ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΙΓΑΙΟΥ
ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΑΚΩΝ ΚΑΙ ΕΠΙΚΟΙΝΩΝΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

CLI Tool for Security Checking of Docker Containers

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

Δημήτριου Ιάσονα Κωστούλα

Επιβλέπων: Κυριάκος Κρητικός

Μέλη εξεταστικής επιτροπής: Δημήτριος Σκούτας, Σπύρος Κοκολάκης

Σάμος, Οκτώμβιος 2024

Η σελίδα αυτή είναι σκόπιμα λευκή.

Πρόλογος και ευχαριστίες

Η παρούσα διπλωματική εργασία πραγματοποιήθηκε στο πλαίσιο ολοκλήρωσης των σπουδών μου στο Πανεπιστήμιο και έχει ως στόχο την ανάπτυξη και μελέτη ενός εργαλείου διεπαφής γραμμής εντολών (CLI) για την ασφάλεια των Docker δοχείων. Η τεχνολογία των δοχείων Docker έχει σημειώσει ραγδαία άνοδο στον χώρο της ανάπτυξης και παράδοσης λογισμικού, ενώ παράλληλα οι προκλήσεις που αφορούν στην ασφάλειά της καθιστούν την έρευνα σε αυτό το πεδίο ιδιαίτερα ενδιαφέρουσα και επίκαιρη. Μέσα από την εργασία αυτή, επιχειρήθηκε η ανάλυση και αντιμετώπιση αυτών των προκλήσεων, με τη δημιουργία ενός πρακτικού εργαλείου για την ενίσχυση της ασφάλειας σε περιβάλλοντα ανάπτυξης και παραγωγής.

Θα ήθελα να ευχαριστήσω ιδιαίτερα τον επιβλέποντα της διπλωματικής εργασίας, καθηγητή κ. Κυριάκο Κρητικό, για την αμέριστη υποστήριξη, την καθοδήγηση και τις πολύτιμες συμβουλές που μου παρείχε καθ' όλη τη διάρκεια της εκπόνησης της εργασίας. Η συμβολή του ήταν καταλυτική για την επιτυχή ολοκλήρωση της παρούσας μελέτης.

Επίσης, θα ήθελα να ευχαριστήσω τα μέλη της εξεταστικής επιτροπής, κ. Δημήτριο Σκούτα και κ. Σπύρο Κοκολάκη, για τον χρόνο τους, την προσοχή και τις εποικοδομητικές παρατηρήσεις τους, που συνέβαλαν στην ολοκλήρωση της εργασίας με τον καλύτερο δυνατό τρόπο.

ΔΗΜΗΤΡΙΟΥ ΙΑΣΟΝΑ ΚΩΣΤΟΥΛΑ

Τμήμα Μηχανικών Πληροφοριακών και Επικοινωνιακών Συστημάτων

ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΙΓΑΙΟΥ

Η σελίδα αυτή είναι σκόπιμα λευκή.

Πίνακας περιεχομένων

Λίστα Σχημάτων

Λίστα Πινάκων

Ακρωνύμια

CLI	Command Line Interface
CI/CD	Continuous Integration / Continuous Delivery

Περίληψη

Η παρούσα διπλωματική εργασία εστιάζει στην ανάπτυξη ενός εργαλείου διεπαφής γραμμής εντολών (CLI) για τον έλεγχο ασφάλειας των Docker δοχείων ή των εικόνων τους. Το εργαλείο αυτό επιτρέπει στους χρήστες να ανιχνεύουν ευάλωτα σημεία και κακόβουλο λογισμικό, να εξετάζουν τις εξαρτήσεις και να ανακαλύπτουν μη κρυπτογραφημένα μυστικά στα δοχεία ή τις εικόνες τους. Η τελική αναφορά του ελέγχου αποθηκεύεται τοπικά, προσφέροντας μια εύχρηστη και ευέλικτη λύση για την ασφαλή διαχείριση δοχείων σε περιβάλλοντα ανάπτυξης και παραγωγής. Το ανεπτυγμένο εργαλείο παρουσιάζει επίσης το πλεονέκτημα πως μπορεί να ενσωματωθεί εύκολα σε αγωγούς CI/CD (Continuous Integration / Continuous Delivery) (Συνεχούς Ενοποίησης / Συνεχούς Παράδοσης).

Λέξεις Κλειδιά: *Docker, ασφάλεια, ανίχνευση τρωτοτήτων, ανίχνευση κακόβουλου λογισμικού, ανάλυση εξαρτήσεων, εντοπισμός μυστικών, διεπαφή γραμμής εντολών (CLI), ασφάλεια δοχείων, σάρωση εικόνας, συνεχής ενσωμάτωση, συνεχής ανάπτυξη, CI/CD, DevOps*

Abstract

This thesis focuses on the development of a command line interface (CLI) tool for security testing of Docker containers or their images. This tool allows users to scan for vulnerabilities and malware, examine dependencies, and discover unencrypted secrets in their containers or images. The final audit report is stored locally, offering an easy-to-use and flexible solution for secure container management in development and production environments. The developed tool also has the advantage of being easily integrated into CI/CD (Continuous Integration / Continuous Delivery) pipelines.

Keywords: *Docker, security, vulnerability scanning, malware detection, dependency analysis, secret detection, command-line interface (CLI), container security, image scanning, continuous integration, continuous deployment, CI/CD, DevOps*

1

Εισαγωγή

1.1 Δοχεία & Ασφάλεια

Η τεχνολογία δοχείων προσφέρει μια αποδοτική μέθοδο απομόνωσης εφαρμογών και των εξαρτήσεών τους, επιτρέποντας την ανάπτυξή τους σε διάφορα περιβάλλοντα χωρίς προβλήματα συμβατότητας. Παρέχει ελαφρύτερες και πιο ευέλικτες λύσεις από τις παραδοσιακές εικονικές μηχανές, με ταχύτερη εκκίνηση και μικρότερο αποτύπωμα πόρων. Η υιοθέτηση της τεχνολογίας δοχείων έχει αυξηθεί ραγδαία τα τελευταία χρόνια, με εργαλεία όπως το Docker και συστήματα ορχηστρώσεων όπως το Kubernetes να παίζουν κεντρικό ρόλο στην αγορά, ειδικά στις υποδομές DevOps και στο Υπολογιστικό Νέφος (Cloud Computing). [1]

Η ανάγκη για ασφαλή διαχείριση της τεχνολογίας δοχείων αναδεικνύεται ολοένα και πιο κρίσιμη καθώς οι επιθέσεις σε λογισμικό και υποδομές γίνονται πιο εξελιγμένες. Τα υπάρχοντα εργαλεία ελέγχου ασφάλειας δοχείων συχνά αποτυγχάνουν να προσφέρουν μια ενοποιημένη και ευρέως προσβάσιμη λύση, καθιστώντας απαραίτητη την ανάπτυξη ενός ενιαίου, εύχρηστου εργαλείου.

1.2 Συνεισφορά Διπλωματικής

Αυτή η διπλωματική εργασία προσδιορίζει τα κενά ως προς τις επιθέσεις κακόβουλου λογισμικού, τρωτά σημεία στις εξαρτήσεις, την ύπαρξη ορατών μη κρυπτογραφημένων μυστικών στον πηγαίο κώδικα, καθώς και ευάλωτα σημεία, ενώ προσφέρει μια λύση που ενσωματώνει πολλαπλά εργαλεία σάρωσης σε ένα συνεκτικό Command Line Interface (CLI) (Διεπαφή Χρήσης Γραμμής Εντολών), βελτιώνοντας την ακρίβεια και την ευκολία χρήσης ως προς την ανίχνευση και αντιμετώπιση αυτών των προβλημάτων. [2]

Τα πλεονεκτήματα της συνεισφοράς αυτής περιλαμβάνουν τη βελτίωση της ακρίβειας και της αποδοτικότητας των σαρώσεων, καθώς και την απλοποίηση της διαδικασίας ελέγχου ασφάλειας μέσω της χρήσης ενός εύχρηστου CLI. Οι χρήστες μπορούν εύκολα να διαμορφώσουν τις σαρώσεις και να λάβουν λεπτομερείς αναφορές για την κατάσταση ασφάλειας των δοχείων τους, κάτι που είναι εξαιρετικά χρήσιμο για την ενσωμάτωση του εργαλείου σε αγωγούς CI/CD.

Ακρίβεια

Η ενσωμάτωση πολλαπλών εργαλείων σκαναρίσματος σε ένα ενιαίο CLI αυξάνει την ακρίβεια των ελέγχων ασφάλειας με τους εξής τρόπους:

- **Πολλαπλές Πηγές Δεδομένων**: Χρησιμοποιώντας δεδομένα από διαφορετικά εργαλεία και βάσεις δεδομένων, το εργαλείο μπορεί να εντοπίζει περισσότερες και πιο ποικιλόμορφες ευπάθειες.
- **Συγκριτική Ανάλυση**: Η δυνατότητα να παραθέτει και να συγκρίνει αποτελέσματα από διάφορα εργαλεία αυξάνει την αξιοπιστία των ανιχνεύσεων, καθώς οι ευπάθειες που επιβεβαιώνονται από πολλαπλές πηγές είναι πιο πιθανό να είναι έγκυρες.
- **Συνεχής Ενημέρωση**: Η χρήση εργαλείων που ενημερώνονται συνεχώς με τις τελευταίες πληροφορίες για ευπάθειες διασφαλίζει ότι οι σαρώσεις είναι πάντα βασισμένες στα πιο πρόσφατα δεδομένα.

Ευκολία Χρήσης

Η ευκολία χρήσης επιτυγχάνεται μέσω της συνεκτικής διεπαφής και των αυτοματοποιημένων διαδικασιών:

- **Ενοποιημένη Διεπαφή**: Αντί να απαιτείται από τους χρήστες να μάθουν και να χρησιμοποιούν πολλαπλά διαφορετικά εργαλεία με διαφορετικές διεπαφές χρήστη, η ενιαία διεπαφή CLI επιτρέπει την εκτέλεση όλων των σαρώσεων από ένα σημείο.
- **Αυτοματοποιημένες Σαρώσεις**: Η δυνατότητα ενσωμάτωσης του εργαλείου σε CI/CD αγωγούς επιτρέπει την αυτόματη εκτέλεση των ελέγχων ασφάλειας σε κάθε φάση της ανάπτυξης και της παραγωγής.
- **Συγκεντρωτική Αναφορά**: Το εργαλείο παράγει ενιαίες αναφορές που περιλαμβάνουν αποτελέσματα από όλα τα εργαλεία σάρωσης, διευκολύνοντας την ανάλυση και την αντιμετώπιση των προβλημάτων.

Πολλαπλές Δυνατότητες Ελέγχου Ασφάλειας

Οι βασικές λειτουργίες ελέγχου ασφάλειας που δεν καλύπτονται συνδυαστικά από τα διαθέσιμα εργαλεία περιλαμβάνουν την ανίχνευση κακόβουλου λογισμικού, την ανάλυση των εξαρτήσεων, την ανίχνευση τρωτοτήτων και τον εντοπισμό μη κρυπτογραφημένων μυστικών. Κάθε εργαλείο συνήθως εξειδικεύεται σε έναν από αυτούς τους τομείς, αλλά δεν παρέχει ολοκληρωμένη κάλυψη όλων των πιθανών ζητημάτων ασφάλειας. Για παράδειγμα, το ClamAV είναι εξαιρετικό στην ανίχνευση κακόβουλου λογισμικού, αλλά δεν προσφέρει λειτουργίες ανάλυσης εξαρτήσεων ή ανίχνευσης τρωτοτήτων. Αντίστοιχα, το Dependency-Check

επικεντρώνεται στις ευπάθειες εξαρτήσεων, αλλά δεν μπορεί να ανιχνεύσει κακόβουλο λογισμικό ή μη κρυπτογραφημένα μυστικά.

1.3 Δομή Αναφοράς

Η υπόλοιπη δομή της αναφοράς είναι ως εξής:

- **Υπόβαθρο**: Παρουσιάζονται οι βασικές έννοιες των δοχείων, η ασφάλεια των δοχείων Docker και τα εργαλεία ελέγχου ασφάλειας.
- **Ανάπτυξη**: Αναλύεται το μοντέλο ανάπτυξης του εργαλείου και τα κύρια αποτελέσματα από την εφαρμογή του ανά δραστηριότητα ανάπτυξης.
- **Εγκατάσταση & Επίδειξη**: Παρέχονται οδηγίες εγκατάστασης του εργαλείου και επίδειξη της λειτουργίας του μέσω σεναρίων χρήσης.
- **Συμπεράσματα & Μελλοντική Εργασία**: Συνοψίζονται τα ευρήματα της εργασίας και προτείνονται κατευθύνσεις για μελλοντική βελτίωση και επέκταση του εργαλείου.

2

Υπόβαθρο

2.1 Δοχεία (Containers)

Τα δοχεία είναι ελαφριές, φορητές, και αυτόνομες μονάδες λογισμικού που περιέχουν όλες τις εξαρτήσεις που χρειάζονται για να τρέξουν έναν κώδικα [3]. Συγκριτικά με τις εικονικές μηχανές, προσφέρουν μεγαλύτερη ευελιξία και αποδοτικότητα, κάτι που τα καθιστά ιδανικά για τη διαχείριση μικροϋπηρεσιών και εφαρμογών νέφους [4].

Τα δοχεία βασίζονται στην τεχνολογία της εικονικοποίησης, αλλά σε αντίθεση με τις παραδοσιακές εικονικές μηχανές, δεν περιλαμβάνουν ολόκληρο το λειτουργικό σύστημα. Αυτό τα καθιστά πιο ελαφριά και ταχύτερα στην εκκίνηση. Χρησιμοποιώντας κοινόχρηστους πόρους από τον κεντρικό πυρήνα του λειτουργικού συστήματος, τα δοχεία μπορούν να κλιμακώνονται πιο αποτελεσματικά και να προσφέρουν καλύτερη απομόνωση εφαρμογών. [2]

Μια άλλη βασική πτυχή των δοχείων είναι η φορητότητά τους. Ένα δοχείο μπορεί να δημιουργηθεί σε ένα περιβάλλον ανάπτυξης και στη συνέχεια να μεταφερθεί και να εκτελεστεί ακριβώς το ίδιο σε οποιοδήποτε άλλο περιβάλλον, είτε αυτό είναι τοπικό είτε στο νέφος [3]. Αυτή η ιδιότητα μειώνει δραστικά τα προβλήματα που σχετίζονται με τις διαφορές περιβάλλοντος, κάτι που ήταν ένα σημαντικό πρόβλημα στην παραδοσιακή ανάπτυξη λογισμικού [5].

Επίσης, τα δοχεία υποστηρίζουν τη διαχείριση μικροϋπηρεσιών. Οι μικροϋπηρεσίες αντιστοιχούν σε μια αρχιτεκτονική όπου η εφαρμογή διασπάται σε μικρότερα, ανεξάρτητα κομμάτια δηλ. τις μικροϋπηρεσίες), το καθένα από τα οποία μπορεί να αναπτυχθεί, να δοκιμαστεί, και να αναβαθμιστεί ξεχωριστά [2]. Τα δοχεία διευκολύνουν αυτή την προσέγγιση, καθώς κάθε μικροϋπηρεσία μπορεί να τρέχει μέσα σε ένα δικό της απομονωμένο περιβάλλον.

Τέλος, η χρήση δοχείων προσφέρει σημαντικά οφέλη στη συνεχή ανάπτυξη και παράδοση λογισμικού (CI/CD). Με τη δυνατότητα γρήγορης δημιουργίας, δοκιμής και ανάπτυξης δοχείων (και άρα

δοχειοποιημένων συστατικών ενός λογισμικού), οι ομάδες ανάπτυξης μπορούν να επιταχύνουν τον κύκλο ζωής του λογισμικού και να εξασφαλίσουν υψηλότερη ποιότητα και σταθερότητα στις εκδόσεις τους. [1]

2.2 Ασφάλεια Δοχείων Docker

Η ασφάλεια των Docker δοχείων περιλαμβάνει την προστασία των εικόνων, των δοχείων και των δεδομένων εκτέλεσης τους. Τα βασικά θέματα ασφαλείας περιλαμβάνουν επιθέσεις μέσω κακόβουλου λογισμικού, ευάλωτα σημεία, "προβληματικές" εξαρτήσεις και την ορατότητα μη κρυπτογραφημένων μυστικών. [4], [5]

Επιθέσεις μέσω Κακόβουλου Λογισμικού

Τα δοχεία Docker είναι ευάλωτα σε επιθέσεις από κακόβουλο λογισμικό που μπορεί να εισαχθεί είτε κατά την κατασκευή της εικόνας είτε κατά την εκτέλεσή της. Ένα κακόβουλο λογισμικό μπορεί να εκμεταλλευτεί τα δοχεία για διάφορες επιθέσεις, όπως είναι η κλοπή δεδομένων, η καταστροφή δεδομένων, η εξάπλωση σε άλλα μέρη του συστήματος ή του δικτύου, και η δημιουργία κερκόπορτων (backdoors). Είναι κρίσιμης σημασίας να εντοπίζονται και να αφαιρούνται τέτοιες απειλές για να διατηρείται η ακεραιότητα και η ασφάλεια των εφαρμογών και των δεδομένων τους.

Εξαρτήσεις

Οι εξαρτήσεις που χρησιμοποιούνται στις εικόνες Docker μπορούν να περιέχουν γνωστά τρωτά σημεία, τα οποία μπορούν να εκμεταλλευτούν οι επιτιθέμενοι. Αυτές οι ευπάθειες μπορούν να προκύψουν από παρωχημένα πακέτα, μη ασφαλείς βιβλιοθήκες. Η μη έγκαιρη ενημέρωση των εξαρτήσεων μπορεί να αφήσει τα δοχεία εκτεθειμένα σε επιθέσεις που θα μπορούσαν να αποφευχθούν. Συνεπώς, η συνεχής παρακολούθηση και ενημέρωση των εξαρτήσεων είναι απαραίτητη για τη διατήρηση της ασφάλειας.

Αποκάλυψη Μη Κρυπτογραφημένων Μυστικών

Μυστικά, όπως κωδικοί πρόσβασης, κλειδιά API και άλλες ευαίσθητες πληροφορίες, μπορούν να αποθηκευτούν σε μη κρυπτογραφημένη μορφή ακούσια μέσα στις εικόνες Docker ή στα αρχεία διαμόρφωσης. Η αποκάλυψη αυτών των πληροφοριών μπορεί να επιτρέψει σε επιτιθέμενους να αποκτήσουν πρόσβαση σε συστήματα και υπηρεσίες, θέτοντας σε κίνδυνο ολόκληρες υποδομές. Η ασφαλής διαχείριση και αποθήκευση των μυστικών είναι απαραίτητη για την προστασία από τέτοιες διαρροές.

Ευάλωτα Σημεία

Τα ευάλωτα σημεία περιλαμβάνουν οποιοδήποτε αδύναμο σημείο σε ένα σύστημα που μπορεί να εκμεταλλευτεί ένας επιτιθέμενος για να προκαλέσει βλάβη ή να αποκτήσει μη εξουσιοδοτημένη πρόσβαση. Στα Docker δοχεία, αυτά τα σημεία μπορεί να περιλαμβάνουν μη ασφαλείς ρυθμίσεις δικτύου, αδύναμα σημεία στον κώδικα των εφαρμογών, καθώς και κακόβουλες ρυθμίσεις στο σύστημα. Είναι ζωτικής σημασίας να εντοπίζονται και να διορθώνονται τέτοιες ευπάθειες για την προστασία των συστημάτων και των δεδομένων από επιθέσεις.

Αυξημένα δικαιώματα

Τα δοχεία Docker μπορούν να προσφέρουν πρόσβαση σε πόρους του συστήματος που δεν θα έπρεπε να είναι διαθέσιμοι, ειδικά όταν εκτελούνται με αυξημένα δικαιώματα (privileged mode). Αυτή η κατάσταση μπορεί να επιτρέψει σε επιτιθέμενους να εκτελούν κακόβουλες ενέργειες με προνόμια διαχειριστή, οδηγώντας σε πλήρη παραβίαση του συστήματος. Είναι σημαντικό να περιορίζεται η πρόσβαση και να εφαρμόζονται αρχές ελάχιστων προνομίων για τη διατήρηση της ασφάλειας.

Η κατανόηση και η αντιμετώπιση αυτών των προβλημάτων είναι απαραίτητη για την ασφαλή διαχείριση και λειτουργία των Docker δοχείων, διασφαλίζοντας την ακεραιότητα και την ασφάλεια των εφαρμογών και των δεδομένων που φιλοξενούνται σε αυτά.

2.3 Έλεγχοι Ασφάλειας

Οι έλεγχοι ασφάλειας στα δοχεία Docker περιλαμβάνουν την ανίχνευση κακόβουλου λογισμικού, την ανάλυση των εξαρτήσεων, την ανίχνευση τρωτοτήτων και τον εντοπισμό μυστικών. Αναλύονται και συγκρίνονται διάφορα εργαλεία για κάθε τύπο ελέγχου, και επιλέγονται τα καταλληλότερα για ενσωμάτωση στο CLI εργαλείο.

Για να δικαιολογήσουμε την επιλογή των εργαλείων που ενσωματώθηκαν στο CLI εργαλείο για τον έλεγχο ασφάλειας των Docker δοχείων, παρακάτω παρατίθεται μια σύγκριση και ανάλυση διαφόρων διαθέσιμων εργαλείων ανά τύπο ελέγχου. Αυτή η ανάλυση βοηθά στην κατανόηση των πλεονεκτημάτων και των μειονεκτημάτων κάθε εργαλείου και εξηγεί γιατί επιλέχθηκαν συγκεκριμένα εργαλεία για την ανάπτυξη του συστήματος.

2.3.1 Ανίχνευση Κακόβουλου Λογισμικού

2.3.1.1 ClamAV

- **Πλεονεκτήματα**: Δωρεάν και ανοιχτού κώδικα, ευρέως διαδεδομένο, υποστηρίζει πολλές μορφοποιήσεις αρχείων. Είναι ένα από τα πιο διαδεδομένα εργαλεία για την ανίχνευση ιών και κακόβουλου λογισμικού σε Linux περιβάλλοντα.
- **Μειονεκτήματα**: Μπορεί να έχει χαμηλότερη ακρίβεια σε σχέση με εμπορικά εργαλεία. Απαιτεί συχνές ενημερώσεις της βάσης δεδομένων του για να παραμένει αποτελεσματικό.
- **Επιλογή**: Ενσωματώθηκε λόγω της αξιοπιστίας του και της δυνατότητας ανίχνευσης κακόβουλου λογισμικού σε συστήματα Linux (τα οποία χρησιμοποιούνται συχνά για servers και κρίσιμες υποδομές) και Docker. Η ευρεία χρήση και υποστήριξή του από την κοινότητα το καθιστούν αξιόπιστη επιλογή.

2.3.1.2 Maldet (Linux Malware Detect)

- **Πλεονεκτήματα**: Ειδικά σχεδιασμένο για συστήματα Linux, εύκολο στη χρήση, καλύπτει κοινές απειλές και κακόβουλο λογισμικό.
- **Μειονεκτήματα**: Περιορισμένη βάση δεδομένων σε σύγκριση με άλλα εμπορικά εργαλεία. Εστιάζει κυρίως σε γνωστές απειλές και ίσως να μην καλύπτει πιο εξειδικευμένα ή νέα κακόβουλα λογισμικά.
- **Επιλογή**: Ενσωματώθηκε λόγω της εστίασης σε Linux και της δυνατότητας να ανιχνεύει κακόβουλο λογισμικό σε περιβάλλοντα Docker. Η χρήση του σε συνδυασμό με το ClamAV παρέχει ένα ολοκληρωμένο σύστημα ανίχνευσης.

2.3.1.3 Sophos Antivirus

- **Πλεονεκτήματα**: Ισχυρή προστασία από κακόβουλο λογισμικό, ευρεία υποστήριξη από την κοινότητα και την εταιρεία. Περιλαμβάνει δυνατότητες ανίχνευσης σε πραγματικό χρόνο.
- **Μειονεκτήματα**: Εμπορικό εργαλείο, απαιτεί συνδρομή. Μπορεί να είναι πιο βαρύ σε πόρους σε σύγκριση με άλλα εργαλεία.
- **Επιλογή**: Παρά την ισχυρή προστασία του, δεν επιλέχθηκε λόγω του κόστους και της ανάγκης για συνδρομή. Προτιμήθηκαν εργαλεία ανοιχτού κώδικα, όπως το ClamAV και το Maldet, που προσφέρουν καλό επίπεδο προστασίας χωρίς κάποιο κόστος.

2.3.1.4 Bitdefender Antivirus

- **Πλεονεκτήματα**: Ισχυρή προστασία από κακόβουλο λογισμικό, ευρεία υποστήριξη και συνεχείς ενημερώσεις. Αναγνωρίζεται για την υψηλή ακρίβεια ανίχνευσης.
- **Μειονεκτήματα**: Εμπορικό εργαλείο, απαιτεί συνδρομή. Μπορεί να είναι πιο βαρύ σε πόρους και να απαιτεί περισσότερο χρόνο για τη σάρωση.
- **Επιλογή**: Παρά την αποτελεσματικότητα του, δεν επιλέχθηκε λόγω του κόστους και της ανάγκης για συνδρομή. Προτιμήθηκαν εργαλεία ανοιχτού κώδικα για την ενσωμάτωση στο CLI εργαλείο.

2.3.1.5 F-Prot Antivirus

- **Πλεονεκτήματα**: Εξαιρετική ανίχνευση κακόβουλου λογισμικού, ευρεία υποστήριξη για πολλαπλά λειτουργικά συστήματα, τακτικές ενημερώσεις.
- **Μειονεκτήματα**: Εμπορικό εργαλείο, απαιτεί συνδρομή. Μπορεί να είναι πιο ακριβό από άλλα εργαλεία.
- **Επιλογή**: Παρόλο που είναι ισχυρό εργαλείο, προτιμήθηκαν τα ClamAV και Maldet λόγω της ευκολίας στη χρήση, της ευρείας υποστήριξης και της ανοιχτού κώδικα φύσης τους.

2.3.2 Ανάλυση Εξαρτήσεων

2.3.2.1 OWASP Dependency-Check

- **Πλεονεκτήματα**: Ειδικά σχεδιασμένο για ανάλυση εξαρτήσεων, μεγάλη βάση δεδομένων ευπαθειών, συνεχείς ενημερώσεις από την κοινότητα OWASP. Ανιχνεύει γνωστές ευπάθειες σε βιβλιοθήκες ανοιχτού κώδικα.
- **Μειονεκτήματα**: Μπορεί να είναι αργό κατά την ανάλυση εξαρτήσεων σε μεγάλες εικόνες Docker. Η βάση δεδομένων του μπορεί να μην περιλαμβάνει όλες τις τελευταίες ευπάθειες χωρίς συχνές ενημερώσεις.
- **Επιλογή**: Ενσωματώθηκε λόγω της εξειδίκευσής του στην ανάλυση εξαρτήσεων και της υποστήριξής του από την κοινότητα OWASP. Η δυνατότητα του να εντοπίζει ευπάθειες σε εξαρτήσεις το καθιστά κρίσιμο εργαλείο για την ασφάλεια.

2.3.2.2 Snyk

- **Πλεονεκτήματα**: Γρήγορο, ευρεία βάση δεδομένων ευπαθειών, ενσωμάτωση με CI/CD pipelines, δυνατότητα διόρθωσης ευπαθειών με pull requests.
- **Μειονεκτήματα**: Περιορισμένη δωρεάν έκδοση, απαιτεί συνδρομή για πλήρη λειτουργικότητα. Μπορεί να παράγει ψευδείς θετικές ανιχνεύσεις.
- **Επιλογή**: Παρόλο που είναι ισχυρό εργαλείο, προτιμήθηκε το OWASP Dependency-Check και το Dockle λόγω της εστίασης στην ανοιχτού κώδικα κοινότητα και της πλήρους δωρεάν λειτουργικότητας τους.

2.3.2.3 WhiteSource Bolt

- **Πλεονεκτήματα**: Εύκολη ενσωμάτωση σε CI/CD pipelines, μεγάλη βάση δεδομένων ευπαθειών, παρέχει λεπτομερείς αναφορές και προτάσεις διόρθωσης.
- **Μειονεκτήματα**: Εμπορικό εργαλείο με περιορισμένη δωρεάν έκδοση. Απαιτεί συνδρομή για πλήρη λειτουργικότητα και μπορεί να είναι πιο πολύπλοκο στη χρήση.
- **Επιλογή**: Παρόλο που είναι ισχυρό εργαλείο, προτιμήθηκε το OWASP Dependency-Check και το Dockle λόγω της εστίασης στην ανοιχτού κώδικα κοινότητα και της πλήρους δωρεάν λειτουργικότητας τους.

2.3.2.4 Sonatype Nexus IQ

- **Πλεονεκτήματα**: Ισχυρό εργαλείο για ανάλυση εξαρτήσεων, ευρεία βάση δεδομένων ευπαθειών, εύκολη ενσωμάτωση σε CI/CD pipelines.
- **Μειονεκτήματα**: Εμπορικό εργαλείο, απαιτεί συνδρομή για πλήρη λειτουργικότητα. Μπορεί να είναι πιο ακριβό από άλλα εργαλεία.
- **Επιλογή**: Παρόλο που είναι ισχυρό εργαλείο, προτιμήθηκε το OWASP Dependency-Check και το Dockle λόγω της ευκολίας στη χρήση, της πλήρους δωρεάν λειτουργικότητας τους και της εστίασης στην ανοιχτού κώδικα κοινότητα.

2.3.3 Ανίχνευση Τρωτοτήτων

2.3.3.1 Trivy

- **Πλεονεκτήματα**: Γρήγορο και εύκολο στη χρήση, ενσωματώνει πολλαπλές βάσεις δεδομένων ευπαθειών, ανοιχτού κώδικα, καλή ενσωμάτωση με CI/CD αγωγούς.
- **Μειονεκτήματα**: Μπορεί να παράγει ψευδείς θετικές ανιχνεύσεις. Απαιτεί συχνές ενημερώσεις για να παραμένει αποτελεσματικό.
- **Επιλογή**: Ενσωματώθηκε για την ταχύτητα και την ευκολία χρήσης του, καθώς και για την δυνατότητα ανίχνευσης ευπαθειών σε Docker εικόνες και πακέτα.

2.3.3.2 Dockle

- **Πλεονεκτήματα**: Ελέγχει τις βέλτιστες πρακτικές για Docker εικόνες, ανιχνεύει ευπάθειες και προβλήματα διαμόρφωσης, υποστηρίζει πολιτικές ασφάλειας και συμμόρφωσης.
- **Μειονεκτήματα**: Περιορισμένη ανάλυση. Επικεντρώνεται περισσότερο στη διαμόρφωση.
- **Επιλογή**: Ενσωματώθηκε για έλεγχο βέλτιστων πρακτικών και διαμόρφωσης. Η συνδυαστική χρήση του με τα υπόλοιπα εργαλεία παρέχει μια πιο ολοκληρωμένη προσέγγιση.

2.3.3.3 Grype

- **Πλεονεκτήματα**: Ενσωματώνει μεγάλη βάση δεδομένων ευπαθειών, παρέχει λεπτομερείς αναφορές, καλή ενσωμάτωση με CI/CD αγωγούς. Ανοιχτού κώδικα.
- **Μειονεκτήματα**: Μπορεί να είναι πιο αργό κατά την ανίχνευση ευπαθειών σε μεγάλες εικόνες Docker. Απαιτεί συχνές ενημερώσεις για να παραμένει αποτελεσματικό.
- **Επιλογή**: Ενσωματώθηκε για τη λεπτομερή ανάλυση και τις αναφορές του. Η συνδυαστική χρήση του με το Trivy εξασφαλίζει την πλήρη κάλυψη των ευπαθειών.

2.3.3.4 Anchore Engine

- **Πλεονεκτήματα**: Ισχυρό εργαλείο για ανίχνευση ευπαθειών και (λανθασμένων/αδύναμων) διαμορφώσεων, καλή ενσωμάτωση με CI/CD pipelines, μεγάλη βάση δεδομένων ευπαθειών.
- **Μειονεκτήματα**: Περιορισμένη δωρεάν έκδοση, απαιτεί συνδρομή για πλήρη λειτουργικότητα. Μπορεί να είναι πιο πολύπλοκο στη ρύθμιση και χρήση σε σύγκριση με άλλα εργαλεία.
- **Επιλογή**: Παρόλο που είναι ισχυρό εργαλείο, προτιμήθηκαν τα Trivy και Grype λόγω της ευκολίας στη χρήση τους και της πλήρους δωρεάν λειτουργικότητας τους.

2.3.3.5 Aqua Microscanner

- **Πλεονεκτήματα**: Γρήγορο, ενσωματώνεται εύκολα σε CI/CD pipelines, στηρίζεται σε μεγάλη βάση δεδομένων ευπαθειών. Παρέχει αναλυτικές αναφορές και προτάσεις διόρθωσης.
- **Μειονεκτήματα**: Περιορισμένη δωρεάν έκδοση, απαιτεί συνδρομή για πλήρη λειτουργικότητα. Μπορεί να απαιτεί περισσότερους πόρους σε σύγκριση με άλλα εργαλεία.
- **Επιλογή**: Παρόλο που είναι ισχυρό εργαλείο, προτιμήθηκαν τα Trivy και Grype λόγω της ευκολίας στη χρήση τους και της πλήρους δωρεάν λειτουργικότητας τους.

2.3.3.6 SonarQube

- **Πλεονεκτήματα**: Εξαιρετική ανάλυση τρωτοτήτων και ποιότητας κώδικα, ενσωματώνεται εύκολα σε CI/CD pipelines, μεγάλη κοινότητα υποστήριξης.
- **Μειονεκτήματα**: Μπορεί να είναι πιο βαρύ σε πόρους σε σύγκριση με άλλα εργαλεία, απαιτεί συνδρομή για πλήρη λειτουργικότητα.
- **Επιλογή**: Παρά την εξαιρετική ανάλυση ποιότητας κώδικα, προτιμήθηκαν τα Trivy και Grype για την εστίασή τους στην ανίχνευση ευπαθειών και την ευκολία χρήσης τους σε Docker περιβάλλοντα.

2.3.4 Εντοπισμός Μυστικών

2.3.4.1 Gitleaks

- **Πλεονεκτήματα**: Ανιχνεύει γνωστά μοτίβα μυστικών, ανοιχτού κώδικα, εύκολο στη χρήση και ρύθμιση. Χρησιμοποιείται ευρέως για τον έλεγχο αποθετηρίων κώδικα.
- **Μειονεκτήματα**: Μπορεί να παράγει ψευδείς θετικές ανιχνεύσεις. Απαιτεί συνεχή ενημέρωση για να αναγνωρίζει νέα μοτίβα μυστικών.
- **Επιλογή**: Ενσωματώθηκε λόγω της εξειδίκευσής του στον εντοπισμό γνωστών μοτίβων μυστικών. Η χρήση του βοηθά στον εντοπισμό και την αντιμετώπιση των εκτεθειμένων μυστικών.

2.3.4.2 TruffleHog

- **Πλεονεκτήματα**: Ανιχνεύει συμβολοσειρές υψηλής εντροπίας που μπορεί να είναι μυστικά, εντοπίζει μυστικά που δεν ακολουθούν τα κοινά μοτίβα.
- **Μειονεκτήματα**: Μπορεί να χρειάζεται χειροκίνητη επιβεβαίωση των αποτελεσμάτων. Η διαδικασία ανίχνευσης μπορεί να είναι πιο αργή σε μεγάλα αποθετήρια κώδικα.
- **Επιλογή**: Ενσωματώθηκε για την δυνατότητα εντοπισμού υψηλής εντροπίας συμβολοσειρών και την συμπληρωματική ανάλυση με το Gitleaks. Η χρήση του εξασφαλίζει ότι ακόμη και τα λιγότερο προφανή μυστικά εντοπίζονται.

2.3.4.3 Shhgit

- **Πλεονεκτήματα**: Εντοπίζει εκτεθειμένα μυστικά σε πραγματικό χρόνο, ιδιαίτερα χρήσιμο για την παρακολούθηση αποθετηρίων κώδικα σε πλατφόρμες, όπως το GitHub.
- **Μειονεκτήματα**: Περιορισμένη χρήση για σάρωση στατικών εικόνων Docker. Εστίαση σε αποθετήρια κώδικα και όχι σε Docker περιβάλλοντα.
- **Επιλογή**: Παρά την ισχυρή ανίχνευση σε πραγματικό χρόνο, δεν επιλέχθηκε λόγω της εστίασής του σε αποθετήρια κώδικα. Προτιμήθηκαν εργαλεία που μπορούν να εντοπίσουν μυστικά σε Docker εικόνες και αρχεία διαμόρφωσης.

2.3.4.4 Detect Secrets

- **Πλεονεκτήματα**: Ανοιχτού κώδικα, ανιχνεύει γνωστά μοτίβα μυστικών, εύκολο στη χρήση και ενσωμάτωση σε CI/CD pipelines.
- **Μειονεκτήματα**: Μπορεί να παράγει ψευδείς θετικές ανιχνεύσεις. Απαιτεί συνεχή ενημέρωση για να αναγνωρίζει νέα μοτίβα μυστικών.
- **Επιλογή**: Παρόλο που είναι καλό εργαλείο, προτιμήθηκαν τα Gitleaks και TruffleHog λόγω της πληρέστερης κάλυψης και της εξειδίκευσής τους στον εντοπισμό μυστικών.

2.3.4.5 Gitrob

- **Πλεονεκτήματα**: Εντοπίζει ευαίσθητα αρχεία και μυστικά σε αποθετήρια Git, ιδιαίτερα χρήσιμο για τον έλεγχο αποθετηρίων κώδικα σε πλατφόρμες, όπως το GitHub.
- **Μειονεκτήματα**: Περιορισμένη χρήση για σάρωση στατικών εικόνων Docker. Εστιάζεται κυρίως σε αποθετήρια κώδικα και όχι σε Docker περιβάλλοντα.
- **Επιλογή**: Παρόλο που είναι καλό εργαλείο, δεν επιλέχθηκε λόγω της εστίασής του σε αποθετήρια κώδικα. Προτιμήθηκαν εργαλεία που μπορούν να εντοπίσουν μυστικά σε Docker εικόνες και αρχεία διαμόρφωσης.

Με βάση την παραπάνω αναλυτική σύγκριση, επιλέχθηκαν τα εργαλεία ClamAV, Maldet, OWASP Dependency-Check, Dockle, Trivy, Grype, Gitleaks και TruffleHog για ενσωμάτωση στο CLI εργαλείο. Αυτή η επιλογή καλύπτει πλήρως τις ανάγκες ανίχνευσης κακόβουλου λογισμικού, ανάλυσης εξαρτήσεων, ανίχνευσης τρωτοτήτων και εντοπισμού μη κρυπτογραφημένων μυστικών, εξασφαλίζοντας την ασφάλεια και την αξιοπιστία των Docker δοχείων.

3

Ανάπτυξη

Η ανάπτυξη του συστήματος λογισμικού ακολούθησε το μοντέλο του καταρράκτη, όπου οι δραστηριότητες ανάπτυξης εκτελούνται με σειριακό τρόπο, η μία μετά την άλλη. Στο υπόλοιπο του κεφαλαίου αυτού θα αναλύσουμε τις ενέργειες που έγιναν σε κάθε δραστηριότητα και τα κύρια αποτελέσματά τους σε ξεχωριστές ενότητες. Η τελευταία ενότητα του κεφαλαίου αποσκοπεί στο να παραθέσει και να αιτιολογήσει τον βαθμό ικανοποίησης των απαιτήσεων που είχαν τεθεί για το σύστημα.

3.1 Ανάλυση Απαιτήσεων

Παραθέτουμε παρακάτω τις λειτουργικές και μη λειτουργικές απαιτήσεις που τέθηκαν για το εργαλείο προς ανάπτυξη.

3.1.1 Λειτουργικές Απαιτήσεις

- **Σάρωση εικόνας Docker**: Δυνατότητα σάρωσης μιας εικόνας Docker για τρωτότητες, κακόβουλο λογισμικό, εξαρτήσεις και μυστικά.
    - **Ανίχνευση Τρωτοτήτων (Vulnerability Scanning)**

      Περιγραφή: Η ανίχνευση τρωτοτήτων αφορά τον εντοπισμό γνωστών ευπαθειών σε εικόνες Docker. Αυτές οι ευπάθειες μπορεί να περιλαμβάνουν αδύναμες ρυθμίσεις ασφαλείας, μη ενημερωμένα λογισμικά και βιβλιοθήκες που περιέχουν γνωστά προβλήματα ασφαλείας.

Τι περιλαμβάνει: Περιλαμβάνει την ανάλυση του λειτουργικού συστήματος της εικόνας, των πακέτων λογισμικού που περιέχονται και των βιβλιοθηκών που χρησιμοποιούνται για να εντοπιστούν γνωστές ευπάθειες. Οι σάρωσες αυτές ενημερώνονται συνεχώς με τις τελευταίες πληροφορίες για ευπάθειες από βάσεις δεδομένων, όπως το NVD "https://nvd.nist.gov/" (National Vulnerability Database).

Έξοδος: Το αποτέλεσμα της σάρωσης θα περιλαμβάνει μια αναλυτική αναφορά με τις ευπάθειες που βρέθηκαν, το επίπεδο κινδύνου τους (π.χ. χαμηλό, μεσαίο, υψηλό), τις βιβλιοθήκες ή τα πακέτα στα οποία εντοπίστηκαν οι ευπάθειες και πιθανές προτάσεις για τη διόρθωσή τους.

- **Ανίχνευση Κακόβουλου Λογισμικού (Malware Scanning)**

  Περιγραφή: Η ανίχνευση κακόβουλου λογισμικού έχει ως στόχο να εντοπίσει κακόβουλο λογισμικό που μπορεί να έχει εισαχθεί στην εικόνα Docker είτε κατά την κατασκευή της είτε κατά την εκτέλεσή της.

  Τι περιλαμβάνει: Περιλαμβάνει τη σάρωση του συστήματος αρχείων της εικόνας για την ανίχνευση γνωστών ιών, Δούρειων Ίππων (trojans), σκουληκιών (worms)) και άλλων ειδών κακόβουλου λογισμικού. Αυτό μπορεί να περιλαμβάνει ανάλυση αρχείων και διεργασιών για γνωστά μοτίβα και υπογραφές κακόβουλου λογισμικού.

  Έξοδος: Το αποτέλεσμα της σάρωσης θα περιλαμβάνει μια αναφορά με τα αρχεία ή τις διεργασίες που βρέθηκαν να περιέχουν κακόβουλο λογισμικό, το είδος του κακόβουλου λογισμικού και προτάσεις για την απομάκρυνσή του.

- **Ανάλυση Εξαρτήσεων (Dependency Analysis)**

  Περιγραφή: Η ανάλυση εξαρτήσεων αφορά τον εντοπισμό προβλημάτων και αδυναμιών στις εξαρτήσεις του κώδικα που χρησιμοποιείται στην εικόνα Docker. Αυτές οι εξαρτήσεις μπορεί να περιλαμβάνουν βιβλιοθήκες, πακέτα λογισμικού και άλλα στοιχεία που απαιτούνται για την εκτέλεση της εφαρμογής. Η ανάλυση εξαρτήσεων εστιάζει πιο συγκεκριμένα στην ολοκληρωμένη λειτουργικότητα των πακέτων, δηλαδή εάν χρησιμοποιούνται σωστές εκδόσεις που είναι συμβατές μεταξύ τους. Από την άλλη, η ανίχνευση τρωτοτήτων αφορά τον εντοπισμό ασφαλείας αυτών των βιβλιοθηκών, δηλαδή αν υπάρχουν γνωστά κενά ή αδυναμίες που μπορούν να εκμεταλλευτούν οι επιτιθέμενοι

  Τι περιλαμβάνει: Περιλαμβάνει την εξέταση όλων των εξαρτήσεων για γνωστές ευπάθειες, μη ενημερωμένα πακέτα και άλλες αδυναμίες που μπορεί να θέτουν σε κίνδυνο την ασφάλεια της εικόνας.

  Έξοδος: Το αποτέλεσμα της ανάλυσης θα περιλαμβάνει μια αναφορά με τις ευπάθειες που βρέθηκαν στις εξαρτήσεις, το επίπεδο κινδύνου τους, τις βιβλιοθήκες ή τα πακέτα στα οποία εντοπίστηκαν τα προβλήματα και προτάσεις για την ενημέρωση ή την αντικατάστασή τους.

- **Εντοπισμός Μυστικών (Secret Detection)**

  Περιγραφή: Ο εντοπισμός μη κρυπτογραφημένων μυστικών έχει ως στόχο να ανακαλύψει ευαίσθητες πληροφορίες, όπως κωδικούς πρόσβασης, κλειδιά API και άλλα μυστικά, που

μπορεί να έχουν αποθηκευτεί κατά λάθος σε μη κρυπτογραφημένη μορφή μέσα στην εικόνα Docker.

Τι περιλαμβάνει: Περιλαμβάνει τη σάρωση του συστήματος αρχείων και των αρχείων διαμόρφωσης για γνωστά μοτίβα μυστικών, καθώς και τον εντοπισμό υψηλής εντροπίας συμβολοσειρών που μπορεί να υποδεικνύουν την παρουσία μυστικών.

Έξοδος: Το αποτέλεσμα της σάρωσης θα περιλαμβάνει μια αναφορά με τα μυστικά που βρέθηκαν, τα αρχεία ή τις τοποθεσίες όπου εντοπίστηκαν και προτάσεις για την ασφαλή αποθήκευση και διαχείρισή τους.

Σημειώνουμε εδώ πως σε όλα τα είδη ελέγχων ασφάλειας που προδιαγράφηκαν προηγουμένως, η σάρωση θα γίνεται αποκλειστικά σε μια εικόνα αντί για το δοχείο που προκύπτει από αυτήν, επειδή τα δοχεία μπορεί να αλλάζουν συχνά και να έχουν μεγάλο όγκο δεδομένων. Επομένως, αν υπάρχει η ανάγκη σάρωσης ενός δοχείου, αυτή μπορεί να καλυφθεί με τον εξής τρόπο: για να "παγώσουν" οι τρέχουσες αλλαγές στο δοχείο, ο χρήστης μπορεί να χρησιμοποιήσει την εντολή "docker commit <container-name> <image-name:tag>" και να αποθηκεύσει το δοχείο σε μια εικόνα προς μετέπειτα σάρωση (από το εργαλείο προς ανάπτυξη).

- **Δυνατότητα επιλογής ελέγχων**: Το εργαλείο πρέπει να επιτρέπει στον χρήστη να επιλέγει την χρήση ενός ή παραπάνω ελέγχων από αυτούς που υποστηρίζονται από το εργαλείο, όπως ανίχνευση τρωτοτήτων, ανίχνευση κακόβουλου λογισμικού, έλεγχος εξαρτήσεων και εντοπισμός μυστικών.

Η επιλογή των ελέγχων διαμορφώνεται μέσω οδηγιών στην γραμμή εντολών. Ειδικότερα, ο χρήστης μπορεί να χρησιμοποιήσει κατάλληλα flags για να καθορίσει τους τύπους των σαρώσεων που επιθυμεί να εκτελεστούν.

- **Παραμετροποίηση εξόδου**: Το εργαλείο θα παράγει μια ενιαία αναφορά των ευρημάτων, η οποία θα περιλαμβάνει τα αποτελέσματα από όλα τα εργαλεία που χρησιμοποιούνται για τις σαρώσεις. Επιπλέον, θα πρέπει να προσφέρει τη δυνατότητα στον χρήστη να ορίζει αν τα αποτελέσματα θα πρέπει να αποθηκευθούν σε κάποιο αρχείο στο τοπικό σύστημα φιλοξενίας. Εφόσον δοθεί το flag αυτό, τότε θα πρέπει να υποστηριχθούν δύο εναλλακτικές περιπτώσεις: (α) παρέχεται επιπλέον flag από τον χρήστη που προσδιορίζει την ακριβή τοποθεσία και όνομα του αρχείου αποτελεσμάτων προς αποθήκευση και (β) δεν παρέχεται επιπλέον flag οπότε το εργαλείο θα πρέπει να υποστηρίζει μια προκαθορισμένη τοποθεσία και όνομα για το αρχείο εξόδου. Μάλιστα, το προκαθορισμένο μοτίβο του ονόματος θα πρέπει να είναι τέτοιο ώστε να αποφεύγονται συγκρούσεις ονομάτων (μεταξύ των αρχείων εξόδου).

- **Αυθεντικοποίηση του χρήστη**: Σε περίπτωση που ο χρήστης χρησιμοποιεί κάποιο αποθετήριο εικόνων (δοχείων), θα πρέπει να έχει την δυνατότητα να παρέχει τα διαπιστευτήριά του ώστε να επιτρέπει στο εργαλείο να προχωρήσει σε αυθεντικοποίηση εκ μέρους του και εκφόρτωση της επιθυμητής εικόνας. Το εργαλείο θα πρέπει να υποστηρίζει την αυθεντικοποίηση χρήστη τόσο για το Docker Hub όσο και για οποιοδήποτε ιδιωτικό αποθετήριο (private registry).

3.1.2 Μη Λειτουργικές Απαιτήσεις

- **Ευκολία Ενσωμάτωσης**: Το εργαλείο αυτό θα πρέπει να υλοποιηθεί με τη μορφή μιας Docker εικόνας ώστε να μπορεί να εκτελεστεί σε οποιαδήποτε πλατφόρμα. Με αυτό τον τρόπο, θα ήταν και εύκολη η ενσωμάτωσή του με περιβάλλοντα CI/CD, τα οποία υποστηρίζουν την εκτέλεση βημάτων αγωγών CI/CD με τη μορφή δοχείων. Αυτό επιτρέπει την αυτόματη εκτέλεση των ελέγχων ασφάλειας σε κάθε φορά που τροποποιείται ο πηγαίος κώδικας του λογισμικού, εξασφαλίζοντας ότι οι εικόνες Docker που χρησιμοποιούνται (και συνήθως αντιστοιχούν σε συστατικά του λογισμικού) είναι ασφαλείς.

- **Ευκολία χρήσης**: Η διεπαφή χρήστη πρέπει να είναι απλή και άμεσα κατανοητή, ακόμα και για χρήστες που δεν έχουν προηγούμενη εμπειρία με εργαλεία CLI.

- **Αναλυτικές αναφορές**: Τα αποτελέσματα από όλες τις σαρώσεις θα πρέπει να παρέχουν αναλυτικές πληροφορίες και να είναι μορφοποιημένα με φιλικό προς τον χρήστη τρόπο. Οι αναφορές θα περιλαμβάνουν λεπτομέρειες για κάθε εύρημα, όπως την περιγραφή του προβλήματος, το αρχείο ή το μέρος της εικόνας Docker που εντοπίστηκε, και τη σοβαρότητα του ευρήματος. Ο κώδικας θα συλλέγει τα αποτελέσματα από τα εργαλεία σάρωσης, θα τα μορφοποιεί και θα τα αποθηκεύει σε ένα ενιαίο αρχείο αναφοράς.

- **Αξιοπιστία και σταθερότητα**: Το εργαλείο θα πρέπει να δοκιμαστεί εκτενώς ώστε να εγγυηθεί η αξιοπιστία και η σταθερότητά του κατά τη χρήση. Επίσης, το εργαλείο θα πρέπει να παρέχει ωραία και κατάλληλα μηνύματα λάθους σε περίπτωση σφαλμάτων των χρηστών.

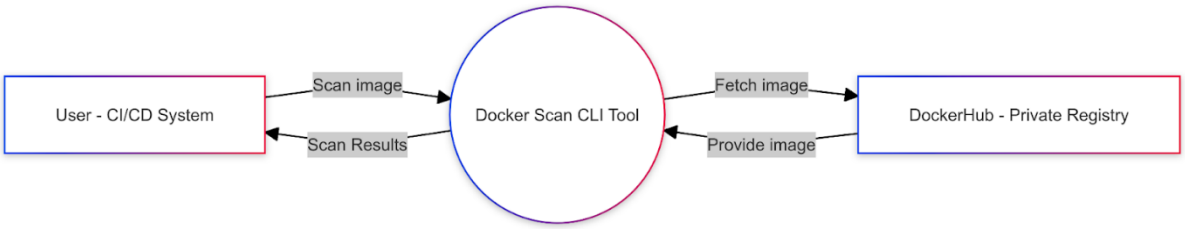
- **Απόδοση**: Το εργαλείο πρέπει να έχει καλή απόδοση και να ανταποκρίνεται άμεσα στα αιτήματα των χρηστών. Αυτό σημαίνει ότι οι σαρώσεις θα πρέπει να εκτελούνται σε εύλογο χρονικό διάστημα, ακόμα και για μεγάλες και πολύπλοκες εικόνες Docker.

- **Λογιστική Καταγραφή (Logging)**: Το εργαλείο πρέπει να υποστηρίζει την πλήρη καταγραφή όλων των πιο σημαντικών ενεργειών, των σφαλμάτων και των αποτελεσμάτων του σε αρχείο καταγραφής (log file), διασφαλίζοντας ότι κάθε βήμα της διαδικασίας σάρωσης είναι τεκμηριωμένο. Αυτό θα βοηθήσει στη διαγνωστική των προβλημάτων και στη βελτίωση της αξιοπιστίας του εργαλείου.

3.2 Σχεδίαση

Παρουσιάζονται τα σχεδιαστικά μοντέλα του εργαλείου που παράχθηκαν κατά τη δραστηριότητα της σχεδίασης του συστήματος:

3.2.1 Διάγραμμα Περιβάλλοντος

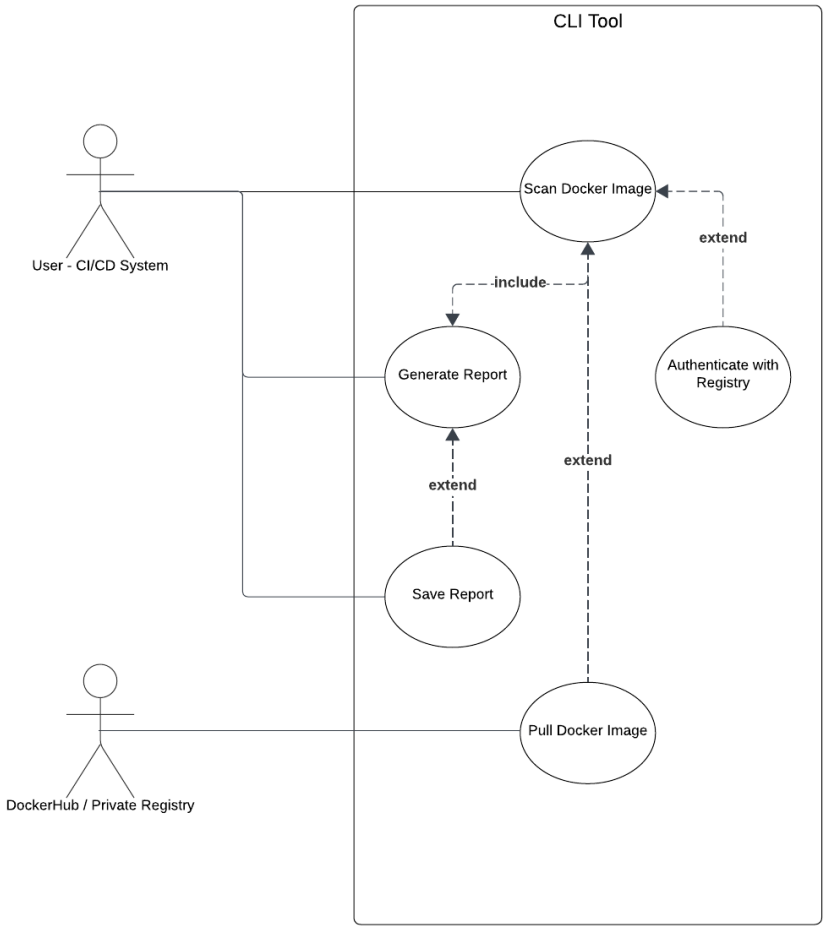


Εικόνα 1. Διάγραμμα περιβάλλοντος

Το διάγραμμα απεικονίζει την κυκλική ροή της διαδικασίας και τονίζει την αλληλεπίδραση μεταξύ χρήστη, εργαλείου και αποθετηρίου.

3.2.2 Διάγραμμα Περιπτώσεων Χρήσης

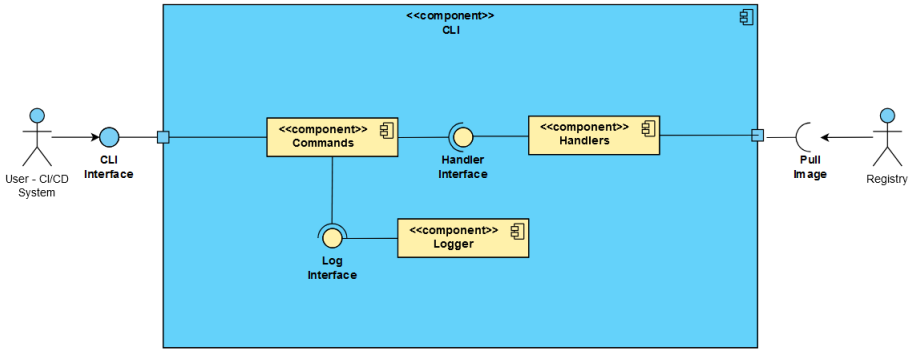
Το διάγραμμα περιπτώσεων χρήσης που παρουσιάζεται απεικονίζει τις αλληλεπιδράσεις μεταξύ χρηστών και του εργαλείου CLI κατά τη διάρκεια της χρήσης του.



Εικόνα 2. Διάγραμμα περιπτώσεων χρήσης

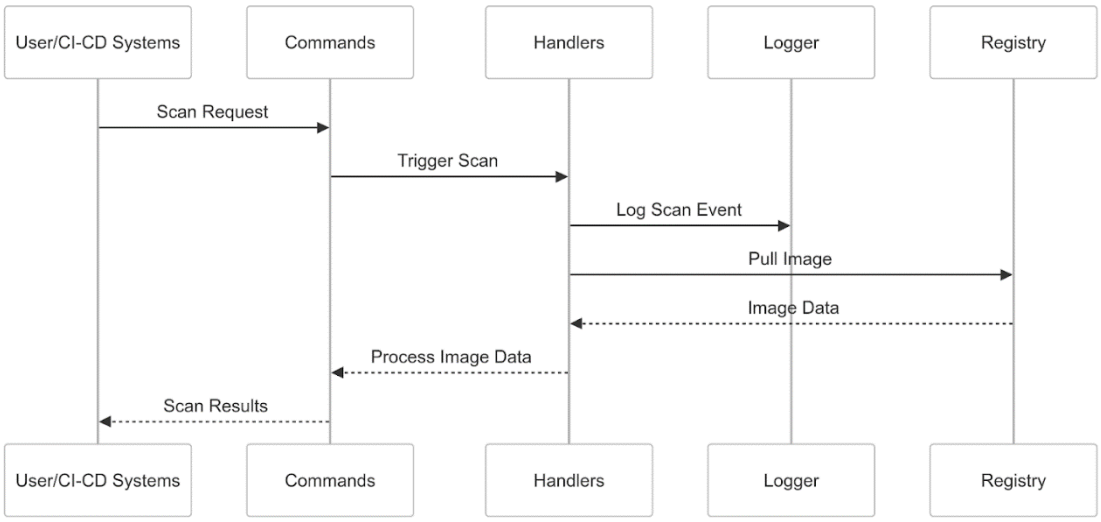
3.2.3 Διάγραμμα Αρχιτεκτονικής / Συστατικών

Το διάγραμμα συστατικών απεικονίζει τη δομή του εργαλείου CLI με τα κύρια τμήματα του συστήματος και τις αλληλεπιδράσεις τους.



Εικόνα 3. Διάγραμμα συστατικών

3.2.4 Διάγραμμα Ακολουθίας



Εικόνα 4. Διάγραμμα ακολουθίας

Το διάγραμμα ακολουθίας αντικατοπτρίζει τη ροή επικοινωνίας μεταξύ των στοιχείων στο σύστημά με βάση το διάγραμμα συστατικών, συμπεριλαμβανομένης της λήψης εικόνας από το αποθετήριο και των αλληλεπιδράσεων καταγραφής.

3.3 Υλοποίηση

3.3.1 Παραδοτέα / Τεχνήματα Υλοποίησης

Η υλοποίησή μας είναι διαθέσιμη ως εικόνα Docker στο Docker Hub, παρέχοντας έναν εύκολο και βολικό τρόπο για τη χρήση του εργαλείου χωρίς να χρειάζεται η εγκατάσταση των απαιτούμενων εξαρτήσεων τοπικά. Ο πηγαίος κώδικας του εργαλείου είναι επίσης διαθέσιμος στο GitHub.

Πηγαίος Κώδικας στο GitHub: https://github.com/DimitrisK01/sec-cli-tool.git

Αποθετήριο Εικόνας Docker στο Docker Hub: dkostoulas01/cli-sec-tool - ας χρησιμοποιήσετε την latest (τρέχουσα) έκδοση της εικόνας.

3.3.2 Αρχείο Κατασκευής Εικόνας Εργαλείου

Παρακάτω παρατίθεται το αρχείο Dockerfile που χρησιμοποιήθηκε για τη δημιουργία της εικόνας Docker του εργαλείου μας:

```
Dockerfile > FROM
FROM python:3.10-slim

WORKDIR /app

RUN apt-get update && apt-get install -y \
    curl \
    gnupg \
    apt-transport-https \
    lsb-release \
    software-properties-common \
    ca-certificates \
    docker.io \
    clamav \
    wget \
    tar \
    unzip \
    default-jdk

RUN apt-get update && apt-get install -y git

# Install Trivy
RUN wget -qO - https://aquasecurity.github.io/trivy-repo/deb/public.key | apt-key add - && \
    echo deb https://aquasecurity.github.io/trivy-repo/deb $(lsb_release -sc) main | tee -a /etc/apt/sources.list.d/trivy.list && \
    apt-get update && apt-get install -y trivy

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

# Install Node.js and npm
RUN curl -fsSL https://deb.nodesource.com/setup_16.x | bash - && \
    apt-get install -y nodejs

# Install Dockle
RUN wget https://github.com/goodwithtech/dockle/releases/download/v0.4.6/dockle_0.4.6_Linux-64bit.deb && \
    dpkg -i dockle_0.4.6_Linux-64bit.deb && \
    rm dockle_0.4.6_Linux-64bit.deb

# Install Gitleaks
RUN curl -sSfL https://github.com/gitleaks/gitleaks/releases/download/v8.18.4/gitleaks_8.18.4_linux_x64.tar.gz -o gitleaks.tar.gz && \
    tar -xzf gitleaks.tar.gz -C /usr/local/bin gitleaks && \
    chmod +x /usr/local/bin/gitleaks

# Install OWASP Dependency-Check with permission verification
RUN VERSION=$(curl -s https://jeremylong.github.io/DependencyCheck/current.txt) && \
    curl -Ls "https://github.com/jeremylong/DependencyCheck/releases/download/v$VERSION/dependency-check-$VERSION-release.zip" --output dependency-check.zip && \
    unzip dependency-check.zip -d /usr/local/bin/ && \
    rm dependency-check.zip && \
    chmod -R +x /usr/local/bin/dependency-check/bin && \
    ls -la /usr/local/bin/dependency-check/bin # This will list permissions in the build output

# Install Maldet
RUN wget http://www.rfxn.com/downloads/maldetect-current.tar.gz && \
    tar -xzf maldetect-current.tar.gz && \
    cd maldetect-* && \
    ./install.sh

COPY . /app

# Set NVD API key environment variable
ENV NVD_API_KEY=[redacted]

# Copy the dependency-check.properties file
COPY dependency-check.properties /app/

# Install Python dependencies
RUN pip install --no-cache-dir click trufflehog3

# Run cli.py when the container launches
ENTRYPOINT ["python", "cli.py", "scan"]
```

Εικόνα 5. Αρχείο Dockerfile

Επεξήγηση Dockerfile:

- Χρησιμοποιεί την επίσημη slim εικόνα της Python 3.10 ως βάση, η οποία είναι πιο ελαφριά και περιέχει μόνο τα απαραίτητα στοιχεία (και για λόγους ασφαλείας).
- Ορίζει τον κατάλογο εργασίας ως /app και εγκαθιστά διάφορα απαραίτητα πακέτα.
- Εγκαθιστά τα εργαλεία σάρωσης.
- Αντιγράφει τα αρχεία της εφαρμογής από το τοπικό σύστημα φιλοξενίας και τον τρέχων φάκελο '.' στον '/app' της εικόνας.
- Ορίζει την περιβαλλοντική μεταβλητή NVD_API_KEY για την χρήση από το OWASP Dependency-Check.

3.3.3 Εργαλεία & Βιβλιοθήκες Ανάπτυξης

Το εργαλείο για τον έλεγχο ασφάλειας δοχείων Docker αναπτύχθηκε χρησιμοποιώντας την γλώσσα προγραμματισμού Python και τη βιβλιοθήκη Click. Η επιλογή της Python βασίστηκε στην ευκολία χρήσης της, την ευρεία υποστήριξη βιβλιοθηκών και εργαλείων, και την ισχυρή κοινότητα προγραμματιστών που διαθέτει. Η Python είναι γνωστή για την απλότητα και την καθαρότητα του κώδικά της, γεγονός που την καθιστά ιδανική για την ανάπτυξη εργαλείων που απαιτούν συνδυασμό διαφορετικών τεχνολογιών και βιβλιοθηκών.

Η Click είναι μια Python βιβλιοθήκη που χρησιμοποιείται για τη δημιουργία διεπαφών (χρήσης) γραμμής εντολών (Command Line Interfaces - CLI). Παρέχει έναν εύκολο και αποτελεσματικό τρόπο για την ανάπτυξη CLI εφαρμογών, επιτρέποντας στους προγραμματιστές να δημιουργούν σύνθετες δομές εντολών με απλές και κατανοητές συντακτικές δομές. Η Click υποστηρίζει τη δημιουργία σύνθετων εντολών με πολλές επιλογές, καθώς και τη διαχείριση λαθών και την παροχή βοήθειας στους χρήστες.

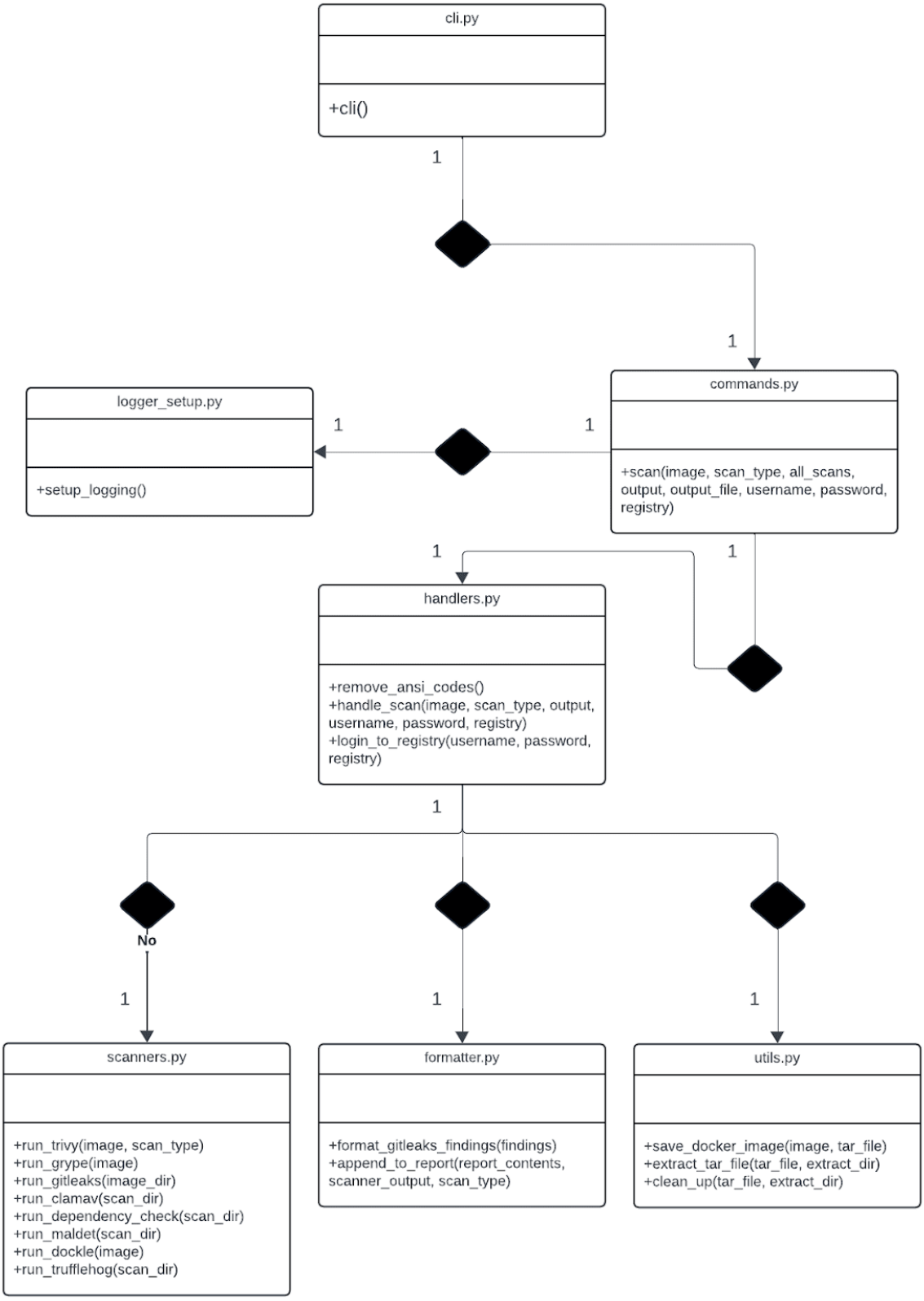
Τα εργαλεία ελέγχου ασφάλειας που ενσωματώθηκαν στο εργαλείο που αναπτύχθηκε περιλαμβάνουν τα εξής (όπως αναφέρθηκε και στην ενότητα 3.3): Trivy, Grype, Dockle, OWASP Dependency-Check, ClamAV, Maldet, Gitleaks και TruffleHog.

3.3.4 Πρόνοιες Βελτίωσης Απόδοσης Εργαλείου

Χρησιμοποιήθηκε εικόνα python:slim για την κατασκευή του εργαλείου, εξασφαλίζοντας ότι το περιβάλλον εκτέλεσης είναι ελαφρύ και γρήγορο. Βελτιστοποιήθηκε ο κώδικας για να εκτελεί σαρώσεις παράλληλα όπου αυτό είναι εφικτό, χρησιμοποιώντας threading ή multiprocessing. Χρησιμοποιήθηκαν αποδοτικές βιβλιοθήκες και εργαλεία σάρωσης που είναι γνωστά για τις επιδόσεις τους, όπως τα Trivy, Grype και ClamAV.

3.3.5 Δομή & Ανάλυση Κώδικα

Παραθέτουμε την δομή του πηγαίου κώδικα υλοποίησης του εργαλείου μέσω διαγράμματος κλάσεων και έπειτα αναλύουμε τις (κύριες) κλάσεις που τον απαρτίζουν.



Εικόνα 6. Διάγραμμα κλάσεων

Κύριες Κλάσεις

- **cli.py**: Entry point για την εφαρμογή CLI.

- **commands.py**: Καθορίζει την εντολή σάρωσης με επιλογές για διαφορετικούς τύπους σάρωσης και έξοδο.
- **handlers.py**: Περιέχει τη λογική χειρισμού της διαδικασίας σάρωσης.
- **scanners.py**: Υλοποιεί λειτουργίες για την εκτέλεση μεμονωμένων εργαλείων σάρωσης ασφαλείας.
- **utils.py**: Βοηθητικές λειτουργίες για αποθήκευση εικόνων Docker, εξαγωγή αρχείων tar και καθαρισμό.
- **formatter.py**: Λειτουργίες μορφοποίησης της αναφοράς
- **logger_setup.py**: Σύστημα καταγραφής (logging), όπου μετατρέπει μηνύματα από τα output και error streams σε καταγραφές μέσω ενός logger. Ανακατευθύνει τα outputs από το console στο αρχείο καταγραφής, αφαιρώντας τους ANSI κωδικούς από τα μηνύματα πριν τα καταγράψει.

3.4 Βαθμός Ικανοποίησης Απαιτήσεων

Παρουσιάζεται πίνακας που προσδιορίζει για κάθε απαίτηση τον βαθμό ικανοποίησής της από το εργαλείο, με αντίστοιχη δικαιολόγηση όπου απαιτείται.

Πίνακας 1. Βαθμός ικανοποίησης απαιτήσεων

Απαίτηση	**Περιγραφή**	**Βαθμός Ικανοποίησης**	**Δικαιολόγηση**
Λειτουργική Απαίτηση 1	Σάρωση εικόνας Docker για τρωτότητες, κακόβουλο λογισμικό, προβληματικές εξαρτήσεις, κρυπτογραφημένα μυστικά.	Πλήρης	Το εργαλείο ενσωματώνει Trivy και Grype για την ανίχνευση τρωτοτήτων, ClamAV και Maldet για την ανίχνευση κακόβουλου λογισμικού, Dockle και Dependency-Check για τον έλεγχο εξαρτήσεων, Gitleaks και TruffleHog για την ανίχνευση μη κρυπτογραφημένων μυστικών.
Λειτουργική Απαίτηση 2	Υποστήριξη πολλαπλών ειδών σαρώσεων	Πλήρης	Το εργαλείο επιτρέπει στον χρήστη να επιλέξει τα είδη σαρώσεων που επιθυμεί να εκτελέσει.
Λειτουργική Απαίτηση 3	Δημιουργία αναφοράς σε μορφή αρχείου κειμένου	Πλήρης	Η αναφορά παράγεται σε αρχείο κειμένου, το οποίο περιλαμβάνει όλα τα αποτελέσματα των σαρώσεων.
Λειτουργική Απαίτηση 4	Αυθεντικοποίηση χρήστη	Πλήρης	Δίνετε η δυνατότητα στον χρήστη να κάνει αυθεντικοποίηση και για

			Docker Hub αλλά και για ιδιωτικό αποθετήριο.
Μη Λειτουργική Απαίτηση 1	Ενσωμάτωση σε CI/CD αγωγούς	Πλήρης	Το εργαλείο μπορεί να ενσωματωθεί εύκολα σε CI/CD αγωγούς, προσφέροντας αυτοματοποιημένο έλεγχο ασφάλειας.
Μη Λειτουργική Απαίτηση 2	Εύχρηστη διεπαφή CLI	Πλήρης	Η διεπαφή CLI είναι σχεδιασμένη με το Click framework, επιτρέποντας εύκολη χρήση και διαμόρφωση.
Μη Λειτουργική Απαίτηση 3	Αναλυτικές αναφορές	Πλήρης	Οι αναφορές που παράγονται περιλαμβάνουν λεπτομερή αποτελέσματα από όλες τις σαρώσεις, παρέχοντας αναλυτικές πληροφορίες.
Μη Λειτουργική Απαίτηση 4	Αξιοπιστία και σταθερότητα εργαλείου	Πλήρης	Το εργαλείο δοκιμάστηκε εκτενώς για να διασφαλιστεί η αξιοπιστία και η σταθερότητά του κατά την εκτέλεση σαρώσεων.
Μη Λειτουργική Απαίτηση 5	Απόδοση	Ικανοποιητικό	Το εργαλείο είναι γρήγορο, αλλά μπορεί να επιβραδυνθεί σε πολύ μεγάλες εικόνες Docker.
Μη Λειτουργική Απαίτηση 6	Καταγραφή ενεργειών και αποτελεσμάτων (Logging)	Πλήρης	Το εργαλείο παρέχει πλήρη καταγραφή των ενεργειών και αποτελεσμάτων, χρησιμοποιώντας τη βιβλιοθήκη `logging` της Python για την τεκμηρίωση και την παρακολούθηση της διαδικασίας σάρωσης.
Ο πίνακας δείχνει ότι όλες οι απαιτήσεις έχουν ικανοποιηθεί πλήρως, εξασφαλίζοντας έτσι τη λειτουργικότητα και την αποδοτικότητα του εργαλείου για τον έλεγχο ασφάλειας δοχείων Docker.			

4

Εγκατάσταση & Επίδειξη

4.1 Εγκατάσταση

Παρακάτω παραθέτουμε τις οδηγίες εγκατάστασης & εκτέλεσης του εργαλείου και των απαραίτητων προαπαιτούμενων.

4.1.1 Προαπαιτούμενα

Βεβαιωθείτε ότι έχετε εγκαταστήσει το Docker στο μηχάνημά σας. Σχετικές οδηγίες εγκατάστασης ανάλογα με το λειτουργικό σας σύστημα παρέχονται εδώ: https://docs.docker.com/engine/install/.

4.1.2 Οδηγίες Εγκατάστασης & Εκτέλεσης

- Κάντε clone το repository που περιέχει τα αρχεία του έργου:
  - git clone https://github.com/DimitrisK01/sec-cli-tool.git
- Μεταβείτε στον φάκελο του έργου:
  - cd sec-cli-tool
- Κάντε build την εικόνα του δοχείου του εργαλείου μας μέσω του υπάρχοντος Dockerfile στον φάκελο του έργου:

- docker build -t cli-container-security .

- Δημιουργήστε ένα δοχείο (container) Docker με την απαραίτητη προσάρτηση αποθηκευτικού χώρου (volume mount) για πρόσβαση στο Docker socket (απαραίτητη ρύθμιση για να μπορεί το δημιουργούμενο δοχείο να δημιουργεί και διαχειρίζεται άλλα δοχεία) και συμπληρώστε την εικόνα που θέλετε να σαρώσετε καθώς και τους επιθυμητούς τύπους σάρωσης. Σε περίπτωση που θέλετε να αποθηκεύσετε τα αποτελέσματα που οπτικοποιούνται από το εργαλείο, συμπληρώστε "-o -f <text_name>.txt" στο τέλος.
    - Γενικό παράδειγμα εκτέλεσης:
        - docker run --rm -v /var/run/docker.sock:/var/run/docker.sock -v $(pwd)/output:/app/output cli-container-security -i <image-tag> -t vulnerabilities,malware,dependencies,secrets -o -f <file-name>.txt --username '' --password '' –registry ''
        - Στο παράδειγμα αυτό παρατηρήστε τα εξής:
            - -v /var/run/docker.sock:/var/run/docker.sock - εδώ έχουμε την προσάρτηση που αναφέραμε προηγουμένως
            - -v $(pwd)/output:/app/output - εδώ έχουμε αντιστοίχιση του τοπικού φακέλου output στον φάκελο /app/output στο δημιουργούμενο δοχείο
            - -i <image-tag>: εδώ προσδιορίζουμε την ετικέτα της εικόνας (η γενική μορφή της ετικέτας είναι <{repo}/image-tag:image-version> όπου το μέρος {repo} είναι προαιρετικό και θα πρέπει να χρησιμοποιηθεί εφόσον η εικόνα εντοπίζεται σε αποθετήριο εικόνων και όχι τοπικά στο σύστημα φιλοξενίας) προς σάρωση.
            - -t <scan-type>: προσδιορισμός τύπου σάρωσης. Δυνατές τιμές: vulnerabilities, malware, dependencies & secrets. Εφόσον επιθυμούμε υποστήριξη πολλαπλών τύπων σάρωσης, θα πρέπει να παρέχονται οι τιμές τους στο τρέχων flag με ',' ως διαχωριστικό και χωρίς κενό. Για παράδειγμα: -t vulnerabilities,malware.
            - -o -f <file-name>.txt: προσδιορίζουμε με το flag -o ότι θέλουμε αποθήκευση αρχείου και με το flag -f παρέχεται το όνομα του αρχείου (αναφοράς) προς αποθήκευση στο φάκελο /app/output του δοχείου.
            - –username και password προσδιορίζουμε το όνομα χρήστη και κωδικό πρόσβασης για χρήση αποθετηρίου και –registry αμα θέλουμε να προσδιορίσουμε αποθετήριο, στην περίπτωση που δεν χρησιμοποιηθεί το –registry το αποθετήριο που χρησιμοποιείται είναι το DockerHub και αν δεν χρησιμοποιηθούν –username και –password τοτε το εργαλείο ψαχνει τοπικά την εικόνα.

4.2 Επίδειξη

Η επίδειξη γίνεται μέσω μιας σειράς από σενάρια που αναλύονται στις επόμενες υπο-ενότητες.

4.2.1 Σενάριο 1: Έλεγχος Ασφάλειας Πριν την Διάταξη

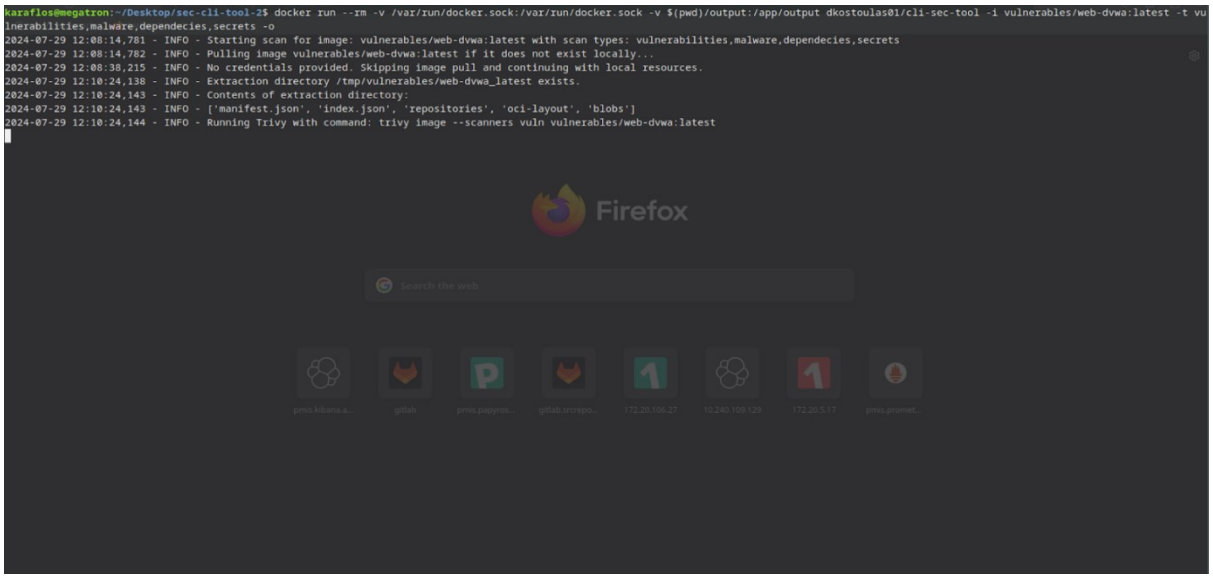
Πρόβλημα: Οι προγραμματιστές δημιουργούν νέες Docker εικόνες για την εφαρμογή τους. Πριν την διάταξη της εφαρμογής σε περιβάλλον παραγωγής, θέλουν να είναι σίγουροι ότι οι εικόνες δεν περιέχουν γνωστές ευπάθειες ή κακόβουλο λογισμικό.

Χρήση Εργαλείου: Οι προγραμματιστές χρησιμοποιούν το εργαλείο CLI για να σαρώσουν αυτές τις εικόνες Docker, το οποίο εκμεταλλεύεται τα τελευταία δεδομένα ευπαθειών που παρέχονται από τα ενσωματωμένα εργαλεία σάρωσης, για την ανίχνευση και αναφορά πιθανών ζητημάτων ασφάλειας. Οι προγραμματιστές έπειτα λαμβάνουν ένα αναλυτικό αρχείο αναφοράς, το οποίο περιγράφει τυχόν ευπάθειες ή κακόβουλο λογισμικό και τις δυνατές επιπτώσεις τους. Το εργαλείο δεν διατηρεί τη δική του βάση δεδομένων ευπαθειών, αλλά ενσωματώνει και ενοποιεί τα αποτελέσματα των εργαλείων σάρωσης που χρησιμοποιεί.

Εντολή Εκτέλεσης:

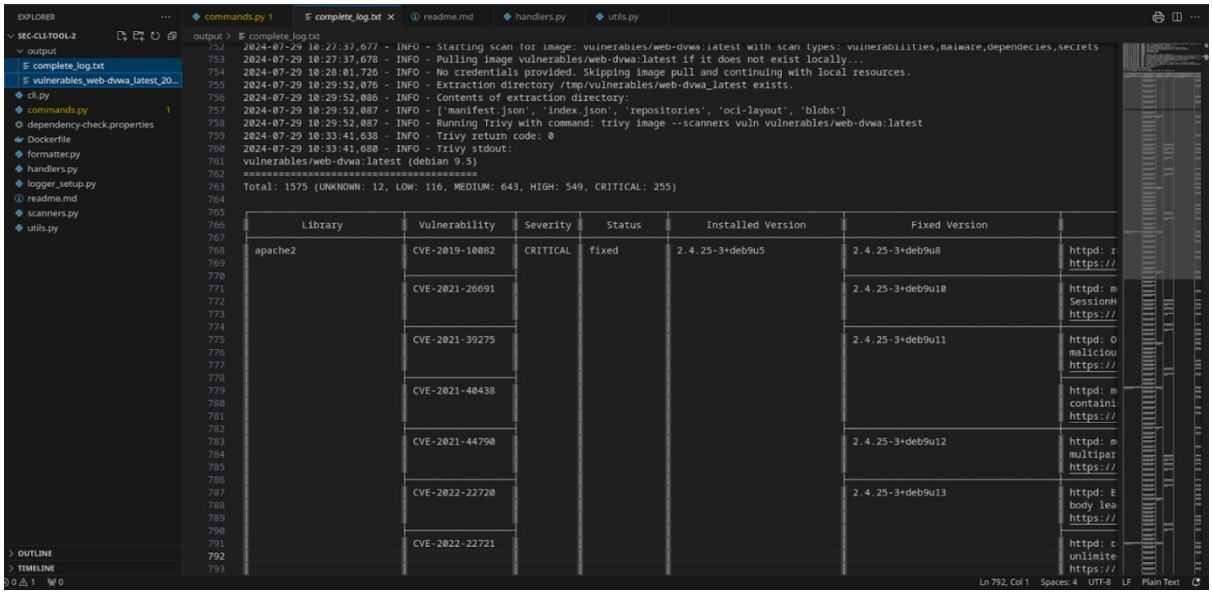
docker run --rm -v /var/run/docker.sock:/var/run/docker.sock -v $(pwd)/output:/app/output cli-sec-tool -i <image>:<tag> -t vulnerabilities,malware,dependencies,secrets -o

Η παραπάνω εντολή θα πρέπει να εκτελεστεί για κάθε εικόνα της εφαρμογής προς σάρωση. Όπως θα παρατηρήσετε, εφαρμόζονται όλοι οι έλεγχοι ασφάλειας κατά την εκτέλεσή της.



Εικόνα 7. Εκτέλεση εντολής σάρωσης με όλους τους ελέγχους ασφάλειας

Η παρακάτω εικόνα είναι από το complete_log.txt και μας δείχνει ευπάθειες που το εργαλείο trivy εντοπίζει μέσω και της χρήσης της CVE βάσης δεδομένων (Common Vulnerabilities and Exposures) για τα πακέτα και τις βιβλιοθήκες.



Εικόνα 8. Ολοκληρωμένο αρχείο καταχωρήσεων, παραγόμενο από το εργαλείο trivy

4.2.2 Σενάριο 2: Ενσωμάτωση σε CI/CD Pipeline

Πρόβλημα: Ένας οργανισμός θέλει να ενσωματώσει αυτόματους ελέγχους ασφάλειας στους αγωγούς (pipeline) CI/CD του για να διασφαλίσει ότι όλες οι εικόνες που αναπτύσσονται είναι ελεύθερες από γνωστές ευπάθειες και κακόβουλο λογισμικό και άρα μπορούν να χρησιμοποιηθούν για την διάταξη της εφαρμογής σε περιβάλλοντα (όπως παραγωγής) και την ενημέρωσή της σε αυτά τα περιβάλλοντα.

Χρήση Εργαλείου: Το εργαλείο CLI ενσωματώνεται στον αγωγό CI/CD, ο οποίος περιλαμβάνει ένα βήμα για την αυτόματη σάρωση της κάθε νέας εικόνα που δημιουργείται. Αν ανιχνευθεί κάποιο πρόβλημα, το εργαλείο το καταγράφει στην αναφορά, η οποία αποθηκεύεται σαν artifact, ενημερώνοντας τους προγραμματιστές για την ανάγκη αναθεώρησης της "προβληματικής" εικόνας που εντοπίστηκε.

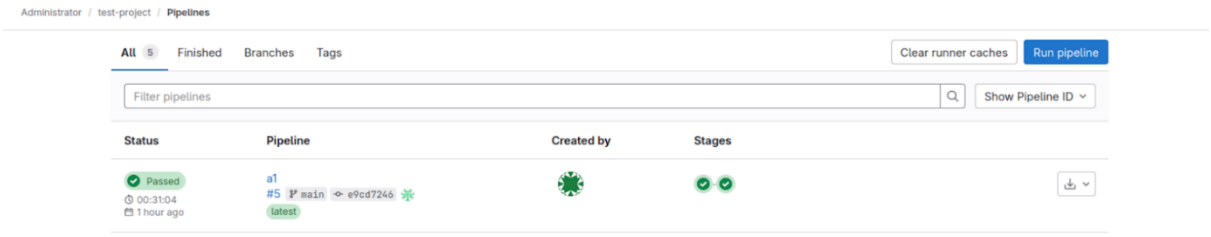
Οι αγωγοί ορίζονται και εκτελούνται στο Gitlab. Παρακάτω θα δούμε την ενσωμάτωση της χρήσης του εργαλείου σε έναν αγωγό CI/CD που αφορά μια συγκεκριμένη εφαρμογή.

Εντολή Εκτέλεσης με Gitlab Pipeline (.gitlab-ci.yml):

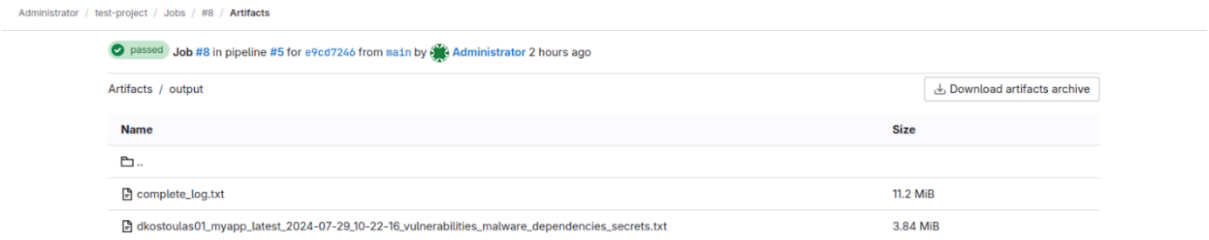
```
image: docker:latest  # Χρησιμοποιείται το Docker image για την εκτέλεση των εντολών


stages:
  - build  # Στάδιο για το build της εφαρμογής
  - scan   # Στάδιο για το scan ασφαλείας της εφαρμογής
```

```
variables:
  DOCKER_DRIVER: overlay2  # Ορίζουμε το docker driver σε overlay2, για καλύτερη απόδοση και αποθήκευση των images
  DOCKER_REGISTRY: "<url-repo>"  # URL του docker registry όπου θα γίνει το push του image
  DOCKER_USER: "xxx"  # Όνομα χρήστη για το docker registry
  DOCKER_PASSWORD: 'xxx'  # Κωδικός για το docker registry
  DOCKER_TLS_CERTDIR: ""  # Απενεργοποιούμε το TLS για επικοινωνία με το Docker daemon
  DOCKER_HOST: "tcp://docker:2375"  # Επικοινωνία με τον Docker daemon σε συγκεκριμένη πόρτα χωρίς TLS

services:
  - docker:dind  # To service docker-in-docker, που επιτρέπει την εκτέλεση docker commands εντός container

before_script:
  - unset DOCKER_HOST  # Καθαρίζουμε τη μεταβλητή DOCKER_HOST για να αποφύγουμε conflicts
  - docker login -u "$DOCKER_USER" -p "$DOCKER_PASSWORD"  # Κάνουμε login στο Docker registry με τα credentials

build:
  stage: build  # Ορισμός του σταδίου build
  script:
    - docker build -t $DOCKER_REGISTRY/myapp:latest .  # Χτίζουμε το Docker image με tag 'latest' και το αποθηκεύουμε στο registry
    - docker push $DOCKER_REGISTRY/myapp:latest  # Κάνουμε push το image στο docker registry

scan:
  stage: scan  # Ορισμός του σταδίου scan
  script:
    - mkdir -p $(pwd)/output  # Δημιουργία του φακέλου output για αποθήκευση των αποτελεσμάτων του scan
    - docker pull dkostoulas01/cli-sec-tool:latest  # Τραβάμε το image του εργαλείου ασφαλείας για scan
    - echo "Starting scan container..."  # Εκτύπωση μηνύματος ότι ξεκινάει το scan
    - docker run --rm --privileged -v /var/run/docker.sock:/var/run/docker.sock -v $(pwd)/output:/app/output dkostoulas01/cli-sec-tool -i $DOCKER_REGISTRY/myapp:latest -t vulnerabilities,malware,dependecies,secrets -o  # Εκτελούμε το scan για το image και αποθηκεύουμε τα αποτελέσματα στο φάκελο output
```

```
  - echo "Contents of the output directory on the host:"  # Εκτύπωση μηνύματος ότι δείχνουμε τα περιεχόμενα του φακέλου output
  - ls -la $(pwd)/output  # Προβολή των αρχείων που υπάρχουν στον φάκελο output
 artifacts:
  paths:
   - output/  # Καθορισμός του φακέλου output ως artifact για αποθήκευση
  expire_in: never  # Το artifact δεν θα λήξει ποτέ
 dependencies:
  - build  # Το στάδιο scan εξαρτάται από το build, δηλαδή θα εκτελεστεί αφού ολοκληρωθεί το build
```



Εικόνα 9. Dashboard του Gitlab όπου φαίνεται η κατάσταση των εκτελούμενων αγωγών



Εικόνα 10. Παραγόμενα τεχνήματα από την εκτέλεση συγκεκριμένου αγωγού που περιλαμβάνει το εργαλείο μας

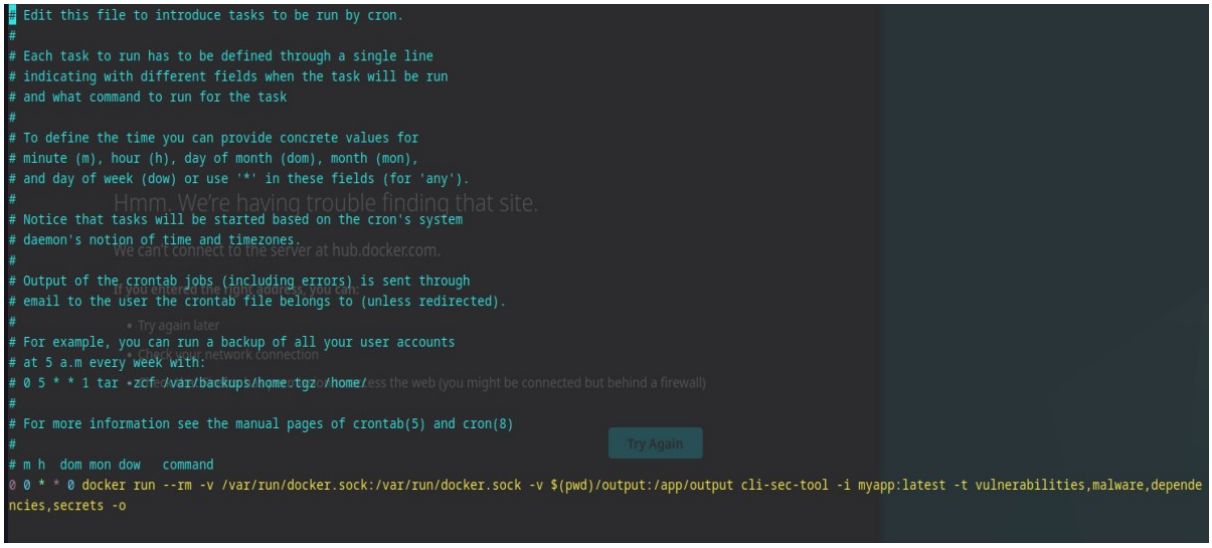
4.2.3 Σενάριο 3: Τακτικός Έλεγχος Εικόνων σε Παραγωγή

Πρόβλημα: Οι εικόνες Docker που χρησιμοποιούνται σε παραγωγικά περιβάλλοντα πρέπει να ελέγχονται τακτικά για να διαπιστώνονται νέες ευπάθειες που ενδέχεται να έχουν αποκαλυφθεί μετά την αρχική ανάπτυξη της εικόνας.

Χρήση Εργαλείου: Το εργαλείο CLI ρυθμίζεται να εκτελείται περιοδικά (π.χ. κάθε εβδομάδα) για να σκανάρει όλες τις εικόνες που είναι σε χρήση στο παραγωγικό περιβάλλον. Τα αποτελέσματα των ελέγχων καταγράφονται και επισκοπούνται για να ενημερώνονται οι σχετικές ομάδες ανάπτυξης και λειτουργίας (DevOps).

Εντολή Εκτέλεσης (χρησιμοποιώντας cron job):

crontab -e # Ανοίγει τον επεξεργαστή crontab για να επεξεργαστείς τα cron jobs
Επεξήγηση:
- Το πρώτο "0" αντιστοιχεί στα λεπτά (00 λεπτά).
- Το δεύτερο "0" αντιστοιχεί στην ώρα (00:00, δηλαδή μεσάνυχτα).
- Το πρώτο "*" σημαίνει ότι εκτελείται κάθε ημέρα του μήνα.
- Το δεύτερο "*" σημαίνει ότι εκτελείται κάθε μήνα.
- Το "0" αντιστοιχεί στην Κυριακή (0 = Κυριακή, 1 = Δευτέρα, κ.ο.κ.).
0 0 * * 0 docker run --rm -v /var/run/docker.sock:/var/run/docker.sock -v $(pwd)/output:/app/output cli-sec-tool -i <image>:<tag> -t vulnerabilities,malware,dependencies,secrets -o



Εικόνα 11. Περιεχόμενο crontab αρχείου για την περιοδική εκτέλεση σάρωσης ασφάλειας εικόνας

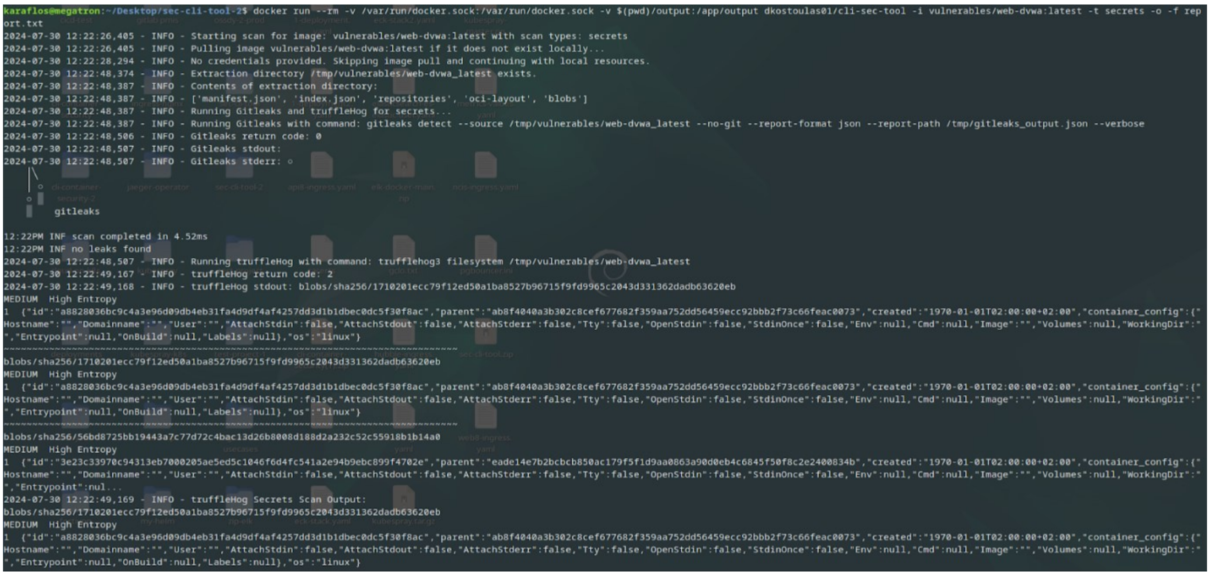
4.2.4 Σενάριο 4: Εντοπισμός και Διαχείριση Μη Κρυπτογραφημένων Μυστικών

Πρόβλημα: Μυστικά που αποθηκεύονται ακραίφνως σε εικόνες Docker μπορεί να αποτελέσουν σοβαρό κίνδυνο ασφάλειας.

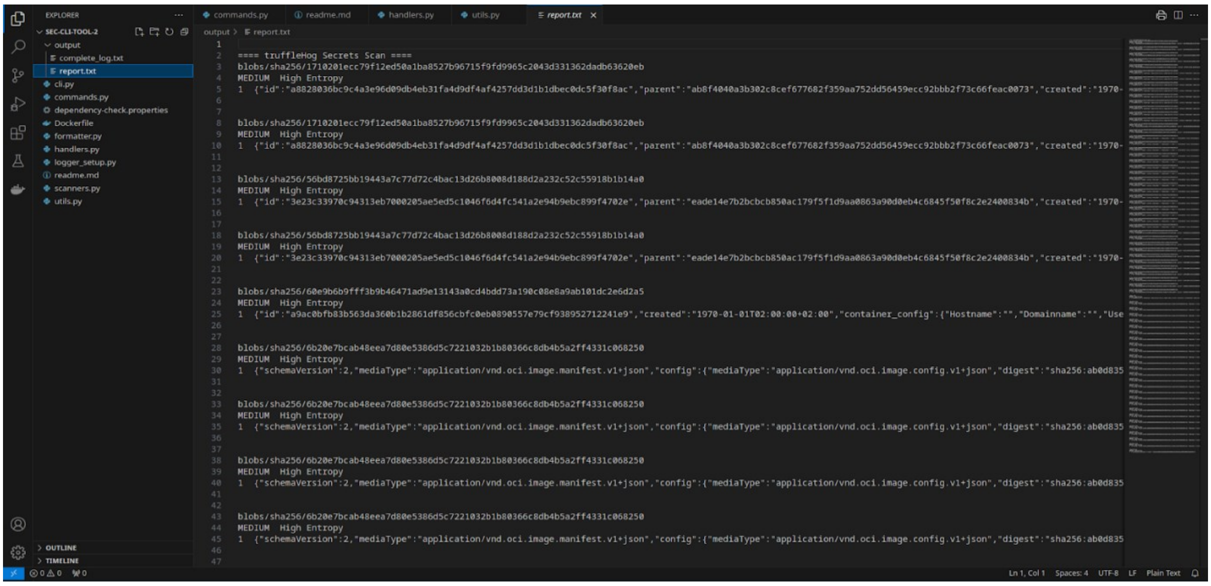
Χρήση Εργαλείου: Το εργαλείο παρέχει τη δυνατότητα σάρωσης για μη κρυπτογραφημένα μυστικά (secrets) μέσα στις εικόνες Docker. Με τη χρήση του, μπορεί να εντοπιστούν API keys, κωδικού (passwords) και άλλα ευαίσθητα δεδομένα που δεν θα έπρεπε να είναι ελεύθερα προσβάσιμα ή αποθηκευμένα σε ανασφαλή τοποθεσία, αλλά και να παραχθεί αναφορά με συγκεκριμένη ονομασία.

Εντολή Εκτέλεσης:
docker run --rm -v /var/run/docker.sock:/var/run/docker.sock -v $(pwd)/output:/app/output cli-sec-tool -i <image>:<tag> -t secrets -o -f <report-name>.txt

Παρακάτω βλέπουμε την εκτέλεση του εργαλείου, το οποίο ψάχνει με το Gitleaks και TrufleHog για μη κρυπτογραφημένα μυστικά.



Εικόνα 12. Εκτέλεση του εργαλείου μας για την ανίχνευση μη κρυπτογραφημένων μυστικών



Εικόνα 13. Τα αποτελέσματα της εκτέλεσης στο αρχείο της παραγόμενης αναφοράς

Εδώ βλέπουμε τα αποτελέσματα της εκτέλεσης στο αρχείο report.

Κάθε σενάριο χρήσης αποτελεί ένα παράδειγμα πραγματικής ανάγκης για ασφάλεια και συντήρηση των Docker εικόνων και δοχείων, καθιστώντας το εργαλείο CLI μια πολύτιμη προσθήκη κάθε ομάδας ανάπτυξης και λειτουργίας λογισμικού.

5

Συμπεράσματα & Μελλοντική Εργασία

5.1 Συμπεράσματα

Στα πλαίσια της εργασίας αυτής αναπτύχθηκε ένα ολοκληρωμένο εργαλείο για τον έλεγχο ασφαλείας δοχείων Docker, με εύχρηστη διεπαφή CLI και ενσωματωμένους πολλούς τύπους σάρωσης ασφάλειας.

Το εργαλείο αυτό παρέχει στους χρήστες τη δυνατότητα να εκτελούν σαρώσεις ευπαθειών, ανίχνευσης κακόβουλου λογισμικού, ελέγχου εξαρτήσεων και εντοπισμού μη κρυπτογραφημένων μυστικών σε Docker εικόνες. Χρησιμοποιώντας κορυφαία εργαλεία σάρωσης ανοικτού κώδικα, όπως το Trivy, το Grype, το Dockle, το OWASP Dependency-Check, το ClamAV, το Maldet, το Gitleaks και το TruffleHog, διασφαλίζεται η αξιοπιστία και η πληρότητα των ελέγχων ασφάλειας.

Ένα από τα βασικά πλεονεκτήματα του εργαλείου είναι η ευκολία χρήσης του μέσω της διεπαφής CLI, η οποία επιτρέπει στους χρήστες να εκτελούν πολλαπλούς τύπους σαρώσεων με απλές εντολές. Επιπλέον, το εργαλείο ενσωματώνεται εύκολα σε περιβάλλοντα CI/CD, προσφέροντας αυτοματοποιημένους ελέγχους ασφάλειας κατά τη διάρκεια της ανάπτυξης και της διάθεσης των εφαρμογών.

Το εργαλείο παράγει αναλυτικές και φιλικές προς τον χρήστη αναφορές, οι οποίες συνδυάζουν τα αποτελέσματα από τα διάφορα εργαλεία σάρωσης. Αυτό επιτρέπει στους προγραμματιστές να έχουν μια συνολική εικόνα της ασφάλειας των Docker εικόνων τους και να λαμβάνουν κατάλληλα μέτρα για την αντιμετώπιση των ευπαθειών.

Το εργαλείο μπορεί να σαρώνει τόσο τοπικές όσο και απομακρυσμένες εικόνες που είναι αποθηκευμένες σε αποθετήρια. Επίσης, μπορεί να σαρώνει και δοχεία, εφόσον αυτά έχουν γίνει snapshot σε συγκεκριμένες εικόνες.

Η εμπειρία από την εκπόνηση αυτής της εργασίας ενίσχυσε τη γνώση στη χρήση και ανάπτυξη εργαλείων ασφάλειας, καθώς και στη χρήση τεχνολογιών Docker, προσφέροντας ένα ισχυρό εργαλείο που μπορεί να βελτιώσει σημαντικά την ασφάλεια των εφαρμογών που βασίζονται σε Docker δοχεία.

5.2 Μελλοντική εργασία

Σε αυτή την ενότητα παρέχουμε ορισμένες κατευθύνσεις μελλοντικής εργασίας με σκοπό την βελτίωση και επέκταση του εργαλείου μας.

Η ενσωμάτωση περισσότερων εργαλείων ελέγχου ασφάλειας μπορεί να βελτιώσει την πληρότητα και την ακρίβεια των σαρώσεων. Υπάρχουν πολλά διαθέσιμα εργαλεία στην αγορά που εξειδικεύονται σε διαφορετικούς τύπους ευπαθειών και απειλών. Ενσωματώνοντας επιπλέον εργαλεία, όπως το Clair για αναλύσεις ευπαθειών σε container images ή το SonarQube για στατική ανάλυση κώδικα, το εργαλείο μπορεί να προσφέρει ακόμη πιο εκτεταμένες δυνατότητες ελέγχου. Τονίζουμε βέβαια πως λόγω της φύσης του εργαλείου θα είναι προτιμότερη η ενσωμάτωση κυρίως εργαλείων ανοικτού κώδικα. Ώστε το εργαλείο να μπορεί να προσφέρεται κι αυτό σε αυτή την μορφή για να εξυπηρετεί αφιλοκερδώς την κοινότητα των προγραμματιστών.

Η βελτιστοποίηση της απόδοσης του εργαλείου είναι σημαντική για τη διασφάλιση της ταχείας και αποδοτικής εκτέλεσης των σαρώσεων. Αυτό μπορεί να επιτευχθεί μέσω της βελτίωσης του κώδικα για την εκτέλεση των σαρώσεων με υψηλότερο βαθμό παραλληλοποίησης (από τον τρέχων) ή της χρήσης cache για την αποθήκευση αποτελεσμάτων σαρώσεων που έχουν ήδη εκτελεστεί. Η χρήση cache για την αποθήκευση αποτελεσμάτων σαρώσεων που έχουν ήδη εκτελεστεί είναι μια στρατηγική που στοχεύει στη βελτίωση της ταχύτητας και της αποδοτικότητας των διαδικασιών σάρωσης. Όταν εκτελούνται σαρώσεις, τα αποτελέσματα μπορούν να αποθηκεύονται προσωρινά σε μια cache. Αυτό σημαίνει ότι αν η ίδια σάρωση χρειαστεί να επαναληφθεί στο μέλλον, το εργαλείο δεν θα χρειάζεται να επαναλάβει όλη τη διαδικασία από την αρχή, αλλά μπορεί να ανακτήσει τα ήδη αποθηκευμένα αποτελέσματα από την cache. Αυτή η προσέγγιση όχι μόνο εξοικονομεί χρόνο, αλλά και μειώνει την κατανάλωση πόρων, όπως η CPU και η μνήμη, δεδομένου ότι αποφεύγονται περιττές υπολογιστικές διαδικασίες. Ειδικότερα σε περιβάλλοντα όπου οι σαρώσεις εκτελούνται συχνά ή σε μεγάλα σύνολα δεδομένων, η αποδοτική χρήση της cache μπορεί να προσφέρει σημαντικά οφέλη, βελτιώνοντας την ταχύτητα εκτέλεσης και την εμπειρία του χρήστη.

Επιπλέον, η αναβάθμιση της υποδομής και η χρήση πιο αποδοτικών αλγορίθμων δηλαδή χρήση βιβλιοθηκών που υποστηρίζουν παραλληλία (π.χ. threading της python που χρησιμοποιήθηκε στο εργαλείο OWASP Dependency check) για την εκτέλεση πολλαπλών σαρώσεων ή διεργασιών ταυτόχρονα, εκμεταλλευόμενοι τους πολλούς πυρήνες επεξεργαστή. Αυτό μπορεί να συμβάλλει στη βελτίωση της συνολικής απόδοσης.

Η επέκταση της υποστήριξης για διαφορετικά περιβάλλοντα ανάπτυξης και υποδομές μπορεί να καταστήσει το εργαλείο πιο ευέλικτο και χρήσιμο σε διαφορετικά σενάρια χρήσης. Αυτό περιλαμβάνει την υποστήριξη για διαφορετικούς τύπους container runtimes πέραν του Docker, όπως το Kubernetes, και την επέκταση της συμβατότητας με διάφορα λειτουργικά συστήματα και αρχιτεκτονικές.

Η συνεχιζόμενη συντήρηση και ενημέρωση του εργαλείου με βάση τις τελευταίες εξελίξεις στην ασφάλεια και την τεχνολογία των containers είναι κρίσιμη για τη διασφάλιση της αποτελεσματικότητάς του. Αυτό περιλαμβάνει την τακτική ενημέρωση των ενσωματωμένων εργαλείων ελέγχου, τη διόρθωση σφαλμάτων και την προσθήκη νέων λειτουργιών βάσει των αναγκών και των προτάσεων της κοινότητας χρηστών.

6

Βιβλιογραφία

[1] J. Shetty, "A State-of-Art Review of Docker Container Security Issues and Solutions," *Am. Int. J. Res. Sci. Technol. Eng. Math.*, Jan. 2017.
[2] F. Loukidis-Andreou, I. Giannakopoulos, K. Doka, and N. Koziris, "Docker-Sec: A Fully Automated Container Security Enhancement Mechanism," in *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*, Jul. 2018, pp. 1561–1564. doi: 10.1109/ICDCS.2018.00169.
[3] A. Martin, S. Raponi, T. Combe, and R. Pietro, "Docker ecosystem – Vulnerability Analysis," *Comput. Commun.*, vol. 122, Mar. 2018, doi: 10.1016/j.comcom.2018.03.011.
[4] Doç. Dr. A. Efe, U. Aslan, and A. Kara, "Securing Vulnerabilities in Docker Images," *Int. J. Innov. Eng. Appl.*, vol. 4, pp. 31–39, Jun. 2020, doi: 10.46460/ijiea.617181.
[5] M. Patra, Kumari, B. Sahoo, and A. Turuk, "Docker Security: Threat Model and Best Practices to Secure a Docker Container," Dec. 2022, pp. 1–6. doi: 10.1109/iSSSC56467.2022.10051481.