Πανεπιστήμιο Πειραιώς
Σχολή Τεχνολογιών Πληροφορικής και Επικοινωνιών
Τμήμα Ψηφιακών Συστημάτων
Π.Μ.Σ. Ασφάλεια Ψηφιακών Συστημάτων

Ασφάλεια σε Περιβάλλοντα Docker Container
(Docker Container Security)

Διπλωματική Εργασία

Δημήτριος Πατούνης

Επιβλέποντες:

Καθηγητής Χρήστος Ξενάκης

Γεώργιος Βάσιος, Αξιωματικός ΚΕΠΥΕΣ

Πειραιάς, 2022

Πανεπιστήμιο Πειραιώς
Σχολή Τεχνολογιών Πληροφορικής και Επικοινωνιών
Τμήμα Ψηφιακών Συστημάτων
Π.Μ.Σ. Ασφάλεια Ψηφιακών Συστημάτων

Ασφάλεια σε Περιβάλλοντα Docker Container
(Docker Container Security)

Διπλωματική Εργασία

Δημήτριος Πατούνης

Αριθμός Μητρώου	E-mail
MTE2026	dpatounis@ssl-unipi.gr

Επιβλέποντες:
Καθηγητής Χρήστος Ξενάκης xenakis@unipi.gr
Γεώργιος Βάσιος, Αξιωματικός ΚΕΠΥΕΣ

“Sometimes a scream is better than a thesis.”

Manfred Eigen

Ευχαριστίες

Με την παρούσα διπλωματική εργασία ολοκληρώνονται οι σπουδές μου στο μεταπτυχιακό πρόγραμμα σπουδών «Ασφάλεια Ψηφιακών Συστημάτων» του Τμήματος Ψηφιακών Συστημάτων του Πανεπιστημίου Πειραιώς και θα ήθελα να εκφράσω τις θερμές μου ευχαριστίες σε όλους όσους συνέβαλαν στην εκπόνησή της.

Θα ήθελα να ευχαριστήσω θερμά τον επιβλέποντα καθηγητή μου, κ. Χρήστο Ξενάκη για την καθοδήγησή του και τις πολύτιμες συμβουλές του σε κάθε φάση της φοιτητικής μου σταδιοδρομίας, ως προπτυχιακός αλλά και ως μεταπτυχιακός φοιτητής.

Ιδιαίτερα επιθυμώ να ευχαριστήσω τον κ. Γεώργιο Βάσιο του ΚΕΠΥΕΣ για την καθοδήγηση και τις εποικοδομητικές του αποδείξεις, οι οποίες συνέβαλαν στην ολοκλήρωση αυτής της εργασίας.

Τέλος, θα ήθελα να ευχαριστήσω την οικογένειά μου, την κοπέλα μου αλλά και τους φίλους μου για τη στήριξή, τη συμπαράσταση και την υπομονή τους.

Περίληψη

Το docker είναι ένα από τα πιο περιζήτητα open-source εργαλεία για το containerization. Επιτρέπει στους προγραμματιστές να δημιουργούν εφαρμογές σε containers τα οποία είναι τυποποιημένα εκτελέσιμα στοιχεία που συνδυάζουν τον πηγαίο κώδικα εφαρμογής με τις βιβλιοθήκες του λειτουργικού συστήματος και τα απαιτούμενα dependencies για την εκτέλεση αυτού του κώδικα σε οποιοδήποτε περιβάλλον.

Ο σκοπός αυτής της διπλωματικής εργασίας είναι η περιγραφή της τεχνολογίας των containers και docker. Στη συνέχεια θα γίνει ανάλυση των βέλτιστων πρακτικών για έναν απλό docker web server. Έπειτα θα παρουσιαστούν βασικές έννοιες και μεθοδολογίες για έναν οργανισμό όπως είναι το red και blue teaming, το SIEM και η ανάλυση ευπαθειών. Θα γίνει περιγραφή του Center for Internet Security Framework μαζί με το CIS Benchmark.

Το επόμενο στάδιο θα είναι η δημιουργία ενός περιβάλλοντος και η παρουσίαση των εργαλείων που θα χρησιμοποιηθούν στην μελέτη περίπτωσης που είναι το πρακτικό κομμάτι της εργασίας. Σε αυτό το κομμάτι, θα εφαρμοστούν οι βέλτιστες πρακτικές που παρουσιάστηκαν, θα αναπτυχθεί ένα SIEM για την παρακολούθηση του περιβάλλοντος που δημιουργήσαμε. Το πρακτικό κομμάτι θα ολοκληρωθεί με την αξιολόγηση του επιπέδου ασφάλειας του συστήματος. Τέλος, θα παρουσιαστούν τα συμπεράσματα που προέκυψαν τόσο από τη θεωρητική μελέτη αλλά και από το πρακτικό κομμάτι της διπλωματικής εργασίας.

Λέξεις Κλειδιά

Docker, Containers, Security, Hardening, Monitoring, Auditing, Compliance, Best Practices

Abstract

Docker is one of the most sought after open-source developer tools. It allows developers to create applications in containers, the standard executable components, that combine the source code of the application with the libraries of the operating system and the dependencies required to execute this code in any environment.

The purpose of this thesis is to describe the technology of containers and of docker. Then we analyze the best practices for a simple docker web server. Then basic concepts and methodologies for an organization such as red and blue teaming, SIEM and vulnerability analysis will be presented. The Center for Internet Security Framework will be described together with the CIS Benchmark.

The next stage will be the creation of an environment and the presentation of the tools that will be used in the case study, which is the practical part of the work. In this part, the best practices will be applied and a SIEM will be developed to monitor the environment we have created. The practical part will be completed with the evaluation of the security level of the system. Finally, the conclusions that emerged from both the theoretical study and the practical part of the thesis will be discussed.

Keyword

Docker, Containers, Security, Hardening, Monitoring, Auditing, Compliance, Best Practices

Πίνακας Περιεχομένων

Πίνακας Εικόνων

Κεφάλαιο 1. Περιγραφή Εργασίας

Η παρούσα διπλωματική εργασία εκπονήθηκε στο πλαίσιο του Π.Μ.Σ. “Ασφάλεια Ψηφιακών Συστημάτων” του “Τμήματος Ψηφιακών Συστημάτων” της “Σχολή Τεχνολογιών Πληροφορικής και Επικοινωνιών” του Πανεπιστημίου Πειραιώς από το μεταπτυχιακό φοιτητή Πατούνη Δημήτριο.

Ο στόχος της συγκεκριμένης διπλωματικής εργασίας αποτελείται από το πρώτο τμήμα που είναι η περιγραφή των container, της τεχνολογίας docker καθώς και των βέλτιστων πρακτικών για την επιτυχή σκλήρυνση και ασφάλεια ενός τέτοιου περιβάλλοντος. Το δεύτερο τμήμα αποτελείται με την εφαρμογή των βέλτιστων αυτών πρακτικών σε ένα Case Study.

Κεφάλαιο 2. Εισαγωγή

Η τεχνολογία Docker container πρωτοεμφανίστηκε το 2013 παρουσιάζοντας μια πλατφόρμα ανοιχτού κώδικα που βοήθησε στην επανάσταση προς τη δημιουργία container στην ανάπτυξη λογισμικού. Τα container βοηθούν στην επίλυση του προβλήματος της αποστολής λογισμικού από το μηχάνημα Α στο μηχάνημα Β με αξιόπιστο και αυτοματοποιημένο τρόπο και με αυτό το τρόπο έχουν βοηθήσει τις ομάδες ανάπτυξής αλλά και τις ομάδες DevOps να αυξήσουν την ευελιξία και να επιταχύνουν την ανάπτυξη και την παράδοση εφαρμογών.

Ειδικά καθώς οι εφαρμογές γίνονται πιο περίπλοκες με την πάροδο των ετών και η στοίβα λογισμικού και η υποδομή έγιναν πιο περίπλοκες, οι προγραμματιστές αντιμετωπίζουν συχνά προβλήματα να πάρουν κώδικα για να εκτελείται αξιόπιστα σε διάφορα μηχανήματα.

Πλέον, το Docker είναι η πιο δημοφιλής τεχνολογία containerization. Με τη σωστή χρήση, μπορεί να αυξήσει το επίπεδο ασφάλειας. Από την άλλη πλευρά όμως, ορισμένες εσφαλμένες ρυθμίσεις παραμέτρων μπορεί να οδηγήσουν σε υποβάθμιση του επιπέδου ασφάλειας ή ακόμη και στην εισαγωγή νέων τρωτών σημείων. Ένα μόνο παραβιασμένο Docker container μπορεί να απειλήσει όλα τα άλλα container καθώς και τον κεντρικό υπολογιστή.

Στη παρούσα διπλωματική εργασία θα προσπαθήσουμε να δώσουμε μια λύση σε αυτό το πρόβλημα που είναι η ασφάλεια της τεχνολογίας Docker. Στην αρχή θα γίνει μια περιγραφή της τεχνολογίας του containerization και της τεχνολογίας Docker. Θα γίνει ανάλυση των αρχιτεκτονικών, των πλεονεκτημάτων σε σχέση με άλλες τεχνολογίες αλλά και μια αρίθμηση των εφαρμογών αυτής της τεχνολογίας. Στο τέλος αυτού του κεφαλαίου θα γίνει μια εμβάθυνση στις σωστές μεθοδολογίες υλοποίησης αλλά και μια καταγραφή των βέλτιστων πρακτικών για την σωστή ανάπτυξη και υποστήριξη ενός περιβάλλοντος Docker.

Στο επόμενο κεφάλαιο, θα ασχοληθούμε με τον ορισμό των blue, red και purple ομάδων αλλά και με τη χρησιμότητά τους σαν ομάδα σε έναν οργανισμό. Θα συνεχίσουμε με την περιγραφή της έννοιας security information and event management, την ανάλυση της αρχιτεκτονικής αλλά και τα οφέλη που έχει αυτή η τεχνολογία. Το επόμενο κομμάτι θα παρουσιάσει τη σημασία της αξιολόγησης

της τρωτότητας και τον τρόπο που γίνεται μια σάρωση για την εύρεση ευπαθειών. Το κεφάλαιο θα τελειώσει με τη λεπτομερή εξέταση της ορολογίας του CIS Framework και CIS Benchmark, τον τρόπο ανάπτυξής τους και τα οφέλη που προσφέρουν.

Τα επόμενα δύο κεφάλαια θα αφιερωθούν στην περιγραφή της δημιουργίας του περιβάλλοντος που θα πραγματοποιηθεί το use case και την σύντομη παρουσίαση των εργαλείων που θα χρησιμοποιηθούν.

Στο προτελευταίο κεφάλαιο θα πραγματοποιήσουμε ένα case study με εφαρμογή των βέλτιστων πρακτικών ασφάλειας και την ανάπτυξη και εφαρμογή ενός open-source SIEM, το οποίο θα χρησιμοποιηθεί για τη παρακολούθηση του συστήματος. Θα τελειώσουμε το case study με τον έλεγχο του συστήματος μας σύμφωνα με τα πρότυπα του CIS Framework.

Η εργασία θα ολοκληρωθεί με παράθεση των συμπερασμάτων που εξήχθησαν από εκπόνησή της και θα παρουσιαστούν ιδέες για μελλοντική έρευνα πάνω σε αυτό τον τομέα.

Κεφάλαιο 3. Docker

Το Docker είναι μια ανοιχτή πλατφόρμα για ανάπτυξη, αποστολή και εκτέλεση εφαρμογών. Το Docker επιτρέπει τον διαχωρισμό των εφαρμογών από τη υποδομή, ώστε να παραδίδεται ένα λογισμικό γρήγορα.

Το Docker παρέχει τη δυνατότητα συσκευασίας και εκτέλεσης μιας εφαρμογής σε ένα απομονωμένο περιβάλλον που ονομάζεται container. Η απομόνωση και η ασφάλεια επιτρέπουν την εκτέλεση πολλών container ταυτόχρονα σε έναν δεδομένο κεντρικό υπολογιστή. Τα containers είναι ελαφριά και περιέχουν όλα όσα χρειάζονται για την εκτέλεση της εφαρμογής.

Το Docker μπορεί να συσκευάσει μια εφαρμογή και τα dependencies της σε ένα εικονικό container που μπορεί να εκτελεστεί σε οποιοδήποτε υπολογιστή Linux, Windows ή macOS.

3.1 Λειτουργικό Σύστημα Linux

Linux, λειτουργικό σύστημα υπολογιστή που δημιουργήθηκε στις αρχές της δεκαετίας του 1990 από τον Φινλανδό μηχανικό λογισμικού Linus Torvalds και το Free Software Foundation (FSF).

Το Linux αναπτύχθηκε αρχικά ως έργο χόμπι από τον Linus Torvalds. Εμπνεύστηκε από το Minix, ένα μικρό σύστημα UNIX που αναπτύχθηκε από τον Andy Tanenbaum και οι πρώτες συζητήσεις σχετικά με το Linux έγιναν στην ομάδα ειδήσεων USENET comp.os.minix. Το 1991 κυκλοφόρησε την έκδοση 0.02. Η έκδοση 1.0 του Linux Kernel κυκλοφόρησε το 1994 [1].

Σήμερα, το Linux είναι ένας πλήρης κλώνος Unix, ικανός να τρέξει το X Window System, TCP/IP, Emacs, Web, mail και άλλα. Σχεδόν όλα τα μεγάλα δωρεάν πακέτα λογισμικού έχουν μεταφερθεί στο Linux και διατίθεται εμπορικό λογισμικό. Στην πραγματικότητα, πολλοί προγραμματιστές ξεκινούν γράφοντας εφαρμογές για Linux και τις μεταφέρουν αργότερα σε άλλα συστήματα Unix.

Ένα Linux Distribution ή distro είναι ένα λειτουργικό σύστημα που δημιουργείται από μια συλλογή λογισμικού που βασίζεται στον πυρήνα του Linux και, συχνά, σε ένα σύστημα διαχείρισης πακέτων.

Ένα τυπικό distro περιλαμβάνει ένα Linux Kernel, εργαλεία και βιβλιοθήκες GNU, πρόσθετο λογισμικό, documentation, windows system window manager και ένα περιβάλλον επιφάνειας εργασίας. Υπάρχουν πάνω από 600 distros και περίπου 500 αναπτύσσονται [2].

3.2 Containerization

Η έννοια του containerization παρουσιάστηκε με την ανάπτυξη του chroot το 1979. Το Chroot, είναι η διαδικασία δημιουργίας εικονικού περιβάλλοντος σε λειτουργικό σύστημα Unix και προστέθηκε στην έκδοση 7 του Unix. Το Chroot σηματοδότησε την αρχή της απομόνωσης των containers, περιορίζοντας την πρόσβαση αρχείων μιας εφαρμογής σε έναν συγκεκριμένο κατάλογο/directory. Ένα βασικό όφελος του διαχωρισμού του chroot ήταν η βελτίωση της ασφάλειας του συστήματος, έτσι ώστε ένα απομονωμένο περιβάλλον να μην μπορεί να θέσει σε κίνδυνο τα εξωτερικά συστήματα εάν εκμεταλλευτεί μια εσωτερική ευπάθεια.

Τα Process Containers, που κυκλοφόρησαν από την Google το 2006, μπήκαν σε μια πιο λεπτομερή απομόνωση περιέχοντας διαδικασίες και όχι μόνο εφαρμογές. Τα containers έχουν σχεδιαστεί για τον περιορισμό (limit), τον απολογισμό (account) και την απομόνωση (isolation) της χρήσης πόρων.

Τα Cgroups εισήλθαν στον τομέα του containerization για τον έλεγχο των σχέσεων μεταξύ των διαδικασιών και περιορίστηκαν στην πρόσβαση των χρηστών σε συγκεκριμένες δραστηριότητες και όγκους μνήμης. Η έννοια του cgroup εισήχθη με σκοπό να προσθέσει ακόμη μεγαλύτερη απομόνωση για να κρατήσει τις διαδικασίες ξεχωριστές. Τα Cgroups απορροφήθηκαν στο core του Linux τον Ιανουάριο του 2008, Linux kernel v2.6.24, μετά την οποία προέκυψε η τεχνολογία Linux container LXC.

Στο containerization, η απομόνωση είναι το πιο βασικό στοιχείο. Για να προσθέσουμε μια άλλη πτυχή της απομόνωσης, το Linux Namespaces, ένα χαρακτηριστικό του Linux Kernel που χωρίζει τους πόρους του πυρήνα, ήρθε για να απομονώσει τους γενικούς πόρους του συστήματος μεταξύ ανεξάρτητων διαδικασιών. Τα namespaces παρέχουν τη βάση για την ασφάλεια του δικτύου

των containers, η οποία χρησιμοποιείται για την απόκρυψη της δραστηριότητας ενός χρήστη ή μιας ομάδας από άλλους στο ίδιο δίκτυο.

Τα containers υποστηρίζουν μια αρχιτεκτονική που βασίζεται σε μικροϋπηρεσίες, μια προσέγγιση για τον επαναπροσδιορισμό έργων λογισμικού μεγάλης κλίμακας ώστε να είναι πιο κλιμακούμενη και αρθρωτή/modular. Η τεχνολογία των containers μπορεί επίσης να διευκολύνει την εκτέλεση εφαρμογών σε διαφορετικά περιβάλλοντα εργασίας υπό διαφορετικές συνθήκες επειδή παρέχει ένα σταθερό runtime περιβάλλον [3].

Τα containers είναι τρόποι που “συσκευάζουν” ένα κώδικα και να τον επιτρέπουν να λειτουργεί σε οποιοδήποτε μηχάνημα. [4]

Ο κώδικας μπορεί να είναι σε διαφορετικές γλώσσες, να βασίζεται σε άλλο λογισμικό, να εγκαθιστά πακέτα από ένα repository στο διαδίκτυο και να έχει συγκεκριμένες απαιτήσεις συστήματος, όπως το μέγεθος της μνήμης, για να λειτουργεί σωστά. Κατά τη διαδικασίας μεταφοράς, τα containers συγκρατούν όλα τα απαραίτητα εργαλεία που απαιτούνται για την εκτέλεση ενός προγράμματος. Το container δίνει το δικό του περιβάλλον εκτέλεσης για να περιλαμβάνει πράγματα που επιτρέπουν την εκτέλεση ενός προγράμματος, όπως:

- Αρχεία
- Βιβλιοθήκες
- Μεταβλητές περιβάλλοντος

Αυτό διαφέρει από τις παραδοσιακές προσεγγίσεις για την εικονικοποίηση, όπου κάθε εφαρμογή απαιτεί ένα λειτουργικό σύστημα ή μια ολόκληρη εικονική μηχανή (Virtual Machine - VM) για εκτέλεση. Τα containers είναι ευέλικτα επειδή μπορούν να λειτουργήσουν σε:

- Bare-metal servers
- Cloud Servers
- Ένα μεμονωμένο VM σε ενα Server

Έτσι, αντί χρησιμοποιούν περιβάλλονταν άλλων, τα containers επιτρέπουν στους κωδικοποιητές να δημιουργούν δικά τους περιβάλλοντα σε διαφορετικούς servers. Οι servers είναι απλώς προσωπικοί υπολογιστές χωρίς όλα τα πλεονεκτήματα ενός κομψού σχεδιασμού και ενός φανταχτερού

λειτουργικού συστήματος. Όσον αφορά την απόδοση του προϊόντος, η μεταφορά containers διασφαλίζει:

- Αξιοπιστία
- Συνοχή

Οι προδιαγραφές στο πίσω μέρος των πακέτων λογισμικού ήταν ο τρόπος για τους προγραμματιστές λογισμικού να περιορίσουν τον κίνδυνο αποτυχίας των εφαρμογών στις συσκευές του χρήστη, ειδικά όταν αναπτύσσουν την εφαρμογή τους ευρέως σε πολλούς τύπους συστημάτων υπολογιστών. Σήμερα, οι προγραμματιστές εφαρμογών μπορούν να:

- Γίνονται πιο στοχευμένοι.
- Χρησιμοποιούν απομακρυσμένους υπολογιστές.
- Δίνουν οδηγίες για το ποιο ακριβώς σύστημα χρειάζεται το λογισμικό τους για να λειτουργήσει σωστά.

Στην πραγματικότητα, οι προγραμματιστές μπορούν να κατασκευάσουν μόνοι τους ένα container και να διασφαλίσουν ότι το λογισμικό λειτουργεί και δεν θα αποτύχει.

Κατά τη διαδικασίας αυτή, υπάρχουν δύο αρχεία προδιαγραφών που καθορίζουν τι λογισμικό απαιτείται σε ένα μηχάνημα και τι υλικό απαιτείται για τη λειτουργία του μηχανήματος. Αυτά είναι:

1. Το Dockerfile που ελέγχει το περιβάλλον εκτέλεσης και την εγκατάσταση των απαραίτητων πακέτων για επαρκή λειτουργία.
2. Το αρχείο Yaml που ελέγχει το υλικό και τις απαιτήσεις ασφάλειας δικτύου

Εκτός από την κατανόηση του framework, πρέπει επίσης να κατανοηθούν τα αποτελέσματα. Αφού δημιουργηθεί ένα container, σχηματίζει ένα image. Το image προκύπτει από το πλήρες runtime set σε ένα μόνο container. Μόλις ο κώδικας αποθηκευτεί και δημιουργηθεί το image, μπορεί στη συνέχεια να αναπτυχθεί σε έναν επιλεγμένο κεντρικό υπολογιστή. Αυτά μπορεί να είναι:

1. Σε τοπικά μηχανήματα
2. Σε μηχανές cloud

Τα container είναι η απάντηση σε πολλά προβλήματα όταν οι ομάδες αναπτύσσουν κώδικα:

- Διαφορετικές βάσεις κώδικα
- Κοινή χρήση κώδικα σε διαφορετικά συστήματα
- Κοινή χρήση κώδικα σε διαφορετικά runtime περιβάλλοντα
- Ασφάλεια
- Έκδοση

Ένα βασικό συστατικό της εικονικοποίησης είναι η απομόνωση, η πράξη του διαχωρισμού πόρων για κάθε εφαρμογή. Τα Container Engines έχουν καλύτερη απόδοση από τα VMs όταν πρόκειται για απομόνωση. Αλλά η επιλογή της χρήσης του ενός έναντι του άλλου χρειάζεται εξέταση και εξαρτάται από την περίπτωση χρήσης.

Σε γενικές γραμμές, σε σύγκριση με τα VMs, τα containers έχουν:

- Ταχύτερους χρόνους εκκίνησης
- Λιγότερο πλεονασμό
- Καλύτερη κατανομή πόρων

Στους περισσότερους προγραμματιστές αρέσουν τα containers επειδή είναι μια ευέλικτη, φιλική προς τους πόρους προσέγγιση στην ανάπτυξη λογισμικού. Οι εταιρείες μπορούν να διασφαλίσουν ότι πληρούνται όλες οι ανάγκες ανάπτυξης και εγκατάσταση τους, διατηρώντας παράλληλα τον χώρο του διακομιστή, είτε είναι φυσικός, είτε εικονικός είτε είναι νέφος.

Σε σύγκριση με τα VMs, τα containers:

- Χρησιμοποιούν λιγότερο χώρο, συνήθως μετριέται σε MB και όχι σε GB
- Μπορούν να περιοριστούν στην κατανάλωση ελάχιστου ποσού πόρων

Επειδή οι εφαρμογές container μπορούν να εκτελούνται σε cloud servers, είναι γενικά πιο προσιτές από άλλες εφαρμογές. Ο προγραμματισμός containers προσφέρει μια φορητή προσέγγιση ανάπτυξης λογισμικού.

Η πιο διαδεδομένη πλατφόρμα είναι το Docker, ένα σύστημα container ανοιχτού κώδικα που βασίζεται στο runC. Τα Docker images λειτουργούν σε διάφορες πλατφόρμες υπηρεσιών, καθιστώντας τα πιο ευέλικτα από ορισμένους ανταγωνιστές.

3.2.1 Πότε να χρησιμοποιούνται τα containers

Τα containers είναι ιδανικά για κάθε επιχείρηση ή οργανισμό που επιδιώκει να ενισχύσει τη διαχείριση της ψηφιακής επιχείρησης μέσω λύσεων που προσφέρουν αξιοπιστία, φορητότητα, ευελιξία και αναπαραγωγιμότητα σε ένα εικονικό περιβάλλον [4].

Οι εταιρείες που κινούνται προς μια DevOps κουλτούρα θα πρέπει να εισάγουν τα containers προσεκτικά. Κατά την εισαγωγή containers συνιστώνται τα ακόλουθα βήματα:

1. Αξιολόγηση των αναγκών των επιχειρήσεων. Θα πρέπει να πραγματοποιηθεί εξάσκηση στα containers σε μικρή κλίμακα και για να εξετάσουν ταιριάζουν στο επιχειρηματικό μοντέλο και τη κουλτούρα.
2. Πιλοτικά containers με την ομάδα DevOps.
3. Μετάβαση της φάσης παραγωγής και ανάπτυξη containers υποδομή της εταιρίας.

Οι οργανισμοί υιοθετούν όλο και περισσότερο τεχνολογία βασισμένη σε containers επειδή παρέχει πρωτοφανή φορητότητα που τους επιτρέπει να μετακινούν εφαρμογές σε διαφορετικές πλατφόρμες και περιβάλλοντα και να τις εκτελούν πιο ομαλά.

Όταν οι εφαρμογές μετακινούνται από μηχανή προγραμματιστή σε staging περιβάλλον, από staging περιβάλλον σε περιβάλλον παραγωγής ή από φυσική μηχανή σε εικονική μηχανή (VM), συχνά προκύπτουν ζητήματα ασυμβατότητας. Για παράδειγμα, ας υποθέσουμε ότι δοκιμάζεται το ίδιο λογισμικό στη σταδιοποίηση με την Python έκδοση 2.7, αλλά χρησιμοποιείται Python έκδοση 3.0 σε παραγωγή. Σε αυτό το σενάριο, θα μπορούσαν να αντιμετωπιστούν αρκετά ζητήματα μη συμβατότητας κατά την εκτέλεση της εφαρμογής μόλις μεταφερθεί στην παραγωγή.

Είναι γνωστό ότι τα εικονικά μηχανήματα μπορούν να περιλαμβάνουν ένα πλήρες λειτουργικό σύστημα με drivers, binaries και βιβλιοθήκες, ακόμη και πραγματικές εφαρμογές. Κάθε λειτουργικό σύστημα βρίσκεται πάνω από έναν hypervisor που ελέγχει το hardware του διακομιστή. Ωστόσο, ένα γνωστό πρόβλημα με την προσέγγιση VM είναι ότι μπορεί να χρησιμοποιήσει υπερβολικά τη μνήμη διακομιστή και να επηρεάσει την αποτελεσματικότητα.

Το containerization, από την άλλη πλευρά, αντιπροσωπεύει μια πολύ πιο εξορθολογισμένη/streamlined προσέγγιση στο DevOps [software

development (Dev) and IT operations (Ops)], πράγμα που σημαίνει ότι όλοι οι σχετικοί servers μπορούν να ενημερωθούν αμέσως μετακινώντας αλλαγές από σύστημα σε σύστημα. Το containerization μπορεί να βοηθήσει στη μείωση των πόρων που χρησιμοποιούνται επειδή κάθε container διατηρεί μόνο την εφαρμογή που διαχειρίζεται και τα σχετικά binaries ή τις απαιτούμενες βιβλιοθήκες.

Επιπλέον, το containerization μπορεί να παρέχει ένα runtime περιβάλλον, απαιτούμενα dependencies, βιβλιοθήκες, binaries και αρχεία που σχετίζονται με το configuration, όλα σε ένα πακέτο. Με τη μεταφορά της πλατφόρμας μιας εφαρμογής και των dependencies της, αφαιρούνται επίσης οι διαφορές στη διανομή λειτουργικού συστήματος και στην υποκείμενη υποδομή (underlying infrastructure) [3].

Τα containers μπορούν να παρέχουν πολλαπλά οφέλη, όπως:

- Απομόνωση εφαρμογών μεταξύ τους
- Απομόνωση εφαρμογών από τον κεντρικό υπολογιστή
- Βελτίωση της ασφάλειας των εφαρμογών περιορίζοντας τις δυνατότητές τους
- Ενθάρρυνση υιοθέτησης της αρχής του ελάχιστου προνομίου

3.2.2 Containers και Προκλήσεις Ασφάλειας

Το containerization έχει τη δυνατότητα να βελτιώσει τη συνολική παραγωγικότητα σε έναν οργανισμό και να επιταχύνει τη διαδικασία παράδοσης λογισμικού. Τα containers διαδραματίζουν κεντρικό ρόλο στην επιτυχία του DevOps, καθώς τα container images χρησιμεύουν ως πρότυπα για την πλήρη στοίβα πληροφορικής, από το λειτουργικό σύστημα, το μεσαίο λογισμικό και έως τον κωδικό εφαρμογής. Επομένως, ο τρόπος λειτουργίας ενός container κατά την ανάπτυξη είναι ο ίδιος τρόπος διασφάλισης ποιότητας (QA - quality assurance) και παραγωγής, με αποτέλεσμα την εκτέλεση της εφαρμογής χωρίς ταλαιπωρία όταν η εφαρμογή μεταφέρεται από το ένα περιβάλλον στο άλλο, γεγονός που μπορεί να βοηθήσει στην ταχεία ανάπτυξη εφαρμογών.

Όπως συμβαίνει με κάθε νέα τεχνολογία, τα containers έχουν δυνητικά οφέλη για μια επιχείρησή και συνοδεύεται από μοναδικές προκλήσεις ασφάλειας που πρέπει να γνωρίζει και κάθε οργανισμός που σκοπεύει να τα χρησιμοποιήσει.

Με τα container, το attack vector είναι μεγαλύτερο επειδή όλες οι εφαρμογές σε container μοιράζονται τον ίδιο πυρήνα λειτουργικού συστήματος. Επομένως, η πρόσβαση σε root στον κεντρικό υπολογιστή θα μπορούσε να επιτρέψει σε έναν εισβολέα να έχει πρόσβαση σε όλα τα container και να δει τι τρέχει μέσα σε αυτά που δεν έχουν την εξουσιοδότηση να φτάσουν.

Τα containers είναι επίσης πιο ευάλωτα σε επιθέσεις λειτουργικού συστήματος λόγω της μεγαλύτερης επιφάνειας επίθεσης που σχετίζεται με τη διεπαφή κλήσης συστήματος OS σε σύγκριση με την πολύ μικρότερη διεπαφή μεταξύ ενός VM και ενός hypervisor. Οι ευπάθειες στις κλήσεις συστήματος μπορούν να επιτρέψουν πιθανή πρόσβαση στον kernel. Μια τέτοια προνομιακή πρόσβαση θα μπορούσε να αποτελέσει πηγή συμβιβασμού ολόκληρου του συστήματος όπου φιλοξενούνται τα container.

Αλλά τα ζητήματα ασφάλειας μπορούν να αντιμετωπιστούν κατά τη φάση σχεδιασμού για να μπορέσουν οι ομάδες να επωφεληθούν από τη μεταφορά τους [3,5].

3.3 Αρχιτεκτονική Docker

Το Docker είναι μια πλατφόρμα ως προϊόν (platform as a service - PaaS) που χρησιμοποιεί εικονικοποίηση για την παράδοση λογισμικού σε πακέτα που ονομάζονται container. Η εταιρεία προγραμματιστών είναι η Docker, Inc. και ξεκίνησε για πρώτη φορά το 2013 [6]. Το λογισμικό horst ονομάζεται Docker Engine και τα κύρια συστατικά του είναι:

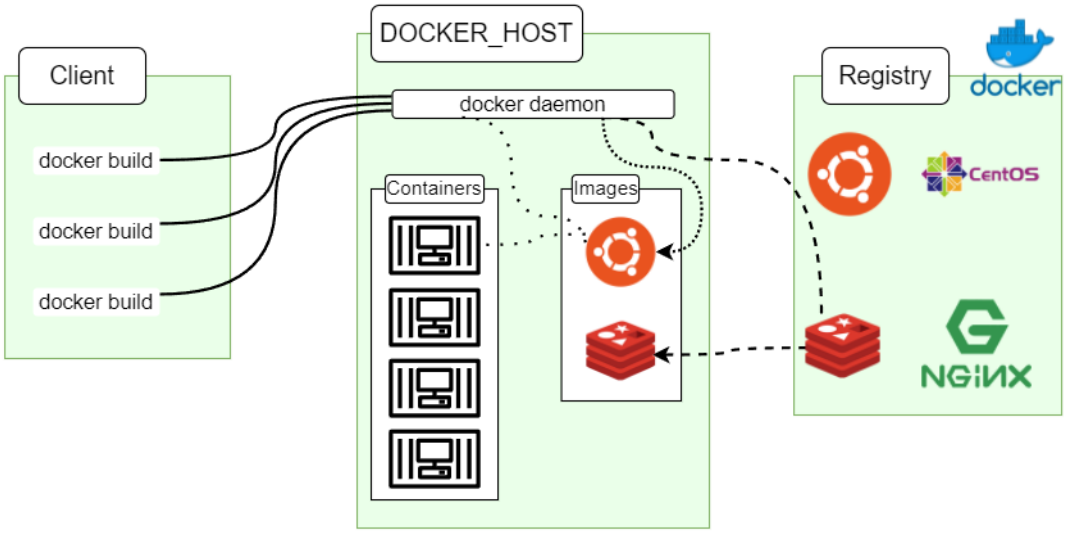
- Software
- Objects
- Registries

Τα κύρια εργαλεία του Docker είναι:

- Docker Compose

- Docker Swarm
- Docker Volume

Το Docker χρησιμοποιεί αρχιτεκτονική client-server. Το Docker client μιλά με το Docker daemon, το οποίο εκτελεί τις περισσότερες διαδικασίες Το client-server Docker και το daemon μπορούν να εκτελεστούν στο ίδιο σύστημα ή μπορούν να συνδεθούν ένα client σε ένα daemon. Το client και το daemon επικοινωνούν χρησιμοποιώντας ένα REST API, μέσω UNIX socket ή διεπαφής δικτύου. Ένα άλλο Docker client είναι το Docker Compose, που επιτρέπει την εργασία με εφαρμογές που αποτελούνται από ένα σύνολο από containers [7]. Η αρχιτεκτονική Docker παρουσιάζεται στην Εικόνα 1.



Εικόνα 1 Docker Αρχιτεκτονική

3.3.1 Objects

Όταν χρησιμοποιείται το Docker, δημιουργούνται και χρησιμοποιούνται images, containers και άλλα objects. Αυτά τα objects είναι εικόνες:

- Images
- Containers
- Networking
    - Bridge network
    - Overlay network
    - Macvlan network

- Storage
    - Data Volumes
    - Data Volume Container
    - Directory Mounts
    - Storage Plugins

3.3.2 Registries

Ένα Docker Registry αποθηκεύει Docker Images. Το Docker Hub είναι ένα δημόσιο registry που μπορεί να χρησιμοποιήσει ο καθένας και το Docker έχει ρυθμιστεί να αναζητά images στο Docker Hub από προεπιλογή. Μπορείτε ακόμη να δημιουργηθεί ένα ιδιωτικό registry.

Όταν χρησιμοποιούνται οι εντολές pull ή run docker, τα απαιτούμενα images εξάγονται από το διαμορφωμένο registry. Όταν χρησιμοποιείται η εντολή push docker, το image μεταφέρεται στο διαμορφωμένο registry.

Τα registries είναι ένα σημαντικό συστατικό στο οικοσύστημα Docker. Χρησιμοποιούνται σε διαφορετικά στάδια του κύκλου ζωής της εφαρμογής:

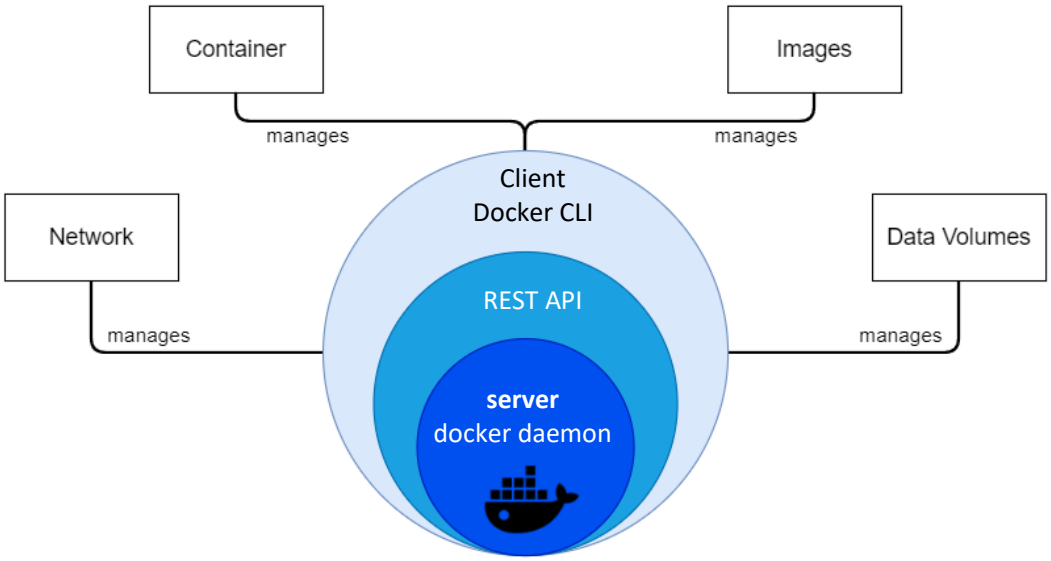
- Οι Developers χρησιμοποιούν το δημόσιο registry docker.io ως κάποιο είδος github για να αποκτήσουν εύκολα έτοιμα προς χρήση containers με application stacks (Java, PHP, Rails…)
- Οι Ops μπορούν να συνεισφέρουν σε αυτά τα containers που είναι έτοιμα για ενδιάμεσο λογισμικό, εφαρμόζοντας μοτίβα εσωτερικής ασφάλειας και ανάπτυξης, εάν χρειάζεται.
- Οι Devs χρησιμοποιούν την τοπική τους παρουσία Docker και τα εσωτερικά registries Docker ως Git για να δημιουργήσουν εφαρμογές έτοιμες για ανάπτυξη.

3.3.3 Docker Daemon

Το Docker daemon λαμβάνει αιτήματα API Docker και διαχειρίζεται Docker objects όπως images, containers, δίκτυα και τόμους. Ένα daemon μπορεί επίσης να επικοινωνήσει με άλλα daemons για να διαχειριστεί τις υπηρεσίες Docker.

3.3.4 Docker Client

Το Docker client είναι ο κύριος τρόπος με τον οποίο πολλοί χρήστες Docker αλληλεπιδρούν με το Docker. Όταν χρησιμοποιούνται εντολές όπως το docker run, το client στέλνει αυτές τις εντολές στο docker, το οποίο τις εκτελεί. Η εντολή docker χρησιμοποιεί το Docker API. Το Docker client μπορεί να επικοινωνήσει με περισσότερους από έναν daemons. Το Docker Engine απεικονίζεται στην Εικόνα 2.



Εικόνα 2 Docker Engine

3.3.5 Image VS Container

Ένα imgae είναι ένα πρότυπο μόνο για ανάγνωση με οδηγίες για τη δημιουργία ενός Docker container. Συχνά, ένα image βασίζεται σε κάποιο άλλο image, με κάποια πρόσθετη προσαρμογή. Για παράδειγμα, γίνεται να δημιουργηθεί ένα image που βασίζεται στην εικόνα του ubuntu, αλλά εγκαθιστά τον διακομιστή ιστού Apache και μία ακόμα εφαρμογή, καθώς και τις λεπτομέρειες διαμόρφωσης που απαιτούνται για να εκτελεστεί αυτή η εφαρμογή.

Το image είναι ένα αδρανές, αμετάβλητο, αρχείο που είναι ουσιαστικά ένα στιγμιότυπο ενός container. Τα images δημιουργούνται με την εντολή build και παράγουν ένα container όταν ξεκινούν με την εκτέλεση. Τα images αποθηκεύονται σε Docker registry όπως το registry.hub.docker.com. Επειδή

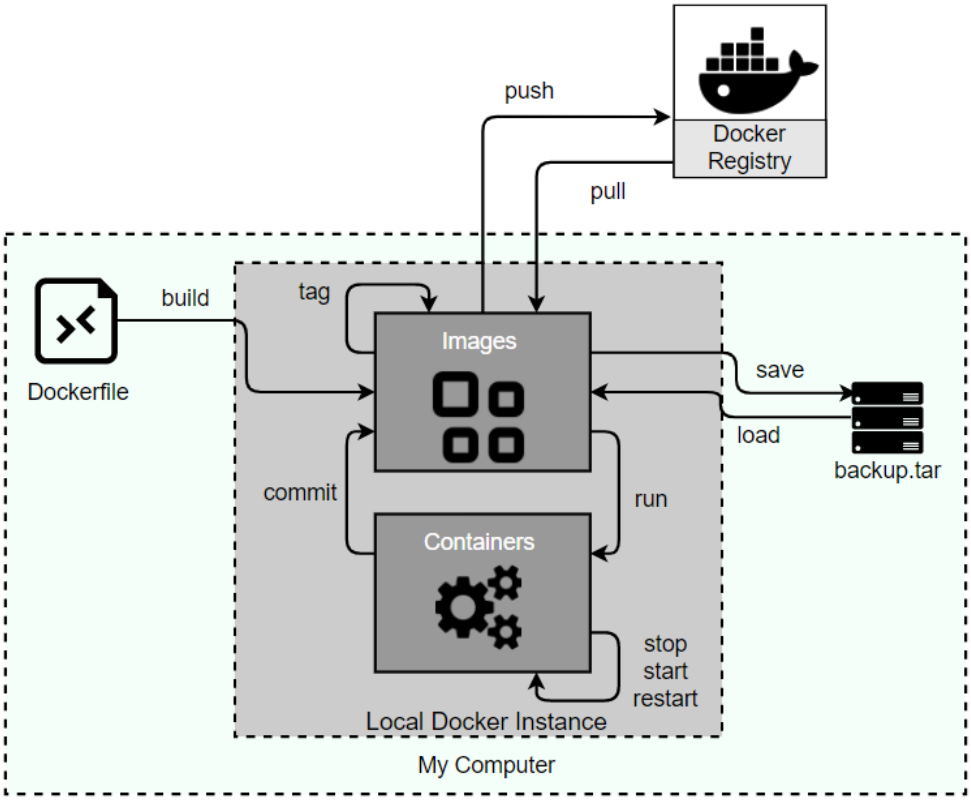
μπορεί να γίνουν αρκετά μεγάλα, τα images έχουν σχεδιαστεί ώστε να αποτελούνται από στρώματα άλλων images, επιτρέποντας την αποστολή ενός ελάχιστου όγκου δεδομένων κατά τη μεταφορά images μέσω του δικτύου.

Η κύρια διαφορά μεταξύ ενός container και ένα image είναι το επάνω εγγράψιμο στρώμα. Τα containers εκτελούν τις πραγματικές εφαρμογές. Ένα container περιλαμβάνει μια εφαρμογή και όλες τις εξαρτήσεις της. Όταν διαγραφεί το container, διαγράφεται και το εγγράψιμο στρώμα. Το υποκείμενο image παραμένει αμετάβλητη.

Από προεπιλογή, ένα container είναι σχετικά καλά απομονωμένο από άλλα containers και το host computer.

Ένα container ορίζεται από το image του καθώς και από τυχόν επιλογές διαμόρφωσης που του προστεθεί κατά τη δημιουργία ή την εκκίνηση. Όταν αφαιρείται ένα container, τυχόν αλλαγές στην κατάστασή του που δεν είναι αποθηκευμένες σε επίμονη αποθήκευση εξαφανίζονται.

Για να χρησιμοποιήσετε μια μεταφορά προγραμματισμού, εάν ένα image είναι μια κλάση, τότε ένα container είναι ένα παράδειγμα μιας κλάσης, δηλαδή ένα αντικείμενο εκτέλεσης. Η διαφορά στο τρόπο λειτουργίας των Images και των Containers στη αρχιτεκτονική του docker φαίνεται στην Εικόνα 3.



Εικόνα 3 Διαφορά Docker Image και Docker Container

3.4 Πλεονεκτήματα του Docker

Τα κύρια πλεονεκτήματα του Docker είναι [8]:

- **Αποδοτικότητα πόρων**: Η απομόνωση σε επίπεδο διαδικασίας και η χρήση του πυρήνα του κεντρικού κεντρικού υπολογιστή είναι πιο αποτελεσματική σε σύγκριση με την εικονικοποίηση ενός ολόκληρου διακομιστή υλικού.
- **Συνεχής ανάπτυξη και δοκιμή**: Η ικανότητα για συνεπή περιβάλλοντα και η ευελιξία με την ενημέρωση κώδικα έχει κάνει το Docker μια εξαιρετική επιλογή για ομάδες που θέλουν να περάσουν από το καταρράκτη στη σύγχρονη προσέγγιση DevOps στην παράδοση λογισμικού.
- **Ταχεία ανάπτυξη εφαρμογής**: Το Docker απαιτεί ελάχιστο χρόνο εκτέλεσης για οποιαδήποτε εφαρμογή, μειώνοντας το μέγεθος και επιτρέποντας γρήγορη ανάπτυξη.
- **Φορητότητα**: Μια εφαρμογή με όλες τα dependencies της μπορεί να ομαδοποιηθεί σε ένα μόνο container, το οποίο είναι ανεξάρτητο από την έκδοση κεντρικού υπολογιστή του πυρήνα Linux, το μοντέλο ανάπτυξης ή τη διανομή πλατφόρμας. Αυτό το container μπορεί απλά να μεταφερθεί σε άλλο μηχάνημα που εκτελεί το Docker και να εκτελεστεί εκεί χωρίς προβλήματα συμβατότητας.
- **Έλεγχος έκδοσης και επαναχρησιμοποίηση dependencies**: Το Docker ξαναχρησιμοποιεί dependencies από τα προηγούμενα layers, γεγονός που τα καθιστά lightweight.
- **Κοινή χρήση**: Είναι δυνατή η χρήση ενός απομακρυσμένου repository για τη χρήση του Docker container με άλλους.
- **Ελαφρύ αποτύπωμα και ελάχιστα έξοδα**: Τα Docker images είναι συνήθως πολύ μικρά, γεγονός που βοηθά στην ταχεία παράδοση και μειώνει το χρόνο ανάπτυξης τυχόν νέων container εφαρμογών.
- **Απλοποιημένη συντήρηση**: Το Docker μειώνει επίσης την προσπάθεια και τον κίνδυνο προβλημάτων που προκαλούνται λόγω των dependencies.

- **Ασφάλεια**: Από την άποψη της ασφάλειας, το Docker διασφαλίζει ότι οι εφαρμογές που εκτελούνται σε container διαχωρίζονται πλήρως μεταξύ τους, παρέχοντάς πλήρη έλεγχο της ροής και της διαχείρισης της κυκλοφορίας. Κανένα container Docker δεν μπορεί να εξετάσει διεργασίες που εκτελούνται μέσα σε άλλο container.

3.5 Εφαρμογές Docker

Οι οργανισμοί χρησιμοποιούν το Docker για την ανάπτυξη εφαρμογών που είναι:

- Αποτελεσματικά βελτιστοποιημένες
- Πολύ κλιμακούμενες
- Φορητές
- Ευκίνητες

Το Docker απλοποιεί τον κύκλο ζωής της ανάπτυξης επιτρέποντας στους προγραμματιστές να εργάζονται σε τυποποιημένα περιβάλλοντα χρησιμοποιώντας τοπικά κοντέινερ που παρέχουν τις εφαρμογές και τις υπηρεσίες σας. Τα container είναι εξαιρετικά για συνεχή ενοποίηση και ροές εργασίας συνεχούς παράδοσης (Continuous integration & Continuous Delivery - CI/CD).

Η φορητότητα και η ελαφριά φύση του Docker διευκολύνουν επίσης τη δυναμική διαχείριση του φόρτου εργασίας, αυξάνοντας ή καταστρέφοντας εφαρμογές και υπηρεσίες, όπως επιβάλλουν οι επιχειρηματικές ανάγκες, σε σχεδόν πραγματικό χρόνο.

Το Docker παρέχει μια βιώσιμη, οικονομικά αποδοτική εναλλακτική λύση σε VMs που βασίζονται σε hypervisor. Το Docker είναι ιδανικό για περιβάλλοντα υψηλής πυκνότητας και για μικρές και μεσαίες εφαρμογές όπου πρέπει να εκτελεστούν περισσότερα πράγματα με λιγότερους πόρους.

3.6 Ασφάλεια Docker και Βέλτιστες Πρακτικές

Τα containers, μαζί με ενορχηστρωτές όπως το Kubernetes, έχουν εισάγει μια νέα εποχή μεθοδολογίας ανάπτυξης εφαρμογών, επιτρέποντας αρχιτεκτονικές μικροϋπηρεσιών καθώς και συνεχή ανάπτυξη και παράδοση. Το Docker είναι μακράν το πιο κυρίαρχο container runtime engine, με ποσοστό 91% σύμφωνα με την τελευταία μας έκθεση για την κατάσταση των containers και του Kubernetes [9].

Το πρόβλημα είναι όμως, ότι ένα μόνο παραβιασμένο container Docker μπορεί να απειλήσει όλα τα άλλα container καθώς και τον υποκείμενο κεντρικό υπολογιστή, υπογραμμίζοντας τη σημασία της ασφάλειας του Docker.

Η ασφάλεια του Docker μπορεί να κατηγοριοποιηθεί σε δύο τομείς: ασφάλεια-security και σκλήρυνση-hardening του κεντρικού υπολογιστή, έτσι ώστε μια παραβίαση σε ένα container να μην οδηγήσει επίσης σε παραβίαση του κεντρικού υπολογιστή και ασφάλεια των άλλων Docker [5].

Οι βέλτιστες πρακτικές για την εξασφάλιση μιας αρχιτεκτονικής που βασίζεται σε Docker έχει σε τρεις βασικούς τομείς [10]:

- Infrastructure
- Images
- Access και Authentication

Πρώτα από όλα, η έκδοση Docker θα πρέπει να είναι πάντα ενημερωμένη. Οι παρωχημένες εκδόσεις είναι επιρρεπείς σε επιθέσεις ασφαλείας. Οι εκδόσεις νέων εκδόσεων περιέχουν συχνά επιδιορθώσεις και διορθώσεις σφαλμάτων που αντιμετωπίζουν τα τρωτά σημεία των παλαιότερων εκδόσεων.

Το ίδιο ισχύει και για το περιβάλλον του κεντρικού υπολογιστή. Θα πρέπει να διασφαλιστεί ότι οι υποστηρικτικές εφαρμογές είναι ενημερωμένες και χωρίς γνωστά σφάλματα ή κενά ασφαλείας.

Ένα εκτεταμένο container περιβάλλον διευρύνει την επιφάνεια επίθεσης και είναι συγκριτικά πιο επιρρεπές σε παραβιάσεις της ασφάλειας από τις αδύνατες ρυθμίσεις. Για την αποφυγεί αυτού, θα διαμορφωθούν τα container

ώστε να περιέχουν μόνο τα απαραίτητα στοιχεία που τα διατηρούν να λειτουργούν, όπως:

- Software packages
- Libraries
- Configuration files

Το Docker Engine χρησιμοποιεί API HTTP για επικοινωνία σε δίκτυο. Τα κακώς διαμορφωμένα API φέρουν ελαττώματα ασφαλείας που μπορούν να εκμεταλλευτούν οι hackers.

Για να την αποφυγή του, στο container θα πρέπει να διαμορφωθεί με ασφάλεια το API που περιορίζει από τη δημόσια έκθεση. Μία προσέγγιση είναι η επιβολή κρυπτογραφημένης επικοινωνίας ενεργοποιώντας τον έλεγχο ταυτότητας βάσει πιστοποιητικών.

Ο περιορισμός πρόσβασης του Docker Daemon σε λίγους βασικούς χρήστες είναι απαραίτητος.

Τα πιστοποιητικά TLS πρέπει να χρησιμοποιούνται για κρυπτογράφηση επικοινωνίας σε επίπεδο κεντρικού υπολογιστή. Είναι επίσης απαραίτητο να είναι απενεργοποιημένες τα αχρησιμοποίητα ports.

Η εφαρμογή φίλτρων για τα system calls που επιτρέπουν τον έλεγχο των κλήσεων που μπορούν να πραγματοποιηθούν μεταξύ του container κοντέινερ και του Linux kernel. Αυτή η προσέγγιση επιτρέπει μια ασφαλή λειτουργία υπολογισμού, μειώνοντας έτσι τα πιθανά σημεία έκθεσης και αποτρέπει την εκμετάλλευση των ευπαθειών του πυρήνα.

Από προεπιλογή, οι διεργασίες εντός των Docker containers έχουν δικαιώματα root που τους παρέχουν root πρόσβαση τόσο στο container όσο και στον κεντρικό υπολογιστή. Αυτό ανοίγει τα containers και τον υποκείμενο κεντρικό υπολογιστή σε ευπάθειες ασφαλείας.

Όσον αφορά τη διαχείριση της αποκατάστασης ευπάθειας (Vulnerability Assessment Process), η ενημέρωση κώδικα λειτουργεί διαφορετικά σε container περιβάλλον. Στα containers, υπάρχουν δύο συστατικά: το base image και το application image. Πρέπει πρώτα να ενημερωθεί το base image και μετά να ξαναφτιαχτεί το application image. Ο καθορισμός μιας σωστής διαδικασίας αξιολόγησης ευπάθειας είναι το κλειδί για τον προσδιορισμό των ευπαθειών [11].

Τα περισσότερα commercial vulnerability scanners προσφέρουν σάρωση container για τον εντοπισμό γνωστών τρωτών σημείων και ζητημάτων εσφαλμένης διαμόρφωσης (misconfiguration). Τα scanners είναι συνήθως σχεδιασμένοι για να ελέγχουν πακέτα λογισμικού που περιλαμβάνονται στο image του container. Μπορούν να διαμορφωθούν ώστε να διασταυρώνουν τα πακέτα με βάσεις δεδομένων ευπάθειας, όπως το National Vulnerability Database (NVD), όπου μπορούν να αναζητήσουν ποια Common Vulnerabilities and Exposures (CVE), εάν υπάρχουν, ισχύουν για αυτό το ακριβές σύνολο πακέτων.

Αυτά τα scanners είναι αυτοματοποιημένα εργαλεία και υπάρχουν αρκετά που διατίθενται προς το παρόν στην αγορά, συμπεριλαμβανομένων προϊόντων εταιρικής ποιότητας και ορισμένων δωρεάν λύσεων. Ο εντοπισμός τρωτών σημείων μπορεί να ακούγεται αρκετά απλός, αλλά μπορεί να υπάρχουν προκλήσεις σε αυτήν τη διαδικασία. Για παράδειγμα, για οποιοδήποτε λειτουργικό σύστημα, το πακέτο Α ενδέχεται να περιλαμβάνει μια ευπάθεια που μπορεί να αξιοποιηθεί μόνο εάν υπάρχει το πακέτο Β ή εάν χρησιμοποιείται ένα συγκεκριμένο πρωτόκολλο δικτύου. Εάν το πακέτο Β δεν υπάρχει ή δεν χρησιμοποιείται συγκεκριμένο πρωτόκολλο δικτύου, είναι πιθανό ότι δεν θα ζητηθεί επιδιόρθωση (patch) για το πακέτο Α. Επίσης, εάν το patch δεν είναι συμβατό με τον κωδικό εφαρμογής, τότε επίσης δεν θα ζητηθεί να εφαρμογή του [11].

Πολλά Linux distribution backport fixes είναι διαθέσιμα για την ενημέρωση παλαιότερων εκδόσεων πακέτων, αντί να τις αντικαθιστούν. Τα τυπικά scanners που λειτουργούν σαν version-dependent εμφανίζουν ευπάθειες με βάση την έκδοση των εγκατεστημένων πακέτων. Κατά τη σάρωση, εάν βρεθεί ένα παρωχημένο (obsolete) πακέτο, θα επισημάνει την ευπάθεια. Η εμπιστοσύνη μόνο στα αποτελέσματα του scanner μπορεί να οδηγήσει σε πολλά false-positives. Αυτός είναι ο λόγος για τον οποίο είναι σημαντική η χειροκίνητη επαλήθευση των ευπαθειών.

Η ικανότητα αποτελεσματικής διαχείρισης των false-positive αποτελεί βασικό διαφοροποιητή μεταξύ των scanners. Η χρήση image container και η αναδημιουργία νέων images, όποτε απαιτείται αλλαγή κώδικα, μπορεί να βελτιώσει αποτελεσματικά τη διαδικασία επιδιόρθωσης. Ταυτόχρονα,

συμβάλλει στη μείωση του αριθμού των τρωτών σημείων για τη βελτίωση της συνολικής ανθεκτικότητας.

Θα πρέπει να εφαρμοστεί Centrally Managed Access Control C-MAC στις αλλαγές ή εντολές που μπορεί να εκτελέσει ένας χρήστης βάσει του ρόλου του και όχι των δυνατοτήτων του. Αυτό μπορεί να βοηθήσει στον καθορισμό και την επιβολή του κατάλληλου ελέγχου πρόσβασης σε ενεργά containers.

Δεδομένου ότι τα containers διαχειρίζονται έναν δυναμικό και ευέλικτο κύκλο ανάπτυξης, τα πράγματα κινούνται γρήγορα, καθιστώντας σημαντικό να παρακολουθείται ποιος έκανε αλλαγές στις ρυθμίσεις διαμόρφωσης και ποιος ξεκίνησε ή έκλεισε τα containers. Με τους χρήστες να έχουν πρόσβαση σε container ως root user, ο προσδιορισμός του ποιος έκανε αλλαγές στη διαμόρφωση είναι σχεδόν αδύνατος. Μερικές φορές, η παροχή πρόσβασης root μπορεί να είναι ο ευκολότερος τρόπος για τους προγραμματιστές να κάνουν τη δουλειά τους, αλλά στη συνέχεια καταλήγουν να έχουν ή να έχουν υπερβολική πρόσβαση.

Τα υπερβολικά επίπεδα πρόσβασης μπορούν να δημιουργήσουν προβλήματα όταν οι ομάδες θέλουν να ελέγξουν ένα ζήτημα εντοπίζοντας τι συνέβη και πότε. Επιπλέον, εάν ένας εισβολέας αποκτήσει πρόσβαση root, μπορεί να αποκτήσει πρόσβαση σε όλα τα containers και να αυξήσει τον αρνητικό αντίκτυπο της επίθεσης.

Υπάρχουν πολλά παραδείγματα ανίχνευσης απειλών στις μέρες μας, συμπεριλαμβανομένης του behavioral baseline, ενός μηχανισμού που επικεντρώνεται στην κατανόηση της τυπικής συμπεριφοράς μιας εφαρμογής ή συστήματος για τον εντοπισμό ανωμαλιών. Η βασική συμπεριφορά περιλαμβάνει τη δημιουργία ενός συστήματος ή μιας βάσης εφαρμογής, συνεχή παρακολούθηση των βασικών διαμορφώσεων και ανίχνευση και απόκριση σε τυχόν αλλαγές στις βασικές διαμορφώσεις. Το active response είναι ένας καλός τρόπος για την αντιμετώπιση μιας επίθεσης, συμβιβασμού ή ανωμαλίας μόλις εντοπιστεί.

Τα responses μπορούν επίσης να έχουν πολλές διαφορετικές μορφές, όπως:

- ειδοποίηση υπεύθυνου προσωπικού.
- επικοινωνία με ticketing system.

- εφαρμογή προκαθορισμένων διορθωτικών ενεργειών σε συστήματα και εφαρμογές.

Σε περιβάλλον container, ένα response θα μπορούσε να σημαίνει την εφαρμογή επιπλέον Logging, την εφαρμογή πρόσθετων κανόνων απομόνωσης, την απενεργοποίηση ενός χρήστη δυναμικά ή ακόμα και την ενεργή διαγραφή του container.

Είναι κρίσιμο να εφαρμοστούν συγκεκριμένες διαδικασίες για τον έλεγχο όλων των απαντήσεων σε περιστατικά που συμβαίνουν σε container περιβάλλοντα.

Εάν κάποιος εισβολέας θέσει σε κίνδυνο το σύστημα ενός κεντρικού υπολογιστής, η απομόνωση του container και οι εγγυήσεις ασφαλείας δεν θα κάνουν μεγάλη διαφορά. Εκτός αυτού, τα container λειτουργούν πάνω από τον kerbel του host από σχεδίαση [12]. Αν εστιάσουμε μόνο στο πλαίσιο Docker:

- Πρέπει να είναι σίγουρο πως η διαμόρφωση του κεντρικού υπολογιστή και του Docker είναι ασφαλής. Για παράδειγμα η περιορισμένη και πιστοποιημένη πρόσβαση, η κρυπτογραφημένη επικοινωνία και άλλα. Συνιστάται η χρήση ενός audit tool για τον έλεγχο του Docker.
- Η διατήρηση του βασικού συστήματος ενημερωμένο.
- Η χρήση minimal, container-center συστημάτων που βασίζονται σε containers, όπως το CoreOS, το Red Hat Atomic και το RancherOS, θα μειώσει την πιθανότητα επίθεσης.
- Μπορεί να πραγματοποιηθεί ο υποχρεωτικό έλεγχο πρόσβασης - Mandatory Access Control για να την αποτροπή ανεπιθύμητων ενεργειων, τόσο στον host υπολογιστή όσο και στα container, σε επίπεδο kernel χρησιμοποιώντας εργαλεία όπως το Seccomp, το AppArmor ή το SELinux.

Ο όρος "breakout container" χρησιμοποιείται για να δηλώσει ότι το container Docker έχει παρακάμψει ελέγχους απομόνωσης, πρόσβαση σε ευαίσθητες πληροφορίες από τον host υπολογιστή ή απόκτηση πρόσθετων προνομίων. Για να αποφευχθεί αυτό, πρέπει να μειωθούν τα προεπιλεγμένα δικαιώματα container [12]. Για παράδειγμα, το daemon Docker εκτελείται ως root από προεπιλογή, αλλά γίνεται να δημιουργηθεί ένα namespace σε επίπεδο

χρήστη ή να αφαιρεθούν (drop) μερικές από τις δυνατότητες container root. Για την αποφυγή του breakout προτείνεται:

- Αφαίρεση-drop δυνατοτήτων είναι ο λεπτομερής έλεγχος πρόσβασης πέρα από το root, που δεν απαιτούνται από το λογισμικό σας.
    - Το CAP_SYS_ADMIN είναι ιδιαίτερα από την άποψη της ασφάλειας γιατί παρέχει ένα ευρύ φάσμα δικαιωμάτων επιπέδου root: τοποθέτηση συστημάτων αρχείων, εισαγωγή kernel namespaces, λειτουργίες ioct και άλλα
- Δημιουργία ενός απομονωμένου user namespace για να τον περιορισμό των μέγιστων προνομίων των container έναντι του host υπολογιστή. Αποφυγή της χρήσης container ως uid 0.
- Εάν πρέπει να τρέξει ένα privileged container, πρέπει να γίνει έλεγχος ότι προέρχεται από αξιόπιστη πηγή.
- Παρακολούθηση επικίνδυνων σημείων αναφοράς από το host υπολογιστή: την υποδοχή Docker (/var/run/docker.sock),/proc,/dev, κ.λπ. Συνήθως, αυτές οι ειδικές βάσεις απαιτούνται για την εκτέλεση της βασικής λειτουργικότητας του container. Μερικές φορές απλώς η έκθεση του συστήματος αρχείων με δικαιώματα μόνο για ανάγνωση θα πρέπει να είναι αρκετή. Δε θα πρέπει να δοθεί πρόσβαση σε εγγραφή. Σε κάθε περίπτωση, το Docker κάνει αντιγραφή-εγγραφή για να αποτρέψει αλλαγές σε ένα τρέχον container που επηρεάζουν το βασικό image, το οποίο μπορεί να χρησιμοποιηθεί για άλλο container.

Τα container είναι πολύ περισσότερα από τα VMs κατά μέσο όρο, είναι ελαφριά και γίνεται να δημιουργηθούν μεγάλες ομάδες από αυτά σε μέτριο υλικό/hardware. Αυτό είναι σίγουρα ένα πλεονέκτημα, αλλά υπονοεί ότι πολλές οντότητες λογισμικού ανταγωνίζονται για τους πόρους του host. Σφάλματα λογισμικού, λανθασμένοι υπολογισμοί σχεδιασμού ή σκόπιμη επίθεση κακόβουλου λογισμικού μπορούν εύκολα να προκαλέσουν άρνηση υπηρεσίας (DoS) εάν δεν διαμορφωθούν σωστά τα όρια πόρων.

Το πρόβλημα μεγαλώνει όταν υπάρχουν αρκετοί διαφορετικοί πόροι που πρέπει να προστατευθούν: CPU, κύρια μνήμη, χωρητικότητα αποθήκευσης, εύρος ζώνης δικτύου, εύρος ζώνης εισόδου/εξόδου και άλλα.

Υπάρχουν ορισμένοι πόροι Kernel που δεν είναι τόσο εμφανείς καθώς και τα αναγνωριστικά χρήστη (UIDs).

Τα όρια σε αυτούς τους πόρους απενεργοποιούνται από προεπιλογή στα περισσότερα συστήματα, η διαμόρφωσή τους πριν από την ανάπτυξη στην παραγωγή είναι βασικά απαραίτητη. Υπάρχουν τρία βασικά βήματα:

1. Χρήση των δυνατοτήτων περιορισμού πόρων που συνοδεύουν τον Linux Kernel και/ή το container.
2. Προσπάθεια να επανάληψης των φορτίων παραγωγής στην προπαραγωγή. Μερικοί άνθρωποι χρησιμοποιούν συνθετικό stress test, άλλοι επιλέγουν να «επαναλάβουν» την πραγματική κυκλοφορία παραγωγής σε πραγματικό χρόνο. Ο έλεγχος φορτίου είναι ζωτικής σημασίας για να την αναγνώριση των φυσικών ορίων που είναι το φυσιολογικό εύρος των λειτουργιών.
3. Εφαρμογή παρακολούθησης και προειδοποίησης (monitoring and alerting) Docker.

Το λογισμικό χρειάζεται ευαίσθητες πληροφορίες για να λειτουργήσει όπως τα hashes των κωδικών πρόσβασης χρήστη, πιστοποιητικά από server, κλειδιά κρυπτογράφησης και άλλα. Χρειάζεται μια αυτόματη και ασφαλή διαδικασία για να το μοίρασμα αυτών των ευαίσθητων πληροφοριών.

- Δε πρέπει να γίνεται χρήση μεταβλητών περιβάλλοντος για μυστικά, αυτή είναι μια πολύ συνηθισμένη αλλά πολύ ανασφαλής πρακτική.
- Δε πρέπει να ενσωματωθεί κανένα μυστικό στο container image.
- Χρήση ενός Docker credentials management software.

Τα containers είναι απομονωμένα “μαύρα κουτιά”, εάν κάνουν τη δουλειά τους όπως αναμενόταν, είναι εύκολο να ξεχαστεί ποιο λογισμικό και έκδοση τρέχει συγκεκριμένα μέσα. ίσως ένα container να λειτουργεί τέλεια από λειτουργική άποψη, αλλά εκτελεί την έκδοση X του server ιστού, η οποία τυχαίνει να έχει ένα κρίσιμο ελάττωμα ασφαλείας. Αυτό το ελάττωμα έχει διορθωθεί πολύ καιρό πριν, αλλά όχι στην τοπικό image. Αυτό το είδος προβλήματος μπορεί να περάσει απαρατήρητο για μεγάλο χρονικό διάστημα εάν δεν ληφθούν τα κατάλληλα μέτρα.

Η απεικόνιση των containers ως αμετάβλητων ατομικών μονάδων είναι πολύ ωραία για το σχεδιασμό της αρχιτεκτονικής, ωστόσο, από την άποψη της ασφάλειας, πρέπει να επιθεωρεί τακτικά το περιεχόμενό τους:

- Ενημέρωση και ανακατασκευή των images περιοδικά για την απόκτηση των νεότερων επιδιορθωμάτνω ασφαλείας,.
    - Το Live-patching των containers θεωρείται κακή πρακτική, το πρότυπο είναι να ξαναχτιστεί ολόκληρο το image με κάθε ενημέρωση.
    - Χρήση λογισμικού από έναν διανομέα που εγγυάται ενημερώσεις ασφαλείας και οτιδήποτε εγκατασταθεί με μη αυτόματο τρόπο από τη διανομή, θα πρέπει να γίνεται ενημέρωση του κώδικα ασφαλείας χειροκίνητα.
    - Οι προσεγγίσεις που βασίζονται σε Docker και μικροϋπηρεσίες θεωρούν τη σταδιακή αναβάθμιση των ενημερώσεων χωρίς να διαταράσσεται ο χρόνος λειτουργίας ως βασική προϋπόθεση του μοντέλου.
    - Τα δεδομένα χρήστη διαχωρίζονται σαφώς από τα images, καθιστώντας όλη αυτή τη διαδικασία ασφαλέστερη.
- Keep it simple. Τα ελάχιστα συστήματα αναμένουν λιγότερο συχνές ενημερώσεις.
    - Χρήση σαρωτή ευπάθειας. Υπάρχουν πολλά, τόσο δωρεάν όσο και εμπορικά. Ενσωμάτωση αυτού του σαρωτή ευπάθειας ως υποχρεωτικό βήμα του CI/CD (Continuous Integration/Continuous Development). Αυτοματοποίηση όπου είναι δυνατόν.

Οι διαδικασίες συνεχούς ολοκλήρωσης (Continuous Integration processes) αντιμετωπίζουν σημαντικές εγγενείς προκλήσεις στον κυβερνοχώρο. Η ιδέα δεν είναι μόνο η επίλυση και ο έλεγχος τυπικών ή κανονιστικών απαιτήσεων ασφάλειας του πηγαίου κώδικα/source code, αλλά και η τήρηση των ίδιων αρχών στο ίδιο το CI pipeline [13].

Το CI είναι μία από τις πολλές πρακτικές ανάπτυξης λογισμικού που αποσκοπεί στο να βοηθήσει τους οργανισμούς να επιταχύνουν την ανάπτυξη και την παροχή λειτουργιών λογισμικού χωρίς συμβιβασμούς στην ποιότητα.

Σύμφωνα με τους Fitzgerald και Stol [14] μπορεί να οριστεί ως “μια διαδικασία που συνήθως ενεργοποιείται αυτόματα και περιλαμβάνει αλληλοσυνδεόμενα βήματα όπως η σύνταξη κώδικα, η εκτέλεση μονάδων και οι δοκιμές αποδοχής, η επικύρωση κάλυψης κώδικα, ο έλεγχος της συμμόρφωσης με τα πρότυπα κωδικοποίησης και η κατασκευή πακέτων ανάπτυξης”.]

Σύμφωνα με μία προσέγγιση [15], κάθε ροή εργασίας CI πρέπει να περιλαμβάνει τα ακόλουθα 6 στοιχεία:

- Automation Server. Εφαρμόζει το CI/CD (Continuous Integration/Continuous Development) pipeline και δημιουργεί έναν τοπικό χώρο εργασίας στον οποίο πραγματοποιούνται τα βήματά του.
- Orchestrator. Ενεργοποιεί διαδοχικά κάθε βήμα του pipeline επικοινωνώντας με τα υπόλοιπα εξαρτήματα.
- Code Retriever. Μεταφέρει τον πηγαίο κώδικα από το αποθετήριο στον τοπικό χώρο εργασίας.
- Unit Tester. Εκτελεί αυτοματοποιημένες δοκιμές μονάδας στον πηγαίο κώδικα.
- Artifact builder. Δημιουργεί επεξεργασμένα artifacts από τον πηγαίο κώδικα.
- Image Generator. Δημιουργεί, επαληθεύει, αποθηκεύει και αναπτύσσει ένα image που θα χρησιμοποιηθεί εντός του pipeline.

Συνοπτικά, ακολουθεί μια λίστα με τις βέλτιστες πρακτικές που προέρχονται από τα βιομηχανικά πρότυπα για την ασφαλή διαμόρφωση των containers και των images του Docker.

1. Χρήση πάντα της πιο ενημερωμένης έκδοσης του Docker [16]. Η ευπάθεια του runC, για παράδειγμα, επιδιορθώθηκε γρήγορα μετά την ανακάλυψή της με την κυκλοφορία της έκδοσης Docker 18.09.2.
2. Να επιτρέπεται ο έλεγχος του Docker daemon μόνο από έμπιστους/trusted χρήστες, διασφαλίζοντας ότι μόνο οι αυτοί είναι μέλη της ομάδας Docker [16].
3. Πρέπει να έχουν θεσπιστεί κανόνες που επιτρέπουν ένα audit trail (διαδρομή ελέγχου) για:

a. Docker daemon
b. Docker files και directories

4. Χρήση registries που διαθέτουν έγκυρο πιστοποιητικό ή αυτά που χρησιμοποιούν TLS για να την ελαχιστοποίηση του κινδύνου υποκλοπής κυκλοφορίας.
5. Χρήση namespaces.
6. Να απαγορεύονται τα containers από την απόκτηση νέων προνομίων/permissions. Από προεπιλογή, επιτρέπεται στα containers να αποκτήσουν νέα προνόμια, οπότε αυτή η διαμόρφωση πρέπει να οριστεί ρητά. Ένα άλλο βήμα είναι η κατάργηση δικαιωμάτων setuid και setgid στα images.
7. Εκτέλεση του container ως μη root χρήστης (UID όχι 0). Από προεπιλογή, τα containers εκτελούνται με δικαιώματα root [18].
8. Χρήση μόνο αξιόπιστων images κατά την κατασκευή των containers.
9. Θα πρέπει να γίνεται χρήση minimal base images που δεν περιλαμβάνουν περιττά πακέτα λογισμικού που θα μπορούσαν να οδηγήσουν σε μεγαλύτερο attack vector.
10. Εφαρμογή μιας ισχυρής πολιτικής που επιβάλλει τη συχνή σάρωση των images. Τα παλία images ή τα images που δεν έχουν σαρωθεί πρόσφατα θα πρέπει να απορριφθούν ή να επανασαρμοστούν πριν από τη μετάβαση στο στάδιο κατασκευής.
11. Δημιουργία μιας ροής εργασίας που τακτικά προσδιορίζει και αφαιρεί παλιά ή μη χρησιμοποιημένα images και containers από τον κεντρικό υπολογιστή.
12. Όταν εκτελείται ένα container, θα πρέπει να αφαιρούνται όλες οι δυνατότητες που δεν απαιτούνται για τη λειτουργία του.
13. Δε θα πρέπει να γίνει **ποτέ εκτέλεση** ενός container με **--privileged flag**, καθώς αυτός ο τύπος container θα έχει τις περισσότερες από τις δυνατότητες που διαθέτει ο κεντρικός υπολογιστής.
14. Δε θα πρέπει να γίνεται χρήση της εντολής docker exec ως privileged ή ως επιλογή user=root, καθώς αυτή η ρύθμιση θα μπορούσε να δώσει στο container extender δυνατότητες Linux.
15. Απαγορεύεται η εκτέλεση sshd μέσα σε container. Από προεπιλογή, το ssh daemon δεν θα εκτελείται σε ένα κοντέινερ και δεν πρέπει να

εγκατασταθεί το ssh για να την απλοποίηση της διαχείρισης ασφάλειας του SSH server.

16. Κανένα port κάτω από το 1024 δε θα πρέπει να γίνει map σε ένα container καθώς θεωρούνται privileged επειδή μεταδίδουν ευαίσθητα δεδομένα. Από προεπιλογή, το Docker κάνει map τα ports σε ένα που είναι εντός του εύρους 49153-65525. Κατά γενικό κανόνα, ότι μόνο τα απαραίτητα ports είναι ανοιχτα στο container [5].
17. Το root filesystem του container θα πρέπει να οριστεί σε read-only. Μόλις εκτελεστούν, τα containers δεν χρειάζονται αλλαγές στο root filesystem.
18. Τα namespaces (network, process, IPC, user, UTS) του host δε θα πρέπει να μοιράζονται χώρο μεταξύ τους, για να διασφαλιστεί η σωστή απομόνωση μεταξύ των Docker container και του κεντρικού υπολογιστή.
19. Επιβολή ορίου PID. Ένα από τα πλεονεκτήματα των containers είναι ο αυστηρός έλεγχος αναγνώρισης διαδικασίας (PID). Κάθε διαδικασία στον πυρήνα φέρει ένα μοναδικό PID και τα containers αξιοποιούν το Linux PID namespace για να παρέχουν ξεχωριστή προβολή της ιεραρχίας PID για κάθε container. Η θέσπιση ορίων στα PID περιορίζει αποτελεσματικά τον αριθμό των διαδικασιών που εκτελούνται σε κάθε container.
20. Το bridge "docker0" δε θα πρέπει να χρησιμοποιείται. Η χρήση της προεπιλεγμένης γέφυρας αφήνει ανοιχτό το σύστημα σε επιθέσεις ARP soofind και MAC flooding. Αντίθετα, τα containers πρέπει να βρίσκονται σε ένα δίκτυο καθορισμένο από τον χρήστη και όχι στο "docker0".
21. Δε θα πρέπει να γίνει mount το Docker socket μέσα σε ένα container, καθώς αυτή η προσέγγιση θα επιτρέψει σε μια διαδικασία εντός του container να εκτελέσει εντολές που του δίνουν τον πλήρη έλεγχο του κεντρικού υπολογιστή [18].

3.6.1 Ασφάλεια Kubernetes

Ως το de facto πρότυπο για ενορχήστρωση των containers, το Kubernetes διαδραματίζει κεντρικό ρόλο στη διασφάλιση της ασφάλειας των

εφαρμογών. Για την αποτελεσματική εξασφάλιση των εφαρμογών που βρίσκονται σε containers, πρέπει να αξιοποιηθούν οι πληροφορίες σχετικά με τα συμφραζόμενα από το Kubernetes, καθώς και τις εγγενείς δυνατότητες επιβολής πολιτικής [5].

Για παράδειγμα, το Kubernetes διαθέτει πολλές ενσωματωμένες δυνατότητες ασφαλείας που διευκολύνουν τη λειτουργία της ασφάλειας του πλήρους κύκλου ζωής, συμπεριλαμβανομένων των Kubernetes RBAC, Πολιτικών δικτύου και Ελεγκτών Εισόδου.

Παρακάτω είναι μερικές βέλτιστες πρακτικές ασφάλειας της Kubernetes που βοηθούν στη λειτουργία της ασφάλειας του πλήρους κύκλου ζωής ενός container:

1. Για το RBAC, θα πρέπει να γίνει καθορισμός στους Ρόλους και τα Συμπλέγματα Ρόλων σε συγκεκριμένους χρήστες ή ομάδες χρηστών.
2. Αποφυγή της επικάλυψης δικαιωμάτων κατά τη χρήση του Kubernetes RBAC, καθώς αυτό θα μπορούσε να δημιουργήσει λειτουργικά ζητήματα.
3. Κατάργηση των αχρησιμοποίητων ή των ανενεργών ρόλων ρόλους RBAC, ώστε να γίνει εστίαση της προσοχής ενεργούς ρόλους κατά την αντιμετώπιση προβλημάτων ή τη διερεύνηση περιστατικών ασφαλείας.
4. Χρήση των πολιτικών δικτύου Kubernetes για να την απομόνωση των pods.
5. Εάν τα pods χρειάζονται πρόσβαση στο Διαδίκτυο (είτε εισέρχονται είτε εξέρχονται), τότε θα πρέπει να δημιουργηθεί κατάλληλη πολιτική δικτύου που θα επιβάλλει τον σωστό κανόνα segmentation/firewalling.
6. Χρήση του ελεγκτή εισόδου PodSecurityPolicy για να την διασφάλιση της εφαρμογής των σωστών πολιτικών Ο ελεγκτής PodSecurityPolicy μπορεί να αποτρέψει τη λειτουργία των containers ως root ή να βεβαιωθεί ότι το root filesystem ενός container είναι read-only.
7. Χρήση του ελεγκτή εισόδου Kubernetes για να την επιβολή πολιτικών image registry, έτσι ώστε όλα τα images λαμβάνονται από μη αξιόπιστα registries να απορρίπτονται αυτόματα.

Κεφάλαιο 4. Ανάλυση Εννοιών και Μεθοδολογιών

4.1 Teaming

Σε μια εταιρία ή έναν οργανισμό, την ασφάλεια των assets, την αναλαμβάνει το blue team. Η εργασία σχετίζεται με το hardening των docker containers, κάτι που θα ήταν στις υποχρεώσεις του blue team σε έναν οργανισμό. Για αυτό το λόγο, σε αυτή την υποενότητα θα γίνει περιγραφή του blue team, καθώς και των red και purple teams.

4.1.1 Blue Teaming

Ένα blue team αποτελείται από επαγγελματίες ασφαλείας που έχουν εσωτερική και εξωτερική άποψη του οργανισμού. Ο στόχος τους είναι να προστατεύσουν τα κρίσιμα περιουσιακά στοιχεία - assets του οργανισμού από κάθε είδους απειλή. Γνωρίζουν καλά τους επιχειρηματικούς στόχους και τη στρατηγική ασφάλειας του οργανισμού. Ως εκ τούτου, το καθήκον τους είναι να ενισχύσουν την άμυνα, ώστε κανένας εισβολέας να μην μπορεί να θέσει σε κίνδυνο τον οργανισμό.

Το blue team θα εντοπίσει και θα εξουδετερώσει τις πιο εξελιγμένες επιθέσεις και θα παρακολουθήσει στενά τις τρέχουσες και τις αναδυόμενες απειλές για να υπερασπιστεί προληπτικά τον οργανισμό.

Το blue team συλλέγει πρώτα δεδομένα, τεκμηριώνει τι ακριβώς πρέπει να προστατευτεί και πραγματοποιεί εκτίμηση κινδύνου. Στη συνέχεια, ενισχύουν την πρόσβαση στο σύστημα με πολλούς τρόπους, συμπεριλαμβανομένης της εισαγωγής ισχυρότερων πολιτικών κωδικού πρόσβασης και εκπαίδευσης του προσωπικού για να διασφαλίσει ότι κατανοεί και συμμορφώνεται με τις διαδικασίες ασφαλείας.

Συχνά δημιουργούνται εργαλεία παρακολούθησης, τα οποία επιτρέπουν την καταγραφή και έλεγχο πληροφοριών σχετικά με την πρόσβαση στα συστήματα για ασυνήθιστες δραστηριότητες. Τα blue teams θα πραγματοποιούν τακτικούς ελέγχους στο σύστημα, για παράδειγμα, ελέγχους

DNS, σαρώσεις ευπάθειας εσωτερικού ή εξωτερικού δικτύου και καταγραφή δείγματος κίνησης δικτύου για ανάλυση.

Τα blue teams πρέπει να θεσπίσουν μέτρα ασφαλείας γύρω από τα βασικά περιουσιακά στοιχεία ενός οργανισμού. Ξεκινούν το αμυντικό τους σχέδιο προσδιορίζοντας τα κρίσιμα περιουσιακά στοιχεία, τεκμηριώνουν τη σημασία αυτών των περιουσιακών στοιχείων για την επιχείρηση και τι αντίκτυπο θα έχει η απουσία αυτών των περιουσιακών στοιχείων [19].

Στη συνέχεια, τα blue teams πραγματοποιούν αξιολογήσεις κινδύνου εντοπίζοντας απειλές κατά κάθε περιουσιακού στοιχείου και τις αδυναμίες που μπορούν να εκμεταλλευτούν αυτές οι απειλές. Με την αξιολόγηση των κινδύνων και την ιεράρχησή τους, το blue team αναπτύσσει ένα σχέδιο δράσης για την εφαρμογή ελέγχων που μπορούν να μειώσουν τον αντίκτυπο ή την πιθανότητα υλοποίησης απειλών έναντι περιουσιακών στοιχείων.

Η συμμετοχή των ανώτερων διευθυντικών στελεχών είναι ζωτικής σημασίας σε αυτό το στάδιο, καθώς μόνο αυτοί μπορούν να αποφασίσουν να αποδεχτούν έναν κίνδυνο ή να εφαρμόσουν ελαφρυντικούς ελέγχους εναντίον του. Η επιλογή των ελέγχων βασίζεται συχνά σε μια ανάλυση κόστους-οφέλους για να διασφαλιστεί ότι οι έλεγχοι ασφαλείας αποδίδουν τη μέγιστη αξία στην επιχείρηση [20].

Οι στόχοι και τα καθήκοντα του blue team περιλαμβάνουν:

- Κατανόηση κάθε φάσης ενός περιστατικού και κατάλληλη ανταπόκριση.
- Παρατήρηση ύποπτων μοτίβων κίνησης και προσδιορισμός δεικτών συμβιβασμού.
- Άμεση επιδιόρθωση κάθε μορφής παραβίασης/compromise.
- Εντοπισμός των command and control servers (C&C or C2) των red teams (C&C ή C2) και αποκλεισμός της σύνδεσης τους με τον στόχο.
- Ανάλυση και εγκληματολογικές (forensics) δοκιμές στα διάφορα λειτουργικά συστήματα που εκτελεί ο οργανισμός τους.

Οι μέθοδοι του blue team περιλαμβάνουν:

- Επανεξέταση και ανάλυση δεδομένων καταγραφής - log data.

- Χρήση μιας πλατφόρμας πληροφοριών ασφάλειας και διαχείρισης συμβάντων (security information and event management - SIEM) για ορατότητα και ανίχνευση ζωντανών εισβολών και για τη δοκιμή συναγερμών σε πραγματικό χρόνο.
- Συλλογή νέων πληροφοριών για νέες απειλές και ιεράρχηση των κατάλληλων ενεργειών στο πλαίσιο των κινδύνων.
- Εκτέλεση ανάλυσης επισκεψιμότητας και ροής δεδομένων.
- Βεβαίωση ότι τα στοιχεία ελέγχου πρόσβασης firewall έχουν διαμορφωθεί σωστά και ότι το λογισμικό προστασίας από ιούς είναι ενημερωμένο
- Εφαρμογή λογισμικού IDS και IPS ως εντοπισμού και προληπτικού ελέγχου ασφαλείας.
- Διαχωρισμός δικτύων και βεβαίωση ότι έχουν διαμορφωθεί σωστά.
- Χρήση λογισμικού σάρωσης ευπάθειας σε τακτά χρονικά διαστήματα.
- Ασφάλιση συστημάτων χρησιμοποιώντας λογισμικό προστασίας από ιούς ή λογισμικό προστασίας από κακόβουλο λογισμικό.
- Εκτέλεση ελέγχων DNS για την πρόληψη επιθέσεων ηλεκτρονικού ψαρέματος (phishing), την αποφυγή παρωχημένων προβλημάτων DNS, την αποφυγή διακοπής λειτουργίας από διαγραφές εγγραφών DNS και την πρόληψη/μείωση των επιθέσεων DNS και web attacks.
- Εγκατάσταση λογισμικού ασφαλείας endpoint σε εξωτερικές συσκευές όπως φορητούς υπολογιστές και smartphone.

4.1.2 Red Teaming

Ένα red team είναι συνήθως ανεξάρτητο από την εταιρεία-στόχος και προσλαμβάνεται για να δοκιμάσει τις άμυνές της. Η ομάδα αποτελείται από εξειδικευμένους ηθικούς hacker, στόχος των οποίων είναι να εντοπίσουν και να εκμεταλλευτούν με ασφάλεια τρωτά σημεία στον κυβερνοασφάλεια ή τις φυσικές περιμέτρους του στόχου.

Το red team αναπτύσσει εργαλεία και τεχνικές hacking αιχμής που έχουν σχεδιαστεί για να διεισδύουν σε συστήματα και χώρους [19,20].

Οι στόχοι και τα καθήκοντα του red team περιλαμβάνουν:

- Παραβίαση της ασφάλειας του στόχου με εξαγωγή πληροφοριών, διείσδυση στα συστήματά του ή παραβίαση των φυσικών περιμέτρων του.
- Αποφυγή εντοπισμού από το blue team.
- Εκμετάλλευση σφαλμάτων και αδυναμιών στην υποδομή του στόχου. Αυτό τονίζει τα κενά στην τεχνική ασφάλεια του οργανισμού που απαιτούν διόρθωση, βελτιώνοντας έτσι τη στάση ασφαλείας του.
- Έναρξη ενός sophisticated penetration testing για την κατανόηση των αμυντικών δυνατοτήτων του blue team.

Οι μέθοδοι του red team περιλαμβάνουν:

- Αρχική αναγνώριση/reconnaissance - open source intelligence OSINT για τη συλλογή πληροφοριών σχετικά με τον στόχο.
- Penetration testing, γνωστό και ως ethical hacking.
- Το social engineering για να ξεγελάσει τα μέλη του προσωπικού ώστε να αποκαλύψουν τα credentials τους ή να επιτρέψουν την πρόσβαση σε μια περιορισμένη περιοχή.
- Ανάπτυξη command-and-control servers (C&C ή C2) για τη δημιουργία επικοινωνίας με το δίκτυο του στόχου.
- Χρήση δολώματος για να μπερδέψουν το blue team.
- Εφαρμογή τεχνικών social engineering και phishing για χειραγώγηση των εργαζομένων ώστε να εκθέσουν ή να αποκαλύψουν πληροφορίες για να θέσουν σε κίνδυνο τους υπολογιστές.
- Δοκιμές φυσικής και ψηφιακής διείσδυσης.
- Κλωνοποίηση της κάρτας ασφαλείας ενός υπαλλήλου για την παροχή πρόσβασης σε περιοχές περιορισμένης πρόσβασης, όπως αίθουσα των servers.
- Phishing – ηλεκτρονικό ψάρεμα.

4.1.3 Purple Teaming

Ένα purple team δεν είναι μόνιμο και έχει ως υποχρέωση να επιβλέπει και να βελτιστοποιεί την άσκηση του red και του blue team. Συνήθως αποτελείται από αναλυτές ασφαλείας ή ανώτερο προσωπικό ασφαλείας εντός του οργανισμού.

Εάν τα red και blue teams λειτουργούν καλά, ένα purple team μπορεί να γίνει περιττό. Μπορεί να είναι περισσότερο μια έννοια παρά μια λειτουργία, οδηγώντας το red team να δοκιμάσει και να στοχεύσει συγκεκριμένα στοιχεία των αμυντικών και ανιχνευτικών δυνατοτήτων του blue team [19,20].

4.2 Security Information and Event Management - SIEM

Η τεχνολογία SIEM, η οποία υπήρχε από τα μέσα της δεκαετίας του 2000, πρωτοεμφανίστηκε από την διαχείρισης αρχείων καταγραφής, τις συλλογικές διαδικασίες και τις πολιτικές που χρησιμοποιούνται για τη διαχείριση και τη διευκόλυνση της δημιουργίας, μετάδοσης, ανάλυσης, αποθήκευσης, αρχειοθέτησης και διάθεσης μεγάλου όγκου δεδομένων καταγραφής που δημιουργήθηκαν μέσα σε ένα πληροφοριακό σύστημα.

Οι αναλυτές της Gartner Inc. επινόησαν τον όρο SIEM σε μία αναφορά του Gartner το 2005 με τίτλο "Βελτίωση της Ασφάλειας Πληροφορικής με Διαχείριση Ευπάθειας" [21]. Στην αναφορά, οι αναλυτές πρότειναν ένα νέο σύστημα πληροφοριών ασφαλείας βασισμένο σε Security Information Management SIM και Security Event Management SEM.

Ενσωματωμένο σε παλαιότερα συστήματα διαχείρισης συλλογής αρχείων καταγραφής, το SIM εισήγαγε μακροπρόθεσμη ανάλυση αποθήκευσης και αναφορά δεδομένων καταγραφής. Ενσωμάτωσε επίσης αρχεία καταγραφής με τις πληροφορίες απειλής. Το SEM είναι υπεύθυνο για τον εντοπισμό, τη συλλογή, την παρακολούθηση και την αναφορά συμβάντων που σχετίζονται με την ασφάλεια σε λογισμικό, συστήματα ή υποδομές πληροφορικής [22].

Έτσι, δημιουργήθηκε το SIEM συνδυάζοντας το SEM, το οποίο αναλύει δεδομένα καταγραφής και συμβάντος σε πραγματικό χρόνο, παρέχοντας

παρακολούθηση απειλών, συσχέτιση συμβάντων και απόκριση συμβάντων, με SIM, το οποία συλλέγει, αναλύει και αναφέρει δεδομένα καταγραφής.

4.2.1 Πώς λειτουργεί το SIEM;

Οι βασικές αρχές κάθε συστήματος SIEM είναι η συλλογή σχετικών δεδομένων από πολλαπλές πηγές, η αναγνώριση αποκλίσεων από τον κανόνα και η ανάληψη κατάλληλης δράσης. Για παράδειγμα, όταν εντοπιστεί μια ύποπτη δραστηριότητα, ένα σύστημα SIEM ενδέχεται να καταγράψει πρόσθετες πληροφορίες, να δημιουργήσει μια ειδοποίηση και να δώσει οδηγίες σε άλλα χειριστήρια ασφαλείας να σταματήσουν την πρόοδο μιας δραστηριότητας [22,23].

Σε βασικό επίπεδο, ένα σύστημα SIEM μπορεί να βασίζεται σε κανόνες ή να χρησιμοποιεί μια στατιστική μηχανή συσχέτισης για να δημιουργήσει σχέσεις μεταξύ των αρχείων καταγραφής συμβάντων. Το λογισμικό SIEM δημιουργεί στη συνέχεια ειδοποιήσεις ασφαλείας όταν εντοπίζει πιθανά ζητήματα ασφαλείας. Χρησιμοποιώντας ένα σύνολο προκαθορισμένων κανόνων και πολιτικών, οι οργανισμοί μπορούν να ορίσουν αυτές τις ειδοποιήσεις ως χαμηλής ή υψηλής προτεραιότητας [23].

Τα συστήματα SIEM λειτουργούν αναπτύσσοντας πολλαπλούς παράγοντες συλλογής με ιεραρχικό τρόπο για τη συλλογή συμβάντων που σχετίζονται με την ασφάλεια από τελικούς χρήστες, διακομιστές και εξοπλισμό δικτύου, καθώς και εξειδικευμένο εξοπλισμό ασφαλείας όπως τείχη προστασίας, ιούς ή συστήματα πρόληψης εισβολών (IPSes). Οι συλλέκτες προωθούν εκδηλώσεις σε μια κεντρική κονσόλα διαχείρισης, όπου οι αναλυτές ασφάλειας μεταδίδουν θόρυβο, συνδέουν τελείες και δίνουν προτεραιότητα σε συμβάντα ασφαλείας

Σε γενικές γραμμές οι ικανότητες ενός SIEM μπορούν να κατηγοριοποιηθούν στις παρακάτω ομάδες [22-24]:

- Συγκέντρωση δεδομένων
- Συσχέτιση χαρακτηριστικών
- Προειδοποίηση
- Πίνακας ελέγχων

- Διατήρηση
- Συμμόρφωση/Compliance
- Εγκληματολογική ανάλυση

4.2.2 Τα οφέλη του SIEM

Τα οφέλη του SIEM είναι τα ακόλουθα [21-27]:

- Μειώνει το χρόνο που απαιτείται για τον σημαντικό εντοπισμό των απειλών, ελαχιστοποιώντας τη ζημιά από αυτές τις απειλές.
- Παρέχει μια ολιστική εικόνα του περιβάλλοντος ασφάλειας πληροφοριών ενός οργανισμού, διευκολύνοντας τη συλλογή και ανάλυση πληροφοριών ασφαλείας για τη διατήρηση της ασφάλειας του συστήματος, όλα τα δεδομένα ενός οργανισμού πηγαίνουν σε ένα κεντρικό αποθετήριο όπου είναι αποθηκευμένα και εύκολα προσβάσιμα.
- Μπορεί να χρησιμοποιηθεί από εταιρείες για ποικίλες χρήσεις που περιστρέφονται γύρω από δεδομένα ή αρχεία καταγραφής, συμπεριλαμβανομένων προγραμμάτων ασφάλειας, ελέγχου και συμμόρφωσης, υποστήριξης και αντιμετώπισης προβλημάτων δικτύου.
- Υποστηρίζει μεγάλο όγκο δεδομένων, έτσι ώστε οι οργανισμοί να μπορούν να συνεχίσουν την κλιμάκωση και την ανάπτυξη των δεδομένων τους.
- Παρέχει ειδοποιήσεις απειλής και ασφάλειας.
- Μπορεί να διενεργήσει λεπτομερή ιατροδικαστική ανάλυση σε περίπτωση σημαντικών παραβιάσεων της ασφάλειας.

4.2.3 Αρχιτεκτονική SIEM

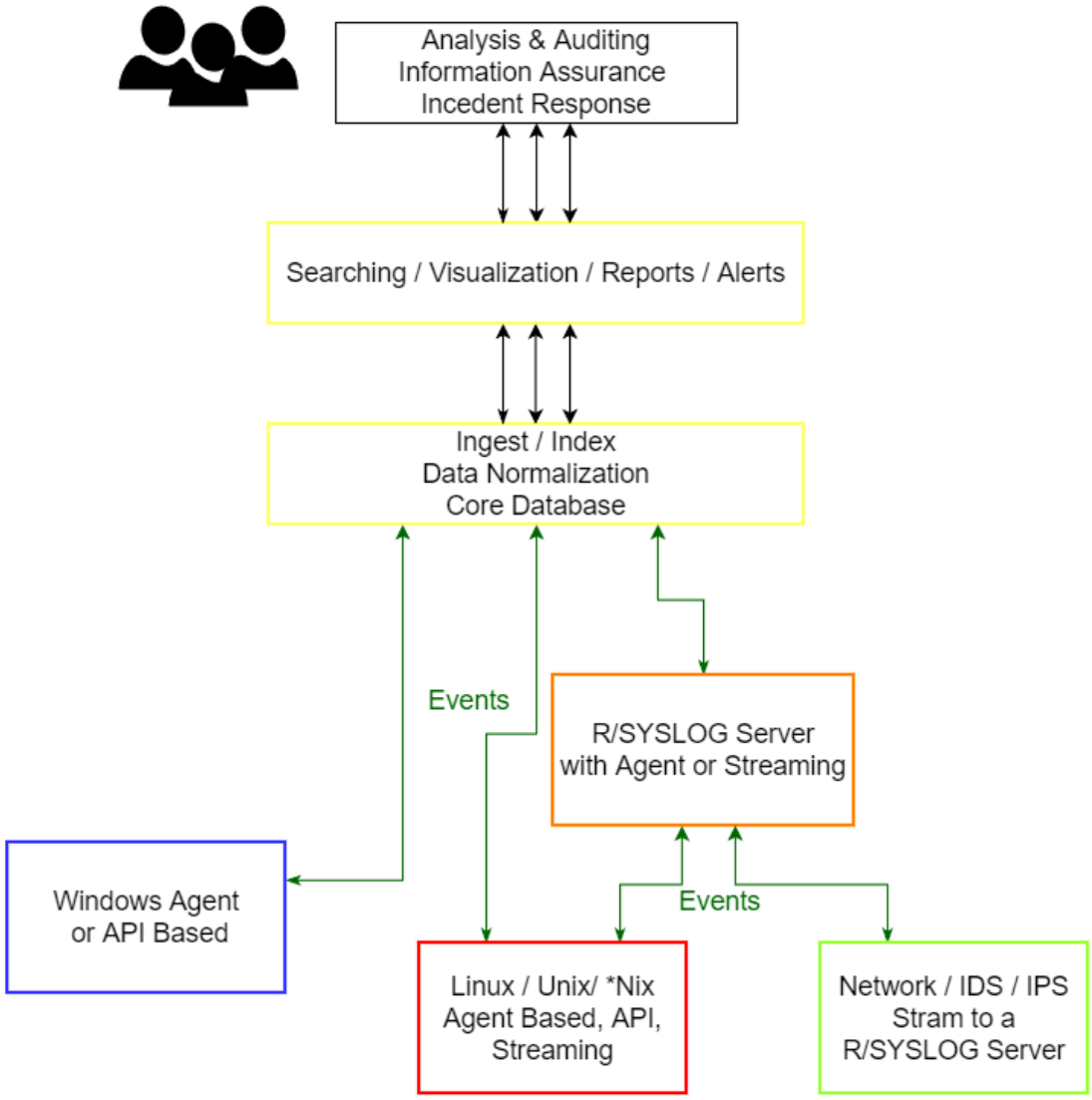
Οι αρχιτεκτονικές SIEM ενδέχεται να διαφέρουν ανάλογα με τον προμηθευτή [25-27]. Ωστόσο, γενικά, τα βασικά στοιχεία ενός SIEM είναι τα εξής:

- Ένας συλλέκτης δεδομένων, ο οποίος προωθεί επιλεγμένα αρχεία καταγραφής ελέγχου από έναν κεντρικό υπολογιστή (ροή καταγραφής

με βάση παράγοντα ή κεντρικό υπολογιστή σε ευρετήριο και σημείο συγκέντρωσης)

- Ένα σημείο συγκέντρωσης σημείου απορρόφησης και ευρετηρίου για ανάλυση, συσχέτιση και ομαλοποίηση δεδομένων
- Ένας κόμβος αναζήτησης που χρησιμοποιείται για οπτικοποίηση, ερωτήματα, αναφορές και ειδοποιήσεις (η ανάλυση πραγματοποιείται σε έναν κόμβο αναζήτησης/search node)

Μια βασική υποδομή SIEM απεικονίζεται στην Εικόνα 4.



Εικόνα 4 Αρχιτεκτονική SIEM

4.2.4 Σενάρια Χρήσης ενός SIEM

Ο ερευνητής ασφάλειας υπολογιστών Chris Kubecka εντόπισε τις ακόλουθες περιπτώσεις χρήσης του SIEM [28], που παρουσιάστηκαν στο συνέδριο χάκερ 28C3 (Chaos Communication Congress):.

- Η ανίχνευση ανωμαλιών SIEM θα μπορούσαν να βοηθήσουν στην ανίχνευση zero-day virus ή πολυμορφικού κώδικα. Κυρίως λόγω των χαμηλών ποσοστών ανίχνευσης ιών έναντι αυτού του τύπου ταχέως μεταβαλλόμενου κακόβουλου λογισμικού.
- Η ανάλυση, η κανονικοποίηση καταγραφής και η κατηγοριοποίηση μπορούν να πραγματοποιηθούν αυτόματα, ανεξάρτητα από τον τύπο του υπολογιστή ή της συσκευής δικτύου, εφόσον μπορεί να στείλει ένα αρχείο καταγραφής.
- Η οπτικοποίηση με ένα SIEM χρησιμοποιώντας συμβάντα ασφαλείας και αποτυχίες καταγραφής μπορεί να βοηθήσει στην ανίχνευση προτύπων.
- Οι ανωμαλίες πρωτοκόλλου που μπορεί να υποδεικνύουν λανθασμένη διαμόρφωση ή πρόβλημα ασφάλειας μπορούν να αναγνωριστούν με ένα SIEM χρησιμοποιώντας ανίχνευση προτύπων, ειδοποιήσεις, βασικές γραμμές και πίνακες ελέγχου.
- Το SIEM μπορεί να εντοπίσει κρυφές, κακόβουλες επικοινωνίες και κρυπτογραφημένα κανάλια.
- Ο κυβερνοπόλεμος μπορεί να εντοπιστεί από τα SIEM με ακρίβεια, ανακαλύπτοντας και τους επιτιθέμενους και τα θύματα.

4.3 Vulnerability Assessment & Vulnerability Scanners

4.3.1 Vulnerability Assessment

Η αξιολόγηση ευπάθειας δικτύου είναι μια διαδικασία εντοπισμού τρωτών σημείων ασφαλείας στα συστήματα, ποσοτικοποίησης και ανάλυσης αυτών και αποκατάστασης αυτών των ευπαθειών βάσει προκαθορισμένων κινδύνων. Οι αξιολογήσεις αποτελούν ουσιαστικό μέρος ενός ολιστικού

προγράμματος ασφάλειας και αναφέρονται από πολλά βιομηχανικά πρότυπα και κανονισμούς συμμόρφωσης.

Υπάρχει ένα ευρύ φάσμα πλεονεκτημάτων που σχετίζονται με την εκτίμηση τρωτότητας, ειδικά λαμβάνοντας υπόψη την αύξηση των επιθέσεων στον κυβερνοχώρο που συνέβησαν τα τελευταία χρόνια. Μόνο διευκολύνοντας τον προσδιορισμό, την ανάλυση και την αποκατάσταση των κινδύνων και των τρωτών σημείων, μπορεί μια επιχείρηση να αντιληφθεί την πλήρη αξία που μπορεί να προσφέρει η διαδικασία αξιολόγησης ευπάθειας [29].

Η διαχείριση ευπάθειας (Vulnerability Management) είναι η «κυκλική πρακτική εντοπισμού, ταξινόμησης, ιεράρχησης, αποκατάστασης και μετριασμού» τρωτών σημείων λογισμικού και είναι αναπόσπαστο μέρος της ασφάλειας υπολογιστών και δικτύων και δεν πρέπει να συγχέεται με την εκτίμηση ευπάθειας (Vulnerability Assessment) [30].

4.3.1.1 Οφέλη της Εκτίμησης Ευπάθειας

Η μείωση των κενών στην ασφάλεια του δικτύου αναφέρεται συνήθως ως σκλήρυνση/hardening του συστήματος. Αυτή είναι μια διαδικασία δημιουργίας ασφαλέστερων συστημάτων μειώνοντας το attack vector που μπορεί να εκμεταλλευτεί ένας κάποιος κακόβουλος.

Μερικά από τα κύρια βήματα του hardening του συστήματος περιλαμβάνουν το κλείσιμο περιττών ports και υπηρεσιών, την ενημέρωση του outdated λειτουργικού συστήματος και του λογισμικού και την αφιέρωση ενός ξεχωριστού server για κάθε σημαντική υπηρεσία. Οι εκτιμήσεις ευπάθειας, πιο συγκεκριμένα οι σαρώσεις ευπάθειας,βοηθούν στον αποτελεσματικό εντοπισμό των πιθανών διανυσμάτων επίθεσης και να βοηθούν στη μείωσή τους.

Η συμμόρφωση/compliant είναι κάτι για το οποίο προσπαθεί κάθε εταιρεία είναι να συμμορφώνεται με ορισμένα πρότυπα και κανονισμούς επειδή την καθιστά πιο ανταγωνιστική στην αγορά. Σύμφωνα με μια έρευνα από το Ινστιτούτο Ponemon, το 60% των παραβιάσεων το 2019 αφορούσε ευάλωτες ευπάθειες που δεν είχαν συμμορφωθεί. Ανάλογα με τη γεωγραφική θέση και τον κλάδο στον οποίο δραστηριοποιείται μια συγκεκριμένη εταιρεία, τα

παρακάτω είναι μερικά από τα βασικά πρότυπα συμμόρφωσης που προσπαθούν να πετύχουν οι εταιρείες:

- HIPAA – Health Insurance Portability and Accountability Act
- PCI DSS – Payment Card Industry Data Security Standard
- GDPR – General Data Protection Regulation
- ISO 27001 – information security standard from the International Organization for Standardization
- SOX – Sarbanes-Oxley Act
- FISMA – Federal Information Security Management Act of 2002
- GLBA – Gramm–Leach–Bliley Act

Δεν απαιτούν όλα τα παραπάνω πρότυπα συμμόρφωσης την ύπαρξη διαδικασίας αξιολόγησης ευπάθειας. Ωστόσο, η διαδικασία συμμόρφωσης είναι σχεδόν αδύνατη χωρίς αξιολόγηση.

4.3.1.2 Βήματα της Διαδικασίας Αξιολόγησης Ευπάθειας

Τα βήματα που χρειάζονται για μια Αξιολόγηση Ευπάθειας είναι:

1. Προσδιορισμός και ανάλυση κινδύνου

Η διαδικασία αναγνώρισης και ανάλυσης κινδύνου ξεκινά με τον προσδιορισμό όλων των περιουσιακών στοιχείων που αποτελούν μέρος ενός πληροφοριακού συστήματος σε μια εταιρεία. Με έναν πλήρη κατάλογο όλου του εξοπλισμού πληροφορικής, οι εταιρείες μπορούν να αρχίσουν να εκχωρούν κινδύνους σε κάθε περιουσιακό στοιχείο προκειμένου να λογοδοτούν για τις περισσότερες καταστάσεις που μπορεί να προκύψουν.

Στη συνέχεια, η εκχώρηση αξίας και η εκτέλεση της ανάλυσης θα καθορίσουν τον πραγματικό κίνδυνο που αντιμετωπίζει κάθε περιουσιακό στοιχείο. Με τους κινδύνους που εντοπίζονται και αναλύονται, η διαδικασία αξιολόγησης ευπάθειας αρχίζει να διαμορφώνεται και η εστίαση μετατοπίζεται αργά στα περιουσιακά στοιχεία με τον μεγαλύτερο κίνδυνο.

2. Πολιτικές και διαδικασίες σάρωσης ευπάθειας

Οι γραπτές πολιτικές και διαδικασίες αποτελούν το backbone κάθε σημαντικής δράσης που σχεδιάζεται να αναληφθεί. Συνεπώς, όλες οι δραστηριότητες που εκτελούνται πρέπει να βρίσκονται εντός των ορίων αυτών των πολιτικών και διαδικασιών.

Η σάρωση ευπάθειας δεν είναι διαφορετική. Δομώντας ολόκληρη τη διαδικασία μέσα σε μια πολιτική ή μια διαδικασία, θα υπάρχει ένα καθορισμένο σύνολο κανόνων και βημάτων που πρέπει να ληφθούν και σημεία που καθορίζουν απαγορευμένη συμπεριφορά.

Πριν από την έναρξη οποιασδήποτε σάρωσης ευπάθειας, θα πρέπει να υπάρχουν υπάρχουσες πολιτικές και διαδικασίες που θα καλύπτουν ολόκληρη τη διαδικασία της σάρωσης.

3. Σάρωση ευπάθειας

Ανάλογα με το μέρος του συστήματος που πρέπει να σαρωθούν, οι σαρώσεις ευπάθειας χωρίζονται σε δύο κατηγορίες, εξωτερικές και εσωτερικές. Οι εξωτερικές σαρώσεις περιλαμβάνουν όλους τους δημόσια διαθέσιμους πόρους, ενώ οι εσωτερικές σαρώσεις στοχεύουν όλα τα εσωτερικά στοιχεία που είναι απρόσιτα από το διαδίκτυο.

Ανάλογα με το ποιος πραγματοποιεί τη σάρωση ευπάθειας, μπορεί να ταξινομηθεί ως εσωτερική σάρωση ή ως σάρωση τρίτου μέρους. Οι σαρώσεις ευπάθειας εκτελούνται συνήθως από εξειδικευμένο προσωπικό ασφαλείας και διαμορφώνονται σε διάφορα εργαλεία που διατίθενται ως λύση λογισμικού επί πληρωμή ή σε μορφή ανοιχτού κώδικα.

4.3.2 Vulnerability Scanner

Ο σαρωτής ευπάθειας είναι ένα πρόγραμμα σχεδιασμένο για την αξιολόγηση υπολογιστών, δικτύων ή εφαρμογών για γνωστές αδυναμίες. Αυτοί οι σαρωτές χρησιμοποιούνται για να ανακαλύψουν τις αδυναμίες ενός δεδομένου συστήματος. Χρησιμοποιούνται για την αναγνώριση και ανίχνευση τρωτών σημείων που προκύπτουν από λανθασμένες διαμορφώσεις ή

ελαττωματικό προγραμματισμό σε ένα στοιχείο δικτύου, όπως τείχος προστασίας, δρομολογητή, διακομιστή Ιστού, διακομιστή εφαρμογών κ.λπ.

Οι σύγχρονοι σαρωτές επιτρέπουν τις αυθεντικοποιημένες και μη σαρώσεις. Είναι συνήθως διαθέσιμοι ως SaaS (Λογισμικό ως Υπηρεσία - Software as a Service), παρέχονται μέσω διαδικτύου και παραδίδονται ως διαδικτυακή εφαρμογή. Ο σύγχρονος σαρωτής ευπάθειας έχει συχνά τη δυνατότητα να προσαρμόζει τις αναφορές ευπάθειας καθώς και το εγκατεστημένο λογισμικό, τα ανοιχτά port, τα πιστοποιητικά και άλλες πληροφορίες για το host υπολογιστή που μπορούν να κρατηθούν ως μέρος της ροής εργασίας του [32].

- Οι αυθεντικοποιημένες σαρώσεις επιτρέπουν στον σαρωτή να έχει άμεση πρόσβαση σε στοιχεία δικτύου χρησιμοποιώντας απομακρυσμένα πρωτόκολλα διαχείρισης, όπως SSH ή πρωτόκολλο απομακρυσμένης επιφάνειας εργασίας (Remote Desktop Protocol - RDP) και τον έλεγχο ταυτότητας χρησιμοποιώντας τα παρεχόμενα διαπιστευτήρια συστήματος. Αυτό επιτρέπει στον σαρωτή ευπάθειας να έχει πρόσβαση σε δεδομένα χαμηλού επιπέδου, όπως συγκεκριμένες υπηρεσίες και λεπτομέρειες διαμόρφωσης του λειτουργικού συστήματος κεντρικού υπολογιστή. Στη συνέχεια, είναι σε θέση να παρέχει λεπτομερείς και ακριβείς πληροφορίες σχετικά με το λειτουργικό σύστημα και το εγκατεστημένο λογισμικό, συμπεριλαμβανομένων ζητημάτων διαμόρφωσης και ελλείψεων ενημερώσεων κώδικα ασφαλείας.
- Οι μη πιστοποιημένες σαρώσεις είναι μια μέθοδος που μπορεί να οδηγήσει σε μεγάλο αριθμό false-positives και δεν είναι σε θέση να παρέχει λεπτομερείς πληροφορίες σχετικά με τα λειτουργικά συστήματα και το εγκατεστημένο λογισμικό. Αυτή η μέθοδος χρησιμοποιείται συνήθως από threat actors ή αναλυτές ασφαλείας που προσπαθούν να καθορίσουν τη στάση ασφαλείας των εξωτερικά προσβάσιμων στοιχείων [32].

Επιπλέον, ανάλογα με το μέρος του συστήματος που πρέπει να σαρωθούν, οι σαρώσεις ευπάθειας χωρίζονται σε δύο κατηγορίες, εξωτερικές και εσωτερικές. Οι εξωτερικές σαρώσεις περιλαμβάνουν όλους τους δημόσια

διαθέσιμους πόρους, ενώ οι εσωτερικές σαρώσεις στοχεύουν όλα τα εσωτερικά στοιχεία που είναι απρόσιτα από το διαδίκτυο.

Ανάλογα με το ποιος πραγματοποιεί τη σάρωση ευπάθειας, μπορεί να ταξινομηθεί ως εσωτερική σάρωση ή ως σάρωση τρίτου μέρους. Οι σαρώσεις ευπάθειας εκτελούνται συνήθως από εξειδικευμένο προσωπικό ασφαλείας και διαμορφώνονται σε διάφορα εργαλεία που διατίθενται ως λύση λογισμικού επί πληρωμή ή σε μορφή ανοιχτού κώδικα.Ο στόχος μιας σάρωσης ευπάθειας είναι να εντοπίσει και να κατηγοριοποιήσει τα τρωτά σημεία που βρίσκονται στο δίκτυο.

Τα τρωτά σημεία του δικτύου έχουν πολλές μορφές, αλλά οι πιο συνηθισμένοι τύποι είναι [33]:

- Malware, συντομογραφία για malicious software, όπως Trojans, viruses και worms που είναι εγκατεστημένα σε υπολογιστή ενός χρήστη ή host server.
- Social Engineering επιθέσεις που ξεγελούν τους χρήστες να εγκαταλείψουν προσωπικές πληροφορίες, όπως username ή password.
- Outdated ή unpached software που εκθέτει τα συστήματα που τρέχουν την εφαρμογή και ενδεχομένως ολόκληρο το δίκτυο. Σύμφωνα με μια έρευνα από το Ινστιτούτο Ponemon, το 60% των παραβιάσεων το 2019 αφορούσε ευάλωτες ευπάθειες που ήταν upached [34].
- Λάθος διαμορφωμένα firewalls / λειτουργικά συστήματα που επιτρέπουν ή έχουν ενεργοποιήσει τις προεπιλεγμένες πολιτικές.

4.3.2.1 Πώς λειτουργεί ένας σαρωτής ευπάθειας ανοιχτού κώδικα;

Ενώ κάθε σαρωτής ευπάθειας ανοιχτού κώδικα χρησιμοποιεί διαφορετική τεχνολογία, μπορούμε να προσδιορίσουμε μια διαδικασία τριών σταδίων που περνούν οι περισσότεροι σαρωτές:

1. Σάρωση στοιχείων ανοιχτού κώδικα

Ο σαρωτής εξετάζει όλα τα στοιχεία ανοιχτού κώδικα στο έργο λογισμικού, συχνά αναλύοντας code repositories, package managers και build tools. Καθιερώνει μια απογραφή στοιχείων και εξαρτήσεων ανοιχτού κώδικα και

προσδιορίζει σχετικά μεταδεδομένα, συμπεριλαμβανομένης της προέλευσης, της άδειας και της έκδοσης.

2. Επαλήθευση της συμμόρφωσης της άδειας

Οι περισσότεροι σαρωτές ευπάθειας μπορούν να εντοπίσουν άδειες χρήσης λογισμικού (software licenses) για στοιχεία ανοικτού κώδικα και να επαληθεύσουν εάν έρχονται σε σύγκρουση με τις οργανωτικές πολιτικές. Για παράδειγμα, ορισμένες άδειες ανοικτού κώδικα μπορεί να είναι επικίνδυνες για χρήση σε εμπορικά έργα, να εκθέσουν πολύτιμη πνευματική ιδιοκτησία ή να έχουν νομικές επιπτώσεις σε ολόκληρο το έργο ανάπτυξης λογισμικού. Οι σαρωτές μπορούν να ειδοποιούν για προβληματικές άδειες, τόσο για τα κύρια στοιχεία ανοιχτού κώδικα που χρησιμοποιούνται όσο και για τις εξαρτήσεις τους.

3. Εντοπισμός τρωτών σημείων

Οι σαρωτές ευπάθειας λαμβάνουν τα αποτελέσματα και τα ελέγχουν με μία ή περισσότερες βάσεις δεδομένων που περιέχουν πληροφορίες σχετικά με ευπάθειες, συμπεριλαμβανομένων των βάσεων δεδομένων Common Vulnerabilities and Exposures (CVE) που αποτελούν τυποποιημένη λίστα ευπάθειας που είναι γνωστή στους ερευνητές ασφαλείας και ιδιόκτητες βάσεις δεδομένων έρευνας ασφάλειας. Τέλος, προειδοποιούν τον χρήστη σχετικά με τα ευπαθή σημεία που εντοπίστηκαν και προτείνουν μια διαδρομή αποκατάστασης.

4.3.2.2 Εργαλεία Σάρωσης Ευπάθειας

Τα εργαλεία λογισμικού για σαρώσεις ευπάθειας διαφέρουν ανάλογα με τους στόχους της άσκησης. Το είδος της άσκησης θα επηρεάσει επίσης τα εργαλεία που χρησιμοποιούνται [19].

- Nessus – είναι ένα εξαιρετικά εύκολο στη χρήση εργαλείο σάρωσης ευπάθειας που σχεδιάστηκε για να μειώσει το συνολικό χρόνο και τις προσπάθειες που μπορεί να χρειαστείτε για ζητήματα σάρωσης, ιεράρχησης προτεραιότητας και αποκατάστασης. Λειτουργεί δοκιμάζοντας όλες τις θύρες σε μια συσκευή ξεχωριστά για να καθορίσει

το λειτουργικό της σύστημα. Στη συνέχεια εξετάζει το λειτουργικό σύστημα για να ελέγξει για τυχόν γνωστή ευπάθεια.

- OpenVas – είναι έξυπνα σχεδιασμένο για να βοηθήσει την εκτέλεση deep scan και τη διαχείριση πακέτων ευπάθειας, σαρώνοντας όλες τις συσκευές δικτύου και τους servers. Λειτουργεί επιλέγοντας τον στόχο, π.χ. μια καθορισμένη διεύθυνση IP και στη συνέχεια ξεκινώντας τις σαρώσεις σύμφωνα με τον προτιμώμενο τύπο σάρωσης για τη διάγνωση αδυναμιών.
- Netspark – είναι ένα εργαλείο ελέγχου σαρωτή ασφάλειας που χρησιμοποιείται για την αυτοματοποίηση των δοκιμών εφαρμογών ιστότοπου. Το λογισμικό είναι σε θέση να εντοπίσει επιθέσεις cross-site scripting και SQL injection. Αυτό είναι ιδιαίτερα σημαντικό για προγραμματιστές, καθώς μπορούν να χρησιμοποιούν το Netspark στους ιστότοπούς τους, στις υπηρεσίες ιστού και στις εφαρμογές ιστού.

4.4 CIS Framework

Το Κέντρο για την Ασφάλεια στο Διαδίκτυο - Center for Internet Security (CIS) είναι ένας μη κερδοσκοπικός οργανισμός, που ιδρύθηκε τον Οκτώβριο του 2000 και του οποίου η αποστολή είναι να βελτιώσει την ετοιμότητα και την ανταπόκριση στον κυβερνοχώρο στον δημόσιο και ιδιωτικό τομέα [35].

Οι ιδρυτικοί οργανισμοί το CIS περιλαμβάνουν μερικούς από τους πιο σεβαστούς ηγέτες στον τομέα της ασφάλειας πληροφορικής στον κόσμο, όπως το ISACA, το American Institute of Certified Public Accountants (AICPA), το Institute of Internal Auditors (IIA), το International Information Systems Security Certification Consortium (ISC2) και το Ινστιτούτο SANS [36].

Μια παλιά παροιμία λέει ότι δεν πρέπει κάποιος να επιχειρήσει να ανακαλύψει ξανά τον τροχό, καθώς είναι χάσιμο χρόνου και έχει ήδη τελειοποιηθεί. Αντίθετα, είναι κοινώς αποδεκτό ότι οπουδήποτε υπάρχει ένα υπάρχον πρότυπο, θα πρέπει να χρησιμοποιείται αυτό. Η βιομηχανία ασφάλειας πληροφορικής δεν διαφέρει.

Το CIS δημοσιεύει οδηγίες που περιγράφουν λεπτομερώς τον τρόπο ρύθμισης παραμέτρων σε μια ποικιλία λειτουργικών συστημάτων, εφαρμογών

και συσκευών δικτύου, έτσι ώστε να βρίσκονται σε γνωστή "καλή και ασφαλή" κατάσταση.

Τα CIS Benchmarks καλύπτουν μια σειρά από περισσότερα από 100 διαφορετικά λειτουργικά συστήματα, εφαρμογές και συσκευές δικτύου [36]. Αυτό περιλαμβάνει αλλά δεν περιορίζεται στα:

- Microsoft Windows operating systems.
- Cisco network devices.
- Major internet browser applications.
- Multi-function printers.
- Cloud providers.
- Database software.
- Mobile applications and operating systems.

Οι οργανισμοί που παρέχουν προϊόντα κυβερνοασφάλειας ως υπηρεσία μπορούν να λάβουν πιστοποίηση CIS για το προϊόν. Αυτό πιστοποιεί ότι το εν λόγω προϊόν είναι συμβατό με τις συστάσεις για την κυβερνοασφάλεια στο σχετικό σημείο αναφοράς CIS.

4.4.1 CIS Benchmarks

Τα CIS Benchmarks είναι βασικές γραμμές διαμόρφωσης και βέλτιστες πρακτικές για την ασφαλή διαμόρφωση ενός συστήματος. Κάθε μία από τις συστάσεις καθοδήγησης αναφέρεται σε έναν ή περισσότερους ελέγχους CIS που αναπτύχθηκαν για να βοηθήσουν τους οργανισμούς να βελτιώσουν τις δυνατότητές τους στον κόσμο της κυβερνοάμυνα;. Οι έλεγχοι CIS αντιστοιχούν σε πολλά καθιερωμένα πρότυπα και ρυθμιστικά πλαίσια, όπως το NIST Cybersecurity Framework (CSF) και το NIST SP 800-53, τη σειρά προτύπων ISO 27000, PCI DSS, HIPAA και άλλα [37-39].

Επιπλέον, πολλά CIS Security Benchmarks υπάρχουν σε open standard schemas, συμπεριλαμβανομένης του Extensible Configuration Checklist Description (XCCDF). Το CIS ασχολείται επίσης με τη διερεύνηση πρόσθετων ευκαιριών για την υποστήριξη της παραγωγής και της συμβολής νέων σχημάτων και περιεχομένου Open Vulnerability and Assessment Language

(OVAL) για την εκτέλεση αυτοματοποιημένων ελέγχων συστημάτων βασισμένων σε πρότυπα βάσει της πολιτικής XCCDF για συστάσεις για τα CIS Benchmarks [38].

Κάθε Benchmark υποβάλλεται σε δύο φάσεις συναίνεσης. Το πρώτο συμβαίνει κατά την αρχική ανάπτυξη όταν οι εμπειρογνώμονες συνέρχονται για να συζητήσουν, να δημιουργήσουν και να δοκιμάσουν προσχέδια εργασίας μέχρι να καταλήξουν σε συναίνεση ως προς αυτά. Κατά τη δεύτερη φάση, μετά τη δημοσίευση του, η ομάδα συναίνεσης εξετάζει τα σχόλια από την κοινότητα του διαδικτύου για ενσωμάτωση στο benchmark.

Υπάρχουν επί του παρόντος πάνω από 70 σημεία αναφοράς CIS που καλύπτουν 14 ομάδες τεχνολογίας και διατίθενται ως αυτοματοποιημένοι, προσαρμόσιμοι πόροι σε μορφές αναγνώσιμες από μηχανή για τα μέλη των CIS Security Benchmarks [35]. Διατίθενται πάνω από 100 Benchmarks σε 14 τεχνολογικές ομάδες, συμπεριλαμβανομένων περιουσιακών στοιχείων που παράγονται από τη Microsoft, τη Cisco, την AWS και την IBM [40].

Τα Security Technical Implementation Guides (STIGs), του Defense Information Systems Agency (DISA), είναι ένα σύνολο κανόνων βέλτιστων πρακτικών για την εγκατάσταση και υποστήριξη συστημάτων πληροφορικής. Κάθε STIG σχετίζεται με ένα συγκεκριμένο asset ,για παράδειγμα ένα λειτουργικό σύστημα ή μια εφαρμογή και εκθέτει τις πρακτικές διαμόρφωσης και συντήρησης που είναι απαραίτητες για τη διασφάλιση ότι το asset έχει διασφαλιστεί και έχει ρυθμιστεί με ασφάλεια.

Τα STIGs που αναπτύχθηκαν από τον DISA για λογαριασμό του Υπουργείου Άμυνας, είναι τα αποδεκτά πρότυπα που χρησιμοποιούνται από τις ομοσπονδιακές κυβερνητικές οργανώσεις και εργολάβους για τη διασφάλιση της ασφάλειας των κυβερνητικών πληροφοριών [40].

Τα CIS Benchmarks αναπτύσσονται μέσω των γενναιόδωρων εθελοντικών προσπαθειών ειδικών θεμάτων, πωλητών τεχνολογίας, μελών της δημόσιας και ιδιωτικής κοινότητας και της ομάδας ανάπτυξης CIS Benchmark. Τα CIS Benchmarks περιλαμβάνουν πολλαπλά προφίλ διαμόρφωσης. Ένας ορισμός προφίλ περιγράφει τις διαμορφώσεις που έχουν εκχωρηθεί σε προτάσεις αναφοράς [37].

- Το προφίλ **επιπέδου 1** θεωρείται μια βασική σύσταση που μπορεί να εφαρμοστεί αρκετά γρήγορα και έχει σχεδιαστεί για να μην έχει εκτεταμένο αντίκτυπο στην απόδοση. Ο σκοπός του επιπέδου αναφοράς προφίλ επιπέδου 1 είναι να μειώσει την επιφάνεια επίθεσης του οργανισμού σας διατηρώντας παράλληλα τη χρήση των μηχανών και μη εμποδίζοντας τη λειτουργικότητα της επιχείρησης.
- Το προφίλ **επιπέδου 2** θεωρείται "άμυνα σε βάθος/defense in depth" και προορίζεται για περιβάλλοντα όπου η ασφάλεια είναι υψίστης σημασίας. Οι συστάσεις που σχετίζονται με το προφίλ επιπέδου 2 μπορεί να έχουν δυσμενείς επιπτώσεις στον οργανισμό σας εάν δεν εφαρμοστούν κατάλληλα ή χωρίς τη δέουσα προσοχή.

Το προφίλ STIG αντικαθιστά το προηγούμενο επίπεδο 3. Το προφίλ STIG παρέχει όλες τις συστάσεις που σχετίζονται με το STIG. Η επικάλυψη συστάσεων από άλλα προφίλ, δηλαδή το επίπεδο 1 και το επίπεδο 2, υπάρχουν στο προφίλ STIG, όπως ισχύει.

Όπως και με τα DISA STIGs, η διασφάλιση της συμμόρφωσης με τα CIS Benchmarks σε όλα τα asset μπορεί να είναι κάτι πολύπλοκο.

Μια αυτοματοποιημένη λύση μπορεί να καταστήσει την εφαρμογή και τη διατήρηση της συμμόρφωσης με ένα πλαίσιο όπως τα DISA STIGs ή CIS Benchmark πολύ ταχύτερη και με λιγότερη απαίτηση πόρων.

Επιπλέον, με τη συνεχή σάρωση του περιβάλλοντός με την πάροδο του χρόνου και την επισήμανση λανθασμένων διαμορφώσεων που προκύπτουν, αυτές οι λύσεις διευκολύνουν επίσης τη διατήρηση της συμμόρφωσης.

Υπάρχουν επτά βασικές κατηγορίες CIS Benchmarks [39]:

1. Τα Benchmarks λειτουργικών συστημάτων καλύπτουν διαμορφώσεις ασφαλείας βασικών λειτουργικών συστημάτων, όπως Microsoft Windows, Linux και Apple OSX.
2. Τα Benchmarks λογισμικού server καλύπτουν διαμορφώσεις ασφαλείας λογισμικού server που χρησιμοποιείται ευρέως, συμπεριλαμβανομένων των Microsoft Windows Server, SQL Server, VMware, **Docker** και Kubernetes. Αυτά τα σημεία αναφοράς περιλαμβάνουν προτάσεις για τη διαμόρφωση των πιστοποιητικών Kubernetes PKI, τις ρυθμίσεις

διακομιστή API, τα στοιχεία ελέγχου διαχειριστή server, τις πολιτικές vNetwork και τους περιορισμούς αποθήκευσης.

3. Τα Benchmarks παρόχου cloud παρέχουν ρυθμίσεις ασφαλείας για τις υπηρεσίες Web Amazon (AWS), Microsoft Azure, Google, IBM και άλλα δημοφιλή δημόσια cloud.
4. Τα Benchmarks για φορητές συσκευές απευθύνονται σε λειτουργικά συστήματα κινητής τηλεφωνίας, συμπεριλαμβανομένων iOS και Android, και εστιάζουν σε τομείς όπως επιλογές και ρυθμίσεις προγραμματιστή, διαμορφώσεις απορρήτου λειτουργικού συστήματος, ρυθμίσεις προγράμματος περιήγησης και δικαιώματα εφαρμογής.
5. Τα Benchmarks συσκευών δικτύου προσφέρουν γενικές και συγκεκριμένες οδηγίες για τη ρύθμιση παραμέτρων ασφαλείας για συσκευές δικτύου και εφαρμοστέο υλικό από Cisco, Palo Alto Networks, Juniper και άλλα.
6. Τα Benchmarks λογισμικού επιφάνειας εργασίας καλύπτουν διαμορφώσεις ασφαλείας για μερικές από τις πιο συχνά χρησιμοποιούμενες εφαρμογές λογισμικού επιφάνειας εργασίας, συμπεριλαμβανομένων των Microsoft Office και Exchange Server, Google Chrome, Mozilla Firefox και Safari Browser.
7. Τα Benchmarks συσκευών εκτύπωσης πολλαπλών λειτουργιών σκιαγραφούν τις βέλτιστες πρακτικές ασφάλειας για τη διαμόρφωση εκτυπωτών πολλαπλών λειτουργιών σε ρυθμίσεις γραφείου και καλύπτουν θέματα όπως οι διαμορφώσεις TCP/IP, η διαμόρφωση ασύρματης πρόσβασης, η διαχείριση χρηστών και η κοινή χρήση αρχείων.

4.4.1.1 Ανάπτυξη ενός CIS Benchmark

Τα CIS Benchmarks αναπτύσσονται με τη βοήθεια μιας σειράς εθελοντών εμπειρογνωμόνων στον τομέα της ασφάλειας στον κυβερνοχώρο και της πληροφορικής. Κάθε CIS Benchmarks ολοκληρώνει μια διαδικασία συναίνεσης σε δύο βήματα.

Στο πρώτο βήμα, μια ομάδα ή ειδικοί στον τομέα της ασφάλειας στον κυβερνοχώρο δημιουργούν, συζητούν και δοκιμάζουν ένα σχέδιο έκδοσης των συστάσεων αναφοράς. Μόλις οι εμπειρογνώμονες επιτύχουν συναίνεση σχετικά με το σχέδιο καθοδήγησης για το CIS Benchmark, δημοσιεύεται για ανασκόπηση από την ευρύτερη κοινότητα εμπειρογνωμόνων στον τομέα της ασφάλειας στον κυβερνοχώρο.

Το δεύτερο βήμα έχει ένα δίκτυο επαγγελματιών στον τομέα της ασφάλειας στον κυβερνοχώρο από όλο τον κόσμο που αναθεωρούν τις συστάσεις της Συνοριακής Συνόδου CIS. Τα σχόλια από την ευρύτερη κοινότητα συλλέγονται και αναθεωρούνται από την ομάδα εμπειρογνωμόνων, με το δείκτη αναφοράς να τροποποιείται για να διασφαλίσει τα πρότυπα βέλτιστων πρακτικών [41].

4.4.1.2 Τα Οφέλη των CIS Benchmarks

Τα CIS Benchmark βοηθούν τους οργανισμούς να δημιουργήσουν συστήματα πληροφορικής και τεχνολογίας για να εξασφαλίσουν βέλτιστες πρακτικές άμυνας στον κυβερνοχώρο. Οι κατευθυντήριες γραμμές παίζουν σημαντικό ρόλο στη διαμόρφωση της πολιτικής κυβερνοασφάλειας ενός οργανισμού. Υπάρχουν σημεία αναφοράς για πολλούς τύπους τεχνολογιών, συμπεριλαμβανομένων δημοφιλών λειτουργικών συστημάτων και προγραμμάτων περιήγησης [39,41].

Κάθε στοιχείο του δικτύου πληροφορικής ενός οργανισμού μπορεί να έχει ευπάθειες στον κυβερνοχώρο εάν δεν έχει ρυθμιστεί σωστά. Ακολουθώντας τα σημεία αναφοράς CIS, οι οργανισμοί μπορούν να εξασφαλίσουν συστήματα πληροφορικής χρησιμοποιώντας ένα πλαίσιο που αναπτύχθηκε από κορυφαίους ειδικούς στον τομέα της ασφάλειας στον κυβερνοχώρο.

Τα οφέλη επίσης περιλαμβάνουν:

- Ενίσχυση των τρωτών σημείων που μπορούν να προκαλέσουν σοβαρά περιστατικά κυβερνοασφάλειας.
- Δωρεάν λήψη και ενσωμάτωση.
- Ένα σαφές εργαλείο για τη βελτίωση της διαδικασίας διακυβέρνησης.

- Προστασία ζωτικών συστημάτων πληροφορικής μέσα σε έναν οργανισμό, από λειτουργικά συστήματα έως δίκτυα.
- Η συλλεγμένη τεχνογνωσία μιας παγκόσμιας κοινότητας επαγγελματιών πληροφορικής και κυβερνοασφάλειας.
- Τακτικά ενημερωμένη, βήμα προς βήμα καθοδήγηση για την ασφάλεια κάθε περιοχής της υποδομής πληροφορικής.
- Συνοχή διαχείρισης συμμόρφωσης.
- Ένα πρότυπο ευελιξίας για την ασφαλή υιοθέτηση νέων υπηρεσιών cloud και για την εκτέλεση στρατηγικών ψηφιακού μετασχηματισμού.
- Εύκολη εγκατάσταση διαμορφώσεων για βελτιωμένη λειτουργική αποδοτικότητα και βιωσιμότητα.

4.4.3 CIS Controls

Τα CIS Controls, ή οι κρίσιμοι έλεγχοι ασφαλείας του CIS για αποτελεσματική κυβερνοάμυνα, είναι ένα σύνολο σαφών δράσεων για τους οργανισμούς για την ενίσχυση της ασφάλειας στον κυβερνοχώρο. Τα CIS Controls είναι ένα ξεχωριστό πρόγραμμα από το CIS, αλλά αναφέρονται σε όλα τα CIS Benchmarks.

Ο σκοπός του CIS Controls είναι να παρέχει σαφείς, εστιασμένες ενέργειες που θα έχουν αντίκτυπο σε σοβαρές απειλές για συστήματα πληροφορικής. Υπάρχουν 20 διαφορετικοί έλεγχοι CIS, οι οποίοι αποτελούνται από μια σειρά δράσεων για τη βελτίωση της ανθεκτικότητας στις κυβερνοεπιθέσεις. Έχουν σχεδιαστεί για να είναι απλές και αποτελεσματικές, συμβάλλοντας στον μετριασμό των πιθανών ζημιών από γνωστές απειλές στον κυβερνοχώρο.

Ενώ τα CIS Benchmarks επικεντρώνονται στη βασική γραμμή κυβερνοασφάλειας ενός συγκεκριμένου συστήματος ή προϊόντος, τα CIS Controls αποτελούν κατευθυντήριες γραμμές για ολόκληρο το σύστημα πληροφορικής. Αποτελούν σημαντικά εργαλεία για τυχόν στρατηγικές αποφάσεις διακυβέρνησης πληροφορικής ή διαδικασίες διαχείρισης κινδύνων. Τα 20 CIS Controls ομαδοποιούνται σε τρεις κατηγορίες για να βοηθήσουν στην εφαρμογή. Τα πρώτα έξι ανήκουν στην «βασική» κατηγορία και αποτελούνται

από σαφείς βασικές ενέργειες για να βοηθήσουν κάθε οργανισμό να προετοιμάσει την άμυνα στον κυβερνοχώρο. Τα επόμενα οκτώ ανήκουν στην κατηγορία «θεμελιώδη», οι οποίες παρέχουν τεχνικές ενέργειες για την περαιτέρω βελτίωση της άμυνας στον κυβερνοχώρο σε όλους τους οργανισμούς.

Οι τέσσερις τελευταίοι έλεγχοι CIS ανήκουν στην κατηγορία «οργανωτικά», οι οποίοι ασχολούνται με τη γενική λειτουργία του συστήματος πληροφορικής. Αυτή η κατηγορία εστιάζει στη δομή του ίδιου του οργανισμού, συμπεριλαμβανομένων διαδικασιών για την αντιμετώπιση περιστατικών και ευρύτερων προγραμμάτων κατάρτισης [41].

Τα 20 CIS Controls είναι:

1. Inventory and Control of Hardware Assets
2. Inventory and Control of Software Assets
3. Continuous Vulnerability Management
4. Controlled Use of Administrative Privileges
5. Secure Configuration for Hardware and Software on Mobile Devices, Laptops, Workstations and Servers
6. Maintenance, Monitoring and Analysis of Audit Logs

Foundational CIS Controls

7. Email and Web Browser Protections
8. Malware Defenses
9. Limitation and Control of Network Ports, Protocols and Services
10. Data Recovery Capabilities
11. Secure Configuration for Network Devices, such as Firewalls, Routers and Switches
12. Boundary Defense
13. Data Protection
14. Controlled Access Based on the Need to Know
15. Wireless Access Control
16. Account Monitoring and Control

Organizational CIS Controls

17. Implement a Security Awareness and Training Program

18. Application Software Security
19. Incident Response and Management
20. Penetration Tests and Red Team Exercises

Υπάρχουν τρεις ομάδες υλοποίησης:

1. **Ομάδα υλοποίησης 1**. Μικρότεροι οργανισμοί με περιορισμένους πόρους για διάθεση στην κυβερνοασφάλεια. Η ευαισθησία δεδομένων μπορεί να είναι χαμηλή και οι οργανισμοί πιθανότατα θα χρησιμοποιούν λογισμικό εκτός τεχνολογίας και συστήματα πληροφορικής.
2. **Ομάδα υλοποίησης 2**. Μεγαλύτεροι οργανισμοί με πολλά τμήματα και πιο περίπλοκα συστήματα πληροφορικής. Μπορεί να υπάρχει ανάγκη συμμόρφωσης με την ασφάλεια στον κυβερνοχώρο και οι οργανισμοί πιθανότατα θα χρησιμοποιούν συστήματα και προϊόντα πληροφορικής σε επίπεδο επιχείρησης.
3. **Ομάδα υλοποίησης 3**. Σύνθετοι οργανισμοί με απαιτήσεις συμμόρφωσης με την κυβερνοασφάλεια. Ενδέχεται να υπάρχουν ειδικοί στον τομέα της ασφάλειας στον κυβερνοχώρο εντός του οργανισμού, με σύνθετες ευθύνες διακυβέρνησης και διαχείρισης κινδύνων. Τα CIS Controls θα βοηθήσουν στη μείωση του κινδύνου στοχευμένων απειλών στον κυβερνοχώρο.

Και στις τρεις ομάδες οργανώσεων, τα CIS Controls στοχεύουν στη βελτίωση της άμυνας στον κυβερνοχώρο και στην ευθυγράμμιση των πόρων για το καλύτερο δυνατό αποτέλεσμα. Αντιστοιχίζονται σε άλλα πρότυπα κυβερνοασφάλειας, όπως το πλαίσιο ITIL, ώστε να μπορούν εύκολα να ενσωματωθούν στα υπάρχοντα συστήματα διακυβέρνησης πληροφορικής. Οι έλεγχοι CIS βοηθούν τους οργανισμούς να δώσουν προτεραιότητα στους πόρους τους για τον μεγαλύτερο αντίκτυπο στην άμυνα της κυβερνοασφάλειας.

4.4.4 CIS Benchmarks για Docker

Τα containers, μαζί με orchestrators όπως το Kubernetes, έχουν εγκαινιάσει μια νέα εποχή μεθοδολογίας ανάπτυξης εφαρμογών, επιτρέποντας

αρχιτεκτονικές μικροϋπηρεσιών καθώς και συνεχή ανάπτυξη και παράδοση. Το Docker είναι μακράν το πιο κυρίαρχο runtime engine, με ποσοστό 91% σύμφωνα με την έκθεση ασφαλείας της κατάστασης του εμπορευματοκιβωτίου και της Kubernetes [42,43].

Το containerization έχει πολλά οφέλη και ως εκ τούτου έχει υιοθετηθεί ευρέως. Η Gartner [44] προβλέπει ότι έως το 2023, το 70% των οργανισμών θα τρέχουν τρεις ή περισσότερες εφαρμογές σε container. Τα container, τα Kubernetes και τα μοντέλα εφαρμογών μικροϋπηρεσιών είναι τρεις από τους κορυφαίους παράγοντες καινοτομίας και ψηφιακού μετασχηματισμού επιχειρήσεων. Οι εταιρείες έχουν αγκαλιάσει αυτές τις τεχνολογίες για τα πλεονεκτήματά τους στην ανάπτυξη και ανάπτυξη εφαρμογών.

Ωστόσο, η δημιουργία εφαρμογών χρησιμοποιώντας κοντέινερ Docker εισάγει επίσης νέες προκλήσεις και κινδύνους ασφαλείας. Ένα μόνο παραβιασμένο Docker container μπορεί να απειλήσει όλα τα άλλα container καθώς και τον υποκείμενο κεντρικό υπολογιστή, υπογραμμίζοντας τη σημασία της ασφάλειας του Docker.

Το CIS ερευνά τις βέλτιστες πρακτικές για την ασφάλεια στον κυβερνοχώρο σε περιβάλλοντα με containers. Το CIS δημοσιεύει το Docker CIS Benchmark, μια ολοκληρωμένη λίστα βέλτιστων πρακτικών που μπορούν να βοηθήσουν στην εξασφάλιση στην παραγωγή Docker container [45].

Το Docker CIS Benchmark επικεντρώνεται στη διασφάλιση ότι τα runtime των Docker containers διαμορφώνονται όσο το δυνατόν ασφαλέστερα.

Η ασφάλεια στο Docker μπορεί να χωριστεί σε δύο τομείς:

1. Ασφάλιση και σκλήρυνση του host έτσι ώστε μια παραβίαση container να μην οδηγήσει επίσης σε παραβίαση του κεντρικού υπολογιστή.
2. Ασφάλιση και σκλήρυνση των Docker containers κατά τη διάρκεια της φάσης κατασκευής και ανάπτυξης και προστασίας τους κατά τη διάρκεια του runtime.

Για να βοηθήσει τους διαχειριστές συστήματος και εφαρμογών, ειδικούς ασφαλείας, ελεγκτές, γραφείο βοήθειας και προσωπικό ανάπτυξης πλατφόρμας που σχεδιάζουν να αναπτύξουν, να αναπτύξουν, να αξιολογήσουν ή να εξασφαλίσουν λύσεις που ενσωματώνουν τεχνολογία Docker 1.6 ή μεταγενέστερη, το CIS κυκλοφόρησε το πρώτο Benchmark

διαμόρφωσης ασφαλείας για το Docker 1.6, το οποίο κάνει περισσότερες από 80 συστάσεις για τη διαμόρφωση και τη λειτουργία του Docker σε περιβάλλοντα παραγωγής. Το Benchmark δημιουργήθηκε με συναίνεση με εκπροσώπους των Docker, VMware, Cognitive Scale, International Securities Exchange, Rakuten και CIS.

Το CIS Benchmark για το Docker καλύπτει οκτώ κατηγορίες συστάσεων, οι οποίες θα καλύψουν εδώ σύντομα. Οι συστάσεις αντιστοιχίζονται επίσης στους ελέγχους CIS για να υπάρχει συνέπεια μεταξύ αυτών των βέλτιστων πρακτικών [42-43,45].

1. Host Configuration

Αυτή η ενότητα καλύπτει τις συστάσεις ασφαλείας που πρέπει να ακολουθηθούν για την προετοιμασία του Host Computer που θα χρησιμοποιηθεί για την εκτέλεση εργασιών με containers. Η διασφάλιση του κεντρικού υπολογιστή Docker και η παρακολούθηση των βέλτιστων πρακτικών ασφάλειας της υποδομής θα δημιουργήσει ένα στέρεο και ασφαλές θεμέλιο για την εκτέλεση φόρτων εργασίας με containers [46]. Η ενότητα περιλαμβάνει βέλτιστες πρακτικές όπως:

- Το Docker πρέπει να είναι ενημερωμένο για να μετριάσει τα τρωτά σημεία του λογισμικού.
- Θα πρέπει να έχει δημιουργηθεί ένα ξεχωριστό διαμέρισμα για container για να μη γεμίσει γρήγορα ο κατάλογος /var/lib/docker με αποτέλεσμα τόσο το Docker όσο και ο κεντρικός υπολογιστής να καταστούν άχρηστοι.
- Μόνο οι έμπιστοι χρήστες επιτρέπεται να ελέγχουν το Docker Daemon για να ελαχιστοποιήσουν τα δικαιώματα διαχειριστή και να χρησιμοποιούν λογαριασμούς διαχειριστή μόνο όταν απαιτούνται.
- Κατάλληλη διαμόρφωση για το Docker Daemon, τα Docker Files και Docker Directories για τον έλεγχο των δραστηριοτήτων και της χρήσης του δαίμονα που εκτελείται ως root.

2. **Docker Daemon Configuration**

Αυτή η ενότητα παραθέτει τις συστάσεις που αλλάζουν και διασφαλίζουν τη συμπεριφορά του δαίμονα Docker [46]. Η ενότητα περιλαμβάνει βέλτιστες πρακτικές όπως:

- Βεβαίωση ότι η κυκλοφορία δικτύου είναι περιορισμένη μεταξύ των containers στο default bridge. Σε αντίθετη περίπτωση, μπορεί να οδηγήσει σε ακούσια και ανεπιθύμητη αποκάλυψη πληροφοριών σε άλλα.
- Βεβαίωση ότι το Docker έχει τη δυνατότητα να κάνει αλλαγές στα iptables αυτόματα, προκειμένου να αποφευχθούν εσφαλμένες διαμορφώσεις δικτύωσης που θα μπορούσαν να επηρεάσουν την επικοινωνία μεταξύ containers και με τον έξω κόσμο.
- Βεβαίωση ότι δεν χρησιμοποιούνται μη ασφαλή registries. Ένα ασφαλές registry χρησιμοποιεί TLS. Ένα μη ασφαλές είναι αυτό που δεν διαθέτει έγκυρο πιστοποιητικό ή ένα που δεν χρησιμοποιεί TLS.
- Βεβαίωση ότι ο έλεγχος ταυτότητας TLS για το Docker Daemon έχει διαμορφωθεί
- Ενεργοποίηση της υποστήριξης "user namespace" για την παροχή πρόσθετης ασφάλειας στο σύστημα κεντρικού υπολογιστή Docker επιτρέποντας σε ένα container να έχει ένα μοναδικό εύρος αναγνωριστικών χρηστών και ομάδων που βρίσκονται εκτός του παραδοσιακού εύρους χρηστών και ομάδων.
- Βεβαίωση ότι τα containers είναι περιορισμένα από την απόκτηση νέων προνομίων. Αυτό μειώνει τους κινδύνους ασφαλείας που σχετίζονται με πολλές επικίνδυνες λειτουργίες, επειδή υπάρχει πολύ μειωμένη ικανότητα ανατροπής προνομιακών binary αρχείων.

3. **Docker Daemon Configuration Files**

Αυτή η ενότητα καλύπτει αρχεία και δικαιώματα καταλόγου που σχετίζονται με το Docker και ιδιοκτησία. Η διατήρηση των αρχείων και των καταλόγων, που μπορεί να περιέχουν ευαίσθητες παραμέτρους, είναι σημαντική για τη σωστή και ασφαλή λειτουργία του Docker daemon. Αυτό απαιτείται για την ελαχιστοποίηση των δικαιωμάτων διαχείρισης και τη χρήση λογαριασμών διαχειριστή μόνο όταν απαιτούνται [46].

4. **Container Images and Build File Configuration**

Τα docker base images και τα αρχεία δημιουργίας διέπουν τις βασικές αρχές του τρόπου με τον οποίο θα συμπεριφερόταν ένα container κοντέινερ από ένα συγκεκριμένο image. Η διασφάλιση ότι γίνεται χρήση των κατάλληλων base images και των κατάλληλων αρχείων δημιουργίας μπορεί να είναι πολύ σημαντική για τη δημιουργία της υποδομής με container [46]. Οι συστάσεις περιλαμβάνουν βέλτιστες πρακτικές όπως:

- Εκτέλεση του κοντέινερ ως μη root χρήστης, όπου είναι δυνατόν.
- Βεβαίωση ότι η εμπιστοσύνη περιεχομένου/content trust για το Docker είναι ενεργοποιημένη. Η εμπιστοσύνη περιεχομένου παρέχει τη δυνατότητα χρήσης ψηφιακών υπογραφών για δεδομένα που αποστέλλονται και λαμβάνονται από απομακρυσμένα Docker Registries.
- Βεβαίωση ότι έχουν προστεθεί οδηγίες HEALTHCHECK στα container images. Ένας σημαντικός έλεγχος ασφαλείας είναι αυτός της διαθεσιμότητας. Η προσθήκη της οδηγίας HEALTHCHECK διασφαλίζει ότι το Docker engine ελέγχει περιοδικά τις παρουσίες του container που εκτελούνται σε σχέση με αυτήν την οδηγία για να διασφαλίσει ότι τα containers εξακολουθούν να λειτουργούν.

5. **Container Runtime Configuration**

Υπάρχουν πολλές επιπτώσεις στην ασφάλεια που σχετίζονται με τους τρόπους εκκίνησης των κοντέινερ. Μπορούν να παρέχονται ορισμένες παραμέτρους χρόνου εκτέλεσης που έχουν συνέπειες ασφάλειας που θα μπορούσαν να θέσουν σε κίνδυνο τον κεντρικό υπολογιστή και τα κοντέινερ που λειτουργούν σε αυτόν [46]. Η διαμόρφωση του χρόνου εκτέλεσης του container θα πρέπει να αναθεωρηθεί σύμφωνα με την πολιτική ασφάλειας του οργανισμού.

- Θα πρέπει να είναι ενεργοποιημένο ένα προφίλ AppArmor. Το AppArmor προστατεύει το λειτουργικό σύστημα και τις εφαρμογές από διάφορες απειλές επιβάλλοντας μια πολιτική ασφαλείας η οποία είναι επίσης γνωστή ως προφίλ AppArmor. Η ενεργοποίηση αυτής της

δυνατότητας επιβάλλει πολιτικές ασφαλείας σε container όπως ορίζονται στο προφίλ.

- Οι επιλογές ασφαλείας SELinux θα πρέπει να έχουν οριστεί για να προσθέσουν ένα επιπλέον επίπεδο ασφάλειας στα δοχεία.
- Οι δυνατότητες του Linux Kernel πρέπει να είναι περιορισμένες εντός container, γιατί διαφορετικά μπορούν να δώσουν σε έναν εισβολέα με πρόσβαση σε ένα container τη δυνατότητα να δημιουργήσει πλαστογραφημένη κίνηση δικτύου.
- Δεν πρέπει να χρησιμοποιούνται προνομιακά δοχεία.
- Οι ευαίσθητοι κατάλογοι συστήματος κεντρικού υπολογιστή, όπως /, /boot, /etc δεν τοποθετούνται σε container. Εάν οι ευαίσθητοι κατάλογοι είναι τοποθετημένοι σε λειτουργία ανάγνωσης-εγγραφής, θα μπορούσαν να γίνουν αλλαγές σε αρχεία εντός τους. Αυτό έχει προφανείς επιπτώσεις στην ασφάλεια και πρέπει να αποφευχθεί.
- Το SSH Daemon (sshd) δεν πρέπει να εκτελείται μέσα σε container, επειδή αυξάνει την πολυπλοκότητα της διαχείρισης ασφάλειας.
- Βεβαίωση ότι το network namespace του κεντρικού υπολογιστή δεν είναι κοινόχρηστο. Η επιλογή αυτή είναι δυνητικά επικίνδυνη και θα μπορούσε ενδεχομένως να πραγματοποιήσει ανεπιθύμητες ενέργειες, όπως το κλείσιμο του κεντρικού υπολογιστή Docker. Αυτή η επιλογή δεν πρέπει να χρησιμοποιείται εκτός εάν υπάρχει πολύ συγκεκριμένος λόγος για την ενεργοποίησή της.
- Η χρήση μνήμης για τα container πρέπει να είναι περιορισμένη.
- Το root filesystem του container είναι μόνο για ανάγνωση. Αυτή η επιλογή μειώνει τα attack vectors ασφαλείας, καθώς το σύστημα αρχείων της περίπτωσης κοντέινερ δεν μπορεί να παραβιαστεί ή να γραφτεί, εκτός εάν έχει ρητά δικαιώματα ανάγνωσης-εγγραφής στο φάκελο και τους καταλόγους του συστήματος αρχείων.
- Η εισερχόμενη κίνηση container θα πρέπει να συνδέεται με μια συγκεκριμένη διεπαφή κεντρικού υπολογιστή. Αυτή η διεπαφή μπορεί επίσης να προστατεύεται με υπηρεσίες όπως ανίχνευση εισβολών, πρόληψη εισβολών, τείχος προστασίας, εξισορρόπηση φορτίου κ.λπ. για την παρακολούθηση της εισερχόμενης κίνησης.

6. **Docker Security Operations**

Αυτή η ενότητα καλύπτει ορισμένα από τα ζητήματα λειτουργικής ασφάλειας που σχετίζονται με τις αναπτύξεις Docker. Αυτές είναι οι βέλτιστες πρακτικές που πρέπει να ακολουθούνται όπου είναι δυνατόν. Οι περισσότερες από τις συστάσεις σε αυτήν την ενότητα λειτουργούν απλώς ως υπενθυμίσεις ότι οι οργανισμοί πρέπει να επεκτείνουν τις τρέχουσες βέλτιστες πρακτικές και πολιτικές ασφάλειας, ώστε να περιλαμβάνουν container [46].

7. **Docker Swarm Configuration**

Αυτή η ενότητα παραθέτει τις συστάσεις που αλλάζουν και διασφαλίζουν τη συμπεριφορά του Docker Swarm. Εάν δεν χρησιμοποιείτε το Docker Swarm τότε οι συστάσεις σε αυτήν την ενότητα δεν ισχύουν. Όταν η λειτουργία Docker Swarm είναι ενεργοποιημένη σε μια μηχανή Docker, ανοίγουν πολλαπλές θύρες δικτύου στο σύστημα και διατίθενται σε άλλα συστήματα του δικτύου για σκοπούς διαχείρισης συμπλέγματος και επικοινωνιών κόμβων. Το άνοιγμα θυρών δικτύου σε ένα σύστημα αυξάνει την επιφάνεια επίθεσης και αυτό πρέπει να αποφεύγεται, εκτός εάν απαιτείται [46].

4.4.4.1 Docker Security Benchmark Tools

Το Docker CIS Benchmark παρέχει εκατοντάδες λεπτομερείς συστάσεις για τη διαμόρφωση του Docker. Έχουν δημιουργηθεί εργαλεία για να βοηθήσουν τις οντότητες και τους οργανισμούς να συμμορφωθούν σύμφωνα με αυτές τις συστάσεις. Κάποια από τα πιο δημοφιλή εργαλεία είναι τα ακόλουθα [45].

Docker Bench for Security

Το Docker Bench for Security είναι ένα open source script που ελέγχει τα containers σύμφωνα με τις βέλτιστες πρακτικές του CIS. Πραγματοποιεί δοκιμές βάσει CIS Benchmarks και καταγράφει τα ευρήματά.

Για κάθε Benchmark, το εργαλείο παρέχει πληροφορίες, όπως τα προβλήματα που βρέθηκαν και ενημερώνει με προειδοποίηση, δηλαδή ότι το container δεν ικανοποιεί τη σύσταση, ή με Pass, που σημαίνει ότι το container την ικανοποιεί. Μπορείτε να εκτελεστεί από το Docker host ή να γίνει clone με το Docker Compose.

OpenSCAP Workbench

Το OpenSCAP περιλαμβάνει πολλαπλές κατευθυντήριες γραμμές αναφοράς ασφαλείας, κριτήρια διαμόρφωσης και εργαλεία ανοιχτού κώδικα που μπορούν να βοηθήσουν στη δοκιμή ζητημάτων ασφαλείας, συμπεριλαμβανομένου του CIS Benchmark. Επικεντρώνεται στο πρωτόκολλο Secure Content Automation (SCAP) με πιστοποίηση NIST, το οποίο περιλαμβάνει πολλές αυτοματοποιημένες πολιτικές ασφαλείας.

Το OpenSCAP πηγαίνει ευρύτερα από τις συστάσεις του CIS, συμπεριλαμβανομένων πολλών άλλων συστάσεων, μερικές από τις οποίες δεν αφορούν συγκεκριμένα περιβάλλοντα με container. Μπορεί να είναι χρήσιμο για τον εντοπισμό πρόσθετων προβλημάτων ασφάλειας που δεν καλύπτονται από τις κατευθυντήριες γραμμές του CIS.

Anchore

Το Anchore Engine είναι ένα εργαλείο για την ανάλυση container images. Μπορεί να εντοπίσει ευπάθειες που βασίζονται σε CVE, και επιτρέπει επίσης στους χρήστες να ορίσουν προσαρμοσμένες πολιτικές και να τις χρησιμοποιήσουν για την αξιολόγηση των Docker images. Οι πολιτικές μπορούν να βασίζονται στη λίστα επιτρεπόμενων, τη μαύρη λίστα, τα διαπιστευτήρια, το περιεχόμενο του αρχείου και τις διαμορφώσεις.

Ενώ το Anchore δεν υποστηρίζει επίσημα τις κατευθυντήριες γραμμές αναφοράς CIS, μπορούν να οριστούν προσαρμοσμένες πολιτικές για να καλύψουν όλες τις προτάσεις αναφοράς. Για κάθε πολιτική, το Anchore επιστρέφει ένα αποτέλεσμα επιτυχίας ή αποτυχίας.

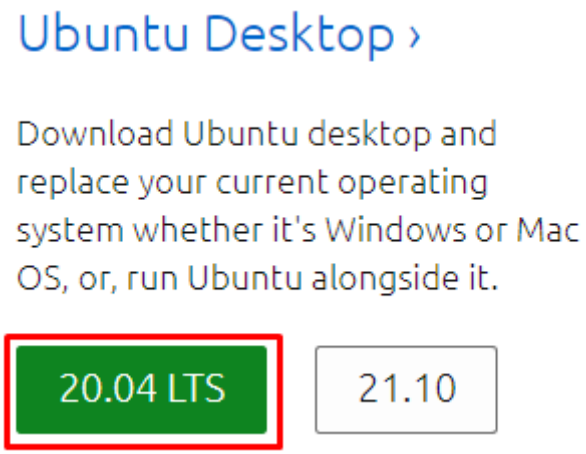
Το Anchore μπορεί να εκτελεστεί ως Docker image container, εντός Kubernetes ή ως αυτόνομο binary. Ενσωματώνεται με δημοφιλή εργαλεία CI/CD όπως το Jenkins και το GitLab.

Κεφάλαιο 5. Δημιουργία Περιβάλλοντος

Στο συγκεκριμένο κεφάλαιο θα παρουσιαστεί η δημιουργία του περιβάλλοντος στο οποίο θα πραγματοποιηθεί το Case Study. Θα χρησιμοποιήσουμε VMWare Workstation Pro και Ubuntu 20.04 LTS, καθώς το χρόνο της σύνταξης της διπλωματικής ήταν η τελευταία LTS Έκδοση.

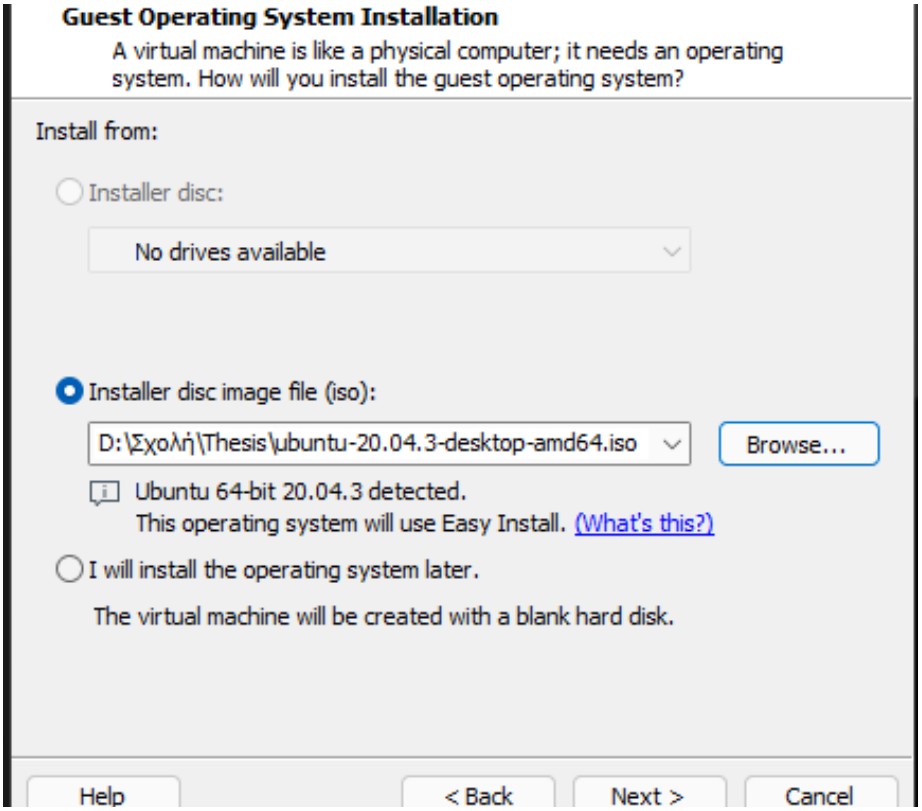
5.1 Δημιουργία VM

Αρχικά θα κατεβάσουμε το Αρχείο Εικόνας Δίσκου για το Ubuntu 20.04 LTS από το επίσημο site (https://ubuntu.com/).



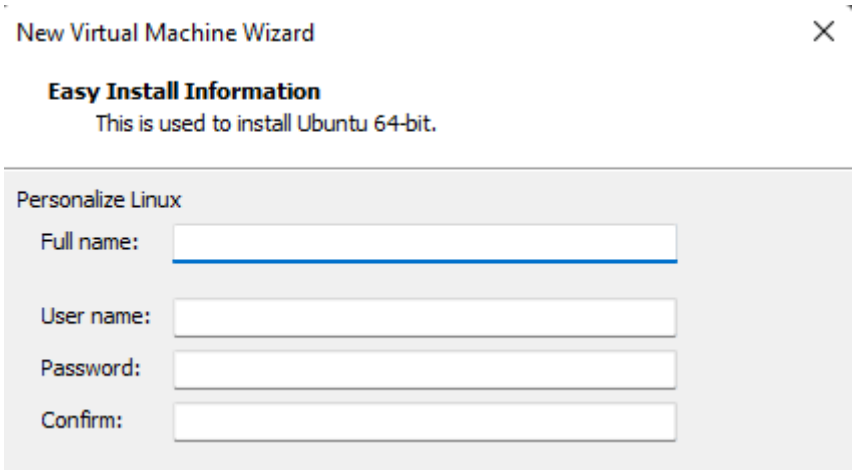
Εικόνα 5 Επιλογή της τελευταίας LTS Έκδοσης

Στη συνέχεια ανοίγουμε το VMWare Workstation Pro, επιλέγουμε το File και έπειτα το New Virtual Machine. Όταν ανοίξει το Wizard επιλέγουμε Advanced και πατάμε Next. Σε αυτό το στάδιο θα επιλέξουμε το ISO που κατεβάσαμε και θα πατήσουμε Next.



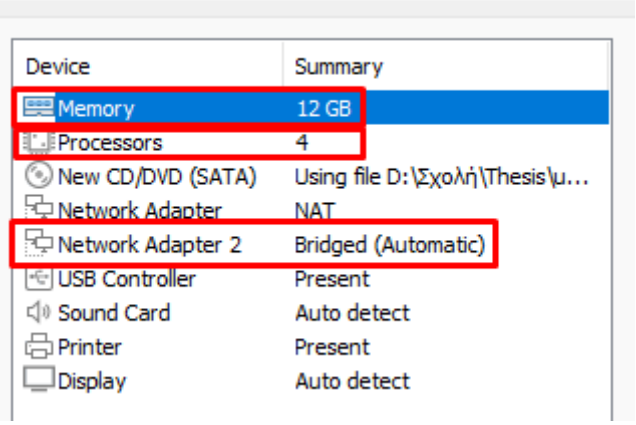
Εικόνα 6 Επιλογή ISO για εγκατάσταση

Το επόμενο στάδιο αφορά το Όνομα, το Username και το κωδικό του VM.



Εικόνα 7 VM Credentials

Μετά από αυτό, θα ρυθμίσουμε του πόρους που θα μπορεί να χρησιμοποιεί το VM. Θα αυξήσουμε τη RAM, τον αριθμό των Πυρήνων και θα προσθέσουμε ακόμα ένα Network Adapter, ορίζοντάς το ως Bridged.

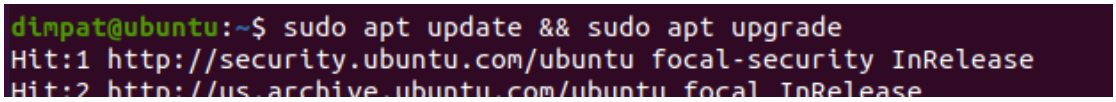


Εικόνα 8 VM Resources

Τώρα τελειώσαμε με τη ρύθμιση του VM από το VMware, πατάμε εγκατάσταση και αρχίζει η διαδικασία και αφήνουμε να κάνει τις απαραίτητες διεργασίες

Εικόνα 9 Διαδικασία Εγκατάστασης

Όπως έχουμε αναφέρει στα προηγούμενα κεφάλαια, σαν βέλτιστη πρακτική είναι η διατήρηση του περιβάλλοντος ενημερωμένο. Χρησιμοποιούμε το CLI για να ενημερώσουμε το σύστημα και να εγκαταστήσουμε τις τελευταίες αλλαγές.



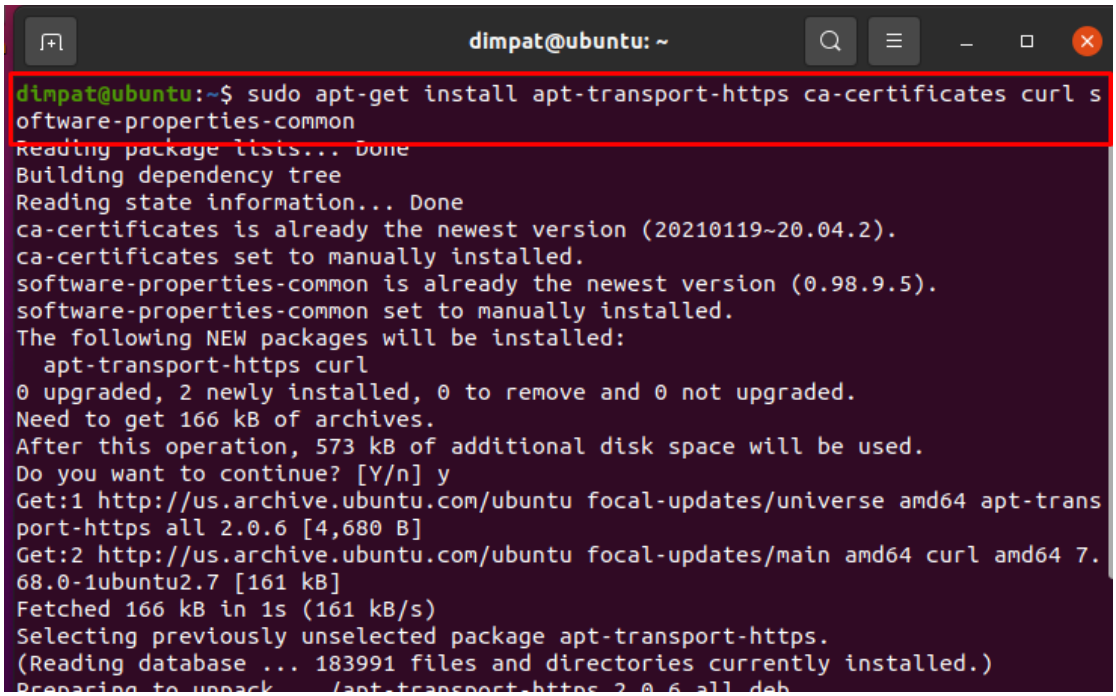
Εικόνα 10 Ενημέρωση του Συστήματος

Το λειτουργικό σύστημα είναι πλέον πλήρως ενημερωμένο, οπότε μπορούμε να εγκαταστήσουμε το Docker.

5.2 Εγκατάσταση Docker

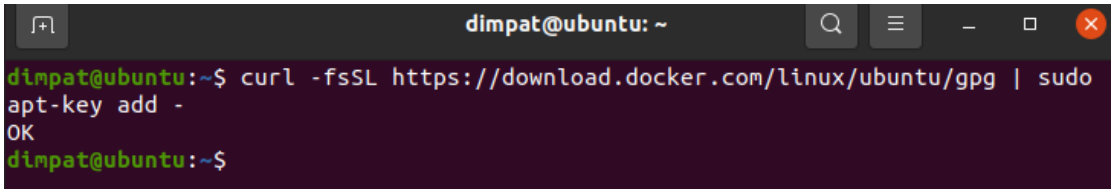
Θα χρησιμοποιήσουμε το official Docker Repository για να εγκαταστήσουμε το Docker στο VM. Αφού ξέρουμε ότι το σύστημά μας είναι ενημερωμένο, μπορούμε να εγκαταστήσουμε τα απαραίτητα dependencies Θα πρέπει να επιτρέψουμε στο Ubuntu να έχει πρόσβαση στο repository μέσω HTTPS.

Για να γίνει αυτό, θα πρέπει να δώσουμε άδεια στο package manage να μεταφέρει αρχεία και δεδομένα και να αφήσουμε το σύστημα να ελέγχει τα πιστοποιητικά ασφαλείας. Επιπλέον θα εγκαταστήσουμε το curl και θα προσθέσουμε ένα script για τη διαχείριση λογισμικού. Η εντολή για τις παραπάνω ενέργειες φαίνεται στην Εικόνα 11.



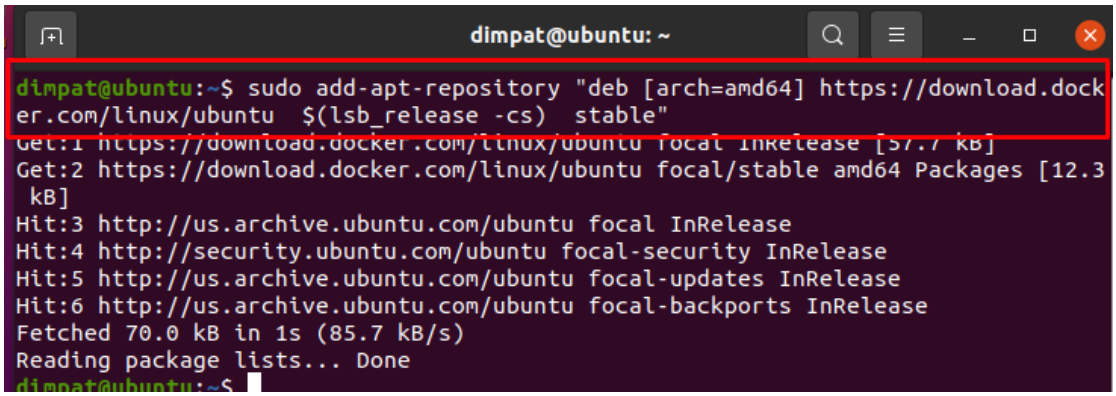
Εικόνα 11 Εγκατάσταση των απαραίτητων Dependencies

Το επόμενο βήμα είναι τα προσθέσουμε το GPG κλειδί του Docker όπως απεικονίζεται στην Εικόνα 12.



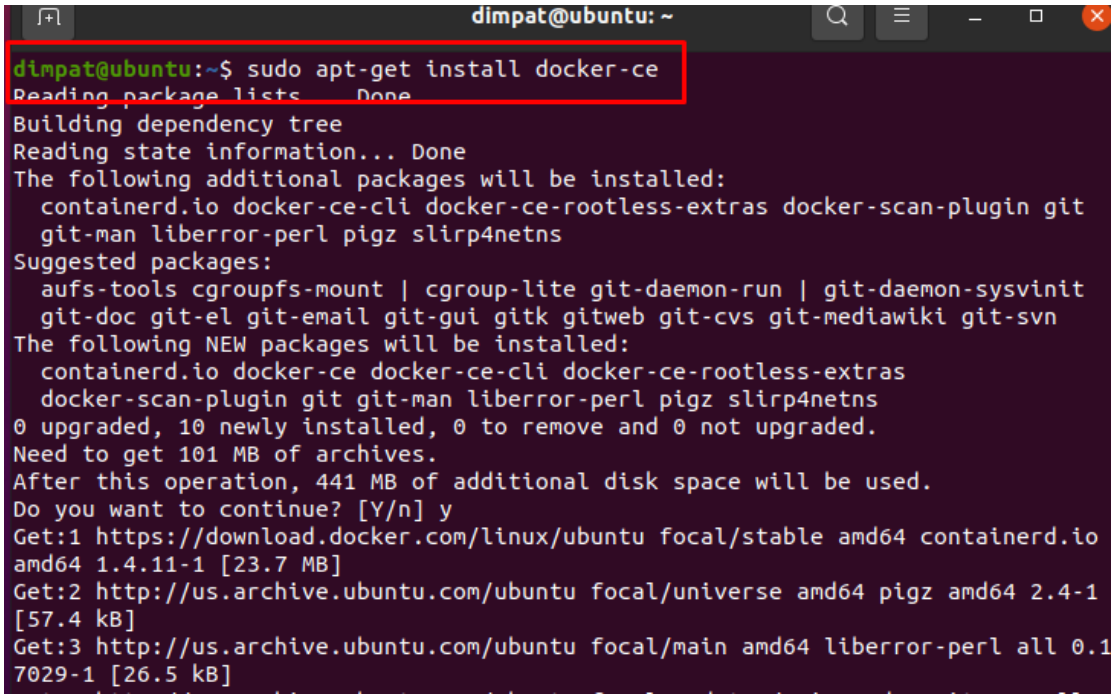
Εικόνα 12 Προσθήκη του κλειδιού GPG του Docker

Μετά από αυτό, θα εγκαταστήσουμε το Docker Repository. Η εντολή φαίνεται στη Εικόνα 13.



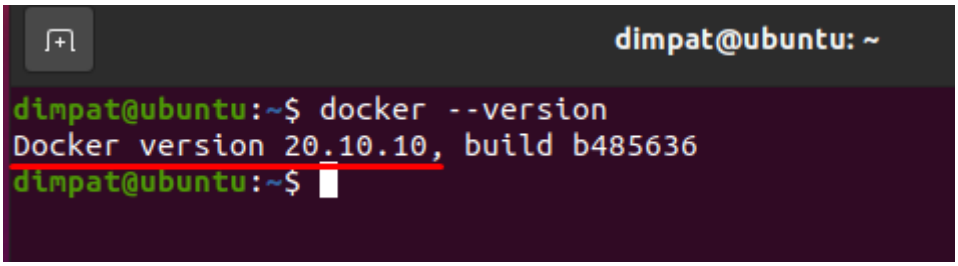
Εικόνα 13 Εγκατάσταση του Docker Repository

Στη συνέχεια θα εγκαταστήσουμε τη τελευταία έκδοση του Docker με τη εντολή που παρουσιάζεται στην Εικόνα 14.



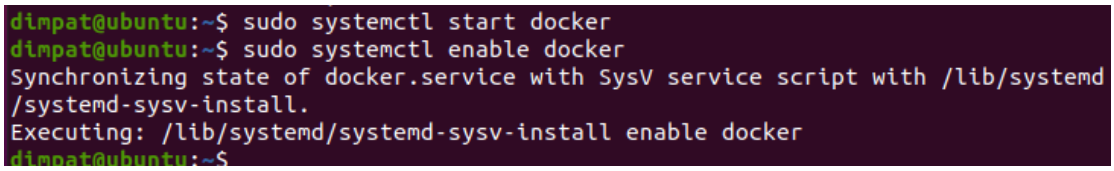
Εικόνα 14 Εγκατάσταση Docker

Τέλος θα ελέγξουμε την έκδοση του Docker, όπως στην Εικόνα 15.



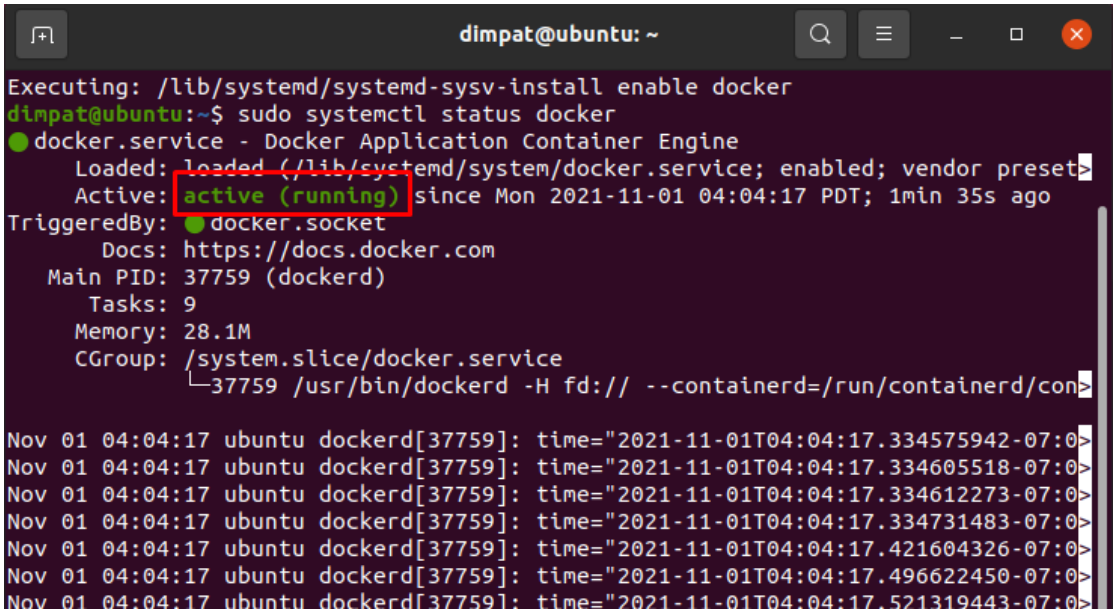
Εικόνα 15 Έλεγχος Έκδοσης

Αφού βεβαιωθούμε πως έχουμε την τελευταία έκδοση με το σωστό build, μπορούμε να ξεκινήσουμε το Docker Service. Οι εντολές για αυτό φαίνονται στην Εικόνα 16.



Εικόνα 16 Έναρξη του Docker Service

Μπορούμε να ελέγξουμε το status του service με την εντολή της Εικόνας 17.



Εικόνα 17 Έλεγχος Status

Η δημιουργία του περιβάλλοντος και η εγκατάσταση του Docker έχει ολοκληρωθεί. Μπορούμε να χρησιμοποιήσουμε το Docker το command:

sudo docker [option] [command] [argument]

Κεφάλαιο 6. Περιγραφή Εργαλείων

Σε αυτή τη διπλωματική εργασία θα ασχοληθούμε με open-source tools. Ο όρος open-source αναφέρεται σε κάτι που οι άνθρωποι μπορούν να τροποποιήσουν και να μοιραστούν επειδή ο σχεδιασμός του είναι δημόσια προσβάσιμος. Ένα open-source tool δεν είναι απαραίτητα και δωρεάν. Πολλοί πάροχοι μπορούν να προσφέρουν διαφορετικά tier ενός τέτοιου εργαλείου, το οποίο μπορεί ένα tier να είναι δωρεάν για όλους και ένα άλλο να είναι επί πληρωμή αλλά να παρέχει περισσότερα features ή να παρέχουν κάποιο έξτρα support.

Υπάρχουν πολλοί λόγοι που κάποιος μπορεί να επιλέξει το open-source ενάντια σε κάποια commercial λύση. Οι πιο βασικοί λόγοι είναι οι ακόλουθοι:

- Ευελιξία
- Έλεγχος
- Εκπαίδευση
- Ασφάλεια
- Σταθερότητα
- Κοινότητα

6.1 Nginx

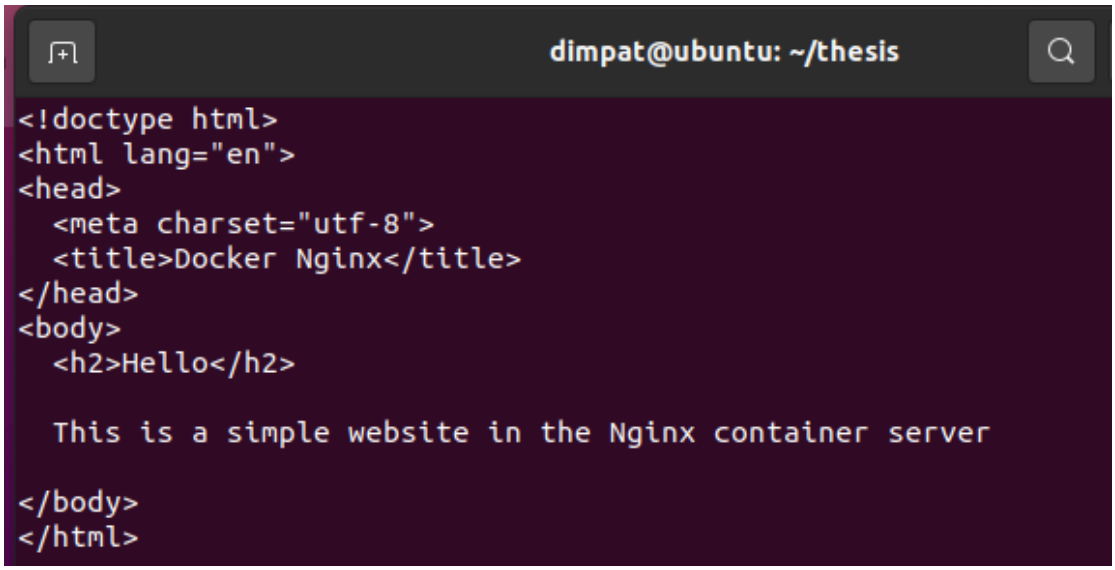
Το nginx, προφέρεται “engine-x” είναι ένα open-source λογισμικό που δημιουργήθηκε από τον Igor Sysoev και κυκλοφόρησε δημόσια το 2004. Μπορεί να χρησιμοποιηθεί για web service, reverse proxy, caching, load balancing, media streaming και πολλά άλλα. Ξεκίνησε ως web server σχεδιασμένος για μέγιστη απόδοση και σταθερότητα. Εκτός από τις δυνατότητες HTTP server, το NGINX μπορεί επίσης να λειτουργήσει ως proxy server για email (IMAP, POP3 και SMTP) και reverse proxy server και load balancer για HTTP servers, TCP και UDP [47].

Μια εταιρεία με το ίδιο όνομα ιδρύθηκε το 2011 για να παρέχει υποστήριξη και λογισμικό επί πληρωμή Nginx Plus.

Σύμφωνα με τα στατιστικά της σελίδας w3techs, το nginx ήταν πρώτο σε χρήση για webservers το Νοέμβριο 2021, με ποσοστά 33.5% και 33%, ενώ τη δεύτερη θέση είχε το Apache με ποσοστό 31.5% [48].

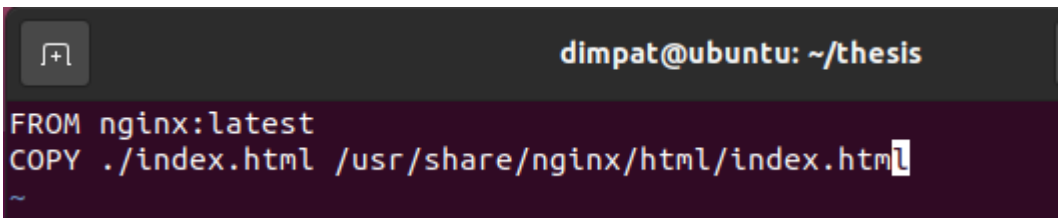
6.1.1 Nginx Docker Image

Θα χρησιμοποιηθεί το nginx για να δημιουργήσουμε ένα docker webserver. Θα χρησιμοποιήσουμε το docker image του nginx για να φτιάξουμε το server. Το πρώτο βήμα είναι να φτιάξουμε το απλό html αρχείο που θα ανεβάσουμε στο server. Η μορφή του αρχείου φαίνεται στην Εικόνα 18.



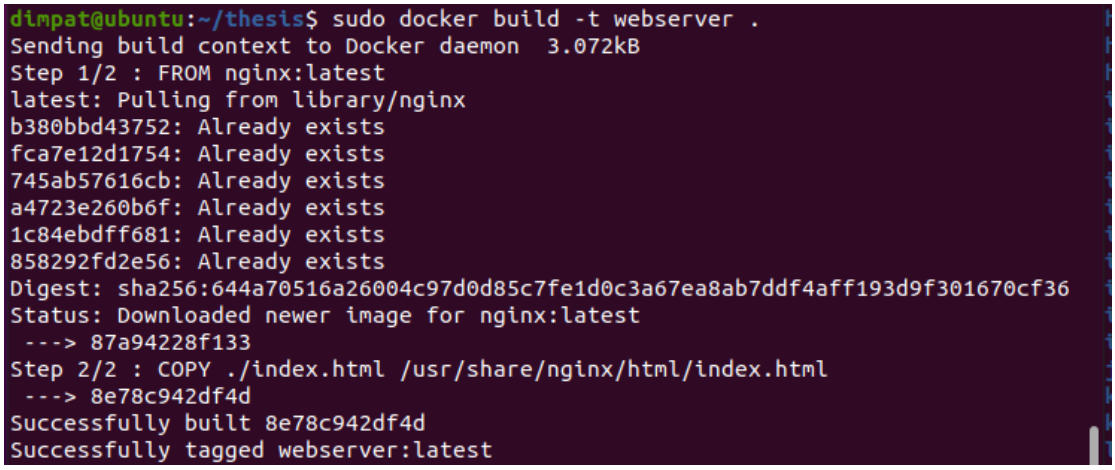
Εικόνα 18 HTML Αρχείο

Στη συνέχεια θα πρέπει να φτιάξουμε το dockerfile για να δείξουμε από που θα πάρουμε το image, και να δώσουμε το path για το html αρχείο, όπως φαίνεται στην Εικόνα 19.



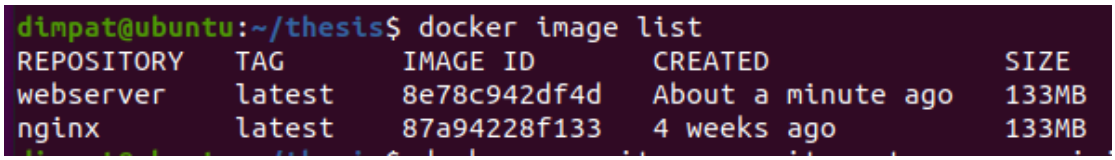
Εικόνα 19 Dockerfile

Έπειτα θα κάνουμε built το dockerfile για να δημιουργήσουμε το custom image. Η εντολή, docker build, φαίνεται στην Εικόνα 19.



Εικόνα 20 Docker Build

Τέλος, μπορούμε να ελέγξουμε ότι το image έχει δημιουργηθεί με την εντολή η οποία απεικονίζεται στην Εικόνα 21.



Εικόνα 21 Το Docker Image είναι έτοιμο

Πλέον, το docker image, μπορούμε να το χρησιμοποιήσουμε με την εντολή ***sudo docker run –d –p 8080 webserver,*** αλλά στο επόμενο κεφάλαιο θα δείξουμε ποιες είναι οι βέλτιστες πρακτικές για τη σωστή χρήση της εντολής ***-run***.

6.2 Seccmomp

Το seccomp (secure computing mode) είναι μια ασφάλεια υπολογιστή στο Linux kernel. Το seccomp επιτρέπει σε μια διεργασία να κάνει μια μονόδρομη μετάβαση σε μια "ασφαλή" κατάσταση όπου δεν μπορεί να πραγματοποιήσει κλήσεις συστήματος εκτός από exit(), sigreturn(), read() και write() σε ήδη ανοιχτούς file descriptor. Εάν επιχειρήσει άλλες κλήσεις συστήματος, ο πυρήνας θα τερματίσει τη διαδικασία με το SIGKILL ή το SIGSYS. Υπό αυτή την έννοια, δεν εικονικοποιεί τους πόρους του συστήματος, αλλά απομονώνει πλήρως τη διαδικασία από αυτούς [49,50].

Η λειτουργία Seccomp ή Secure Computing, συνοπτικά, είναι μια δυνατότητα του πυρήνα Linux που μπορεί να λειτουργήσει ως φίλτρο syscall. Το Seccomp έχει 2 λειτουργίες.

1. Η λειτουργία seccomp ενεργοποιείται μέσω της κλήσης συστήματος prctl(2) χρησιμοποιώντας το όρισμα PR_SET_SECCOMP
2. Το seccomp-bpf είναι μια επέκταση του seccomp που επιτρέπει το φιλτράρισμα των κλήσεων συστήματος χρησιμοποιώντας μια διαμορφώσιμη πολιτική που εφαρμόζεται χρησιμοποιώντας κανόνες φίλτρου πακέτων Berkeley. Χρησιμοποιείται από το OpenSSH και το vsftpd καθώς και από τα προγράμματα περιήγησης ιστού Google Chrome/Chromium σε Chrome OS και Linux. Το Seccomp-bpf υποστηρίζεται από το Docker για τον περιορισμό των syscalls από τα κοντέινερ μειώνοντας αποτελεσματικά την επιφάνεια.

6.3 Apparmor

Το AppArmor είναι ένα σύστημα Υποχρεωτικού Ελέγχου Πρόσβασης/Mandatory Access Control (MAC) που είναι μια βελτίωση του kernel (LSM) για τον περιορισμό προγραμμάτων σε ένα περιορισμένο σύνολο πόρων. Το μοντέλο ασφαλείας του AppArmor είναι να δεσμεύει χαρακτηριστικά ελέγχου πρόσβασης σε προγράμματα και όχι σε χρήστες. Ο περιορισμός του AppArmor παρέχεται μέσω προφίλ που φορτώνονται στο kernel, συνήθως κατά την εκκίνηση. Τα προφίλ AppArmor μπορούν να είναι σε μία από τις δύο λειτουργίες: επιβολή και καταγγελία. Τα προφίλ που φορτώνονται σε λειτουργία επιβολής θα έχουν ως αποτέλεσμα την επιβολή της πολιτικής που ορίζεται στο προφίλ καθώς και την αναφορά προσπαθειών παραβίασης πολιτικής, είτε μέσω syslog είτε μέσω ελέγχου. Τα προφίλ σε λειτουργία καταγγελίας δεν θα επιβάλλουν την πολιτική, αλλά θα αναφέρουν προσπάθειες παραβίασης πολιτικής [51].

Το AppArmor διαφέρει από ορισμένα άλλα συστήματα MAC στο Linux γιατί βασίζεται σε paths και επιτρέπει τη μίξη προφίλ επιβολής και παραπόνων. Χρησιμοποιεί αρχεία για να διευκολύνει την ανάπτυξη και έχει πολύ χαμηλότερο εμπόδιο εισόδου από άλλα δημοφιλή συστήματα MAC [52].

Το default-docker profile παρέχει:

- Πρόσβαση σε όλα τα δίκτυα.
- Δεν ορίζει κανένα capability.
- Δεν επιτρέπεται η εγγραφή σε οποιοδήποτε αρχείο /proc.
- Δεν επιτρέπεται η πρόσβαση σε άλλους υποκαταλόγους/αρχεία των /proc και /sys για ανάγνωση/εγγραφή/κλείδωμα/σύνδεσμο/εκτέλεση.
- Δεν επιτρέπεται το mount

6.4 Capabilities

Τα capabilities του Linux παρέχουν ένα υποσύνολο των διαθέσιμων δικαιωμάτων root σε μια διεργασία. Αυτό διαχωρίζει αποτελεσματικά τα δικαιώματα root σε μικρότερες και διακριτικές μονάδες. Κάθε μία από αυτές τις μονάδες μπορεί στη συνέχεια να παραχωρηθεί ανεξάρτητα σε διεργασίες. Με αυτόν τον τρόπο μειώνεται το πλήρες σύνολο των προνομίων και μειώνονται οι κίνδυνοι εκμετάλλευσης.

Επιτρέπουν καλύτερο έλεγχο για τις δυνατότητες που μπορούν να επιτραπούν στον χρήστη root και το docker τα χρησιμοποιεί για να περιορίσει τις λειτουργίες που μπορούν να γίνουν μέσα σε ένα container, ανεξάρτητα από τον τύπο του χρήστη [53].

6.5 Cgroups

Το cgroups (control groups) είναι μια δυνατότητα του Linux kernel που περιορίζει, λαμβάνει υπόψη και απομονώνει τη χρήση πόρων (CPU, μνήμη, είσοδος/εξόδου δίσκου, δίκτυο, κ.λπ.) μιας συλλογής διεργασιών.

Οι μηχανικοί της ξεκίνησαν την εργασία σε αυτό το χαρακτηριστικό το 2006 με το όνομα "process containers". Στα τέλη του 2007, η ονοματολογία άλλαξε σε "control groups" για να αποφευχθεί η σύγχυση που προκαλείται από πολλαπλές έννοιες του όρου "container" στο πλαίσιο του Linux kernel και η λειτουργικότητα των ομάδων ελέγχου συγχωνεύτηκε στην κύρια γραμμή του kernel στην έκδοση πυρήνα 2.6.24. που κυκλοφόρησε τον Ιανουάριο του 2008.

Από τότε, οι προγραμματιστές έχουν προσθέσει πολλές νέες δυνατότητες και ελεγκτές, όπως υποστήριξη για kernf το 2014, firewall και unified hierarchy. Το cgroup v2 συγχωνεύτηκε στον πυρήνα Linux 4.5 με σημαντικές αλλαγές στη διεπαφή και την εσωτερική λειτουργικότητα [54].

Τα cgroups μπορούν να χρησιμοποιηθούν με πολλούς τρόπους:

- Με μη αυτόματη πρόσβαση στο εικονικό σύστημα αρχείων cgroup.
- Δημιουργώντας και διαχειρίζοντας ομάδες χρησιμοποιώντας εργαλεία όπως cgcreate, cgexec και cgclassify (από το libcgroup).
- Μέσω του "rules engine daemon" που μπορεί να μετακινήσει αυτόματα διαδικασίες ορισμένων χρηστών, ομάδων ή εντολών σε cgroups όπως καθορίζεται στη διαμόρφωσή του.
- Έμμεσα μέσω άλλου λογισμικού που χρησιμοποιεί cgroups, όπως το Docker, Firejail, LXC, libvirt, systemd, Open Grid Scheduler/Grid Engine και το αναπτυξιακά ανενεργό lmctfy της Google.

6.6 Wazuh

Το Wazuh είναι μια δωρεάν, open-source και έτοιμη για επιχειρήσεις λύση παρακολούθησης ασφάλειας για ανίχνευση απειλών, παρακολούθηση ακεραιότητας, απόκριση συμβάντων και συμμόρφωση [55]. Τα κυριότερα στοιχεία του είναι τα:

- Security Analytics
- Intrusion Detection
- Log Data Analysis
- File Integrity Monitoring
- Vulnerability Detection
- Configuration Assessment
- Incident Response
- Regulatory Compliance
- Cloud Security
- Containers Security

Το Wazuh παρέχει ορατότητα ασφαλείας στους κεντρικούς υπολογιστές και τα docker container, παρακολουθώντας τη συμπεριφορά τους και εντοπίζοντας απειλές, τρωτά σημεία και ανωμαλίες. Το Wazuh agent έχει εγγενή ενοποίηση με το Docker engine που επιτρέπει στους χρήστες να παρακολουθούν images, volumes, ρυθμίσεις δικτύου και container που τρέχουν.

Το Wazuh συλλέγει και αναλύει συνεχώς λεπτομερείς πληροφορίες χρόνου εκτέλεσης. Για παράδειγμα, ειδοποίηση για container που εκτελούνται σε προνομιακή λειτουργία, ευάλωτες εφαρμογές, shell που εκτελείται σε container, αλλαγές σε persistent volumes ή images και άλλες πιθανές απειλές [55].

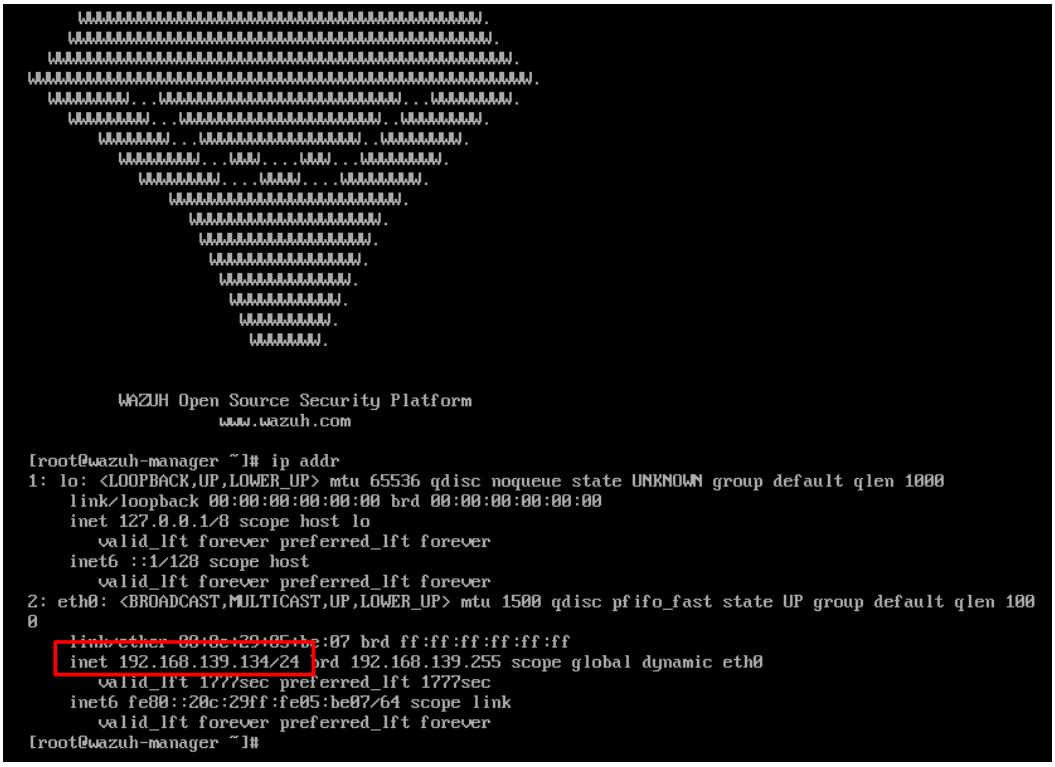
Υπάρχουν δύο διαφορετικές επιλογές για την ανάπτυξη του Wazuh:

1. All-in-one: Το Wazuh server και το Elastic Stack, ένα από τα πιο δημοφιλή open-source εργαλεία για τη καταγραφή δεδομένων, είναι εγκατεστημένα σε ένα έτοιμο προς χρήση OVA.
2. Distributed: Κάθε στοιχείο εγκαθίσταται σε ξεχωριστό υπολογιστή ως σύμπλεγμα ενός κόμβου ή πολλών κόμβων.

Το OVA που παρέχεται αποτελείται από τα ακόλουθα στοιχεία:

- CentOS 7
- Wazuh manager: 4.2.5
- Open Distro για Elasticsearch: 1.13.2
- Filebeat-OSS: 7.10.2
- Kibana: 7.10.2
- Πρόσθετο Wazuh Kibana: 4.2.5-7.10.2

Εμείς θα χρησιμοποιήσουμε το OVA, για να δημιουργήσουμε ένα νέο VM πάνω στο οποίο θα τρέχει το SIEM. Αυτό θα το κάνουμε γιατί, δε πρέπει να τρέχουν όλα τα συστήματα παρακολούθησης στον ίδιο server. Επιπλέον, ένα SIEM που καταγράφει συνέχεια πληροφορίες και δεδομένα χρειάζεται δικό του storage volume. Μόλις κάνουν import το Wazuh OVA, μπορούμε να το ανοίξουμε και μόλις βάλουμε τα απαραίτητα credentials, μπορούμε να δούμε πάνω σε ποια IP τρέχει, όπως φαίνεται στην Εικόνα 22.



Εικόνα 22 Wazuh Manager είναι έτοιμο προς χρήση

6.7 Anchore

Το Anchore Engine, είναι ένα open-source λογισμικό για επιθεώρηση, ανάλυση και πιστοποίηση container images. Το anchore μπορεί να χωριστεί σε δύο εργαλεία, το Syft και το Grype.

Το Syft δημιουργεί ένα ολοκληρωμένο Software Bill of Materials (SBOM) με τη χρήση του CLI. Παρέχει “ορατότητα” σε file level και αυτοματοποιήσει το SBOM στο CI/CD pipeline του οργανισμού.

Το Grype είναι ένα εργαλείο CLI και γραμμένο στη Go για τη δημιουργία ενός SBOM από images και συστήματα αρχείων. Είναι ένας σαρωτής ευπάθειας για εικόνες κοντέινερ και συστήματα αρχείων. Παρέχει γρήγορη σάρωση και δημιουργία μιας λίστας γνωστών τρωτών σημείων από ένα SBOM, ένα image ή έναν κατάλογο έργου [56].

6.8 Dockerbench Security

Το Docker Bench for Security είναι ένα script που ελέγχει δεκάδες κοινές βέλτιστες πρακτικές σχετικά με την ανάπτυξη Docker container. Οι δοκιμές είναι όλες αυτοματοποιημένες και βασίζονται στο CIS Docker Benchmark v1.3.1.

Είναι διαθέσιμο ως βοηθητικό πρόγραμμα, ώστε η κοινότητα του Docker να μπορεί να έχει έναν εύκολο τρόπο να αυτοαξιολογήσει τους κεντρικούς υπολογιστές και τα container σε σχέση με το CIS Benchmark [58].

Από προεπιλογή, το Docker Bench for Security θα εκτελεί όλες τις διαθέσιμες δοκιμές CIS και θα παράγει αρχεία καταγραφής στον τρέχοντα κατάλογο με το όνομα docker-bench-security.sh.log.json και docker-bench-security.sh.log.

Το Docker Bench for Security επιτρέπει στους διαχειριστές να δημιουργήσουν μια ασφαλή γραμμή βάσης στη διαδικασία ανάπτυξης του Docker.

Σαρώνει τον υπολογιστή για κοινά ζητήματα διαμόρφωσης, όπως χαλαρές ρυθμίσεις σε αρχεία διαμόρφωσης, δικαιώματα συστήματος και αμφισβητήσιμες προεπιλογές. Το εργαλείο βασίζεται σε μια βάση δεδομένων Common Vulnerabilities and Exposures (CVE) για τον έλεγχο των βιβλιοθηκών και των εκτελέσιμων στο εν λόγω σύστημα.

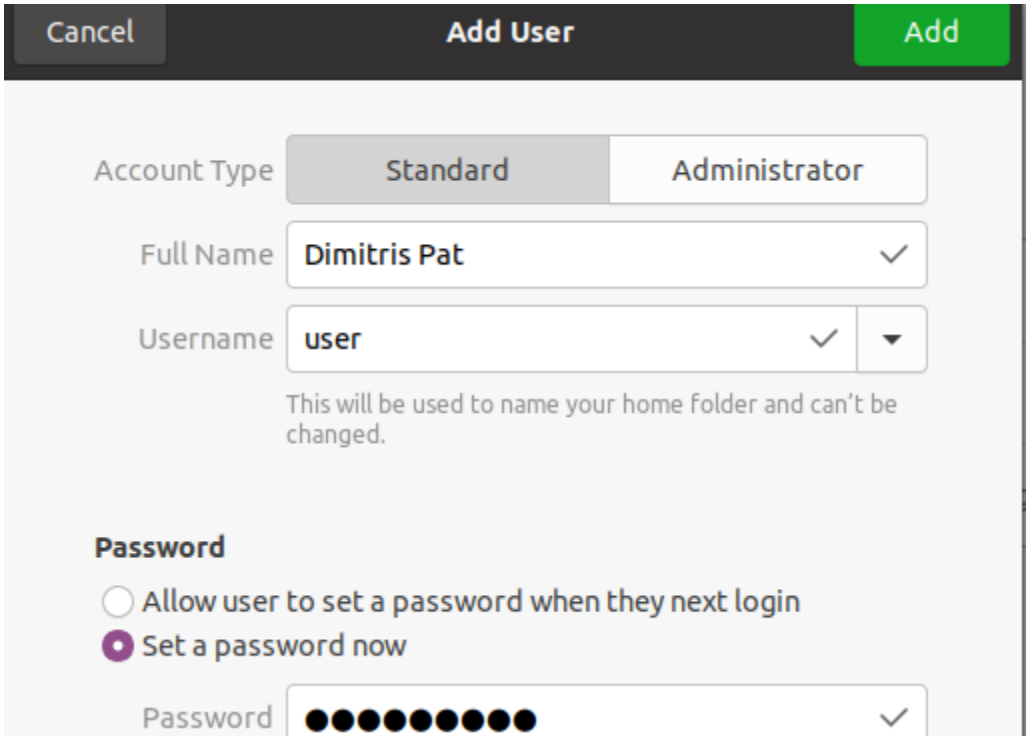
Στο τέλος κάθε σάρωσης, παρέχει μια βαθμολογία. Οι διαχειριστές μπορούν να παρακολουθούν τη βαθμολογία Docker Bench for Security μιας διαμόρφωσης κεντρικού υπολογιστή για να επισημαίνουν βελτιώσεις με την πάροδο του χρόνου [58].

Κεφάλαιο 7. Docker Hardening, Security Auditing and Compliance Case Study

Σε αυτό το κεφάλαιο θα εφαρμόσουμε τις βέλτιστες πρακτικές για έναν docker server, τις οποίες τις αναλύσαμε στο Κεφάλαιο 4. Στη συνέχεια θα κάνουμε deploy ένα SIEM για να έχουμε τη δυνατότητα να παρακολουθούμε το σύστημα. Τέλος, θα ελέγξουμε το περιβάλλον με auditing tools για να δούμε αν οι βέλτιστες πρακτικές που έχουμε ακολουθήσει είναι σωστές.

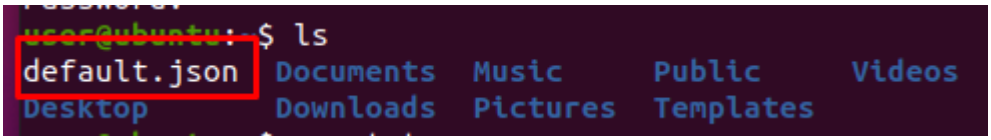
7.1 Docker Container Hardening

Είχαμε δημιουργήσει ένα nginx image με όνομα webserver (βλ. Κεφάλαιο 6). Όπως είχαμε αναφέρει, ως βέλτιστη πρακτική είναι να μην τρέχουμε το docker container ως root και σε ένα namespace. Υπάρχουν πολλά διαφορετικά είδη namespaces όπως το process isolation, network interface, user, unix timesharing system και άλλα. Εμείς θα χρησιμοποιήσουμε το user namespace και με αυτό το τρόπο επιτυγχάνουμε και να τρέχουμε το docker ως no root user. Αρχικά θα πρέπει να φτιάξουμε έναν δεύτερο user ο οποίος δε θα πρέπει να έχει root privileges. Αυτή η διαδικασία φαίνεται στην Εικόνα 23.



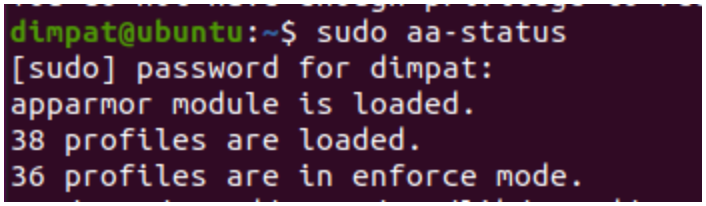
Εικόνα 23 Δημιουργία non root user

Το επόμενο βήμα είναι να προσθέσουμε τα seccomp και apparmor profiles. Επειδή ο user δεν μπορείς να διαβάσει το seccomp προφίλ από τον root χρήστη, θα πρέπει να περάσουμε το default.json αρχείο στο φάκελο του user, όπως παρουσιάζεται στην Εικόνα 24. Το seccmomp μπορούμε να το προσθέσουμε στην ενεργοποίηση του container με το flag ***--security-opt seccomp=default.json***.

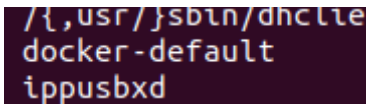


Εικόνα 24 Default seccomp profile

Έπειτα, μέσα από το χρήστη με δικαιώματα root μπορούμε να δούμε τα apparmor profiles που έχει το σύστημα μας το οποίο φαίνεται στην Εικόνα 25 και να ελέγξουμε αν υπάρχει και το docker profile το οποίο απεικονίζεται στην Εικόνα 26. Το apparmor μπορούμε να το προσθέσουμε στην ενεργοποίηση του container με το flag ***--security-opt apparmor=docker-default***.

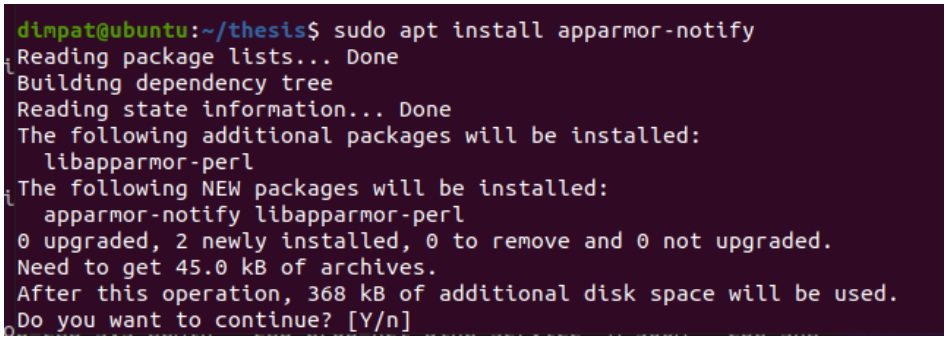


Εικόνα 25 Apparmor profiles



Εικόνα 26 Apparmor docker-default profile

Σε αυτό το στάδιο μπορούμε να εγκαταστήσουμε το apprmor-notify όπως φαίνεται στην Εικόνα 27.



Εικόνα 27 Εγκατάσταση του apparmor-notify

Αφού πλέον έχουμε φτιάξει το χρήστη και έχουμε τα απαραίτητα tools, μπορούμε να εστιάσουμε τη προσοχή μας στην εντολή που θα χρησιμοποιήσουμε για να τρέξουμε το docker container server. Το πρώτο πράγμα που πρέπει να κάνουμε είναι να βρούμε ένα όνομα για το container. Θα το ονομάσουμε thweb που προέρχεται από το **th**esis **web**server.

Μια ακόμα βέλτιστη πρακτική είναι να κάνουμε drop τα capabilities τα οποία δε χρειάζεται το container μας. Θα αρχίσουμε από τα πιο σημαντικά που είναι το cap_sys_admin και θα αφαιρέσουμε επίσης το net_bind_service αφού το container θα το βάλουμε τα τρέχει στο port 8080 και δε χρειάζεται να έχει πρόσβαση στα port κάτω από το 1024 [18]. Τα Capabilities μπορούμε να το αφαιρέσουμε κατά την ενεργοποίηση του container, προσθέτοντας με τα flags ***--cap-drop=cap_sys_admin --cap-drop=net_bind_service***.

Μετά από τα capabilities θα ασχοληθούμε με τα cgroups. Θα περιορίσουμε το user space memory σε 500m και το CPU Share σε 512. Το CPU Share είναι μια αναλογία που ελέγχει τη χρήση της CPU του container. Έχει μια προεπιλεγμένη τιμή 1024 και κυμαίνεται μεταξύ 0 και 1024. Εάν τρία containers έχουν το ίδιο μερίδιο CPU 1024, κάθε container μπορεί να πάρει έως και 33% της CPU σε περίπτωση “σύγκρουσης” για τους πόρους της CPU. Μπορούμε να περιορίσουμε το kernel memory και το blkioweight, το οποίο ελέγχει τα Container’s IO, όμως, αυτές οι επιλογές θα καταργηθούν και θα αφαιρεθούν στις επόμενες εκδόσεις του docker [58]. Για να περιορίσουμε το CPU share και το user space memory κατά την ενεργοποίηση του container, θα πρέπει να προσθέσουμε τα flags ***-m 500M --cpu-shares 512***.

Αν σε ένα container ένας attacker καταφέρει να αποκτήσει πρόσβαση ως low privilege user, και σε περίπτωση που υπάρχει ένα binary suid αρχείο που δεν έχει γίνει configure σωστά, τότε ο attacker μπορεί να κάνει κατάχρηση αυτού του miss-configured αρχείου και να προχωρήσει σε privileges escalation επίθεση και στη συνέχεια να καταφέρει μια docker escape επίθεση. Για να αποτρέψουμε ένα τέτοιο escalation, θα προσθέσουμε την επιλογή no-new-privileges στην ενεργοποίηση του container, με το *flag* ***--security-opt=no-new-privileges:true***.

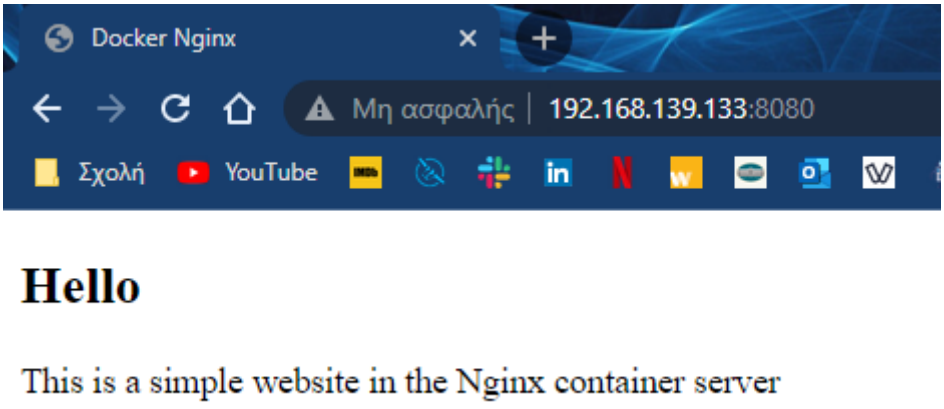
Πλέον έχουμε κατανοήσει τα best practices και μπορούμε να τα εφαρμόσουμε με την παρακάτω εντολή:

docker run -it --security-opt seccomp=default.json --security-opt apparmor=docker-default --cap-drop=cap_sys_admin --cap-drop=net_bind_service -m 500M --cpu-shares 512 --security-opt=no-new-privileges:true --rm -d -p 8080 --name thweb webserver

Με την παραπάνω εντολή τρέχουμε το container με όνομα thweb το οποίο βασίζεται στο image webserver και χρησιμοποιούμε διάφορα flags. Τα flags που έχουμε χρησιμοποιήσει είναι:

- -it: είναι συντομογραφία των flags -t (για να κάνουμε allocate ένα pseudo-tty) -i (για να διατηρήσουμε το STDIN ανοιχτό ακόμα και αν δεν είναι attached)
- --security-opt seccomp=default.json: προσθέτουμε το security option για το seccomp
- --security-opt apparmor=docker-default: προσθέτουμε το security option για το apparmor
- --cap-drop=cap_sys_admin: κάνουμε drop το cap_sys_admin που επιτρέπει πολλά πράγματα όπως BFP operations και system administration operations
- --cap-drop=net_bind_service: κάνουμε drop το net_bind για τα ports κάτω από το 1024
- -m 500M: περιορισμός του user space memory
- --cpu-shares 512: περιορισμός του CPU Share
- --security-opt=no-new-privileges:true: προσθέτουμε το security option για το no-new-privileges
- --rm: έτσι ώστε να διαγραφεί το container όταν γίνει exit
- -d: για να τρέχει το container στο background.
- -p 8080: ορίζουμε το port σε 8080
- --name thweb webserver: ορίζουμε το όνομα σε thweb και χρησιμοποιούμε το image webserver

Αν συνδεθούμε στη διεύθυνση του server με το port που έχουμε ορίσει, μπορούμε να δούμε ότι ο server λειτουργεί σωστά, όπως φαίνεται στην Εικόνα 28.



Εικόνα 28 Πρόσβαση στη σελίδα του server

Το apprmor κάνει συνεχόμενα monitor το σύστημα, οπότε μπορούμε με την εντολή ***journalctl -fx*** να δούμε τα logs όπως φαίνεται στην Εικόνα 29.

```
dimpat@ubuntu:~/thesis$ sudo journalctl -fx
-- Logs begin at Mon 2021-11-01 00:54:30 PDT. --
Nov 09 10:17:18 ubuntu NetworkManager[835]: <info>  [1636481838.4729] device (d
ocker0): carrier: link connected
Nov 09 10:17:18 ubuntu kernel: IPv6: ADDRCONF(NETDEV_CHANGE): veth75f68da: link
 becomes ready
Nov 09 10:17:18 ubuntu kernel: docker0: port 1(veth75f68da) entered blocking st
ate
Nov 09 10:17:18 ubuntu kernel: docker0: port 1(veth75f68da) entered forwarding
state
Nov 09 10:17:18 ubuntu gnome-shell[7672]: Removing a network device that was no
t added
Nov 09 10:17:20 ubuntu avahi-daemon[828]: Joining mDNS multicast group on inter
face veth75f68da.IPv6 with address fe80::cc6c:c8ff:fe7b:5cd1.
Nov 09 10:17:20 ubuntu avahi-daemon[828]: New relevant interface veth75f68da.IP
v6 for mDNS.
Nov 09 10:17:20 ubuntu avahi-daemon[828]: Registering new address record for fe
80::cc6c:c8ff:fe7b:5cd1 on veth75f68da.*.
Nov 09 10:17:24 ubuntu sudo[17383]:   dimpat : TTY=pts/1 ; PWD=/home/dimpat/the
sis ; USER=root ; COMMAND=/usr/bin/journalctl -fx
Nov 09 10:17:24 ubuntu sudo[17383]: pam_unix(sudo:session): session opened for
user root by (uid=0)
```

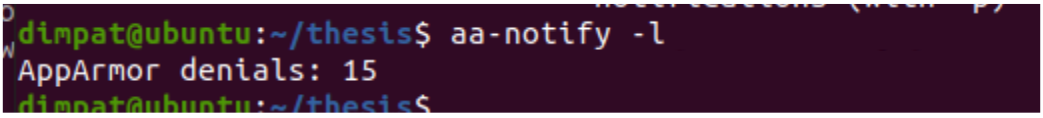
Εικόνα 29 Logs από το apparmor

Μέσα από τα logs μπορούμε να παρατηρήσουμε, στην Εικόνα 30, ότι το docker0 interface είναι disable το οποίο βρίσκεται και αυτό στη λίστα με τις βέλτιστες πρακτικές.

```
Nov 09 10:41:05 ubuntu containerd[944]: time="2021-11-09T10:41:05.575881152-08:00" level=err
cf/fd/12: file already closed"
Nov 09 10:41:05 ubuntu kernel: docker0: port 1(veth5fd8a3b) entered disabled state
Nov 09 10:41:05 ubuntu kernel: veth8345a40: renamed from eth0
Nov 09 10:41:05 ubuntu NetworkManager[835]: <info>  [1636483265.6284] manager: (veth8345a40)
anager/Devices/148)
Nov 09 10:41:05 ubuntu systemd-udevd[19101]: ethtool: autonegotiation is unset or enabled, t
```

Εικόνα 30 Το docker0 έχει απενεργοποιηθεί

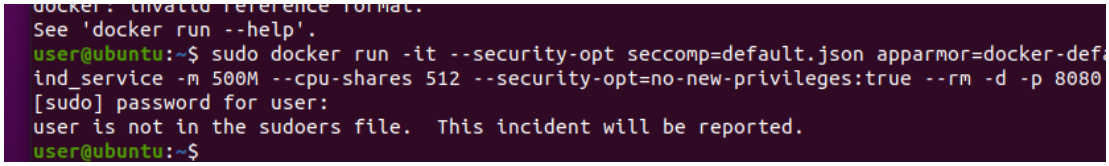
Μπορούμε επίσης να δούμε τα denials που έχουν προκύψει από το apparmor, με τη χρήση του notify όπως φαίνεται στην Εικόνα 31.



Εικόνα 31 Apparmor-Notify Denials

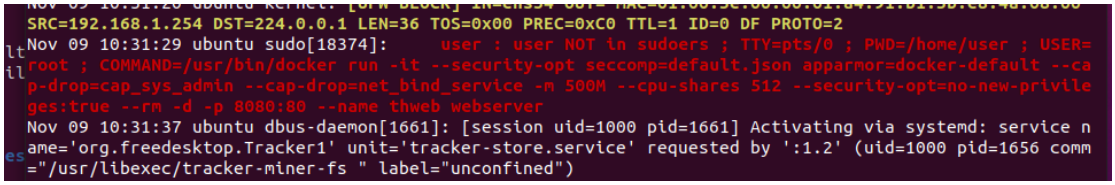
7.1.1 Apparmor Usecase

Όπως αναφέραμε στη προηγούμενη ενότητα το apparmor είναι ενεργοποιημένο και απλά “φορτώνουμε” το profile με το αντίστοιχο flag όταν πάμε να τρέξουμε το container. Σε περίπτωση που ο χρήσης προσπαθήσει να τρέξει το docker container ως root, το default-profile του apparmor θα τον σταματήσει και θα κάνει report το incident. Η προσπάθεια του χρήστη να τρέξει το docker container σαν root φαίνεται στην Εικόνα 32.



Εικόνα 32 Το apparmor μπλοκάρει το user

Μέσα από το journalctl που μας δείχνει τα logs του apparmor, μπορούμε να δούμε το report του incident όπως παρουσιάζεται στην Εικόνα 33.



Εικόνα 33 Report από το μπλοκάρισμα του user

7.2 Monitoring & Alerting

Από τη στιγμή που έχουμε κάνει import το OVA του Wazuh και το έχουμε ανοίξει, μπορούμε μέσα από το άλλο VM να μπούμε στη σελίδα για να μπορέσουμε να έχουμε πρόσβαση στο SIEM. Για να μπορέσει όμως το VM να δίνει τα απαραίτητα στοιχεία στο wazuh-manager θα πρέπει να εγκαταστήσουμε το wazuh-agent, να το ρυθμίσουμε με τις κατάλληλες IP

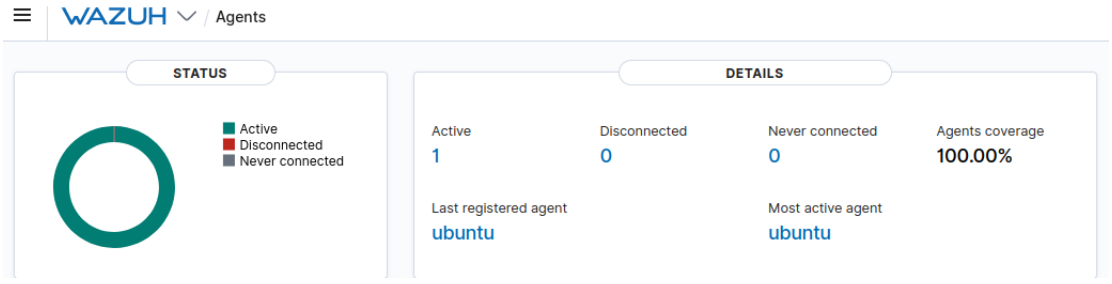
χρησιμοποιόντας την εντολή ***WAZUH_MANAGER="192.168.139.134" yum install wazuh-agent***. Η IP που βάλαμε είναι αυτή του Wazuh VM. Στη συνέχεια αφού ξεκινήσουμε κάνουμε reload το daemon, enable τον agent και τον ξεκινήσουμε (start), μπορούμε να δούμε τη κατάστασή του όπως φαίνεται στην Εικόνα 34.

```
root@ubuntu:/home/ssl-vm1# systemctl status wazuh-agent
● wazuh-agent.service - Wazuh agent
     Loaded: loaded (/lib/systemd/system/wazuh-agent.service; enabled; vendor p>
     Active: active (running) since Thu 2021-11-18 02:35:48 PST; 17s ago
    Process: 96047 ExecStart=/usr/bin/env /var/ossec/bin/wazuh-control start (c>
      Tasks: 33 (limit: 4619)
     Memory: 440.6M
     CGroup: /system.slice/wazuh-agent.service
             ├─95874 /var/ossec/bin/wazuh-execd
             ├─95882 /var/ossec/bin/wazuh-agentd
             ├─95896 /var/ossec/bin/wazuh-syscheckd
             ├─95909 /var/ossec/bin/wazuh-logcollector
             └─95925 /var/ossec/bin/wazuh-modulesd

Nov 18 02:35:45 ubuntu systemd[1]: Starting Wazuh agent...
Nov 18 02:35:46 ubuntu env[95852]: Completed.
Nov 18 02:35:46 ubuntu env[96047]: Starting Wazuh v4.2.5...
Nov 18 02:35:46 ubuntu env[96047]: wazuh-execd already running...
Nov 18 02:35:46 ubuntu env[96047]: wazuh-agentd already running...
Nov 18 02:35:46 ubuntu env[96047]: wazuh-syscheckd already running...
Nov 18 02:35:46 ubuntu env[96047]: wazuh-logcollector already running...
Nov 18 02:35:46 ubuntu env[96047]: wazuh-modulesd already running...
Nov 18 02:35:48 ubuntu env[96047]: Completed.
Nov 18 02:35:48 ubuntu systemd[1]: Started Wazuh agent.
```

Εικόνα 34 Κατάσταση του wazuh agent

Αφού ο agent στέλνει πλέον πληροφορίες στο manager, μπορούμε να κάνουμε reload τη σελίδα του wazuh και να δούμε ότι ο agent είναι Active και connected. Αυτή η σύνδεση παρουσιάζεται στην Εικόνα 35.



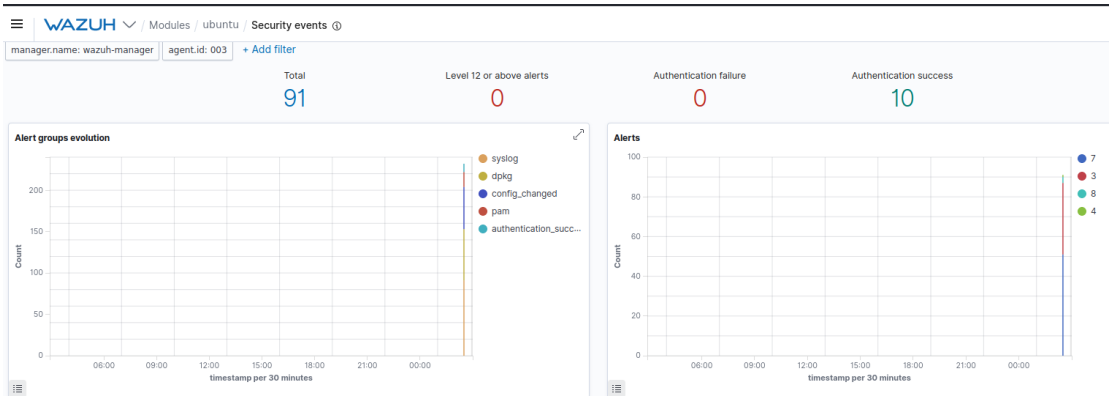
Εικόνα 35 Wazuh SIEM

Μέσα από το SIEM, μπορούμε να δούμε της πληροφορίες του ενεργού agent, το οποίο φαίνεται στην Εικόνα 36.

OS	Cluster node	Version	Registration date	Last keep alive	Status
Ubuntu 20.04.2 LTS	node01	v4.2.5	Nov 18, 2021 @ ...	Nov 18, 2021 @ ...	active

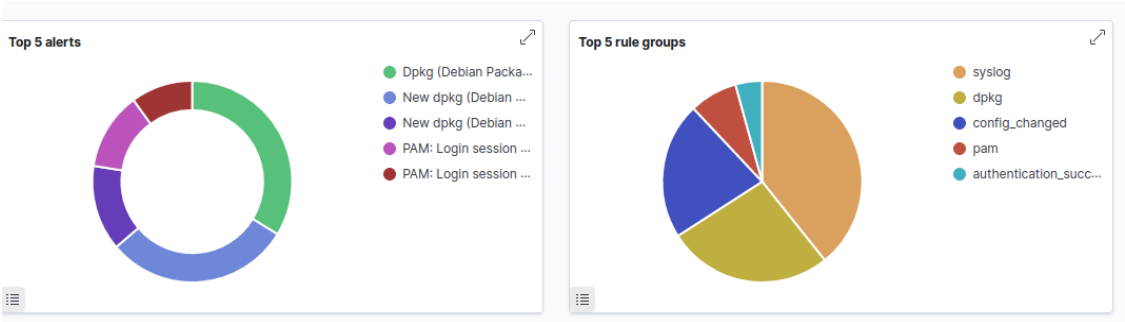
Εικόνα 36 Στοιχεία Agent

Μόλις επιλέξουμε το συγκεκριμένο agent παρουσιάζονται τα security events και βλέπουμε ότι δεν έχουμε κάποιο event με security threat Level 12 ή παραπάνω. Τα alerts είναι 91 και τα επιτυχημένα authentication 10. Μπορούμε να εξετάσουμε τα groups των alerts και να δούμε ότι τα περισσότερα προέρχονται από το dpkg. Αυτό φαίνεται στην Εικόνα 37.

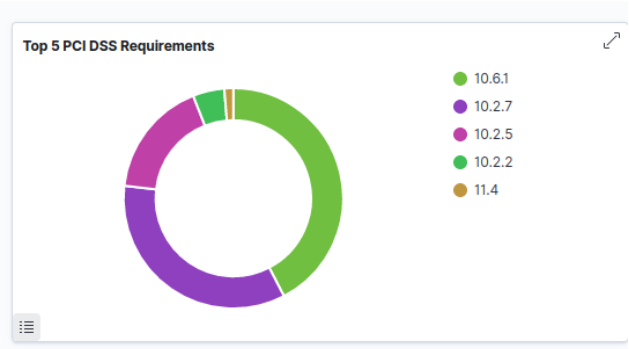


Εικόνα 37 Security Events

Στο ίδιο page (Security Events) μπορούμε να δούμε τα alerts σε διαφορετική μορφή (pie chart), τα rules groups, Εικόνα 38, καθώς και τα PCI DSS Requirements, Εικόνα 39.

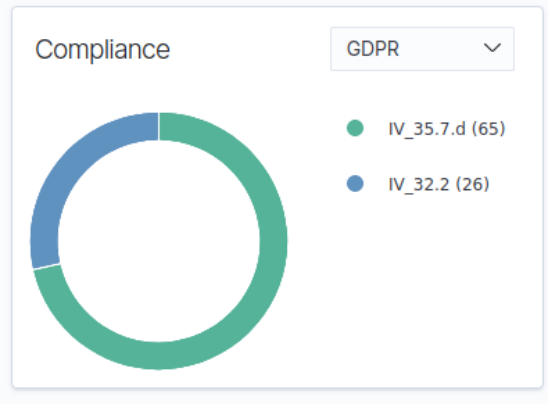


Εικόνα 38 Alerts & Rule Groups



Εικόνα 39 PCI DSS Requirements

Επιπλέον μπορούμε να ελέγξουμε τα GDPR Compliances, τα οποία παρουσιάζονται στην Εικόνα 40.



Εικόνα 40 GDPR Compliances σύμφωνα με το SIEM

Τέλος, αν κάνουμε scroll στο τέλος της σελίδας μας δίνεται η δυνατότητα να δούμε αναλυτικά τα security alerts, Εικόνα 41.

Security Alerts

Time ↓	Technique(s)	Tactic(s)	Description	Level	Rule ID
Nov 18, 2021 @ 02:59:46.330			Listened ports status (netstat) changed (new port opened or closed).	7	533
Nov 18, 2021 @ 02:54:12.006	T1040	Credential Access, Discovery	Interface entered in promiscuous(sniffing) mode.	8	5104
Nov 18, 2021 @ 02:54:11.963	T1040	Credential Access, Discovery	Interface entered in promiscuous(sniffing) mode.	8	5104
Nov 18, 2021 @ 02:54:11.952			PAM: Login session closed.	3	5502
Nov 18, 2021 @ 02:54:03.947	T1078	Defense Evasion, Initial Access, Persistence, Privilege Escalation	PAM: Login session opened.	3	5501
Nov 18, 2021 @ 02:53:53.979	T1078	Defense Evasion, Initial Access, Persistence, Privilege Escalation	PAM: Login session opened.	3	5501

Εικόνα 41 Αναλυτικά τα Security Alerts

Σε αυτό το σημείο μας δείχνει την ώρα του alert, την τεχνική και την τακτική, μια σύντομη περιγραφή, το Level Threat και το Rule ID. Επιλέγουμε το τελευταίο Alert το οποίο είναι Level 7 και μπορούμε να δούμε αναλυτικά το τι έχει συμβεί. Το alert απεικονίζεται στην Εικόνα 42.

agent.name	ubuntu
agent.id	003
manager.name	wazuh-manager
rule.firedtimes	1
rule.mail	false
rule.level	7
rule.pci_dss	10.2.7, 10.6.1
rule.hipaa	164.312.b
rule.tsc	CC6.8, CC7.2, CC7.3
rule.description	Listened ports status (netstat) changed (new port opened or closed).
rule.groups	ossec
rule.id	533
rule.nist_800_53	AU.14, AU.6
rule.gpg13	10.1
rule.gdpr	IV_35.7.d
decoder.name	ossec
full_log	ossec: output: 'netstat listening ports': tcp 0.0.0.0:22 0.0.0.0:* /usr tcp6 127.0.0.1:631 0.0.0.0:* 3197/cupsd tcp6 ::1:631 :::* 3197/cupsd udp 0. daemon tcp 0.0.0.0:8080 0.0.0.0:* 102707/docker-proxy tcp6 :::8080 ::
location	netstat listening ports

Εικόνα 42 Security Alert Level 7

Στην Εικόνα 43, παρουσιάζεται το full_log από το συγκεκριμένο Alert. Μπορεί κάποιος να το διαβάσει αν έχει τη συγκεκριμένη Διπλωματική σε μορφή PDF και κάνει ζουμ, γιατί η εικόνα είναι πολύ μεγάλη.

Εικόνα 43 Full_log Security Alert Level 7

7.3 Docker Container Security Audit & Compliance

Για το audit και compliance θα χρησιμοποιήσουμε δύο διαφορετικά tools, το anchore και το docker bench security. Και τα δύο εργαλεία θα τα τρέξουμε από το χρήστη με root privileges.

7.3.1 Anchore

Το anchore αποτελείται από το syft και το grype. Αρχικά το syft κάνει scan του image, το οποίο στη συγκεκριμένη περίπτωση είναι το custom image, webserver και μας δείχνει το όνομα, την έκδοση και τον τύπο του πακέτου. Τα αποτελέσματα του syft παρουσιάζονται στην Εικόνα 44.

```
dimpat@ubuntu:~$ syft packages webserver
 ✔ Loaded image
 ✔ Parsed image
 ✔ Cataloged packages      [136 packages]

NAME                        VERSION                 TYPE
adduser                     3.118                   deb
apt                         1.8.2.3                 deb
base-files                  10.3+deb10u11           deb
base-passwd                 3.5.46                  deb
bash                        5.0-4                   deb
bsdutils                    1:2.33.1-0.1            deb
ca-certificates             20200601~deb10u2        deb
coreutils                   8.30-3                  deb
curl                        7.64.0-4+deb10u2        deb
```

Εικόνα 44 Anchore - Syft

Από την άλλη, με το grype, μπορούμε να κάνουμε scan όλα τα layers του image που θέλουμε, Εικόνα 45.

```
dimpat@ubuntu:~$ grype webserver --scope all-layers
 ⠧ Vulnerability DB        [4.7 MB / 89 MB]
 ✔ Loaded image
 ✔ Parsed image
 ✔ Cataloged packages      [220 packages]
```

Εικόνα 45 Anchore - Grype Scan

Μόλις το scan ολοκληρωθεί το grype δείχνει όλες τις ευπάθειες που έχει το image, τα CVE των ευπαθειών καθώς και το severity των ευπαθειών, Εικόνα 46.

```
dimpat@ubuntu:~$ grype webserver --scope all-layers
 ✔ Vulnerability DB        [updated]
 ✔ Loaded image
 ✔ Parsed image
 ✔ Cataloged packages      [220 packages]
 ✔ Scanned image           [277 vulnerabilities]
NAME            INSTALLED              FIXED-IN        VULNERABILITY     SEVERITY
apt             1.8.2.3                                CVE-2011-3374     Negligible
bash            5.0-4                                  CVE-2019-18276    Negligible
bsdutils        1:2.33.1-0.1           (won't fix)     CVE-2021-37600    Low
coreutils       8.30-3                 (won't fix)     CVE-2016-2781     Low
coreutils       8.30-3                                 CVE-2017-18018    Negligible
curl            7.64.0-4+deb10u2       (won't fix)     CVE-2021-22924    Low
curl            7.64.0-4+deb10u2       (won't fix)     CVE-2021-22946    High
curl            7.64.0-4+deb10u2       (won't fix)     CVE-2021-22947    Medium
curl            7.64.0-4+deb10u2       (won't fix)     CVE-2021-22898    Low
curl            7.64.0-4+deb10u2                       CVE-2021-22922    Negligible
```

Εικόνα 46 Anchore - Grype Αποτελέσματα

7.3.2 Docker Bench Security

Με το docker bench security, θα μπορέσουμε να δούμε τα compliances με το CIS Docker Benchmark v1.3.1. Το μόνο που χρειάζεται να κάνουμε είναι να τρέξουμε το tool, και αυτό μόλις κάνει τα αυτοματοποιημένα τεστ που χρειάζεται, δείχνει τα αποτελέσματα στις 7 κατηγορίες του CIS Docker Benchmark (βλ. Κεφ. 4.4). Τα αποτελέσματα, φαίνονται στις Εικόνες 47-53.

```
[INFO] 1 - Host Configuration
[INFO] 1.1 - Linux Hosts Specific Configuration
[WARN] 1.1.1 - Ensure a separate partition for containers has been created (Automated)
[INFO] 1.1.2 - Ensure only trusted users are allowed to control Docker daemon (Automated)
[INFO]       * Users: dimpat,user
[WARN] 1.1.3 - Ensure auditing is configured for the Docker daemon (Automated)
[WARN] 1.1.4 - Ensure auditing is configured for Docker files and directories -/run/containerd (Automated)
[WARN] 1.1.5 - Ensure auditing is configured for Docker files and directories - /var/lib/docker (Automated)
[WARN] 1.1.6 - Ensure auditing is configured for Docker files and directories - /etc/docker (Automated)
[WARN] 1.1.7 - Ensure auditing is configured for Docker files and directories - docker.service (Automated)
[INFO] 1.1.8 - Ensure auditing is configured for Docker files and directories - containerd.sock (Automated)
[INFO]       * File not found
[WARN] 1.1.9 - Ensure auditing is configured for Docker files and directories - docker.socket (Automated)
[WARN] 1.1.10 - Ensure auditing is configured for Docker files and directories - /etc/default/docker (Automated)
[WARN] 1.1.11 - Ensure auditing is configured for Dockerfiles and directories - /etc/docker/daemon.json (Automated)
[WARN] 1.1.12 - 1.1.12 Ensure auditing is configured for Dockerfiles and directories - /etc/containerd/config.toml (Automated)
[INFO] 1.1.13 - Ensure auditing is configured for Docker files and directories - /etc/sysconfig/docker (Automated)
[INFO]       * File not found
[WARN] 1.1.14 - Ensure auditing is configured for Docker files and directories - /usr/bin/containerd (Automated)
[WARN] 1.1.15 - Ensure auditing is configured for Docker files and directories - /usr/bin/containerd-shim (Automated)
[WARN] 1.1.16 - Ensure auditing is configured for Docker files and directories - /usr/bin/containerd-shim-runc-v1 (Automated)
[WARN] 1.1.17 - Ensure auditing is configured for Docker files and directories - /usr/bin/containerd-shim-runc-v2 (Automated)
[WARN] 1.1.18 - Ensure auditing is configured for Docker files and directories - /usr/bin/runc (Automated)
[INFO] 1.2 - General Configuration
[NOTE] 1.2.1 - Ensure the container host has been Hardened (Manual)
[PASS] 1.2.2 - Ensure that the version of Docker is up to date (Manual)
[INFO]       * Using 20.10.10, verify is it up to date as deemed necessary
```

Εικόνα 47 Host Configuration

```
[INFO] 2 - Docker daemon configuration
[NOTE] 2.1 - Run the Docker daemon as a non-root user, if possible (Manual)
[WARN] 2.2 - Ensure network traffic is restricted between containers on the default bridge (Scored)
[PASS] 2.3 - Ensure the logging level is set to 'info' (Scored)
[PASS] 2.4 - Ensure Docker is allowed to make changes to iptables (Scored)
[PASS] 2.5 - Ensure insecure registries are not used (Scored)
[PASS] 2.6 - Ensure aufs storage driver is not used (Scored)
[INFO] 2.7 - Ensure TLS authentication for Docker daemon is configured (Scored)
[INFO]      * Docker daemon not listening on TCP
[INFO] 2.8 - Ensure the default ulimit is configured appropriately (Manual)
[INFO]      * Default ulimit doesn't appear to be set
[WARN] 2.9 - Enable user namespace support (Scored)
[PASS] 2.10 - Ensure the default cgroup usage has been confirmed (Scored)
[PASS] 2.11 - Ensure base device size is not changed until needed (Scored)
[WARN] 2.12 - Ensure that authorization for Docker client commands is enabled (Scored)
[WARN] 2.13 - Ensure centralized and remote logging is configured (Scored)
[WARN] 2.14 - Ensure containers are restricted from acquiring new privileges (Scored)
[WARN] 2.15 - Ensure live restore is enabled (Scored)
[WARN] 2.16 - Ensure Userland Proxy is Disabled (Scored)
[PASS] 2.17 - Ensure that a daemon-wide custom seccomp profile is applied if appropriate (Manual)
[INFO] Ensure that experimental features are not implemented in production (Scored) (Deprecated)
```

Εικόνα 48 Docker daemon configuration

```
[INFO] 3 - Docker daemon configuration files
[PASS] 3.1 - Ensure that the docker.service file ownership is set to root:root (Automated)
[PASS] 3.2 - Ensure that docker.service file permissions are appropriately set (Automated)
[PASS] 3.3 - Ensure that docker.socket file ownership is set to root:root (Automated)
[PASS] 3.4 - Ensure that docker.socket file permissions are set to 644 or more restrictive (Automated)
[PASS] 3.5 - Ensure that the /etc/docker directory ownership is set to root:root (Automated)
[PASS] 3.6 - Ensure that /etc/docker directory permissions are set to 755 or more restrictively (Automated)
[INFO] 3.7 - Ensure that registry certificate file ownership is set to root:root (Automated)
[INFO]      * Directory not found
[INFO] 3.8 - Ensure that registry certificate file permissions are set to 444 or more restrictively (Automated)
[INFO]      * Directory not found
[INFO] 3.9 - Ensure that TLS CA certificate file ownership is set to root:root (Automated)
[INFO]      * No TLS CA certificate found
[INFO] 3.10 - Ensure that TLS CA certificate file permissions are set to 444 or more restrictively (Automated)
[INFO]       * No TLS CA certificate found
[INFO] 3.11 - Ensure that Docker server certificate file ownership is set to root:root (Automated)
[INFO]       * No TLS Server certificate found
[INFO] 3.12 - Ensure that the Docker server certificate file permissions are set to 444 or more restrictively (Automate
[INFO]       * No TLS Server certificate found
[INFO] 3.13 - Ensure that the Docker server certificate key file ownership is set to root:root (Automated)
[INFO]       * No TLS Key found
[INFO] 3.14 - Ensure that the Docker server certificate key file permissions are set to 400 (Automated)
[INFO]       * No TLS Key found
[PASS] 3.15 - Ensure that the Docker socket file ownership is set to root:docker (Automated)
[PASS] 3.16 - Ensure that the Docker socket file permissions are set to 660 or more restrictively (Automated)
[PASS] 3.17 - Ensure that the daemon.json file ownership is set to root:root (Automated)
[PASS] 3.18 - Ensure that daemon.json file permissions are set to 644 or more restrictive (Automated)
[PASS] 3.19 - Ensure that the /etc/default/docker file ownership is set to root:root (Automated)
[INFO] 3.20 - Ensure that the /etc/sysconfig/docker file permissions are set to 644 or more restrictively (Automated)
[INFO]       * File not found
[INFO] 3.21 - Ensure that the /etc/sysconfig/docker file ownership is set to root:root (Automated)
[INFO]       * File not found
[PASS] 3.22 - Ensure that the /etc/default/docker file permissions are set to 644 or more restrictively (Automated)
[PASS] 3.23 - Ensure that the Containerd socket file ownership is set to root:root (Automated)
[PASS] 3.24 - Ensure that the Containerd socket file permissions are set to 660 or more restrictively (Automated)
```

Εικόνα 49 Docker daemon configuration files

```
[INFO] 4 - Container Images and Build File
[WARN] 4.1 - Ensure that a user for the container has been created (Automated)
[NOTE] 4.2 - Ensure that containers use only trusted base images (Manual)
[NOTE] 4.3 - Ensure that unnecessary packages are not installed in the container (Manual)
[NOTE] 4.4 - Ensure images are scanned and rebuilt to include security patches (Manual)
[WARN] 4.5 - Ensure Content trust for Docker is Enabled (Automated)
[WARN] 4.6 - Ensure that HEALTHCHECK instructions have been added to container images (Automated)
[WARN]      * No Healthcheck found: [webserver:latest]
[INFO] 4.7 - Ensure update instructions are not used alone in the Dockerfile (Manual)
[NOTE] 4.8 - Ensure setuid and setgid permissions are removed (Manual)
[INFO] 4.9 - Ensure that COPY is used instead of ADD in Dockerfiles (Manual)
[NOTE] 4.10 - Ensure secrets are not stored in Dockerfiles (Manual)
[NOTE] 4.11 - Ensure only verified packages are are installed (Manual)
```

Εικόνα 50 Container Images and Build File

```
[INFO] 5 - Container Runtime
[PASS] 5.1 - Ensure that, if applicable, an AppArmor Profile is enabled (Automated)
[PASS] 5.2 - Ensure that, if applicable, SELinux security options are set (Automated)
[WARN] 5.3 - Ensure that Linux kernel capabilities are restricted within containers (Automated)
[PASS] 5.4 - Ensure that privileged containers are not used (Automated)
[PASS] 5.5 - Ensure sensitive host system directories are not mounted on containers (Automated)
[PASS] 5.6 - Ensure sshd is not run within containers (Automated)
[PASS] 5.7 - Ensure privileged ports are not mapped within containers (Automated)
[WARN] 5.8 - Ensure that only needed ports are open on the container (Manual)
[PASS] 5.9 - Ensure that the host's network namespace is not shared (Automated)
[PASS] 5.10 - Ensure that the memory usage for containers is limited (Automated)
[PASS] 5.11 - Ensure that CPU priority is set appropriately on containers (Automated)
[WARN] 5.12 - Ensure that the container's root filesystem is mounted as read only (Automated)
[WARN] 5.13 - Ensure that incoming container traffic is bound to a specific host interface (Automated)
[WARN]       * Port being bound to wildcard IP: 0.0.0.0 in thweb
[WARN] 5.14 - Ensure that the 'on-failure' container restart policy is set to '5' (Automated)
[WARN]       * MaximumRetryCount is not set to 5: thweb
[PASS] 5.15 - Ensure that the host's process namespace is not shared (Automated)
[PASS] 5.16 - Ensure that the host's IPC namespace is not shared (Automated)
[PASS] 5.17 - Ensure that host devices are not directly exposed to containers (Manual)
[INFO] 5.18 - Ensure that the default ulimit is overwritten at runtime if needed (Manual)
[INFO]       * Container no default ulimit override: thweb
[PASS] 5.19 - Ensure mount propagation mode is not set to shared (Automated)
[PASS] 5.20 - Ensure that the host's UTS namespace is not shared (Automated)
[PASS] 5.21 - Ensurethe default seccomp profile is not Disabled (Automated)
[NOTE] 5.22 - Ensure that docker exec commands are not used with the privileged option (Automated)
[NOTE] 5.23 - Ensure that docker exec commands are not used with the user=root option (Manual)
[PASS] 5.24 - Ensure that cgroup usage is confirmed (Automated)
[PASS] 5.25 - Ensure that the container is restricted from acquiring additional privileges (Automated)
[WARN] 5.26 - Ensure that container health is checked at runtime (Automated)
[WARN]       * Health check not set: thweb
[INFO] 5.27 - Ensure that Docker commands always make use of the latest version of their image (Manual
[WARN] 5.28 - Ensure that the PIDs cgroup limit is used (Automated)
[WARN]       * PIDs limit not set: thweb
[INFO] 5.29 - Ensure that Docker's default bridge 'docker0' is not used (Manual)
[PASS] 5.30 - Ensure that the host's user namespaces are not shared (Automated)
[PASS] 5.31 - Ensure that the Docker socket is not mounted inside any containers (Automated)
```

Εικόνα 51 Container Runtime

```
[INFO] 6 - Docker Security Operations
[INFO] 6.1 - Ensure that image sprawl is avoided (Manual)
[INFO]      * There are currently: 10 images
[INFO]      * Only 0 out of 10 are in use
[INFO] 6.2 - Ensure that container sprawl is avoided (Manual)
[INFO]      * There are currently a total of 4 containers, with 1 of them currently running
```

Εικόνα 52 Docker Security Operations

```
[INFO] 7 - Docker Swarm Configuration
[PASS] 7.1 - Ensure swarm mode is not Enabled, if not needed (Automated)
[PASS] 7.2 - Ensure that the minimum number of manager nodes have been created in a swarm (Automated) (Swarm mode not enabled)
[PASS] 7.3 - Ensure that swarm services are bound to a specific host interface (Automated) (Swarm mode not enabled)
[PASS] 7.4 - Ensure that all Docker swarm overlay networks are encrypted (Automated)
[PASS] 7.5 - Ensure that Docker's secret management commands are used for managing secrets in a swarm cluster (Manual) (Swarm mode not enabled)
[PASS] 7.6 - Ensure that swarm manager is run in auto-lock mode (Automated) (Swarm mode not enabled)
[PASS] 7.7 - Ensure that the swarm manager auto-lock key is rotated periodically (Manual) (Swarm mode not enabled)
[PASS] 7.8 - Ensure that node certificates are rotated as appropriate (Manual) (Swarm mode not enabled)
[PASS] 7.9 - Ensure that CA certificates are rotated as appropriate (Manual) (Swarm mode not enabled)
[PASS] 7.10 - Ensure that management plane traffic is separated from data plane traffic (Manual) (Swarm mode not enabled)
```

Εικόνα 53 Docker Swarm Configuration

Τέλος το docker bench security βγάζει ένα τελικό σκορ, το οποίο απεικονίζεται στην Εικόνα 54.

```
Section C - Score

[INFO] Checks: 116
[INFO] Score: 11
```

Εικόνα 54 Docker bench security - Score

Κεφάλαιο 8. Συμπεράσματα

Το Docker ως μια από τις πιο γνωστές τεχνολογίες για την ανάπτυξη εφαρμογών παρέχει πολλά οφέλη στις ομάδες ανάπτυξης. Ο ηγετικός ρόλος του docker στον τομέα των containers, το κάνει στόχο για πολλούς. Για αυτό το λόγο, η ασφάλιση αυτής της τεχνολογίας είναι κάτι πολύ σημαντικό αλλά παράλληλα κάτι περίπλοκο.

Η ασφάλιση ενός container Docker δεν διαφέρει από την ασφάλιση άλλων container. Όμως, θα πρέπει να πραγματοποιηθεί μια ολοκληρωμένη προσέγγιση αν κάποιος θέλει πραγματικά να ασφαλίσει την υποδομή του, κάτι που είναι δύσκολο για πολλούς οργανισμούς. Ο κεντρικός υπολογιστής, το δίκτυο αλλά και οποιοσδήποτε άλλος ενδιάμεσος παράγοντας μπορεί να προκαλέσει κάποιο κενό στην ασφάλεια του συστήματος. Για την πλήρη ασφάλιση ενός docker container κάποιος θα πρέπει να ασφαλίσει όλα τα παραπάνω στοιχεία, χωρίς όμως να περιορίζει τη λειτουργικότητα του container.

Στη συγκεκριμένη διπλωματική, ασχοληθήκαμε με την εύρεση των βέλτιστων πρακτικών για τα περιβάλλοντα Docker αλλά και με τη πρακτική τους εφαρμογή. Η παρακολούθηση του συστήματος, η αξιολόγηση αλλά και η συμμόρφωση σύμφωνα με frameworks ήταν επίσης ένα μεγάλος μέρος της εργασίας.

Η χρήση images από αξιόπιστες πηγές, η αποφυγή χρήσης containers που έχουν προνόμια και η συνεχής ενημέρωση του συστήματος είναι κάποια από τα πιο βασικά βήματα για τη σωστή ανάπτυξη ενός docker περιβάλλοντος. Η συνεχής παρακολούθηση και η σάρωση είναι επίσης πράγματα που δε θα πρέπει να απουσιάζουν.

Κάθε αλυσίδα όμως, είναι τόσο δυνατή όσο είναι ο πιο αδύναμος της κρίκος και στον τομέα της ασφάλειας αυτός ο κρίκος είναι ο άνθρωπος. Η ασφάλεια είναι ένας τομέας που είναι αναπτύσσεται και μεταβάλλεται μέρα με τη μέρα. Κάτι που ισχύει σήμερα, μπορεί να θεωρείται απαρχαιωμένο αύριο. Για τους δύο παραπάνω λόγους το πιο σημαντικό από όλα, είναι η σωστή εκπαίδευση και ενημέρωση του προσωπικού.

8.1 Περαιτέρω Έρευνα

Ο σημαντικότερος τομέας που απαιτεί περαιτέρω έρευνα είναι η ασφάλεια των docker cluster, το swarm mode καθώς και των διάφορων orchestrators. Ιδιαίτερο ενδιαφέρον αποτελεί η δημιουργία custom profiles, των apparmor και seccomp, τα οποία θα κατασκευαστούν για να μπορούν να καλύπτουν όλες τις ανάγκες των docker containers καθώς και της υποδομής στην οποία βρίσκονται. Τέλος, η ανάπτυξη και ο έλεγχος της συμπεριφοράς ενός SIEM που συνεργάζεται με κάποιον orchestrator είναι κάτι που θα πρέπει να μελετηθεί.

Βιβλιογραφία

[1] Welsh, Matt, et al. *Running LINUX*. 3. ed, O'Reilly & Associates, Inc., 1999.

[2] Kiarie, James. "10 Linux Distributions and Their Targeted Users."*TecMint*, 25 Sept. 2020, www.tecmint.com/linux-distro-for-power-users.

[3] Mulgund, Devdatta. "A Brief History of Containerization: Why Container Security Best Practices Need to Evolve Now." Security Intelligence, 9 May 2019, securityintelligence.com/posts/a-brief-history-of-containerization-why-container-security-best-practices-need-to-evolve-now/.

[4] Johnson, Jonathan. "Containers & Containerization: A Beginner's Guide." BMC Blogs, 19 Apr. 2021, www.bmc.com/blogs/what-is-a-container-containerization-explained/.

[5] StackRox. "Docker Container Security 101: Risks and 33 Best Practices." *Stackrox.Io*, 19 Sept. 2019, www.stackrox.io/blog/docker- security-101.

[6] Mazin, Arnaud. "Docker Registry First Steps." *Octo Talks*, 11 Feb. 2014, https://blog.octo.com/en/docker-registry-first-steps/.

[7] "Docker Overview." *Docker Documentation*, https://docs.docker.com/get-started/overview/.

[8] "Docker Architecture." *Aqua*, www.aquasec.com/cloud-native-academy/docker-container/docker-architecture/.

[9] RedHat. "Kubernetes Adoption, Security, and Market Trends Report 2021." RedHat, 14 July 2021, www.redhat.com/en/resources/kubernetes-adoption-security-market-trends-2021-overview.

[10] Sengupta, Sudip. "Docker Security: 14 Best Practices for Securing Docker Containers." BMC Blogs, 19 Feb. 2021, www.bmc.com/blogs/docker-security-best-practices/.

[11] Mulgund, Devdatta. "8 Best Practices for Application Container Security." SecurityIntelligence, 22 June 2019, securityintelligence.com/posts/8-best-practices-for-application-container-security.

[12] Burillo, Mateo. "7 Docker Security Vulnerabilities and Threats." Sysdig, 25 Aug. 2017, https://sysdig.com/blog/7-docker-security-vulnerabilities/.

[13] González, David Fernández, et al. "SecDocker: Hardening the Continuous Integration Workflow." arXiv preprint arXiv:2104.07899 (2021).

[14] Fitzgerald, Brian, and Klaas-Jan Stol. "Continuous software engineering: A roadmap and agenda." Journal of Systems and Software 123 (2017): 176-189.

[15] Bass, Len, et al. "Securing a deployment pipeline. In 2015 IEEE/ACM 3rd International Workshop on Release Engineering." IEEE, may. 2015.

[16] “The RunC Vulnerability — A Deep Dive on Protecting Yourself | StackRox Community.” StackRox, 21 Feb. 2019, www.stackrox.io/blog/the-runc-vulnerability-a-deep-dive-on-protecting-yourself.

[17] Li, Ying. “Introducing Docker Secrets Management.” Docker Blog, 9 Feb. 2017, www.docker.com/blog/docker-secrets-management.

[18] “Docker Security.” Docker Documentation, https://docs.docker.com/engine/security/.

[19] Firch, Jason. “Red Team VS Blue Team: What’s The Difference?” Purplesec, 27 Sept. 2020, purplesec.us/red-team-vs-blue-team-cyber-security.

[20] Hargreaves, Amy, and James Chamberlain. “The Roles of Red, Blue and Purple Teams.” Itlab, www.itlab.com/blog/understanding-the-roles-of-red-blue-and-purple-security-teams.

[21] Williams, Amrit, and Mark Nicolett. "Improve it security with vulnerability management." Gartner ID G00127481 (2005).

[22] Webcast. “Security Event Management.” WayBackMachine, web.archive.org/web/20141019131638/http://securityinformationeventmanagement.com/security-event-management.php.

[23] Gordon, Scott. “Compliance Management and Compliance Automation – How and How Efficient, Part 1 | AccelOps.” WayBackMachine, 29 Apr. 2010, web.archive.org/web/20110723002943/http://www.accelops.net/blog/?p=149

[24] Verizon. “2018 Data Breach Digest.” Verizon, 2018, enterprise.verizon.com/resources/reports/2018-data-breach-digest.pdf.

[25] Kotenko, Igor, Olga Polubelova, and Igor Saenko. "The ontological approach for SIEM data repository implementation." 2012 IEEE International Conference on Green Computing and Communications. IEEE, 2012.

[26] Kotenko, Igor, and Andrey Chechulin. "Common framework for attack modeling and security evaluation in SIEM systems." 2012 IEEE International Conference on Green Computing and Communications. IEEE, 2012.

[27] Azodi, Amir, et al. "Pushing the limits in event normalisation to improve attack detection in IDS/SIEM systems." 2013 International Conference on Advanced Cloud and Big Data. IEEE, 2013.

[28] Kubecka, Chris. “28C3: Security Log Visualization with a Correlation Engine.” Fahrplan, 29 Dec. 2011, fahrplan.events.ccc.de/congress/2011/Fahrplan/events/4767.en.html.

[29] Foreman, P. (2009). Vulnerability Management (1st ed.). Auerbach Publications. https://doi.org/10.1201/9781439801512

[30] Bitchkei, Silvia. “Vulnerability Assessments vs. Vulnerability Management.” Systèmes de Sécurité Hitachi, 19 Feb. 2018, hitachi-systems-security.com/the-difference-between-vulnerability-assessments-and-vulnerability-management.

[31] Aqua Security. “Open Source Vulnerability Scanning: Methods and Top 5 Tools.” Aqua, 23 Sept. 2021, www.aquasec.com/cloud-native-academy/vulnerability-management/open-source-vulnerability-scanning.

[32] Scarfone, Karen, et al. "Technical guide to information security testing and assessment." NIST Special Publication 800.115 (2008): 2-25.

[33] Firch, Jason Mba. “Common Types Of Network Security Vulnerabilities In 2021.” PurpleSec, 23 Oct. 2020, purplesec.us/common-network-vulnerabilities.

[34] Ponemon. “Ponemon Study on Gaps in Vulnerability Response - ServiceNow.” Servicenow, www.servicenow.com/lpayr/ponemon-vulnerability-survey.html.

[35] Center for Internet Security. “Comments from the CenterforInternet Security in Response to the NISTRequest for Information on ‘Developing a FrameworkTo Improve CriticalInfrastructure Cybersecurity.’” NIST, www.nist.gov/system/files/documents/2017/06/01/040513_center_for_internet_security.pdf.

[36] Payne, Chris. “6 Essential Things to Know about CIS Benchmarks.” Advanced Cyber Solutions, 14 Aug. 2018, www.advancedcyber.co.uk/it-security-blog/six-essential-things-to-know-cis-benchmark.

[37] CIS (Center for Internet Security). “BenchmarksTM FAQ.” CIS, www.cisecurity.org/cis-benchmarks/cis-benchmarks-faq.

[38] Mazzoli, Robert. “Center for Internet Security (CIS) Benchmarks - Microsoft Compliance.” Microsoft Docs, 24 Aug. 2021, docs.microsoft.com/en-us/compliance/regulatory/offering-cis-benchmark.

[39] IBM Cloud Education. “CIS Benchmarks.” IBM, 17 July 2020, www.ibm.com/cloud/learn/cis-benchmarks.

[40] Ogden, Jacqueline von. “System Hardening with DISA STIGs and CIS Benchmarks.” Cimcor, 4 Jan. 2021, www.cimcor.com/blog/system-hardening-with-disa-stigs-and-cis-benchmarks.

[41] Trotter, Matt. “CIS Security Benchmarks and Compliance | What Is CIS Compliance?” Diligent Insights, 30 Dec. 2020, insights.diligent.com/compliance/what-is-cis-compliance.

[42] Smith, Beth. “Securing Docker with CIS Controls.” IT Security Guru, 17 June 2020, www.itsecurityguru.org/2020/05/22/securing-docker-with-cis-controls.

[43] Greig, Jonathan. “Security Concerns Hampering Adoption of Containers and Kubernetes.” TechRepublic, 19 Feb. 2020, www.techrepublic.com/article/security-concerns-hampering-adoption-of-containers-and-kubernetes.

[44] Christiansen, Robert. “More Enterprises Are Using Containers; Here’s Why.” CIO, 26 Aug. 2019, www.cio.com/article/3434010/more-enterprises-are-using-containers-here-s-why.html.

[45] Aqua Security. “Docker CIS Benchmark: Best Practices in Brief.” Aqua, 26 Sept. 2021, www.aquasec.com/cloud-native-academy/docker-container/docker-cis-benchmark.

[46] Center for Internet Security (CIS), “CIS Docker Benchmark – v1.3.1”, 18 May 2021, Online.

[47] NGINX, Inc. “What Is NGINX?” NGINX, www.nginx.com/resources/glossary/nginx.

[48] w3techs. “Historical Trends in the Usage Statistics of Web Servers, November 2021.” W3techs, w3techs.com/technologies/history_overview/web_server. Accessed Nov. 2021.

[49] “Seccomp(2) - Linux Manual Page.” Man7, man7.org/linux/man-pages/man2/seccomp.2.html.

[50] “Seccomp - HackTricks.” Hacktricks, book.hacktricks.xyz/linux-unix/privilege-escalation/docker-breakout/seccomp.

[51] “AppArmor - Ubuntu Wiki.” Ubuntu Wiki, wiki.ubuntu.com/AppArmor.

[52] “AppArmor - HackTricks.” Hacktricks, book.hacktricks.xyz/linux-unix/privilege-escalation/docker-breakout/apparmor.

[53] “Linux Capabilities - HackTricks.” Hacktricks, book.hacktricks.xyz/linux-unix/privilege-escalation/linux-capabilities.

[54] Red Hat. “Chapter 1. Introduction to Control Groups (Cgroups) Red Hat Enterprise Linux 6.” Red Hat Customer Portal, access.redhat.com/documentation/en-us/red_hat_enterprise_linux/6/html/resource_management_guide/ch01.

[55] Wazuh. “Welcome to Wazuh · Wazuh Documentation.” Wazuh, documentation.wazuh.com/current/index.html.

[56] Anchore. “Open Source Container Security Container Security | Vulnerability Scanning & SBOMs •.” Anchore, anchore.com/opensource.

[57] Docker. “GitHub - The Docker Bench for Security.” GitHub, github.com/docker/docker-bench-security.

[58] “Deprecated Engine Features.” Docker Documentation, https://docs.docker.com/engine/deprecated/.