ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΙΓΑΙΟΥ
ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΑΚΩΝ ΚΑΙ ΕΠΙΚΟΙΝΩΝΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

Εργαλείο Αποτίμησης Ρίσκου για Εικόνες Δοχείων Docker (A Risk Assessment Tool for Docker Container Images)

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

Μανουήλ Ιωάννη Γαλάνη

Μέλη Εξεταστικής Επιτροπής:

Επιβλέπων Διπλωματικής:	Μέλος Επιτροπής	Μέλος Επιτροπής
Κυριάκος Κρητικός	Σπυρίδων Κοκολάκης	Δημήτρης Σκούτας
Αναπληρωτής Καθηγητής	Καθηγητής	Επίκουρος Καθηγητής
Πανεπιστήμιο Αιγαίου	Πανεπιστήμιο Αιγαίου	Πανεπιστήμιο Αιγαίου

Σάμος, Οκτώβριος 2023

Πρόλογος και ευχαριστίες

Θα ήθελα να ευχαριστήσω όλους όσους με βοήθησαν για την εκπόνηση της διπλωματικής εργασίας και κυρίως τον Αναπληρωτή Καθηγητή του Τμήματος Μηχανικών & Πληροφοριακών Συστημάτων και υπεύθυνο για αυτή τη διπλωματική, κ. Κυριάκο Κρητικό, για την βοήθεια και την καθοδήγηση του.

Πίνακας περιεχομένων

Λίστα Σχημάτων

Λίστα Πινάκων

Ακρωνύμια

API	Application Programming Interface
CVE	Common Vulnerabilities and Exposures
CVSS	Common Vulnerability Scoring System
IDE	Integrated Development Environment
NIST	National Institute of Standards and Technology
NVD	National Vulnerability Database

Περίληψη

Η παρούσα διπλωματική εργασία αφορά την δημιουργία και υλοποίηση ενός εργαλείου υπολογισμού του ρίσκου τόσο για μεμονωμένες εικόνες δοχείου όσο και για ένα σύνολο από εικόνες δοχείων που απαρτίζουν μια εφαρμογή / σύστημα. Για να επιτευχθεί αυτός ο σκοπός υλοποιούνται δύο αλγόριθμοι, ένας για την συλλογή όλων των απαραίτητων δεδομένων και ένας δεύτερος για τον υπολογισμό του ρίσκου στα δύο προαναφερόμενα επίπεδα (μεμονωμένων εικόνων και εφαρμογής). Επίσης, χρησιμοποιούνται δύο εργαλεία εύρεσης ευπαθειών σε εικόνες δοχείων, το Clair-scanner και το Grype-analyzer, για καλύτερη ακρίβεια στον εντοπισμό των ευπαθειών. Για την συλλογή των δεδομένων, πέρα από τα ίδια τα προαναφερόμενα εργαλεία, χρησιμοποιούνται και δύο online βάσεις δεδομένων, η NVD και η cveapi.com, από τις οποίες αντλούνται όλες οι λεπτομέρειες για τις ευπάθειες που έχουν εντοπιστεί. Το προτεινόμενο εργαλείο αποτίμησης ρίσκου υλοποιήθηκε στην γλώσσα προγραμματισμού Java ενώ μπορεί να εκτελεστεί αποκλειστικά σε Linux-based λειτουργικά συστήματα.
Η τρέχουσα αναφορά της διπλωματικής αυτής εργασίας, εκτός από το να αναλύει ολοκληρωμένα το υλοποιημένο εργαλείο υπολογισμού ρίσκου, το επιδεικνύει με βάση σενάρια χρήσης καθώς και το αποτιμά ως προς την ακρίβειά του. Τέλος, παρουσιάζει τα συμπεράσματα που προέκυψαν μέσα από την μελέτη και υλοποίηση του συγκεκριμένου εργαλείου, καθώς και βελτιώσεις που θα μπορούσαν να λάβουν θέση μελλοντικά για να αυξήσουν την απόδοση και την αποτελεσματικότητα του.

Λέξεις κλειδιά

έλεγχος για ευπάθειες, εικόνες δοχείων, επίπτωση, πιθανότητα, αποτίμηση ρίσκου, CVSS, CVE, NVD, Clair-scanner, Grype Analyzer.

Abstract

The hereunto dissertation concerns the creation and implementation of a tool, which calculates the risk both for individual container images and for the overall application / system comprising a set of container images. In order to achieve this target, two algorithms are implemented, one for the retrieval of all the necessary data and one for the calculation of the risk at the two aforementioned levels (container images and whole application). Further, two tools for vulnerability detection in container images are used, the Clair scanner and the Grype-analyzer. For data retrieval purposes, apart from the aforementioned tools, two online data bases, NVD and cveapi.com, are also used from which all the details of the detected vulnerabilities are drawn. The suggested risk assessment tool is implemented in the Java programming language while it can run exclusively on Linux-based operating systems.

The current report of this dissertation, in addition to comprehensively analyzing the suggested risk computation tool, demonstrates it based on usage scenarios as well as evaluates it in terms of its accuracy. It also presents the conclusions reached through the study and implementation of the suggested tool, as well as improvements that could be followed in the future to increase its efficiency and effectiveness.

Keywords

Vulnerability scanning, container images, impact, probability, risk assessment, CVSS, CVE, VND, Clair-scanner, Grype Analyzer.

1. Εισαγωγή

Στην σημερινή εποχή της ψηφιοποίησης όλο και περισσότερων διαδικασιών, οι οποίες ενδέχεται να εμπλέκουν ευαίσθητα προσωπικά δεδομένα ανθρώπων και επιχειρήσεων, η εξασφάλιση ότι τα δεδομένα αυτά καθώς και οι παρεχόμενες υπηρεσίες – διαδικασίες θα είναι ασφαλείς είναι επιτακτική ανάγκη.

Τα τελευταία χρόνια, παρατηρούμε την αυξανόμενη χρήση των υπηρεσιών υπολογιστικού νέφους, σε ολοένα και περισσότερους τομείς της καθημερινότητας. Ειδικότερα, ολοένα και περισσότερες υπηρεσίες, εφαρμογές και διαδικασίες υλοποιούνται σε μια εγγενή αρχιτεκτονική νέφους (cloud-native architecture) μέσω δοχείων (containers) καθώς και εκτελούνται σε κατάλληλα περιβάλλοντα νέφους. Οπότε, επιβάλλεται η εξασφάλιση της ασφάλειας των δοχείων εκτός από την ασφάλεια των περιβαλλόντων στα οποία εκτελούνται τα δοχεία αυτά, η οποία μπορεί να αναλαμβάνεται από τον αντίστοιχο πάροχο.

Η παρούσα διπλωματική εργασία έχει ως στόχο να αποτιμήσει την ασφάλεια των εικόνων δοχείων (container images) έτσι ώστε να ενισχυθεί η ασφάλεια των εφαρμογών και υπηρεσιών νέφους μέσω της χρήσης ασφαλών εικόνων δοχείων χαμηλού ρίσκου. Ειδικότερα, πραγματεύεται την δημιουργία ενός εργαλείου, το οποίο θα μπορεί να εντοπίζει τις ευπάθειες εικόνων δοχείων Docker, διότι η τεχνολογία Docker έχει επικρατήσει, και να υπολογίζει την επίπτωση και το αντίστοιχο ρίσκο που ενδέχεται να έχει σε μια εφαρμογή ή ένα σύστημα η εκμετάλλευση τους. Για καλύτερη ακρίβεια στον εντοπισμό των ευπαθειών χρησιμοποιούνται πολλαπλά εργαλεία ανίχνευσης ευπαθειών. Επιπροσθέτως, για να είναι εφικτός ο υπολογισμός του ρίσκου με ολοκληρωμένο τρόπο, το προτεινόμενο εργαλείο ενώνει τις πληροφορίες που παράγουν τα εργαλεία ανίχνευσης ευπαθειών μέσα από την χρήση γνώσης, η οποία προέρχεται από διαδικτυακές βάσεις δεδομένων (τρωτοτήτων).

Ορισμένα πλεονεκτήματα που προσφέρει το εργαλείο που αναπτύξαμε είναι τα ακόλουθα:

- Μεγαλύτερη ακρίβεια στον υπολογισμό ευπαθειών και αντίστοιχου ρίσκου
- Υπολογισμός ρίσκου τόσο στο επίπεδο των εικόνων δοχείων όσο και στο επίπεδο των εφαρμογών, οι οποίες μπορεί να απαρτίζονται από εικόνες δοχείων
- Ρεαλιστικός υπολογισμός ρίσκου με βάση δεδομένα που μπορούν σίγουρα να ληφθούν από εργαλεία ανίχνευσης τρωτοτήτων και online βάσεις δεδομένων τρωτοτήτων
- Παροχή υποστήριξης στην επιλογή ασφαλών εικόνων δοχείων κατά την σχεδίαση μιας εφαρμογής

- Υψηλό επίπεδο παραμετροποίησης του εργαλείου
- Ευκολία εγκατάστασης & χρήσης

1.1 Δομή της διπλωματικής

Το θεωρητικό υπόβαθρο για τον αλγόριθμο υπολογισμού του ρίσκου που περιλαμβάνει τις παραμέτρους στις οποίες αυτός βασίζεται, τις βάσεις δεδομένων που χρησιμοποιήθηκαν καθώς και τα διαθέσιμα εργαλεία που διατίθενται για τον εντοπισμό ευπαθειών σε εικόνες δοχείων παρουσιάζονται στο Κεφάλαιο 2. Στο Κεφάλαιο 3 περιγράφονται εργασίες σχετικές με την αποτίμηση ρίσκου. Στο Κεφάλαιο 4 αναλύεται η αρχιτεκτονική του εργαλείου που υλοποιήθηκε, ενώ στο Κεφάλαιο 5 παρουσιάζεται η υλοποίηση του. Στο Κεφάλαιο 6 γίνεται μια πρώτη επίδειξη και αποτίμηση του εργαλείου με βάση την απόδοση του και την ακρίβεια των αποτελεσμάτων του. Τέλος, στο Κεφάλαιο 7 εξάγονται κάποια συμπεράσματα και παρουσιάζονται ορισμένες κατευθύνσεις για ενδεχόμενη βελτίωση του προτεινόμενου εργαλείου.

2. Υπόβαθρο

2.1 Παράμετροι υπολογισμού του ρίσκου

Για να επιτύχουμε τον καλύτερο και ακριβέστερο υπολογισμό του ρίσκου τόσο για μια εικόνα όσο και για μια εφαρμογή / ένα σύστημα (δηλ. ένα σύνολο εικόνων) έπρεπε να λάβουμε υπόψιν ποικίλους παράγοντες, οι οποίοι θα μπορούν να περιγράψουν πλήρως όλες τις πλευρές μιας ευπάθειας και να μας δώσουν μια σαφή εικόνα για την σοβαρότητα και τις επιπτώσεις που ενδέχεται να έχει. Ο πιο εύκολος και συνάμα αποδοτικός τρόπος να γίνει αυτό είναι με την χρήση του πρότυπου CVSS (Common Vulnerability Scoring System) (Ενιαίο Σύστημα Βαθμολόγησης Ευπαθειών).

Το πρότυπο αυτό βρίσκεται αυτή τη στιγμή στην τρίτη του έκδοση. Η πρώτη έκδοση δημοσιεύτηκε το 2005 από το National Infrastructure Advisory Council των ΗΠΑ. Η δεύτερη έκδοση του δημοσιεύθηκε το 2007 και χρησιμοποιήθηκε ευρέως. Η τρίτη έκδοση δημοσιεύτηκε για πρώτη φορά το 2015 και ανανεώθηκε τελευταία φορά το 2019. Η τρίτη έκδοση, παρότι βελτίωσε σε ένα βαθμό την ακρίβεια των πληροφοριών για τις διάφορες ευπάθειες, δεν κατόρθωσε να διορθώσει όλα τα ζητήματα της δεύτερης έκδοσης [6], όπως για παράδειγμα το γεγονός ότι αδυνατούσε να βαθμολογήσει σωστά "απομακρυσμένες" ευπάθειες, οι οποίες είχαν μικρό αντίκτυπο. Επίσης, παρέμεινε το πρόβλημα αναφορικά με το γεγονός ότι παρότι η μετρική "περιπλοκότητα πρόσβασης" αποτελείται από τρία επίπεδα (Low, Medium, High), στην πράξη χρησιμοποιούνται μόνο τα δυο. Αυτό, σε συνδυασμό με το γεγονός ότι δημιούργησε κάποια νέα ζητήματα:

- αύξηση της πολυπλοκότητας του συστήματος βαθμολόγησης με την εισαγωγή νέων μετρικών
- μη ενσωμάτωση στην φόρμουλα βαθμολόγησης της αλληλεπίδρασης των χρηστών που απαιτείται για την εκμετάλλευση ορισμένων ευπαθειών
- συμπερίληψη μιας καινούργιας μετρικής (Attack Vector), η οποία διαδραματίζει καθοριστικό ρόλο στο τελικό αποτέλεσμα και πολλές φορές είναι υποκειμενική με συνέπεια το τελικό αποτέλεσμα να διαφέρει αρκετά ανάλογα με το ποιος κάνει την αξιολόγηση

τα οποία δεν υπήρχαν στην προηγούμενη έκδοση, έχουν οδηγήσει στην καθυστέρηση της μετάβασης των αλγορίθμων που χρησιμοποιούν αυτό το πρότυπο από την χρήση της δεύτερης στην χρήση της τρίτης έκδοσής του. Ταυτόχρονα, η τρίτη έκδοση δεν έχει ευρέως χρησιμοποιηθεί για την περιγραφή των ευπαθειών σε σχέση με την δεύτερη έκδοση με αποτέλεσμα τα αποθετήρια / βάσεις

δεδομένων τρωτοτήτων να περιέχουν μικρό αριθμό από ευπάθειες για τις οποίες παρέχεται περιγραφή στην 3η έκδοση του CVSS.
Επιπλέον, ο συνδυασμός των δύο εκδόσεων δεν είναι μια βιώσιμη λύση καθότι υπάρχουν αρκετές διαφορές τόσο στον τρόπο βαθμολόγησης όσο και στο εύρος βαθμολόγησης που προδιαγράφουν. Επομένως, ένας συνδυασμός τους θα οδηγούσε σε πολλά σφάλματα και αστοχίες. Για τους παραπάνω λόγους αποφασίσαμε να χρησιμοποιήσουμε την δεύτερη έκδοση του συστήματος βαθμολόγησης CVSS.

Το σύστημα βαθμολόγησης CVSS 2.0 συντίθεται από τρία γκρουπ μετρικών:

1. Ομάδα βασικών μετρικών (Base metric group).
2. Ομάδα δυναμικών μετρικών (Temporary metric group).
3. Ομάδα περιβαλλοντικών μετρικών (Environmental metric group).

Η πρώτη ομάδα αναπαριστά τα βασικά χαρακτηριστικά των ευπαθειών, τα οποία παραμένουν σταθερά ανεξάρτητα από την πάροδο του χρόνου ή το περιβάλλον στο οποίο βρίσκονται. Αυτά μπορούν να συνοψιστούν στα παρακάτω:

- *Access Vector (Τρόπος Πρόσβασης)*: Αναφέρεται στο "πώς" η ευπάθεια εκτελείται.
- *Access Complexity (Πολυπλοκότητα Πρόσβασης)*: Αναφέρεται στο πόσο πολύπλοκη είναι η διαδικασία της εκμετάλλευσης μιας ευπάθειας από την στιγμή που έχουμε πρόσβαση στο σύστημα-στόχο. Όσο χαμηλότερη είναι η πολυπλοκότητα τόσο μεγαλύτερη είναι η βαθμολογία.
- *Authentication (Αυθεντικοποίηση)*: Αναφέρεται στο πόσες φορές χρειάζεται να ταυτοποιηθεί ο επιτιθέμενος κατά την διάρκεια της επίθεσης.
- *Confidentiality Impact (Επίπτωση Εμπιστευτικότητας)*: Επιπτώσεις στην εμπιστευτικότητα των δεδομένων που επεξεργάζεται το σύστημα από την εκμετάλλευση της συγκεκριμένης ευπάθειας.
- *Integrity Impact (Επίπτωση Ακεραιότητας)*: Επιπτώσεις στην ακεραιότητα των δεδομένων και εν γένει του συστήματος από την εκμετάλλευση της συγκεκριμένης ευπάθειας.
- *Availability Impact (Επίπτωση Διαθεσιμότητας)*: Επιπτώσεις στην διαθεσιμότητα των δεδομένων του συστήματος από την εκμετάλλευση της συγκεκριμένης ευπάθειας.

Οι τρεις πρώτες μετρικές περιγράφουν το πώς η ευπάθεια είναι προσβάσιμη και το αν απαιτείται η ικανοποίηση και άλλων συνθηκών για να γίνει αυτή εκμεταλλεύσιμη, ενώ τα επόμενα τρία περιγράφουν το πως θα επηρεαστεί (αρνητικά) το σύστημα σε περίπτωση που η εκμετάλλευση αυτή επιτευχθεί.
Αναλυτικότερα οι τιμές που μπορούν να πάρουν οι παραπάνω "μετρικές" καθώς και η λογική σημασία των τιμών αυτών φαίνονται στον παρακάτω πίνακα:

Πίνακας 1: Οι τιμές για τις βασικές μετρικές του CVSS

Τίτλος Βασικής Μετρικής	**Τιμές Βασικής Μετρικής**	**Περιγραφή**
Επίπεδο Πρόσβασης (Access Vector)	Local (L)	Απαιτεί από τον επιτιθέμενο να έχει τοπική πρόσβαση για να εκτελέσει την ευπάθεια.
	Adjacent Network (A)	Απαιτεί από τον επιτιθέμενο να έχει πρόσβαση είτε σε κάποια broadcast εκπομπή είτε σε κάποιο collision domain του ευπαθούς λογισμικού (πχ. τοπικό IP υπο-δίκτυο).
	Network (N)	Δεν απαιτεί τοπική πρόσβαση. Η ευπάθεια μπορεί να εκτελεστεί απομακρυσμένα.
Πολυπλοκότητα Πρόσβασης (Access Complexity)	High (H)	Υπάρχουν εξειδικευμένες συνθήκες πρόσβασης.
	Medium (M)	Οι συνθήκες πρόσβασης είναι εν μέρη εξειδικευμένες.
	Low (L)	Δεν υπάρχουν εξειδικευμένες συνθήκες πρόσβασης.
Αυθεντικοποίηση (Authentication)	Multiple (M)	Η εκτέλεση της ευπάθειας προϋποθέτει ότι ο επιτιθέμενος πρέπει να αυθεντικοποιηθεί στο σύστημα-στόχο δύο ή παραπάνω φορές.
	Single (S)	Η εκτέλεση της ευπάθειας απαιτεί ο επιτιθέμενος να ταυτοποιηθεί μόνο μια φορά στο σύστημα-στόχο.
	None (N)	Δεν απαιτείται αυθεντικοποίηση.
Επίπτωση Εμπιστευτικότητας (Confidentiality Impact)	None (N)	Δεν υπάρχει κάποια επίπτωση στην εμπιστευτικότητα του συστήματος.
	Partial (P)	Υπάρχει μια σημαντική αποκάλυψη πληροφορίας.
	Complete (C)	Υπάρχει πλήρης αποκάλυψη των πληροφοριών.
Επίπτωση Ακεραιότητας (Integrity Impact)	None (N)	Δεν υπάρχει κάποια επίπτωση στην ακεραιότητα του συστήματος.
	Partial (P)	Υπάρχει κάποια επίπτωση στην ακεραιότητα του συστήματος.
	Complete (C)	Υπάρχει μεγάλη παραβίαση της ακεραιότητας του συστήματος.

Επίπτωση Διαθεσιμότητας (Availability Impact)	None (N)	Δεν υπάρχει κάποια επίπτωση στην διαθεσιμότητα του συστήματος.
	Partial (P)	Μείωση ή πλήρης διακοπή της διαθεσιμότητας του συστήματος.
	Complete (C)	Πλήρης τερματισμός λειτουργίας για τους πόρους που επηρέασε η εκμετάλλευση της ευπάθειας.

Η δεύτερη ομάδα αναπαριστά τις μετρικές εκείνες που μπορούν να αλλάξουν με την πάροδο του χρόνου. Οι μετρικές αυτές είναι οι εξής:

1. Exploitability (Δυνατότητα εκμετάλλευσης): Αναφέρεται στην διαθεσιμότητα των συστατικών (τεχνικές εκμετάλλευσης, κώδικας εκμετάλλευσης, κλπ) προς εκμετάλλευση μιας ευπάθειας.
2. Remediation Level (Επίπεδο Αντιμετώπισης): Αναφέρεται στο πόσο καλά αντιμετωπίζεται η ευπάθεια.
3. Report Confidence (Αναφορά Εμπιστοσύνης): Αναφέρεται στην αξιοπιστία των πληροφοριών που είναι διαθέσιμες για την ευπάθεια καθώς και στο επίπεδο αυτοπεποίθησης πως η ευπάθεια αυτή υπάρχει.

Από αυτές τις τρεις μετρικές η μόνη που θα χρησιμοποιηθεί στο προτεινόμενο εργαλείο της διπλωματικής είναι η δεύτερη (επίπεδο αντιμετώπισης). Αυτή ουσιαστικά δίνει πληροφορία για το αν η ευπάθεια έχει επιδιορθωθεί με κάποιο τρόπο ή σε πιο στάδιο είναι η εργασία/μελέτη της επιδιόρθωσης της, πράγμα που την καθιστά αρκετά σημαντική.

Ο λόγος για τον οποίο οι άλλες δύο μετρικές δεν θα χρησιμοποιηθούν στον αλγόριθμο μας είναι ότι δεν μπορούμε να αντλήσουμε πληροφορία για την τιμή τους ούτε από τα εργαλεία ανίχνευσης τρωτοτήτων αλλά ούτε και από τις διαδικτυακές βάσεις δεδομένων (τρωτοτήτων).

Οι τιμές που μπορεί να λάβει η μεταβλητή Remediation Level παρουσιάζονται στον παρακάτω πίνακα:

Πίνακας 2: Οι τιμές για την μετρική Remediation Level του CVSS

Τιμή Μετρικής	**Περιγραφή**
Official Fix (OF)	Υπάρχει διορθωτική λύση για την ευπάθεια μέσω αναβάθμισης ή επίσημης επιδιόρθωσης (patch) από τον πάροχο.
Temporary Fix (TF)	Υπάρχει κάποια διορθωτική λύση από τον πάροχο, αλλά είναι προσωρινή.
Workaround (W)	Υπάρχει κάποια διορθωτική λύση, αλλά όχι από τον επίσημο πάροχο.
Unavailable (U)	Είτε δεν υπάρχει λύση είτε δεν μπορεί να εφαρμοστεί.
Not Defined (ND)	Αυτή η τιμή δεν θα επηρεάσει το αποτέλεσμα (αποτίμησης).

Η τρίτη ομάδα μετρικών αναπαριστά τα χαρακτηριστικά που έχει η εκάστοτε ευπάθεια ανάλογα με το περιβάλλον στο οποίο βρίσκεται (για παράδειγμα ανάλογα με τον σκοπό και την χρήση του εκάστοτε πληροφοριακού συστήματος ή εταιρίας). Οι μετρικές αυτές μπορούν να συνοψισθούν ως εξής:

1. Collateral Damage Potential (CDP) (Δυνατότητα Παράπλευρης Ζημιάς): Αναφέρεται στην πιθανότητα κλοπής ή καταστροφής εξοπλισμού (φυσικού / λογισμικού) ή ακόμα και σε οικονομικές συνέπειες.
2. Target Distribution (TD) (Κατανομή Στόχου): Αναφέρεται στον υπολογισμό του ποσοστού των συστημάτων που ενδέχεται να επηρεαστούν από την εκμετάλλευση της ευπάθειας.
3. Security Requirements (Απαιτήσεις Ασφάλειας): Επιτρέπει την ρύθμιση/τελειοποίηση του σκορ περιβάλλοντος (δηλ. του μερικού σκορ για την τρίτη ομάδα μετρικών) ως προς το περιβάλλον και τις απαιτήσεις του χρήστη:
    - Confidentiality Requirement (CR) (Απαίτηση Εμπιστευτικότητας Δεδομένων)
    - Integrity Requirement (IR) (Απαίτηση Ακεραιότητας Δεδομένων)
    - Availability Requirement (AR) (Απαίτηση Διαθεσιμότητας Δεδομένων)

Από τις παραπάνω μετρικές, στον αλγόριθμο συμπεριλαμβάνεται μόνο η μετρική Security Requirements. Οι τιμές που μπορεί να πάρει η μετρική αυτή και ειδικότερα οι υπο-μετρικές της καθορίζονται εξ ‘αρχής από τον διαχειριστή του συστήματος, ανάλογα με την σημασία που έχει για το εκάστοτε σύστημα / εταιρία η μη ικανοποίηση κάποιας από τις απαιτήσεις ασφάλειας αλλά και την σχέση που έχουν τα συστατικά του συστήματος (που αντιστοιχούν σε αντίστοιχες εικόνες δοχείων που ελέγχονται για τρωτότητες) με την απώλεια αυτών των απαιτήσεων ασφαλείας. Για τον λόγο αυτό δεν μπορούμε να πάρουμε αυτές τις τιμές από κάποια online βάση δεδομένων (τρωτοτήτων). Οι τιμές που μπορεί να δώσει ο διαχειριστής στις τρεις παραπάνω υπο-μετρικές απαιτήσεων ασφάλειας παρουσιάζονται στον παρακάτω πίνακα:

Πίνακας 3: Οι τιμές για την μετρική Security Requirements του CVSS

Τιμή Μετρικής	**Περιγραφή**
Low (L)	Πιθανώς μικρή επίπτωση από την απώλεια τους.
Medium (M)	Πιθανώς σοβαρή επίπτωση από την απώλεια τους.
High (H)	Πιθανώς πολύ σημαντική επίπτωση από την απώλεια τους.
Not Defined (ND)	Αυτή η τιμή δεν θα επηρεάσει το αποτέλεσμα (αποτίμησης).

Χρειάζεται, ωστόσο, πολύ προσοχή από τον διαχειριστή του συστήματος στο τι τιμές θα θέσει της συγκεκριμένες παραμέτρους διότι μπορεί να επηρεάσει σε μεγάλο βαθμό τα αποτελέσματα του εργαλείου. Αν θέσει την ελάχιστη τιμή, τότε ο αλγόριθμος θα θεωρεί ότι η επίπτωση της εκμετάλλευσης της ευπάθειας δεν θα είναι μεγάλη για το σύστημα, με αποτέλεσμα το σκορ για την συγκεκριμένη ευπάθεια να είναι χαμηλότερο, το οποίο με την σειρά του μπορεί να οδηγήσει στο να αγνοηθεί η ευπάθεια και, εν τέλει, όταν εκτελεστεί να προκαλέσει μεγαλύτερη ζημιά από την αναμενόμενη. Από την άλλη μεριά, αν ο διαχειριστής θέσει την υψηλότερη τιμή, τότε είναι

πιθανότερο να θεωρηθεί επικίνδυνη για το σύστημα η εκάστοτε ευπάθεια και να ξεκινήσουν οι διαδικασίες αντιμετώπισης της. Όμως, αυτό ενέχει τον κίνδυνο να χαρακτηριστούν ως επικίνδυνες ευπάθειες που δεν θα έπρεπε, με αποτέλεσμα να σπαταλούνται πόροι και χρόνος από την αντιμετώπιση των ουσιαστικά επικινδύνων ευπαθειών.
Άρα, λοιπόν, θα πρέπει οι τιμές αυτές να προκύπτουν μετά από ενδελεχή έλεγχο των συστατικών της εφαρμογής / συστήματος ώστε να εντοπιστούν οι ανάγκες αυτών αλλά και η σωστή εκτίμηση της σημαντικότητας του κάθε συστατικού για την εφαρμογή / σύστημα ανάλογα με την φύση τους και τα δεδομένα που επεξεργάζονται.

2.2 Εργαλεία ανίχνευσης ευπαθειών

Για την ανίχνευση των ευπαθειών σε μια εικόνα δοχείου έχει εντοπισθεί μια πληθώρα από διαθέσιμα εργαλεία, τα οποία περιλαμβάνουν:

1. Clair scanner
2. Grype
3. Dagda
4. Docker Bench
5. Harbor

Τα εργαλεία αυτά αναλύονται στην συνέχεια. Τα δύο πρώτα αναλύονται εκτενέστερα διότι τελικά, όπως αναφέρεται και παρακάτω, είναι αυτά που θα χρησιμοποιηθούν στην παρούσα εργασία για την ανίχνευση των ευπαθειών.

2.2.1 Clairscanner

Το εργαλείο clair-scanner[1] είναι μια ελαφρώς διαφοροποιημένη – πιο απλή στη χρήση - μορφή/έκδοση του εργαλείου clair, το οποίο έχει αρχικά δημιουργηθεί από την CoreOS. Το clair-scanner λειτουργεί ως εξής:

1. Επικοινωνεί με έναν clair-server (διακομιστή Clair) ώστε να αντλήσει περαιτέρω πληροφορίες για μια τρωτότητα που ανίχνευσε στην εικόνα δοχείου. Αυτός ο διακομιστής παρέχει πρόσβαση σε μια βάση δεδομένων στην οποία είναι αποθηκευμένες πληροφορίες για γνωστές ευπάθειες.
2. Συγκρίνει τις ευπάθειες που “ανίχνευσε” με ένα whitelist. Ουσιαστικά παρέχεται η δυνατότητα στον χρήστη του εργαλείου να “εξαιρέσει” κάποιες ευπάθειες από την λίστα με τις ευπάθειες που εντοπίστηκαν προσθέτοντας τις σε ένα αρχείο (whitelist). Έτσι, αν το εργαλείο εντοπίσει κάποια από αυτές στην εικόνα δοχείου που εξετάζει, δεν θα την συμπεριλάβει στην τελική αναφορά που πρέπει να παράγει.
3. Στην περίπτωση που ανιχνευθούν ευπάθειες που δεν περιλαμβάνονται στο whitelist τότε εμφανίζει τις εξής πληροφορίες για αυτές:

[1] https://github.com/arminc/clair-scanner

1. Το όνομα (πεδίο: featurename) και την έκδοση του λογισμικού (asset) στο οποίο εντοπίστηκε η ευπάθεια.
2. Μια μικρή περιγραφή της ευπάθειας καθώς και έναν εξωτερικό σύνδεσμο για περισσότερες πληροφορίες.
3. Τιμή για την παράμετρο “severity” του CVSS Scoring System.
4. Ένα πεδίο με τίτλο “fixedby”, το οποίο αναφέρεται στην ύπαρξη η μη κάποιος διόρθωσης για την αντιμετώπιση της ευπάθειας.
5. Ένα πεδίο με τίτλο “namespace”, το οποίο αναφέρεται στο γενικότερο πλαίσιο του λογισμικού στο οποίο βρέθηκε η εκάστοτε ευπάθεια. .
6. Το αναγνωριστικό της ευπάθειας (πεδίο: Vulnerability) (αντιστοιχεί σε ένα αναγνωριστικό cve-id με βάση το πρότυπο CVE).

Όσον αφορά τα αποτελέσματα, το εργαλείο έχει την δυνατότητα να τα αποθηκεύει σε ένα αρχείο με μορφοποίηση JSON χρησιμοποιώντας την επιλογή -r και το όνομα του αρχείου.
Πρακτικά οι εξαγόμενες πληροφορίες από το εργαλείο απεικονίζονται με έναν πίνακα που αποτελείται από JsonObject (αντικείμενα JSON) και κάθε JsonObject περιλαμβάνει τις πληροφορίες για την εκάστοτε ευπάθεια που εντόπισε. Κάθε πληροφορία αποτελεί ένα ζεύγος κλειδιών-τιμών μέσα σε αυτό το JsonObject.

Για την εκμετάλλευση των αποτελεσμάτων του εργαλείου clair-scanner μας ενδιαφέρουν οι τιμές featurename, Vulnerability και fixedby διότι σχετίζονται με ορισμένες μετρικές του CVSS αλλά και με βάση ορισμένες από αυτές (πχ. τιμή πεδίου Vulnerability) μπορούμε να αντλήσουμε περισσότερες πληροφορίες για μια ευπάθεια.

2.2.2 Grype

Το εργαλείο Anchore Grype[2] είναι ανεπτυγμένο από την εταιρία Anchore και είναι διαθέσιμο σε εκδόσεις συμβατές με λειτουργικά συστήματα Linux και MacOs. Χρησιμοποιεί μια βάση ευπαθειών που είναι αποθηκευμένη στο τοπικό σύστημα αρχείων. Η βάση αυτή δημιουργείται εξάγοντας δεδομένα από διάφορες δημόσια διαθέσιμες πηγές όπως :

- Alpine Linux SecDB: https://secdb.alpinelinux.org/
- Amazon Linux ALAS: https://alas.aws.amazon.com/AL2/alas.rss
- RedHat RHSAs: https://www.redhat.com/security/data/oval/
- Debian Linux CVE Tracker: https://security-tracker.debian.org/tracker/data/json
- Github GHSAs: https://github.com/advisories
- National Vulnerability Database (NVD): https://nvd.nist.gov/vuln/data-feeds
- Oracle Linux OVAL: https://linux.oracle.com/security/oval/
- RedHat Linux Security Data: https://access.redhat.com/hydra/rest/securitydata/
- Suse Linux OVAL: https://ftp.suse.com/pub/projects/security/oval/
- Ubuntu Linux Security: https://people.canonical.com/~ubuntu-security/

[2] https://github.com/anchore/grype

Το εργαλείο φροντίζει αυτόματα από μόνο του για την διαχείριση της βάσης δεδομένων (π.χ ενημέρωση για νέες ευπάθειες), ωστόσο η διαχείριση αυτή είναι παραμετροποιήσιμη από τον χρήστη. Επίσης, προσφέρει ποικίλες δυνατότητες όπως η εξέταση τόσο εικόνων δοχείου όσο και δοχείων που εκτελούνται ήδη. Ακόμη, παρέχει, επιλογή διαμόρφωσης για την αποθήκευση των αποτελεσμάτων σε διάφορες μορφοποιήσεις με την χρήση της παραμέτρου (διαμόρφωσης) -o ως εξής: `grype <image> -o <format>` όπου <format> είναι η επιθυμητή μορφοποίηση. Οι διαθέσιμες επιλογές μορφοποίησης είναι οι εξής:

1. table: εξαγωγή των αποτελεσμάτων σε μορφή πίνακα (προεπιλογή)
2. cyclonedx (XML)
3. json
4. template: ο τύπος (πρότυπο) εξόδου του εργαλείου καθορίζεται από τον χρήστη. Σε αυτή την περίπτωση, το πρότυπο αυτό θα πρέπει να έχει δημιουργηθεί με το εργαλείο goTemplate[3] και να είναι αποθηκευμένο σε ένα αρχείο με κατάληξη .template στο τοπικό σύστημα αρχείων ενώ θα πρέπει να δοθεί το μονοπάτι προς αυτό το αρχείο κατά την εκτέλεση του εργαλείου.

Παρόμοια με το εργαλείο clair-scanner, έτσι και το Grype παρουσιάζει ως προς το περιεχόμενο των αναφορών του μεταξύ άλλων και κάποιες βασικές πληροφορίες για τις ευπάθειες:

1. Το αναγνωριστικό της ευπάθειας (πεδίο: id)
2. Έναν εξωτερικό σύνδεσμο για περισσότερες πληροφορίες.
3. Ένα πεδίο με τίτλο “namespace”, το οποίο αναφέρεται στο γενικότερο πλαίσιο του λογισμικού στο οποίο βρέθηκε η εκάστοτε ευπάθεια..
4. Τιμή για την παράμετρο “severity” του CVSS.
5. Το όνομα του λογισμικού (asset) στο οποίο εντοπίστηκε η ευπάθεια (πεδίο: name).
6. Ένα πεδίο «state» που δηλώνει την ύπαρξη η όχι κάποιας διόρθωσης (της ευπάθειας).

Παρόμοια με την περίπτωση του εργαλείου Clair Scanner, οι πληροφορίες ευπάθειας που μας ενδιαφέρουν αφορούν τις τιμές των πεδίων id, name, και “state” διότι σχετίζονται με ορισμένες μετρικές του CVSS αλλά και με βάση ορισμένες από αυτές (πχ. τιμή πεδίου id) μπορούμε να αντλήσουμε περισσότερες πληροφορίες για μια ευπάθεια.

2.2.3 Dagda

Το εργαλείο Dagda[4] χρησιμοποιείται για την στατική ανάλυση γνωστών ευπαθειών, δούρειων ίππων (trojans), ιών, κακόβουλων λογισμικών και άλλων απειλών σε εικόνες δοχείων Docker. Για να μπορέσει να επιτελέσει το στόχο αυτό, το εργαλείο εισάγει δεδομένα από online πηγές δεδομένων που περιέχουν τις αντίστοιχες πληροφορίες σε μια τοπική βάση δεδομένων (Mongo DB), η οποία εγκαθίστανται μαζί με το εργαλείο και αναβαθμίζεται περιοδικά από το εργαλείο αυτό. Έτσι όταν ο χρήστης επιθυμεί να εκτελέσει μια ανάλυση, το εργαλείο αναζητά σε αυτή τη

[3] https://pkg.go.dev/text/template
[4] https://github.com/eliasgranderubio/dagda/

βάση τα αντίστοιχα δεδομένα, όπως πακέτα λειτουργικού συστήματος (OS packages) και εξαρτήσεις προγραμματιστικής γλώσσας (programming languages dependencies). Κάθε ανάλυση που πραγματοποιείται στο τέλος αποθηκεύεται στην ίδια βάση δεδομένων από την οποία αντλήθηκαν τα δεδομένα.

Για την εκτέλεση του εργαλείου θα πρέπει να χρησιμοποιηθεί η παρακάτω εντολή:

```
python3 dagda.py check --docker_image <όνομα εικόνας δοχείου>
```

2.2.4 Docker Bench

Το Docker Bench[5] είναι ένα πρόγραμμα (script), το οποίο ελέγχει εικόνες δοχείων, για τυχόν ευπάθειες και "κακές πρακτικές" κατά την υλοποίηση των δοχείων αυτών. Είναι διαθέσιμο για τις διάφορες διανομές του λειτουργικού συστήματος Linux καθώς και για MacOS. Είναι βασισμένο στην έκδοση 1.4.0 του Center for Internet Security (CIS) Docker Benchmark[6]. Το CIS Benchmark είναι προϊόν μιας διαδικασίας κοινοτικής συναίνεσης και περιλαμβάνει οδηγίες για την ασφαλή διαμόρφωση του Docker. Ειδικότερα, είναι ένα σύνολο βέλτιστων πρακτικών και συστάσεων ασφάλειας για την ανάπτυξη δοχείων Docker, το οποίο παρέχει ένα περιεκτικό οδηγό για την ασφάλιση των δοχείων Docker καλύπτοντας διάφορες πτυχές διαμόρφωσης τους.

Μπορούμε να κατεβάσουμε το πρόγραμμα του Docker Bench από το github με την εντολή:

```
git clone https://github.com/docker/docker-bench-security.git
```

Για να εκτελέσουμε το συγκεκριμένο εργαλείο, θα πρέπει να έχουμε εγκατεστημένη στο σύστημά μας τουλάχιστον την έκδοση 1.13.0 της μηχανής δοχείων του Docker. Αφού εκφορτώσουμε το script με την προηγούμενη εντολή, θα πρέπει να το μετατρέψουμε σε εκτελέσιμο και εν συνεχεία να το εκτελέσουμε με την εντολή :

```
sudo sh docker-bench-security.sh
```

Η εκτέλεση του εργαλείου δεν απαιτεί την σύνδεση σε κάποιον διακομιστή. Επίσης, το εργαλείο δεν πραγματοποιεί κάποια αναζήτηση σε βάσεις δεδομένων ευπαθειών αλλά πραγματοποιεί κάποιους ελέγχους ασφαλείας βασισμένους στο CIS Docker Benchmark.

2.2.5 Harbor

Το εργαλείο Harbor[7] είναι ένα εργαλείο ανοιχτού κώδικα, το οποίο προστατεύει τις εικόνες δοχείων με πολιτικές ασφάλειας και αντίστοιχα συστήματα ελέγχου πρόσβασης βασισμένα σε ρόλους. Επίσης, εξασφαλίζει ότι οι εικόνες δοχείων έχουν ελεγχθεί για τυχόν ευπάθειες και εκείνες με καμία ή λίγες ευπάθειες πολύ μικρής κρισιμότητας μαρκάρονται ως "έμπιστες". Το Harbor αποτελεί ένα από τα graduated project του CNFC (Cloud Native Computing Foundation).

[5] https://github.com/dev-sec/cis-docker-benchmark
[6] https://github.com/docker/docker-bench-security

Μπορεί να εγκατασταθεί απευθείας σε κάποιο περιβάλλον kubernetes ή σε οποιοδήποτε σύστημα είναι εγκατεστημένο το Docker.
Για την ανίχνευση ευπαθειών χρησιμοποιεί τα εργαλεία clair και Trivy, ενώ η εξαγωγή των αποτελεσμάτων μπορεί να γίνει και σε JSON μορφοποίηση.
Για την εγκατάσταση του εργαλείου:

1. Θα πρέπει πρώτα να ελέγξουμε ότι το σύστημα μας καλύπτει τις ελάχιστες απαιτήσεις για την εκτέλεση του harbor, όπως αρκετό αποθηκευτικό χώρο, μνήμη και χωρητικότητα CPU.
2. Θα πρέπει να έχουμε εγκατεστημένο το Docker.
3. Θα πρέπει να κατεβάσουμε το πακέτο του εργαλείου από το github
4. Θα πρέπει να κάνουμε της κατάλληλες ρυθμίσεις ανάλογα με το σύστημα μας στο αρχείο "harbor.yml".
5. Τέλος Θα πρέπει να εκτελέσουμε το script της εγκατάστασης.

Το εργαλείο Harbor μπορεί να εκτελεστεί σε Linux, Windows ή MacOS λειτουργικό σύστημα.

2.2.6 Σύγκριση των πέντε προαναφερθέντων εργαλείων

Στον παρακάτω πίνακα γίνεται μια σύγκριση των εργαλείων που αναλύθηκαν προηγουμένως με βάση κάποια σημαντικά χαρακτηριστικά, τα οποία θα μας βοηθήσουν στην επιλογή των κατάλληλων εργαλείων.

Πίνακας 4: Σύγκριση των εργαλείων ανίχνευσης ευπαθειών

Εργαλείο	**Συμβατότητα με Λ/Σ**	**Μορφοποίηση αποτελεσμάτων σε JSON**	**Ευκολία εγκατάστασης**
Clair-scanner	Linux	Ναι	Εύκολη
Grype Analyzer	Linux	Όχι προκαθορισμένα	Εύκολη
Dagda	Linux	Όχι προκαθορισμένα	Μέτρια
Docker Bench	Linux, windows, MacOS	Όχι προκαθορισμένα	Μέτρια
Harbor	Linux	Ναι	Περίπλοκη

Οι σημαντικότερες από αυτές είναι η μορφοποίηση των αποτελεσμάτων στο πρότυπο JSON, έτσι ώστε να είναι εύκολη και κοινή για όλα τα εργαλεία η διαχείριση των αποτελεσμάτων, καθώς και η ευκολία στην εγκατάσταση τους.

Με βάση την ανάλυση που πραγματοποιήθηκε και αποτυπώθηκε στον παραπάνω πίνακα, τα εργαλεία που ξεχωρίζουν είναι τα clair-scanner και Grype Analyzer, διότι επιτρέπουν και την μορφοποίηση των αποτελεσμάτων σε μορφή JSON και έχουν εύκολη διαδικασία εγκατάστασης. Τα

εργαλεία Dadga & Docker Bench έρχονται δεύτερα διότι έχουν μέτρια ευκολία εγκατάστασης. Το εργαλείο Harbor έρχεται τελευταίο λόγω της πολυπλοκότητας της διαδικασίας εγκατάστασής του.

3. Σχετικές εργασίες

Σε αυτό το σημείο θα θέλαμε να αναφερθούμε σε σχετικές εργασίες που έχουν πραγματοποιηθεί πάνω στο αντικείμενο της διπλωματικής εργασίας.
Μια σημαντική εργασία [1], από την οποία άντλησε την προσέγγιση εκτίμησης ρίσκου των ευπαθειών εικόνων δοχείων η τρέχουσα διπλωματική, πραγματεύεται την δημιουργία ενός αλγορίθμου υπολογισμού του ρίσκου για εικόνες δοχείων βασιζόμενο στο CVSS V2. Οι διαφορές της εργασίας αυτής σε σχέση με την διπλωματική μας εργασία μπορούν να συνοψιστούν στα παρακάτω:

1. Το άρθρο αυτό παρουσίαζε μια θεωρητική προσέγγιση του αλγορίθμου ενώ η παρούσα εργασία τον υλοποιεί με πρακτικό τρόπο φιλτράροντας ορισμένα μέρη του που είναι αδύνατο να υπολογισθούν με βάση τις παρεχόμενες πληροφορίες από τα εργαλεία εντοπισμού τρωτοτήτων και χρησιμοποιώντας πολλαπλά εργαλεία ανίχνευσης/εντοπισμού ευπαθειών για την αύξηση της ακρίβειας εντοπισμού.
2. Στο άρθρο ο αλγόριθμος ήταν περισσότερο πολύπλοκος, με δυσκολότερες μαθηματικές πράξεις, κυρίως με την χρήση της λογικής των πιθανοτήτων αλλά και με την χρήση περισσότερων παραμέτρων στον υπολογισμό του ρίσκου. Η αυξημένη πολυπλοκότητα του αλγορίθμου θα οδηγούσε σε πολλές εμφωλευμένες δομές επανάληψης, οι οποίες θα καθιστούσαν το εργαλείο υπερβολικά αργό στον υπολογισμό του ρίσκου.

Στην εργασία [2] υλοποιείται ένα “φορητό” - καθότι για την λειτουργία του χρησιμοποιούνται ειδικά ρυθμισμένες εικόνες δοχείου Linux (Linux Containers – LXCs) - δοκιμαστικό περιβάλλον, για την δυναμική αξιολόγηση των εικόνων δοχείων, δηλαδή την αξιολόγηση της ασφάλειας και κατά την εκτέλεση των δοχείων των εικόνων μέσω της χρήσης του εργαλείου Dagda ώστε να εντοπιστούν ανώμαλες δραστηριότητες. Για την αξιολόγηση του περιβάλλοντος αυτού χρησιμοποιήθηκαν MongoDB Singularity δοχεία. Το περιβάλλον αυτό είναι αρκετά αποδοτικό ως προς την ευκολία στη χρήση και την κατανάλωση λίγων πόρων από το σύστημα, ενώ το γεγονός ότι είναι “φορητό” του δίνει ένα ακόμα πλεονέκτημα, μαζί με το γεγονός ότι χρησιμοποιούνται διαφορετικά εργαλεία ανίχνευσης ευπαθειών (Nessus, OpenVAS, Qualys). Ωστόσο η εργασία αυτή εστιάζει μόνο σε δοχεία MongoDB και όχι δοχεία για οποιοδήποτε είδος συστατικού, όπως η παρούσα διπλωματική, με αποτέλεσμα να μειώνεται σημαντικά το εύρος εφαρμογής του αλγορίθμου της σε σχέση με το προτεινόμενο εργαλείο της διπλωματικής. Επιπροσθέτως, η

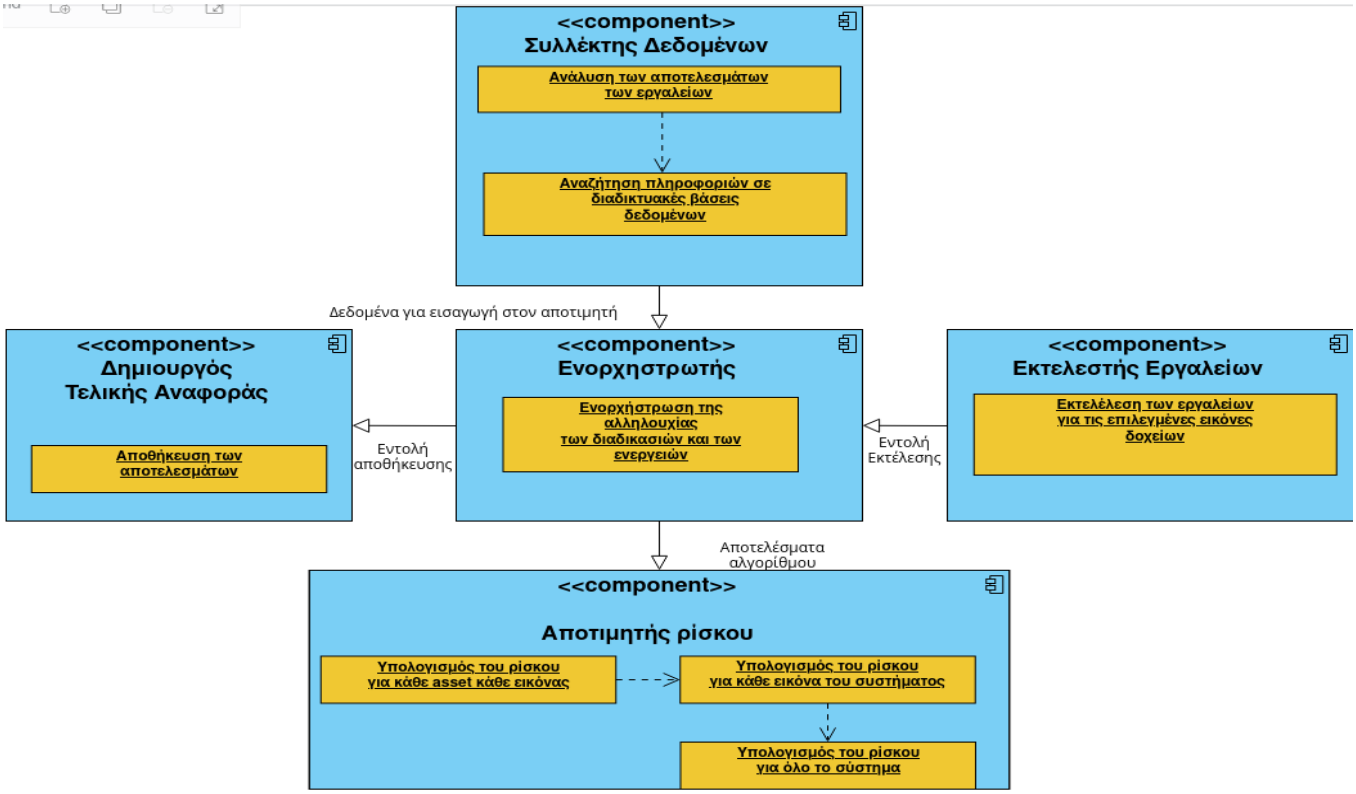
εργασία αυτή εστιάζει σε Singularity δοχεία και όχι σε δοχεία Docker, όπως κάνει η παρούσα διπλωματική εργασία.

Στην εργασία [3] παρουσιάζεται ένας αλγόριθμος για τη βελτίωση της λειτουργίας του μητρώου Docker (Docker registry), επικεντρώνοντας στην αύξηση της δυνατότητας επεκτασιμότητας της, δηλαδή της δυνατότητας διαχείρισης των αιτημάτων που γίνονται προς αυτή, στον έλεγχο πρόσβασης και στην αξιολόγηση εικόνων. Για την αξιολόγηση των εικόνων χρησιμοποιείται το εργαλείο Anchore Grype. Ένα πλεονέκτημα που προσφέρει η εργασία αυτή είναι ο έλεγχος πρόσβασης στο μητρώο Docker προκειμένου να αυξηθεί η ασφάλεια του, ωστόσο η ακρίβεια των αποτελεσμάτων είναι μειωμένη σε σχέση με την παρούσα εργασία, καθότι χρησιμοποιείται μόνο ένα εργαλείο ανίχνευσης τρωτοτήτων.

Η τελευταία εργασία [4] προσπαθεί να αξιολογήσει τα κενά ασφαλείας των εικόνων δοχείων Docker, τα οποία είναι διαθέσιμα δημόσια στο Docker Hub, υλοποιώντας το πλαίσιο DIVA (Docker Image Vulnerability Analysis framework). Το πλαίσιο αυτό χρησιμοποιεί το εργαλείο clair και το common Vulnerability Scoring System (CVSS) για την τελική αξιολόγηση των ευπαθειών. Οι βασικές διαφορές της εργασίας αυτής σε σχέση με αυτής της διπλωματικής είναι οι ακόλουθες. Η χρήση δύο εργαλείων, που υλοποιεί η παρούσα διπλωματική εργασία, αλλά και ο αλγόριθμος υπολογισμού του ρίσκου που λαμβάνει υπόψιν του περισσότερες παραμέτρους και συνδυάζει περισσότερα δεδομένα αυξάνει την ακρίβεια των αποτελεσμάτων όσων αφορά την αξιολόγηση των ευπαθειών.

4. Αρχιτεκτονική του Εργαλείου

Στο παρακάτω διάγραμμα συστατικών απεικονίζεται η αρχιτεκτονική του προτεινόμενου εργαλείου:



Εικόνα 1: Διάγραμμα συστατικών

Τα συστατικά από τα οποία απαρτίζεται το προτεινόμενο εργαλείο είναι τα εξής:

- *Εκτελεστής Εργαλείων Ανίχνευσης Ευπαθειών*
- *Συλλέκτης Δεδομένων*
- *Αποτιμητής Ρίσκου*
- *Δημιουργός Τελικής Αναφοράς*
- *Ενορχηστρωτής*

Συνοπτικά, ο *Εκτελεστής Εργαλείων* ενσωματώνει τα εργαλεία, τα εκτελεί και παρέχει πρόσβαση στα αρχεία αποτελεσμάτων τους. Ο *Συλλέκτης Δεδομένων* υλοποιεί τον αλγόριθμο συλλογής

δεδομένων, έχοντας δυνατότητα επικοινωνίας με εξωτερικές βάσεις δεδομένων ευπαθειών ενώ ο *Αποτιμητής Ρίσκου* υλοποιεί τον αλγόριθμο υπολογισμού του ρίσκου έχοντας ως είσοδο τα αποτελέσματα του αλγορίθμου συλλογής δεδομένων. Ο *Δημιουργός Τελικής Αναφοράς* είναι υπεύθυνος για την αποθήκευση των αποτελεσμάτων και την εκτύπωση τους στην οθόνη. Τέλος, ο *Ενορχηστρωτής* φροντίζει για την ενορχήστρωση της εκτέλεσης της λειτουργικότητας των άλλων τεσσάρων συστατικών και για την σωστή επικοινωνία τους.

Στην συνέχεια, θα αναλύσουμε το κάθε συστατικό της προαναφερόμενης αρχιτεκτονικής σε ξεχωριστές ενότητες του τρέχοντος κεφαλαίου.

4.1 Εκτελεστής Εργαλείων Ανίχνευσης Ευπαθειών

4.1.1 Επιλογή των εργαλείων ανίχνευσης τρωτοτήτων

Όπως αναφέρθηκε και παραπάνω, υπάρχει μια πληθώρα εργαλείων ανίχνευσης τρωτοτήτων που μπορούν να χρησιμοποιηθούν, ωστόσο μετά από την έρευνα και τις δοκιμές, οι οποίες αφορούν την ευκολία εκτέλεσης των εργαλείων, την δομή, το περιεχόμενο και τον τρόπο αποθήκευσης των αποτελεσμάτων τους, που επιτελέσαμε, αποφασίσαμε να χρησιμοποιήσουμε τα εξής δύο (2) εργαλεία:

1. Clair-scanner
2. Grype analyzer

Οι λόγοι αυτής της επιλογής συνοψίζονται στους εξής:

1. Τα 2 αυτά εργαλεία ήταν ευκολότερα στην εγκατάσταση και στην ρύθμιση ώστε να είναι πλήρως λειτουργικά.
2. Μπορούσαν να εξάγουν και να αποθηκεύσουν τα ευρήματα τους σε μορφή JSON ώστε να μπορούν να επεξεργαστούν με τους κατάλληλους αλγορίθμους και να μην είναι αναγκαία η μετατροπή ή μετάφρασή τους από μια άλλη μορφή στην μορφή JSON.
3. Ήταν εύκολα προσβάσιμα μέσα από την πλατφόρμα του Github.
4. Μπορούσαν να εκτελεστούν μέσω τερματικών (Terminal) εύκολα και γρήγορα.
5. Παρέχουν πληροφορία η οποία είναι απαραίτητη για την υλοποίηση του αλγορίθμου αποτίμησης ρίσκου και η οποία δεν παρέχεται από τα υπόλοιπα εργαλεία, όπως η πληροφορία για την κατάσταση επιδιόρθωσης.

4.1.2 Εκτέλεση των εργαλείων ανίχνευσης τρωτοτήτων.

Ο εκτελεστής των εργαλείων, λοιπόν, παρέχει μια διεπαφή, η οποία ενσωματώνει την δυνατότητα εκτέλεσης των εργαλείων αυτών. Έπειτα, εφόσον κληθεί η αντίστοιχη μέθοδος της διεπαφής, χρησιμοποιούνται οι κατάλληλες (σε περιβάλλον Linux) εντολές έτσι ώστε να εκτελεστούν σωστά τα δύο αυτά εργαλεία και να παράγουν το επιθυμητό αποτέλεσμα.

4.2 Συλλέκτης Δεδομένων

4.2.1 Βάσεις δεδομένων άντλησης πληροφοριών

Για την εκτίμηση του ρίσκου για μια εικόνα δοχείου θα πρέπει να χρησιμοποιούνται περισσότερες πληροφορίες σε σχέση με αυτές που διατίθενται από τα 2 προαναφερόμενα εργαλεία ανίχνευσης. Συνεπώς, για την άντληση των πληροφοριών αυτών χρησιμοποιήθηκαν online βάσεις δεδομένων ευπαθειών. Για την αναζήτηση των δεδομένων σε αυτές τις βάσεις αποφασίσαμε ότι το καλύτερο "κλειδί" αποτελεί το πεδίο "cve-id" καθότι χαρακτηρίζει μοναδικά μία ευπάθεια. Υπάρχουν διάφορες βάσεις δεδομένων από τις οποίες μπορούμε να αντλήσουμε τις απαραίτητες πληροφορίες, όπως η National Vulnerability Database (NVD), η οποία παρέχεται τον οργανισμό N.I.S.T (National Institute of Standards and Technology), ο ιστότοπος https://cveapi.com/, ο ιστότοπος https://www.cve-search.org, ο ιστότοπος https://cve.circl.lu/ καθώς και ο ιστότοπος https://vuldb.com/. Μετά από μελέτη των παραπάνω βάσεων δεδομένων καταλήξαμε στο συμπέρασμα ότι οι πλέον κατάλληλες βάσεις είναι η NVD και ο ιστότοπος cveapi. Σαν πρώτη επιλογή ο αλγόριθμος στρέφεται στην NVD διότι αποτελεί την μεγαλύτερη και πιο ευρέως διαδεδομένη βάση δεδομένων όσον αφορά την εύρεση ευπαθειών. Επίσης, ανανεώνεται από έναν επίσημο φορέα με μεγάλο κύρος, πράγμα που αυξάνει την αξιοπιστία της. Ωστόσο, μερικές φορές η αναζήτηση σε αυτή την βάση αποτυγχάνει, διότι μπορεί να μην υπάρχουν πληροφορίες για την συγκεκριμένη ευπάθεια που αναζητείται. Για τον λόγο αυτό, όταν έχουμε αποτυχία της αναζήτησης στην NVD, εκτελούμε αντίστοιχη αναζήτηση στον ιστότοπο cveapi. Ο λόγος που επιλέχθηκε ο συγκεκριμένος ιστότοπος είναι διότι ο όγκος και είδος της πληροφορίας που παρέχεται είναι ίδιος ή παρόμοιος σε σχέση με την NVD, ενώ και οι πληροφορίες είναι δομημένες με παρόμοιο τρόπο και επομένως δεν χρειάζεται να γίνουν σημαντικές αλλαγές στον κώδικα του προγράμματος.

4.2.2 Αλγόριθμος συλλογής δεδομένων

Η πληροφορία που παράγεται από τα εργαλεία εντοπισμού ευπαθειών πρέπει να εμπλουτιστεί από διάφορες πηγές ώστε να παραχθεί μια ολοκληρωμένη πληροφορία ευπαθειών που μπορεί έπειτα να χρησιμοποιηθεί για την αποτίμηση του σχετικού ρίσκου. Επιπλέον, χρειάζεται να ενωθούν οι πληροφορίες που παράγουν τα εργαλεία για τις ευπάθειες σε μια ενιαία πληροφορία. Για αυτούς τους λόγους, αναπτύχθηκε αλγόριθμος συλλογής της απαραίτητης πληροφορίας για ευπάθειες από διάφορες πηγές δεδομένων.

Σε αυτόν τον αλγόριθμο που αναπτύξαμε χρησιμοποιήσαμε ένα συγκεκριμένο και ενιαίο τύπο δεδομένων που θα ήταν ταυτόχρονα εύκολος τόσο στην ανάγνωση από τον άνθρωπο όσο και στην επεξεργασία από κάποια γλώσσα προγραμματισμού. Για τους παραπάνω λόγους επιλέξαμε να αποθηκεύουμε τα αποτελέσματα αυτά σε μορφή JSON. Επιπροσθέτως, ένας ακόμα λόγος που

χρησιμοποιήθηκε αυτή η μορφοποίηση είναι ότι και τα 2 εργαλεία που τελικά επιλέξαμε έχουν την δυνατότητα να αποθηκεύουν τα αποτελέσματά που παράγουν σε μορφή JSON.

Αφού, λοιπόν, εκτελέσουμε τα εργαλεία για κάποια εικόνα δοχείου, αποθηκεύουμε τα δεδομένα που παρήχθησαν σε ένα .json αρχείο με όνομα ένα συνδυασμό του ονόματος της εικόνας και του εργαλείου που χρησιμοποιείται κάθε φορά, μέσα σε ένα φάκελο με όνομα μια χρονοσφραγίδα, η οποία αναφέρεται στην χρονική στιγμή της εκτέλεσης του εργαλείου, συνδυασμένη με ένα αλφαριθμητικό ("scan" ή "risk"). Πριν ξεκινήσουμε την εξόρυξη των δεδομένων από το εκάστοτε αρχείο, πρέπει να εξετάσουμε την δομή του, η οποία και διαφέρει στα δύο εργαλεία. Επομένως, πρέπει να εξετάσουμε το καθένα ξεχωριστά.

Για την υλοποίηση του εργαλείου αποτίμησης ρίσκου μας ενδιαφέρουν οι ακόλουθες τρεις βασικές πληροφορίες, τις οποίες εξαγάγουμε από τα εργαλεία που χρησιμοποιήσαμε, ενώ ο τρόπος που τις λαμβάνουμε θα εξηγηθεί παρακάτω:

(1) *cve-id*. Με τον όρο "cve-id" θα προσδιοριστεί το αναγνωριστικό της κάθε ευπάθειας, το οποίο στην συνέχεια θα χρησιμοποιηθεί τόσο στον αλγόριθμο συλλογής δεδομένων όσο και στον αλγόριθμο υπολογισμού του ρίσκου. Επίσης, είναι το κλειδί με το οποίο θα αναζητήσουμε περισσότερες πληροφορίες για τις ευπάθειες από τις online βάσεις δεδομένων.

(2) *asset name*. Με τον όρο "asset name" προσδιορίζουμε εκείνο το "κομμάτι λογισμικού" της εικόνας που έχει την συγκεκριμένη ευπάθεια - ένα asset μπορεί να έχει παραπάνω από μια ευπάθειες. Αυτή η πληροφορία μας είναι χρήσιμη διότι θέλουμε να προσδιορίσουμε το ρίσκο που υπάρχει σε κάθε asset μιας εικόνας δοχείου.

(3) *fix state*. Με τον όρο "fix state" θα προσδιορίζουμε εάν η συγκεκριμένη ευπάθεια στο συγκεκριμένο "asset" έχει διορθωθεί ή όχι. Εφόσον μια ευπάθεια που έχει εντοπιστεί έχει αντιμετωπιστεί με κάποιο τρόπο, αυτό μειώνει τόσο την πιθανότητα να μπορεί να «εκτελεστεί» όσο και την επίπτωση που μπορεί αυτή να έχει για το σύστημα. Επιπλέον, χρειαζόμαστε αυτή την πληροφορία γιατί χρησιμοποιείται από τον αλγόριθμο υπολογισμού του ρίσκου.

Για να μειωθεί ο χρόνος εκτέλεσης του εργαλείου που κατασκευάζουμε, η επεξεργασία των δεδομένων στα αποτελέσματα των δύο εργαλείων ανίχνευσης (clair-scanner & Grype) βασίζεται σε ένα κοινό HashMap (finalAssets <String,Set>), με τιμή ένα σύνολο από τα cve-id's των ευπαθειών που βρέθηκαν σε κάποιο asset μιας εικόνας και κλειδί το εκάστοτε asset. Με αυτόν τον τρόπο δεν χρειάζεται στην συνέχεια να εξετάσουμε για διπλότυπα και να συγχωνεύσουμε δύο διαφορετικά HashMaps (ένα για την επεξεργασία των αποτελεσμάτων κάθε εργαλείου). Πριν προστεθεί ένα ζεύγος κλειδιού/asset και αναγνωριστικού ευπάθειας στο κοινό HashMap, ελέγχουμε αν υπάρχει ήδη εγγραφή με το ίδιο κλειδί/asset που εξετάζουμε. Εφόσον βρεθεί εγγραφή (ουσιαστικά ένα σύνολο), τότε προσθέτουμε σε αυτήν την νέα ευπάθεια (Σημ.: αν υπάρχει ήδη η ευπάθεια, δεν θα αποθηκευθεί διότι έχουμε να κάνουμε με ένα σύνολο και όχι μια λίστα). Διαφορετικά, αν δεν υπάρχει εγγραφή με το asset, τότε προστίθεται νέα εγγραφή με κλειδί το asset και τιμή ένα κενό σύνολο, στο οποίο εισάγεται η αντίστοιχη, εντοπισμένη ευπάθεια.

Για να εξάγουμε τις παραπάνω πληροφορίες από τα αποτελέσματα του εργαλείου clair-scanner εκτελούμε τα εξής βήματα:

1) Άνοιγμα και διάβασμα του αρχείου που παράγεται από το εργαλείο αυτό.
2) Αναζήτηση για τη λέξη κλειδί "vulnerabilities" και εκχώρηση των αποτελεσμάτων σε ένα in-memory πίνακα JSON (JsonArray).
3) Σε μια επαναληπτική διαδικασία κάνουμε τα εξής βήματα:
    1. Αναζητούμε τις εγγραφές στον πίνακα που αντιστοιχούν στην πληροφορία για το αναγνωριστικό της ευπάθειας ("vulnerability") και το λογισμικό στο οποίο βρέθηκε ("featurename") και τα αποθηκεύουμε στην προαναφερόμενη δομή HashMap (finalAssets <String,–Set>).
    2. Εξετάζουμε εάν το πεδίο "fixedby" είναι κενό ή έχει τιμή. Αν είναι κενό προσθέτουμε μια εγγραφή στο HashMap isFixed (<String, Double>) με κλειδί το cve-id της ευπάθειας και τιμή μηδέν (0.0), διότι δεν υπάρχει κάποια διόρθωση και επομένως η ευπάθεια είναι πιο επικίνδυνη. Σε κάθε άλλη περίπτωση η εγγραφή που προσθέτουμε έχει κλειδί το cve-id της ευπάθειας και τιμή 0.15.

Για το εργαλείο Anchore Grype διεξάγουμε τα εξής βήματα για να αποκτήσουμε πρόσβαση στην πληροφορία που μας ενδιαφέρει:

1. Άνοιγμα και διάβασμα του αρχείου που παράγεται από αυτό το εργαλείο.
2. Όλα τα δεδομένα βρίσκονται "μέσα" σε ένα μεγάλο JsonArray με αναγνωριστικό "matches". Κάθε στοιχείο αυτού του πίνακα περιέχει τις πληροφορίες για μια ευπάθεια που εντοπίστηκε από το εργαλείο.
3. Μέσα σε μία επαναληπτική διαδικασία για κάθε στοιχείο του πίνακα αναζητούμε διαδοχικά τα εξής:
    i. "vulnerability": JsonObject που μας επιστρέφει όλα τα χαρακτηριστικά της ευπάθειας που εντοπίστηκε. Τα πεδία/κλειδιά που μας ενδιαφέρουν είναι τα εξής:
        - "id": String που αντιστοιχεί στο "cve-id" – αναγνωριστικό της ευπάθειας.
        - "fix" : JsonObject που μας δίνει πληροφορίες για το αν υπάρχει διαθέσιμη κάποια επιδιόρθωση για την συγκεκριμένη ευπάθεια.
    ii. "matchdetails": JsonArray που μας δίνει περισσότερες πληροφορίες για το "λογισμικό" στο οποίο εντοπίστηκε η ευπάθεια και περιλαμβάνει την εξής δομή:
        - "searchedBy": JsonObject μέσα στον πίνακα matchdetails.
            - "package": JsonObject μέσα στο προηγούμενο ("searchedBy"). Μας δίνει πληροφορίες για ποιο "κόμματι" της εικόνας έχει την συγκεκριμένη ευπάθεια.
                - "name": String με το όνομα του "λογισμικού" – asset. Βρίσκεται μέσα στο JsonObject "package".
    iii. "artifact": δίνει πληροφορίες για το asset στο οποίο εντοπίσθηκε η ευπάθεια. Ουσιαστικά εδώ έχουμε έναν εναλλακτικό τρόπο να λάβουμε το όνομα του asset εφόσον δεν βρίσκεται μέσα στον matchdetails JsonArray. Περιλαμβάνει το εξής ενδιαφέρον πεδίο/κλειδί:

- “name”: String με το όνομα του “λογισμικού” – asset μέσα στο JsonObject “artifact”.

Σε αυτό το σημείο προσθέτουμε το asset και το cve-id που εντοπίσαμε στην προαναφερόμενη δομή HashMap (finalAssets <String,Set>) με βάση την διαδικασία που αναλύθηκε προηγουμένως.

```
"vulnerability": {
 "id": "CVE-1999-1338",
 "dataSource": "https://nvd.nist.gov/vuln/detail/CVE-1999-1338",
 "namespace": "nvd",
 "severity": "Medium",
 "urls": [
  "http://marc.info/?l=bugtraq&m=93259112204664&w=2"
 ],
 "description": "Delegate proxy 5.9.3 and earlier creates files and directories in the
 "cvss": [
  {
   "version": "2.0",
   "vector": "AV:N/AC:L/Au:N/C:N/I:P/A:N",
   "metrics": {
    "baseScore": 5,
    "exploitabilityScore": 10,
    "impactScore": 2.9
   },
   "vendorMetadata": {}
  }
 ],
 "fix": {
  "versions": [],
  "state": "unknown"
 },
 "advisories": []
},
"relatedVulnerabilities": [],
"matchDetails": [
 {
  "type": "cpe-match",
  "matcher": "ruby-gem-matcher",
  "searchedBy": {
   "namespace": "nvd",
   "cpes": [
    "cpe:2.3:a:*:delegate:0.3.0:*:*:*:*:*:*:*",
    "cpe:2.3:a:delegate:delegate:0.3.0:*:*:*:*:*:*:*"
   ]
  },
  "found": {
   "versionConstraint": "<= 5.9.3 (unknown)",
   "cpes": [
    "cpe:2.3:a:delegate:delegate:*:*:*:*:*:*:*:*"
   ]
  }
 }
],
"artifact": {
 "name": "delegate",
 "version": "0.3.0",
 "type": "gem",
 "locations": [
  {
   "path": "/usr/local/lib/ruby/gems/3.2.0/specifications/default/delegate-0.3.0.gemspe
   "layerID": "sha256:b0aef552c1e3c1a3bd6a8ec58d41a6f793b06da01d892ebd93486fafdbb8e360"
  }
 ],
 "language": "ruby",
 "licenses": [
  "Ruby",
  "BSD-2-Clause"
 ],
 "cpes": [
  "cpe:2.3:a:yukihiro-matsumoto:delegate:0.3.0:*:*:*:*:*:*:*",
  "cpe:2.3:a:yukihiro_matsumoto:delegate:0.3.0:*:*:*:*:*:*:*",
  "cpe:2.3:a:ruby-lang:delegate:0.3.0:*:*:*:*:*:*:*",
  "cpe:2.3:a:ruby_lang:delegate:0.3.0:*:*:*:*:*:*:*",
  "cpe:2.3:a:delegate:delegate:0.3.0:*:*:*:*:*:*:*",
  "cpe:2.3:a:ruby:delegate:0.3.0:*:*:*:*:*:*:*",
  "cpe:2.3:a:*:delegate:0.3.0:*:*:*:*:*:*:*"
 ],
 "purl": "pkg:gem/delegate@0.3.0",
 "upstreams": []
}
},
{
```


Εικόνα 2: Παράδειγμα περιεχομένου αρχείου αποτελεσμάτων που παράγεται από την εκτέλεση του Anchore Grype

Το τελευταίο βήμα για την συλλογή των δεδομένων είναι με βάση τα cve-id που συλλέξαμε να αναζητήσουμε στις online βάσεις δεδομένων τις πληροφορίες που χρειαζόμαστε για την κάθε ευπάθεια. Για να γίνει αυτό, θεωρήσαμε ότι ο πιο αποδοτικός και εύκολος τρόπος να επεξεργαστούμε τις πληροφορίες που αντλήσαμε από αυτές τις βάσεις είναι με την χρήση του πρότυπου JSON.

Για την αναζήτηση στη βάση NVD χρησιμοποιούμε την εξής διεύθυνση URL: https://services.nvd.nist.gov/rest/json/cve/1.0/"cve-id", ενώ για την βάση δεδομένων από τον ιστότοπο cveapi η αντίστοιχη διεύθυνση URL είναι: https://v1.cveapi.com/"cve-id".json

Κατόπιν χρησιμοποιώντας πάλι JsonObject "διαβάζουμε" τα δεδομένα από την εκάστοτε ιστοσελίδα. Ο αλγόριθμος για να το επιτύχουμε αυτό είναι παρόμοιος και στις δύο περιπτώσεις. Στην περίπτωση που αντλούμε τα δεδομένα από την NVD πρέπει να εκτελέσουμε τρία (3) περισσότερα βήματα σε σχέση με την βάση τρωτοτήτων από το cevapi.

Τα βήματα του αλγορίθμου ανά ιστοσελίδα / βάση είναι τα εξής :

1. Δημιουργούμε ένα αντικείμενο JSON με την πληροφορία που αντλήσαμε από την βάση.
2. Αναζητούμε το πεδίο «result» και το αποθηκεύομε σε ένα αντικείμενο Json (j1)
3. Στο παραπάνω αντικείμενο αναζητούμε το πεδίο «CVE_Items» και το αποθηκεύουμε σε ένα JsonArray (j2).
4. Από αυτό το JsonArray παίρνουμε την πρώτη γραμμή και την αποθηκεύουμε σε ένα νέο JsonObject (j3).
5. Σε αυτό το JsonObject αναζητούμε το πεδίο «impact» και το αποθηκεύουμε σε ένα αντικείμενο Json (j4).
6. Σε αυτό το JsonObject αναζητούμε το πεδίο «baseMetricV2» και το αποθηκεύουμε σε ένα αντικείμενο Json (j5).
7. Σε αυτό το JsonObject αναζητούμε το πεδίο «cvssV2» και το αποθηκεύουμε σε ένα αντικείμενο Json (j6).

Αυτό το JsonObject περιέχει την πληροφορία που θέλουμε και θα χρησιμοποιήσουμε αργότερα στον αλγόριθμο (αποτίμησης ρίσκου).

Για την αναζήτηση στη βάση δεδομένων cveapi χρησιμοποιούμε μόνο τα βήματα 1,5,6 και 7.

4.3 Αποτιμητής Ρίσκου

Για τον αλγόριθμο υπολογισμού του ρίσκου βασιστήκαμε στο άρθρο [1].
Για τον υπολογισμό του ρίσκου, βασική προϋπόθεση είναι να μετατρέψουμε τις τιμές των παραμέτρων/μεταβλητών/μετρικών, που περιγράψαμε στην Ενότητα 2.1 Παράμετροι υπολογισμού του ρίσκου, σε πραγματικούς αριθμούς. Οι αντιστοιχήσεις, όπως αντλήθηκαν από το άρθρο [1], παρουσιάζονται στον παρακάτω πίνακα:

Πίνακας 5: Η αντιστοίχιση από ποιοτικές σε αριθμητικές τιμές των βασικών μετρικών του CVSS

Τίτλος Βασικής Μετρικής	**Τιμή Βασικής Μετρικής**	**Τιμή ως Πραγματικός Αριθμός**
crValue	High (H)	0.5
	Medium (M)	0.75
	Low (L)	1.0
irValue	High (H)	0.5
	Medium (M)	0.75
	Low (L)	1.0
arValue	High (H)	0.5
	Medium (M)	0.75
	Low (L)	1.0
Access Vector	Local (L)	0.4
	Adjacent Network (A)	0.6
	Network (N)	1.0
Access Complexity	High (H)	0.5
	Medium (M)	0.75
	Low (L)	1.0
Authentication	Multiple (M)	0.5
	Single (S)	0.55
	None (N)	1.0
Confidentiality	None (N)	0.0
	Partial (P)	0.5
	Complete (C)	1.0
Integrity	None (N)	0.0
	Partial (P)	0.5
	Complete (C)	1.0
Availability	None (N)	0.0
	Partial (P)	0.5
	Complete (C)	1.0

Εφόσον θέσαμε αλγεβρικές τιμές στις παραμέτρους μας, τώρα μπορούμε να αναλύσουμε τον αλγόριθμο που ακολουθείται για τον υπολογισμό του ρίσκου. Ο αλγόριθμος αυτός αποτελείται από 3 μέρη:

(α) υπολογισμός ρίσκου ανά asset εικόνας.
(β) υπολογισμός ρίσκου για κάθε εικόνα.
(γ) υπολογισμός συνολικού ρίσκου συστήματος.

4.3.1 Υπολογισμός Ρίσκου ανά Asset Εικόνας

Έστω ότι σε μια εικόνα έχουμε εντοπίσει N ευπάθειες. Τότε, το ρίσκο για κάθε ευπάθεια j υπολογίζεται πολλαπλασιάζοντας την πιθανότητα να συμβεί η ευπάθεια με την επίπτωση που αυτή θα έχει στο σύστημα σε περίπτωση που εκτελεστεί, δηλαδή:

$$R_j = P_j * IM_j \quad (1)$$

όπου:
R_j είναι το ρίσκο της ευπάθειας,
P_j είναι η πιθανότητα να εκτελεστεί η ευπάθεια,
IM_j είναι η επίπτωση από την εκτέλεση της ευπάθειας.

Η πιθανότητα (P_j) να εκτελεστεί μια ευπάθεια υπολογίζεται ως το γινόμενο τριών παραγόντων:

1. Access Vector
2. Access Complexity
3. Authentication

Ο λόγος που χρησιμοποιούνται οι τρεις παραπάνω παράγοντες είναι διότι αυτοί αποτελούν τις βασικές μετρικές του συστήματος βαθμολόγησης που χρησιμοποιούμε.
Επομένως, ο τύπος υπολογισμού της πιθανότητας είναι ο εξής:

$P_j = avValue_j * acValue_j * auValue_j$ (2).

Η επίπτωση (IM_j) που θα έχει η εκτέλεση της ευπάθειας υπολογίζεται από τον παρακάτω τύπο:

$I_j = \big(ImpactAvailability_j + ImpactIntegrity_j + ImpactConfidentiality_j\big)/\big(3 * (1 - RL)\big)$. ή καλύτερα
$IM_j = \big(IA_j + II_j + IC_j\big)/\big(3 * (1 - RL)\big)$. (3).

Όπου :

- $IA_j = Availability_j * arValue_j$
- $II_j = Integrity_j * irValue_j$
- $IC_j = Confidentiality_j * crValue_j$

όπου οι παράμετροι $Availability_j$, $Integrity_j$ και $Confidentiality_j$ είναι αυτοί που αντλήσαμε από την online βάση δεδομένων και οι αντίστοιχες απαιτήσεις AR_j, IR_j και CR_j (σημασιολογικά Availability Requirement, Integrity Requirement και Confidentiality Requirement, αντίστοιχα) δίνονται από τον διαχειριστή του συστήματος ως παράμετροι εισόδου κατά την εκκίνηση του προγράμματος. Οι τιμές των απαιτήσεων αυτών εξαρτώνται από την φύση και τις απαιτήσεις του εκάστοτε συστήματος / εταιρίας αλλά και το περιεχόμενο κάθε εικόνας δοχείου.

Έστω τώρα ότι σε μία εικόνα υπάρχουν k assets (κομμάτια λογισμικού στα οποία έχει εντοπιστεί μια ευπάθεια). Προκειμένου να αυξήσουμε την ακρίβεια του αλγορίθμου μας πρέπει να λάβουμε υπόψιν και τα assets στα οποία βρέθηκαν οι ευπάθειες. Ένας τρόπος για να γίνει αυτό είναι η υλοποίηση της συνάρτησης *weighted_consolid,* η οποία προσπαθεί να εντοπίσει το γεγονός πως τουλάχιστον μια τρωτότητα θα συμβεί για ένα συγκεκριμένο asset k και έπειτα πολλαπλασιάζει την πιθανότητα του γεγονότος αυτού με ένα άθροισμα με βάρη των επιπτώσεων των τρωτοτήτων, όπου το κάθε βάρος είναι η σταθισμένη πιθανότητα της αντίστοιχης τρωτότητας. Ο τύπος υπολογισμού είναι ο εξής :

$$R_k = Paone_k * WR_k = \left(\prod_j P_j\right)\frac{\sum_j P_j IM_j}{\sum_j P_j}$$

$$R_k = Paone_k * WR_k = \left(\prod_j P_j\right) \Big/ \left(\sum_j P_j \sum_j P_j\, IM_j\right)$$

όπου:
$Paone_k$: είναι η πιθανότητα τουλάχιστον μια τρωτότητα να συμβεί στο asset k.
WR_k: είναι ένα άθροισμα με βάρη των επιπτώσεων των τρωτοτήτων.

4.3.2 Υπολογισμός Ρίσκου Ανά Εικόνα

Για τον υπολογισμό του ρίσκου σε ολόκληρη την εικόνα δοχείου χρησιμοποιείται μια από τις παρακάτω 4 συναρτήσεις, ανάλογα με την επιλογή του διαχειριστή:

1. *Avg*: δέχεται σαν είσοδο τις τιμές των ρίσκων που υπολογίστηκαν για τα asset που περιέχει η εικόνα και υπολογίζει τον μέσο όρο τους. Επομένως το ολικό ρίσκο για μια εικόνα υπολογίζεται από τον τύπο: $IR_i = \sum_k R_k$ / |K| (5) όπου IR_k είναι το συνολικό ρίσκο της εικόνας i, R_k είναι το ρίσκο για ένα asset k της εικόνας και |K| είναι το πλήθος των assets της εικόνας.
2. *Max*: θεωρεί ότι το ρίσκο της εικόνας είναι ίσο με το μεγαλύτερο ρίσκο που υπολογίστηκε στα assets της εικόνας. Ειδικότερα, $IR_i = ax_k\{R_k\}$ (6)
3. *Weight_aggr_manual*: υπολογίζει το ρίσκο με βάση κάποια βάρη τα οποία θέτει ο διαχειριστής και σχετίζονται με το πόσο σημαντικό ή κρίσιμο είναι το ένα asset k για την εκάστοτε εικόνα σύμφωνα με τον τύπο: $IR_i = \sum_k (w_k * R_k) / \sum_k w_k$ (7) όπου:
    1. W_k είναι το "βάρος" που έδωσε ο διαχειριστής για το asset k της εικόνας.
    2. R_k είναι το ρίσκο που έχει υπολογισθεί για αυτό το asset k .

4. *Weight_aggr_auto*: υπολογίζει το ρίσκο με βάση κάποια βάρη, τα οποία τίθενται αυτόματα από το σύστημα βάσει πιθανοτήτων, ως εξής:
   $IR_i = \sum_k (Paone_k * R_k) / \sum_k Paone_k$ (8)
   όπου :
   - $Paone_k$ είναι η πιθανότητα να συμβεί τουλάχιστον μια τρωτότητα από αυτές που έχουν εντοπιστεί για το asset k της εικόνας i.
   - R_k είναι το ρίσκο που έχει υπολογισθεί για το asset k.

4.3.3 Υπολογισμός Συνολικού Ρίσκου Συστήματος

Τέλος, ο υπολογισμός του ρίσκου (R) για ολόκληρο το σύστημα γίνεται με παρόμοιο τρόπο. Χρησιμοποιούμε αυτή τη φορά πέντε πιθανές συναρτήσεις σύνθεσης των ρίσκων για τις εικόνες του συστήματος:

1. Avg: δέχεται ως είσοδο τα υπολογισμένα ρίσκα για κάθε εικόνα i και έπειτα υπολογίζει τον μέσο όρο τους. Επομένως, το ολικό ρίσκο του συστήματος υπολογίζεται από τον τύπο: $R = \sum_i R_i / \sum_i |i|$ (9)
   όπου $|i|$ είναι το συνολικό πλήθος εικόνων του συστήματος.
2. Max: θεωρεί ότι το ρίσκο του συστήματος R είναι το μεγαλύτερο ρίσκο που υπολογίστηκε στις εικόνες του. $R = \{max_i\}(R_i)$.
3. Weight_aggr_manual: υπολογίζει το ρίσκο σύμφωνα με χειρωνακτικά βάρη (που παρέχονται από τον διαχειριστή) με βάση τον ακόλουθο τύπο: $R = \sum_i (W_i R_i) / \sum_i R_i$, (10) όπου:
   1. w_i είναι το "βάρος" που έδωσε ο διαχειριστής για την εικόνα i και σχετίζεται με το πόσο σημαντική ή κρίσιμη είναι η εικόνα για το συνολικό σύστημα.
   2. R_i είναι το ρίσκο που έχει υπολογισθεί για την εικόνα i.
4. Weight_aggr_auto: υπολογίζει το ρίσκο με βάση κάποια βάρη, τα οποία τίθενται αυτόματα από το σύστημα βάσει πιθανοτήτων, ως εξής: $R = \sum_i (P_i * R_i) / \sum P$ (11) όπου :
   - P_i είναι η πιθανότητα να συμβεί τουλάχιστον μια τρωτότητα j από αυτές που έχουν εντοπιστεί για την εικόνα i, με τύπο:
     $P_i = 1 - \prod_j (1 - Paone_j)$ (12).
   - R_i είναι το ρίσκο που έχει υπολογισθεί για την εικόνα i.
5. Ταυτοτική: Στην περίπτωση που έχουμε μια εικόνα το ρίσκο του συστήματος ισοδυναμεί με το ρίσκο της εικόνας (επιλογή no).

4.4 Δημιουργός Τελικής Αναφοράς

Ο ρόλος του είναι να παίρνει τα αποτελέσματα από τον αποτιμητή ρίσκου και να τα αποθηκεύει σε ένα αρχείο στο κατάλληλο φάκελο στο σύστημα αρχείων καθώς και να τα απεικονίζει στην οθόνη. Το αρχείο αυτό θα περιέχει πληροφορίες για ευπάθειες σε assets που εντοπίστηκαν σε κάθε εικόνα

του συστήματος καθώς και για το υπολογισθέν ρίσκο τόσο για τις εικόνες όσο και για το συνολικό σύστημα. Το όνομα του αρχείου είναι πάντοτε results.json.

4.5 Ενορχηστρωτής

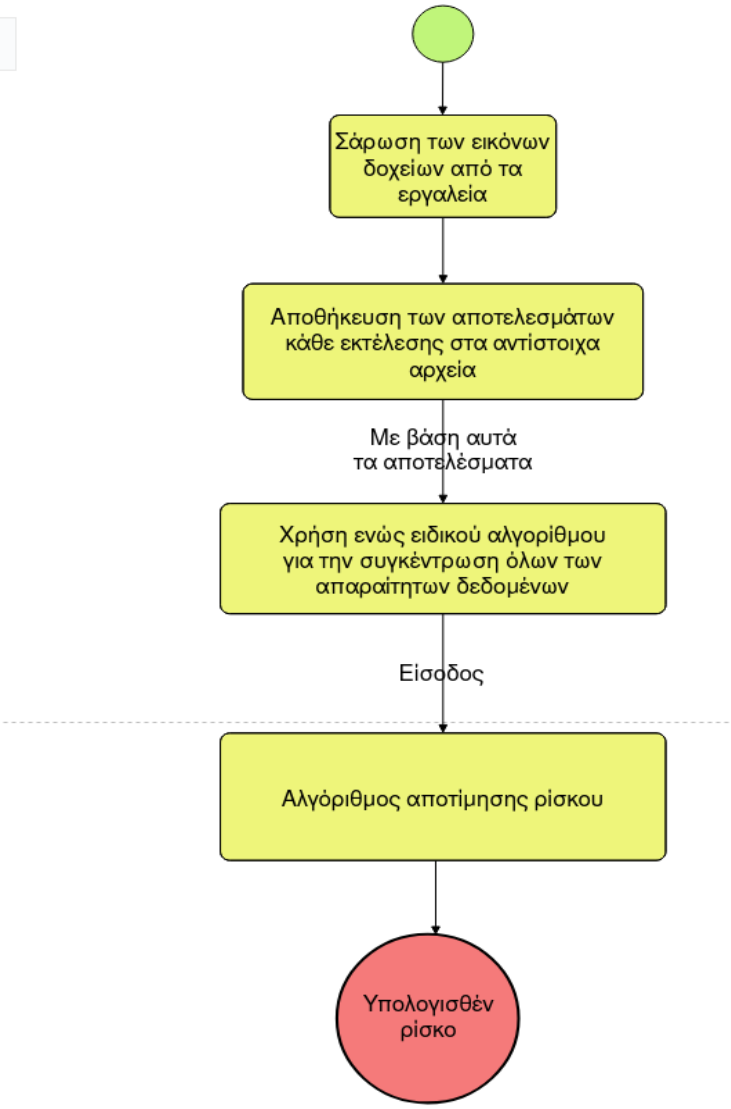
Ο ρόλος του ενορχηστρωτή είναι να εξασφαλίζει την ομαλή λειτουργία των άλλων τεσσάρων συστατικών καθώς και την ενορχηστρωμένη εκτέλεσή τους.

4.6 Συνολική Διαδικασία Αποτίμησης Ρίσκου

Η συνολική διαδικασία που ακολουθεί το εργαλείο για την αποτίμηση του ρίσκου ενός συστήματος είναι η εξής (δείτε και την Εικόνα 3):

1. Αρχικά γίνεται η σάρωση των εικόνων που έχει επιλέξει ο διαχειριστής του συστήματος από τα δύο (2) εργαλεία (υπεύθυνο συστατικό: *Εκτελεστής Εργαλείων Ανίχνευσης Ευπαθειών*).
2. Κατόπιν τα αποτελέσματα από κάθε εκτέλεση αποθηκεύονται αυτόματα στα αντίστοιχα αρχεία σε μορφή JSON (υπεύθυνο συστατικό: *Εκτελεστής Εργαλείων Ανίχνευσης Ευπαθειών*).
3. Στη συνέχεια με βάση τα αποτελέσματα που παρήγαγαν τα εργαλεία και με την βοήθεια ενός ειδικού αλγορίθμου (αλγόριθμος συλλογής δεδομένων) συγκεντρώνονται όλες οι απαραίτητες πληροφορίες για τις ευπάθειες (υπεύθυνο συστατικό: *Συλλέκτης Δεδομένων*).
   - Για το παραπάνω βήμα εκτός από τα αποτελέσματα των εργαλείων ανίχνευσης χρησιμοποιούνται και διαδικτυακές βάσεις δεδομένων, από τις οποίες με τα κατάλληλα διαμορφωμένα αιτήματα αντλούμε τα δεδομένα.
4. Τέλος, οι πληροφορίες που συλλέξαμε στο προηγούμενο βήμα χρησιμοποιούνται ως είσοδος για τον αλγόριθμο υπολογισμού του ρίσκου που υλοποιείται σε αυτό το βήμα (υπεύθυνο συστατικό: *Δημιουργός Τελικής Αναφοράς*).

Άρα, λοιπόν, βλέπουμε ότι η αλληλεπίδραση των συστατικών του συστήματος είναι άμεση και κάθε αλλαγή στην είσοδο σε ένα από τα συστατικά αυτά μπορεί να επιφέρει μια μικρότερη ή μεγαλύτερη αλλαγή στο τελικό αποτέλεσμα.



Εικόνα 3: Συνολική διαδικασία αποτίμησης ρίσκου

5. Υλοποίηση

5.1 Ανάλυση υλοποίησης

Το πρόγραμμα υλοποιήθηκε στη γλώσσα προγραμματισμού Java και χρησιμοποιήθηκε το Eclipse ως το ενοποιημένο περιβάλλον ανάπτυξης (Integrated Development Environment – IDE) .

Ο κώδικάς μας οργανώθηκε σε πακέτα όπου κάθε πακέτο αντιστοιχεί σε ένα βασικό συστατικό του εργαλείου μας. Τα πακέτα αυτά είναι τα εξής:

- *Ενορχηστρωτής*:
    1. Orchestrator class
- *Wrapper (Εκτελεστής Εργαλείων Ανίχνευσης Ευπαθειών)*, το οποίο περιλαμβάνει τις εξής κλάσεις:
    1. GrypeRunner class
    2. ClairScannerRunner class

    καθώς και την διεπαφή ToolRunner
- *Συλλέκτης Δεδομένων*, το οποίο περιλαμβάνει τις κλάσεις
    1. RetrieveDataFromOnlineDB class
    2. GrypeAnalyzer class
    3. ClairAnalyzer class

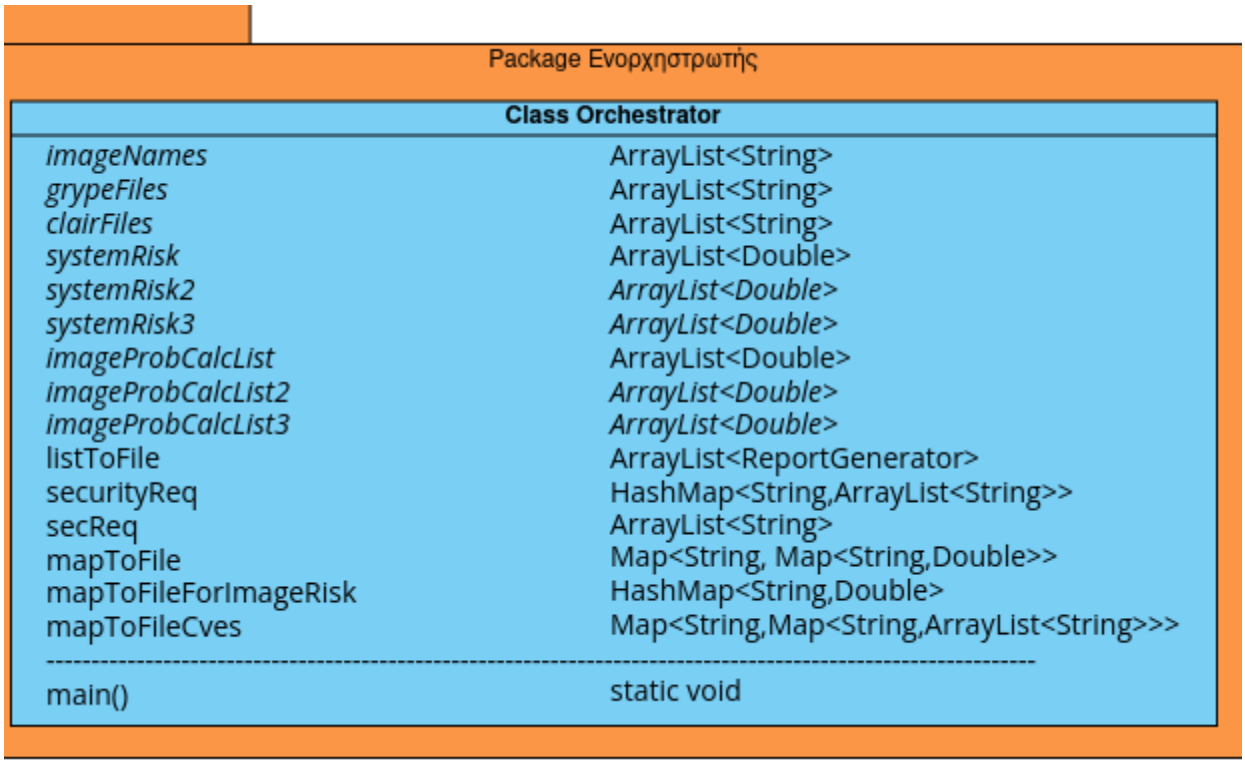
    και την διεπαφή: ToolReportAnalyzer
- *Αποτιμητής Ρίσκου* που περιλαμβάνει τις εξής κλάσεις:
    1. CalcAssetRisk class
    2. ImageRiskCalcAlgo class
    3. OverallRisk class
- *Δημιουργός Τελικής Αναφοράς*:
    1. ReportGenerator class

Παρακάτω αναλύουμε σε μεγαλύτερο βαθμό λεπτομέρειας το περιεχόμενο των προαναφερόμενων πακέτων.

5.1.1 Ενορχηστρωτής

Orchestrator class

Η κλάση Orchestrator είναι η κύρια κλάση του εργαλείου. Είναι η κλάση που ελέγχει την ροή των δεδομένων και της εκτέλεσης των συστατικώ. Μέσω αυτής εισάγονται στο εργαλείο οι επιλογές του διαχειριστή, βάσει των οποίων θα γίνει η αποτίμηση του ρίσκου. Η μοναδική μέθοδος σε αυτή τη κλάση είναι η main(), η οποία διαχειρίζεται την ροή εκτέλεσης & δεδομένων του εργαλείου.



Εικόνα 4: Το πακέτο Ενορχηστρωτής με τα περιεχόμενα της κλάσης του

5.1.2 Εκτελεστής Εργαλείων (ToolRunner)

Το συστατικό αυτό (*Εκτελεστής Εργαλείων*) αντιστοιχεί σε μια διεπαφή (ToolRunner), η οποία θα πρέπει να υλοποιηθεί από κλάσεις που περιτυλίγουν την κύρια λειτουργικότητα του κάθε εργαλείου ανίχνευσης.

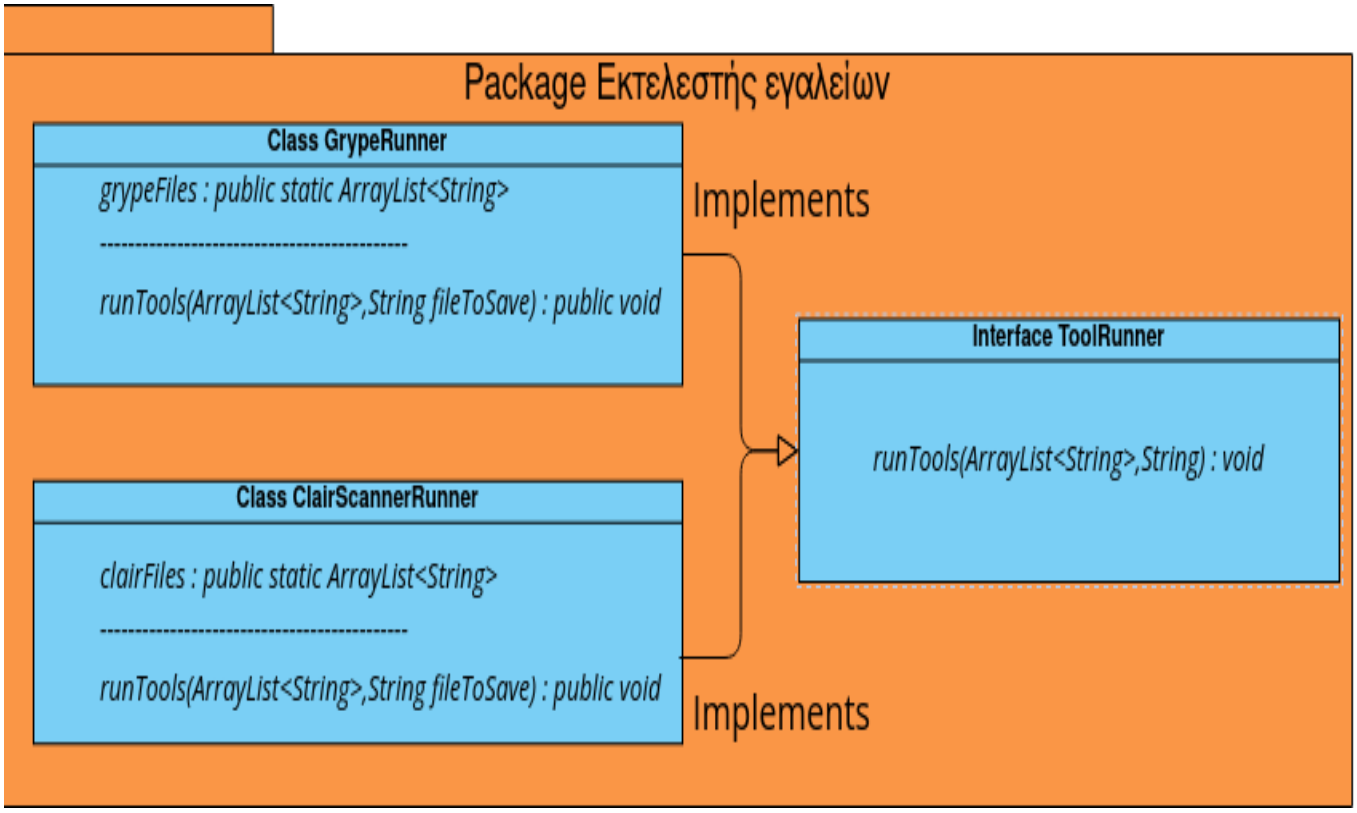
Διεπαφή ToolRunner:
Η διεπαφή ToolRunner περιέχει την μέθοδο runTools(), η οποία υλοποιείται από τις κλάσεις, οι οποίες είναι υπεύθυνες για την εκτέλεση των δύο εργαλείων που χρησιμοποιούμε, αντίστοιχα. Στόχος της μεθόδου είναι να εξασφαλίσει την σωστή και αποτελεσματική εκτέλεση των εργαλείων.

Οι κλάσεις που αναλαμβάνουν την εκτέλεση των εργαλείων είναι οι εξής:

- ClairScannerRunner: είναι υπεύθυνη για την εκτέλεση του εργαλείου clair-scanner.

- GrypeRunner: είναι υπεύθυνη για την εκτέλεση του εργαλείου Grype.

Ουσιαστικά αυτές οι κλάσεις χρησιμοποιούν την αντίστοιχη εντολή/εντολές λειτουργικού συστήματος για να εκτελεστεί το εκάστοτε εργαλείο, σαν να το εκτελούσαμε από το τερματικό (terminal) του linux, και αποθηκεύουν τα αποτελέσματα του σε αρχείο με μορφοποίηση JSON.



Εικόνα 5: Το πακέτο Εκτελεστής Εργαλείων

Αναλυτικότερα για τις κλάσεις έχουμε τα εξής:

GrypeRunner class

Η κλάση αυτή περιέχει μία και μόνο ιδιότητα, μια λίστα (grypeFiles) στην οποία αποθηκεύονται τα ονόματα των αρχείων που περιέχουν τα αποτελέσματα από την εκτέλεση του εργαλείου Grype για την εκάστοτε εικόνα δοχείου:

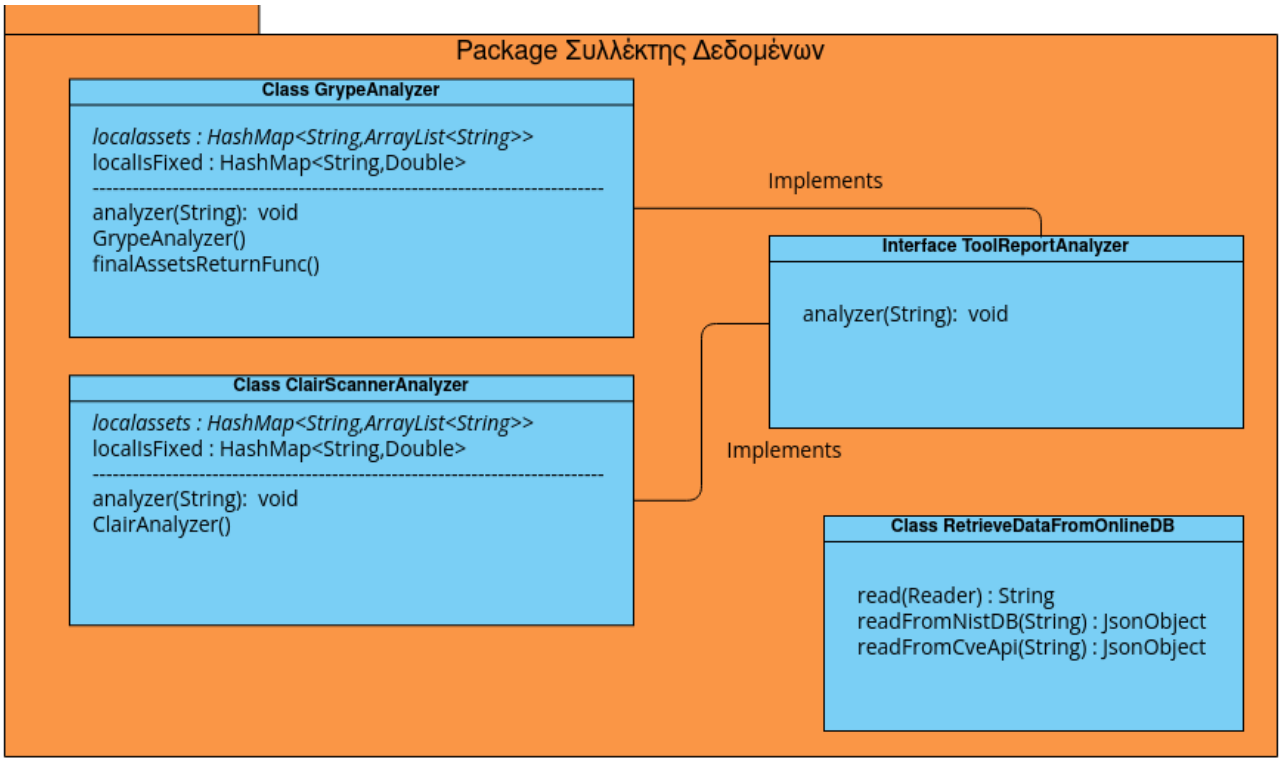
```
public static ArrayList<String> grypeFiles
```

ClairScannerRunner class

Η κλάση αυτή περιέχει μία και μόνο ιδιότητα, μια λίστα (*clairFiles*) στην οποία αποθηκεύονται τα ονόματα των αρχείων που περιέχουν τα αποτελέσματα από την εκτέλεση του εργαλείου clair-scanner για την εκάστοτε εικόνα δοχείου:

```
public static ArrayList clairFiles
```

5.1.3 Συλλέκτης Δεδομένων (Data Collector)



Εικόνα 6: Το πακέτο Συλλέκτης Δεδομένων

Σε αυτό το πακέτο (*Συλλέκτης Δεδομένων*), τον ρόλο της επεξεργασίας των δεδομένων που παράγουν οι δύο προαναφερόμενες κλάσεις (του πακέτου *Εκτελεστής Εργαλείων*) έχουν επιφορτιστεί οι κλάσεις ClairAnalyzer και GrypeAnalyzer για τα εργαλεία clair-scanner και grype, αντίστοιχα. Οι δύο τελευταίες κλάσεις υλοποιούν την διεπαφή ToolReportAnalyzer.

Διεπαφή ToolReportAnalyzer
Η διεπαφή αυτή περιέχει την μέθοδο analyzer(String), την οποία χρησιμοποιούν οι κλάσεις GrypeAnalyzer και ClairAnalyzer προκειμένου να μπορέσουν να απομονώσουν από τα αποτελέσματα των εργαλείων τις πληροφορίες εκείνες που μας είναι απαραίτητες στον αλγόριθμο αποτίμησης ρίσκου που έχουμε αναπτύξει.

GrypeAnalyzer class και ClairAnalyzer class
Και οι δύο κλάσεις περιέχουν τις εξής δύο ιδιότητες:

- HashMap με όνομα localassets, το οποίο χρησιμοποιούμε για να υπολογίσουμε το συνολικό ρίσκο αν χρησιμοποιούσαμε ξεχωριστά το κάθε εργαλείο.
- HashMap με όνομα localIsFixed, το οποίο χρησιμοποιείται στην διαδικασία που περιγράφηκε στο προηγούμενο bullet.

Οι κοινές μέθοδοι που ενσωματώνουν οι κλάσεις αυτές είναι τρεις:

- Ο προκαθορισμένος δομητής.

- η μέθοδος με όνομα `analyzer()` από την διεπαφή ToolReportAnalyzer. Δέχεται σαν παράμετρο το όνομα του αρχείου που είναι αποθηκευμένα τα αποτελέσματα από την εκτέλεση του εκάστοτε εργαλείου για κάθε εικόνα και με μια σειρά από εντολές απομονώνει τις πληροφορίες που μας ενδιαφέρουν και τις αποθηκεύει σε μια δομή HashMap με όνομα finalAssets.
- η μέθοδος `finalAssetsReturnFunc()` που, όπως μαρτυράει και το όνομα της, επιστρέφει το HashMap finalAssets (δείτε Ενότητα 4.2.2 Αλγόριθμος συλλογής δεδομένων). Η τελευταία δομή περιέχει τα assets αντιστοιχισμένα με όλες τις ευπάθειες που έχουν εντοπιστεί σε αυτά.

Αυτές οι δύο κλάσεις είναι που υλοποιούν και τον αλγόριθμο συλλογής δεδομένων. Σε αυτή τη φάση έχουμε μετατρέψει τα δεδομένα μας σε μια μορφή, η οποία μπορεί να χρησιμοποιηθεί στη συνέχεια από τον αλγόριθμο υπολογισμού του ρίσκου.

Οι δομές finalAssets και isFixed βρίσκονται στην κλάση CalcAssetRisk - η οποία αναλύεται παρακάτω στην Ενότητα 5.1.4 Αποτιμητής Ρίσκο. Ουσιαστικά, είναι οι βασικές δομές για την υλοποίηση του απομιμητή ρίσκου. Χρησιμοποιούνται στις κλάσεις που αναλύουμε εδώ για να αποθηκεύσουν τα αποτελέσματα από την εκτέλεση των εργαλείων ανίχνευσης ευπαθειών. Ο λόγος που γίνεται αυτό είναι πως, αντί να υπάρχει ξεχωριστή υλοποίηση της κάθε δομής στις δύο παραπάνω κλάσεις, χρησιμοποιούνται οι δομές της κλάσης CalcAssetRisk για εξοικονόμηση χρόνου. Αν κάθε εργαλείο αποθήκευε σε ξεχωριστή δομή τα αποτελέσματα της εκτέλεσης του για μια εικόνα, τότε θα έπρεπε στη συνέχεια να συγκριθούν αυτές οι δομές για διπλότυπα, ενώ με αυτόν τον τρόπο η σύγκριση αυτή γίνεται αυτόματα, κατά την εισαγωγή των στοιχείων στη δομή.

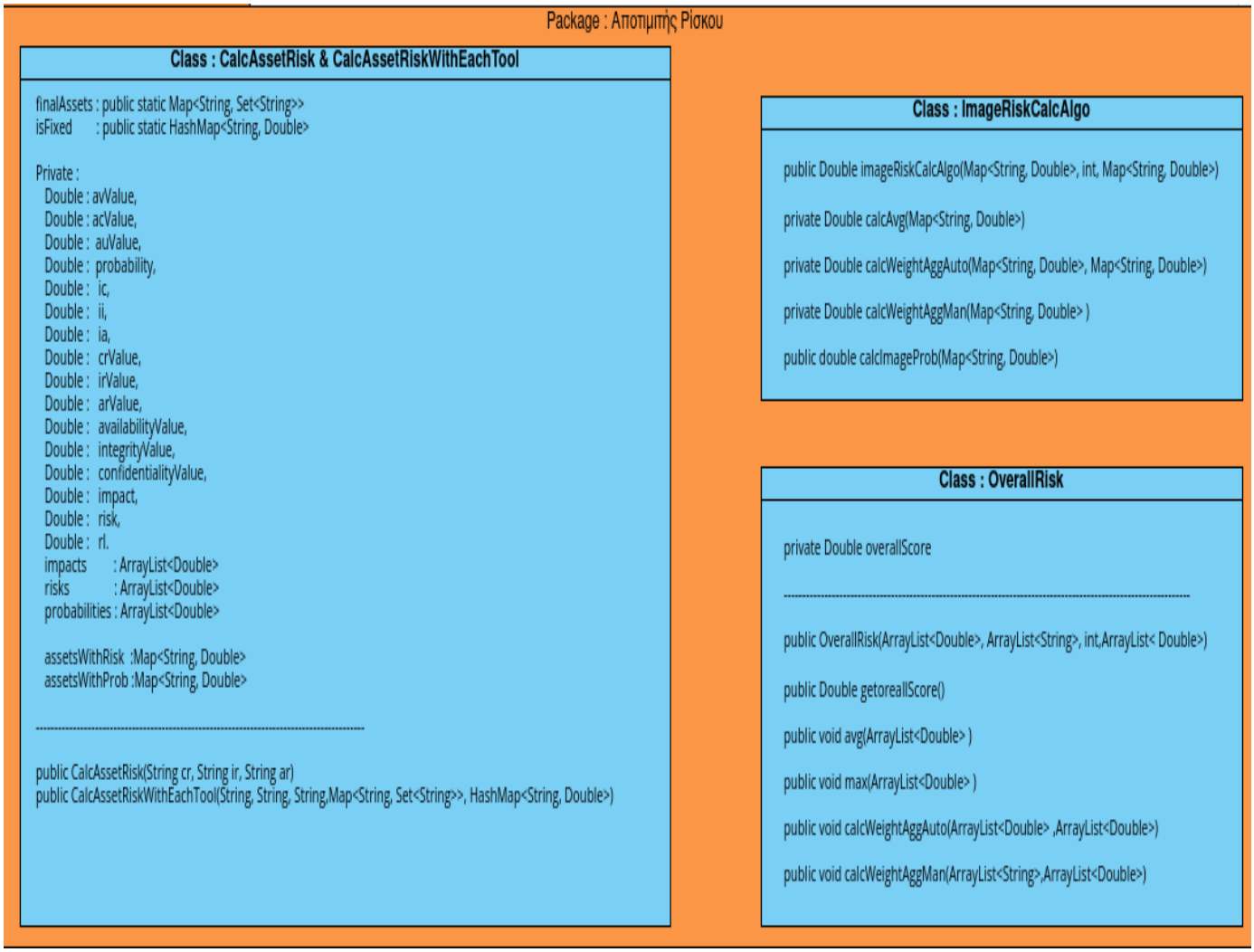
RetrieveDataFromOnlineDB class

Αυτή η κλάση χρησιμοποιείται για να "διαβάζουμε" τα δεδομένα από την εκάστοτε διαδικτυακή βάση δεδομένων (είτε NVD είτε cveapi). Οι δύο αυτές βάσεις χρησιμοποιούνται σειριακά, δηλαδή πρώτα αναζητούμε την πληροφορία που θέλουμε στην βάση NVD και αν δεν βρεθούν αποτελέσματα, τότε προχωράμε την αναζήτηση μας στην βάση cveapi,
Η κλάση RetrieveDataFromOnlineDB περιέχει τις εξής μεθόδους:

1. `read()`. Η μέθοδος αυτή επιστρέφει ένα αλφαριθμητικό, το οποίο είναι σε μορφή JSON και περιέχει όλη την πληροφορία που ανακτήθηκε από την εκάστοτε διαδικτυακή βάση δεδομένων, ενώ παίρνει σαν παράμετρο ένα αντικείμενο της κλάσης Reader της Java.
2. `readFromNistDB()` και `readFromCveApi()`. Οι μέθοδοι αυτές είναι παρόμοιες. Λαμβάνου ως είσοδο το αντικείμενο που παρήγαγε η μέθοδος `read()`και επιστρέφουν ένα JsonObject στο οποίο έχει απομονωθεί, από την συνολική πληροφορία που ανακτήθηκε από την βάση δεδομένων NVD και cevapi αντίστοιχα, το κομμάτι εκείνο που περιέχει τις πληροφορίες που θα χρησιμοποιηθούν στη συνέχεια για τον υπολογισμό του ρίσκου.

Η ύπαρξη των δύο τελευταίων διαφορετικών συναρτήσεων είναι αναγκαία διότι, τόσο η αναπαράσταση των δεδομένων στις δύο βάσεις όσο και η δομή τους αυτή καθ' αυτή είναι διαφορετική.

5.1.4 Αποτιμητής Ρίσκου



Εικόνα 7: Το πακέτο Αποτιμητής Ρίσκου

CalcAssetRisk class

Η κλάση αυτή χρησιμοποιείται για τον υπολογισμό του ρίσκου σε κάθε asset μιας εικόνας. Διαβάζει τις πληροφορίες από τα αποτελέσματα των εργαλείων αλλά και αυτές που υπάρχουν στις διαδικτυακές βάσεις δεδομένων, προχωρά στις απαραίτητες μετατροπές από αλφαριθμητικές σε αλγεβρικές τιμές και τέλος εκτελεί τις απαραίτητες πράξεις προκειμένου να υπολογισθεί το ρίσκο σε ένα asset.

Η μοναδική μέθοδος που περιέχει είναι ο δομητής, ο οποίος δέχεται σαν παράμετρο τρεις μεταβλητές τύπου αλφαριθμητικού, οι οποίες αναπαριστούν τις απαιτήσεις ασφάλειας και στο

τέλος υπολογίζει το ζητούμενο ρίσκο για κάθε asset της εικόνας, το οποίο τελικά θα αποθηκευτεί στην μεταβλητή finalRisk, ενώ το ζεύγος asset – risk θα αποθηκευτεί στην δομή assetWithRisk.

CalcAssetRiskWithEachTool class

Η κλάση αυτή είναι υπεύθυνη για τον υπολογισμό του ρίσκου χρησιμοποιώντας κάθε εργαλείο ξεχωριστά, χωρίς να συνδυάζονται τα αποτελέσματά τους.

Η δομή της είναι παρόμοια με την κλάση CalckAssetRisk με την διαφορά ότι στον κατασκευαστή της δέχεται σαν παράμετρο 2 Map (δείτε επόμενη παράγραφο). Μια άλλη διαφορά είναι ότι αυτή η κλάση καλείται ξεχωριστά για κάθε εργαλείο και όχι για τα συγκεντρωτικά αποτελέσματά.

Η μοναδική μέθοδος που περιέχει είναι ο κατασκευαστής, ο οποίος δέχεται σαν παράμετρο τρεις μεταβλητές τύπου αλφαριθμητικού, οι οποίες αναπαριστούν:

- τις απαιτήσεις ασφάλειας
- ένα Map που περιέχει όλα τα assets μιας εικόνας αντιστοιχισμένα με τα cve ids που βρέθηκαν σε αυτά (localAssets)
- ένα Map που περιέχει την πληροφορία για το αν έχει βρεθεί κάποιο fix(localIsFixed) για κάθε εντοπισμένη ευπάθεια στην εικόνα

και στο τέλος υπολογίζει το ζητούμενο ρίσκο.

ImageRiskCalcAlgo class

Η κλάση αυτή χρησιμοποιείται για τον υπολογισμό του ρίσκου σε μια εικόνα δοχείου. Η κλάση αυτή δεν περιέχει καμία ιδιότητα, αλλά μόνο έξι μεθόδους, από τις οποίες οι τέσσερεις πρώτες χρησιμοποιούνται για τον υπολογισμό του ρίσκου σε μια εικόνα δοχείου, ανάλογα με την επιλογή που έκανε ο διαχειριστής κατά την εκκίνηση του εργαλείου, η πέμπτη είναι βοηθητική για τον υπολογισμό του ρίσκου σε ορισμένες μόνο περιπτώσεις (`calcImageProb()`), ενώ η τελευταία είναι ο κατασκευαστής της κλάσης. Αναλυτικότερα:

1. `calcMax()`. Μέθοδος που βρίσκει το μεγαλύτερο από τα υπολογισμένα ρίσκα για τα assets και ορίζει αυτό ως το ρίσκο για όλη την εικόνα. Δέχεται σαν παράμετρο το Map assetsWithRisks που περιέχει τα assets που βρέθηκαν σε μια εικόνα αντιστοιχισμένα με το ρίσκο που υπολογίσθηκε για αυτά και επιστρέφει μια μεταβλητή τύπου Double, η οποία αντιστοιχεί στο υπολογισθέν ρίσκο.
2. `calcAvg()`. Μέθοδος που υπολογίζει το ρίσκο σε μια εικόνα, υπολογίζοντας τον απλό μέσο όρο των ρίσκων των assets που εντοπίστηκαν σε αυτή την εικόνα. Δέχεται σαν παράμετρο το Map assetsWithRisks που περιέχει τα assets που βρέθηκαν σε μια εικόνα αντιστοιχισμένα με το ρίσκο που υπολογίσθηκε για αυτά και επιστρέφει μια μεταβλητή τύπου Double, η οποία αντιστοιχεί στο υπολογισθέν ρίσκο.
3. `calcWeightAggAuto()`. Μέθοδος που υπολογίζει το ρίσκο σε μια εικόνα, με βάση κάποια αυτόματα βάρη που έχουν τοποθετηθεί στα assets που εντοπίστηκαν σε αυτή την εικόνα. Αυτά τα βάρη υπολογίζονται με βάση την πιθανότητα να εκτελεστεί κάποια από τις ευπάθειες που περιέχουν. Δέχεται σαν παράμετρο το Map assetsWithRisks που περιέχει τα

assets που βρέθηκαν σε μια εικόνα αντιστοιχισμένα με το ρίσκο που υπολογίσθηκε για αυτά καθώς και το Map assetsWithProb, το οποίο περιέχει όλα τα assets που βρέθηκαν σε μια εικόνα, αντιστοιχισμένα με την πιθανότητα εκτέλεσης τουλάχιστον μίας ευπάθειας από αυτές που περιέχουν. Επιστρέφει μια μεταβλητή τύπου Double, η οποία αντιστοιχεί στο υπολογισθέν ρίσκο.

4. `calcWeightAggMan()`. Μέθοδος που υπολογίζει το ρίσκο σε μια εικόνα, με βάση κάποια βάρη, τα οποία θέτει ο διαχειριστής του συστήματος κατά την εκτέλεση του εργαλείου για κάθε asset που υπάρχει σε αυτή την εικόνα. Δέχεται σαν παράμετρο το Map assetsWithRisks που περιέχει τα assets που βρέθηκαν σε μια εικόνα αντιστοιχισμένα με το ρίσκο που υπολογίσθηκε για αυτά. Λαμβάνοντας υπόψιν τα βάρη που δίνει ο διαχειριστής επιστρέφει μια μεταβλητή τύπου Double, η οποία αντιστοιχεί στο υπολογισθέν ρίσκο.
5. `calcImageProb()`. Μέθοδος που υπολογίζει για κάθε εικόνα την πιθανότητα να εκτελεστεί μια ευπάθεια σε τουλάχιστον ένα asset της. Δέχεται σαν παράμετρο το Map assetsWithRisks που περιέχει τα assets που βρέθηκαν σε μια εικόνα αντιστοιχισμένα με το ρίσκο που υπολογίσθηκε για αυτά και επιστρέφει μια μεταβλητή τύπου Double, η οποία αντιστοιχεί στο υπολογισθέν ρίσκο.
6. `ImageRiskCalcAlgo()`. Αυτή η μέθοδος είναι ο κατασκευαστής της κλάσης. Αυτός είναι υπεύθυνος ώστε να εκτελεστεί η σωστή μέθοδος από τις παραπάνω ανάλογα με την επιλογή του διαχειριστή. Δέχεται σαν παράμετρο το Map assetsWithRisk που περιέχει τα assets που βρέθηκαν σε μια εικόνα αντιστοιχισμένα με το ρίσκο που υπολογίσθηκε για αυτά καθώς και το Map "assetsWithProb", το οποίο περιέχει όλα τα assets που βρέθηκαν σε μια εικόνα αντιστοιχισμένα με την πιθανότητα εκτέλεσης τουλάχιστον μίας ευπάθειας από αυτές που περιέχουν καθώς και μια ακέραια τιμή η οποία υποδηλώνει ποια μέθοδος θα χρησιμοποιηθεί. Στο τέλος επιστρέφει μια μεταβλητή τύπου Double, η οποία αντιστοιχεί στο υπολογισθέν ρίσκο.

OverallRisk class

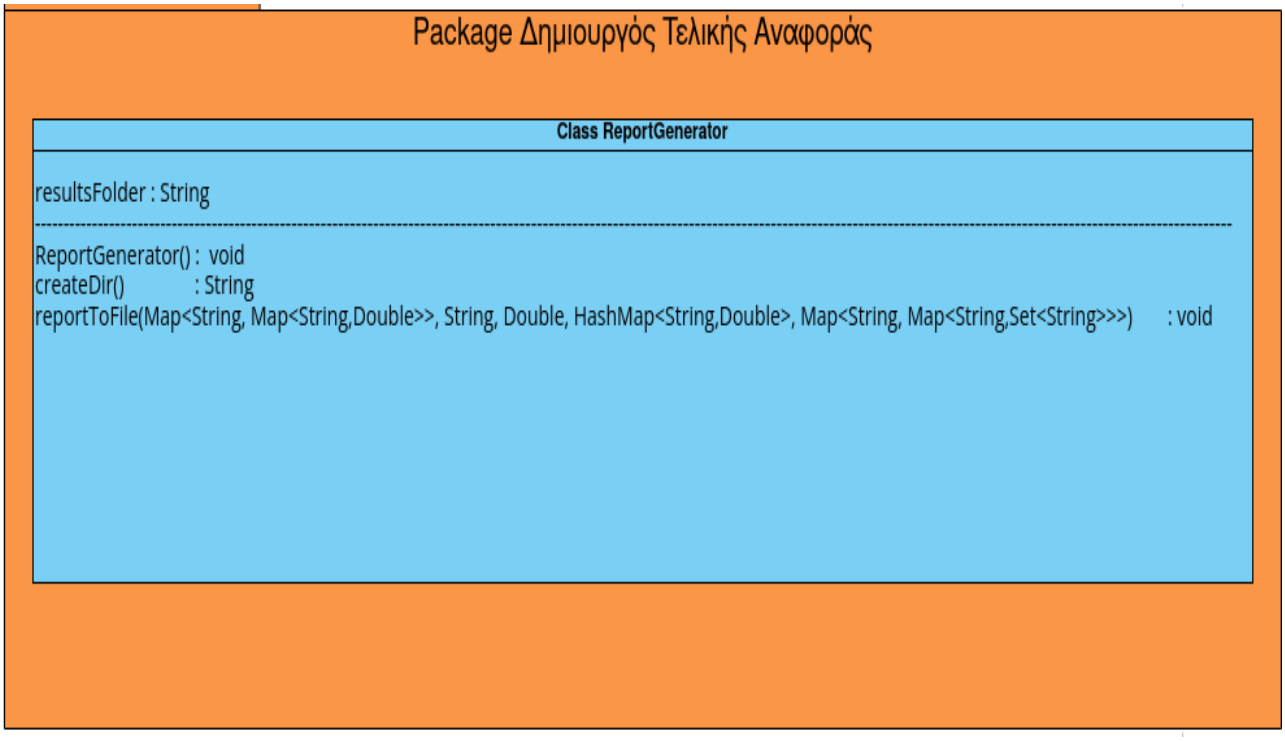
Η κλάση αυτή χρησιμοποιείται για τον υπολογισμό του ρίσκου σε ολόκληρο το σύστημα. Η δομή και η λειτουργικότητα της είναι παρόμοια με την προηγούμενη. Περιέχει μια ιδιωτική ιδιότητα με όνομα overrallScore τύπου Double, η οποία αναφέρεται στο συνολικό ρίσκο του συστήματος και συνολικά τις έξι παρακάτω μεθόδους:

1. `max()`. Μέθοδος που βρίσκει το μεγαλύτερο από τα υπολογισμένα ρίσκα για τις εικόνες και ορίζει αυτό ως το ρίσκο για το συνολικό σύστημα. Δέχεται σαν παράμετρο την λίστα risks που περιέχει τα ρίσκα που υπολογίστηκαν για τις εικόνες που περιέχει το σύστημα και εκχωρεί το υπολογισθέν ρίσκο στην ιδιότητα overallScore της κλάσης.
2. `avg()`. Μέθοδος η οποία υπολογίζει το ρίσκο για το συνολικό σύστημα, υπολογίζοντας τον απλό μέσο όρο των ρίσκων των εικόνων δοχείων. Δέχεται σαν παράμετρο την λίστα risks που περιέχει τα ρίσκα που υπολογίστηκαν για τις εικόνες και εκχωρεί το υπολογισθέν ρίσκο στην ιδιότητα overallScore της κλάσης.
3. `calcWeightAggAuto()`. Μέθοδος που υπολογίζει το ρίσκο για το συνολικό σύστημα, με βάση κάποια αυτόματα βάρη που έχουν τοποθετηθεί στις εικόνες δοχείων. Αυτά τα βάρη

υπολογίζονται με βάση την πιθανότητα να εκτελεστεί κάποια από τις ευπάθειες που περιέχουν οι εικόνες. Δέχεται σαν παράμετρο την λίστα risks και την λίστα imagesWithProb, η οποία περιέχει την πιθανότητα να εκτελεστεί κάποια από τις ευπάθειες κάθε εικόνας, ενώ εκχωρεί το υπολογισθέν ρίσκο στην ιδιότητα overrallScore της κλάσης.

4. `calcWeightAggMan()`. Μέθοδος, που υπολογίζει το ρίσκο για το συνολικό σύστημα, με βάση κάποια βάρη, τα οποία θέτει ο διαχειριστής του συστήματος για κάθε εικόνα. Δέχεται σαν παράμετρο την λίστα risks και την λίστα weights που περιέχει τα βάρη όλων των εικόνων δοχείων ενώ εκχωρεί το υπολογισθέν ρίσκο στην ιδιότητα overallScore της κλάσης.
5. `getOverallScore()`. Μέθοδος που επιστρέφει το υπολογισθέν ρίσκο για όλο το σύστημα.
6. `OverallRisk()`. Αυτή η μέθοδος είναι ο κατασκευαστής της κλάσης. Είναι υπεύθυνος ώστε να εκτελεστεί η σωστή μέθοδος από τις παραπάνω, ανάλογα με την επιλογή του διαχειριστή. Δέχεται σαν παράμετρο τις λίστες risks (Γεμίζει στην κλάση Orchestrator και περιέχει το ρίσκο που υπολογίσθηκε για κάθε εικόνα), weights και imagesWithProb (Η δομή αυτή υπολογίζεται από την κλάση ImageRiskCalcAlgo κατά την διάρκεια του υπολογισμού του ρίσκου για την εκάστοτε εικόνα) καθώς και μια ακέραια τιμή η οποία υποδηλώνει ποια μέθοδος θα χρησιμοποιηθεί. Στο τέλος, εκχωρεί το υπολογισθέν ρίσκο στην ιδιότητα overrallScore της κλάσης.

5.1.5 Δημιουργός Τελικής Αναφοράς (Final Report Generator)



Εικόνα 8: Το πακέτο Δημιουργός Τελικής Αναφοράς

ReportGenerator class

Η κλάση αυτή χρησιμοποιείται για την αποθήκευση των αποτελεσμάτων από την εκτέλεση του εργαλείου και την παραγωγή της τελικής αναφοράς. Για κάθε εκτέλεση του προγράμματος δημιουργεί ένα καινούριο (υπο-)φάκελο (μέθοδος `createDir()`) με όνομα την τρέχουσα ώρα του υπολογιστή. Αυτό συμβαίνει έτσι ώστε τα αποτελέσματα (ενδιάμεσα και τελικά) κάθε εκτέλεσης να αποθηκεύονται μοναδικά.

Περιλαμβάνει μια ιδιότητα, η οποία περιέχει το όνομα του υπο-φακέλου στον οποίο θα αποθηκευτούν τα αποτελέσματα από την εκτέλεση του εργαλείου:

1. public String resultsFolder

Η κλάση αυτή περιέχει τις εξής μεθόδους:

1. `ReportGenerator()`. Αυτή η μέθοδος αποτελεί τον κατασκευαστή της κλάσης.
2. `createDir()`. Μέθοδος που δημιουργεί τον νέο υποφάκελο στο μονοπάτι που θα αποθηκευτούν τα αποτελέσματα του εργαλείου (results/<ημερομηνία και ώρα εκτέλεσης> όπου <ημερομηνία και ώρα εκτέλεσης> είναι το όνομα του υπο-φακέλου και results είναι ο φάκελος αποθήκευσης όλων των αποτελεσμάτων κλήσεων του εργαλείου).
3. `reportToFile()`. Αυτή η μέθοδος χρησιμοποιείται για να αποθηκεύσει όλη την πληροφορία που παρήγαγε το εργαλείο αποτίμησης ρίσκου σε ένα αρχείο με όνομα results.json. Δέχεται σαν παράμετρο ένα Map που περιέχει όλα τα asset μιας εικόνας αντιστοιχισμένα με το ρίσκο που υπολογίσθηκε για αυτά (πρώτη παράμετρος τύπου Map<String, Map<String, Double>>), το όνομα του φακέλου που θα αποθηκευτεί το αρχείο (ίδιος φάκελος με τα προηγούμενα αρχεία - String), το ολικό ρίσκο που υπολογίσθηκε για το σύστημα(Double), ένα Map που περιέχει τις εικόνες αντιστοιχισμένες με το ρίσκο που εντοπίστηκε για αυτές (τέταρτη παράμετρος τύπου Map<String,Double>) καθώς και ένα Map που περιέχει τα cve ids αντιστοιχισμένα με το asset στο οποίο βρέθηκαν στην εκάστοτε εικόνα (τελευταία παράμετρος τύπου Map<String, Map<String, Set<String>>>).

5.2 Οδηγίες εκτέλεσης και μεταγλώττισης του κώδικα

5.2.1 Οδηγίες εγκατάστασης των εργαλείων ανίχνευσης τρωτοτήτων

Η εγκατάσταση του εργαλείου clair-scanner είναι σχετικά απλή και γίνεται με τα εξής βήματα:

1. Λήψη του εκτελέσιμου αρχείου[8].
2. Εγκατάσταση του εκτελέσιμου αρχείου, σε όποιο φάκελο επιθυμούμε. Πρέπει να έχουμε στο νου μας ότι θα χρειαστούμε το μονοπάτι συστήματος προς τον αρχείο αυτό στη

[8] https://github.com/arminc/clair-scanner/releases

συνέχεια για την εκτέλεση του προτεινόμενου εργαλείου, καθώς αυτό χρησιμοποιείται στην εντολή εκτέλεσης του εργαλείου clair-scanner.

Εναλλακτικά, μπορούμε να χρησιμοποιήσουμε το αποθετήριο του clair-scanner στο github[9]. Ειδικότερα, θα πρέπει να κλωνοποιήσουμε το αποθετήριο τοπικά, να εισέλθουμε στον αντίστοιχο φάκελο που δημιουργήθηκε και τέλος να κάνουμε build (κατασκευή) και cross-compile (ενσωμάτωση των εξαρτήσεων) με τις εντολές:

1) `make build`
2) `make cross`

Για εγκατάσταση του εργαλείου Grype στο λειτουργικό σύστημα linux προτείνεται η χρήση της παρακάτω εντολής:

```
curl -sSfL
https://raw.githubusercontent.com/anchore/grype/main/install.sh |
sh -s -- -b /usr/local/bin
```

Επίσης χρησιμοποιήθηκε η έκδοση της Java : SE-17 (επομένως θα πρέπει να είναι διαθέσιμη στο σύστημα φιλοξενίας).

Εντολές μεταγλώττισης και εκτέλεσης κώδικα εργαλείου:

Για να μεταγλωττίσουμε τον κώδικα αρκεί να εκτελέσουμε την παρακάτω εντολή:

```
sudo javac -cp <Μονοπάτι προς τον φάκελο του project>/libs/org.json-
20120521.jar
:<Μονοπάτι προς τον φάκελο του project>/libs/json-simple-1.1.1.jar
-d <Μονοπάτι προς τον φάκελο του project>/bin/
-s <Μονοπάτι προς τον φάκελο του project>/src/*.java
```

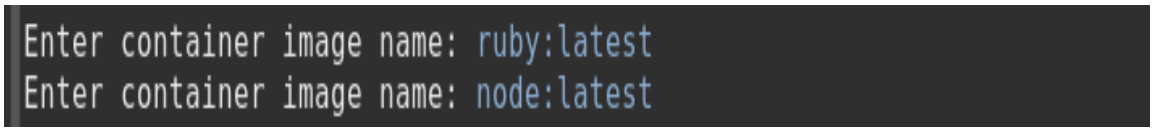
Χρησιμοποιούμε τις δύο βιβλιοθήκες (“.jar” αρχεία) που είναι απαραίτητες για την εκτέλεση του κώδικα και βρίσκονται στον υποφάκελο “libs” μέσα στον φάκελο του έργου. Με την παράμετρο “d” επιλέγουμε τον φάκελο στον οποίο θα αποθηκευτούν τα “.class” αρχεία που θα δημιουργηθούν κατά την μεταγλώττιση. Στην προκειμένη περίπτωση είναι ο υποφάκελος “bin” στο φάκελο του έργου. Με την παράμετρο “s” επιλέγουμε το φάκελο από τον οποίο θα αντληθούν τα αρχεία του πηγαίου κώδικα. Στην προκείμενη περίπτωση, τα πηγαία αρχεία βρίσκονται στον υποφάκελο “src” .

Αφού μεταγλωττίσουμε τον κώδικα, για να τον εκτελέσουμε αρκεί να πληκτρολογήσουμε την παρακάτω εντολή:

```
sudo java -cp <Μονοπάτι για τον φάκελο του project>/libs/org.json-
20120521.jar
:<Μονοπάτι για τον φάκελο του project>/libs/json-simple-1.1.1.jar
:<Μονοπάτι για τον φάκελο του project>/bin/ Orchestrator
```

[9] https://github.com/quay/clair

Όμοια με την εντολή μεταγλώττισης, στην εντολή εκτέλεσης πρέπει να εισάγουμε τις βιβλιοθήκες που χρησιμοποιήθηκαν (μονοπάτι και όνομα αρχείου) μαζί με το αρχείο .class που περιλαμβάνει την κύρια μέθοδο (μονοπάτι και όνομα αρχείου).
Στην περίπτωση που επιθυμούμε να εκτελέσουμε πλήρως την λειτουργικότητα του εργαλείου, θα πρέπει να δώσουμε το ακριβές όνομα των εικόνων δοχείων που έχουμε αποθηκευμένα στο σύστημά μας και θέλουμε να ελέγξουμε, όπως φαίνεται στην παρακάτω εικόνα.



Εικόνα 9: Εικόνες δοχείων προς εξέταση

Η διαδικασία εκτέλεσης (βήματα και επιλογές εισαγωγής) του εργαλείου παρουσιάζεται αναλυτικά στο επόμενο κεφάλαιο.

6. Επίδειξη & Αξιολόγηση Εργαλείου Αποτίμησης Ρίσκου

6.1 Επίδειξη Εργαλείου

Στόχος της επίδειξης είναι να παρουσιάσουμε λεπτομερώς όλες τις πτυχές της λειτουργικότητας του εργαλείου και να σχολιάσουμε τα αποτελέσματα του καθώς και τα πλεονεκτήματα που προσφέρει.
Για να ελέγξουμε την ορθή εκτέλεση του εργαλείου επιλέξαμε να κατεβάσουμε 2 εικόνες δοχείων από την πλατφόρμα Docker Hub και να εφαρμόσουμε πάνω σε αυτές το εργαλείο που δημιουργήσαμε. Αυτές οι δύο εικόνες είναι οι εξής :

- node:latest
- ruby:latest

Οι λόγοι για τους οποίους επιλέχθηκαν οι συγκεκριμένες εικόνες δοχείων είναι πολλοί. Καταρχάς, είναι αρκετά πολύπλοκες και περιέχουν ένα μεγάλο αριθμό από διαφορετικά assets. Αυτό σημαίνει ότι υπάρχουν περισσότερες πιθανότητες να βρεθούν ευπάθειες στις συγκεκριμένες εικόνες. Ταυτόχρονα, είναι εικόνες που μπορούν να χρησιμοποιηθούν σε πολλά και διαφορετικά συστήματα και κατά συνέπεια αποτελούν ένα καλό παράδειγμα για την χρησιμότητα και την αποτελεσματικότητα του εργαλείου. Επιπροσθέτως, μας επιτρέπουν να δείξουμε τόσο την αύξηση της ακρίβειας στον υπολογισμό του ρίσκου αλλά και την αύξηση στον αριθμό των εντοπισμένων ευπαθειών που επιδεικνύει το προτεινόμενο εργαλείο σε σχέση με τα εργαλεία ανίχνευσης (που αυτό χρησιμοποιεί) μεμονωμένα.

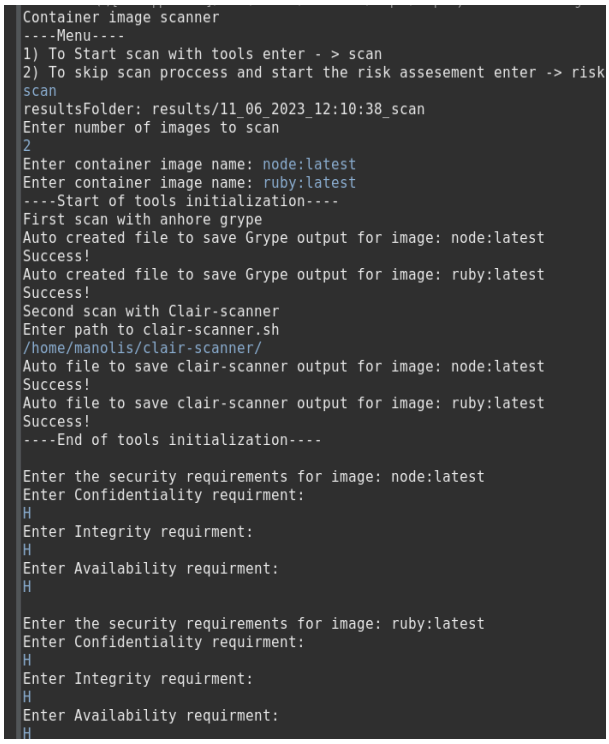
6.1.1 Σενάριο Πλήρους Εκτέλεσης του Εργαλείου

Στο πρώτο σενάριο επιθυμούμε να εκτελέσουμε πλήρως όλη την λειτουργικότητα του εργαλείου οπότε χρησιμοποιούμε την επιλογή “scan” (δείτε Εικόνα 10) στην πρώτη ερώτηση που μας κάνει το εργαλείο.
Έπειτα, παρέχουμε τον αριθμό 2 διότι, όπως προαναφέραμε, θα εξετάσουμε 2 εικόνες δοχείων. Κατόπιν, εισάγουμε τα ονόματα των εικόνων, όπως είναι αποθηκευμένα στο σύστημα. Παρατηρούμε ότι το εργαλείο δημιουργεί αυτόματα τα απαραίτητα αρχεία για να αποθηκεύσει τα αποτελέσματα του εργαλείου grype (δείτε Εικόνα 10), ενώ στην συνέχεια μας ζητά να εισάγουμε τα απαραίτητα στοιχεία για το clair-scanner, δηλαδή το μονοπάτι για την θέση που είναι αποθηκευμένο το αρχείο clair-scanner.sh (δείτε Εικόνα 10).

Τα αρχεία με τα αποτελέσματα από την εκτέλεση και των δύο εργαλείων αποθηκεύονται μέσα στον φάκελο results, σε έναν υποφάκελο με όνομα το “<timestamp της εκτέλεσης του εργαλείου>_scan” και έχουν την εξής μορφή:
<όνομα_εικόνας>_res_from_<όνομα_εργαλείου>.json, όπου το όνομα του εργαλείου στην πρώτη περίπτωση θα είναι “grype” και στην δεύτερη “clairScanner” .

Έπειτα, μας ζητείται να θέσουμε τιμές στις τρεις απαιτήσεις ασφαλείας για κάθε εικόνα (δείτε Εικόνα 10). Τέλος, επιλέγουμε τις μεθόδους που θα χρησιμοποιηθούν στον αλγόριθμο σύνθεσης ρίσκου (δείτε Εικόνα 11).
Κατόπιν ξεκινάει η εκτέλεση του προγράμματος.
Μόλις ολοκληρωθεί η εκτέλεση του εργαλείου, αυτό παρουσιάζει τα αποτελέσματα στην εξής μορφή:

- Πρώτα εμφανίζεται το ρίσκο για μια εικόνα από το πρώτο εργαλείο ανίχνευσης, μετά από το δεύτερο και στη συνέχεια από το προτεινόμενο εργαλείο (δείτε Εικόνα 12).
- Μετά εμφανίζονται οι ευπάθειες που εντόπισε κάθε εργαλείο ξεχωριστά για την εικόνα και μετά εμφανίζονται οι ευπάθειες που εντόπισε το προτεινόμενο εργαλείο (δείτε Εικόνα 12).
- Η παραπάνω διαδικασία επαναλαμβάνεται για όλες τις εικόνες που ζητήθηκαν να αναλυθούν.
- Τέλος εμφανίζει το συνολικό ρίσκο που υπολογίσθηκε με χρήση του πρώτου εργαλείου μόνο, μετά μόνο του δεύτερου εργαλείου και στο τέλος του προτεινόμενου εργαλείου (δείτε Εικόνα 12).



Εικόνα 10: Εκτέλεση του εργαλείου 1/6

```
Choose aggregate function for image risk calculation
For max function choose : 1
For average function choose : 2
For weight_aggr_auto function choose : 3
For weight_aggr_manual function choose : 4
2
Choose aggregate function for system risk calculation
For max function choose : 1
For average function choose : 2
For weight_aggr_auto function choose : 3
For weight_aggr_manual function choose : 4
For one image (no aggregate) chooce 5
 2
```

Εικόνα 11: Εκτέλεση του εργαλείου 2/6

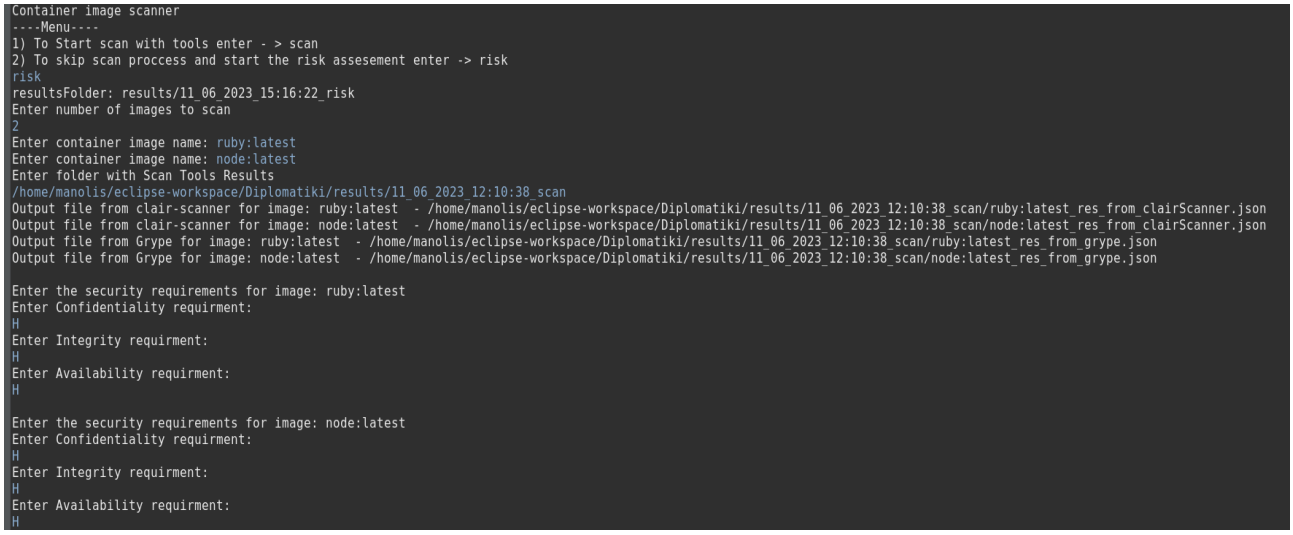
```
Results:
----Start risk assesement for image: node:latest----
Risk Assesment for image: 1 from : 2
files names:  clair: results/11_06_2023_12:10:38_scan/node:latest_res_from_clairScanner.json grype: results/11_06_2023_12:10:38_scan/node:latest_res_from_
--------------- START THE PROCCESS ON CLAIR ---------------------
--------------- END THE PROCCESS ON CLAIR ---------------
--------------- START THE PROCCESS ON GRYPE ---------------------
--------------- END THE PROCCESS ON GRYPE ---------------------
IMAGE: node:latest IS CALCULATED WITH 109.29 RISK , USING ONLY GRYPE ANALYZER
Cve's found from grype analyzer : 41 for image node:latest
IMAGE: node:latest IS CALCULATED WITH 86.16 RISK , USING ONLY CLAIR SCANNER
Cve's found from clair scanner : 54 for image node:latest
IMAGE: node:latest IS CALCULATED WITH 96.15 RISK , USING BOTH TOOLS
----End risk assesement for image: node:latest----
Cve's found from both tools : 54 for image node:latest
----Start risk assesement for image: ruby:latest----
Risk Assesment for image: 2 from : 2
files names:  clair: results/11_06_2023_12:10:38_scan/ruby:latest_res_from_clairScanner.json grype: results/11_06_2023_12:10:38_scan/ruby:latest_res_from_
--------------- START THE PROCCESS ON CLAIR ---------------------
--------------- END THE PROCCESS ON CLAIR ---------------
--------------- START THE PROCCESS ON GRYPE ---------------------
--------------- END THE PROCCESS ON GRYPE ---------------------
IMAGE: ruby:latest IS CALCULATED WITH 96.82 RISK , USING ONLY GRYPE ANALYZER
Cve's found from grype analyzer : 49 for image ruby:latest
IMAGE: ruby:latest IS CALCULATED WITH 0.0 RISK , USING ONLY CLAIR SCANNER
Cve's found from clair scanner : 54 for image ruby:latest
IMAGE: ruby:latest IS CALCULATED WITH 87.99 RISK , USING BOTH TOOLS
----End risk assesement for image: ruby:latest----
Cve's found from both tools : 62 for image ruby:latest
OverallRisk for all images: 92.07
OverallRisk for all images using grype analyzer: 103.055
OverallRisk for all images using clair scanner: 43.08
```

Εικόνα 12: Εκτέλεση του εργαλείου 3/6

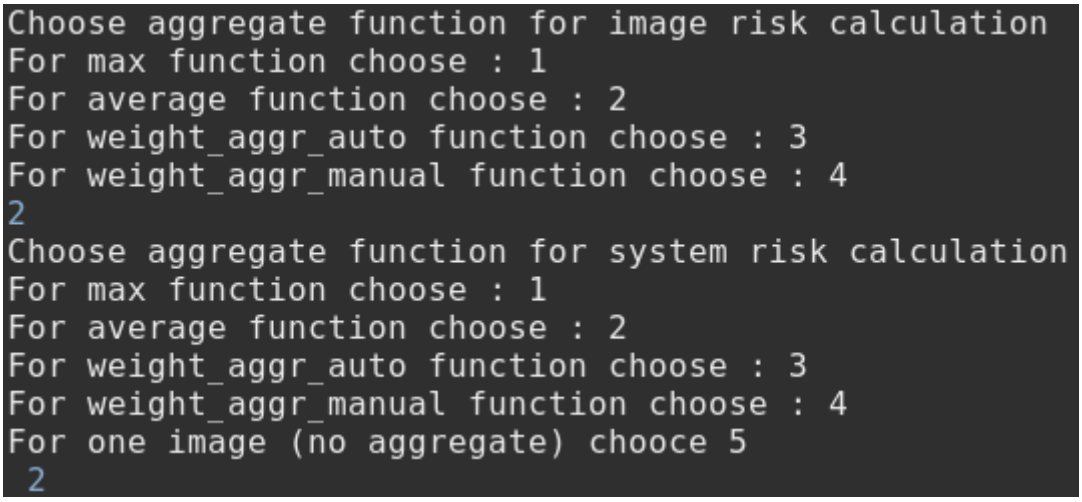
6.1.2 Σενάριο Εκτέλεσης Μόνο Αποτίμησης Ρίσκου

Στο δεύτερο σενάριο έχουμε ήδη τα αποτελέσματα αποθηκευμένα σε αρχεία (από προηγούμενη εκτέλεση του προτεινόμενου εργαλείου) και θα τα χρησιμοποιήσουμε για να υπολογίσουμε το ρίσκο απευθείας. Η διαδικασία σε αυτό το σενάριο έχει κάποιες ομοιότητες αλλά και αρκετές διαφορές σε σχέση με αυτήν που ακολουθήθηκε στο πρώτο σενάριο. Αυτό αναλύεται παρακάτω.

Σε αυτό το σενάριο εκτέλεσης (ενότητα 6.1.2) του κώδικα (περίπτωση “risk”) τα αποθηκευμένα αρχεία για τα αποτελέσματα πρέπει να βρίσκονται μέσα σε έναν υποφάκελο του φακέλου “results” και να έχουν την μορφή “<image name>_res_from_<Tool name>.json”.



Εικόνα 13: Εκτέλεση του εργαλείου 4/6



Εικόνα 14: Εκτέλεση του εργαλείου 5/6

```
Results:
----Start risk assesement for image: ruby:latest----
Risk Assesment for image: 1 from : 2
files names:  clair: /home/manolis/eclipse-workspace/Diplomatiki/results/11_06_2023_12:10:38_scan/ruby:latest_res_from_clairScanner.json grype: /home/manolis/eclipse-workspace/Diplomati
--------------- START THE PROCCESS ON CLAIR --------------------
--------------- END THE PROCCESS ON CLAIR ---------------
--------------- START THE PROCCESS ON GRYPE --------------------
--------------- END THE PROCCESS ON GRYPE --------------------
IMAGE: ruby:latest IS CALCULATED WITH 96.82 RISK , USING ONLY GRYPE ANALYZER
Cve's found from grype analyzer : 49 for image ruby:latest
IMAGE: ruby:latest IS CALCULATED WITH 0.0 RISK , USING ONLY CLAIR SCANNER
Cve's found from clair scanner : 54 for image ruby:latest
IMAGE: ruby:latest IS CALCULATED WITH 87.64 RISK , USING BOTH TOOLS
----End risk assesement for image: ruby:latest----
Cve's found from both tools : 62 for image ruby:latest
----Start risk assesement for image: node:latest----
Risk Assesment for image: 2 from : 2
files names:  clair: /home/manolis/eclipse-workspace/Diplomatiki/results/11_06_2023_12:10:38_scan/node:latest_res_from_clairScanner.json grype: /home/manolis/eclipse-workspace/Diplomati
--------------- START THE PROCCESS ON CLAIR --------------------
--------------- END THE PROCCESS ON CLAIR ---------------
--------------- START THE PROCCESS ON GRYPE --------------------
--------------- END THE PROCCESS ON GRYPE --------------------
IMAGE: node:latest IS CALCULATED WITH 109.29 RISK , USING ONLY GRYPE ANALYZER
Cve's found from grype analyzer : 41 for image node:latest
IMAGE: node:latest IS CALCULATED WITH 86.16 RISK , USING ONLY CLAIR SCANNER
Cve's found from clair scanner : 54 for image node:latest
IMAGE: node:latest IS CALCULATED WITH 87.99 RISK , USING BOTH TOOLS
----End risk assesement for image: node:latest----
Cve's found from both tools : 62 for image node:latest
OverallRisk for all images: 87.815
OverallRisk for all images using grype analyzer: 103.055
OverallRisk for all images using clair scanner: 43.08
```

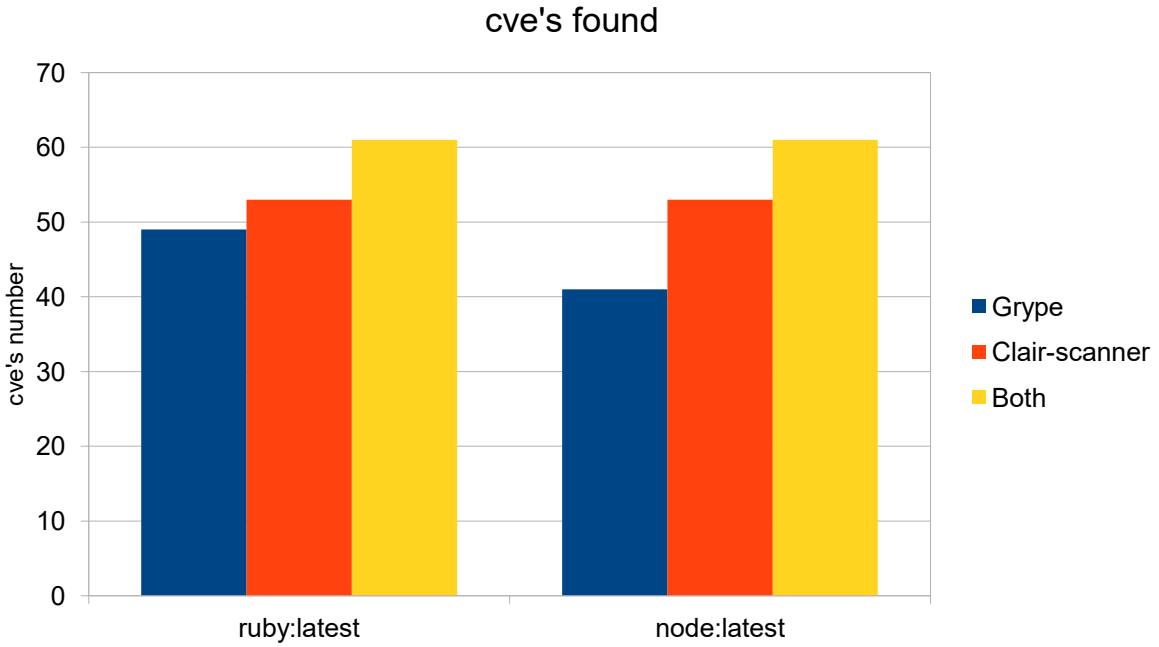
Εικόνα 15: Εκτέλεση του εργαλείου 6/6

Αρχικά, επιλέγουμε “risk” στην πρώτη ερώτηση που θέτει το εργαλείο (δείτε Εικόνα 13). Έπειτα, βάζουμε τον αριθμό 2 διότι, όπως προαναφέραμε, θα εξετάσουμε 2 εικόνες δοχείων.

Κατόπιν, εισάγουμε τα ονόματα των εικόνων, όπως είναι αποθηκευμένα στο σύστημα.
Στην συνέχεια, θα πρέπει ο διαχειριστής να δώσει το όνομα του φακέλου στον οποίο είναι αποθηκευμένα τα αρχεία με τα αποτελέσματα από την εκτέλεση των δύο εργαλείων για τις συγκεκριμένες εικόνες (δείτε Εικόνα 13). Τα αρχεία αυτά θα πρέπει να έχουν συγκεκριμένο όνομα, το οποίο αποτελείται από το όνομα της εικόνας + το όνομα του εργαλείου + την λέξη “file” και με κατάληξη “.json”.
Κατόπιν, μας ζητείται να θέσουμε τιμές στις τρεις απαιτήσεις ασφαλείας (δείτε Εικόνα 13).
Τέλος επιλέγουμε τις μεθόδους που θα χρησιμοποιηθούν στον αλγόριθμο σύνθεσης ρίσκου (δείτε Εικόνα 14).
Κατόπιν ξεκινάει η εκτέλεση του προγράμματος.
Όταν ολοκληρωθεί η εκτέλεση του εργαλείου, παρουσιάζονται τα αποτελέσματα στην ίδια μορφή με αυτή του πρώτου σεναρίου (δείτε Εικόνα 15).

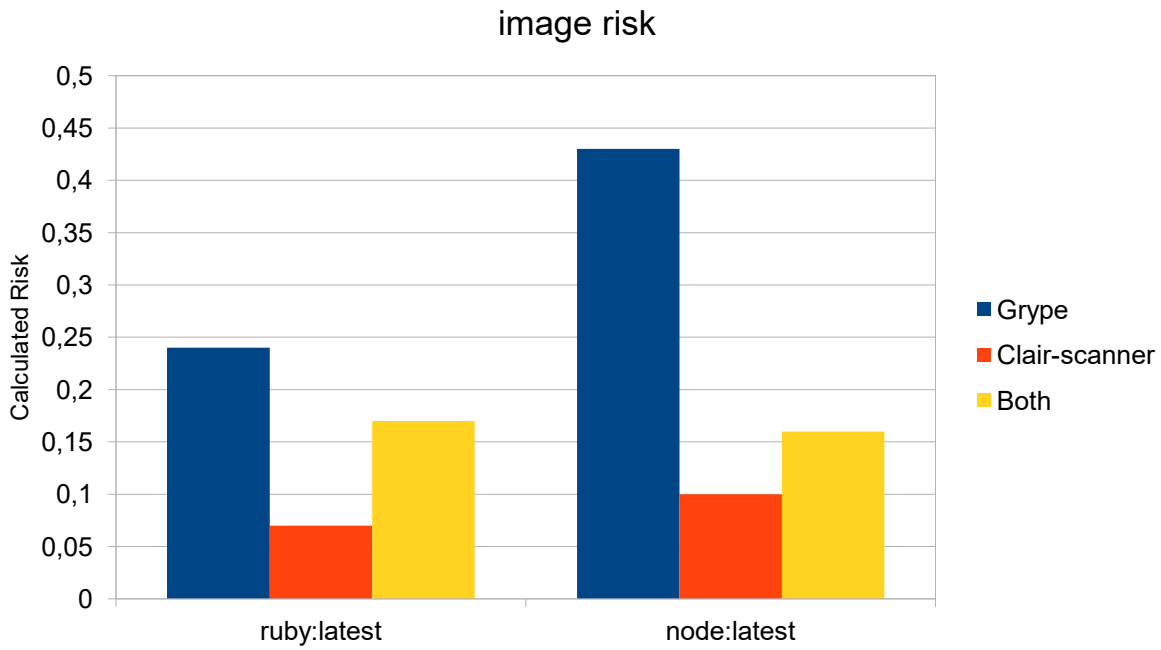
6.2 Αξιολόγηση Εργαλείου

Η αποτίμηση/αξιολόγηση του εργαλείου πραγματοποιήθηκε με βάση τις 2 προαναφερόμενες εικόνες που εκφορτώθηκαν από το Docker Hub. Βασίζεται στον υπολογισμό του ρίσκου και στον συνολικό αριθμό των ευπαθειών που εντοπίστηκαν τόσο για το προτεινόμενο εργαλείο, όταν αυτό εκμεταλλεύεται και τα 2 εργαλεία ανίχνευσης, όσο και για 2 τις περιπτώσεις όπου το εργαλείο μας εκμεταλλεύεται μόνο ένα από τα 2 εργαλεία ανίχνευσης αντίστοιχα.

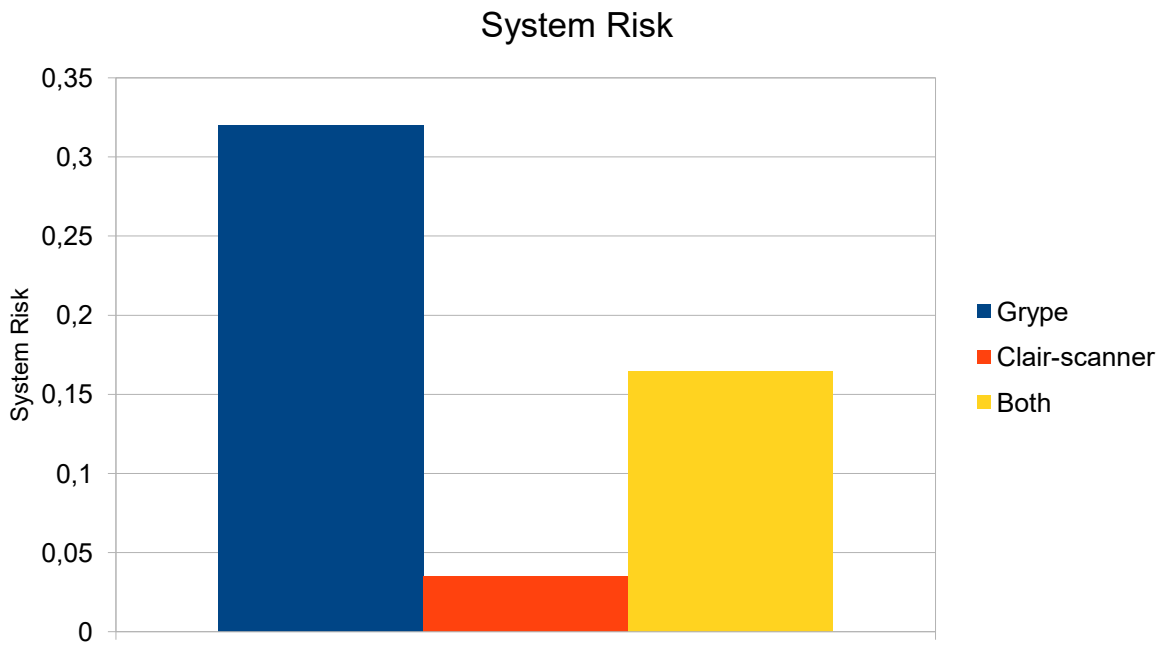


Εικόνα 16: Σύγκριση στον αριθμό των ευπαθειών που βρέθηκαν

Όπως αποτυπώνεται στο διάγραμμα στην Εικόνα 16 ο συνολικός αριθμός των ευπαθειών είναι μεγαλύτερος όταν χρησιμοποιηθούν και τα δύο εργαλεία ανίχνευσης, πράγμα που αυξάνει την αποτελεσματικότητα του εργαλείου μας σε σχέση με την χρήση μόνο ενός από τα δύο εργαλεία ανίχνευσης (Clair-scanner και Grype Analyzer).



Εικόνα 17: Σύγκριση στο υπολογισθέν ρίσκο για μια εικόνα



Εικόνα 18: Σύγκριση στο υπολογισθέν ρίσκο για το συνολικό σύστημα

Στα δύο παραπάνω διαγράμματα (δείτε Εικόνα 17 και Εικόνα 18) βλέπουμε ότι παρά το γεγονός ότι στο δικό μας εργαλείο ανιχνεύσαμε περισσότερες ευπάθειες, το συνολικό ρίσκο που υπολογίσαμε, τόσο για μια εικόνα όσο και για το συνολικό σύστημα, δεν είναι πάντα μεγαλύτερο από αυτό που υπολογίζεται όταν χρησιμοποιείται μόνο ένα από τα δύο εργαλεία ανίχνευσης. Ο κύριος λόγος που συμβαίνει αυτό είναι εξαιτίας της χρήσης των συναρτήσεων σύνθεσης τόσο για τον υπολογισμό του ρίσκου για μια εικόνα όσο και για όλο το σύστημα. Επίσης, μπορεί να οφείλεται στο γεγονός ότι η ύπαρξη περισσότερων ευπαθειών δεν σημαίνει αναγκαστικά και μεγαλύτερο κίνδυνο, διότι μπορεί να υπάρχουν πολλές αναγνωρισμένες και πλήρως ή μερικώς αντιμετωπισμένες ευπάθειες, οι οποίες δεν χρήζουν σημαντικής ανησυχίας.

7. Συμπεράσματα & Μελλοντική Εργασία

7.1 Συμπεράσματα και εμπειρίες

Μέσα από την εκπόνηση της παρούσας διπλωματικής εργασίας κατανοήσαμε καλύτερα και σε βάθος τις τεχνολογίες πίσω από το υπολογιστικό νέφος. Ιδιαίτερα, τον ρόλο και την λειτουργία των εικόνων δοχείων Docker στο υπολογιστικό νέφος και στην ανάπτυξη εφαρμογών νέφους. Μάθαμε διάφορες λεπτομέρειες για τα εργαλεία που χρησιμοποιούνται για να ανιχνεύουν ευπάθειες σε αυτές τις εικόνες, όπως πως λειτουργούν, τι αποτελέσματα βγάζουν και πως μπορούμε να τα χρησιμοποιήσουμε για να κάνουμε μια εκτίμηση του ρίσκου που έχει το κάθε asset μια εικόνας, μια εικόνα και συνολικά μια εφαρμογή/σύστημα. Επιπροσθέτως, αποκτήσαμε γνώσεις τόσο για τον τρόπο με τον οποίο μπορούμε να συλλέγουμε δεδομένα από διαδικτυακές βάσεις δεδομένων όσο και το πως να τα επεξεργαζόμαστε. Παράλληλα, επεκτείναμε τις γνώσεις μας πάνω στην γλώσσα προγραμματισμού Java καθώς όλο το εργαλείο δημιουργήθηκε με την χρήση αυτής της γλώσσας, μαθαίνοντας ειδικότερα πως να διαχειριζόμαστε αρχεία JSON αλλά και πως να εκτελούμε εντολές τερματικού μέσα από κώδικα Java.

Επιπροσθέτως, μπορούμε να συμπεράνουμε ότι με την χρήση 2 (ή και περισσότερων) εργαλείων ανίχνευσης συνδυαστικά αυξάνεται ο αριθμός των ευπαθειών που εντοπίζονται, διότι τα δύο εργαλεία δεν ανιχνεύουν ακριβώς τις ίδιες ευπάθειες, με αποτέλεσμα ο συνδυασμός τους να αναγνωρίζει συνδυαστικά όλες τις ευπάθειες που αυτά εντόπισαν (συνολικά). Ταυτόχρονα, αυξάνουμε και την ακρίβεια των αποτελεσμάτων της αποτίμησης ρίσκου διότι γνωρίζουμε περισσότερες ευπάθειες που ενδέχεται να υπάρχουν σε μια εικόνα δοχείου και στα assets αυτής.

Ο συνδυασμός των εργαλείων προϋποθέτει ότι είτε τα αποτελέσματα που αυτά παράγουν θα έχουν μια κοινή μορφή (format) είτε θα μπορούν, μετά από κάποια επεξεργασία, να έρθουν σε μια κοινή μορφή. Τα εργαλεία που χρησιμοποιήθηκαν εδώ προσφέρουν την δυνατότητα να αποθηκεύσουν τα αποτελέσματα που παράγουν στην ίδια μορφοποίηση (JSON). Επομένως, ο συνδυασμός τους δεν ήταν δύσκολος διότι η χρήση JSON παρείχε τα εχέγγυα για μια εύκολη επεξεργασία των αποτελεσμάτων προς μια κοινή μορφή.

7.2 Μελλοντική εργασία

Το εργαλείο που εξετάζεται στην παρούσα εργασία χρησιμοποιεί, όπως αναφέρθηκε και παραπάνω, στον αλγόριθμο υπολογισμού του ρίσκου το ενιαίο σύστημα βαθμολόγησης (CVSS) στην δεύτερη έκδοση του (v2). Επομένως, μια ενδεχόμενη βελτίωση που θα μπορούσε να γίνει, όταν οι συνθήκες το επιτρέψουν, είναι η μεταφορά του εργαλείου μας στην τρίτη έκδοση του CVSS που θα οδηγούσε σε καλύτερη ακρίβεια στην αποτίμηση του ρίσκου. Επιπροσθέτως, μια ακόμα δυνατότητα βελτίωσης του συστήματος μπορεί να γίνει μέσω της ένταξης σε αυτό περισσότερων παραγόντων (Exploitability, Report Confidence), όπως αυτοί περιγράφονται στο άρθρο [1].
Επίσης, μια ακόμα βελτίωση που επιδέχεται το αναπτυχθέν εργαλείο είναι να μπορεί να εκτελεστεί και σε άλλα περιβάλλοντα, όπως το λειτουργικό σύστημα Windows.
Μια άλλη βελτίωση θα ήταν και η προσθήκη περισσότερων εργαλείων ανίχνευσης ευπαθειών με σκοπό να γίνει το τελικό αποτέλεσμα αποτίμησης ακόμη πιο ακριβές. Τέτοια εργαλεία θα μπορούσαν να είναι το Dagda και το Harbor.
Επιπροσθέτως, μια ουσιαστική βελτίωση του εργαλείου θα ήταν να δημιουργηθεί μια εικόνα δοχείου, η οποία να παρέχει την δυνατότητα εκτέλεσης του εργαλείου. Με αυτόν τον τρόπο, θα μπορούσε να εκτελεστεί το εργαλείο σε οποιοδήποτε λειτουργικό σύστημα και όχι μόνο σε λειτουργικά συστήματα βασισμένα σε Linux.
Μια άλλη μικρή επέκταση θα ήταν το διάβασμα όλων των τιμών διαμόρφωσης του εργαλείου από ένα αρχείο διαμόρφωσης (που μπορεί να βρίσκεται σε προκαθορισμένη τοποθεσία), όπως και η ενσωμάτωση των οδηγιών χρήσης στην εντολή εκτέλεσης του εργαλείου.
Τέλος, θα μπορούσαμε να επεκτείνουμε την χρήση του εργαλείου μας και να το χρησιμοποιήσουμε σε ένα αποθετήριο εικόνων (πχ. το Docker Hub) προκειμένου να μπορούμε να κάνουμε μια αποτίμηση ρίσκου για όλες τις εικόνες που υπάρχουν σε αυτό.

8. Βιβλιογραφία

1. M. U. Aksu, M. Hadi Dilek, E. İslam Tatlı, K. Bicakci, H. İbrahim Dirik, M. Umut Demirezen, and T. Aykır et al., "A quantitative CVSS-based cyber security risk assessment methodology for IT systems," *2017 International Carnahan Conference on Security Technology (ICCST)*, Madrid, Spain, 2017, pp. 1-8, doi: 10.1109/CCST.2017.8167819.
2. A. Mailewa Dissanaya, S. Mengel, L. Gittner, and H. Khan, "Dynamic and Portable Vulnerability Assessment Testbed with Linux Containers to Ensure the Security of MongoDB in Singularity LXCs," *Supercomputing 2018 (SC18)*, Dallas, Texas, USA, 2018.
3. M. Nizam Mohd Mydin, B. Ikhwan Ismail, R. Kandan, H. Ahmad and F. Khalid. An Operational View into Docker Registry with Scalability, Access Control and Image Assessment. Advanced Computing Lab, MIMOS Berhad, Kuala Lumpur, Malaysia, 2021.
4. Rui Shu, Xiaohui Gu, and William Enck. 2017. A Study of Security Vulnerabilities on Docker Hub. In *Proceedings of the Seventh ACM on Conference on Data and Application Security and Privacy (CODASPY '17)*. Association for Computing Machinery, New York, NY, USA, 269–280. https://doi.org/10.1145/3029806.3029832

9. Παράρτημα

Clair scanner

Για να εκτελεστεί το συγκεκριμένο εργαλείο είναι απαραίτητη η επικοινωνία με ένα διακομιστή για την πρόσβαση στη βάση δεδομένων. Ωστόσο επειδή αυτό δεν είναι πάντοτε εφικτό, υπάρχει η δυνατότητα να εκτελεστεί το πρόγραμμα χρησιμοποιώντας μια ειδικά διαμορφωμένη βάση δεδομένων η οποία ανανεώνεται καθημερινά και επικοινωνεί με το clair-scanner. Η βάση αυτή έχει δημιουργηθεί με την χρήση ενός Travis Scheduled job και είναι προσβάσιμη μέσα από το docker hub .

Για την εκτέλεση του εργαλείου χρειάζεται να γνωρίζουμε το path για το αρχείο clair-scanner.sh και μπορούμε να το εκτελέσουμε μαζί με ένα wishlist αρχείο αν υπάρχει για το αρχείο που θέλουμε να σκανάρουμε ή και χωρίς . Οι επιλογές που έχουμε είναι οι εξής:

6. -w, --whitelist="" -> Path to the whitelist file εάν θέλουμε να χρησιμοποιηθεί μαζί και κάποιο whitelist .
7. -t, --threshold -> CVE severity threshold. Μπορεί να πάρει τις εξής τιμές : Defcon1, Critical, High, Medium, Low, Negligible, Unknown.
8. -c, --clair="" -> Clair URL . Για παράδειγμα : “http://127.0.0.1:6060” . Το “127.0.0.1” δείχνει ότι η βάση τρέχει τοπικά , ενώ το port number (6060) είναι αυτό που ορίσαμε παραπάνω (στην δεύτερη εντολή για την βάση δεδομένων)
9. --ip="" -> IP address where clair-scanner is running on . Για παράδειγμα “localhost” αν τρέχει τοπικά.
10. -l, --log="" -> Log to a file
11. --all, --reportAll = true . Εμφανίζει όλες τις ευπάθειες που εντόπισε ακόμη και αν αυτές είναι εγκεκριμένες από κάποιο whitelist.
12. -r, --report="" -> Report output file, as JSON
13. --exit-when-no-features = false -> Exit with status code 5 when no features are found for a particular image.

Η εντολή για να εκτελέσουμε το εργαλείο είναι η εξής:
Path_to_clair-scanner.sh/./clair-scanner [OPTIONS] IMAGE_NAME

Το εργαλείο αυτό έχει την δυνατότητα να αποθηκεύσει τα αποτελέσματα που παρήγαγε σε ένα αρχείο στην μορφή json χρησιμοποιώντας την επιλογή -r και το όνομα του αρχείου .

Τα αποτελέσματα του εργαλείου έχουν την παρακάτω μορφή :
{
“featurename” -> αναφέρεται στο software (asset / κομμάτι) της εικόνας δοχείου στο οποίο βρέθηκε η ευπάθεια .
Featureversion” -> αναφέρεται στην έκδοση (version) του παραπάνω λογισμικού.
“vulnerability” -> περιέχει το cveId της ευπάθειας όπως αυτό αντλήθηκε από την βάση δεδομένων .
“namespace”
“description” -> περιέχει μια μικρή περιγραφή για την ευπάθεια .
“link” -> περιέχει έναν υπερσύνδεσμο όπου μπορούμε να βρούμε ακόμα περισσότερες πληροφορίες για την ευπάθεια .
“severity” -> αντιπροσωπεύει την ομώνυμη παράμετρο του CVSS Scoring System.
“fixedby” -> περιέχει πληροφορίες για κάποιο fix εάν αυτό υπάρχει για την συγκεκριμένη ευπάθεια.
}

grype analyzer

Μερικά από τα χαρακτηριστικά που κάνουν το grype να ξεχωρίζει είναι τα εξής:

4. Μπορεί να ανιχνεύσει ευπάθειες για μια πληθώρα πακέτων λειτουργικών συστημάτων όπως :
    1) Alpine
    2) Amazon Linux
    3) BusyBox
    4) CentOS
    5) Debian
    6) Distroless
    7) Oracle linux
    8) Red Hat (RHEL)
    9) Ubuntu
5. Μπορεί να εντοπίσει ευπάθειες που αναφέρονται σε συγκεκριμένα πακέτα γλωσσών όπως:
    - Ruby
    - Java (JAR, WAR, EAR, JPI, HPI)
    - JavaScript (NPM, Yarn)
    - Python (Egg, Wheel, Poetry, requirements.txt/setup.py files)
    - Dotnet (deps.json)
    - Golang (go.mod)
    - PHP (Composer)
    - Rust (Cargo)

6. Υποστηρίζει Docker και OCI μορφοποίηση.

Επιπλέον το εργαλείο μας δίνει την δυνατότητα να εξετάσουμε για πιθανές ευπάθειες τα κομμάτια λογισμικού σε όλα τα επίπεδα μια εικόνας δοχείου και όχι μόνο σε εκείνα που εμφανίζονται σε μεγαλύτερο ποσοστό . Για να το επιτύχουμε αυτό αρκεί να εκτελέσουμε την παρακάτω εντολή:
grype <image_name> --scope all-layers

Επιπροσθέτως μας προσφέρει και την δυνατότητα να εξετάσουμε και δοχεία που εκτελούνται ήδη (running containers) εκτελώντας την παρακάτω εντολή:
docker run --rm \ --volume /var/run/docker.sock:/var/run/docker.sock \
--name Grype anchore/grype:latest \
$(ImageName):$(ImageTag)

Ταυτόχρονα παρέχει την δυνατότητα να εξετάσουμε για γνωστές ευπάθειες τόσο singularity images όσο και directories με τις εντολές grype path/to/image.sif και grype dir:path/to/dir αντίστοιχα .

Μια άλλη δυνατότητα που παρέχεται από το εργαλείο grype είναι η δυνατότητα που έχει να “αγνοεί” συγκεκριμένες ταυτοποιήσεις ευπαθειών , οι οποίες έχουν οριστεί από τον χρήστη του εργαλείου . Αυτό μπορεί να πραγματοποιηθεί τροποποιώντας το configuration file του εργαλείου , χρησιμοποιώντας το tag “ignore” και στην συνέχεια γράφοντας τους κανόνες στους οποίους θα βασίζεται . Οι κανόνες αυτοί μπορεί να βασίζονται σε οποιαδήποτε πληροφορία που περιέχει η ευπάθεια όπως για παράδειγμα το cve_id , το package name κλπ.

Η δομή των αποτελεσμάτων είναι η εξής:
{
“vulnerability” : {
},
“relatedVulnerabilities” : {
},
“MatchDetailes” : {
},
“artifact” :
{
},
}

Στην παραπάνω δομή κάθε {} αποτελεί ένα jsonObject .Στο αναγνωριστικό “vulnerability” περιλαμβάνονται διάφορες πληροφορίες για την εκάστοτε ευπάθεια όπως το cve_id , severity , CVSS Score , fix information και άλλα. Στο αναγνωριστικό “relatedVulnerabilities” περιλαμβάνονται πληροφορίες για σχετικές ευπάθειες με αυτή που βρέθηκε απο το εργαλείο. Στο αναγνωριστικό “MatchDetailes” περιλαμβάνονται πληροφορίες που αφορούν την διαδικασία με την οποία έγινε τελικά η αναγνώριση της ευπάθειας , όπως για παράδειγμα τα κλειδιά αναζήτησης,

ή λεπτομερείς πληροφορίες για το πακέτο στο οποίο περιλαμβάνεται η ευπάθεια. Στο αναγνωριστικό "artifact" περιλαμβάνει περισσότερες και πιο λεπτομερείς αναφορές για το πακέτο στο οποίο βρέθηκε η ευπάθεια καθώς και την "θέση" του στην εικόνα.

Για να αντλήσει τις πληροφορίες για τις ευπάθειες το εργαλείο grype χρησιμοποιεί μια βάση δεδομένων η οποία είναι αποθηκευμένη τοπικά στο σύστημα αρχείων και η οποία είναι κατασκευασμένη από το εργαλείο "τραβώντας" δεδομένα από διάφορες δημόσιες πηγές όπως:
1) Alpine Linux SecDB: https://secdb.alpinelinux.org/
2) Amazon Linux ALAS: https://alas.aws.amazon.com/AL2/alas.rss
3) RedHat RHSAs: https://www.redhat.com/security/data/oval/
4) Debian Linux CVE Tracker: https://security-tracker.debian.org/tracker/data/json
5) Github GHSAs: https://github.com/advisories
6) National Vulnerability Database (NVD): https://nvd.nist.gov/vuln/data-feeds
7) Oracle Linux OVAL: https://linux.oracle.com/security/oval/
8) RedHat Linux Security Data: https://access.redhat.com/hydra/rest/securitydata/
9) Suse Linux OVAL: https://ftp.suse.com/pub/projects/security/oval/
10) Ubuntu Linux Security: https://people.canonical.com/~ubuntu-security/

Στην πράξη η βάση δεδομένων είναι ένα αρχείο Sql με όνομα "vulnerability.db"και δεν απαιτεί κάποια ενέργεια ή διαχείριση από τον χρήστη , καθότι τα διαχειρίζεται όλα αυτόματα το εργαλείο.

Dagda

Χρησιμοποιεί και το εργαλείο ClamAv ως αντιικό λογισμικό για να μπορεί να ανιχνεύει trojans , ιούς , κακόβουλο λογισμικό και άλλες απειλές . Το εργαλείο Dagda υποστηρίζει μια πληθώρα από διανομές linux όπως :

4. Alpine
5. Debian / Ubuntu
6. OpenSUSE
7. Red Hat / CentOS / Fedora

Ταυτόχρονα μπορεί να αναλύσει dependencies από διάφορες γλώσσες συμπεριλαμβανομένων :

- java
- php
- python
- nodejs
- js
- ruby

Επιπροσθέτως ενσωματώνει το λογισμικό Falco , το οποίο του δίνει την δυνατότητα να παρακολουθεί τα δοχεία που εκτελούνται ήδη (running containers) , προκειμένου να ανιχνεύσει κάποια κακόβουλη δραστηριότητα.

Για την εγκατάσταση του εργαλείου απαιτούνται τα εξής :

4. Python 3.8
5. MongoDB 3.6
6. Pip3

Επίσης μπορούμε να εκτελέσουμε το εργαλείο απομακρυσμένα με την παρακάτω εντολή:

python3 dagda.py agent localhost:5000 -i <Docker image>

Docker Bench

Για να εκτελέσουμε το docker bench αρκεί να γράψουμε την παρακάτω εντολή:

docker run –rm –net host –pid host –userns host –cap add audit_control \

-e DOCKER_CONTENT_TRUST=$DOCKER_CONTENT_TRUST \

-v /*etc:*etc:ro \

-v /*usr/*bin/containerd:/usr/bin/containerd:ro \

-v /*usr/*bin/runc:/usr/runc:ro \

-v /*usr/*bin/systemd:/usr/systemd:ro \

-v *var/*lib:/var/lib:ro \

-v *var/run/docker.sock*:/var/*run/docker.sock*::ro \

--label docker_bench_security \

docker/docker-bench-security

Ωστόσο ανάλογα με την διανομή linux στην οποία εκτελείται το εργαλείο πρέπει να γίνουν κάποιες αλλαγές.

Harbor

Για την εγκατάσταση του απαιτεί να υπάρχουν εγκατεστημένα στο σύστημα :

2. Docker engine
3. Docker compose
4. Openssl

Όσον αφορά το hardware το εργαλείο Harbor έχει κάποιες ελάχιστες απαιτήσεις :

2. 2 cpu cores
3. Memory 4 GB
4. Disk 40 GB

Επιπλέον για να μπορέσει να λειτουργήσει το συγκεκριμένο εργαλείο απαιτεί να είναι ανοιχτές οι εξής δικτυακές πόρτες :

2. 443 με πρωτόκολλο HTTPS στην οποία το API του εργαλείου αποδέχεται HTTPS αιτήματα.
3. 4445 με πρωτόκολλο HTTPS για την σύνδεση με την υπηρεσία Content Trust του Docker.
4. 80 με πρωτόκολλο HTTP στην οποία το API του εργαλείου αποδέχεται HTTP αιτήματα.

Για να εγκαταστήσουμε το εργαλείο πρέπει πρώτα να βεβαιωθούμε ότι ικανοποιούνται οι παραπάνω συνθήκες . Κατόπιν πρέπει να εκτελέσουμε τα παρακάτω βήματα:

4. Download Harbor installer (https://github.com/goharbor/harbor/releases)
5. Ρύθμιση της HTTPS πρόσβασης στο Harbor
6. Ρύθμιση του YML αρχείου του Harbor
7. Ρύθμιση της εσωτερικής επικοινωνίας μεταξύ των διάφορων components του Harbor ώστε να χρησιμοποιεί και το πρωτόκολλο TLS.
8. Run installer file.

Τέλος να επισημάνουμε μερικά από τα σημαντικότερα “εξαρτήματα” του εργαλείου :

4. Postgresql
5. Redis
6. Beego
7. Chartmuseum
8. Docker/distribution
9. Docker/notary
10. Helm
11. Swagger-ui

Κλάση Orchestrator

Αρχικά στον χρήστη δίνεται η δυνατότητα να επιλέξει αν θέλει να εκτελεστούν τα εργαλεία για τις εικόνες που έχει (επιλογή scan)ή αν έχει ήδη στην κατοχή του τα αποτελέσματα από τα εργαλεία να εισάγει μόνο τα αρχεία με τα αποτελέσματα σε μορφή json(επιλογή risk).
Κατόπιν του ζητείται να εισάγει τον αριθμό των εικόνων για τις οποίες επιθυμεί να υπολογιστεί το ρίσκο.
Ανάλογα με την προηγούμενη επιλογή του εκτελούνται τα ακόλουθα:

Αν επέλεξε scan :

1. Εισάγει το path για το αρχείο clair-scanner.sh
2. Τα αρχεία στα οποία θα αποθηκευτούν τα αποτελέσματα από την εκτέλεση των 2 εργαλείων θα δημιουργηθούν αυτόματα από το εργαλείο .

Αν επέλεγε risk:

1. Σε αυτή τη περίπτωση πρέπει να εισάγει μόνο το μονοπάτι του φακέλου που είναι αποθηκευμένα τα αρχεία με τα αποτελέσματά των εργαλείων για τις εικόνες που τον ενδιαφέρουν.
2. ΣΗΜΑΝΤΙΚΌ. Για να μπορέσει το εργαλείο να διαβάσει τα αρχεία θα πρέπει :

1. Να έχουν συγκεκριμένο όνομα (όπως αυτό ορίζεται στις οδηγίες εκτέλεσης του προγράμματος)
2. Να βρίσκονται στο σωστό directory.

Στην συνέχεια πρέπει να ορίσει τις τιμές για τις μεταβλητές που αφορούν τις απαιτήσεις ασφάλειας (security requirements) και είναι:

1. cr -> confidentiality requirement
2. ir -> Integrity requirement
3. ar -> Availability requirement

Αμέσως μετά ξεκινά η διαδικασία υπολογισμού του ρίσκου στην οποία για κάθε αρχείο αποτελεσμάτων που διαθέτουμε (2 αρχεία κάθε φορά , ένα από κάθε εργαλείο , για μία εικόνα τη φορά) εκτελούμε τα παρακάτω βήματα:

1. Για το αρχείο που προέρχεται από την εκτέλεση του εργαλείου grype δημιουργούμε ένα αντικείμενο της κλάσης GrypeAnalyzer και περνάμε σαν παράμετρο το αρχείο αυτό.
2. Για το αρχείο που προέρχεται από την εκτέλεση του εργαλείου clair δημιουργούμε ένα αντικείμενο της κλάσης ClairAnalyzer και περνάμε σαν παράμετρο το αρχείο αυτό.
3. Δημιουργούμε ένα αντικείμενο της κλάσης CalcAssetRisk αι περνάμε τις εξής παραμέτρους:
   a) confidentiality requirement
   b) Integrity requirement
   c) Availability requirement
4. Κατόπιν επιλέγει την συνάρτηση που θα χρησιμοποιηθεί τόσο στον υπολογισμό του ρίσκου για μια εικόνα όσο και για το σύστημα . Έχει 4 επιλογές για το ρίσκο όσον αφορά την εικόνα δοχείου και πέντε για το σύστημα συνολικά .

   Όσον αφορά το συνολικό σύστημα οι επιλογές είναι οι εξής :
   a) no (Σε αυτή τη περίπτωση έχουμε μόνο μια εικόνα)
   b) Max (δίνεται η μέγιστη τιμή από όλα τα asset)
   c) average (υπολογίζεται ο μέσος όρος από τα asset)
   d) weight_aggr_auto (υπολογίζεται με βάση κάποια βάρη που υπολογίζει το σύστημα αυτόματα)
   e) weight_aggr_maual (υπολογίζεται με βάση κάποια βάρη που δίνει ο χρήστης – διαχειριστής του συστήματος)

   Για τις εικόνες έχει τις εξής επιλογές :
   f) Max (δίνεται η μέγιστη τιμή από όλα τα asset)
   g) average (υπολογίζεται ο μέσος όρος από τα asset)
   h) weight_aggr_auto (υπολογίζεται με βάση κάποια βάρη που υπολογίζει το σύστημα αυτόματα)
   i) weight_aggr_maual (υπολογίζεται με βάση κάποια βάρη που δίνει ο χρήστης – διαχειριστής του συστήματος .
5. Τώρα δημιουργούμε ένα directory με βάση την ημερομηνία και τη ώρα εκτέλεσης του προγράμματος έτσι ώστε κάθε εκτέλεση να καταγράφεται μοναδικά.

6. Δημιουργούμε αντικείμενο της κλάσης ImageRiskCalcAlgo η οποία υπολογίζει το ρίσκο για μια εικόνα και αποθηκεύουμε το αποτέλεσμα σε μια μεταβλητή με όνομα riskRes.
7. Κατόπιν εκτελούμε το αλγόριθμο υπολογισμού του ρίσκου για κάθε ένα απο τα εργαλεία ξεχωριστά δημιουργώντας δύο αντικείμενα της κλάσης CalcAssetRiskWithEachTool και καλώντας δύο φορές την μέθοδο imageRiskCaclAlgo με τις κατάλληλες παραμέτρους κάθε φορά.
8. Προσθέτουμε τις τιμές αυτές σε διαφορετικές λίστες για τον υπολογισμό αργότερα του ρίσκου ολόκληρου του συστήματος.
9. Κατόπιν καλούμε την μέθοδο calcImageProb της κλάσης ImageRiskCalcAlgo για να αποθηκεύσουμε σε μια λίστα την τιμή της πιθανότητας να εκτελεστεί τουλάχιστον μια ευπάθεια σε κάποιο asset μιας εικόνας . Αυτή η λίστα θα την χρησιμοποιήσουμε αργότερα για τον υπολογισμό του συνολικού ρίσκου στο σύστημα.
10. Στη συνέχεια δημιουργούμε ένα αντικείμενο της κλάσης Save_to_json_file και κατόπιν καλούμε την μέθοδο writeToFile για να αποθηκεύσουμε μέσα στο irectory που δημιουργήσαμε προηγουμένως ένα json αρχείο με τα αποτελέσματα για κάθε εικόνα και την writeToFileCves για να αποθηκεύσουμε τα cves που βρήκαμε για την συγκεκριμένη εικόνα.
11. Κατόπιν δημιουργούμε αντικείμενο της κλάσης OverallRisk για να υπολογίσουμε το ρίσκο για ολόκληρο το σύστημα περνώντας σαν παραμέτρους την λίστα που περιέχει τα υπολογισμένα ρίσκα για κάθε εικόνα και την λίστα με όλες τις εικόνες.
12. Μετά δημιουργούμε ένα νέο αντικείμενο της κλάσης Save_to_json_file , αλλά αυτή τη φορά καλούμε την μέθοδο storeSystemRisk για να αποθηκεύσουμε το συνολικό ρίσκο για στο σύστημα στο ίδιο directory , όπως και πριν.
13. Τέλος, δημιουργούμε δύο αντικείμενα της κλάσης OverrallRisk για να υπολογίσουμε το ρίσκο για ολόκληρο το σύστημα αν χρησιμοποιούσαμε κάθε εργαλείο ξεχωριστά.

GrypeRunner

Η δομή είναι η εξής :

1. Σε μια επαναληπτική διαδικασία δημιουργείται ένα αντικείμενο BufferedWriter και FileWriter για την αποθήκευση των αποτελεσμάτων στο αρχείο.
2. Δημιουργία ενός αντικειμένου της κλάσης Process της java προκειμένου να μπορέσουμε να εκτελέσουμε εντολή μέσα από το τερματικό.
3. Εκτέλεση της εντολής για την εκτέλεση του εργαλείου και αποθήκευση των αποτελεσμάτων στο αρχείο.

ClairScannerRunner

Η δομή είναι η εξής :

1. Ο χρήστης πρέπει να εισάγει την ip διεύθυνση του συστήματος στο οποίο εκτελείται το εργαλείο καθώς και το path για την θέση στο σύστημα αρχείων , όπου είναι αποθηκευμένο το αρχείο clair-scanner.sh .
2. Σε μια επαναληπτική διαδικασία δημιουργείται ένα αντικείμενο BufferedWriter και FileWriter για την αποθήκευση των αποτελεσμάτων στο αρχείο.
3. Δημιουργία ενός αντικειμένου της κλάσης Process της java προκειμένου να μπορέσουμε να εκτελέσουμε εντολή μέσα από το τερματικό.
4. Εκτέλεση της εντολής για την εκτέλεση του εργαλείου και αποθήκευση των αποτελεσμάτων στο αρχείο.

GrypeAnalyzer class και ClairAnalyzer class

Οι δυο παραπάνω κλάσεις έχουν παρόμοια δομή όπως είπαμε, υλοποιούν έναν κατασκευαστή, την μέθοδο και την μέθοδο analyzer την οποία και θα αναλύσουμε εδώ:
Τα βήματα που ακολουθούνται σε αυτή τη μέθοδο είναι τα εξής:

1. Δημιουργία ενός αντικειμένου JsonParser και FileReader για την ανάγνωση από αρχείο.
2. Διάβασμα των δεδομένων και μετατροπή τους σε ένα JsonObject (obj).
3. «Αποθήκευση» αυτού του obj σε ένα άλλο JsonObject με όνομα grypeFileData για τα αποτελέσματα του εργαλείου Grype και clairFileData για αυτά του εργαλείου clair-scanner.
4. Δημιουργούμε ένα JsonArray (j1) και «αποθηκεύουμε» σε αυτό:
    1. Για το εργαλείο clair-scanner:
        a) Την πληροφορία που αντιστοιχεί στο αναγνωριστικό «vulnerabilities»
    2. Για το εργαλείο Grype:
        a) Την πληροφορία που αντιστοιχεί στο αναγνωριστικό «matches»
5. Έπειτα και για τα δύο εργαλεία εκτελούμε μια επαναληπτική διαδικασία για κάθε στοιχείο του JSonArray j1 και εκτελούμε τα παρακάτω βήματα :
    1. Δημιουργούμε ένα JSonObject (j2) και «αποθηκεύουμε σε αυτό το πρώτο στοιχείο του πίνακα j1.
    2. Για το εργαλείο clair-scanner:
        a) Δημιουργούμε ένα Αλφαριθμιτικό (String) με όνομα asset και «αποθηκεύουμε» σε αυτό την πληροφορία από το αναγνωριστικό «featurename» που πήραμε από το j2
        b) Δημιουργούμε ένα Αλφαριθμιτικό με όνομα j3 και «αποθηκεύουμε» σε αυτό την πληροφορία από το αναγνωριστικό «fixedby» που πήραμε από το j2.
    3. Για το εργαλείο Grype:
        a) Δημιουργούμε ένα JSonObject με όνομα j3 και «αποθηκεύουμε» σε αυτό την πληροφορία από το αναγνωριστικό «vulnerability» που πήραμε από το j2
        b) Δημιουργούμε ένα JSonObject με όνομα j4 και «αποθηκεύουμε» σε αυτό την πληροφορία από το αναγνωριστικό «fix» που πήραμε από το j3.
        c) Δημιουργούμε ένα JsonArray (j6) και «αποθηκεύουμε» σε αυτό την πληροφορία από το αναγνωριστικό «matchDetails» που πήραμε από το j2.

d) Δημιουργούμε ένα JSonObject με όνομα j7 και «αποθηκεύουμε» σε αυτό την πληροφορία από το πρώτο στοιχείο του πίνακα j6
e) Δημιουργούμε ένα JSonObject με όνομα j8 και «αποθηκεύουμε» σε αυτό την πληροφορία από το αναγνωριστικό «searchedBy» που πήραμε από το j7.
f) Δημιουργούμε ένα JSonObject με όνομα j9 και «αποθηκεύουμε» σε αυτό την πληροφορία από το αναγνωριστικό «package» που πήραμε από το j8.
g) Δημιουργούμε ένα Αλφαριθμιτικό με όνομα asset και «αποθηκεύουμε» σε αυτό την πληροφορία από το αναγνωριστικό «name» που πήραμε από το j9.

Από εδώ και πέρα η διαδικασία είναι ίδια και για τα δύο εργαλεία και είναι η εξής :

4. Δημιουργούμε ένα σλυνολο τύπου <String> με όνομα swap.
5. αποθηκεύουμε το όνομα του asset και το cve-id στις αντίστοιχες μεταβλητές.
6. Ελέγχουμε εάν το hashMap finalAssets είναι άδειο.
   a) Αν είναι άδειο τότε:
      1. Δημιουργούμε ένα σύνολο με όνομα temp τύπου <String>.
      2. Προσθέτουμε σε αυτή το αναγνωριστικό της ευπάθειας που (μεταβλητή cveId)
      3. Αντιγράφουμε το σύνολο temp στην λίστα swap.
      4. Δημιουργούμε μια νέα εγγραφή στο HashMap finalAssets με κλειδί το asset (μεταβλητή assetName) και σαν value το σύνολο swap.
   b) Αν δεν είναι άδειο:
      1. Ελέγχουμε αν υπάρχει εγγραφή με το assetName
         I. Αν δεν υπάρχει τότε εκτελούμε τα βήματα 1,2 και 4 από το βήμα 6.a
         II. Αν υπάρχει τότε:
            1. Αν το σύνολο με τις ευπάθειες για την συγκεκριμένη εγγραφή του HashMap finalAssets (δηλαδή για το συγκεκριμένο asset) περιέχει την ευπάθεια που διαβάσαμε δεν κάνουμε τίποτα.
            2. Αν δεν την περιέχει τότε προσθέτουμε την ευπάθεια στο συγκεκριμένο σύνολο.

Αφού ολοκληρωθεί η παραπάνω διαδικασία τότε και στις δύο κλάσεις ακολουθείται η ίδια λογική με παραπάνω μόνο που αυτή τη φορά τα αποτελέσματα αποθηκεύονται σε τοπικές δομές της κάθε κλάσης προκειμένου να χρησιμοποιηθούν για τον υπολογισμό του ρίσκου χρησιμοποιώντας το κάθε εργαλείο ξεχωριστά.

RetrieveDataFromOnlineDB class

Αυτή η κλάση χρησιμοποιείται για να "διαβάζουμε" τα δεδομένα από την εκάστοτε διαδικτυακή βάση δεδομένων (είτε nist είτε cveapi) και λειτουργεί ως εξής:

1. Διαβάζουμε το περιεχόμενο από την εκάστοτε βάση δεδομένων με την μέθοδο read και την αποθηκεύουμε στην μεταβλητή Text η οποία είναι τύπου αλφαριθμητική.
2. Την μετατρέπουμε σε JsonObject με όνομα json.
3. Στην περίπτωση που χρησιμοποιούμε την βάση nist εκτελούμε τα παρακάτω βήματα:

1. Δημιουργούμε ένα JSonObject με όνομα j1 και «αποθηκεύουμε» σε αυτό την πληροφορία από το αναγνωριστικό «result» που πήραμε από το json.
2. Δημιουργούμε ένα JSonArray με όνομα j2 και «αποθηκεύουμε» σε αυτό την πληροφορία από το αναγνωριστικό «CVE_Items» που πήραμε από το j1.
3. Δημιουργούμε ένα JSonObject με όνομα j3 και «αποθηκεύουμε» σε αυτό την πληροφορία από το πρώτο στοιχείο του πίνακα j2.
4. Δημιουργούμε ένα JSonObject με όνομα j4 και «αποθηκεύουμε» σε αυτό την πληροφορία από το αναγνωριστικό «impact» που πήραμε από το j3.
5. Δημιουργούμε ένα JSonObject με όνομα j5 και «αποθηκεύουμε» σε αυτό την πληροφορία από το αναγνωριστικό «baseMetricV2» που πήραμε από το j4.
6. Δημιουργούμε ένα JSonObject με όνομα j6 και «αποθηκεύουμε» σε αυτό την πληροφορία από το αναγνωριστικό «cvssV2» που πήραμε από το j5.

4. Στην περίπτωση που χρησιμοποιούμε την βάση cevapi εκτελούμε μόνο τα βλήματα 4,5 και 6.

CalcAssetRisk class

Είναι η κλάση στην οποία συγκεντρώνονται τα αποτελέσματα αφότου έχουν αναλυθεί από τις κλάσεις ClairAnalyzer και GrypeAnalyzer και υπολογίζεται το τελικό ρίσκο για κάθε εικόνα. Η δομή της κλάσης έχει ως εξής :
Αρχικά δημιουργούμε επτά (7) δομές για την διαχείριση των δεδομένων:

1. ArrayList<Double> risks
2. ArrayList<Double> probabilities
3. Map<String, Double> assetWithRisk
4. HashMap <String, Double> isFixed
5. Map<String, ArrayList<String>> *finalAssets*
6. Map<String, ArrayList<String>> assetWithProb
7. ArrayList<Double> impacts

Ο κατασκευαστής της κλάσης δέχεται σαν παραμέτρους τα εξής:

1. String cr,
2. String ir,
3. String ar

Οι τρεις αυτές παράμετροι είναι οι απαιτήσεις ασφαλείας που εισήγαγε ο χρήστης - διαχειριστής κατά την εκτέλεση του προγράμματος και θα χρησιμοποιηθούν στον υπολογισμό του ρίσκου.
Αρχικά για κάθε εγγραφή του Map finalAssets αναζητήσουμε την αντίστοιχη εγγραφή στις 2 online βάσεις δεδομένων που έχουμε επισημάνει παραπάνω σειριακά . Η αναζήτηση αυτή γίνεται ανά asset (κάθε εγγραφή του Map περιέχει σε μια λίστα όλες τις ευπάθειες που βρέθηκαν σε αυτό το asset) με βάση το εκάστοτε CVE-ID.

Εφόσον εντοπιστεί η ευπάθεια στην βάση τότε χρησιμοποιούμε την κλάση RetrieveDataFromOnlineDB για να φέρουμε πίσω τα δεδομένα σε μορφή JSON και με τη χρήση ενός jsonObject να μπορέσουμε να τα διαβάσουμε.
Χρησιμοποιώντας την μέθοδο get παίρνουμε την αλφαριθμητική τιμή που υπάρχει σε κάθε πεδίο του jsonObject και στη συνέχεια ανάλογα με την τιμή αυτή εισάγουμε και την αντίστοιχη αλγεβρική τιμή στην αντίστοιχη μεταβλητή που αναπαριστά το συγκεκριμένο πεδίο στον αλγόριθμο.
Κατόπιν , με βάση τις τιμές του προηγούμενου βήματος , υπολογίζουμε τον όρο “probability” .
Εν συνεχεία με βάσει τις αλφαριθμητικές τιμές που έχει δώσει ο χρήστης για τις απαιτήσεις ασφάλειας εκχωρούμε τις αντίστοιχες αλγεβρικές τιμές στις αντίστοιχες μεταβλητές .
Ακολούθως ελέγχουμε αν υπάρχει εγγραφή με το συγκεκριμένο cve-id στο HashMap isFixed. Αν υπάρχει τότε σημαίνει ότι η συγκεκριμένη ευπάθεια βρίσκεται στο στάδιο της αντιμετώπισης .
Επομένως εκχωρούμε στην αντίστοιχη μεταβλητή την τιμή που υπάρχει στο HashMap για το συγκεκριμένο cve-id και θέτουμε την τιμή της μεταβλητής flag = true. Εάν δεν βρεθεί τότε θα εκχωρήσουμε στην συγκεκριμένη μεταβλητή την τιμή 0.0.
Μετά υπολογίζουμε τους όρους impact , το probabilitySum καθώς και το probForP_one.
Τέλος υπολογίζουμε το finalRisk και το στρογγυλοποιούμε στα 2 δεκαδικά ψηφία για την συγκεκριμένη ευπάθεια .
Όταν ολοκληρωθεί η εκτέλεση της συγκεκριμένης κλάσης έχουν υπολογιστεί όλα τα ρίσκα για τις ευπάθειες που ανιχνεύθηκαν στο συγκεκριμένο asset.

ImageRiskCalcAlgo class

Η κλάση αυτή χρησιμοποιείται για τον υπολογισμό του ρίσκου σε μια εικόνα δοχείου.
Αναλυτικότερα:
Για την μέθοδο calcImageProb :
Σε μια επαναληπτική διαδικασία για κάθε στοιχείο του προαναφερθέντος Map (assetsWithProb) χρησιμοποιώντας την παρακάτω εντολή :

```
imageProb = imageProb * (1 – entry.getValue());
```

υπολογίζουμε το πρώτο παράγοντα για της σχέσης για τον υπολογισμό της πιθανότητας .
Κατόπιν χρησιμοποιούμε την εντολή : imageProb = 1 – imageProb; για να υπολογίζουμε και τελευταίο κομμάτι της σχέσης που θα μας δώσει τελικά την ζητούμενη τιμή της πιθανότητας.

Για την μέθοδο calcWeightAggMan:
Σε μια επαναληπτική διαδικασία για τα στοιχεία του πρώτου Map :

1. Διαβάζει από το χρήστη το βάρος για κάθε asset και το εκχωρεί στην μεταβλητή Wk.
2. Κατόπιν παίρνει από το πρώτο Map την τιμή για το υπολογισθέν ρίσκο του asset και την εκχωρεί στην μεταβλητή Rk .
3. Μετά υπολογίζει και το άθροισμα του γινομένου των δύο παραπάνω μεταβλητών και το εκχωρεί στην μεταβλητή multiSum.

Τέλος υπολογίζει το ρίσκο για την εικόνα βάσει του τύπου 1 / Sumk(Wk) * Sumk(Wk * Rk) και το στρογγυλοποιεί στα 2 δεκαδικά ψηφία .

Για την μέθοδο calcWeightAggAuto:

Σε μια επαναληπτική διαδικασία για τα στοιχεία του πρώτου Map :

1. Με την εντολή assetsWithProb.get(entry.getKey()); παίρνει για το κάθε asset του πρώτου Map (assetsWithRisks) την τιμή για την πιθανότητα από το δεύτερο Map (assetsWithProb) και την εκχωρεί στη αντίστοιχη μεταβλητή (Pk) .
2. Κατόπιν παίρνει από το πρώτο Map την τιμή για το υπολογισθέν ρίσκο του asset και την εκχωρεί στην μεταβλητή Rk .
3. Μετά υπολογίζει και το άθροισμα του γινομένου των δύο παραπάνω μεταβλητών και το εκχωρεί στην μεταβλητή multiSum.
4. Τέλος υπολογίζει το ρίσκο για την εικόνα βάσει του τύπου 1 / Sumk(Pk) * Sumk(Pk * Rk) και το στρογγυλοποιεί στα 2 δεκαδικά ψηφία .

OverallRisk class

Η κλάση αυτή χρησιμοποιείται για τον υπολογισμό του ρίσκου σε ολόκληρο το σύστημα.
Για την μέθοδο calcWeightAggMan:
Σε μια επαναληπτική διαδικασία για τα στοιχεία του πρώτου Map :

1. Διαβάζει από το χρήστη το βάρος για κάθε εικόνα και το εκχωρεί στην μεταβλητή Wk.
2. Κατόπιν παίρνει από την λίστα με τα υπολογισμένα ρίσκα την αντίστοιχη τιμή και την προσθέτει στη μεταβλητή Rk.
3. Μετά υπολογίζει και το άθροισμα του γινομένου των δύο παραπάνω μεταβλητών και το προσθέτει στην μεταβλητή multiSu
4. Τέλος υπολογίζεται το συνολικό ρίσκο για το σύστημα σύμφωνα με τον τύπο 1 / Sumk(Wk) * Sumk(Wk * Rk) και στρογγυλοποιείται στα δύο δεκαδικά ψηφία (Ο χρήστης εισήγαγε 4).

Για την μέθοδο calcWeightAggAuto:
Σε μια επαναληπτική διαδικασία τα στοιχεία της λίστας με τα ονόματα:

1. Υπολογίζει την τιμή (άθροισμα) για την πιθανότητα να εκτελεστεί κάποια ευπάθεια σε κάποιο asset , παίρνοντας τις αντίστοιχες τιμές για κάθε εικόνα από την λίστα assetsWithProb και την εκχωρεί στη αντίστοιχη μεταβλητή (Pk) .
2. Κατόπιν παίρνει από την λίστα με τα ρίσκα για κάθε εικόνα (r) την τιμή για το κάθε υπολογισθέν ρίσκο και την προσθέτει στην μεταβλητή Rk .
3. Μετά υπολογίζει και το άθροισμα του γινομένου των δύο παραπάνω μεταβλητών και το προσθέτει στην μεταβλητή multiSum.
4. Τέλος υπολογίζει το ρίσκο για την εικόνα βάσει του τύπου 1 / Sumk(Pk) * Sumk(Pk * Rk) και το στρογγυλοποιεί στα 2 δεκαδικά ψηφία (Ο χρήστης εισήγαγε 3).

Εάν ο χρήστης εισήγαγε 5 τότε σημαίνει ότι ελέγχθηκε μια μόνο εικόνα και κατά συνέπεια το ολικό ρίσκο του συστήματος είναι ίσο με το ρίσκο της εικόνας . Έτσι σε αυτή τη περίπτωση στη μεταβλητή overallScore εκχωρούμε το περιεχόμενο της λίστας risks με την εντολή :
overallScore = risks.get(0);

Τέλος στρογγυλοποιούμε στα δύο δεκαδικά ψηφία .

CalcAssetRiskWithEachTool class

Η κλάση αυτή είναι υπεύθυνη για τον υπολογισμό του ρίσκου χρησιμοποιώντας κάθε εργαλείο ξεχωριστά, χωρίς να συνδυάζονται τα αποτελέσματά τους.
Ο κατασκευαστής της κλάσης δέχεται σαν παραμέτρους τα εξής:

1. String cr,
2. String ir,
3. String ar
4. Map<String, ArrayList<String>> finalAssets
5. HashMap<String, Double> isFixed

Οι τρεις πρώτες παράμετροι είναι οι απαιτήσεις ασφαλείας που εισήγαγε ο χρήστης - διαχειριστής κατά την εκτέλεση του προγράμματος και θα χρησιμοποιηθούν στον υπολογισμό του ρίσκου.
Αρχικά για κάθε εγγραφή του Map finalAssets αναζητήσουμε την αντίστοιχη εγγραφή στις 2 online βάσεις δεδομένων που έχουμε επισημάνει παραπάνω σειριακά . Η αναζήτηση αυτή γίνεται ανά asset (κάθε εγγραφή του Map περιέχει σε μια λίστα όλες τις ευπάθειες που βρέθηκαν σε αυτό το asset) με βάση το εκάστοτε CVE-ID.
Εφόσον εντοπιστεί η ευπάθεια στην βάση τότε χρησιμοποιούμε την κλάση RetrieveDataFromOnlineDB για να φέρουμε πίσω τα δεδομένα σε μορφή JSON και με τη χρήση ενός jsonObject να μπορέσουμε να τα διαβάσουμε.
Χρησιμοποιώντας την μέθοδο get παίρνουμε την αλφαριθμητική τιμή που υπάρχει σε κάθε πεδίο του jsonObject και στη συνέχεια ανάλογα με την τιμή αυτή εισάγουμε και την αντίστοιχη αλγεβρική τιμή στην αντίστοιχη μεταβλητή που αναπαριστά το συγκεκριμένο πεδίο στον αλγόριθμο.
Κατόπιν , με βάση τις τιμές του προηγούμενου βήματος , υπολογίζουμε τον όρο “probability” .
Εν συνεχεία με βάσει τις αλφαριθμητικές τιμές που έχει δώσει ο χρήστης για τις απαιτήσεις ασφάλειας εκχωρούμε τις αντίστοιχες αλγεβρικές τιμές στις αντίστοιχες μεταβλητές .
Ακολούθως ελέγχουμε αν υπάρχει εγγραφή με το συγκεκριμένο cve-id στο HashMap isFixed. Αν υπάρχει τότε σημαίνει ότι η συγκεκριμένη ευπάθεια βρίσκεται στο στάδιο της αντιμετώπισης .
Επομένως εκχωρούμε στην αντίστοιχη μεταβλητή την τιμή που υπάρχει στο HashMap για το συγκεκριμένο cve-id και θέτουμε την τιμή της μεταβλητής flag = true. Εάν δεν βρεθεί τότε θα εκχωρήσουμε στην συγκεκριμένη μεταβλητή την τιμή 0.0.
Μετά υπολογίζουμε τους όρους impact , το probabilitySum καθώς και το probForP_one.

Τέλος υπολογίζουμε το finalRisk και το στρογγυλοποιούμε στα 2 δεκαδικά ψηφία για την συγκεκριμένη ευπάθεια .

Όταν ολοκληρωθεί η εκτέλεση της συγκεκριμένης κλάσης έχουν υπολογιστεί όλα τα ρίσκα για τις ευπάθειες που ανιχνεύθηκαν στο συγκεκριμένο asset.

ReportGenerator class

Η κλάση αυτή είναι υπεύθυνη για την αποθήκευση των αποτελεσμάτων στα κατάλληλα αρχεία.

Μέθοδος reportToFile:

Με την χρήση ενός bufferWritter και JsonObjects γράφουμε όλες τις πληροφορίες που μας ενδιαφέρουν σε ένα Json αρχείο.

Μέθοδος createDir:

Δεν δέχεται κάποια παράμετρο. Με την χρήση ων κλάσεων SimpleDateFormat , Date και File της java, δημιουργεί μέσα στο φάκελο results , νέο φάκελο με την ημερομηνία και την ώρα της εκτέλεσης του εργαλείου, στον οποίο θα αποθηκευτούν όλα τα αποτελέσματα.