ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΙΓΑΙΟΥ

ΔΙΔΑΚΤΟΡΙΚΗ ΔΙΑΤΡΙΒΗ

Προηγμένες μέθοδοι μηχανικής μάθησης στην ανίχνευση δικτυακών επιθέσεων

Συγγραφέας
Δημήτριος Χ. Παπαμαρτζιβάνος

Επιβλέπων
Αναπλ. Καθ. Γεώργιος Καμπουράκης

ΔΙΑΤΡΙΒΗ
για την απόκτηση Διδακτορικού Διπλώματος
στο

Εργαστηρίο Ασφάλειας Πληροφοριακών και Επικοινωνιακών Συστημάτων
Τμήμα Μηχανικών Πληροφοριακών και Επικοινωνιακών Συστημάτων
Πολυτεχνική Σχολή
Πανεπιστήμιο Αιγαίου

Σάμος, Ιούνιος 2019

University of the Aegean

Doctoral Thesis

Advanced machine learning methods for network intrusion detection

Author
Dimitrios C. Papamartzivanos

Supervisor
Associate Prof. Georgios Kambourakis

A thesis submitted in fulfilment of the requirements
for the degree of Doctor of Philosophy

at the

Laboratory of Information and Communication Systems Security
Department of Information and Communication Systems Engineering
School of Engineering
University of the Aegean

Samos, June 2019

Υπεύθυνη Δήλωση

Εγώ ο Δημήτριος Χ. Παπαμαρτζιβάνος, δηλώνω ότι είμαι ο αποκλειστικός συγγραφέας της υποβληθείσας Διδακτορικής Διατριβής με τίτλο «Προηγμένες μέθοδοι μηχανικής μάθησης στην ανίχνευση δικτυακών επιθέσεων». Η συγκεκριμένη Διδακτορική Διατριβή είναι πρωτότυπη και εκπονήθηκε αποκλειστικά για την απόκτηση του Διδακτορικού διπλώματος του Τμήματος Μηχανικών Πληροφοριακών και Επικοινωνιακών Συστημάτων. Κάθε βοήθεια, την οποία είχα για την προετοιμασία της, αναγνωρίζεται πλήρως και αναφέρεται επακριβώς στην εργασία.

Επίσης, επακριβώς αναφέρω στην εργασία τις πηγές, τις οποίες χρησιμοποίησα, και μνημονεύω επώνυμα τα δεδομένα ή τις ιδέες που αποτελούν προϊόν πνευματικής ιδιοκτησίας άλλων, ακόμη κι εάν η συμπερίληψη τους στην παρούσα εργασία υπήρξε έμμεση ή παραφρασμένη. Γενικότερα, βεβαιώνω ότι κατά την εκπόνηση της Διδακτορικής Διατριβής έχω τηρήσει απαρέγκλιτα όσα ο νόμος ορίζει περί διανοητικής ιδιοκτησίας και έχω συμμορφωθεί πλήρως με τα προβλεπόμενα στο νόμο περί προστασίας προσωπικών δεδομένων και τις αρχές της Ακαδημαϊκής Δεοντολογίας.

Υπογραφή:

Ημερομηνία: Ιούνιος 7, 2019

Declaration of Authorship

I, Dimitrios C. Papamartzivanos, declare that this thesis entitled, "Advanced machine learning methods for network intrusion detection" and the work presented in it are my own. I confirm that:

- This work was done wholly while in candidature for a research degree at this University.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: _____

Date: June 7, 2019 _____

Advising Committee of this Doctoral Thesis:

Professor Stefanos Gritzalis, Advisor
Department of Digital Systems
University of Piraeus, Greece

Associate Professor Georgios Kambourakis, Supervisor
Department of Information and Communication Systems Engineering
University of the Aegean, Greece

Assistant Professor Panagiotis Rizomiliotis, Advisor
Department of Informatics and Telematics
Harokopio University, Greece

University of the Aegean, Greece
2019

Approved by the Examining Committee:

Stefanos Gritzalis
Professor, University of Piraeus, Greece

Georgios Kambourakis
Associate Professor, University of the Aegean, Greece

Spyros Kokolakis
Associate Professor, University of the Aegean, Greece

Christos Xenakis
Associate Professor, University of Piraeus, Greece

Emmanouil Kalligeros
Assistant Professor, University of the Aegean, Greece

Constantinos Patsakis
Assistant Professor, University of Piraeus, Greece

Panagiotis Rizomiliotis
Assistant Professor, Harokopio University, Greece

University of the Aegean, Greece
2019

Dimitrios C. Papamartzivanos

Department of Information and Communication Systems Engineering
School of Engineering
University of the Aegean

Abstract

Department of Information and Communication Systems Engineering
School of Engineering
University of the Aegean

Doctor of Philosophy
by Dimitrios C. Papamartzivanos

The contemporary Information and Communications Technology infrastructures have become undoubtedly the "land of opportunity" for ill-motivated entities, which aim to threaten the confidentiality, integrity and availability of the underlying systems. The ever-increasing magnitude and sophistication of cyber attacks leave no room for rest to the defenders. In this context, the quest for full-fledged and versatile defensive frameworks and methodologies is of high priority. In this direction, Intrusion Detection and Response Systems are essential entities in a network topology aiming to safeguard the protected systems and provide remediation actions against offensive incidents. However, such mechanisms need to be supported by intelligent methods to sustain a high operational capability. In this context, this doctoral thesis focuses on advanced machine learning methods that can deliver beneficial characteristics to intrusion detection and response systems.

More specifically, this Phd thesis comprises three tightly interrelated axes, namely a) the provisioning of optimal countermeasures in the context of intrusion response systems, b) the induction of accurate detection rules to enable misuse network intrusion detection, and c) the integration of self-adaptation properties to those systems.

In relation to the first axis, this work provides a comprehensive analysis on reactions frameworks which aim to provide cost-benefit countermeasures against cyber attacks. Our analysis aims to critically scrutinize the pertinent works in this field, to pinpoint the Artificial Intelligence methods utilized by them, and to offer an in-depth discussion and side-by-side comparison among them based on several criteria. Also, an extensive discussion is offered to highlight on the shortcomings and future research challenges and directions in this timely field of research.

Driven by the fact that reactive frameworks should be triggered upon accurate predictions on the nature of offensive incidents, the second axis of the doctoral thesis at hand focuses on the design and implementation of a rule induction methodology, called *Dendron*, for misuse intrusion detection systems. More specifically, our methodology

takes advantage of both Decision Trees and Genetic Algorithms for the sake of evolving linguistically interpretable and accurate detection rules. *Dendron* is able to rightly designate the category where attacks belong to, and achieves superior results over other legacy techniques under several classification metrics.

Additionally, with the aim of tackling the major limitation of misuse intrusion detection systems to adapt to new network conditions, the third axis pursues the development of a self-adaptive methodology, which can revitalize a detection engine through the automation of its retraining process. Considering the extended size of modern networks and the complexity of big network traffic data, the adaptation problem exceeds the limits of human managing capabilities. Thus, through the utilization of deep-learning based methods, our approach is able to grasp an attack's nature based on generalized feature reconstructions stemming directly from the unknown environment and its unlabeled data. The experimental results reveal that our methodology can breathe new life into an intrusion detection system, thus outperforming rigid detection approaches.

Greek Abstract

(Εκτεταμένη Περίληψη)

Τμήμα Μηχανικών Πληροφοριακών και Επικοινωνιακών Συστημάτων
Πολυτεχνική Σχολή
Πανεπιστήμιο Αιγαίου

Διδακτορική διατριβή
του Δημητρίου Χ. Παπαμαρτζιβάνου

Οι σύγχρονες υποδομές τεχνολογίας πληροφοριών και επικοινωνίας έχουν μετατραπεί χωρίς αμφιβολία σε ένα χώρο ευκαιριών για κακόβουλες οντότητες, οι οποίες απειλούν την εμπιστευτικότητα, την ακεραιότητα και διαθεσιμότητα αυτών των συστημάτων. Το συνεχώς αυξανόμενο μέγεθος και η πολυπλοκότητα των κυβερνοεπιθέσεων δεν αφήνουν περιθώρια επανάπαυσης στους αμυνόμενους. Σε αυτό το πλαίσιο, η αναζήτηση ολοκληρωμένων και ευέλικτων αμυντικών μηχανισμών και μεθόδων καθίσταται υψίστης σημασίας. Σε αυτήν την κατεύθυνση, τα συστήματα ανίχνευσης και αντιμετώπισης εισβολών αποτελούν απαραίτητες οντότητες σε ένα δίκτυο για την προστασία των συστημάτων και την παροχή ενεργειών αποκατάστασης εναντίον των επιθέσεων. Ωστόσο, τέτοιου είδους μηχανισμοί είναι απαραίτητο να υποστηρίζονται από ευφυείς μεθόδους, για να είναι σε θέση να διατηρούν υψηλή επιχειρησιακή ετοιμότητα. Σε αυτό το πλαίσιο, η παρούσα διδακτορική διατριβή εστιάζει σε προηγμένες μεθόδους μηχανικής μάθησης, οι οποίες μπορούν να προσδώσουν ωφέλιμα χαρακτηριστικά σε συστήματα ανίχνευσης και αντιμετώπισης εισβολών.

Πιο συγκεκριμένα, η παρούσα διατριβή αποτελείται από τρεις άξονες: α) την παροχή βέλτιστων αντιμέτρων στο πλαίσιο μηχανισμών αντιμετώπισης εισβολών, β) την εξαγωγή αξιόπιστων κανόνων ανίχνευσης για συστήματα ανίχνευσης εισβολών κακής χρήσης (Misuse Detection IDS), και γ) την ενσωμάτωση χαρακτηριστικών αυτοπροσαρμογής σε αυτά τα συστήματα.

Σχετικά με τον πρώτο άξονα, η παρούσα διατριβή παρέχει μία εκτενή ανάλυση μηχανισμών αντιμετώπισης εισβολών, οι οποίοι στοχεύουν στην παροχή βέλτιστων αντίμετρων εναντίον κυβερνοεπιθέσεων. Η ανάλυσή μας στοχεύει να εξετάσει λεπτομερώς και με κριτικό πνεύμα τις σχετικές δημοσιεύσεις του συγκεκριμένου τομέα, να εντοπίσει τις μεθόδους τεχνίτης νοημοσύνης που αυτές αξιοποιούν και να προσφέρει μία σε βάθος συζήτηση και αναλυτική σύγκριση βάσει κριτηρίων. Επιπλέον, επισημαίνονται οι ελλείψεις και οι μελλοντικές ερευνητικές προκλήσεις του συγκεκριμένου ερευνητικού πεδίου.

Ορμώμενοι από το γεγονός ότι οι μηχανισμοί αντιμετώπισης επιθέσεων θα πρέπει να ενεργοποιούνται βάσει ακριβούς πρόβλεψης της φύσης των επιθέσεων, ο δεύτερος άξονας της παρούσας διατριβής εστιάζει στο σχεδιασμό και την ανάπτυξη μίας μεθοδολογίας εξαγωγής κανόνων, με την ονομασία *Dendron*, για συστήματα ανίχνευσης εισβολών κακής χρήσης. Συγκεκριμένα, η μεθοδολογία μας εκμεταλλεύεται Δέντρα Απόφασης (Decision Trees) και Γενετικούς Αλγορίθμους (Genetic Algorithms), με σκοπό την ανάπτυξη μεταφράσιμων και αξιόπιστων κανόνων ανίχνευσης. Το *Dendron* είναι ικανό να προσδιορίζει σωστά την κατηγορία στην οποία ανήκουν οι επιθέσεις, ενώ επιτυγχάνει καλύτερη απόδοση, σε σύγκριση με άλλες κλασικές τεχνικές, στις περισσότερες μετρικές κατηγοριοποίησης.

Επιπρόσθετα, με σκοπό την αντιμετώπιση του σημαντικότερου μειονεκτήματος των συστημάτων ανίχνευσης κακής χρήσης, που είναι η αδυναμία προσαρμογής σε νέες δικτυακές συνθήκες, ο τρίτος άξονας της διατριβής αποσκοπεί στην ανάπτυξη μίας αυτοπροσαρμοζόμενης μεθοδολογίας, η οποία μπορεί να αναζωογονήσει μία μηχανή ανίχνευσης μέσω της αυτοματοποίησης του μηχανισμού επανεκπαίδευσής της. Λαμβάνοντας υπόψη την εκτεταμένη κλίμακα των σύγχρονων δικτύων και την πολυπλοκότητα των δικτυακών δεδομένων, το πρόβλημα της προσαρμογής υπερβαίνει κατά πολύ τις δυνατότητες διαχείρισης από έναν ειδικό ασφάλειας. Έτσι μέσω της αξιοποίησης μεθόδων Βαθιάς Μάθησης (Deep Learning), η μεθοδολογία μας μπορεί να αντιληφθεί τη φύση μίας επίθεσης βάσει γενικευμένων ανασχηματισμένων χαρακτηριστικών (generalized feature reconstructions) που προέρχονται απευθείας από το άγνωστο δικτυακό περιβάλλον και τα δικτυακά δεδομένα, από τα οποία απουσιάζει η κατηγορική ετικέτα κλάσης. Τα πειραματικά αποτελέσματα δείχνουν ότι η μεθοδολογία μας μπορεί να αναζωογονήσει ένα σύστημα ανίχνευσης εισβολών, και επιπλέον επιτυγχάνει καλύτερη απόδοση σε σχέση με κλασικές μη-ευέλικτες προσεγγίσεις.

Acknowledgements

Firstly, I would like to express my sincere gratitude to my supervisor and mentor Associate Prof. Georgios Kambourakis for the continuous support during this long journey. I would like to thank him for being always there for me by giving me confidence and encouraging my research. Without his valuable guidance this work would not have been materialized.

I would like to heartily thank Dr. Félix Gómez Mármol, who has not only been my advisor but also a friend. He gave me the chance to broaden my scientific horizons by working with him at NEC Laboratories Europe, Heidelberg and at the University of Murcia, Spain.

I would like to express my gratitude to my mentor Prof. Stefanos Gritzalis, who always believed in me and in my potential. In addition, I would like to thank Assoc. Prof. Spyros Kokolakis, Assoc. Prof. Christos Xenakis, Assist. Prof. Emmanouil Kalligeros, Assist. Prof. Constantinos Patsakis and Assist. Prof. Panagiotis Rizomiliotis, members of the examining committee, for investing some of their time in reviewing this thesis and offering advice. I would also like to thank Assist. Prof. Constantinos Kolias for his insightful comments and suggestions.

I am deeply grateful to my colleagues and friends, Sia Douma, Dr. Marios Anagnostopoulos, Dr. Dimitrios Damopoulos, Dr. Zisis Tsiatsikas, Alexandros Fakis, Dr. Christos Merkatas, Dr. Konstantinos Kaloudis, Pantaleone Nespoli, and Mattia Zago for sharing constructive ideas and moments of joy. I hope they will fulfill their goals and have a fortunate life.

A special thanks goes to Alice Sagianou for being always next to me during this long journey. Her love and warm encouragement gave me the strength to keep working hard for my goals.

Nothing would have been possible without the financial, emotional and moral support of my family. My parents Christos and Konstantini, my sister Athanasia, and my grandparents Dimitrios and Kyratsouda, Sokrates and Kyriaki are part of myself and I am grateful to them. My parents passed to me the passion for gaining knowledge and the love for sciences. Their belief in me, their encouragement and unconditional love gave me strength throughout my research.

Dedicated to my family,
Christos Papamartzivanos, Konstantini Pagkou & Athanasia Papamartzivanou.

Contents

List of Figures

List of Tables

Abbreviations

ACO	Ant Colony Optimization
ACT	Attack Countermeasure Tree
ADR	Attack Detection Rate
ADT	Attack Defense Tree
AG	Attack Graph
AI	Artificial Intelligence
AIS	Artificial Immune Systems
ALE	Annualized Loss Expectancy
AMSEC	Attack Modeling and Security Evaluation Component
ANN	Artificial Neural Networks
ART	Attack Response Tree
ASR	Asset Summary Report
AT	Attack Tree
AV	Antivirus
BAG	Bayesian Attack Graph
BIRCH	Balanced Iterative Reducing and Clustering using Hierarchies
BPSO	Binary Particle Swarm Optimization
BYOD	Bring Your Own Device
CAPEC	Common Attack Pattern Enumeration and Classification
CCE	Common Configuration Enumeration
CCSS	Common Configuration Scoring System
CIA	Confidentiality Availability and Integrity
CMDP	Competitive Markov Decision Process
CMSS	Common Misuse Scoring System
CPE	Common Platform Enumeration

CPSO	Chaos Particle Swarm Optimization
CRE	Common Remediation Enumeration
CRF	Conditional Random Fields
CVE	Common Vulnerability Exposure
CVRF	Common Vulnerability Report Format
CVSS	Common Vulnerability Scoring System
CWE	Common Weakness Enumeration
CWRAF	Common Weakness Risk Analysis Framework
CWSS	Common Weakness Scoring System
CybOX	Cyber Observable eXpression
DAG	Directed Acyclic Graph
DCAE	Dilated Convolutional Autoencoders
DDoS	Distributed Denial-of-Service
DGSOT	Dynamically Growing Self-Organizing Tree
DNN	Deep Neural Networks
DoS	Denial of Service
DSS	Domain Specific System
DT	Decision Tree
ERI	Extended Remediation Information
FAR	False Alarm Rate
FFAE	FeedForward Autoencoder
FTP	File Transfer Protocol
FW	Firewall
GA	Genetic Algorithm
GUI	Graphical User Interface
HMM	Hidden Markov Model
ICT	Information and Communications Technology
IDMEF	Intrusion Detection Message Exchange Format
IDS	Intrusion Detection System
IODEF	Incident Object Description Exchange Format
IoT	Internet of Things
IPS	Intrusion Prevention System
IRS	Intrusion Response System

KL	Kullback-Leibler
KPCA	Kernel Principal Component Analysis
LCPD	Local Conditional Probability Distribution
LCS	Learning Classifier Systems
LOA	Level of Automation
LSTM	Long Short-Term Memory
MAEC	Malware Attribute Enumeration and Characterization
MAPE-K	Monitor, Analyze, Plan, Execute - Knowledge
MARS	Multivariate Adaptive Regression Splines
MCDM	Multi-Criteria Decision Making
MCLP	Multiple Criteria Linear Programming
MDP	Markov Decision Process
MFM	Mean F-Measure
ML	Machine Learning
MLP	Multilayer perceptron
MOOP	Multi-Objective Optimization Problem
MOTS	Multi-Objective Tabu Search
MTD	Moving Target Defense
NB	Naive Bayes
NDAE	Non-Symmetric Deep Autoencoder
NSGA-II	Non-dominated Sorting Genetic Algorithm II
NVD	National Vulnerability Database
OpenIOC	Open Indicator Of Compromise
OVAL	Open Vulnerability and Assessment Language
POCMDP	Partially Observable Competitive Markov Decision Process
POMDP	Partially Observable Markov Decision Process
PRB	Probing
R2L	Remote to Local
RAOM	Risk Assessment and Optimization Model
RBM	Restricted Boltzmann Machine
ROI	Return-On-Investment
RORI	Return-On-Response-Investment
RRE	Response and Recovery Engine

SAE	Sparse Autoencoder
SC	Spectral Clustering
SCAP	Security Content Automation Protocol
SDG	Service Dependency Graph
SIEM	Security Information and Event Management
SLA	Service Level Agreement
SMO	Sequential Minimal Optimization
SOAR	Security Orchestration, Automation and Response
SOHO	Small Office Home Office
SOOP	Single-Objective Optimization Problem
STIX	Structured Threat Information eXpression
STL	Self-taught Learning
SVM	Support Vector Machines
TAXII	Trusted Automated eXchange of Indicator Information
TCP	Transmission Control Protocol
TMSAD	Trust Model for Security Automation Data
TN	True Negative
TP	True Positive
U2R	User to Root
UDP	User Datagram Protocol
VANETs	Vehicular ad-hoc networks
VFDT	Very Fast Decision Tree
VM	Virtual Machine
WSN	Wireless Sensor Network
XCCDF	Extensible Configuration Checklist Description Format

Chapter 1

Introduction

Intrusion detection systems (IDSs) are considered a mainstay when it comes to the protection of Information and Communications Technology (ICT) infrastructures. The concept of monitoring and surveilling computer security threats was set back in 80's, when James P. Anderson's work [1] became a stepping stone for the intrusion detection research field. Since then, the technological progress has formed a landscape of a densely interconnected world, which has become the "land of opportunity" for ill-motivated entities that introduce new security and privacy threats in an everyday basis. In fact, the everlasting battle between defenders and attackers has taken the form of an "arm race", where both sides constantly upgrade their arsenals in order to prevail against each other. The emergence of new attacks spurs the academia and industry to investigate for novel methodologies, which are able to closely monitor this race and adapt rapidly to the changes in the field.

Undoubtedly, IDSs has become an irreplaceable weapon in this battle, as both the industry and the research community have offered a plethora of tools and methodologies to oppose to offensive security incidents. IDSs monitor and analyze passively network and system activities for recognizing attack patterns. In fact, there is a wide spectrum of IDSs depending on their deployment position in the infrastructure and their operational profile. Regarding the deployment position, the IDSs can be classified into *Host-based* and *Network* systems. The former approach monitors processes and actions accruing in a specific host, while the latter is placed to strategic points of a network and monitors all the traffic exchanged among its nodes. Regarding the operational profile, the IDSs can

be classified into *Misuse-based* (sometimes also called signature-based) and *Anomaly-based* (sometimes also called profile-based) systems. The former class bases the detection process on previously identified signatures, which can take a form of rules or patterns, while the latter regulates its detection engine to identify as intrusive incidents those that exhibit deviations from a predefined normal behavioral profile. Misuse systems are destined to identify known type of attacks, whereas they are able to keep False Alarm Rate (FAR) under acceptable levels. Hence, the detection ability of such a system is directly affected by the freshness of the detection rules it possesses. On the other hand, anomaly detection systems are able to detect previously unseen attacks, but they usually suffer from a FAR. This limitation renders such systems an impractical solution for protecting a large and sensitive infrastructure as the definition of the normal behavior is normally a demanding task. All the aforementioned approaches bear advantages and disadvantages and a potential deployment needs to take them into consideration for building a defense mechanism able to meet the security requirements of the host infrastructure.

However, the detection of an incident is just a milestone in a multistep process which has to be triggered for safeguarding an infrastructure. The steps that follow the detection of an offensive action aim to prevent possible negative consequences and respond promptly by taking the proper actions. In this direction, Intrusion Prevention Systems (IPS) are introduced to actively trigger basic remediation actions. In contrast to IDSs, IPSs are installed inline with the network so that to actively detect and react on the current traffic. IPS actions include drop, reset or custom-scripted actions and all of that occurs immediately upon signature match to proactively mitigate the consequences. Thus, a potentially falsified action can lead to loss of revenue as the IPS might drop legitimate traffic. Under this prism, it becomes clear that any remediation action, especially if it is destined to operate with a given automation level, should be based on a detection mechanism which is able, not only to accurately detect an offensive event, but to infer the exact category within the incident belongs to and to keep false alarms to relatively low levels. Additionally, as cyber attacks constantly gain in sophistication and ICT infrastructures become even more complex and dynamic, simplistic IPSs become inadequate.

Due to the above reasons, there is a constant need to extend security solutions beyond legacy surveillance and preventative technologies and to use advanced intelligent

methods to enable automation across the entire network security architecture. In this direction, Data Mining, Artificial Intelligence (AI) and Machine Learning (ML) are the mainstays for developing novel methods, which are able to bring intelligence into legacy security solutions by leveraging the inherent information of data. The seminar works of Lee and Stolfo [2],[3] showed the way for designing ML-based detection methods by utilizing data generated from a data mining approach. Since then, ML-based IDSs are always within the scope of researchers as they can deduce the detection rules (or patterns) in an automated manner by grasping feature structures in the data. Additionally, AI-enabled intrusion response systems (IRS) attract the interest of the research community for supporting the decision making and providing optimal remediation during an incident response process.

However, the evolution of ICT systems and the parallel continuous and intensive advance of attacking techniques fuels the development of defensive systems with the aim of bringing back the advantage to the defenders' side. Additionally, the limitations of the existing detection and response techniques leave room for further research and innovation. In fact, current attack detection methodologies lack agility and lead to rather rigid detection systems. To fill these gaps, this doctoral thesis seeks for new methods that can address the reported limitations and empower detection systems with agile and adaptable characteristics. Under this prism, our work:

(a) provides a thorough analysis over cost-benefit reactive methodologies for offering optimal countermeasures.

(b) utilizes nature-inspired evolutionary techniques to enable objective-driven detection rules induction.

(c) aims to introduce self-adaptation properties to misuse detection systems through Deep Learning methodologies.

As further explained in the next section, the PhD thesis at hand aims in introducing versatile properties to misuse network IDSs through advanced machine learning methodologies.

1.1 Motivation and Objectives

From the conceptualization of the network surveillance systems [1] and the introduction of data mining techniques to enable ML-based intrusion detection [2] [3] until the present days, it becomes clear that the battle has become even more intensive, as the diversity of the contemporary systems creates many opportunities for aggressors. This fact, motivates us to investigate for new methodologies which can address key limitations in this field.

Keeping an IDS's rules database up-to-date is a challenging procedure that requires system administrators' engagement. Handling and analyzing a huge volume of network traffic is an intensive engineering task, which needs to be supported by automated tools able to grasp the characteristics of the network traffic. Hence, such a tool should not only aim to distinguish between legitimate and malicious traffic, but also to infer the specific class of an attack occurring in the target system. We argue that a detection system which can answer simply the question "Attack or not?" is considered incompetent given the variety of contemporary cyber attacks. Thus, a modern rule induction methodology should adopt a multi-classed model to handle the network traffic.

However, such an approach raises challenges mainly due to specific properties of the network traffic flows. Currently, state-of-the-art approaches neglect minor classes that indicate a malicious behavior due to their scarcity in the network data. Nevertheless, such kind of incidents can have major impact on a targeted system. In this context, a rule induction methodology should be driven by objectives that ensure a high Attack Detection Rate (ADR) and a low FAR, but additionally high recall and precision scores in predicting the exact class of an incident. Additionally, the inducted rules need to be human interpretable in order to help administrators understand the semantics of the detection process.

Even thought in the related literature exist a great amount of publications, the above-mentioned qualities cannot be found in a single methodology. Moreover, such properties are a cardinal concern when a detection process is followed by a responsive procedure with a certain automation level. The accuracy of the incident inference process can affect the efficacy of a response mechanism to a great extend, as possibly incorrectly identified

incidents could lead to inappropriate countermeasures, or even worse, to unnecessary actions in case of false alarms.

A rule induction methodology can alleviate to a great extend the burden of the administrators' shoulders, but is it not a panacea. It can be used to extract detection rules automatically out of network traffic flows, but security specialists need to put significant effort to annotate them properly before those are given as input to a supervised machine learning method. It becomes clear that the retraining of an intrusion detection model still remains a demanding and rigid procedure and needs to take place anytime a drop in efficacy appears. Undoubtedly, there are several factors that can affect the efficiency of an IDS, but most of them pertain to the *environmental changes*. The latter term refers to any aspect of a network that can affect the profile of the generated network traffic. This reality, combined with the lack of adaptable detection engines, forces the IDS to become quickly outdated and inadequate as it inevitably has to operate in new and "unknown" or unforeseen environments for which its engine was not trained. In fact, this is a key limitation in regard to any ML-based IDS and the quest of proper ways to automatize the retraining process comprises a major research challenge in the field.

Given the above mentioned observations, the motivation of the PhD thesis at hand is to investigate for ML methods that can provide agile and versatile characteristics to network IDSs. In a nutshell, the objectives (and simultaneously the research pillars) of this PhD thesis are as follows:

Objective 1: We intent to shed light on IRSs systems proposed in the literature so far and to provide a thorough analysis over the methodologies used to empower cost-benefit reactive systems for countermeasure provision. Through this analysis we intent to identify open research challenges in both IDS and IRS ecosystems and provide future directions.

Objective 2: We aim to deliver a rule induction methodology that can provide highly accurate detection rules, which can enable the discrimination among distinct classes of offensive incidents with a high precision. The rules must be human interpretable for assisting security administrators to comprehend the detection semantics.

Objective 3: We aim to expand further the second objective and further explore ML methodologies that can provide self-adaptation and agile properties to misuse network IDSs, so that to bring intelligence in the retraining process of their engine.

As detailed in the next subsection, the novelties of this work mainly lie in the last two objectives, while the first one basically explores the related literature for identifying possible gaps, shortcomings, and research directions.

1.2 Contributions

As already pointed out, the main intention of this PhD work is to investigate and develop ML approaches that can provide agile and versatile characteristics to network intrusion detection and response systems. To this end, our work exploits the beneficial characteristics of nature-inspired evolutionary computation and deep learning for addressing the limitations identified in the related literature.

Initially, our work provides a comprehensive analysis on reaction frameworks which aim to provide optimal countermeasure selection against cyber attacks. Currently, even though there is a plethora of solutions to analyze the security state of a network and to detect offensive incidents, the decisions during an incident response process are taken by security experts. To the best of our knowledge, there are no tools that can support the decision making and provide cost-benefit countermeasures. Instead, this process is undertaken by security experts who use their experience and – hopefully – well documented incident response policies to base their decisions.

However, as explained also in the previous subsections, any remediation action applied with a given level of automation has to be based on an accurate prediction in order to avoid inappropriate countermeasures, or unnecessary actions in case of a false alarm. In this context and in conjunction with the second objective of this thesis, we provide a prototype implementation of an evolutionary classifier for detecting network offensive incidents.

Additionally, our work tackles the major limitation of a misuse network IDS, which is its inefficiency to adapt to new and "unknown" environments. By utilizing a deep learning approach, we aim to build and evaluate the first to our knowledge proof of concept for

self-adaptive misuse network IDSs. That is, our approach aims to address the cardinal challenge for any IDS designer, namely to bring intelligence and find proper ways to automatize, at least to a certain degree, the retrain process of the detection engine.

More specifically, the contribution of this PhD thesis with respect to our publications in scientific journals and patent filings is as follows:

- A thorough analysis of the intrusion reaction frameworks proposed in the literature [1]. The main pillars of this contribution are:
  - Provide a detailed overview over the fundamental components utilized by reaction frameworks.
  - Identify primary characteristics of reaction frameworks (and their chief enablers, namely IDS) and provide a side-by-side comparison by highlighting on the advantages and limitations of the surveyed proposals.
  - Identify research challenges and future directions in the cost-benefit intrusion response and detection research fields.
- The design and implementation of a novel classification algorithm, called *Dendron*, to enable misuse network intrusion detection [2]. In this context, we:
  - Provide a new methodology for detection rules induction through the combination of Genetic Algorithms and Decision trees.
  - Introduce a selection probability function to steer the evolutionary process towards delivering a multiclassed individual with desirable characteristics.
  - Provide a rule induction methodology driven by desirable performance objectives which is able to treat fairly all the network traffic classes.
- *Dendron* is used to empower a detection rule induction method and system which led to a Patent filing by NEC Corporation, for reserving the exploitation right of the invention [3].

[1] P. Nespoli, D. Papamartzivanos, F. Gómez Mármol, and G. Kambourakis. Optimal countermeasures selection against cyber attacks: A comprehensive survey on reaction frameworks. IEEE Communications Surveys Tutorials, 20(2):13611396, Secondquarter 2018. ISSN 1553-877X. doi: 10.1109/COMST.2017.2781126.

[2] D. Papamartzivanos, F. Gómez Mármol, and G. Kambourakis. Dendron : Genetic trees driven rule induction for network intrusion detection systems. Future Generation Computer Systems, 79:558 574, 2018. ISSN 0167-739X. doi: 10.1016/j.future.2017.09.056.

[3] D. Papamartzivanos and F. Gómez Mármol. Intrusion detection and prevention system and method for generating detection rules and taking countermeasures, September 25 2018. US Patent 10,084,822.

- The design and implementation of a novel deep learning-based self-adaptive network misuse IDS [4]. The contributions of this work are as follows:
  - Provide a scalable approach which combines the benefits of Self-Taught Learning (STL) and MAPE-K methodologies (see section 4.3) to enable IDS self-adaptation.
  - Automatize the retraining process of an IDS by utilizing new data feature representations through transfer learning from unlabeled data, which stem from the unknown environment where the IDS operates.
  - Evaluate the adaptation and detection ability of the IDS by exposing it to consecutive and drastic environmental changes.
- The filing of a patent application by NEC Corporation for reserving the exploitation rights of a method and apparatus inspired by our work on self-adaptive intrusion detection [5].

[4] D. Papamartzivanos, F. Gómez Mármol, and G. Kambourakis. Introducing deep learning self-adaptive misuse network intrusion detection systems. IEEE Access, 7:1354613560, 2019. ISSN 2169-3536. doi: 10.1109/ACCESS.2019.2893871.

[5] D. Papamartzivanos, F. Gómez Mármol, G. Kambourakis, and R. Bifulco. Adaptive network intrusion detection, November 9 2018. US provisional patent application 62/757,769.

1.3 Thesis Structure

The next chapter presents the fundamental background of Intrusion Detection, Prevention and Response systems. More specifically, the Chapter 2 comprises two major sections:

- Section 2.2 elaborates on Machine Learning methodologies and datasets used to realize intrusion detection systems.
- Section 2.3 provides a survey over reaction frameworks for optimal countermeasure provision against cyber attacks. This section aims to provide a side-by-side comparison of the works presented in the literature, to uncover research challenges and provide future directions.

Chapter 3 deals with the design and development of *Dendron*, a novel rule induction methodology for network intrusion detection. More specifically, this chapter provides a detailed overview of the evolutionary components utilized to steer the induction process towards developing detection rules of beneficial characteristics.

Chapter 4 details on the applicability of deep learning techniques for bringing the quality of self-adaptation to network IDSs. Precisely, this chapter introduces the combination of Self-Taught Learning and MAPE-K methodologies to provide an holistic approach for autonomous and self-adaptive intrusion detection.

The last chapter provides a discussion over the results and the contributions of this PhD thesis. Additionally, it includes potential extensions of the proposed schemes as well as future research directions.

Chapter 2

Background

This chapter provides an overview of the intrusion detection and response research field, while it also elaborates on machine learning approaches which have been recruited among several researches. Initially, the chapter elaborates on fundamental notions of cyber defense and in Section 2.2 proceeds to the analysis and taxonomy of ML-based intrusion detection approaches of the field. Section 2.3 provides an extensive analysis of reactive methodologies which aim to provide optimal countermeasures and responses against cyber attacks.

2.1 The OODA loop for incident response

As the engagement between attackers and defenders has taken the form of an ever-lasting battle, several military terms and phrases have been adopted by information security experts for describing tactics in the cyberspace. Among them, *OODA* loop, which stands for ***O**bserve, **O**rient, **D**ecide, and **A**ct*, is a principal military strategy used to enhance the decision making in a battle field. U.S. Air Force Colonel John Boyd conceptualized the OODA loop [4], and nowadays is a practical analogy adopted in the incident response tactics for aiding the defense procedures to adapt properly considering the moves of the attackers.

2.1.1 Observe

This phase consists of information gathering and monitoring actions for collecting as much information as possible regarding the steps of an adversary and the security state of the protected system. The detailed and fast discovery of suspicious events can be of great value for understanding an adversary's target and aiding in optimal countermeasure selection in later steps.

Primary source of information in this phase are the Network or Host-based IDSs which are able to identify patterns of known attacks or traces of anomalous behavior in the network or on a host respectively. Several IDS solutions exist in the wild and they offer diverse monitoring capabilities. Among them, Snort [5], Suricata [6] and Zeek [7] (formerly known as Bro [8]) are the most famous network IDSs, while OSSEC [9] is a host-based IDS.

Another valuable tool that can offer great visibility in the network traffic is the Netflow analyzers. Such tools enable visibility in traffic flows and help administrators to take a deeper look into threats and locate traffic exchanges among assets of high importance. Tools, such as Ntop [10] and Nfdump [11] with NfSen [12], can offer network traffic visibility through proper visualization techniques.

Vulnerability scanners, such as OpenVAS [13], Nessus [14] and Nexpose [15], are a valuable asset for observing the security situation of an infrastructure. This kind of tools are able to discover and asses existing vulnerabilities on systems, and uncover weak points that can become points of failure of the entire security architecture.

2.1.2 Orient

This phase require to logically analyze the collected evidences and correlate them with previously faced incidents, the security experts' experience and any other source of threat intelligence. The aim of this phase is to interpret the evidences in order to understand the attack strategy and evaluate the possible impact on business assets. That said, a security analyst needs to deploy a defensive strategy based on the available observations and understand the existing environment state.

In practice, getting the full picture of the situation during an attack is almost infeasible due to several reasons, like the attack sophistication or the lack of accurate observations. That is, the orientation phase is revisited any time additional observations come into view. In this process, asset inventory tools can provide valuable information about the current situation of the network and host assets. Such tool can be proved really useful, especially when the protected infrastructure is a dynamic one, where assets or services may enter or leave the network any time. Additionally, open threat intelligence databases offer a wide spectrum of information regarding the latest threats, like malwares, botnets, blacklisted IPs, and more.

2.1.3 Decide

Given the observations and the outcome of the orientation phase a defender has to decide about a defensive strategy that can eradicate the threat but can have the minimum negative implications to the business runtime. This phase can be supported by documented incident response procedures and the security policy of the organization. However, the decisions are taken by security experts based on their experience as, to the best of our knowledge, there are no tools that can support the decision making in this phase.

Hence, the decision making for providing optimal countermeasures against cyber attacks requires significant effort to counterbalance the cost and the benefit of a defensive strategy, while the whole process takes place under stressful conditions.

2.1.4 Act

In this phase a security expert applies the defensive actions decided in the previous phase. The aim of this step is to remediate the affected systems and recover any interrupted business workflow. A security expert or a team must act quickly but considering also the business cost of any remediation action. That is, the OODA loop may occur repetitively until an optimal solution is defined.

The remediation and recovery actions usually engage a wide spectrum of tools. Forensic tools can be used in order to uncover and analyze the facts of an incident and to create a legal audit trail. System backup and recovery tools can be utilized to revert the affected systems back to the normal operational state. The final goal of the OODA loop is, not

only to tackle an incident, but also to gradually increase security awareness through the analysis of the weak points of the security architecture. At the end of this process, a security team has the chance to improve the incident response procedures by exploiting the recent observations, orientation outcomes and decisions.

2.2 Machine Learning-based intrusion detection systems

As described in Chapter 1, the novelties of this PhD thesis mainly fall into the ML-based intrusion detection field. That is, this section provides an overview of the detection methodologies, the datasets and the widely used ML methods in the intrusion detection literature.

Machine Learning is a sub-domain of AI which exploits the inherent information of data in order to automatically identify patterns and help with decision making. A wide variety of specialized algorithms and techniques synthesize a powerful tool for designing intelligent predictive systems for a wide spectrum of business tasks. In the case of cybersecurity, and more specifically in network intrusion detection, we use classification or clustering algorithms to discover anomalies and offensive incidents occurring in a protected network. The power of ML resides in the ability of ML algorithms to automatically identify patterns based on thousands of features within massive amounts of network traffic records, which otherwise would require manual mining on behalf of an expert. The extracted patterns are used to build generalized models, which can predict automatically the legitimacy of future network traffic instances or determine their proximity to anomalous or legitimate behaviors. Hence, ML-based IDSs goes beyond simplistic approaches of setting thresholds and monitoring specific metrics, as they are destined to scale and adapt to the demanding nature of large network environments.

2.2.1 Detection approaches

IDSs can be classified into three classes, namely *misuse*, *anomaly* and *specification*-based, according to their operational profile [16]. Table 2.1 summarizes the advantageous and weak characteristics of the aforementioned systems.

Misuse detection systems (sometimes also called signature-based) aim to detect signatures of known attacks [3]. Primarily, such systems base the detection process on the formation of rules or patterns which can match these signatures. The advantage of misuse detection systems lies on their simplicity and effectiveness when it comes to the detection of known attacks. Additionally, their functional profile enables the contextual analysis of an incident on behalf of a security expert, while the detection rules can be shared across information security communities. In fact, the high detection rates for known attacks comes along with low False Alarm Rates (FAR), but they miss to detect new or deviations of known attacks. Consequently, the detection performance of a misuse detection system is strictly connected to the freshness of its rule database, which in practice is a demanding engineering task.

On the other hand, anomaly-based detection systems identify as anomaly any deviation from a standardized normal behavioral profile [17, 18]. In contrast to misuse detection systems, these systems are able to detect unforeseen attacks or deviations of known ones, but they suffer from high a FAR. In fact, this is the main shortcoming of this detection approach, which lies on the fact that the standardization of a normal operational profile is an error-prone process. Given that the operational profile of a system can change even upon routine actions, such as software/OS updates, one could say that anomaly-based detection systems is an impractical solution for protecting large or dynamic infrastructures. Additionally, anomaly-based systems provide little insight to the nature of the offensive event, as the latter is identified as an anomaly. That is, further investigation is needed to determine the nature of an attack.

In an effort to combine the best of both worlds, there have been reported also solutions that join the benefits of misuse and anomaly-based systems, known as hybrid detection systems [19].

Specification-based detection systems are destined to monitor specific functionalities, system configurations or protocol states of the protected systems [20]. In a sense, they share the same idea of anomaly-based systems as they aim to identify changes of a standard norm. Specification-based IDSs are widely used for resource constrained devices, which present a minimal spectrum of actions or, in case of communication protocol surveillance, they follow the established protocol standards.

TABLE 2.1: Classification of intrusion detection approaches.

Misuse detection systems	**Anomaly-based systems**	**Specification-based systems**
Advantages		
• High detection rates for known attacks • Enable contextual analysis of incidents • Enable rule sharing among communities • Low false alarm rates	• Detection of unforeseen attacks • Detection of zero Day attacks	• Sound determination of correct behavior • Detection of unexpected states or actions • Suitable for resource constrained devices
Shortcomings		
• Inability to detect new or deviations of known attacks • Hard to keep them up-to-date	• High false alarm rates • Hard to determine the normal profile • Unsuitable for complex and dynamic systems	• System specific implementations • Unable to detect attacks that mimic the legitimate behavior

Regarding the deployment position, the IDSs can be classified into *Host-based* and *Network* systems. The former approach monitors processes and actions accruing in a specific host, while the latter is placed to strategic points of a network and monitors all the traffic exchanged among its nodes.

Since, the contribution of this PhD thesis focuses mainly on the misuse network intrusion detection approaches, the discussion offered later in this section focuses on the ML methods and algorithms utilized in this specific class of detection systems.

2.2.2 Datasets for network intrusion detection

This section presents a collection of widely used datasets in the intrusion detection literature. Our research ideas presented later in Chapters 3 and 4 utilize the KDDCup'99 [21], NSL-KDD [22] and UNSW-NB15[23] datasets to build our evaluation testbed.

KDDCup'99 This dataset [21] constitutes a standard in machine learning approaches for building IDS to detect web service abuses. Even though this dataset is rather old and there have been researches [22] reporting its flaws, it is still considered as standard and used by recent studies in the field [24]. This is mainly because KDDCup'99 poses the same challenges with the network traffic and is used as a benchmark for conducting a valid comparison among the proposed methods in the literature.

The simulated traffic contained in this dataset includes a variety of intrusions (22 distinct types of attacks - see Table 2.2) under different probability distributions all of them falling into four major categories:

Class	Attacks of KDDCup'99	Attacks of NSL-KDD
DoS	back, land, neptune, pod, smurf, teardrop	back, neptune, smurf, teardrop, land, pod, apache2, mailbomb, processtable, udpstorm
PRB	ipsweep, nmap, portsweep, satan	satan, portsweep, ipsweep, nmap, mscan, saint
R2L	ftp_write, guess_passwd, imap, multihop, phf, spy, warezclient, warezmaster	ftp_write, warezclient, spy, named, warezmaster, multihop, xsnoop, sendmail, snmpguess, imap, snmpgetattack, worm, ftp_write, xlock, phf, guess_passwd
U2R	buffer_overflow, loadmodule, perl, rootkit	buffer_overflow, httptunnel, loadmodule, perl, rootkit, xterm, ps, sqlattack

TABLE 2.2: Attack categories in KDDCup'99 and NSL-KDD

- **DoS**: Denial of Service.
- **PRB**: Probing - Surveillance and other means of probing.
- **R2L**: Remote to Local - Unauthorized access from a remote machine.
- **U2R**: User to Root - Unauthorized access to local superuser (root) privileges.

The raw traffic data captured during the simulations were transformed into machine learning labeled instances representing "good" and "bad" connections to the target system. The initial size of the dataset (approx. 5M instances) renders the set too bulky for training machine learning algorithms. Thus, the vast majority of researchers make use of sampling techniques to create a smaller fraction (usually 10%) of the dataset by keeping its initial properties.

Moreover, KDDCup'99 includes several duplicated network instances, which tend to bias ML-based predictive models. That is, it is a common strategy in ML-based IDS research to preprocess the dataset and remove these redundant instances for building more reliable detection schemes.

NSL-KDD As already mentioned in the previous paragraph, KDDCup'99 has been criticized for several inherent deficiencies [22]. Those deficiencies motivated the authors in [22] to release NSL-KDD dataset in 2009 and since then several intrusion detection researches based their evaluation on it. While NSL-KDD kept the advantageous and challenging characteristics of KDDCup'99, including the imbalanced number of instances and the variety of classes, its creators made a step forward by eliminating the documented limitations. More specifically, NSL-KDD is free of redundant records that bias machine learning techniques to favorable results, while the list of the incorporated

attacks has been extended as it can be observed from Table 2.2. Notably, the prominent characteristic of NSL-KDD is that it was compiled towards maximizing the difficulty level of prediction. To do so, the authors evaluated the initial dataset using several benchmark classifiers and they annotated each instance with the number of its successful predictions only to finally group them into five difficulty levels. Eventually, the authors compiled a training set comprising from instances among all difficulty levels but giving higher priority to those of the higher difficulty levels, i.e., those instances annotated by a lower number of successful predictions.

UNSW-NB15 This data collection was proposed in [23] as a modernized dataset which reflects the contemporary network traffic characteristics and new low-footprint attack scenarios. The authors created a testbed for generating both malicious and normal traffic, which in the first place was captured in pcap files. The attacks launched during the network experiment acquired from the CVE (Common Vulnerability Exposure) database to ensure the contemporary threat environment. The authors used Argus [25] and Bro-IDS [8] tools to manipulate the captured network traffic in order to extract machine learning features and label the instances of the dataset. The authors extracted 49 features in total to reflect the nature of the network traffic, while the malicious traffic was finally grouped into 9 classes as can be seen in Table 2.3.

Class	No. of instances
Normal	2,218,761
DoS	16,353
Reconnaissance	13,987
Fuzzers	24,246
Shellcode	1,511
Exploits	44,525
Generic	215,481
Backdoor	2,329
Analysis	2,677
Worms	174
Total	2,540,044

TABLE 2.3: UNSW-NB15 dataset instances distribution

ISCX Datasets In an effort to provide contemporary evaluation testbeds for intrusion detection, the Canadian Institute for Cybersecurity makes available a set of datasets, which according to the creators are modifiable, extensible, and reproducible [26]. The

ISCX datasets follow the systematic approach presented in [27], which aims to the creation of datasets by combining diverse *profiles*. The profiles contain detailed descriptions of intrusions and abstract distribution models for applications, protocols, or lower level network entities. Real traces are analyzed to create profiles for agents that generate real traffic for HTTP, SMTP, SSH, IMAP, POP3, and FTP. The profiles can be used to create datasets with desirable characteristics and multi-stage attack scenarios.

AWID The prominence of wireless communication technologies and the absence of wireless-specific datasets in the intrusion detection field, led the authors in [28] to present AWID [29]. AWID is a publicly available dataset, which contains real traces of offensive and normal wireless traffic over the WEP 802.11 standard. The authors emulated a SOHO infrastructure hosting a number of legitimate clients and a single mobile attacker. The dataset contains 16 attack classes, which fall into 3 major classes namely, flooding, impersonation and injection attacks, while the instances comprise 156 features.

2.2.3 Evaluation metrics

There is a wide spectrum of evaluation metrics to quantify the goodness of an ML approach and uncover its strong and weak aspects. Classification accuracy is a metric usually used to evaluate the performance of an ML model but, especially in the context of intrusion detection, more complex metrics are needed for ensuring the robustness of a detection mechanism. The evaluation metrics given below are designed to measure the efficacy of an ML-based IDS, which aims to distinctly detect diverse attack classes (multi-classification approach) rather than simply predict between normal and malicious events (binary approach).

Accuracy This metric measures the frequency of correct decisions. It is a fraction of the correct decisions made among all the classes (true positives, or TP_i) divided by the total number of instances in the dataset (N).

$$Accuracy = \frac{\sum_{i=1}^{|\mathcal{C}|} TP_i}{N} \tag{2.1}$$

Mean F-Measure (MFM) F-Measure is used to measure the balance between the precision and the recall. In the case of a multi-classed problem, the mean F-Measure is calculated based on the following formulas:

$$MeanFMeasure = \frac{\sum_{i=1}^{|\mathcal{C}|} FMeasure_i}{|\mathcal{C}|} \tag{2.2}$$

$$FMeasure_i = \frac{2 \cdot Recall_i \cdot Precision_i}{Recall_i + Precision_i} \tag{2.3}$$

$$Precision_i = \frac{TP_i}{TP_i + FP_i} \tag{2.4}$$

$$Recall_i = \frac{TP_i}{TP_i + FN_i} \tag{2.5}$$

where:

- FP_i, or false positives, represent instances with actual class other than the i-th, but wrongly predicted to belong in the i-th class.
- FN_i, or false negatives, represent instances with i-th being the actual class, but falsely predicted to belong to another class.

Average Accuracy It is calculated as the average recall among all the classes of the dataset.

$$AvgAccuracy = \frac{1}{\mathcal{C}} \sum_{i=1}^{|\mathcal{C}|} Recall_i \tag{2.6}$$

Attack Accuracy This metric is used to measure the ability of a model to detect solely the attack classes by not taking into consideration the normal traffic. Index $i = 1$ stands for the normal traffic class.

$$AttackAccuracy = \frac{1}{\mathcal{C} - 1} \sum_{i=2}^{|\mathcal{C}|} Recall_i \tag{2.7}$$

Attack Detection Rate (ADR) It stands for the accuracy rate for the attack classes.

$$ADR = \frac{\sum_{i=2}^{|\mathcal{C}|} TP_i}{\sum_{i=2}^{|\mathcal{C}|} TP_i + FP_i} \tag{2.8}$$

False Alarm Rate (FAR) This metric focuses on the normal traffic and quantifies the FNs, that is to say, normal instances misclassified as attacks.

$$FAR = \frac{FN_1}{TP_1 + FN_1} \tag{2.9}$$

The aforementioned evaluation metric are used to evaluate the performance of the intrusion detection methodologies, which are presented later in Chapters 3 and 4.

2.2.4 Machine learning methods for intrusion detection

This section provides a overview of fundamental machine learning methods, which have been adopted in the design of network IDSs. Additionally, we refer briefly to related works of this field.

2.2.4.1 Artificial Neural Networks (ANN)

Artificial Neural Networks (ANNs) are a technical equivalent of the neural network of animals' brain. ANN structures imitate the connections among the neurons of a brain to create a network of simple, but highly interconnected processing units, which are able to compute output values given a set of input values [30]. The information is traversed through the network via "edges", which connect the artificial neurons and help to interact among each other. These connections are regulated by "weights", which are being adapted during the learning process. Thus, the weights increase or decrease the strength of the connections with the aim of identifying patterns in the input data.

ANNs are comprised by several layers of neurons. The first is the input layer, which receives the input data in the form of numerical vectors. That is, all real-world data must be translated first into this form. The last layer is the output layer. Between the aforementioned layers might exist one or more layers, which are called hidden layers. An

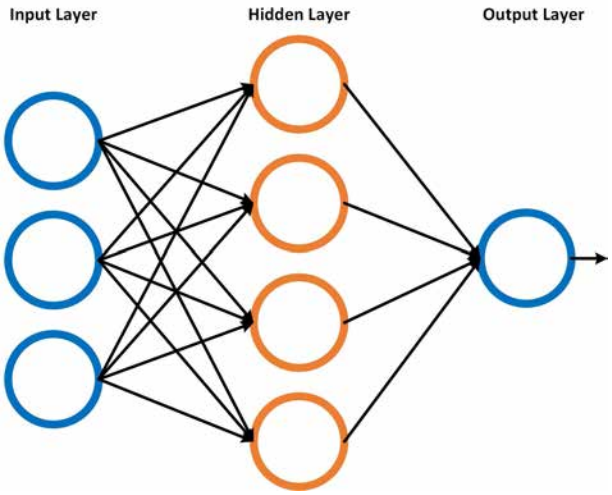


FIGURE 2.1: Example of an Artificial Neural Network

ANN structure that has more than one hidden layers is called deep neural network [31]. An example of a simple ANN structure is given in Figure 2.1.

There are two types of ANNs based on the way the information flows throughout the units, namely FeedForward and Feedback. In the former type, the information flow is unidirectional and passes sequentially from the input layer to the hidden units until the output, while in the latter, the flow can create cycles/loops. The developments in the field led to numerous ANN structures for diverse purposes, such as clustering, classification, regression, reinforcement learning and feature extraction.

The intrusion detection literature has offered a wide spectrum of works which adopt a ANN structure to enable either misuse or anomaly-based detection of offensive events. To name a few, the work in [32] combines ANNs with fuzzy clustering to offer a predictive model called FC-ANN, which performs adequately for low-frequent attacks. By utilizing the fuzzy clustering technique the authors create smaller datasets of KDDCup'99 [21], which are fed to different ANNs. Using a "divide and conquer" approach, the authors advocate that this separation of the dataset helps each ANN to achieve more precise results for the low-frequent classes of the dataset. The results of the different ANN are aggregated using a fuzzy meta-learner to deliver a complete prediction model. An intrusion detection model called MOVCIDS is presented in [33]. This model is able to visualize the network traffic data through a functional and mobile visualization interface, which reveals the internal structure of the data. This approach can provide valuable insights to security administrators. In [34], the authors recruited self-organizing maps for

data clustering and Multilayer perceptron (MLP) neural networks to deliver an anomaly-based detection system.

The developments in hardware accelerated machine learning and the remarkable results of deep learning methods in the field of image and speech recognition, triggered the interest of cybersecurity reseach community and led to promising intrusion detection proposals [35, 36, 37, 38, 39]. Note that, one of the major contributions of this doctoral thesis, namely the introduction of self-adaptive properties for misuse IDS, is based on a deep learning methodology presented in chapter 4.

2.2.4.2 Decision Trees

Decision Trees (DTs) are a widely used model of supporting decision making in the context of machine learning. DTs are constructed as graphs, where internal nodes represent conditions for testing attribute values of instances in a dataset with the aim of inferring a target value. This value can be either a district value, i.e., the classes of the instances (Classification tree), or a continues value (Regression tree).

In the case of classification trees, each leaf represents a class that classifies an instance if its input values satisfy the conditions of the nodes that sequentially construct a path from the root to the corresponding leaf. This path can be represented as an IF $\{Conditions\}$ THEN $\{Class\}$ rule, as shown below. Thus, the simplicity of the decision rules enables one to easily understand the conditions and the outcome.

$$IF\ \{Condition_1\ is\ Value_1 \wedge \cdots \wedge Condition_n\ is\ Value_n\}\ THEN\ Class = C$$

Several classification algorithms have been proposed for building DTs. Amongst them, C4.5 [40], ID3 [41] and CART [42] are considered as standards in decision making and in the context of IDS. Such algorithms are used in order to create a DT based on training instances and then their classification ability is measured during a testing period on previously unseen data.

On the downside, DTs pose some notable limitations and challenges. DTs algorithms usually generate complex structured DTs and pruning techniques should be adopted to

minimize the size of the trees, while discretization techniques have to be applied on continous dataset features. Moreover, algorithms driven by information gain to construct a DT are biased toward the major classes of a dataset achieving low classification accuracy for those classes represented by a smaller percentage of instances in a dataset [43]. This poses the main challenge in the context of IDS as network traffic data follow this behavior.

In the context of intrusion detection, several works utilized decision tree structures usually combined with other methods to improve the detection accuracy. For instance, the work in [44] presented a light weight IDS based on a wrapper approach and DTs. The wrapper approach aimed to identify an efficient subset of feature of the dataset to improve the overall performance of the system. The authors used multiple neural networks in an assembled fashion in order to pre-process the dataset and derive a new one. The new dataset was given in C4.5 [40] tree classifier to realize the final detection model. Additionally, the authors in [45] utilized a genetic algorithm to identify a subset of features of the KDDCup'99 dataset in order to maximize the performance of a C4.5 classifier.

Note that, one of the major contributions of this doctoral thesis is the realization of a rule induction methodology, which combines decision trees and genetic algorithms under a novel setup. More details regarding the methodology and relevant tree-based proposals can be found in Chapter 3.

2.2.4.3 Ensemble Learning

Ensemble learning is a machine learning methodology which combines multiple algorithms to create a final predictive model having improved performance. This methodology adopts a divide and conquer approach, but collectively the final model combines the observations of the constituent learning algorithms and offers an end-model which has higher stability and prediction power compared to the constituent ones.

The aforementioned models can be virtually all different from each other. In fact this strategy can provide better results as it increases the diversity among the learners [46]. However, the size of the ensemble learner, i.e., the number of the internal learners, can affect significantly the overall prediction accuracy. In fact, defining the optimal number

of component learners remains an open issue, which is strongly connected to the nature of a problem and the variance and noise of the data. However, according to [47], having as many constituent learners as the number of the classes of the dataset leads to a more accurate end-model.

The most prominent ensemble methods are Bagging and Boosting. Bagging method (Bootstrap aggregating) deploys a set of similar learners, which are fed with random dataset samples. Thus, a model is created for each dataset sample and the final result comes of a voting process, where the votes have equal wights. This approach aids in reducing the variance error. Random Forest [48] is a notable classification algorithm which uses the Bagging approach, but additionally selects a random set of the dataset features for each of the dataset samples.

Boosting is a sequential approach, where each learner aims to revise the misclassifications of the previous one. Thus, the first learner is trained on the entire dataset and each subsequent learner refines the residuals of the one built previously. This technique is known for decreasing the bias error, but it faces over-fitting issues. The most common boosting algorithm is Adaboost (Adaptive Boosting) [49].

Overall, the ensemble methods can offer accurate and robust results due to the combination of multiple prediction models. However, the high accuracy comes at a price, as the computational complexity of the end model makes it impractical for online applications. Additionally, the sequential models obfuscate the interpretabilty of the model and provide less insights to the analysts.

Several works have adopted an ensemble approach in the intrusion detection field. To name a few, the work in [50] built a wide testbed for evaluating ensemble methods in the context of intrusion detection. Specifically, the authors compared the performance of multiple ensemble setups and tested the performance of diverse voting methods to combine the final result. According to their results an ensemble methods based on k-NN and SVM over a Particle swarm optimization (PSO) method provides more accurate results in contrast to the well-known weighted majority algorithm (WMA). Of course, several works proposed IDSs based on the use of Random Forest, as it has been proved a very accurate method, but with interpretability issues [51, 52]. However, ensemble methods are not limited to simple learners combinations as more complex structures can be designed. This is the case for the work in [53], where the authors utilized Artificial

Neural Networks (ANNs), Support Vector Machines (SVMs) and Multivariate Adaptive Regression Splines (MARS) classifiers to provide an end-model with high generalization performance by using the majority voting approach.

2.2.4.4 Evolutionary Computation

Evolutionary computation [54] is a subfield of AI which is strongly inspired by nature. Evolutionary algorithms aim to solve optimization problems by imitating the biological process of evolution based on Darwin's theory [55]. Genetic Algorithms (GA), Evolutionary Programming (EP) and Genetic Programming (GP) are some of the most prominent approaches in this filed. These approaches aim to solve optimization problems through the development of a population of possible solutions (individuals). Thus, given an initial population, a process of natural selection is applied in order to evolve a new population, which includes more optimized solutions than the previous one. Even though the aforementioned approaches have some differences, they share common operations. That is, in this section we focus on GAs [56], as it is a methodology which is utilized later in this doctoral thesis.

In the evolutionary process of GAs, the individuals are selected for reproduction in order to born offsprings for the next generation. Hopefully, the offsprings inherit the beneficial characteristics of their parents, they develop new ones, and they pass them to their descendants. This is an iterative process where the fittest individuals have more chances to survive and pass their characteristics to the next generations.

Each individual has a standard representation of a vector, which contains a set of properties, called genes. Depending on the problem, the genes can be represented by diverse data structures, such as real numbers, bit arrays and tree-like structures, i.e., GAs can have diverse encodings.

The evolutionary process of a GA deploys 5 operations, namely Initialization, Selection, Crossover, Mutation, and Replacement. Apart from these operations, many other operations exist depending on the problem domain. However, these five operations can be found is the vast majority of genetic algorithm implementations.

Initialization: During the initialization step, a random initial population is created. The population contains individuals, each of which is a possible solution for the given optimization problem. The initial individuals have low fitness, but beneficial characteristics will be developed through the evolutionary process.

Selection: During the evolutionary process some of the individuals will be selected for reproduction in order to create a new generation. Typically, a selection process involves an evaluation step for measuring the fitness of the individuals using a fitness function. This function is always problem dependent and aims to quantify the quality of a candidate solution. Typically, a selection process gives higher chances to fittest individuals to be selected as parents.

Crossover: This operation (also known as recombination) aims to exchange a piece of genetic information between two parents and generate a new offspring. The parents are derived from the selection process and the crossover operation is applied usually on a randomly chosen gene of their structure.

Mutation: This operation changes one or more genes of the individual for the sake of introducing and preserving the diversity in the evolutionary process. The mutation occurs probabilistically on a randomly chosen gene of the individual. The mutation prevents the individuals from becoming too similar to each other and thus, adheres new characteristics in the next generations.

Replacement: After the aforementioned evolutionary steps, the replacement operation defines which individuals will be moved in the next generation. That is, a selection process takes place, where the parents and their children are evaluated and the fittest individuals will be moved to the new generation.

Genetic algorithms is a widely used optimization technique that has various advantages. A genetic algorithm can provide a set of near-optimal solutions and not just one solution, while parallel processing can be used in the evolutionary operations. Additionally, the fitness function can steer the evolutionary process to optimize multi-objective problems. Last but not least, GAs always converge to a solution in contrast to other optimization methods. However, GAs pose also shortcomings as they introduce randomness in the process, while they have the tendency to "stuck" in local optima [57]. In addition, the

individuals have to be evaluated repeatedly and this can increase the computational cost.

As described in Chapter 1, one of the contributions of this doctoral thesis is the design and implementation of an evolutionary methodology in the context of network intrusion detection, which is described in Chapter 3. The interested reader can refer to Section 3.6, which provides a discussion of related works that utilize genetic algorithms.

2.2.4.5 Support Vector Machines

Support Vector Machines (SVM) [58] is a widely used machine learning algorithm, which can achieve high performance results based on a simple concept. The objective of this algorithm is to find a hyperplane that distinctly classifies the data points of a given dataset in an N-dimensional space.

In a simple example where a dataset contains two classes, the objective of an SVM algorithm is to define a plane that has the maximum perpendicular distance (i.e., margin) between the closest points of both classes. These points are called the support vectors. The wider is this margin between the hyperplane and the support vectors, the higher is the generalization ability of the predictive model.

In principle, the discrimination of data points, i.e., the definition of an optimal hyperplane, in real data is a difficult task, as those may not be linearly separable. Thus, transformations are applied to augment the input dimensions of the problem with the aim of achieving the class separation. These transformations are called kernels. Several kernels can be used to transform the input space into higher dimensions, such as a Polynomial Kernel and a Radial Kernel. This is called the Kernel Trick [59].

An SVM model needs to search for the coefficients of the hyperplane. This is done using optimization procedures. The most popular method for training an SVM classifier is the sequential minimal optimization algorithm (SMO) [60].

SVMs are designed to address binary classification problems. However, multi-classification problems can be solved by breaking down the problem into multiple binary classification tasks and combining the final result. Common approaches in this context are the *one-versus-all* or pairwise classification *one-versus-one* [61, 62] approaches.

In the context of intrusion detection, the work in [63] uses an SVM classifier to build an anomaly IDS. The training process of the classifier is enhanced using Dynamically Growing Self-Organizing Tree (DGSOT) algorithm for clustering analysis in order to pinpoint the support vectors. Thus, according to the authors, the SVM training becomes faster and achieves higher generalization, while it achieves higher scores in detecting attacks compared to other legacy methods. A similar approach was used also in [64], where BIRCH hierarchical clustering algorithm was used to create a smaller KDDCup'99 dataset [21] with abstracted data points, which aid the SVM classifier to build a more accurate model. Four SVMs were trained separately to cover all the major classes of the dataset. Additionally, the work in [65] presented an IDS which uses an GA-based optimization methodology to define an optimal feature subset and the kernel parameters (C, γ) of an SVM classifier. It can be inferred that in the context of intrusion detection, SVMs can achieve great results but they require demanding preprocessing and parameter optimization steps due to the complex nature and the high input dimensions of the data. Additionally, the design of multi-classed detection approaches require the combination of multiple binary classification tasks.

2.2.4.6 Swarm Intelligence

Swarm Intelligence (SI) is a subfield of artificial intelligence, which focuses on the collective behavior of distributed, self-organized artificial systems [66]. SI systems are inspired by nature and involve algorithms imitating the behavior of insects or animals. Typically, these algorithms involve a population of agents, which interact with their environment autonomously and in a distributed way, but their actions intend to serve the goals of the population. In fact, the agents, as individuals, may act with stochasticity, randomness and messiness as ants, bees and birds do. However, when act in synergy they are able to converge to a problem solution successfully, and notably, without the presence of any single authority for instructing the right actions [67].

Among others, Ant Colony Optimization (ACO) [68] and Particle Swarm Optimization (PSO) [69] are the most well-known classes of SI algorithms, which have contributed in the intrusion detection field.

ACO is an optimization algorithm inspired by the behavior of an ant colony and it was firstly introduced by M. Dorigo et al. in [68]. The ants have the ability to locate the

shortest path from their nest to a food source. Notably, ants have limited vision, while they lack the ability to talk [67]. Instead, their coordination is based on the sense of smell and the pheromone exudation. The ants roam in the quest of food. Once an ant detects a food source, it deposits pheromone on its way back to the nest. Soon, more ants will carry food to the nest by depositing pheromone. Eventually, the ants converge to the shortest path, which is the one having the higher concentration of pheromone. By following these principles, agents roam through the search space of a problem and once a solution is found, they converge to the optimal one.

ACO has been used in numerous intrusion detection proposals. In [70], a distributed IDS for wireless networks was proposed. The authors developed a classification rule induction methodology, called TermID, based on ACO. Their experiments conducted using AWID wireless dataset [28] with the aim of constructing a list of rules in a distributed manner via data parallelism to ensure privacy preservation. In the same spirit, the work in [71] used Ant-Miner [72] to form a new algorithm, called Ant-Classifier, for inducting prediction rules for misuse detection. The authors introduced a Multiple Ant Colony Optimization (MACO) methodology to tackle the limitation of Ant-Miner, where the ants tend to favor one class of a dataset. That is, MACO assigns one colony to each class. Additionally, ACO-based algorithms have been utilized in feature reduction approaches [73] or for defining smaller but representative datasets to improve the training process of IDSs [74].

PSO draws inspiration from the coordinated movement of bird flocks or fish school [75]. PSO is a global optimization algorithm, where particles move through the n-dimensional search space guided by a fitness criterion. In fact, each particle in the flock represents a possible solution. The movement of each particle is affected by its local best position and the best known position of the flock. Thus, every time a better position is found, it is communicated among all particles so that to improve their movement towards a global optimum.

PSO offered a variety of intrusion detection solutions. The work presented in [76] proposes a variation of PSO, namely Simplified Swarm Optimization (SSO), for building an anomaly-based IDS. The authors implement a Weighted Local Search (WLS) strategy, which improves the position update strategy of each particle an leads to more efficient detection rules. The authors in [77] proposed an IDS based on a method which combines

binary particle swarm optimization (BPSO) and support vector machine (SVM). The BPSO is used to optimize the problem of finding simultaneously the optimum features and parameters for the SVM. Additionally, the authors in [78] make use of time-varying chaos particle swarm optimization (CPSO) with the aim of optimizing the parameters and feature selection of Multiple Criteria Linear Programming (MCLP) and SVM classifiers. According to the authors, by adopting the time-varying inertia weight factor (TVIW) and time-varying acceleration coefficients (TVAC) they avoid being stuck to local optima during the optimization process. Thus, the detection efficiency of the proposed systems surpasses those of legacy PSO and CPSO approaches.

2.3 Intrusion reaction frameworks for optimal countermeasures provision against cyber attacks

This section details on AI-based IRSs which aim to support the decision making when ICT infrastructures face offensive incidents. Such systems aim to provide optimal selections of countermeasures that fit to the nature of ongoing attacks occurring against the protected infrastructure. Nowadays, there is more than ever a need to reactively counteract intrusive events upon their occurrence in order to dynamically eradicate potential consequences on the protected systems with the aim of minimizing the security risk and financial losses. In this section, we will refer to this approach using the term *Countermeasure Strategy*, explaining our understanding of the problem and underpinning its major components.

2.3.1 Introduction

Battling against intrusive incidents occurring in a network infrastructure is a challenging task due to several parameters, including the attacker's level of expertise, the variety and sophistication of attacks, the network size and topology, the number and diversity of zero-day vulnerabilities, the robustness of the deployed IDS, etc. [79, 80, 17, 81]. Security administrators bare the burden of dealing with such demanding tasks, and most of the time, they have to manually react against intrusions while having security budget constraints and a strict reaction time frame. Putting it another way, security administrators often face multi-criteria decision making (MCDM) problems that have to

be solved in a timely and cost-effective manner [82]. Also, by taking into consideration other fundamental parameters of the problem, one can safely argue that it is almost infeasible for a human to appropriately deal with all these requirements. As a result, efficient fully or semi-automated decision support systems are needed to address the resource bottleneck by human operators and security officials [83].

Several works in the literature propose cost-benefit quantitative approaches in applying optimal countermeasure strategies. At a high level, all these contributions share a common goal; they define an optimal set of countermeasures to counteract cyber attacks. In this direction, a plethora of methodologies have been introduced by incorporating a diversity of fundamental notions. More specifically, as detailed in the following sections, cyber attack modeling methodologies such as Attack Graphs (AGs), Attack Trees (ATs) [84] and other graphical structures are used to accurately portray the interconnections and possible dependencies among the various network assets. In particular, ATs aim to formally represent the security states of a system and to visualize the different ways in which it can be attacked using a tree structure. On the other hand, AGs combine information pertaining to network topology and possible vulnerabilities and exploits, with the aim of providing a visual representation of the attack paths that a potential aggressor could traverse in an effort to reach a specific network target. Alongside with these graphical representations, several probabilistic models have been used to describe the system's security state transitions, which constitute the actual search space of the problem. To do so, several optimization algorithms and quantitative risk assessment methods have been recruited, and combined together, to deliver competent solutions able to provide optimal sets of countermeasures regarding the system's security states. In a nutshell, the works presented in the literature can be divided into two major categories, namely *static* and *dynamic* reaction systems. The former are used to proactively secure a monitored system, while the latter are destined to operate reactively upon the occurrence of an attack incident [85].

The numerous solutions and their diverse above-mentioned characteristics spur us on to provide a comprehensive analysis of the pertinent literature and present the state-of-the-art frameworks in this ecosystem. Specifically, this chapter focuses on works which aim to provide countermeasure recommendations as a result of automated processes driven by quantitative security metrics. By going through the literature, we identified 7 common basic qualitative features (detailed in Subsection 2.3.2.2) and used them as the

basis for comparing the capabilities of the proposed frameworks. The goal of our analysis is to identify the pros and cons of proposed solutions and to shed light on the limitations of this particular field. That is, by breaking down the proposed solutions and comparing their characteristics, we identified several research challenges that should be taken into account by researchers intending to contribute in the field. Our analysis emphatically focuses on the methods and theories applied fundamentally on the background of such solutions. Moreover, the reviewed works are field-independent as their concepts could be applied in a wide range of ICT domains

The rest of this chapter is organized as follows. The next Section presents an in-depth overview of the problem and the qualitative features used to compare the various works are given in Section 2.3.2. A detailed description, analysis, and comparison among the works gathered from the literature is included in Section 2.3.3. The last but one Section provides a discussion on the research challenges in the field and offers pointers to future research. The last Section concludes by summarizing the most significant findings of our work.

2.3.2 Problem Statement

As already pointed out, the goal of any reactive system is to assist the security administrators in the decision making for counteracting possible security incidents. Aiming at providing a holistic view of the problem, this section elaborates on the core ingredients of a *Countermeasure Strategy*, while at the same time underlines the most important features included in the various surveyed studies.

2.3.2.1 Countermeasure strategy

Current information systems contain a plethora of assets along with the associated security controls which aim to ensure a specific level of security for each of them. The volume of information produced by these controls is the baseline that has to be used for building a robust defense strategy, but at the same time, it represents also a problem by itself given its huge size.

In this context, security administrators have to face many security issues, including multistep, new and sophisticated type of attacks (or polymorphic ones [86]), asset exposures,

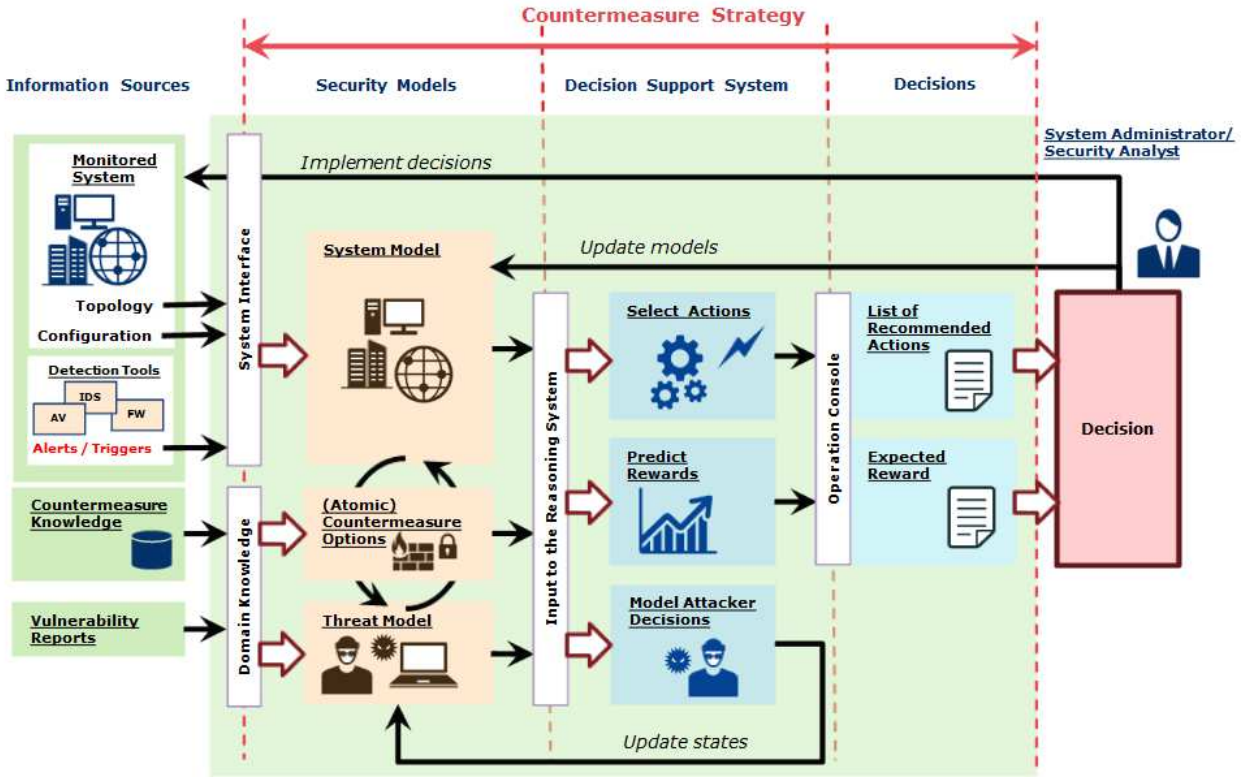


FIGURE 2.2: An abstract view of countermeasure strategy architecture.

distributed and heterogeneous physical network topologies, etc. Moreover, they have to work within specific budget constraints that may preclude them from implementing all possible hardening measures or even those that provide remedies to hopefully cover the great majority of weaknesses. In addition, their decision must consider time limitations, because usually in any reaction strategy the time is of essence. Thus, system administrators need to find the optimal trade-off between the implementation cost of a specific countermeasure and the overall security level of the system, also known as system administrator dilemma [87]. In the context this doctoral thesis, we define the term *countermeasure strategy* as follows.

Definition. [Countermeasure Strategy] A generic set of methodologies, procedures, and processes that aim at reacting to security incidents in a given system and eradicate them.

As is detailed later in Section 2.3.3, so far, many approaches have been presented in the literature to address the problem of finding the optimal combination of security and cost parameters in order to achieve the optimal collection of countermeasures. Based on

our analysis, Figure 2.2 shows the basic components of a countermeasure strategy. We define them as follows:

- *Monitored system*: the physical system to be protected; from this system, core information is extracted, including *network topology* and *asset configuration*.
- *Detection tools*: collection of tools that send all events happening within the monitored system to the appropriate controls of the countermeasure strategy. Such events include intrusion alerts, software updates, hardware installations, and so on. Examples of these tools are IDS, *Antiviruses* (AVs) and *FWs*.
- *Countermeasure knowledge*: initial raw knowledge about the reaction steps which must be triggered to cope with security issues. As detailed in Sections 2.3.4.2 and 2.3.4.3, this knowledge is acquired from external sources and the security administrator, and is typically represented in heterogeneous formats. Few examples of such ilk of countermeasures are "close TCP/UDP ports", "redirect incoming traffic", "apply a patch", etc..
- *Vulnerability reports*: it represents knowledge on the vulnerabilities. The sources of this information are the expertise of the system administrators, the public available databases, such as CVE [88], and the open threat sharing networks [89].
- *System model*: given the information gathered from the monitored system, such as network topology and configuration, a model which synthesizes these pieces of data is created to report the core parameters which will be used for further analysis. This model uses *architecture description languages*, like ArchiMate [90], to conceptually model the structure, behavior, and components of the system [91].
- *Atomic countermeasure options*: given the raw knowledge about the remediation steps, represented in the countermeasure knowledge, a list of possible countermeasures is created, trying to combine the above-mentioned remediation steps to effectively defeat possible attacks.
- *Threat model*: based on the vulnerability reports and system model, a threat model is created, which represents possible attack patterns. As detailed in Section 2.3.2.2, AGs and ATs are prominent examples for this model, as they are usually employed as a formal representation which aims to describe possible attacker's actions against the monitored system[84].

- *Select actions*: in this component, the selection of the specific countermeasures is conducted, balancing the existing trade-off between the security level of the system and the cost of the reaction. Specifically, this component considers not only the monetary cost of activating the selected actions but also the possible negative impact due to their enforcement, including the possible availability decrease for one or more services. Therefore, for a specific attack, a set of possible countermeasures are analyzed and ranked, trying to maximize their effectiveness and to minimize their cost.

- *Predict rewards*: depending on the selections done by the previous component, a calculation of the produced security and economic benefit is performed and forwarded to the system administrator.

- *Model attacker decisions*: based on the choice of the "Select Actions" component, the threat model is updated, so as to be able to predict similar kind of attacks. When a countermeasure is selected, the attack patterns are modified reflecting this choice.

- *List of recommended actions/expected reward*: based on the selection made by the decision support system, a list of countermeasures is prepared to be presented to the system administrator. This report must include the expected revenue derived from the implementation of the selected countermeasures, such as an improvement of the overall risk level of the system or an economic reward due to the attack blocking.

- *System operator decision*: by getting the previous list and associated data, the administrator may decide to implement the reaction procedure on the monitored system, and update the configuration on the previous models (i.e., system and threat models).

It can be safely argued that the presence of the aforementioned components in the above abstract strategy is expected to result in more accurate and efficient countermeasures. This is basically because the different components can offer a modularized but holistic view of the problem and provide the correct kind and amount of information for aiding the administrator to decide optimally on an appropriate reaction plan. In this vision of the countermeasure strategy, the role of security administrators is of major importance.

This view relies on the fact that the administrator not only has a deep knowledge of the particular system but also the responsibility and the privileges to take an important decision. Moreover, the administrators usually manage system hardening procedures with budget limitations, hence they need to balance the trade-off between impact and cost of the countermeasure's enforcement and system security level. This procedure is twofold; on the one hand, it represents the actions which will be implemented on the system. On the other, it represents an important feedback for the decision support system. This learning process for the decision support system is crucial, since this component has to provide a decision on which action must be implemented on the system maximizing the reward.

2.3.2.2 Features for comparison of works

So far, numerous works have been proposed in the literature dealing with a countermeasure strategy [92, 93]. Most of them share a common ground but also exhibit significantly different features in terms of the adopted system representation and countermeasure derivation methodologies. To this end, after analyzing the various works, we have identified the following 7 features that can be used as a basis for our analysis. Note however that the scope of countermeasures selection is moderately wide. Hence, it is unlikely to find solutions that simultaneously address all the posed challenges of the field, while the process of extracting the necessary information to enable a foolproof comparison proved to be demanding. The aforementioned challenges led us to conclude to the following comparison features:

- Attack modeling
- Countermeasures provision techniques
- Outcomes assessment
- Type of reaction
- Used standards
- Automation level
- Performance

It has to be mention that we treat the aforementioned features as having an equivalent importance weight. That is, since each work may have different characteristics and it may have been evaluated in a specific environment, we preferred not to assign diverse (unequal) importance weights to the identified features, as this could bias unjustifiably

the provided analysis. To this end, Section 2.3.3.14 offers a comprehensive comparison of the various works analyzed as part of this survey based on these equivalent features.

A) Attack modeling An attack model is a formal representation aiming to describe possible attacker's actions focusing on the vulnerabilities and misconfigurations which are present within systems [94]. This knowledge about the possible attack steps is essential to counteract malefactors. That is, via these steps it is possible to block the most probable paths that an attacker can follow, eradicating in this way the intrusion. Next, we describe the most common modeling techniques, highlighting on their advantages and drawbacks.

Attack Graph (AG) This technique is widely used in the literature for modeling cyber attacks [95]. Its popularity is due to its ability to synthesize several system-related aspects to construct a complete representation of the infrastructure intended to be protected. AGs combine information about the network topology, possible vulnerabilities, and exploits appearing on the assets of the network, aiming to provide a visual representation of attack paths that an attacker could traverse in an effort to reach specific network targets. In other words, an AG visualizes the vulnerability dependencies in a network and enumerates its possible states. The states of the network are represented as nodes in the graph, while the edges represent the interconnections among them. The edges reveal a *cause-consequence* relationship between the nodes of the graph. AGs enable the defender to identify the weaknesses of a network and/or pinpoint risky paths on the graph, so that to proceed to the necessary actions removing or remediating nodes and/or edges that threaten the network assets. Figure 2.3 depicts an AG describing a possible File Transfer Protocol (FTP) buffer overflow attack carried on asset 1.

This technique poses limitations as well. Discovering all the dependencies in a network cannot be considered an easy task, thus inefficiencies may appear. Moreover, a known limitation is the scalability of the graphs [95]. As the size of the network increases, an AG becomes bulky since the dependencies among the nodes are numerous. As a result, the defender faces difficulties in understanding and analyzing the graph. Several tools are available to create AGs such as NetSPA [96] and MulVAL [97].

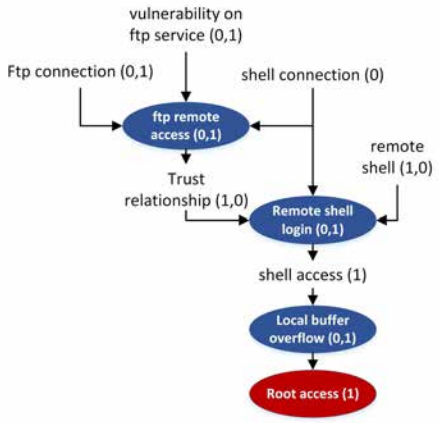
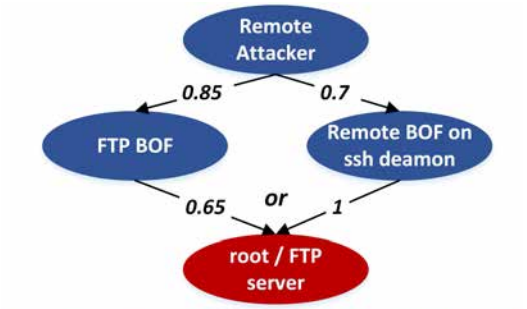


FIGURE 2.3: Attack graph showing ftp buffer overflow attack executed by asset 0 on asset 1

Bayesian Attack Graph (BAG) [98] can be considered as an extension to legacy AGs. Specifically, this type of AGs introduces probabilities on the edges for modeling the uncertainty in the state transitions. In the example depicted in Figure 2.4, the vulnerabilities discovered in a system are marked with likelihood values on the edges, representing the overall probability for an external attacker to exploit them. Then, the score for the final goal is computed considering the possible combinations of the previous conditions, which in the example are presented in disjunction. In this way, BAGs are able to take into account the likelihood of exploitation for a certain vulnerability with finer granularity. On the downside, this technique inherits the limitations of AGs, adding also the computation and assignment of the probabilities as extra parameters, which in turn augments the overall complexity.



ssh	ftp	Pr (root)	Pr (¬root)
1	1	1	0
1	0	0.65	0.35
0	1	1	0
0	0	0	0

FIGURE 2.4: Bayesian attack graph showing ftp and ssh buffer overflow attacks with an OR condition on the target node

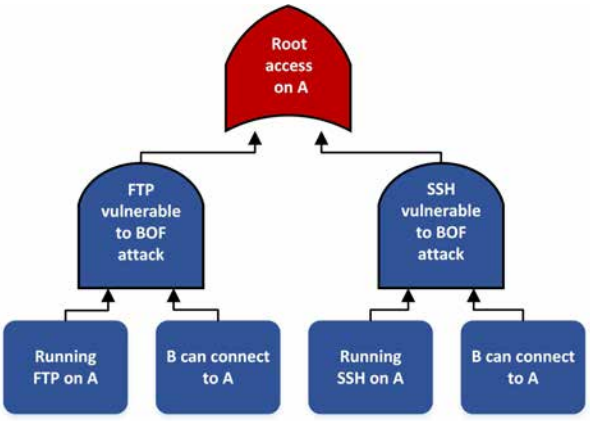


FIGURE 2.5: Attack tree showing ftp and ssh buffer overflow attacks with an OR condition on the root node

Attack Tree (AT) ATs introduced in [84] aim to formally represent the security states of a system and to visualize the different ways in which it can be attacked. The root of the tree represents the attacker's ultimate goal, while its leafs correspond to the entry points for the attacker. The intermediate nodes are the attacker's sub-goals which are connected with AND/OR conditions. These conditions create multiple paths that connect the leafs of the tree with the root. An example of AT is shown in Figure 2.5, where SSH and FTP buffer overflow attacks are connected with an OR condition on the root node. In the AT representation, the nodes usually include also values (either continuous or nominal) to describe the attack paths in a more detailed way. The attack likelihood, the financial cost of exploitation, time, and the cost of defense resources are just some examples of node values. Using these values and starting from a leaf node, the defender is able to sum up the total defense cost up to the root. As with the previous technique, an AT can become large and complex containing many nodes and numerous paths between the root and the leaves. Therefore, in large and complex topologies, where the interconnections of the network assets and the possible vulnerabilities are numerous, the increased number of security states could result to ATs with hundreds or thousands of different paths. As a result, the addition of only one security state (i.e. a new node) results to several new interconnected paths, leading to state explosion in the search problem. In addition, due to the fact that the root can represent only one ultimate goal, it is reasonable to say that several trees may be needed in order to create an holistic security overview of an infrastructure. This can result in a forest of ATs.

Among others, extensions of ATs are *Attack Countermeasure Tree* (*ACT*) [99], *Attack Response Tree* (*ART*) [100], and *Attack Defense Tree* (*ADT*) [101]. These techniques

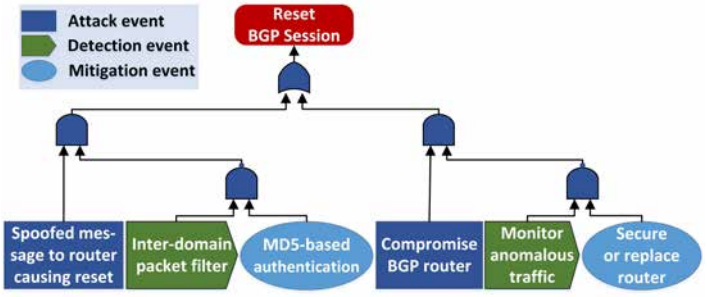


FIGURE 2.6: Attack Countermeasure Tree illustrating an attack against a BGP router

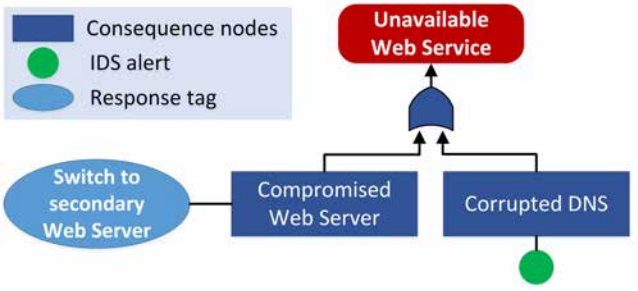


FIGURE 2.7: Attack Response Tree showing an attack against a Web Server

enable particular features keeping the same tree representation. In particular, the ACT formalism uses a non-state-space approach to represent attack, detection, and mitigation events on the same tree structure. These concepts are depicted in Figure 2.6, where the possible steps of an attacker to reset a BGP session are represented using ACT formalism. ARTs use a state-space model (partially observable stochastic game model [102]) to find the optimal set of countermeasure, including both attacks and responses on the same tree. In this regard, Figure 2.7 shows an example of ART to visualize a possible attack against a web service. Lastly, ADT is a node-labeled rooted tree describing the actions an attacker can take to attack a system and the defenses that a defender can employ to protect it. Figure 2.8 illustrates an instance of ADT for a possible attack against a server. Note that the node labeled as "Firewall" represents a defense action against the attack node "Outsider Attack".

Service Dependency Graph (SDG) These are dependency graphs which represent the relationships among multiple services in a system [103]. To exemplify this situation, Figure 2.9 shows the interdepencies for a web mail service. The relationships are defined as privileges, which have been granted to the dependent service from the antecedent one. The dependencies can reveal how a dependent service can be affected in terms of confidentiality, integrity, and availability, if a related service faces an intrusive incident

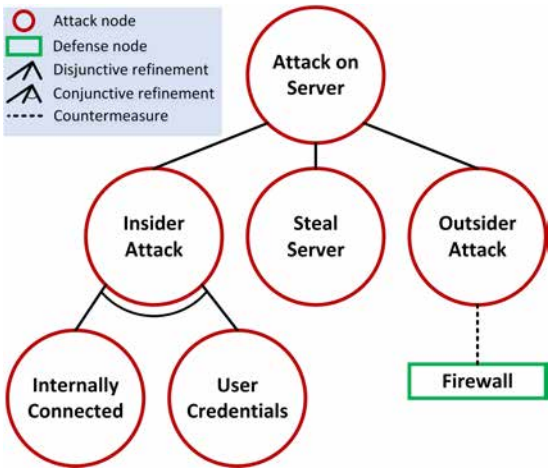


FIGURE 2.8: Attack Defense Tree

[104]. However, identifying and representing the interdependences of all the services in an infrastructure can be proved a cumbersome task, which in turn can lead to inefficiencies.

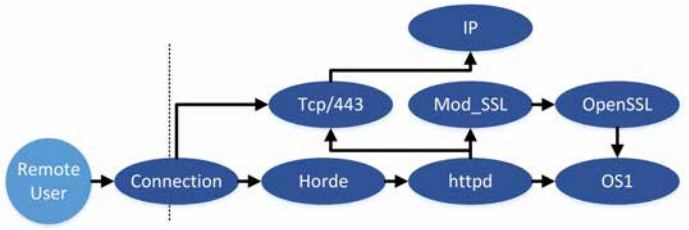


FIGURE 2.9: Service dependency graph for a mail web service

Markov Decision Process (MDP) This technique provides a mathematical methodology to model decision making in situations where outcomes are not totally under the control of the decision maker, since some variables in the process are stochastic [105]. In particular, one can realize a MDP as a *discrete time stochastic control process*, where at each time step the decision maker can cherry-pick between a set of actions for a specific state. A reward is associated with each of these actions. In this way, the probability of moving to a new state is influenced by the previous state and by the selected action. In this respect, MDPs are an extension of the Markov chains [106] by adding to them the concepts of action and reward. In Figure 2.10 an example of MDP is depicted, in which a system is represented using two states and two possible actions, together with the associated rewards. However, identifying all the possible states and actions in a system represents a difficult task mainly due to the complexity of the modern networks.

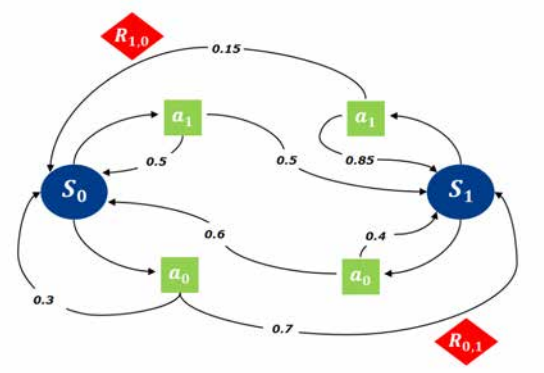


FIGURE 2.10: Markov decision process with two states, two possible actions and associated rewards

The Markov models are often used alongside the other three attack modeling techniques in order to statistically assign probabilities to the paths on the graph. With this probabilistic analysis, the defender can become aware of the most probable paths that an attacker could follow [107].

Among others, extensions of MDP are *Competitive Markov Decision Process* (*CMDP*) [108] and *Partially Observable Markov Decision Process* (*POMDP*) [109], in which Markov processes are adapted to represent different situations. More specifically, CMDP is used to model the system states as a stochastic game where the competitors are the adversary and the defender who both aim to increase their profit. POMDP on the other hand is applied when the states of the system are not entirely observable.

More specifically, CMDP is used to model the system states as a stochastic game where the competitors are the adversary and the defender who both aim to increase their profit. On the other side, POMDP is applied when the states of the system are not totally observable.

For an easy reference, Table 2.4 summarizes all the analyzed attack modeling techniques, highlighting on their main advantages and drawbacks. Both main attack models and their extensions have been summarized. The reader should keep in mind that the extended representations (i.e., those marked with $\triangleright$) inherit the characteristics of the main model or try to address the limitations found in the main representations.

B) Countermeasure provision techniques The ultimate goal of the countermeasure strategies is to come up with the optimal countermeasure or a set of them depending on the events occurring in the system and its current security state. To this end, a

TABLE 2.4: Overview of the presented attack modeling techniques with a focus on strengths and weaknesses

Attack Representation	Strengths	Weaknesses
Attack Graph *[95]*	✓ Holistic view of the system ✓ Visual representation of possible attack paths	✗ State explosion for complex network ✗ Probabilities and defense points are not represented
▷ *Bayesian Attack Graph [98]*	✓ Likelihood on the edges to model uncertainties	✗ Extra computation for the probabilities calculation and assignment
Attack Tree *[84]*	✓ Formal representation of the system states ✓ Visual illustration of possible attacker paths using AND/OR conditions ✓ Absence of defense nodes	✗ Numerous paths between leaves and root ✗ Forest of trees to protect a complex system
▷ *Attack Countermeasure Tree [99]*	✓ Attack, detection, and mitigation events on the same tree structure	✗ Countermeasure nodes cannot be refined over time
▷ *Attack Response Tree [100]*	✓ State-space model, including attacks and responses on the tree	✗ State explosion due to the use of POMDP
▷ *Attack Defense Tree [110]*	✓ Improvements of the tree structure though defense point and countermeasure representation	✗ Detection and mitigation points are represented in an unique node
Service Dependancy Graph *[103]*	✓ Visual representation of interdependencies between services ✓ Quantitative attack impact assessment using CIA attributes	✗ Identification of service dependencies requires huge effort from a security expert ✗ Integration with an attacker-centric representation is needed to model possible attack decisions
Markov Decision Process *[105]*	✓ Representation of decision making process ✓ Concepts of state, reward, and action	✗ State explosion for complex systems ✗ Often used alongside with other attack modeling techniques
▷ *Competitive Markov Decision Process [108]*	✓ System states modeled as a stochastic game between attacker and defender	✗ Computation of the attacker and defender steps augments the problem complexity
▷ *Partially Observable Markov Decision Process [109]*	✓ Representation of unobservable system states	✗ Interaction with the environment to receive information on unobservable states increases the complexity

methodology needs to take into consideration several metrics and balance the trade-off among them in order to conclude to the optimal solution that achieves the desired level of asset protection with respect to the protection cost.

A metric can be defined as "the assignment of a value to the characteristics of an object or event, which, in this manner, can be compared with other objects or events" [111]. Metrics such as the attack cost, defense implementation cost, attack impact, operational cost, and others are only some examples of factors that can be used for defining the optimal solution [112].

For finding the optimal solution, some methodologies consider also statistical-based optimization techniques for optimizing and calibrating metric functions. In this context,

both *single objective optimization problem* (SOOP) and *multiple objective optimization problem* (MOOP) [113] have been proposed to support the decision making in identifying the optimal countermeasures.

In this direction, until now, several methodologies have been recruited. Evolutionary computing techniques, i.e., *Genetic Algorithms* (GAs) [57], have been used in constructing solutions using in their fitness functions synthesized metrics for tuning the solution. Also, *Arm Race* is a bio-inspired technique which describes the competition between two different populations (attacker and defender), which compete with each other to defeat the opponent [114]. The evolution of the competitive populations progresses in parallel. The fitness of the individuals in the one population competes against the fitness of the individuals evolved in the other. Similar proposals utilize *Ant Colony Optimization* (ACO) [68] with graphs with the aim of defining the optimal set of countermeasures based on the pheromone paths constructed by the ants. Precisely, the ants roam probabilistically on the graph based on the probabilities indicated by two parameters, namely the attractiveness and trail level of the move. Note that the latter parameter also incorporates the cost of the trace. Precisely, the ants roam probabilistically on the graph based on the probabilities indicated by two parameters, namely the attractiveness and trail level of the move, where the latter incorporates also the cost of the trace. *Tabu Search* [115] has been also used in this field; this technique tries to find an optimal solution avoiding sticking in local sub-optimal regions.

Generally, when facing an MDP, the *Bellman's optimization method* is usually applied in order to solve it with dynamic programming [116]. Bellman equation solves discrete-time problems regarding the optimal control theory. This is done by optimizing iteratively the objective function and keeping track of the changes. However, other researchers prefer to use *heuristic* implementations for countermeasure selection's algorithms which best fit to their needs.

C) Outcomes assessment In the context of this chapter, we consider the evaluation of the surveyed systems with regard to their outcome, that is, providing countermeasures, as a critical feature in our analysis. To assess the results produced by the analyzed works, we extract two commonly used characteristics regarding the outcome assessment, namely *testbed* and *admin's role*.

Testbed In the context of the reaction strategies, the testing environment plays a significant role, because it directly refers to the applicability of the proposed solution. In this survey, we refer to the used testbed, using the terms *simulated, emulated, and real*.

Admin's role As already stated, the role of the system administrator is central to our vision of automated reaction strategy. Specifically, based on the surveyed works, we identify two distinct roles for the system administrator:

- *Tuning* - The administrator is tasked with setting the goals, objectives and metrics, which are subsequently used by the countermeasure system.
- *Feedback* - The administrator assesses the outcomes of the countermeasure system, selects the optimal solution, and provides feedback to it.

According to the literature, both the aforementioned tasks can be benefited from the use of reinforcement learning [117].

D) Type of reaction Reaction to security incidents can be achieved by following two main approaches, namely *static* and *dynamic*. As static approaches we perceive those which are capable of acting proactively against security incidents, while as dynamic those which are able to react upon the occurrence of cyber threats. These methods differ in the intent, the effort required to implement them, and their outcome from a defender's viewpoint.

Static reaction It deals mainly with *security risk assessment* [118], which represents the process to identify potential security risks that may reside in an ICT system. It begins with the identification of the system characteristics, including weaknesses and exposures, and potential threat sources. The accurate estimation of the amount of risk per asset in the system is a effortful task, and often this judgment is driven by the administrator's subjective belief. This estimation could be useful to locate weak spots during the system design phase, but also proactively, via the use of penetration testing tools[119]. The main limitation of this approach is the lack of a dynamic model capable to follow the flow of events of a potential attacker who could try to exploit system's

vulnerabilities. In such a scenario, countermeasures must be taken on-the-fly, in order to limit if not eradicate the intrusion efficiently.

Dynamic reaction It is concerned with the system response capabilities to a possible ongoing attack. In this case, a deep knowledge of the system vulnerabilities, together with the evaluation of attacker's skills and response time are pivotal factors. Due to its features, this approach covers the limitation of the static one, making more effective the countermeasure selection. On the downside, it requires more computational power. Considering all the parameters which must be taken into account, the task to identify the optimal set of actions for dynamically blocking an advancing intrusion is hard. The probability of the attack paths, the countermeasure's effectiveness as well as their cost are just few examples of the parameters whose computation has to be done in real-time.

E) Use of standards For conducting a quantitative analysis of the adopted reaction strategy, one needs to use *security metrics*, which can measure network security in an objective and cross-platform manner. In computer and network security, a plethora of metrics has been proposed, with the goal of capturing different aspects, including attacks, intruders, network topologies, costs, and vulnerabilities metrics[112].

Nevertheless, to be effective, a metric should belong to a highly shared and used measurement system. The various *standards* help in addressing this problem. The use of standards allows the comparison among published works and solutions, giving quantitative and qualitative measurement of their effectiveness. In the context of this survey, we concentrate on the standards reported in Table 2.5.

CVSS The *Common Vulnerability Scoring System* (CVSS) is an open framework for communicating the characteristics and severity of software vulnerabilities[143]. In the recent years, CVSS has become the *de facto* standard, adopted from the research community to measure the effectiveness of the proposed works dealing with vulnerabilities' impacts and scores [144, 145].

Moreover, great effort has been put to standardize the information flow regarding security threats intending to help enterprises worldwide to use common means of fighting against them. Many of the cited standardization attempts in Table 2.5 are part of the

Security Content Automation Protocol (*SCAP*) [146], which is maintened by NIST, and MITRE corporation [147]. This source database duality sometimes results in redundancies in the standards. For instance, regarding the "cyber threat information sharing and analysis" category, both TMSAD [135] and TAXII [138] aim to create a trust model to exchange information in the field of security automation.

F) Automation level As with the previous features, especially for large network topologies, the great amount of parameters to consider in the process of finding the optimal set of countermeasures often becomes unmanageable. In addition, the response time is a crucial factor, therefore it can be said that a certain level of automation is required in the deployment of a reaction strategy. Until now, several level of automation (LOA) taxonomies have been presented, dealing with human/machine interaction [148]. In the context of our analysis, we classify them into three levels:

- *Manual* - The administrator performs the tasks concerning the reaction strategy, including monitoring the state of the system, selecting the appropriate action to perform and physically implementing it.
- *Semi-Automated* - The system generates a ranked list of decision options based on specific criteria. The system administrator can select one of them, thus giving a feedback used to calibrate the future system decisions.
- *Fully Automated* - The system selects the best option to implement and directly enforces it based on predefined directives given by the administrator.

As observed, the concept of automation is strictly connected to the role of the system administrator. That is, their participation in the countermeasure strategy is a core parameter, not only due to their expertise and privileges, which could lead to a better solution, but also because they have to deal with a specific economic budget.

G) Performance Few will oppose that performance is of great importance in any countermeasure strategy.. This means that quality metrics [149] must be considered when a big project has to be developed. In others words, the reaction plan needs to take into account some inherent limitations of the existing devices in the system.

For example, in an *Internet of Things* (IoT) network topology, the various devices are generally resource-constrained [150].

From the analysis of the surveyed works, we concluded that the performance factors which strongly influence the implementation of the countermeasure strategy are *scalability, time complexity,* and *response time.*

Scalability It is one of the most desirable attributes of a network, system or process. It refers to the ability of a system to accommodate an increasing number of elements or nodes, to process growing volumes of work gracefully, and to be susceptible to enlargement [151]. In Section 2.3.3, we refer to the scalability of the analyzed works as the property of the proposed solution to preserve a polynomial behavior on the response when the number of nodes in the system increases. We classify this factor using the linguistic values in the following set: $\{Low, Medium, High\}$.

Time complexity It quantifies the amount of time taken by an algorithm to run as a function of the length of the string representing the input [152]. In Section 2.3.3, we refer to this factor as the property of the countermeasure provision solution to preserve a polynomial behavior when the input (cardinality of the countermeasure set, number of nodes, etc.) increases. This factor is estimated using the same scale of three values as that in Scalability.

Response time This third factor can be defined as the elapsed time that a computer system takes to respond to a given input. In Section 2.3.3, we define this factor as the ability of the examined solutions to provide a reaction in an acceptable time frame in the context of dynamic reaction. To estimate response time we rely on the following scale: $\{Fast, Average, Slow\}$.

2.3.3 Survey of works

This Section contains a detailed survey of the current major works on the cyber attack countermeasure strategies ecosystem. The survey is given in chronological order with oldest proposals first. For each work, we first provide a succinct description, followed by a constructive analysis of its features.

2.3.3.1 Dewri et al. [153]

Descriprion Dewri et al. [153] introduce an evolutionary technique for defining an optimal defense strategy. They conduct a cost-benefit analysis to determine the optimal defense strategies that have to be taken against dynamically changing attack endeavors by maximizing the *Return-On-Investment* (ROI) index. ATs are chosen as the modeling technique to describe the dependencies among the security states of the protected system. The AT induction methodology used here was firstly introduced in [87]. The authors developed their own solution in which the tree induction process considers as input an initial vulnerability table, the network topology, and online vulnerability exposure databases, namely BugTraq [154], CERT/CC [155], and NetCat [156].

Specifically, the first is an electronic mailing list dedicated to issues about computer security such as vulnerabilities, vendor security-related announcements, methods of exploitation, and possible remedies [154]. The second is the coordination center of the computer emergency response team (CERT) for the Software Engineering Institute (SEI), which researches software bugs that impact software and internet security [155]. The latter is a computer networking utility for reading from and writing to network connections using TCP or UDP. It is also used as a vulnerability scanner during penetration testing [156].

The evolutionary process is based on the non-dominated sorting GA NSGA-II [157]. This algorithm is suitable for applying multi-objective optimization, while sustaining the diversity of the solutions to a high level. The defense and attack strategies are modeled as binary vectors that represent the leafs of the AT. The binary values for a defense strategy signify whether a defense measure is enabled on a leaf (value = 1) or the leaf is unprotected (value = 0). From the adversary's point of view, if a leaf is chosen in an attack strategy, then the corresponding value is 1, otherwise 0. The GA generates defense and attack strategies, while fitness functions are used to infer on the superior strategies.

To conduct a quantitative assessment of the problem, the authors adopt the Butler's multi-attribute risk assessment framework [158, 159]. They introduce several complex metrics which are based on different types of individual metrics like the installation and operation cost of a defense (monetary metrics), system downtime (time), law penalty (severity), and others. In total, authors define the following metrics:

- Potential Damage of an attribute (P)
- Residual Damage (RD)
- Total Security Control Cost (SCC)
- Damage Inflicted (DI)
- Attack Strategy Cost (ASC)
- Breach Loss (BL)

Using P on an attribute of the tree, an augmented AT is built. More specifically, this corresponds to a normal AT which also bares the aggregated cost information to every attribute.

The authors initially define and then evaluate their solution with reference to the following 4 problems: SOOP, MOOP, Multi-objective Robust Optimization problem, and Attacker-Defender Arms Race problem. For the first problem, they define a weighted function that minimizes RD and SCC metrics based on the selection of a boolean vector of possible security controls. For the second problem, the multi-objective strategy aims to minimize several metrics individually, thus the computed solution is more reliable considering the correlation of several parameters. For the third problem, the authors conduct a fault tolerance analysis, where they try to define the most robust defense solution, that is, a solution that is less sensitive to failures in security controls. This is achieved by calculating RD and considering potential failures in these controls. The latter problem differs significantly from the previous ones. Namely, the authors utilize a competitive co-evolution technique to emulate an "arms race" between the attacker and the defender. As already pointed out in the previous section, this methodology incorporates the evolution of two competitive populations in parallel. The fitness of the individuals in the one population competes against the fitness of the individuals evolved in the other. This means that the success of a defense strategy implies the defeat of the attacker and vice versa. The fitness functions are correspondingly based on the Payoff Functions for the Defender (POD) and the attacker (POA). These functions incorporate combinations of the above mentioned metrics introduced by the authors to reflect the benefit of the two competitors. The optimization goal for a defender is to find a defense strategy that maximizes POD against all possible attacks, whereas the attacker aims to maximize POA.

The experimental results for the arms race approach revealed that it was easier for the attacker to find strategies to bypass the measures of the defender. At the same time, a low improvement in the attacker's payoff resulted into a significant drop in the defender's payoff. However, it seems that the system finds an equilibrium point after several generations.

Analysis The authors adopt the AT representation and they also integrate the potential damage metric to build an augmented AT. While this methodology is able to hierarchically represent possible sub-goals of the attacker and enable a quantitative approach of the problem, it remains rather complex. Even worse, the complexity of the proposed solution is increased further as the competitive co-evolutionary process demands the evolution of two antagonistic populations. However, the notion of the "arms race" introduced by the authors reflects the relationship between the attacker and defender and emulates the dynamic engagement between them. Defining an equilibrium point is an appropriate technique to achieve the goal of adequately protecting an asset without over-protecting it. On the downside, the approach of single and multi-objective optimization seems incapable of providing the optimal hardening solution. This is mainly due to the difficulty of tuning the weighted fitness function and the extreme values of *RD* metric produced, which in turn overprotect the assets. The robust optimization approach employed by the authors is an interesting notion, while the fault tolerant scheme they use approaches the problem realistically as it cannot be taken for granted that a defense is unbreakable.

The quantitative evaluation of the proposed scheme is based on a framework proposed in [158] and [159], while the metrics introduced by the authors consider monetary, severity, and time values. This handling is suitable for creating a holistic quantitative evaluation for a scheme. Even so, the metrics employed are not aligned with globally accepted standards, as showed in Table 2.5, to enable a quantitative comparison with other solutions.

A limitation of the proposed methodology is that the attacker's expertise level and the dynamic nature of the network are not taken into consideration. This however may have a severe impact on the implementation of any similar system. Under realistic conditions, a decision maker should adapt their action by considering also the abilities of the adversary, while an accurate representation of the network is vital. The authors'

proposal is not destined to deliver an automated reaction system but a static tool to conduct risk assessment for identifying the exploitation possibilities through the evolutionary approach. Besides, in the experimental phase, the authors consider 19 defenses, whereas the AT has 13 leafs. This is translated into 2^{19} different defense strategies and 2^{13} attacks. It can be inferred that even for a middle-sized network and a considerable number of countermeasures the search space of the problem is being expanded exponentially. It is noteworthy that the countermeasures proposed are limited to disabling and patching actions as this is the case for the majority of the documented works. Finally, the competitive "arms race" approach reflects the relationship between the attacker and the defender, yet defining the equilibrium point could be a really effortful task if the model is to be extended to fulfill the requirements of a dynamic response framework.

2.3.3.2 Poolsappasit et al. [98]

Description In this work, the authors propose a security risk assessment methodology based on BAGs. These can be considered as an extension of AGs as presented in Section 2.3.2.2, adding also a likelihood to the occurring events, which in turn can modify their state. That is, the main difference of a BAG compared to a classical AG is the existence of a non-zero probability corresponding to the case that the attacker is not able to take advantage of the exploit, even if all the preconditions are met. This probability per network node is represented by a *Local Conditional Probability Distribution* (*LCPD*).

In order to compute these probability distributions, the authors use CVSS [143] to estimate the attack likelihood. Specifically, they calculate this probability using three base metrics, namely: *access vector, access complexity*, and *authentication instances*. Using the proposed technique, they are able to perform:

- *Static Risk Assessment* - It identifies system characteristics, potential threat sources and attacker capabilities, often judged by the system administrator with the aim of creating the initial probability values.
- *Dynamic Risk Assessment* - It uses the knowledge about the happened attack incident to update the probabilities using the bayesian inference technique.
- *Risk Mitigation Analysis* - It is aimed at countering risks either or both in a proactive or reactive manner.

Regarding the last point, the authors define a security control as a *preventive* measure that minimizes or eliminates the likelihood of an attack; the enforcement of this control modifies the LCPD of a node, and indirectly influences the unconditional probabilities of other attributes in the network. Therefore, a *security mitigation plan* is intended as a set of security controls, where each of them has a specific cost.

By merging the concepts of LCPD and mitigation plan, the authors define the *Augmented-BAG*, which incorporates the security controls with the expected loss/gain per node. From a system administrator point of view, the derived cost model represents a hard problem to solve. Toward this direction, the authors present *SOOP* and *MOOP*, with GA as means of resolution [157].

The authors include an evaluation section to assess these choices, showing the feasibility of the model in a dynamic context as well. In the conducted experiments, the calculated mitigation plans for the static and dynamic environment present many similarities, suggesting the application of similar sequence of security controls.

Analysis The authors use BAGs to model attack's probabilities into the system. By doing so, they are able to consider also the intrinsic *uncertainty* of a real attack incident. Even though this representation can provide a more detailed and realistic view of the problem, the likelihood assignment and computation are extra parameters that augment the complexity of the AG. Particularly, the algorithm of generation for the BAGs cannot go beyond $O((A \times M)^3)$, where A represents the number of attributes to consider, and M represents the number of machines in the system. Even if the initial AG generation is a one-time cost, for a large-scale and power-constraint network this complexity may be unsustainable. Furthermore, the computation of the marginal probabilities on the BAG is even more challenging; both for prior and posterior probability, when thinking of a dynamic scenario, the algorithmic complexity is exponential. It is therefore obvious that this complexity should be decreased by, say, using a heuristic-based algorithm to realize a dynamic reaction in an acceptable time.

The authors also demonstrate the methodology of providing countermeasures using the cost model as a core parameter. Specifically, the problem is focused on the *system administrator's dilemma* [87], which states that the administrator often has to deal with a limited budget that could preclude them to implement all the possible hardening

measures. Thus, they demonstrate the need of optimization techniques to provide a priority ranking of the countermeasures. They claim that by using a greedy selection or the subjective belief of an expert, the results may be inaccurate. So, to solve this problem, they propose the use of GAs and demonstrate that this methodology presents many advantages both for SOOP and MOOP, which do succeed in achieving a more precise classification of the security controls. One can safely argue that this procedure is extensible for a bigger set of hardening measures, making the process more realistic for a real network.

The proposed reaction strategy by the authors is mainly static. For a given network, they show how it is possible to plan a mitigation strategy for the security incidents that decreases the overall risk level, taking into account the cost model. They also test their framework in a dynamic environment, and via the use of two different attack scenarios, they compute the corresponding reaction plan. It can be said that further experimentation is needed in this direction, considering as a core parameter also the *time of the reaction*, that cannot be easily considered using BAGs.

2.3.3.3 Roy et al. [160]

Description The work in [160] aims to provide a cost-benefit countermeasure system for dealing with cyber attacks. By capitalizing on their previous work in [99], authors rely on ACTs with the purpose of providing a scalable solution for the problem at hand. According to the authors, ACTs perform better than ARTs which use POMDP as a solution technique. This is because the latter model leads to state-explosion issues. More specifically, ACTs provide a non-state-space approach, and according to the authors, this technique is less expensive than state-space driven approaches.

In their model, ACTs are able to represent three types of events as internal nodes in the structure, namely, *atomic attack*, *detection*, and *mitigation*. The authors build two example trees to conduct their experiments and to evaluate and compare the performance of their model against the ART-based solution proposed in [161]. They also suggest two algorithmic approaches to define the optimal set of countermeasures, namely *explicit enumeration* (greedy approach) and *branch and bound*, and they introduce the following four objective functions:

- Selecting the minimum number of countermeasures.
- Selecting countermeasures by minimizing the Security investment Cost.
- Maximization of profit in terms of ROI for a set of countermeasures.
- Multi-objective function for minimizing the probability of attack success and security investment cost.

The first objective function may not be able to provide an optimal set of countermeasures for an attack due to limitations, including security budget constrains or lack of countermeasures for specific attacks. In this case, partial solutions for defending only a critical part of the infrastructure should be taken. The use of the second objective function can restrain the countermeasure cost to specific investment cost. According to the authors, the next two functions pose computational challenges to the proposed framework as the optimization problem is non-linear. To compress the computational requirements, the authors use Watters' transformation [162]. Specifically, this transformation converts zero-one polynomial formulations to equivalent linear zero-one formulations by simply adding constraints on the product terms. This downgrades the problem to a linear integer programming one.

The authors provide experiments to prove the ability of their proposal to converge fast to an optimal solution. The results show that their framework can yield a solution in 38 seconds for an ACT of 5000 leaf nodes. Specifically, the authors noticed that for large ACTs, the first objective function tends to apply countermeasures to the higher nodes of the trees. For the rest of the functions, the countermeasures are applied at the lower levels as the goal is to minimize the security cost and measures at higher nodes tend to be more sophisticated. Also, the authors give a comparison between their solution and the Response and Recovery Engine (RRE) framework proposed in [161] (recall that the latter uses an ART structure). Their approach resulted to an optimal solution in approximately 17 seconds, while RRE needed approximately 3 minutes when using the third objective function.

Analysis The authors make use of an ACT structure to model the system's atomic attacks and countermeasure events with the intention of protecting the goal residing in

the root of the tree. For a small-scale network consisting of 7 hosts (with 12 vulnerabilities each) the proposed solution converge to an optimal solution in a reasonable time frame. Particularly, the ACT-based method computes an optimal countermeasure set within 17 seconds ± 2 seconds. The authors technique to cast the optimization problem to an integer programming one with linear complexity is reflected to the improved performance of the system. However, even if ACTs perform better than ARTs, it has to be stated that the testbed used corresponds to a small-scale network topology. Another potential limitation of this work is the number of ACTs needed to represent the possible attacker's goals in the system. As every tree has one root, one can argue that for protecting multiple assets or properties in the infrastructure there is a need to deploy several ACTs, thus increasing the computational requirements.

Albeit the authors introduce a multi-objective function, they do not elaborate on the time needed for their proposal to provide a solution when using it. However, a multi-objective function is of major importance in a countermeasure system, where the optimal goal is to define optimal solutions which achieve a balanced trade-off between the several metrics. In addition, the diverse cost functions used to compose the objective functions do not follow any specific standard, while the attack cost and attack impact values used are derived from rather aged works.

The ability of the system to adapt dynamically during an attack incident is questionable. Specifically, the authors claim that ACT could be transformed in a hierarchical model that supports Markov chains for modeling sequential attack events. Even so, no evaluation results are given on this aspect of the problem, and thus the authors' proposal can be characterized as static. The administrator's engagement level is not defined by the authors, but we can infer that the creation of the ACTs has to be supervised by a human. This is because a decent knowledge of attack patterns is needed for identifying the internal nodes, the relationships among them, and finally the objective residing at the root of the tree.

2.3.3.4 Viduto et al. [163]

Description This work proposes a novel *Risk Assessment and Optimization Model* (RAOM) to cope with the problem of security countermeasure selection. Considering the budget limitations and the high demand in security measures, the authors present a

model which addresses these issues and at the same time maintains an acceptable level of performance.

Based on NIST SP800-30 [164], which aims to provide guidance for conducting risk assessments of federal information systems and organizations, the proposed model offers a quantitative improvement to it mainly for conducting impact analysis and risk determination. For the first task, the authors calculate the impact of the identified vulnerabilities in relation to *Confidentiality, Integrity, Availability* (CIA) impact, expressed by CVE [88]. The different combinations of impact levels (Partial, Complete, None) are grouped into an impact scale with discrete level [10, 50, 100]. For the risk determination task, the methodology proposes to calculate a *Total Initial Risk* (TIR), which is defined as the initial risk in an organization when no security countermeasure has been applied. During this procedure, an analysis of the system vulnerabilities is conducted with the purpose of identifying their source and properties. This stage can be executed in different ways; specifically, using automated vulnerability scanning tools, performing penetration tests on systems, employing vulnerability modeling techniques, and assessing ICT documentation of a previous risk assessment. The next step in the model is to perform a threat analysis, because the idea of the authors is that vulnerabilities can only be translated into risk if there is a threat able to exploit them. To gather information about threats, they suggest to use historical databases of previous attacks recorded by the organization, or to use threat modeling techniques, such as AGs or ATs. A matching between threats and vulnerabilities is conducted to estimate the likelihood of a threat over a specific vulnerability. Also, in this case, the likelihood can adopt values in the discrete range [0.1, 0.5, 1], where the values of 0.1, 0.5, 1 correspondingly represents low, medium, and high likelihood of a specific threat exploiting a certain vulnerability.

Then, the authors propose a list of generic countermeasures, following a classification similar to NIST report in [165]. In particular, three categories are presented (Technical, Management, Operational) with four subcategories (Support, Prevent, Detect, Recovery). The countermeasures are then associated with vulnerabilities. Actually, the association method between countermeasures and vulnerabilities they follow has been initially proposed in [166] and later demonstrated in [167], where a matching value is assigned to each countermeasure. This value is within the discrete range [-1, -0.5, 0, 0.5, 1], where [0.5, 1] represents a partial or total addressing of the vulnerability, 0 means

no match, and [-1, -0.5] implies that the application of the countermeasure directly or indirectly creates another vulnerability.

Each of the listed security measures has an associated cost of implementation, namely purchase cost, operational cost, training cost and manpower. The total cost for the security controls, together with the total risk level of the system, compose the multi-objective optimization function, which must be minimized.

To solve this MOOP, the authors develop a *Multi-Objective Tabu Search* (MOTS) technique [115], where the objective function is evaluated for different values of the binary countermeasure vector. Also, an experimental section is given, where the characteristics of MOTS are compared with the traditional exhaustive search, which can find the exact set of optimal solutions. In particular, comparisons in terms of speed and quality are shown.

Analysis In the described work, the authors introduce a model to optimize the selection of the security countermeasures. Given a set of k generic countermeasures, represented as a single bit in the countermeasure vector, the search space has 2^k possible solutions. Precisely, as the size of the problem increases, that is, the number of the possible security hardening options becomes considerable, the time required to solve the problem increases exponentially.

The solution presented by the authors consists of a multi-objective tabu search. This technique is able to find the Pareto optimal solutions for the problem, avoiding sticking in a local minimum. Using this methodology, they were able to present a stable pool of founded solutions after 8000 iterations of the algorithm. In particular, 54 non-dominated solutions are discovered in 163 seconds. Overall, the authors argue that in comparison to exhaustive search, which finishes its execution after 2466 seconds, the tabu search offers superior performance. Another experiment is conducted on the quality of the solutions. In this regard, the authors show that their algorithm can reach up to 30% of the optimal solutions, and also the rest differ slightly from the optimum. From these results, we can infer that MOTS methodology performs better than exhaustive search, but in our opinion a study should be conducted to compare it with other multi-objective optimization techniques, so that the eventual advantages are clearly visible. In this mindset, Saraha et al. [168] argue that a pure tabu search is not

enough to comprehensively explore the search space, proposing instead a solution which incorporates also some ilk of GA.

The paper deals mainly with risk assessment, which as we stated in Section 2.3.2.2, represents the process to identify potential risks that may be involved in an ICT system. Even though this phase has a crucial role in cyber defense, it represents a preventive action to defeat eventual cyber crooks, lacking of dynamic capabilities to react against an ongoing attack. Obviously, an interesting improvement is to try to adapt this methodology for use in a dynamic environment.

Similar to other surveyed works, the authors neglect attack modeling. They only suggest threat modeling techniques, such as AGs, to gather information about potential attacks. Instead, the assignment of the probability of a specific threat acting over vulnerability is done using logged attack attempts and self-expertise. The value of this likelihood belongs to a discrete interval composed of three values. However, the effectiveness of this procedure is debatable, even for a static type of reaction. That is, the use of historical databases lacks in considering first experienced or zero-day attacks, thus exposing the system to undue risk. Moreover, the know-how of the security experts is a good source of knowledge, which has to be used, but relying on human subjective belief may lead to a number of errors too. In this direction, the use of a decision support system can be valuable in helping the system administrator in the decision making process, so that a limitation to these errors can be imposed.

The authors highlight in their work the possibility of matching countermeasures and vulnerabilities. Note that this idea is not novel in this particular field, but represents an interesting feature of the model. This correlation considers also the chance that a countermeasure can indirectly or directly create a new vulnerability. Nonetheless, this match is effective if a complete knowledge about security controls and vulnerabilities exists. Gathering these pieces of data is an arduous task, so a broader study should be conducted in this direction, since such correlation is useful to build a dynamic strategy.

2.3.3.5 Chung et al. [169]

Description In this work, the authors concentrate on the protection of cloud infrastructures. In such scenario, there are several peculiarities which need to be considered

by the security expert: (1) administrators do not have the complete control over virtual machines (VMs), so they may be unable to patch the system vulnerabilties as in the case of common data centers; (2) cloud users can install vulnerable software on their own VMs; (3) the compliance of the *SLAs* (Service Level Agreements) is a priority, so the reaction strategy to cyber attacks should be included in the Business Continuity Plan [170]. Furthermore, an attacker can benefit from these security issues exploiting vulnerabilities on a much greater number of machines, which can be used as *zombies* to carry out a large-scale Distributed Denial-of-Service (DDoS) attack.

To address these problems, the authors propose *NICE* (Network Intrusion detection and Countermeasure sElection in virtual network systems), a framework which is able to detect and counteract possible attacks against the cloud infrastructure. Taking advantage of the SDN network control approach [171], where network functions can be controlled and programmed through software switches and the OpenFlow protocol [172], the authors deploy NICE with the following constituents:

- *NICE-A* - A mirroring-based NID agent installed on each cloud server, which filters and analyzes the incoming traffic.
- *VM profiler* - It uses the knowledge about services, connections and states to create an accurate profile of each VM.
- *Network Controller* - It supports the programmable network capabilities to realize the virtual network reconfiguration feature based on OpenFlow protocol. This feature is crucial for the entire framework, because it controls also the traffic flows within the network clusters.
- *Attack Analyzer* - It uses a classical AG representation to correlate the alerts stemming from the agents and select the most appropriate countermeasure.

When a vulnerability is discovered or some VMs are identified as suspicious, several countermeasures can be triggered to limit attacker's capabilities. Based on CVSS [143], CVE [88], and NVD [121], the system is able to calculate the score for each vulnerability and use them for constructing on the AG. Based on the above metrics and scores, a *virtual-networking-based* countermeasure pool is built, where each countermeasure is presented with (I) cost, defined in a range from 1 to 5, (II) intrusiveness, which represents

the possible negative impact on the SLAs, (III) condition, which is the requirement to enable the countermeasure implementation, and (IV) effectiveness, expressed as a percentage of probability that the state of a specific node can change after the application of a countermeasure. The optimal countermeasure selection is shown as a MOOP, which has to minimize both the cost and impact while maximizing the benefit (via the use of the ROI index [173]).

The authors offer an initial implementation of NICE in a small public cloud environment, and then they extend their analysis using a bigger private one. They monitor the introduced overhead and the security performance using a VM Security Index (VSI) [174].

Analysis NICE is presented as a framework which is capable to cover two different phases of the network life cycle pertaining to attack incidents, namely detection and reaction. However, the target of this work is the countermeasure strategy, so our analysis focuses on the latter.

The idea of developing a countermeasure system in a cloud environment is innovative. To the best of our knowledge, this is the first attempt to deploy an architecture that is able to detect and react in a virtual scenario. This feature carries also the disadvantage that without a solid background of research, this work could not address all the issues which are present in a real environment. Moreover, the novelty in the usage of cloud infrastructure makes the attacker modeling still incomplete. That is, it is very difficult to find a great number of known attacks in the literature, and this makes also the defense strategy still incomplete.

AG is the selected model to represent the possible attack paths into the cloud system. It is built based on (1) cloud system information, collected from the controller, (2) the virtual network topology and configuration information, which include also the raw traffic data, and (3) vulnerabilities information generated from periodic vulnerability scanning and via penetration testing using the public available vulnerability databases. Once a new vulnerability is discovered or a countermeasure is implemented, the graph is updated. Apart from the scalability problem regarding this attack representation, the authors make a strong assumption; the hypervisor is secure and free of any vulnerability. However, the literature is full of documented attacks targeting it [175].

To select the appropriate countermeasure, the presented algorithm has a complexity of $O(V \times CM)$, where V is the number of vulnerabilities, and CM represents the cardinality of the countermeasure set. In this way, the authors claim that they solve a MOOP based on the ROI index in an effort to avoid a negative impact on the SLAs. A sample of possible actions is presented with subjective values of intrusiveness and cost. Specifically, as expected in a cloud environment, these countermeasures pertain to layer-2 and 3 of the OSI stack of protocols. The exploration of a wide range of possible reaction steps with different values for the suggested security metrics would greatly benefit this work.

The described system incorporates both static and dynamic reaction. To enable them, the authors consider both periodic vulnerabilities and agent-based traffic controls. A dynamic reaction to an attack is presented on a small-scale cloud system, showing that the countermeasure selection process works well with a limited number of countermeasures, and the performance is better in comparison with other proxy-based Network IDS. However, further work is needed to extend the deployment in a large-scale network. In such a setting, there is also the option of distributing the computational overhead of the control center, which naturally represents a *single point of failure*.

An interesting feature of the authors' framework is the capability to react in a *completely autonomous* way; once an alarm is generated from the *alert correlation engine* (due to the exceed of a specific threshold), the system selects a countermeasure from the pool. From an administrator's viewpoint, this reduces the human effort, especially in a virtual environment where many users share the same physical resources. Nevertheless, one can argue that in particular situations, and for critical resources, the reaction strategy cannot completely exclude the administrator approval, but it has to consider their expertise in the decision process.

A notable shortcoming of this work is the handling of zero-day vulnerabilities. The solution proposed by the authors is a profile database for the VMs, but one can argue that this is insufficient to solve the problem, and the proposed IDS agent by the authors (Snort [5]) is a rule-based one, thus incapable of detecting zero-days.

2.3.3.6 Wang et al. [145]

Description The work in [145] introduces a probabilistic approach for optimal security hardening. The authors aim to bridge the gap between system vulnerabilities and organization-level security metrics. To do so, they propose a methodology which incorporates AGs and Hidden Markov Model (HMM) to describe probabilistically the interconnections of the numerous security states of a system. Precisely, the authors extend Multiple Prerequisite (MP) graphs [96] by introducing 3 types of labels to the observable subjects of the network. The tags *Solid*, *Soft*, and *Dark* are used for marking the physical assets (e.g. servers), any measurable notion (e.g. network traffic), and the system vulnerabilities in the system. According to the authors, this approach reduces the size of the graphs, while important characteristics of the attacks can be identified which in turn leads to a better estimation regarding the security state of the system.

By taking advantage of the AGs portraying the interconnections in the network, the authors apply HMM to estimate probabilistically the possible security states of the system. HMM enables the quantitative analysis of the security hardening problem, while it can be used to model uncertainties introduced in the process. By employing HMM the authors can probabilistically predict the system's state transitions given a set of network observations. In this way, the defender can be notified about which state the target of the attacker is going to be, and thus take the appropriate decisions for blocking potentially harmful transitions.

For conducting a cost-benefit analysis, the authors extend the AG into a *directed state contact* one which is able to represent the state transitions [145]. Based on the interconnections of the graphs, the authors use a cost function to quantify the cost induced due to system transitions. The cost function is a weighted one which considers two individual costs, namely the *attack cost* caused by potential vulnerabilities, and the *defense cost* derived from hardening measures. Both these costs are modeled using the Butler's risk assessment framework [158]. This enables the synthesis of several problem-specific security metrics, which can be reviewed and put in practice by the security administrators. The problem is modeled as an optimization one aiming to minimize the cost function.

In this direction, the authors utilize a heuristic algorithm based on the ACO family [68]. Precisely, the *ants* roam probabilistically on the graph based on the probabilities indicated by the HMM and add pheromone on the edges of the graph. Ant path selection

decisions are taken by considering two parameters, namely the *attractiveness* and *trail level* of the move, where the latter incorporates also the cost of the trace.

In the conducted experiments, the authors manually construct a *directed acyclic graph* (DAG) and determine the HMM elements. They also assign values on the edges considering NVD-CVSS [143] framework and experts' knowledge. Throughout the experiments, the authors tested the ability of their solution to identify the root causes of three different attacks, and they demonstrated that their proposal is able to balance the trade-off between the defense and attack costs and deliver near-optimal solutions.

Analysis As already pointed out, the authors make use of AGs and HMM for representing the problem and rely on ACO to define optimal hardening solutions. Following the discussion of Section 2.3.2.2, the AG technique faces scalability problems as the size of the network increases and the possible interconnections between the different nodes lead to a system state explosion. On top of AGs, authors utilize HMM to create a probabilistic mapping between the AG and the various system states, and to model the uncertainties. As with the majority of works in the literature, the experiments are based on a rather small network topology with a limited number of system states and 4 types of countermeasures, namely *Disconnect, Patch, Disable* and *Configure*. More specifically, two testbeds have been created having 14 and 18 states, respectively. In this context, the proposed framework is able to provide a near optimal defense solution in a reasonable time for both testbeds, respectively 27.23 and 29.11 seconds. However, as the time complexity of the algorithm is $O(M \times N^2)$ (where M represents the network states and N the number of hosts) it can safely be asserted that the solution is not scalable for large-sized networks.

The quantitative analysis of the framework is based on the Butler's framework [158], very similar to [153]. This framework enables the introduction of several types of security metrics in order to define complex cost functions. Further, the authors refer to CVSS [143] for defining security metrics that will allow them to quantify both attack and defense consequences. As already mentioned, the use of standards, can add value to a work as it caters for a solution which is aligned with globally accepted and deployed ICT practices.

It is to be mentioned that the proposed model is not destined to react dynamically, but to infer on countermeasures that can achieve the near-optimal trade-off between the attack and defense costs. That is, authors' proposal aims to assist the decision maker, while demanding the active engagement of a security administrator for determining the assets under protection and tuning the parameters of the framework.

The optimization strategy followed in this work is quite interesting. The authors argue that it is impractical to exhaustively traverse the search space of the problem to define the optimal defense strategy. This is why they utilize ACO to pinpoint the root causes of specific attack scenarios. This approach narrows down the focus of the problem significantly and it can be used to identify critical assets in the infrastructure. In addition, ant colonies are a fast solution that can provide a satisfactory solution quickly [67]. ACO also guarantees that the algorithm always converges to a solution. The aforementioned ACO qualities could be proved beneficial in dynamically changing graphs, where new systems states may appear and a defense mechanism should define fresh optimal solutions in a reasonable time frame.

2.3.3.7 Zonouz and Haghani [176]

Description Zonouz and Haghani [176] introduce a framework called EliMet with the aim of estimating whether the security requirements in a system are effectively met. EliMet combines expert knowledge and reinforcement learning to support decision making against intrusive incidents. The system is driven by system-specific security measures to infer and identify risky system states with the purpose of suggesting the administrator appropriate healing actions. The system passively observes administrator's reactions against offensive incidents to calculate the aforementioned security measures. According to the authors, by using these measures, the system not only is able to respond in an autonomous manner, but also to predict potential security threats that administrators should take care in advance.

More specifically, the state of the system is modeled as CMDP [108], where the competitors are the adversary and the defender who both aim to increase their profit. For estimating the system's states, the proposed model considers the following parameters: i) the security state space, ii) a set of actions divided into *adversarial* and *response* actions, iii) a security measure function, and iv) a probability of shifting to a new system's

state. In addition to these parameters, EliMet models the insufficiency introduced by potentially false alerts of the IDS. Thus, the *belief* about every security state is measured probabilistically based on the previous state of the system combined with the alerts observed for the current state.

Given the CMDP model, the system identifies the optimal defensive solutions based on the security measures. EliMet treats the reaction procedure as a *maxmin* problem and finds the optimal defence policy by maximizing the security measures through Bellman's optimization method [177], known as dynamic programming.

As already pointed out, EliMet passively observes the security expert's reaction on intrusive incidents. In parallel with passive observation, the system uses an inverse reinforcement learning algorithm. The latter incorporates the expert's response policy and the CMDP model with the aim of iteratively refining the security measures so that they converge with the expert knowledge. The proposed algorithm takes also into consideration the defender's expertise level in calculating security measures. After a specified period of time, the system concludes to an optimal policy for the corresponding CMDP model, which could be used later in an automated response system.

The reinforcement procedure is able to result in efficient policies when it comes to common incidents that appear frequently in a system's defense life cycle. However, for rare system states, where the policy uncertainty is high in terms of Shannon entropy [178], the system queries the expert for the appropriate action. The incidents which will be queried to the expert are decided based on two criteria. On the one hand, the less information the system knows about the rare state the higher the possibility for the system to generate a query to the expert. On the other, a query for a system's state may arise or not, based on the potential return benefit for the defended system. In other words, the more information gain stems from a state transition the higher the chances for a query to be generated.

Another feature of EliMet is its ability to perform contingency analysis. The system imitates the adversary by choosing the actions that increase the offensive benefit, while at the same time takes the optimal defense actions. In this way, the system identifies risky states that the administrator should take care of and critical assets which should be protected more intensively.

Analysis EliMet is an interesting framework as it combines several techniques to deal with the problem of providing optimal countermeasures. More specifically, the authors utilize CMDP graphs to represent the system's security states. According to their evaluation, the graph creation occurs in a reasonable time frame, that is, 400 milliseconds for networks with 37 nodes, while the number of possible system states affects the response time of the system proportionally. CMDP is built based on the topology of the system and global access control policies, while the system considers IDS alerts as the sole input events. However, possible critical system states can appear upon system updates or the emergence of new vulnerabilities. It seems that such information is not taken under consideration by the authors during the graph creation. In general, the CMDP graph representation seems promising, but as the authors state, dynamic changes in the system topology are sure to pose a challenge.

Besides its ability to dynamically apply defense policies, another strong aspect of this proposal is its capacity to perform also static risk assessment for identifying critical assets proactively. However, both the risk assessment and the security metrics utilized to quantitatively evaluate the system's effectiveness do not follow any specific standard. The authors provide a general purpose framework, yet it is unclear how the security measures are defined and how complex and multi-objective measures could affect the overall performance of the framework.

The authors utilize reinforcement learning in an effort to capture and integrate the experts' wisdom in their framework. In their experiments, the refined security measures generated by the reinforcement learning algorithm are compared with the measures produced by the expert's actions. According to the produced security measures, the algorithm seems able to imitate the expert's actions, but in some cases, the actions taken by the algorithm overprotect the states of the system. Naturally, the ability of EliMet to converge with the expert's action is a positive feature. However in the context of a dynamic reaction system, if the system fusses over its assets may cause service disruptions and monetary losses.

Finally, the authors do not provide any information about the set of actions which constitute the defensive policies and the ways the size of this set can affect the performance of the system. Even though the CMDP seems to be scalable, a possibly hefty set of defensive actions could increase significantly the response time of the system.

2.3.3.8 Zonouz et al. [161]

Description The authors in [161] propose an automated system for cost-sensitive intrusion response. They model the intrusion response problem as a multi-step, sequential, hierarchical, non-zero-sum, two-player stochastic game between the adversary and the defender. The two entities present conflicting interests, and their utmost goal is to increase their benefit by taking sequential actions. The authors utilize ARTs to model system states which later are transformed automatically into partially observable competitive Markov decision processes (POCMDPs). In fact, they adopt this representation in order to apply Bellman's optimization method [177] that will allow them to define an optimal defense policy based on the state of the system. Bear in mind that this approach is used by the authors in EliMet framework [176].

Here, authors utilize a game-theoretic approach to model the relationship between the attacker and the defender. Under this prism, the one entity adapts its behavior according to the strategy of the other. The defender is the first who makes a move in the game and then the attacker responds in a sequential Stackelberg stochastic game [179]. This sequential behavior continues to an infinite-horizon and every movement leads probabilistically to a new system state. The state transitioning is modeled in POCMDP by also considering the uncertainty for the exact state of the system caused due to uncertainties derived from the IDS. In this way, POCMDP conceptualizes the system's states as *belief states* instead of exact ones.

The authors proposed a decentralized architecture to implement their mechanism. Their notion incorporates local engines placed in individual hosts in the network and a global engine located in the RRE server. The local engines are subscribed to an intrusion alert database in order to get notified when an intrusive incident relative to their host appears. Additionally to the IDS alerts, the local engines undertake also the creation of ARTs for the local host. Hence, the local engines are responsible for modeling the local system and react against offensive events. Every local engine has at most 3 ARTs where each one is used to infer on possible violations against the CIA of the host. The root node of each ART holds an accumulative security estimation value (δ) which is sent to the global engine of the RRE. This engine is responsible to inspect the security level of the whole infrastructure.

The global engine takes as input the network topology, the acknowledged system vulnerabilities, and the connectivity matrix and creates a CMDP automatically. The same engine aims to safeguard the system using global objectives defined by the administrator of the system. For this purpose, the authors utilize a Fuzzy control-based technique to enable the administrator define Fuzzy rules in the form of IF <premises> THEN <consequent>, which are then used to infer on the security level of the system. The δ values derived from the local engines are converted to qualitative values (high, medium, low) to meet the ones set in the rules by the administrator. Then, a center of gravity defuzzification method [180] takes place to provide a quantitative score for the security level.

Analysis The authors introduce several methodologies to deal with the problem of the dynamic intrusion response. The main contribution of this work is the distributed nature of the proposed framework. Based on the literature, we can safely extrapolate that finding the optimal defense solution against an attack is a computationally intensive task. In this direction, a distributed model to deal with this problem can improve the scalability and performance of a response system as the computational burden is relocated to its hosts. On the downside, this approach poses also certain limitations. The purpose of every host in a system is to deliver one or more services, and thus adding an additional intensive task could lead to service availability issues. This resolution can be used in networks where the participating hosts are able to manage this kind of procedures. However, in environments with limited processing power on the edge nodes the same approach may be impractical. Furthermore, trust issues emerge in distributed mechanisms destined to deliver security services. As a result, trust models [181] have to be used to ensure the legitimacy of the nodes, given that misleading results may occur due to contributions stemming from compromised nodes. Finally, decisions made in an automated and autonomous manner to the edges of a network may disrupt its normal behavior, while at the same time the administrator may have partial observability of the system state.

The authors use ARTs to represent the security state of the system, but for defining the optimal defense solution the methodology instructs the transformation of ARTs to a CMDP. This of course comes at the price of additional overhead. Moreover, authors do not rely on specific and global accepted metrics to support the optimization process.

Instead, they provide a formula which can feed a cost function to enable a quantitative analysis. As already pointed out, every local node has 3 ARTs, one per CIA property. Even though it is essential to quantify the impact on CIA properties, it is debatable whether or not a defense solution should consider more metrics to provide a complete quantitative defense analysis.

The local engines are capable of reacting against intrusive incidents automatically. Still, the system demands the complete engagement of the administrator for defining the global security objective of the system in terms of Fuzzy rules. According to authors' evaluation, the system performs well for quite large sized ARTs. However, they do not elaborate on the number of available countermeasures to cope with every possible state of the system. In case there is a pool of likely reactions against an incident, this is translated to an expanded search space and the problem's complexity could be further expanded by incorporating multi-objective cost functions in the optimization process.

2.3.3.9 Gonzalez-Granadillo et al [182]

Description This contribution proposes a geometrical model to select the optimal combination of countermeasures based on the *Return-On-Response-Investment* (RORI) index with the aim of counteracting cyber attacks against critical systems.

The authors present their improved version of RORI index, discussed in [183, 184] by extending the approach proposed initially by Kheir et al. [103]. Specifically, they modify the initial formula to also include the infrastructure value expressed as *Annual Infrastructure Value* (AIV), and to handle the possibility of applying a null set of countermeasures in specific scenarios.

Also, the authors extend the definition of attack surface of a given system, presented in [185]. To do so, they present a "volumes" model, which represents systems, attacks and countermeasures in a three dimensional coordinate system. The dimensions correspond to the users, communication channels and system resources. Specifically, the volumes are defined as follows:

- *System Volume* - It represents the maximal space a given system exposes to users and attackers.

- *Attack Volume* - The portion of the total volume being targeted by a given attack based on the vulnerabilities it is able to exploit.
- *Countermeasure Volume* - It represents the percentage of system volume that is covered and controlled by a given countermeasure.

The three dimensions are defined following the access control methodology [186, 187], and are identified as the ones which contribute directly to the execution of a given attack. That is, user account as the *subject*, resource as the *object*, and channel as the way to execute an *action*. The dimensions are then populated, and by following the CARVER methodology [188], a weighting factor is assigned to each element represented in the Cartesian system. In this way, a numerical bijection is created between the real elements and their representation within the coordinate system.

In this three dimensional system, the represented volumes appear as 3D geometrical figures (parallelepipeds) within the system volume. Also, the attack and countermeasures volumes are calculated and represented considering the used dimensions. For multiple attacks and countermeasures, a study on the volume union and intersection is conducted, showing the possible ways to calculate the contribution of both joint and disjoint volumes.

The main idea of the authors is that by using this graphical representation, it is possible to determine not only the impact of each attack and countermeasure (or the impact of a group of them), but also the residual risk (i.e., the volume of the system that is being attacked but is not covered by any countermeasure) as well as the potential collateral damage (i.e., the volume of the system that is not being attacked, but is covered by a countermeasure).

The authors also offer a prototype implementation, in a form of an application, which generates the above mentioned 3D graphical representation of the system, attacks, and countermeasures. The same application is able to automatically evaluate, rank and select the optimal set of countermeasures against complex attacks. The platform is composed by two modules, namely the Attack volume application, and the RORI application. The first one is responsible to map the attacks and the countermeasures into the Resource-Channel-User 3D system by calculating their monetary impacts. The second is in charge of performing the evaluation of individual and combined countermeasures, taking as

input all the needed parameters for RORI calculation. If some of them are missing (i.e., *Annual Loss Expectancy* (ALE) or *Risk Mitigation* (RM)), they will be requested to the Attack Volume engine, as they depend on the specific attack context.

To demonstrate their solution, the authors offer a case study pertaining to a critical infrastructure control system. Using this paradigm, they show the selection of combined countermeasures for a particular attack. To complete this task, the authors follow the approach presented in [184], which also considers any possible restrictions among the countermeasures (mutually and partially exclusive, restrictive).

Analysis This proposal builds on previous works by the same authors with the aim of using the RORI-based countermeasure selection together with an interesting geometrical representation of the involved defensive or offensive quantities. More specifically, in [183], they highlight two main limitations of the RORI-based model, in particular the accuracy in the estimation of the different parameters in the formula, and the unconsidered interdependence among the various countermeasures. Actually, the first limitation derives from the difficulty of getting a real assessment of variable parameters in the RORI formula. That is, they identify the ALE of an attack and the RM level of a specific countermeasure as the most challenging, thus requiring a considerable effort. A great improvement toward this goal is made in [189], where the authors propose statistical methodologies to estimate the above parameters using epistemic uncertainty [190]. The second limitation is only partially addressed in [184], where the authors study the results of their model applying a combination of two or more countermeasures.

It is to be noted that the authors decide to neglect the attack modeling. This is obvious because in the presented case study, the process starts with the evidence of an attack in the system. In this way, they are able to demonstrate the feasibility of their procedure in a dynamic environment. Nevertheless, even though the detection belongs to a different phase in the network life cycle, modeling the attacker's steps aids the reaction phase as well. This is because the defender is able to predict the attacker's trajectory in the system more easily. Overall, attack modeling is essential, because nowadays cyber attacks are increasingly disruptive, and the reaction time is a decisive factor. Moreover, the process in charge of generating the system, attack, and countermeasure volumes adds a further delay. It seems that more experimentation needs to be done to obtain a better view of the authors' proposal in terms of performance. Another idea would be to

consider extending the volume representation with extra dimensions, that is by adding time as an extra axis.

Recall that for calculating the overall contribution to the volume representation of each selected category within an axis, the authors follow the CARVER methodology, whose goal is to measure the priority of each element in a given system. This measurement is based on 6 factors, namely *Criticality*, *Accessibility*, *Recuperability*, *Vulnerability*, *Effect*, and *Recognizability*. The proposed methodology assigns a numerical value to each considered factor on a scale of 1-10 and places them in a decision matrix. The sum of the values indicate the severity of a given dimension. The CARVER matrix was developed from US special operation forces, and it can be used both from an offensive or defensive perspective. Nevertheless, this methodology is not widely adopted by ICT community, which rather prefer to use other open standards.

In the authors' model, countermeasures are proposed to be implemented for a short period, that is, from the moment of the detection of the attack incidents until the system returns to a safe state. According to the authors, this approach is preferable because it does not need them to compute long-term investments in the proposed procedure. Due to the complexity of the search space and the inaccuracy of the results, they discourage the usage of genetic and heuristic algorithms. However, this option is debatable, because nature-inspired techniques may be proved particularly effective, especially when the pool of the available countermeasures becomes large.

In the context of the 3D representation used by the authors, the coverage of a countermeasure is defined as the percentage of system volume it is able to cover. In this mindset, they are able to calculate the percentage of volume that a countermeasure can cover for a specific attack. However, in practice, it is not straightforward to establish a direct mapping between countermeasures and attacks, especially if one considers representing them in another reference system. In addition, they present this countermeasure volume coverage as a percentage. In this way, the evaluation of joint or disjoint volumes is not simple, and it requires a preventive analysis. The negative impact of a combined solution is not considered as well, while only the cost impact is computed in the model. Nevertheless, the latter requires more effort, especially for evaluating the impact on the availability of the service for a combined countermeasure, whose effects are not expected to be negligible.

2.3.3.10 Miehling et al. [191]

Description In [191] authors model the defense problem using the notion of *POMDP*. Their goal is to provide optimal dynamic defense policies to counteract in-progress cyber attacks occurring against the protected system. To do so, they utilize BAGs and Moving Target Defense (MTD) [192] schemes for implementing defense policies able to adapt to the adversary's behavior in a dynamic manner. The proposed model is guided by security metrics that quantify the trade-off among the *CIA* of the system in an effort to secure it without interrupting the provided services.

More specifically, the authors utilize BAGs to represent the adversary's possible movements in the system by also considering probabilities in every step (edge) on the paths that connect the entering (leaf) nodes toward the root nodes of the graph. The spreading ability of an attacker depends on the type of the node expressed by AND/OR relations, the previous state of the attacker, and the exploitation probability of the node. In addition to these aspects, the authors engage also a factor to emulate the defender's partial observability. Under realistic terms, malicious actions may occur in a system without raising intrusion alerts. This is why the authors introduce the *probability of detection* factor in their model as well. This uncertainty about the current state of the system forms a *belief* about the present network's state.

The countermeasure strategy is defined as a set of individual actions where, based on an incident, an optimal subset of those can be used to counteract an intrusion. Every defending action directly affects CIA metrics. More specifically, a cost function should be able to reflect both the negative and the positive impact to the CIA metrics for the applied actions. To this end, the authors treat the defense problem as an optimization one where the ultimate goal is to define an optimal defense policy. They take into consideration both the *belief* for the current system state and the future states to optimize the cost function through dynamic programming (Bellman's method [177]).

The evaluation of the proposal has been conducted on a small BAG with 12 attributes and only one of them as critical (root). The graph has two leafs and two possible countermeasure actions. According to the authors, the possible states of the system is 2^{12}. In order to reduce the search space of the problem, the authors add restrictions by considering only *AND* relationships among the attributes and monotonic threat propagation (i.e., the attacker aims to constantly increase their benefit) and result to a minimized

problem with 29 states. The POMDP model of the example network was solved in terms of identifying the optimal solution using a POMDP-solver written in C programming language. The outcome is a network *heat map* which represents the critical nodes of the system. The countermeasure strategy is shown intuitively and can be inferred from the heat map. The authors suggest the adaptation of those countermeasures that reduce the exploitation probability of the risky nodes.

Analysis As stated previously, the authors make use of BAGs to model the system's security states. Albeit this type of representation is suitable for representing also the uncertainties which are introduced, the limitation of scalability affects significantly the efficacy of this solution. Specifically, as shown by the authors' experiments, the conceivable states of the system combined with the numerous possible countermeasures expand significantly the search space of the problem. The experiments conducted in a small-scale scenario and under specific assumptions, aiming to minimize the search space. On top of that, the example scenario engages only two countermeasures and one root node in the graph. To this end, one can argue that the complexity of the solution is high and can be significantly increased if the solution is utilized in a dynamic environment trying to model the behavior of a moving attacker. However, the employment of the detection probability as well as the probabilistic contagion spreading model they use emulate the imperfect environment in which a security expert has to take actions.

The cost function used for quantifying the cost of an attack or the deployment of a defense action is calculated based on CIA metrics. CIA properties should be in the core of the countermeasure strategy, but we argue that a framework should include several types of metrics to achieve a better quantification of the problem. Apart from that, it seems that the authors do not adopt any globally accepted security automation standards in their model.

Finally, although the authors claim that their proposal suggests an automated defense tool, they do not evaluate the performance of their proposal in this direction, while the countermeasures taken in the conducted experiments aim to block or disable a given service in the system. These actions, however, can be considered as extreme measures to deal with an attack. The goal of a countermeasure strategy is to explore optimal policies to deal with a security incident instead of applying extreme measures.

2.3.3.11 Shameli-Sendi and Dagenais [104]

Description The work in [104] presents a model able to dynamically evaluate the positive and negative effects of defense actions on a system under attack. The problem of providing defense actions which maximize the security performance but simultaneously minimize the negative effects of the applied measures is treated as a MOOP. The ultimate goal of the framework called ORCEF is to provide optimal defense actions while simultaneously sustaining the quality of the services provided to the end-users.

Notably, ORCEF utilize AGs and SDGs in an effort to respectively allocate the defense points in the network and to evaluate the response negative impact. AGs are used to enable the system to find the attacker's position and final goals based on a confidence level, and to define the optimal network points where appropriate defense measures should be applied. In this way, ORCEF creates a dynamic mapping between the possible attack paths and the network topology. Further, SDGs are used for the sake of identifying the interconnections among the provided services. In this way, ORCEF is able to conduct a quantitative analysis by considering also the service dependencies and the number of users affected by the malfunctioning services.

ORCEF's response engine is triggered by IDS alerts and tries to locate the attacker on the AG. Also, it is aware of a number of parameters, including the network topology, the number of provided services in the infrastructure, and the population of users. Based on a pre-defined pool of responses and the aforementioned parameters, ORCEF calculates the positive and negative aftermath of every response by utilizing the MCDM methods SAW [193] and WP [194]. To do so, the positive effects take into account the outcomes on CIA and the performance of the system. The negative corollaries consider the effect induced on the availability of a service and other dependent on that services, the users' inaccessibility to services, and the setup cost of a defense. The positive outcomes and the cost setup of a defense are calculated statically, while the rest of the metrics are dynamically adjusted during an in-progress attack to the system. The determination of the static metrics and their importance is done during system bootstrapping, where the administrators are asked for their opinion. More specifically, ORCEF captures the security experts' opinions in the form of linguistic variables by utilizing a Fuzzy model.

Once the positive and negative effects for every response are calculated, a Pareto optimal set of defenses is generated. The optimal defenses are those which achieve the most

efficient trade-off between the security level and the negative impacts. Finally, the Pareto optimal responses are ordered depending on the state of the system. If the attacker is highly skilled, then the solutions that minimize the damage cost should be selected. Also, if the resource is of high value, then those countermeasures which minimize the negative impacts should be selected.

The authors evaluated ORCEF in a topology consisting of 5 sub-networks under two attack scenarios modeling both an external and internal attacker. The system was able to react in about 489 milliseconds and 456 milliseconds for the two scenarios, respectively.

Analysis ORCEF is a framework able to respond dynamically on ongoing attacks as it is capable to allocate the possible attack paths based on IDS alerts and the defense points of the network. Toward this goal, the proposed framework incorporates both AGs and SDGs. Although the authors' approach combines the advantages of the aforementioned representations to deliver a cost-benefit response framework, it seems that the authors do not consider the cost of initializing or adapting the graphs. Under this prism, we consider the protected topology to be static. That is, under realistic conditions a network topology can dynamically change, say, as new nodes join or leave the network. These kind of changes should be reflected also in the AG dynamically. Further, the new nodes may bare vulnerabilities, existing nodes may get vulnerable at any time, new users and services may be added, and this is why the SDG needs to be updated or recreated from scratch.

By utilizing the MCDM framework and Pareto optimal methodology, the proposed system is able to come up with optimal defense solutions on different defense points in the attack path of the topology within a short time frame. Still, the authors do not mention if their evaluation is limited in protecting a single service in the topology. In this case, the framework should be assessed under a situation where multiple services need to be protected and several countermeasures must be applied to diverse defensive points. If so, the complexity of the optimization problem is expected to further augment. Based on the results, ORCEF seems to perform fast for the given scenario. On the downside, the evaluation metrics for quantifying the impact of defense decisions on the target system do not follow any specific standard. An exception is that ORCEF incorporates the CVE standard when it comes to the alerts generated by Snort [5]. However, the authors' approach to utilize a fuzzy model for capturing the expert's opinion in the form

of linguistic variables can improve the system's experience. Furthermore, the ORCEF administrators need to pass through a demanding phase of initializing the system as they have to assign a great amount of linguistic values to the system's parameters. The defenses used by the authors are applied on several guarding points according to the attack path in the topology. This is an interesting approach which minimizes the effort and the cost as the countermeasures are applied only in the part of interest on the graph. Finally, as the authors state, ORCEF is not able to generate combinations of defenses for combating an offensive incident in a more efficient way. This is because ORCEF concludes only to one single optimal solution.

2.3.3.12 Kotenko et al. [195, 196, 197, 198, 199, 200, 201, 202, 203, 204, 205, 206]

Kotenko et al. [195, 196, 197, 198, 199, 200, 201, 202, 203, 204, 205, 206] presented a series of works dealing with countermeasure strategies against cyber attacks. We can categorize them into 1) those which cope with the countermeasure selection in SIEM systems based on AGs and SDGs [195, 196, 197, 198, 199], and 2) those which deal with attack modeling and security evaluation in SIEM systems [200, 201, 202, 203, 204, 205, 206]. Below, we elaborate on each category.

A) Countermeasure selection in SIEM system

Description The authors define an *ontological representation* for security metrics[195], viewed as a core element of a countermeasure decision support module within a SIEM system. In their vision, the adequacy of the eventual application of a countermeasure action depends mainly on the speed and the reliability of selection and calculation of a subset of security metrics. However, in modern networks, this concept represents a hard task for the security administrator, given the huge amount of data. To solve this problem, they propose an *ontology*, which is seen as a flexible tool for describing objects, classes, relationships and attributes of a domain of arbitrary complexity. In this way, the ontology is used to select the most fitting countermeasures based on the calculated values of metrics and rules of logical reasoning. The main classes existing in the ontology are *reactions*, which in turn are divided into two subclasses, namely *alarms*

and countermeasures, and *metrics*. The latter class is an abstract superclass from which several abstract subclasses are generated by applying the relationship *is-a*:

- *Malefactor metrics* - They incorporate important parameters regarding the attacker profile, e.g., *Attacker Skill Level* and *Successful Exploitation Probability*.
- *Topological metrics* - They integrate information about the network topology, e.g., hosts, applications, and vulnerabilities.
- *Attack metrics* - They describe the main characteristics of an offensive attempt, e.g., *Attack Impact* and *Confidence Level*.
- *System metrics* - They consider *Attack Surface* and overall *Security Level*.
- *Zero-day metrics* - They attempt to measure the impact of an eventual zero-day exploitation in the system, e.g., *Probabilistic Vulnerability Measure* [207] and *k-zero day safety* [79].
- *Cost metrics* - They take into account *Annual Lost Expectancy* and the *RORI* index [183].

Based on the presented classification, the authors introduce their approach to the countermeasure selection [197, 198], considering the following main requirements: (1) *security metrics*, as suggested in the corresponding ontology; (2) *AGs*, created on the base of existing vulnerabilities, network configuration and attacker capabilities by following an attacker-centric model; (3) *SDGs*, considering information of the interconnections between network services; (4) application of the SCAP for the specification of input data; (5) integration within *SIEM systems*, considered to be in active development in the recent years. In particular, the third requirement is defined by the necessity to consider possible negative impacts of the countermeasure selection in the objective functions of the system under protection. Instead, the fourth requirement is connected with the strong need to automate the security analysis process and reaction in the modern systems, especially when considering the possibility of reacting dynamically.

The authors also discuss the countermeasure selection technique, distinguishing between two main modes, namely *static* and *dynamic*. The first one is understood as a general improvement of the security level of the system, taking into account the values of metrics

previously defined as input data. The latter one is seen as preventive actions for a specific ongoing attack, taking into account the SIEM events as its main input data. Specifically, in [199], the authors concentrate on the events level, because it allows considering the dynamic aspect of the security assessment and countermeasure selection. This level is based on the incoming security events, stemming from different sensors in the network and from the SIEM system. The *event model* connects this information with models of the previous level (AG, attacker models), mapping the attacker position on the AG and calculating security metrics that reflect with high accuracy the security state of the system. Based on these metrics, a list of possible countermeasures is generated. In a next step, the countermeasures are ranked taking into account the cost, effectiveness, and collateral damage.

Analysis The described works use both AGs, which allow them to define the possible steps of an attacker in the system, and SDGs, which consider also the interconnections between network services. By doing so, the authors are able to obtain the advantages of both these two attack modeling techniques, thus achieving a more accurate security evaluation. This includes the mapping of the attacker's position and their most probable paths, together with a cost-sensitive analysis, which represents the most important results. However, the algorithm proposed by the authors to evaluate these graphs based on the aforementioned metrics is not presented. The authors only describe the methodology used to achieve their goals. Yet, without an optimized methodology to analyze the abovementioned graphs, the caused overhead makes the evaluation unpractical in case of ongoing attacks which demand a dynamic reaction [95].

To specify a common approach in the development of a countermeasure model, standards from SCAP are applied. Particularly, authors use the *CRE* and *ERI* standards. The employment of these standards along with CPE for network configuration, and CVE, CWE, and CVSS for network vulnerabilities, makes the authors' model interoperable and quantitatively comparable to others. Nevertheless, the authors assume that the system has already a pool of countermeasures, which can be selected by applying an *ad-hoc* algorithm. The generation of such a pool is made based on the knowledge and expertise of the system administrators. On the downside, this assumption requires an initial significant effort from the expert, who has to fill the *knowledge database* of countermeasures, followed by a stable phase where the system is capable of reacting

automatically. Therefore, one could argue that the authors' proposal lacks of flexibility, because this database of countermeasures needs to be continuously updated, and the participation of the administrators, at least in some specific and critical situations, is unavoidable.

From an attack response viewpoint, the proposed solution supports both static and dynamic modes. In this respect, it is theoretically able to cover a wide range of assessments regarding the network security level. However, only few examples are presented by the authors, including some type of attacks on small-scale networks with the use of generated attack sequences and poor attacker modeling. Nevertheless, a more detailed experimental section is needed to elaborate on both the input data regarding the attacks and the network topology. Moreover, the authors claim that for a small number of security events and short attack sequences, the effectiveness of the countermeasure selection is reduced, but the accuracy and efficiency of the implemented solution satisfy the initial requirements. Still, the presentation of the results is not sufficient, which constitutes the authors' work not directly comparable against others in the same field.

B) Network Attack Modeling and Security Evaluation Framework

Description This series of works by the same authors is different from the previous one, not only in their scope, but also because it is backed by a great implementation process. In [200], the authors present an *Attack Modeling and Security Evaluation Component* (*AMSEC*), which if deployed in tandem with a SIEM system is capable of (1) generating *ATs* and *SDGs* based on the topological vulnerability analysis along with zero-day vulnerabilities, (2) applying *anytime algorithms* to provide a near real-time attack modeling, (3) analyzing AGs to predict *future attacker's steps*, (4) calculating *security metrics* that also reflect the response impact, and (5) selecting the optimal security solution through an *interactive decision support* process.

The authors' proof-of-concept implementation supports two main modes, namely *design* and *exploitation*. In the first mode, AMSEC operates offline taking as input a model of the analyzed system. It then produces a list of weak spots and a set of ATs (one per attack scenario), which will eventually build the AG. In the latter mode, AMSEC

operates in real-time or near real-time, so it can adjust existing ATs, succeeding in predicting attacker's steps and generating the appropriate countermeasures.

The proposed system is connected to external open databases of vulnerabilities, attacks, and configurations (NVD [121], CVE [88], CAPEC [131]), and translates the gathered information into a format recognizable by the AMSEC's security data repository. The latter also stores data obtained from network scanning tools and admin's knowledge. All the gathered information is used to build and analyze the AGs. The authors present a prototype which contains three basic components:

- *VDBUpdater* - It updates the internal database of vulnerabilities using information obtained from NVD.
- *Network Constructor* - It aims to create and modify network models.
- *Security Level Evaluator* - It evaluates the overall security level of the system starting from the analysis of the AGs and the associated security metrics.

An improvement of the previous model is presented in [203], where the authors present *CAMIAC* (*Cyber Attack Modeling and Impact Assessment Framework*), a framework which optimizes the AG building and analysis processes with the goal to enable their usage in near real-time operations.

Kotenko and Chechulin [205] extend their AMSEC prototype, by proposing a novel approach to construct, modify, and analyze the AG in a faster way, showing that it can achieve better results if the analyzed network presents a limited number of changes. In this way, they claim that it is possible to monitor a large-scale network by updating only the topological changes.

Analysis In the presented works, authors use both AGs and SDGs, as with their previous series of works described in Section 2.3.3.12. However, in this case, the difficulty of generating and maintaining these graphs in real-time is demystified. Also, it is made clear that these representations keep their actuality for a limited period of time, until significant changes in the security policies or in the network topology occur. The suggested solution for this problem is to use these models constructed *in advance*, and *updating* them with the help of ad-hoc algorithms. By doing so, the computational power

required for the operations related with the construction and maintenance of the graphs decreases significantly, thus offering the possibility to represent large-scale networks and analyze the ongoing events in real-time. As already pointed out, the performance of the proposed framework in terms of applicability is critical. So, the scalability of the model should be further demonstrated with data obtained from real networks instead of simulated ones.

Regarding the type of reaction, the main idea of the authors is equivalent to that discussed in Section 2.3.3.12. Nevertheless, their proof-of-concept is only partially implemented, thus limiting its capabilities to the detection of the attacker inside the monitored system. Therefore, the prototype shows limited risk analysis capabilities, and only recommendations are given to counteract security incidents. However, very few would argue that the framework capabilities of automatically (or semi-automatically) reacting to malicious activities is a key requirement for any countermeasure strategy.

The authors' framework works in combination with a *Security Data Repository*, used to store information updated from external sources and the results of the security evaluation of the system. This database makes an extensive use of standards (CPE, CVE, CVSS and CAPEC), thus providing a standard way to represent and report cybersecurity-related information. Note however that in the context of these works this information is mainly used for the attack modeling phase, without giving the proper importance to the reaction one.

Lastly, an interesting feature is the presence of an *Interactive Decision Support Module*. This component interacts with the admins through a graphical user interface (GUI) to let them select the most appropriate security solutions, and define their preferences regarding the different types of requirements (risks, costs, benefits). The GUI is also able to visualize the attacker's position on the network map and uses different colors to show different risk levels. Nevertheless, only recommendations are presented, without any real implementation of the suggested counteractions directly on the various network assets.

2.3.3.13 Shameli-Sendi et al. [208]

Description Shameli-Sendi et al [208] propose a novel IRS architecture to select and deploy the optimal countermeasure in the context of dynamic reaction to cyber threats. The problem of providing countermeasures which maximize the security performance and simultaneously minimize the negative effects of the applied measures (i.e. impact on the system services and cost) is treated as a MOOP.

Particularly, the proposed architecture leverages the capabilities of SDGs and ADTs in an effort to respectively allocate possible defense nodes and to evaluate the attack damage cost. SDG are used to identify the interconnections among the services in the network, so that both the negative impact of attacks and countermeasures can be quantitatively evaluated based on CIA attributes. Additionally, ADTs are employed to extract the paths relative to an incoming attack, thus specific countermeasures can be allocated on defense points to block the intrusion.

When an alert is raised by an IDS, the proposed IRS maps the alert to the ADT. In this way, attack paths and defense points are identified. Starting from a predefined pool of countermeasures, the IRS computes which of those can be implemented based on 3 parameters that are evaluated independently, namely *security benefit, security impact, and security cost*. Once the Pareto set is generated, SAW method [193] is used to extract the optimal solution. Moreover, the IRS evaluates the possibility to combine multiple countermeasures. To do so, the authors propose the *vulnerabilities surface coverage*, which represents the vulnerabilities a countermeasure is able to heal. In this way, joint and disjoint surfaces are computed and, consequently, the countermeasure combination which covers the maximum number of vulnerabilities is selected. Further, the selected countermeasure is deployed and its effectiveness is evaluated through a 5 secs window. Specifically, the reaction is effective if no other attacks are detected in this time frame. The effectiveness values per countermeasures are afterwards stored in the database.

Lastly, the authors present a detailed experimental section, in which they deploy a real cloud environment with 6 VMs and 15 vulnerabilities in total. An ad-hoc multi-step attack scenario is created to exploit the above-mentioned vulnerabilities. During 6 months, the authors generate 5691 attacks, recording the parameters relative to the countermeasures deployment. In addition, the SAW methodology is compared against other 2 well-known scoring methods, namely PW and TOPSIS [194]. The authors

conclude that SAW is more reliable than the other, performing better for their scenario. Furthermore, a performance analysis is conducted, in which the authors show that the proposed framework is able to respond in 449 milliseconds for their attack scenario.

Analysis As stated previously, the authors use ADTs and SDGs in an effort to model possible attack steps within the monitored system and estimate their impact on the provided services. Although this dual endeavor may result in a more accurate attack representation, it has to be noticed that the proposed framework performs with one ADT used to protect a single asset. Notably, in real conditions, the protection of a complex system will require a forest of ADTs. This increases the complexity of the problem and the time required to analyze these trees. That is, more research is needed in this direction to safely argue on the feasibility of the presented approach. Additionally, the difficulty of generating and maintaining SDGs remains an open challenge, since the authors state that the importance and dependence between the services is pre-defined by a security expert. However, identifying and representing the interdependencies of all the services in a complex infrastructure is a cumbersome task and therefore can lead to inefficiencies.

An interesting feature of this work is the dynamism of the defense strategy. That is, the ADT is created first based on the SDG, and then updated over time, when services or vulnerabilities are added or removed. This aspect is surely appreciable, particularly in the modern systems which are characterized by a strong fluidity in their topology and configuration. Moreover, the computation of the reaction considers also the already deployed countermeasures. Still, this feature is not reflected in the experimental section, in which the authors replicate an ad-hoc attack scenario on a static network. Thus, more experiments are needed to test this promising capability and to demonstrate its feasibility.

Notably, the authors define the security performance of a given countermeasure as the number of vulnerabilities it covers multiplied by the history of its success/failure. Since multiple countermeasures can be deployed simultaneously to counteract an ongoing attack, a quite extensive study on the joint surfaces is presented. Although this methodology allows to quantitatively estimate the countermeasures' performance, it is not straightforward to define a direct correlation between vulnerabilities and countermeasures. Actually, this association method has been initially proposed in [166] and later demonstrated in [167], and overall it is proved to be a complicated task.

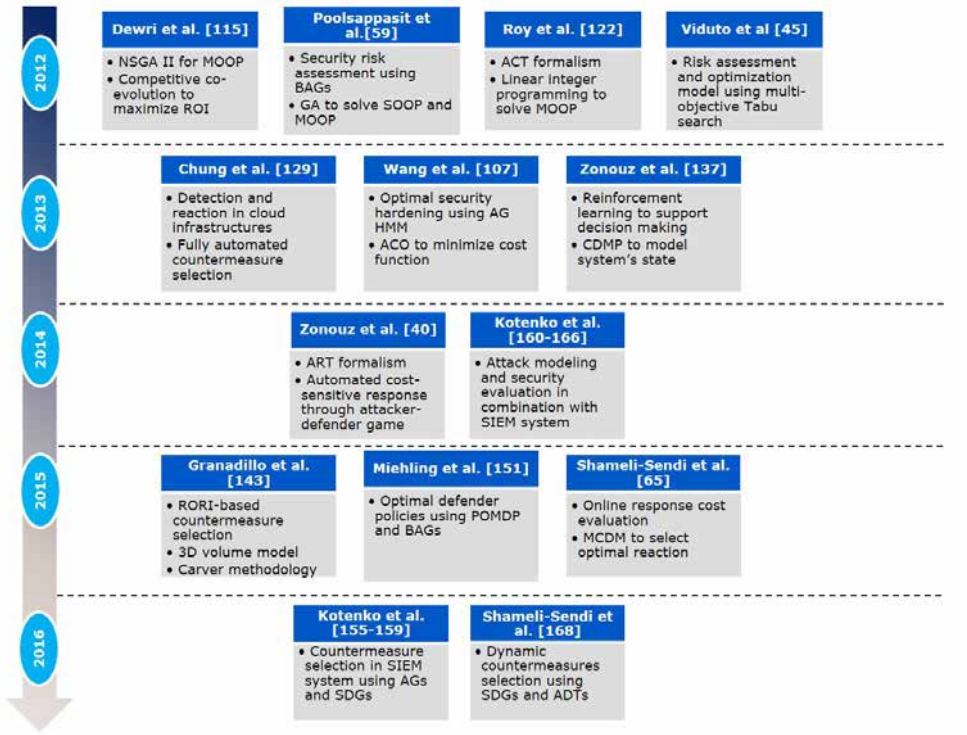


FIGURE 2.11: A timeline of the surveyed works, highlighting on their novelty and core characteristics.

The presented testbed includes only 6 VMs connected by 4 virtual switches. One could argue that such a scenario does not reflect the modern IT infrastructures, where hundreds of machines are connected for providing services to the end-users. Additionally, the complexity of the designed framework is reported as $O(|CM|^2 + (|S| + |W| + |V| + |CM_a|) \times |CM|)$, where $|CM|$ represents the number of possible countermeasures, $|S|$ is the number of services in the SDG, $|W|$ is the number of time windows used in the countermeasure goodness evaluation, $|V|$ represents the number of vulnerabilities within the system, and $|CM_a|$ is the number of current deployed countermeasures. It is clear that the presented framework lacks in scalability, thus it unsuitable in the context of dynamic reaction for complex networks.

2.3.3.14 Summary and comparison

This Section offers a comprehensive comparison of the various works analyzed as part of this survey based on the seven features introduced in Section 2.3.2.2. For easy reference, a summary of features per examined work is included in Table 2.6. In addition, an overview of the surveyed works with reference to the publication year and their chief

characteristics is depicted in Figure 2.11. This figure summarizes the evolution of the surveyed research area, in which several authors contribute diverse reaction frameworks against cyber attacks. Finally, we elaborate on both the positive and negative aspects of each work.

Regarding the first feature, namely attack modeling decision, Table 2.6 reveals that all the works analyzed, except of those presented in [182] [163], make use of a graphical representation to reflect the dependencies and the interconnections among the possible assets and properties in the monitored system. The authors in [163] suggest the use of AGs for modeling the attack space, but eventually they do not integrate such a model in their solution. Likewise, the work in [182], instead of using an attack representation model, it focuses on applying countermeasures solely on one specific node representing an asset in the monitored system. From our analysis, it can be safely deduced that a representation which is able to formally model the system's numerous dependencies is a necessity both for depicting the system, but also for creating the search space of the problem. In parallel with a formal representation of the system, several probabilistic schemes are utilized in an effort to model the system's security states transitioning. In this direction, the authors adopt BAGs [98] or HMMs [145] to model the uncertainties which are introduced in the process.

Considering the numerous sources of uncertainty (e.g., IDS false alarms, possible misconfigurations, target choices of the attackers, etc.) and possible unpredictable states of the system (e.g., zero day attacks) the adoption of such a probabilistic approach contributes toward more realistic solutions that can effectively support the decision maker. However, all the reviewed works, besides those presented in [195, 196, 197, 198, 199],[208], treat the attack model representation in a static manner. Under realistic terms, a network's parameters (e.g., topology, vulnerabilities, configurations) are in a constant flux and this inevitably brings changes to the attack modeling representations as new states, nodes, and transitions occur. This ilk of changes happening in a system should be dynamically reflected in the modeling process, but unfortunately this feature is almost sure to increase the complexity of the derived models. Interestingly, none of the reviewed works considers this feature when it comes to the evaluation of the overall complexity of the framework.

Another important conclusion of our survey is that most of the analyzed works deal with

the problem of cost-benefit attack counteraction by utilizing optimization techniques. This kind of solutions aim to define an optimal trade-off mainly between metrics that reflect the potential attack cost and those that quantify the impact and effectiveness of applying defense strategies. In this direction, GAs [153] and ACO [145] were utilized by leveraging single and multi-objective optimization cost functions for providing optimal sets of countermeasures. The works presented in [161, 176, 191] make use of Bellman's optimization on top of MDPs to identify the optimal defense strategies, while integer optimization and Tabu Seach [115] were respectively used in [160, 163]. In addition, multicriteria decision-making techniques like SAW, PW and TOPSIS has been recruited in [208] in the context of MOOP. Yet, the works presented in [195, 196, 197, 198, 199, 200, 201, 202, 203, 204, 205, 206], [169], [182] do not fall into this category of solutions, as the authors propose their own heuristic optimization method to guide their system to the optimal solution. The works presented in [153] and [161] are of special interest as their models reflect the dynamic relationship of the attacker and the defender. More specifically, [153] uses two competitive populations in the context of GAs that imitate an "arms race" between the two entities with the aim to define equilibrium points, that is, a set of countermeasures that can stop the attacker from increasing their gain. In the same direction, the work presented in [161] models the relationship of the attacker and the defender as a game, where the two players make sequential moves to increase their benefit. A different approach has been adopted in [104] where MCDM methodologies were used to infer on a set of possible countermeasures for eliminating the detected incidents. From this set, a Pareto optimal set is derived on a later step.

Another interesting fact stemming from Table 2.6 is that most of the proposed frameworks in the literature are destined to dynamically adapt to the events transpiring in the protected system. As stated in Section 2.3.2, a dynamic approach fits better the needs of a countermeasure mechanism since the system is able to adapt in real time to ongoing offensive incidents. However, such a dynamic approach increases the complexity of a countermeasure mechanism, and if we additionally consider the complexity introduced by the attack modeling representation, then we can conclude that the performance of the proposed solutions is questionable. On the other hand, static solutions are designed to deliver a quantitative risk assessment for the protected system at a given time instance and to aid security administrators in diagnosing weak points on assets appearing during the life cycle of the system. Even though static solutions can be proved a valuable

asset for identifying weaknesses in a network topology, the lack of adaptability against unfolding offensive incidents is not suitable to support a reactive countermeasure system.

In our analysis, we have distinguished the type of reaction (dynamic/static) from the automation level which actually reflects the administrators' engagement level in the reaction process. As it can be observed from Table 2.6, only the works in [145] and [153] require the manual engagement of the administrator, while the rest of them provide a higher level of automation. The majority of the latter belong to the semi-automated category, where the administrators supervise the countermeasure system and the enforcement of any defensive policy requires their approval before applied on the protected system. Still, the works in [191, 169], [208] proposed a fully-automated system, in which the defense mechanisms react in an autonomous manner requiring minor intervention by the administrators. More specifically, the administrators assign predefined directives that have to be followed by the countermeasure system. The proposed system in [208] can be characterised as fully-automated, but the administrator has to identify the services' dependencies and maintain the SDG of the system. Overall, although fully-automated solutions can respond immediately against ongoing attacks, the applied decisions may differ from a security expert's perspective, resulting to unwanted after-effects like the over or under-protection of network assets.

All the analyzed works adopt assessment methodologies to provide a quantitative analysis of the defense strategy they propose, but as we can observe from Table 2.6, this is being done mainly without adopting any security standards. A model which is able to quantify the trade-off between the attack versus defense cost in applying countermeasure policies is a vital feature for the countermeasure strategy. In this direction, as discussed in Section 2.3.2.2, several methodologies that engage and combine diverse types of metrics have been proposed. However, it is notable that even though those methodologies can provide a quantitative security assessment, half of the surveyed works do not adopt any globally accepted standard. We argue that a countermeasure strategy should adopt the use of standards, so that security administrators can perceive in an accurate and foolproof way the security state of the system at any time. In addition, the adoption of standards ensures interoperability and enables a countermeasure mechanism to operate across diverse systems. As it can be observed in Table 2.6, CVSS [143] is the most prominent standard used for describing the threats in a system. Even though, CRE [132] and CAPEC [131] are scarce among the used standards, we argue that they can

not only significantly contribute in the automation of countermeasure solutions, but also improve the accuracy in predicting the security state of a system.

It is important to analyze the way the various works evaluate the effectiveness of their proposal and specifically the ways a security administrator (who relies on a countermeasure system) can assess the outcome of the provided defensive actions. To do so, we included in Table 2.6 the scale and the type of the environments used in the evaluation of the examined frameworks. We also report on the role of the administrator to reflect the way they interact with the system. As it can be observed from the table, 10 out of a total of 14 works ([153], [98], [160], [169], [145], [161], [182], [191], [195, 196, 197, 198, 199], [208]) rely on a rather small-scale environment to evaluate their solution, while 3 others ([163], [176], [104]) employed a medium-scale one. The sole work that considers a large-scale testbed is that in [200, 201, 202, 203, 204, 205, 206]. It is also notable that only the works in [169] and [208] utilized a real environment instead of a simulation. The role of the administrator in most of the surveyed works is to tune the system, while only in [176] this entity is in charge to respond by giving feedback. Also, the exact role of the administrator was not appreciable in contributions [182, 169, 153], and [191].

Continuing on the performance criterion, the works presented in [176] and [153] are of special interest because of the methods used to ensure the optimality of the result. Precisely, the authors in [176] adopt a reinforcement learning approach, where the administrator is able to give a feedback to the system and boost the learning process to more accurate results. On the other hand, the framework proposed in [153] ensures the effectiveness of the defense policy as it tests the latter against several attack strategies based on the "arms race" method. The goal is to define a policy that will impede the attacker from penetrating further into the protected system. In the works presented in [161, 145],[200, 201, 202, 203, 204, 205, 206],[195, 196, 197, 198, 199], [98], [104], [160], [163] the administrator tunes the process of providing optimal countermeasures by setting objectives to the system. In [208] on the other hand, the administrator needs to update the dependences among the system's services. These objectives are represented in an abstract way, either by specific trade-offs in system's metrics or IF-THEN rules, which are used to infer on the security level of the system.

Finally, it is of significant importance to elaborate on the performance of the proposed frameworks in terms of scalability, time complexity, and response time.

Scalability As observed from Table 2.6, the majority of the reported works are characterized by a low scalability due to the inability of the attack modeling representations to scale. This feature seems to be a substantial limitation in the field. As already mentioned, this shortcoming becomes a major one if we consider that none of the reported works but those in [195, 196, 197, 198, 199], [208], consider dynamic changes in the network parameters. Moreover, even though the authors in [195, 196, 197, 198, 199],[208] consider a dynamically changing environment, this is not advocated by their experiments, as no change occurs in the employed topologies. However, according to their authors, the works presented in [169, 161, 176] have a high scalability. In fact, this is why the solutions in [161, 176] incorporate distributed architectures which disseminate the computational task to the edge nodes instead of dealing with the problem solely in a central point. Also, it can be said that the work in [176] presents a high scalability as it incorporates CMDP to the attack modeling phase. Although this approach looks potent, the level of information about the security state of the system, which can be reflected by the CMDP representation, is debatable. In addition, as with the vast majority of the surveyed works, this solution is tested in a small-scale simulated environment and therefore its overall complexity in a real environment might be considerably higher.

Time Complexity Regarding the complexity sub-criterion, half of the reported solutions present a high complexity, while the rest can be classified into low or medium regarding the same metric. Unfortunately, the fact that the vast majority of the proposals were tested under simulated and small-scale environments raises questions about their performance in a real networking environment. In addition, another parameter that can augment the overall complexity is the number of countermeasure actions that can be applied in each particular case. In fact, the reviewed works engage only a small fraction of the possible countermeasures. The existence of several possible countermeasures for every possible security state of the system can lead to a search space explosion and challenge the optimization algorithms. On the contrary, the works proposed in [169, 161] adopt a different approach, as they model the system in a distributed manner, so that the computational cost is not undertaken solely by a specific machine. However, these solutions introduce other limitations inherited from the distributed environments, including trust issues and the lack of accurate insight of the system's state at a central node in the architecture.

Response time It is obvious from Table 2.6 that it was practically infeasible to extract the response time for all the surveyed works. This is because 6 of them ([153], [98], [169], [182], [191], [195, 196, 197, 198, 199]) do not provide details about the response time of the framework they propose when activating the appropriate countermeasures. This is actually a controversial discovery as the effectiveness of a response system is also determined by its ability to react within a reasonable time frame. Especially for dynamic proposals, which have to respond instantly against ongoing incidents, this metric is of high significance. To sum up, only the works in [104, 176],[208] were classified as fast-performing solutions, while 3 presented average response time ([160], [163], [161]) and 2 seem to be slower regarding this metric ([145], [200, 201, 202, 203, 204, 205, 206]). At this point, we have to note that during the evaluation none of the works considered the time needed to create the attack modeling representation, while the small-size simulated environments cannot guarantee the preciseness in response time estimation.

All in all, it can be safely summarized that virtually all the so far proposed solutions in the field of countermeasures elicitation suffer from scalability issues due to the bulky attack modeling representations, the numerous possible countermeasures, and the dynamic nature of the monitored systems.

TABLE 2.5: Standardization attempts for security automation

Category	Acronym	Name	Description
Vulnerability management	CVE [88]	*Common Vulnerabilities and Exposures*	Provides a reference method for publicly known vulnerabilities and exposures. It is available in several format, such as CVRF, XML and HTML.
	CVRF [120]	*Common Vulnerability Report Format*	XML-based language that enables different stakeholders across different organizations to share critical security-related information in a single format.
	NVD [121]	*National Vulnerability Database*	U.S. government repository of standards based vulnerability management data represented using SCAP. It can be accessed using JSON, XML or RSS feeds.
	OVAL [122]	*Open Vulnerability and Assessment Language*	XML-based Information security community effort to standardize how to assess and report upon the machine state of a computer system.
Configuration management	CCE [123]	*Common Configuration Enumeration*	Provides unique identifiers to system configuration issues for facilitating fast and accurate correlation of configuration data across multiple info sources. It is available in XML and Excel format.
	CCSS [124]	*Common Configuration Scoring System*	Set of measures of the severity of the software security configuration issues.
Asset management	CPE [125]	*Common Platform Enumeration*	Standardized XML-based method for describing and identifying class of application, operating systems, and hardware devices present in enterprise's computing assets.
	ASR [126]	*Asset Summary Report*	XSD data model to express the transport format of summary information about one or more set of assets.
Software assurance	CWE [127]	*Common Weakness Enumeration*	Provides a common language to discuss, find and deal with the causes of software security vulnerabilities as they are found in code, design or system architecture. It is available in several format, including CSV, XML and HTML.
	CWSS [128]	*Common Weakness Scoring System*	Provides a mechanism for prioritizing software weakness in a consistent, flexible and collaborative manner.
	CMSS [129]	*Common Misuse Scoring System*	Set of measures of the severity of software feature misuse (trust assumptions made when designing software features abused to violate security).
	CWRAF [130]	*Common Weakness Risk Analysis Framework*	Part of the Common Weakness Enumeration (CWE) project. It provides a graphical framework for scoring software weaknesses.
Attack taxonomy	CAPEC [131]	*Common Attack Pattern Enumeration and Classification*	Offers a publicly available catalog of common attack patterns classified in an intuitive manner. It can be acquired in XML and CSV format.
Remediation information	CRE [132]	*Common Remediation Enumeration*	Suite of XML-based remediation specifications that enables automation and enhanced correlation of remediation activities.
	ERI [133]	*Extended Remediation Information*	XML dictionary with additional data about each CRE, including references to CPE, CVE, and CCE.
Intrusion detection	IDMEF [134]	*Intrusion Detection Message Exchange Format*	Using XML schema, it defines data formats and exchange procedures for sharing information of interest to IDS/IPS and to the management systems that may need to interact with them.
Cyber threat information sharing and analysis	TMSAD [135]	*Trust Model for Security Automation Data*	Common trust model that can be applied to XML specification within security automation domain.
	OpenIOC [136]	*Open Indicator Of Compromise*	An extensible XML schema that allows the description of the technical characteristics that identify a known threat, an attacker's methodology, or other evidence of compromisation.
	STIX [137]	*Structured Threat Information eXpression*	Collaborative community-driven effort to define and develop a language to represent structured threat information. It is based on XML schemes.
	TAXII [138]	*Trusted Automated eXchange of Indicator Information*	Open transport mechanism that standardizes the automated exchange of cyber threat information.
	CybOX [139]	*Cyber Observable eXpression*	Standardized XML-based language for encoding and communicating high-fidelity information about cyber observables, that are noticeable events or properties in the operational cyber realm.
Security benchmark	XCCDF [140]	*eXtensible Configuration Checklist Description Format*	XML-based specification language to write security checklists, benchmarks and related documents.
Incident management	IODEF [141]	*Incident Object Description Exchange Format*	Defines a data representation that provides a framework for sharing information commonly exchanged by Computer Security Incident Response Teams (CSIRTs) about computer security incidents.
Malware management	MAEC [142]	*Malware Attribute Enumeration and Characterization*	Standardized language for encoding and communicating high-fidelity information about malware based upon attributes such as behaviors, artifacts, and attack patterns. It is available in XSD and HTML format.

TABLE 2.6: Side-by-side comparison of the surveyed works based on the features presented in Section 2.3.2.2

Surveyed Works	Features for countermeasures strategy solutions								
	Attack modeling	Countermeasures provision techniques	Outcomes assessment	Type of reaction	Used standards	Automation level	Performance		
							Complexity		
							Scalability		Response
(2012) Dewri et al.[153], Section 2.3.3.1	• Augmented Attack Trees	• Genetic Algorithm for single & multi-objective optimization • Attacker-Defender "Arms Race"	• Test bed: Small scale simulation • Admin's role: N/A	Static	No	Manual	Low	High	N/A
(2012) Poolsappasit et al. [98], Section 2.3.3.2	• Bayesian Attack Graphs	• Genetic algorithm for single & multi-objective optimization	• Test bed: Small scale simulation • Admin's role: Tuning	Static	CVSS	Semi-automated	Low	High	N/A
(2012) Roy et al. [160], Section 2.3.3.3	• Attack Countermeasure Trees	• Branch & Bound integer optimization algorithm	• Test bed: Small scale simulation • Admin's role: Tuning	Static	No	Semi-automated	Med	Low	Avg
(2012) Viduto et al. [163], Section 2.3.3.4	• No attack modeling. An attack model could be applied	• Multi-objective tabu search	• Test bed: Medium scale simulation • Admin's role: Tuning	Static	CVE	Semi-automated	N/A	Low	Avg
(2013) Chung et al. [169], Section 2.3.3.5	• Attack Graphs	• Heuristic optimization method	• Test bed: Small scale real virtual network • Admin's role: N/A	Both	CVSS, CVE NVD	Fully-automated	High	Low	N/A
(2013) Wang et al.[145], Section 2.3.3.6	• Attack Graphs • Hidden Markov Model	• Ant Colony Optimization	• Test bed: Small scale simulation • Admin's role: Tuning	Static	NVD, CVSS	Manual	Med	Med	Slow
(2013) Zonouz et al. [176], Section 2.3.3.7	• Network connectivity matrix • Competitive Markov decision process	• Bellman's optimization method	• Test bed: Medium scale simulation • Admin's role: Feedback	Both	No	Semi-automated	High	Med	Fast
(2014) Zonouz et al [161], Section 2.3.3.8	• Attack Response Trees • Partially observable Markov decision Process • Game theory two- player Stackelberg stochastic game	• Bellman's optimization method	• Test bed: Small scale simulation • Admin's role: Tuning	Dynamic	No	Semi-automated	High	N/A	Avg
(2015) Granadillo et al. [182], Section 2.3.3.9	• No attack modeling. Static allocation of attacker using network evidence	• Heuristic RORI-based optimization method	• Test bed: Small scale simulation • Admin's role: N/A	Dynamic	No	Semi-automated	Low	High	N/A
(2015) Miehling et al.[191], Section 2.3.3.10	• Partially observable Markov decision Process • Bayesian Attack Graphs	• Bellman's optimization method	• Test bed: Small scale simulation • Admin's role: N/A	Dynamic	No	Fully-automated	Low	High	N/A
(2015) Shameli-Sendi et al. [104], Section 2.3.3.11	• Attack Graphs • Service Dependency Graphs	• Simple Additive Weighting and Weighted Product MCDM • Pareto optimality	• Test bed: Medium scale simulation • Admin's role: Tuning	Dynamic	CVE	Semi-automated	Low	Low	Fast
(2013-2016) Kotenko coutermeasure selection in SIEM [195, 196, 197, 198, 199], Section 2.3.3.12	• Attack Graphs • Service Dependency Graphs	• Heuristic optimization method	• Test bed: Small scale simulation • Admin's role: Tuning	Both	SCAP : CRE, ERI, CVE, CVSS, CWE, CPE	Semi-automated	Low	High	N/A
(2012-2014) Kotenko attack modeling [200, 201, 202, 203, 204, 205, 206], Section 2.3.3.12	• Attack Graphs • Service Dependency Graphs	• Heuristic optimization method	• Test bed: Large scale simulation • Admin's role: Tuning	Both	CPE, CVE, CVSS, CAPEC	Semi-automated	Low	High	Slow
(2016) Shameli-Sendi et al. [208], Section 2.3.3.13	• Attack Defense Trees • Service Dependency Graphs	• Simple Additive Weighting • Pareto optimality	• Test bed: Small scale real virtual network • Admin's role: Tuning	Dynamic	CVE	Fully-automated	Low	Medium	Fast

2.3.4 Research Challenges and Future Directions

This Section builds on top of the previous one by detailing on the challenges in the area of countermeasure strategies against cyber attacks. The discussion revolves around 6 factors that, according to the conducted survey, seem to be the most prominent in the development of solutions in this particular area.

2.3.4.1 Scalability

As it is shown in Table 2.6, one of the main limitations of the analyzed works is the poor scalability of the proposed solutions. Only a few of them propose an approach which is scalable, in the sense that complexity does not increase exponentially with the number of parameters included in the attack model. In fact, this characteristic is clearly reflected in the pilot implementations of the described works; one can easily notice that there is a lack in including an experimental section which assesses the feasibility of the solution on a large-scale environment in terms of number of hosts and interconnections. It is therefore clear that the presented implementations can be seen only as prototypes and they cannot reliably reflect the size and complexity of real-life networks.

We identify the attack modeling as the main cause of this deficiency. That is, most of the reported works use AGs and ATs as the referring model for exploring possible paths, which may be selected by an attacker. As stated in Section 2.3.2.2, these representations are widely used by the research community to model the attacker's steps, because they are able to reveal the cause-consequence relationship among the represented nodes of the graph, taking into account some elemental parameters of the network, including its topology, connections, vulnerabilities, and so forth. The described features are really useful in case of multi-step attacks, where the ability of predicting the attacker's pathway has a key role. However, the size of these representations becomes quickly unmanageable as the size of the network grows and the interconnections among the nodes become denser. Moreover, alongside with the graphical representation of the attacker's steps, a computation of the paths' probability must be executed. However, this task increases the overall complexity of the attack modeling.

Some methodologies, like that in [95], do attempt to reduce this limitation. One obvious improvement can be achieved by building the graphs *off-line*, and then update them

when the input parameters change. In this way, the graphs do not have to be generated continuously, and the saved resources can be allocated for the analysis phase [98]. However, this process does not consider that, especially for dynamic environments, there is a high probability that the input used to build the graph (i.e., network topology, asset vulnerabilities, and others) may frequently change over time. In this case, the graph generation cannot be considered as a one-time cost. That is, a dynamic procedure is needed to update this representation by adding or removing nodes without regenerating the graph from scratch.

Another possible amelioration in the analysis of the graphs is the usage of special algorithms which are able to decrease the analysis complexity and calculate the path probabilities faster [209, 210]. The contribution of these algorithms lies mainly in a reduction of the search time in graphs, thus it can be safely argued that the application of these methodologies is suitable for this kind of problems. Given the evidence of an intrusion on a graph node, all the connected paths must be extracted. Besides this, the probabilities assignment on the paths has to be computed in an efficient way, so as to make the graph computation affordable for dynamic scenarios.

Perhaps, this inherent scalability issue can be solved if looking at it from a broader perspective, that is, by focusing on all the development stages of the reaction system:

- At the design stage, the system should be arranged as a distributed architecture, which scales better for complex networks.
- At the implementation stage, more efficient analysis algorithms should be used for the purpose of calculating faster the likelihoods and the possible connections of the attacker paths, and predicting their next steps.

2.3.4.2 Countermeasure knowledge

Another shortcoming that emerged during this study is the lack regarding the *countermeasure knowledge*. Each one of the surveyed works presents only a limited set of countermeasures, which is used to counteract specific attacks reported to the monitored system. In this way, the selection process loses a great part of its importance and effectiveness, because the usage of an optimization technique on a limited search space is not advantageous, but rather it increases the algorithmic complexity.

In our opinion, a comprehensive pool of countermeasures is a *sine qua non* for any reactive system. This pool must contain atomic actions (or a combination of them) which can be undertaken to block and eradicate possible attacks. By doing so, the number of possible choices for the defender to react will be considerable, giving them two main advantages: (a) the possibility to defeat more potential intrusions, and (b) the potential to select the most appropriate countermeasure (or a set of them), which satisfies the required trade-off between cost, impact, and effectiveness of the hardening measures. Moreover, with a large pool of countermeasures, the use of an optimization algorithm is sure to offer its own advantages, giving its ability to react based on a combination of countermeasures.

Nevertheless, there is a need for reliable sources of information to build this knowledge. One possible solution is to use security administrators' expertise. As already pointed out, we do consider the important role that security administrators should play in this strategy. These people have certain budget constraints, so the selection of countermeasures cannot exclude their analysis and approval. Moreover, the task to define and control the trade-off between effectiveness and cost of the countermeasures is assigned to them, as they represent in this vision a central point in which the flow of knowledge must pass through. Having in mind these considerations, a security expert should not be the *only* source of knowledge; rather, they should directly act only in critical situations, and provide feedback to the countermeasure selection process.

As described in Section 2.3.2.2, another possible way to tackle this problem is to use open standard platforms. The CVE database [88], for example, presents a list of known vulnerabilities which have been discovered in computer systems during the recent years. The format of a CVE entry provides a *reference* field, which normally provides a link to an HTML page describing the problem and, in most cases, possible workarounds. Following this methodology, a translator, say, in the form of a software gateway can be used for acquiring knowledge in text format and transform it into another that is understandable by the underlying machines [211]. Extending this concept to more vulnerability platforms of this kind, one can anticipate the acquisition of a nearly exhaustive knowledge of countermeasure solutions to perform an accurate and successful reaction.

2.3.4.3 Standard representation

Another deficiency which arose during the study of the various works consists in the absence of a *standard representation* for the countermeasures. This issue is directly linked with the previous one, meaning that with the presence of a common and shared reaction intelligence, a standard format which represents the counteractions will be greatly appreciated.

Following the same reasoning exposed for the countermeasure knowledge, we consider a standard countermeasures representation as a key feature in this context, as it can enable essential information sharing among the different actors, thus leading to an increase in the effectiveness of the implemented actions in the mid or long-run.

It is true that so far some attempts have been done towards this direction. An example is the *Common Remediation Enumeration* (CRE) [132], which is a component of the SCAP [146]. A CRE entry is a set of properties that describe a specific remediation instance, including a single configuration setting change, the application of a patch, installation/de-installation of software, a system reboot, and many others. Specifically, a CRE entry has the following parameters:

- A *CRE-ID*, a textual identifier which can be used to uniquely name a single remediation instance.
- A textual, human-understandable *description* of the entry detailing the method and the effect of the remediation instance;
- An optional list of *parameters* applicable to the entry with specific constraints.
- The platform on which the entry is valid, expressed as a *Common Platform Enumeration Applicability Language* expression. The latter defines a standardized way to describe IT platforms by forming complex logical expressions out of individual CPE names and references [212].
- Supporting *references* and *metadata* related to the CRE entry.

A first draft of the CRE was given in 2011, but it was discontinued as of March 2016 for reasons not publicly known. Obviously, more research is needed to better appreciate

the helpfulness of this representation, and to find ways in which it can be exploited for maximizing its benefits.

2.3.4.4 Correlation between countermeasures and attacks

Once a comprehensive pool of countermeasures has been created, a thorough study on the *correlation* between atomic reaction steps and attacks is also needed. This is another arduous aspect regarding the countermeasure strategy, especially when considering the great mass and the complexity of possible attacks. Until now, a handful of attempts have been conducted to solve this issue in [163, 167], but all of them require a great effort from the security expert. This is because this kind of solutions rely on administrator's knowledge about each threat. Hence, a systematic methodology is needed toward intertwining the available attack set with the appropriate countermeasures.

The first action to achieve such a correlation is to study per given countermeasure the number (or category) of attacks it can cover. For example, let us consider the countermeasure "*block a certain range of IP addresses*". This can be easily done by adding a specific access control list in the FW. Also, such a rule is able to defeat different categories of attacks, including probing, DoS and bruteforce, among others.

Then, an analysis needs to be conducted on the effect of the combined countermeasures against a specific attack. This requirement is needed for complex attacks, when applying a single countermeasure may be not enough, or when a combination of security measures is preferred because their execution is more effective or more convenient as the case may be. For instance, suppose that an information leakage is detected in a specific machine in the network. The first action which should be undertaken is to remove all the privileges assigned to that machine in order to block the malicious activity. However, this may be not enough to stop the intrusion; the machine should be also isolated by blocking its network connections.

Once the *one-to-many* relationships between countermeasures and attacks have been constructed, the logical sequel is to build a *many-to-many* relationship between them. This interlinking should consider also the *context* in which the countermeasures are applied. This means that a specific reaction can be useful in a specific context, but ineffective or useless if it is applied in another scenario.

2.3.4.5 Metrics and scoring system

As discussed in Section 2.3.2.2, the use of metrics is needed to *quantitatively* analyze the experimental results provided by each of the surveyed works, where applicable. So far, in the literature, a plethora of security metrics has been proposed, with the aim of capturing different aspects of the problem, including attacks, networks topologies, cost and vulnerabilities metrics [112]. Nevertheless, from our study, we realized that there is a prominent lack of specific and commonly used measurement systems for reliable countermeasure assessment. While the application of a specific countermeasure is for blocking an attack, it may also involve side effects, which should be considered and quantified for optimizing the whole process.

As already pointed out in Section 2.3.2.2, currently a scoring system for the vulnerabilities has been developed by the *Forum of Incident Response and Security Teams* (FIRST) [143].

In this mindset, the creation of a countermeasure scoring system is highly desirable. An important aspect to consider in this regard is the possibility of adapting the countermeasure score depending on temporal and environmental aspects, as CVSS does for the vulnerabilities. That is, as a vulnerability evolves during time and changes its impact depending on the system where it is applied, also a countermeasure should be considered as an *evolving entity* which, depending on the above factors, updates its score.

From the survey of works conducted in the context of this chapter, a countermeasure scoring system should consider the following parameters:

- *Effectiveness* of the solution. It is expressed as a percentage of coverage or a probability of success.
- *Scope* of the reaction. It is expressed as the ability of impacting other components in the system, which could be directly or indirectly affected by its enforcement.
- *Maturity* of the solution. It reflects the elapsed time from which the solution has been deemed as functional.
- *Impact* on the system. It is expressed as a function of *availability*, *confidentiality* and *integrity* impact of the countermeasure on the ICT system.

- *Implementation cost.* It is part of the direct costs of the countermeasure, regarding its activation in the system.
- *Maintenance cost.* It represents the economic cost to sustain the implementation of a countermeasure for long-term reactions; it is calculated only for the long-term countermeasures.
- *Indirect cost.* It reflects the collateral economic damage caused by a countermeasure, as the effect of the countermeasure on a legitimate user of the system, the deactivation cost for a particular invasive remediation, and so forth.

Considering the possible adjustments using temporal and environmental aspects, the score assigned to a given countermeasure can be considered as *adaptable*, solving in this way also the lack of adaptability existing in the analyzed works. Once a suitable measurement system has been developed, a testing phase to judge upon its capabities is desiderable. In this mindset, the construction of a full-fledged benchmark dataset containing both single and multi-step attacks based on real network data would be a great pointer for future research in this field. The proposal of Shiravi et al. [27] can be of great help in this direction as it introduces a set of guidelines on how to build valid datasets, which can be followed to create new ones.

2.3.4.6 Mitigating zero-day attacks

A last research challenge extracted from the survey contucted in Section 2.3.3 is the mitigation of zero-day attacks. The dissection of the works reveals a notable shortcoming of the literature, as the vast majority of the proposed countermeasure provision solutions neglect any kind of reaction against unknown offensive incidents. Even though the authors in [176, 153, 196, 200, 203] elaborate on the problem of responding upon zero-day attacks, their implementations and the experimental testbeds advocate that they cannot be considered as concrete solutions. Without doubt, detecting attacks derived from zero-day vulnerabilities is a challenging task. One could say that counteracting such attacks is even more challenging. In fact, until a fix is published to patch the zero-day vulnerability, the corresponding systems remain unprotected. This gives additional value to a countermeasure system which is able to provide cost-benefit mitigation strategies, and to a comprehensive static countermeasure planning.

The inefficiency of the current proposals stems mainly from the fact that the attack modelling representations are capable of representing specifically defined states of the system, thus omitting attack paths that could appear under zero-day vulnerabilities. POMDP and HMM representations have been used in [191, 79] respectively to quantify uncertainties concerning the attacker's position or their chosen attack path, but not to model the potential existence of unknown system's security states. To this end, such models could be proved beneficial in conjunction with zero-day vulnerability metrics [207, 79] and security automation standards such as CWE[127]. Additionally, an approach of generating AGs by taking into account zero-day vulnerabilities [200] sounds very promising for building countermeasure systems in this direction. Without doubt, the effectiveness and responsiveness of such a system constitute an additional research challenge.

2.3.5 Conclusions

The convergence of network technologies around IP and the openness to the Internet and IoT, present major challenges from a security viewpoint. Today, more than ever, organizations are facing a plethora of highly diversified cyber attacks, which tend to be more ingenious and decisive. In this highly offensive and dynamic terrain, the need for full-blown, fine-grained, and possibly automated reaction strategies in terms of optimal countermeasure selection is highly demanded and urgently needed. In fact, this necessity is observed in the recent literature of ICT security by a number of works published in well-respected journals and conferences during the last 5 years. In this context, this chapter offers a comprehensive study of these works, fulfilling the following three goals. First, it extracts common criteria that can be used as a basis for comparing the various existing (and future) works in this evolving field. Second, it delves into each of the studied works, and through a critical discussion, pinpoints its advantages and disadvantages. This, in synergy with the identified criteria, leads to a comprehensive side-by-side comparison of the included works and helps the reader to obtain a holistic view of this particular field. Last, it elaborates on the future research directions and challenges in this topic, which can be used as a reference to anyone interested in grasping the diverse facets of this area of research.

Chapter 3

Dendron: Genetic trees driven rule induction for network intrusion detection systems

The previous chapter provided an overview of the intrusion detection and response realm by presenting and analyzing a range of proposed systems and methodologies. Naturally, in order to properly respond and prevent the negative consequences of a cyber attack, it is a primary premise to detect it first. In fact, any responsive mechanism destined to provide cost-benefit countermeasures should be triggered upon a correctly identified incident. In this regard, a high False Alarm Rate (FAR) of a detection system could lead to an outburst of unnecessary responses, which in turn would affect the availability of services and introduce additional workload to the administrators. Additionally, it is of major importance to detect the exact type (class) of an attack, as an optimal response should be tailored to the nature of an attack and different types of attacks require different counteraction strategies.

To this end, it becomes obvious that the effectiveness of a responsive system is strongly connected to the performance of the detection engine. Hence, a detection system, which is in synergy with a responsive system, should be capable to keep the FAR to acceptable levels and accurately designate the class where cyber attacks belong to. In this direction, misuse (also known as signature-based) IDSs seems to be a more appropriate solution in contrast to anomaly IDSs, as the latter are known for producing high FARs and they

solely distinguish between legitimate and anomalous behavior instead of defining the exact attack class. Misuse IDSs are equipped with a set of rules which are posed against network traffic with the aim of matching attack descriptions, i.e., signatures of attack vectors. On the other hand, while these systems are able to detect known attacks, they miss to identify novel attacks or variations of known ones. Thus, the detection ability of a misuse detection system is directly affected by the freshness of the detection rules it possesses.

In view of the above, this chapter presents *Dendron*, a rule induction methodology based on a hybrid approach of Genetic Algorithms (GAs) and Decision Trees (DTs). *Dendron* aims to evolve DTs through an evolutionary process and result to a set of rules for building a misuse IDS, which is able to detect multiple attack classes and keep FARs to acceptable levels.

3.1 Introduction

Keeping a detection rules database up-to-date is a challenging task that involves system administrators' supervision. Considering the huge traffic volume passing through central network nodes like an IDS, one easily concludes that the rule generation process is necessary to be supported by automated tools able not only to distinguish between legitimate and malicious traffic, but also to infer the specific class of an attack occurring in the target system. Moreover, the set of the detection rules should enable the system to identify attacks with high Attack Detection Rate (ADR) while keeping the FAR low. Generally, false alarms are a cardinal concern in the field, especially when an IDS is involved in collaborative infrastructures [70], [213] and reputation systems [214], [215]. However, there is always a trade-off between FAR and ADR.

Analyzing network traffic in the context of IDS and Machine Learning is proven to be a challenging task mainly because of specific properties of the network traffic flows. Under realistic terms, a network is flooded with normal traffic flows and only a smaller fraction of the traffic may indicate malicious behavior. Currently, state-of-the-art approaches are able to generate rules for detecting popular classes of attacks but largely neglect the minority attack classes. Even if these types of attacks are less common, their impact on

the targeted system may be destructive. For instance, attacks such as remote vulnerability exploitation or privilege escalation could lead to a system being compromised by an evildoer or confidential information being leaked, causing financial losses and harming the trustworthiness of the organization. Moreover, a network analysis process focuses on several features that have to be taken into consideration for distinguishing legitimate from malicious traffic.

Overall, the aforementioned data properties combined with the numerous attack types pose a significant challenge for Machine Learning solutions as they expand the search space of the problem and lead to computational intensive procedures. In addition, the imbalanced data hinders the detection accuracy at a great extent. In short, misuse IDSs deal with datasets characterized by:

- Being multi-classed (diverse types of attacks).
- Being multi-featured (several network traffic attributes).
- Being highly un-balanced (many instances of normal network traffic, but very few instances of rare attacks).

In this context, we propose *Dendron*, a novel misuse detection system able to accurately detect both popular and rare types of attacks. Our solution utilizes Decisions Trees (DTs) blended with evolutionary techniques in order to generate detection rules under two main premises. On the one hand, the rules must enable the IDS to take accurate decisions concerning all types of attacks, even the most scarce ones. On the other, emphasis is put on the readability of the rules, that is the generated rules should be linguistically interpretable for human comprehension in order to add value to the system administration task. More specifically, DTs ensure the interpretability of the rules, as the transformation of a DT into classification rules is straightforward. However, DTs are known to significantly neglect the minor classes in a dataset [43], while the size of the tree and the discretization of continuous features poses a challenge. Therefore, *Dendron* takes into consideration a feature reduction strategy to minimize the length of the detection rules, while the equal-frequency discretization [216] technique is adopted to handle the continuous features of the dataset. Moreover, our approach utilizes Genetic Algorithms (GA) which is a well-known optimization solution to increase the classification accuracy of classification models by gradually evolving populations of *Individuals*

based on procedures inspired by natural evolution. By the term *Individuals* we refer to DTs which are gradually evolved through the evolutionary process aiming to generate an accurate and unbiased decision tree able to detect both popular and rare intrusive events.

In short, the contributions of the work presented in this chapter are as follows:

- We propose a new methodology that combines the benefits of DTs and GA with the aim of providing linguistically interpretable and accurate intrusion detection rules. Our method is able to deal with imbalanced datasets and generate intrusion detection rules enabling an IDS to identify both popular and rare types of attacks.
- In the context of GA, we provide a weighted selection probability function for evolving balanced DTs which are not biased toward neglecting those classes represented by a smaller percentage of instances (records) in the dataset.
- Our approach is compared against other state-of-the-art and legacy machine learning proposals in an extensive testbed comprising of three intrusion detection datasets, namely: KDDCup'99 [21], NSL-KDD [22] and UNSW-NB15 [23].
- Our proposal is evaluated and compared against others using six distinct classification metrics. The results show that the proposed solution surpasses equivalent methods in terms of Mean F-Measure (MFM), Average Accuracy (AvgAcc) and Attack Accuracy (AttAcc) classification metrics, while keeping false alarms at acceptable levels.

The rest of this chapter is organized as follows: The next section presents background information concerning the methodology of DTs in the context of GA. The proposed methodology is given in Section 3.3, while in Section 3.4 we provide the evaluation results and the complexity analysis of our proposal. Section 3.5 provides a discussion on the findings. Section 3.6 reviews the related work in the field. The last section concludes and provides pointers to future research.

3.2 Preliminaries

As already pointed out, our methodology is based on a combination of DTs with GA in favor of finding the optimal solution for generating intrusion detection rules. By

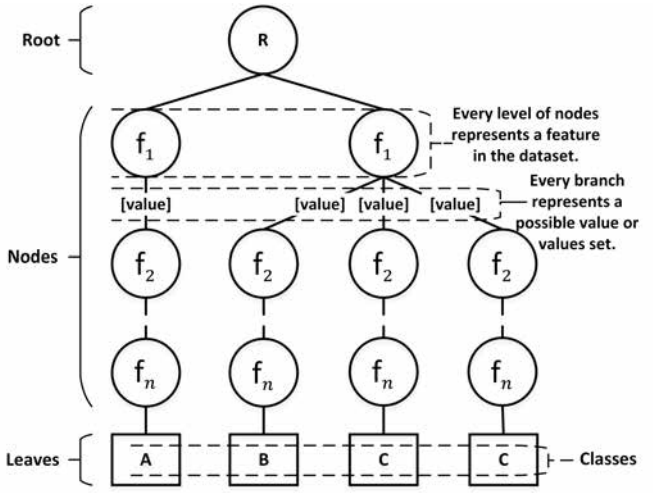


FIGURE 3.1: Structure and properties of a DT

doing so, we meet two main qualities in the field, namely: i) the interpretability of the detection rules to augment human comprehension, and ii) the generation of accurate detection rules.

3.2.1 Decision Trees

DTs are a well-known classification model for supporting decision making in the context of machine learning. An example of a DT is given in Figure 3.1, while the interested reader can refer to section 2.2.4 for more details on DTs.

DTs present several advantages: Transforming a DT into decision rules is straightforward; as shown below, every branch leading from the root of the tree to a leaf, can be represented as an IF $\{Conditions\}$ THEN $\{Class\}$ rule, where the IF clause contains the conjunction of the conditions (derived from the nodes) and the outcome is the class of the leaf. The IF part of the rule is also known as *antecedent* and it is expressed as a sequence of tuples $<feature, operator, value>$, while the prediction part of the rule is also known as *consequent*.

$$IF \left\{ f_1 \; is \; V_1 \wedge \cdots \wedge f_n \; is \; V_n \right\} THEN \; Class = C$$

The simplicity of the decision rules enables one to easily understand the conditions and the outcome. Thus, the premise of human-understandable rules is met. However,

DT algorithms which are driven by information gain significantly neglect those classes represented by a smaller percentage of instances in a dataset [43].

3.2.2 Genetic Algorithms

Evolutionary computation [57] is based on algorithms imitating the evolutionary principles introduced by Darwin's theory. The most prevalent type of evolutionary algorithms is the GAs [217] which are based on the concept of population evolution. GAs are generally composed by the following fundamental steps: (i) Initial population creation, (ii) Individuals selection, (iii) Individuals crossover, (iv) Individuals mutation, (v) Population replacement. These sequential steps are repeated until a termination condition is met. The aim of the evolutionary steps is to explore the search space of the problem and come up with the best solution among the feasible ones. The "best" solution is the one with the highest fitness value, which is calculated by a fitness function. The latter is any metric, in the context of a problem, that can be used to evaluate the candidate solutions, i.e., the individuals of the particular population. GAs are a well-known optimization technique for improving the performance of classification models. For more details about GAs, please refer to Section 2.2.4.

3.2.3 Combining Decision Trees and Genetic Algorithms

In the context of our research, we take advantage of GAs to evolve populations of DTs. Thus, a DT in our approach is considered as an individual in the population which is being evolved through the GA. In the rest of this chapter, the reader should consider the individuals as DTs and vise versa. Decision nodes are considered as genes of individuals. The crossover and mutation operations are applied between two selected individuals, upon a specifically selected gene, with the aim of generating new individuals hopefully "superior" than their ancestors. In the context of our work, any legacy classification metric could be used as a fitness function to measure the classification ability of a multi-classed DT. The evolutionary process evolves individuals toward maximizing their fitness. In this way, our approach of DTs and GAs leads to a set of accurate DTs where the final model is the most accurate among them. More details about the fitness metrics are given in Section 2.2.2, while our approach is explained in Section 3.3.

Parameter	Description
$\mathcal{D}$	Dataset
$\mathcal{P}, \lvert\mathcal{P}\rvert$	Population, Size of Population
$\lvert\mathcal{F}\rvert$	Number of features
$\mathcal{N}$	Number of iterations
$\mathcal{P}', \lvert\mathcal{P}'\rvert$	Temporary population, Size of temporary population
λ	Number of Individuals per class
μ	Mean value of Gaussian Distribution
σ	Standard Deviation of Gaussian Distribution
$parent_i$	i-th Selected Parent
$child_i$	i-th Crossovered Child
$child'_i$	i-th Mutated Child
$\mathcal{I}$	Individual
$\widehat{\mathcal{I}}$	Best Individual
$\widehat{\mathcal{NI}}$	New Best Individual
$\mathcal{C}$	Number of classes in the dataset
$\tilde{\mathcal{C}}$	Missing classes in the best individual
$\lvert\mathcal{L}\rvert$	Total number of leaves in the individual
$\lvert\mathcal{L}_j\rvert$	Number of j-th's class leaves in the individual
$\lvert\mathcal{L}_m\rvert$	Total number of leaves of the missing classes
α, β, γ	Alpha, Beta and Gamma weights
p_i	Percentage of instances of the i-th class in the dataset
$\mathcal{SP}_i$	Selection probability of i-th individual
$depth$	Depth indexing to the node for crossover and mutation
$branch_i$	i-th branch of an Individual of a specific node
φ_i	i-th splitting point of a branch
ι_i	The information gain value for the i-th splitting point
φ	Optimal splitting point
$\mathcal{P}''$	Next Population
θ	Number of extra individuals to be added

TABLE 3.1: Pseudocode parameters.

3.3 Proposed Methodology

The proposed methodology involves specific GA steps to enhance the classification ability of the end model. More specifically, the GA is applied on a population of DTs aiming to traverse the search space of the problem by expanding them. The goal is to explore the possible efficient paths, which will subsequently form the final detection rules. Every step introduces heuristic methods in order to overcome challenges which are mainly posed by the nature of the problem. An overview of the proposed methodology is given in Algorithm 1, while the reader can refer to the pseudocodes that describe low-level operations. Also, to ease the reader in the understanding of the text and algorithms, all the associated parameters (terms) are included in Table 3.1 along with the recommended symbols.

Algorithm 1: Evolutionary Algorithm and Decision Trees.

Data: $\mathcal{N}$, λ, $\mathcal{D}$, σ
Result: Classification Model

1 $|\mathcal{F}| \leftarrow$ `GetNumOfFeatures(`$\mathcal{D}$`)`;
2 $\{p_1, \ldots, p_{|\mathcal{C}|}\} \leftarrow$ `GetClassPercentages(`$\mathcal{D}$`)`;
3 $\mathcal{P} \leftarrow$ `CreateInitPopulation(`λ`)`;
4 $\widehat{\mathcal{I}} \leftarrow$ `Evaluate(`$\mathcal{P}$`)`;
5 $\mu \leftarrow 0$;
6 **for** $i \leftarrow 1$ **to** $\mathcal{N}$ **do**
7 $\mathcal{P}' \leftarrow$ null;
8 $\{SP_1, SP_2 \ldots, SP_{|\mathcal{P}|}\} \leftarrow$ `ComputeSelectionProbability(`$\mathcal{P}$, $\widehat{\mathcal{I}}$, $\{p_1, \ldots, p_{|\mathcal{C}|}\}$`)`;
9 **while** $|\mathcal{P}'| < |\mathcal{P}|$ **do**
10 $\{parent_1, parent_2\} \leftarrow$ `SelectParents(`$\mathcal{P}$, $\{SP_1, SP_2 \ldots, SP_{|\mathcal{P}|}\}$`)`;
11 depth $\leftarrow$ `SelectDepth(`μ, σ, $|\mathcal{F}|$`)`;
12 $\{child_1, child_2\} \leftarrow$ `Crossover(`$\{parent_1, parent_2\}$, depth`)`;
13 $\{child'_1, child'_2\} \leftarrow$ `Mutate(`$\{child_1, child_2\}$, depth`)`;
14 $\mathcal{P}' \leftarrow \mathcal{P}' \bigcup \{\{child'_1, child'_2\}\}$;
15 $\mu \leftarrow (\mu + 1) \bmod |\mathcal{F}|$;
16 **end**
17 $\mathcal{P} \leftarrow$ `Replace(`$\mathcal{P}$, $\mathcal{P}'$`)`;
18 $\widehat{\mathcal{NI}} \leftarrow$ `Evaluate(`$\mathcal{P}$`)`;
19 **if** `GetFitness(`$\widehat{\mathcal{NI}}$`)` $>$ `GetFitness(`$\widehat{\mathcal{I}}$`)` **then**
20 $\widehat{\mathcal{I}} \leftarrow \widehat{\mathcal{NI}}$;
21 **end**
22 **end**
23 **return** $\widehat{\mathcal{I}}$;

3.3.1 Initial Population Creation

The first step is to create an initial population of individuals (alg:1, line:3). Recall that the term *Individual* stands for a DT in the population. The initial population consists of DTs with minimum classification capabilities as they are only able to classify instances of one class and their node's conditions are always true. Exemplary DTs having these properties are depicted in Figure 3.2, where one can easily observe that the trees are not branched at all. Instead, if any dataset instance is submitted to a tree of this kind, then all its attributes will satisfy the corresponding condition of the node and as a result the classification outcome will be the class represented in that leaf. The DTs will be gradually evolved through the crossover and mutation processes described in Sections 3.3.4 and 3.3.5, respectively. The size of the initial population ($|\mathcal{P}|$) is an adjustable parameter based on the number of the traffic classes $|\mathcal{C}|$, as indicated by the dataset, and

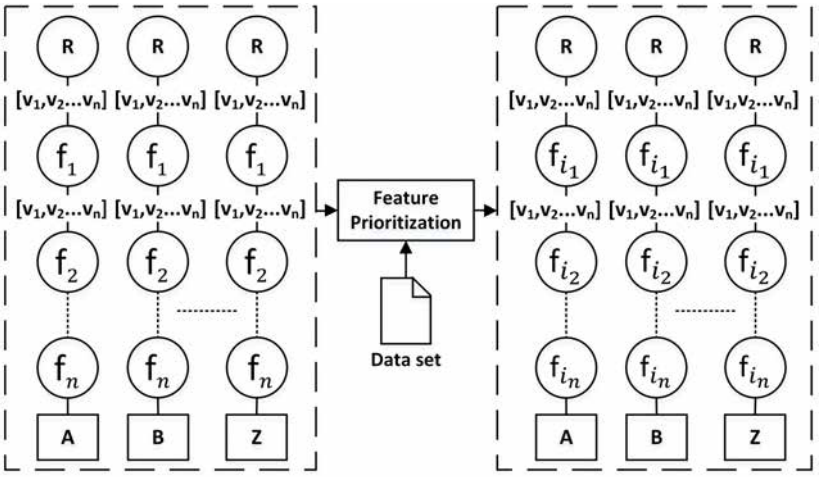


FIGURE 3.2: Initial population creation: The initial individuals are not branched at all. Their nodes are prioritized based on their information gain.

Function 1: CreateInitPopulation(λ, $\mathcal{D}$)

Data: λ, $\mathcal{D}$
Result: $\mathcal{P}$

```
prioritizedFeatures ← CalculateInformationGain(D);
P ← null;
for class ∈ C do
    for j ← 1 to λ do
        decisionTree ← GenerateDecisionTree(class, prioritizedFeatures);
        P ← P ⋃ {decisionTree};
    end
end
return P;
```

the desired number of individuals per class (λ), as indicated by equation (3.1).

$$|\mathcal{P}| = |\mathcal{C}| \cdot \lambda \tag{3.1}$$

The sequence of the nodes of the DTs is a vital aspect for the classification accuracy of the individuals. In this direction, the prioritization of the decision nodes is driven by the information gain for every node, given the input dataset (func:1). The information gain metric is used to evaluate the worth of an attribute with respect to the class in the dataset. We evaluated the worth of the attributes by utilizing the InfoGainAttributeEval class as used by the WEKA Data Mining software [218]. In this way, the individuals are constructed by placing the more significant decision nodes closer to the root of the tree. It must be noted that the sequence of nodes of each tree in the population is the same. The aforementioned procedure is intuitively represented in Figure 3.2.

Once the initial population is created, it is given as an input to the genetic evolutionary process for maximizing the classification accuracy of the individuals.

3.3.2 Parents Selection

The GA involves a selection process (func:6) aiming to gradually create a new population, possibly better than the previous one. To do so, the selection process is driven by a probabilistic method to come up with two individuals, the parents $\{parent_1, parent_2\}$, who will be crossovered and mutated in subsequent steps in order to construct two new individuals (alg:1, line:10).

The probabilistic method for calculating the selection probability for each of the individuals $\mathcal{I}_i$ in the population is given by formula 3.2, while a pseudocode implementation is given in function 5.

$$F(\mathcal{I}_i) = \alpha \cdot g_1(\mathcal{I}_i) + \beta \cdot g_2(\mathcal{I}_i) + \gamma \cdot g_3(\mathcal{I}_i) \tag{3.2}$$

where:

- $g_1(\mathcal{I}_i) \in [0, 1]$ is the actual fitness function for the i-th individual.
- $g_2(\mathcal{I}_i) \in [0, 1]$ is the class-based selection function for the i-th individual.
- $g_3(\mathcal{I}_i) \in [0, 1]$ is the missing classes function of the best individual.
- α, β, $\gamma \in [0, 1]$ are the weights of g_1, g_2 and g_3 respectively (where $\alpha + \beta + \gamma = 1$).

Fitness function The fitness function g_1 stands for the classification metric chosen to measure the classification ability of the individual (func: 2). Several legacy classification

Function 2: g_1=GetFitness($\mathcal{I}$, $\mathcal{D}$)

Data: $\mathcal{I}$
Result: Fitness Metric

```
1 confusionMatrix ← GenerateConfusionMatrix(I, D);
2 fitness ← { Accuracy(confusionMatrix);
              AverageAccuracy(confusionMatrix);
              MeanF-Measure(confusionMatrix);
              AttackAccuracy(confusionMatrix);
              AttackDetectionRate(confusionMatrix);
              FalseAlarmRate(confusionMatrix);
3 return fitness;
```

metrics could be used as a fitness function. More details on this process are given in Section 3.4, while Section 2.2.3 provides the definitions of the classification metrics.

Class-based Selection function The Class-based Selection function $g_2(\mathcal{I}_i)$ aims to instruct the system toward maximizing the selection probability of those individuals containing more leaves of the minority classes for the given dataset (func:3). This function is represented by the following formula:

$$g_2(\mathcal{I}_i) = \sum_{j=1}^{|\mathcal{C}|} \Big((1 - p_j) \frac{|\mathcal{L}_j|}{|\mathcal{L}|} \Big), \tag{3.3}$$

where:

- $|\mathcal{C}|$: the number of classes in the dataset.
- p_j: the percentage of instances of the j-th's class in the dataset.
- $|\mathcal{L}_j|$: the number of j-th's class leaves in the individual $\mathcal{I}_i$.
- $|\mathcal{L}|$: the total number of leaves in $\mathcal{I}_i$.

Missing Classes function The missing classes function g_3 aims to instruct the system toward maximizing the selection probability of those individuals containing leaves for classes missing from the best individual (func:4).

$$g_3(\mathcal{I}_i) = \frac{\sum_{j=1}^{m} (|\mathcal{L}_j|)}{|\mathcal{L}|}, \tag{3.4}$$

where:

- m: the number of missing classes in the best individual.

The proposed probabilistic method $F(\mathcal{I}_i)$ is weighted by three attributes α, β and γ, applied to g_1, g_2 and g_3, respectively, in order to adjust the final result of the system accordingly.

The α and β weights are adjusted during the initialization of evolutionary process, while γ weight has a non-zero value if and only if there are missing classes at the leaves of

Function 3: g_2=ClassSizeBasedSelection($\mathcal{I}$, $\{p_1, \ldots, p_{|\mathcal{C}|}\}$)

Data: $\mathcal{I}$, $\{p_1, \ldots, p_{|\mathcal{C}|}\}$
Result: Class size-based selection probability

1 selectionProbability $\leftarrow 0$;
2 **for** class $\in \mathcal{C}$ **do**
3 $\quad |\mathcal{L}_{class}| \leftarrow$ `GetNumberOfLeavesForClass`(class, $\mathcal{I}$);
4 $\quad |\mathcal{L}| \leftarrow$ `GetTotalLeaves`($\mathcal{I}$);
5 $\quad$ selectionProbability $\leftarrow$ selectionProbability $+ \left((1 - p_{\text{class}})\frac{|\mathcal{L}_{class}|}{|\mathcal{L}|}\right)$;
6 **end**
7 **return** selectionProbability;

Function 4: g_3=MissingClassesFunction($\mathcal{I}$, $\widehat{\mathcal{I}}$)

Data: $\mathcal{I}$, $\widehat{\mathcal{I}}$
Result: Selection probability based on missing classes in the best individual

1 $\tilde{\mathcal{C}} \leftarrow$ `GetMissingClasses`($\{\ \widehat{\mathcal{I}}\ \}$);
2 **for** class $\in \tilde{\mathcal{C}}$ **do**
3 $\quad |\mathcal{L}_{class}| \leftarrow$ `GetNumberOfLeavesForClass`(class, $\mathcal{I}$);
4 $\quad |\mathcal{L}_m| \leftarrow |\mathcal{L}_m| + |\mathcal{L}_{class}|$;
5 **end**
6 $|\mathcal{L}| \leftarrow$ `GetTotalLeaves`($\mathcal{I}$);
7 selectionProbability $\leftarrow \frac{|\mathcal{L}_m|}{|\mathcal{L}|}$;
8 **return** selectionProbability;

the best individual. If the best individual is "normal", in the sense that all the classes indicated in the dataset are present in its leaves, then γ is equal to 0. Therefore, in this case, missing classes function g_3 is not applied in $F(I_i)$. The formula for γ calculation is given in eq. (3.5).

$$\gamma = 1 - \frac{m}{|\mathcal{C}|}(\alpha + \beta), \tag{3.5}$$

The α and β weights are complementary ($\alpha + \beta = 1$) and gradually change during the evolution of the population (func:5, line:1). A possible initial configuration could be an assignment of a high value to β at the initiation of the algorithm (e.g., $\alpha = 0.2$ and $\beta = 0.8$) in order to augment the selection probability $F(\mathcal{I}_i)$ of individuals containing more leaves of the minority classes. Then weight α is being gradually increased, while β is being decreased until finally reaching $\alpha = 1$ and $\beta = 0$. During this gradual alteration

Function 5: ComputeSelectionProbability($\mathcal{P}$,$\widehat{\mathcal{I}}$, $\{p_1, \ldots, p_{|\mathcal{C}|}\}$)

Data: $\mathcal{P}$,$\widehat{\mathcal{I}}$, $\{p_1, \ldots, p_{|\mathcal{C}|}\}$
Result: Selection Probability for each individual

```
{α, β, γ} ← CalculateWeights();
for I ∈ P do
    x ← α· GetFitness(I);
    y ← β· ClassSizeBasedSelection(I, {p_1, ..., p_|C|});
    z ← γ· MissingClassesFunction(I, Î);
    SP_I ← x + y + z ;
end
return {SP_1, SP_2 ..., SP_|P|};
```

of α and β, γ calculation formula (3.5) is applied if and only if there are missing classes in the best individual.

From formula (3.5) it can be inferred that γ gets specific values based on the total number of classes in the dataset $|\mathcal{C}|$ and the missing classes in the best individual m. When $\gamma \neq 0$, α and β have to be normalized in order to satisfy the rule $(\alpha + \beta + \gamma) = 1$. Hence, if $\gamma \neq 0$, the normalized values of α and β are calculated as follows:

$$\alpha' = \alpha \cdot \frac{m}{|\mathcal{C}|} \tag{3.6}$$

$$\beta' = \beta \cdot \frac{m}{|\mathcal{C}|} \tag{3.7}$$

Discussion The configuration described above is a key concept of the proposed methodology. According to the literature [43], classifiers tend to neglect the minor classes of a dataset, achieving lower accuracy contrary to the major ones. To counteract this phenomenon, we introduce the selection function $F(\mathcal{I}_i) \in [0, 1]$ in combination with α, β and γ weights. This approach leads the selection function toward selecting parents who are able not only to classify accurately, but also to take into consideration the minor classes of the dataset by increasing the selection probability with respect to the inverse percentage of the instances of a class (parameter $(1 - p_j)$ in eq.(3.3)). The aforementioned trend can be also weighted accordingly using α and β weights. On the other hand, γ weight is introduced in order to ensure that all the classes indicated by the dataset are present in the best individual. As stated in Section 3.3.1, the initial DTs are single-class only. The final classification model should be able to infer on traffic instances of all classes, thus via the evolutionary approach our methodology has to

Function 6: SelectParents($\mathcal{P}$, $\{SP_1, SP_2 \ldots, SP_{|\mathcal{P}|}\}$)

Data: $\mathcal{P}$, $\{SP_1, SP_2 \ldots, SP_{|\mathcal{P}|}\}$
Result: $\{parent_1, parent_2\}$

1 $\{parent_1, parent_2\} \leftarrow$ `RouletteWheelSelection(`$\mathcal{P}$,$\{SP_1, SP_2 \ldots, SP_{|\mathcal{P}|}\}$`);`
2 **return** $\{parent_1, parent_2\}$;

ensure the presence of all classes in the best individual. This is why γ weight is applied with $g_3(\mathcal{I}_i)$ in equation (3.2). Moreover, in order to select the two parents which will be crossovered and mutated, we utilize the Roulette Wheel [219] selection technique (func:6, line:1). This technique is known for giving higher chances to the candidate individuals with higher selection probability to be selected. However, at the same time, it does not completely neglect those with lower selection probability.

3.3.3 Depth Selection

This step is a preparatory process for the crossover and mutation operations. The depth selection (alg:1, line:11) utilizes a Gaussian distribution (func: 7) with μ (mean) varying from 0 to the maximum depth of DTs (which is equal to the total number of features in the dataset $|\mathcal{F}|$). More specifically, μ is gradually increased by 1 every time 2 parents are selected. In this way, every next selected parents are crossovered and mutated potentially to a lower node. Figure 3.3 depicts the gradually moving distribution. When μ reaches its maximum value is then reinitialized to 0. In this way, the parents will start again being crossovered and mutated in higher levels, and so forth. The standard deviation σ can be adjusted accordingly.

We should keep in mind that the problem of generating detection rules is a search problem with a quite large search space. This is why the problem of learning an optimal DT is known to be NP-complete [220]. The goal of Depth Selection approach is to

Function 7: SelectDepth(μ, σ, $|\mathcal{F}|$)

Data: μ, σ, $|\mathcal{F}|$
Result: depth
1 depth $\leftarrow$ -1;
2 **do**
3 | depth $\leftarrow$ `GaussianGenerator(`μ,σ`);`
4 **while** (depth < 0) $\vee$ (depth $> |\mathcal{F}|$);
5 **return** depth;

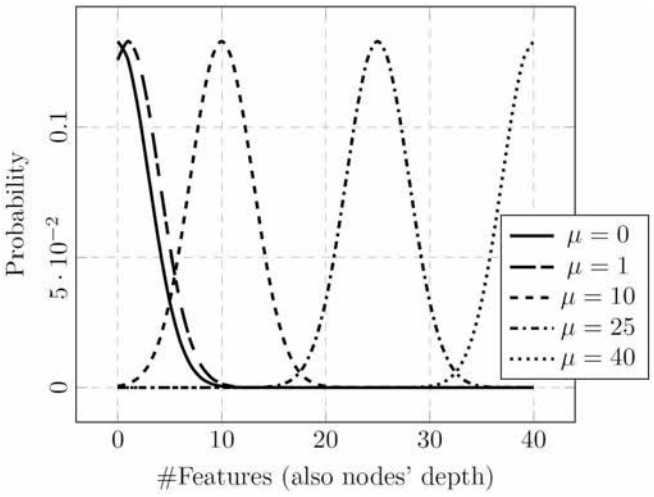


FIGURE 3.3: Gradually moving Gaussian distribution.

gradually create highly branched individuals on multiple levels (nodes) and explore the search space of the problem in a more efficient way.

3.3.4 Parents Crossover

The crossover process (func:8) is applied to the individuals and more specifically to the two parents $\{parent_1, parent_2\}$ who were selected from the selection step described in Section 3.3.2. The evolutionary strategy synthesizes individual's characteristics by swapping randomly chosen branches between the selected individuals with the aim of constructing individuals of high variance. The swapping process is intuitively represented in Figure 3.4, where the dashed branches are swapped resulting into two new individuals $\{child_1, child_2\}$. This synthesis occurs based on a specific probability (Crossover probability) and to a specific depth among the nodes of DTs. The depth (the node) on which the swapping is applied is indicated by the depth selection process described in Section 3.3.3.

Function 8: Crossover($\{parent_1, parent_2\}$, depth)

Data: $\{parent_1, parent_2\}$, depth
Result: $\{child_1, child_2\}$

1 $branch_1 \leftarrow$ `SelectBranchRandomly(parent`$_1$`,depth)`;
2 $branch_2 \leftarrow$ `SelectBranchRandomly(parent`$_2$`,depth)`;
3 $\{child_1, child_2\} \leftarrow$ `SwapBranches(`$\{parent_1, parent_2\}$, $branch_1$, $branch_2$`)`;
4 **return** $\{child_1, child_2\}$;

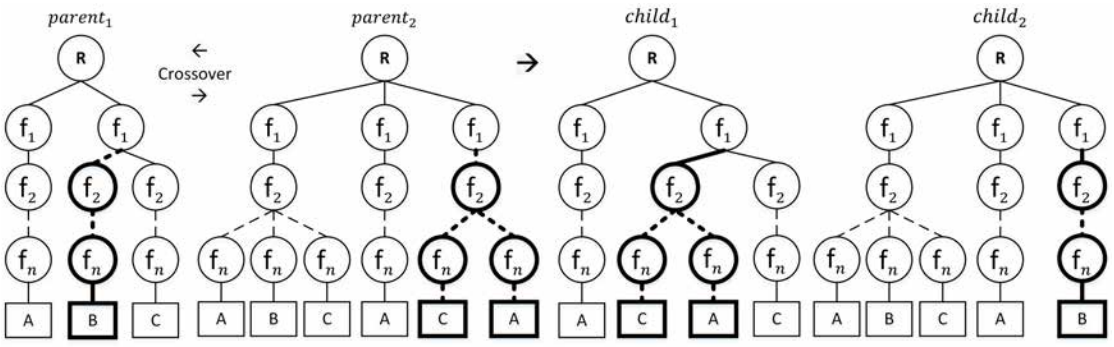


FIGURE 3.4: Crossover operation: Swapping branches between two individuals ($parent_{1,2}$) to a specific node, results to two new individuals ($child_{1,2}$), as described in Algorithm 1.

The crossover process creates populations with a high variation contributing in this way to the problem of exploring the search space more efficiently.

3.3.5 Parents Mutation

The mutation process (func:9) is applied on the crossovered individuals $\{child_1, child_2\}$ in terms of splitting decision nodes on specific predefined splitting points. The mutation process is depicted in Figure 3.5. Actually, this operation is responsible for expanding the DTs by generating new branches for the tree nodes. In this way, new paths connecting the root with the leaves are created. This affects the trees in two ways: on the one hand, the generation of a branch contributes to the exploration of the search space and, on the other, it creates a new set of rules.

Function 9: Mutate($\{child_1, child_2\}$, depth)

Data: $\{child_1, child_2\}$, depth
Result: $\{child'_1, child'_2\}$

for $\mathcal{I} \in \{child_1, child_2\}$ **do**
  $\{branch_1, \ldots, branch_n\} \leftarrow$ `GetBranches(`$\mathcal{I}$, depth`)`;
  $\{\varphi_1, \ldots, \varphi_n\} \leftarrow$ `GetSplittingPoints(`$\{branch_1, \ldots, branch_n\}$`)`;
  if $\{\varphi_1, \ldots, \varphi_n\} \neq \emptyset$ **then**
    $\{\iota_1, \ldots, \iota_n\} \leftarrow$ `ComputeInfoGain(`$\{\varphi_1, \ldots, \varphi_n\}$`)`;
    $\varphi \leftarrow$ `GetBestSplittingPoint(`$\{\varphi_1, \ldots, \varphi_n\}$, $\{\iota_1, \ldots, \iota_n\}$`)`;
    mutatedIndividual $\leftarrow$ `SplitBranchAtPoint(`$\mathcal{I}$, φ`)`;
  end
  $\{child'_1, child'_2\} \leftarrow \{child'_1, child'_2\} \bigcup \{\text{mutatedIndividual}\}$;
end
return $\{child'_1, child'_2\}$;

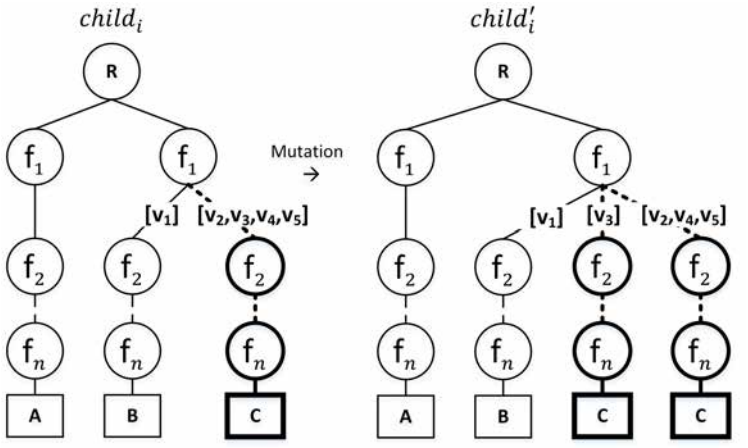


FIGURE 3.5: Mutation operation: Creating a new branch to the individual on a specific node for the value that achieves the highest information gain ($[V_3]$ in this example).

A feature in the dataset represented by a node can be either continuous or discrete. The splitting values of a discrete attribute are straighforward but for the continuous attributes a discretization technique is needed in order to define specific splitting points in the continuous interval. Among several methodologies proposed in the literature [221][222] we adopted the *equal-frequency discretization* [216]. This approach decides upon the splitting values so that each resulting interval contains approximately the same number of instances. We adopt this approach mainly because the generated splitting values are less that those derived from the approaches proposed in [221] and [222], minimizing this way significantly the search space.

For example, if equal-frequency discretization is used to discretize the continuous attribute *src_bytes* from KDDCup'99 [21], the derived split intervals are the following:

- **src_bytes**: [0,40571] → [0,1], (1, 202], (202, 249], (249, 330], (330, 40571]

If the value 249 is chosen to split the node then two new branches will emerge:

- **src_bytes**: [0, 249] → [0,1], (1, 202], (202, 249]
- **src_bytes**: (249, 40571] → (249, 330], (330, 40571]

The first derived branch can be split further on 1 and 202, while the second branch could be further split only on 330.

One the other hand, the discrete features are split in a more simple way. In this case, we adopted a "Pull-Left" [223] approach where a new branch is generated by isolating

the value that achieves the maximum information gain. For example, if we assume that the *service* feature (from KDDCup'99 [21]) has 4 distinct values, then as shown below, there are 4 different splitting possibilities.

- **service**: {http, smtp, pop3, ftp}
  - {http}, {smtp, pop3, ftp}
  - {smtp}, {http, pop3, ftp}
  - {pop3}, {http, smtp, ftp}
  - {ftp}, {http, smtp, pop3}

Every splitting decision (both for discrete and discretized continuous features) is taken based on the information gain produced on every possible splitting point for the selected decision node. The splitting point that produces the highest information gain is the one where a new branch is generated (func:9 lines:3-7). If a node cannot be further split or its splitting point does not produce any information gain, based on the given dataset, then the mutation operation is not applied. As in the crossover process, parents mutation occurs based on a specific probability (mutation probability) and the depth selection process (Section 3.3.3) indicates the depth of the node intended to be split. The mutation process leads to individuals who are as compact as possible as their expansion is guided by the information gain. In practical terms, this is translated into a relatively small set of detection rules.

The mutated individuals $\{child'_1, child'_2\}$ will be added to a temporary population. The Selection, Crossover and Mutation operations are repeated (alg:1, lines:7-14) until the size of this temporary population reaches the size of the initial one. When this condition is met, the population replacement operation is initiated.

3.3.6 Population Replacement

The replacement operation (func:10) is responsible for choosing the most suitable individuals between the initial and the temporary population in order to form the next population, which will be used again as input to the evolutionary process. In this process, we evaluate the classification accuracy of the individuals one-by-one and those with the highest fitness are finally moved to the next population (func:10, lines:1-9). It has

Function 10: Replace($\mathcal{P}$, $\mathcal{P}'$, θ)

Data: $\mathcal{P}$, $\mathcal{P}'$
Result: $\mathcal{P}''$

```
1  for i ← 1 to |P| do
2  |  if ( GetFitness(P_i) > GetFitness(P'_i) ) then
3  |  |  I ← P_i
4  |  else
5  |  |  I ← P'_i
6  |  end
7  |  P'' ← P'' ⋃ {I};
8  end
9  missingClasses ←GetMissingClasses(P'');
10 while missingClasses ≠ ∅ do
11 |  I ← FindIndividual(P, missingClasses);
12 |  for j ← 1 to θ do
13 |  |  P'' ← P'' ⋃ {I};
14 |  end
15 |  missingClasses ← GetMissingClasses(P'')
16 end
17 return P'';
```

to be mentioned that the individuals in every population are ordered in a descending order based on their Fitness. The fitness function could be any classification metric from those referred in Section 2.2.3. During the replacement process, several measures have to be taken for ensuring the "normality" of the next population. A "normal" population requires all the classes indicated by the dataset to be present in the leaves of the individuals. That is, if a class is missing from a population, then all the next generations will miss this class, resulting to an undesirable outcome. In this direction, we check for potentially missing classes in the population and if there are any, then we proceed to the addition of extra individuals from the previous population containing the missing classes in their leaves (func:10, lines:10-17). We add several instances of the same individual in the next population in order to maximize the probability to be selected as a parent later in the execution. This is depicted in Figure 3.6 in an intuitive manner. The number of individuals added to the next population is given by parameter θ adjusted proportionally to the total size of the population.

The operations described above are repeated until a specific condition is met. This condition could be a specific number of iterations of the GA, a specific score of a classification metric, or even a time lapse. During this repetitive process, the accuracy of the individuals is gradually increased, whereas the individuals are getting branched. The ultimate goal is to conclude to an end model able to classify all possible types of attacks

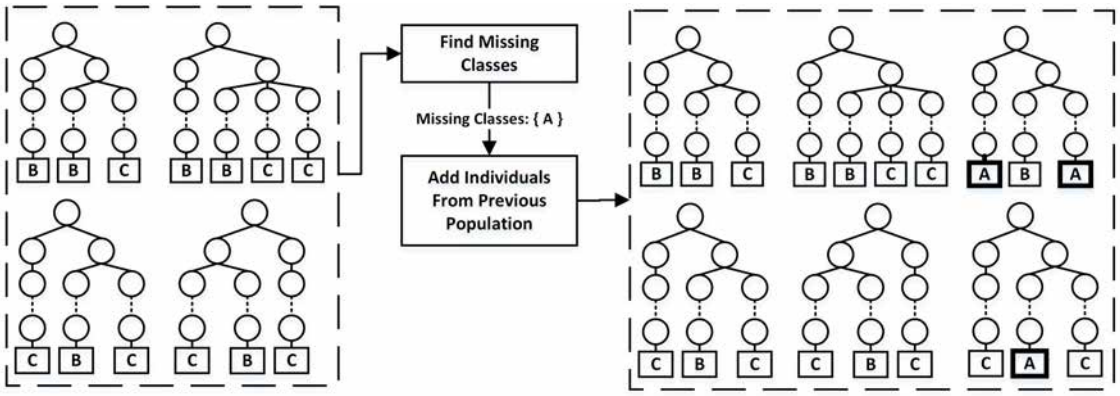


FIGURE 3.6: Population replacement: The new population must be "normal" in the sense that all the classes indicated by the dataset must be present in the leaves of the population.

with the highest possible accuracy.

3.3.7 Population Evaluation

The evaluation process is to decide upon the best DT of a population (func:11). In order to nominate the best individual, the evaluation process can be based on several legacy classification metrics. The evaluation metric is essential because it is used as the actual fitness function in the GA. In this direction, the evolutionary process is guided toward maximizing the chosen metric.

The definition of aspirant classification metrics for a multi-classed approach was given in Section 2.2.3.

Function 11: Evaluate($\mathcal{P}$)

Data: $\mathcal{P}$
Result: $\widehat{\mathcal{I}}$

```
1  bestFitness ← -1;
2  Î ← null;
3  for I ∈ P do
4  |   fitness ← GetFitness(I);
5  |   if ( fitness > bestFitness ) then
6  |   |   bestFitness ← fitness;
7  |   |   Î ← I;
8  |   end
9  end
10 return Î;
```

3.3.8 Feature Selection

In practical terms, feature selection is considered a necessity when dealing with multi-dimensional machine learning problems. According to [224], a feature reduction strategy can improve the prediction performance while providing faster and more cost-effective models. In the context of the current work, the adoption of a feature reduction strategy is translated into shorter in length detection rules improving the human-readability of the rules. As described in Sections 3.3.3 and 3.3.5, the complexity of the Individuals is intensively increased since the possible paths, i.e., detection rules, among the nodes and the corresponding splitting points, are numerous.

In an effort to minimize the complexity of our model and contribute to the overall accuracy, we concluded to 21 features out of the 41 of the KDDCup'99 [21] and NSL-KDD [22]. Precisely, we selected the final subset of the features based on an empirically study of our model and our knowledge on the domain of the problem. Since we aim to balance the accuracy between the major and the minor classes of the dataset, we chose features that can represent U2R (user-to-root) and R2L (remote-to-local) attacks like *root_shell, num_root, num_shells, num_compromised*. But of course, we chose also representative features for the PRB (probing) and DoS (denial of service) attacks like *dst_host_same_srv_rate* and *wrong_fragment* respectively. Regarding the UNSW-NB15 [23] dataset, we took advantage of the Wrapper technique [225], which recursively evaluates diverse feature combinations to define an efficient subset. This method provided us with the features presented in Table 3.3 for the given dataset.

The value of the selected features per dataset will be evaluated based on their information gain in order to prioritize the nodes of the DTs as stated in Section 3.3.1. Tables 3.2 and 3.3 summarize the selected subsets of features for KDDCup'99, NSL-KDD and UNSW-NB15 datasets, respectively. The features are presented prioritized based on their information gain in a descending order. For more details about the various features, the interested reader could refer to the dataset's corresponding publication. Notably, the authors in [224] provide a checklist with steps that could be taken to solve the feature selection problem. Domain knowledge is suggested as a main strategy for constructing efficient sets of features. However, we envision to add an embedded feature selection technique as an extension to our algorithm in a future work.

#	Feature	Type
1	service	discrete
2	flag	discrete
3	src_bytes	continuous
4	count	continuous
5	dst_host_same_srv_rate	continuous
6	dst_host_serror_rate	continuous
7	dst_host_srv_serror_rate	continuous
8	logged_in	discrete
9	dst_host_rerror_rate	continuous
10	duration	continuous
11	protocol_type	discrete
12	hot	continuous
13	num_compromised	continuous
14	wrong_fragment	continuous
15	root_shell	discrete
16	num_file_creations	continuous
17	num_root	continuous
18	num_access_files	continuous
19	num_failed_logins	continuous
20	num_shells	continuous
21	su_attempted	discrete

TABLE 3.2: Selected features for KDDCup'99 and NSL-KDD

#	Feature	Type
1	ct_state_ttl	continuous
2	sttl	continuous
3	dttl	continuous
4	dload	continuous
5	ct_srv_dst	continuous
6	service	discrete
7	proto	discrete
8	dmean	continuous
9	dbytes	continuous
10	state	discrete
11	dpkts	continuous
12	ct_srv_src	continuous
13	ct_dst_sport_ltm	continuous
14	sjit	continuous
15	dloss	continuous
16	sbytes	continuous
17	smean	continuous
18	ct_src_dport_ltm	continuous
19	ct_dst_src_ltm	continuous
20	dwin	continuous
21	response_body_len	continuous
22	ct_flw_http_mthd	continuous
23	trans_depth	continuous

TABLE 3.3: Selected features for UNSW-NB15

3.4 Evaluation

In this section, we evaluate *Dendron* using the three aforementioned intrusion detection datasets. The performance of *Dendron* is also compared against other state-of-the-art and legacy machine learning algorithms under several classification metrics.

3.4.1 KDDCup'99 Dataset

For evaluating *Dendron* using the KDDCup'99, we adopted a set of preprocessing steps which are broadly used in the literature. We sampled the dataset to generate a smaller one, while keeping its initial properties. Moreover, we removed all duplicates from the partial dataset. Table 3.4 gives a detailed overview of the instances included in the training and testing sets we used throughout our experiments. The training set contains 10% of the instances of every category except for the U2R class where 50% of the instances were taken into account. The reader can refer to Section 2.2.2 or to the original publication of the dataset [21] for more details.

Class	KDDCup'99		NSL-KDD	
	Training	Testing	Training	Testing
Normal	8,783	79,049	6,735	60,608
DoS	5,457	49,115	4,600	41,327
PRB	213	1,918	1,164	10,483
R2L	100	899	109	886
U2R	26	26	26	26
Total	14,579	131,007	12,634	131,330

TABLE 3.4: Training and testing set instances for KDDCup'99 and NSL-KDD datasets

3.4.2 NSL-KDD Dataset

For evaluating *Dendron* using the NSL-KDD dataset, we followed the same strategy followed also for KDDCup'99, as we sampled randomly the NSL-KDD to create a training set consisting of the 10% of the initial instances, while the rest 90% was used in the testing phase. The training set contains 10% of the instances of every category except for the U2R class, where 50% of the instances were taken into account. Table 3.4 gives a detailed overview of the instances included in the training and testing sets used throughout our experiments. The reader can refer to Section 2.2.2 or to the original publication of the dataset [22] for more details.

3.4.3 UNSW-NB15 dataset

As it was also the case for the KDDCup'99 and the NSL-KDD datasets, we randomly sampled the UNSW-NB15 dataset to create two subsets for training and testing purposes. The training set consists of approximately the 10% of the instances while the rest 90% was used to build the testing set. This percentage segmentation is applied for each of the 9 classes of the dataset except for the "Worms" class, where 50% of the instances were taken into account. Since we approach the problem of intrusion detection as a multi-classification problem and not as a binary one (i.e. Normal/Anomaly), we excluded the feature that annotates each instance as Normal or Anomaly, but we keep only the actual class of the instances. However, after removing also the "id" feature, which is used to assign a unique number to each instance, we found out that there were duplicated instances in the dataset. By following sound practices as documented in the literature [22], we removed the duplicated records concluding to the training and testing sets, which are detailed in Table 3.5

Class	Training Set	Testing Set
Normal	3,420	30,786
DoS	172	1,546
Reconnaissance	270	2,433
Fuzzers	485	4,353
Shellcode	40	338
Exploits	760	6,849
Generic	366	3,291
Backdoor	40	306
Analysis	45	401
Worms	22	22
Total	5,620	50,325

TABLE 3.5: Training and testing set instances for UNSW-NB15 dataset

3.4.4 Testbed parameters and results

As described in the previous sections, apart from the fact that GAs introduce randomness in the process, our approach brings in several parameters which can affect the results. These two factors render the evaluation of the model hard in the sense that our solution is not deterministic. Therefore, for a specific input, the model will not create the same exact output. Regardless of the variance of the results among the different runs, the best solution is the one which ultimately will be chosen to generate the detection rules. Table 3.6 contains the parameters and the values used for the evaluation of the model for each dataset. We chose to evaluate our model using Mean F-Measure (MFM) as the fitness metric. Our goal is to maximize this average metric to ensure that *Dendron* treats the several classes of the dataset equally. In the context of multi-classification problems, MFM is destined to measure the equilibrium between Recall and Precision

Symbol	Parameters	KDDCup'99	NSL-KDD	UNSW-NB15
$\|\mathcal{C}\|$	Number of Classes	5	5	10
$\|\mathcal{F}\|$	Number of Features	21	21	23
σ	Standard deviation	3	3	3
	Crossover probability	0.95	0.95	0.95
	Mutation probability	1.0	1.0	1.0
λ	Individuals per class	15	10	5
$\mathcal{N}$	Iterations	1400	2400	7000
β	Beta weight	40	40	70
	Num. of discretized bins	6	6	5
θ	Additional individuals	20%	20%	20%
	Selection method	Roulette Wheel	Roulette Wheel	Roulette Wheel

TABLE 3.6: Values of evaluation parameters per dataset

metrics across all the classes of the dataset. The Recall and Precision metrics take into account the FNs and the FPs of a class respectively. Thus, MFM is an aspirant metric to sustain the balance between detection rates and false alarms. Moreover, we extensively compare the performance of *Dendron* against *"rule of thumb"* algorithms under the metrics presented in Section 2.2.3. Precisely, in the case of KDDCup'99 [21], we compare our results against those presented in [24], and more specificaly against the FARCHD-OVO and C4.5 [40] algorithms.

In the experiments conducted, we used 1400, 2400 and 7000 iterations (i.e., population generations) for KDDCup'99, NSL-KDD and UNSW-NB15, respectively. One could say that the number of generations is quite high, but this is justified by the fact that MFM is a complex metric. As MFM tries to keep the balance between Recall and Precision, several populations need to be evolved in order to achieve a stable model. In the case of UNSW-NB15, the number of generations is far greater than those of the other datasets. This is because UNSW-NB15 is a more complex dataset since it contains 10 traffic classes instead of the 5 included in the others. It is also noteworthy that the mutation probability is set to 1. This means that a selected individual is always mutated as the branch expansion decision is based on the information gain. As stated in Section 3.3.5, if the split of a branch produces information gain, then it will be split. If not, the mutation operation is not applied.

In order to minimize the search space of the problem and keep the complexity under an acceptable threshold, we made use of 21 features of the KDDCup'99 [21] and NSL-KDD [22], while 23 features were used for UNSW-NB15 [23]. The number of the split intervals for the continuous features was set to 6 for KDDCup'99 and NSL-KDD (i.e., 5 splitting values), while 5 intervals were used for UNSW-NB15. By augmenting the number of the splitting points the search space increases exponentially, but it is not to be taken for granted that this action maximizes the purity at the leaves of the trees. In addition, the standard deviation σ of the Gaussian distribution was set equal to 3 in order to imitate a "top-down" branch expansion approach. Precisely, an $\sigma = 3$ generates a quite narrow distribution bell that can support the "top-down" approach, while sustaining at the same time the randomness of the GA. The λ parameter is of great importance for the final size of the population. Even though the initial DTs are single-class only, during the evolutionary process the trees become multi-classed and each one becomes a candidate for the best solution. In principle, the bigger the population, the

higher the diversity and complexity of the produced solutions. Given that Gaussian distribution is quite narrow ($\sigma = 3$), a potentially big population cannot guarantee a higher population diversity since it is almost sure that there will be similarly branched individuals. That is, $\lambda = 15$, 10 and 5 were chosen to generate populations of 75, 50 and 50 individuals for the three datasets respectively.

The results are summarized in Tables 3.7 - 3.14. More specifically, for each dataset, we provide the confusion matrix derived from the testing process of *Dendron* and a table which accumulates the performance of the selected algorithms in terms of several detection metrics.

3.4.5 Analysis

From the evaluation results we can safely argue that our approach is able to create accurate and balanced intrusion detection rules for supporting a misuse detection system. MFM is proven to be a proper fitness metric, as *Dendron* achieves high performance for each of the three investigated datasets. More specifically, *Dendron* surpasses its competitors in 3 classification metrics when it comes to KDDCup'99 and UNSW-NB15 datasets, while for NSL-KDD our approach achieves higher performance over 2 metrics. Generally, *Dendron* seems to favourably achieve high performance for MFM, AvgAcc and AttAcc metrics, while for the rest of them scores comparable results. Notably, *Dendron* outperforms representative algorithms derived from the Bayesian, SVM, Neural Networks and Decision Tree families. Among those algorithms, C4.5 is proved to be the most competitive one.

KDDCup'99 *Dendron* scores 85.77% of MFM, 89.85% of AvgAcc and 87.50% of AttAcc, while the FAR of 0.75% can be considered as a directly comparable percentage with those reported in the literature. When comparing the results against FARCHD-OVO, we can state that *Dendron* surpasses the latter under four classification metrics. Besides the superiority in the MFM, AvgAcc and AttAcc metrics, our approach achieves additionally a higher ADR of 98.24%. On the downside, *Dendron* lacks slightly when it comes to Acc and FAR. It is noteworthy that for this purpose FARCHD-OVO utilises the One-Vs-One pairwise learning scheme [62] to deal with the multi-classed data traffic. *Dendron* on the other hand achieves the same goal without such a need. Moreover,

	Normal	DoS	PRB	U2R	R2L	Recall (%)
Normal	78,459	211	162	12	205	99.25
DoS	350	48,590	175	0	0	98.93
PRB	66	208	1,640	4	0	85.51
U2R	3	3	0	20	0	76.92
R2L	86	11	3	2	797	88.65
Precision (%)	99.36	99.12	82.83	52.63	79.54	

TABLE 3.7: Confusion Matrix of testing process for KDDCup'99

Method	Acc	MFM	AvgAcc	AttAcc	ADR	FAR
Dendron	98.85	**85.77**	**89.85**	**87.50**	98.24	0.75
FARCHD-OVO	99.00	84.12	89.32	86.70	97.77	0.19
C4.5	99.59	81.81	87.79	84.79	99.29	0.20
Naive Bayes	88.55	52.95	69.29	65.31	93.61	9.87
SMO (SVM)	98.13	76.91	71.76	64.89	96.42	0.46
MultilayerPerceptron	98.90	85.59	85.13	81.50	97.74	0.20
ID3	98.50	83.42	84.22	80.47	97.66	0.49

TABLE 3.8: Metrics summary and comparison for KDDCup'99 (%)

Dendron leapfrogs C4.5 in terms of MFM, AvgAcc and AttAcc. This is reflected by an increment of approximately 4% in MFM, 2% in AvgAcc, and 3% in AttAcc. However, C4.5 performs slightly better than *Dendron* in terms of FAR and non-average metrics Acc and ADR. That is because C4.5 falls into the category of algorithms which perform better for the major classes of the dataset, while it neglects the minor classes [43]. Consequently, the high values of Acc and ADR are due to the high number of TPs for the major classes of the dataset. Opposite to this tendency, *Dendron* achieves high detection accuracy for all the classes of the dataset and this is reflected by the increased values of the average metrics of MFM, AvgAcc and AttAcc.

NSL-KDD When it comes to NSL-KDD, *Dendron* outperforms all algorithms under the AvgAcc and AttAcc metrics, while for the rest of the metrics C4.5 outperforms all the compared algorithms (see Table 3.10). NSL-KDD proves its increased difficulty level as all algorithms, apart from SMO, score lower ADR and higher FAR compared to KDDCup'99. However, *Dendron* ranked second for the Acc, MFM and ADR metrics. Additionally, as it can be observed from Table 3.9, the recall performance for the minor classes of the dataset, namely U2R and R2L, remains as high (76.92% and 83.75%, respectively) as it was also the case for KDDCup'99. This fact, advocates the ability

	Normal	DoS	PRB	U2R	R2L	Recall (%)
Normal	59,954	95	238	15	306	98.92
DoS	827	39,650	850	0	0	95.94
PRB	180	100	10,186	6	11	97.17
U2R	6	0	0	20	0	76.92
R2L	124	7	1	12	742	83.75
Precision (%)	98.14	99.49	90.34	37.74	70.07	

TABLE 3.9: Confusion Matrix of testing process for NSL-KDD

Method	Acc	MFM	AvgAcc	AttAcc	ADR	FAR
Dendron	97.55	83.35	**90.54**	**88.44**	95.97	1.08
C4.5	99.36	86.57	88.09	85.23	99.19	0.26
Naive Bayes	68.30	44.77	61.56	62.54	80.55	29.79
SMO (SVM)	96.14	66.18	64.60	56.07	93.12	0.68
MultilayerPerceptron	97.41	65.96	66.46	58.29	95.42	0.47
ID3	96.32	69.34	66.36	58.20	93.69	0.54

TABLE 3.10: Metrics summary and comparison for NSL-KDD (%)

of *Dendron* to treat fairly the minor classes of the dataset regardless of the increased difficulty level of NSL-KDD.

UNSW-NB15 As explained in Section 3.4.3, this dataset is quite different from the previous two. Since UNSW-NB15 is a more recent creation, it reflects a more contemporary and complex threat environment. The increased number of attack classes and its highly imbalanced records (Table 3.5) pose a significant challenge to every machine learning approach. As observed from Table 3.12, the performance metrics of every classifier lack significantly from those achieved using the other datasets. However, under equal terms, *Dendron* leapfrogs the other algorithms under the MFM, AvgAcc and AttAcc metrics. Once again, C4.5 is our main competitor as *Dendron* lacks slightly in Acc and FAR metrics, while the difference in ADR is approximately 4% in favour of C4.5. It can be inferred that the scarce data pertaining to the minor classes of the dataset (Worms, Backdoor, Analysis, Shellcode), affect negatively the performance of the all algorithms. Nevertheless, *Dendron* is the one achieving the highest values under the mean average metrics (MFM, AvgAcc and AttAcc), and therefore it can be asserted that our approach is able to provide balanced and accurate detection rules.

The high value of MFM indicates that *Dendron* is able to keep the balance between FPs and FNs across all the classes. This is mainly due to the more balanced values

	Normal	Backdoor	Analysis	Fuzzers	Shellcode	Recon.	Exploits	DoS	Worms	Generic	Recall (%)
Normal	29,982	1	0	313	63	125	257	43	1	1	97.39
Backdoor	25	206	0	7	0	9	28	29	1	1	67.32
Analysis	32	149	82	7	0	0	7	124	0	0	20.45
Fuzzers	999	247	5	2,804	31	66	62	135	0	4	64.42
Shellcode	23	0	0	44	123	116	30	2	0	0	36.39
Reconnaissance	148	63	0	280	261	1,121	375	167	16	2	46.07
Exploits	361	255	26	233	127	225	5,220	303	66	33	76.22
DoS	270	242	6	57	72	66	599	221	4	9	14.29
Worms	0	0	0	2	4	2	8	0	4	2	18.18
Generic	109	2	1	46	83	34	263	67	8	2,678	81.37
Precision (%)	93.84	17.68	68.33	73.93	16.10	63.55	76.22	20.26	4.00	98.10	

TABLE 3.11: Confusion Matrix of testing process for UNSW-NB15

Method	Acc	MFM	AvgAcc	AttAcc	ADR	FAR
Dendron	84.33	**48.81**	**52.21**	**47.19**	63.76	2.61
C4.5	85.15	48.79	49.33	44.14	67.88	2.54
Naive Bayes	48.45	25.55	39.20	37.82	43.42	42.48
SMO (SVM)	74.47	28.41	26.63	18.52	34.77	0.25
MultilayerPerceptron	73.89	26.61	27.91	20.57	42.25	4.56
ID3	76.31	30.49	29.19	21.36	40.46	0.27

TABLE 3.12: Metrics summary and comparison for UNSW-NB15 (%)

between Recall and Precision in our model and the ability of our algorithm to treat all the classes of the dataset under equal terms. This is advocated by the high Recall and Precision, and of course the TPs achieved for the minor classes, while only a small fraction of intrusive instances go unreported. The higher classification metrics indicate that the proposed method can be considered as a robust solution for detecting intrusive incidents, while at the same time enables the distinguishability among the several attack classes in a more accurate way.

3.4.6 Complexity Analysis of Dendron

The computational complexity of *Dendron* is the combined complexity of its components. In this direction, we use the Big-O notation in order to give a perception of our algorithm's behaviour upon the size of input parameters, as it is independent from the implementation's and enviroment's details. Since *Dendron* emerges DTs through an evolutionary process, its complexity derives from the constituent tasks of the GA and the cost of evaluating the individuals. In the context of our work, where individuals are actually DTs, the evaluation cost is that of assessing the classification ability of a DT for a given dataset. The cost of classifying one instance using a DT of height $|\mathcal{F}|$ is in the worst case $O(|\mathcal{F}|)$ [226]. That is, evaluating a dataset of size $|\mathcal{D}|$ takes $O(|\mathcal{D}||\mathcal{F}|)$. In our

analysis, we focus on the evolutionary process that takes place between lines 6 to 22 in Algorithm 1. Also, we do not consider one-time cost functions and pre-processing steps, since their complexity is always overshadowed by the more complex tasks according to the Big-O notation.

More specifically, the combined complexity derives for the following tasks: (i) Computation of selection probabilities, (ii) Parent selection, (iii) Crossover, (iv) Mutate, (v) Replace, and (vi) Evaluate. Table 3.13 gives a summary of the complexity of each process.

Function 5 is the heart of the methodology discussed in Section 3.3.2 and invokes functions 2, 3, 4 sequentially for each DT in the population. The cost of function 2, is $O(1)$ since the fitness of each individual is calculated once when the whole population is evaluated (alg:1, line:18) and it is stored as an individual's property. Function 3 needs $O(|\mathcal{C}|)$ to enumerate the number of leaves of each class in a DT and return the selection probability. Finally, function 4 needs $O(|\mathcal{C}|)$ to compute the selection probability based on the missing classes of the best individual. Since the aforementioned functions are invoked $|\mathcal{P}|$ times, the total complexity is $O(|\mathcal{P}||\mathcal{C}|))$.

The complexity of the parent selection process (func:6) is that of the Roulette Wheel Selection [219] technique which is $O(|\mathcal{P}|)$. The crossover process (func:8) incorporates three functions of constant execution time, resulting in complexity of $O(1)$. The same applies also for the case of depth selection function where a Gaussian random number generator is triggered.

The mutation process (func:9) includes several steps for defining the best splitting point of a DT node. For each available splitting point $|\varphi|$ of a selected node, the mutation process needs to calculate the Information Gain produced per splitting decision. To do so, it is necessary first to locate the subset of instances that satisfy the nodes' conditions, that is, the path which leads to the selected node. Therefore, the worst case implies a splitting decision to be taken at the very last node of the DT (i.e. $|\mathcal{F}|$) and the corresponding subset of instances that reach that node to be the whole dataset (i.e. $|\mathcal{D}|$). Hence, this subset allocation has a complexity of $O(|\mathcal{F}||\mathcal{D}|)$. Then, the entropy of this subset is computed with complexity of $O(|\mathcal{D}| + |\mathcal{C}|)$. This subset induction and its entropy calculation are needed for calculating the Information Gain of each of the $|\varphi|$ splitting points. For one splitting decision, it is required to (i) identify which of

the instances of the subset falls into each of the two newly derived branches ($O(\lceil|\mathcal{D}|)$, and (ii) calculate their entropy ($O(2(\lceil|\mathcal{D}|+|\mathcal{C}|))$. The combined complexity of these two tasks is $O(\lceil|\mathcal{D}|+2(\lceil|\mathcal{D}|+|\mathcal{C}|))$ which is reduced to $O(\lceil|\mathcal{D}|+|\mathcal{C}|)$. Eventually, the splitting procedure itself takes place for creating the child nodes below the newly derived branch. Since we assumed that the worst case is to have a splitting decision at the last node of the DT, the complexity is that of just adding the leaf, which is $O(1)$. To sum up, the total complexity of mutating two DTs at a time at a branch of $|\varphi|$ possible splitting points is: $2\Big[O(\lceil|\mathcal{F}||\mathcal{D}|)+O(\lceil|\mathcal{D}|+|\mathcal{C}|)+|\varphi|\left[O(\lceil|\mathcal{D}|+|\mathcal{C}|)\right]+O(1)\Big]$, which is reduced to $O(\lceil|\mathcal{F}||\mathcal{D}|+|\varphi|(\lceil|\mathcal{D}|+|\mathcal{C}|))$.

After creating a new population, the replacement method (func:10) occurs to choose among the individuals that will move on to the next one. At first, in order to synthesize the next population $\mathcal{P}''$ a *for* loop of $O(\lceil|\mathcal{P}|)$ takes place by choosing individuals from the previous and temporary populations. Since the next population needs to be complete, we need to check for potential absence of a class. This is done in $O(\mathcal{P})$ as the population is examined sequentially. In the worst case, the population will be missing $\mathcal{C}-1$ classes, leading the function to seek for appropriate DTs in the previous population, add them θ times in the next population, and check again for the population's completeness. That is, the summed up complexity is $O(\lceil|\mathcal{P}|)+O(\lceil|\mathcal{P}|)+|\mathcal{C}-1|\left[O(\lceil|\mathcal{P}|+\theta+\left|\mathcal{P}''\right|)\right]$. This can be reduced to $O(\lceil|\mathcal{C}-1|\left|\mathcal{P}''\right|)$, since $\theta<|\mathcal{P}|\leq\left|\mathcal{P}''\right|$, and θ is expressed as a percentage of $|\mathcal{P}|$.

The last step before the evolutionary process proceeds to a new generation is the evaluation of the whole population (func: 11) and the allocation of the best individual $\widehat{\mathcal{I}}$. The former entails a total complexity of $O(\lceil|\mathcal{P}||\mathcal{F}||\mathcal{D}|)$, while the latter costs $O(1)$.

All in all, the overall complexity is derived by taking into account (i) all the above mentioned procedures, and (ii) the number of generations $\mathcal{N}$, and the *while* loop which is executed for $|\mathcal{P}|/2$. This results to a total complexity of $\mathcal{N}\Big[O(\lceil|\mathcal{P}||\mathcal{C}|)+\frac{|\mathcal{P}|}{2}\Big[O(\lceil|\mathcal{P}|)+O(1)+O(1)+O(\lceil|\mathcal{F}||\mathcal{D}|+|\varphi|(\lceil|\mathcal{D}|+|\mathcal{C}|))\Big]+O(\lceil|\mathcal{C}-1|\left|\mathcal{P}''\right|)+O(\lceil|\mathcal{P}||\mathcal{F}||\mathcal{D}|)+O(1)\Big]$, which is reduced to $O(\mathcal{N}\Big[\left|\mathcal{P}^2\right|+|\mathcal{P}||\mathcal{D}|(\lceil|\mathcal{F}|+|\varphi|)+|\mathcal{C}-1|\left|\mathcal{P}''\right|\Big])$.

As it was expected, the overall complexity of the algorithm is affected by the size of the population, the size of the dataset, the height of the decision trees, and finally the number of splitting points of the features. The only factor that has a squared complexity

Process	Complexity
Computation of selection probability	$O(\lvert\mathcal{P}\rvert\lvert\mathcal{C}\rvert))$
Parent selection	$O(\lvert\mathcal{P}\rvert)$
Depth selection	$O(1)$
Crossover	$O(1)$
Mutate	$O(\lvert\mathcal{F}\rvert\lvert\mathcal{D}\rvert + \lvert\varphi\rvert(\lvert\mathcal{D}\rvert + \lvert\mathcal{C}\rvert))$
Replace	$O(\lvert\mathcal{C}-1\rvert\lvert\mathcal{P}''\rvert)$
Evaluate	$O(\lvert\mathcal{P}\rvert\lvert\mathcal{F}\rvert\lvert\mathcal{D}\rvert)$
Allocate best individual	$O(1)$
Total	$O(\mathcal{N}\left[\lvert\mathcal{P}^2\rvert + \lvert\mathcal{P}\rvert\lvert\mathcal{D}\rvert(\lvert\mathcal{F}\rvert + \lvert\varphi\rvert) + \lvert\mathcal{C}-1\rvert\lvert\mathcal{P}''\rvert\right])$

TABLE 3.13: Summary of complexity

is the size of the population. However, this fact cannot affect the overall performance to a great extend since the size is a parameter usually bounded in a low range.

3.5 Discussion

The methodology described and evaluated in the previous sections combines the benefits of DTs and GAs to produce linguistically interpretable and accurate detection rules, able to infer on multiple attack types. This approach makes the output, that is, the detection rules, human comprehensive, alleviating in this way the burden of the system administrators in understanding the network traffic and the attacks themselves. Moreover, the use of heuristics in the evolutionary process enables *Dendron* to deal with the challenges posed by the nature of the network traffic data. More specifically, the selection probability function in combination with the α, β and γ weights, balances the trend of machine learning algorithms to be biased toward the major classes of attacks contained in the dataset. This phenomenon is also partially addressed by the average classification metric of MFM used as the fitness function. By selecting to maximize this average metric, *Dendron* evolves the DTs with the aim of increasing the detection ratio considering all the attack classes in the dataset. This is confirmed by the provided results (Table 3.14) under the evaluation of *Dendron* using three intrusion detection datasets. Our proposal, outperformed its competitors over the average metrics MFM, AvgAcc and AttAcc, while it achieved also high ADR and kept the FAR under an acceptable threshold. However, the reader would notice that *Dendron* falls slightly short in the accuracy metric, especially when compared with C4.5. We argue that such a

Dataset	Acc	MFM	AvgAcc	AttAcc	ADR	FAR
KDDCup'99	98.85	85.77	89.85	87.50	98.24	0.75
NSL-KDD	97.55	83.35	90.54	88.44	95.97	1.08
UNSW-NB15	84.33	48.81	52.21	47.19	63.76	2.61

TABLE 3.14: Dendron metrics summary for all datasets (%)

behavior is normal, because as already pointed out, Dendron focuses on treating all the classes of the dataset under equal terms. This means that for algorithms like Dendron the mean average metrics become significantly more important than the accuracy one. Putting it another way, a high accuracy in the context of multi-classification problems with imbalanced datasets derives from an algorithm's tendency to decide in favor of the major classes of the dataset, while neglecting the minor ones.

Moreover, the use of DTs produces a straightforward output, that is decision rules which could be used in real-life IDS. In practice, this feature can be proved a strong advantage as the other solutions reviewed in Section 3.6 are based on multilayered approaches or Fuzzy association rule-based methods requiring additional data transformations or pre-processing steps to infer on detection rules. On the bright side, *Dendron* can be implemented in a straightforward way on the provided dataset. Furthermore, in our experiments, we used 10% of the dataset to train our model and the rest 90% for the evaluation. Given the small size of the training data, we argue that our concept of evolving several learning models, namely DTs, in a population multiplies the possibilities to construct an end-model that fits better on the data, even if those are scarce.

Conversely, the evolution of several individuals increases the complexity of *Dendron*. That is, the optimal balance between the size of the population and the produced results must be determined given an acceptable complexity level. In any case, the termination condition could be adjusted to a desirable outcome. As already pointed out, GAs explore the search space of a problem randomly and this renders our solution non deterministic in the sense that for the same input data the model will not create the exact same set of detection rules. Even though the DTs are generated randomly, the outcome of *Dendron* is a DT representing a static set of detection rules, which are as accurate as the tree's performance metrics. In any case (and this applies to every method), the search space of a network traffic analysis problem can be huge and this indicates that several

experiments should be conducted over the same input to finally select the more accurate set of detection rules.

3.6 Related work

Network intrusion detection utilizing metaheuristic optimization algorithms is a field that constantly attracts the research community's attention, thus the plurality of the solutions that have been proposed is rather high [67]. Therefore, in the context of this section, we only consider related work that falls under the umbrella of misuse detection, utilizes DTs or GA, and additionally focuses on detecting intrusive events of minority classes. We classify the documented works in three major categories based on the methodology they utilise.

3.6.1 On Decision Trees

The work in [227] introduces two hybrid classifiers, namely hybrid decision tree (DT) and hybrid naïve Bayes (NB). The first one implements a naïve classifier to remove troublesome instances in the dataset which could overfit the classifier. After removing those instances, a DT algorithm driven by information gain is used to classify the free-of-noise dataset. On the other hand, the hybrid NB classifier is used to find an optimal subset of attributes of the dataset. The authors report average sensitivity (recall) of 81.9% and 82.6% of precision for the hybrid DT classifier and 82.3% and 83.6% for the Hybrid NB respectively.

The authors in [44] propose an IDS based on Neurotrees and GAs for feature reduction. Precisely, chromosomes are formed as binary strings {0, 1} of the length of 41 features of KDDCup'99 dataset. The fitness function of the GA is based on the Sensitivity and Specificity metrics of the neurotree. The authors conclude to 16 features to support their classifier but they miss to successfully classify 6 minority attack classes out of 23.

In [228] a two-layer IDS is proposed. At the first layer, a coarse-grained IDS is applied for detecting attacks based on 5 features, whereas at the second layer a fine-grained IDS is used based on 20 features having the highest information gain. The authors suggest the use of Very Fast Decision Tree (VFDT) to improve the performance of their system.

These trees are gradually expanded in a top-down approach by replacing leaves with decision nodes. The proposed system was not able to detect U2R and R2L attacks but the use of VFDT improved the training time.

3.6.2 On Genetic Algorithms

The work in [229] proposes a combination of neuro-fuzzy classifiers and GAs. Specifically, the ANFIS classifier is used for generating fuzzy rules per traffic class, while a Mamdani fuzzy inference system with defuzzification strategy infers upon intrusive and normal events. Then, a GA is used to optimize the solution by having as genes parameters of Membership Functions used in the fuzzy decision engine. Even though the proposed algorithm is able to infer accurately on normal, probing, and DoS traffic, the detection rate for the minority classes is rather low.

The authors in [230] apply a Pareto-based evolutionary algorithm within the detection engine of Snort IDS with the aim to optimize the rule generation process. The GA evolves a population of individuals where each one is composed of a certain number of signatures (rules). They evaluate their proposal under two optimization modes. First, a single objective weighted function is used to minimize the FPs and FNs, while in the second mode, a Pareto-optimization approach is used to optimize two different fitness functions. The authors assess their proposal under different inputs arriving to the conclusion that the higher the population size, the better the quality of the solutions.

In [231] a combination of a SVM model and kernel principal component analysis (KPCA) with GAs is proposed. The SVM is used as a multilayered classifier to infer on attacks, while KPCA is used to select a subset of features. GAs serve as an optimization scheme for the system's parameters. More specifically, the authors formed 5 different training and testing sub-datasets by using sampling methods. Overall, even if the authors apply several normalization and feature reduction techniques to the dataset, the results for detecting rare types of attacks are unsatisfactory.

The authors in [24] propose a misuse detection approach based on genetic Fuzzy Systems and pairwise learning. They take advantage of the One-Vs-One technique to build binary classifiers. During classification, an instance is submitted to all the classifiers and the prediction level is a combination of the overall classification. Although their approach is

quite accurate, it is based on significantly complex classification techniques, hence the generated results may not be straightforwardly exploited without utilizing de-fuzzing steps. Their solution utilizes GAs in the context of fuzzy systems to form a classifier considering only a subset of the search space. Even though the theoretical background of this work significantly differs from ours, as explained in Section 3.3, we use it in order to compare our system as being a recent work that achieves remarkably high detection rates.

3.6.3 On other machine learning techniques

The study in [232] proposes a layered approach for intrusion detection. The authors make use of Conditional Random Fields (CRFs) aiming to distinguish better the overlapping features in a dataset. One classification model is created for each of the 4 major attack classes of the KDDCup'99 dataset and trained individually. Therefore, each layer does not contain knowledge of other layers. For each layer, the authors conduct feature reduction and try to identify relations among features to improve accuracy. They achieved 29.6% and 86.3% detection accuracy for R2L and U2R attacks, respectively.

The authors in [64] propose a hierarchical SVM-based IDS. They utilize BIRCH, a clustering algorithm, to pre-process KDDCup'99 dataset in order to minimize the training instances before they are fed to a SVM classifier. BIRCH produces 5 datasets which correspond to the major traffic classes of the dataset. These datasets are used to train 4 SVM classifiers separately which, finally, are combined to form the eventual model. Also, the authors apply a simple strategy to identify the useful features per traffic class. The proposed model performs well for DoS and Probe attacks, but it achieves low detection rate for the minority classes.

The work in [32] capitilizes on a combination of Artificial Neural Networks (ANNs) and fuzzy clustering. More specifically, the proposed model is divided into three steps: i) Fuzzy clustering for generating training subsets, ii) Training different ANNs, and iii) Aggregation and combination of the results of the different ANNs. The authors report a 58.57% and 76.92% recall for R2L and U2L attacks, respectively.

In the context of big data analytics, the authors in [233] designed a defense mechanism able to receive security analysts' feedback. The proposed system tries to address three

main challenges: i) Lack of labeled data, ii) Constantly evolving attacks, and iii) Limited investigative time and budget. The authors utilize a system for transforming raw traffic data into labeled data combined with both supervised and unsupervised machine learning techniques. Supervised solutions like the one we propose could be applied under this concept. Also, the work in [233] focuses on outlier detection for rare incidents. According to the authors' evaluation, the proposed system is able to act against unseen attacks, while the time interval between attack detection and prevention is reduced. Their system produces threefold detection ratio, while minimizes the FPs by 5 times.

3.6.4 Discussion

Based on the literature review given above, it can be argued that the vast majority of the works use GAs with the aim of coming up with a subset of features which could increase the classification accuracy. Opposite to that observation, in our solution, we take advantage of GAs for evolving DTs with the purpose of exploring the search space of the problem. Also, to the best of our knownlegde, we are the first to use this concept in the context of IDS. So far, throughout the literature, only the work presented in [230] shares a common ground with our methodology. More specifically, the authors use Snort rules as individuals of string representations, while we use DTs, which are expanded through the crossover and mutation operations. Another noticeable difference with [230] is the selected fitness functions that guide the solution. That is, we utilize metrics which apply in the context of multi-classifiaction problems, as shown in Section 2.2.3, whereas [230] employs a Pareto-optimization approach in the fitness function.

Another observation is that nearly all the works use KDDCup'99 [21] dataset as the basic benchmark for assessing the proposed systems under a limited number of metrics. Contrary to this, our work employs an extensive testbed, which engages two more challenging and far more recent datasets (NSL-KDD[22] and UNSW-NB15[23]), while the performance of *Dendron* is evaluated using a plethora of metrics.

In addition, the great mass of the proposed solutions do not meet the quality of rules interpretability, as they apply several data transformations to serve the requirements of the selected classifiers. It is also worth of noting that in an effort to deal with the imbalanced network datasets, several works use multilayered approaches for detecting attack types individually at each layer or by combining several trained models in a

merging phase. On the bright side, our solution is single-layered and produces one end-model (a DT) that represents the whole set of the generated rules. Finally, one could say that the problem of detecting rare intrusive events remains largely an open issue. Namely, some solutions are not able to detect this kind of incidents while others do, but still the detection accuracy they present is not adequate. To cope with this shortcoming, the work at hand has put considerable effort into fine-tuning the algorithm and developing accurate individuals that are able to equally identify both common and rare intrusive incidents.

3.7 Conclusions and future work

In this chapter, we presented *Dendron*, a novel methodology for evolving DT classifiers using GA with the aim of generating detection rules in the context of misuse detection systems. Our proposal delivers linguistically interpretable rules in an effort to increase the benefit of security administrators and to ease their tasks. To this end, our methodology deals with the several challenges posed by the nature of the network traffic and balances the trend of machine learning algorithms to largely neglect the minority attack classes. The weighted selection probability function in the evolutionary process in combination with the MFM fitness metric lead *Dendron* to outperform legacy and state-of-the-art solutions in the field. *Dendron* achieves increased attack detection accuracy, while keeping the balance of Recall and Precision metrics among all the classes of the dataset, even for the minor ones. The proposed scheme was evaluated using three intrusion detection datasets, namely KDDCup'99 [21], NSL-KDD [22] and UNSW-NB15 [23] and scored superior results for the average metrics MFM, AvgAcc, and AttAcc. Especially for the minor classes, that pose a significant challenge to intrusion detection solutions, *Dendron* reached high detection accuracy with respect to their size.

Chapter 4

Introducing Deep Learning Self-Adaptive Misuse Network Intrusion Detection Systems

In chapter 3 we presented *Dendron*, a misuse detection system based on an evolutionary methodology that enables effective detection rules induction given a dataset. Even though such a mechanism can be of major importance in the process of updating the detection engine of an IDS, still, its adaptation remains a highly demanding engineering task for the administrators. Machine learning models need to be re-evaluated periodically. That is, an ML-based IDS is not "set it and forget it", as models are only as good as the data they analyze. Robust protection requires frequent, rigorous re-training of the model by providing data with high fidelity to the real world [234].

In this direction, this chapter details on a self-adaptive methodology, which introduces intelligence in the retraining process of a misuse IDS with the aim of alleviating the IDS administrators from adapting the detection engine upon the occurrence of an efficiency drop. In fact, the presented methodology, as pointed out in Section 1.1, aims to address the main limitation of misuse IDS, which is the lack of agility in adapting to new and "unknown" network states.

4.1 Introduction

Undoubtedly, as has been widely reported in the literature, misuse IDSs lack the ability of identifying new attack patterns or deviations from known ones, and their performance depends on the freshness of the signatures database. Hence, the IDSs administrator needs to put significant effort to keep the misuse detection model up to date. If we additionally consider the fact that the protected environment is a dynamic ecosystem where new devices and/or services may appear or leave the network at any moment (e.g., the Internet of Things (IoT)), it becomes clear that the adaptability issue becomes a burden on administrators' shoulders. This burden becomes even heavier as the growth of communication networks pushes IDSs into the big data era, where the increased volume of the transmitted data surpasses the limits of human processing capabilities.

In view of the above, adaptive IDSs are becoming an active research field as new researches [235, 236] aim to address the inherent limitations of legacy intrusion detection systems. So far, interesting artificial intelligence-based methods that bear the feature of adaptability have been reported as promising approaches. To name a few, Learning Classifier Systems (LCS) [237], Artificial Immune Systems [238] and Swarm Intelligence [67] combine adaptation and evolution aspects. However, this research topic has still many challenges to face as systems and attacking tactics become more sophisticated.

Keeping any type of IDS up-to-date is a demanding task for several reasons where most of them pertain to the *environmental changes*. The latter term refers to any aspect of a network that can change and consequently affect the profile of the generated network traffic. In practice, the addition (or disengagement) of a device in a network can affect different network aspects, including the topology, the running services, the open ports, the communication protocols and/or applications, the network traffic load, and others. In turn, these environmental changes affect fundamental security features such as the vulnerabilities appearing in the network, which can generate multiple penetration paths for the attackers. Considering a more dynamic network like an IoT environment, an Ad Hoc network, or even a corporate network with a Bring your own device (BYOD) policy applied, one can understand that the attack surface of the network can be increased unexpectedly. It is plausible that, say, the newly introduced device might be already infected by a malware and act as a stepping stone for an attacker to conquer more assets within the network. Yet, new devices are not the only enemies of an IDS in a network, as also already installed devices will eventually proceed with software/OS updates or new software installations that again will bring in alterations in the environment.

Overall, the above mentioned changes are routine actions that constantly appear in every common network, rather than unusual events. Actually, virtually all sort of modifications

can significantly affect the performance of an IDS which is placed to protect an ever-changing infrastructure. This reality, combined with the lack of adaptable detection engines, forces the IDS to become quickly outdated and inadequate as it inevitably has to operate in new and "unknown" or unforeseen environments for which its engine was not trained. Thus, security administrators undertake the task of constantly retraining the IDS by considering all the new environmental changes to regain the reliability and the performance of the detection system. All in all, the cardinal challenge for any IDS designer, namely find proper ways to automatize at least to a certain degree the retrain process, remains largely unsolved.

To this end, this chapter presents a novel adaptive methodology which can significantly boost the performance of a misuse IDS when it is dragged into new, previously unseen environmental states. This novel solution brings in intelligence in the detection engine update process with the aim of extending its lifetime and sustain the detection ratio above considerably higher levels than it would reach without such intelligence. At least, in network setup transition periods, this ability gives the necessary time to the administrators to smoothly retrofit the IDS to fully meet the new environmental conditions. To do so, we take advantage of the benefits of the Self-Taught Learning (STL) methodology [239], for enabling Transfer Learning from unlabeled data for the sake of assisting the IDS when dealing with unknown environments. Our evaluation proves that the qualities of the STL methodology can fit well in the particular problem and address the challenges raised in the field of adaptive IDSs. Our adaptive methodology is also supported by the MAPE-K model [240] for delivering a self-adaptive IDS that follows the sound practices of autonomic computing.

In short, the contributions of the work presented in this chapter are as follows:

- We propose a novel methodology for designing a scalable, self-adaptive and autonomous misuse intrusion detection systems based on advanced artificial intelligence (AI) techniques.
- We take advantage of deep learning methodologies to identify new data feature representations that stem from the unknown environment where the IDS operates. These new representations are used to retrain the IDS in an automated way so as to adapt to the new environment.
- We integrate our proposal in the context of MAPE-K methodology that draws the frame for autonomous and self-adaptive systems.
- We extensively evaluate our system over several metrics and diverse environmental states to deliver a proof of concept, which is supported by experimental results and demonstrates its potentiality for further extension.

The rest of this chapter is organized as follows. The next section includes all the necessary information to introduce the reader in our methodology. In Section 4.3 we present our methodology and elaborate on its beneficial characteristics, while in Section 4.4 we provide the evaluation results. Section 4.5 provides a discussion on the key findings. Section 4.6 reviews the related works in the field. The last achieved section concludes and provides pointers to future research.

4.2 Preliminaries

Our work recruits two different concepts to provide a holistic framework for self-adaptive and autonomous misuse detection systems. Before introducing the reader to our idea, we first provide an overview of the basic concepts related to our methodology. That is, the following subsections elaborate on the MAPE-K [240] and Self-Taught Learning [239] methodologies.

4.2.1 MAPE-K Control Loop

MAPE-K control loop is a reference model in autonomic computing firstly introduced by IBM [240]. Since then, MAPE-K is used to set the principles for self-adaptive systems and significant effort has been put to standardize and formalize the methodology [241, 242]. MAPE-K is a control loop comprised of five activities, namely, *Monitor, Analyze, Plan, and Execute* over a *Knowledge* base, as can be seen in Figure 4.1. As further explained in subsection 4.3.2, these activities provide a general framework for developing self-adaptive systems which are able to sense changing events from their environment. This is done via the use of specialized *Sensors*, and eventually harmonize their behavior by taking actions through *Actuators*.

More specifically, the *Monitor* senses the environment and collects data/events of interest where their presence indicates the need of system adaptation. The collected data/events are gathered in the *Knowledge* database for later reference. Next, the *Analyzer* undertakes the task of processing the collected events to identify patterns of failure or critical events and act upon by initiating a proper adaptation strategy. The *Plan* activity orchestrates the decision making and determines the changes which should be taken for keeping the system aligned with its objectives. Finally, the *Executor* instructs the appropriate alterations to the system through the *Actuators*. The control loop described above is initiated whenever the system identifies the need to adapt its behavior to the underlying environment and always aims to meet the objectives which - in principle - are set by the administrator.

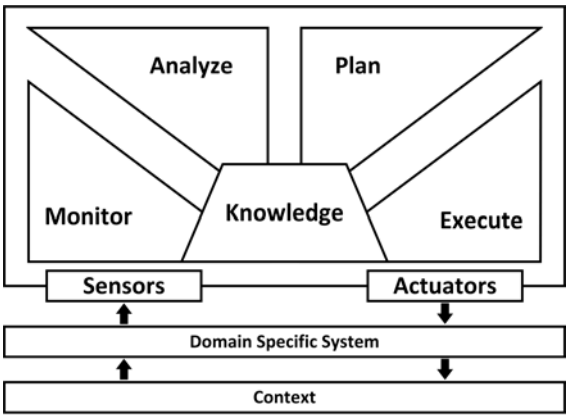


FIGURE 4.1: MAPE-K methodology sets the principles for an adaptive and autonomous IDS.

Every single activity in the MAPE-K control loop can contain other autonomic elements that can be used to fulfill sub-objectives of the main activities. These elements can interact among each other by exchanging signals and messages. Overall, the system and its sub-components are coordinated with the aim of providing a fault tolerant self-adaptive system, which is driven by pre-determined objectives.

4.2.2 Self-Taught Learning

Self-Taught Learning (STL) is a machine learning framework which is able to exploit unlabeled data with the purpose of improving a supervised classification problem [239]. The motivation of the authors in [239] derives from the fact that labeled data are an "expensive" source of information, as it requires a significant investigation budget to acquire and update them. The question posed was whether unlabeled data could be used to improve a given classification task.

In the STL concept, one is provided with both labeled and unlabeled data. The labeled data are used as the initial training set of m samples for a given classification task $T = \{(x_l^{(1)}, y^{(1)}), (x_l^{(2)}, y^{(2)}), ..., (x_l^{(m)}, y^{(m)})\}$, where $x_l^{(i)} \in \mathbb{R}^n$ is the i-th sample with n features, $y^{(i)} \in \{1, ..., C\}$ is the class label, and the l symbol stands for "labeled". On the other hand, the set of k unlabeled samples $U = \{x_u^{(1)}, x_u^{(2)}, ...x_u^{(k)}\} \in \mathbb{R}^n$, where $x_u^{(i)} \in \mathbb{R}^n$ is the i-th unlabeled sample with n features, and u stands for "unlabeled". U is given as input to a sparse coding algorithm to learn a higher level structure of those data. This structure is then used as a basis to transform the initial labeled dataset T and obtain a new training set $\hat{T} = \{(a_l^{(1)}, y^{(1)}), (a_l^{(2)}, y^{(2)}), ..., (a_l^{(m)}, y^{(m)})\}$, where $a_l^{(i)}$ represents the i-th new training example. In consequence, the new training dataset $\hat{T}$ can be used to train a supervised learning algorithm.

More specifically, sparse coding is a type of unsupervised methods that aims to reconstruct input data as accurately as possible and express them as linear combinations of a basis vector b. The basis vector b enables to accurately capture the inherent information of the input and identify strong patterns in it. Additionally, sparse coding regulates the sparsity of the data by using coefficients (or activations) a_i and encourages most of the coefficients to be zero. In fact, this is an optimization problem that aims to reconstruct the input data by minimizing the reconstruction error, and at the same time to maximize the sparsity of the output. Given the bases b and the training set T, the STL algorithm transforms the inputs $x_l^{(i)}$ to sparse non-linear combinations of the basis b to form a new, but more informative, training set $\hat{T}$.

The interested reader can refer to [239] for more details regarding the STL method. In the following section, we further elaborate on the beneficial features of the STL and explain how these features in conjunction with MAPE-K control loop can deliver a holistic methodology for building a misuse adaptive IDS.

4.3 Proposed Methodology

This section details on our methodology for adaptive misuse IDSs. With reference to Section 4.1, we first define the term "environmental state" and elaborate on the challenges that IDSs face whenever a change appears in the underlying network topology. Next, we combine the two methodologies described in subsections 4.2.1 and 4.2.2 for the purpose of presenting our full-fledged approach along with its advantages.

4.3.1 Environmental States and Network flows

Undoubtedly, computer networks are highly volatile environments, which can be characterized as a mosaic of diverse interconnected devices usually from different vendors. On top of that, one needs to consider that modern networks like IoT networks, Wireless Sensor Networks (WSN), VANETs, and others come with such an extended size, dynamics and complexity that far exceed the limits of human managing capabilities. Similarly, IDSs which aim to protect such dynamic networks are unable to automatically adapt to the changes occurring to their environment and their adjustment requires significant human effort. In fact, there are several reasons that can lead a network to a new state, and therefore lower the efficiency of the deployed IDS.

At any given time, a new device can join a network. This is the most common situation, especially in wireless networks, but this event alone can lead to a set of subsequent events

that can bring instability. That is, the newcomer may be a host of new services, ports, applications, communication protocols, communication patterns, network workload, and even new vulnerabilities. In the worst case scenario, the new device may be also infected by some ilk of malware that can attempt to exploit other network assets to penetrate the network. A join operation is not the only reason for facing network state changes. The already existing network assets can modify their operational profile as they can be subjected to OS/application upgrades and new application installations. These modifications, apart from introducing changes in the network state, can sometimes increase the attack surface of the network as they can bring in new and zero-day vulnerabilities. The aforementioned changes, even when occurring individually, can significantly affect the decision engine of an IDS and produce a high level of false alarms. Even worse, if the IDS is not retrained to deal with these changes it can become the single point of failure of the infrastructure's security planning.

In the context of our work, we perceive any of the above mentioned events to lead the network into a new state, and thus affect the IDS's operational environment. Such changes also affect the network's behavioral profile, which in turn is reflected in the network flows. According to RFC 2722 [243], a network flow can be seen as an artificial logical equivalent to a call or connection, which has as attribute values aggregated quantities which reflect the events that take place during this connection. These attribute values can bear valuable information regarding numerous aspects of the network's behavior ranging from the topology to the workload and the active services. Thus, network flows are a rich source of information that can improve the network security visibility as they can be leveraged by security analysts to identify and assess hostile actions, new attacks, and the network's security state in general. As a result, when a network is overwhelmed by unknown and previously unseen network flows, an IDS which has been trained to defend a network based on a static training set needs to be retrained in order to sustain a credible security level. This however implies the need of a demanding process on behalf of the security analyst to identify and label manually new network instances for creating a new dataset that can be used to retrain the IDS. Considering that most of the network changes are common actions that can happen regularly, it becomes clear that there is a need for methods capable of automating the retraining process.

To this end, our methodology aims to offer an automated way to keep the detection ratio of a misuse IDS to acceptable levels regardless of the environmental changes that may indicate the presence of previously unknown attacks. In a nutshell, our methodology can empower autonomous and self-adaptive misuse IDSs by enabling them to adapt to their environment and significantly contribute in keeping a high or at least acceptable security level. This quality also significantly alleviates security experts from the demanding task of retraining the IDS.

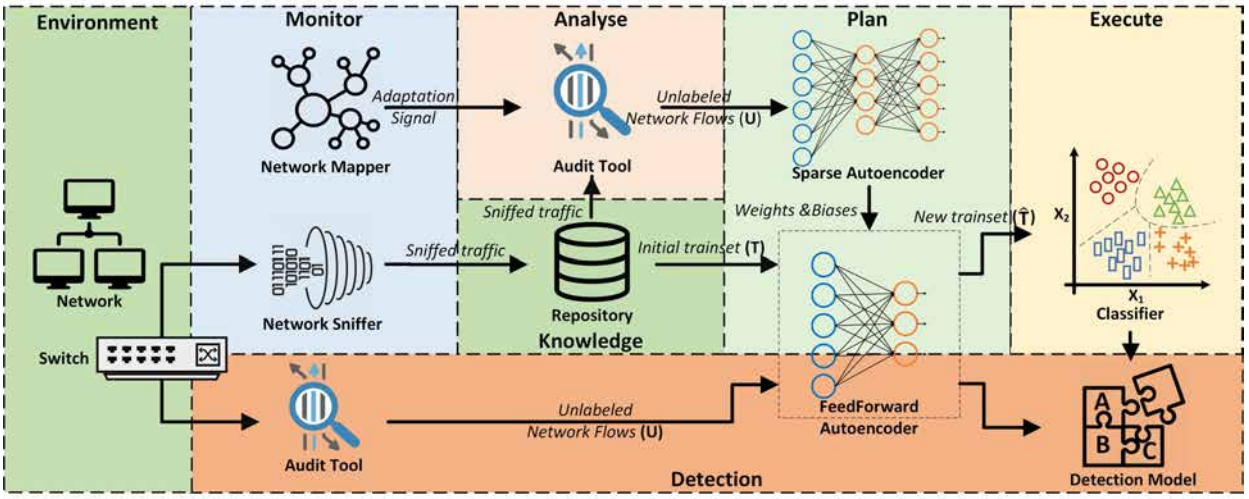


FIGURE 4.2: Architectural overview of the proposed system.

4.3.2 Blending MAPE-K and Self-Taught Learning

As described in Section 4.2, MAKE-K is a reference model to build autonomous and self-adaptive systems. This subsection details on the ways the benefits of MAPE-K and STL can co-work toward coping with the challenges of this particular field and building a solid basis for misuse adaptive IDSs.

As observed from Figure 4.1, MAPE-K comprises 5 activities that operate over a *Domain Specific System (DSS)* and a *Context*. In our case, the DSS is the IDS per se, while the Context can be adjusted to any given type of network where there is a need of an adaptive IDS. Figure 4.2 depicts the proposed system and details on its components, which are described next.

4.3.2.1 Monitor

This activity undertakes the task of coordinating the sensors for acquiring the basic knowledge that will reveal the need of triggering the adaptive control loop. Network mappers can be used as the basic sensors for network inventory. Such entities are able to determine a gamut of network characteristics, including its topology, the available hosts, the running services, open ports, the operating systems, and even potential vulnerabilities. By collecting such information, the *Monitor* is able to determine any alteration event that requires an IDS adaptation. The monitoring activity is able to grasp the environmental changes in collaboration with the Knowledge activity, which serves as a repository for reference purposes. The *Monitor* can schedule the network mapping process to occur periodically according to the characteristics of the network.

In parallel, another sensor type which is controlled by the *Monitor* is the Network Sniffers. The latter are used to capture the network traffic. This traffic is used as the basis to extract in a later stage the network flows which have to pass through the detection engine of the IDS. Additionally, the network traffic is stored in the *Knowledge* component to serve the purpose of adaptivity as it is described further down in the Plan/Execute activities.

4.3.2.2 Analyze

After collecting the necessary data, the *Analyzer* performs the transformation of the raw network traffic into network flows. By using the stored traffic of the knowledge component, the *Analyzer* utilizes network audit tools such as Argus [25] and CICFlowMeter [244] in order to generate the network flows. These tools are able to analyze large amounts of network traffic even in an in-line manner and process them accordingly to generate highly informative network flows with various features. These features comprise the machine learning features of the network traffic instances, which are fed into the IDS engine for detection purposes. Note that these flows constitute the unlabeled dataset, which on the one hand are given into the supervised model of the IDS to detect potential attacks, while on the other are used as the unlabeled data fed to the STL to fuel the adaptive process. This implies that during the IDS operation, the adaptive process is simultaneously executed with the aim of coming up with a new detection model that will replace the existing one.

4.3.2.3 Plan

The planning activity undertakes the key process of leveraging the unlabeled data for initiating the machine learning adaptive process. Until that point, the *Monitor* and the *Analyzer* identified environmental changes in the network, while the *Knowledge* component consolidated the network flows, which were generated by the time that the change(s) occurred. This moment is the beginning of a crucial time interval when the IDS may face unknown network traffic instances that can undermine its performance. In this direction, the *Plan* activity aims to cope with this ambiguity by utilizing unsupervised feature learning techniques. In the context of our work, we utilize a Sparse autoencoder as the unsupervised learning algorithm to learn new informative and sparse representations of the unlabeled data, and thus benefit the supervised task of the misuse IDS.

Sparse autoencoder An autoencoder is a neural network that applies backpropagation [245] and aims to reconstruct a given input to an output that approximately resembles to the initial input. That is, the neural network attempts to learn a function $h_{W,b}(x) \approx x$, where the W, b vectors denote the Weights and Biases among the layers and their units of the neural network. This process can be driven also by other objectives apart from minimizing solely the reconstruction error. As already pointed out, in our work we utilize a Sparse autoencoder in order to learn sparse representations of the input data. In practice, the backpropagation process is driven by the following cost function.

$$\begin{aligned} J(W,b) = & \frac{1}{k}\sum_{i=1}^{k}\left(\frac{1}{2}\|x_u^{(i)} - \hat{x}_u^{(i)}\|_2^2\right) \\ & + \frac{\lambda}{2}\sum_{L=1}^{2}\sum_{i=1}^{s_L}\sum_{j=1}^{s_{L+1}}\left(W_{ji}^{(L)}\right)^2 \\ & + \beta\sum_{j=1}^{s_2} KL(\rho\|\hat{\rho}_j) \end{aligned} \tag{4.1}$$

where:

- $x_u^{(i)} \in \mathbb{R}^n$ is the i-th input unlabeled example.
- $\hat{x}_u^{(i)} \in \mathbb{R}^n$ is the i-th output given the i-th input example.
- k is the number of the examples in the unlabeled training set.
- λ is the weight decay parameter.
- L index denotes the number of a layer.
- s_L is the number of nodes in the L-th layer.
- β is the weight of the sparsity penalty.
- $\|x_u^{(i)} - \hat{x}_u^{(i)}\|_2^2$ is the squared L^2 norm.

Through backpropagation the Sparse autoencoder aims to minimize equation (4.1). As can be seen, the equation consists of three terms (one per line). The first term represents the average accumulated squared error among the input and the reconstructed output. That is, the network tries to reconstruct the output and achieve high similarity with the input. Note that, $\hat{x}_u^{(i)} \in \mathbb{R}^n$ is a vector of n features, i.e., $\hat{x}_u^{(i)} = \{\hat{x}_{u_1}^{(i)}, \hat{x}_{u_2}^{(i)}, \ldots, \hat{x}_{u_n}^{(i)}\}$. Thus, each $\hat{x}_{u_p}^{(i)} = h_{W,b}(x_{u_p}^{(i)}) = f\big(\sum_{j=1}^{s_2} W_{pj}^{(2)} a_j^{(2)} + b_p^{(2)}\big)$, $p = 1, \ldots, n$ and $a_j^{(2)}$ are the activations of the hidden units (2nd layer). The sigmoid function $f(z) = \dfrac{1}{1+\exp(-z)}$ has been chosen as the activation function for the neurons. This activation function gives values between 0 and 1, while it regulates the weights of the network to change gradually and output better results. Additionally, the sigmoid function introduces non-linearity

into the model, thus aiding in capturing non-linear combinations of the input data. The second line refers to the weight decay term that tries to decrease the magnitude of the weights ($W_{ji}^{(L)}$) among the nodes of the layers, while λ controls the importance of the weight decay term. The last term is a function that applies the sparsity penalty, where $KL(\rho\|\hat{\rho}_j) = \rho \log \frac{\rho}{\hat{\rho}_j} + (1-\rho) \log \frac{1-\rho}{1-\hat{\rho}_j}$ is the Kullback-Leibler (KL) divergence that can determine the difference between two distributions having ρ and $\hat{\rho}_j$ mean values respectively. That is, ρ defines a desired level of sparsity, while $\hat{\rho}_j$ is the average activation of the j-th hidden unit. The magnitude of the sparsity penalty is regulated by the β weight.

Feedforward autoencoder The training process of the Sparse autoencoder will define the Weights and Biases vectors $(W, b) = (W^{(1)}, b^{(1)}, W^{(2)}, b^{(2)})$. Next, these vectors can be used in a Feedforward manner over a new input for finding a new and more informative structure of this input. In other words, the knowledge acquired from the unlabeled data U that fed into the Sparse autoencoder can now be exploited for restructuring another dataset. This reconstruction is driven by a new representation which is learned out from unlabeled data, i.e., data that stem from an unknown environment.

Following this principle, our methodology can generate a new representation of the labeled dataset $T = \{(x_l^{(1)}, y^{(1)}), (x_l^{(2)}, y^{(2)}), ..., (x_l^{(m)}, y^{(m)})\}$, that was initially used to train the IDS. Note that, $x_l^{(i)} \in \mathbb{R}^n$ is a vector of n features, i.e., $x_l^{(i)} = \{x_{l_1}^{(i)}, x_{l_2}^{(i)}, \ldots, x_{l_n}^{(i)}\}$. The new representation is a new labeled training set $\hat{T}$ that has as features the activations of the hidden units. That is, given T as the new input in a Feedforward autoencoder, we can calculate the new activation vectors using the Weights and Biases of the first layer $W^{(1)}$, $b^{(1)}$ by applying the activation function. As a result, the system produces a new dataset $\hat{T} = \{(a_l^{(1)}, y^{(1)}), (a_l^{(2)}, y^{(2)}), ..., (a_l^{(m)}, y^{(m)})\}$, where $a_l^{(i)}$ represents the i-th new training example. Thus, each $a_l^{(i)}$ example is a vector of s_2 activations, $a_l^{(i)} = \{a_{l_1}^{(i)}, a_{l_2}^{(i)}, ..., a_{l_{S_2}}^{(i)}\}$, and each activation is given as follows:

$$a_{l_p}^{(i)} = f\big(\sum_{j=1}^{s_1} W_{pj}^{(1)} x_{l_j}^{(i)} + b_p^{(1)}\big), p = 1, \ldots, s_2 \tag{4.2}$$

Finally, the new training dataset $\hat{T}$ can be used to train a supervised learning algorithm. Note that, the acquisition of $\hat{T}$ can occur virtually indefinitely as long as the planning activity is fed with new unlabeled network traffic.

The reader may notice that the Feedforward autoencoder adds an extra layer of data transformation. That is, any instance which will be subjected into the supervised learning algorithm for detection purposes needs to pass first through the Feedforward autoencoder to acquire the same transformation properties.

4.3.2.4 Execute

The outcome of the planning activity is a Feedforward autoencoder which is used for reconstructing the initial labeled dataset T and acquire $\hat{T}$. Hence, the execution activity undertakes the training of a supervised model based on the new dataset $\hat{T}$. This step does not impose any constrains regarding the supervised algorithm that can be used to empower the detection model. In our case, we make use of Softmax Regression to deliver a multi-classification detection module. After training the new model, the old one, which due to the environmental changes had started facing efficiency problems, can be replaced through the actuators.

4.3.2.5 Knowledge

During the adaptive control loop, the *Knowledge* component is accountable for storing purposes. In fact, *Knowledge* is a repository that supports the adaptive functions and helps exchanging the inputs and outputs of each activity among them. More specifically, the repository stores the sniffed network traffic during the monitoring phase. Upon the adaptation signal of the network mapper, these captures will become the input of the network audit tool for generating the network flows. Additionally, the repository holds the initial labeled dataset T, which is used as a basis every time the adaptive control loop is triggered.

4.3.3 Discussion

Figure 4.2 provides a high-level view of the proposed system by highlighting the building blocks of the MAPE-K adaptive control loop. Additionally, in the figure, we can observe the interconnections among the diverse components and follow the flow of the system's actions. As already pointed out at the beginning of this subsection, the combination of MAPE-K and STL serves as the basis for building adaptive and autonomic misuse IDSs. That is, while MAPE-K provides the essential principles to realise such an IDS, STL contributes several features that cover the missing parts of the misuse IDS adaptation puzzle.

- STL is destined to utilize unlabeled data with the aim of improving a supervised learning task. This feature fits directly in the nature of the problem. Once a misuse IDS is powered, it faces unlabeled network instances and tries to classify them. Inevitably, due to environmental changes, the statically trained IDS will face efficiency issues. Nevertheless, since STL is able to capture informative structures from unlabeled data, the ambiguity of the new environment is exploited to generate new knowledge. This, in turn, will reinforce the misuse IDS by generating a new representation of the initial dataset.

- STL cannot be seen as an unsupervised feature learning or a semi-supervised technique, but as a more powerful setup. This is because the unlabeled data x_u can: 1) be of any class and not necessarily coincide with the classes of the labeled data x_l, and 2) be drawn from a different distribution from the labeled data x_l. Crucially, these are two essential qualities that advocate the suitability of STL as in practice, in an unknown environment, an IDS will face both known and unknown attacks, which both stem from different distributions. Thus, our methodology guarantees the adaptability and autonomy of the misuse IDS.

- When put in the frame of MAPE-K, the characteristics of STL can significantly alleviate the burden of retraining the IDS every time a change appears in its environment. For the retraining process, one needs also to consider the significant effort required to assign labels by hand to Big Data such as network data. That is, if the IDS is trained with a basic labeled dataset, the proposed methodology can significantly extend its autonomy. Though, a reasonable requirement is that the basic labeled dataset needs to be representative enough and this requirement is delegated to the administrator. It is to be stated that we do not claim that our approach eliminates completely the need for human intervention, but it can significantly diminish it.

- STL is able to handle big data in a scalable manner. In fact, the more unlabeled data are given as input to the autoencoder, the more informative will be the new representation.

- STL can significantly uncover strong structures in the data, especially when the data features are statistically correlated.

Overall, the aforementioned characteristics of the combination of STL and MAPE-K address to a great extent the major limitation of misuse IDSs, namely their inability to deal with unknown situations. Considering the fact that misuse IDS are widely used over anomaly detection IDSs, our approach becomes even more impactful.

4.4 Evaluation

In this section we evaluate our novel methodology. More specifically, in order to highlight the advantages of our proposal and to make it comparable to other approaches, we present our results under a variety of legacy classification metrics, as those are defined in Section 2.2.3. Additionally, we detail on the used dataset and the setup of our evaluation experiments.

4.4.1 KDDCup'99 and NSL-KDD Datasets

We evaluate our methodology using KDDCup'99 [21] and NSL-KDD [22] well-known benchmarks datasets. Note that, this work does not aim to evaluate a detection algorithm per se. Instead, we aim to prove that our methodology is able to exploit unknown network flows to boost the detection efficiency in ambiguous environments. This means that the inherent limitations of KDDCup are orthogonal to our testbed. The reader can refer to Section 2.2.2 or to the original publications of the datasets for more details.

Precisely, in the context of this work, we merged all the datasets provided by KDDCup'99 and NSL-KDD for creating a single voluminous dataset that bears as many network traffic instances and as many attack classes as possible. Table 4.1 presents the instances of the used dataset. In total, the compiled dataset has of proximately 1.3 million network instances and comprises 40 classes (1 normal + 39 attacks), which come under different probability distributions and fall into the following 5 major categories:

- **Normal**: Normal traffic instances.
- **DoS**: Denial of Service.
- **PRB**: Probing - Surveillance and other means of probing.
- **R2L**: Remote to Local - Unauthorized access from a remote machine.
- **U2R**: User to Root - Unauthorized access to local superuser (root) privileges.

We removed all duplicates from the merged dataset to avoid any bias to the classification end-model. KDDcup dataset has some beneficial characteristics. To begin with, its variety in attack classes seems ideal for evaluating our methodology. Its 39 attack classes can be used to emulate a realistic and challenging testbed, where an IDS has to face unknown traffic instances every time an environmental change occurs. Additionally, the fact that the attack instances are drawn from different probability distributions directly challenges the STL method. Recall that according to its properties, STL is destined to handle efficiently large amounts of unlabeled data with that exact property. Moreover, KDDCup dataset has an imbalanced number of instances among its classes

TABLE 4.1: Normal and attack classes in KDDCup'99 and NSL-KDD.

Class	KDDCup'99 and NSL-KDD subclasses and the number of instances	#Instances
Normal	Normal traffic is not divided into sub-classes	936,152
DoS	back (2,633), neptune (297,085), smurf (6,688), teardrop (1,828), land (46), pod (448), apache2 (1,531), mailbomb (601), processtable (1,429), udpstorm (4)	312,293
PRB	satan (10,226), portsweep (6,787), ipsweep (7,411), nmap (3,200), mscan (2,044), saint (683)	30,351
R2L	ftp_write (22), warezclient (1,783), spy (4), named (34), warezmaster (1,986), multihop (50), xsnoop (8), sendmail (29), snmpguess (690), imap (25), snmpgetattack (357), worm (4), xlock (18), phf (12), guess_passwd (2,639)	7,661
U2R	buffer_overflow (102), httptunnel (278), loadmodule (22), perl (10), rootkit (46), xterm (26), ps (31), sqlattack (4)	519
Total	40 subclasses	1,286,976

and this feature reflects a realistic network condition. Finally, the KDDCup dataset created over a network experiment that lasted for 9 weeks and the final result was a dataset of approximately 7 million network instances with duplicates. Our compiled dataset consists of 1.3 million instances without duplicates. This implies that our dataset corresponds to a data collection period of at least 12 days. Hence, apart from the beneficial characteristics mentioned above, the compiled dataset comprises a realistic collection of network traffic that spans adequately over time and it is thus suitable for evaluating an adaptive mechanism.

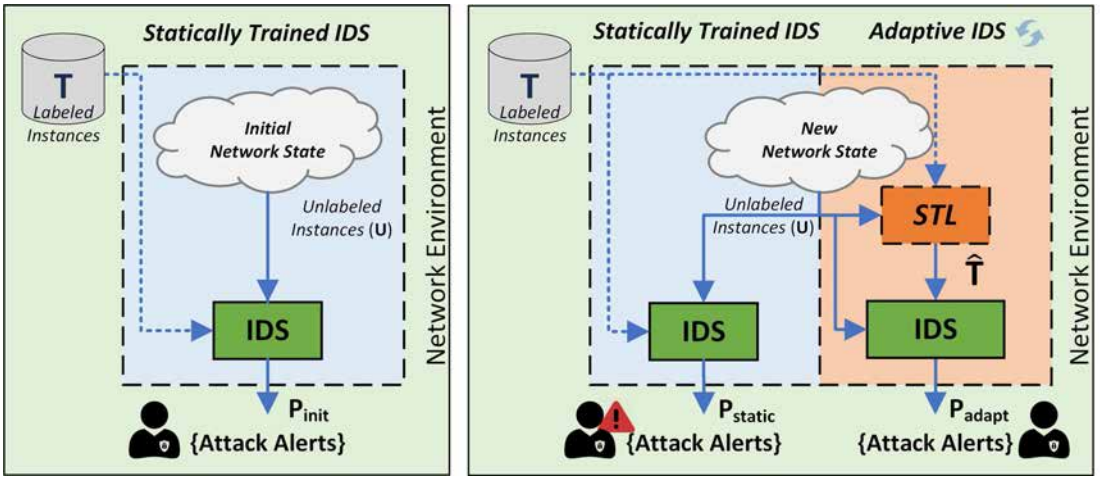


FIGURE 4.3: In the initial network state, a statically trained IDS can achieve an acceptable performance P_{init} (Left). In new network states, the adaptive IDS has the ability to sustain an acceptable performance in contrast to the statically trained IDS. The goal of our methodology is to improve the efficiency of the IDS so that $P_{adapt} > P_{static}$ (Right).

TABLE 4.2: Parameters' setup.

Parameter	Value
Sparse autoencoder	
• Number of hidden neurons	30
• Number of optimization iterations	500
• λ	0.0008
• ρ	0.06
• β	3
Softmax Regression	
• Weight decay parameter	0.0001
• Number of optimization iterations	100

4.4.2 Testbed and parameters

To emulate an ever-changing environment for the adaptive IDS, we came up with the following strategy. To emulate the initial state of the IDS, we train the IDS using Softmax Regression with an initial dataset T, which consists of a fraction of 10% of normal traffic and a randomly chosen subset of attack traffic. This attack-focused subset consists of 3, 3, 3 and 4 attacks subclasses of the major classes DoS, PRB, U2R, and R2L respectively. As it is the case with any legacy machine learning-based IDS, we cross-evaluate the IDS for achieving a robust end-model of more than 99% prediction accuracy. In a legacy situation, this end-model would be the one to defend any future environmental state of the network. Consequently, for emulating a new environmental state, we randomly select another piece of the dataset U which consists of 10% of normal traffic and 5, 5, 5, 8 attacks subclasses of the major classes DoS, PRB, U2R and R2L, respectively. *U might or might not* contain the classes or the instances gathered in T. Hence, depending on the divergence between T and U the new environment can be slightly or very different from the initial one. That is, it is expected to witness a low or even high drop of the IDS efficiency respectively. However, according to the proposed methodology, the adaptive IDS is able to exploit the U in order to obtain a better representation of the new environment by transforming the initial dataset T, and thus resisting to this efficiency drop. Note that U is in practice an unlabeled set of instances which is fed to the detection engine for prediction. Naturally this applies also in our case, but we are beforehand aware of the hidden classes of the instances for being able to measure the efficiency of the adaptive IDS contrary to the static one. If we denote the performance of the IDS in the initial phase, in the new but static phase, and after the adaptation as $P_{init}, P_{static}, P_{adapt}$ respectively, the goal of our methodology is to improve the efficiency of the IDS so that $P_{adapt} > P_{static}$. The evaluation strategy described in this section is presented intuitively in Figure 4.3.

As described in Section 4.3, the *Planning* activity is based on the Sparse autoencoder for identifying the new structure out of the unlabeled data. The autoencoder has to be tuned

beforehand by the security administrator. This process requires an initial tuning period on behalf of the administrator that relies basically on the nature of the network and the utilized data features. In our case, we selected the testbed's parameters (Table 4.2) based on an empirical study and our knowledge on the domain of the problems.

4.4.3 Results

This subsection elaborates on the results and compares the performance of the adaptive IDS against the static approach, based on the metrics presented in Section 2.2.3. In total, we subjected the IDS to 100 environmental changes, i.e., 100 diversely compiled datasets, blended with diverse attacks of each major class of the compiled KDDCup dataset. Figures 4.4 to 4.8 provide a graphical representation of the recorded metrics over the 100 environmental states.

As can be seen in Figure 4.4, our adaptive approach surpasses the static one in most of the environmental states. More specifically, in 84% of the states the adaptive approach achieved a higher accuracy score compared to the static one. The average accuracy of the static approach is 59.71%, while the adaptive's one is 77.99%. This means that in average our approach performs better by 18.28% over the 100 unknown states. Additionally, the standard deviation is 30.79% and 18.78% for the static and the adaptive approaches respectively. This fact quantifies what intuitively can be observed from Figure 4.4, where the adaptive curve witnesses less and smaller efficiency drops over the vast majority of the states. It is important to note that the maximum positive accuracy difference between the two approaches is 56.92% (state #8), while the maximum negative difference is -1.6% (state #36). In fact, as can be seen in Figure 4.4, in critical cases where the IDS accuracy drops significantly due to a state's high deviation with respect to the initial training set (T), the adaptive methodology demonstrates a significantly higher contribution that can sustain the IDS to acceptable detection levels. In total, 38% of the states present higher accuracy difference than the standard deviation (18.77%), and for these states, the average accuracy is increased by almost 48%. Hence, we can safely argue that the adaptive approach can significantly contribute to the overall security level in sudden network environmental changes (including attack incidents), while in cases where the IDS accuracy drops to some extent, the adaptive approach achieves almost the same performance as the static one. More precisely, for those 46 states where the accuracy difference per state is positive and less than the standard deviation, the average accuracy of the adaptive approach is greater than the static one by 0.51%. However, only for the 16 out of 100 states where the static approach performs better than the adaptive one, the average performance is 0.57% in favour of the static approach. All in all, the adaptive approach greatly outperforms the static one especially when it comes to critical states.

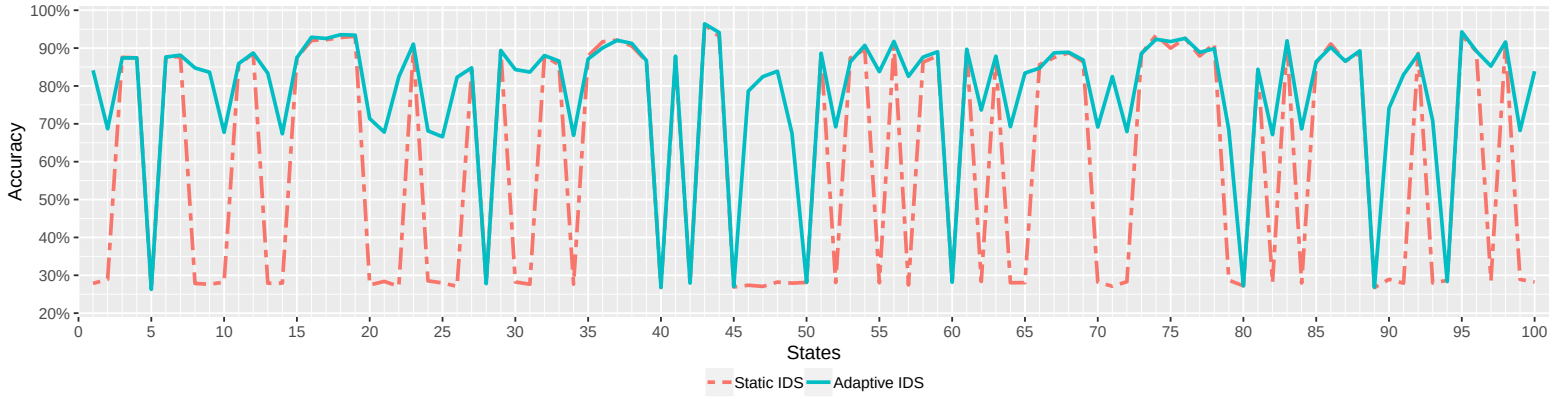


FIGURE 4.4: Deviation of IDS accuracy over 100 consecutive environmental states.

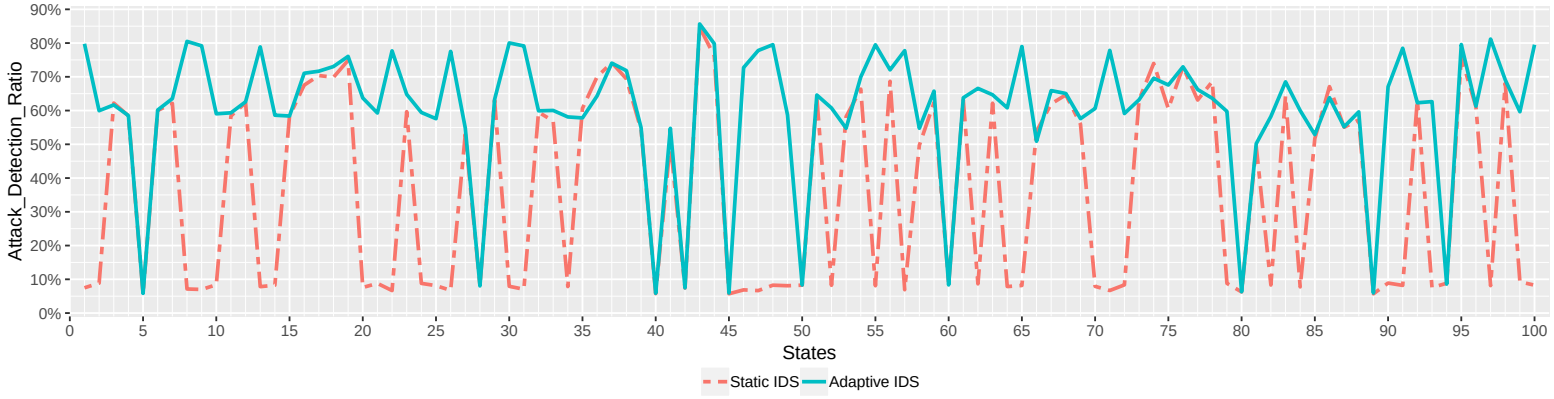


FIGURE 4.5: Deviation of IDS Attack Detection Ratio (ADR) over 100 consecutive environmental states.

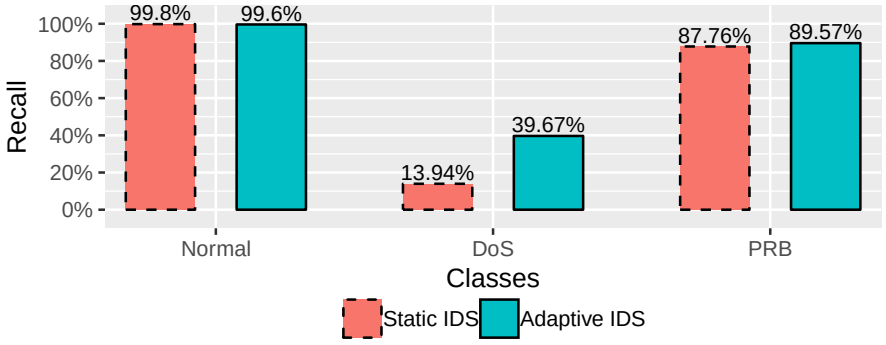


FIGURE 4.6: Accuracy (Recall) of all classes over the 100 states.

Figure 4.5 presents the ADR performance over the 100 environmental states. Recall from Section 2.2.3 that ADR measures the accuracy in detecting exclusively attacks instances, and thus reveals the performance in offensive incident detection. Overall, the adaptive approach scores an average ADR of 60.34% and outweighs the static one by 23.8%, as the latter scores an average ADR of 36.54%. The standard deviations are 28.34% and 19.69% for the static and the adaptive approach respectively. In total, the adaptive approach is proved better for the 86% of the states and, notably, the maximum ADR increment is 73.37% (state #8), while the maximum deficient percentage is -5.67% (state #36). As in the case of the accuracy metric, ADR achieves high scores for those states where the static approach witnesses significant performance drops. For those 38 critical states, where the ADR difference between the two approaches is greater than the standard deviation (19.69%), the average increment in the adaptive approach is 60.83%. Additionally, 48 states present an ADR difference smaller than the standard deviation, but again the adaptive approach performs better by 1.84% on average. Finally, only in 14 states the static approach performed better by 2.14% on average. Since ADR is considered a cardinal metric to measure the performance of an IDS, our results suggest that our adaptive method can significantly contribute in sustaining the detection ability of an IDS to high levels. The value of our novel proposal lies in the fact that it can breathe new life into the IDS in critical/sudden situations and increase ADR by up to 73.37%. In principle, in critical situations where the IDS performance drops significantly there is an urgent need for human intervention. Namely, in these cases, ADR can drop to such deficient levels that most of the attacks occurring in the network can go completely unnoticed. Hence, instead of triggering a process of manually retraining the IDS, our solution provides a self-adaptive and autonomous way to keep the IDS's operational ability to high levels.

Furthermore, Figure 4.6 gives an overview of the average accuracy (recall) per class included in the dataset. For each class, we offer a side-by-side comparison of the performance of the static versus the adaptive method. In the figure, the accuracy for the

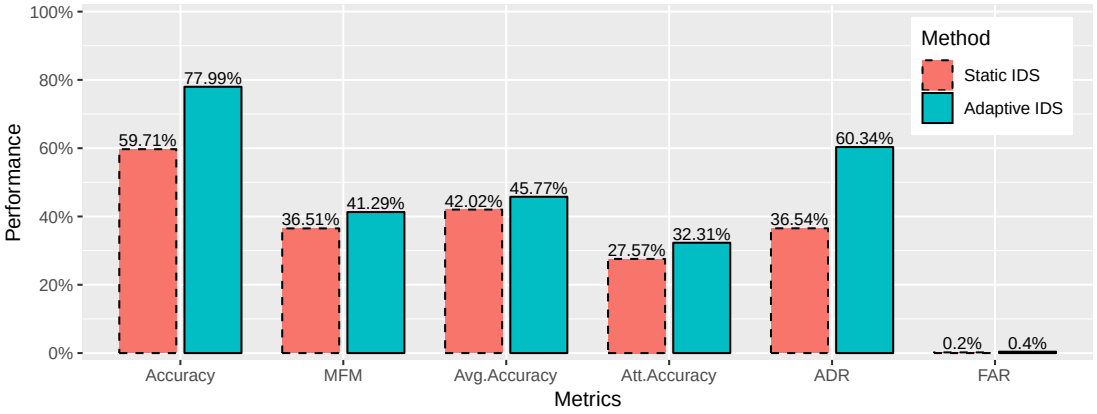


FIGURE 4.7: Performance comparison over all average metrics.

Normal class is almost identical for the two methods. In fact, there is a tiny difference of 0.2% in favor of the static approach. Given that this accuracy is the average of the recall over 100 states, this difference is characterized as minor. On the bright side, the difference concerning the accuracy of the DoS attack class is significant. More specifically, the adaptive method achieves an average accuracy of 39.67% in contrast to the static one that achieves 13.94%. This difference of 25.73% on an average metric is noteworthy. The maximum difference of DoS accuracy recorded among all the 100 states is 80.34% and is perceived in state #8. Actually, in that state, the static IDS witnessed a critical situation as the accuracy in DoS detection was only 0.52% due to the high deviation induced by the new environment. On the other hand, the adaptive approach was able to acquire the necessary knowledge for the unknown network traffic and boost the accuracy metric to 80.87%. Additionally, regarding PRB attack detection accuracy, once again the adaptive approach achieves an average score of 89.57% against the static one, which scored 87.76%. Regarding the attack classes of U2R and R2L, both methods were incapable of detecting any attack vector of those classes. The reason behind this fact is that the aforementioned classes have a small number of instances. Recall from subsection 4.4.2 that any new environmental state stems from a sampling technique. Apparently, the small numbers of instances of those normal/attack classes and the sampling technique to simulate the new environment do not provide adequate instances to build a solid ground truth for the classifier. Overall, the adaptive approach is capable of keeping a stable accuracy on identifying the normal traffic, while its attack detection performance is originated primarily from the DoS attack accuracy and secondarily from the PRB attacks detection accuracy.

The overall performance of the adaptive and the static methods is illustrated in Figure 4.7. The dominance of the adaptive method is verified by all the metrics. Apart from the accuracy and the ADR metrics analyzed above in detail, also the rest of metrics prove the superiority of our methodology. The difference of 4.78% in the MFM

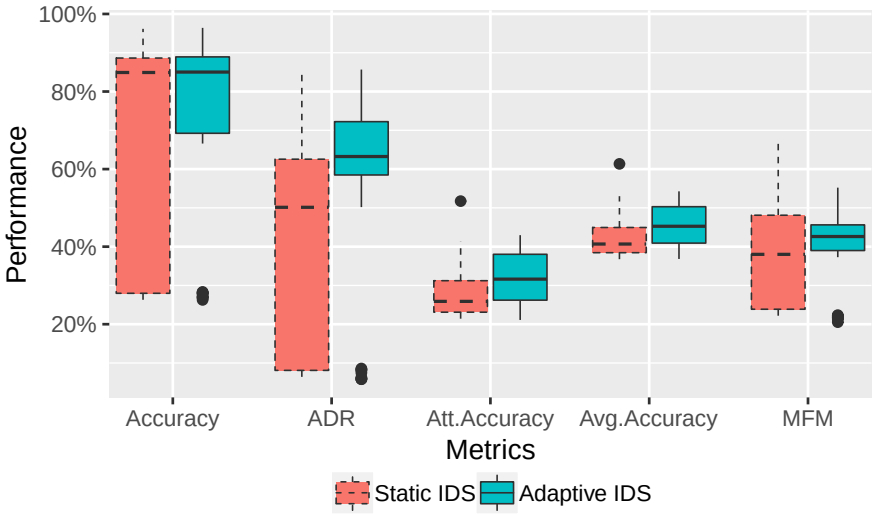


FIGURE 4.8: Side-by-side Boxplots of metrics for both methods.

metric reveals that the adaptive approach is able to keep the balance between Recall and Precision among all the dataset classes to a greater extent. Note that the MFM metric, as defined in Section 2.2.3, is the unweighted average of recall and precision. That is, the unweighted MFM constitutes a more strict metric to evaluate our method, as it treats all classes equally independently of the classes' size. This means that the adaptive method is not only able to provide better attack detection rates, but it is also capable of identifying with higher precision the correct class where the attack instances belong to. Finally, the small deficiency (0.2%) reported in the accuracy metric of the Normal class for the adaptive approach is reflected in the slightly increased FAR of 0.4% in contrast to the 0.2% achieved by the static approach.

Additionally, in order to better understand the distributional characteristics of our results over the chosen metrics for the two methods, Figure 4.8 provides a side-by-side comparison of the boxplots of the metrics. Note that the FAR metric is absent since its deviation is minor. In fact, Figure 4.8 puts in a nutshell the behavior of the two methods as it was already portrayed in Figure 4.4 and Figure 4.5. The dominance of the adaptive approach becomes clear as in all boxes the medians are comparatively higher than those in the static approach. Especially for the boxes representing the Accuracy and ADR metrics, we can notice a significant difference. Regarding the static method, the long size of the second quartile, both for the Accuracy and the ADR, reveals the inefficiency of this method to sustain an acceptable detection level for the IDS in critical situations. This is not the case for the adaptive method, as the concise inter-quartile reveals an overall high stability in both metrics. Additionally, it can be observed that the box of the Accuracy metric of the adaptive approach is slightly higher for the observations above the median (third quartile). Regarding the box of the ADR metric, the adaptive

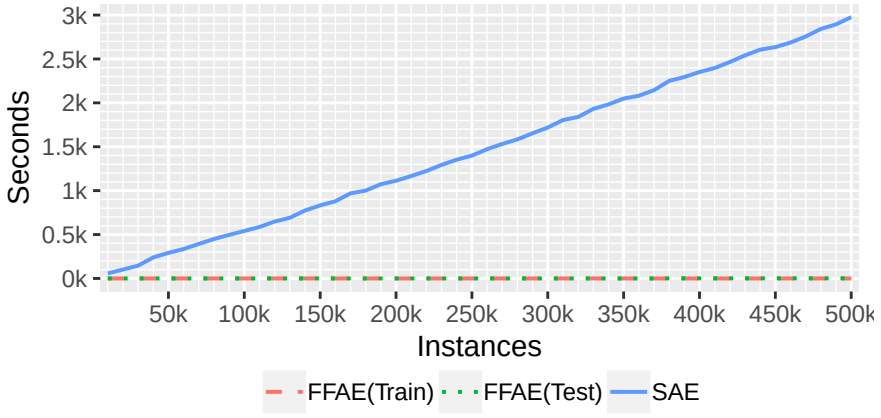


FIGURE 4.9: Time performance analysis for autoencoders.

approach achieves significantly higher scores as it is noteworthy that its third quartile starts at that point where the third quartile of the static approach ends. This proves the higher robustness of the adaptive method in detecting offensive incidents in previously unseen environments. Regarding the Attack Accuracy, Average accuracy, and the MFM metrics, once again the boxplots reveal the benefits of the adaptive approach. Note that the aforementioned metrics are average metrics, and thus it is normal to present a lower deviation in contrast to the Accuracy and the ADR metrics. It is noteworthy that the concise size of the MFM boxplot demonstrates the ability of the adaptive approach to keep the balance between the recall and precision over all the classes and across all the environmental states.

4.4.4 Time performance analysis

Since our proposal aims to provide an autonomous method for IDS self-adaptation it is critical to analyse the time performance and scalability of its core components. More specifically, the Sparse autoencoder (SAE) and the FeedForward autoencoder (FFAE) of the planning activity are those components that enable the IDS to adapt to a new environmental state. Hence, we measured the time performance of the autoencoders using an incrementing size of dataset instances to evaluate also their scalability. As can can be seen in Figure 4.9, we fed the autoencoders with dataset pieces ranging from 10k to 500k instances having a pivot step of 10k instances. Figure 4.9 reveals that the SAE behaves linearly, while the FFAE (both in the training and testing phase) needs less that 1 sec to obtain the transformation of the whole dataset. The SAE training phase needed 2.977 secs, i.e., ∼49.5 min. given a dataset of 500k instances. The linear performance behavior of the SAE advocates the ability of our approach to scale in big data network environments. From our analysis, one could say that the adaptation cycle can occur virtually indefinitely, while the extra transformation layer added by the

FFAE is negligible. Our python implementation was executed on a server empowered with an Intel Xeon E5-2630 v4 @ 2.20GHz CPU. For measuring the time performance, we exclusively utilized only one thread of the CPU. Naturally, the training time could be significantly reduced with the use of GPU accelerators.

4.5 Discussion

The novel methodology described and evaluated in this chapter combines the benefits of STL [239] and MAPE-K [240] to deliver a holistic deep learning-based methodology toward self-adaptive and autonomous misuse IDSs. Our solution addresses the challenges of this particular field to a great extent and, to the best of our knowledge, it is the first one to evaluate an IDS under consecutive environmental changes. That is, we prove its ability not only to adapt to new and unknown environments, but to achieve significantly higher scores contrary to a static approach. In fact, as detailed in section 4.4.2, the compiled dataset which consists of 39 attack classes along with the strategy of simulating new environmental states out of it, reflects with high consistency a realistic situation. The robustness and agility of the proposed methodology is advocated by its superiority over a wide range of legacy classification metrics and over 100 different environmental states. It is worth mentioning that across the vast majority of states our novel approach was able to deliver a significantly higher detection ratio surpassing the static IDS by up to 73.37%. In few states we achieved better but comparable results with the static approach, while only in a handful of states the static approach proved better for a small percentage (2.14% on average). In a nutshell, the acquired results demonstrate that the proposed methodology can revitalize a misuse IDS and boost its ADR by up to 73.37% in critical situations.

Additionally, our simulation posit practical challenges to our methodology. In realistic occasions, an IDS will have to deal with unknown attacks of any class, where their features might be drawn of different probability distributions. Our sampling approach tries to imitate such a challenge and stresses our method over consecutive environment changes. However, this challenge is compensated inherently by the STL properties. Moreover, our evaluation is given in the context of a multi-classification instead of a binary one (Attack/Normal). This provides deeper insights about the performance of the presented approach. Namely, it is important for an IDS solution, not only to be able to detect an attack, but also to designate the class where the attack belongs to in order to aid well-defined counteraction plans [246].

Our proposal comes to deal with a well-known disadvantage of misuse IDSs, namely their stiffness to adapt upon changes. Note that we do not claim that our proposal is able to

identify new attack classes, but it is indeed able to grasp an attack's nature based on generalized features reconstructions stemming directly from the unknown environment and its unlabeled data. Crucially, this reconstruction is a product of a scalable method which is able to handle big network data. We prove that given an initial labeled training set that serves as a basis, the proposed solution is able to revitalize the efficiency of the IDS without the constant need to refresh it, and then retrain the IDS. The transformation of the initial training set – based on the knowledge acquired directly from the current state of the network – provides a high lever of automation in the retraining process. That is, the presented methodology addresses the inherent limitation of misuse IDSs to adapt to new environments, while it significantly alleviates the burden of administrators of constantly refreshing its training set.

4.6 Related work

Machine learning IDSs is a research topic that attracted the attention of the research community for decades. In the recent years, several neural networks-based solutions arose and offered promising solutions. This section offers an overview of deep and shallow learning methods, while it also elaborates on whether they bear any adaptable features. This section also refers to methodologies that aim to provide a level of automation in the IDS adaptation and discusses our contributions over them.

4.6.1 Neural Networks-based approaches

The work in [36] introduces a deep learning solution for NIDSs. The authors utilize the self-taught learning methodology exclusively as an unsupervised feature learning method for supporting a statically trained IDS. Instead, our solution provides a more powerful setup of the STL in conjunction with MAPE-K methodology to deliver a deep learning methodology for adaptive IDSs. In fact, contrary to [36], we pass the IDS through environmental changes to prove that our approach is able to generate knowledge out of unknown environments.

The authors in [247] propose an IDS algorithm based on Spectral Clustering (SC) [248] and Deep Neural Networks (DNN) [31]. Through SC the proposed method is able to identify cluster centers that divide a raw dataset into data clusters with similar features. Those data clusters are fed as training data into DNN's of multiple layers. The algorithm trains as many DNN's as the clusters identified by the SC and aggregates the final result in an ensembled way. However, the proposed deep learning approach does not provide any kind of adaptiveness to the system.

In the context of MAPE-K control loop, the authors in [236] proposed an adaptive IDS in terms of network environmental changes. By exploiting the MAPE-K model, the authors were able to perceive the environment changes and plan appropriate update actions in the Snort IDS detection rules. The authors in [236] utilize MAPE-K model for regulating the adaptive process. However, our approach goes beyond that point and exploits MAPE-K model to build a deep learning misuse IDSs in the adaptive frame.

The authors in [37] use deep learning techniques to detect network anomalies in 5G networks. The authors utilize DNN and Long Short-Term Memory (LSTM) Recurrent Networks to empower the anomaly detection. Their proposal takes self-adaptation actions with respect to the network load requirements by applying management policies. Nevertheless, the adaptation policies are applied to the network load management rather the detection framework per se.

The work in [249] proposes a deep learning method for detecting DoS attacks based on Restricted Boltzmann Machine (RBM). Specifically, the authors used a Gaussian–Bernoulli RBM with 7 hidden layers with 100 neurons each. Through the aforementioned setup the proposed method learns a reduced set of new features of the NSL-KDDTrain+_20Percent dataset [22] (train: 25,194 instances, test: 4,508 instances). According to the authors, the Gaussian—Bernoulli RBM is able to outperform the deep learning approach of Bernoulli-Bernoulli RBM and Deep Belief Network, and the legacy machine learning methods of SVM (radial basis), SVM (epsilon-SVR), and decision tree. Even though the proposed method learns new features representation out of unlabeled data, it is not exposed to unknown data for supporting an adaptable approach.

The authors in [250] designed an IDS model by stacking dilated convolutional autoencoders (DCAEs) for learning features representations from unlabeled data. In their experiments, the authors tested the generalization ability of their detection model by testing it with previously unknown attacks. Even though the authors pose their trained model against new attacks, they do not proceed to any automated retraining method. Rather they aim to exclusively test the generalization ability of the learned features employed to statically train the IDS. Additionally, the authors approach the problem as a binary one (normal/attack), while in our classification case we deliver a multi-classification method.

The authors in [38] propose a new type of autoencoder namely non-symmetric deep autoencoder (NDAE) and they utilized it in a classification model using stacked NDAEs. According to the authors, the NDAE engaged only an encoding phase for reducing the complexity of the network with minor effect on the accuracy of the model. At the end of the stacked NDAE, the authors attach Random Forest algorithm that undertakes the classification task based on the features learned from the NDAEs. The proposed setup

achieves high accuracy rates, but the author's methodology does not bear adaptability characteristics.

4.6.2 Adaptive methodologies in IDS realm

Although the IDS research on IDS has offered an large amount of works [251, 70, 24, 252] the vast majority of them focus on providing highly accurate end-models with minor false alarm rates. Thus, the adaptability property remains an open issue and it is a well-known drawback, especially for the misuse IDS domain. Still, there are methodologies that, according to the literature [235], could provide the necessary foundations to adaptive IDSs. Among them, Learning Classifier Systems (LCS) [237], Artificial Immune Systems (AIS) [238], Swarm intelligence [67], Evolutionary computing [253] and Reinforcement learning [254]. There is no doubt that all of these approaches can offer their principles to build adaptive systems for practical problems. However, the intrusion detection problem has some inherent characteristics that one has to take into consideration when aiming to build a self-adaptive and autonomous IDS. That is, a self-adaptive and autonomous approach implies the complete disengagement of the human factor or at least a minimal interaction in the form of a supervision.

The self-adaptation property implies that an IDS should be able to adapt itself to the needs of a new environment even without the need of feedback from the administrators. This means that the adaptation process of an IDS cannot be based on labeled data because labeling network data of a new environment is a demanding engineering task. That is, approaches that base their adaptation on labeled data can be considered adaptive, but not self-adaptive. As a result, to support self-adaptation in the context of intrusion detection, we need to invest on methods that can exploit unlabeled data to improve the detection performance.

Additionally, in an on-line machine learning problem, where new instances need to be classified instantly and accurately, there is a need of approaches that can adapt to new environments also in a realtime fashion. This entails that solutions that rely their adaptation on a trial-and-error approach as those based on reinforcement learning, seem to be impractical for this nature of problems. In fact, an IDS cannot learn in the same way that, say, a robot does. For example, if a robot encounters an obstacle then it learns out of this incident and proceeds to an adaptation of its objectives. Unfortunately, in the intrusion detection context, the only entity that can identify a mis-classification (i.e., an obstacle) is the administrator who notices an ongoing attack. An IDS is not in position to know if a new instance is misclassified or not. In other words, the IDS cannot see the "obstacles", but the "obstacles" are in practice attacks that go unreported or

normal instances which are detected as attacks and increase the false alarm rate. In this sense, we need to invest on methods that can learn from the unknown environment in an autonomous way.

To this end, our proposal tries to address the aforementioned challenges. The combination of Self-taught learning [239] and MAPE-K [240] brings together the benefits of transfer learning from unlabeled data and places this ability in the frame of autonomic computing. That is, our self-adaptive and autonomous method enables an IDS to extract new features out of the unlabeled and unknown traffic of a new environment and exploit them for retraining in an autonomous way its detection engine. Furthermore, our solution enables the IDS to adapt according to the dynamics of the new environment even if this is overwhelmed by previously unseen traffic. Given an initial training set, a new features construction is learned using a neural networks-based sparse autoencoder, and via a feed-forward autoencoder the initial training set is updated to meet the challenges of the new environment.

4.7 Conclusions

This chapter focused on a novel methodology that advances the state-of-the-art in the literature of misuse IDS. The highlight of the presented scheme is that it can practically render any misuse IDS autonomous, i.e., self-adaptive to the ever-lasting changes made in its network environment. This means that in such frequently occurring (and sometimes sudden) transition periods, the IDS is able to maintain an at least acceptable attack detection rate, which otherwise is fated to drop abruptly, rendering the IDS useless. This quality is also a great relief to the security administrators who after a network environment change are granted enough time to possibly update the IDS's detection model. The proposed methodology uniquely blends the MAPE-K reference model and a deep-learning technique called self-taught learning to enable an IDS to identify previously unseen attacks via reconstructions made on unlabeled data. The linear performance behavior for acquiring the aforementioned reconstructions, renders our proposal suitable for contemporary big data network environments. The effectiveness of our proposal is demonstrated through extensive experimentation considering several metrics and a plethora of attacks included in widely used datasets.

Conclusions and Future Directions

4.8 Conclusions

In an era when ICT infrastructures gain in size and complexity at a fast pace, their protection exceeds the limits of human processing capabilities. Hence, it is crucial to investigate for methods which not only achieve high detection efficacy, but they are able to support the decision making in critical situations. In this direction, such methods should bring intelligence in the security mechanisms, enable automation and self-adaptation for alleviating the security administrators. Reaching to the end of this doctoral thesis, it becomes clear that intrusion detection and reaction remains a highly active research area as the battle between aggressors and defenders has taken the form of an "arm-race". Through our research, we investigate the current state-of-the-art on cost-benefit intrusion response methodologies for providing optimal security countermeasures, while we also introduce novel methodologies for enabling versatile intrusion detection.

More specifically, as described in chapter 2, intrusion response systems is a research field which encompasses several research challenges. Through an extensive analysis, we reveal a gamut of open research challenges and we provide future directions and best practices for building reactive and remediative mechanisms that can support security administrators during critical conditions. However, as described in chapter 3, an optimal incident response strategy should be triggered upon a correctly identified and classified incident. That said, this doctoral thesis introduces two novel methodologies for improving the state-of-the-art in network intrusion detection. Chapter 3 proposes *Dendron*, a methodology for evolving detection rules, which can detect both popular and rare intrusive incidents utilizing evolutionary computing techniques. Additionally, the methodology presented in chapter 4 is destined to tackle the main limitation of misuse detection systems, which is the lack of agility in adapting to new and "unknown" environments.

Objective	Chapter	Contribution	Publication	Patents
Obj. 1	2	A detailed overview of fundamental characteristics of reaction frameworks.	[246]	
Obj. 1	2	A comprehensive survey and side-by-side comparison on reaction frameworks for optimal countermeasure selection.	[246]	
Obj. 2	3	Design and implementation of an evolutionary classifier called *Dendron*.	[251]	
Obj. 2	3	A method and systems for generating detection rules and taking countermeasures.		[252]
Obj. 3	4	Introducing Deep Learning Self-Adaptive Misuse Network Intrusion Detection Systems.	[255]	[256]

TABLE 4.3: Overall PhD Thesis Contribution.

This chapter summarizes the contribution of this doctoral thesis and provides research directions for future work.

4.9 Thesis Contributions

In accordance to the objectives presented in chapter 1, this doctoral thesis aims to shed light to the state-of-the-art methodologies which aim to provide optimal and cost-benefit counteraction of cyber attacks in the context of intrusion response systems. Additionally, it focuses on the deployment of advanced machine learning techniques to tackle known limitations of the intrusion detection literature. A side-by-side comparison of the thesis objectives and our contributions in terms of publications in peer-reviewed venues and patents/patent applications is given in Table 4.3.

Initially, our research concentrated on the analysis of IRSs, which aim to provide optimal countermeasures against cyber attacks in a reactive manner. In fact, there are several solutions to enable the observability and monitoring networks for identifying misuse or anomalous incidents, as well as asset inventory tools to draw the security landscape of a network. Even though those tools can shape a generic picture of a security state of an infrastructure, the decision making for applying recovery actions upon incidents remains an open issue and the community still lacks proper tools. This shortcoming motivated us to shed light on today's research solutions which aim to provide optimal countermeasures in a cost-benefit manner. In this direction, and in accordance to the first objective of this thesis, chapter 2 and [246] provided an extensive analysis of reactive methodologies resulting the following research challenges, which remain widely open among the reviewed solution.

- Lack of scalable solutions.
- Limited countermeasure knowledge.
- Correspondence between countermeasures and attacks.
- Lack of standard representation.
- Metrics and scoring system.
- Mitigating Zero-Day attacks.

Given the aforementioned challenges, our work advocates that the automation of decision making for incident response has still many steps to take for reaching the point of providing complete counteraction tools that can assist the security administrators to their duties. Until then, administrators will act based on their knowledge and experience and by following hopefully documented security policies.

Based on the analysis provided in chapter 2 and [246], it becomes clear that any response strategy must be triggered upon a correctly identified incident, otherwise an automated responsive mechanism could trigger unnecessary remediation actions, which could disrupt the well-adjusted operational state of a system. In this direction, this doctoral thesis developed a methodology which enables a signature-based IDS to accurately identify the class within an incident belongs to. By utilizing this methodology, an IDS is in position to identify accurately both popular and rare incidents, as the latter are significantly neglected by related works. Our work blended the beneficial characteristics of Decision Trees and Genetic algorithms for evolving a population of Decision Trees with the purpose of acquiring an accurate multi-classed end-model able to keep the balance among all the network traffic classes. This feature is achieved via a weighted selection probability function, which guides the evolutionary process to come up with an individual, which is not biased towards neglecting those classes represented by a smaller percentage of instances in a dataset. According to the evaluation results presented in chapter 3 and [251], our approach was able to surpass well-known algorithms in terms of key performance metrics over 3 distinct benchmark datasets, namely KDDCup'99 [21], NSL-KDD [22] and UNSW-NB15 [23].

The aforementioned novelties were published in [251], while a Patent was filed by NEC Corporation for reserving the exploitation rights of a corresponding system and method [252]. Summarizing the above, on can understand that the contribution of our work in [251] is twofold. On the one hand, the multi-classification approach of the problem and the intention to designate an incident to a specific class are aligned to the requirements of incident response mechanism, since an optimal countermeasure must be tailored to the particular characteristics of a threat. On the other, the rule induction method aims to assist the security administrator in understanding the network traffic and the attacks themselves, as the rules derived out of a decision tree can be interpreted by a human.

Even though a methodology for detection rules induction can be proved beneficial and alleviates the burden of security administrators, the major limitation of misuse detection systems remains. These systems are rather static and demand a rather huge engineering task to update the detection system. In principle, an ML-based IDS needs to be re-evaluated periodically, as it cannot be "set it and forget it". Robust intrusion detection

requires frequent, rigorous re-training of the model by providing data with high fidelity to the real world [234]. In this direction and in accordance to the third objective of this doctoral thesis, the work presented in chapter 4 and published in [255], came to tackle the lack of agility of misuse IDSs. By blending the beneficial characteristics of Self-Taught Learning [239] and MAPE-K [240] methodologies, we built a novel self-adaptive misuse IDS, which is able to exploit the inherent uncertainty of the network environment to revitalize its detection engine and keep it up-to-date. By adopting state-of-the-art deep learning methodologies, the proposed system is able to grasp an attack's nature based on generalized feature reconstructions stemming directly from the unknown environment and its unlabeled data. We argue that our approach is a perfect fit to the nature of the problem, as it can exploit unlabeled data drawn of any distribution and of any class. These beneficial characteristics advocate the suitability of this methodology in practice.

In addition to the above-mentioned contributions, our methodology is suitable for Big Data environments, as it presents a linear performance. Our evaluation results support the ability of our proposal to adapt to new network environmental states, achieving attack detection scores which surpass a static detection approach by up to 73.37%. We argue that our research sets new standards in the field of adaptive detection systems, as the automation of the retraining process can significantly eliminate the human intervention. The aforementioned novelties were published in [255], while a Patent was filed by NEC Corporation for reserving the exploitation rights of a corresponding system and method [256].

4.10 Future Research Directions

This Phd thesis has mainly contributed to the field of network IDSs by introducing versatile methodologies with an eye towards addressing key limitations in this field of research. Additionally, a significant step was taken in the direction of IRSs, by reporting and decomposing the reactive methodologies, which have been proposed in the literature so far, through an extensive analysis. Undoubtedly, the quest for novel defense and response mechanisms must be the bastion for the future battles. It becomes clear that several steps have to be taken for introducing more versatile detection and reaction techniques that can bring the advantage to the defensive side. To this end, this subsection elaborates on possible research directions.

- *Self-adaptation and autonomicity* - In a digital world which is constantly expanding, Research & Development programs aim to foster innovation on AI-assisted autonomous and adaptive systems for cyberdefense [257]. Our work presented

in [255] and our invention in [256] are aligned to this direction and aim to make the difference against rigid detection systems. An in depth analysis of aspirant AI-driven schemes and systems adaptation methodologies that can convey those qualities to detection and reaction systems can be beneficial for the research community and foster innovation in this particular field.

- *Deep learning approaches* - Deep learning techniques have witnessed special recognition the last few years mainly in the field of image recognition. The hardware solutions, such as GPU accelerators, gave a boost to the field. The investigation of deep neural network approaches can still benefit the efficacy of detection and reaction systems, while further research must be conducted regarding the interpretation of their behavior in the context of network intrusion detection.

- *Big Data and parallelization techniques* - As ICT infrastructures gain in size and complexity, modern networks present high capacity for supporting highly active environments. Hence, a detection methodology needs to be scalable for handling large volumes of data in a timely manner. However, evolutionary algorithms takes a considerable amount of time for training and delivering an end-model. That said, it would be worthwhile to investigate the performance of such methods in conjunction with parallelization techniques and GPU accelerators.

- *Designing cost-benefit reaction systems* - Given the analysis conducted in this doctoral thesis regarding the intrusion reaction mechanisms, a potential future direction is the design of a cost-benefit reaction mechanism. By taking into consideration the research challenges of the field, it is worth to investigate for scalable methodologies which are driven by widely adopted security scoring systems and threat intelligence. Such methods can be proved a great defense asset to support decision making. In fact, Security Orchestration, Automation and Response (SOAR) solutions promise to go beyond what is possible in today's Security information and event management (SIEM) platforms [258].

- *Joined deployment of detection and reaction system* - A reactive system is triggered upon the detection of an offensive incident. This chain of events needs to be investigated in tandem as the former can significantly affect the efficacy of the latter. Hence, a future work could illuminate the challenges and best practices for fine-tuning AI-based detection and reaction mechanisms, which work in synergy.

- *Building new challenging datasets* - It has been widely reported in the literature that the community lacks of contemporary datasets that reflect the modern network conditions and attack characteristics. Hence, an aspirant future direction is to design multidisciplinary datasets that combine characteristics from various

realms like IoT devices and smart devices, and investigate for unified and interoperable detection solutions.

- *Wireless IDS idiosyncrasies* - Network traffic from wireless and mobile devices has already exceeded that produced by wired devices. This calls for IDS/IPS/IRS that are specially designed to cope with the distributed and heterogeneous idiosyncrasies of the wireless terrain, not to mention the absence of rigid access control, the highly dynamic network conditions, and the still limited bandwidth that also apply to these networks. So far, only a handful of approaches touched upon this challenge [70], and thus it would be interesting to see if advanced methodologies like *Dendron* [251] and our self-adaptive detection methodology [255] can be equally effective if deployed in distributed wireless terrains comprised by picocells and femtocells, including current 4G and imminent 5G networks.

Bibliography

[1] James P Anderson. Computer security threat monitoring and surveillance. *Technical Report, James P. Anderson Company*, 1980.

[2] Wenke Lee and Salvatore J. Stolfo. Data mining approaches for intrusion detection. In *Proceedings of the 7th Conference on USENIX Security Symposium - Volume 7*, SSYM'98, pages 6–6, Berkeley, CA, USA, 1998. USENIX Association. URL http://dl.acm.org/citation.cfm?id=1267549.1267555.

[3] Wenke Lee, S. J. Stolfo, and K. W. Mok. A data mining framework for building intrusion detection models. In *Proceedings of the 1999 IEEE Symposium on Security and Privacy (Cat. No.99CB36344)*, pages 120–132, May 1999. doi: 10.1109/SECPRI.1999.766909.

[4] J. Boyd, G.T. Hammond, and Air University (U.S.). Press. *A Discourse on Winning and Losing*. Air University Press, 2017. ISBN 9781585662791.

[5] Snort, network intrusion detection & prevention system. https://snort.org/. Accessed: 2019-04-11.

[6] Suricata, open source IDS/IPS/NSM engine. https://suricata-ids.org/. Accessed: 2019-04-11.

[7] The zeek network security monitor. https://www.zeek.org/. Accessed: 2019-04-11.

[8] Vern Paxson. Bro: a System for Detecting Network Intruders in Real-Time. *Computer Networks*, 31(23-24):2435–2463, 1999. URL http://www.icir.org/vern/papers/bro-CN99.pdf.

[9] OSSEC, open source HIDS security. http://www.ossec.net/. Accessed: 2019-04-11.

[10] ntop – high performance network monitoring solutions based on open source an commodity hardware. https://www.ntop.org/. Accessed: 2019-04-11.

[11] nfdump - a toolset for collecting and processing netflow and sflow data. http://nfdump.sourceforge.net/. Accessed: 2019-04-11.

[12] Nfsen - netflow sensor. https://www.ntop.org/. Accessed: 2019-04-11.

[13] Openvas - open vulnerability assessment system. http://www.openvas.org/. Accessed: 2019-04-11.

[14] Nessus vulnerability assessment solution. https://www.tenable.com/products/nessus/nessus-professional. Accessed: 2019-04-11.

[15] Nexpose - vulnerability management software. https://www.rapid7.com/products/nexpose/. Accessed: 2019-04-11.

[16] Hung-Jen Liao, Chun-Hung Richard Lin, Ying-Chih Lin, and Kuang-Yuan Tung. Intrusion detection system: A comprehensive review. *Journal of Network and Computer Applications*, 36(1):16 – 24, 2013. ISSN 1084-8045. doi: https://doi.org/10.1016/j.jnca.2012.09.004. URL http://www.sciencedirect.com/science/article/pii/S1084804512001944.

[17] P. García-Teodoro, J. Díaz-Verdejo, G. Maciá-Fernández, and E. Vázquez. Anomaly-based network intrusion detection: Techniques, systems and challenges. *Computers & Security*, 28(1-2):18 – 28, 2009. ISSN 0167-4048. doi: http://dx.doi.org/10.1016/j.cose.2008.08.003.

[18] M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita. Network anomaly detection: Methods, systems and tools. *IEEE Communications Surveys Tutorials*, 16(1):303–336, First 2014. ISSN 1553-877X. doi: 10.1109/SURV.2013.052213.00046.

[19] A. L. Buczak and E. Guven. A survey of data mining and machine learning methods for cyber security intrusion detection. *IEEE Communications Surveys Tutorials*, 18(2):1153–1176, Secondquarter 2016. ISSN 1553-877X. doi: 10.1109/COMST.2015.2494502.

[20] Prem Uppuluri and R. Sekar. Experiences with specification-based intrusion detection. In Wenke Lee, Ludovic Mé, and Andreas Wespi, editors, *Recent Advances in Intrusion Detection*, pages 172–189, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg. ISBN 978-3-540-45474-8.

[21] Wenke Lee and Salvatore J. Stolfo. A framework for constructing features and models for intrusion detection systems. *ACM Trans. Inf. Syst. Secur.*, 3(4):227–261, November 2000. ISSN 1094-9224. doi: 10.1145/382912.382914.

[22] Mahbod Tavallaee, Ebrahim Bagheri, Wei Lu, and Ali A. Ghorbani. A detailed analysis of the kdd cup 99 data set. In *Proceedings of the Second IEEE International Conference on Computational Intelligence for Security and Defense Applications*, CISDA'09, pages 53–58, Piscataway, NJ, USA, 2009. IEEE Press. ISBN 978-1-4244-3763-4.

[23] Nour Moustafa and Jill Slay. UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In *2015 Military Communications and Information Systems Conference, MilCIS 2015, Canberra, Australia, November 10-12, 2015*, pages 1–6, 2015. doi: 10.1109/MilCIS.2015.7348942.

[24] Salma Elhag, Alberto Fernández, Abdullah Bawakid, Saleh Alshomrani, and Francisco Herrera. On the combination of genetic fuzzy systems and pairwise learning for improving detection rates on intrusion detection systems. *Expert Systems with Applications*, 42(1):193 – 202, 2015. ISSN 0957-4174. doi: 10.1016/j.eswa.2014.08.002.

[25] Argus. The network audit record generation and utilization system. URL https://qosient.com/argus/index.shtml.

[26] Canadian Institute for Cybersecurity. ISCX datasets. URL https://www.unb.ca/cic/datasets/ids.html.

[27] Ali Shiravi, Hadi Shiravi, Mahbod Tavallaee, and Ali A. Ghorbani. Toward developing a systematic approach to generate benchmark datasets for intrusion detection. *Computers & Security*, 31(3):357 – 374, 2012. ISSN 0167-4048. doi: 10.1016/j.cose.2011.12.012.

[28] Constantinos Kolias, Georgios Kambourakis, Angelos Stavrou, and Stefanos Gritzalis. Intrusion detection in 802.11 networks: Empirical evaluation of threats and a public dataset. *IEEE Communications Surveys and Tutorials*, 18(1):184–208, 2016. doi: 10.1109/COMST.2015.2402161. URL http://dx.doi.org/10.1109/COMST.2015.2402161.

[29] AWID. Aegean wifi intrusion dataset. URL http://icsdweb.aegean.gr/awid/.

[30] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359 – 366, 1989. ISSN 0893-6080. doi: https://doi.org/10.1016/0893-6080(89)90020-8. URL http://www.sciencedirect.com/science/article/pii/0893608089900208.

[31] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85 – 117, 2015. ISSN 0893-6080. doi: 10.1016/j.neunet.2014.09.003.

[32] Gang Wang, Jinxing Hao, Jian Ma, and Lihua Huang. A new approach to intrusion detection using artificial neural networks and fuzzy clustering. *Expert Systems with Applications*, 37(9):6225 – 6232, 2010. ISSN 0957-4174. doi: 10.1016/j.eswa.2010.02.102.

[33] Emilio Corchado and Álvaro Herrero. Neural visualization of network traffic data for intrusion detection. *Applied Soft Computing*, 11(2):2042 – 2056, 2011. ISSN 1568-4946. doi: https://doi.org/10.1016/j.asoc.2010.07.002. URL http://www.sciencedirect.com/science/article/pii/S1568494610001559. The Impact of Soft Computing for the Progress of Artificial Intelligence.

[34] J. Shun and H. A. Malki. Network intrusion detection system using neural networks. In *2008 Fourth International Conference on Natural Computation*, volume 5, pages 242–246, Oct 2008. doi: 10.1109/ICNC.2008.900.

[35] C. Yin, Y. Zhu, J. Fei, and X. He. A deep learning approach for intrusion detection using recurrent neural networks. *IEEE Access*, 5:21954–21961, 2017. ISSN 2169-3536. doi: 10.1109/ACCESS.2017.2762418.

[36] Ahmad Javaid, Quamar Niyaz, Weiqing Sun, and Mansoor Alam. A deep learning approach for network intrusion detection system. In *Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies (Formerly BIONETICS)*, BICT'15, pages 21–26. ICST, 2016. ISBN 978-1-63190-100-3. doi: 10.4108/eai.3-12-2015.2262516.

[37] Lorenzo Fernández Maimó, Alberto Huertas Celdrán, Manuel Gil Pérez, Félix J. García Clemente, and Gregorio Martínez Pérez. Dynamic management of a deep learning-based anomaly detection system for 5G networks. *Journal of Ambient Intelligence and Humanized Computing*, May 2018. ISSN 1868-5145. doi: 10.1007/s12652-018-0813-4.

[38] N. Shone, T. N. Ngoc, V. D. Phai, and Q. Shi. A deep learning approach to network intrusion detection. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 2(1):41–50, Feb 2018. doi: 10.1109/TETCI.2017.2772792.

[39] Sanjiban Sekhar Roy, Abhinav Mallik, Rishab Gulati, Mohammad S. Obaidat, and P. V. Krishna. A deep learning based artificial neural network approach for intrusion detection. In Debasis Giri, Ram N. Mohapatra, Heinrich Begehr, and Mohammad S. Obaidat, editors, *Mathematics and Computing*, pages 44–53, Singapore, 2017. Springer Singapore. ISBN 978-981-10-4642-1.

[40] J. Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993. ISBN 1-55860-238-0.

[41] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, Mar 1986. ISSN 1573-0565. doi: 10.1007/BF00116251. URL https://doi.org/10.1007/BF00116251.

[42] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Taylor & Francis Group, New York, 1984. ISBN 9781315139470. doi: https://doi.org/10.1201/9781315139470.

[43] Nitesh V. Chawla. C4.5 and imbalanced data sets: investigating the effect of sampling method, probabilistic estimate, and decision tree structure. In *In Proceedings of the ICML'03 Workshop on Class Imbalances*, 2003.

[44] Siva S. Sivatha Sindhu, S. Geetha, and A. Kannan. Decision tree based light weight intrusion detection using a wrapper approach. *Expert Systems with Applications*, 39(1):129 – 141, 2012. ISSN 0957-4174. doi: 10.1016/j.eswa.2011.06.013.

[45] Gary Stein, Bing Chen, Annie S. Wu, and Kien A. Hua. Decision tree classifier for network intrusion detection with ga-based feature selection. In *Proceedings of the 43rd Annual Southeast Regional Conference - Volume 2*, ACM-SE 43, pages 136–141, New York, NY, USA, 2005. ACM. ISBN 1-59593-059-0. doi: 10.1145/1167253.1167288.

[46] Ludmila I. Kuncheva and Christopher J. Whitaker. Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. *Machine Learning*, 51(2):181–207, May 2003. ISSN 1573-0565. doi: 10.1023/A:1022859003006. URL https://doi.org/10.1023/A:1022859003006.

[47] Hamed R. Bonab and Fazli Can. A theoretical framework on the ideal number of classifiers for online ensembles in data streams. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, CIKM '16, pages 2053–2056, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-4073-1. doi: 10.1145/2983323.2983907. URL http://doi.acm.org/10.1145/2983323.2983907.

[48] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, Oct 2001. ISSN 1573-0565. doi: 10.1023/A:1010933404324. URL https://doi.org/10.1023/A:1010933404324.

[49] Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119 – 139, 1997. ISSN 0022-0000. doi: https://doi.org/10.1006/jcss.1997.1504. URL http://www.sciencedirect.com/science/article/pii/S002200009791504X.

[50] Abdulla Amin Aburomman and Mamun Bin Ibne Reaz. A novel svm-knn-pso ensemble method for intrusion detection system. *Applied Soft Computing*, 38:360 – 372, 2016. ISSN 1568-4946. doi: https://doi.org/10.1016/j.asoc.2015.10.011. URL http://www.sciencedirect.com/science/article/pii/S1568494615006328.

[51] J. Zhang, M. Zulkernine, and A. Haque. Random-forests-based network intrusion detection systems. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 38(5):649–659, Sep. 2008. ISSN 1094-6977. doi: 10.1109/TSMCC.2008.923876.

[52] F. Gharibian and A. A. Ghorbani. Comparative study of supervised machine learning techniques for intrusion detection. In *Fifth Annual Conference on Communication Networks and Services Research (CNSR '07)*, pages 350–358, May 2007. doi: 10.1109/CNSR.2007.22.

[53] Srinivas Mukkamala, Andrew H. Sung, and Ajith Abraham. Intrusion detection using an ensemble of intelligent paradigms. *Journal of Network and Computer Applications*, 28(2):167 – 182, 2005. ISSN 1084-8045. doi: https://doi.org/10.1016/j.jnca.2004.01.003. URL http://www.sciencedirect.com/science/article/pii/S1084804504000049. Computational Intelligence on the Internet.

[54] Thomas Back, David B. Fogel, and Zbigniew Michalewicz, editors. *Handbook of Evolutionary Computation*. IOP Publishing Ltd., Bristol, UK, UK, 1st edition, 1997. ISBN 0750303921.

[55] Charles Darwin. *The Origin of Species: By Means of Natural Selection Or the Preservation of Favored Races in the Struggle for Life*, volume 1. Modern library, 1872.

[56] Melanie Mitchell. *An Introduction to Genetic Algorithms*. MIT Press, Cambridge, MA, USA, 1998. ISBN 0262631857.

[57] Kenneth De Jong. Evolutionary computation: A unified approach. In *Proceedings of the 14th Annual Conference Companion on Genetic and Evolutionary Computation*, GECCO '12, pages 737–750, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1178-6. doi: 10.1145/2330784.2330914.

[58] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, Sep 1995. ISSN 1573-0565. doi: 10.1007/BF00994018. URL https://doi.org/10.1007/BF00994018.

[59] John Shawe-Taylor and Nello Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, New York, NY, USA, 2004. ISBN 0521813972.

[60] Zhi-Qiang Zeng, Hong-Bin Yu, Hua-Rong Xu, Yan-Qi Xie, and Ji Gao. Fast training support vector machines using parallel sequential minimal optimization. In *2008 3rd International Conference on Intelligent System and Knowledge Engineering*, volume 1, pages 997–1001, Nov 2008. doi: 10.1109/ISKE.2008.4731075.

[61] Chih-Wei Hsu and Chih-Jen Lin. A comparison of methods for multiclass support vector machines. *Trans. Neur. Netw.*, 13(2):415–425, March 2002. ISSN 1045-9227. doi: 10.1109/72.991427. URL https://doi.org/10.1109/72.991427.

[62] Trevor Hastie and Robert Tibshirani. Classification by pairwise coupling. In *Proceedings of the 1997 Conference on Advances in Neural Information Processing Systems 10*, NIPS '97, pages 507–513, Cambridge, MA, USA, 1998. MIT Press. ISBN 0-262-10076-2.

[63] Latifur Khan, Mamoun Awad, and Bhavani Thuraisingham. A new intrusion detection system using support vector machines and hierarchical clustering. *The VLDB Journal*, 16(4):507–521, Oct 2007. doi: 10.1007/s00778-006-0002-5. URL https://doi.org/10.1007/s00778-006-0002-5.

[64] Shi-Jinn Horng, Ming-Yang Su, Yuan-Hsin Chen, Tzong-Wann Kao, Rong-Jian Chen, Jui-Lin Lai, and Citra Dwi Perkasa. A novel intrusion detection system based on hierarchical clustering and support vector machines. *Expert Systems with Applications*, 38(1):306 – 313, 2011. ISSN 0957-4174. doi: 10.1016/j.eswa.2010.06.066.

[65] M.R. Gauthama Raman, Nivethitha Somu, Kannan Kirthivasan, Ramiro Liscano, and V.S. Shankar Sriram. An efficient intrusion detection system based on hypergraph - genetic algorithm for parameter optimization and feature selection in support vector machine. *Knowledge-Based Systems*, 134:1 – 12, 2017. ISSN 0950-7051. doi: https://doi.org/10.1016/j.knosys.2017.07.005. URL http://www.sciencedirect.com/science/article/pii/S0950705117303209.

[66] Gerardo Beni and Jing Wang. Swarm intelligence in cellular robotic systems. In Paolo Dario, Giulio Sandini, and Patrick Aebischer, editors, *Robots and Biological Systems: Towards a New Bionics?*, pages 703–712, Berlin, Heidelberg, 1993. Springer Berlin Heidelberg.

[67] Constantinos Kolias, Georgios Kambourakis, and M. Maragoudakis. Swarm intelligence in intrusion detection: A survey. *Computers & Security*, 30(8):625–642, 2011. doi: 10.1016/j.cose.2011.08.009. URL http://dx.doi.org/10.1016/j.cose.2011.08.009.

[68] M. Dorigo, V. Maniezzo, and A. Colorni. Ant system: optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 26(1):29–41, Feb 1996. ISSN 1083-4419. doi: 10.1109/3477.484436.

[69] J. Kennedy and R. Eberhart. Particle swarm optimization. In *Proceedings of ICNN'95 - International Conference on Neural Networks*, volume 4, pages 1942–1948 vol.4, Nov 1995. doi: 10.1109/ICNN.1995.488968.

[70] Constantinos Kolias, Vasilis Kolias, and Georgios Kambourakis. TermID: a distributed swarm intelligence-based approach for wireless intrusion detection. *International Journal of Information Security*, 16(4):401–416, Aug 2017. ISSN 1615-5270. doi: 10.1007/s10207-016-0335-z.

[71] J. He, D. Long, and C. Chen. An improved ant-based classifier for intrusion detection. In *Third International Conference on Natural Computation (ICNC 2007)*, volume 4, pages 819–823, Aug 2007. doi: 10.1109/ICNC.2007.206.

[72] R. S. Parpinelli, H. S. Lopes, and A. A. Freitas. Data mining with an ant colony optimization algorithm. *IEEE Transactions on Evolutionary Computation*, 6(4): 321–332, Aug 2002. ISSN 1089-778X. doi: 10.1109/TEVC.2002.802452.

[73] Chi-Ho Tsang and Sam Kwong. *Ant Colony Clustering and Feature Extraction for Anomaly Intrusion Detection*, pages 101–123. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006. ISBN 978-3-540-34956-3. doi: 10.1007/978-3-540-34956-3_5. URL https://doi.org/10.1007/978-3-540-34956-3_5.

[74] Yinhui Li, Jingbo Xia, Silan Zhang, Jiakai Yan, Xiaochuan Ai, and Kuobin Dai. An efficient intrusion detection system based on support vector machines and gradually feature removal method. *Expert Systems with Applications*, 39(1):424 – 430, 2012. ISSN 0957-4174. doi: https://doi.org/10.1016/j.eswa.2011.07.032. URL http://www.sciencedirect.com/science/article/pii/S0957417411009948.

[75] Craig W. Reynolds. Flocks, herds and schools: A distributed behavioral model. *SIGGRAPH Comput. Graph.*, 21(4):25–34, August 1987. ISSN 0097-8930. doi: 10.1145/37402.37406. URL http://doi.acm.org/10.1145/37402.37406.

[76] Yuk Ying Chung and Noorhaniza Wahid. A hybrid network intrusion detection system using simplified swarm optimization (sso). *Applied Soft Computing*, 12(9):3014 – 3022, 2012. ISSN 1568-4946. doi: https://doi.org/10.1016/j.asoc.2012.04.020. URL http://www.sciencedirect.com/science/article/pii/S1568494612002116.

[77] J. Ma, X. Liu, and S. Liu. A new intrusion detection method based on bpso-svm. In *2008 International Symposium on Computational Intelligence and Design*, volume 1, pages 473–477, Oct 2008. doi: 10.1109/ISCID.2008.65.

[78] Seyed Mojtaba Hosseini Bamakan, Huadong Wang, Tian Yingjie, and Yong Shi. An effective intrusion detection framework based on mclp/svm optimized by time-varying chaos particle swarm optimization. *Neurocomputing*, 199:90 – 102, 2016. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2016.03.031. URL http://www.sciencedirect.com/science/article/pii/S0925231216300510.

[79] Lingyu Wang, Sushil Jajodia, Anoop Singhal, and Steven Noel. K-zero day safety: Measuring the security risk of networks against unknown attacks. In *Proceedings of the 15th European Conference on Research in Computer Security*, ESORICS'10, pages 573–587, Berlin, Heidelberg, 2010. Springer-Verlag. ISBN 3-642-15496-4, 978-3-642-15496-6.

[80] Aleksandr Lenin, Jan Willemson, and Dyan Permata Sari. *Attacker Profiling in Quantitative Security Assessment Based on Attack Trees*, pages 199–212. Springer International Publishing, Cham, 2014. ISBN 978-3-319-11599-3. doi: 10.1007/978-3-319-11599-3_12.

[81] Symantec. Internet security threat report — government. https://www.symantec.com/content/dam/symantec/docs/reports/istr-21-2016-government-en.pdf, April 2016. Accessed: 2016-09-21.

[82] Salvatore Greco, J Figueira, and M Ehrgott. Multiple criteria decision analysis: State of the art surveys. *Springer's International series*, 2005. doi: 10.1007/978-1-4939-3094-4.

[83] Andrew Fielder, Emmanouil Panaousis, Pasquale Malacaria, Chris Hankin, and Fabrizio Smeraldi. Decision support approaches for cyber security investment. *Decision Support Systems*, 86:13 – 23, 2016. ISSN 0167-9236. doi: http://dx.doi.org/10.1016/j.dss.2016.02.012.

[84] Bruce Schneier. Attack trees. *j-DDJ*, 24(12):21–22, 24, 26, 28–29, December 1999. ISSN 1044-789X.

[85] Daniel O. Díaz López, Ginés Dólera Tormo, Félix Gómez Mármol, and Gregorio Martínez Pérez. Dynamic counter-measures for risk-based access control systems: an evolutive approach. *Future Generation Computer Systems*, 55:321–335, 2016. doi: 10.1016/j.future.2014.10.012. URL http://dx.doi.org/10.1016/j.future.2014.10.012.

[86] Prahlad Fogla, Monirul Sharif, Roberto Perdisci, Oleg Kolesnikov, and Wenke Lee. Polymorphic blending attacks. In *Proceedings of the 15th Conference on USENIX Security Symposium - Volume 15*, USENIX-SS'06, Berkeley, CA, USA, 2006. USENIX Association.

[87] Rinku Dewri, Nayot Poolsappasit, Indrajit Ray, and Darrell Whitley. Optimal security hardening using multi-objective optimization on attack tree models of networks. In *Proceedings of the 14th ACM Conference on Computer and Communications Security*, CCS '07, pages 204–213, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-703-2.

[88] CVE, common vulnerabilities and exposures. https://cve.mitre.org/. Accessed: 2016-08-18.

[89] AlienVault Open Threat Exchange (OTX). https://www.alienvault.com/open-threat-exchange. Accessed: 2016-09-14.

[90] Archimate, architecture description language. http://www.opengroup.org/subjectareas/enterprise/archimate. Accessed: 2016-09-23.

[91] Paul C. Clements. A survey of architecture description languages. In *Proceedings of the 8th International Workshop on Software Specification and Design*, IWSSD '96, pages 16–, Washington, DC, USA, 1996. IEEE Computer Society. ISBN 0-8186-7361-3.

[92] Andreas Schilling and Brigitte Werners. Optimal selection of it security safeguards from an existing knowledge base. *European Journal of Operational Research*, 248 (1):318–327, 2016.

[93] Karel Durkota, Viliam Lisy, Christofer Kiekintveld, and Branislav Bosansky. Game-theoretic algorithms for optimal network security hardening using attack graphs. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, pages 1773–1774. International Foundation for Autonomous Agents and Multiagent Systems, 2015.

[94] Barbara Kordy, Ludovic Piétre-Cambacédés, and Patrick Schweitzer. DAG-based attack and defense modeling: Don't miss the forest for the attack trees. *Computer Science Review*, 13-14:1 – 38, 2014. ISSN 1574-0137. doi: http://dx.doi.org/10.1016/j.cosrev.2014.07.001.

[95] Xinming Ou, Wayne F. Boyer, and Miles A. McQueen. A scalable approach to attack graph generation. In *Proceedings of the 13th ACM Conference on Computer and Communications Security*, CCS '06, pages 336–345, New York, NY, USA, 2006. ACM. ISBN 1-59593-518-5. doi: 10.1145/1180405.1180446.

[96] K. Ingols, M. Chu, R. Lippmann, S. Webster, and S. Boyer. Modeling modern network attacks and countermeasures using attack graphs. In *Computer Security Applications Conference, 2009. ACSAC '09. Annual*, pages 117–126, Dec 2009. doi: 10.1109/ACSAC.2009.21.

[97] Xinming Ou, Sudhakar Govindavajhala, and Andrew W. Appel. Mulval: A logic-based network security analyzer. In *Proceedings of the 14th Conference on USENIX Security Symposium - Volume 14*, SSYM'05, pages 8–8, Berkeley, CA, USA, 2005. USENIX Association.

[98] N. Poolsappasit, R. Dewri, and I. Ray. Dynamic security risk management using bayesian attack graphs. *IEEE Transactions on Dependable and Secure Computing*, 9(1):61–74, Jan 2012. ISSN 1545-5971. doi: 10.1109/TDSC.2011.34.

[99] Arpan Roy, Dong Seong Kim, and Kishor S. Trivedi. Attack countermeasure trees (ACT): towards unifying the constructs of attack and defense trees. *Security and Communication Networks*, 5(8):929–943, 2012. ISSN 1939-0122. doi: 10.1002/sec.299.

[100] W. H. Sanders. RRE: A game-theoretic intrusion response and recovery engine for process control applications. In *Critical Infrastructures, 2009. CRIS 2009. Fourth International Conference on*, pages 1–1, March 2009. doi: 10.1109/CRIS.2009.5071485.

[101] Barbara Kordy, Piotr Kordy, Sjouke Mauw, and Patrick Schweitzer. *ADTool: Security Analysis with Attack–Defense Trees*, pages 173–176. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013. ISBN 978-3-642-40196-1. doi: 10.1007/978-3-642-40196-1_15.

[102] Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1):99 – 134, 1998. ISSN 0004-3702. doi: http://dx.doi.org/10.1016/S0004-3702(98)00023-X. URL http://www.sciencedirect.com/science/article/pii/S000437029800023X.

[103] Nizar Kheir, Nora Cuppens-Boulahia, Frédéric Cuppens, and Hervé Debar. *A Service Dependency Model for Cost-Sensitive Intrusion Response*, pages 626–642. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010. ISBN 978-3-642-15497-3. doi: 10.1007/978-3-642-15497-3_38.

[104] Alireza Shameli-Sendi and Michel Dagenais. ORCEF: Online response cost evaluation framework for intrusion response system. *Journal of Network and Computer*

Applications, 55:89 – 107, 2015. ISSN 1084-8045. doi: http://dx.doi.org/10.1016/j.jnca.2015.05.004.

[105] Richard Bellman. A markovian decision process. *Indiana University Mathematics Journal*, 6:679–684, 1957. ISSN 0022-2518.

[106] John G Kemeny, James Laurie Snell, et al. *Finite markov chains*, volume 356. van Nostrand Princeton, NJ, 1960.

[107] S. Jha, O. Sheyner, and J. Wing. Two formal analyses of attack graphs. In *Computer Security Foundations Workshop, 2002. Proceedings. 15th IEEE*, pages 49–63, 2002. doi: 10.1109/CSFW.2002.1021806.

[108] Jerzy Filar and Koos Vrieze. *Competitive Markov Decision Processes*. Springer-Verlag New York, Inc., New York, NY, USA, 1996. ISBN 0-387-94805-8.

[109] M. H. A. Davis and P. Varaiya. Dynamic programming conditions for partially observable stochastic systems. *SIAM Journal on Control*, 11(2):226–261, 1973. doi: 10.1137/0311020.

[110] Barbara Kordy, Sjouke Mauw, and Patrick Schweitzer. Foundations of attack-defense trees. *Formal Aspects in Security and Trust*, 6561:80–95.

[111] Elazar J Pedhazur and Liora Pedhazur Schmelkin. *Measurement, design, and analysis: An integrated approach*. Psychology Press, 2013.

[112] Andrew Jaquith. *Security metrics*. Pearson Education, 2007.

[113] Sanghamitra Bandyopadhyay and Sriparna Saha. *Some Single- and Multiobjective Optimization Techniques*, pages 17–58. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013. ISBN 978-3-642-32451-2. doi: 10.1007/978-3-642-32451-2_2.

[114] R. Dawkins and J. R. Krebs. Arms races between and within species. *Proceedings of the Royal Society of London B: Biological Sciences*, 205(1161):489–511, 1979. ISSN 0080-4649. doi: 10.1098/rspb.1979.0081.

[115] Fred Glover and Manuel Laguna. Principles of tabu search. 2007.

[116] Dimitri P. Bertsekas. *Dynamic Programming and Optimal Control, Vol. II*. Athena Scientific, 3rd edition, 2007. ISBN 1886529302, 9781886529304.

[117] R.S. Sutton and A.G. Barto. *Reinforcement learning: An introduction*, volume 116. Cambridge Univ Press, 1998.

[118] J.S. Magdych, T. Rahmanovic, J.R. McDonald, B.E. Tellier, A.C. Osborne, and N.P. Herath. Network-based risk-assessment tool for remotely detecting local computer vulnerabilities, August 22 2006. US Patent 7,096,503.

[119] David Maynor. *Metasploit toolkit for penetration testing, exploit development, and vulnerability research*. Elsevier, 2011.

[120] CVRF, common vulnerability report format. http://www.icasi.org/cvrf/. Accessed: 2016-08-18.

[121] NVD, national vulnerability database. https://nvd.nist.gov/. Accessed: 2016-08-23.

[122] OVAL, open vulnerability and assessment language. https://oval.mitre.org/. Accessed: 2016-08-22.

[123] CCE, common configuration enumeration. https://cce.mitre.org/. Accessed: 2016-08-18.

[124] Karen Scarfone and Peter Mell. The common configuration scoring system (CCSS): Metrics for software security configuration vulnerabilities. *NIST Interagency/Internal Report (NISTIR) - 7502*, 2010.

[125] Common platform enumeration. https://cpe.mitre.org/. Accessed: 2016-08-18.

[126] ASR, asset summary report. https://scap.nist.gov/specifications/asr/. Accessed: 2016-08-22.

[127] CWE, common weakness enumeration. https://cwe.mitre.org/. Accessed: 2016-08-18.

[128] CWSS, common weakness scoring system. https://cwe.mitre.org/cwss/cwss_v1.0.1.html. Accessed: 2016-08-18.

[129] Elizabeth LeMay, Karen Scarfone, and Peter Mell. The common misuse scoring system (CMSS): Metrics for software feature misuse vulnerabilities. *NIST Interagency/Internal Report (NISTIR) - 7864*, 2012.

[130] CWRAF, common weakness risk analysis framework. https://cwe.mitre.org/cwraf/. Accessed: 2016-08-18.

[131] CAPEC, common attack pattern enumeration and classification. https://capec.mitre.org/. Accessed: 2016-08-18.

[132] CRE, common remediation enumeration. https://scap.nist.gov/specifications/cre/. Accessed: 2016-08-18.

[133] Waltermire David, Johnson Christopher, Kerr Matthew, Wojcik Matthew, and Wunder John. Proposed open specifications for an enterprise remediation automation framework (draft). *NIST Interagency/Internal Report (NISTIR) - 7670*, 2011.

[134] Hervé Debar, David A Curry, and Benjamin S Feinstein. IDMEF, intrusion detection message exchange format. 2007.

[135] TMSAD, trust model for security automation data. https://scap.nist.gov/specifications/tmsad/. Accessed: 2016-08-22.

[136] OpenIOC, open indicator of compromise. http://www.openioc.org/, . Accessed: 2016-08-22.

[137] STIX, structured threat information eXpression. https://stixproject.github.io/. Accessed: 2016-08-22.

[138] TAXII, trusted automated exchange of indicator information. https://taxiiproject.github.io/. Accessed: 2016-08-22.

[139] CybOX, cyber observable expression. https://cyboxproject.github.io/. Accessed: 2016-08-22.

[140] XCCDF, eXtensible configuration checklist description format. https://scap.nist.gov/specifications/xccdf/. Accessed: 2016-08-22.

[141] Roman Danyliw, Jan Meijer, and Yuri Demchenko. IODEF, the incident object description exchange format. 2007.

[142] MAEC, malware attribute enumeration and characterization. https://maecproject.github.io/. Accessed: 2016-08-22.

[143] Common vulnerabilities scoring system. https://www.first.org/cvss. Accessed: 2016-08-18.

[144] M. Keramati, A. Akbari, and M. Keramati. CVSS-based security metrics for quantitative analysis of attack graphs. In *Computer and Knowledge Engineering (ICCKE), 2013 3th International eConference on*, pages 178–183, Oct 2013. doi: 10.1109/ICCKE.2013.6682816.

[145] Shuzhen Wang, Zonghua Zhang, and Youki Kadobayashi. Exploring attack graph for cost-benefit security hardening: A probabilistic approach. *Computers & Security*, 32:158 – 169, 2013. ISSN 0167-4048. doi: http://dx.doi.org/10.1016/j.cose.2012.09.013.

[146] SCAP, security content automation protocol. https://scap.nist.gov/. Accessed: 2016-08-22.

[147] Mitre corporation. https://www.mitre.org/. Accessed: 2016-08-22.

[148] Mica R Endsley. Level of automation effects on performance, situation awareness and workload in a dynamic control task. *Ergonomics*, 42(3):462–492, 1999.

[149] Hassan Gomaa. *Software modeling and design: UML, use cases, patterns, and software architectures*. Cambridge University Press, 2011.

[150] Alessandro Sforzin, Félix Gómez Mármol, Mauro Conti, and Jens-Matthias Bohli. Raspberry Pi IDS: A fruitful Intrusion Detection System for IoT. In *13th IEEE International Conference on Advanced and Trusted Computing (ATC 2016)*, pages –, Toulouse, France, 2016.

[151] André B. Bondi. Characteristics of scalability and their impact on performance. In *Proceedings of the 2Nd International Workshop on Software and Performance*, WOSP '00, pages 195–203, New York, NY, USA, 2000. ACM. ISBN 1-58113-195-X. doi: 10.1145/350391.350432.

[152] Michael Sipser. *Introduction to the Theory of Computation; 2nd ed.* Thomson Course Technology, Cambridge, 2006.

[153] Rinku Dewri, Indrajit Ray, Nayot Poolsappasit, and Darrell Whitley. Optimal security hardening on attack tree models of networks: a cost-benefit analysis. *International Journal of Information Security*, 11(3):167–188, 2012. ISSN 1615-5270. doi: 10.1007/s10207-012-0160-y.

[154] BugTraq, security focus mailing list. http://www.securityfocus.com/archive. Accessed: 2017-07-04.

[155] CERT/CC, cert coordination center. http://www.cert.org/. Accessed: 2017-07-04.

[156] Jan Kanclirz. *Netcat power tools*. Syngress, 2008.

[157] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6 (2):182–197, Apr 2002. ISSN 1089-778X. doi: 10.1109/4235.996017.

[158] S. A. Butler. Security attribute evaluation method: a cost-benefit approach. In *Software Engineering, 2002. ICSE 2002. Proceedings of the 24rd International Conference on*, pages 232–240, May 2002. doi: 10.1109/ICSE.2002.1007971.

[159] Shawn A. Butler and Paul Fischbeck. Multi-attribute risk assessment. In *Proceedings of the 3rd Symposium on Requirements Engineering for Information Security*, CERIAS '02, pages 2:1–2:12, West Lafayette, IN, 2002. CERIAS - Purdue University.

[160] Arpan Roy, Dong Seong Kim, and Kishor S. Trivedi. Scalable optimal countermeasure selection using implicit enumeration on attack countermeasure trees. In *Proceedings of the 2012 42Nd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, DSN '12, pages 1–12, Washington, DC, USA, 2012. IEEE Computer Society. ISBN 978-1-4673-1624-8.

[161] S. A. Zonouz, H. Khurana, W. H. Sanders, and T. M. Yardley. RRE: A game-theoretic intrusion response and recovery engine. *IEEE Transactions on Parallel and Distributed Systems*, 25(2):395–406, Feb 2014. ISSN 1045-9219. doi: 10.1109/TPDS.2013.211.

[162] Lawrence J. Watters. Reduction of integer polynomial programming problems to zero-one linear programming problems. *Operations Research*, 15(6):1171–1174, 1967. ISSN 0030364X, 15265463.

[163] Valentina Viduto, Carsten Maple, Wei Huang, and David López-Peréz. A novel risk assessment and optimisation model for a multi-objective network security countermeasure selection problem. *Decis. Support Syst.*, 53(3):599–610, June 2012. ISSN 0167-9236. doi: 10.1016/j.dss.2012.04.001.

[164] Gary Stoneburner, Alice Y. Goguen, and Alexis Feringa. SP 800-30. risk management guide for information technology systems. Technical report, Gaithersburg, MD, United States, 2002.

[165] PTA, a practical threat analysis case study: next generation call accounting. http://www.ptatechnologies.com/Documents/TACSCase.pdf. Accessed: 2016-09-09.

[166] Robert H Anderson, Phillip M Feldman, Scott Gerwehr, Brian Houghton, and Richard Mesic. Securing the us defense information infrastructure: A proposed approach. Technical report, DTIC Document, 1999.

[167] Mukul Gupta, Jackie Rees, Alok Chaturvedi, and Jie Chi. Matching information security vulnerabilities to organizational security profiles: A genetic algorithm approach. *Decis. Support Syst.*, 41(3):592–603, March 2006. ISSN 0167-9236. doi: 10.1016/j.dss.2004.06.004.

[168] R. Sarala, G. Zayaraz, and V. Vijayalakshmi. *Optimal Selection of Security Countermeasures for Effective Information Security*, pages 345–353. Springer India, New Delhi, 2016. ISBN 978-81-322-2674-1. doi: 10.1007/978-81-322-2674-1_33.

[169] C. J. Chung, P. Khatkar, T. Xing, J. Lee, and D. Huang. NICE: Network intrusion detection and countermeasure selection in virtual network systems. *IEEE Transactions on Dependable and Secure Computing*, 10(4):198–211, July 2013. ISSN 1545-5971.

[170] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, and Matei Zaharia. A view of cloud computing. *Commun. ACM*, 53(4):50–58, April 2010. ISSN 0001-0782.

[171] Open Networking Fundation. Software-defined networking: The new norm for networks. *ONF White Paper*, 2012.

[172] Openflow. https://www.opennetworking.org/sdn-resources/openflow, . Accessed: 2016-08-25.

[173] Guofei Gu, Junjie Zhang, and Wenke Lee. Botsniffer: Detecting botnet command and control channels in network traffic. 2008.

[174] M. Tupper and A. N. Zincir-Heywood. VEA-bility security metric: A network security analysis tool. In *Availability, Reliability and Security, 2008. ARES 08. Third International Conference on*, pages 950–957, March 2008.

[175] Diego Perez-Botero, Jakub Szefer, and Ruby B. Lee. Characterizing hypervisor vulnerabilities in cloud computing servers. In *Proceedings of the 2013 International Workshop on Security in Cloud Computing*, Cloud Computing '13, pages 3–10, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-2067-2.

[176] Saman Zonouz and Parisa Haghani. Cyber-physical security metric inference in smart grid critical infrastructures based on system administrators' responsive behavior. *Computers & Security*, 39, Part B:190 – 200, 2013. ISSN 0167-4048. doi: http://dx.doi.org/10.1016/j.cose.2013.07.003.

[177] Richard Bellman. *Dynamic Programming*. Princeton University Press, Princeton, NJ, USA, 1 edition, 1957.

[178] C. E. Shannon. A mathematical theory of communication. *SIGMOBILE Mob. Comput. Commun. Rev.*, 5(1):3–55, January 2001. ISSN 1559-1662. doi: 10.1145/584091.584093.

[179] Guillermo Owen. *Game theory*. Emerald Group Publishing Limited, 3 edition, 1995. ISBN 978-0125311519.

[180] Shi-Jay Chen and Shyi-Ming Chen. Fuzzy risk analysis based on similarity measures of generalized fuzzy numbers. *IEEE Transactions on Fuzzy Systems*, 11(1): 45–56, Feb 2003. ISSN 1063-6706. doi: 10.1109/TFUZZ.2002.806316.

[181] Félix Gómez Mármol and Gregorio Martínez Pérez. Security threats scenarios in trust and reputation models for distributed systems. *Computers & Security*, 28 (7):545 – 556, 2009. ISSN 0167-4048. doi: http://dx.doi.org/10.1016/j.cose.2009.05.005.

[182] G. Gonzalez-Granadillo, J. Garcia-Alfaro, E. Alvarez, M. El-Barbori, and H. Debar. Selecting optimal countermeasures for attacks against critical systems using the attack volume model and the RORI index. *Comput. Electr. Eng.*, 47(C):13–34, October 2015. ISSN 0045-7906.

[183] Gustavo Gonzalez Granadillo, Hervé Débar, Grégoire Jacob, Chrystel Gaber, and Mohammed Achemlal. Individual countermeasure selection based on the return on response investment index. In *Proceedings of the 6th International Conference on Mathematical Methods, Models and Architectures for Computer Network Security: Computer Network Security*, MMM-ACNS'12, pages 156–170, Berlin, Heidelberg, 2012. Springer-Verlag. ISBN 978-3-642-33703-1.

[184] G. G. Granadillo, G. Jacob, H. Debar, and L. Coppolino. Combination approach to select optimal countermeasures based on the rori index. In *Innovative Computing Technology (INTECH), 2012 Second International Conference on*, pages 38–45, Sept 2012.

[185] P. K. Manadhata and J. M. Wing. An attack surface metric. *IEEE Transactions on Software Engineering*, 37(3):371–386, May 2011. ISSN 0098-5589. doi: 10.1109/TSE.2010.60.

[186] Ravi S. Sandhu, Edward J. Coyne, Hal L. Feinstein, and Charles E. Youman. Role-based access control models. *Computer*, 29(2):38–47, February 1996. ISSN 0018-9162. doi: 10.1109/2.485845.

[187] A. A. E. Kalam, R. E. Baida, P. Balbiani, S. Benferhat, F. Cuppens, Y. Deswarte, A. Miege, C. Saurel, and G. Trouessin. Organization based access control. In *Policies for Distributed Systems and Networks, 2003. Proceedings. POLICY 2003. IEEE 4th International Workshop on*, pages 120–131, June 2003. doi: 10.1109/POLICY.2003.1206966.

[188] Federation of american scientists, special operations forces intelligence and electronic warfare operations, appendix d: Target analysis process, 1991. http://www.fas.org/irp/doddir/army/fm34-36/appd.htm.

[189] Gustavo Gonzalez Granadillo, Malek Belhaouane, Hervé Debar, and Grégoire Jacob. Rori-based countermeasure selection using the orbac formalism. *International journal of information security*, 13(1):63–79, 2014.

[190] Laura P Swiler, Thomas L Paez, and Randall L Mayes. Epistemic uncertainty quantification tutorial. In *Proceedings of the 27th International Modal Analysis Conference*, 2009.

[191] Erik Miehling, Mohammad Rasouli, and Demosthenis Teneketzis. Optimal defense policies for partially observable spreading processes on bayesian attack graphs. In *Proceedings of the Second ACM Workshop on Moving Target Defense*, MTD '15, pages 67–76, New York, NY, USA, 2015. ACM. ISBN 978-1-4503-3823-3. doi: 10.1145/2808475.2808482.

[192] Department of homeland security. moving target defense. https://www.dhs.gov/science-and-technology/csd-mtd. Accessed: 2016-12-01.

[193] Ching-Lai Hwang and Kwangsun Yoon. *Methods for Multiple Attribute Decision Making*, pages 58–191. Springer Berlin Heidelberg, Berlin, Heidelberg, 1981. ISBN 978-3-642-48318-9. doi: 10.1007/978-3-642-48318-9_3.

[194] R.T. Marler and J.S. Arora. Survey of multi-objective optimization methods for engineering. *Structural and Multidisciplinary Optimization*, 26(6):369–395, Apr 2004. ISSN 1615-1488. doi: 10.1007/s00158-003-0368-6. URL https://doi.org/10.1007/s00158-003-0368-6.

[195] Igor Kotenko, Olga Polubelova, Igor Saenko, and Elena Doynikova. The ontology of metrics for security evaluation and decision support in siem systems. In *Availability, Reliability and Security (ARES), 2013 Eighth International Conference on*, pages 638–645. IEEE, 2013.

[196] Igor V Kotenko and Elena Doynikova. Evaluation of computer network security based on attack graphs and security event processing. *JoWUA*, 5(3):14–29, 2014.

[197] Elena Doynikova and Igor Kotenko. Countermeasure selection based on the attack and service dependency graphs for security incident management. In *International Conference on Risks and Security of Internet and Systems*, pages 107–124. Springer, 2015.

[198] Igor Kotenko and Elena Doynikova. Countermeasure selection in siem systems based on the integrated complex of security metrics. In *2015 23rd Euromicro International Conference on Parallel, Distributed, and Network-Based Processing*, pages 567–574. IEEE, 2015.

[199] Igor Kotenko and Elena Doynikova. Dynamical calculation of security metrics for countermeasure selection in computer networks. In *2016 24th Euromicro International Conference on Parallel, Distributed, and Network-Based Processing (PDP)*, pages 558–565. IEEE, 2016.

[200] Igor Kotenko and Andrey Chechulin. Attack modeling and security evaluation in siem systems. *International Transactions on Systems Science and Applications*, 8: 129–147, 2012.

[201] Igor Kotenko and Andrey Chechulin. Common framework for attack modeling and security evaluation in siem systems. In *Green Computing and Communications (GreenCom), 2012 IEEE international conference on*, pages 94–101. IEEE, 2012.

[202] Evgenia Novikova and Igor Kotenko. Analytical visualization techniques for security information and event management. In *2013 21st Euromicro International Conference on Parallel, Distributed, and Network-Based Processing*, pages 519–525. IEEE, 2013.

[203] Igor Kotenko and Andrey Chechulin. A cyber attack modeling and impact assessment framework. In *Cyber Conflict (CyCon), 2013 5th International Conference on*, pages 1–24. IEEE, 2013.

[204] Igor Kotenko and Andrey Chechulin. Computer attack modeling and security evaluation based on attack graphs. In *Intelligent Data Acquisition and Advanced Computing Systems (IDAACS), 2013 IEEE 7th International Conference on*, volume 2, pages 614–619. IEEE, 2013.

[205] Igor Kotenko and Andrey Chechulin. Fast network attack modeling and security evaluation based on attack graphs. *Journal of Cyber Security and Mobility*, 3(1): 27–46, 2014.

[206] Igor Kotenko, Andrey Shorov, Andrey Chechulin, and Evgenia Novikova. Dynamical attack simulation for security information and event management. In *Information Fusion and Geographic Information Systems (IF AND GIS 2013)*, pages 219–234. Springer, 2014.

[207] M. S. Ahmed, E. Al-Shaer, and L. Khan. A novel quantitative approach for measuring network security. In *INFOCOM 2008. The 27th Conference on Computer Communications. IEEE*, April 2008. doi: 10.1109/INFOCOM.2008.260.

[208] Alireza Shameli-Sendi, Habib Louafi, Wenbo He, and Mohamed Cheriet. Dynamic optimal countermeasure selection for intrusion response system. *IEEE Transactions on Dependable and Secure Computing*, 2016.

[209] Eric A. Hansen and Shlomo Zilberstein. LAO*: A heuristic search algorithm that finds solutions with loops. *Artificial Intelligence*, 129(1):35 – 62, 2001. ISSN 0004-3702. doi: http://dx.doi.org/10.1016/S0004-3702(01)00106-0.

[210] Andrew V. Goldberg and Chris Harrelson. Computing the shortest path: A search meets graph theory. In *Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '05, pages 156–165, Philadelphia, PA, USA, 2005. Society for Industrial and Applied Mathematics. ISBN 0-89871-585-7.

[211] IBM. *Cognitive security (white paper)*. 2016.

[212] David Waltermire, Paul Cichonski, and Karen Scarfone. CPE, common platform enumeration: Applicability language specification. *NIST Interagency/Internal Report (NISTIR) - 7698*, 2011.

[213] Manuel Gil Pérez, Félix Gómez Mármol, and Gregorio Martínez Pérez. Improving attack detection in self-organizing networks: A trust-based approach toward alert satisfaction. In *2015 International Conference on Advances in Computing, Communications and Informatics, ICACCI 2015, Kochi, India, August 10-13, 2015*, pages 1945–1951, 2015. doi: 10.1109/ICACCI.2015.7275903.

[214] Manuel Gil Pérez, Félix Gómez Mármol, Gregorio Martínez Pérez, and Antonio F. Skarmeta Gómez. Building a reputation-based bootstrapping mechanism for newcomers in collaborative alert systems. *Journal of Computer and System Sciences*, 80(3):571 – 590, 2014. ISSN 0022-0000. doi: 10.1016/j.jcss.2013.06.012. Special Issue on Wireless Network Intrusion.

[215] Manuel Gil Pérez, Félix Gómez Mármol, Gregorio Martínez Pérez, and Antonio F. Skarmeta Gómez. Repcidn: A reputation-based collaborative intrusion detection network to lessen the impact of malicious alarms. *Journal of Network and Systems Management*, 21(1):128–167, 2013. ISSN 1573-7705. doi: 10.1007/s10922-012-9230-8.

[216] Jiawei Han, Micheline Kamber, and Jian Pei. *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 3rd edition, 2011. ISBN 0123814790, 9780123814791.

[217] Pengfei Guo, Xuezhi Wang, and Yingshi Han. The enhanced genetic algorithms for the optimization design. In *Biomedical Engineering and Informatics (BMEI), 2010 3rd International Conference on*, volume 7, pages 2990–2994. IEEE, 2010.

[218] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. The weka data mining software: An update. *SIGKDD Explor.*

Newsl., 11(1):10–18, November 2009. ISSN 1931-0145. doi: 10.1145/1656274.1656278.

[219] Thomas Bäck. *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*. Oxford University Press, Oxford, UK, 1996. ISBN 0-19-509971-0.

[220] Laurent Hyafil and Ronald L. Rivest. Constructing optimal binary decision trees is np-complete. *Information Processing Letters*, 5(1):15 – 17, 1976. ISSN 0020-0190. doi: 10.1016/0020-0190(76)90095-8.

[221] Usama M. Fayyad and Keki B. Irani. Multi-interval discretization of continuous-valued attributes for classification learning. In *IJCAI*, pages 1022–1029, 1993.

[222] Igor Kononenko. On biases in estimating multi-valued attributes. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'95, pages 1034–1040, San Francisco, CA, USA, 1995. Morgan Kaufmann Publishers Inc. ISBN 1-55860-363-8.

[223] Don Coppersmith, Se June Hong, and Jonathan R.M. Hosking. Partitioning nominal attributes in decision trees. *Data Mining and Knowledge Discovery*, 3(2): 197–217, 1999. ISSN 1573-756X. doi: 10.1023/A:1009869804967.

[224] Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. *J. Mach. Learn. Res.*, 3:1157–1182, March 2003. ISSN 1532-4435.

[225] Ron Kohavi and George H. John. Wrappers for feature subset selection. *Artificial Intelligence*, 97(1):273 – 324, 1997. ISSN 0004-3702. doi: http://dx.doi.org/10.1016/S0004-3702(97)00043-X.

[226] Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. *Introduction to Data Mining, (First Edition)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2005. ISBN 0321321367.

[227] Dewan Md. Farid, Li Zhang, Chowdhury Mofizur Rahman, M.A. Hossain, and Rebecca Strachan. Hybrid decision tree and naïve bayes classifiers for multi-class classification tasks. *Expert Systems with Applications*, 41(4, Part 2):1937 – 1946, 2014. ISSN 0957-4174. doi: 10.1016/j.eswa.2013.08.089.

[228] Safaa O. Al-mamory and Firas S. Jassim. On the designing of two grains levels network intrusion detection system. *Karbala International Journal of Modern Science*, 1(1):15 – 25, 2015. ISSN 2405-609X. doi: 10.1016/j.kijoms.2015.07.002.

[229] Adel Nadjaran Toosi and Mohsen Kahani. A new approach to intrusion detection based on an evolutionary soft computing model using neuro-fuzzy classifiers. *Computer Communications*, 30(10):2201 – 2212, 2007. ISSN 0140-3664. doi: 10.1016/j.comcom.2007.05.002.

[230] J. Gómez, C. Gil, R. Baños, A. L. Márquez, F. G. Montoya, and M. G. Montoya. A pareto-based multi-objective evolutionary algorithm for automatic rule generation in network intrusion detection systems. *Soft Computing*, 17(2):255–263, 2013. ISSN 1433-7479. doi: 10.1007/s00500-012-0890-9.

[231] Fangjun Kuang, Weihong Xu, and Siyang Zhang. A novel hybrid kpca and svm with ga model for intrusion detection. *Applied Soft Computing*, 18(Complete): 178–184, 2014. doi: 10.1016/j.asoc.2014.01.028.

[232] Kapil Kumar Gupta, Baikunth Nath, and Ramamohanarao Kotagiri. Layered approach using conditional random fields for intrusion detection. *IEEE Transactions on Dependable and Secure Computing*, 7(1):35–49, 2010. ISSN 1545-5971. doi: 10.1109/TDSC.2008.20.

[233] K. Veeramachaneni, I. Arnaldo, V. Korrapati, C. Bassias, and K. Li. AI^2: Training a big data machine to defend. In *2016 IEEE 2nd International Conference on Big Data Security on Cloud (BigDataSecurity), IEEE International Conference on High Performance and Smart Computing (HPSC), and IEEE International Conference on Intelligent Data and Security (IDS)*, pages 49–54, April 2016. doi: 10.1109/BigDataSecurity-HPSC-IDS.2016.79.

[234] Mike Duffy for AT&T Cybersecurity. Machine learning myths, September 18, 2017. URL https://www.alienvault.com/blogs/security-essentials/machine-learning-myths.

[235] Hassina Bensefia and Nassira Ghoualmi-Zine. Adaptive intrusion detection systems: The next generation of idss. *Network Security Technologies: Design and Applications: Design and Applications*, page 239, 2013.

[236] Ki-Hyun Lee and Young B. Park. A study of environment-adaptive intrusion detection system. In James J. (Jong Hyuk) Park, Yi Pan, Gangman Yi, and Vincenzo Loia, editors, *Advances in Computer Science and Ubiquitous Computing*, pages 625–630, Singapore, 2017. Springer Singapore. ISBN 978-981-10-3023-9.

[237] Ryan J. Urbanowicz and Jason H. Moore. Learning classifier systems: A complete introduction, review, and roadmap. *J. Artif. Evol. App.*, 2009:1:1–1:25, January 2009. ISSN 1687-6229. doi: 10.1155/2009/736398.

[238] Jungwon Kim, Peter J. Bentley, Uwe Aickelin, Julie Greensmith, Gianni Tedesco, and Jamie Twycross. Immune system approaches to intrusion detection – a review. *Natural Computing*, 6(4):413–466, Dec 2007. ISSN 1572-9796. doi: 10.1007/s11047-006-9026-4.

[239] Rajat Raina, Alexis Battle, Honglak Lee, Benjamin Packer, and Andrew Y. Ng. Self-taught learning: Transfer learning from unlabeled data. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, pages 759–766, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-793-3. doi: 10.1145/1273496.1273592.

[240] Jeffrey O. Kephart and David M. Chess. The vision of autonomic computing. *Computer*, 36(1):41–50, January 2003. ISSN 0018-9162. doi: 10.1109/MC.2003.1160055.

[241] Yuriy Brun, Giovanna Di Marzo Serugendo, Cristina Gacek, Holger Giese, Holger Kienle, Marin Litoiu, Hausi Müller, Mauro Pezzè, and Mary Shaw. *Engineering Self-Adaptive Systems through Feedback Loops*, pages 48–70. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009. ISBN 978-3-642-02161-9. doi: 10.1007/978-3-642-02161-9_3.

[242] Rogério de Lemos, Holger Giese, Hausi A. Müller, and Mary Shaw. *Software Engineering for Self-Adaptive Systems: A Second Research Roadmap*, pages 1–32. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013. ISBN 978-3-642-35813-5. doi: 10.1007/978-3-642-35813-5_1.

[243] Nevil Brownlee, Cyndi Mills, and Greg Ruth. RFC 2722 - Traffic flow measurement: Architecture. Technical report, 1999.

[244] CICFlowMeter. UNB CIC network traffic flow generator (formerly iscxflowmeter). URL http://www.unb.ca/cic/datasets/flowmeter.html.

[245] Honglak Lee, Alexis Battle, Rajat Raina, and Andrew Y. Ng. Efficient sparse coding algorithms. In *Proceedings of the 19th International Conference on Neural Information Processing Systems*, NIPS'06, pages 801–808, Cambridge, MA, USA, 2006. MIT Press. URL http://dl.acm.org/citation.cfm?id=2976456.2976557.

[246] P. Nespoli, D. Papamartzivanos, F. G. Mármol, and G. Kambourakis. Optimal countermeasures selection against cyber attacks: A comprehensive survey on reaction frameworks. *IEEE Communications Surveys Tutorials*, 20(2):1361–1396, Secondquarter 2018. ISSN 1553-877X. doi: 10.1109/COMST.2017.2781126.

[247] Tao Ma, Fen Wang, Jianjun Cheng, Yang Yu, and Xiaoyun Chen. A hybrid spectral clustering and deep neural network ensemble algorithm for intrusion detection in sensor networks. *Sensors*, 16(10), 2016. ISSN 1424-8220. doi: 10.3390/s16101701.

[248] Ulrike von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, Dec 2007. ISSN 1573-1375. doi: 10.1007/s11222-007-9033-z.

[249] Yadigar Imamverdiyev and Fargana Abdullayeva. Deep learning method for denial of service attack detection based on restricted boltzmann machine. *Big Data*, 6 (2):159–169, 2018. doi: 10.1089/big.2018.0023.

[250] Yang Yu, Jun Long, and Zhiping Cai. Network intrusion detection through stacking dilated convolutional autoencoders. *Security and Communication Networks*, 2017, 2017. doi: 10.1155/2017/4184196.

[251] Dimitrios Papamartzivanos, Félix Gómez Mármol, and Georgios Kambourakis. Dendron : Genetic trees driven rule induction for network intrusion detection systems. *Future Generation Computer Systems*, 79:558 – 574, 2018. ISSN 0167-739X. doi: 10.1016/j.future.2017.09.056.

[252] Dimitrios Papamartzivanos and Felix Gomez Mármol. Intrusion detection and prevention system and method for generating detection rules and taking countermeasures, September 25 2018. US Patent 10,084,822.

[253] Aimin Zhou, Bo-Yang Qu, Hui Li, Shi-Zheng Zhao, Ponnuthurai Nagaratnam Suganthan, and Qingfu Zhang. Multiobjective evolutionary algorithms: A survey of the state of the art. *Swarm and Evolutionary Computation*, 1(1):32 – 49, 2011. ISSN 2210-6502. doi: 10.1016/j.swevo.2011.03.001.

[254] L. Bu¿oniu, R. Babu *hats*ka, and B. D. Schutter. A comprehensive survey of multiagent reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 38(2):156–172, March 2008. ISSN 1094-6977. doi: 10.1109/TSMCC.2007.913919.

[255] D. Papamartzivanos, F. Gómez Mármol, and G. Kambourakis. Introducing deep learning self-adaptive misuse network intrusion detection systems. *IEEE Access*, 7:13546–13560, 2019. ISSN 2169-3536. doi: 10.1109/ACCESS.2019.2893871.

[256] Dimitrios Papamartzivanos, Felix Gomez Mármol, Georgios Kambourakis, and Roberto Bifulco. Adaptive network lntrusion detection, November 9 2018. US provisional patent application 62/757,769.

[257] DARPA: Defense Advanced Research Projects Agency. AI Next Campaign, September 2018. URL https://www.darpa.mil/work-with-us/ai-next-campaign.

[258] Eric Ahlm for Gartner Research. Emerging technology analysis: SOAR Solutions, December 2018.