ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΙΓΑΙΟΥ

ΔΙΔΑΚΤΟΡΙΚΗ ΔΙΑΤΡΙΒΗ

Ανίχνευση και πρόληψη επιθέσεων άρνησης εξυπηρέτησης στα πρωτόκολλα SIP και SDP

Συγγραφέας
Ζήσης Κ. Τσιάτσικας

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, Απρίλιος 2019

University of the Aegean

Doctoral Thesis

Detection and prevention of denial of service attacks in SIP and SDP

Author

Zisis K. Tsiatsikas

Supervisor

Associate Professor 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, April 2019

Υπεύθυνη Δήλωση

Εγώ ο Ζήσης Κ. Τσιάτσικας, δηλώνω ότι είμαι ο αποκλειστικός συγγραφέας της υποβληθείσας Διδακτορικής Διατριβής με τίτλο «Ανίχνευση και πρόληψη επιθέσεων άρνησης εξυπηρέτησης στα πρωτόκολλα SIP και SDP». 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: Απρίλιος 12, 2019

Declaration of Authorship

I, Zisis K. Tsiatsikas, declare that this thesis entitled, "Detection and Prevention of Denial of Service attacks in SIP and SDP" 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: April 12, 2019 _____

Advising Committee of this Doctoral Thesis:

Professor Stefanos Gritzalis, Advisor
Department of Information and Communication Systems Engineering
University of the Aegean

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

Lecturer Dimitrios Geneiatakis, Advisor
Department of Electrical and Computer Engineering
Aristotle University of Thessaloniki
Researcher Joint Research Centre - European Commission, Italy

---

University of the Aegean, Greece
2019

Approved by the Examining Committee:

Professor Stefanos Gritzalis
University of the Aegean, Greece

Professor Vasilios Katos
Bournemouth University, UK

Professor Costas Lambrinoudakis
University of Piraeus, Greece

Associate Professor Georgios Kambourakis
University of the Aegean, Greece

Associate Professor Spyros Kokolakis
University of the Aegean, Greece

Assistant Professor Christos Goumopoulos
University of the Aegean, Greece

Lecturer Dimitrios Geneiatakis
Aristotle University of Thessaloniki, Greece
Researcher Joint Research Centre - European Commission, Italy

---

University of the Aegean, Greece
2019

Zisis K. Tsiatsikas

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 Zisis K. Tsiatsikas

VoIP services in general, and Session Initiation Protocol (SIP) ones in particular, continue to grow at a fast pace and have already become a key component of Next Generation Networks (NGN). Despite this proliferation, SIP-based services expose a large attack surface which make them prone to a plethora of attacks. Among them, Denial of Service (DoS) is perhaps the most powerful and devastating one, as it aims to drain the underlying resources of a service and make it inaccessible to its legitimate users. So far, various detection and/or prevention schemes have been proposed to detect, deter, and eliminate DoS occurrences in SIP ecosystems. It is true that while this topic has been addressed in the literature to a great extent, none of the so far proposed works seems to be complete in assessing in both realtime and offline modes if a system remains free of such types of attacks. Additionally, only a handful of works examine the potential of Machine Learning (ML) techniques to detect DoS in SIP-based networks, and even fewer do so in realtime. Additionally, the current attack detection and prevention schemes focus almost completely on the SIP headers neglecting the message body. An additional significant gap of the literature of this topic is the examination of the potential of communicating hidden information in SIP using the message body. The latter can be applied either with respect to covert channels, or malformed messages. That is, while a significant mass of works in the literature cope with covert communication channels, only a very limited number of them rely on SDP to realize its goals.

The major contribution of this PhD thesis is focused on the design and implementation of an Intrusion Detection System (IDS) framework, with the aim to battle against DDoS attacks in SIP. This contribution is met with the design and development of a framework which analyzes audit trails. Such network trails are considered a rich source of information toward flushing out DoS incidents, and evaluating the security level of the system of interest. Specifically, we introduce an end-user privacy-friendly service to assess whether or not a SIP service provider suffers a DoS by examining either the recorded audit trails (in a forensic-like manner) or the realtime traffic. To do so, we employ 2 statistical methods namely Entropy and Hellinger distance.

A second contribution of this thesis revolves around the assessment of the potential of 5 different ML-driven techniques in combating SIP-driven Distributed Denial of Service (DDoS) Attacks. In our analysis, we take into account both high and low-rate Distributed DDoS when assessing the efficacy of ML techniques in SIP intrusion detection. Additionally, we employ this type of solutions both in an offline and realtime fashion. In the context of this analysis, and for DDoS attacks, we compare our results with those produced by the two aforementioned statistical and anomaly-based detection methods, namely Entropy and Hellinger distance.

The last contribution of this PhD thesis pertains to the exploitation of Session Description Protocol (SDP) as an attack vector. As already mentioned, while so far several works in the literature have been devoted to the detection of DoS attacks in SIP ecosystems, the focus is on those which exploit SIP headers neglecting the message body. In this respect, in the context of this PhD thesis we examine the following two categories of attacks (a) those that exploit SDP information residing in SIP requests, with the aim to hide data in plain sight, and (b) those which capitalize on SDP to mount malformed message attacks with the purpose of causing DoS either to the SIP proxy or the end-users. More precisely, with respect to the first type of attack, the PhD thesis at hand demonstrates and evaluates the feasibility of a simple but very effective in terms of stealthiness and simplicity SDP-based covert channel for botnet Command and Control (C&C). To this end, we do not only scrutinize this ilk of attacks and demonstrate their effect against the end-user, but also develop an open source extensible SDP parser module capable of detecting intentionally or unintentionally crafted SDP segments parasitizing in SIP requests. Following a firewall-based logic, currently, the parser incorporates 100 different rules organized in 4 categories (policies) based on RFC 4566 [1].

Greek Abstract

(Εκτεταμένη Περίληψη)

Τμήμα Μηχανικών Πληροφοριακών και Επικοινωνιακών Συστημάτων
Πολυτεχνική Σχολή
Πανεπιστήμιο Αιγαίου

Διδακτορική διατριβή
του Ζήση Κ. Τσιάτσικα

Οι πολυμεσικές υπηρεσίες που παρέχονται μέσω του Διαδικτύου, και συγκεκριμένα αυτές που αξιοποιούν το πρωτόκολλο SIP για τη διαχείριση της σηματοδοσίας, συνεχίζουν να αναπτύσσονται ραγδαία. Επιπρόσθετα, οι εν λόγω υπηρεσίες αποτελούν ένα από τα βασικά δομικά στοιχεία των δικτύων ύστερων γενεών (5G). Ανεξαρτήτως του ρυθμού ανάπτυξης τους, οι πολυμεσικές υπηρεσίες που αξιοποιούν το SIP παρουσιάζουν μια ευρεία επιφάνεια ευπαθειών, καθιστώντας τις αντίστοιχες υπηρεσίες επιρρεπείς σε ένα μεγάλο σύνολο επιθέσεων. Μεταξύ αυτών, οι επιθέσεις άρνησης εξυπηρέτησης ανήκουν στην κατηγορία ισχυρότερων και καταστρεπτικότερων, καθώς στοχεύουν στην κατανάλωση των πόρων των εμπλεκομένων συστημάτων και δικτύων, με σκοπό να τα παραλύσουν και να προκαλέσουν δυσαρέσκεια στον τελικό χρήστη. Μέχρι στιγμής, η βιβλιογραφία στο συγκεκριμένο ερευνητικό πεδίο εμφανίζει ένα σημαντικό πλήθος εργασιών με σκοπό την ανίχνευση, την αποτροπή και την ελαχιστοποίηση των επιπτώσεων των επιθέσεων άρνησης εξυπηρέτησης. Συνολικά, αν και η συγκεκριμένη περιοχή έχει μελετηθεί εκτενώς στη βιβλιογραφία, φαίνεται να υπάρχει ένα σημαντικό κενό αναφορικά με την αποτίμηση των μεθόδων ή/και πλαισίων ασφαλείας, και την αξιολόγηση τους, όταν αυτά λειτουργούν τόσο σε πραγματικό, όσο και σε μη πραγματικό χρόνο. Επιπρόσθετα, μόνο ένα μικρό σύνολο των εν λόγω εργασιών ασχολείται με την εφαρμογή τεχνικών μηχανικής μάθησης για την ανίχνευση περιστατικών άρνησης εξυπηρέτησης σε περιβάλλοντα SIP. Τέλος, ενώ το μεγαλύτερο μέρος της σχετικής βιβλιογραφίας ασχολείται με τις επιθέσεις άρνησης εξυπηρέτησης που έχουν ώς φορέα τις κεφαλίδες των μηνυμάτων SIP, ελάχιστες εξετάζουν το κυρίως σώμα του μηνύματος. Επιπλέον, ενώ μέχρι τώρα ένας σημαντικός όγκος των σχετικών εργασιών στην παρούσα ερευνητική περιοχή ασχολούνται με επιθέσεις στο πρωτόκολλο SIP, ελάχιστες από αυτές ασχολούνται με τη δημιουργία συγκεκαλυμμένων καναλιών, και καμία με την αξιοποίηση του πρωτοκόλλου SDP ως φορέα αυτών.

Η σημαντικότερη συνεισφορά της παρούσας διατριβής αφορά στην ανάπτυξη και το σχεδιασμό ενός συστήματος ανίχνευσης εισβολών, με σκοπό την καταπολέμηση κατανεμημένων επιθέσεων άρνησης εξυπηρέτησης στο SIP. Ο στόχος αυτός, εκπληρώνεται με το σχεδιασμό και την ανάπτυξη μηχανισμών ανίχνευσης εισβολών που βασίζονται στην ανάλυση αρχείων λογιστικών καταγραφών του δικτύου στο επίπεδο εφαρμογής. Τα αρχεία αυτά θεωρούνται μια πλούσια πηγή πληροφορίας αναφορικά με την ανίνευση περιστατικών άρνησης εξυπηρέτησης, καθώς και για την γενικότερη αξιολόγηση του επιπέδου ασφαλείας ενός συστήματος. Συγκεκριμένα, σχεδιάζουμε, υλοποιούμε και αξιολογούμε μια υπηρεσία ανίχνευσης εισβολών που επιπλέον διατηρεί την ιδιωτικότητα του τελικού χρήστη με γνώμονα την ιδιωτικότητα, για την αξιολόγηση επιθέσεων άρνησης εξυπηρέτησηςη ενός παρόχου υπηρεσιών VoIP, είτε εξετάζοντας αρχεία λογιστικών καταγραφών, ή εξετάζοντας τη δικτυακή κίνηση σε πραγματικό χρόνο. Για να επιτευχθεί αυτό, αξιοποιούμε δύο στατιστικές μεθόδους που βασιζονται στην εντροπία και την απόσταση Hellinger.

Μια δεύτερη συνεισφορά της παρούσας διατριβής επικεντρώνεται στην αξιολόγηση της εφαρμογής πέντε διαφορετικών τεχνικών μηχανικής μάθησης, με σκοπό την καταπολέμηση των επιθέσεων κατανεμημένης άρνησης εξυπηρέτησης. Στην αξιολόγηση των μεθόδων μας λαμβάνουμε υπόψη επιθέσεις τόσο υψηλού όσο και χαμηλού όγκου δεδομένων και ρυθμών μετάδοσης μηνυμάτων. Επιπρόσθετα, στο πλαίσιο της ανάλυσης μας, και για την περίπτωση των κατανεμημένων επιθέσεων άρνησης εξυπηρέτησης, συγκρίνουμε τις επιδόσεις του συστήματος με αυτές που παρήχθησαν κατά την εφαρμογή των στατιστικών μεθόδων, δηλαδή της εντροπίας και της απόστασης Hellinger.

Υπό αυτό το πρίσμα, εξετάζουμε τις ακόλουθες δύο βασικές κατηγορίες επιθέσεων (α) αυτές που χρησιμοποιούν το πρωτόκολλο SDP - το οποίο αξιοποιείται από το SIP - ως φορέα προκειμένου να μεταφέρουν κρυφή πληροφορία, και (β) αυτές που αξιοποιούν συγκεκριμένα τμήματα ενός μηνύματος SDP με σκοπό να προκαλέσουν άρνηση εξυπηρέτησης στα εμπλεκόμενα συστήματα ή/και στον χρήστη της υπηρεσίας. Ως παράδειγμα του πρώτου τύπου επιθέσεων παρουσιάζουμε και αξιολογούμε τη λειτουργία ενός απλού αλλά ιδιαίτερα αποτελεσματικού κρυφού καναλιού επικοινωνίας, το οποίο βασίζεται αποκλειστικά στο πρωτόκολλο SDP, για την επικοινωνία μολυσμένων συσκευών (ρομπότ) μεταξύ τους ή με τον διακομιστή εντολών (C&C). Σε αυτό το πλαίσιο, όχι μόνο εξερευνούμε αυτή την κατηγορία επιθέσεων, αλλά επίσης καταδεικνύουμε την επίδραση τους στον τελικό χρήστη. Επιπρόσθετα, υλοποιούμε ένα επεκτάσιμο άρθρωμα ανοιχτού λογισμικού, ικανό να ανιχνεύει εκούσια ή ακούσια αλλοιωμένα μηνύματα SDP, τα οποία παρασιτούν στα μηνύματα SIP. Ακολουθώντας μια λογική παρόμοια με αυτή ενός αναχώματος ασφαλείας, το εν λόγω άρθρωμα λογισμικού μέχρι στιγμής ενσωματώνει 100 διαφορετικούς κανόνες, οργανωμένους σε 4 κατηγορίες (πολιτικές) σύμφωνα με το RFC 4566 [1].

Acknowledgements

I want to express my deepest gratitude to my supervisor Associate Professor Georgios Kambourakis for the invaluable help, the guidance, and the opportunities he offered me during my PhD studies. A highly skilled Professor, and a exceptional educator, always boosting his students to gain knowledge.

Additionally, i want to thank Dr. Dimitrios Geneiatakis for helping me in each one of the works we succeeded in this six-year journey, during my MSc and PhD studies at the University of the Aegean. His invaluable experience and knowledge in the area of VoIP security and privacy helped me to overcome the obstacles and move on.

Furthermore, i want to thank Professor Stefanos Gritzalis for the priceless guidance and the help he offered me during my studies at the University of the Aegean.

Moreover, my thanks go to all the colleagues we cooperated these years in the Department of Information and Communication Systems Engineering: Mrs. Anastasia Douma, Dr. Marios Anagnostopoulos, Mr. Alexandros Fakis, Mr. Dimitrios Papamartzivanos. Additionally, i want to thank Mr. Sozon Lambrou for his help.

I also want to thank Mrs. Elina Tsakalidou, as well as Mrs. Maria Lymperi for their help in the proofreading of specific parts in this thesis.

After that i want to sincerely thank every person that helped me and contributed on making my life easier during my stay in Samos island. Thank you all!

Finally, i want to thank my parents Konstantinos Tsiatsikas and Maria Theodoridou for the moral and financial support they offered me during my studies. Nothing would be possible without their help. A thank goes also to my sister Styliani.

I would like to thank the University of the Aegean and the Greek Ministry of Development-General Secretariat of Research and Technology for the funding i received during my PhD studies. This support was received in the context of my participation in the 5179 (SCYPE) research project, implemented within the context of the Greek Ministry of Development-General Secretariat of Research and Technology funded program "Excellence II / Aristeia II", co-financed by the European Union/European Social Fund - Operational program "Education and Life-long Learning" and National funds.

Dedicated to my parents
Konstantinos Tsiatsikas & Maria Theodoridou.

Contents

List of Figures

List of Tables

Abbreviations

ABNF	**A**ugmented **B**ackus-**Naur** **F**orm
AIB	**A**uthenticated **I**dentity **B**ody
AIS	**A**rtificial **I**mmune **S**ystem
ANN	**A**rtificial **N**eural Networks
ARFF	**A**ttribute- **R**elated **F**ile **F**ormat
B2BUA	**B**ack **T**o **B**ack **UA**
C&C	**C**ommand & **C**ontrol
CNN	**C**onvonutional **N**eural Networks
CPA	**C**hange **P**oint **A**nalysis
CPS	**C**alls **P**er **S**econd
CPU	**C**entral **P**rocessing **U**nit
CSV	**C**omma **S**eparated Values
CUSUM	**CU**mulative **SUM**
CVE	**C**ommon **V**ulnerabilities **E**xposures
DDoS	**D**istributed **D**enial **O**f **S**ervice
DES	**D**iscrete **E**vent **S**ystem
DMZ	**D**emilitarized **Z**one
DNS	**D**omain **N**ame **S**ystem
DNS	**D**irectory **N**umber
DNS	**D**omain **N**ame **S**ystem
DoS	**D**enial **O**f **S**ervice
DPI	**D**eep **P**acket **I**nspection
DWT	**D**iscrete **W**avelet **T**ransform
EENA	**E**uropean **E**mergency **N**umber **A**ssociation
ESINET	**E**mergency **S**ervices **I**nternet **P**rotocol **N**etwork

ESRP	**E**mergency **S**ervices **R**outing **P**roxy
EWMA	**E**xponential **W**eighted **M**oving **A**verage
FIDS	**F**low-based **I**ntrusion **D**etection **S**ystem
FIFO	**F**irst **I**n **F**irst **O**ut
FN	**F**alse **N**egative
FP	**F**alse **P**ositive
FSM	**F**inite **S**tate **M**achine
GC	**G**arbage **C**ollector
GUI	**G**raphical **U**ser **I**nterface
GW	**G**ateway
HD	**H**elinger **D**istance
HMAC	**H**ash-based **M**essage **A**ithentication **C**ode
HMM	**H**idden **M**arkov **M**odels
HTTP	**H**yper **T**ext **T**ransfer **P**rotocol
I-CSCF	**I**nterrogating **C**all **S**ession **C**ontrol **F**unction
IETF	**I**nternet **E**ngineering **T**ask **F**orce
IMS	**I**internet **M**ultimedia **S**ubsystem
IoT	**I**internet **o**f **T**hings
JNI	**J**AVA **N**ative **I**nterface
JVM	**J**AVA **V**irtual **M**achine
LMS	**L**og **M**anagement **S**ystem
LTE	**L**ong **T**erm **E**volution
MAC	**M**essage **A**uthentication **C**ode
MCS	**M**ulti **C**lasification **S**ystems
MITM	**M**an **I**n **T**he **M**iddle
ML	**M**achine **L**earning
MLP	**M**ultilayer **P**erceptron
MOS	**M**ean **O**pinion **S**core
MTU	**M**aximum **T**tansmission **U**nit
NAT	**N**etwork **A**ddress **T**ranslation
NENA	**N**ational **E**mergency **N**umber **A**ssociation
NGN	**N**ext **G**eneration **N**etworks
NIDS	**N**etwork **I**ntrusion **D**etection **S**ystems

NIST	**N**ational **I**nstitute of **S**tandards and **T**echnology
NTP	**N**etwork **T**ime **P**rotocol
PBX	**P**rivate **B**ranch **eX**change
PDF	**P**robability **D**ensity **F**unction
P-CSCF	**P**roxy **C**all **S**tate **C**all **F**unction
P-GW	**P**roxy **GW**
PSAP	**P**ublic **S**afety **A**nswering **P**oint
PSTN	**P**ublic **S**witched **T**elephony **N**etwork
P2P	**P**eer **T**o **P**eer
QoS	**Q**uality **O**f **S**ervice
RFC	**R**equest **F**or **C**omments
ROC	**R**eceiver **O**perating **C**haracteristic
RTP	**R**ealtime **T**ransport **P**rotocol
SBC	**S**ession **B**order **C**ontroller
SDN	**S**oftware **D**efined **N**etworking
SDP	**S**ession **D**escription **P**rotocol
SFW	**S**IP **F**ire **W**all
SHA	**S**ecure **H**ash **A**lgorithm
SIP	**S**ession **I**nitiation **P**rotocol
SM	**S**ignalling **M**anager
SMO	**S**equential **M**inimal **O**ptimization
SPIT	**S**pam **O**ver **I**nternet **T**elephony
SSL	**S**ecure **S**ocket **L**ayer
SVM	**S**upport **V**ector **M**achines
S-GW	**S**ystem **GW**
TCP	**T**ransmission **C**ontrol **P**rotocol
TDoS	**T**elephony **D**enial **O**f **S**ervice
TLS	**T**ransport **L**ayer **S**ecurity
TP	**T**rue **P**ositive
UA	**U**ser **A**gent
UAC	**U**ser **A**gent **C**lient
UAS	**U**ser **A**gent **S**erver
UDP	**U**ser **D**atagram **P**rotocol

URI	**U**niform **R**esource **I**dentifier
VoIP	**V**oice **o**ver **I**nternet **P**rotocol
VoLTE	**V**oice **o**ver **LTE**
VPN	**V**irtual **P**rivate **N**etwork
WebRTC	**W**eb **R**eal- **T**ime **C**ommunication
3GPP	**3**rd **G**eneration **P**arthnership **P**roject

Chapter 1

Introduction

According to recent marketing analysis reports [2, 3, 4], Voice over IP (VoIP) services are mushrooming on a daily basis. As in Public Switch Telephone Networks (PSTN), a central role in VoIP communications plays a signaling protocol responsible for managing (establish, update, terminate) user sessions. Although various signaling protocols, including H.323 [5], SIP [6], Media Gateway Control Protocol (MGCP) [7], have been proposed, the Session Initiation Protocol (SIP) [6] seems to be the predominant one. This is because SIP inherits from the HTTP [8] model and structure, thus providing a high degree of freedom to easily develop new multimedia services and products.

Despite the advantages users enjoy due to SIP flexibility, various attacks have been identified against SIP-based VoIP services [9, 10, 11]. To alleviate, if not eliminate, such security flaws a diversity of detection and prevention solutions have been proposed in the literature [12, 13, 14, 15, 16]. However, while all these security mechanisms and countermeasures may be of considerable value, they do not capitalize on log files collected by the providers. So, it might mistakenly be taken for granted that the underlying services are secure, while in fact they are prone to security breaches, which have gone undetected. This especially applies to low-volume Denial of Service (DoS) attacks, which are lately on the rise and admittedly remain hard to detect and repel. In fact, the value of audit trail data in identifying security violations and flaws in applications has been highlighted by several researchers, security fora and organizations, including the National Institute of Standards and Technology (NIST) [17].

On the downside, personal data contained in audit trails - and especially those stemming from the application layer as that of SIP - are subject to various legal restrictions and regulations. This fact alone makes the processing and exchange of audit trails among multimedia providers highly troublesome and problematic. This is because the exposure of sensitive personal information contained in audit trails to unauthorized entities is prone to several malicious acts that clearly violate the users' private sphere [18, 19, 20, 21]. For instance, an ill-motivated actor is able to learn the user's real identities and next eavesdrop on which services are being accessed by them, thus violating the principle of user anonymity [22, 23]. In the long term, when this kind of information is systematically gathered, the end-user can be profiled and sensitive information (e.g., preferred services) can be inferred. So, while audit trail analysis in VoIP ecosystems may be of great value, this needs to happen after a data-neutralization process takes place. This is necessary in order to obfuscate certain pieces of personal information contained in log files and preserve the privacy of the end-users.

Moreover, by examining the rather rich literature on SIP security, one can distinguish several categories of assaults ranging from SQL injection to Denial of Service (DoS) [24, 25, 26, 27]. It can be safely argued that the latter category attracts the greater attention, and seems to be the most perilous and difficult to confront since it is closely related with the signaling nature of the protocol per se. So, focusing on this kind of attacks, so far, several protection and detection methods have been proposed. Roughly, we can categorize them into misuse-detection and anomaly-detection ones. Generally, the first family of methods monitors network activity with exact signatures of known malicious behavior (e.g., observe the network traffic for singular byte sequences), while the second possesses a knowledge of normal activity and warns against any deviation from that profile. The latter category of methods, is usually realized by means of tools borrowed from the Machine Learning (ML) community. This refers to algorithms that get trained first in an either supervised or unsupervised manner with reference input to learn its particulars, and then are fed with unknown input for accomplishing the real detection process. Specifically for SIP, although the DoS threat has been stressed out and dealt with a significant number of researches [9, 28], the applicability and effectiveness of ML techniques to cope against such incidents are still being assessed and certainly in need for further development.

Naturally, this is mainly due to the increased overhead that these methods may bear

- especially when it comes to realtime detection and a training phase is required - in comparison to misuse-based or purely statistical ones. Nevertheless, in this PhD thesis we argue that ML techniques can be particularly fruitful for examining the high-volume log files of a given VoIP realm in an offline fashion if they contain DoS incidents. Also, this category of methods may show better results when used for the detection of low-rate DoS (also known with the term "low and slow"), which although not used to paralyze the target system at a fast pace, it consumes valuable network, CPU and memory resources. Ultimately, this results in service delays which in turn cause customer dissatisfaction with direct negative results in the provider's market share.

In a typical DoS assault, the attacker tries to paralyse the victim by either sending against it a surge of SIP requests or a number of malformed messages. In the former case, the victim is unable to serve the voluminus number of incoming requests, while in the latter the sufferer is incapable of parsing or handling properly the incoming request, and the service crashes. On the other hand, as VoIP services rely on the open Internet, providers need to ensure availability levels similar to PSTN. This means that among other well-documented threats [24, 26, 25, 27] they need to cope with resource consumption attacks targeting the application layer, namely Denial Service (DoS) as well as their distributed form (DDoS) that causes service disruptions and sometimes even complete outages. This is of high importance especially for critical voice services e.g., emergency numbers. This threat is further aggravated as the current predominant VoIP signaling protocol, namely Session Initiation Protocol (SIP), can be easily exploited by an attacker. This is mainly due to SIP text nature that allows the aggressor to straightforwardly craft and send large volumes of SIP requests toward its victim with the aim of paralysing it. The perpetrator is also able to exercise more clever attacks, including low and slow ones [29] in order to consume a considerable amount of VoIP server's resources and network bandwidth, and thus degrade the quality of the service.

The common denominator of the latter kind of attacks is the manipulation of SIP message headers by the attacker so as to hamper or preferably paralyse the parsing process at the SIP server or client. To cope with this threat, the great mass of works [30, 31, 25, 32] in the literature propose some way for the SIP server to detect malformed SIP headers, and thus discard the corresponding messages outright. However, what is largely neglected is that similar assaults may take advantage of the Session Description Protocol (SDP) [1] part of a SIP message. Recall that SDP is responsible for negotiating the

media information among the communicating peers, and as a result, SDP information is present in a diverse type of SIP requests and replies. Moreover, according to the literature [33], SDP can be exploited to build covert communication channels in ignorance of the SIP network. For instance, such hidden channels are typically exploited by botnets for realizing a command and control (C&C) infrastructure.

Bearing SDP-driven DoS attacks in mind, the goal of this thesis is dual; first to provide proof of the pernicious nature of this type of attacks, on real-life SIP clients and servers, and second to introduce a lightweight and flexible filtering mechanism for effectively coping with them. Moreover, a defensive solution is provided. The latter comes in the form of a software module either embedded in the SIP server or running in a separate machine in front of the former. In this way, the defender is able to timely detect and silently drop messages that do not fully comply with the standard [1]. Also, as a side advantage, the parser is capable of rejecting SIP requests that are found suspicious to carry information that may be part of a covert communication channel.

1.1 Thesis motivation and objectives

Attackers always find ways to elude the employed security mechanisms of a system, no matter how strong they are. Nevertheless, audit trails - which as a rule of thumb are kept by any service provider - store all the events pertaining to the service of interest. Therefore, audit trail data can be a valuable ally when it comes to the certification of the security level of a given service. This stands especially true for critical realtime services such as multimedia ones, which nowadays are on the rise.

In a nutshell, audit trails, especially those of large volume as in the case of multimedia services, are rarely utilized properly so as to prove service abuse. As already pointed out this is mainly due to privacy restrictions. Therefore, as a general rule, any solution focusing on digital forensic analysis should deduce services security level with respect to audit trails (as well), but it is important to do so without violating the privacy of the end-user. Overall, it would be very beneficial if the existing security audit controls and certification methodologies could be seconded by appropriate tools that are able to automatically analyze the collected audit trails and determine in a formal way whether or not the provided service suffers a security incident, e.g., a flooding attack.

To cope with this threat, several SIP-oriented intrusion detection and prevention systems have been presented in the literature so far. Focusing on proposals to defend against application layer DoS in these environments, one can identify simple statistical schemes as those given in [13, 34]. In this category of solutions detection relies on different network statistics, including incoming traffic rate, and uses a predefined threshold above which the received traffic is classified as malicious. It is obvious, though, that such a solution cannot protect SIP services effectively against low-rate application layer DoS attacks, as it is cumbersome to constantly adapt itself, say, by recalculating the underlying threshold to reflect the ongoing characteristics of the attack.

Furthermore, most of the existing solutions do not consider SIP different inherent features and characteristics which can be exploited by an attacker to launch DoS, while the majority of them are privacy-invasive. A comprehensive analysis of existing protection solutions against DoS in SIP can be found in [9]. Therefore, as the aggressors become more sophisticated, there is a need for advanced DDoS detection methods, in the application layer, that are able to automatically adjust their behavior to the attack traffic patterns. A way to achieve this is to use well-established classifiers from the Machine Learning (ML) toolbox.

Moreover, the literature so far largely misses the exploration of SIP malformed message attacks, which specifically leverage SDP information conveyed by certain SIP messages. Even the RFC 4475 [35], which is dedicated to SIP torture test messages, does not explicitly elaborate on SDP malformed message attacks. To this end, this PhD thesis not only contributes proofs of the feasibility and impact of this ilk of attack against real-life diverse SIP entities, but it additionally offers a publicly available parsing mechanism that can remedy it. Also, the same solution can help into deterring evildoers from exploiting the SIP infrastructure for building secret communication channels. As already discussed, SDP-based covert channels are scarcely addressed in the literature. That is, the majority of the existing works concentrates on the applicability of information hiding techniques in VoIP-related protocols in general. This includes SIP, Real Time Protocol (RTP) and RTP Control Protocol (RTCP). The delivered channels may be used in a variety of ways, aiming to establish secret paths of communication.

Taking into account the previously-mentioned findings, this PhD thesis aims to provide a complete framework, capable of thwarting application layer DDoS attacks in SIP and

SDP. The framework has the following characteristics: a) it works both in a realtime and offline fashion. b) It supports 2 statistical schemes and 5 ML classifiers. c) It performs a privacy-preserving analysis. d) It examines syntactically wrong SDP bodies which may cause either DoS attacks or convey hidden commands in the context of a C&C channel. To this end, the objectives of this work can be summarized as follows:

Objective 1: The first objective of the PhD thesis at hand is focused on the privacy-preserving analysis of audit-trails to detect security incidents. Moreover, the investigation of the applicability of 2 statistical schemes in the detection of SIP-based application layer DDoS attacks is considered a basic parameter of the proposed IDS framework. We aim at evaluating the statistical schemes both in a offline and realtime manner.

Objective 2: The second objective concerns the extension of the framework to support ML-driven detection of SIP-based application layer DDoS attacks. In this context we want to employ different ML classifiers, both in offline and realtime fashion, in order to evaluate their effectiveness. Using the results of Obj. 1 and Obj. 2, we intent to perform a side to side comparison of statistical schemes with ML techniques. This comparison will be based on the time overhead, false alarms, and detection accuracy between 2 statistical schemes and 5 ML techniques. Furthermore, it will provide proofs for building more efficient and robust IDS systems, to cope against DDoS attacks in SIP ecosystems.

Objective 3: The third objective of this thesis pertains to the investigation and mitigation of application layer SDP-driven attacks. More specifically, we examine the feasibility of exploiting SDP as a Command and Control (C&C) channel aiming to convey hidden information over malformed SDP segments to an SIP botnet and launch attacks. Additionally, we study the impact of malformed SDP messages which target the application layer in order to cause DoS. In this respect, the last goal of this thesis revolves around the design and development of a protection mechanism to combat against the 2 types of attacks we mentioned earlier, by detecting malformed SDP bodies.

1.2 Thesis contributions

As already pointed out in the previous subsection, the main intention of this PhD thesis is the design and development of a privacy-preserving framework to trace application layer DDoS attacks in SIP and SDP. In this direction, the privacy-preserving analysis of audit

trails consists one of the main pillars of this framework. Additionally, the applicability and the assessment of 2 statistical schemes and 5 ML classifiers against application layer DDoS attacks in SIP, consists the main weaponry of this framework. Finally, the examination of the impact of SDP-driven malformed message attacks which target the application layer remains the last goal of this PhD thesis. More specifically, SDP-based C&C channels and DoS attacks which are caused due to malformed SDP segments are examined. Finally, a protection mechanism to remedy the previous-mentioned attacks in SDP is given.

All in all, the contributions of this study are fully aligned with the previously-analyzed objectives.

More precisely, to meet Obj.1 we study the privacy-preserving analysis of audit trail files which contain SIP traffic. In this direction we apply 2 pure statistical schemes, namely Entropy and Hellinger Distance, with the aim of detecting application layer DDoS attacks. We design and develop a software module, in order to address a method for the first scheme. Chapter 5 presents in detail the contributions with respect to the statistical schemes.

Moreover, with regards to Obj. 2, we employ 5 well-known classifiers for the detection of application layer DDoS attacks. Additionally, we study the overhead these methods introduce in a realtime system for both the training and the detection phase. Finally, we perform a side-by-side comparison with the schemes presented in the first objective. A detailed overview of these contributions is provided in Chapter 6.

Concerning Obj. 3, we study the capacity of a SDP-driven C&C channel. Additionally, we examine the impact of application layer DoS attacks which exploit SDP. Finally, in order to thwart these vulnerabilities, we introduce a software module capable of filtering malformed SDP segments. Chapter 7 presents a thorough view of these contributions.

All in all, the contributions of this PhD thesis with reference to publications in scientific journals and conference proceedings are as follows:

- The introduction of an Entropy-driven, efficient, and easily deployable method to analyze audit trail data from a security point of view in both realtime and offline

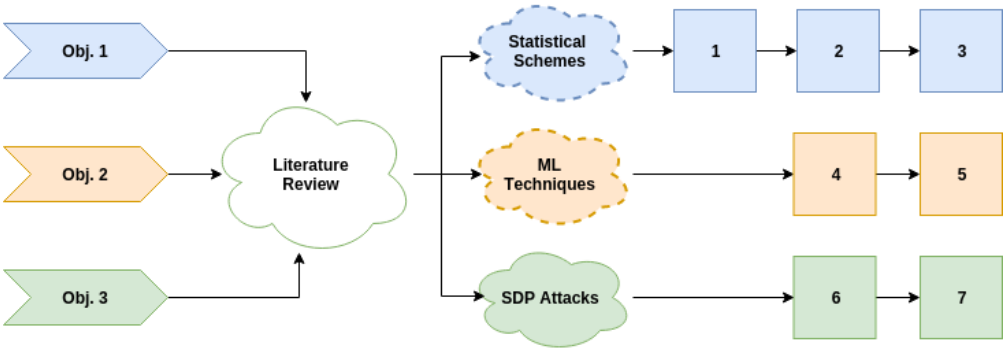

FIGURE 1.1: PhD thesis - Objectives

fashion [1] [2]. On top of that, this contribution enables VoIP providers to share their audit trails with trusted authorities in charge of analyzing its security status. This is possible because we mandate all data to be anonymized prior to being communicated between the different entities and get processed. In this respect, it is argued that the proposed solution bridges the gap between the limitations of existing approaches to identify security flaws by examining the audit trails, while at the same time it is orthogonal to the current defensive weaponry.

- The applicability of the well-known Hellinger Distance (HD) metric [36] to identify application layer DoS attacks in SIP-based multimedia services [3].

- The assessment of the effectiveness of 5 well-known classifiers to detect application layer DDoS incidents in SIP is examined both in offline [4] and realtime [5] modes. More specifically, :

  - A method to calculate SIP message headers occurrences from a given log file in a privacy-preserving way based on a predefined message window. The output of this process are fed to the ML algorithm as the case may be.

  - For DDoS scenarios, a comparison between two other anomaly-based detection

[1] Tsiatsikas, Z., Geneiatakis, D., Kambourakis, G., Keromytis, A. D. (2013, September). A Privacy-Preserving Entropy-Driven Framework for Tracing DoS Attacks in VoIP. (ARES 2013), DOI: 10.1109/ARES.2013.30

[2] Tsiatsikas, Z., Geneiatakis, D., Kambourakis, G., Keromytis, A. D. (2015). An efficient and easily deployable method for dealing with DoS in SIP services. Computer Communications, https://doi.org/10.1016/j.comcom.2014.11.002

[3] Tsiatsikas, Z., Kambourakis, G., Geneiatakis, D. Exposing resource consumption attacks in internet multimedia services, (ISSPIT 2014), DOI: 10.1109/ISSPIT.2014.7446237

[4] Tsiatsikas, Z., Fakis, A., Papamartzivanos, D., Geneiatakis, D., Kambourakis, G., Kolias, C. (2015, July). Battling against DDoS in SIP: Is Machine Learning-based detection an effective weapon?, (SECRYPT 2015), DOI: 10.5220/0005549103010308

[5] Tsiatsikas, Z., Geneiatakis, D., Kambourakis, G., Gritzalis, S. (2016, September). Real-time DDoS Detection in SIP Ecosystems: Machine Learning Tools of the Trade. NSS 2016, DOI: https://doi.org/10.1007/978-3-319-46298-1$_9$

methods proposed in the literature (i.e., Entropy, and Hellinger Distance) and ML-powered detection in terms of effectiveness, is given.

- A simple but powerful in terms of stealthiness covert communication protocol to exchange botnet C&C messages over SDP data in SIP requests [6]. More specifically, we demonstrate:

  – The evaluation of the effectiveness of the covert channel by controlling several bots and launching two different (i.e., a volumetric and a protocol-based) Denial of Service (DoS) types of attacks.

  – The attack impact in terms of resource consumption at the victim side is also examined.

- We provide proofs of the pernicious nature of SDP-driven application layer DoS attacks, on real-life SIP clients and servers. Additionally, a lightweight and flexible filtering mechanism for effectively copying with them is introduced [7]. This defensive solution comes in the form of an SDP parser software module either embedded in the SIP server or running in a separate machine in front of the former. In this way, the defender is able to timely detect and silently drop messages that do not fully comply with the standard [1]. Also, as a side advantage, the parser is capable of rejecting SIP requests that are found suspicious of carrying information that may be part of a covert communication channel. More specifically, in the area of SDP-driven malformed message attacks we provide:

  - A study of the impact of SDP malformed messages on a variety of SIP software and hardware phones, and servers.

  - A publicly available open-source software module capable of detecting malformed SDP messages lurking in SIP requests is given. Message parsing is done based on RFC 4566 [1], while the implemented software can work alongside the SIP server or independently in a separate machine.

  - We extensively assess the performance of the proposed scheme in terms of service time.

[6] Tsiatsikas, Z., Anagnostopoulos, M., Kambourakis, G., Lambrou, S., Geneiatakis, D. (2015, September). Hidden in plain sight. SDP-Based covert channel for botnet communication, (TrustBus 2015), DOI: https://doi.org/10.1007/978-3-319-22906-5$_4$

[7] Tsiatsikas, Z., Kambourakis, G., Geneiatakis, D., Wang, H. (2019). The Devil is in the Detail: SDP-Driven Malformed Message Attacks and Mitigation in SIP Ecosystems. IEEE Access, 7, 2401-2417.

Finally, figure 1.1 presents the contributions of this PhD thesis with respect to the objectives described in the previous subsection.

1.3 Thesis structure

The next Chapter presents the fundamental background of SIP and SDP. In this context, the basic messages, syntax, structure and network entities are presented.

Chapter 3 presents SIP and SDP threat models with respect to application layer DDoS attacks, malformed messages, and C&C channels.

Chapter 4 analyzes the literature review and emphasizes on three main pillars. The first one concerns the analysis of log files to detect security incidents. The second pertains to the detection of application layer DDoS attacks. The third one relates to C&C channels.

Chapter 5 elaborates in the detection of DDoS attacks in SIP using statistical means. In this context, two different schemes are presented building over Entropy theory and Hellinger Distance.

Chapter 6, presents and evaluates the applicability of five Machine Learning classifiers in the detection of application layer DDoS attacks in SIP.

Chapter 7 analyzes SDP-driven malformed messages and C&C channels. Additionally, a misuse-based prevention mechanism is presented against the aforementioned types of attacks.

All in all, the epilogue is draw in Chapter 8, by analyzing the results obtained from this PhD thesis. Additionally, defensive and preventive best practices, with respect to application layer DDoS attacks in SIP and SDP, are provided. Finally, future directions, with special emphasis on application layer DDoS attacks targeting SIP-based emergency calling centers are also provided.

Chapter 2

Background

2.1 VoIP services

During the last years Voice over IP (VoIP) technologies and services have penetrated the market and for many of us became an integral part of our software and/or hardware portfolio. Recent reports indicate that this market will grow to reach about USD 190 billion until 2024 [2]. In both mobile and fixed networks, Session Initiation Protocol (SIP) seems to be the predominant means for establishing and managing a VoIP session. On the downside, the text and open nature of the protocol has given rise to a plethora of attacks against it.

As the Internet dominates communication nowadays, telcos and other companies are trying to exploit its advantages to offer low cost voice multimedia communication services to their users. Very similar to legacy telecommunication systems the basis to do so is a signaling protocol in charge of managing the multimedia session. As already pointed out, nowadays, SIP seems to attract the major piece of attention. Intrinsically, SIP is designed to create, modify, and terminate voice and other more advanced multimedia sessions over the Internet. SIP is text-based with syntax similar to that of HTTP. SIP messages can be either a request or an acknowledgment to a corresponding request, consisting of the appropriate header fields and optionally a message body, depending on the nature of the request or response. An example of a typical SIP request message is given in figure 2.1.

Whenever a user wishes to use a SIP service she should announce its presence by registering their current IP address to the registration service (registrar) through a SIP REGISTER message. After that, the user is able to initiate a session with other registered or interconnected User Agents (UA) by sending a SIP INVITE message to its local SIP proxy. After the call has been established, the two endpoints, namely the caller and callee are able to start the multimedia session with the help of Realtime Transport Protocol (RTP) [37]. At any time, either the caller or the callee may terminate the call by sending a SIP BYE message toward the other endpoint.

2.2 SIP service architecture

SIP is a text-based protocol with syntax similar to that of HTTP. As shown in figure 2.1, a SIP message (request in this case) consists of two basic parts. The upper one corresponds to the message headers and carries information in regards to the sender (caller) and the recipient (callee) of the message. The second part is known as the message body and carries the media details. Communications resources in SIP are assigned a SIP Uniform Resource Identifier (URI), e.g., with reference to the first line of figure 2.1, sip:tzisis@msip.aegean.gr. Every SIP message is processed by the appropriate SIP component. A basic SIP infrastructure consists of:

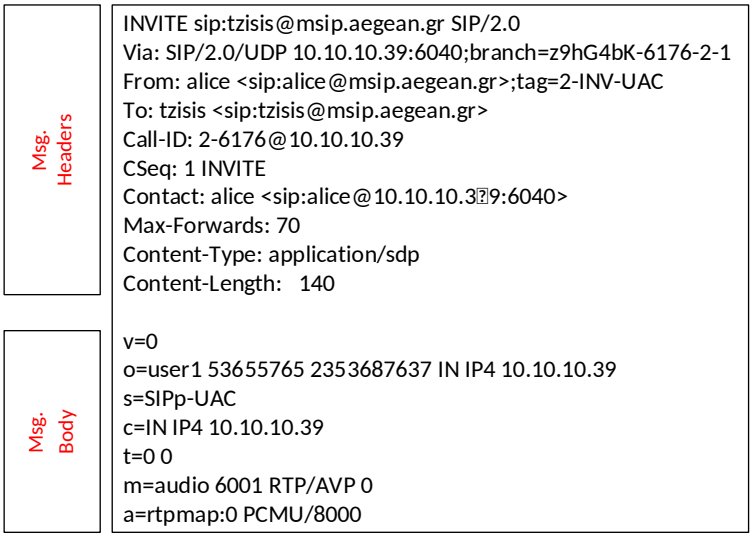

FIGURE 2.1: A typical SIP INVITE request

FIGURE 2.2: A typical SDP body

- **SIP User Agent (UA)** - Represents the end points of the SIP protocol, that is, the User Agent Client (UAC) and the User Agent Server (UAS) which are able to initiate or terminate a session using a SIP software or hardware client.
- **SIP Proxy Server** - An intermediate entity which plays the role of the client and the server at the same time. Its task is to route all the packets being sent and received by the users participating in a SIP session. Note that one or more SIP proxies may exist between any two UAs.
- **Registrar** - Handles the authentication and register requests initiated by the UAs. For this reason, this entity stores the user's credentials and UA location information.

The media session establishment process is initiated by the UAC (caller) sending a SIP INVITE request to the UAS (callee) via one or more SIP proxies. Upon receiving the request, the SIP proxy extracts the callee's username and queries the Registrar for obtaining the corresponding location information. Then, the SIP proxy forwards the request to

the callee. After that, the session is considered established and a media protocol takes over the management of the audio and video packets between the endpoints. Realtime Tranport Protocol (RTP) [37] has been acclaimed as the most appropriate for this purpose. At any time, either the caller or callee may terminate the media session by sending a BYE request toward the other endpoint. SIP architecture offers a total of 88 messages for session management. Among them, 14 are used as requests and the rest as responses. A SIP message may consist of two main parts, namely the SIP message headers (the upper part of the message) and the message body carrying SDP information. Specifically, the upper part carries information regarding the method, the sender, the recipient of the message, and the communication path. The SDP part, which is the focus of this work, conveys information concerning the media of the session and it may be present in specific SIP requests and responses, including INVITE, ACK, 180 RINGING, 183 Session Progress, 200 OK. That is, with the help of SDP, SIP employs the offer/answer model [38] to establish and manage multimedia sessions. Figure 2.2 presents the SDP part of a SIP INVITE request generated by the open source SIP UA Jitsi [39]. According to the corresponding RFC [1], every SDP message can contain up to 5 mandatory and 15 optional different descriptors. Table 7.1 provides a succinct overview per descriptor along with a typical example.

2.3 Protocol messages

SIP provides a set of requests for session management. These requests are presented in detail in Table 2.1. All of these requests can be used as a means to launch application layer DDoS attacks. In any case, those which are usually exploited by the attacks for these purposes are the SIP REGISTER and INVITE requests. Chapter 3 details on these types of attacks. Furthermore, all of these requests can be used for attacking a victim using malformed messages [40, 41, 42, 43]. Using this type of attack, the attacker aims on causing application layer DoS in the victim side. Finally, all of the previously-mentioned requests can be used for conveying hidden information in the context of a C&C covert channel [44, 45]. In any case, keep in mind that SDP-driven C&C's can be only build over those requests which are used for media negotiation (i.e., they carry a SDP body). Chapter 7 details on these messages.

TABLE 2.1: Basic SIP requests

Type of Request	Description
REGISTER	It is used for Registering on a service
INVITE	It is used for initiating a session
ACK	Acknowledges a SIP INVITE request
OPTIONS	It is used as a ping mechanism
SUBSCRIBE	Creates a subscription between a client and a service
NOTIFY	Provides information to the subscriber
PRACK	Acknowledges provisional responses
PUBLISH	Transfer information from the service to the broker
INFO	Conveys application level information
REFER	It is used for transfering a call
MESSAGE	Provides the ability to send instant messages
UPDATE	Updates the parameters of a session prior establishment
BYE	Terminates a session
CANCEL	Cancels the establishing of a session

Table 2.2 presents the set of responses which are used in SIP. The responses which convey SDP information can be exploited by an attacker with the aim to launch the 2 types of attacks this PhD thesis investigates in Chapter 7. In general, the rest of the responses can be used in order to launch other types of DoS attacks (i.e., race conditions, etc.). Such attacks are presented in Chapter 3.

TABLE 2.2: Basic SIP responses

Type of Response	Description
1xx	Set of responses used for provisional purposes
2xx	Set of responses used for success purposes
3xx	Set of responses used for redirection purposes
4xx	Set of responses used for client failure purposes
5xx	Set of responses used for server failure purposes
6xx	Set of responses used for global failure purposes

2.4 SIP message flow

SIP [6] is an application-layer signaling protocol for creating, modifying, and terminating multimedia sessions among two or more participants. Actually, SIP is text-based with syntax similar to that of HTTP. SIP messages can be either a request or an acknowledgment to a corresponding request, consisting of the appropriate header fields and optionally a message body, depending on the nature of the request or response. An example of a typical SIP request message is given in figure 5.2.

Whenever a user wishes to use a SIP service they should announce its presence by registering their current IP address to the registration service (registrar) through a SIP REGISTER message. After that, the user is able to initiate a session with another registered or interconnected User Agent (UA) by sending a SIP INVITE message to its local SIP proxy. After the call has been established, the two peers (the caller and callee) can start the multimedia session with the help of Realtime Transport Protocol (RTP) [37]. At any time, either the caller or the callee may terminate the call be sending a SIP BYE message toward the other end. These procedures are succinctly illustrated in figure 2.3.

Note that this kind of network logistic data pertaining to SIP calls are kept by default by all VoIP providers in order to fulfill important tasks including billing, network management and planning, security assessment, etc. So, independently of the file format each provider uses to store them, these raw data are solely consisted of SIP requests and responses.

2.5 SIP log files anonymization

The concept of the end-user privacy is very crucial for any service. In this direction, SIP-based applications can be assisted by appropriate tools that are able to automatically

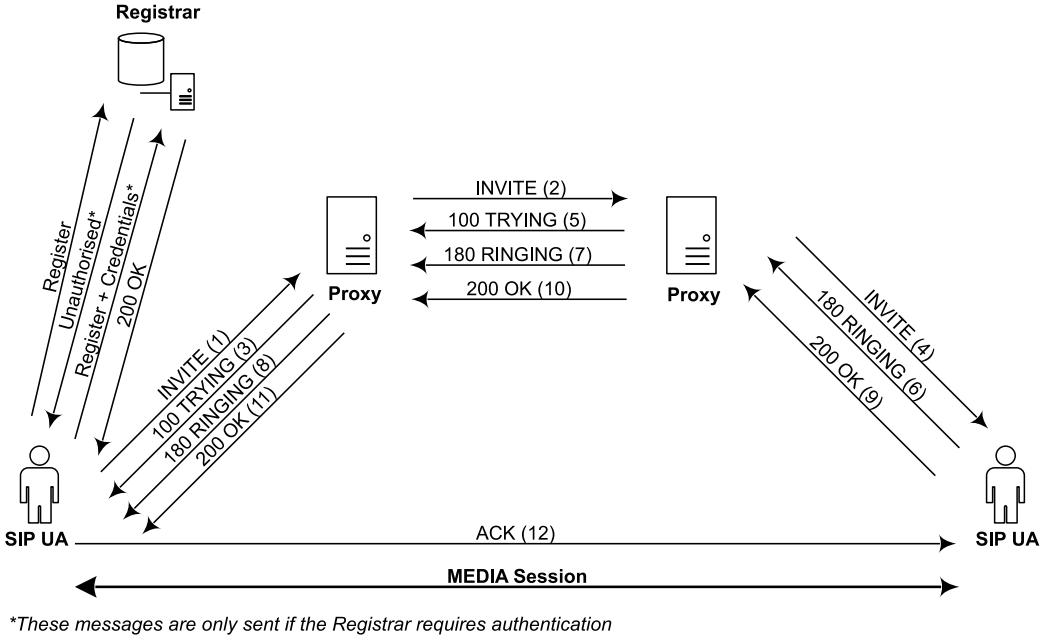

FIGURE 2.3: A simplified SIP message flow example

analyze the collected audit trails and determine in a formal way whether or not the provided service suffers a security incident, e.g., a flooding attack. In any case, these tools must meet specific requirements with respect to the end-users' privacy.

The privacy and security requirements for any VoIP provider are strongly related to the robustness of the anonymization process conducted on the log files. Therefore, there is a need for a fast-performing solution able to cope with realtime detection, but also strong enough to deter de-anonymization attacks when offline analysis is performed.

In this context, various techniques have been proposed in the literature for anonymizing data that include private information [46, 47]. Since anonymization and privacy cannot be considered as binary properties for any system, each solution is suitable for employment to a specific architecture and context depending on the requirements at hand. In the following, a brief description of such approaches is offered showing their advantages and disadvantages pertaining to our case. Note however that a comprehensive analysis of all possible anonymization schemes for preserving the privacy of network log files remains out of scope of this work.

Randomization was introduced in works [48, 49] for defending against zero-day attacks. A similar approach can be followed for data anonymization. To do so, data are transformed via the use of a randomized function. However, the transformation pattern should be also known to the third party that wishes to analyze the transformed data.

Generalization [50] divides data (e.g., sensitive headers per SIP message) into Quasi-Identifier (QI) groups, and changes their QI-values into less explicit forms, in a way that data in the same QI-group are indistinguishable by their QI-value (corresponding to the number of SIP message in our case). The philosophy of this method has been embraced and evolved toward forming more advanced ones including anatomy [51], permutation anonymization [52] and others.

K-anonymity is a privacy preserving approach [53] that constitutes k records indistinguishable. In this scheme a set of k data records are k|anonymized if for any data record with a given set of attributes there are at least k-1 other records that match with those attributes. To achieve this, sensitive attributes are hidden in order to obstruct leakage of real identities.

Symmetric encryption can be used to provide not only confidentiality services, but also data anonymization [54]. Sensitive record's attributes are encrypted using a secret key. The output looks random, while decryption is computationally infeasible, unless you know secret key. Symmetric encryption schemes can be used either in a deterministic way, i.e., the same input produces the same output for the same key, or in a semantically secure mode, i.e., a publicly known input modifies the output per encryption for the same key.

Message Authentication Codes (MACs) are symmetric schemes that are used for data integrity protection. When the plaintext space is much smaller than the tags space, the tag can be used as pseudonym. Similarly to symmetric encryption schemes, MACs are keyed constructions that can be either deterministic or non-deterministic.

Hash functions are instances of one way functions. They are very efficient keyless schemes that are computationally difficult to invert. Secure hash functions possess several nice cryptographic properties, like pre-image, second pre-image and collision security.

Searchable encryption has gained a lot of attention the last few years. It enables keyword search on encrypted data [55] by employing either symmetric or asymmetric cryptography. The asymmetric schemes are based mainly on homomorphic encryption and functional encryption, and so far they add excessive overhead. On the other hand, symmetric solutions are more practical. They are based on a combination of data structures and symmetric encryption algorithms. More precisely, the user, for a specific collection of documents, creates a corresponding index of terms (keywords) with the help of which one can execute queries. This process has inherently at least linear preprocessing complexity on the number of files.

To sum up, although each of the aforementioned techniques could be employed to impose log files anonymity, the selected anonymization technique must fulfill the following requirements:

- Users' anonymity, including SIP URI, IP address, etc. must be preserved.
- Perform fast both in realtime and offline.
- The Entropy of the original data after anonymization must remain intact.
- It must be computationally expensive to execute de-anonymization attacks.

Based on the above requirements the anonymization technique must be a property-preserving scheme and more precisely an Entropy preserving one. While randomization and generalization kind of solutions perform fast, they produce negative effects on symbols' frequency, thus affecting Entropy. Furthermore, to perform any analysis on the anonymized data requires the transformation method to be exchanged between the peers. Consequently, these schemes can be regarded as more complex, and naturally vulnerable to attacks, as this additional information is required to be stored in a secure manner. For example, in [56] it is noted that anonymization can have severe undesirable outcomes if implemented incorrectly. This negative effect is discussed by the authors in [57], showing that the correlation of anonymous data with publicly available Internet movie database information made possible to reveal the real identities of many of its customers. In this line, K-anonymity and its variants can be used to hide sensitive data effectively, but affect also the Entropy of the anonymized data. Finally, non-deterministic symmetric schemes (encyrption and MACs) are not Entropy preserving as well. The outputs are based on some input randomness and for the same input different outputs are produced.

Thus, among the proposed solutions, we have to choose one of the deterministic algorithms, hash functions, deterministic symmetric encryption schemes and deterministic MACs. All three of them are considered among the fastest cryptographic primitives and they are adequate for real time applications.

Regarding privacy protection, hash functions are the weakest ones. In our setting, the plaintext space is rather small. Thus, the keyless nature of hash functions make them vulnerable to brute force attacks. Let us assume an adversary who possess parts of the exchanged traffic (corresponding to a dictionary) and the anonymized data. Then, she is in position to execute a brute force attack aiming to match the dictionary records with the ciphertext and reveal, say, the URI of the end-users. To examine this possibility, we performed such an attack assuming that the adversary has built a dictionary of about 1 million records of different SIP INVITE requests for the same provider obtained, say, by eavesdropping. Using a laptop machine incorporating an i7 2.20 GHz processor and 6 GB RAM we managed to reveal a SHA-256 hashed header in 12.892 secs. On the other hand, the use of keyed solution, like a deterministic symmetric encryption or MAC algorithm, renders the application of a brute force infeasible.

In terms of performance, depending on the platform each technique excels. For instance,

encrypting the following SIP request INVITE sip:zisis@195.251.161.166 SIP/2.0 with AES-128 takes approximately 0.38 msec. For the same header, a keyed-hash message authentication code (HMAC)-SHA256 requires 0.29 msec, and a SHA256 digest 0.13 msec. We consider the mean value of 1,000 iterations executed on the same machine described above.

However, it is known that using deterministic algorithms makes the encrypted data vulnerable to frequency analysis. To minimize the effect of such an attack, we modify the parameters, i.e., the secret key, of the MAC algorithm per outsourced log file. Thus, using the HMAC-SHA256 scheme parametrised with a specific secret key for a set of log entries, the identical headers will produce the same digest and the Entropy will be preserved. For the next log file, a new randomly chosen key will be used for anonymization. In this way, an adversary is still able to process anonymous log records, but she cannot correlate different anonymised data sets.

Of special interest are searchable encryption schemes and especially, the symmetric ones. The last few years, several symmetric searchable encryption (SSE) algorithms have been proposed that have practical implementations [58]. However, the most efficient among them by design allow some leakage. More precisely, the leakage contain both the search pattern and the access pattern, i.e., as soon a keyword is queried, the server knows when the same keyword is accessed again. Thus, these schemes cannot protect from frequency analysis. In order to hide access pattern, SSE must use Oblivious RAM solutions, which are inadequate for real time applications, since they add polylogarithmic overhead (around $O(\log^2(N)$ per search, where N the number of data blocks) [59].

Summarizing, one needs to make a trade-off between performance and the level of anonymity offered. In this respect, the use of an HMAC scheme seems to be a fast-performing, well-respected and tested choice. Further analysis on the appropriateness of an HMAC scheme over that of a hash function in terms of server overhead pertaining to a real time detection service is given in section 5.1.11.

Chapter 3

DDoS attacks in SIP networks

3.1 Attacks in the SIP ecosystem

As already discussed, so far a plethora of attacks against SIP-based services have been identified in the literature. These include eavesdropping, flooding, SQL injection, manipulation of SIP messages, and so on [24, 25, 26, 27, 11]. One of the categories we address in this PhD thesis pertain to application layer DDoS attacks. These are caused by malicious entities who send a surge of SIP messages against their target, that is, a SIP server or UA. From an attacker's viewpoint, this category of attacks are considered quite straightforward mostly because of the text-based nature of the protocol, the lack of built-in countermeasures, and the existence of open source publicly available attack tools [60]. On the other hand, the impact of such an attack on the target is considerable and may vary from loss of service to entire network paralysis. From a VoIP provider viewpoint, this may result in many dissatisfied customers and loss of profit.

3.2 Types of DDoS attacks

Three main types of DDoS attacks are known in the literature. The first one concerns the volumetric attacks. In this type of attack the attacker aims on saturating the target bandwidth using for example UDP or ICMP floods. The impact of this category of attacks is measured in bits per second. The second category pertains to protocol attacks. SYN floods and ping of death belong in this category. The impact of the attack is

measured in packets per second. The last category relates to the application layer, and it makes use of requests which are seem to be innocent. The impact of the attack is measured in packets per second.

3.2.1 SIP-based application layer DDoS attacks

Figure 3.1 presents the high level categorization of DoS attacks we take into account in this PhD thesis. That is, in the upper level, the attacker can launch either a DoS or a DDoS attack. These types of resource consumption attacks can be launched due to a surge of syntactically correct signalling messages, or malformed ones. For each one of the previously-mentioned categories, there are 3 different attack strategies.

The first strategy requires the aggressor to send the attack messages using a slow pace. This strategy does not affect directly the target, but, instead, it introduces a small processing overhead which is may crucial for the underlying service. In the second type, the attacker sends an increased volume of traffic, which finally leads the victim to paralysis. The latter one is used as a mix of the other 2 categories. That is, the attacker initiates the attack using a low-rate traffic, and gradually increases the volume.

The importance of detecting and/or preventing these types of attacks is very crucial for all types of services. The SIP ecosystem requires the design and development of lightweight, efficient and easily deployable solutions, with the aim to deter the aforementioned types of attacks. The latter becomes even more crucial when one considers the increased attack surface in the context of SIP-driven solutions for public safety (i.e., NG9-1-1 emergency calling centers).

3.2.2 Other types of DoS attacks against SIP services

Many realtime systems build over state machines with the aim to provide a deterministic and stable functionality. State machines are usually operate in the context of a specific process in the system. Using this approach there are certain events which advance the state machine to the next state. In this direction, there are 2 paths of communication for the processes which are also well-known as the signalling managers of the system (SM). The first path of communication pertains to the network interface (i.e., the path of communication with the external entities). The second one relates to the inter-process

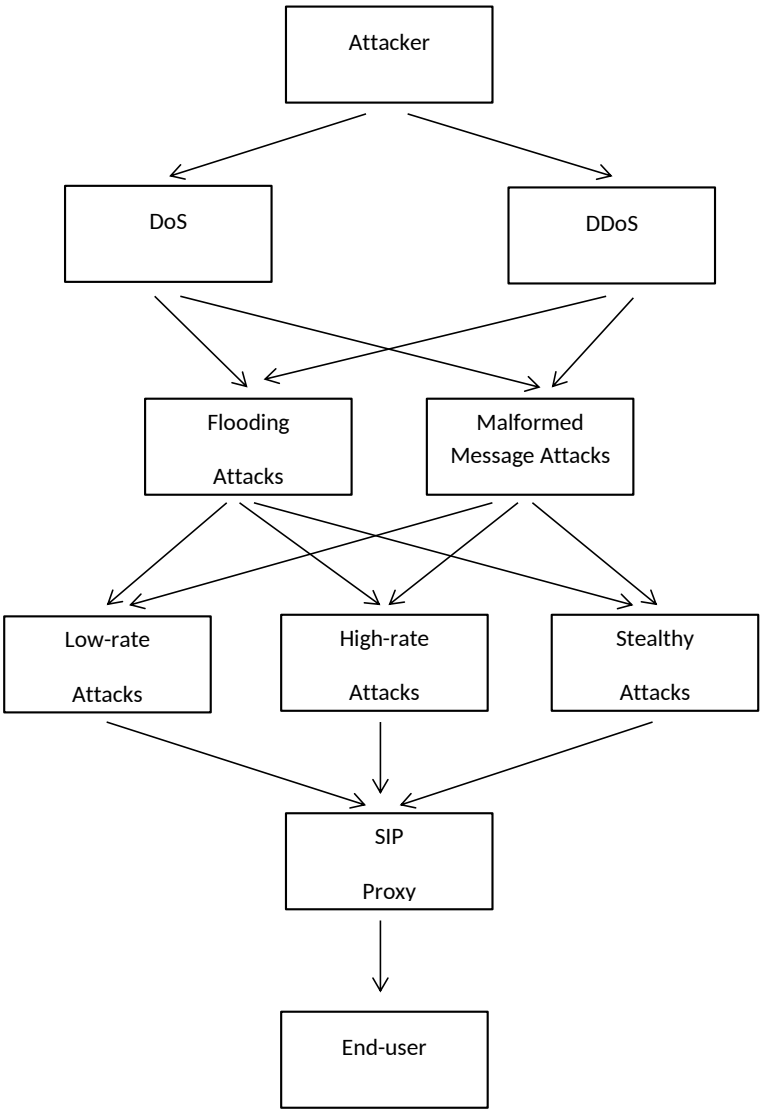

FIGURE 3.1: Categorization of application layer DDoS attacks in SIP and SDP

communication (i.e., the communication with the internal entities). In many cases the design of the system is performed using the assumption that the inter-process communication is faster than the external network entities. In this direction, there is a big chance that this hypothesis will fail.

As it has already been discussed, SIP networks are prone to a plethora of DoS attacks. One of these types pertain to race conditions. That is, taking into account all the above, an attacker could easily exploit the lack of state integrity between the different SIP entities (i.e., race condition) with the aim to cause a DoS attack. Figure 3.2 presents 2 network entities which exchange SIP messages. In this case we assume that a network entity sends an initial SIP INVITE request to the SM which handles the SIP messages

in the SIP server. Assuming a state machine implementation in the SIP server, the transition between the different states will be based on the messages exchanged between the processes and the SIP messages between the network entities. If the interprocess communication presents significant delays, then this might cause an unexpected behavior in the state machine handling.

There are certain cases where the network delays are negligible and the process communication is quite slow. These cases may occur either randomly or due to malicious activity. For example, figure 3.2 depicts process A which communicates with process B using Message 2. It is easily observable that process B responds in step 6 after receiving the SIP ACK message (i.e., Message 5). This inconsistency will probably cause a DoS attack in the system. That is, if an attacker is in position to affect both the message in the network and the delays in the inter-process communication, then she will easily launch a DoS attack.

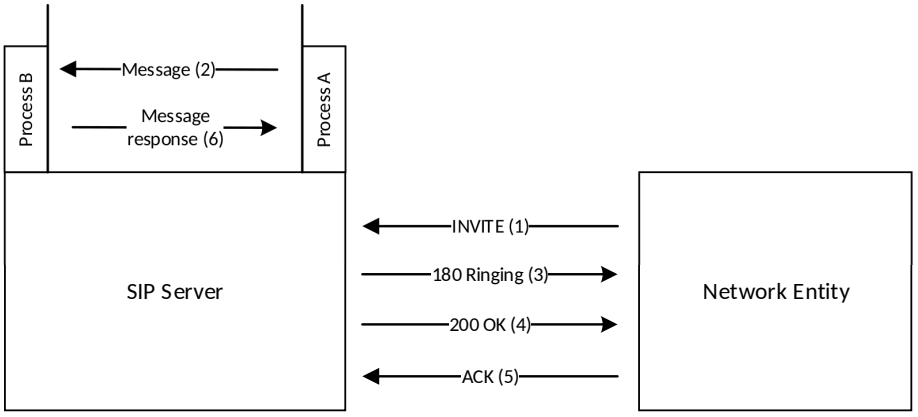

FIGURE 3.2: Race condition between SIP entities creates a DoS attack

3.3 SIP threat model

The formulation of a threat model in the context of this PhD thesis, has to do with two types of adversaries; external and internal ones. The former category includes malicious entities trying to cause DoS or collect information about the service. Such adversaries will act from the perimeter of the network, meaning that they have no direct access to the resources of the service itself. On the other hand, internal adversaries are assumed to be honest-but-curious and reside either in the service provider or within a third party to whom the provider has outsourced one of its security-related services. Prior to explaining

further, we make the hypothesis that the integrity of the log files is assured by the use of some well-accredited method [61, 62]. In fact, this requirement is a *sine qua non* for any service provider and it is also mandated by law in most countries worldwide [63, 64]. More specifically, the following assumptions are made:

Malicious external adversaries: The flexibility in message coding SIP offers can be of great advantage to an adversary when planning and executing, say, a flooding attack. Therefore, in this case, it is reasonable for one to assume a Dolev-Yao threat model [65] in which the adversary is among others able to eavesdrop, forge, and replay messages, and the only constraint is that of the cryptographic methods used. The latter, however, is not the case here as the tunneling of SIP traffic over, say, TLS or IPSec is not a widely-used practice, mainly due to the need of some sort of Public Key Infrastructure (PKI). So, for instance, the aggressor is able to launch a SIP INVITE or BYE flooding attack with the aim to paralyze the victim as reported in [66, 67] or execute a low-volume DoS to silently consume valuable network resources. This is for sure to gradually increase user discontent, which in turn leads to reducing provider's market share. Such type of assaults, especially the low-volume ones, may go totally unnoticed. In any case, however, the traces of the attack will remain hidden in the corresponding audit trails. Note that this kind of threats has to do with availability and integrity of the VoIP service itself, and do not focus on the (de)anonymization of log files.

Honest-but-curious third parties: While log files have special worth to multimedia providers for managing their network and billing purposes, they do not include only personal data but subscribers call history as well. Hence, due to the added value that such raw data have in terms of profit for different types of organizations, it can tempt any insider into gaining access to them. So, regarding the privacy preservation of the log files, we consider honest-but-curious (also known as semi-honest) third parties to which the service provider has outsourced the security analysis of its log files. Collaborating service providers who exchange log files in pursuit of shared goals also fall in this category. Insiders, that is, individuals working for the provider itself or a collaborating third party can also behave this way. This category of adversaries is supposed to have access to some version of the data and behave in a semi-honest manner. Namely, they might arbitrarily try to infer some additional information from the log files, but they obey the bilateral agreement in place as the case may be. Note that this category cannot be regarded as malicious because any insider attempt to corrupt the detection service is generally

detectable if the assumption on the integrity of the log files holds. The capabilities of such an adversary are included in the following:

- They might learn which services are being accessed by the end-users of some provider by just observing the information contained in <From> and <To> headers. This information can be used towards profiling certain users. As already mentioned in the previous section such privacy breaches clearly violate the principle of user anonymity [68].
- They might copy (steal) log files with the intention to sell them to, say, advertising companies for profit.
- An adversary working for a given provider has access to the audit trails of another provider, and/or the employee of a public analysis (detection) service is able to snoop on records contained in the raw data of one or multiple VoIP providers. This is of significant importance as the confidentiality protection of audit trails is of matter to the VoIP provider itself. For example, many providers would be interested in hiding data about which their most popular (accessed) service is.
- Following the previous issue, an adversary working for the detection service or a provider can correlate pieces of data contained in log files of different providers in order to deduce more information about the end-users of interest.

From the previous analysis it becomes obvious that the sharing of network logistic data among providers and between providers and third-parties (for outsourcing purposes) remain highly questionable due to the type of data contained in them. As already pointed out, this is of utmost important here as the security analysis service may be outsourced to an external entity and/or some collaborating providers may share their data based on a common agreement. This situation becomes even more complex, assuming service providers operating in different countries or continents (mainly due to diverse legislation and legal requirements applying to each particular country). Cloud-based operation, which is currently on the ascend, adds one more dimension to worry about.

3.4 SDP threat model

For analyzing DoS assaults caused by malformed SDP segments, we consider an adversary model which includes two main types of opponents; the ones who register with the SIP service (insiders), and those who remain unregistered (outsiders). The former category typically refers to honest-but-curious parties that establish and maintain some bond of trust with the VoIP service, while the latter to malicious external adversaries, i.e., aggressors who try to attack the service from the network perimeter. Specifically, insiders are normally considered to be trusted or semi-trusted, say, they possess a SIP Uniform Resource Identifier (URI) and the matching credentials to authenticate to the local SIP registrar. On the downside, as explained in [69], trust may be the key element for launching more silent type of attacks, without violating the communication protocol or causing any obvious damage. On the other hand, external evil-doers reside outside the local network, may muster an army of attack-bots (e.g., an IP stresser and/or zombie machines), and employ techniques like IP spoofing to minimize the footprint of their attack. This is straightforward for protocols like SIP which support both TCP and UDP. Both these types of adversaries are capable of eavesdropping on SIP traffic, creating SIP messages that contain a SDP segment, injecting data into the exchanged SIP messages, and ultimately launching two basic types of assaults as illustrated in figure 3.3:

- DoS attacks against specific or random UAs. That is, by manipulating SDP messages, the attacker attempts to cause DoS to the corresponding SIP entity. These messages can be crafted and sent at will to specific or randomly selected users or be part of an ongoing session between a caller and a callee, where the attacker acts as a man-in-the-middle. Additionally, the attacker may send such specially crafted malformed messages to a SIP server with the aim of paralyzing its message parser.
- The creation of *"hide-in-plain-sight"* type of communication channels used to secretly convey information via SDP.

In the first case, the SDP part of the message have been intentionally crafted with the aim to crash the SDP parser at the SIP phone. This is usually accomplished using a packet manipulation program [70]. Such a tool allows for the capturing and crafting of SIP messages. However, keep in mind that a malformed message may be also unintentionally

generated due to a software/hardware error. When the malformed SDP data are received by the target phone, three outcomes are possible. First, the phone may not be able to decode the SDP segment, and it will display an error message. Second, the phone will abnormally crash because for example its parser enters to an infinite loop. Lastly, the malformed data will go unnoticed. This typically happens because the malformed data lie in non-critical fields, i.e., those that do not directly affect the session establishment process. Overall, the effectiveness of this type of attack depends on which specific part (or parts) of the SDP segment have been manipulated. It is also to be noted that the same attack strategy can be used against SIP servers. That is, the attacker sends different SDP malformed messages to the SIP server in hopes of driving its parser to paralysis. Nevertheless, as discussed later, in section 7.3.2, at least two major SIP servers are found immune to this attack because they simply do not check the SDP part of the message (while they should). This means that the SIP server will simply forward the malformed SDP information toward its final destination without bothering to check its soundness. Of course, this negligence renders SIP clients highly prone to the same kind of attack.

As already pointed out, the second type of adversaries are assumed to use the SDP part of SIP messages to create a C&C communication channel as described in detail in [33]. Briefly, this scenario assumes that the SDP part of SIP messages are used for secretly exchanging commands with the sole purpose of coordinating a botnet. Therefore, in such a case, the attacker's goal, acting as a botmaster, is to realize a C&C without affecting the underlying infrastructure and running services. This is accomplished by malformed SDP messages that go unnoticed, i.e., do not induce an error condition on the target device.

Having in mind all the above, in the following we formulate an attacker-centric threat model. Specifically, we assume that the perpetrator is able to fabricate a SIP message by simply spoofing its headers. The most appropriate SIP requests to achieve such a goal are INVITE, REGISTER, and OPTIONS [6]. The attacker spoofs certain headers of the message (i.e., Request-Line, <Via>, <From>, <To>, <Call-ID>, <Contact>) to create a flooding effect towards the victim and obfuscate the forensic signal of the attack. For instance, if the attacker knows the URI of a certain user, is then able to mount a high-rate application layer DoS attack with INVITE requests to choke the user's softphone. An indicative example of a device that is prone to such a vulnerability is Cisco SIP Phone

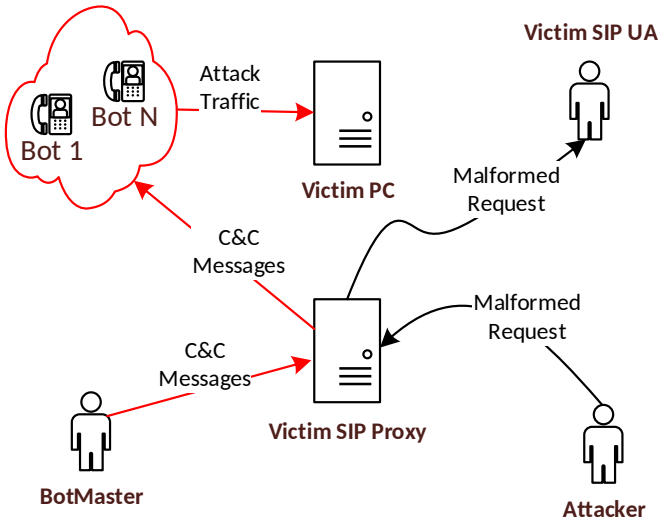

FIGURE 3.3: The malicious entity sends either malformed requests or (in red) communicates with bots

3905 [71]. An alternative attack strategy aims at paralyzing critical components of SIP infrastructure, including SIP Proxy or Registrar.

As already pointed out, various vulnerabilities have been presented so far in the literature concerning SIP [25, 11, 72]. The formulation of a threat model with respect to SDP-driven C&C channels, has to do with adversaries who try to capitalize on SIP as a covert channel. We consider two different cases depending on who controls the SIP Registrar with which the bots need to be registered.

In the first one, the botmaster controls the Registrar, e.g., she is the owner of this server or she has compromized it in some way. As a result, the botmaster is able of registering users with the SIP proxy. This way she solves the problem of randomly assigning and updating usernames to the bots. Moreover, she is capable of further eliminating the chances of getting detected by applying IP and Domain Fluxing to the SIP proxy without significant modification to the proposed architecture. In the case of IP flux, the botmaster would regularly alter the IP address pertaining to the Fully Qualified Domain Name (FQDN) of SIP Registrar by owning or controlling a group of PCs dedicated to that purpose. On the other, by applying domain fluxing, she continuously modifies and associates multiple FQDNs to the SIP Registrar. For example, every day, the botmaster could assign a new domain name to the SIP Registrar. These names might be generated by a hash function taking as input the current global date and a secret string. With the same way, the various bots could produce the domain name of a specific day.

The second scenario is the opposite of the former, i.e., the botmaster does not control the Registrar. In this case, the easiest workaround for the botmaster is to register the bots and herself to a SIP public service provider. A list of such providers is included in [73]. However, the problem of assigning usernames in this case may not be so trivial. The botmaster and consequently the bots must know which usernames are still available (not taken by other users). This requires either a public directory or a P2P protocol for sharing and updating a list which contains the already assigned usernames. Another more straightforward solution lies in exploiting SIP protocol requests to determine if a UA is alive. For example, an OPTIONS request could be used by the botmaster (or a bot) to identify if a username has already been assigned to another user. According to RFC 3261 [6], this request is used by a UA for identifying the capabilities either of another UA or a SIP proxy. Therefore, one could take advantage of this functionality to build a list of the already occupied usernames. A third option is for the botmaster to assign totally randomly generated usernames for the bots, but this may attract the attention of the proxy administrator. Such a list can be shared between the botmaster and each bot beforehand. Generally, it can be argued that the more realistic the usernames the less the chances of being detected as malicious.

In our case, we assume that the Registrar is in the possession of the bot-herder. We also hypothesize that the bots have been installed in the host machines following an infection. Nonetheless, this infection phase remains out of scope of this work.

A botnet can be considered as a network consisting of infected and compromized computers, called bots, zombies or slaves, which are controlled by an attacker known as the botmaster or bot-herder. A bot agent obeys every command received by its botmaster ordering it to initiate or terminate an attack. Botnets pose a serious threat to the Internet, since they are capable of disrupting the normal operation of services, networks and systems at will of their botmaster. For instance, botnets could be used for launching Distributed DoS (DDoS) attacks [74], sending spam emails on a massive scale, performing identity theft, distributing malware or even copyrighted material, and so forth.

Perhaps the most vital demand for maintaining control of the entire botnet is the ability for a bot to constantly stay in touch with its C&C infrastructure through a reliable and undetectable covert channel. That is, a bot will not be able to receive new instructions if the C&C cannot be located, and continue to probe the vanished C&C in vain. In

this direction, botmasters employ a number of techniques to not only minimize the probability of bots losing contact with their C&C infrastructure, but also to render their botnet more agile to hijacking and stoppage attempts. Depending on how the bots are remotely controlled by their master, i.e., how the C&C channel is structured, one is able to classify them into centralized, decentralized or hybrid architectures.

The centralized infrastructure is based on the client-server model, where all bots are directly connected with one or few C&C servers. These servers undertake to coordinate the bots and instruct them to take action. Although a centralized botnet exhibits optimum coordination and rapid dissemination of commands, it also poses a single point of failure. From the moment the C&C server is detected and deactivated the entire botnet is turned off. Usually, a bot-herder conveys its command through a well-known protocol. This way, she is able to hide the C&C traffic into a legitimate one. As a rule of thumb, the communication channels in this approach are based on HTTP or IRC protocol [75]. In the first case, the communication is disguised inside the normal Web network traffic as the usage of Web is allowed in most networks, including corporate ones. On the other hand, in IRC-based architecture the bots are connected to IRC channels and waiting for commands from the bot-herder. Of course, the messages on the IRC channel are in an obfuscated custom dialect, e.g., encrypted or hashed to avoid disclosure. In the context of this PhD thesis, a centralized infrastructure is employed, where one or more SIP proxies are responsible for dispatching the commands to bots. Furthermore, we are not based on the aforementioned protocols, but rather we utilize SIP as a covert channel. Although, centralized approach seems easily detectable, the botmaster is capable of evading defence mechanisms by applying fluxing techniques. As detailed later in Chapter 7, section 7.2, fluxing allows the aspiring botmaster to frequently change the IP and/or the domain name of the proxy.

Alternatively, a decentralized architecture may be selected to carry out the C&C mechanism. In this approach, there is not a central C&C server, but rather the various bots communicate with each other via Peer-to-Peer (P2P) protocols. In other words, the bots behave as C&C server and client at the same time. Therefore, if any of the bots is tracked down and deactivated, there are no implications to the robustness of the entire network [76]. The hybrid architecture combines the advantages of both the centralized and decentralized ones. That is, in this setting, the bot agents exhibit diverse functionalities. Some of them, temporarily undertake the C&C server role, with the aim to coordinate

the botnet and disseminate the instructions, while the others wait for commands before springing to action [77].

Chapter 4

Literature review

This Chapter provides a comprehensive survey in the area of security & privacy in SIP, with special emphasis on application layer DDoS, malformed messages, and C&C channels. Furthermore, among the different security mechanisms presented in the Chapter, the first part is focused on works which capitalize on log files with the aim to track security incidents.

The researches presented in this Chapter have been classified in 3 main categories based on the way they manipulate their inputs. In this respect, we categorized the works based on the context they process and analyze the input data. The first type of class namely "online", pertains to works which process the data the same time they arrive on the system. On the other hand, the "offline" category refers on researches which operate over stored data. Finally, works which lack a real implementation are classified under the other class.

Concerning the chronological range, we picked works spanning from 1998 to 2018. This time period has been applied only to the proposals which rely on log files. For all the other categories we touched works presented from 2012 to 2018. This has been done with the aim to avoid an overlapping with existing researches which are mature and complete in this research field [11]. Figure 4.1 provides an abstract overview of the literature map. In total, 91 researches have been collected and analyzed in the context of this PhD thesis.

The next subsection provides the background of intrusion detection systems with respect to their detection approach.

4.1 Introduction to intrusion detection systems

Intrusion detection systems (IDS) can be categorized under 3 basic classes based on their detection approach. These are the following ones: anomaly detection, misuse detection, and specification-based detection [78, 79, 80].

Anomaly detection [81], [82] relies on the detection of deviations from a predefined threshold. The latter is usually computed over an attack-free training dataset. After extracting the thresholds, the testing data are examined for deviations from the normal traffic. An important issue in these types of systems pertains to the attack-free data assumption [83]. Based on this assumption, the normal traffic which is used as a base to extract the legitimate traffic thresholds, must be attack-free. In a different case, the anomaly detection will not be able to identify the attack patterns.

Misuse-based detection [84], [85] relies on specific patterns of the attack traffic. This means that the examined traffic is compared over a set of predefined rules or signatures. In case the IDS finds a match, the attack pattern is verified. While this category reaches a 100% of detection accuracy, it introduces the limitation of detecting only known patterns of attack traffic. Systems which reside in this category cannot detect new attacks.

Specification detection [86] builds over the fact that each entity in a system must conform to specific actions (i.e., security policy). Any entity which deviates from this set of normal behavior is considered suspicious.

Table 4.1 summarizes the 3 aforementioned types of IDS and highlights the advantages and the weaknesses of each one. That is, the first category is considered a good ally in tracing new attacks with an increased number of false alarms. The second category pertains to systems which are in the position to only detect an attack if it matches a signature. From this point of view these systems are considered monolithic. On the other hand, the process of updating the rules is fast, and thus it is considered much faster in contrast to anomaly detection. Finally, the last category is considered as a top performer in the false negative performance. The main disadvantage in this category relies on the difficulty of generating formal specification.

TABLE 4.1: Intrusion detection systems

Type of IDS	Pros	Cons
Anomaly detection	Detects New attacks	Increased False alarms
Misuse detection	Fast rule update	Cannot detect new attacks
Specification-based detection	Low false negative	Effort to generate formal specification

4.2 Exploitation of log files to detect security incidents

Network audit trails, especially those composed of application layer data, can be a valuable source of information regarding the investigation of attack incidents. In this respect, the fast and accurate analysis of those files, is very critical because they may contain pieces of attack data which have not raised the security alarms.

Motivated by this fact, 2 years before the beginning of the 3rd millennium, Schneier et al. [62] presented a computationally efficient scheme to preserve log file integrity and confidentiality in compromised machines. The authors rely on the fact that log files are created using secret keys which already exist in untrusted machines. These keys are used in untrusted machines, with the aim to communicate with trusted verification machines. According to the proposed method, the overall security level is summarized in 4 different steps. The first one relates to a hashed authentication key. The second one concerns the creation of the encryption key. The third one is related to the authentication of all the previous entries in the log file using a hash chain. The latter is produced in a one to one mapping from the authentication key of each log entry. Finally, the last one relates to role-based security systems, and the capability of partial access. This proposal presents only the theoretical model and thus it is categorized under the other category.

Two years later, Biskup et al. [87] researched the area of transaction-driven pseudonyms. The authors exploited a threshold-based approach, driven by the concept of the "legal purpose". In this direction, they provided an approach to retrieve the data subject identity, upon exceeding a specific threshold. Additionally, the proposed design was driven by Shamir's approach [88]. More specifically, based on the proposed scheme, the condition to retrieve the subject's identity pertains to the excess of a threshold related to the number of the pseudonymous actions, and a predetermined purpose. The proposal details on the proposed scheme only in a theoretical level, and thus it is categorized under the other class.

The same year, Sah [89] introduced a new way to manage a high volume of enterprise log data by introducing a novel LMS. The latter supports operations such as the compression, the management, and the analysis of log files. The authors verified the performance of the proposed scheme using five PC's running on RedHat 7.1. According to the results they obtained, the proposed data management platform exceeded the processing of 20,000 number of weblogs records per second. This proposal is categorized under the other class.

One more work in the field was presented the same year by Flegel [90]. This research presented a set of tools devoted to the pseudonimyzation of log files. As the authors state, they performed the necessary actions with respect to the requirements for anonymity from the user's perspective, and the requirements for accountability from the service provider. According to the introduced threat model and the presented architecture, the proposed design fits well with existing Unix systems. This proposal is categorized under the online category.

No year later than 2004, Rouillard [91] presented the Simple Event Correlation (SEC). Using SEC, the authors contributed on the reduction of false alarms, introduced by the process of automated log file analysis. Additionally, using a set of rules, they aimed to maximize the efficiency to problem response. A main disadvantage highlighted by the authors pertained to the performance with respect to the throughput. More specifically, the rule-driven approach of SEC introduced a linear complexity with respect to the search operations. Aiming to eliminate this drawback, the authors proposed to optimize the solution by moving the most used rules on the top of the list. This work is classified under the online class.

The same year, Xu et al. [92] proposed a correlation approach, driven by the concept of "low-level" events. According to the authors, these types of events comprise the triggering mechanism for generating alerts in a system. Additionally, the system triggers alerts which may finally fit into the same attack category or cluster. Furthermore, they detailed on the input and output resources. Using the resources, one is able to correlate different types of attacks and extract attack scenarios. Based on the experimental results, the alert clustering created 512 clusters. Among them 17 pertained to 2-alert clusters while the rest to single ones. Finally, based on their findings and with respect to the attack reconstruction capability of their proposal, the proposed method succeeded

in constructing 10 attack scenarios. This work is categorized under the other class. This is because the authors exploited a simulator in order to simulate the behavior of sensors and obtain the alert streams.

Two years later, Lincoln et al. [93] introduced a new scheme with the aim to research the concept of log file sharing. According to the authors, the proposed scheme does not require trusted third parties, while it is solely based on multiparty computation schemes. Additionally, it does not require sophisticated key management. Based on the collected results, the authors reported an overhead equal to 110.34 and 106.20 secs per 1 million records of alerts. These results considered the applicability of SHA-1 and HMAC algorithms on source and destination IPs'. Finally, the results have been perceived using a controlled lab environment. Based on the operation classification this work is characterized under the online category.

The same year, Slagell et al. [94] conducted a survey on the proposals pertaining to log file sharing. In this direction, the authors elaborated on the problem definition and they provided a number of obstacles pertaining to large scale log file sharing. More specifically, the authors highlighted the need of a protocol agnostic framework for log file sharing. Additionally, they pointed the need for anonymization and balancing between security and usability. This work is presented only on a theoretical level, and thus it is classified under the other class.

A year later, Godínez et al. [95] exploited n-grams with the aim to reduce information parasitizing in audit trails. The motivation behind this work stemmed from the training overhead introduced in models like the Hidden Markov Model (HMM). Based on the authors this cost is proportional to the size of log files. In this direction, the authors offered a method to reduce the log file size. Their proposal relied on the process of grouping subsequences of system calls which appear with high frequency. Based on the results they obtained, the authors reported an improvement which ranges between a factor of 3.6 and 4.8 in terms of audit trail size reduction. Finally, the proposed method offers an increased performance without affecting the detection accuracy. This proposal resides under the offline class.

One year later, Stathopoulos et al. [96] presented a framework for secure logging in public communication networks. The authors conducted a study on the research area, considering a threat model devoted to insider attacks. The proposal is based on a trusted

"Regulatory Authority" which is responsible for the integrity of log files. The authors assumed a semi-trusted environment and they defined 6 different phases. The first phase corresponds to the definitions of the networks and the operational events. The second one concerns the requirements of log files. The 2 next phases touch the security measures employed for combating the external and internal attacks correspondingly. In the next phase they deal with the design of implementation. Finally, the last phase revolves around the log file verification procedure. This work is categorized under the other class.

The same year, CP. Lee et al. [97] introduced a forensic-driven analysis tool namely FlowTag. Based on the authors, the proposed software has advanced characteristics compared to other traditional tools. Such characteristics pertain to the quick analysis, the reporting and the sharing of the attack data. All these features are driven from the property of tagging specific flows. Using this approach the authors stated that one can easily grasp a better understanding of a given network file. This seems to be especially helpful for analysts as they are able to maintain the context without saving details. Additionally, the authors highlighted the advantage of switching over different tags really fast. Finally, the advantage of using tags is highlighted by the authors in the process of generating corresponding reports and exporting "attack packages". This proposal is grouped under the other category.

Casey et al. [98] conducted a study on the investigation of sophisticated security breaches. After studying the topic, the authors highlighted the importance of the immediate and on target actions. The latter refer to actions performed by the investigators. In this direction, the authors put special attention in the process of collecting digital evidence. Additionally, they focused on the understanding of the actual breach. Furthermore, special emphasis has been put on the attacker tactics which may include digital evidence . For this reason, they stressed the need for speedy actions with the aim to preserve the digital evidence integrity. This is so in order to perform advanced analysis using forensic tools and techniques, and finally to seize the attacker without applying "hacking back" techniques. This work is grouped under the other category.

Escobar-Jeria et al. [99] provided a review on the main applications of Fuzzy logic to web mining. Initially, the authors elaborated on Web usage mining and they highligted the process of constructing user profiles in the web. Furthermore, they provided results regarding the applicability of Fuzzy Association Rules and fuzzy clustering in Web Server

log files. The latter has been done with the aim to extract usage patterns. After analyzing a bunch of CSV log files, they came up with a confidence factor which reaches 1. This work is categorized under the other class.

Saleh et al. [100] proposed a model checking approach for the formally forensic analysis of log files. The author exploited a tree structure aiming to model a set of logs from a specific system. To accomplish the previously-mentioned goal, they employed algebraic operations. The latter supported formalization of the actions parasitizing in the log files and captured by the logging system. In order to highlight the advantages of the tree-driven approach, they referred to the easiness of correlating Windows, system and network logs into the same tree. After defining the patterns of traces, the forensic investigator is able to submit a hypothesis in the form of a property, in the component namely "model checker". The latter will perform a search operation for these patterns over the tree data structure. This proposal is grouped under the other class.

Monteiro et al. [101] used a variation of the Needham Schroeder protocol [102] in order to authenticate and validate syslogs. The authors examined the authentication of the records upon their creation. In contrast to the existing works in the field, the examined work does not rely on the reactive forensic analysis of the log file records, but instead, it relies on fingerprints, with the aim to achieve the authentication of the file contents. The proposed method consists of 6 phases as follows: 1. User authentication, 2. System connection establishment, 3. System connection establishment response, 4. Application event entry generation, 5. Applications termination, 6. System connection termination. Using the proposed method the authors managed to protect log files against truncation and man-in-the-middle attacks. The latter has been done with the aim to preserve the forensic viability of log files.

The proposed scheme includes a server which receives fingerprints from the different entities (i.e., users', applications', system). Additionally, fingerprints are created using the algorithm RS [103]. Using this approach, the files meet the following requirements: a) the fingerprints provide integrity, b) the authentication is reassured, c) the files can be used as evidence is a court.

The metrics are used aiming to quantify the ability of applying forensic analysis on a file. The phase of detection relies on the fact that normal pieces of data have already been fed into the system. Using this approach, the system generates the detection thresholds.

Moreover, increased values denote that a forensic analysis is viable and fruitful. On the contrary, low threshold values, indicate that the pieces of data included in the log files are not enough to perform the analysis. This work is categorized under the other class.

Goel et al. [104] introduced a technique for reducing the size of audit trails. The authors accomplished this task by simplifying the process of query analysis. Additionally, they enabled the execution of analysis tools interactively. The previously-mentioned drawback seems to create problems in the analysis from a system under investigation based on their auditing system. As they state, the large volume system-level audit trails create problems to the analysis tools. Aiming to bypass this problem, the authors employed interval tables for storing the lifetimes of system objects or their attributes. According to the authors, these pieces of data seem to play a vital role in determining the system state. As far as it concerns the overhead of the proposed method, the authors state that the necessary queries are 4-5 times slower without implementing the interval tables. The aforementioned results perceived under a ftpd attack. The latter required less than 10% of the overall loading overhead for their creation. This proposal is classified under the online category.

The same year, Xia et al. [105] conducted a research on the reconstruction of probable sequences of events created on the journaling process in the Ext3 file system. This process makes use of program behavior signatures. The proposed architecture exploits the metadata residing in the Ext3 file system in order to accomplish the process of reconstruction. Additionally, as the authors state, principal investigators can be helped by the proposed architecture using a program behavior database. The latter consists of data extracted from system call monitoring and metadata archiving. Finally, as the authors report, computational overhead is not given for the proposed architecture due to lack of optimization. This work is categorized under the online class.

One year later, Pun et al. [106] provided a integrated framework for evaluating application servers. More precisely, the authors investigated the key factors affecting intensive web applications. The proposed framework introduced three main components. The first one pertained to the audit trail analyzer, the second one to the web server log analyzer, and the last one to the stress testing tool. The audit trail analyzer concentrates on the business logic of the server. The web server log analyzer deals with the transactional details. Finally, the stress testing tool provides an overview of the server capabilities

with respect to the external volume of traffic. In this direction the authors focused on the result integration, collected from audit trails from the application and web servers and the stressing tool. For this purpose they exploited 20, 18, and 7 different statistic characteristics. This work is categorized under the offline class.

The same year, Barradas-Acosta et al. [107] presented a system for evidence classification. The authors proposed an algorithm building over the recurrent neural networks (RNN) classifier. Using this algorithm they aimed on reducing the cost of analyzing big volumes of traffic residing in log files. Based on the proposed architecture, the proposed system comprises of a traffic analyzer devoted to the data parsing operation. Additionally, it offers the preprocessing operation, used for extracting the useful pieces of information. The latter is applied directly in the log file, and the data are fed to RNN for the learning phase. Finally, the RNN is used in order to obtain a "possible evidence". This work is categorized under the offline class due to the fact that the input the data are fed to the system from the log files.

Myers et al. [108] conducted a study on insider threat detection. The authors employed a method for detecting insider attacks using only web server logs. Based on their findings, the increase number of events and devices improves the detection of insider attacks. In any case, they recognized the complexity and compatibility issues that arise from this approach. Finally, their next steps focus on the analysis and the correlation of log files. This work is supported only on a theoretical level and thus it is categorized under the other group.

A year later, Nagappan [109] introduced a component-driven approach for the analysis of log files. The authors emphasized on cloud systems. The proposed solution uses a range of algorithms and tools, with the aim to find the optimized combination. The latter is supported using a set of components. The first one pertains to the log abstraction component which deals with the contents of the log file and extracts a canonical form of the collected data. The second one deals with the common format that exists in every log file, using specific pieces of data. The next component operates on the analysis techniques and the corresponding tools. The author defines the classic requirements for the framework evaluation. In this direction, the complexity in terms of time and space and also the precision are the major pillars for evaluating this proposal. This work is categorized under the other category.

One year later, Mazza et al. [110] proposed a complete framework to achieve accountability. The authors highlighted the contribution of their work with regards to the production of evidence for a legal dispute. More specifically, the proposed model builds over B-method, and relies on agents for the organization of the log files. The latter, emphasizes both on the data and the behavior of the system entities at the transaction phase. That is, the authors make the assumption that they are in a position to know the source of the exchanged packets. Additionally, they assume that log files do not contain inconsistencies or wrong records. In order to achieve the goal of non-repudiation, they make use of digital signatures. Moreover, a set of agents monitors these actions and records the events in the form of a chain. Using this approach it is possible to represent the event history, assuming that the agents are in the position to come to an agreement. This work is grouped under the other class.

Six years ago, KH. Lee et al [111] introduced an audit logging system called LogGC. The proposed system focused on the reduction of the log files size, taking also into account the garbage collection process over the log files. Based on the authors, the proposed system reduces the log size by 14 and 37 times without compression, when one considers regular and server systems correspondingly. As the authors indicate, this performance is reached without affecting the accuracy of forensic analysis. This system is categorized under the offline group.

Yen et al. [112] presented a system, namely Beehive, for the automatic mining and extraction of knowledge from large enterprise environments. The authors emphasized on the high volume of data which is generated from the variety of different security products. The proposed system relies on a set of operations for the processing of log files, and the noise removal. These operations preserve the accurate timestamps and retain a consistent ip-to-host mapping for locating devices which change IP address frequently. Using the proposed system the authors reported a 74% reduction on the data inspected by the proposed system (i.e., from 300 million to 80 million log messages per day). This system is classified under the online class.

Additionally, King [113] introduced a new system to achieve non-repudiation. After introducing the minimum characteristics which must appear in a log file, the authors presented the statistical metrics which need to be employed in order to ensure non-repudiation. The

corresponding actions concern, the creation, the projection, the alteration, and the deletion of the system resources. Furthermore, the authors took into account security-related actions which pertained to the user rights and the authentication process. This work is classified under the other class.

Singh et al. [114] detailed on the exploitation of log files to detect web attacks. Finally, the authors provided a comparison between the buffer overflow and the iframe injection attacks. In this comparison they detailed on the amount of the data transferred in these types of attacks. Based on the obtained results, the buffer overflow attack is much more devastating in contrast to the iframe one. In this direction, the attack impact is proportional to the attack input. This work is categorized under the offline class.

The next year, Gu et al. [115] introduced a anomaly-based detection system building over supervised statistical learning, for thwarting camouflaged attacks. The authors highlighted the difficulty of purely statistical methods to combat these types of attacks. Based on the authors, the main problem introduced in these types of attacks, pertains to the process of computing accurate values of the normal thresholds. Moreover, this goal is difficult to be achieved due to the noise which is injected to the traffic. For this reason the authors exploit control flow graph (CFG) of each application with the aim to reduce the noise. This work is categorized under the offline category.

The same year, Ambre et al. [116] elaborated on insider attacks by introducing a log/file monitor filter. The proposed solution offers both misuse-based log analysis and event correlation. The proposed mechanism comprises of 4 different modules as follows: The first one pertains to the log collection, the second is responsible for the log analysis, the next one performs event correlation, and the last one calculates the corresponding probability. According to the authors, their solution detects DDoS attacks using the random matrix theory. This work is classified under the offline class.

The same year, Juvonen et al. [117] investigated the applicability of 3 different methods for reducing the dimensions of log files. The methods that have been touched by the authors are as follows: random projection (RP), principal component analysis (PCA) and diffusion maps (DM). According to the results they obtained from HTTP analysis, they proposed to exploit the first and the third techniques together. Furthermore, the authors highlighted the results they obtained using real-world traffic, even using only the message headers. More specifically, after analyzing a log file with 2693 of records,

they report a performance penalty, in terms of time, which fluctuates between 2.4 secs, for the RP methodology and 196.6 secs using DM. This work is categorized under the offline class.

One year later, Moh et al. [118] introduced a two-tier system for analyzing log files with the aim to detect and prevent new attacks. The authors focused on SQL attacks. The proposed architecture relies on 3 main components. The first one pertains to the logging system, the second one details on the data preprocessing, and the last one focuses on the detection methods. The combination of using pattern matching and Bayes Net model gives an accuracy reaching 94.7%. In general, using the proposed architecture, the authors reported a maximum detection accuracy equal to 95.4%. This research is categorized under the offline class.

Finally, the next year Breier et al. [119] researched the concept of analyzing log files for the detection of new attacks. In this direction, the authors investigated a set of data mining techniques for the dynamic generation of rules. Additionally, the authors employed data reduction techniques with the aim to handle the huge amount of data. Based on the results they obtained, they report false alarms rates which do not exceed 10%. This work is categorized under the offline class.

4.3 Detection of DoS attacks

This section details on works which focus on the detection of resource consumption attacks using statistical means.

4.3.1 Statistical schemes

A subset of the statistical schemes presented in this subsection rely on the Change Point Analysis (CPA) tool [120]. One of these works applied this solution almost seven years ago by Chen et al. [121]. The authors worked on the detection of flooding attacks in IMS, by presenting a solution which builds over the non-parametric CUSUM algorithm. In essence, the proposed solution relies on a changing point detection method for fast tracking of alterations in the examined pieces of data. In this respect, the authors model the attack traffic considering the REGISTER flooding attack as an example.

More specifically, to apply this method, the authors track the differences between the SIP REGISTER requests and the corresponding 200 OK responses. Additionally, they employ the IMS platform with the aim to examine the detection accuracy of the proposed solution. Using a sampling period of 5 secs, they obtained a threshold equal to 30. After launching a flooding attack with a rate between 200 to 300 cps, they reported only 1 false alarm. This work is categorized under the offline category.

The same year, Stachtiari et al. [122] studied the field of admission control policies, with the aim to cope against DDoS attacks targeting CAPTCHA protection mechanisms. More specifically, the proposed approach defends against attackers which aim at causing DoS to CAPTHA systems, by draining the bandwidth of the protection mechanism. In essence, the authors provided an additional layer of protection to anti-SPIT mechanisms which pertains to admission control. More specifically, their solution pertains to 4 different probabilistic models which have been developed on the PRISM model checking toolset. Each model presents a VoIP server under DDoS attack. Additionally, they authors researched the cost and net benefit metrics, using reward structures. The first metric details on the probability to reject an incoming request and the bandwidth which has been lost during a DDoS attack. The second one represents the probability to accept an incoming request. Based on the obtained results, the reservation of 60% of bandwidth in contrast to 80% saves unexploited bandwidth. Finally, the probability of accepting new requests is increased. This work is classified under the other category.

The same year, Dassouki et al. [123] introduced a mechanism to combat against signalling attacks in SIP. The proposed mechanism focuses on the authentication of SIP messages with the aim to prevent message manipulations. In order to achieve this goal, the authors exploit message fingerprints. As the authors state, the computational resources for the proposed solution are driven by the cryptographic operations and the length of the SIP messages. Additionally, the authors provide a comparison of the proposed work to RFC 3893 [124]. Based on the results they obtained, they state that the proposed mechanism achieves 0.2 less CPU cycles compared to those obtained from [124]. Moreover they present the different results, in terms of CPU cycles, they obtained for the rest of the scenarios, and they report a 33% and 66% reduction respectively. This work is categorized under the online class.

Two years later, Zargar et al. [125] presented an anomaly-based approach to prevent

and detect DoS attacks in SIP. The proposed approach builds over Entropy theory and makes use of a compressed summarized table of the packet data, i.e., a three-dimensional Sketch structure. As the authors state, their method has been tested on the Spirent server. According to the results they obtained, the acurracy of the proposed scheme seems promising for both low-rate DoS attacks and their distributed form. Moreover, based on the authors, the proposed method induces negligible overhead which does not exceed the testing time over 1.5 times. More specifically, based on their results obtained from 3 different data sets, the false alarm rate fluctuates between 2.1% and 4.2%. Finally, the authors performed a comparison of the proposed approach to the Hellinger Distance. Based on their results the false alarm rates for the Hellinger Distance solution fluctuated between 12.8% and 17.8%. This result indicates that the proposed Entropy-driven method outperforms the Hellinger Distance one. This work is categorized under the offline class.

J. Tang et al. [126] researched the area of DoS attacks in SIP. The authors offered a detection and prevention system based on Hellinger distance (HD) and the sketch data structure. The proposed system exploits the sketch data structure in order to represent the different attributes presented in a SIP message. This is achieved using two-dimentional hash tables to store the different attributes. The authors exemplified this architecture using 4 different tables for each one of the investigated SIP attributes. In this direction, they highlighted the system performance to detect multi-attribute flooding attacks. After the generation of the sketch structures, the authors compute the HD metric over the stored hashes for the training and the testing period. According to the results they obtained, the authors report a detection accuracy spanning between 88% to 100%. This work is categorized under the online category.

Under the same pillar, J. Tang et al. [32] presented an approach driven by Hellinger Distance. In essence, the authors enhanced their previous solution presented in [126]. The main difference from their previous work pertains to the detection of DDoS attacks. Based on the authors, this proposal aimed on detecting low-rate DoS attacks with dynamic background conditions and multi-attribute traffic. As a drawback, the authors highlighted the weakness to combat against stealthy attacks. This is caused due to the limitation of the proposed method to create significant deviations in the Hellinger Distance between the normal and the attack distributions. Finally, based on the obtained results, the maximum computational cost generated by the Hellinger Distance and the

multi-dimentional sketch does not exceed 0.82 secs. Additionally, concerning the detection accuracy, the false alarm rate seems to be zeroed when K equals 32. Moreover, it does not exceed 7.81% when K becomes equal to 8. The parameter K corresponds to the attribute hash-table size. This work is categorized under the online category.

The same year, J. Lee et al. [127] researched a scheme for the detection of SPIT and DoS attacks. The authors considered DoS attacks which are triggered due to increased volumes of traffic. The latter may contain and malformed messages. More specifically, the DoS attack detection module is based on the completeness of the sequence of messages that need to be exchanged between a UAC and a UAS bi-directionally. The correlation of the various SIP messages is done by analyzing the L7 information residing in the SIP headers. In the case of SPIT, the module makes use of a DB to store users' past call patterns. As the authors state, the SPIT detection if performed with a small number of false positives equal to 0.16%. On the other hand, the DoS detection rate reaches 98.05%, when the traffic of SIP messages equals 400 packets/sec. Finally, the same module reaches 100% accuracy when a malformed packet filter is applied to the proposed scheme. The proposed scheme is categorized under the online category.

The same year, Tsiatsikas et al. [128] detailed on DoS attacks in SIP. The authors presented a privacy-friently scheme, building over the Hellinger distance metric, with the aim to trace DoS assaults. The authors reported a FP rate spanning between 0.2 to 7.6%, and a FN rate which does not exceed 0.002%. This work is categorized under the offline class.

A year later, Tsiatsikas et al. [34] proposed an entropy-driven framework for battling against DoS attacks in SIP. In essence, the authors extended their previous work introduced in [129]. As the authors stated, their scheme operates both as a standalone application and a software module destined to SIP proxies. In the first case, the framework performs fundamental parsing operations in audit trail files, using 6 different features. In the second case, the same anonymization scheme is employed on the fly in each incoming request. The processing of the traffic in both the offline and the realtime software modules is performed in a privacy-friendly manner, after employing a HMAC-SHA256 algorithm over the data. In both cases the Entropy calculations are performed over a set of messages, using a specific message window which is pre-configurable. The authors exemplified their results using the SIP INVITE request, but as they state the proposed

scheme can be applied in any SIP request. According to the results obtained from the analysis, the scheme introduces negligible time overhead. Moreover, it presents a detection accuracy which remains lower than 2% for the FP, in the case of the realtime processing. On the other hand, for the offline module it fluctuates between 0.3% and 10.6%. In the same direction, the minimum FN for both modules remain low, almost equal to 1%. For the maximum FN, the scheme presents a significant difference, almost equal to 24%. Finally, the mean overhead introduced from the anonymization scheme, does not exceed 16.8 msec for all the scenarios. This work belongs to the online class.

Semerci et al. [130] elaborated on DDoS attacks by providing a scheme that is capable of detecting alterations in the network flow. The scheme is based on changing point analysis. The same approach was followed by Kurt et al. [131]. More specifically, the authors delivered a framework building over the Bayesian multiple change model. Additionally, aiming to evaluate their proposal, they provided a probabilistic SIP simulation system exploiting also F-score. The latter relies upon precision and recall measures. Furthermore, it indicates a good performance when the value converges to 1. The best average score equals 0.94 and is given with a lag value of L=5. As the authors state, the increase of L parameter more than 5, does not necessarily indicate an increase in the F-score. Finally, they compare their framework with a distance-driven method, building over the Helinger distance method. When the distance method reach the true positive values, they report a 15-20% false positive ratio. This work is classified under the offline category.

Three years ago, Golait et al. [132] researched the area of flooding attacks in SIP. The authors exploited the Poison distribution with the aim to generate normal profiles. The latter are generated using a set of statistical observations, extracted from the original dataset. The proposed scheme follows the same logic in both phases of training and testing. Thus, in the testing period the same profiles are generated and compared to those obtained from the training phase. The proposed system consists of the following components. The first one pertains to the probabilistic distribution of the training data. The second one details on the threshold probability δ for event. The last one is devoted to the Number of Events in T Time Period from Testing Data. Based on the results they obtained, the authors report a 100% detection accuracy for all the scenarios. This work is classified under the offline category.

Recently, Dassouki et al. [133] presented an approach to combat against flooding attacks in SIP. The proposed research concentrates on flooding attacks launched in cloud infrastructures. The detection scheme consists of two main algorithms. The first one concerns the phase of detecting an attack, while the second one pertains to the mitigation phase. The detection algorithm relies on temporal characteristics of SIP, as well as the fingerprints of SIP messages. The second algorithm makes use of a fingerprint database, following a whitelist approach. This case concerns the SIP INVITE request fingerprints, which preserve a constant size stored for all the introduced scenarios, equal to 100 records. In terms of detection accuracy, the presented results concern a flash crowd, a stealthy flooding attack, and a DDoS one. In the first case, the attack was successfully detected with a 0.24% false alarm rate. In the second case the proposed approach required 1.95 secs, while in the last case, the method introduced a delay of 1.65 secs for low and high rate DDoS. In each one of the previous scenarios, the authors provide a comparison to the Chi-square test-driven approach. Finally, after comparing the mitigation method with other researches in the literature, they reported a 4% CPU overhead for 100 to 1,000,000 users. For the same number of users, the rest of the approaches in the literature report an overhead which varies between 4% to 47%. This work is categorized under the offline class.

Golait et al. [134] studied flooding and coordinating attacks. For detecting these types of attacks, and more specifically to identify anomalies, they modeled the SIP operations related to transactions and dialogs, using a Discrete Event System (DES). More specifically, they designed a new state machine namely Probabilistic Counting Deterministic Timed Automata (PCDTA). In this respect, they identified attacks using a mapping from DoS attacks to anomalies in the DES model. Furthermore, they train the PCDTA to identify probabilities of transitions and timings between SIP operations. Based on their results, the PCDTA hits a 100% detection accuracy in flooding attacks. Finally, they compare their solution to the one provided in [127]. The latter reaches a maximum detection accuracy equal to 98.14% in flooding attacks, and a 6.3% false alarm rate. Finally, as they state, for their experiments they used simulated SIP traffic. This work is categorized under the online class.

Two years earlier Kurt et al. [131] elaborated on the detection of DDoS attacks in SIP. The authors proposed a scheme based on the change point model to thwart DDoS incidents. The authors extensively evaluated their scheme under 4 different datasets. Each

one of the datasets they used contained 40 DDoS attacks. Additionally, they exploited the Poison and Multinomial observation models using 242 observation models. Finally, they implemented a detection scheme prototype based on the Hellinger distance metric. In addition, they offered a comparison between the aforementioned scheme to those presented in [31] and [128]. This work is categorized under the online class.

Lastly, two years ago Wu et al. [135], presented two schemes to cope with SIP-based flooding attacks in VoLTE. The proposed schemes build over the Counting Bloom Filter (CBF) data stucture. For the second one namely PFilter, the authors state that it is easy to expose a bunch of malicious messages inside a crowd of legitimate messages. This statement is defended based on the fact that using a CBF one is in a position to easily extract simple statistics over repeated occurences for an element, assuming a constant time period. The second scheme introduced in this work comprises a robust version of the first one with regards to multi-attribute flooding attacks. In this direction, the main drawback of the first solution pertains to the memory consumption (overflow). In order to overcome this problem, the authors introduced a second scheme which is based on a lightweight CBF-driven data structure to cope with the overflow limitation. Furthermore, the detection accuracy for PFilter reaches 76.4% when the proposed solution battles against a flooding attack with rate equal to 10 cps. Additionally, it reaches a maximum of 100% detection accuracy for a flooding attack of 100 cps. Finally, the authors perform a side-by-side comparison with the research presented in [126]. As they state, the proposed solution presented an increased detection accuracy for all the experiments the authors conducted, except the last one which was not applied in the proposed solution (i.e., flooding rate equal to 500).

The latest work in this subsection was given by Semerci et al. [136]. The authors elaborated on the detection of DDoS attacks by introducing a scheme which relies on change point model. As the authors state, the proposed scheme detects alterations in the Mahalanobis distance metrics of the feature vectors. The scheme consists of a change point detector and a identification module for the malicious entities. In every scenario, the factor which seems to affect the accuracy pertains to the fluctuations of the traffic. In general, increased values in the fluctuations of the traffic correspond to lower F-score values. In any case, the lowest reported value equals to 0.70. This work is categorized under the online category.

4.3.2 Machine learning techniques

Until now, a handful of works discussed the suitability of methods borrowed from the ML community to detect DoS attacks in VoIP realms and particularly in SIP.

Almost seven years ago, Ferdous researched the area of anomalous messages in SIP networks [137]. The authors offered a two-tier system to cope with malformed messages. The proposed architecture comprises of a lexical analyzer and a ML-driven classifier based on SVM. The latter has been employed with the aim to expose semantically anomalous messages which have gone undetected from the first layer. Based on the results they obtained, the authors reported a 0.45 msec overhead per SIP message regarding the classification process. Regarding the performance of the proposed architecture, the authors reported a 99.9% accuracy when they employed their scheme on real traces. This work is categorized under the offline class.

The same year Mehta et al. [138] explored the applicability of Euclidean distance classifiers to cope against malformed SIP messages. The vectors used to feed the classifiers were obtained using substrings from each SIP message. That is, using a sliding window and the n-gram technique, they extracted a set of 100,000 features. Using the high frequency n-grams, and the Kolmogorov Smirnov (K-S) test, they successfully reduced the feature set to 2,000. Based on the authors, this was the first work which elaborated on the applicability of multiple-classifier systems (MCS) in SIP. Using the logistic Regression approach, the authors selected the most prominent classifiers in terms of performance, and they correlated their results using weighed the linear combination method. More specifically, they used a combination function to correlate the pieces of information between the different classifiers, with the aim to reach on a consensus between them. As the authors state, their approach reaches 97.56% accuracy. This proposal is categorized under the offline group.

Ferdous et al. [139] presented a two-tier system for defending against malformed SIP messages. The proposed system comprises of two logical entities, and classifies the messages in two classes, as "good" or "bad". The first level of the system performs lexical analysis on SIP messages, while the second one detects structural and/or content anomalies. The level of deviation is calculated by taking as a reference to past messages which were classified as "good". Moreover, the authors presented a bunch of configuration steps in order to calibrate the environment for the use of SVM using 26 features extracted from

800 pre-classified SIP messages. On the other hand, for the calibration they used SIP messages captured on a real VoIP infrastructure. Finally, the proposed system induces an overhead of 0.45 msecs/msg. This work is categorized under the offline category.

Two years later, Akbar et al. [140] proposed a set of spatial and temporal packet-related features to protect SIP infrastructures against application layer DoS, DDoS attacks, and SPIT. The framework assembles vectors of certain features contained in SIP messages, which are provided as input to Naive Bayes and J48 classifiers. The authors highlighted the advantage of the proposed framework to perform analysis over a bunch of packets, compared to other existing works in the field which need to analyze streams. More specifically, the spatial features used in this work pertain to the IP address contained in the <From> header and the call ratio of SIP INVITE requests. As in their previous work [141], the authors synthetically injected attack messages to the normal SIP traffic with the aim to estimate the detection accuracy of their proposal. Based on their results the proposed mechanism induces negligible false alarm rates. More specifically, they report 0% to 0.7% and 0% to 25% for the FP and FN metrics, respectively. This work is categorized under the online category.

The same year, Mohamadi et al. [142] proposed an anomaly-driven method building over Artificial Immune Systems (AIS). The authors aimed to counteract various types of known and unknown attacks in SIP. Aiming to evaluate the proposed algorithms, the authors exploited existing, well-known datasets, namely as follows: KDD99, Darpa98, NSLKDD, INRIASip. Furthermore, they introduced IUSTSip and DataSetMe datasets. All of the aforementioned datasets contain a variety of known attacks, like, for example, DoS, flooding, and SPIT. Additionally, they launched a DDoS attack namely Torshammer, with the aim to evaluate their method on learning new attacks. That is, based on the provided results, the proposed method successfully detected the new attack. This work is categorized under the offline class.

The next year, Tsiatsikas et al. [143] provided an offline assessment of 5 different ML classifiers in the battle against DoS and DDoS assaults in SIP. This has been done by examining the recorded SIP audit trails in a forensic-driven manner. Based on the results they obtained, the authors stated that ML-powered methods outperform legacy statistical schemes [129], [36]. Additionally, they highlighted the fact that some classifiers achieved satisfactory results even in the case of low-rate flooding attacks. For instance,

according to the results presented, the Neural Networks classifier succeeded a FP rate of 5.2% and zero FN. This work is classified under the offline category. One year later, the same authors [144] researched the applicability of the same classifiers in a realtime fashion. In this direction, they presented clues of efficiently thwarting DDoS assaults and performance applicability details concerning the overhead of ML classifiers on a realtime system. More specifically, based on their results, Naive Bayes and Random Forest were the top performers in terms of false alarms. Additionally, the classification overhead for all the classifiers remained lower than 4 msec. This work is categorized under the offline class.

4.3.3 Other approaches

Another research focusing on SIP-based DoS attacks was published almost seven years ago by Voznak et al. [145]. This study was the first one presenting a realtime hybrid system using Snort, SnortSam, and IPTables. The proposed solution relied on the statistics over signatures. That is, after reaching a threshold of matchings, with respect to the aforementioned signatures, the SnortSam plugin informs the IPTables component to apply the corresponding rules. Based on their results, the system introduces delays in the response phase. This delay is irrelevant to the signature-driven detection scheme and is considered as a second drawback of the solution. Using sipp and Asterisk, they created a set of scenarios, considering attacks generated using the following SIP requests: REGISTER, INVITE, ACK, BYE, CANCEL, OPTIONS. Moreover, they launched UDP and TCP SYN flooding attacks. Based on their results, REGISTER and OPTIONS were the most devastating ones. Finally, considering the lack of protection directly in the SIP proxy, they proposed an alternative network topology, using a Demilitarized zone (DMZ). This approach protects the SIP proxy from internal and external attacks. This work belongs to the online class.

Maccherani et al. [146] extended the NetServ node architecture, using OpenFlow. Using this approach, the authors presented a flow-based approach for intrusion detection (FIDS) which reaches an increased performance in high speed networks. The latter is evaluated in contrasto to DPI systems. The proposed system relies on 2 distinctive steps. The first one pertains to the flow exporting, and the second one to the flow collection. Based on the authors, the main difference of the proposed system in contrast to the prior art,

relates to the distributed functionality. That is, the proposed system can be easily applied in a distributed manner, using the NetServ service, by deploying the solution in every node. Additionally, in order to exemplify their proposal, the authors picked the GENI platform. As an example, they picked SIP and more specifically DoS attacks. Based on their results, the reaction time on 2 attacks does not exceeded 2.4 sec. Finally, based on their results, the use of 2 PUs doubles the handled traffic. This work is categorized under the online class.

Rodnikorn et al. [147] presented a new protection mechanism against signaling attacks, forgery and message manipulation, namely SIPE-SAP. The proposed mechanism emphasizes on confidentiality and message integrity. Additionally, it constists of 3 different modules, namely as follows: a. KeyAg module, b. EDCrypt module, and c. MD. The first sub-module is devoted to the key exchange process, the second one pertains to the confidentiality, and the latter one is used for the digest generation. In order to assess the proposed mechanism, the authors created a testbed using OpenSIPS, and the SIPp tool. After applying SIPE-SAP, a registration response time similar to the original registration process is reported. In addition, concerning the overhead, for the call setup the authors mention an improvement smaller than the one presented with TLS. In the latter case, the average overhead with SIPE-SAP equals 1.7 msecs, while for TLS it is almost 2 msecs higher. This proposal is categorized under the online class.

Not later than 2012, Rontti et al. [148] performed a demonstration on how it would be possible to expose vulnerabilities in NGN networks by employing specification-based fuzzing. Using the proposed method, the authors aimed to prevent DoS and DDoS attacks against IMS. More specifically, using the SIP ABNF [149] the researchers developed a fuzzer with specification-based rules. As the authors state, the purpose of this approach is to test the internal software modules and dive into the functionality as much as possible. After this phase, an attacker can send the packet with the malformed content. The latter assures that the scarlet packet will affect the corresponding state machine in the underpinnings. After that, they tested 4 critical core NGN interfaces. The P-CSCF and the I-CSCF in the core network, and the S-GW and the P-GW in the home network. Based on the results obtained from the proposed method, they managed to expose vulnerabilities. More specifically, the DoS time for S-GW and P-GW was 11.5 secs, while for P-CSCF the maximum DoS time was 33.7 secs, while for I-CSCF 0.5 secs. This research is categorized under the other class.

The same year, Stanek et al. [150] contributed to the detection of DDoS attacks by introducing a new architecture entitled SIP protector. The main components of the architecture pertain to a redirect server, a rate-limiting firewall, a NAT element and a mapping function. Using these elements, the authors constructed the proposed architecture which as they state is not appropriate for a real-life deployment. This is due to the results obtained from the stress testing, which indicated a processing capacity of approximately 20000 rps. Additionally, in their experiments, the authors simulated both General and Tailored DDoS attacks, using the SIPp-DD [151] tool. To complete the testbed, they employed the widely-known SIP server Asterisk. When the authors launched a typical DDoS attack, this was successfully thwarted on the second level of the proposed architecture (i.e., the rate-limiting firewall). For the second type of attack, the proposed architecture mitigated 75% of the attack traffic. This work is classified under the online class.

The same year, Farley et al. [152] proposed a lightweight mechanism devoted to the mitigation of MITM, counterfeit and replay attacks, namely VoIP Shield. The proposed mechanism relies on pre-shared keys and the pairing bonds between UACs' and SIP servers'. As the authors state, the architecture allows one pairing connection from the side of each client, and multiple pairings from the side of server. Using simple hash functions, the proposed scheme employs a Message Authentication Code (MAC) in every SIP message. Using this approach, every message can be authenticated in the recipient shield for the integrity and the authenticity of the sender. In order to test the proposed mechanism, the authors utilized 23 different attacks. Based on their results, 22 of these attacks are successfully detected, while one of them is prevented. This work is categorized under the online category.

Lahmadi et al. [153] designed and developed a framework namely SecSIP, for preventing exploitations from well-known vulnerabilities in SIP. As the authors state, their method relies on captured traffic which is analyzed in order to create prevention-specifications in the form of rules. The latter are coded in a language called VeTo. In this respect, event graphs are employed with the aim to correlate protocol activities with prevention specifications. Based on their statements, genetic algorithms are employed with the aim to automatically extract prevention specifications. As a paradigm, they make use of flooding and DoS attacks. Using OpenSIPS and SIPp, they created the legitimate traffic which was captured using the SecSIP framework on a realtime fashion. As they state,

SecSIP introduces an overhead which ranges between 1 and 10 msecs, with a call rate lower than 800 packets/sec. Finally, based on their results, the maximum time overhead reaches 900 msecs on a heavy load of 900 concurrent calls. This work is categorized under the online class.

Asgharian et al. [154] presented a combined approach to detect DoS attacks in SIP. In this direction, the authors provided a publicly available dataset. Regarding the detection scheme, their proposal consisted of anomaly-driven approaches and specification-based detection [155]. As they state, the intrusion detection engine relies upon a finite state machine. In this respect, the main idea of the proposed mechanism relies on the concept of complete SIP transactions. In order to monitor the transactions, the authors selected features which describe uniquely SIP transactions and dialogs. That is, they touched the <From> and <To> headers tags, the branch parameter in the <Via> header, and the <CSeq> correspondingly. Using a labeled dataset, they reported an average of 85% and 7.7% for detection and false alarm rates correspondingly. On the other hand, using the dataset provided in [156], they reported average detection and false alarm rates equal to 85% and 15% respectively. Finally, using the ROC curves [157], they calibrateed the proposed scheme and extracted the corresponding thresholds. This work is categorized under the online class.

Vrakas et al. [158] were the first who managed to introduce a realtime system capable of detecting and preventing a wide set of spoofing and DDoS attacks. In their analysis, they considered a set of spoofing attacks, like, for example SIP signaling, identity theft, masquareading, and MITM. The architecture of the proposed system is structured based on the policies that must be applied, the spoofing, and the flooding attacks. The basic functionality of the system relies on the cross correlation of 6 different values which parasitize in different layers of the network stack. Using these values, they created vectors which are then forwarded in each different module. The initial decision for forwarding a message is always obtained by using the policy enforcer module. Furthermore, in order to achieve memory efficiency, they employed a bloom filter for performing the search operation of the vector. Finally, based on the obtained results, the false alarm rate is almost zeroed for the spoofing module. For the flooding modules, the authors mentioned a large number of false positives in scenarios which exceed the training period. Additionally, they mentioned a constant number of false alarms in simple flooding attacks

with minimum alterations in the traffic patterns. This work is categorized under the online class.

Raza et al. [159] researched the concept of restrictive modeling, with the aim to detect and prevent DoS attacks. The proposed model operates on the transport layer, using UDP. The authors introduced the "WAIT" and the "GO" messages. The first one restricts the caller to flood the SIP proxy with redundant messages. This happens by sending a "WAIT" message to the caller. The "GO" message, is exploited with the aim to inform the caller about the restriction revocation. Based on the results they obtained from a 10 call attack, a packet drop reduction to 3.2 secs is reported. Additionally, the authors highlighted the 1 second improvement with respect to service availability. This work is categorized under the other category.

The same year, Hussain et al. [160] presented a lightweight mechanism to detect SIP flooding attacks. Based on the authors, the simplicity of the proposed scheme relies on the use of one transition table in the SIP proxy, and one new custom header field in each SIP REGISTER request. The transition table is used with the aim to match requests, with source and destination IPs', along with the arrival time. Moreover, the addition of a new header entitled as "Critical number", serves the purpose of limiting the requests that can be handled in the UA side. Using this approach the SIP proxy can limit the number of SIP INVITE requests, based on this value. Moreover, based on the results they obtained on a realtime assessment, the proposed mechanism presents improved response times spanning between 0.5 to 1.0 μsecs. Finally, even for stressing scenarios the proposed scheme outperforms the original core functionality of openSER. This work is categorized under the online class.

Seo et al. [14] conducted a study on resource consumption attacks targeting SIP proxies. More specifically, the scope of this research is twofold, researching both malformed messages and flooding attacks. In order to enhance the security of IMS, with respect to the previously-mentioned attacks, the authors introduced a stateful mechanism called SIPAD. Based on the authors, the main advantage of their proposal is concentrated on the performance, which relies on the applicability of an optimized data structure. Based on the results they obtained, SIPAD was the top performer in the category of malformed message attacks, with a detection accuracy reaching 100%. This research is classified under the online class.

Roh et al. [161] researched the area of DoS attacks in SIP. The authors introduced a whitelisting approach to defeat those types of assaults. The proposed scheme consisted of three fundamental modules. The first one, namely the whitelist manager, is related to the process of whitelisting. The second one pertains to a filter, and the last one to the attack detector. Based on the proposed architecture, these modules are placed in front of the SIP server, and thus they do not create additional costs in terms of overhead in the SIP server side. As the authors state, the main advantage of the proposed scheme compared to the rest of the related researches, relies on the bloom filter data structure which outperforms the other solutions in terms of memory consumption. In essence, the attack detector performs the first layer processing of the messages in order to identify anomalies. After that, the traffic is forwarded to the filtering module which takes the final decision. That is, in case an attack has been detected, the packets will be forwarded to the SIP server only in case the whitelist manager has instructed to do so. The authors reported false positive alarms below 0.1%, and false negatives which reach 0.2%. These results have been obtained when the system receives attack traffic of 600 messages/sec. Finally, the authors perform a side by side comparison to the whitelisting-based work presented in [162]. Based on the results they obtained from this comparison, the proposed method is the only one which exhibits a tolerance with respect to the throughput, which is not lower that 1. This work is categorized under the online class.

The same year Zhang et al. [163] proposed a bloom filter-driven approach, to cope against resource consumption attacks in SIP proxies. More specifically, the authors provided insights on the reasons affecting the time latency in DNS queries. After exemplifying how an attacker could launch a DoS attack by utilizing hard-to-resolve domain names, they proposed a solution related to blacklisting. More specifically, instead of using the traditional blacklisting approach which is limited by the *O(N)* search operation, they employed a counter bloom filter. Based on their results, the proposed mechanism does not give any false negative alarms, while it presents a few false positives which are caused due to size issues pertaining to the data structure. Aiming to surpass this problem, they provided information for best practices using bigger structures. This research is classified under the online category.

One year later Sebastian et al. [164] proposed a mechanism to prevent registration flooding attacks in SIP infrastructures. The proposed mechanism consists of 3 basic layers. The first one relies on simple calculations for blocking packets which exceed specific

thresholds. The second one banks on the concept of "reverse routing technique" [165]. The last one counts on a anti-spoofing mechanism which monitors the source IP address of the network layer, and the <Contact> header. Aiming to evaluate their mechanism on a realtime fashion, they developed a software module using the C language. More specifically, they employed OpenSIPS along with SIPp for providing SIP service and creating the attack traffic accordingly. Based on their results, the introduced overhead does not exceed 2 secs in terms of CPU processing. The latter comes true when the authentication is not enabled on the server. On the contrary, after enabling the authentication process, a maximum of approximately 7 secs overhead is reported. This research is categorized under the online group.

The same year, J. Tang et al. [40] conducted a study on stealthy attacks and malformed messages. With respect to the first category of attacks, the authors provided a detection mechanism building over the sketch data structure. In this case, the authors picked every SIP address as the key of a record in the data structure, and the sum of the INVITE messages originated from this address as the value. In the next phase of the detection, the proposed solution performs wavelet analysis over the output perceived from the first phase. Moreover, the authors employed the EWMA [166] method for handling the normal traffic, using dynamic thresholds. Based on the provided results, the authors stated that their proposal is directly comparable to the HD solution, considering a normal flooding scenario. On the contrary, their proposal outperformed HD in the case of stealthy flooding attacks. Concerning the second category of attacks, the researchers focused on attacks which rely on the exploitation of the "Session-expires" header, and the open nature of the wireless protocols in general. This concerns the attacks which aim on draining the target resources by simply allocating resources on the SIP proxy. In this direction, the SIP proxy reserves resources for the time period defined in the "Session-expired" header, which is not upper bounded by RFC 4028 [167]. Taking into account this type of attack, the authors presented a detection mechanism building over the Anderson–Darling test. In the examined problem the different Session-expired values composed the values of the distribution. Based on the proposed method, using a 400s session timer the attack was successfully detected upon receiving 436 samples. This work is categorized under the offline category.

Liu et al. [168] explored DoS attacks in SIP borrowing models from queuing theory. More specifically, the authors modeled the victim of this type of assault as a M/M/1/K

system [169]. Additionally, they proposed a new priority queue in order to enhance the existing SFW architecture. They achieved this by using a new architecture consisting of 2 different priority queues. The first one, the low-priority queue, deals with the SIP INVITE requests. The second one, the high-priority one, with the rest. In any case, as they state, this can be configured for addressing the needs of every different infrastructure. Furthermore, they employed a bandwidth policy with the aim to trigger the SDN controller and reform the network. Finally, based on their results, using an arrival rate equal to = 450, they obtained a better response time from the priority queue which is almost 50% better than the single FIFO case. This proposal is categorized under the other class.

Marchal et al. [41] employed MCS in order to achieve a consensus between different classifiers, in the battle against DoS attacks in SIP. More specifically, they investigated a new type of DoS attack in SIP, called mimicry TDoS. In this type of assault, the attacker sends a volume of malformed messages which contain minor changes in contrast to the normal ones. As the authors stated, using their approach, they totally bypassed the parser module needed in the detection side. In this respect, they exploited MCS aiming to classify the message. Moreover, they employed Kamailio and SIPp tool, with the aim to generate SIP REGISTER malformed requests. Using the Rieck's et al. [170] method, they employed a 4-gram approach to extract the necessary features. After testing two combination functions, namely "the voting rule" and LR, they achieved the combination of different classifiers. The proposed MCS reached a detection accuracy, with a probability spanning between 98.5% and 99.12%. Additionally, this approach induces a constant algorithmic complexity equal to *O(1)*. Finally, they reported that the major drawback of MCS in detecting malformed SIP messages, pertains to the classification of inconsistencies in the <Content-Length> header. This happens because there is a difficulty in comparing this SIP header to the actual payload. This work is classified under the online category.

Venilla et al. [171] presented a hybrid approach to detect flooding attacks and bogus messages in SIP. The authors employed a two-tier architecture, consisting of a less than 10-feature SVM classifier in the first layer, and an Entropy-driven approach in the second one. More specifically, the first layer (preprocessor) SVM, is utilized as a first step classification to differentiate irrelevant VoIP traffic. The second level is based on Entropy which takes as an input the packet count. Additionally, the Entropy is used in conjuction

with EWMA [166], with the aim to handle dynamic thresholds. In order to assess the proposed solution, the authors created a testbed using the Asterisk PBX, SIPp, a bunch of VoIP applications, and Nsauditor for creating the attack traffic. Based on the provided results, the authors reported an accuracy which exceeds 92%. Finally, they provided results of performance after using a different kernel SVM function. This work is categorized under the offline group.

Bansal et al. [172] researched the area of flooding attacks in SIP. The authors offered a mitigation mechanism which correlates the number of SIP messages. That is, the authors make the assumption that under normal conditions a legitimate user will send 1 to 2 signalling messages in order to initiate a call. Additionally, they stated that each new call from the same user should follow a SIP BYE request. In this direction, if a user wants to initiate a new call she should first disconnect the previous call. Under this pillar, the authors state that an attacker could be identified by measuring and correlating the number of SIP INVITE and BYE requests. Based on the results they obtained under a DoS attack, the authors the proposed scheme improved the system performance by 158%, in terms of CPU consumption. This work is categorized under the online class.

Jyothi et al. [173] conducted a study on the overhead introduced in the pre-processing phase of Deep Packet Inspection (DPI). Even the latter consists one of the fundamental operations in every NIDS, the authors stated that it comes with the L7 extraction overhead. This operation is considered to be quite expensive because the usual form of field extractors is driven by software solutions. In this respect, the authors contributed with a deep packet field extraction engine, namely DPFEE, capable of extracting L7 fields into hardware. Based on their results, DPFEE presented an improved performance compared to PJSIP parser and Snort. The total improvement ranging between 22X-80X. Finally, as the authors state, using their proposal, one can achieve a 30% reduction in the system load. This work is categorized under the online category.

Akbar et al. [174] introduced a scheme building over HD, with the aim to detect low-rate and multi-attribute DDoS attacks. Additionally, they considered the SIP BYE tear down attacks. For HD, they used the normalized frequencies of INVITE, BYE, 200 OK, and ACK. Using a training period of 120 secs, they obtained the necessary metrics for the normal values of the distribution. On the other hand, they performed the classification using a 10 secs testing period. As they state, they used Kamailio as

a SIP proxy and load balancer. In order to achieve that, they amended "rate control", "pike", and "dispatcher" modules correspondingly. In order to create the attack traffic, they used SIPp. Finally, they compared their proposal to the Adaptive threshold and CUSUM. Based on the results they obtained, the proposed method presented a higher accuracy in all the scenarios, achieving a 98.3% detection accuracy in the worst case in the low-rate attack. In any case, the proposed method presented a better performance in all scenarios. This research is categorized under the online class.

Shah et al. [175] presented an approach to enhance SIP security with respect to malformed and spoofed messages. The proposed approach comprised of 2 conceptually separated entities. The first one is devoted to Header validation using a rule-driven approach. The latter goal is achieved using Snort. The second one deals with the IP validation through whitelist. In this context, they used Asterisk PBX as a SIP proxy, and 2 softphones to generate the normal traffic. Additionally, they exploited VoIPER as an attack generator. Based on the results they obtained, their approach touched a detection accuracy equal to 87.27% . The latter was perceived when the attack tool generated a traffic of 100 packets. At the same time, 96 packets were detected as an alert in the proposed scheme. This research is categorized under the offline category.

Sabra et al. [176] exploited the concept of clusterheads with the aim to protect users' privacy. The authors considered a model with 2 different clusterheads which communicate directly. The latter preserves the users' privacy and unlinkability by broadcasting the packets to the rest of the nodes, belonging in the same area. One of the criteria they picked in order to select a node as a clusterhead pertained to the remaining battery of a node. This was induced from the fact that the forwarding functionality of a clusterhead drains its resources. The authors examined the concept of energy consumption by using a laptop and a smartphone. For the first one they reported an energy consumption of approximately 0.031% per session. For the other, the energy consumption reached 1.6% per session. The authors employed the OPNET software in order to grab a better understanding of the model performance. In this direction, they touched the MOS metric [177] with the aim to assess the QoS at the network. In this direction, the authors assessed different codecs regarding jitter, MOS, and end-to-end delay. Based on the results they obtained, the G.729b codec scored the worst results. This work is categorized under the other class.

Two years earlier, Cadet et al. [178] detailed on a realtime hybrid scheme for protecting SIP infrastructures from DoS and DDoS attacks. The authors employed Snort [179] in conjuction with iptables, with the aim to counteract flooding attacks. In the first layer the SIP messages are classified based on the rules of Snort. In the next layer (i.e., iptables filtering), a final decision is granted with respect to the luck of a SIP packet. That is, the packet is forwarded to the SIP server or it is dropped. The authors employed the NFQUEUE kernel module aiming to redirect the traffic from iptables to Snort and back again. Furthermore, they employed the rules presented in [172] along with a static threshold equal to 4. The latter was selected using the results obtained in [172]. Based on these results it was found that a legitimate user sends 1 to 2 SIP INVITE requests in every call initiation. The proposed scheme was evaluated using a testbed consisted of the Asterisk PBX and the SIPp tool. Based on the obtained results, the applicability of Snort reduced the CPU consumption by 20%. Finally, the authors reported a betterment to the bandwidth utilization, after applying SNORT. This work is categorized under the online class.

Sun et al. [180] introduced SFADS, a two-layer, novel flooding attack detection scheme. The first one builds over the CUSUM algorithm and more particularly relies on 2 features, one internal and one external. The features are extracted in the initial phase of session establishment. Additionally, the output of the first layer is fed in the second layer, which relies on fuzzy logic. The latter is used with 25 rules, with the aim to assist on taking decisions prior reaching the predefined thresholds. Furthermore, the authors employed OpenIMScore and sipp in order to test the proposed scheme. As the authors state, the main advantages of the proposed solution compared to the CUSUM algorithm, are highlighted as follows: a. SFADS detects more accurate low-rate flooding attacks. b. The detection performance in terms of time detection are far lower for SFADS, considering the scenarios with a number of SIP INVITE requests which does not exceed 100. Additionally, the false alarms are zeroed for all the comparable scenarios, except the second one, which reaches 0.2%. Finally, for the worst detection time of CUSUM solution which equals 7.3 msec, the proposed solution introduced a delay equal to 4.8 msec. This research is categorized under the online category.

The same year, Hosseinpour et al. [181] presented an anomaly-driven detection method, to detect DoS attacks in SIP. After employing an FSM to extract specific characteristics from normal traffic, the authors make use of fuzzy logic to detect attacks based on the

extracted characteristics. More specifically, they rely on the average of time between differences in the FSM states and the number of messages obtained in the various states. For the testbed they used Spirent Abacus 5000, along with Asterisk, and SIPp. In the flooding scenario, they launched a bunch of SIP OPTION requests in a variety of rates. Prior reaching the SIP proxy, the proposed scheme detected 99.9% of the attack traffic. Additionally, they provided a comparison to the work presented in [14]. As they report, the false alarm rates are quite lower than in SIPAD. Finally, their approach is irrelevant to the type of processed SIP messages. This work is categorized under the offline group.

The same year, Dassouki et al. [182] researched the area of DoS attacks in SIP. The authors introduced a new specification-driven mechanism, exploiting an FSM for each session initiated using the SIP INVITE request. The main pillar of the proposed mechanism relies on the delayes of ACK messages with respect to the callee entity. In this direction, the proposed scheme makes use of 2 counters which correspond to the different levels of analysis, the session and the aggregation level. That is, the first counter is increased in the session level when an ACK delay is encountered. In case this delay persists for a specific time range, then the second, aggregation level counter will increase. When the latter exceeds a predefined threshold, then a SIP INVITE flooding attack is detected. Finally, the authors provided a comparison to the works presented in [183] and [184]. Based on the obtained results, the proposed mechanism detected a slow flooding attack in 2.6 secs. On the other hand, the solution presented in [184] detected the same attack in 9.2 secs. Additionally, for the same attack pattern, the method presented in [184] introduced a delay of approximately 10 secs. This work is marked under the online category.

Hosseinpour et al. [42] presented a new method for thwarting out of sequence messages, malformed messages, and flooding attacks. The authors considered the detection and prevention of DoS attacks in SIP, by modeling the states of normal traffic using an FSM. This was achieved by using a parser which correlates session information among different SIP messages. In order to detect the attack patterns, the authors defined 3 different attack states, namely “Normal”, “Alarm”, and “Attack”. Moreover, they compared their results with those obtained from SIPAD [14]. The results indicated that especially for low-rate attacks, the proposed method performed better in terms of detection probability. More specifically, their method presented a detection probability varying between

38.27% and 98.41% for all the scenarios. This happened because SIPAD employs specific thresholds which make very difficult to detect low-rate attacks. Additionally, with respect to malformed messages, the proposed method reached a 100% accuracy, in contrast to SIPAD which reached a 33.13% one. This work is categorized under the offline category.

Recently, Tsiatsikas et al.[43], researched the area of SDP-driven malformed attacks and their impact in the server and client side. The authors touched 9 different softphones and 1 hardware phone in order to highlight the impact of malformed message attacks to real-life SIP equipment. Additionally, they researched the tolerance of SIP proxies to the handling of big SDP segments. In this direction they successfully managed to convey a SDP segment equal to 12,116 bytes. This finding indicated that SIP proxies may be suceptible to volumetric DDoS attacks. In order to combat the aforementioned types of attacks, the authors provided a software module capable of preventing SDP malformed message attacks and SDP-driven covert channels. Based on the presented results, the proposed solution introduced negligible overhead in terms of processing time. Additionally, the authors provided a side-by-side comparison to the widely-known open source SIP proxy Kamailio in terms of processing overhead. As the authors stated, the parser detection accuracy reached 100%. This is explained due to the fact that the parser relies on misuse detection, and thus it can detect a set of 100 implemented rules, based on RFC 4566 [1]. This work is categorized under the online class.

4.4 C&C channels over SIP

This section succinctly reports on works that have been presented in the literature so far regarding C&C channels over SIP.

Zhao et al. [44] exploited 3 randomly generated strings included in SIP messages in order to create a covert channel. In this respect, the aim of this work is to detect hidden information in SIP headers using chaos theory. Following this approach and using only a small portion of normal messages, the authors analyzed and reconstruct the random numbers included in the <Call-ID> header and the various Tags of a SIP message. This is undertaken by means of a time series analysis. Additionally, the authors highlighted the advantage of using the proposed scheme on a realtime fashion, based on

the computational complexity and the detection accuracy. According to the obtained results, the complexity reaches *O(k)*, and the detection accuracy does not exceed 91.9% on average. This work is categorized under the offline category.

Two years later, Mehic et al. [185] researched the area of VoIP steganography with respect to the capacity of covert channels. The authors calculated the number and the type of SIP messages which can be used to convey hidden traffic during established calls. In this direction they exploited AD Snort, with the aim to capture network traffic. After that they employed a profile generator in order to perform network traffic predictions. For the profile generator the authors picked the Naive and Moving average methods. Using the predicted results, they aimed on comparing the current network traffic with the predicted one. Based on the metrics obtained from their testbed and the calculated bandwidth, a value of 4 cps, i.e., 1856kbps, was perceived without being exposed by Snort. This work is categorized under the other category.

One year later, Tsiatsikas et al. [186] evaluated the capacity of covert channels over SDP. The authors exploited one mandatory and one optional descriptor with the aim to convey secret information. As they stated, they picked *<a=ptime:>* and *<o=>* with the aim to eliminate the chances of triggering communication problems in the session. The first descriptor has been used as a means to convey the attack type, while the second one has been used for communicating the IP address of the victim. For their experiments, they used 6 different infected machines. Using the proposed C&C channel, they successfully managed to simulate a botnet consisting of infected phones. More specifically, they managed to simulate the attack capacity of their army by using 2 types of attacks. The PING and the SYN flooding one. Based on their results, they obtained a 6 MB/sec volume of traffic on the victim side when the number of threads in every bot reached a number equal to 30. This work is categorized under the other class.

Wendzel et al. [187] provided a unified description method for the research area of network steganography. The authors provided a literature review with regards to the researches touching data hiding methods. After that, they collected a set of 131 hiding methods and they provided a method for helping the classification and evaluation of future works in the field. That is, they created a unified method to describe the different steganographic approaches presented in the corresponding research area. The authors recognized 3 main areas of hiding methods. These are a. “general information about the hiding method”,

b. "description of the hiding process", and c. "potential or tested countermeasures". The first two consist of mandatory and optional sub-categories. Furthermore, aiming to help researchers, they proposed to combine their method, with the one given in [188]. Using this combination, they state that it is easy to identify the novelty of a new work with respect to data hiding methods. Moreover, as they state this correlation can be applied in the peer review process. This contribution is categorized under the other group.

The same year, Shrestha et al. [189] presented a ML-driven framework to detect covert channels. This proposal exploited SVM using fingerprints obtained from network communications. The proposed framework consisted of 3 different units. These are the traffic filter, the fingerprint extractor and, the SVM framework. In essence, these components capture the traffic, extract the necessary features, and finally feed the features to the SVM framework. As the authors stated, the obtained results have been perceived by training the classifier using 4 diverse covert timing channel (CTC) algorithms. For the feature extraction process, the authors utilized the following types of statistical fingerprints: the Kolmorov-Smirnov (K-S) score, the Regularity score, the Entropy, and the Corrected Conditional Entropy (CCE). Finally, they reported that the classification accuracy of covert and overt traffic in the case of a On-Off channel reached 100% and 97% respectively. The latter was perceived on a system examining block sizes of 2,000 samples. In the case of a L-Bits-to-N-Packets, the authors reported a 100% and a 96% accuracy for covert and overt communication. This work is categorized under the offline class.

Finally, the same year, Mazurczyk [45], presented an IP-driven steganographic method namely LACK. Building over useless Real time Transport Protocol (RTP) packets that may exist in a communication network, they builded a hidden channel of communication. Additionally, they evaluated their method in terms of steganographic bandwidth, using different voice codecs. Based on their results, they stated that high bit rate codecs are prefereable. In order to exemplify the previously-mentioned statement they exploited the G.711 codec. This was further verified by their experimental results, which provided a robustness on packet losses over 5%, and a MOS score [177] greater than 3. Finally, they stressed the robustness of their method, in terms of steganalysis attacks, which in turn was justified from the "naturallity" of the packet loss process. This work is categorized under the other category.

4.5 Discussion

As already discussed in the beginning of this section, the contributions of this PhD thesis in contrast to the existing works in the literature are depicted in figure 4.1. The red circles correspond to works which have been published in the context of in this PhD thesis. On the other hand, the green circles refer to works which have used as a reference, the contributions of the PhD thesis at hand.

Tables 4.2 and 4.3 present the analysis of the literature contributions. The first table is devoted to the analysis of the researches which exploit log files to detect security incidents. The second table details on the works which refer to DoS attacks and C&C channels.

TABLE 4.2: Analysis of the literature: Log files

No.	Reference	Year	Contribution area	Processing
1	Schneier et al.[62]	1998	integr./confid.	other
2	Biskup et al. [87]	2000	privacy	other
3	Sah [89]	2002	integr./confid.	other
4	Flegel [90]	2002	privacy	online
5	Rouillard [91]	2004	log correlation	online
6	Xu et al. [92]	2004	log correlation	other
7	Lincoln et al. [93]	2004	privacy	online
8	Slagell et al. [94]	2005	anonymity	other
9	Godinez et al. [95]	2005	reduce log size	offline
10	Stathopoulos et al. [96]	2006	integrity	other
11	CP. Lee et al. [97]	2006	trace viewing	other
12	Casey et al. [98]	2006	log analysis (forensics)	other
13	VH. Escobar-Jeria et al. [99]	2007	log analysis	other
14	Saleh et al. [100]	2007	log analysis	other
15	Monteiro et al. [101]	2008	log authentication	other
16	Goel et al. [104]	2008	system state	online
17	Xia et al. [105]	2008	probabilistic forensics	online
18	Pun et al. [106]	2009	log analysis	offline
19	Barradas-Acosta et al. [107]	2009	log analysis	offline
20	Myers et al. [108]	2009	malicious insiders	other
21	Nagappan [109]	2010	log analysis	other
22	Mazza et al. [110]	2011	log analysis	other
23	KH. Lee et al [111]	2013	garbage collection	offline
24	Yen et al. [112]	2013	knowledge extraction	online
25	King [113]	2013	non-repudiation	other
26	Singh et al. [114]	2014	log analysis	offline
27	Gu et al. [115]	2015	log analysis (extract info)	offline
28	Ambre et al. [116]	2015	log analysis (extract info)	offline
29	Juvonen et al. [117]	2015	log analysis (extract info)	online
30	Moh et al. [118]	2016	ML (training)	offline
31	Breier et al. [119]	2017	log correlation	offline

TABLE 4.3: Analysis of the literature: DoS attacks & C&C Channels over SIP

No.	Reference	Year	Protocol	Protects/Proposes	Type	Processing	Comparison
32	Voznak et al. [145]	2012	SIP	DoS	SnortSam, IPTables	online	N/A
33	Maccherani et al. [146]	2012	SIP	DoS	OpenFlow, NetServ	online	N/A
34	J. Tang et al. [126]	2012	SIP	message forgery	statistical (Sketch, HD)	online	N/A
35	Rodnikorn et al. [147]	2012	SIP	message forgery	new header	online	N/A
36	Rontti et al. [148]	2012	SIP	DDoS	specification-based fuzzing	other	N/A
37	Stanek et al. [150]	2012	SIP	DDoS	three-tier	online	N/A
38	Farley et al. [152]	2012	SIP	MITM, counterfeit	cryptographic-driven	online	N/A
39	Lahmadi et al. [153]	2012	SIP	DoS (malformed, flooding)	rule-based	online	✓(Suricata [190])
40	Asgharian et al. [154]	2012	SIP	DoS (flooding)	state machine	online	N/A
41	Chen et. al. [121]	2012	SIP	DoS (flooding)	statistical (CUSUM)	offline	N/A
42	Ferdous et al. [137]	2012	SIP	DoS (Malformed)	rule-based, ML(SVM)	offline	N/A
43	Mehta et al. [138]	2012	SIP	DoS (Malformed)	ML (MCS)	offline	N/A
44	Ferdous et al. [139]	2012	SIP	DoS (Malformed)	ML (SVM)	offline	N/A
45	Zhao et al. [44]	2012	SIP	C&C	chaos theory	offline	N/A
46	Mazurczyk [45]	2012	RTP	C&C	N/A	online	N/A
47	J. Tang et al. [40]	2013	SIP	DoS (malformed)	statistical (Anderson–Darling)	offline	✓(HD)
48	Vrakas et al. [158]	2013	SIP	DDoS - spoofing attacks	statistical (bloom filter)	online	✓
49	Raza et al. [159]	2013	SIP	DDoS	restrictive model	other	N/A
50	Hussain et al. [160]	2013	SIP	DDoS	statistical, new header	online	N/A
51	Seo et al. [14]	2013	SIP	DoS (malformed, flooding)	rule-based (stateful)	online	✓(SecSip [191], [13], [183])
52	**Tsiatsikas et al. [129]**	2013	SIP	DoS (flooding)	statistical (Entropy)	offline	N/A
53	Roh et al. [161]	2013	SIP	DoS (flooding)	statistical (bloom filter)	online	✓(whitelist [162])
54	Zhang et al. [163]	2013	DNS	DoS (DNS)	statistical (bloom filter)	online	N/A
55	Stachtiari et. al. [122]	2013	SIP	DDoS	N/A	other	N/A
56	Sebastian et al. [164]	2014	SIP	DoS (flooding)	three-tier	online	N/A
57	Zargar et al. [125]	2014	SIP	DoS (flooding)	statistical (Entropy)	offline	✓(HD)
58	Akbar et al. [140]	2014	SIP	DDoS (flooding) - SPIT	ML, Entropy	online	✓(HD)
59	J. Tang et al. [32]	2014	SIP	DDoS (flooding)	statistical (HD)	online	N/A
60	Liu et al. [168]	2014	SIP	DoS (flooding)	priority queue	other	N/A
61	Dassouki et al. [123]	2014	SIP	spoofing	fingerprints	online	✓(AIB [124])
62	**Tsiatsikas et al. [128]**	2014	SIP	DoS (flooding)	statistical (HD)	offline	N/A
63	Mohamadi et al. [142]	2014	SIP	new attacks	ML (AIS)	offline	N/A
64	Mehic et al. [185]	2014	SIP/RTP	C&C	N/A	other	N/A
65	Marchal et al. [41]	2015	SIP	TDoS (mimicry)	ML (MCS)	online	N/A
66	Venilla et al. [171]	2015	SIP	DoS (flooding)	ML (SVM), Entropy	offline	✓
67	Bansal et al. [172]	2015	SIP	DoS(flooding)	statistical	online	N/A
68	Jyothi et al. [173]	2015	SIP	DoS	pre-processing	online	N/A
69	Lee et al. [127]	2015	SIP	DoS (malformed) - SPIT	statistical	online	N/A
70	**Tsiatsikas et al. [34]**	2015	SIP	DoS (flooding)	statistical (Entropy)	online	N/A
71	Akbar et al. [174]	2015	SIP	DDoS	statistical (HD)	online	✓(CUSUM)
72	**Tsiatsikas et al. [143]**	2015	SIP	DDoS (flooding)	ML	offline	✓(Entropy [34], HD [128])
73	**Tsiatsikas et al. [186]**	2015	SDP	C&C	N/A	other	N/A
74	Wendzel et al. [187]	2015	N/A	N/A	N/A	other	N/A
75	Shah et al. [175]	2016	SIP	DoS (malformed), spoofed	rule-based	offline	N/A
76	Sabra et al. [176]	2016	SIP	privacy	cryptographic-driven	offline	N/A
77	Dassouki et al. [182]	2016	SIP	DoS (flooding)	specification-based	online	✓([183], [184])
78	Cadet et al. [178]	2016	SIP	DoS (flooding)	Snort	online	N/A
79	Sun et al. [180]	2016	SIP	DoS (flooding)	CUSUM	online	✓(CUSUM)
80	Hosseinpour et al. [181]	2016	SIP	DoS	FSM	offline	N/A
81	Semerci et al. [130]	2016	SIP	DDoS	statistical (CPA)	offline	N/A
82	Golait et al. [132]	2016	SIP	DoS (flooding)	statistical	offline	N/A
83	**Tsiatsikas et al. [144]**	2016	SIP	DDoS (flooding)	ML	online	N/A
84	Shrestha et al. [189]	2016	N/A	C&C	ML (SVM)	offline	N/A
85	Dassouki et al. [133]	2017	SIP	DoS (flooding)	fingerprints	online	✓
86	Golait et al. [134]	2017	SIP	DoS (flooding, coordinated)	state machine	online	✓(RET [192])
87	Wu et al. [135]	2017	SIP	DoS (flooding)	statistical (bloom filter)	online	✓(Sketch [40])
88	Kurt et al. [131]	2018	SIP	DDoS (flooding)	Statistical	online	✓([193], [128])
89	Hosseinpour et al. [42]	2018	SIP	DDoS (malformed, flooding)	FSM	offline	✓(SIPAD [14], [161])
90	Semerci et al. [136]	2018	SIP	DDoS	statistical (mahalanobis dist.)	online	N/A
91	**Tsiatsikas et al. [43]**	2019	SDP	DoS (malformed) - C&C	rule-based	online	N/A

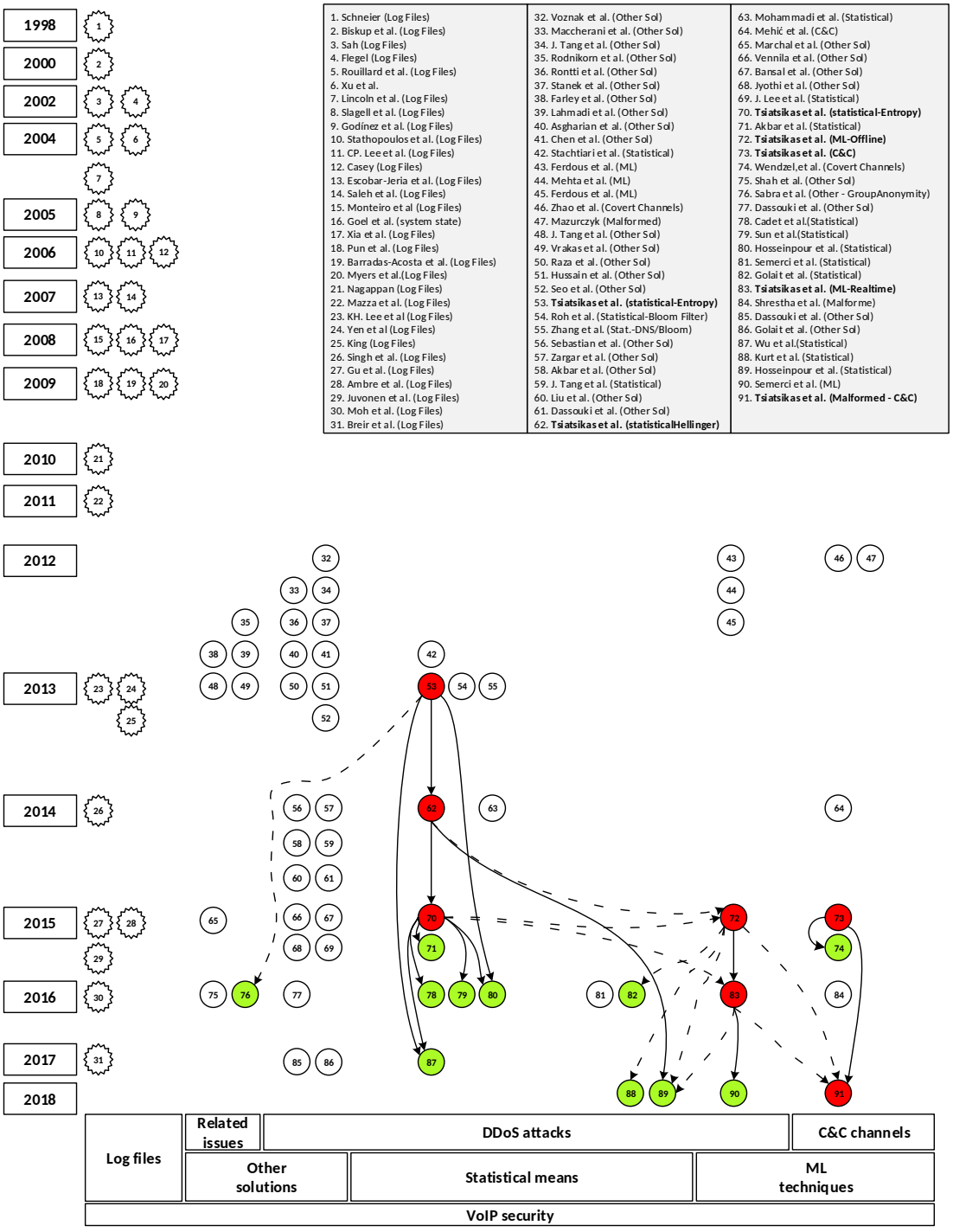

FIGURE 4.1: Overview of the literature map

Chapter 5

Detection of DoS attacks in SIP using statistical means

Until now, various research works [194, 195, 196, 197, 198] have been dedicated to the identification of resource consumption attacks as a part of network Intrusion Detection Systems (IDS). However, as already pointed out, mainly for privacy reasons very few concentrate on the analysis of VoIP audit trails to identify and distinguish uncommon or suspicious traffic. Actually, a straightforward method to analyze audit trail data is to use Entropy. For instance, the authors in [199] employ Entropy theory to detect IP spoofing DoS attacks. This is done by monitoring the distribution of destination/source IP addresses of packets entering or leaving the network. Analogous methods can be utilized in VoIP ecosystem to analyse audit trails (or realtime traffic), but so far their scope is confined to the IP level only. On the other hand, data coming from the application layer are usually rich of information that can be processed towards identifying security incidents. The authors in [129] have identified this potential in theoretical level, but unfortunately the results they provide solely stem from offline analysis using predetermined patterns of SIP traffic.

5.1 Detection of DoS attacks in SIP using Entropy theory

5.1.1 Overview

According to the Entropy theory, symbol redundancy indicates lower Entropy values. This means that symbols having greater frequency occurrence correspond to less "disorder" compared to others coexisting in the same set of messages. In the ideal case, an audit trail should not contain message redundancies, except those that take place due to retransmisions, subscribers peculiarities or habits (a given user calls another very often), and/or other random causes as noted in section 5.1.2. Following this observation, we rely on Entropy to quantify the level of disparity among messages belonging to the same audit trail as well as ones recorded in different log files, and determine whether they contain intrusive records or not. This is because Entropy provides a single-quantity measurement, which corresponds to the randomness of a given audit trail. Note that this quality allow us to even compare audit trails belonging to different multimedia providers, which may be very handy when investigating large-scale distributed DoS (DDoS).

Figure 5.1 offers an overview of the proposed solution to identify DoS incidents in SIP networks. As observed from the figure 5.1, all log files are anonymized before any processing can take place (1). Next, for each message contained in the audit trail, an Entropy value is computed based on the headers of interest and compared against that derived from a reference (attack-free) file (2). If a predetermined threshold is exceeded, then the message is classified as malicious (3). Additionally, if one needs to obtain an estimate on whether the examined audit trail as a whole is suspicious, she can calculate its overall Entropy and have it compared with the value corresponding to the reference file. This means that our scheme allows for auditing the security level of a given audit trail in both per message and per set basis.

It is therefore implied that before we are able to decide whether an audit trail contains a DoS, we require that one of the cross-evaluated audit trail sets is attack-free. Putting it another way, this set is used as a reference (training set) during the detection phase. Hence, the existence of such an attack-free audit trail along with the quality of the latter in terms of proper characterization of the service is of major importance. Actually, this is a well-known issue in intrusion detection, as semi-supervised anomaly detection techniques are able to detect anomalies after being trained by a dataset that has only the

normal instances labeled. Generally, (semi)-supervised detectors rely on the existence of pre-labeled data to build their predictive models for both normal and anomalous classes during the training phase. Newly fed data instances are evaluated against this pre-constructed model, in deployment phase, so that they are categorized in normal or one of the intrusive classes. Realistically, data labeling is not always possible, either due to the high cost of the labeling process, or the sensitive nature of the data. Generally, it is much easier to construct a dataset with normal instances as in most situations normality is the rule. In the case of SIP the correct labeling (identification) of messages can be decisively seconded by the billing service, because these logs are supposed to be accurate and valid.

Entropy is a metric of uncertainty based on the mathematical theory of communication [200] introduced by Shannon. Putting it another way, Entropy quantifies the expected value of the information contained in a message. That is, a reduced uncertainty is quantified in a lower Entropy and vice versa. Hence, the probability of occurrence (certainty of an outcome) of a symbol contained in a SIP message can provide one with knowledge about hidden redundancy in the information received.

Specifically, considering that a symbol S_i in a specific message set (M_{set}) has probability P_{S_i}, then the *itself information* included in this symbol is by definition:

$$I_{S_i} = -\log_b P_{S_i} \tag{5.1}$$

The average of *itself information* with reference to the message set (M_{set}) is called Entropy and is computed using the following formula:

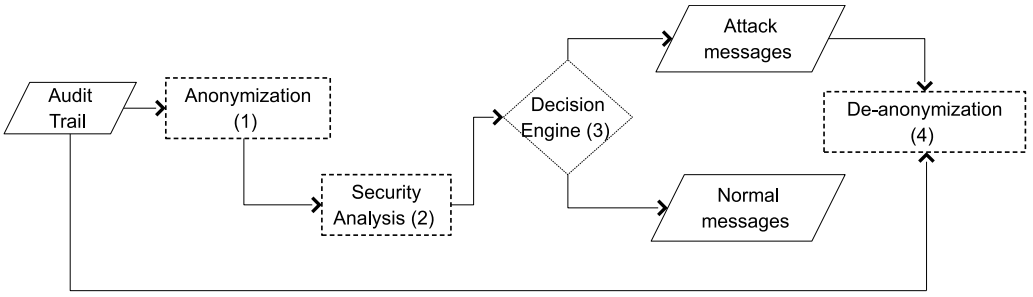

FIGURE 5.1: Abstract view of the proposed audit-trail analysis model

$$H(M_{set}) = -\sum_{i=1}^{n} P_{S_i} * \log_b P_{S_i} \tag{5.2}$$

The Entropy of a source set M_{set} maximizes when all instances (e.g., messages) contained in that set have equal probability of occurrence ($P_{S_i} = 1/n$). This means that the uncertainty of the outcome is augmented, while the redundancy in M_{set} is reduced. With respect to *itself information* this fact indicates that all messages (or symbols corresponding to certain fields of a given message) contain the same amount of information. Note that the greater the probability of a specific message the less information is included in it. Furthermore, in case where two symbols are independent of each other, then the itself information and the Entropy metrics are calculated using the formulas (5.3) and (5.4), respectively.

$$I(A, B) = I(A) + I(B) \tag{5.3}$$

$$H(A, B) = H(A) + H(B) \tag{5.4}$$

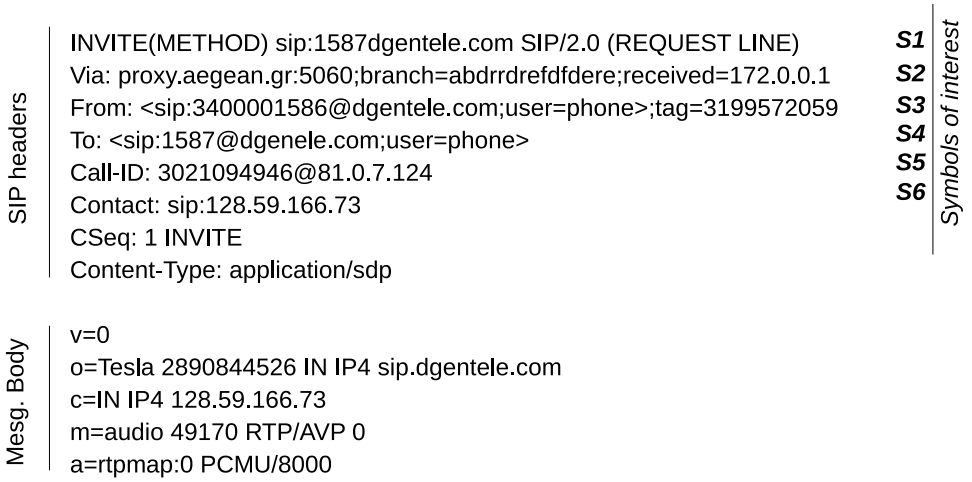

FIGURE 5.2: Symbols of interest in a typical SIP INVITE request

5.1.2 Symbol definition

To apply the above mentioned principles of information theory in the context of a SIP auditing service, we define certain parts of a SIP message (headers) as the symbols of interest, as shown in the right side of figure 5.2. The selection of these symbols reflects the different parts of a SIP message that an attacker could craft in order to launch a resource consumption or other type of attack. In fact, this method of assault is well-documented and evaluated in various researches so far [9, 10, 11, 66]. For instance, a malicious actor could fabricate different SIP messages by modifying some of their parts such as the Request-Line, <Via>, <From>, <To>, <Call-ID> headers (corresponding to symbols S2 – S5, and S1 in figure 5.2) depending on the situation at hand. The last two headers, namely <CSeq>, <Content-Type> shown in figure 5.2 are left out because they bare minimum information regarding message Entropy. That is, their values remain the same across different messages. For example, the latter header will always get the same values corresponding to the session ("application/sdp") parameters of the SIP phone in use. It is to be noted that excluding malicious SIP message tampering, replicated traffic can be also generated due to device misconfiguration or any other random cause. However, this situation is anticipated to happen mostly in small-scale, have short duration, and produce low-volume of extra traffic.

5.1.3 Metrics definition

This section details on the metrics used by the proposed detection scheme.

Actual Information (AI): quantifies the randomness of a message in relation to all the other ones contained in the same set. Recall that in the proposed model, a SIP message is consisted of S1 to S6 symbols and the itself information of each one is calculated using equation (5.1). So, we compute the randomness of each individual message using formula (5.5).

$$AI(M) = \sum_{i=1}^{n} I_{S_i} \tag{5.5}$$

Theoretical Maximum (TM): defines the theoretical maximum randomness value that a message can hold vis-a-vis a particular set of messages (M_{set}). This value is

computed using formula (5.6), where k is the number of symbols defined (6 in our case) and n is the total number of messages existing in that set (M_{set}). Keep in mind that in the case of SIP ACK and BYE messages, the corresponding number of symbols is 5 due to the absence of the <Contact> header.

$$TM = -k * \log_b 1/n \tag{5.6}$$

Normal Average Distance (NAD): represents the average randomness distance of an attack-free traffic from its theoretical maximum value. Its value is computed using formula (5.7).

$$NAD = Avg(TM - AI) \tag{5.7}$$

Actual Information Distance (AID): measures the distance of an examined audit trail message from its theoretical maximum. Its value is computed by formula (5.8).

$$AID = TM - AI \tag{5.8}$$

Normal Threshold (NT): defines the threshold that should not be exceeded by the AID of an examined message in order this message to be classified as normal. The NT value relies on NAD adjusted by a parameter δ, which in turn relies on the characteristics of the examined traffic, NT = NAD + δ.

Audit Trail Entropy (ATE): this metric represents the overall randomness included in an audit trail. It is based on the sum of AI values of all the messages contained in a given M_{set}, and it is computed using formula (5.9).

$$ATE(M_{set}) = \sum_{i=1}^{n} H_{S_i} \tag{5.9}$$

5.1.4 Theoretical example

To exemplify the above metrics, in Table 5.1 we present a simple artificially-created audit trail consisted of 10 messages with different symbols. For instance, the M1 message is comprised of symbols A1, E1, I1, K1, Q1 and U1 corresponding to some of SIP headers existing in the right side of figure 5.2. In this case, the itself information for A1, as calculated by equation (5.1), is 3.32. Also, by using formula (5.5), the AI for M1 is $3.32 + 1 + 0.73 + 0.15 + 1.73 + 2.32 = 9.25$. If we consider this audit trail to be attack-free, and considering that TM is $-6 * \log_2(0.16) = 15.84$, then NAD will be $(15.84 - 9.25)/10 = 0.65$ according to equation (5.7). Moreover, the Entropy values for each symbol S1 to S6 are $3.32, 1, 1.77, 0.46, 1.89$ and 1.92 respectively, while the Entropy calculated over the whole set of messages contained in the audit trail is 10.36. The lower the ATE value is, the less the randomness contained in the examined set.

TABLE 5.1: Entropy method: Example

Msg.	**Symbols**						**AI**
	S1	**S2**	**S3**	**S4**	**S5**	**S6**	–
M1	A1	E1	I1	K1	Q1	U1	9.25
M2	A2	E2	I1	K1	Q2	U2	9.25
M3	A3	E1	I1	K1	Q3	U3	9.25
M4	A4	E2	I1	K1	Q1	U4	8.25
M5	A5	E1	I1	K1	Q2	U4	8.25
M6	A6	E2	I1	K1	Q3	U4	8.25
M7	A7	E1	I2	K1	Q1	U4	8.25
M8	A8	E2	I3	K1	Q2	U1	11.84
M9	A9	E1	I4	K1	Q3	U2	11.84
M10	A10	E2	I5	K2	Q4	U3	16.6
ATE	3.32	1	1.77	0.46	1.89	1.92	–

So, it is straightforward that whenever one wishes to identify whether a message contained in any given audit trail is malicious or not, requires her to compute its AI and AID (subsequently) and then compare the latter against the NT computed over the attack-free audit trail.

5.1.5 The proposed scheme

Initially, the data contained in the audit trail files are anonymized to obfuscate user specific information contained in SIP headers. As already pointed out in section 2.5, this is done by creating a keyed hash (HMAC) [201] value for each header of interest. Bear in

mind that this information corresponds to the symbols S1 to S6 defined in figure 5.2 and detailed in Table 5.2. This (keyed) hashing phase allows one to retain symbols frequency while obscuring the initial values. That is, due to the intrinsic properties of HMAC functions, the chances for someone to reverse the hash is almost non-existent, while the output produced allows for exchanging the data among providers or between a provider and a public data analysis service. Also, it should be noted that even if the key (k) is compromised the original values (x) cannot be reversed engineered from HMAC(k, x) = value. However, such a situation will constitute the data vulnerable to brute force attacks. This observation also applies to cases where the access to the original data and the secret key used to create the corresponding HMAC is managed by different entities.

Thus, as shown in figure 5.1, in case there is a need to identify each one message classified as malicious by the framework, the service provider searches for its hash in the already anonymized audit trail. If a match is found, then the initial message can be retrieved from the original data for further examination. Of course, another possibility is for the detection framework to send back to the service provider only the serial number of the messages detected to be suspicious. This way, we can publicize information and outsource data for security related analysis that otherwise would remain private.

Recall from section 2.5, however, that while this HMAC-powered anonymization scheme is very fast, it is only fair when it comes to unlinkability [202]. That is, the hash values of two headers pertaining to the same user, say, the caller in an INVITE request, will be identical. Therefore, a person having access to the anonymized data is able to deduce certain relationships among the data, even if the real identities of the corresponding persons remain hidden.

TABLE 5.2: Symbols of interest contained in a SIP message

Symbol	**Corresponds To**	**Symbol**	**Corresponds To**
S1	Request-Line (requested resource)	S2	Via header
S3	FROM header	S4	TO header
S5	Call-ID header	S6	Contact header
S7	entire SIP message	-	-

After the obfuscation phase, the randomness included in the audit trail of interest is quantified. Having in mind the metrics defined in section 5.1.3, we calculate (a) the AI per message, (b) the AID for the examined set, and (c) the ATE over the whole set of messages contained in the audit trail file. As already pointed out, to identify abnormalities (which may denote a DoS attack) it is assumed that there exists at least one attack-free audit trail, to be used as a training set for calculating the NAD and (consequently) the NT metric. The latter value is used to decide if a given message is intrusive or not. This procedure is presented in detail in Appendix A.1, algorithm 1.

5.1.6 First-level offline evaluation

For assessing the effectiveness of the proposed method in detecting abnormalities in SIP traffic we used the well-known open source *Kamailio* (http://www.kamailio.org/w/) as a VoIP server. For generating the background and attack traffic we employed *sipp* (`http://sipp.sourceforge.net/`) and *sipsak* (`http://sipsak.org/`) respectively. The VoIP server records all the traffic, which will be used for offline analysis by the proposed framework.

In an effort to assess the effectiveness of our scheme to identify DoS and traffic abnormalities in general, we implemented fourteen scenarios summarized in Table 5.3. Each one was executed for a period of 120 sec. In all the scenarios it is introduced different (legitimate) background traffic, while various attacks have been simulated in order to examine if they can be traced by the proposed solution and to which degree. We should also stress out that the rate of traffic used in each scenario included in Table 5.3 corresponds to the rate the tools (sipp, sipsak) are pre-configured to operate with. Scenarios SN-1, SN-2 and SN-3 serve as references for attack-free traffic in order to assess the proposed solution under different traffic patterns, and therefore are used for calculating the NT as well as the NAD metrics for the sub-scenarios.

5.1.7 First-level offline detection

Initially, we anonymize the audit trail data by hashing the pre-defined symbols per message. Specifically, for each message contained in the audit trail, we employ SHA-1 to obtain the hash of every symbol defined in Table 5.2. Hashing allow us to keep symbol

TABLE 5.3: First-level evaluation: Description of the scenarios evaluated

Scenario Number	**Description**
SN-1	30 legitimate users establishing 5 calls/sec. The maximum number of calls per user/sec is 2. This scenario contains no attack traffic.
SN-1-1, SN-1-2, SN-1-3	These 3 sub-scenarios use the background traffic of SN-1 and single source SIP INVITE flood attack traffic with a rate of 5, 12 and 30 calls/sec.
SN-2	30 legitimate users establishing 5 calls/sec. The maximum number of calls per user/sec is 5. This scenario contains no attack traffic.
SN-2-1, SN-2-2, SN-2-3, SN-2-4	These 4 sub-scenarios use the background traffic of SN-2 and single source SIP INVITE flood attack traffic with a rate of 8, 20, 40 and 80 calls/sec.
SN-3	30 legitimate users establishing 2 calls/sec. The maximum number of calls per user/sec is 600.
SN3.1, SN-3-2, SN-3-3, SN-3-4	The last 4 sub-scenarios use the background traffic of SN-3 and multiple source SIP INVITE flood attack traffic with a rate of 25, 50, 175, 350 calls/sec.

frequency unmodified, while obscuring the initial values. This means that the initial information (i.e., the whole message) can be retrieved only if the audit trail becomes available. In case that it is needed to identify the exact initial messages, the audit trail is anomyzed and each of the attack messages is compared against the anonymized ones. If there is a match then we extract the initial message for further examination.

Naturally, other anonymization techniques [203], discussed already in Chapter 2, and detailed in section 2.5, such as hiding, permutation or enumeration can also be utilized here. However, such approaches require keeping metadata information in secure storage for the case where the initial messages need to be retrieved. This fact constitutes these schemes more complex, and of course vulnerable to attacks, as this additional information is required to be stored in a secure manner. Also, as reported in [56], anonymization can have severe undesirable outcomes if implemented incorrectly.

Next, security analysis is applied to the anonymized data for measuring the uncertainty included in the audit trail. More specifically, in the proposed model we compute: (a) the AI per message, (b) the AID for the examined set, and (c) the ATE over the whole set of

messages contained in the audit trail file. In order to identify abnormalities (which may indicate a DoS attack) we assume that there exist at least one attackfree audit trail, to be used as a training set for calculating the NT as well as the NAD metrics (see section 5.1.3). Then, the computed values are checked against the corresponding thresholds in order to have the messages classified as attack or normal traffic. To extract the exact message details and reveal additional information related to the attack message(s) we use the hash values included in the malicious set and search for the corresponding values in the audit trail. It should be noted that there is no need to access the initial values since there is no match between these values. The initial values are retrieved only if the hash values match. This way, we can publicize information and outsource data for security related analysis that otherwise are considered private.

Figure 5.3 illustrate snapshots of the distribution of the AI metric for scenarios SN-1, SN-1-1, SN-1-2 and SN-1-3. Note that similar distributions have been recorded for the remaining scenarios. When compared to the other scenarios, the AI in SN-1 is closer to its TM value due to many retransmittions and the call pattern used. That is, many retransmissions occurring in a short period of time may falsely indicate DoS traffic. Although, we rely on a simulated environment, this is also the case for real architectures, where users build a specific call pattern during a particular period of time [204], [205].

In all the attack sub-scenarios the AI metric obtains lower values due to excessive symbol repetitions in the examined set of messages. This behavior is distinctively depicted in figure 5.3, as the attack traffic is increased gradually for scenarios SN-1-1 to SN-1-3 respectively. For instance, in SN-1, the TM is 46.06 and the average AI 31.78, while in attack sub-scenarios, say SN-1-1, both the TM and average are increased to 67.78 and 36.75 respectively. Recall that lower values in AI is a strong indication of uncommon behavior. This indication can be used also in cases where an attack-free audit trail is not available. However, as already explained in 5.1.6, in such an unusual case, we should take into account either the theoretical users' behavior or employ other techniques aiming to estimate the appropriate threshold. It is also relevant to note here that the values related to TM vary among different scenarios, since the examined set includes different number of recorded messages.

For all the scenarios the NT metric is adjusted to the corresponding traffic pattern using the δ parameter. In our case, this parameter is equal to the St. Dev. value calculated

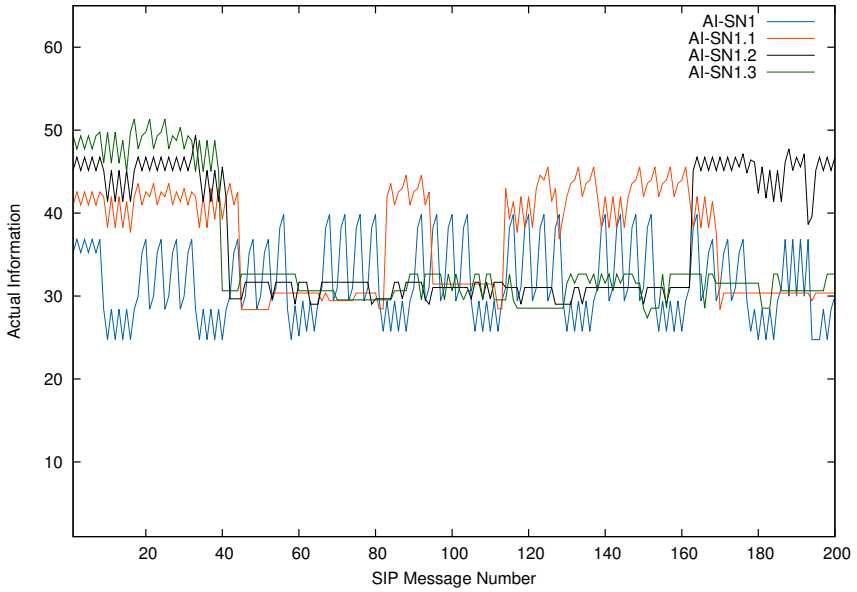

FIGURE 5.3: Sample of actual info (AI) for scenarios SN-1, SN-1-1, SN-1-2 and SN-1-3

over the messages that consist the corresponding normal traffic set. This means that in order to deduce if a particular message is part of an uncommon traffic pattern we compute its AI (according to equation 5.5) and compare it with the NT value. The statistics for all the scenarios are summarized in Table 5.4.

To evaluate the accuracy of the proposed solution in identifying DoS attacks we use legacy IDS error assessment metrics, namely False Positive (FP) and False Negative (FN) [206]. In this context, the attack traffic is logged in parallel and independently in order to use it on a later stage to validate the correctness of our proposal to classify a message as malicious or not. The results of this analysis are summarized in Table 5.4 as well. To further validate the outcomes, we also analyse considerable volume of legitimate traffic having similar patterns to SN-1, SN-2, SN-3. In all these cases, no false alarm was detected meaning that our solution is sound. Regarding the results derived from the rest of the scenarios we were able to detect only FP. Particularly, in all the scenarios, we identify accurately all the attack traffic, while the FP value for all SN-1 to SN-3 sub-scenarios fluctuates between 1% to 3.5%. Bear in mind that the rate of FP is highly affected by possible retransmissions and users' call behavior depending on the case.

In this point, one might argue that an FP of 3.5% is quite significant. However, in forensic analysis, privacy-preserving solutions need to always balance between security and privacy. We argue that this percentage is very promising; however, along with

the calibration of the metrics currently used in our model, there might exist additional parameters that may affect its behavior and thus lead to better results. This is an issue worth of investigating in a future work. It is also important to note that if we solely consider the whole message as an independent symbol (S7), then the AI will receive the maximum theoretical TM value of this set. This happens because every SIP message, either legitimate or not, always presents some additional fields or parameters that uniquely differentiate it from any other. Obviously, if doing so, one will end-up believing that the audit trail under investigation is attack-free. Thus, our model makes use of the AI metric which involves information stemming from different, but clearly defined, symbols of the SIP message structure.

5.1.8 Second-level offline evaluation

This subsection details on the second-level evaluation of the proposed scheme. The second-level evaluation differs from the first-level in the number of the examined scenarios and the side-by-side comparison it offers, with respect to the offline and the realtime analysis.

The proposed detection scheme was implemented as two independent software modules; one for contacting offline analysis of audit trail data, and the other for inspecting SIP messages in realtime. Both modules were programmed in C language and are capable of processing any type of SIP message either request or response. The realtime module is built as an extension of the well-known SIP server Kamailio [207].

For both modules, before doing any security analysis, the data are anonymized through the HMAC-SHA256 function [208] and stored in a hash table. In fact, the hash table structure stores a different record per unique SIP symbol found in the examined messages. The SIP symbol is the alias-key which is used to retrieve the corresponding value from the hash table.

As detailed later in section 5.1.11 and summarized in algorithm 1, in the case of realtime detection, the data remain in the hash table for a predefined message-window M_w. As soon as the M_w expires, the module calculates the NT, NAD metrics over the received messages up to that moment and compares the AID of every next incoming message against that of NAD.

TABLE 5.4: First-level evaluation: False alarm ratio and statistics for all the scenarios

Scen.	Traf.		FP		Stats.			
	RC	AC	In.	%	TM	St. Dev	Thrsh.	AI MV
SN-1	219	-	-	-	46.60	4.78	14.88	31.78
SN-1-1	443	199	0	-	52.70	4.88	19.76	36.60
SN-1-2	715	464	8	1.1	56.89	5.54	20.42	35.76
SN-1-3	963	721	-	-	59.46	5.79	20.67	35.54
SN-2	891	-	-	-	52.80	4.76	13.04	40.13
SN-2-1	892	438	32	3.5	58.80	5.21	18.25	39.43
SN-2-2	1095	644	33	3.0	60.58	5.55	18.59	38.97
SN-2-3	2683	895	28	1.0	62.34	5.78	18.82	38.60
SN-2-4	3655	1389	29	-	65.01	5.94	18.98	38.22
SN-3	2275	-	-	-	60.91	4.39	14.39	46.51
SN-3-1	5031	1422	39	0.7	67.78	4.43	18.82	45.85
SN-3-2	5769	1798	35	0.6	68.96	4.53	18.92	45.70
SN-3-3	9683	3899	32	0.33	73.44	4.81	19.2	45.03
SN-3-4	17317	7800	41	0.23	78.47	5.10	19.49	44.75

For offline analysis the exact same metrics are calculated, but this time over the whole traffic corresponding to the audit trail at hand. This procedure is summarized in algorithms 2, 3. However, if one needs to obtain a security assessment of the messages received during a specific period of time, the audit trail can be split into segments over which the metrics can be calculated. By doing so, one is able to achieve a similar approach to the M_w used in realtime analysis. Algorithm 5, presented in Appendix A.1, exemplifies this procedure.

5.1.9 Testbed setup

To evaluate the effectiveness of the proposed scheme in detecting abnormalities in SIP traffic we used a properly designed testbed as shown in figure 5.4. To simulate different

types of both legitimate and attack (flood) traffic we employed *sipp v.3.2*[1] and *sipsak*[2] tools respectively. Table 5.5 summarizes all the scenarios used for the evaluation of our scheme. The main scenarios, namely SN-1 to SN-5, serve as references for attack-free traffic and therefore are used for calculating the base-value of NAD and subsequently the NT metric. The latter will be compared against that of AID calculated per message for every sub-scenario. To simulate legitimate traffic in terms of incoming calls in a realistic manner we employed an exponential inter-arrival time distribution ($\lambda = 100$), similar to that used in evaluating SIP server performance [209].

Apart from assessing the effectiveness of the modules to detect suspicious traffic, we also measure the introduced overhead of the one destined to realtime operation. For the latter, the SIP server has been configured to operate in single-thread mode, representing a worst case scenario. The server was running on an i7 2.2 GHz machine having 6 GB of RAM.

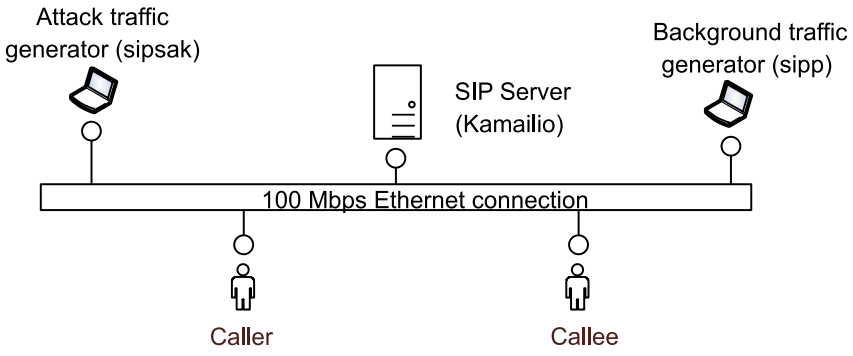

FIGURE 5.4: Deployed testbed

5.1.10 Second-level offline detection

Figure 5.5 illustrates a snapshot of the AI metric distribution for the main scenarios SN-1 and SN-2. It can be observed that this metric follows a nearly similar pattern for both scenarios. Analogous distributions are perceived for the remaining attack-free scenarios as well. In fact, this slight difference is due to the disparate rate in calls per second each scenario incorporates. This call distribution represents the calls initiated by legitimate users in a typical SIP ecosystem as explained in [204, 210]. Contrary to that, as observed from figures 5.6 and 5.7, the AI produced in attack scenarios SN-1-2 and SN-2-2 exhibits

[1] http://sipp.sourceforge.net/
[2] http://sipsak.org/

TABLE 5.5: Second-level evaluation: Description of the scenarios evaluated

Scenario Number	**Description**
SN-1	This attack-free scenario simulates 30 legitimate users establishing 5 calls/sec.
SN-1-1, SN-1-2, SN-1-3	These sub-scenarios use the background traffic of SN1 and simulate a single source SIP INVITE flood attack with a rate of 20, 40, 80 calls/sec respectively.
SN-2	It simulates 30 legitimate users establishing 2 calls/sec. This is an attack-free scenario as well.
SN-2-1, SN-2-2, SN-2-3	These sub-scenarios use the background traffic of SN2 and simulate multiple sources of SIP INVITE flood attack with rates of 50, 175, 350 calls/sec respectively.
SN-3	It simulates 50 legitimate users establishing 120 calls/sec. This scenario contains no attack traffic.
SN-3-1, SN-3-2	These sub-scenarios employ the background traffic of SN3 and simulate a single source SIP INVITE flood attack of 400, 1200 calls/sec respectively.
SN-4	This attack-free scenario incorporates 50 legitimate users establishing 120 calls/sec.
SN-4-1	It relies on the background traffic of SN-4 and simulates 24 single source SIP INVITE floods each one with 800 calls/sec.
SN-5	This last attack-free scenario incorporates 50 legitimate users establishing 20 calls/sec.
SN-5-1	It relies on background traffic of SN-5 and simulates 16 single source SIP INVITE floods each one with 266 calls/sec.

high fluctuations. The calculated statistical results for all the scenarios are summarized in Table 5.6.

It should be noted that in all the attack sub-scenarios the AI obtains greater values due to the higher number of messages existing in the corresponding audit trails. However, when an attack unfolds, the AI for messages belonging to the attack traffic receives lower values. Thus, the NAD metric needs to be adjusted according to the parameter

δ (see section 5.1.3) for calibrating it to the current traffic pattern. In our case, the δ parameter is equal to the *Standard Deviation* value calculated over the messages that consist the corresponding normal traffic set. This means that in order to deduce if a particular message is part of an uncommon traffic stream, we compute its AID (according to equation 5.8) and compare it against the NT value. If the latter is exceeded, the message is characterized as suspicious.

Generally, the existence of excessive symbol repetition in any given audit trail is a high indication of uncommon user behavior because legitimate users are not capable of generating high volumes of traffic in a short period of time, unless, for example, their device is infected by a malware. In opposite to that, low values in symbol (S_i) recurrence is a strong indication of normal traffic, and thus it can be used in cases where an attack-free audit trail is not available. However, in such an unusual case, one should take into account either the theoretical users' behavior or employ other techniques (as in [211, 210]) aiming to estimate the appropriate threshold.

To evaluate the accuracy of this detection module in identifying DoS attacks we use legacy IDS error assessment metrics, namely False Positive (FP) and False Negative (FN) [206]. The first one is related with messages detected as abnormal but they belong to the legitimate traffic, while the latter involves messages detected as normal but they

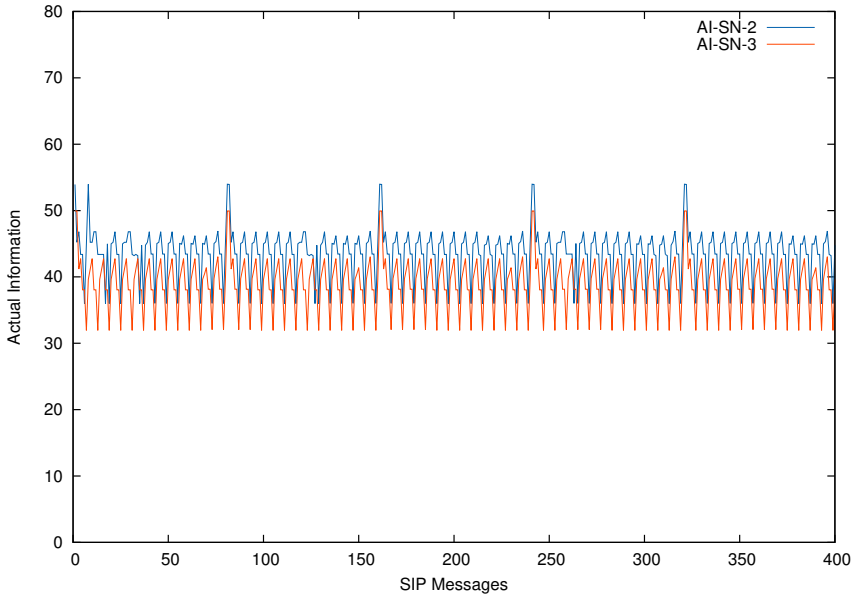

FIGURE 5.5: A snapshot of the AI for scenarios SN-1, SN-2 (normal traffic only)

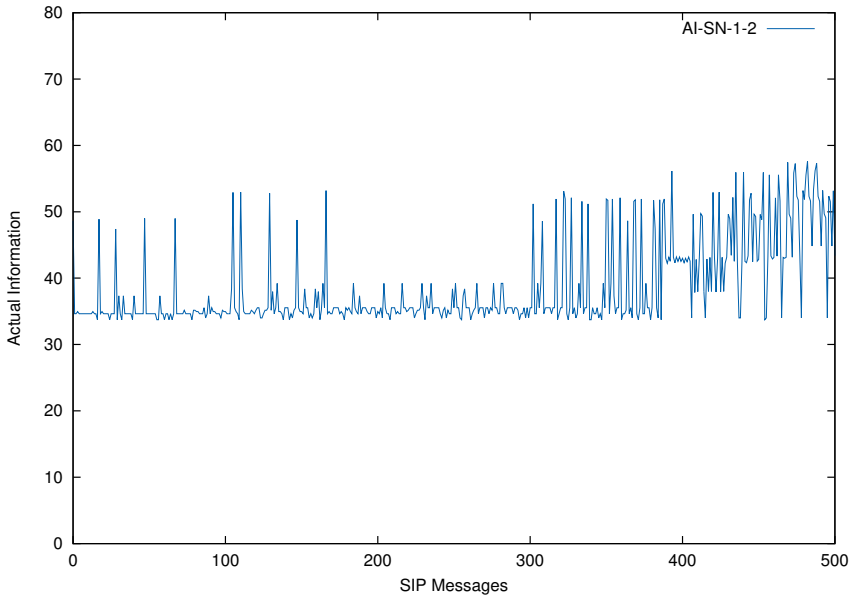

FIGURE 5.6: A (random) snapshot of the AI for scenario SN-1-2

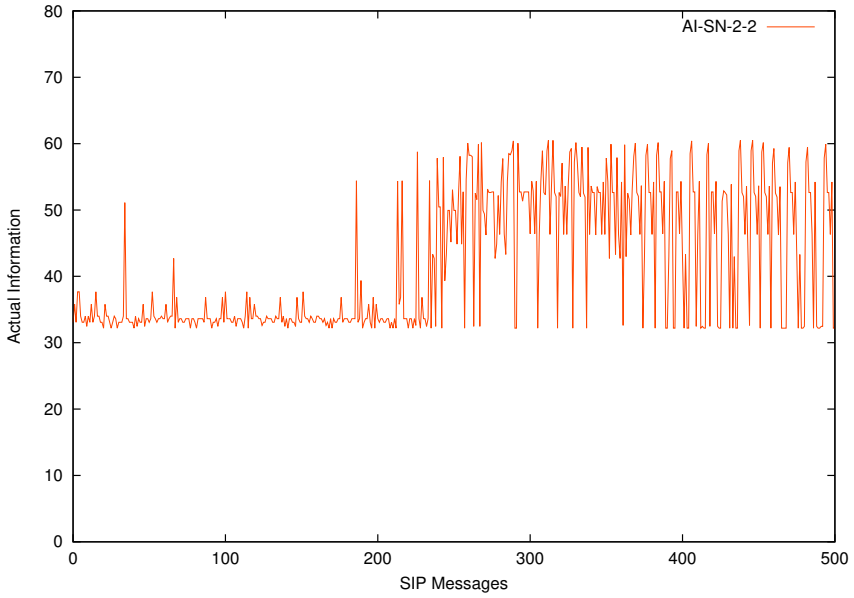

FIGURE 5.7: A (random) snapshot of the AI for scenario SN-2-2

belong to abnormal traffic. To further validate the outcomes at a later stage, we mandate all the attack traffic to be recorded in a separate log file.

The FP and FN results for all the scenarios are summarized in Table 5.7. It is observed that the FP percentage varies from 0.3% to 10.6%, while that of FN reaches 1.8%. For instance, in SN-1-1 to SN-1-3 the FP fluctuates between 7.5% and 10.6%. This is because aggressors usually do not solely rely on random messages to cause DoS, but employ smart

spoofing schemes where the attack messages are crafted to contain legitimate SIP addresses/headers. So, some of these specially manipulated messages would remain hidden under the normal traffic, while other pertaining to normal traffic are being classified as malicious, generating FP alarms. In this respect, it can be said that such an FP is rather expected since it corresponds to spoofing attacks which is very difficult to defend at least in their early stages or in cases the attacker uses "low and slow" attack techniques. On the other hand, the FN percentage is almost negligible, around 2% in the worst case. This result is particularly encouraging as it makes harder for an attack to go totally undetected.

It is also worth noting that if we solely consider the whole message as an independent symbol, then the AI obtains the maximum theoretical TM value. This happens because every SIP message, either legitimate or not, always presents some additional fields or parameters that uniquely differentiate it from any other. Obviously, if doing so, one will end-up believing that the audit trail under investigation is attack-free. Thus, our model makes use of the AI metric which involves information stemming from different, but clearly defined, symbols of the SIP message structure. Of course, more SIP headers can be added or removed depending on the case.

TABLE 5.6: Second-level evaluation: Statistics for all the scenarios

Scenario	**TM**	**St. Deviation (δ)**	**Threshold**	**AI mean value**
SN-1	69.94	3.83	33.3	43.47
SN-1-1	80.75	7.14	43.94	43.95
SN-1-2	81.85	7.01	44.12	44.74
SN-1-3	83.80	6.60	44.92	45.48
SN-2	61.93	3.83	26.87	38.89
SN-2-1	79.67	7.08	45.43	41.32
SN-2-2	83.43	6.83	47.86	42.40
SN-2-3	86.71	5.32	47.18	44.85
SN-3	96.88	4.02	45.88	55.02
SN-3-1	108.92	5.16	56	58.08
SN-3-2	117.15	4.48	61.65	59.98
SN-4	99.80	3.85	46.87	56.78
SN-4-1	117.23	4.95	61.2	60.98
SN-5	89.73	4.09	44.23	49.58
SN-5-1	105.51	4.77	55.31	54.97

TABLE 5.7: Second-level evaluation: False alarm ratio

Scenario	Traffic		FP		FN	
	Total calls	Attack calls	Instances	%	Instances	%
SN-1	3,230	0	0	0	0	0
SN-1-1	11,263	7,204	1,195	10.6%	0	0
SN-1-2	12,790	8,523	1,132	8.8%	0	0
SN-1-3	16,019	11,763	1,203	7.5%	0	0
SN-2	1280	0	0	0	0	0
SN-2-1	9,936	8,291	391	3.9%	0	0
SN-2-2	15,340	13,657	458	2.9%	0	0
SN-2-3	22,420	20,799	496	2.2%	0	0
SN-3	72,553	0	0	0	0	0
SN-3-1	291,634	228,432	3,104	1.0%	4,111	1.4%
SN-3-2	754,510	705,936	12,182	1.6%	7,274	0.9%
SN-4	101,762	0	0	0	0	0
SN-4-1	761,963	694,788	2,479	0.3%	6,873	0.9%
SN-5	31,775	0	0	0	0	0
SN-5-1	196,761	168,470	5,835	2.9%	3,696	1.8%

5.1.11 Second-level realtime detection

This section reports on the results obtained from the realtime detection module running on Kamailio SIP server. The evaluation scenarios remain the same as in the case of offline analysis in an effort not only to assess the accuracy of the realtime module but also to cross-evaluate their outcomes. These scenarios are presented in detail in Table 5.5. It is to be noted that the main difference between the realtime and offline modules is the amount of traffic that is available to the realtime module at a given time. This is because the offline module works based on statistical metrics derived from all the available traffic, which naturally is not the case for the realtime one. Thus, as already pointed out, a M_w is used as a training phase to provide an estimation of NT and NAD metrics. This procedure is illustrated in figure 5.8. Of course, these metrics can be automatically readjusted as further traffic is available, similar to the offline approach. However, the adjustment of TM and NAD metrics needs to take into account the M_w parameter as well. This also means that the selection of M_w is critical to the accuracy of realtime detection. To our knowledge, there is no direct approach to formally define these parameters, mainly because they are highly contextual, i.e., closely bound to the characteristics of the service and underlying network. Therefore, similar to other anomaly-based approaches [32], we adopted an error-trial approach to balance between the *Mw* parameter and the false alarm rate.

In a nutshell, as the M_w expires, the module starts to monitor every incoming SIP message to decide if it is normal or not. This is done by computing its AID according to equation (5.8) and comparing it against the NT value.

Table 5.8 overviews the average processing time per message introduced in SIP server as well as other statistical metrics (min, max, st.dev) for realtime operation for all the scenarios. Note that the table contains the overhead induced when a SHA256 or HMAC-SHA256 operation is involved for data anonymization. This is done to acquire a cleaner view of the applicability of each method in terms of performance. As illustrated in the Table, for the HMAC-SHA256 case, this time penalty remains low at ≈ 3.9 and 1.8 msec considering the first two attack-free scenarios, namely SN-1, SN-2. This is expected as these scenarios have the lowest call rate of 5, 2 calls per second respectively.

However, as the volume and the arrival rate of messages increases, the introduced overhead is expected to augment as well. For example, the average overhead in SN-4 is ≈ 16.3 msec, while for SN-5 is ≈ 14.9 msec. Even in the worst case scenario of SN-3-2, the mean value of this metric is ≈ 16.8 msec. This observation is further supported by the standard deviation per scenario, which fluctuates between 2.8 and 9.7 msec. Therefore, it can be safely argued that the use of HMAC-SHA256 does not generate a significant increase in overhead. So, as already discussed in Chapter 2, and more specifically in section 2.5, this stronger anonymization method is definitely to be preferred over that of a hash function.

To assess the effectiveness of the realtime module in terms of false alarms we use different values for the M_w, ranging from 100 to 1000 messages. This will allow for fine-tuning of the M_w parameter based on the recorded FP, FN metrics, and provide a general estimation on how the M_w affects these metrics. Figure 5.9 illustrates the variation of FP for all the scenarios. Taking SN-1-1 to SN-1-3 as examples, no FP is observed when the M_w varies from 100 to 1000 messages. However, for scenarios SN-2-1 and SN-2-2, when the M_w is set to 1000 messages, a rather negligible FP of $\approx 1.9\%$ is perceived. A

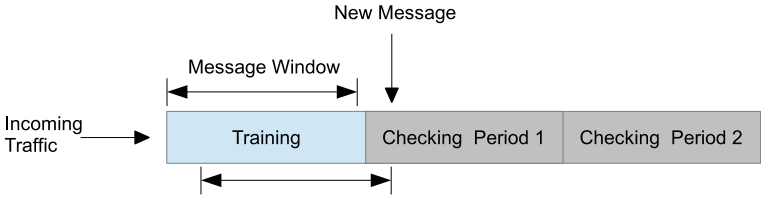

FIGURE 5.8: Abstract view of the *Message window* (M_w)

TABLE 5.8: SIP server overhead (in msec)

Scenario	SHA256 / HMAC-SHA256			
	Min	**Max**	**Average**	**St. Deviation**
SN-1	0.440 / 0.423	36.079 / 44.711	3.812 / 3.899	6.929 / 7.274
SN-1-1	0.325 / 0.413	40.869 / 40.800	15.354 / 15.700	9.311 / 9.251
SN-1-2	0.330 / 0.295	48.262 / 43.710	15.201 / 15.627	9.274 / 9.156
SN-1-3	0.338 / 0.312	43.873 / 43.324	15.658 / 15.951	9.186 / 9.383
SN-2	0.464 / 0.422	32.233 / 28.354	2.124 / 1.837	2.731 / 2.867
SN-2-1	0.396 / 0.342	59.128 / 65.678	15.113 / 15.426	9.290 / 9.429
SN-2-2	0.395 / 0.449	44.651 / 51.038	15.315 / 15.328	9.384 / 9.303
SN-2-3	0.395 / 0.344	178.276 / 58.846	15.476 / 15.510	10.645 / 9.745
SN-3	0.346 / 0.373	45.720 / 124.347	15.874 / 16.028	8.385 / 8.765
SN-3-1	0.308 / 0.276	72.268 / 68.205	16.601 / 16.597	8.537 / 8.678
SN-3-2	0.295 / 0.349	60.860 / 62.937	16.806 / 16.841	8.365 / 8.516
SN-4	0.295 / 0.330	155.355 / 511.134	16.167 / 16.256	8.849 / 8.173
SN-4-1	0.209 / 0.440	300.404 / 56.473	16.511 / 16.744	8.560 / 8.067
SN-5	0.307 / 0.291	36.196 / 47.506	14.734 / 14.894	8.236 / 8.383
SN-5-1	0.319 / 0.320	55.762 / 65.981	15.534 / 15.668	8.722 / 9.090

similar FP distribution is recorded for scenarios SN-4-1 and SN-5-1, showing in all cases an FP percentage below 2%.

It is also important to note that while in the case of offline analysis the FP was approximately 10%, here is much smaller ($\approx 2\%$). This betterment is because the realtime module is able to detect bursts of DoS traffic taking place within the predefined M_w (due

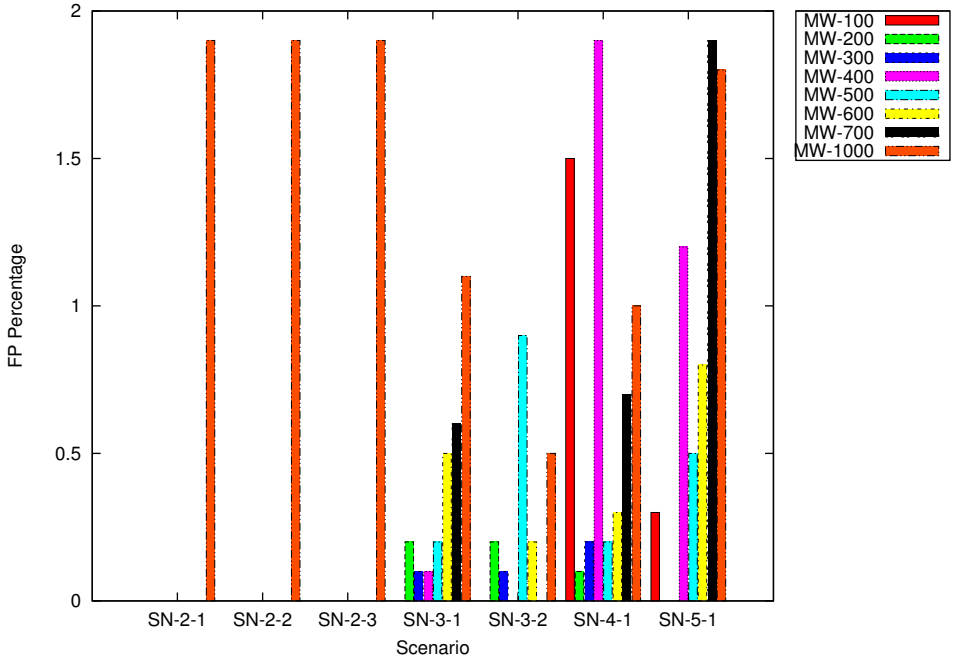

FIGURE 5.9: FP distribution under different M_w (all scenarios)

to the decrease of the AI that in turn triggers the threshold). This includes spoofing attacks exploiting IP addresses belonging to legitimate users. Contrary to that, the offline module cannot detect such cases as the attacks are spread along the entire audit trail. In any case, as mentioned previously, if the audit trail is split into multiple data segments one can achieve similar results for the offline module as well. Nevertheless, in that case, the threshold needs to be adjusted to the appropriate segment following a normalization procedure.

Figures 5.10 and 5.11 offer an overview of FN percentage under different configurations of the M_w parameter. As illustrated in the figure 5.10, when the M_w is configured to a low value it produces high FN percentages in almost every scenario. For instance, in SN-2-* sub-scenarios as the M_w augments from 100 to 1000 messages causes the FN to be decreased rapidly. Particularly, in these sub-scenarios the FN is decreased highly from almost 30% down to $\approx 2\%$ as the M_w increases. In SN-1-*, where the legitimate traffic is slightly increased, compared to that of SN-2-*, the FN mean value is $\approx 5.5\%$ for all the variations of M_w. Note that as the legitimate traffic accumulates, e.g., in SN-3-*, SN-4-* and SN-5-*, the FN is also increased considering the same values of M_w. This behavior is actually expected because the detection engine makes decisions based on a limited volume of incoming traffic. To further verify this result, we experimented with sizes of M_w greater than 1000 messages. It is therefore not surprising that for scenario SN-4-1 when the M_w is set to 4000 messages, an FN of 3.7% is generated (an improvement of 10.6%). Overall, for all the scenarios included in figure 5.11, when the M_w is equal to 1000 we perceive a minimum and maximum FN percentage of 4.8% and 24.5% respectively. On the other hand, when the M_w is increased to 2000, the corresponding values drop to 4.2% and 19.1% respectively. Generally, the results designate that if the M_w is adjusted to the corresponding normal traffic, then the generated FP and (especially) FN alarms can be greatly optimized.

When comparing the FN values outputted by this module against those of the offline one, we can make the following remarks: (a) the minimum value in both cases is very low at about 1%, (b) the maximum values however present a significant difference of $\approx 27\%$ (29-1.8), thus further verifying the above mentioned rule "the larger the M_w the better the FN", (c) elaborating on the latter point, from the results obtained, offline detection seems to be far more robust due to the great mass of information available;

on the downside, the online detection module is easily adjustable, performs sufficiently well, and comes very handy in cases where a timely reaction is desired.

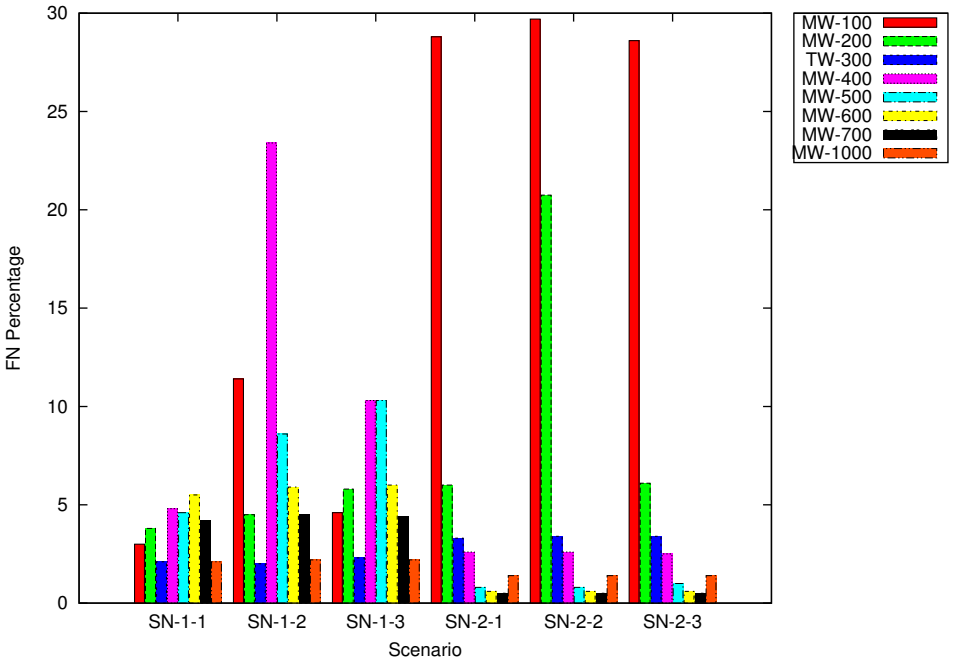

FIGURE 5.10: FN distribution under different M_w (scenarios SN-1-* to SN-2-*)

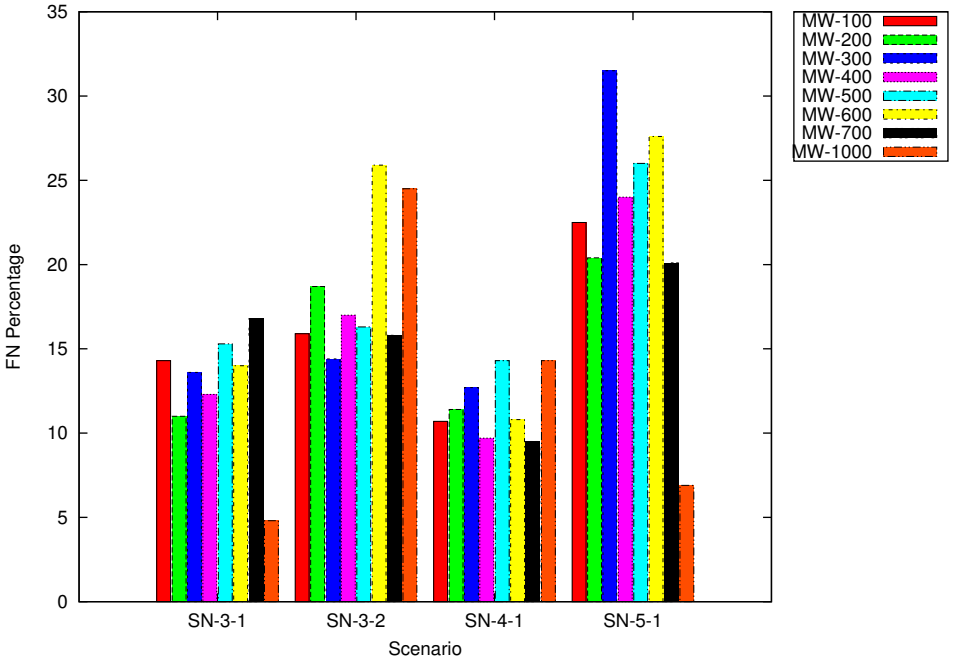

FIGURE 5.11: FN distribution under different M_w (scenarios SN-3-* to SN-5-*)

5.2 Detection of DoS attacks in SIP using Hellinger distance

DoS attacks of high volume are sure to quickly attract the attention of the security guards. But what happens with other security incidents that are of low-volume and usually evade detection? Say, for example, that the aggressor sends out spoofed SIP requests having unresolved IP addresses, but does this at a slow pace and following a carefully designed wake/sleep strategy. Such stealthy incidents may gradually affect the availability of the provided service, mostly reflected to reduced bandwidth, which in turn causes users' dissatisfaction and reduced market share for the provider. To better highlight on this issue we executed a single DoS attack to a standard SIP server with a rate of only 10 INVITE messages per second. Figure 5.12 shows the percentage of memory allocation at the server side during the attack. As observed, under normal operation, the memory consumption induced by the SIP server process is about 35%. However, as this simple attack unfolds, causes the corresponding percentage to increase to around 80%.

It is well-known that depending on the legislation of different countries, Telcos and service providers in general retain signaling data for a certain period of time for billing, auditing and network management and planning purposes. Therefore, these logs can be proved a valuable source of information toward identifying whether or not a given provider has been the target of a DoS. Generally, the analysis of such data could help one to (a) prove the security level of the provided services and investigate the related incidents, and (b) highlight the need of employing additional security protection measures to enhance service availability, say, due to attacks that managed to bypass the already deployed countermeasures.

5.2.1 Overview

We rely on Hellinger distance [36] with the aim to identify abnormal traffic. Hellinger distance is a well-known metric to calculate the deviation between two probability distributions. More specifically, we define as features of the distributions the symbols S_1 to S_6 discussed in 5.1.2 (Entropy) (i.e., Request-Line, <Via>, <From>, <To>, <Call-ID>, <Contact>). These symbols answer to different headers of a SIP message that could be maliciously changed by an adversary.

The distributions are defined according to equation 5.10

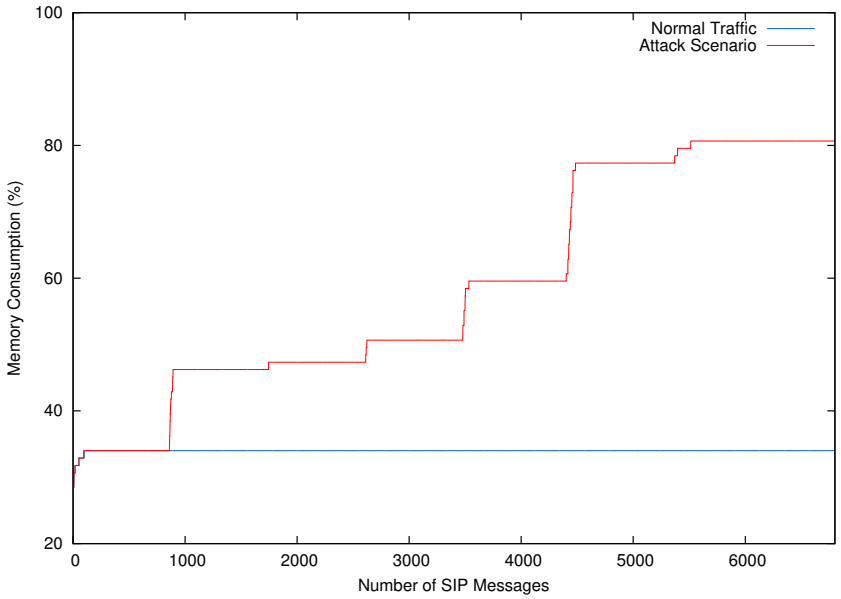

FIGURE 5.12: Memory usage during an attack

$$\begin{aligned} P &= (p_1/N1 + p_2/N1 + ... + p_K/N1) \\ Q &= (q_1/N2 + q_2/N2 + ... + q_K/N2) \end{aligned} \tag{5.10}$$

$$N = \sum_{i=1}^{K} p_i \tag{5.11}$$

$$HD^2 = 1/2 \sum_{i=1}^{K} (\sqrt{p_i} - \sqrt{q_i})^2 \tag{5.12}$$

N is calculated by equation 5.11 in order to normalize the probabilities of the two distributions. Finally, HD can be calculated using formula 5.12.

We take for granted that each distribution consists of K features. In our case this number will vary from 5 to 6 according to the headers of the processed SIP message (request or response). We know that if the two distributions are identical then the HD value will be equal to 0, while the maximum deviation will provide a value of 1.

We assume a training and a testing period. We split the audit trail in segments based on a predefined message window equal to 1000.

Initially, for the training period, after the segmentation of an attack free audit trail, we obtain the frequency occurence for every feature-header (symbol) of a SIP message. After that, we compute the mean HD between this "normal" message and the messages included in the attack free audit trail (1000 messages). The detection threshold will be equal to the mean value plus the standard deviation (we don't use st. dev for the first two scenarios). As soon as we have computed the detection threshold, we calculate the HD value between the normal message and every single message. If the HD value is greater than the predefined threshold then an alert is raised and the message is classified as malicious.

5.2.2 Detection service

To implement a detection service that relies on the Hellinger distance metric to identify traffic abnormalities, one needs to choose certain parts of SIP message as the symbols of interest. By observing figure 5.2 one can conclude that the most important parts of a SIP message are the first 6 lines (headers) corresponding to symbols S1 to S6. According to the literature, these symbols reflect the different types of SIP messages that an aggressor could craft in order to launch a resource consumption attack. For instance, a malicious entity could select to replay the same message or fabricate dissimilar SIP messages by modifying certain segments, including the Request-Line, <Via>, <From>, <To>, <Call-ID>, <Contact> (corresponding to the fourteen different SIP methods) depending on the case. While a detailed analysis of SIP flooding attacks is out of the scope of this work the interested reader could refer to [30, 212].

After the symbols of interest have been selected, the detection service has to be fed with P distribution. That is, a training phase is required taking as input an attack-free log file. Note that P distribution can be updated less or more frequently depending on the provider's needs. On the other hand, the Q distribution is calculated on unknown traffic. Note that contrary to other approaches, P and Q distributions are generated for all the possible types of SIP methods. This has the advantage that it allows one not only to detect an attack incident but also identify its exact type (e.g., an INVITE or BYE flooding). To sum up, for every possible type of message there exists a pair of P and Q distributions. Depending on the case at hand, the audit trail file can be split into several

message segments based on a predefined message window, say, equal to 1000. If so, P and Q have to be computed on the basis of this message window.

Specifically, to train the model we calculate the mean distribution value of every symbol included in the reference sample traffic. This procedure is given in Appendix A.2, algorithm 6. The main role of the training period is to compute the threshold, which is adjusted to the examined traffic by a parameter d, using equation (5.13). In fact, the parameter d coincides to a standard deviation metric, which is equal to the square root of the variance computed over a specific message type in the normal traffic set based on the mean value of HD.

$$Threshold = MeanHD + d \tag{5.13}$$

An unknown log file is examined for its conformity with the already determined normal model. That is, similar to the calculation of P, we reckon symbols distribution for the different requests and responses that formulate the corresponding Q distributions per message type. After that, P and Q distributions for the message type of interest are compared, as detailed in Appendix A.2, algorithm 7. In case that the HD value of the examined message is greater than the predefined threshold, then an alert is raised and the message is classified as malicious.

5.2.3 Theoretical example

To elaborate on the above metric we present an example which is based on two normal messages M1, M2 and one malicious M3, as presented in Table 5.9. The numbers denote the frequency occurence of each symbol in every message.

To calculate the HD of the first two messages (i.e., M1, M2) we will use formulas 5.11 and 5.12.

$N1 = 32 + 3 + 6 + 38 + 6 + 38 = 123$, $N2 = 6 + 1 + 6 + 38 + 6 + 38 = 95$

$$HD = 1/2 * ((\sqrt{32/123} - \sqrt{6/95})^2 + (\sqrt{3/123} - \sqrt{1/95})^2 + (\sqrt{6/123} - \sqrt{6/95})^2 + (\sqrt{38/123} - \sqrt{38/95})^2 + (\sqrt{6/123} - \sqrt{6/95})^2 + (\sqrt{38/123} - \sqrt{38/95})^2) = 0.03$$

We will follow the same steps in order to calculate HD for the messages M1, M3. HD will be equal to $HD = 0.08$. If we additionally calculate the HD between M2, M3 this will provide a value of $HD = 0.16$.

TABLE 5.9: Hellinger distance method: Example

Msg.	Features					
	S1	**S2**	**S3**	**S4**	**S5**	**S6**
M1	32	3	6	38	6	38
M2	6	1	6	38	6	38
M3	993	1	993	1	1	993

5.2.4 Evaluation

The detection accuracy of the proposed scheme has been tested under fifteen different scenarios briefly described in Table 5.10. We simulated distinct patterns for both legitimate and attack traffic using *sipp v.3.2*[3] and *sipsak*[4] tools respectively. Also, for the needs of the experiments, and in order to reproduce realistic call rate conditions, we employed an exponential inter-arrival time distribution ($\lambda = 100$) for legitimate traffic similar to that used in evaluating SIP server performance [209]. The experiments were conducted on an i7 processor 2.2 GHz machine having 6 GB of RAM.

Table 5.11 summarizes the FP and FN results for all the scenarios. As observed, the FP fluctuates between 0.2% and 7.6%, whereas FN reaches a maximum value of 0.002%.

To exemplify the results obtained above, figures 5.13 to 5.16 depict a fragment of HD distribution for scenarios SN-1, SN-1-1, SN-2, SN-2-3, SN-3, SN-3-1, and SN-4, SN-4-1 accordingly. Note that for easy reference and comparison the figures also include HD distribution for the corresponding attack-free traffic scenarios (i.e., SN-1, SN-2, SN-3, SN-4). This is to better conceptualize the fluctuations (increment in our case) exhibited in HD line between normal and intrusive traffic. Taking figure 5.13 and SN-1-1 as an example, one can easily observe that HD values remain as low as ≈ 0.01, while for SN-1-1 the HD fluctuates between ≈ 0.04 and ≈ 0.25. A similar situation is depicted in figures 5.15 and 5.16. Specifically, HD line for SN-3-1 and SN-4-1 abruptly reaches ≈ 0.25 when the attack is initiated. Note that in all the attack scenarios, normal and attack traffic take turns. Naturally, this sudden increase in HD value is due to the

[3] http://sipp.sourceforge.net/
[4] http://sipsak.org/

TABLE 5.10: Description of scenarios used for the Hellinger distance method evaluation

Scenario Number	**Description**
SN-1	It simulates 30 legitimate users establishing 2 calls/sec. This is an attack-free scenario.
SN-1-1, SN-1-2, SN-1-3	These sub-scenarios use the background traffic of SN2 and simulate multiple sources of SIP INVITE flood attack with rates of 50, 175, 350 calls/sec respectively.
SN-2	This attack-free scenario simulates 30 legitimate users establishing 5 calls/sec.
SN-2-1, SN-2-2, SN-2-3	These sub-scenarios use the background traffic of SN-1 and simulate a single source SIP INVITE flood attack with a rate of 20, 40, 80 calls/sec respectively.
SN-3	This last attack-free scenario incorporates 50 legitimate users establishing 20 calls/sec.
SN-3-1	It relies on background traffic of SN-3 and simulates 16 single source SIP INVITE floods each one with 266 calls/sec.
SN-4	This attack-free scenario incorporates 50 legitimate users establishing 120 calls/sec.
SN-4-1	It relies on the background traffic of SN-4 and simulates 24 single source SIP INVITE floods each one with 800 calls/sec.
SN-5	It simulates 50 legitimate users establishing 120 calls/sec. This scenario contains no attack traffic.
SN-5-1, SN-5-2	These sub-scenarios employ the background traffic of SN-5 and simulate a single source SIP INVITE flood attack of 400, 1200 calls/sec respectively.

attack traffic contained in scenario SN-1-1, which in turn is translated into excessive symbol repetition, thus exceeding the predefined threshold. Nevertheless, if the attacker manages to generate traffic that has congruent characteristics to that of normal traffic she may be able to evade detection. On the opposite, however, the impact of the attack is anticipated to be much smaller in terms of probability of occurrence. Also, taking a SIP INVITE method as an example, the aggressor is not able to randomly generate headers of the following format: INVITE sip:x@y:port, where x the username and y the host (domain or IP). This is because the IP used must correspond to an existent address

TABLE 5.11: Summary of evaluation metrics

SN	Traffic (Calls)		FP		FN		Stats (HD)	
	Rec.	Attack	Inst	%	Inst	%	Mean	St. Dev.
SN-1	1426	-	-	-	-	-	0.002	0
SN-1-1	12000	5574	120	1	0	0	-	-
SN-1-2	13000	7238	87	0.6	0	0	-	-
SN-1-3	24530	19622	120	0.4	0	0	-	-
SN-2	3598	-	-	-	-	-	0.041	0.061
SN-2-1	11000	6516	566	5.1	2	0	-	-
SN-2-2	14000	9327	592	4.2	0	0	-	-
SN-2-3	15409	10802	1179	7.6	1	0	-	-
SN-3	12435	-	-	-	-	-	0.161	0.155
SN-3-1	667047	563705	5864	0.8	959	0.001	-	-
SN-4	2505	-	-	-	-	-	0.025	0.028
SN-4-1	178438	168073	8808	4.9	0	0	-	-
SN-5	2004	-	-	-	0	0	0.018	0.021
SN-5-1	261999	195050	1468	0.5	752	0.002	-	-
SN-5-2	667769	601798	1463	0.2	788	0.001	-	-

of the internal network. Otherwise, due to packet filtering rules, the packet will be most likely dropped at the perimeter. If the attacker uses only legitimate IPs to perform the flooding attack, then due to the limited sip URI space, she is not in position to sent too many identical messages.

Generally, the occasional resemblance in terms of probability of occurrence between normal and attack messages is the most prevalent cause of FP alarms. More specifically, FPs are mainly the result of arbitrary device retransmissions or repetitive patterns in user's (call) behavior (e.g., a user calls another one very often). On the other hand, attack messages that appear seldom during the attack incident may generate an FN.

As discussed in the previous section, the threshold for all the scenarios - represented as a flat line in each of the figures - is calculated as the sum of the last two columns of Table 5.11 for all normal traffic scenarios. So, depending on the uniformity of the messages included in each scenario with their mean value, the threshold is expected to set a boundary above which a message is categorized as suspicious. For example, for scenario SN-1 the threshold is nearly zero while for SN-3 is 0.316. Lastly, the scarce upward peaks observed in the normal traffic (blue) line belong to messages that differ considerably from all the others contained in the normal traffic set. That is, their headers appear very frequently causing HD to suddenly spring up. As already pointed out, this may happen due to, say, device retransmissions. On the other hand, the downward

pointing peaks observed in the HD line belong to normal messages that are interposed between the attack ones.

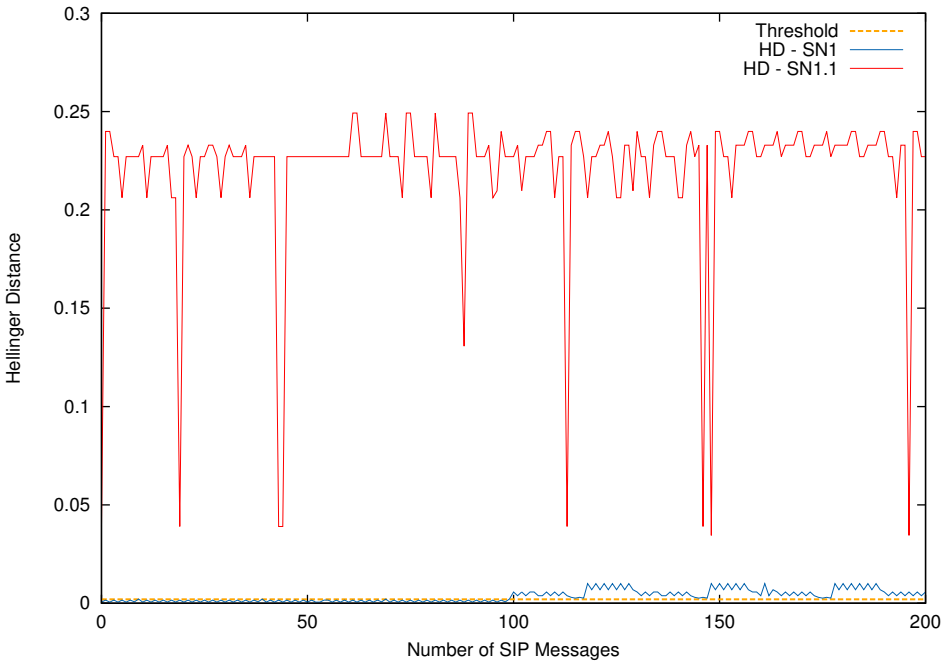

FIGURE 5.13: A fragment of the HD values for scenarios SN-1 and SN-1-1

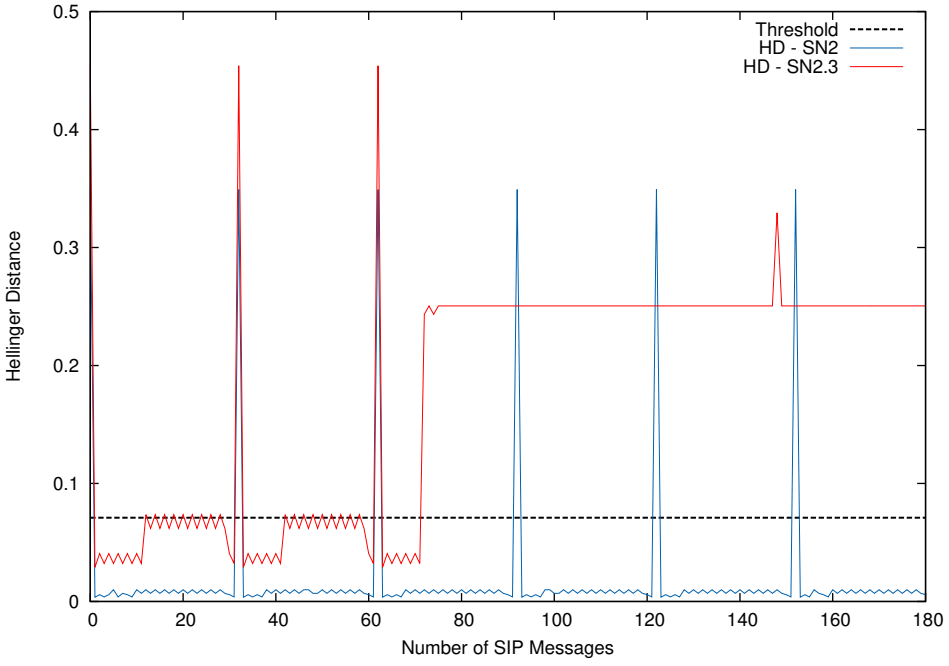

FIGURE 5.14: A fragment of the HD values for scenarios SN-2 and SN-2-3

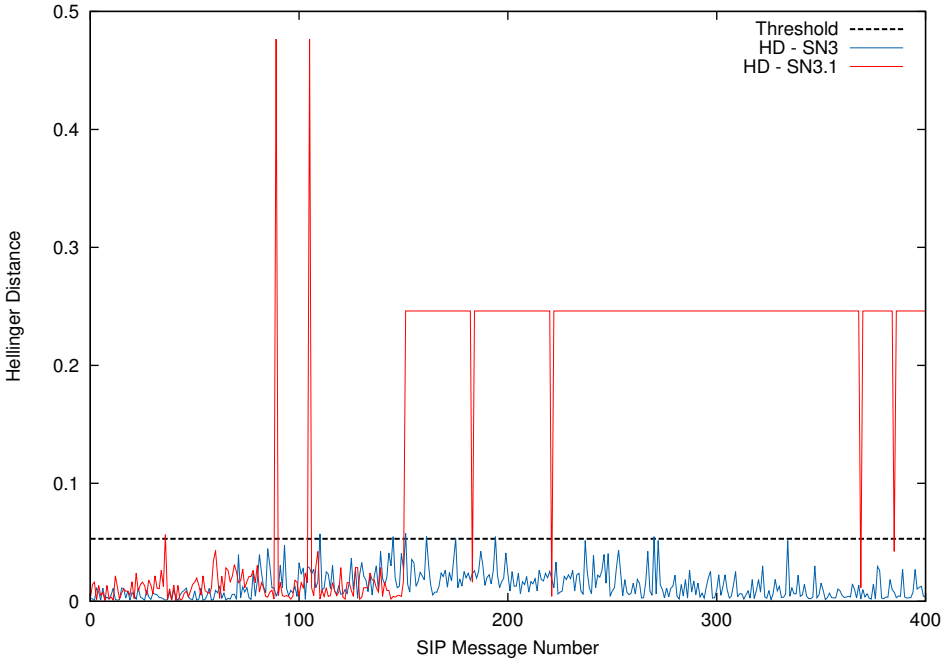

FIGURE 5.15: A fragment of the HD values for scenarios SN-3 and SN-3-1

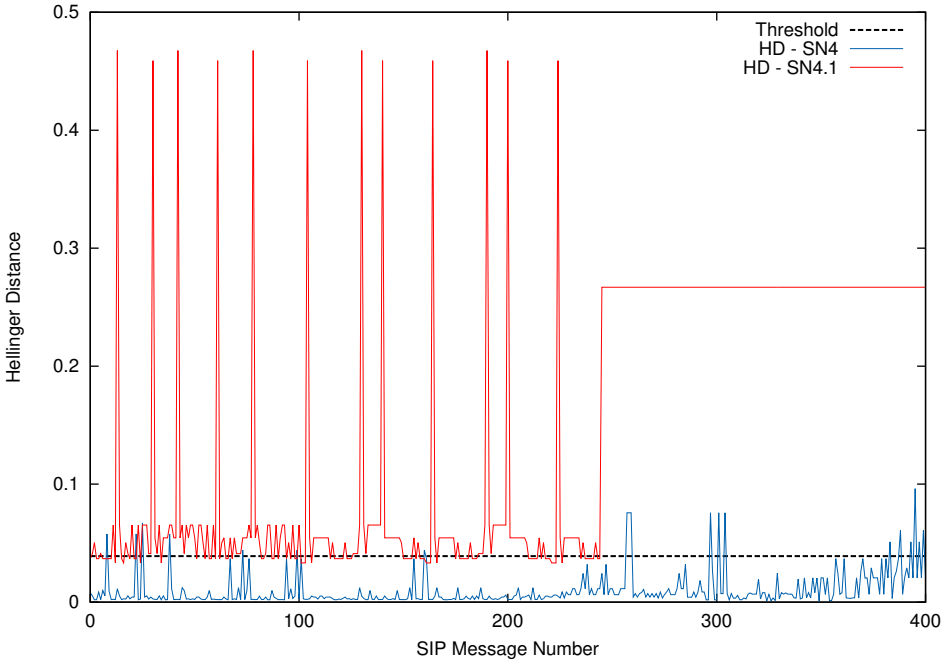

FIGURE 5.16: A fragment of the HD values for scenarios SN-4 and SN-4-1

Chapter 6

Detection of DDoS attacks in SIP using machine learning

6.1 Machine learning techniques

In network intrusion detection, a typical method for exposing attacks is by tracking the network activity for any anomaly. That is, any discrepancy from a previously learned normal profile is identified as suspicious. This procedure is usually done using methods borrowed from the machine learning realm. So far, this potential have been examined in the literature in a great extend. However, as already discussed in Chapter 4 in the case of VoIP in general and SIP in particular, works on this topic are not only scarce but also incomplete. To fill this striking gap, in this Chapter, we try to better assess the power of ML-based techniques to identify DDoS incidents that capitalize on the use of SIP signaling. We consider 5 different popular ML detectors and a plethora of realistically simulated SIP traffic scenarios representing different flavors of DDoS. The results indicate that specific classifiers present high accuracy even in cases of low-rate DoS attacks.

The ML classifiers we touched in the context of this PhD thesis are the following ones: SMO, NaiveBayes, Random Forest, J48, and Neural Networks. Moreover, these classifiers are included in the WEKA tool [213]. We selected these classifiers due to the fact that various researches highlighted their advantages in datasets including numerical data [214], [215].

Based on the classifiers we picked in the context of this PhD thesis, 6 features have been selected for the training and the testing operations. Additionally, we analyzed the audit trails containing the SIP traffic using the software module given in [216]. The analysis has been done with the "ML-train" choice, presented in the corresponding GUI. The analysis of log files resulted in the extraction of several "arff" files, which were finally used for training each one of the selected classifiers.

6.2 ML techniques in audit trail forensic analysis

By examining the rather rich literature on SIP security, one can distinguish several categories of assaults ranging from SQL injection to Denial of Service (DoS) [24, 25, 26, 27]. It can be safely argued that the latter category attracts the greater attention, and seems to be the most perilous and difficult to confront since it is closely related with the signaling nature of the protocol per se. So, focusing on this kind of attacks, so far, several protection and detection methods have been proposed. Roughly, we can categorize them into misuse-detection and anomaly-detection ones. Generally, the first family of methods monitors network activity with exact signatures of known malicious behavior (e.g., observe the network traffic for singular byte sequences), while the second possess a knowledge of normal activity and warns against any deviation from that profile. The latter category of methods, which is the focus of this PhD thesis, is usually realized by means of tools borrowed from the Machine Learning (ML) community. This refers to algorithms that are first get trained in an either supervised or unsupervised manner with reference input to learn its particulars, and then are fed with unknown input for accomplishing the real detection process. Specifically for SIP, although the DoS threat has been stressed out and dealt by a significant number of researches [9, 28], the applicability and effectiveness of ML techniques to cope against such incidents is still being assessed and certainly in need for further development.

Naturally, this is mainly due to the increased overhead that these methods may bear - especially when it comes to realtime detection and a training phase is required - in comparison to misuse-based or purely statistical ones. Nevertheless, in this PhD thesis we argue that ML techniques can be particularly fruitful for examining the high-volume log files of a given VoIP realm in an offline fashion if they contain DoS incidents. Also, this category of methods may show better results when used for the detection of low-rate DoS

(also known with the term "low and slow"), which although is not used to paralyze the target system at a fast pace, it consumes valuable network, CPU and memory resources. Ultimately, this results to service delays which in turn cause customer dissatisfaction with direct negative results to the provider's market share.

Taking the above into consideration, the focus of the PhD thesis at hand is on the applicability of ML techniques to track down DoS incidents, paying special attention to DDoS and low-rate ones.

6.2.1 Classification features

As already mentioned in Chapter 4, to avoid DoS attacks in SIP several solutions have been proposed [32, 30, 34]. Given that this type of attack is as a rule of thumb executed in a distributed manner and may be quite sophisticated regarding its implementation, simple anomaly-detection approaches that rely on the sudden and fast-paced increment of SIP traffic may be not enough. In this regard, ML-powered methods can be a potent ally towards the detection of such perilous events. The key factor here is the log files on the provider side, which can be used to feed a ML classifier in realtime or offline (in case, say, the investigation of an attack aftermath is required). This section elaborates on the use of such techniques in an SIP environment.

In our experiments, we utilize and evaluate the effectiveness of 5 well-known classifiers tested under 15 different attack scenarios. Specifically, we use the SMO, Naive Bayes, Neural Networks, Decision Trees (J48) and Random Forest classifiers. This selection has been made based on the ability of these classifiers to perform better in terms of decision accuracy and speed when it comes to numerical data [217], [214], [215].

In order to take advantage of the aforementioned performance characteristics, we utilize algorithm 8, given in Appendix A.3. Its purpose is twofold. On the one hand, it aims to deal with the sensitive nature of the communication transactions residing in an audit trail by providing an anonymization scheme [34], while on the other allows for automatically extracting the classification features to be used by the classifiers into a numerical form.

The anonymization goal is met using HMAC [218]. HMAC enables one to preserve the anonymity of the communication entities appearing in the underlying audit trail, while the Entropy of messages is preserved leading the subsequent calculations to remain intact.

In fact, revealing the hidden UA identities is as hard as reversing the HMAC procedure itself. The cryptographic key is kept secret and in possession of the entity, who is the legitimate owner of the audit trail. According to the transformation procedure, a log file is examined line-by-line and every privacy-sensitive SIP message header (e.g., <Via>, <From>, <To>, etc) becomes input for the HMAC function (lines 2-4). The algorithm considers only the SIP message headers S1 to S6 as given in figure 6.1. More precisely, the hash function used in our case is the HMAC-SHA256 one combined with a cryptographic key of 256 bits (line 4).

The next stage is to generate the classification features. The steps to achieve this are summarized in lines 5-14 of algorithm 8. The anonymized unique headers are kept in a Hash table data structure (line 5). This table is populated with the number of occurrences of every single header checksum. That is, if a checksum occurs for the first time, then a new instance is generated in the table (lines 8-9). If it is a repeating header, its number of occurrences is increased by 1 (line 6). This procedure is repeated until a certain message window M_w is met (line 11). In our case, the M_w is set to 1,000, but this parameter can be adjusted by the service provider itself, say, according to the average call rates. To our knowledge, there is no foolproof approach to formally define this parameter, mainly because it is eminently contextual. That is, it is closely connected to the characteristics of the service and underlying network. As a result, similar to other anomaly-based approaches, one can follow an error-trial approach to equilibrate between the M_w parameter and the false alarm rate.

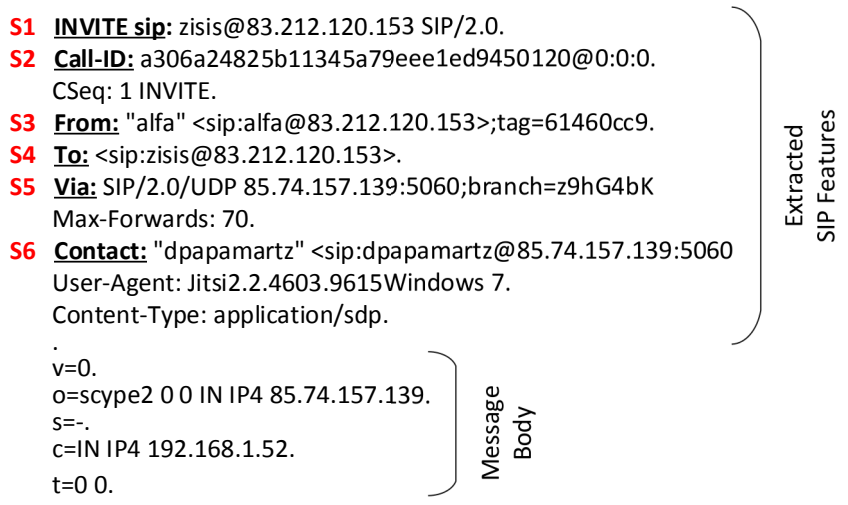

FIGURE 6.1: Symbols s1 to s6 used for ML classification

The result of applying algorithm 8 to an audit trail is a number of specially formatted .arff files (one per M_w), which are afterwards used in the classification process. Each .arff file contains classification vectors, i.e., one vector per SIP message found in the log file being examined. Two instances of such a classification vector follows.

$$V_{attack} = \{926, 4, 988, 4, 4, 3, attack\}$$

$$V_{normal} = \{12, 4, 6, 4, 3, 8, normal\}$$

The first 6 values of each vector represent the occurrences of S1 to S6 SIP headers respectively, and the last one characterizes the class in which the vector belongs. One can easily observe that the first vector introduces a higher number of occurrences in S1 and S3 headers, while the rest remain low, close to those contained in V_{normal}.

6.2.2 Evaluation

In order to evaluate the effectiveness of the aforementioned classifiers in detecting DoS incidents we created a testbed, depicted in figure 6.2. Three different Virtual Machines (VMs) have been used for the SIP proxy, the legitimate users, and the generation of the attack traffic depending on the scenario. All VMs run on an i7 processor 2.2 GHz machine having 6 GB of RAM. For the SIP proxy we employed the widely-known VoIP server Kamailio [207]. We simulated distinct patterns for both legitimate and DoS attack traffic using *sipp v.3.2*[1] and *sipsak*[2] tools respectively. Furthermore, for the simulation of DDoS attack, the SIPp-DD tool has been used [151]. The well-known Weka tool [213] has been employed for ML analysis.

As already pointed out in section 6.2.1, we assessed 5 classifiers under 15 different scenarios the results of which is provided in Table 6.2. It is stressed that both the training and testing scenarios include legitimate and attack traffic. For example, the training scenario is SN-1 and its testing scenarios are SN-1-1, SN-1-2, SN-1-3, and so on. The legitimate traffic for DoS testing scenarios was created using the same call rate as that of the corresponding training scenario. On the other hand, for DDoS we used a range of different call rates aiming to better simulate the possible variations that may appear in a real VoIP service. For example, as observed in Table 6.1, the call rate for SN-6-1

[1] http://sipp.sourceforge.net/
[2] http://sipsak.org/

is given as 20-120, where the first number indicates the call rate of the attack, and the second corresponds to the call rate of the legitimate traffic both occurring in parallel. Keep in mind that for DDoS scenarios about half of the registered users were generating the normal traffic, while the other half were launching the actual attack. Moreover, for all the scenarios, we employed an exponential inter-arrival time distribution ($\lambda = 100$), for producing the legitimate traffic similar to that used in evaluating SIP server performance [209]. The attack traffic for DoS training scenarios was created using randomly generated attacks with call rates varied between 20 to 10,000 calls/sec and time pauses between them spanning from 15 to 360 secs. The same method was used for creating the DDoS training scenarios that is, seven variants were launched in total, having different call rates spanning between 2,000 to 15,000 calls/sec and pauses between them set to 10 to 800 secs.

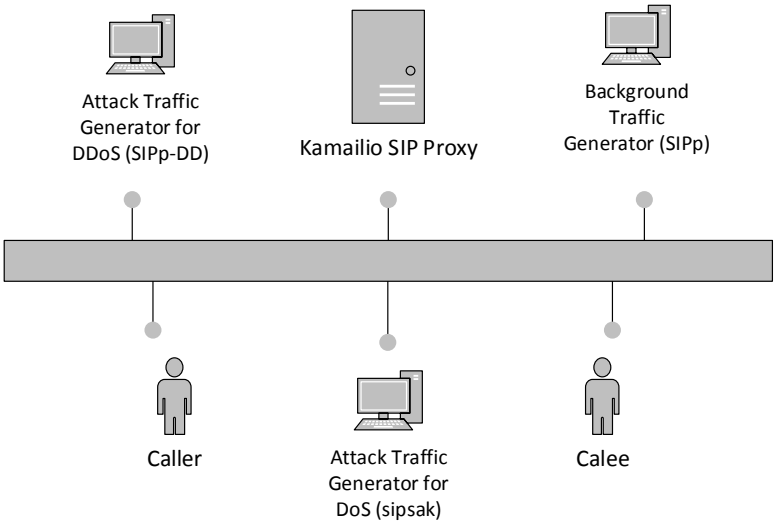

FIGURE 6.2: Offline ML detection: Deployed testbed for DDoS simulations

6.2.3 Results

The obtained results for all the scenarios are given in Table 6.2. This section firstly refers to the DoS attack scenarios and then to DDoS ones. As shown in Table 6.2, we use legacy intrusion detection metrics, namely False Positive (FP) and False Negative (FN) to assess the performance of each algorithm. One can easily observe that in the case of DoS involving scenarios SN-1-1 to SN-5-2, the maximum FP value is equal to 3.7%, scored by both SMO and Neural Networks detectors. For the same scenarios, the FN metric remains low, presenting an average value of 0.02% and a maximum one of

TABLE 6.1: ML techniques: Description of scenarios

Scen.	Num.of Users	Calls/Sec.	Train Scen.	Type of Attack
SN-1	30	2	✓	DoS
SN-1-1	30	50	-	DoS
SN-1-2	30	175	-	DoS
SN-1-3	30	350	-	DoS
SN-2	30	5	✓	DoS
SN-2-1	30	20	-	DoS
SN-2-2	30	40	-	DoS
SN-2-3	30	80	-	DoS
SN-3	30	20	✓	DoS
SN-3-1	30	266	-	DoS
SN-4	30	120	✓	DoS
SN-4-1	30	800	-	DoS
SN-5	50	120	✓	DoS
SN-5-1	50	400	-	DoS
SN-5-2	50	1200	-	DoS
SN-6	60	20	✓	DDoS
SN-6-1	60	20-120	-	low-rate DDoS
SN-6-2	60	120-20	-	high-rate DDoS
SN-7	500	100	✓	DDoS
SN-7-1	500	10-200	-	low-rate DDoS
SN-7-2	500	100-40	-	high-rate DDoS
SN-7-3	500	30-50	-	low-rate DDoS

0.85%. Generally, the best results in the DoS case are obtained by J48 and Random Forest classifiers. The results also indicate that as the attack traffic volume increases the FP and FN rates decrease. For instance, taking SN-3-1 and SN-4-1 as an example, the FP metric decreases significantly when compared to the first three subscenarios, namely SN-1-1 to SN-1-3.

On the downside, the false alarms per classifier augment for scenarios SN-6-1 to SN-7-3 representing a DDoS case. This is rather expected as the occurrences per message header decrease significantly due to the multiple spoofed IPs - that in turn affect headers S3, S5 and S6 of virtually every transmitted SIP message - thus leading to a more difficult separation between the attack and normal messages.

Among all the classifiers the worst results for DDoS scenarios in terms of FP are obtained by SMO and Naive Bayes. Note that FP percentage rates scored in DDoS scenarios for all the algorithms are generally considerably higher than those obtained by the corresponding DoS ones. Taking SN-6-1 for example, FP fluctuates between 0.04% and 17.7%,

having an average value of 6.86%. Similar results are recorded for SN-7-1, with FP varying between 5.2% and 11.3%. When the attack traffic increases, i.e., when the high-rate DDoS scenarios are involved, all the results are improved significantly. This is because the portion of the attack messages inside the same M_w increases proportionally to the rate of the attack. For instance, for scenario SN-6-2, the maximum FP value is rather negligible, equal to 0.55%, while FN is zeroed. Similar results are obtained in the case of the other high-rate DDoS scenario, namely SN7.2, demonstrating a maximum FP value equal to 1%. Finally, SN-7-3 corresponds to a moderate attack rate scenario and presents similar results to the four previously mentioned ones.

Specifically for DDoS scenarios, we compare the results scored by ML detectors against those obtained for the same scenarios but with two other anomaly-detection methods, namely Entropy [200] and Hellinger Distance [36]. More specifically, we compare the results obtained from the ML-driven detection, to those obtained from previous researches [34], [128], employing the previously-mentioned statistical methods. Table 6.3 summarizes the FP and FN results obtained by the two aforementioned schemes. To help the reader compare between the various algorithms, the rightmost columns of the same table contain the corresponding false alarm values as scored by the top ML-based performer. Bear in mind that in contrast to ML techniques the training scenarios (SN-6 and SN-7) used for Entropy and Hellinger Distance do not include attack traffic. This is sensible because non-machine learning approaches rely on deviations between the legitimate messages in order to compute the corresponding thresholds. If an examined message exceeds the predefined threshold, then the message is classified as abnormal.

We can safely argue that the non-machine learning schemes score worse results in comparison with ML-based ones. More precisely, in the case of Entropy metric and for all the five DDoS scenarios, the FP rate reaches the maximum value of 18.1%, while FN varies between 5.41% and 43.5% (and especially for the Entropy metric scores exceedingly high values for all the scenarios but one). Further, the FP for Hellinger Distance fluctuates between 1.8% to 36%. The maximum FN value for the two aforementioned methods is the same, equal to 43.1% perceived in both cases for scenario SN-6-2.

To sum up, the results obtained from Table 6.3 imply that ML-based detectors outperform the non-machine learning ones especially in terms of FN, for all DDoS incidents. In fact, the same category of detectors are overall competitive, presenting high accuracy in

DoS scenarios as well. This is because these schemes learn from a mixed traffic including both normal and attack messages, and thus it is easier for them to separate between the two classes, even with slight differences in header occurrences.

In general, anomaly-detection schemes must cope with a number of issues [219]: (i) A considerable number of false alarms (especially false positives) is normally expected by most classifiers. In our case, this statement seems to be confirmed in its entirety for the Entropy and Hellinger Distance metrics. For the ML ones, we can assert that the same statement stands half-true for FP, and false for FN. Specifically, ML-based detection largely fails in the case of low-rate DDoS (except for Neural Networks, and partially for Decision Trees), but it is effective across all algorithms for high-rate DDoS. This however hardly comes as a surprise as low-rate attacks are generally much harder to detect. (ii) Acquiring attack-free data for training may be a problem. In our case, this point can be dealt with if a VoIP billing system is in place. This will allow the correct labeling of each message because these logs are supposed to be accurate and valid. (iii) Smart aggressors may try to elude detection by increasingly teaching a system to identify intrusive activity as legitimate. To tackle this third point, one can vary the M_w based on mid or long-term statistical observations regarding SIP traffic.

A last point to be emphasized is that in terms of complexity ML-based classifiers require a different and usually significant amount of time to build a model from the given training set. Note that this time does not include that needed to generate an .arff data file from the given log file. For instance, taking SN-4 as an example the training process spans between 0.01 to 154.95 secs for all the classifiers when fed with a file containing 261k records of SIP messages.

6.3 Realtime DDoS detection

In this section we assess the potential of using techniques borrowed from the ML realm to detect DDoS incidents in SIP services. In contrast to the solution presented in section 6.2, the evaluation of the various classifiers is done in realtime in the SIP proxy. Our experiments involve 5 different well-known classifiers and a large variety of attack scenarios ranging from simple DoS to slow and high-rate DDoS. The evaluation is done in terms of both detection accuracy and processing time.

6.3.1 Detection engine

This section elaborates on the architecture of the proposed IDS, which as observed from figure 6.3, is composed of 3 modules. The first module is occupied with the extraction of the required classification features from the headers of the incoming SIP messages. The selected features are forwarded to an anonymisation module, and finally are fed to the classification engine. After a training phase, the latter module can be configured to operate in realtime using an ML algorithm of choice. The subsequent sections elaborate on each of the aforementioned IDS modules.

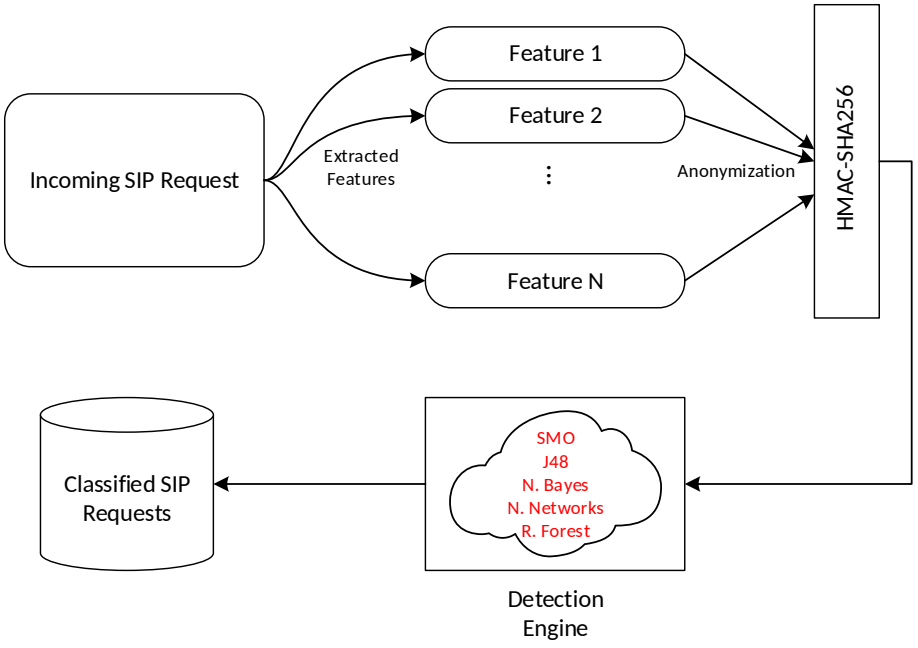

FIGURE 6.3: Detection engine architecture

6.3.2 Feature exctraction and anonymization of data

As already mentioned, the feature extraction module operating on the SIP server examines the incoming SIP traffic for any request, say, INVITE, REGISTER. Next, the request is parsed to isolate the headers of interest, namely, Request-Line, <Via>, <From>, <To>, <Call-ID>, and <Contact>. The extracted headers are anomymized with the help of HMAC-SHA256. The latter aims at preserving end-user privacy in cases where the detection task is outsourced to a third party. Also, it has the dual benefit of preserving the Entropy of the original headers and making deanonymization as hard as reversing

the HMAC-SHA256. The anonymized features along with its frequency of appearance are stored in a hash table data structure.

As described in Appendix A.3, algorithm 9, every time a new (different) feature is extracted from an incoming SIP request a new record is inserted in the hash table with its corresponding value equal to 1. In case an already existing key is inserted in the table the corresponding frequency value is increased by 1. This procedure continues until the number of messages reaches a certain predefined Message Window M_w. For the needs of our experiments we picked arbitrarily a M_w equal to 1,000. Therefore, the detection engine starts the classification process from message 1001. That is, the hash values of the headers of this message will update the corresponding cells of the hash table and the resulting numbers will be fed to the classifier. This process is done in a message-by-message manner for every SIP request arriving after the 1000th.

Obviously, the value of M_w parameter is sure to affect the detection accuracy. Naturally, this parameter can be adjusted by the service provider itself, say, according to the average call rates. Nevertheless, to our knowledge, there is no foolproof approach to formally define this parameter, mainly because it is eminently contextual. That is, it is closely connected to the local characteristics of the service and underlying network. As a result, similar to other anomaly-based approaches, one can follow an error-trial approach to equilibrate between the M_w parameter and the false alarm rate.

6.3.3 Training and operation

Similar to any other ML-powered approach, the detection engine requires the training of the classifier in order to be able to detect anomalies in the incoming traffic. This means that during service initialization, the classifier of choice must be trained based on some pregenerated training data. The creation of such a training set is up to the provider because it mainly depends on the particular services it offers and the characteristics of the underlying network. Also, a renewal and/or amendment of the training set is required as soon the network and/or service operating conditions change. In this PhD thesis, we opt not to address the two aforementioned issues which are left for future work.

Nevertheless, it should be stressed that the training data must contain both legitimate and attack traffic. This is necessary because in order for the classifier to build a realistic

traffic model, the training data must contain classes of both attack and normal traffic. As soon as the training phase is completed, the detection engine starts on the SIP server as a realtime service.

6.3.4 Implementation

The realtime detection service has been implemented as a plug-in module of the well-known SIP proxy Kamailio [207]. Specifically, the module [216] is written in C programming language and is capable of processing any incoming SIP request as described in the previous subsection.

The feature extraction module stores temporarily the processed data to a hash table, while classification relies on Weka [213] a well-known framework for ML analysis. Given that Weka provides a Java interface, we use Java Native Interface (JNI) [220] to make possible the integration between the feature extraction and classification module. In this way, one can easily configure the employment of the appropriate classifier depending on the requirements at hand.

The implementation has been tested for possible memory leaks following an error-trial approach and monitoring the Linux OS memory consumption under various scenarios. This is because JNI implementation should be protected against potential out-of-memory conditions [221]. As already pointed out, our implementation is freely available [216] for further development and experimentation.

6.3.5 Results

This section details on the testbed used and presents the results in terms of both false alarms and processing overhead on the SIP server.

6.3.6 Testbed Setup

As illustrated in figure 6.4, we employed a virtualised testbed running over an i7 processor at 2.2 GHz. Three Virtual Machines (VMs) were created, each one equiped with 6 GB of RAM. These VMs respectively host the SIP proxy, the UAs, and the attack traffic

generator. We created distinct patterns of legitimate, single source DoS and DDoS traffic using *sipp v.3.2* [222], *sipsak* [223], and SIPp-DD [151] tools respectively.

As observed from Table 6.1, twenty two disparate scenarios were created in total to replicate different DDoS incidents. Seven basic scenarios were used for training, while the others represent an attack incident. For all the scenarios, an exponential inter-arrival time distribution ($\lambda = 100$) is followed to produce the legitimate traffic. Note that this kind of distribution inherently presents the "lack-of-memory" property. Specifically, this property considers that the probability of a future event (call arrival) is the same regardless of the previous events that took place in a series of time frames. In our case this is analogous to the traffic used for assessing VoIP systems' performance [209].

Furthermore, a range of different call rates has been used in the cases of DoS and DDoS, with the aim to simulate various call rates which may approximate the traffic patterns of a real VoIP provider. For example, as observed in Table 6.1, the call rate for SN-6-1 is given as 20-120, where the first number indicates the call rate of the attack, and the second designates the call rate of the legitimate traffic both occurring in parallel. Keep in mind that for DDoS scenarios about half of the registered users were generating the normal traffic, while the other half were acting maliciously.

6.3.7 Detection accuracy

In the context of our experiments, we employed 5 well-known classifiers, namely SMO, Naive Bayes, Neural Networks, Decision Trees (J48), and Random Forest. This particular choice has been done because these classifiers present a better detection accuracy when it comes to numerical data [224]. The detection accuracy of each classifier in terms of False Positives (FP) and False Negatives (FN) is estimated and the results are included in Table 6.4. Also, to ease the reading of the table, the mean, minimum, and maximum FN values per classifier and collectively per attack type (DoS/DDoS) are depicted in figures 6.5 and 6.6. From these figures it can be seen that the FN metric fluctuates between 0.9% and 23.7% for DoS scenarios having an average of 14%, while the corresponding values for DDoS are 2%, 62%, and 16%.

Focusing on Table 6.4 and on its lines containing the average values of FN metric, one can conclude that SMO produces the worst results and therefore should be avoided.

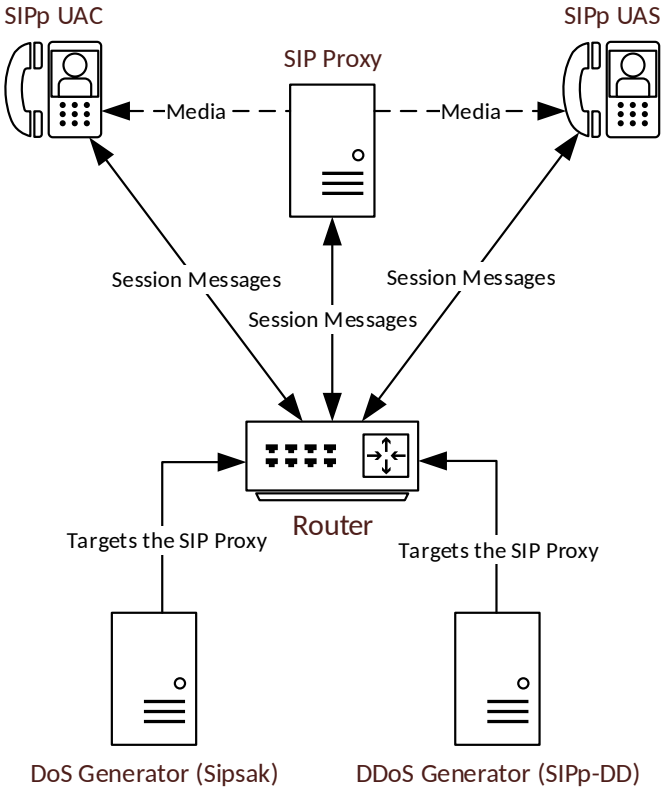

FIGURE 6.4: Realtime ML detection: Deployed testbed for DDoS simulations

The same observation applies to Random Forest but only for DoS scenarios. J48 on the other hand scores an average FN of 11.8% but largely fails in terms of FP (interestingly, the FP metric is almost non-existent for all the algorithms but J48). It can be safely argued that the most reliable classifier across all scenarios and for both FN and FP metrics seems to be Naive Bayes followed by Random Forest. In any case, the FN percentages scored by both the aforementioned algorithms especially for DoS scenarios are not favorable for any real-world IDS. Putting aside SMO and Neural Networks, the rest of the classifiers produce an average FN lesser than 8.5% for DDoS scenarios only. It can be therefore estimated that ML-driven detection shows greater potential in detecting more sophisticated attacks of this kind.

Having a complete view of the results, we consider that further experimentation is needed to obtain a better approximation of the power of ML-driven DDoS detection in SIP realms. In this direction, a future work could concentrate on testing more classifiers and tuning the Mw parameter based on the specific needs of the VoIP service provider.

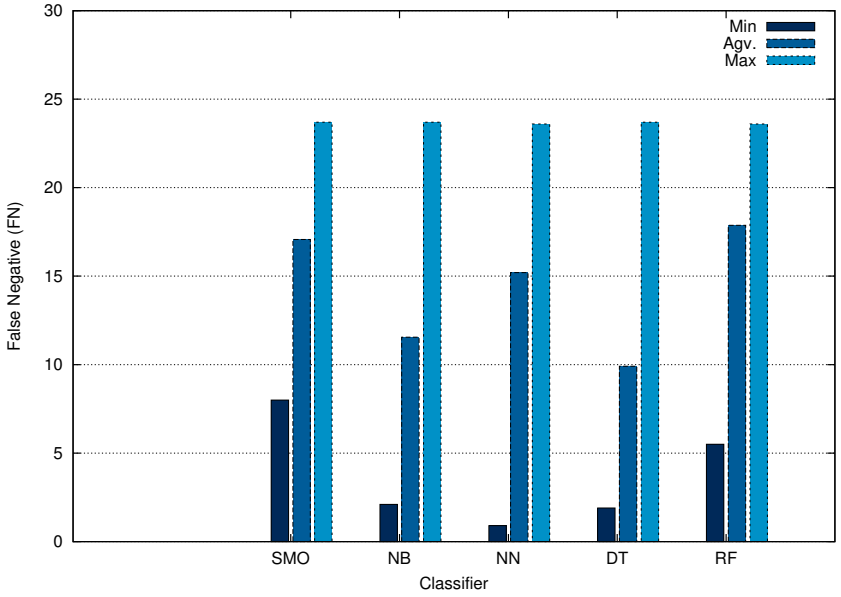

FIGURE 6.5: Dos minimum, maximum and average FN percentage values per classifier

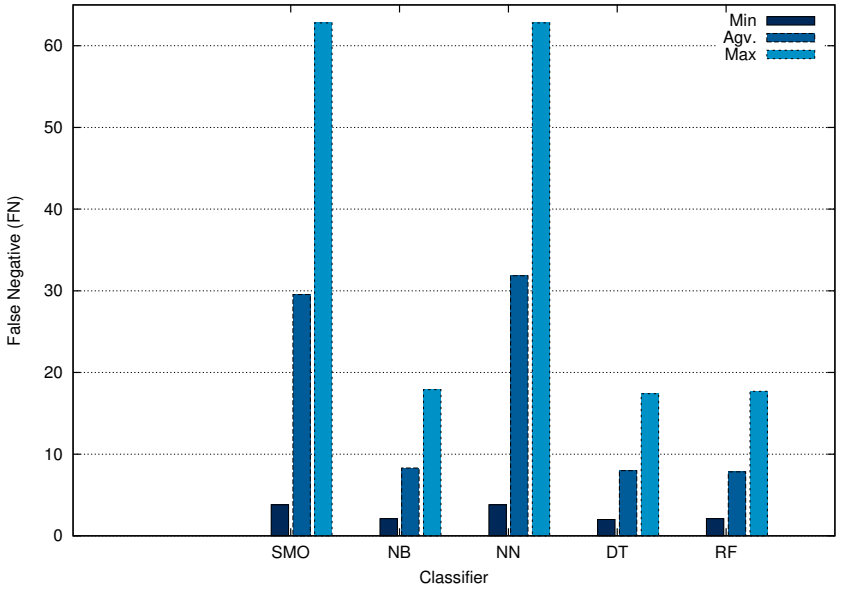

FIGURE 6.6: DDoS minimum, maximum and average FN percentage values per classifier

6.3.8 Performance

Putting aside its effectiveness, the other decisive factor for any IDS is that of performance in terms of service time. This section elaborates on the per message (SIP request) processing time introduced by the realtime detection engine on the SIP server. It is to

be noted that all the time measurements included in this section correspond to a worst case scenario as the SIP proxy for all the tests was configured with one serving thread.

Figures 6.7 and 6.8 illustrate a random snapshot of the processing time introduced by our architecture, while Table 6.5 includes the average, max, min, and standard deviation processing times per classifier for all the 15 attack scenarios. As observed from the table, the average processing time remain under 4ms, while it is easily seen that all classifiers follow a similar tendency. As the SIP proxy is configured in single-thread mode it can be safely argued that the induced overhead is negligible. The maximum values of, say, 300 ms contained in Table 6.5, are unique or very scarce and are due to the activation of single-thread mode at SIP proxy side and the traffic pattern of the examined scenario. For instance, while in SN-1-2 the normal traffic rate is 2 calls/sec, in SN-7-1 is 200 calls/sec resulting to an increment in the classifiers, average processing time.

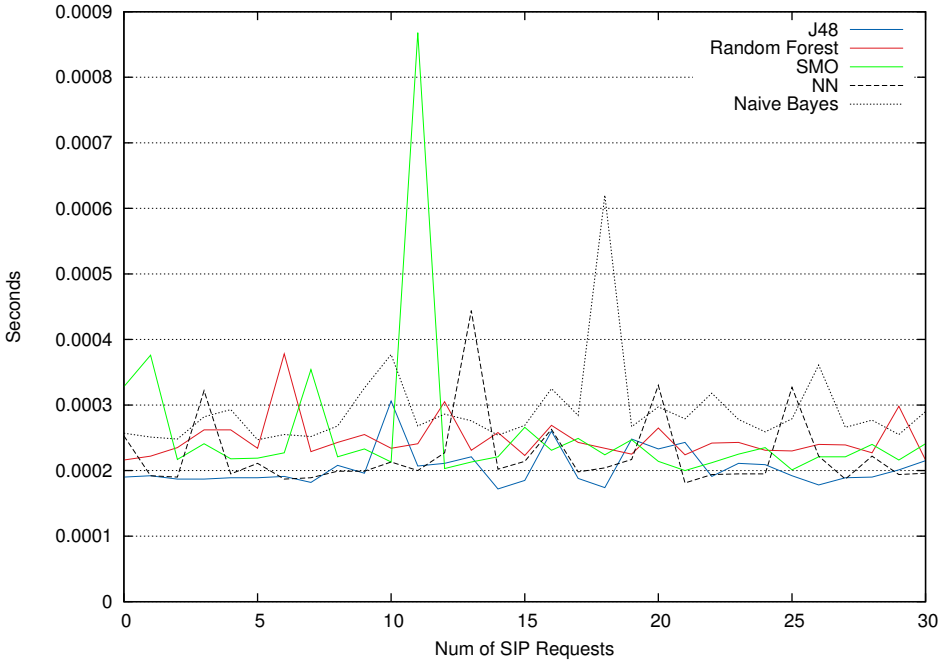

FIGURE 6.7: A random snapshot of the time overhead for scenario SN-1-2

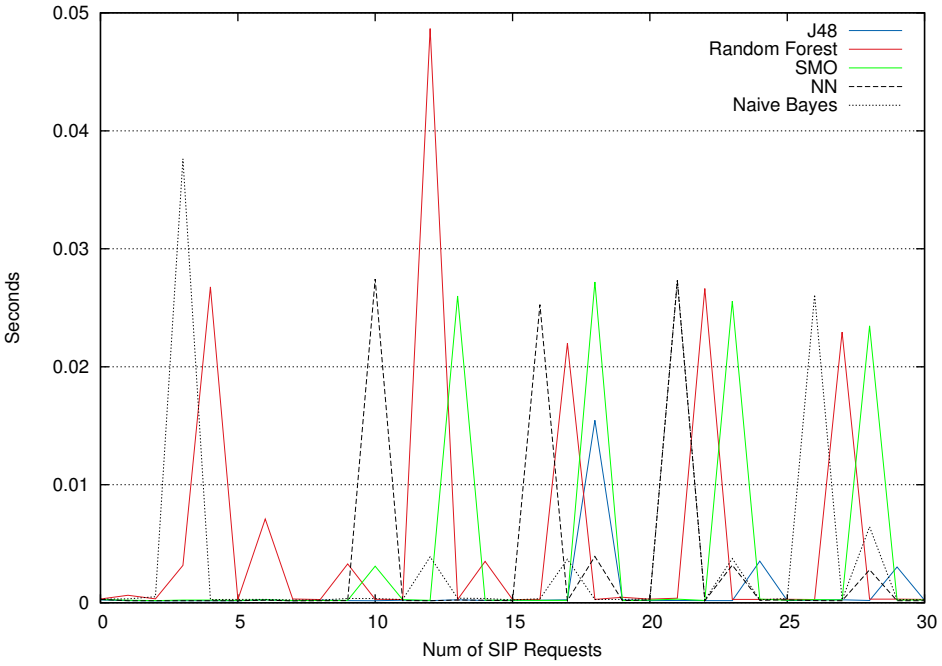

FIGURE 6.8: A random snapshot of the time overhead for scenario SN-7-1

Table 6.2: Offline ML detection: Summary of results for all the scenarios (The best performer per scenario in terms of FP is in bold).

SN	**Traffic (Calls)**		**SMO**		**Naive Bayes**		**Neural Networks**		**Decision Trees (J48)**		**Random Forest**	
	Total Rec.	**Attack Rec.**	FP	FN	FP	FN	FP	FN	FP	FN	FP	FN
			%	%	%	%	%	%	%	%	%	%
SN-1-1	11.3k	9.7k	2.1	0	0.3	0	2	0	**0**	0	**0**	0
SN-1-2	14k	12.3k	1.8	0	0.15	0	1.7	0	**0**	0	**0**	0
SN-1-3	15.4k	11.3k	3.7	0	0.24	0	3.7	0	**0**	0	**0**	0
SN-2-1	12k	7.9k	0.01	0	0.25	0	**0**	0	**0**	0	**0**	0
SN-2-2	13k	9.2k	0.06	0	0.28	0	**0**	0	**0**	0	**0**	0
SN-2-3	24.5k	22.8k	**0**	0	0.11	0	**0**	0	**0**	0	**0**	0
SN-3-1	667k	568k	0.09	0.04	0.02	0.85	0.3	0.01	**0**	0.01	0.04	0.01
SN-4-1	178k	168k	**0**	0.01	**0.01**	0.01	**0**	0	**0**	0	0.02	0
SN-5-1	262k	200k	0.02	0.05	0.05	0.08	0.4	0.01	0.01	0.01	0.06	0.01
SN-5-2	667k	611k	0.02	0.01	0.09	0.01	0.28	0.01	0.03	0.01	0.08	0.01
SN-6-1	175k	23k	17.7	0.01	11.2	0	0.04	0	2.4	0	3	0
SN-6-2	114k	50k	0.18	0	0.55	0	0.01	0	**0**	0	**0**	0
SN-7-1	203k	11k	10.4	0	11.3	0	5.2	0	7.3	0	5.2	0
SN-7-2	144k	50k	0.51	0	1	0	0.25	0	0.27	0	0.25	0
SN-7-3	128k	33k	0.78	0	0.91	0	0.25	0	0.31	0	0.24	0

TABLE 6.3: Summary of evaluation metrics for Statistical Schemes in DDoS scenarios ($M_w = 1,000$).

SN	**Low-rate**	**Entropy**		**Hellinger Distance**		**ML Techniques (Top performer)**	
		FP	FN	FP	FN	FP	FN
		%	%	%	%	%	%
SN-6-1	✓	**0**	13.3	36	0.01	0.04	0
SN-6-2		0.97	43.5	1.8	0	**0**	0
SN-7-1	✓	**4.4**	5.41	8	5.41	5.2	0
SN-7-2		18.1	34.5	3.38	0	**0.25**	0
SN-7-3	✓	**0**	25.7	2.49	5.45	0.24	0

Table 6.4: Realtime ML detection: Summary of results for all the scenarios (The best performer per scenario in terms of FN is in bold).

SN	Traffic (Calls)		SMO		Naive Bayes		Neural Networks		Decision Trees (J48)		Random Forest	
	Total Rec.	Attack Rec.	FP %	FN %	FP %	FN %	FP %	FN %	FP %	FN %	FP %	FN %
SN-1-1	2.4k	1.2k	0	23.7	0	23.7	0	**23.6**	35.1	23.7	0	**23.6**
SN-1-2	2.5k	1.3k	0	22.9	0	**22.8**	0	22.9	33.7	**22.8**	0	23
SN-1-3	2.7k	1.5k	0	**21.8**	0	21.8	0	**21.8**	32	22	0	**21.8**
SN-2-1	2.7k	1.2k	0	21	0	7.3	0	14	36.9	**6.3**	36.3	19.6
SN-2-2	2.7k	1.2k	0	16.1	0	7.3	0	28	29.9	**7.1**	0	9.9
SN-2-3	3k	1.3k	0	12.4	0	**5.5**	0	24.8	44.1	14.7	0	27.3
SN-3-1	5k	2.3k	0	19.8	0	**6.9**	0	9.1	47.1	7.2	0	20
SN-4-1	8k	1.5k	0	8	0	8.9	0	4.7	75.1	**1.9**	0	15.7
SN-5-1	7k	1k	0	11.7	0	9.2	0	**0.9**	0	11.2	0	5.5
SN-5-2	9k	2.6k	0	12.3	0	2.1	0	2.2	59.9	**2**	0	12.3
Avg.	—	—	0	16.97	0	11.55	0	15.2	39.38	11.8	3.63	17.8
SN-6-1	12k	6.6k	0	3.8	0	3.8	0	3.8	0	**3.4**	0	3.7
SN-6-2	4.5k	2k	0	18.2	0	17.9	13.7	28.5	0	**17.4**	0	17.7
SN-7-1	10.5k	3.3k	0	31.1	0	2.1	0	31.2	0	**2**	0	2.1
SN-7-2	7k	2.2k	0	32	0	10.7	0	32.8	0	**10.5**	0	10.6
SN-7-3	9k	6k	0	62.6	0	7	0	62.8	0	6.6	0	**5.1**
Avg.	—	—	0	29.5	0	8.3	2.74	31.8	0	7.98	0	7.84

TABLE 6.5: Summary of classification time overhead for all the scenarios (msec)

Classifier	**Min.**	**Max.**	**Avg.**	**St. Dev.**
SMO	0.08	123.78	3.37	7.73
Naive Bayes	0.15	171.13	3.48	7.76
Neural Networks	0.10	129.20	3.42	7.74
Decision Trees (J48)	0.08	388.01	3.28	7.91
Random Forest	0.08	91.74	3.56	7.72

Chapter 7

Misuse-based detection of attacks against SDP

7.1 Introduction

During the last decade, VoIP services have exhibited a remarkable expansion. As a matter of fact, recent reports [2] indicate that IP multimedia communication services gain ground against the Public Switched Telephone Network (PSTN) ones. This is because VoIP services provide more flexible and inexpensive models, and thus they gradually dominate the market. Among others, multimedia session establishment and management constitutes a fundamental operation in VoIP networks. Nowadays, Session Initiation Protocol (SIP) has been adopted as the prevalent signaling protocol for handling multimedia sessions over the Internet and 3rd Generation partnership Project (3GPP) realms. On

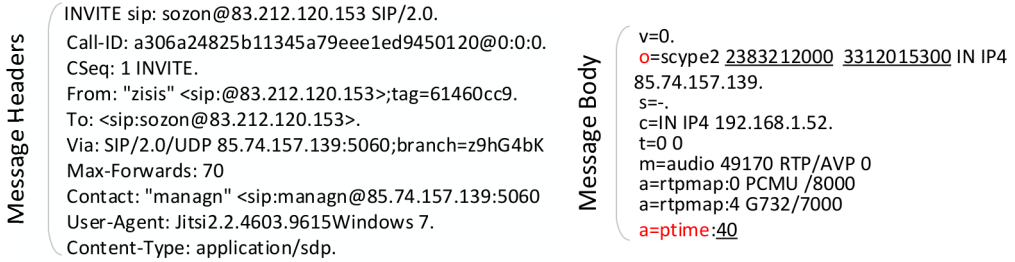

FIGURE 7.1: A typical SIP INVITE request. The SDP segment has been placed in the right side of the figure

the downside, SIP is inherently susceptible to different kinds of attacks [11, 72]. One of them lies in its exploitation as a covert channel. Adversaries usually employ covert channels aiming to communicate information over legitimate data flows. In fact, the text-based nature of SIP fosters such types of attacks. An adversary could easily craft specific parts of the message in order to deliver data with special meaning over legitimate SIP requests. The only requirement for such an attack would be to conform to SIP syntax, otherwise the message parser module at the receiver side would possibly drop the request as malformed. In any case, the exploitation of fields which do not fully comply with the RFC [1] is possible in the context of a C&C. Using this approach an aggressor is in position to create an army of noiseless devices (bots). This case is further discussed in section 7.3.2.

On the downside, as reported in various research works [48, 225, 9] and software flaw databases[1], SIP is well-known to be susceptible to a plethora of attacks ranging from Denial of Service (DoS), SQL injection, and signalling manipulation. In a typical DoS assault, the attacker tries to paralyze the victim by either sending against it a surge of SIP requests or a number of malformed messages. In the former case, the victim is unable to serve the voluminous number of incoming requests, while in the latter the sufferer is incapable of parsing or handling properly the incoming request, and the service crashes.

So far, SIP-based covert channels are scarcely addressed in the literature, and to our knowledge, no implementation exists. That is, the majority of the existing works concentrates on the applicability of information hiding techniques in VoIP-related protocols in general. This includes SIP, Real Time Protocol (RTP) and RTP Control Protocol (RTCP). The delivered channels may be used in a variety of ways, aiming to establish secret paths of communication. In this PhD thesis, we examine the feasibility of exploiting SIP as a Command and Control (C&C) channel aiming to deliver commands to a SIP botnet and launch attacks.

Bearing the above in mind, the goal of the work at hand is dual; first to provide proofs of the pernicious nature of these types of attacks, on real-life SIP clients and servers, and second to introduce a lightweight and flexible filtering mechanism for effectively copying with them. Our defensive solution comes in the form of a SDP parser software module either embedded in the SIP server or running in a separate machine in front of the former.

[1] Common Vulnerabilities and Exposures https://cve.mitre.org

In this way, the defender is able to timely detect and silently drop messages that do not fully comply with the standard [1]. Also, as a side advantage, the parser is capable of rejecting SIP requests that are found suspicious to carry information that may be part of a covert communication channel.

We evaluated our solution in terms of service time overhead using a custom made architecture, in which we simulate several scenarios involving a mix of normal and attack traffic. The results indicate that the proposed mechanism introduces negligible overhead to the processing of incoming and outgoing SIP messages. To the best of our knowledge, this is the first work that specifically focuses on SDP message manipulations. As already mentioned, this is in contrary to other works in the literature [25, 226], which solely deal with deliberate malicious manipulations in SIP message headers. The main contributions of this PhD thesis with respect to SDP-driven attacks are summarized as follows:

- We present a simple but powerful in terms of stealthiness covert communication protocol to exchange botnet C&C messages over SDP data in SIP requests.
- We evaluate the effectiveness of the covert channel by controlling several bots and launching two different Denial of Service (DoS) type of attacks.
- An assessment of the attack impact in terms of resource consumption at the victim side is also included.
- We study the impact of SDP malformed messages on a variety of SIP software and hardware phones, and servers.
- We offer a publicly available open-source software module capable of detecting malformed SDP messages lurking in SIP requests. Message parsing is done based on RFC 4566 [1], while the implemented software can work alongside the SIP server or independently in a separate machine.
- We extensively assess the performance of the proposed scheme in terms of service time.

7.2 SDP C&C channels

A SIP message comprises of several headers and a message body. It is text-based and presents similar structure to that of HTTP. Figure 7.1 depicts a typical SIP INVITE request. As observed from the figure, the various headers contain information related to the sender and the recipient of the message, and also the communication path. Also, as seen in the figure, such a message is comprised of two parts; the left one containing the various headers, and the message body describing streaming media initialization parameters. The latter part is built following the SDP standard format [1]. Given the text nature of the message an adversary could straightforwardly manipulate the data contained in the SIP headers or SDP descriptors with the aim to build a covert channel over the legitimate information. Note that if this is done in a SIP-oriented (natural) way, the channel has many changes of going undetected. However, as explained further down, care must be taken in order not to alter important information that are required by the peers or the proxies to establish communication. Moreover, any manipulation in the various headers or parameters must be syntactically neutral; otherwise, the message could be dropped by the receiver's message parser. One may also think of encrypting the parts of the message to be used as the covert channel carrier. This however would require the provision of some key management process, and more importantly, will attract the attention of network defenses. So, the idea here is to hide the (C&C in our case) information in plain sight by simply mimicking the values contained in the fields of a normal SIP message.

To create a SIP-based covert channel one needs to choose specific parts of the message and use them as data carriers. In fact, several SIP headers or SDP descriptors contained in, say, a SIP request can be used to bear information with special meaning to the communicating parties. In any case, the selection must fulfil the next two requirements. On the one hand, it must be syntactically correct, otherwise the message will be most likely dropped by the parsing process. On the other, it must preserve the communication information at least regarding to the sender and the SIP proxy. Otherwise, the message may never be delivered correctly.

In this work, we concentrate on fields contained in the message body of a SIP request where the literature seems to be quite incomplete. As already pointed out and depicted in the right part of figure 7.1, this part of the message follows the SDP data format.

Precisely, these pieces of data contain information related to the media parameters of a session and are comprized of 5 mandatory and 15 optional fields [1]. We make use of only two descriptors namely as <*o*> and <*a=ptime*>. The first one is mandatory while the second is optional. The <*o*> descriptor carries information in regards to the session originator and it is composed of 5 fields. Among them, the first and the last one point out the username and the IP address of the caller ("skype2" in figure 7.1), while the second and the third indicate a unique session id and the session version. The fourth field is a text string bearing the type of the network ("IN" (Internet) in the normal case). The creation of session id and version fields are up to the creating tool. The RFC [1] suggests that both these parameters must receive numerical string values of at least 10 digits each created based on a Network Time Protocol (NTP) [227] format timestamp in order to ensure uniqueness. Also, RFC states that the <*a=ptime*> descriptor bears the length of time in msecs represented by the media in a packet. So, for example, any decimal value representing time in msecs is considered normal. The selected fields are shown in red in figure 7.1. The interested reader who wishes to get a deeper understanding of SDP can refer to the corresponding RFC [1].

For exploiting the above mentioned fields aiming to deliver a covert channel over legitimate SIP messages one has to set specific values. Table 7.2 summarizes these values in the context of this work. As observed from the table, the protocol relies on three simple commands related to the type, the parameters, and the execution and termination of an attack. That is, the *a=ptime:*<*packet time*> descriptor can receive three values 20, 30 and 40. The first one triggers the UA to extract attack parameters and wait for further commands. The other two values correspond to the initiation and termination of the attack respectively. As shown in the table, the second and the third fields of the <*o*> descriptor bear the first and the second half of the victim's IPv4 accordingly. In the example given in figure 7.1, the second and the third values of this descriptor are equal to 2383212000 and 3312015300 respectively. So, the IP address can be extracted by a bot as follows: Assuming a quad-dotted notation, the first two digits of each 10 digit number represent the number of digits that this half of the address is consisted of. In the example, the first two values of session id are 2 and 3 leading the bot to extract the first half of the IP address, i.e., 83.212. In the same manner, session version starts with 33, thus allowing the bot to extract the remaining half 120.153. The last two digits of the second field of session version instruct the bot about the type of the attack. Specifically,

a value of 00 means a SYN flooding, while 11 designates a PING one. Special care has been taken for these values to appear as perfectly legitimate ones. To do so, both session id and version numbers are appropriately padded with zeros to reach 10 digits, which is the minimum length suggested by the SDP RFC [1].

Keep in mind that the selected SDP descriptors receive values that correspond to fields which do not affect the session establishment, and thus the covert channel remains functional. In this way, a botmaster is able to hide messages in plain sight without being exposed. On the downside, the use of one optional descriptor for the creation of the covert channel adds 10 extra bytes per message. However, it can be safely argued that this presents a negligible increase in the network traffic to be noticed by the underlying defense mechanisms. Even for a large population of bots, where the botmaster needs to send one SIP request per bot, this augmentation shall be in the order of some tenths of kilobytes (e.g., for 10,000 bots it would be $\approx$ 98 kilobytes).

It should be stressed out that the aforementioned descriptors and fields are not the only ones that can be exploited for secretly communicating information between the two ends. Several other selections and combinations are possible. However, each of them should be done in such a way that will attract the minimum attention. For instance, the *k=* descriptor is to be avoided because its use is not recommended by the RFC [1]. Also, the employment of a large number of SDP optional fields for the needs of the covert channel would not only raise suspicions, but also augment the volume of each SIP request. On the other hand, the information carried by the *a=ptime:<packet time>* descriptor in our protocol could be moved to the padded segment of the *<o>* descriptor as given above.

Another point of interest here is that the architecture is fully dynamic because it can be used both by static and mobile UAs. That is, due to SIP intrinsic operation, each bot is reachable from virtually anywhere. As explained in section 3.4, the IP of the Registrar may change but the bots can become aware of this shift via a domain fluxing scheme or otherwise by extending the C&C instruction repertoire. The reader would likely notice that the communication protocol between the bot master and the bots is one-way. That is, a bot does not send any messages toward its bot-herder. Actually, from the botmaster's point of view, this is not really a problem; as she is in control of the SIP registrar she always knows which bot is alive (i.e., has been registered with the Registrar). On the other hand, one can anticipate that this approach also contributes in

keeping the communication channel as hidden as possible. Putting it another way, the less information are transmitted towards a single receiver (the bot-herder) the less the chances of revealing the channel.

7.2.1 TestBed setup

In order to evaluate the effectiveness of the C&C channel we created a testbed depicted in figure 7.2. We used 7 SIP UAs, one of which was used as the Botmaster and the rest as bots. The SIP UA were developed in JAVA language using the JAIN-SIP library [228]. Each UA runs on an Intel i3 3.3GHz processor with 4GB of RAM. The well-known SIP proxy Kamailio [207] has been employed in the cloud as both a SIP server and Registrar. The server machine was equipped with 1GB of RAM. Finally, the victim's machine was running on an Intel Pentium 4 2.8GHz processor having 1GB of RAM available.

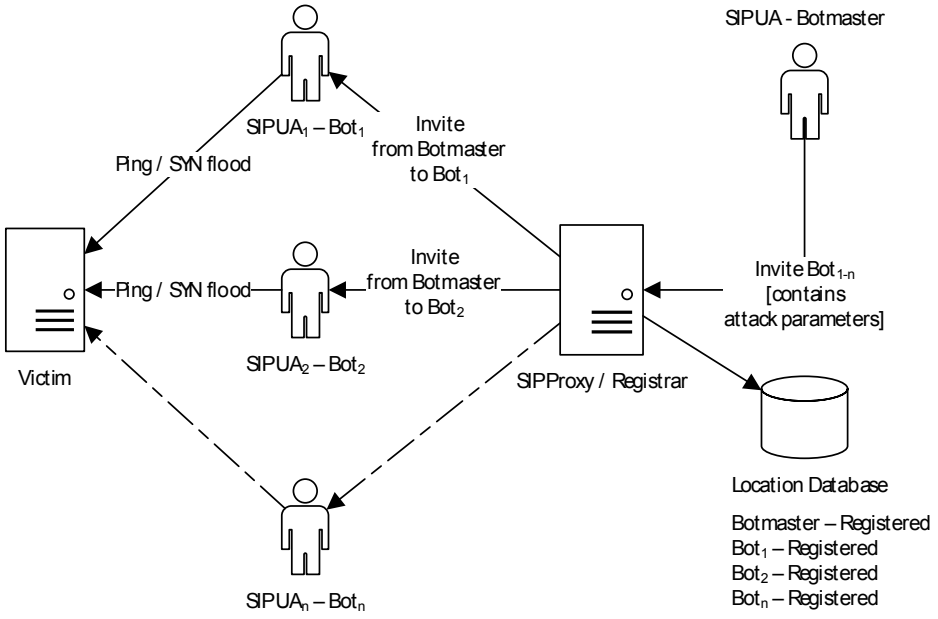

FIGURE 7.2: Deployed testbed and generic attack scenario

We created six different scenarios each one employing a variant number of attack threads launched by each bot. We released both PING and SYN flooding assaults, each one with different number of attack threads. The first three scenarios correspond to a SYN flooding attack, while the rest to a PING one. For SYN flooding we used correspondingly 5, 15 and 30 attack threads per bot, while for PING 30, 80 and 160. This increased number of attack threads in the second type of attack was used in an effort to augment the impact of this particular attack. This is because a SYN flood is generally more powerful in

contrast to a PING one. We used this simulation methodology aiming to grab a better understanding of the attack impact, especially when its volume augments. We employed three metrics to estimate the fallout of each type of attack on the victim's machine; network bandwidth utilization, memory consumption, and CPU usage.

7.2.2 Results

Figures 7.3 and 7.4 present snapshots of the received network traffic and CPU usage at the victim side under a SYN and PING flood attack respectively. From the first figure, one can easily observe that as the number of attack threads per bot remains low, the incoming traffic at the victim's side presents moderate fluctuations. On the downside, when the number of threads increases significantly the incoming traffic doubles. For example, when the threads per bot become equal to 30 the network volume doubles reaching 6MB/sec.

Figure 7.4 depicts CPU usage at the target machine for a different number of PING flooding threads. It is well perceivable that as the number of threads per bot increases, the CPU utilization percentage augments notably. For example, when the number of the attack threads per bot is set to 30 the CPU usage reaches a maximum value equal to 25%. On the other hand, when the number of threads per bot are sextupled (180), CPU usage reaches a peak value of 30%.

Regarding the memory consumption at the victim's side, we perceived a worst case increment of $\approx$ 100% (from 12% to 24%) in the case of PING flooding, and $\approx$ 118% (from 22% to 48%) for the SYN one.

7.3 SDP malformed message attacks

This section details on SDP attacks targeted at both SIP phones and servers. It also offers a concrete example of a botmaster-to-bots covert communication channel realized via SDP.

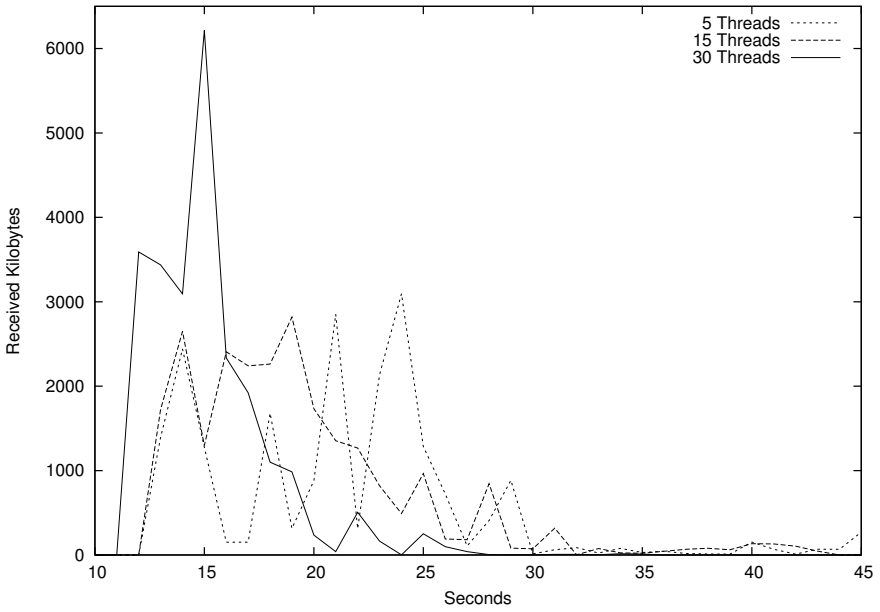

FIGURE 7.3: Network utilization at victim's side under a SYN flood attack

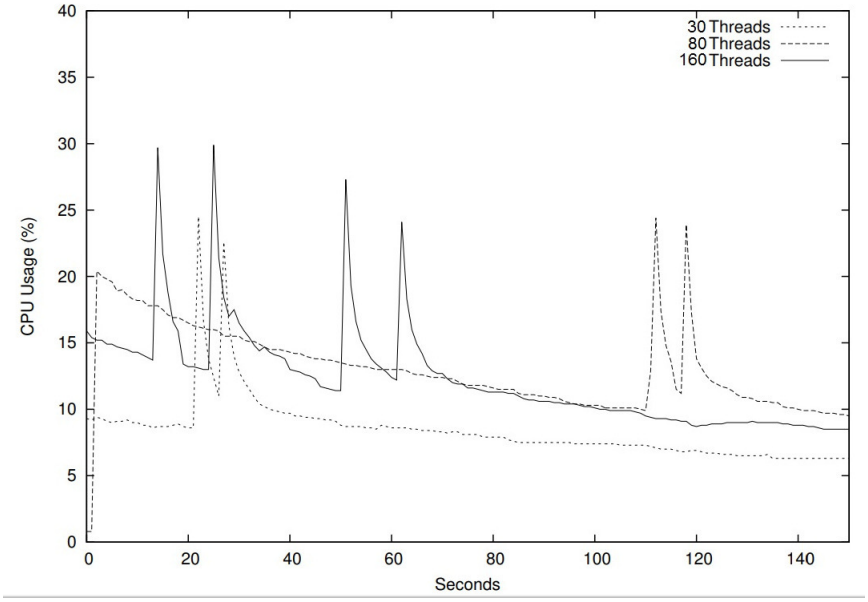

FIGURE 7.4: CPU usage at victim's side under a PING flood attack

7.3.1 Preliminaries and dataset

During the last few years, the industry and research communities came across different kinds of vulnerabilities pertaining to SDP [229, 230, 231]. These vulnerabilities have been collected and categorized in databases [232], [233], and are mostly due to software bugs in the respected products, either VoIP servers or UAs. Such weaknesses may lead

to attacks ranging from DoS to the execution of malicious code. It is therefore obvious that SDP-based attacks can cause unpredictable behaviors on real-life software and/or hardware systems of everyday use.

Attacks which exploit SDP are typically performed in the context of an offer/answer media negotiation. Based on RFC 6337 [38], which describes the offer/answer model [1, 234], the SIP request messages that can contain a SDP segment are INVITE, ACK, PRACK, UPDATE. Moreover, the following messages can be used to convey SDP data in the context of a response: 2xx INVITE, 1xx-rel INVITE, 200 PRACK, 2xx UPDATE. Also, there are some messages which may bear a SDP segment, but they are not part of the offer/answer model. Such messages are for example the OPTIONS request and the corresponding 200 OK answer.

A malformed SDP segment is likely to cause DoS to the SIP server or the UA at different phases of the media negotiation process. That is, a malformed SDP segment which appears in the "offer" phase of media negotiation, may go unnoticed by the SDP parser in the SIP server. These messages target directly the end-user or they are used as vehicles to convey information in the context of a hidden communication channel. On the other hand, an attack against a SIP proxy usually happens in the context of an "answer" phase of media negotiation. This typically occurs because as detailed further down, the message parser in several SIP servers is stateful regarding the SDP parameters. This means that it tries to match the SDP parameters contained in the "offer" phase against those carried by the "answer". In case of inconsistencies, and assuming a software bug, the parser may crash. Bearing the above into mind, Table 7.3 summarizes the different facets of this ilk of assault, the attacker's goals, and the attack-affected victims.

To test both the SDP data hiding and attack capacity, we created a seemingly legitimate SDP segment using only the G.711 PCMU codec. This selection appears in all messages depicted in figures 7.5 and 7.6. That is, the corresponding codec is presented in the $<m=>$ line with code "0". Keep in mind that this static payload does not necessarily need additional information to be decoded. In this way, we guarantee that the phones will not reject the message with a "488 not acceptable here" response. After that, as in [33], we crafted specific parts of the SDP segment at will. Using this approach, we have created a dataset of 12 representative malformed SDP message bodies depicted in figures 7.5 and 7.6. Note that in the course of our experiments, we also tested many

more SDP malformed messages and cherry-picked those included in the aforementioned figures. To our knowledge, no such dataset containing a rich set of SDP malformed messages exists, therefore the only option was to create one by crafting SDP messages and performing an error-trial approach to observe the consequences on the various phones and SIP servers.

All these 12 SDP bodies either miss or contain specific pieces of information, which directly or indirectly violate RFCs 4566 [1] and 3551 [234]. More specifically, *MalSDP1* lacks the mandatory <*v*=> descriptor. Moreover, it is inconsistent regarding the <*t*=> descriptor. That is, the start time is greater than the end time. Finally, the *a=rtpmap* attribute does not correspond to an already declared payload. *MalSDP2* contains the word "RANDOM" in the first <*m*=> line. This value is completely irrelevant to the specification. Also, it contains the same media part twice. The <*t*=> descriptor of *MalSDP3* carries an end value which has a length of 11 digits. This number exceeds the maximum allowed limit of 10 digit. Additionally, numbers 1 and 2 cannot be used as a payload because they are reserved. Finally, the numbers 97, 98 correspond to dynamic payloads [234], but they have not been declared in the media level. Keep in mind that codecs which have been defined in "RTP Audio/Video profile", do not require additional information to be decoded. Nevertheless, the aforementioned session attribute lines (used for dynamic codecs), may also be employed for the static ones.

The <*o*=> descriptor in *MalSDP4* carries a session ID which violates the Network Time Protocol (NTP) format as specified in RFC 4566. *MalSDP5*, conveys a cryptographic key via the <*k*=> descriptor. However, the use of cryptographic keys in favor of supporting older implementations are not recommended by RFC 4566. Finally, *MalSDP6* makes use of a very big number equal to $34,567,999 * 10^{33}$, as a session-id in the <*o*=> descriptor. Additionally, it conveys several voluminous strings. Moreover, all the 6 SDP bodies depicted in figure 7.6 present significant fraudulent alterations in either the session or the media part of their SDP segment. Last but not least, the media level part of these bodies carry the numbers 5, 0, which are nevertheless used only with audio codecs.

7.3.2 Experimentation with real-life SIP equipment

The 6 malformed SDP bodies of figure 7.5 were used to test the robustness of 9 different SIP softphones and 1 hardware phone. For selecting the softphones, we searched the

Google play and Apple store, and cherry-picked some of the most popular ones [235], namely Sipdroid v.4.1 beta, EVA Sip phone v.2.1, CSipSimple v.1.02.03, MizuDroid v.2.4.0, Media5-fone v.4.25.4.13060, Linphone v.3.3.2, SessionChat v.6.0, VaxPhone v.8.6.0.2, and rDialer v.1.1. Panasonic KX-HDV130 (with firmware HDV130/06.101) has been used as the hardware SIP phone. This device is amongst the most popular in the Unified Communications (UC) market [236]. All the above mentioned phones' name, model and type are summarized in Table 7.4.

SIP phones employ a set of audio and video codecs for establishing a multimedia session with peer SIP components. Normally, the phone selects the codecs based on a configurable prioritization list. For instance, ITU-T G.711 [237] is one of the most commonly used in VoIP realm. In this context, when the phone receives a SDP segment containing codecs which are incompatible, it will respond with "488 not acceptable here" SIP response. This means that the call is not established. As observed from Table 7.4, this error message was generated by phones 3, 5 after sending all *MalSDPs* depicted in figure 7.5. Finally, the same error code was generated by the Panasonic phone, after receiving *MalSDP1* and *MalSDP2*. Obviously, this is an indication that the phone produces a generic - and in this case misleading - codec error message instead of a more precise and detailed one pertaining to the specific SDP error. The rest of the phones responded with different messages. Sipdroid produced a "403 forbidden" message. The EVA Sip phone encountered a bug as it kept ringing continuously even after pressing the response button. However, in this case, the corresponding logs showed that the rest of the protocol messages (180 RINGING and 200 OK) have been exchanged successfully. A similar behavior to the EVA SIP phone has been observed with the MizuDroid softphone. Finally, the Linphone ringed normally and the call was answered, but for *MalSDP2* and *MalSDP5* the call time counter had frozen.

We also sent 5 consecutive instances of *MalSDP6* message to all the phones.The EVA Sip phone, Sipdroid, CSipSimple, and Media5-fone showed the same behavior as previously. On the other hand, Linphone and MizuDroid crashed, and additionally for MizuDroid, the smartphone rebooted without a warning.

Regarding the softphones on the iOS platform, SessionChat responded with a "480 Temporarily not available" for *MalSDP2* to *MalSDP5*. For *MalSDP1*, the same softphone displayed a "400 SDP parse error" message. VaxPhone and rDialer did not respond at

all. Finally, the Panasonic phone responded with a "488 Not acceptable here" error message for *MalSDP1* and *MalSDP2*. For the rest, the phone ringed normally and the call was answered. Table 7.4 summarizes the behaviour of the tested software and hardware phones. Generally, a SIP phone which receives a manipulated SDP request and it responds normally, provides the attacker with beneficial information regarding an inherent weakness of the SDP parsing process at the UA side. Then, using this information gathered during a reconnaissance phase, the attacker knows which UAs are suitable for hosting their bot(s). After infection (which is outside the scope of this PhD thesis), the botmaster can uneventfully convey hidden commands toward their bots. An example of this situation is the Panasonic device listed in Table 7.4, which does not produce an error code for messages *MalSDP3* to *MalSDP5*. Details on the creation of such a communication channel are given further down in this section.

As the reader realizes, the root cause of this kind of attacks is that the SIP proxies forwarding SIP messages between the peers do not inspect their SDP bodies for possible inconsistencies. In our tests, we employed two of the most popular SIP servers in the VoIP realm, namely Kamailio ver. 5.0.2 (former OpenSER) [207] and Asterisk ver. 14.6.0 [238]. Both these servers relayed the majority, if not all, of the crafted messages. Precisely, Kamailio relayed all the messages depicted in figures 7.5 and 7.6, while Asterisk only those included in figure 7.6. Generally, Asterisk presented a higher resiliency in crafted SDP bodies, especially when the manipulation concerns the <*m*=> descriptor. This is the main reason that the messages depicted in figure 7.5 were rejected by Asterisk.

In any case, this parsing negligence leaves plenty of room for creating covert communication channels. That is, as already mentioned, the attacker is able to uneventfully convey "in plain sight" whatever information they wish, by simply altering the SDP segment of certain SIP messages. Let's say that a botmaster wishes to construct a simple communication channel consisting of 3 commands pertaining to the type, the parameters, and the initiation/termination of an attack. To do so, they rely on, say, the <*a*=> and <*t*=> descriptors, and exploit their respective fields included in red rectangles in Table 7.1. For the former descriptor, the possible values are chosen to be 20 (notifies the receiving UA-bot to extract the attack parameters and stand by for further instructions), and 30, 40 (correspondingly signify the initiation/termination of the attack. On the other hand, the first and the second half of the victim's IP are included in the second and third fields of the <*t*=> descriptor. Say for example that the victim's IPv4 is 212.120.83.153.

Then, the second and third fields of the <*t*=> descriptor would be 3|3|212|120|00 and 2|3|83|153|110 respectively. Note that the vertical bar character has been included in the numbers to ease the comprehension of the example. That is, assuming a quad-dotted notation, the UA-bot will interpret the aforementioned numbers as follows: the first 2 digits of each number (3, 3 for the first number and 2, 3 for the second) carry the number of digits that comprise this half of the IPv4 address, which immediately follows (i.e., 212.120 for the first half and 83.153 for the second). Also, the type of the attack is included in the first 2 of the last 3 digits (11) of the second number. Note that all the selected values in both these descriptors seem perfectly legitimate. For instance, the length of the 2 numbers, correspondingly denoting the session id and version, is 10 digits, so as to fully comply with RFC 4566. Also, the channel remains functional because the selected <*a*=>, <*t*=> descriptors receive values that correspond to fields which do not affect the session establishment. Naturally, one has to also consider that each optional descriptor adds ≈ 10 extra bytes per message, which however is insignificant and hardly perceivable by the underlying defense mechanisms. Another point to consider is that the above mentioned channel will be blocked by the SDP protection module detailed in the next section. This is simply because the syntax of the <*t*=> descriptor infringes rule no. 62 as referred in Appendix B.

Lastly, more worrisome is the fact that apart from crafted messages, we managed to relay a huge SDP segment using Kamailio. That is, we created a SDP segment with size equal to 12,116 bytes. This segment passed through the SIP proxy intact without returning an error code to the sender. This means that the SIP proxy could potentially relay enormous SDP bodies without hesitation. This naturally renders both the SIP server and registered UAs prone to simple volumetric DDoS attacks.

7.4 Mitigation

This section elaborates on our solution to bolster defense capabilities against SDP-driven attacks.

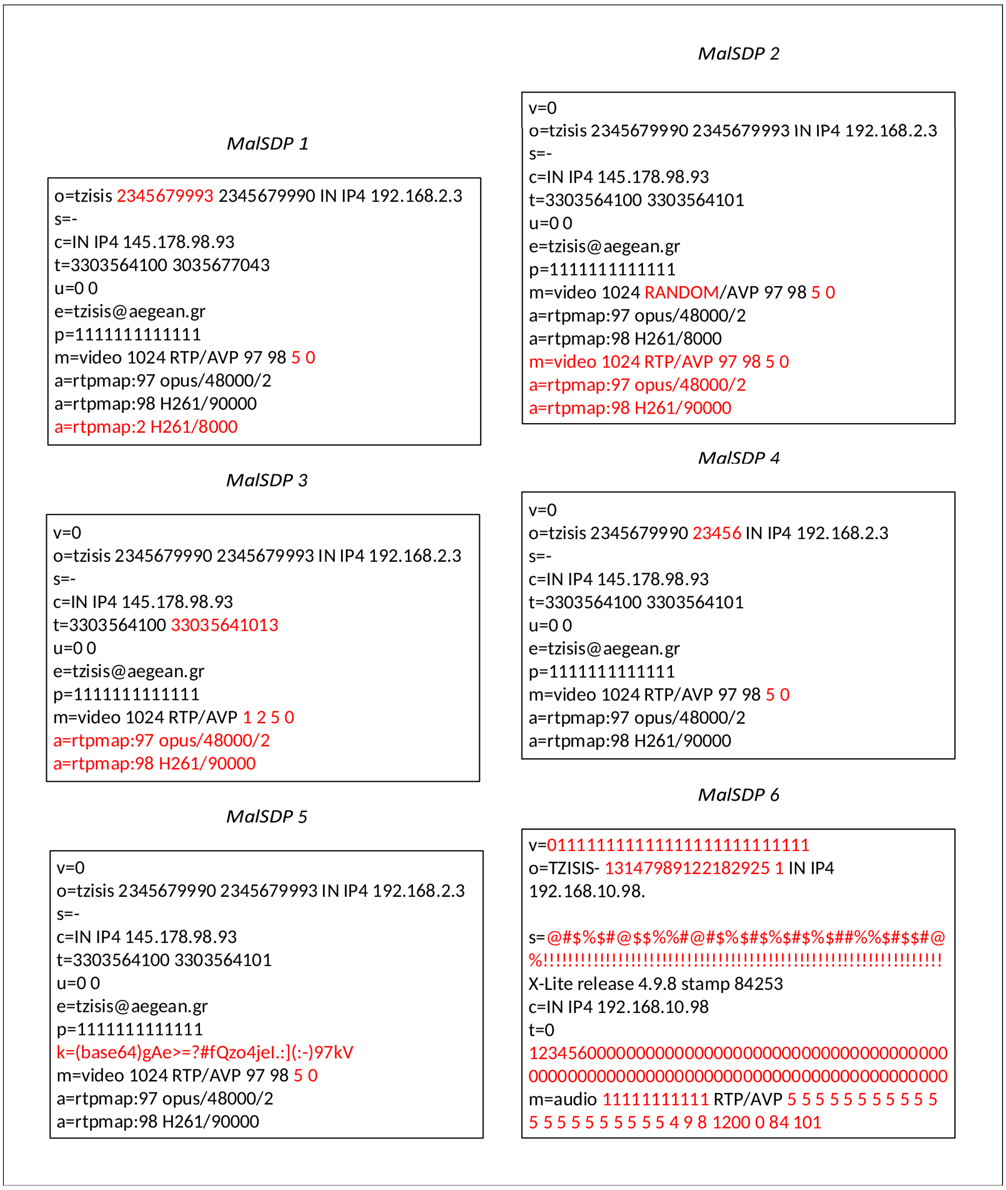

FIGURE 7.5: Malformed SDP bodies relayed from Kamailio (The malformed part is shown in red font)

7.4.1 High-level description and network architecture

A SDP parser is a software module capable of analyzing SDP bodies. Normally, such a module is present in every SIP server and UA. However, as already explained in section 7.3, both these (real-life) entities largely fail when it comes to malformed SDP bodies. So, from an attacker's viewpoint, virtually all major SIP components, either servers or end-user equipment are sitting targets. Also, more advanced attackers, say, botmasters perceive the SIP infrastructure as an alluring means of building their C&C protocol. Once more, as in the case of many other protocols or systems, it seems that security

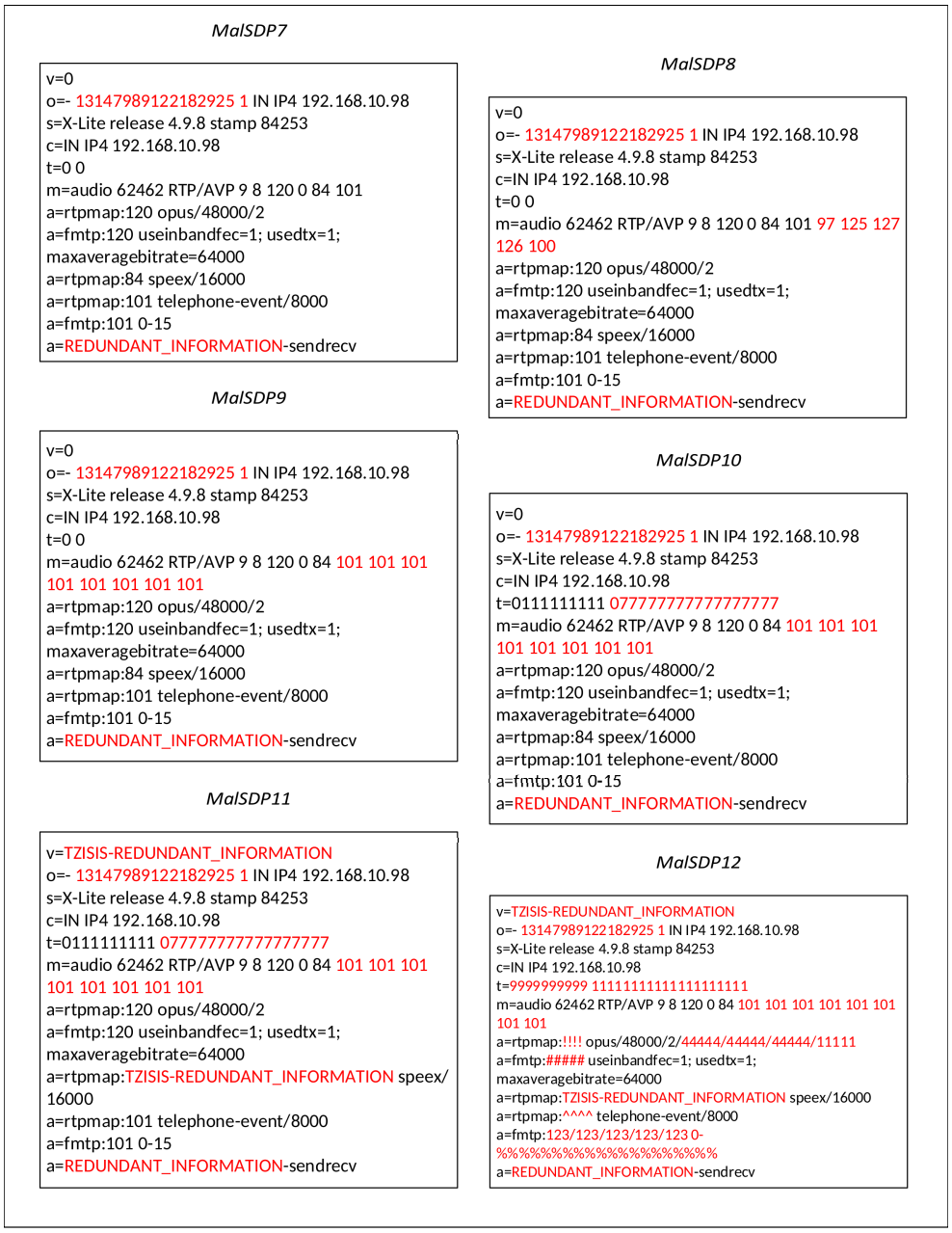

FIGURE 7.6: Malformed SDP bodies relayed from Kamailio and Asterisk (The malformed part is shown in red font)

is not the first priority of the designers/implementors of SIP products and security-by-design is still a far-fetched goal. The solution however is rather straightforward; the SDP parser must be able to tell between a well-formed and malformed message in terms of RFC 4566. Preferably, this must be done for both the incoming and outgoing traffic to/from the SIP server.

To this end, this work introduces an autonomous open-source software module for SDP messages that can be either physically co-located with the SIP proxy or reside in a

different machine, say, in the perimeter of the network. In the context of this work, we opt to select the latter configuration as it is entirely transparent to any SIP compliant component. This option also enables one to integrate the SDP parser with, say, a next generation firewall. The overall architecture of such a network configuration is illustrated in figure 7.7.

Very similar to a firewall, the SDP parser protection module inspects each message based on a parsing policy reflected to one or more sets of rules, and decides if it can be forwarded to the SIP server or silently dropped. That is, the filtering operation starts by extracting the message body from any incoming SIP request. Next, the SIP headers of the message are stripped away and the remaining part containing the SDP message (descriptors) is kept for further processing. Appendix A.4, algorithm 10, provides an overview of the process flow and discrete operations of the SDP parser.

7.4.2 Filtering policy and parser rules

As per RFC 4566, the implemented rules have been divided into 4 major categories (policies). Namely, we compiled a set of SDP parsing rules and we grouped them into the following categories: "MUST", "NOT RECOMMENDED", "SHOULD NOT". For the remaining rules, i.e., those which do not fall within any of the above mentioned categories, we created a fourth category, namely "GENERAL INCONSISTENCIES". The "MUST" policy can be further categorized into 3 sub-policies, namely "MUST APPEAR", "MUST NOT APPEAR", and "MUST HAVE". Until now, altogether the 4 chief categories contain a total of 100 rules, which tackle the vast majority of inconsistencies that may exist in a SDP message. Of course, one is able to add more rules at any time. The full list of the implemented rules is given in Appendix B.

Note that all the rules have been created after grindingly extracting every single piece of information which pertains to the standard syntax of SDP. For example, with reference to figure 7.1, rule 63 describes the length of the <*t*=> descriptor. Based on RFC 4566, the length of this descriptor must be equal to 2, meaning that if this descriptor appears in a SDP session level part, it must have the form *t*=<*start-time*> <*stop-time*>. The same logic has been followed for the rest of the implemented rules. Up to now, our implementation covers all the mandatory descriptors, plus many of the optional ones.

Full coverage is expected in a future version of the parser, which is publicly available at [216].

The implemented rules can be categorized either by using the 4 above mentioned categories, or according to their relevance to the different SDP regions (i.e., with reference to figure 2.2, session level and media level). This three-fold sub-categorization is described under the "Filtering sub-policies" header in the Appendix B. For example, the mandatory descriptors reside only in the session level of the SDP segment. So, the first filtering sub-policy in Appendix B, dictates that all the mandatory descriptors must be present in an SDP message. Otherwise, the message is dropped. Overall, for every defined rule, each time an inconsistency is found, the parser silently drops (and preferably logs) the SIP message. On the contrary, if the message is found to be sound it is forwarded to the SIP proxy.

7.4.3 Design considerations

Misuse detection systems (also known as signature-based detection) rely on known signatures, that is, detection rules aiming to distinguish legitimate traffic instances from the malicious ones. However, while these systems are able to detect known attacks, they miss to recognize novel attacks or variations of known ones. Thus, the detection ability of a misuse detection system, as the one proposed in this work, primarily depends on the newness of the detection rules the system has been configured with. In this context, we selected the specific rules based on the fact that, in the normal case, altogether the mandatory descriptors, namely $<v=>$, $<o=>$, $<s=>$, $<m=>$, $<t=>$ offer a limited number of "variables" in contrast to the optional ones. The term "variable" refers to the part of every descriptor which can be altered by the sending entity, either for benign or malicious purposes. For example, with reference to the fourth column of Table 7.1, and for the $<o=>$ mandatory descriptor, the number of "variables" is equal to 6. In general, altogether the mandatory descriptors but $<m=>$ provide 13 "variables", which can be malignantly altered. In the normal case, $<m=>$ carries 3 "variables" plus a number of payloads, which ranges from 1 to n. Also, in certain cases, e.g., in an audio/video session, there may be more than one instances of $<m=>$.

The 10 optional descriptors on the other hand provide a much larger "variable" space for the assailant, simply because their number may be triple the quantum of the mandatory

ones (some of them may even be repeated in the SDP segment), and pieces of information conveyed by certain optional descriptors may not be critical or simply ignored by the SIP proxy. Therefore, as shown in the example of section 7.3, the optional descriptors are low hanging fruits for attackers who seek to hide information in a SDP body toward creating a covert communication channel. Even worse, considering the offer/answer model (see section 7.3) the UA is capable of renegotiating the SDP parameters at any time and each time present a different SDP segment. For instance, if the caller wishes to put the callee on hold, it needs to send them a *re-INVITE* carrying the SDP <a=inactive> attribute (along with any other crafty SDP alterations).

All in all, the greater the number of "variables" the larger the attack vector the malformed message may yield, because the aggressor is able to arm the message with several different variations of malformed fields in an effort to inflict damage to the victim or pass through a richer set of hidden information. Of course, as already touched upon in section 7.3, the inclusion of many optional descriptors in the message is sure to increase its size (note that any SIP implementation must be capable of handling messages up to the maximum datagram packet size, e.g., for UDP this size is 65,535 bytes, including IP and UDP headers). Also, it is to be noted that in order to avoid the fragmentation of messages over UDP and offer congestion control for larger messages, RFC 3261 states that any request within 200 bytes of the path Maximum Transmission Unit (MTU) or larger than 1,300 bytes if the path MTU is unknown, must be dispatched using TCP. This, on the other hand, complicates the creation of large malformed SDP messages along with IP spoofing at the attacker's side.

Bearing the above in mind, we designed the SDP parser in such a way so the mandatory descriptors - which are critical and cumulatively have the lesser number of "variables" - are always filtered first. After that, the selection of the rules gradually covers the case of voluminous SDP segments, where the parser possibly needs to go through a much greater number of rules. More details regarding the time complexity of the parser are given in the next section.

7.4.4 Implementation

The SDP message parser has been implemented as a standalone multi-threaded Java application. Its first task is to intercept SIP requests and responses carrying a SDP

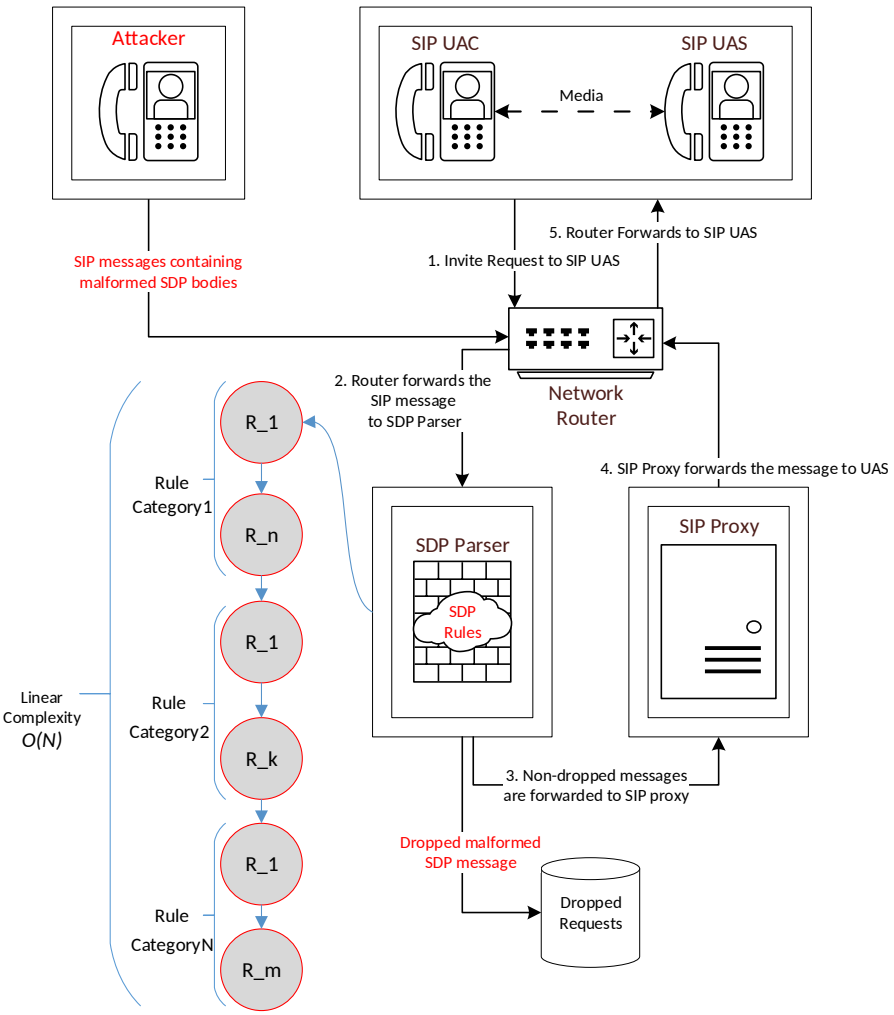

FIGURE 7.7: Overview of the deployed testbed. The letters n, k, m represent the number of rules per rule category

segment (see section 7.3) and destined to the SIP proxy. To do so, we employed the well-known JAIN-SIP stack [239]. The JAIN architecture comprises 3 different layers. The first layer corresponds to the protocol stack. The second refers to the JAIN layer, and the last one to the actual application. Precisely, the latter layer provides the necessary methods to format and send out SIP messages. Furthermore, it provides interfaces capable of extracting and parsing specific message headers. As shown in figure 7.7, we employed the well-known iptables linux utility program to redirect the traffic as follows: for every incoming packet with destination TCP or UDP port equal to 5060 (the standard SIP proxy port used for non-encrypted signaling traffic), we forwarded the packet to the parser, listening on the corresponding TCP or UDP port 6090. If the message is found to be well-formed, it is forwarded to the SIP proxy as normal. This is done with the help

of the *SendRequest* method of the JAIN SIP API.

The parser allows the network administrator to select the descriptors to be filtered based on a graphical user interface (GUI). Namely, the user is capable of selecting a specific policy reflected to one or more categories of rules, as illustrated in figure 7.7 and described in Appendix B. A screenshot of the parser's GUI is presented in figure 7.8. As observed from the figure, the various filtering policies (categories of rules) reside in the upper left section of the GUI. Recall that the parser application is freely available for further development and experimentation [216].

We also developed a Java attack tool capable of generating malformed SIP *INVITE* messages carrying a SDP segment. The application produces a variety of malformed SDP messages, including the ones described in section 4, as well as others with random values in specific SDP fields. Also, the same tool allows one for sending a single or a surge of malformed SDP requests towards a UA or the SIP proxy. While our experiments make use of only SIP *INVITE* requests, the exact same methodology can be followed for any other SIP request/response having a SDP segment, as detailed in section 7.3.

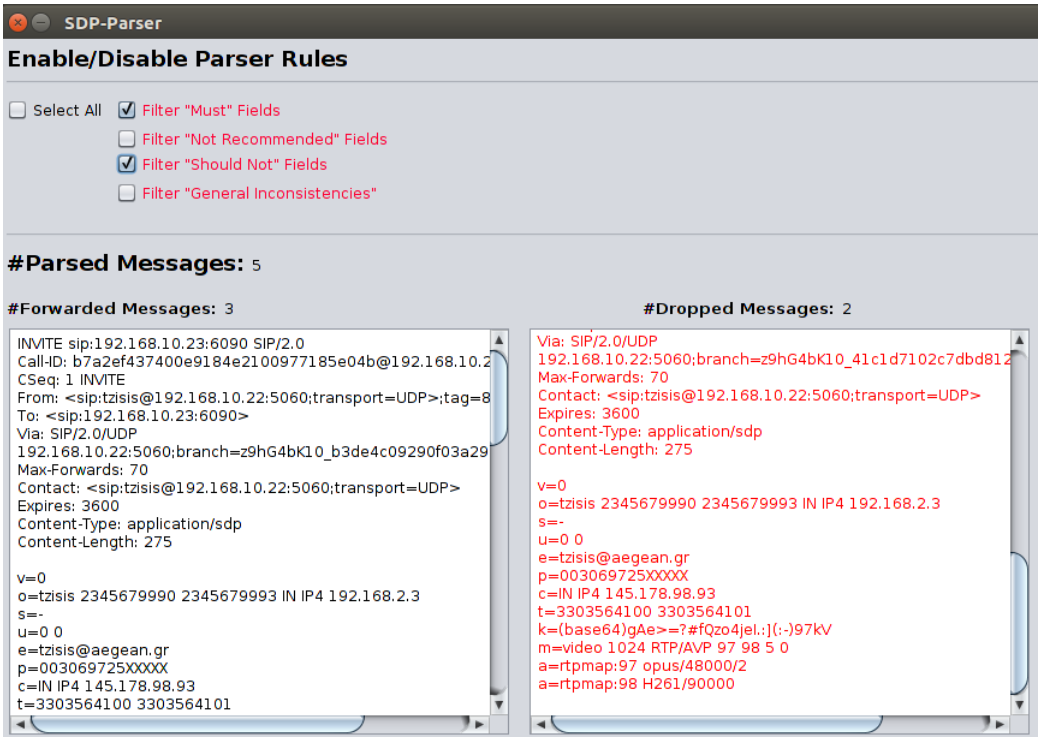

FIGURE 7.8: The front-end of the SDP parser

7.4.5 Performance evaluation

This section reports on the performance evaluation of the proposed scheme. To this end, we employed a SIP testbed architecture in which we launched various attack scenarios to estimate the effectiveness of our SDP parser under different traffic conditions. We assess our solution in terms of the introduced overhead for service provision. The following subsections elaborate on the testbed architecture, the deployed scenarios, and the obtained results.

7.4.6 Testbed

The employed testbed is depicted in figure 7.7. Specifically, 3 different Virtual Machines (VM) have been used to host the UAC/UAS, the SDP malformed message attack tool described in subsection 7.4.4, the SDP parser, and the SIP proxy. The physical machine hosting the VMs is equipped with an Intel i5-4310m processor clocked at 2.7 GHz and 8 GB of RAM. The Kamailio SIP server in ver. 5.0.2 has been used as a SIP registrar and proxy. We utilized the well-known Sipp tool [222] to test the performance of the parser under stress, that is, by simulating legitimate SIP calls between a caller and a callee. The UAC and the UAS operate on the same VM using different port numbers, namely 6040 and 8040 correspondingly. Keep in mind that the conveyed traffic must always be filtered by a firewall.

As already pointed out, for launching the attacks, we used the Java tool already described in subsection 7.4.4. This attack tool dispatches a number of SDP malformed requests (those of figures 7.5, 7.6, plus the one with the huge SDP segment) depending on the needs of each scenario. A detailed description for the employed scenarios can be found in the next subsection. The SDP parser has been logically placed in front of the SIP proxy, but both run on the same VM, sharing a dual-core CPU and 2 GB of RAM. Having the SDP parser hosted by a separate physical machine is estimated to introduce negligible time penalty in terms of communication time, given that the SIP proxy and the parser will normally reside in the same subnetwork.

7.4.7 Scenarios

For evaluating the performance of the parser, and consequently estimate the overhead inflicted by our solution on the provided service, we created a set of 16 scenarios (SN) divided in 4 categories as shown in Table 7.5. That is, as indicated in the third column of the table, for each category of scenarios, we flooded the target proxy with 500, 1500, 2500, and 4000 SIP INVITE requests, armed with malformed SDP segments, using random time intervals of 3 to 10 sec between the flood bursts.

We selected random time intervals because the strategy of the attack (i.e., the use of a certain distribution in the attack pattern) does not affect the detection accuracy. This happens because a syntactically wrong message will be dropped independently of the call distribution. Namely, in the worst case scenario, the system may loose some packets due to lack of CPU resources, but statistically the SDP parser detection accuracy is not affected as the arriving INVITE packets are mirrored to statistically independent events.

Moreover, as shown in the second column of Table 7.5, during the execution of the attacks, the SDP and SIP proxies were stressed with legitimate traffic following a pace of 10, 20, 40 and 80 calls per second (cps).

7.4.8 Performance evaluation

Given that the detection accuracy of the SDP parser depends solely on the implemented rules (i.e., the parser had 100% detection accuracy against the malformed messages of figures 7.5 and 7.6), the focus of this section is on service time provision. To this end, we assess the overhead on both the SDP parser and SIP proxy in terms of message processing time. This means that for each group of scenarios in Table 7.5 (SN-1-1 to SN-1-4, SN-2-1 to SN-2-4, and so on), we measured the processing time per incoming message in both the SDP parser and the SIP proxy. Next, we present the results per group of scenarios using box-and-whisker plots in figure 7.9, where the lower hand edge of the blue box corresponds to the 1st quartile, the upper hand edge of the red box to the 3rd quartile, and the line between the boxes is the median. For instance, in the upper half of the figure, the distribution of message processing time for all scenarios (SN-1-x) in group SN-1 for the SDP parser is marked as “SN-1” in the horizontal axis.

For the last 4 sub-scenarios (SnN-4-x), we used the same number of SDP malformed *INVITE* packets. As explained in subsection 7.4.7, this has been done because the detection accuracy of the SDP parser is not affected by the number of packets. Nevertheless, we used a different number of packets for the first 3 groups of scenarios (SN-1 to SN-3) for the purpose of demonstrating that the SDP parser behavior remains stable independently of the deviation in the rate of the incoming malformed packets. As expected, during the experiments, we witnessed network performance issues expressed as packet loss. This was due to the excessive attack traffic on top of the legitimate calls. Keep in mind that this behavior is expected because of the connectionless nature of UDP, which is often preferred over TCP, especially when the number of devices connecting to the SIP server grows.

Focusing on the upper half of the figure, i.e., the SDP parser, we easily observe that the average message processing (parsing) time per incoming message for all the scenarios fluctuates between 30 to 50 msec. Also, the minimum and maximum values for all the 4 groups of scenarios are between 17 and 60 msec. Overall, we can safely argue that the SDP parser adds a negligible time of the order of tenths of msecs. Even in the most stressing group of scenarios (SN-3, SN-4) where the cps is 40 to 80, the average parsing time does not exceed 60 msec. However, bear in mind that the aforementioned parsing times are only indicative because they do not only depend on the volume and type of SIP traffic, but also on the computing resources available at the parser side (i.e., processor type, available memory, etc).

When comparing the first 3 groups of scenarios, one can also perceive an increment of $\approx$ 10 msec in the average message parsing time proportionally to the cps parameter. That is, from $\approx$ 30 msec for SN-1 to $\approx$ 40 for SN-2 and $\approx$ 50 for SN-3. SN-4 on the other hand, showed faster message parsing times because the volume of the SDP malformed messages sent in all SN-4-x was constant. This is verified by the corresponding Interquartile Range (IQR = Q3-Q1), which is the smallest amongst all four group of scenarios.

In a nutshell, and as explained further in the next subsection, the parser introduces a delay proportional to the number of rules. That is, in terms of asymptotic notation, the parser's upper bound is *O(N)*, where N is the number of rules enabled in the parser. Of course, this number is reflected to the currently activated SDP filtering policies as the case may be.

To acquire a better understanding of the aforementioned results in terms of SDP parsing penalization, we compare the previous times against those produced by Kamailio for exactly the same 4 groups of scenarios. The results are depicted in the bottom half of figure 7.9. From the figure, it is easily perceivable that Kamailio produces very fast and stable message processing times in the order of msec even under excessive stress. This is however highly expected as modern VoIP servers are developed for handling thousands of cps [207]. Specifically, for all the 4 scenarios, the average processing time per message fluctuates between 0.48 and 0.52 msec. This means that even with the addition of the SDP parser in front of the SIP server, one would expect overall message processing times in the order of 100 msec. Also, in the case of a volumetric attack leveraging SDP, the SDP parser is expected to ease the burden of the SIP server because the malformed messages will be dropped.

7.5 Discussion

As already pointed out, the SDP parser has been developed with a firewall-based logic. That is, as shown in Algorithm 10, upon the arrival of a SIP message carrying SDP information, the parser extracts the SDP segment and performs a linear search against the list of the enabled rules, with the aim of detecting inconsistencies. Obviously, a SDP malformed message will pass through the parser undetected in case it contains an inconsistency that has not been implemented as a rule (or due to the current policy, the corresponding set of rules is not enabled). As already mentioned, the SDP parser performs a linear search, thus, assuming N active rules, the worst case will require the examining of N rules, while the best only involves the first rule in the current set. Note that the worst case applies also when the incoming message is well-formed. To put it another way, the overall time required to detect a malformed message is always proportional to the number of the enabled rules and does not depend on the specific set of rules or the combination of the enabled set of rules as the case may be. Therefore, the time complexity of the parser is *O(N)*. On the other hand, the detection accuracy of the parser will always reach 100%, assuming that the presented inconsistency has been addressed in an already implemented rule, which is additionally enabled.

A separate note should be done regarding the selection of the Java programming language for implementing the parser. Note that this was mainly done for the sake of portability. In

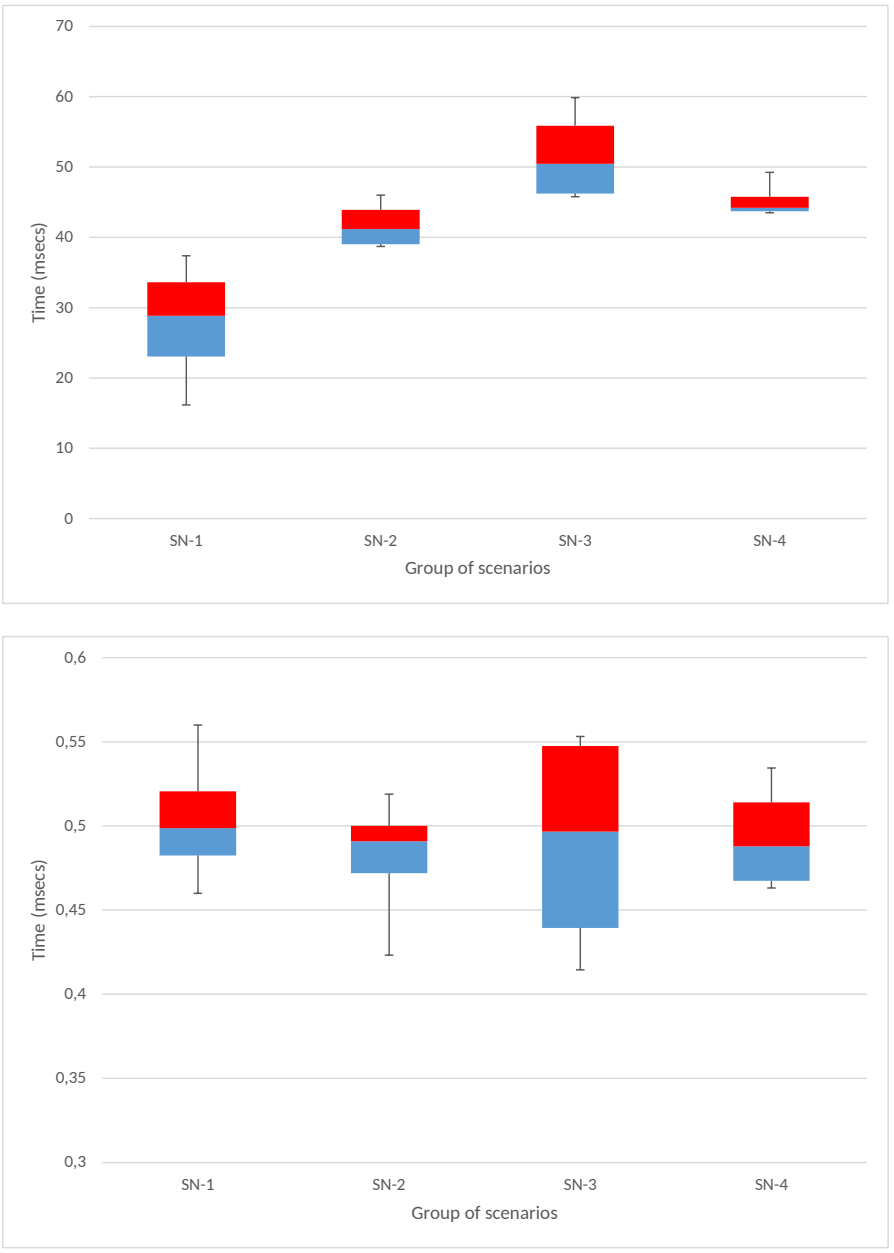

FIGURE 7.9: Time overhead per category of scenarios for the parser module (upper half) and the SIP proxy Kamailio (bottom half)

this respect however, the parser's performance may be negatively affected by a) the Java Virtual Machine translation cost, and b) Java garbage collection process. Therefore, if portability is not a priority, a C-based implementation of the parser may perform faster given that (a) C code is directly compiled into native code, thus is expected to surpass Java bytecode even in the case of Just-in-Time compilation [240], and (b) memory management optimizations are possible when programming with C (but this may on the other hand increase the software bug surface).

TABLE 7.1: SDP descriptors. Some of the optional descriptors may be repeated across the SDP message segment

Descriptor	Mandatory	Meaning	Descriptor format	Example of a valid value
<*v*=>	yes	Carries SDP version	v= (Protocol version)	v=0
<*o*=>	yes	Corresponds to the originator of the session	o=<username> <sess-id> <sess-version> <nettype> <addrtype> <unicast-address>	o=tzisis 2233445566 2233445577 IN IP4 192.168.1.1
<*s*=>	yes	Conveys the textual session name	s=<session name>	s=SDP seminar
<*m*=>	yes	Media description	m=<media> <port> <proto> <fmt> ...	m=audio 49170 RTP/AVP 0 8 97
<*t*=>	yes	Start and end times of a session	t=<start-time> <stop-time>	t=1280656800 1290938400
<*i*=>	no	Provides textual information regarding the session	i=<session description>	i=Department of Information & Communication Systems Engineering
<*u*=>	no	Provides additional info about the session using a URI	u=<uri>	u=http://www.icsd.aegean.gr
<*e*=>	no	Provides a contact email address	e=<email-address>	e=tzisis at aegean.gr
<*p*=>	no	Provides a contact telephone number	p=<phone-number>	p=+30 22730 82XXX
<*c*=>	no	Provides data needed for the connection	c=<nettype> <addrtype> <connection-address>	IN IP4 192.168.1.1
<*b*=>	no	Carries information for the aggregate bandwith limit for the session	b=<bwtype>:<bandwidth>	b=AS:64
<*r*=>	no	Provides information regarding the repeat times of a session	r=<repeat interval> <active duration> <offsets from start-time>	r=7d 1h 0
<*z*=>	no	Carries information for scheduling a session	z=<adjustment time> <offset> <adjustment time> <offset>	z=2882844526 -1h 2898848070 0
<*k*=>	no	Offers information about encryption keys	k=<method>:<encryption key>	k=(base64)gAe1111fQzo4jeJdfrtsY197kV
<*a*=>	no	Used for extending SDP	a=<attribute>:<value>	a=rtpmap: 0 PCMU/8000/1

TABLE 7.2: Description of C&C protocol messages (character X corresponds to a single digit of the victim's IPv4 address, and Z refers to a digit used for another command or it is zero-padded)

Descriptor	**Field**	**Value**	**Hidden Message**
<o>	sess-id	33XXXXXXZZ	First half of Victim's IP
<o>	sess-version	33XXXXXXZZ	Second half of Victim's IP
<o>	sess-version	ZZZZZZZZ00	SYN Flood attack
<o>	sess-version	ZZZZZZZZ11	PING Flood attack
a=ptime:<packet time>	-	20	Save attack parameters & wait
a=ptime:<packet time>	-	30	Launch attack
a=ptime:<packet time>	-	40	Stop attack

TABLE 7.3: Summary of attack types and impact on the victim

No	**Attack Type**	**Attacker's Goal**	**Victim**
1	Flooding with malformed messages	Annoy the user, crash or paralyze the device	SIP Proxy/Client
2	Malformed messages	Crash, freeze the device or execute malicious code	SIP Proxy/Client
3	Creation of covert communication channels	Convey hidden commands	Internal/External entities

TABLE 7.4: SIP Phones used and results (Sw/Hw stands for softphone/hardware phone)

Attack Type (in Table 7.3)	**Phone Name**	**Attacked with MalSDP#**	**Behavior**	**Type/OS**
1	Sipdroid	All *MalSDPs*	Error: Codecs incompatible, 403 Forbidden	Sw/Android
1, 2, 3	EVA Sip Phone	*MalSDP1**, *MalSDP4 to 5*$^{+}$	*Continuous Ringing, $^{+}$Error: 400 Bad SDP	Sw/Android
1	CSipSimple	All *MalSDPs*	488 Not acceptable here	Sw/Android
1, 2, 3	MizuDroid	All *MalSDPs*	Continuous Ringing	Sw/Android
1	Media5-fone	All *MalSDPs*	488 Not acceptable here, Missed call	Sw/Android
1, 2, 3	Linphone	*MalSDP1** and *MalSDP3 to 4**, *MalSDP2*$^{+}$ and *MalSDP5*$^{+}$	*Responded normally, $^{+}$Responded but call time counter froze	Sw/Android
1	SessionChat	*MalSDP1**, *MalSDP2 to 5*$^{+}$	*400 Parse Error, $^{+}$480 Temporarily not available	Sw/iOS
1	VaxPhone	All *MalSDPs*	No Response	Sw/iOS
1	rDialer	All *MalSDPs*	No Response	Sw/iOS
1, 2, 3	Panasonic	*MalSDP1 to 2**, *MalSDP3 to 5*$^{+}$	*488 Not acceptable here, $^{+}$The phone rings and the call is answered	Hw/HDV130

TABLE 7.5: Parameters of attack scenarios

Scenario	**Legitimate traffic (cps)**	**Number of malicious INVITE sent**
SN-1-1	10	500
SN-1-2	10	1500
SN-1-3	10	2500
SN-1-4	10	4000
SN-2-1	20	500
SN-2-2	20	1500
SN-2-3	20	2500
SN-2-4	20	4000
SN-3-1	40	500
SN-3-2	40	1500
SN-3-3	40	2500
SN-3-4	40	4000
SN-4-1	80	1500
SN-4-2	80	1500
SN-4-3	80	1500
SN-4-4	80	1500

Chapter 8

Conclusions and Future Directions

8.1 Conclusions

The proliferation of VoIP services, and especially those based on SIP is expected to significantly augment in the years to come. However, SIP needs to confront several security issues mainly due to its open and text-oriented nature. In this direction, researchers are seeking novel proposals that are able to promptly identify security breaches and apply effective methods of control.

All in all, based on the objectives that have been introduced in Chapter 1, this PhD thesis offers an arsenal of statistical and ML-driven mechanisms to battle against SIP-based application layer DDoS attacks. A side contribution of the PhD thesis lies on the prevention of SDP-based malformed messages and the creation of C&C channels in SDP. Table 8.1 presents an overview of the achievements of the PhD thesis at hand, with respect to the literature.

Precisely, a first contribution of this PhD thesis builds over the idea proposed in [129], that is, the use of Entropy theory to detect abnormalities in raw application data. Specifically, through extensive experimentation, we extend, calibrate and throughly assess the effectiveness of the initial idea, thus offering a complete formalized framework that can be used to trace and detect application layer DDoS attacks in SIP networks. The proposed solution can be used to assess and certify the security level of VoIP provider with reference to application layer DoS attacks, based on the logged traffic without trampling on end-user's privacy. We assert that our framework exhibits several advantages over

TABLE 8.1: Overall PhD Thesis Contribution

Objective	Chapter	Contribution	Publication
Obj. 1	5	A Privacy-Preserving Entropy-Driven Framework for Tracing DoS Attacks in VoIP	[129]
Obj. 1	5	Exposing resource consumption attacks in internet multimedia services	[128]
Obj. 1	5	An efficient and easily deployable method for dealing with DoS in SIP services	[34]
Obj. 2	6	Battling against DDoS in SIP: Is Machine Learning-based detection an effective weapon?	[143]
Obj. 2	6	Realtime DDoS Detection in SIP Ecosystems: Machine Learning Tools of the Trade	[144]
Obj. 3	7	Hidden in plain sight. SDP-Based covert channel for botnet communication	[186]
Obj. 3	7	The Devil is in the Detail: SDP-Driven Malformed Message Attacks and Mitigation in SIP Ecosystems	[43]

those in earlier work. That is, it is lightweight, practical, privacy-preserving, and retains full compatibility with the SIP standard operating effectively in both offline and realtime fashion. It also presents a high degree of flexibility (through the tuning of its parameters) depending on the everyday VoIP traffic each provider has to cope with. Chapter 5 along with [129], [128], and [34] present in detail the solutions build on statistical schemes. These contributions fulfill the requirements that have been set with respect to the first objective (i.e., Obj. 1) of this PhD thesis.

In network intrusion detection, a typical method for exposing attacks is by tracking the network activity for possible deviations from the normal profile. That is, any discrepancy from a previously learned normal profile is identified as suspicious. This procedure is usually done using methods borrowed from the machine learning realm. So far, this potential have been examined in the literature in a great extend. However, as discussed in Chapter 4, in the case of VoIP in general and SIP in particular, works on this topic are not only scarce but also incomplete. To fill this striking gap, an important contribution of this PhD thesis is devoted to the assessment of the power of ML-based techniques, to identify DDoS incidents that capitalize on the use of SIP signaling. We consider 5 different popular ML detectors and a plethora of realistically simulated SIP traffic scenarios representing different flavors of DDoS. The results indicate that specific classifiers present

high accuracy even in cases of low-rate DoS attacks. The best results for DDoS are obtained for the classifier introducing the maximum overhead, and thus accuracy may at a hefty price. To grab a better understanding of the effectiveness of this kind of detection, we compare the obtained results against those generated by two other anomaly-based methods, namely Entropy and Hellinger Distance. From this comparison one can safely argue that ML techniques appreciably surpass non-machine learning ones in terms of FN and up to a certain extend in terms of FP.

In the same direction, another contribution of this PhD thesis lies in the applicability of several well-known ML techniques in detecting application layer DDoS attacks on the fly. That is, we implemented a Kamailio software module to achieve attack detection in realtime and tested both the detection accuracy and processing overhead of each classifier under a significant number of different attack scenarios representing both DoS and DDoS incidents. All the algorithms but one achieved desirable detection accuracy in terms of FP, but only mediocre accuracy (for a real IDS product) when the FN metric is considered. Therefore, further experimentation is needed to better appreciate this potential. From a processing overhead viewpoint, it can be safely argued that realtime operation is feasible as the induced time penalty is negligible even if the SIP server operates in single-thread mode. Chapter 6 along with [143] and [144] detail on the contributions which lie in the use of ML techniques and contribute toward addressing the second objective of this PhD thesis (i.e., Obj. 2).

This PhD thesis also elaborates on the exploitation of SIP as a covert channel for building botnet C&C. We demonstrate that with little effort an aggressor is able to tinker with SDP data contained in SIP requests aiming to convey spurious information secretly. This is also done in an straightforward and simple way, perfectly in line with SIP/SDP standards, and without raising any suspicions or causing the messages to be dropped by the receiving end as malformed. From a network defense viewpoint, little can be done; the messages seem completely legitimate, they are sent only sporadically and do not augment the network traffic significantly. So, even deep and continuous packet inspection at the application or other layer would not reveal something suspicious. The only effective counteraction is to monitor SIP transactions for requests without a matching response. But on the other hand, this mismatch occurs for legitimate SIP transactions quite often too, thus it is to be assumed that it will cause a high false alarm rate at the proxy-side IDS. Overall, we argue that the simplest and more innocuous the covert channel the

less the possibility of detecting it. Additionally, we provide results about the feasibility of such a C&C deployment by implementing two kinds of flooding attacks executed by bots.

Media negotiation comprises a fundamental operation in SIP ecosystems. However, the text nature of SDP creates a fertile ground for attackers aiming to launch DoS attacks by crafting the message bodies. So far, research has concentrated on malformed SIP message headers, largely neglecting SDP bodies. Motivated by this fact, a last contribution of this PhD thesis, is not only to demonstrate the magnitude of SDP-powered attacks to real-life SIP equipment and the user, but also to offer a straightforward to implement and fully compatible with SIP proxy-based method to solve it. The obtained results show that our solution comes at a negligible cost in terms of service time provision while (as with any misuse detection system) its detection accuracy depends solely on the freshness and quality of the implemented rules. Also, it is directly extensible by simply contributing additional rules to one or more of its categories pertaining to SDP message body filtering policies. These policies can be enforced or disabled by the administrator at will. Also, as explained in Chapter 7, and more specifically in section 7.5, each new rule added to the list, affects linearly the parser performance. That is, in terms of algorithmic time complexity, the search operation will always be equal to *O(N)* across the number of rules included in the currently enabled policies. Chapter 7 along with [186] and [43] detail on SDP-driven attacks. These contributions are the cornerstones toward fulfilling the last objective of this PhD thesis (i.e., Obj. 3).

8.2 Future directions

This PhD thesis provided solid clues for the applicability and efficiency of a set of both statistical methods and ML techniques regarding the detection of application layer DDoS attacks in SIP. Additionally, it elaborated on the prevention of SDP-based malformed messages and C&C channels, using misuse detection and filtering policies. Still, a number of research fields remain open for future work.

- In the case of log file analysis, open issues are towards providing more robust data anonymization schemes for use, especially in cases where the exchange of log files among providers or between a provider and a data analysis center is deemed necessary. Particularly, focusing on offline analysis, one may be interested in schemes that offer unlinkability along with anonymity. The employment of machine learning techniques on the audit trail data as a second more comprehensive layer of analysis is another direction worthy of investigation in a future work.

- Regarding the Hellinger Distance statistical scheme, one can expand the solution to work in realtime by inspecting each incoming message on-the-fly. This requires the development of a software module destined to SIP proxies. The applicability of the results of this work to similar services and protocols are also of great interest.

- Regarding ML techniques, a future direction involves extensive experimentation with the *Mw* parameter in an effort to better assess its overall effect on the detection process. The last one pertains to the evaluation of more advanced classifiers regarding its ability to cope with application layer DDoS attacks.

- With respect to SDP-driven malformed messages, the pointers for future work are related to (a) calibrating the parser and populating it with more rules, (b) testing the parser in other application domains, including Web RealTime Communication (WebRTC) environments that leverage SIP as their signaling protocol, and (c) implementing a dual-layer solution, which in addition to rule-based detection, will employ machine learning techniques to cope with previously unseen instances of SDP malicious alterations.

- Another future direction pertains to the enhancement of the SDP-driven C&C with more options for the botmaster. An idea is to find a way to dynamically change

the pattern of communication, that is, the places (message headers or descriptors) where the bytes with special meaning are placed. This way, the detection of the C&C channel would become even harder. On the downside, upon change, this pattern must be communicated to the bot population. A second interesting issue to consider in this context is the possibility of botnet partitioning. That is, in view of what has been discussed in the latter half of section 3.4, having some bots registered to a given public provider and the rest to another one(s). In this case, every bot needs to be informed about which Registrar (domain or IP) must register with, and the botmaster needs to keep and update a list of *{bot-username, Registrar-domain-name}* for being able to correctly dispatch the SIP INVITE messages. This naturally implies an extension of the covert channel to communicate a "Registrar shift" message to the bots.

- Finally, the battle against DDoS attacks in SIP-based NG9-1-1 emergency calling centers remain one of the most challenging and timely topics of research in VoIP, which however is not efficiently tackled by this PhD thesis. These services need to operate on a 24/7 basis, and thus the challenge in this case is related to the classification between attack and normal traffic taking into account the legislation requirements. A detailed analysis of this research problem is provided in the next paragraphs.

Emergency calling services consist one of the most critical components of public safety. The last years, many vendors extended their product portfolio to include emergency calling solutions. Such solutions are mostly targeted on Session Border Controller (SBC), Emergency Services Routing Proxy (ESRP), and/or Public Safety Answering Point (PSAP) solutions. The types of attacks presented in Chapter 3 of this PhD thesis, can be proven to be the single point of failure for emergency calling centers. Thus, the vendors need to design and develop solutions which should classify on the fly the calls entering the ESInet. The fulfillment of this requirement becomes even more challenging in NG9-1-1 environments, which should accept voice, text, and video calls.

Lately, a number of researches [241], [242], [243] has highlighted the need for increased security in NG9-1-1 environments. A major problem in this area has to do with the different media which may be negotiated by the various network entities. For example, if two network entities do not support the same codecs, then the proxy, in this case a

emergency calling center entity, should trigger a transcoding operation. Assuming that an attacker has already compromised a big number of telephony devices (i.e, an army of zombies), then a quite slow-rate and thus silent attack in terms of bandwidth could be launched in order to drain the victim's resources.

Figure 8.1 provides an overview of the SIP-based Emergency services architecture. With reference to the figure, it becomes apparent that emergency services provide an increased attack surface. That is, ESInets receive a plethora of SIP messages which are initiated via a number of different types of interconnected trunks. In this direction, SBC deployments comprise the external guards of the SIP-based emergency calling centers. The latter entities perform the main filtering and classification of SIP messages which enter the ESInet. The message labeling is usually performed with respect to spam and simple application layer DDoS attacks. In this context, more advanced IDS solutions should be installed in order to combat (a) application layer DDoS attacks, (b) SDP-driven malformed messages and, (c) SDP-based C&C channels.

The diversity of the interconnected trunks highlights the significance and the variety of traffic that an emergency calling deployment can receive. Additionally, it gives an overview of the different types of bots that an attacker could compromise in order to launch a more advanced DDoS attack. That is, figure 8.1 depicts an army of infected machines (zombies), which parasitize in different types of networks. After a DDoS attack is launched by the botmaster, the generated attack traffic is directed to the ESInet.

According to suggestions from well-known vendors in the field, including IBM, security should be designed and build into the devices [244]. In this respect, a security-by-design approach should be followed, with the aim to eliminate application layer DDoS attacks launched by botnets [245]. Lately, this approach is also presented in a number of researches [242].

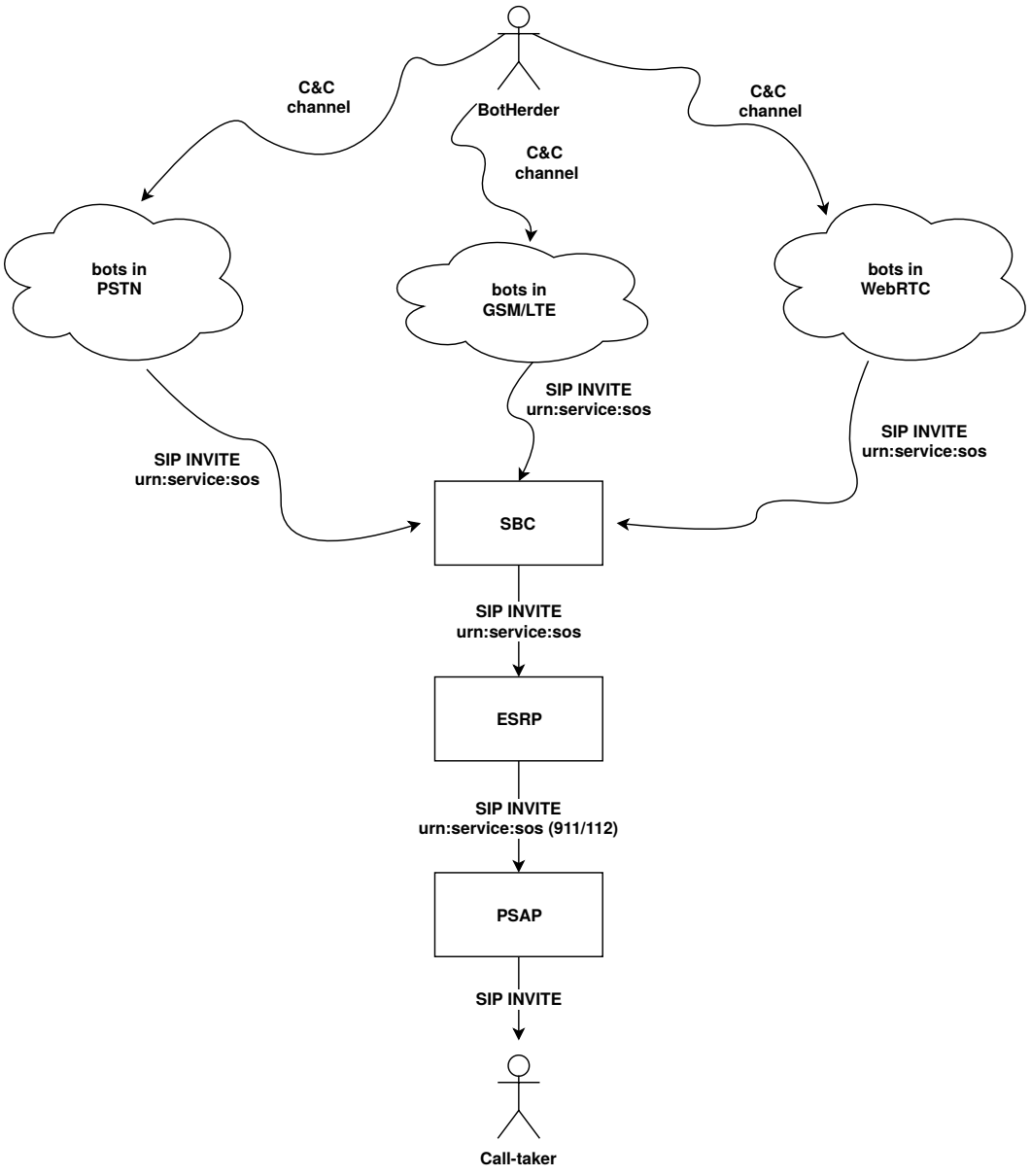

FIGURE 8.1: DDoS attacks in the ESInet

Appendix A

Pseudocodes: Statistical means, ML techniques, SDP parser

A.1 Pseudocodes for the entropy method

The next 5 pseudocodes describe the fundamental operations of the entropy method.

Algorithm 1: EntropyDetectionModule/RealtimeAnalysis

Input: SIPmessage

Output: MessageClassification

AlocateMemory(*HashTable*) ;
$NAD \leftarrow Avg(AID)$;
$SipHeaders \leftarrow SplitSIPMessage(SIPMessage)$;
while *(*$SipHeaders \neq NULL$*)* **do**
    $H_{S_i} \leftarrow$ **HMAC-SHA256**$(SipHeaders)$;
    InsertToHashTable(H_{S_i})
end
$AID \leftarrow ComputeMetricsHashTable(SipMessage, HashTable)$;
if *AID is greater than* $(NAD + \delta)$ **then**
    $print(ALERT - AttackMessage)$
else
    $print(LegitimateMessage)$
end

Algorithm 2: ComputeMetrics/OfflineAnalysis

Input: DataFile

Output: EntropyFile

OpenToRead(DataFile);
$EntropySumArray[S1, S2, S3, S4, S5, S6]$;
$i \leftarrow 1$;
$SymbolCounter \leftarrow 1$;
$SUM \leftarrow 0$;
while *(DataFile* $\neq$ *EOF)* **do**
    if ***Search***$(H_{S_i}$, *DataFile)* **then**
        $Temp \leftarrow readLine$ /*Locate "Ap" in line*/;
        /*Ap(HSi) : Appearances of HSi*/;
        Read(Ap);
        $Prob(H_{S_i}) \leftarrow (Ap/N)$;
        /*N : overall number of occurences*/ ;
        Write $I(H_{S_i})$ in EntropyFile;
        /*I(HSi):Itself Information of HSi*/ ;
        $SUM+ \leftarrow I(H_{S_i})$;
        $EntropySumArray[S_i]+ \leftarrow Prob(H_{S_i}) * I(H_{S_i})$;
    end
    if ***symbolCounter*** *equals* 4 ***or*** 6 **then**
        $ActualInfo(CurrentMessage) \leftarrow SUM$;
        $symbolCounter \leftarrow 1$;
        $SUM \leftarrow 0$;
        $i \leftarrow 1$;
    end
end

Algorithm 3: ComputeSymbolFrequency/OfflineAnalysis

Input: TrafficFile

Output: DataFile

```
1  i ← 1;
2  symbolCounter ← 1;
3  OpenToRead(TrafficFile);
4  while TrafficFile ≠ EOF do
5      OSi ← readLine;
6      /*OSi : Original Symbol*/;
7      if Search(O_Si, HeaderSymbol) or Search(O_Si, symbol[s1 − s6]) then
8          H_Si ← HMAC-SHA256(O_Si);
9          /*HSi : Hash Symbol*/;
10         symbolCounter+ ← 1;
11         InsertToHashTable(H_Si);
12         /*For insertToHashTable see algorithm 4*/
13         Write(Hash(O_Si) : H_Si);
14     else
15         if symbolCounter equals 4 or 6 then
16             i ← 1
17             symbolCounter ← 1
18         end
19     end
20 end
21 OpenToWrite(TrafficFile);
22 while TrafficFile ≠ EOF do
23     if SearchLine equals H_Si then
24         if LookUpHashTable(HashTable) ≠ 0 then
25             Write(H_Si : Ap :: LookUpHashTable(HashTable, H_Si))
26         end
27     end
28 end
```

Algorithm 4: InsertToHashTable

Input: SIPHeader

Output: UpdateHashTable

```
1 if LookUpHashTable(SIPHeader, HashTable) equals 0 then
2 |   Insert(SIPHeader, 0)
3 else
4 |   Insert(SIPHeader, LookUpHashTable(SIPHeader, HashTable) + 1)
5 end
```

Algorithm 5: Timestamps/OfflineAnalysis

Input: TrafficFile
Output: EntropyFile/Interval

```
1 Interal ← userChoice;
2 timestamp ← Read(TrafficFile);
3 while (timestamp ≠ timestamp + Interval) do
4 |   TraficFile ← InsertToFile(HMAC-SHA256(SIPHeader))
5 end
6 ComputeMetrics(ComputeSymbolFrequency(TrafficFile))
```

A.2 Pseudocodes for the Hellinger Distance method

The following 2 pseudocodes describe the fundamental operations of the Hellinger distance method with respect to the training and the detection phases.

Algorithm 6: Obtain Theoretical Messages

Input: Segmented-Attack-Free-File

Output: TheoreticalMessages

```
1  Normalization;
2  while (SegmentedFile ≠ NULL) do
3  |  Line ← ReadLine();
4  |  if Line is equal to FirstLine then
5  |  |  TypeOfMessage = ExtractTypeOfMessage();
6  |  |  TypeOfMessageCounter++;
7  |  else
8  |  |  Occurences ← ExtractOccurences(Line);
9  |  |  switch(TypeOfMessage);
10 |  |  TheoreticalMessages[TypeOfMessage][NumberOfHeader++]+ ← Occurences;
11 |  end
12 end
13 TheoreticalMessages[TypeOfMessage][NumberOfHeader]/ ← TypeOfMessageCounter;
14 while (TheoreticalMessages[TypeOfMessage][SipHeaders] ≠ NULL) do
15 |  Occurences ← ExtractOccurences(SipHeaders);
16 |  Normalization+ ← Occurences;
17 end
18 THeoreticalMessages[TypeOfMessage][SipHeaders]/Normalization;
```

Algorithm 7: Compute Hellinger Distance

Input: TheoreticalMessages, ExaminedMessage
Output: Hellinger Distance

```
1  DistributionEx;
2  NormalizationEx;
3  SipHeaders ← ExtractSipHeaders(ExaminedMessage);
4  while (SipHeaders ≠ NULL) do
5  |  Occurences ← ExtractOccurences(SipHeaders);
6  |  NormalizationEx+ ← Occurences;
7  end
8  DistributionEx ← Occurences(SipHeaders)/NormalizationEx;
9  while (SipHeaders ≠ NULL) do
10 |  Sum+ ← (sqrt(DistributionEx[SipHeaders]) −
   |    sqrt(THeoreticalMessages[Type(Examinedmessage)][SipHeaders]))^2;
11 end
12 HellingerDistance ← 0.5 * Sum
```

A.3 Pseudocodes for the ML techniques

The following 2 pseudocodes describe the fundamental operations of the ML techniques with regards to the training and the classification phases.

Algorithm 8: Obtain Input Data for ML Classifiers

Input: Audit Trail
Output: Input File for Classifiers (.arff format)

```
1  while (AuditTrail ≠ NULL) do
2  |  Line ← ReadLine();
3  |  SIPHeader ← ExtractSipHeader(Line);
4  |  HashedHeader ← HMAC-SHA256(SIPHeader);
5  |  if (InsertToHashTable(HashedHeader) ≠ NULL) then
6  |  |  GetValueofHashTable(HashedHeader)++;
7  |  else
8  |  |  InsertToHashTable(HashedHeader);
9  |  |  SetValueInHashTable(HashedHeader) ← 1;
10 |  end
11 |  if (Message-Window = 1,000) then
12 |  |  TotalMessages ← TotalMessages + Mw;
13 |  |  Re-Initialize(HashTable);
14 |  end
15 |  for (i=1; i ≤ TotalMessages; i++) do
16 |  |  PrintOccurences(GetValueofHashTable(HashedHeader));
17 |  end
18 end
```

Algorithm 9: Extraction of classification features

Input: Incoming SIP messages
Output: Classification result

```
1  SIPHeaders[N] ← ExtractSipHeader(SIP Request);
2  for (i=1; i ≤ N; i++) do
3  |  HashedHeader[i] ← HMAC-SHA256(SIPHeaders[i]);
4  |  if (InsertToHashTable(HashedHeaders[i]) ≠ NULL) then
5  |  |  GetValueofHashTable(HashedHeader[i])++;
6  |  else
7  |  |  InsertToHashTable(HashedHeaders[i]);
8  |  |  SetValueInHashTable(HashedHeader[i]) ← 1;
9  |  end
10 end
11 classificationResult ← classify(#HashedHeaders[1],... ,#HashedHeaders[N]);
12 if (Mw = 1,000) then
13 |  TotalMessages ← TotalMessages + Mw;
14 |  Re-Initialize(HashTable);
15 end
```

A.4 Pseudocode for the SDP parser

The next pseudocode describes the main logic of the SDP parser filtering operation.

Algorithm 10: Filtering process in SDP parser

Input: SIP message
Output: SDP segment classification

```
1  for RuleSet i ← 1 to k do
2      if RuleSet i is enabled then
3          for Rule j ← 1 to N do
4              if Rule j applies on SDPsegment then
5                  Drop message;
6                  return;
7              end
8          end
9      end
10 end
11 Forward message to the proxy;;
```

Appendix B

SDP Parser rules

This appendix contains the full list of the rules that have been implemented in our SDP parser until now.

Filtering Categories (policies):

- Filter MUST descriptors.
- Filter NOT RECOMMENDED descriptors.
- Filter SHOULD NOT descriptors.
- Filter GENERAL INCONSISTENCIES.

Filtering Sub-policies:

- Examine the presence and location of mandatory descriptors in the session level of SDP message. The included rules can be further categorized into 3 sub-policies, namely "MUST APPEAR", "MUST NOT APPEAR", and "MUST HAVE".
- Examine the number of elements per descriptor.
- Examine the length and/or the type of the elements per descriptor.

Set of currently implemented rules in the SDP parser per policy and sub-policy

MUST

1. “ =” A space Must Not appear in the left side of the equal sign.
2. “= ” A space Must Not appear in the right side of the equal sign.
3. “ = ” A space Must Not appear in the left and the right side of the equal sign.
4. The “c=” descriptor Must appear.
5. The “s= ” descriptor Must appear at most once in the session level.
6. The “i= ” descriptor Must appear at most once in the session level.
7. The “u= ” descriptor Must appear at most once in the session level.
8. The “H261” codec Must appear for video media.
9. The “H264” codec Must appear for video media.
10. The “H264-RCD0” codec Must appear for video media.
11. The “H264-SVC” codec Must appear for video media.
12. The “DV” codec Must appear for video media.
13. The “u= ” descriptor Must appear before the first “m” one.
14. The “e= ” descriptor Must appear before the first “m” one.
15. The “H261” codec Must have a clock rate equal to 90000.
16. The “H264” codec Must have a clock rate equal to 90000.
17. The “H264-RCD0” codec Must have a clock rate equal to 90000.
18. The “H264-SVC” codec Must have a clock rate equal to 90000.
19. The “DV” codec Must have a clock rate equal to 90000.

SHOULD NOT & NOT RECOMMENDED

1. Permanent sessions Should Not be used.
2. Unbounded sessions Should Not be used.
3. “k=” descriptor is Not Recommended.
4. “b=X-” descriptor is Not Recommended.

GENERAL INCONSISTENCIES

1. RTP payload number “0” is used for audio.
2. No need for an “a=rtpmap:0” attribute.
3. No need for an “a=fmtp:0” attribute.
4. RTP payload number “1” is RESERVED. It cannot be used.
5. RTP payload number “2” is RESERVED. It cannot be used.
6. RTP payload number “3 to 18” is used only for audio.
7. No need for an “a=rtpmap:” attribute for RTP payloads with numbers “3-18”.
8. No need for an “a=fmtp:” attribute for RTP payloads with numbers “3-18”.
9. RTP payload numbers “19 to 24” are either RESERVED or UNASSIGNED. They cannot be used.
10. RTP payload numbers “25 to 26” are used for video.
11. No need for an “a=rtpmap:” attribute for RTP payloads with numbers “25-26”.
12. No need for an “a=fmtp:” attribute for RTP payloads with numbers “25-26”.
13. RTP payload number “27” is either RESERVED or UNASSIGNED. It cannot be used.
14. RTP payload numbers “28” is used for video.
15. No need for an “a=rtpmap:” attribute for RTP payloads with number “28”.

16. No need for an “a=fmtp:” attribute for RTP payloads with numbers “28”.
17. RTP payload numbers “29” is used for video.
18. No need for an “a=rtpmap:” attribute for RTP payloads with number “29”.
19. No need for an “a=fmtp:” attribute for RTP payloads with numbers “29”.
20. RTP payload numbers “30” is used for video.
21. No need for an “a=rtpmap:” attribute for RTP payloads with number “30”.
22. No need for an “a=fmtp:” attribute for RTP payloads with numbers “30”.
23. RTP payload numbers “33” is used for audio/video.
24. No need for an “a=rtpmap:” attribute for RTP payloads with number “33”.
25. No need for an “a=fmtp:” attribute for RTP payloads with numbers “33”.
26. RTP payload numbers “31” is used for audio/video.
27. No need for an “a=rtpmap:” attribute for RTP payloads with number “31”.
28. No need for an “a=fmtp:” attribute for RTP payloads with numbers “31”.
29. RTP payload numbers “32” is used for audio/video.
30. No need for an “a=rtpmap:” attribute for RTP payloads with number “32”.
31. No need for an “a=fmtp:” attribute for RTP payloads with numbers “32”.
32. RTP payload numbers “34” is used for audio/video.
33. No need for an “a=rtpmap:” attribute for RTP payloads with number “34”.
34. No need for an “a=fmtp:” attribute for RTP payloads with numbers “34”.
35. Codec “G7291” must be used for audio.
36. Codec “VC1” must be used for video.
37. RTP payload numbers “35 to 71” are either RESERVED or UNASSIGNED. They cannot be used.
38. RTP payload numbers “72 to 76” are either RESERVED or UNASSIGNED. They cannot be used.

39. RTP payload numbers “77 to 95” are either RESERVED or UNASSIGNED. They cannot be used.

40. RTP payload numbers “96 to 127” a=rtpmap is not contained in the m line.

41. RTP payload numbers “96 to 127” fmtp is not contained in the m line.

42. The sixth argument in the “o” must exist in the SIP headers.

43. Codec “VC1” must have 2 arguments.

44. Codec “G7221” must have 2 arguments.

45. Check if “o” line contains 6 arguments.

46. RTP payload numbers “>127” are out of range.

47. If the “VC1 fmtp” attribute is present, it Must contain one of the following profile values: “profile=0”, “profile=1”, “profile=2”.

48. If the “VC1 fmtp” attribute, is present, it Must contain one of the following level values: “level=1”, “level=2”, “level=3”.

49. Codec “VC1” must have a clock rate equal to 90000.

50. Codec “G7221” clock rate is either equal to “16000” or “32000”.

51. Codec “G7221” must have a clock rate equal to “24000” “32000” assuming a sample rate equal to “16000”.

52. Codec “G7221” must have a clock rate equal to “32000” “48000” assuming a sample rate equal to “32000”.

53. Codec “G7221” must have a bit rate which is multiple of 400.

54. Codec “G7221” must have one of the values mentioned in rules 50, 51, 52 for the clock rate. Otherwise the value is out of range.

55. The clock rate for codec “G7291” must be “8000”, “12000”, “14000”,“16000”, “18000”, “20000”, “22000”, “24000”, “26000”, “28000”, “30000” or “32000”.

56. The clock rate for codec “L20” must be “8000”, “11025”, “16000”,“22050”, “24000”, “32000”, “44100” or “48000”.

57. The clock rate for codec “L24” must be “8000”, “11025”, “16000”,“22050”, “24000”, “32000”, “44100” or “48000”.

58. A bigger number of “rtpmap” attributes than RTP payloads exists in the SIP body.

59. The “a=orient” attribute can receive the following values: “portrait”, “landscape”, “seascape”.

60. Check the transport protocol in the m line. It should be equal to one of the following values: “udp”, “RTP/AVP”, “RTP/SAVP”.

61. An m line should contain one of the following values: “audio”, “video”, “application”, “text”, “message”.

62. Problem in the start and or end time in the “t” line.

63. The length of the “t” line must be equal to 2.

64. The length of the “c” line must be equal to 3.

65. The length of the “v” line must be equal to 1.

66. The length of the “s” line must be equal to 1.

67. The first argument in the “c” line must be equal to “IN”.

68. The second argument in the “c” line must be equal to “IP4”.

69. The second argument in the “v” line must be equal to “0”.

70. The first argument in the “o” line must be equal to the username or equal to “-”.

71. The second argument in the “o” line must be 10 digits.

72. The third argument in the “o” line must be 10 digits.

73. The fourth argument in the “o” line must be equal to “IN”.

74. The fifth argument in the “o” line must be equal to “IP4”.

75. The first argument in the “t” line must be 10 digits.

76. The second argument in the “t” line must be 10 digits.

77. The port number must be between “1024” and “65535”.

Bibliography

[1] M. Handley, V. Jacobson, and C. Perkins. Sdp: Session description protocol. Internet Requests for Comments, July 2006. URL http://www.rfc-editor.org/rfc/rfc4566.txt.

[2] ClarityTel. VoIP Industry Statistics, 2019. URL https://www.claritytel.com/voip-industry-statistics/. Accessed on 12 April, 2019.

[3] Anthony Cox. Mobile voip users to reach 1 billion by 2017, or one in seven mobile subscribers, 2012. URL http://www.juniperresearch.com/viewpressrelease.php?pr=355. Accessed on 12 April, 2019.

[4] IBISWorld. Voip in the us: Market research report, 2013. URL http://www.ibisworld.com/industry/default.aspx?indid=1269. Accessed on 12 April, 2019.

[5] C. K. Yeo, S. C. Hui, I. Y. Soon, and L. M. Ang. H.323 compliant voice over ip system. *Int. J. Comput. Appl. Technol.*, 16(4):143–153, July 2003. ISSN 0952-8091. doi: 10.1504/IJCAT.2003.000321.

[6] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler. Sip: Session initiation protocol. Internet Requests for Comments, June 2002. ISSN 2070-1721. URL http://www.rfc-editor.org/rfc/rfc3261.txt.

[7] F. Andreasen and B. Foster. Media gateway control protocol. Internet Requests for Comments, Jan 2003. URL https://www.rfc-editor.org/info/rfc3435.

[8] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. Hypertext transfer protocol – http/1.1. Internet Requests for Comments, March 1999. URL http://www.rfc-editor.org/rfc/rfc2616.txt.

[9] Sven Ehlert, Dimitris Geneiatakis, and Thomas Magedanz. Survey of network security systems to counter sip-based denial-of-service attacks. volume 29, pages 225 – 243, 2010. doi: 10.1016/j.cose.2009.09.004.

[10] Dimitris Geneiatakis, Tasos Dagiuklas, Georgios Kambourakis, Costas Lambrinoudakis, Stefanos Gritzalis, Sven Ehlert, and Dorgham Sisalem. Survey of security vulnerabilities in session initiation protocol. volume 8, pages 68–81, 2006. doi: 10.1109/COMST.2006.253270.

[11] Angelos D. Keromytis. A comprehensive survey of voice over ip security research. *IEEE Communications Surveys & Tutorials*, 14(2):514–537, 2012. doi: 10.1109/SURV.2011.031611.00112.

[12] Dimitris Geneiatakis, Georgios Kambourakis, Costas Lambrinoudakis, Tasos Dagiuklas, and Stefanos Gritzalis. A framework for protecting a sip-based infrastructure against malformed message attacks. *Comput. Netw.*, 51(10):2580–2593, July 2007. ISSN 1389-1286. doi: 10.1016/j.comnet.2006.11.014.

[13] Sven Ehlert, Ge Zhang, Dimitris Geneiatakis, Georgios Kambourakis, Tasos Dagiuklas, Jiří Markl, and Dorgham Sisalem. Two layer denial of service prevention on sip voip infrastructures. *Comput. Commun.*, 31(10):2443–2456, June 2008. ISSN 0140-3664. doi: 10.1016/j.comcom.2008.03.016.

[14] Dongwon Seo, Heejo Lee, and Ejovi Nuwere. Sipad: Sip–voip anomaly detection using a stateful rule tree. *Computer Communications*, 36(5):562 – 574, 2013. ISSN 0140-3664. doi: https://doi.org/10.1016/j.comcom.2012.12.004. URL http://www.sciencedirect.com/science/article/pii/S0140366412004124.

[15] Jens Fiedler, Tomas Kupka, Sven Ehlert, Thomas Magedanz, and Dorgham Sisalem. Voip defender: Highly scalable sip-based security architecture. In *Proceedings of the 1st International Conference on Principles, Systems and Applications of IP Telecommunications*, IPTComm '07, pages 11–17, New York, NY, USA, 2007. ACM. ISBN 978-1-60558-006-7. doi: 10.1145/1326304.1326307.

[16] Jin Tang and Yu Cheng. *Intrusion Detection for IP-Based Multimedia Communications over Wireless Networks*. Springer Publishing Company, Incorporated, 2013. ISBN 1461489954, 9781461489955.

[17] Marianne. Swanson and Barbara Guttman. *Generally accepted principles and practices for securing information technology systems [electronic resource]*. National Institute of Standards and Technology, Technology Administration, U.S. Dept. of Commerce, 1996.

[18] Paul Ohm. Broken promises of privacy: Responding to the surprising failure of anonymization. Technical Report ID 1450006, Social Science Research Network, Rochester, NY, July 2012.

[19] Omer Tene. Privacy: The new generations. SSRN Scholarly Paper ID 1710688, Social Science Research Network, Rochester, NY, November 2010.

[20] Latanya Sweeney. Uniqueness of Simple Demographics in the U.S. Population. *LIDAP-WP4, Carnegie Mellon University, Laboratory for International Data Privacy, Pittsburgh, 2000*, 1000.

[21] Philippe Golle. Revisiting the uniqueness of simple demographics in the us population. In *Proceedings of the 5th ACM workshop on Privacy in electronic society*, WPES '06, pages 77–80, New York, NY, USA, 2006. ACM. ISBN 1-59593-556-8. doi: 10.1145/1179601.1179615.

[22] Andreas Pfitzmann and Marit Hansen. A terminology for talking about privacy by data minimization: Anonymity, unlinkability, undetectability, unobservability, pseudonymity, and identity management, 2009.

[23] F. Pereniguez, R. Marin-Lopez, G. Kambourakis, S. Gritzalis, and A.F. Gomez. Privakerb: A user privacy framework for kerberos. volume 30, pages 446 – 463, 2011. doi: 10.1016/j.cose.2011.04.001.

[24] Dimitris Geneiatakis, Georgios Kambourakis, C Lambrinoudakis, T Dagiuklas, and S Gritzalis. Sip message tampering: The sql code injection attack. In *Proceedings of SoftCOM 2005, Split, Croatia*, 2005.

[25] D. Geneiatakis, G. Kambourakis, C. Lambrinoudakis, T. Dagiuklas, and S. Gritzalis. A framework for protecting a sip-based infrastructure against malformed message attacks. *Communications Networks, Elsevier*, 51(10):2580–2593, 2007.

[26] D. Geneiatakis, T. Dagiuklas, G. Kambourakis, C. Lambrinoudakis, S. Gritzalis, K.S. Ehlert, and D. Sisalem. Survey of security vulnerabilities in session initiation

protocol. *IEEE Communications Surveys & Tutorial*, 8(3):68–81, rd 2006. ISSN 1553-877X. doi: 10.1109/COMST.2006.253270.

[27] Georgios Kambourakis, Constantinos Kolias, Stefanos Gritzalis, and Jong Hyuk Park. Dos attacks exploiting signaling in UMTS and IMS. *Computer Communications*, 34(3):226 – 235, 2011. ISSN 0140-3664. doi: http://dx.doi.org/10.1016/j.comcom.2010.02.010. URL http://www.sciencedirect.com/science/article/pii/S014036641000085X.

[28] Angelos D. Keromytis. *Voice over IP Security - A Comprehensive Survey of Vulnerabilities and Academic Research.*, volume 1 of *Springer Briefs in Computer Science*. Springer, 2011. ISBN 978-1-4419-9866-8.

[29] M. Shtern, R. Sandel, M. Litoiu, C. Bachalo, and V. Theodorou. Towards mitigation of low and slow application ddos attacks. In *Cloud Engineering (IC2E), 2014 IEEE International Conference on*, pages 604–609, March 2014. doi: 10.1109/IC2E.2014.38.

[30] Dimitris Geneiatakis, Nikos Vrakas, and Costas Lambrinoudakis. Utilizing bloom filters for detecting flooding attacks against SIP based services. *Computers & Security*, 28(7):578–591, 2009. doi: 10.1016/j.cose.2009.04.007. URL http://dx.doi.org/10.1016/j.cose.2009.04.007.

[31] Hemant Sengar, Haining Wang, Duminda Wijesekera, and Sushil Jajodia. Detecting voip floods using the hellinger distance. *IEEE Trans. Parallel Distrib. Syst.*, 19(6):794–805, June 2008. ISSN 1045-9219. doi: 10.1109/TPDS.2007.70786. URL http://dx.doi.org/10.1109/TPDS.2007.70786.

[32] J. Tang, Y. Cheng, Y. Hao, and W. Song. Sip flooding attack detection with a multi-dimensional sketch design. *IEEE Transactions on Dependable and Secure Computing*, 11(6):582–595, 2014.

[33] Zisis Tsiatsikas, Marios Anagnostopoulos, Georgios Kambourakis, Sozon Lambrou, and Dimitris Geneiatakis. Hidden in plain sight. sdp-based covert channel for botnet communication. In *TrustBus*, pages 48–59, 2015.

[34] Zisis Tsiatsikas, Dimitris Geneiatakis, Georgios Kambourakis, and Angelos D. Keromytis. An efficient and easily deployable method for dealing with dos in sip

services. *Computer Communications*, 57:50 – 63, 2015. ISSN 0140-3664. doi: https://doi.org/10.1016/j.comcom.2014.11.002. URL http://www.sciencedirect.com/science/article/pii/S014036641400348X.

[35] R. Sparks, Ed. A. Hawrylyshen, A. Johnston, J. Rosenberg, and H. Schulzrinne. Session initiation protocol (sip) torture test messages. Internet Requests for Comments, May 2006. URL http://www.rfc-editor.org/rfc/rfc4475.txt.

[36] Hazewinkel, Michiel. Hellinger distance, Encyclopedia of Mathematics. URL https://www.encyclopediaofmath.org/. Accessed on 12 April, 2019.

[37] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. Rtp: A transport protocol for real-time applications. Internet Requests for Comments, July 2003. URL http://www.rfc-editor.org/rfc/rfc3550.txt.

[38] Takuya Sawada and Paul Kyzivat. Session Initiation Protocol (SIP) Usage of the Offer/Answer Model. RFC 6337, August 2011. URL https://rfc-editor.org/rfc/rfc6337.txt.

[39] Emil Ivov. Jitsi. URL https://jitsi.org/. Accessed on 12 April, 2019.

[40] Jin Tang and Yu Cheng. *SIP Stealthy Attack Detection and Resource-Drained Malformed Message Attack Detection*, pages 71–86. Springer New York, New York, NY, 2013. ISBN 978-1-4614-8996-2. doi: 10.1007/978-1-4614-8996-2_5. URL https://doi.org/10.1007/978-1-4614-8996-2_5.

[41] S. Marchal, A. Mehta, V. K. Gurbani, R. State, T. Kam Ho, and F. Sancier-Barbosa. Mitigating mimicry attacks against the session initiation protocol. *IEEE Transactions on Network and Service Management*, 12(3):467–482, Sept 2015. ISSN 1932-4537. doi: 10.1109/TNSM.2015.2459603.

[42] Mahsa Hosseinpour, Mohammad Hossein Yaghmaee, Seyed Amin Hosseini Seno, Hossein Khosravi Roshkhari, and Mohsen Asadi. Anomaly-based dos detection and prevention in sip networks by modeling sip normal traffic. *International Journal of Communication Systems*, 0, 2018.

[43] Z. Tsiatsikas, G. Kambourakis, D. Geneiatakis, and H. Wang. The devil is in the detail: Sdp-driven malformed message attacks and mitigation in sip ecosystems.

IEEE Access, 7:2401–2417, 2019. ISSN 2169-3536. doi: 10.1109/ACCESS.2018.2886356.

[44] Hong Zhao and Xueying Zhang. Sip steganalysis using chaos theory. In *IEEE CMCSN 2012*, pages 95–100, July 2012. doi: 10.1109/CMCSN.2012.25.

[45] Wojciech Mazurczyk. Lost audio packets steganography: The first practical evaluation. *Sec. and Commun. Netw.*, 5(12):1394–1403, December 2012. ISSN 1939-0114. doi: 10.1002/sec.502. URL http://dx.doi.org/10.1002/sec.502.

[46] S.E. Coull, F. Monrose, M.K. Reiter, and M. Bailey. The challenges of effectively anonymizing network data. In *Conference For Homeland Security, 2009. CATCH '09. Cybersecurity Applications Technology*, pages 230–236, March 2009. doi: 10.1109/CATCH.2009.27.

[47] Rinku Dewri, Indrajit Ray, Indrakshi Ray, and Darrell Whitley. Exploring privacy versus data quality trade-offs in anonymization techniques using multi-objective optimization. *Journal of Computer Security*, 19(5):935–974, September 2011. ISSN 0926-227X.

[48] Angelos D. Keromytis. Voice-over-ip security: Research and practice. *IEEE Security & Privacy*, 8(2):76–78, 2010. ISSN 1540-7993. doi: http://doi.ieeecomputersociety.org/10.1109/MSP.2010.87.

[49] Elena Gabriela Barrantes, David H. Ackley, Trek S. Palmer, Darko Stefanovic, and Dino Dai Zovi. Randomized instruction set emulation to disrupt binary code injection attacks. In *Proceedings of the 10th ACM Conference on Computer and Communications Security*, CCS '03, pages 281–289, New York, NY, USA, 2003. ACM. ISBN 1-58113-738-9. doi: 10.1145/948109.948147.

[50] P. Samarati. Protecting respondents' identities in microdata release. *IEEE Trans. on Knowl. and Data Eng.*, 13(6):1010–1027, November 2001. ISSN 1041-4347. doi: 10.1109/69.971193.

[51] Xiaokui Xiao and Yufei Tao. Anatomy: Simple and effective privacy preservation. In *Proceedings of the 32Nd International Conference on Very Large Data Bases*, VLDB '06, pages 139–150. VLDB Endowment, 2006.

[52] Xianmang He, Yanghua Xiao, Yujia Li, Qing Wang, Wei Wang, and Baile Shi. Permutation anonymization: Improving anatomy for privacy preservation in data publication. In *Proceedings of the 15th International Conference on New Frontiers in Applied Data Mining*, PAKDD'11, pages 111–123, Berlin, Heidelberg, 2012. Springer-Verlag. ISBN 978-3-642-28319-2. doi: 10.1007/978-3-642-28320-8_10.

[53] Latanya Sweeney. K-anonymity: A model for protecting privacy. *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.*, 10(5):557–570, October 2002. ISSN 0218-4885. doi: 10.1142/S0218488502001648.

[54] K. Seppanen. Method, apparatus and computer program for anonymization of identification data, November 20 2008. URL http://www.google.com/patents/US20080287118. US Patent App. 12/168,041.

[55] Reza Curtmola, Juan Garay, Seny Kamara, and Rafail Ostrovsky. Searchable symmetric encryption: Improved definitions and efficient constructions. In *Proceedings of the 13th ACM Conference on Computer and Communications Security*, CCS '06, pages 79–88, New York, NY, USA, 2006. ACM. ISBN 1-59593-518-5. doi: 10.1145/1180405.1180417.

[56] Naren Ramakrishnan, Benjamin J. Keller, Batul J. Mirza, Ananth Y. Grama, and George Karypis. Privacy risks in recommender systems. volume 5, pages 54–62, Piscataway, NJ, USA, November 2001. IEEE Educational Activities Department. doi: 10.1109/4236.968832.

[57] Schneier Bruce. Why 'Anonymous' data sometimes isn't. URL https://www.schneier.com/essay-200.html. Accessed on 12 April, 2019.

[58] Elaine Shi, Emil Stefanov, and Charalampos Papamanthou. Practical dynamic proofs of retrievability. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*, CCS '13, pages 325–336, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-2477-9. doi: 10.1145/2508859.2516669. URL http://doi.acm.org/10.1145/2508859.2516669.

[59] Emil Stefanov, Marten van Dijk, Elaine Shi, Christopher Fletcher, Ling Ren, Xiangyao Yu, and Srinivas Devadas. Path oram: An extremely simple oblivious ram protocol. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*, CCS '13, pages 299–310, New York, NY,

USA, 2013. ACM. ISBN 978-1-4503-2477-9. doi: 10.1145/2508859.2516660. URL http://doi.acm.org/10.1145/2508859.2516660.

[60] SIPVicious. Sipvicious. URL http://blog.sipvicious.org/. Accessed on 12 April, 2019.

[61] Jason E. Holt. Logcrypt: Forward security and public verification for secure audit logs, 2005.

[62] Bruce Schneier and John Kelsey. Cryptographic support for secure logs on untrusted machines. In *Proceedings of the 7th Conference on USENIX Security Symposium - Volume 7*, SSYM'98, pages 4–4, Berkeley, CA, USA, 1998. USENIX Association.

[63] Andreas Mitrakas. Assessing liability arising from information security breaches in data privacy. *International Data Privacy Law*, 2011. doi: 10.1093/idpl/ipr001.

[64] European Commission. Directive on privacy and electronic communication - 2002/58/ec, 2002. URL http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=CONSLEG:2002L0058:20091219:EN:PDF.

[65] D. Dolev and Andrew C. Yao. On the security of public key protocols. *Information Theory, IEEE Transactions on*, 29(2):198–208, Mar 1983. ISSN 0018-9448. doi: 10.1109/TIT.1983.1056650.

[66] Dimitris Geneiatakis, Nikos Vrakas, and Costas Lambrinoudakis. Utilizing bloom filters for detecting flooding attacks against sip based services. *Computers & Security*, 28(7):578–591, 2009.

[67] Ge Zhang, Sven Ehlert, Thomas Magedanz, and Dorgham Sisalem. Denial of service attack and prevention on sip voip infrastructures using dns flooding. In *Proceedings of the 1st International Conference on Principles, Systems and Applications of IP Telecommunications*, IPTComm '07, pages 57–66, New York, NY, USA, 2007. ACM. ISBN 978-1-60558-006-7. doi: 10.1145/1326304.1326314.

[68] G. Kambourakis. Anonymity and closely related terms in the cyberspace: An analysis by example. *Journal of Information Security and Applications, Elsevier.*, 2014. doi: dx.doi.org/10.1016/j.jisa.2014.04.001.

[69] Emmanouil Vasilomanolakis, Jan Helge Wolf, Leon Böck, Shankar Karuppayah, and Max Mühlhäuser. I trust my zombies: A trust-enabled botnet. *CoRR*, abs/1712.03713, 2017. URL http://arxiv.org/abs/1712.03713.

[70] Philippe Biondi. Scapy. URL http://www.secdev.org/projects/scapy/. Accessed on 12 April, 2019.

[71] Cisco Security Advisory. Cisco sip phone 3905 resource limitation denial of service vulnerability. URL https://tools.cisco.com/security/center/content/CiscoSecurityAdvisory/cisco-sa-20151202-sip. Accessed on 12 April, 2019.

[72] D. Geneiatakis, T. Dagiuklas, G. Kambourakis, C. Lambrinoudakis, S. Gritzalis, et al. Survey of security vulnerabilities in session initiation protocol. *IEEE Communications Surveys & Tutorials*, 8(3):68–81, 2006. ISSN 1553-877X. doi: 10.1109/COMST.2006.253270.

[73] Sip service providers and carriers, 2015. URL http://www.cs.columbia.edu/sip/service-providers.html. Accessed on 12 April, 2019.

[74] Marios Anagnostopoulos, Georgios Kambourakis, Panagiotis Kopanos, Georgios Louloudakis, and Stefanos Gritzalis. Dns amplification attack revisited. *Computers & Security*, 39:475–485, November 2013. ISSN 0167-4048.

[75] Raquel C.G. Pinto Sergio S.C. Silva, Rodrigo M.P. Silva and Ronaldo M. Salles. Botnets: A survey. *Computer Networks*, 57(2):378 – 403, 2013. ISSN 1389-1286.

[76] Ping Wang, Lei Wu, Baber Aslam, and Cliff Changchun Zou. A systematic study on peer-to-peer botnets. In *IEEE ICCCN 2009*, pages 1–8, Aug. 2009.

[77] Ping Wang, Sherri Sparks, and Cliff Changchun Zou. An advanced hybrid peer-to-peer botnet. *Dependable and Secure Computing, IEEE Transactions on*, 7(2): 113–127, 2010.

[78] R. Sekar, A. Gupta, J. Frullo, T. Shanbhag, A. Tiwari, H. Yang, and S. Zhou. Specification-based anomaly detection: A new approach for detecting network intrusions. In *Proceedings of the 9th ACM Conference on Computer and Communications Security*, CCS '02, pages 265–274, New York, NY, USA, 2002. ACM. ISBN 1-58113-612-9. doi: 10.1145/586110.586146. URL http://doi.acm.org/10.1145/586110.586146.

[79] Robert Mitchell and Ing-Ray Chen. A survey of intrusion detection in wireless network applications. *Computer Communications*, 42:1 – 23, 2014. ISSN 0140-3664. doi: https://doi.org/10.1016/j.comcom.2014.01.012. URL http://www.sciencedirect.com/science/article/pii/S0140366414000280.

[80] I. Butun, S. D. Morgera, and R. Sankar. A survey of intrusion detection systems in wireless sensor networks. *IEEE Communications Surveys & Tutorials*, 16(1): 266–282, First 2014. ISSN 1553-877X. doi: 10.1109/SURV.2013.050113.00191.

[81] 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):18 – 28, 2009. ISSN 0167-4048. doi: https://doi.org/10.1016/j.cose.2008.08.003. URL http://www.sciencedirect.com/science/article/pii/S0167404808000692.

[82] 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.

[83] J. Zhang and M. Zulkernine. Anomaly based network intrusion detection with unsupervised outlier detection. In *2006 IEEE International Conference on Communications*, volume 5, pages 2388–2393, June 2006. doi: 10.1109/ICC.2006.255127.

[84] Sandeep Kumar and Eugene H. Spafford. A pattern matching model for misuse intrusion detection. In *In Proceedings of the 17th National Computer Security Conference*, pages 11–21, 1994.

[85] Sandeep Kumar and Eugene H. Spafford. An application of pattern matching in intrusion detection, 1994.

[86] 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.

[87] Joachim Biskup and Ulrich Flegel. Threshold-based identity recovery for privacy enhanced applications. In *Proceedings of the 7th ACM conference on Computer and communications security*, CCS '00, pages 71–79, New York, NY, USA, 2000. ACM. ISBN 1-58113-203-4. doi: 10.1145/352600.352611.

[88] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *Advances in Cryptology — CRYPTO' 86*, pages 186–194, Berlin, Heidelberg, 1987. Springer Berlin Heidelberg. ISBN 978-3-540-47721-1.

[89] Adam Sah. A new architecture for managing enterprise log data. In *Proceedings of the 16th USENIX Conference on System Administration*, LISA '02, pages 121–132, Berkeley, CA, USA, 2002. USENIX Association. URL http://dl.acm.org/citation.cfm?id=1050517.1050531.

[90] Ulrich Flegel. Pseudonymizing unix log files. In George Davida, Yair Frankel, and Owen Rees, editors, *Infrastructure Security*, pages 162–179, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg. ISBN 978-3-540-45831-9.

[91] John P. Rouillard. Real-time log file analysis using the simple event correlator (sec). In *LISA*, 2004.

[92] Dingbang Xu and Peng Ning. Alert correlation through triggering events and common resources. In *Proceedings of the 20th Annual Computer Security Applications Conference*, ACSAC '04, pages 360–369, Washington, DC, USA, 2004. IEEE Computer Society. ISBN 0-7695-2252-1. doi: 10.1109/CSAC.2004.5.

[93] Patrick Lincoln, Phillip Porras, and Vitally Shmatikov. Privacy-preserving sharing and correction of security alerts. In *Proceedings of the 13th Conference on USENIX Security Symposium - Volume 13*, SSYM'04, pages 17–17, Berkeley, CA, USA, 2004. USENIX Association. URL http://dl.acm.org/citation.cfm?id=1251375.1251392.

[94] A. Slagell and W. Yurcik. Sharing computer network logs for security and privacy: a motivation for new methodologies of anonymization. In *Workshop of the 1st International Conference on Security and Privacy for Emerging Areas in Communication Networks, 2005.*, pages 80–89, Sept 2005. doi: 10.1109/SECCMW.2005.1588299.

[95] Fernando Godínez, Dieter Hutter, and Raúl Monroy. On the role of information compaction to intrusion detection. In Félix F. Ramos, Victor Larios Rosillo, and Herwig Unger, editors, *Advanced Distributed Systems*, pages 83–97, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[96] Vassilios Stathopoulos, Panayiotis Kotzanikolaou, and Emmanouil Magkos. A framework for secure and verifiable logging in public communication networks. In Javier Lopez, editor, *Critical Information Infrastructures Security*, pages 273–284, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg. ISBN 978-3-540-69084-9.

[97] Christopher P. Lee and John A. Copeland. Flowtag: A collaborative attack-analysis, reporting, and sharing tool for security researchers. In *Proceedings of the 3rd International Workshop on Visualization for Computer Security*, VizSEC '06, pages 103–108, New York, NY, USA, 2006. ACM. ISBN 1-59593-549-5. doi: 10.1145/1179576.1179597. URL http://doi.acm.org/10.1145/1179576.1179597.

[98] Eoghan Casey. Investigating sophisticated security breaches. *Commun. ACM*, 49 (2):48–55, February 2006. ISSN 0001-0782. doi: 10.1145/1113034.1113068. URL http://doi.acm.org/10.1145/1113034.1113068.

[99] Víctor H. Escobar-Jeria, María J. Martín-Bautista, Daniel Sánchez, and María-Amparo Vila. Analysis of log files applying mining techniques and fuzzy logic. In Hiroshi G. Okuno and Moonis Ali, editors, *New Trends in Applied Artificial Intelligence*, pages 483–492, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[100] Mohamed Saleh, Ali Reza Arasteh, Assaad Sakha, and Mourad Debbabi. Forensic analysis of logs: Modeling and verification. *Know.-Based Syst.*, 20(7):671–682, October 2007. ISSN 0950-7051. doi: 10.1016/j.knosys.2007.05.002. URL http://dx.doi.org/10.1016/j.knosys.2007.05.002.

[101] Steena D. S. Monteiro and Robert F. Erbacher. An authentication and validation mechanism for analyzing syslogs forensically. *SIGOPS Oper. Syst. Rev.*, 42(3): 41–50, April 2008. ISSN 0163-5980. doi: 10.1145/1368506.1368513. URL http://doi.acm.org/10.1145/1368506.1368513.

[102] Roger M. Needham and Michael D. Schroeder. Using encryption for authentication in large networks of computers. *Commun. ACM*, 21(12):993–999, December 1978. ISSN 0001-0782. doi: 10.1145/359657.359659. URL http://doi.acm.org/10.1145/359657.359659.

[103] R. Sedgewick. *Algorithms in C*. Addison-Wesley series in computer science and information processing. Addison-Wesley Publishing Company, 1990. ISBN 9780201514254. URL https://books.google.gr/books?id=t0UPAQAAMAAJ.

[104] Ashvin Goel, Kamran Farhadi, Kenneth Po, and Wu-chang Feng. Reconstructing system state for intrusion analysis. *SIGOPS Oper. Syst. Rev.*, 42(3):21–28, April 2008. ISSN 0163-5980. doi: 10.1145/1368506.1368511. URL http://doi.acm.org/10.1145/1368506.1368511.

[105] Ying Xia, Kevin Fairbanks, and Henry Owen. A program behavior matching architecture for probabilistic file system forensics. *SIGOPS Oper. Syst. Rev.*, 42 (3):4–13, April 2008. ISSN 0163-5980. doi: 10.1145/1368506.1368509. URL http://doi.acm.org/10.1145/1368506.1368509.

[106] K. I. Pun and Y. W. Si. Audit trail analysis for traffic intensive web application. In *2009 IEEE International Conference on e-Business Engineering*, pages 577–582, Oct 2009. doi: 10.1109/ICEBE.2009.91.

[107] Araceli Barradas-Acosta, Eleazar Aguirre, Mariko Nakano-Miyatake, and Hector Perez-Meana. Neural network based attack detection algorithm. *WSEAS Transactions on Computers*, 8:905–915, 01 2009.

[108] Justin Myers, Michael R. Grimaila, and Robert F. Mills. Towards insider threat detection using web server logs. In *Proceedings of the 5th Annual Workshop on Cyber Security and Information Intelligence Research: Cyber Security and Information Intelligence Challenges and Strategies*, CSIIRW '09, pages 54:1–54:4, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-518-5. doi: 10.1145/1558607.1558670. URL http://doi.acm.org/10.1145/1558607.1558670.

[109] M. Nagappan. Analysis of execution log files. In *2010 ACM/IEEE 32nd International Conference on Software Engineering*, volume 2, pages 409–412, May 2010. doi: 10.1145/1810295.1810405.

[110] Eduardo Mazza, Marie-Laure Potet, and Daniel Le Métayer. A formal framework for specifying and analyzing logs as electronic evidence. In Jim Davies, Leila Silva, and Adenilso Simao, editors, *Formal Methods: Foundations and Applications*, 2011.

[111] Kyu Hyung Lee, Xiangyu Zhang, and Dongyan Xu. Loggc: Garbage collecting audit log. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer Communications Security*, CCS '13, pages 1005–1016, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-2477-9. doi: 10.1145/2508859.2516731.

[112] Ting-Fang Yen, Alina Oprea, Kaan Onarlioglu, Todd Leetham, William Robertson, Ari Juels, and Engin Kirda. Beehive: Large-scale log analysis for detecting suspicious activity in enterprise networks. In *Proceedings of the 29th Annual Computer Security Applications Conference*, ACSAC '13, pages 199–208, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-2015-3. doi: 10.1145/2523649.2523670. URL http://doi.acm.org/10.1145/2523649.2523670.

[113] J. King. Measuring the forensic-ability of audit logs for nonrepudiation. In *2013 35th International Conference on Software Engineering (ICSE)*, pages 1419–1422, May 2013. doi: 10.1109/ICSE.2013.6606732.

[114] Nanhay Singh, Achin Jain, Ram Shringar Raw, and Rahul Raman. Detection of web-based attacks by analyzing web server log files. In Durga Prasad Mohapatra and Srikanta Patnaik, editors, *Intelligent Computing, Networking, and Informatics*, pages 101–109, New Delhi, 2014. Springer India. ISBN 978-81-322-1665-0.

[115] Z. Gu, K. Pei, Q. Wang, L. Si, X. Zhang, and D. Xu. Leaps: Detecting camouflaged attacks with statistical learning guided by program analysis. In *2015 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, pages 57–68, June 2015. doi: 10.1109/DSN.2015.34.

[116] Amruta Ambre and Narendra Shekokar. Insider threat detection using log analysis and event correlation. *Procedia Computer Science*, 45:436 – 445, 2015. ISSN 1877-0509. doi: https://doi.org/10.1016/j.procs.2015.03.175. URL http://www.sciencedirect.com/science/article/pii/S1877050915004184. International Conference on Advanced Computing Technologies and Applications (ICACTA).

[117] Antti Juvonen, Tuomo Sipola, and Timo Hämäläinen. Online anomaly detection using dimensionality reduction techniques for http log analysis. *Computer Networks*, 91:46 – 56, 2015. ISSN 1389-1286. doi: https://doi.org/10.1016/j.comnet.2015.07.019. URL http://www.sciencedirect.com/science/article/pii/S1389128615002650.

[118] M. Moh, S. Pininti, S. Doddapaneni, and T. Moh. Detecting web attacks using multi-stage log analysis. In *2016 IEEE 6th International Conference on Advanced Computing (IACC)*, pages 733–738, Feb 2016. doi: 10.1109/IACC.2016.141.

[119] Jakub Breier and Jana Branišová. A dynamic rule creation based anomaly detection method for identifying security breaches in log records. *Wireless Personal Communications*, 94(3):497–511, Jun 2017. ISSN 1572-834X. doi: 10.1007/s11277-015-3128-1. URL https://doi.org/10.1007/s11277-015-3128-1.

[120] W. Taylor. Change-point analysis: A powerful new tool for detecting changes, 2000. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.36.8387.

[121] Z. Chen, W. Wen, and D. Yu. Detecting sip flooding attacks on ip multimedia subsystem (ims). In *2012 International Conference on Computing, Networking and Communications (ICNC)*, pages 154–158, 2012.

[122] Emmanouela Stachtiari, Yannis Soupionis, Panagiotis Katsaros, Anakreontas Mentis, and Dimitris Gritzalis. Probabilistic model checking of captcha admission control for dos resistant anti-spit protection. In Bernhard M. Hämmerli, Nils Kalstad Svendsen, and Javier Lopez, editors, *Critical Information Infrastructures Security*, pages 143–154, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[123] K. Dassouki, H. Safa, and A. Hijazi. End to end mechanism to protect sip from signaling attacks. In *2014 6th International Conference on New Technologies, Mobility and Security (NTMS)*, pages 1–5, March 2014. doi: 10.1109/NTMS.2014.6814017.

[124] J. Peterson. Session initiation protocol (sip) authenticated identity body (aib) format. Internet Requests for Comments, September 2004. URL http://www.rfc-editor.org/rfc/rfc3893.txt.

[125] R. H. M. Zargar and M. H. Y. Moghaddam. An entropy-based voip flooding attacks detection and prevention system. In *2014 4th International Conference on Computer and Knowledge Engineering (ICCKE)*, pages 691–696, Oct 2014. doi: 10.1109/ICCKE.2014.6993385.

[126] Jin Tang, Yu Cheng, and Yong Hao. Detection and prevention of sip flooding attacks in voice over ip networks. In *INFOCOM, 2012 Proceedings IEEE*, pages 1161–1169, March 2012. doi: 10.1109/INFCOM.2012.6195475.

[127] Jonghan Lee, Kyumin Cho, ChangYong Lee, and Seungjoo Kim. Voip-aware network attack detection based on statistics and behavior of sip traffic. *Peer-to-Peer Networking and Applications*, 8(5):872–880, Sep 2015.

[128] Zisis Tsiatsikas, Georgios Kambourakis, and Dimitris Geneiatakis. Exposing resource consumption attacks in internet multimedia services. In *proceedings of 14th IEEE International Symposium on Signal Processing and Information Technology (ISSPIT)*, Security Track, pages 1–6. IEEE Press, 2014.

[129] Z. Tsiatsikas, D. Geneiatakis, G. Kambourakis, and A.D. Keromytis. A privacy-preserving entropy-driven framework for tracing dos attacks in voip. In *Availability, Reliability and Security (ARES), 2013 Eighth International Conference on*, pages 224–229, Sept 2013. doi: 10.1109/ARES.2013.30.

[130] M. Semerci, M. Yamaç, A. T. Cemgil, B. Sankur, and D. S. Coşar. Detection of (d)dos attacks based on online change point analysis. In *2016 24th Signal Processing and Communication Application Conference (SIU)*, pages 1161–1164, May 2016. doi: 10.1109/SIU.2016.7495951.

[131] Barış Kurt, Çağatay Yıldız, Taha Yusuf Ceritli, Bülent Sankur, and Ali Taylan Cemgil. A bayesian change point model for detecting sip-based ddos attacks. *Digital Signal Processing*, 2017. ISSN 1051-2004. doi: https://doi.org/10.1016/j.dsp.2017.10.009. URL http://www.sciencedirect.com/science/article/pii/S1051200417302361.

[132] D. Golait and N. Hubballi. Voipfd: Voice over ip flooding detection. In *2016 Twenty Second National Conference on Communication (NCC)*, pages 1–6, March 2016. doi: 10.1109/NCC.2016.7561121.

[133] Khaled Dassouki, Haidar Safa, Mohamed Nassar, and Abbas Hijazi. Protecting from cloud-based sip flooding attacks by leveraging temporal and structural fingerprints. *Computers Security*, 70:618 – 633, 2017. ISSN 0167-4048. doi: https://doi.org/10.1016/j.cose.2017.08.003. URL http://www.sciencedirect.com/science/article/pii/S016740481730158X.

[134] D. Golait and N. Hubballi. Detecting anomalous behavior in voip systems: A discrete event system modeling. *IEEE Transactions on Information Forensics and Security*, 12(3):730–745, 2017.

[135] Mingli Wu, Na Ruan, Shiheng Ma, Haojin Zhu, Weijia Jia, Qingshui Xue, and Songyang Wu. Detect sip flooding attacks in volte by utilizing and compressing counting bloom filter. In Liran Ma, Abdallah Khreishah, Yan Zhang, and

Mingyuan Yan, editors, *Wireless Algorithms, Systems, and Applications*, pages 124–135, Cham, 2017. Springer International Publishing.

[136] Murat Semerci, Ali Taylan Cemgil, and Bülent Sankur. An intelligent cyber security system against ddos attacks in sip networks. *Computer Networks*, 136:137 – 154, 2018. ISSN 1389-1286. doi: https://doi.org/10.1016/j.comnet.2018.02.025. URL http://www.sciencedirect.com/science/article/pii/S1389128618300987.

[137] R. Ferdous, R. Lo Cigno, and A. Zorat. Classification of sip messages by a syntax filter and svms. In *2012 IEEE Global Communications Conference (GLOBECOM)*, pages 2714–2719, Dec 2012. doi: 10.1109/GLOCOM.2012.6503527.

[138] A. Mehta, N. Hantehzadeh, V. K. Gurbani, T. K. Ho, and F. Sander. On using multiple classifier systems for session initiation protocol (sip) anomaly detection. In *2012 IEEE International Conference on Communications (ICC)*, pages 1101–1106, 2012.

[139] R. Ferdous, R. L. Cigno, and A. Zorat. On the use of svms to detect anomalies in a stream of sip messages. In *2012 11th International Conference on Machine Learning and Applications*, volume 1, pages 592–597, 2012.

[140] Muhammad Ali Akbar and Muddassar Farooq. Securing sip-based voip infrastructure against flooding attacks and spam over ip telephony. *Knowledge and Information Systems*, 38(2):491–510, Feb 2014. ISSN 0219-3116. doi: 10.1007/s10115-012-0595-5. URL https://doi.org/10.1007/s10115-012-0595-5.

[141] M Ali Akbar and Muddassar Farooq. Application of evolutionary algorithms in detection of sip based flooding attacks. In *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, pages 1419–1426. ACM, 2009.

[142] Mahdi Mohammadi, Ahmad Akbari, Bijan Raahemi, Babak Nassersharif, and Hassan Asgharian. A fast anomaly detection system using probabilistic artificial immune algorithm capable of learning new attacks. *Evolutionary Intelligence*, 6(3): 135–156, Feb 2014.

[143] Z. Tsiatsikas, A. Fakis, D. Papamartzivanos, D. Geneiatakis, G. Kambourakis, and C. Kolias. Battling against ddos in sip: Is machine learning-based detection an

effective weapon? In *2015 12th International Joint Conference on e-Business and Telecommunications (ICETE)*, volume 04, pages 301–308, 2015.

[144] Zisis Tsiatsikas, Dimitris Geneiatakis, Georgios Kambourakis, and Stefanos Gritzalis. Realtime ddos detection in sip ecosystems: Machine learning tools of the trade. In Jiageng Chen, Vincenzo Piuri, Chunhua Su, and Moti Yung, editors, *Network and System Security*, pages 126–139, Cham, 2016. Springer International Publishing.

[145] Miroslav Vozňák and Jakub Safarik. Dos attacks targeting sip server and improvements of robustness. 6:177–184, 2012.

[146] E. Maccherani, M. Femminella, J. W. Lee, R. Francescangeli, J. Janak, G. Reali, and H. Schulzrinne. Extending the netserv autonomic management capabilities using openflow. In *2012 IEEE Network Operations and Management Symposium*, pages 582–585, April 2012. doi: 10.1109/NOMS.2012.6211961.

[147] P. Rodnikorn, N. Elz, and L. Preechaveerakul. Sipe-sap: Sip extension for signaling attacks protection. In *2012 Fourth International Conference on Ubiquitous and Future Networks (ICUFN)*, pages 296–300, July 2012. doi: 10.1109/ICUFN.2012.6261715.

[148] T. Rontti, A. M. Juuso, and A. Takanen. Preventing dos attacks in ngn networks with proactive specification-based fuzzing. *IEEE Communications Magazine*, 50 (9):164–170, 2012.

[149] D. Crocker, Ed. Brandenburg, and P. Overell. Augmented bnf for syntax specifications: Abnf. Internet Requests for Comments, Jan 2008. URL http://www.rfc-editor.org/rfc/rfc5234.txt.

[150] Jan Stanek and Lukas Kencl. Sip protector: Defense architecture mitigating ddos flood attacks against sip servers. In *ICC*, pages 6733–6738. IEEE, 2012. ISBN 978-1-4577-2052-9.

[151] J. Stanek and L. Kencl. Sipp-dd: Sip ddos flood-attack simulation tool. In *Computer Communications and Networks (ICCCN), 2011 Proceedings of 20th International Conference on*, pages 1–7, July 2011. doi: 10.1109/ICCCN.2011.6005946.

[152] R. Farley and X. Wang. Voip shield: A transparent protection of deployed voip systems from sip-based exploits. In *2012 IEEE Network Operations and Management Symposium*, pages 486–489, April 2012. doi: 10.1109/NOMS.2012.6211937.

[153] A. Lahmadi and O. Festor. A framework for automated exploit prevention from known vulnerabilities in voice over ip services. *IEEE Transactions on Network and Service Management*, 9(2):114–127, June 2012. ISSN 1932-4537. doi: 10.1109/TNSM.2012.011812.110125.

[154] Zoha Asgharian, Hassan Asgharian, Ahmad Akbari, and Bijan Raahemi. Detecting denial of service message flooding attacks in sip based services. *AUT Journal of Electrical Engineering*, 44(1):75–85, 2012. ISSN 2588-2910. doi: 10.22060/eej.2012.44. URL http://eej.aut.ac.ir/article_44.html.

[155] Boris Beizer. *Software Testing Techniques*. International Thomson Computer Press, 1990. ISBN 9781850328803.

[156] Mohamed Nassar, Radu State, and Olivier Festor. Labeled voip data-set for intrusion detection evaluation. In Finn Arve Aagesen and Svein Johan Knapskog, editors, *Networked Services and Applications - Engineering, Control and Management*, pages 97–106, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg. ISBN 978-3-642-13971-0.

[157] Tom Fawcett. An introduction to roc analysis. *Pattern Recognition Letters*, 27(8):861 – 874, 2006. ISSN 0167-8655. doi: https://doi.org/10.1016/j.patrec.2005.10.010. URL http://www.sciencedirect.com/science/article/pii/S016786550500303X. ROC Analysis in Pattern Recognition.

[158] Nikos Vrakas and Costas Lambrinoudakis. An intrusion detection and prevention system for ims and voip services. *International Journal of Information Security*, 12(3):201–217, Jun 2013. ISSN 1615-5270. doi: 10.1007/s10207-012-0187-0. URL https://doi.org/10.1007/s10207-012-0187-0.

[159] M. A. Raza, A. u. R. Khan, and M. Raza. A restrictive model (rm) for detection and prevention of invite flooding attack. In *2013 3rd IEEE International Conference on Computer, Control and Communication (IC4)*, pages 1–6, Sept 2013. doi: 10.1109/IC4.2013.6653766.

[160] I. Hussain, S. Djahel, D. Geneiatakis, and F. Naït-Abdesselam. A lightweight countermeasure to cope with flooding attacks against session initiation protocol. In *6th Joint IFIP Wireless and Mobile Networking Conference (WMNC)*, pages 1–5, April 2013. doi: 10.1109/WMNC.2013.6549057.

[161] Byeong hee Roh, Ju Wan Kim, Ki-Yeol Ryu, and Jea-Tek Ryu. A whitelist-based countermeasure scheme using a bloom filter against sip flooding attacks. *Computers Security*, 37:46 – 61, 2013. ISSN 0167-4048. doi: https://doi.org/10.1016/j.cose.2013.04.001. URL http://www.sciencedirect.com/science/article/pii/S0167404813000771.

[162] E. Y. Chen and M. Itoh. A whitelist approach to protect sip servers from flooding attacks. In *2010 IEEE International Workshop Technical Committee on Communications Quality and Reliability (CQR 2010)*, pages 1–6, June 2010. doi: 10.1109/CQR.2010.5619917.

[163] G. Zhang and S. Fischer-Hübner. Counteract dns attacks on sip proxies using bloom filters. In *2013 International Conference on Availability, Reliability and Security*, pages 678–684, Sept 2013. doi: 10.1109/ARES.2013.89.

[164] Bosco Sebastian, Paromita Choudhury, and C. D. Jaidhar. Mechanism for preventing registration flooding attack in sip. In Durga Prasad Mohapatra and Srikanta Patnaik, editors, *Intelligent Computing, Networking, and Informatics*, pages 705–712, New Delhi, 2014. Springer India. ISBN 978-81-322-1665-0.

[165] Gaston Ormazabal, Sarvesh Nagpal, Eilon Yardeni, and Henning Schulzrinne. Secure sip: A scalable prevention mechanism for dos attacks on sip based voip systems. In *Principles, systems and applications of IP telecommunications. Services and security for next generation networks*, pages 107–132. Springer, 2008.

[166] D. R. Cox. Prediction by exponentially weighted moving averages and related methods. *Journal of the Royal Statistical Society. Series B (Methodological)*, 23(2): 414–422, 1961. ISSN 00359246. URL http://www.jstor.org/stable/2984031.

[167] S. Donovan and J. Rosenberg. Session timers in the session initiation protocol (sip). Internet Requests for Comments, April 2005. URL http://www.rfc-editor.org/rfc/rfc4028.txt.

[168] Z. Liu, X. Yin, and H. Lee. An efficient defense scheme against sip dos attack in sdn using cloud sfw. In *2014 Ninth Asia Joint Conference on Information Security*, pages 52–55, Sept 2014. doi: 10.1109/AsiaJCIS.2014.12.

[169] Arnold O. Allen. *Probability, Statistics, and Queueing Theory with Computer Science Applications*. Academic Press Professional, Inc., San Diego, CA, USA, 1990. ISBN 0-12-051051-0.

[170] Konrad Rieck, Stefan Wahl, Pavel Laskov, Peter Domschitz, and Klaus-Robert Müller. A self-learning system for detection of anomalous sip messages. In Henning Schulzrinne, Radu State, and Saverio Niccolini, editors, *Principles, Systems and Applications of IP Telecommunications. Services and Security for Next Generation Networks*, volume 5310 of *Lecture Notes in Computer Science*, pages 90–106. Springer Berlin Heidelberg, 2008. ISBN 978-3-540-89053-9. doi: 10.1007/978-3-540-89054-6_5.

[171] G. Vennila, M. S. K. Manikandan, and S. Aswathi. Detection of sip signaling attacks using two-tier fine grained model for voip. In *TENCON 2015 - 2015 IEEE Region 10 Conference*, pages 1–7, Nov 2015. doi: 10.1109/TENCON.2015.7372954.

[172] A. Bansal and A. R. Pais. Mitigation of flooding based denial of service attack against session initiation protocol based voip system. In *2015 IEEE International Conference on Computational Intelligence Communication Technology*, pages 391–396, Feb 2015. doi: 10.1109/CICT.2015.66.

[173] V. Jyothi, S. K. Addepalli, and R. Karri. Deep packet field extraction engine (dpfee): A pre-processor for network intrusion detection and denial-of-service detection systems. In *2015 33rd IEEE International Conference on Computer Design (ICCD)*, pages 266–272, Oct 2015. doi: 10.1109/ICCD.2015.7357113.

[174] A. Akbar, S. M. Basha, and S. A. Sattar. Leveraging the sip load balancer to detect and mitigate ddos attacks. In *2015 International Conference on Green Computing and Internet of Things (ICGCIoT)*, pages 1204–1208, Oct 2015. doi: 10.1109/ICGCIoT.2015.7380646.

[175] Bela Shah and Kinjal Dave. Sip based intrusion detection system for voip based applications. In *Proceedings of the Second International Conference on Information*

and Communication Technology for Competitive Strategies, ICTCS '16, pages 28:1–28:5, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-3962-9. doi: 10.1145/2905055.2905086. URL http://doi.acm.org/10.1145/2905055.2905086.

[176] Zahraa Sabra and Hassan Artail. Using group anonymity to hide the identity of voip mobile users communicating over hybrid networks while preserving quality of service. *Wireless Communications and Mobile Computing*, 16(17):2792–2808. doi: 10.1002/wcm.2725. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/wcm.2725.

[177] Robert C. Streijl, Stefan Winkler, and David S. Hands. Mean opinion score (mos) revisited: methods and applications, limitations and alternatives. *Multimedia Systems*, 22(2):213–227, Mar 2016. ISSN 1432-1882. doi: 10.1007/s00530-014-0446-1. URL https://doi.org/10.1007/s00530-014-0446-1.

[178] F. Cadet and D. T. Fokum. Coping with denial-of-service attacks on the ip telephony system. In *SoutheastCon 2016*, pages 1–7, March 2016. doi: 10.1109/SECON.2016.7506691.

[179] Martin Roesch. Snort - lightweight intrusion detection for networks. In *Proceedings of the 13th USENIX Conference on System Administration*, LISA '99, pages 229–238, Berkeley, CA, USA, 1999. USENIX Association. URL http://dl.acm.org/citation.cfm?id=1039834.1039864.

[180] Qibo Sun, Shangguang Wang, Ning Lu, Kok-Seng Wong, and Myung Ho Kim. Sfads: A sip flooding attack detection scheme with the internal and external detection features in ims networks. *Journal of Internet Technology*, 17(7):1327–1338, 2016.

[181] M. Hosseinpour, S. A. Hosseini Seno, M. H. Yaghmaee Moghaddam, and H. Khosravi Roshkhari. An anomaly based voip dos attack detection and prevention method using fuzzy logic. In *2016 8th International Symposium on Telecommunications (IST)*, pages 713–718, Sept 2016. doi: 10.1109/ISTEL.2016.7881916.

[182] K. Dassouki, H. Safa, A. Hijazi, and W. El-Hajj. A sip delayed based mechanism for detecting voip flooding attacks. In *2016 International Wireless Communications and Mobile Computing Conference (IWCMC)*, pages 588–593, Sep. 2016. doi: 10.1109/IWCMC.2016.7577123.

[183] Dongwon Seo, Heejo Lee, and Ejovi Nuwere. Detecting more sip attacks on voip services by combining rule matching and state transition models. In Sushil Jajodia, Pierangela Samarati, and Stelvio Cimato, editors, *Proceedings of The Ifip Tc 11 23rd International Information Security Conference*, pages 397–411, Boston, MA, 2008. Springer US. ISBN 978-0-387-09699-5.

[184] Sven Ehlert, Chengjian Wang, Thomas Magedanz, and Dorgham Sisalem. Specification-based denial-of-service detection for sip voice-over-ip networks. In *Proceedings of the 2008 The Third International Conference on Internet Monitoring and Protection*, ICIMP '08, pages 59–66, Washington, DC, USA, 2008. IEEE Computer Society. ISBN 978-0-7695-3189-2. doi: 10.1109/ICIMP.2008.14. URL https://doi.org/10.1109/ICIMP.2008.14.

[185] Miralem Mehić, Martin Mikulec, Miroslav Voznak, and Lukas Kapicak. Creating covert channel using sip. In Andrzej Dziech and Andrzej Czyżewski, editors, *Multimedia Communications, Services and Security*, pages 182–192, Cham, 2014. Springer International Publishing.

[186] Zisis Tsiatsikas, Marios Anagnostopoulos, Georgios Kambourakis, Sozon Lambrou, and Dimitris Geneiatakis. Hidden in plain sight. sdp-based covert channel for botnet communication. In Simone Fischer-Hübner, Costas Lambrinoudakis, and Javier López, editors, *Trust, Privacy and Security in Digital Business*, pages 48–59, Cham, 2015. Springer International Publishing.

[187] Steffen Wendzel, Wojciech Mazurczyk, and Sebastian Zander. Unified description for network information hiding methods. *CoRR*, abs/1512.07438, 2015. URL http://arxiv.org/abs/1512.07438.

[188] Steffen Wendzel and Carolin Palmer. Creativity in mind: Evaluating and maintaining advances in network steganographic research. *CoRR*, abs/1511.08507, 2015. URL http://arxiv.org/abs/1511.08507.

[189] P. L. Shrestha, M. Hempel, F. Rezaei, and H. Sharif. A support vector machine-based framework for detection of covert timing channels. *IEEE Transactions on Dependable and Secure Computing*, 13(2):274–283, 2016.

[190] Suricata. URL https://suricata-ids.org/. Accessed on 12 April, 2019.

[191] A. Lahmadi and O. Festor. Secsip: A stateful firewall for sip-based networks. In *2009 IFIP/IEEE International Symposium on Integrated Network Management*, pages 172–179, June 2009. doi: 10.1109/INM.2009.5188807.

[192] W. Conner and K. Nahrstedt. Protecting sip proxy servers from ringing-based denial-of-service attacks. In *2008 Tenth IEEE International Symposium on Multimedia*, pages 340–347, Dec 2008. doi: 10.1109/ISM.2008.65.

[193] H. Sengar, Haining Wang, D. Wijesekera, and S. Jajodia. Fast detection of denial-of-service attacks on ip telephony. In *Quality of Service, 2006. IWQoS 2006. 14th IEEE International Workshop on*, pages 199–208, June 2006. doi: 10.1109/IWQOS.2006.250469.

[194] Sven Ehlert, Ge Zhang, Dimitris Geneiatakis, Georgios Kambourakis, Tasos Dagiuklas, Jiří Markl, and Dorgham Sisalem. Two layer denial of service prevention on sip voip infrastructures. volume 31, pages 2443–2456, Amsterdam, The Netherlands, The Netherlands, June 2008. Elsevier Science Publishers B. V. doi: 10.1016/j.comcom.2008.03.016.

[195] John Ioannidis and Steven M. Bellovin. Implementing pushback: Router-based defense against ddos attacks. In *NDSS*. The Internet Society, 2002. ISBN 1-891562-13-4. URL http://dblp.uni-trier.de/db/conf/ndss/ndss2002.html#IoannidisB02.

[196] R. Mathew and V. Katkar. Survey of low rate dos attack detection mechanisms. In *Proceedings of the International Conference, Workshop on Emerging Trends in Technology*, ICWET '11, pages 955–958, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0449-8. doi: 10.1145/1980022.1980227.

[197] Jelena Mirkovic and Peter Reiher. D-ward: A source-end defense against flooding denial-of-service attacks. volume 2, pages 216–232, Los Alamitos, CA, USA, jul 2005. IEEE Computer Society Press. doi: 10.1109/TDSC.2005.35.

[198] George Oikonomou, Jelena Mirkovic, Peter Reiher, and Max Robinson. A framework for a collaborative ddos defense. In *Proceedings of the 22nd Annual Computer Security Applications Conference*, ACSAC '06, pages 33–42, Washington, DC, USA, 2006. IEEE Computer Society. ISBN 0-7695-2716-7. doi: 10.1109/ACSAC.2006.5.

[199] WillaK. Ehrlich, Kenichi Futamura, and Danielle Liu. An entropy based method to detect spoofed denial of service (dos) attacks. In S. Raghavan, Bruce Golden, and Edward Wasil, editors, *Telecommunications Modeling, Policy, and Technology*, volume 44 of *Operations Research/Computer Science Interfaces*, pages 101–122. Springer US, 2008. ISBN 978-0-387-77779-5.

[200] C. E. Shannon. A mathematical theory of communication. volume 27, 1948.

[201] National institute of standards and technology. FIPS PUB 198-1, The Keyed-Hash Message Authentication Code , Federal Information Processing Standard (FIPS). Technical report, DEPARTMENT OF COMMERCE, July 2002.

[202] Alissa Cooper, Marit Hansen, Rhys Smith, and Hannes Tschofenig. Privacy terminology and concepts. 2012.

[203] Geoff Kuenning and Ethan L. Miller. Anonymization techniques for urls and filenames. Technical report, 2003.

[204] Pedro O. S. Vaz De Melo, Leman Akoglu, Christos Faloutsos, and Antonio A. F. Loureiro. Surprising patterns for the call duration distribution of mobile phone users. In *Proceedings of the 2010 European conference on Machine learning and knowledge discovery in databases: Part III*, ECML PKDD'10, pages 354–369, Berlin, Heidelberg, 2010. Springer-Verlag. ISBN 3-642-15938-9, 978-3-642-15938-1.

[205] Daqiang Zhang, Athanasios V. Vasilakos, and Haoyi Xiong. Predicting location using mobile phone calls. volume 42, pages 295–296, New York, NY, USA, August 2012. ACM. doi: 10.1145/2377677.2377738.

[206] Guofei Gu, Prahlad Fogla, David Dagon, Wenke Lee, and Boris Skorić. Measuring intrusion detection capability: an information-theoretic approach. In *Proceedings of the 2006 ACM Symposium on Information, computer and communications security*, ASIACCS '06, pages 90–101, New York, NY, USA, 2006. ACM. ISBN 1-59593-272-0. doi: 10.1145/1128817.1128834.

[207] Kamailio – the open source sip server. URL http://www.kamailio.org/w/. Accessed on 12 April, 2019.

[208] D. Eastlake and T. Hansen. Us secure hash algorithms- (sha and sha-based hmac and hkdf), 2011.

[209] R. Krishnamurthy and G.N. Rouskas. Evaluation of sip proxy server performance: Packet-level measurements and queuing model. In *Communications (ICC), 2013 IEEE International Conference on*, pages 2326–2330, June 2013. doi: 10.1109/ICC.2013.6654877.

[210] Alefiya Naveed Hussain. *Measurement and spectral analysis of denial of service attacks*. PhD thesis, Los Angeles, CA, USA, 2005. AAI3196820.

[211] Paul Barford, Jeffery Kline, David Plonka, and Amos Ron. A signal analysis of network traffic anomalies. In *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment*, IMW '02, pages 71–82, New York, NY, USA, 2002. ACM. ISBN 1-58113-603-X. doi: 10.1145/637201.637210.

[212] Sven Ehlert. *Denial-of-service detection and mitigation for SIP communication networks*. PhD thesis, Berlin Institute of Technology, 2009. URL http://opus.kobv.de/tuberlin/volltexte/2010/2496/.

[213] 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.

[214] S. Tsang, B. Kao, K. Y. Yip, W. Ho, and S. D. Lee. Decision trees for uncertain data. In *2009 IEEE 25th International Conference on Data Engineering*, pages 441–444, March 2009. doi: 10.1109/ICDE.2009.26.

[215] S. . Horikawa, T. Furuhashi, and Y. Uchikawa. On fuzzy modeling using fuzzy neural networks with the back-propagation algorithm. *IEEE Transactions on Neural Networks*, 3(5):801–806, Sept 1992. ISSN 1045-9227. doi: 10.1109/72.159069.

[216] Zisis Tsiatsikas, Dimitris Geneiatakis, and Georgios Kambourakis. Research project scype: Software modules. URL https://scype.samos.aegean.gr/tzisis/scype_5179_software/. Accessed on 12 April, 2019.

[217] Ian H. Witten and Eibe Frank. *Data Mining: Practical Machine Learning Tools and Techniques, Second Edition (Morgan Kaufmann Series in Data Management Systems)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2005. ISBN 0120884070.

[218] Donald Eastlake and Tony Hansen. Us secure hash algorithms (sha and sha-based hmac and hkdf). Internet Requests for Comments, May 2011. URL https://www.rfc-editor.org/rfc/rfc6234.txt.

[219] Carrie Gates and Carol Taylor. Challenging the anomaly detection paradigm: A provocative discussion. In *Proceedings of the 2006 Workshop on New Security Paradigms*, NSPW '06, pages 21–29, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-923-4.

[220] Rob Gordon. Essential jni: Java native interface. Upper Saddle River, NJ, USA, 1998. Prentice-Hall, Inc. ISBN 0-13-679895-0.

[221] Oracle. Crashing jvm, 2016. URL http://docs.oracle.com/cd/E15289_01/doc.40/e15059/crash.htm#i1010768. Accessed on 12 April, 2019.

[222] SIPp. Free open source test tool / traffic generator for the sip protocol. URL http://sipp.sourceforge.net/index.html. Accessed on 12 April, 2019.

[223] Nils Ohlmeier. Sip swiss army knife. URL http://sipsak.org/. Accessed on 12 April, 2019.

[224] Ian H. Witten, Eibe Frank, and Mark A. Hall. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, Burlington, MA, 3 edition, 2011. ISBN 978-0-12-374856-0. URL http://www.sciencedirect.com/science/book/9780123748560.

[225] D. Geneiatakis, T. Dagiuklas, G. Kambourakis, C. Lambrinoudakis, S. Gritzalis, K. S. Ehlert, and D. Sisalem. Survey of security vulnerabilities in session initiation protocol. *IEEE Communications Surveys & Tutorials*, 8(3):68–81, rd 2006. ISSN 1553-877X.

[226] E. Y. Chen and M. Itoh. Scalable detection of sip fuzzing attacks. In *2008 Second International Conference on Emerging Security Information, Systems and Technologies*, pages 114–119, Aug 2008.

[227] D. Mills. Network time protocol (version 3) specification, implementation and analysis. Internet Requests for Comments, March 1992. URL http://www.rfc-editor.org/rfc/rfc1305.txt.

[228] Phelim O'Doherty and Mudumbai Ranganathan. JAIN SIP Tutorial - Serving the Developer Community. Technical report, 2003.

[229] Cisco Security Advisory. Cisco Meeting Server Session Description Protocol Media Lines Buffer Overflow Vulnerability. URL https://tools.cisco.com/security/center/content/CiscoSecurityAdvisory/cisco-sa-20161102-cms1. Accessed on 12 April, 2019.

[230] Cisco Security Advisory. Cisco Unified IP Phone 8900/9900 Series Crafted SDP Packet Vulnerability. URL https://tools.cisco.com/security/center/content/CiscoSecurityAdvisory/Cisco-SA-20131010-CVE-2013-5526. Accessed on 12 April, 2019.

[231] Asterisk Project Security Advisory. Buffer Overflow Exploit Through SIP SDP Header. URL http://downloads.asterisk.org/pub/security/AST-2013-001.html. accessed on 12 April, 2019.

[232] CVE Details - The ultimate security vulnerability datasource. Known SDP-driven vulnerabilities. https://www.cvedetails.com/google-search-results.php?q=session+description+protocol+%28SDP%29, Accessed on 12 April, 2019.

[233] NIST. Computer Security Resource Center - National Vulnerability Database. URL https://nvd.nist.gov/vuln/search/results?adv_search=false&form_type=basic&results_type=overview&search_type=all&query=session+description+protocol. Accessed on 12 April, 2019.

[234] H. Schulzrinne and S. Casner. Rtp profile for audio and video conferences with minimal control. Internet Requests for Comments, July 2003. URL http://www.rfc-editor.org/rfc/rfc3551.txt.

[235] How Does Mobile VoIP Work? URL https://www.voip-info.org/wiki/view/Mobile+VoIP. Accessed on 12 April, 2019.

[236] Matt Grech. 10 Simple But Powerful Small Business VoIP Phones for Under $100. URL https://getvoip.com/blog/2016/06/16/small-business-voip-phones/. Accessed on 12 April, 2019.

[237] ITU. G.711 : Pulse code modulation (pcm) of voice frequencies. URL https://www.itu.int/rec/T-REC-G.711/en. Accessed on 12 April, 2019.

[238] Asterisk. Open source communications software | asterisk official site. URL http://www.asterisk.org/. Accessed on 12 April, 2019.

[239] Emmanuel Proulx. An introduction to the jain sip api. URL http://www.oracle.com/technetwork/java/introduction-jain-sip-090386.html. Accessed on 12 April, 2019.

[240] David Lion, Adrian Chiu, Hailong Sun, Xin Zhuang, Nikola Grcevski, and Ding Yuan. Don't get caught in the cold, warm-up your JVM: Understand and eliminate JVM warm-up overhead in data-parallel systems. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, pages 383–400, Savannah, GA, 2016. USENIX Association. ISBN 978-1-931971-33-1. URL https://www.usenix.org/conference/osdi16/technical-sessions/presentation/lion.

[241] A. Zacchi, A. Goulart, and W. Magnussen. A framework for securing the signaling plane in the emergency services ip network (esinet). In *2011 IEEE Consumer Communications and Networking Conference (CCNC)*, pages 515–516, Jan 2011. doi: 10.1109/CCNC.2011.5766525.

[242] Mordechai Guri, Yisroel Mirsky, and Yuval Elovici. 9-1-1 ddos: Threat, analysis and mitigation. *CoRR*, abs/1609.02353, 2016. URL http://arxiv.org/abs/1609.02353.

[243] X. Lu and S. S. Huang. Malicious apps may explore a smartphone's vulnerability to detect one's activities. In *2017 IEEE 31st International Conference on Advanced Information Networking and Applications (AINA)*, pages 787–794, March 2017. doi: 10.1109/AINA.2017.49.

[244] Bret Greenstein. IoT devices used in DDoS attacks, 2016. URL https://www.ibm.com/blogs/internet-of-things/ddos-iot-platform-security/. Accessed on 12 April, 2019.

[245] C. Kolias, G. Kambourakis, A. Stavrou, and J. Voas. Ddos in the iot: Mirai and other botnets. *Computer*, 50(7):80–84, 2017. ISSN 0018-9162. doi: 10.1109/MC.2017.201.