ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΙΓΑΙΟΥ

Σκλήρυνση Μηχανής Δοχείων και Λειτουργικού Συστήματος σε Περιβάλλοντα Linux

του

Χωλίδη Κωνσταντίνου

Εκπόνηση διπλωματικής ως μέρος του
Προπτυχιακού Τίτλου Σπουδών

στην
Πολυτεχνική Σχολή
Τμήμα Μηχανικών Πληροφοριακών και Επικοινωνιακών Συστημάτων

Επιβλέπων Καθηγητής: Δρ. Κρητικός Κυριάκος
Αναπληρωτής Καθηγητής Τμήματος ΜΠΕΣ

Μέλη εξεταστικής επιτροπής:

Δρ. Σκούτας Δημήτριος
Επίκουρος Καθηγητής Τμήματος ΜΠΕΣ

Δρ. Σκιάνης Χαράλαμπος
Καθηγητής Τμήματος ΜΠΕΣ

Σάμος, Μάρτιος 2024

Δήλωση Συγγραφικής Ιδιότητας

Εγώ, ο Χωλίδης Κωνσταντίνος, δηλώνω ότι αυτή η διπλωματική εργασία με τίτλο "Σκλήρυνση Μηχανής Δοχείων και Λειτουργικού Συστήματος σε Περιβάλλοντα Linux" και η δουλειά που παρουσιάζεται σε αυτή είναι δικά μου. Επιβεβαιώνω ότι:

- Αυτή η δουλειά πραγματοποιήθηκε ολοκληρωτικά ή κυρίως κατά την υποψηφιότητά μου για τίτλο προπτυχιακών σπουδών σε αυτό το πανεπιστήμιο.
- Όπου οποιοδήποτε μέρος αυτής της διπλωματικής εργασίας έχει προηγουμένως κατατεθεί για την απόκτηση πτυχίου ή άλλου τίτλου σε αυτό ή άλλο πανεπιστήμιο, αυτό διατυπώνεται ξεκάθαρα.
- Όπου έχω συμβουλευτεί την δημοσιευμένη δουλειά τρίτων, αυτό αποδίδεται ορθώς.
- Όπου έχω παραθέσει δουλειά τρίτων, η πηγή δίνεται πάντα. Με εξαίρεση αυτές τις παραθέσεις, αυτή η διπλωματική εργασία είναι εξ ολοκλήρου προσωπική μου δουλειά.
- Έχω παραθέσει όλες τις κύριες πηγές βοήθειας.
- Όπου αυτή η διπλωματική εργασία είναι βασισμένη σε συνεργατική δουλειά δική μου και τρίτων, έχω καταστήσει ξεκάθαρο ποια κομμάτια έχουν πραγματοποιηθεί από άλλους και πώς συνέβαλα εγώ.

"*UNIX is basically a simple operating system, but you have to be a genius to understand the simplicity.*"

Dennis Ritchie (1941 - 2011)

ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΙΓΑΙΟΥ

Σύνοψη

Πολυτεχνική Σχολή
Τμήμα Μηχανικών Πληροφοριακών και Επικοινωνιακών Συστημάτων

Προπτυχιακός Τίτλος Σπουδών

Σκλήρυνση Μηχανής Δοχείων και Λειτουργικού Συστήματος σε Περιβάλλοντα Linux

του Χωλίδη Κωνσταντίνου

Τη σημερινή εποχή, όλο και περισσότερος κόσμος βασίζεται πλέον σε υπηρεσίες τύπου IaaS (Infrastructure as a Service) έναντι των παραδοσιακών επιτόπιων υποδομών για την παροχή υπολογιστικής υποστήριξης σε εφαρμογές, υπηρεσίες και επιχειρησιακές διαδικασίες. Αυτό συμβαίνει διότι κατ' αυτό τον τρόπο μειώνονται τα έξοδα ενός οργανισμού ή μιας επιχείρησης εφόσον δεν υπάρχει ανάγκη δαπάνης/επένδυσης για την αγορά εξοπλισμού και το λειτουργικό κόστος χρήσης υπηρεσιών IaaS βασίζεται σε ευέλικτα μοντέλα χρέωσης με βάση την χρήση (των πόρων υποδομής που προσφέρονται). Επιπλέον, είναι δυνατή η κλιμάκωση της προσφερόμενης απομακρυσμένης υποδομής ανάλογα με τις ανάγκες του οργανισμού και του τρέχοντος φόρτου εργασίας των υπηρεσιών και εφαρμογών προς υποστήριξη. Με αυτόν τον τρόπο, μεταβιβάζεται η ευθύνη της διάθεσης και συντήρησης του εξοπλισμού σε τρίτους ενώ ταυτόχρονα εισάγεται ένα καινούριο μοντέλο εμπιστοσύνης ανάμεσα στον χρήστη/οργανισμό και τον πάροχο νέφους. Η αύξηση ενδιαφέροντος από τις επιχειρήσεις προς τις τεχνολογίες εικονικοποίησης, οι οποίες αποτελούν τα θεμέλια των υπηρεσιών IaaS, αλλά και η ραγδαία άνοδος της δημοτικότητας των τεχνολογιών δοχείων (containers) όπως είναι το Docker, άρχισαν με την σειρά τους να ενισχύουν την υιοθέτηση της αρχιτεκτονικής μικρο-υπηρεσιών για την ανάπτυξη εφαρμογών. Μιας αρχιτεκτονικής που βασίζεται τόσο στις τεχνολογίες εικονικοποίησης για την στέγαση των εφαρμογών σε υποδομές νέφους όσο και στις τεχνολογίες δοχείων για την διαμέριση των λειτουργιών τους σε ένα σύνολο από συστατικά/δοχεία, προσφέροντας ένα κατάλληλο επίπεδο απόδοσης και κλιμακωσιμότητας [1]. Ωστόσο, οι εφαρμογές αυτές παραμένουν άμεσα ευεπηρέαστες σε ζητήματα ασφάλειας που μπορεί να αφορούν το ίδιο το νέφος ή/και τις τεχνολογίες στις οποίες στηρίζεται.

Στην παρούσα εργασία θα αναλύσουμε πρώτα τα ζητήματα ασφαλείας που αφορούν το νέφος και ειδικότερα αυτά που αφορούν τις τεχνολογίες εικονικοποίησης και δοχείων. Έπειτα, θα αναλύσουμε πως μπορεί να γίνει χρήση αυτών των δύο τεχνολογιών με περισσότερη ασφάλεια. Όμως, ο σκοπός της εργασίας προχωρά πέρα από αυτό και μεταβαίνει σε ένα πρακτικό επίπεδο, προτείνοντας τη λύση ενός εργαλείου που υλοποιεί την προτεινόμενη, ασφαλή χρήση των τεχνολογιών αυτών. Ειδικότερα, το εργαλείο αυτό όχι μόνο μπορεί να δημιουργεί εικονικές μηχανές κατά μήκος πολλών παρόχων νέφους αλλά και να τις σκληραίνει με έναν αυτοματοποιημένο τρόπο. Επιπροσθέτως, είναι ικανό να εγκαθιστά σε αυτές τις εικονικές μηχανές τη μηχανή δοχείων Docker, την οποία επίσης μπορεί να σκληραίνει. Ο βασικός στόχος της εργασίας είναι να διευκολύνει έναν οργανισμό στην εγκατάσταση και διαμόρφωση με αυτοματοποιημένο τρόπο ενός ασφαλούς, κατανεμημένου περιβάλλοντος (φιλοξενίας και λειτουργίας) για την εγκατάσταση και λειτουργία μιας εφαρμογής μικρο-υπηρεσιών. Η αυτοματοποίηση αυτή έγκειται στη σωστή διαμόρφωση του εργαλείου, η οποία δεν προϋποθέτει κάποια ειδική γνώση πάνω σε τεχνικά ζητήματα αλλά ούτε και στον τομέα της ασφάλειας υποδομών και των λειτουργικών συστημάτων.

Λέξεις κλειδιά: Νέφος, Ασφάλεια, Εικονικοποίηση, Εικονικές Μηχανές, Δοχεία, Μηχανή Δοχείων, Μικρο-υπηρεσίες, Αυτοματοποίηση, Σκλήρυνση

University of the Aegean

Abstract

School of Engineering
Department of Information and Communication Systems Engineering

Undergraduate Studies

Container Engine and Operating System Hardening in Linux Environments

of Cholidis Konstantinos

Today, more and more people rely on IaaS (Infrastructure as a Service) services over a traditional on-premise infrastructure to provide computational support to applications, services and business processes. This is due to the fact that the costs of an organization or business are reduced, since there is no need for an upfront investment on the purchase of equipment. Also, the operational cost of using IaaS services is based on flexible billing models according to the usage (of the offered infrastructure resources). In addition, it is possible to scale the offered remote infrastructure, depending on the needs of the organization and the current workload of the services and applications to be supported. In this way, the responsibility for the equipment and its maintenance is transferred to third parties, while at the same time a new trust model is introduced between the user/organization and the cloud provider. The increased interest shown by enterprises when in comes to virtualization technologies (which are a key foundation of IaaS services) in conjunction with the rapid rise in popularity of container technologies such as Docker, has in turn begun to drive the adoption of the microservices architecture for application development. An architecture based on virtualization technologies for hosting applications in cloud infrastructures and container technologies for partitioning their functions into a set of containers and thus, providing an appropriate level of performance and scalability [1]. However, such applications remain vulnerable to security issues that may be tied to the cloud and/or the technologies on which it is based.

In this thesis we will first analyze the security issues related to the cloud and in particular, those related to virtualization and container technologies. Then, we will analyze how these two technologies can be used in a more secure manner. However, the purpose of this thesis goes beyond that and moves to a more practical level by proposing the solution of a tool that can implement the proposed safe use of these technologies. In particular, this tool can not only create virtual machines across multiple cloud providers but also harden them in an automated manner. In addition, it is capable of installing the Docker container engine on these virtual machines, which it can also harden. The main goal of this work is to make it easier for an organization to install and configure in an automated manner a secure, distributed environment for the deployment and operation of a microservices application. This automation lies in the correct configuration of our tool, which does not require any special knowledge on technical or security issues concerning the infrastructure and operating systems.

Keywords: Cloud, Security, Virtualization, Virtual Machines, Containers, Container Engine, Micro-services, Automation, Hardening

Ευχαριστίες

Στο σημείο αυτό, θέλω να ευχαριστήσω όλους τους ανθρώπους που στάθηκαν δίπλα μου καθ' όλη την διάρκεια των σπουδών μου. Κυρίως, ευχαριστώ την οικογένειά μου που με στήριξε και συνεχίζει να με στηρίζει με κάθε τρόπο σε όλα τα επαγγελματικά και προσωπικά μου βήματα. Επίσης, ευχαριστώ βαθύτατα τους φίλους μου, που βρίσκονται πάντα δίπλα μου σε κάθε στάδιο της ζωής μου.

Τέλος, θέλω να ευχαριστήσω τον επιβλέποντα καθηγητή μου, κύριο Κυριάκο Κρητικό, που μου προσέφερε την ευκαιρία να ασχοληθώ με ένα θέμα που με ενδιέφερε κατόπιν συζήτησης μαζί του. Η συνεργασία μας ήταν πολύ εποικοδομητική και η εκπόνηση της παρούσας διπλωματικής εργασίας με βοήθησε να εμβαθύνω τις γνώσεις μου σε όλες τις τεχνολογίες που χρειάστηκαν για την υλοποίησή της.

Περιεχόμενα

Κατάλογος σχημάτων

Κατάλογος πινάκων

Κατάλογος Εντολών

Συντομογραφίες

AMI	**A**mazon **M**achine **I**mage
API	**A**pplication **P**rogramming **I**nterface
AppArmor	**App**lication **Armor**
ARP	**A**ddress **R**esolution **P**rotocol
ATT&CK	**A**dversarial **T**actics, **T**echniques & **C**ommon **K**nowledge
AWS	**A**mazon **W**eb **S**ervices
BIOS	**B**asic **I**nput/**O**utput **S**ystem
CaaS	**C**ontainer **a**s **a** **S**ervice
CAPEC	**C**ommon **A**ttack **P**attern **E**numeration and **C**lassification
CCE	**C**ommon **C**onfiguration **E**numeration
CERT	**C**omputer **E**mergency **R**esponse **T**eam
cgroups	**c**ontrol **groups**
CI/CD	**C**ontinuous **I**ntegration / **C**ontinuous **D**elivery
CIS	**C**enter for **I**nternet **S**ecurity
CISA	**C**ybersecurity & **I**nfrastructure **S**ecurity **A**gency
CPE	**C**ommon **P**latform **E**numeration
CPU	**C**entral **P**rocessing **U**nit
CVE	**C**ommon **V**ulnerabilities and **E**xposures
CVSS	**C**ommon **V**ulnerability **S**coring **S**ystem
CVRF	**C**ommon **V**ulnerability **R**eporting **F**ramework
CWE	**C**ommon **W**eakness **E**numeration
DDoS	**D**istributed **D**enial **o**f **S**ervice
DevOps	**Dev**elopment **Op**eration**s**
DNS	**D**omain **N**ame **S**ystem
DoS	**D**enial **o**f **S**ervice
EBS	**E**lastic **B**lock **S**tore
EC2	**E**lastic **C**ompute **C**loud

Gbps	**G**iga**b**its **p**er **s**econd
GCE	**G**oogle **C**ompute **E**ngine
GHz	**G**iga**H**ert**z**
GiB	**Gi**bi**B**yte
IaaS	**I**nfrastructure **a**s **a** **S**ervice
IAM	**I**dentity and **A**ccess **M**anagement
IaC	**I**nfrastructure **a**s **C**ode
IP	**I**nternet **P**rotocol
IPC	**I**nter **P**rocess **C**ommunication
IPv4	**I**nternet **P**rotocol **v**ersion **4**
IPv6	**I**nternet **P**rotocol **v**ersion **6**
KVM	**K**ernel-based **V**irtual **M**achine
LAN	**L**ocal **A**rea **N**etwork
LUNAR	**L**ockdown **UN**ix **A**uditing and **R**eporting
LXC	**L**inu**x** **C**ontainers
MAC	**M**andatory **A**ccess **C**ontrol
MAC	**M**edium **A**ccess **C**ontrol
MitM	**M**an-**i**n-**t**he-**M**iddle
NFV	**N**etwork **F**unction **V**irtualization
NIST	**N**ational **I**nstitute of **S**tandards and **T**echnology
NVD	**N**ational **V**ulnerability **D**atabase
OCI	**O**pen **C**ontainer **I**nitiative
OS	**O**perating **S**ystem
OVAL	**O**pen **V**ulnerability and **A**ssessment **L**anguage
OWASP	**O**pen **W**eb **A**pplication **S**ecurity **P**roject
PaaS	**P**latform **a**s **a** **S**ervice
PoC	**P**roof **o**f **C**oncept
RAM	**R**andom **A**ccess **M**emory
SaaS	**S**oftware **a**s **a** **S**ervice
SDK	**S**oftware **D**evelopment **K**it
SecComp	**Sec**ure **Comp**uting

SecDep	**Sec**ure **Dep**loyment
SELinux	**S**ecurity-**E**nhanced **Linux**
SSH	**S**ecure **SH**ell
TPM	**T**rusted **P**latform **M**odule
TUI	**T**ext-based **U**ser **I**nterface
UEFI	**U**nified **E**xtensible **F**irmware **I**nterface
VBS	**V**irtualization-**B**ased **S**ecurity
VM	**V**irtual **M**achine
VMBR	**V**irtual **M**achine **B**ased **R**ootkits
VMM	**V**irtual **M**achine **M**onitor
Vuls	**VUL**nerability **S**canner
XML	E**x**tensive **M**arkup **L**anguage
XSS	Cross (**X**) **S**ite **S**cripting
ΛΣ	**Λ**ειτουργικό **Σ**ύστημα
ΤΠ	**Τ**εχνολογίες **Π**ληροφοριών

Την παρούσα διπλωματική εργασία την αφιερώνω στην οικογένειά μου, τους φίλους μου, καθώς και σε όλους τους ανθρώπους που με στήριξαν κατά την διάρκεια των σπουδών μου.

Κεφάλαιο 1. Εισαγωγή

Παραδοσιακά, όταν ένας χρήστης/εταιρεία επιθυμούσε να διαθέτει μια υπηρεσία στο ευρύ κοινό θα έπρεπε να επενδύσει ένα κατάλληλο κεφάλαιο για την αγορά σχετικού εξοπλισμού. Σε περίπτωση που η υπηρεσία αυτή τύγχανε ευρείας αποδοχής, καθίστατο επιτακτική η ανάγκη της επέκτασης του υφιστάμενου εξοπλισμού αλλά και της εγκατάστασης όλων των αναγκαίων πόρων λογισμικού (βάσεων δεδομένων (Databases), λογισμικό της υπηρεσίας, αρχείων ρυθμίσεων, κ.λπ.) στα νέα κομμάτια εξοπλισμού που αγοράσθηκαν. Η παραπάνω διαδικασία απαιτούσε επιπλέον κεφάλαιο και αρκετές εργατοώρες προκειμένου να γίνει πράξη. Επιπρόσθετα, η λειτουργία του εξοπλισμού και η συντήρησή του είχε και αυτή ένα σημαντικό κόστος.

Στις αρχές του 2000 ο τρόπος διεξαγωγής της παραπάνω διαδικασίας άλλαξε ριζικά όταν η Amazon, ψάχνοντας τρόπους να κλιμακώσει τις υπηρεσίες που προσέφερε στον τομέα του ηλεκτρονικού εμπορίου (e-commerce), δημιούργησε την θυγατρική της, AWS (Amazon Web Services). Η AWS άρχισε να προσφέρει υπηρεσίες νεφο-υπολογιστικής (Cloud Computing Services) και συγκεκριμένα, την EC2 (Elastic Compute Cloud), μια υπηρεσία τύπου IaaS (Infrastructure as a Service) (Υποδομή ως Υπηρεσία). Πρόκειται για την πρώτη υπηρεσία ενοικίασης υπολογιστικών πόρων προς αξιοποίηση από οργανισμούς και επιχειρήσεις προκειμένου να επιταχυνθεί η διαδικασία διάθεσης των δικών τους υπηρεσιών. Αυτό το μοντέλο λειτουργίας παρέχει πολλά πλεονεκτήματα, όπως το μικρότερο κόστος εκκίνησης διάθεσης υπηρεσιών, η διευκόλυνση κλιμάκωσής τους (είτε οριζόντιας όπου μοιράζεται ο φόρτος εργασίας σε παραπάνω διακομιστές είτε κάθετης κατά την οποία ένας διακομιστής αναβαθμίζεται με ισχυρότερες προδιαγραφές [2]) και η απαλλαγή ευθύνης σχετικά με την συνεχή λειτουργία του εξοπλισμού στον οποίο στεγάζονται.

Λόγω της ευρείας αποδοχής των υπηρεσιών IaaS της AWS και των πολλών πλεονεκτημάτων που αυτές παρέχουν, πολλές άλλες εταιρείες, όπως η Google, IBM και Microsoft, άρχισαν να προσφέρουν και αυτές υπηρεσίες του ίδιου τύπου. Σήμερα, ο μέσος καταναλωτής έχει στην διάθεσή του μια πληθώρα εταιρειών που προσφέρουν υπηρεσίες νεφο-υπολογιστικής και μάλιστα μερικές από αυτές, όπως η Linode, Vultr και Digital Ocean, προσφέρουν ως την κύρια υπηρεσία τους τη δυνατότητα διάθεσης υπολογιστικών πόρων στους χρήστες με τη μορφή ενοικίασης εικονικών μηχανών (Virtual Machines - VMs).

1.1 Ασφάλεια Περιβαλλόντων Νέφους

Η ασφάλεια των περιβαλλόντων νέφους είναι ένα θέμα που απασχολεί πολύ τους χρήστες και ακόμα περισσότερο τις επιχειρήσεις που βασίζονται σε υπηρεσίες νεφο-υπολογιστικής για την διάθεση των δικών τους υπηρεσιών. Η επίτευξη κατάλληλου επιπέδου ασφάλειας εξαρτάται από τρεις βασικούς παράγοντες. Ο πρώτος είναι η ασφάλεια των υποδομών νέφους, όπου στην περίπτωση χρήσης υπηρεσιών IaaS υπεύθυνος είναι ο πάροχος νέφους. Ο δεύτερος παράγοντας είναι η ασφάλεια των τεχνολογιών εικονικοποίησης που χρησιμοποιούνται, όπου υπεύθυνος είναι εν μέρει ο πάροχος για τις τεχνολογίες που επέλεξε προκειμένου να διαθέσει τους απομακρυσμένους (εικονικούς) διακομιστές του και ο τελικός χρήστης εφόσον προβεί στην χρήση τεχνολογιών δοχείων (containers) (δηλ. ένα είδος τεχνολογίας εικονικοποίησης). Ο τρίτος και τελευταίος παράγοντας είναι οι ενέργειες στις οποίες οφείλει να προβεί ο χρήστης προκειμένου να διατηρήσει ή να ενισχύσει την ασφάλεια της υπηρεσίας του (που θα φιλοξενείται σε μια υποδομή νέφους) σύμφωνα με τις ανάγκες του.

Όταν επιλέξει ένας χρήστης να δημιουργήσει εικονικές μηχανές μέσω μιας από τις διαθέσιμες πλατφόρμες IaaS, προς διευκόλυνσή του η διαδικασία γίνεται και μέσω γραφικού περιβάλλοντος της μορφής Point and Click. Στις εικονικές μηχανές που προσφέρονται, τις περισσότερες φορές διατίθενται διάφορες διανομές λειτουργικού συστήματος Linux, οι οποίες ενδέχεται να έχουν εγκατεστημένα και ρυθμισμένα εκ των προτέρων διάφορα λογισμικά, όπως το σύστημα διαχείρισης βάσεων δεδομένων MySQL[1] και ο διακομιστής ιστού nginx[2]. Με αυτό τον τρόπο, η διαδικασία δημιουργίας εικονικής μηχανής είναι αρκετά εύκολη για τον χρήστη, ο οποίος δεν χρειάζεται να έχει ειδικές γνώσεις στο υλικό για να το πετύχει αυτό. Στην προκειμένη περίπτωση, ενώ υπάρχει μηδενική δυσκολία για την απόκτηση εικονικής μηχανής εξοπλισμένης με λειτουργικό σύστημα και πιθανότατα απαραίτητα προς χρήση προγράμματα, η ασφάλειά της δεν είναι πάντοτε εγγυημένη. Πολλά από τα προγράμματα που θα χρησιμοποιήσει ο χρήστης δεν παρέχουν εξ αρχής ενεργοποιημένες τις παραμέτρους ασφαλείας που μπορεί να υποστηρίζουν διότι δεν δύναται να υπάρχει μια ασφαλής ρύθμιση που να καλύπτει όλα τα πιθανά σενάρια χρήσης. Επιπλέον, η ρύθμιση των παραμέτρων ασφαλείας είναι μια διαδικασία που απαιτεί εξειδικευμένες γνώσεις και είναι αρκετά εύκολο όχι μόνο να παραλειφθεί αλλά και να γίνει λανθασμένα λόγω έλλειψης οικειότητας με τα εργαλεία που έχει στην διάθεση του ο χρήστης.

[1] Oracle. MySQL. URL: https://www.mysql.com/.
[2] NGINX. NGINX. URL: https://nginx.org/en/.

Όπως αναφέρεται και στο [5], υπάρχουν υπηρεσίες IaaS που προσφέρουν διανομές λειτουργικού συστήματος Linux με εγκατεστημένα και ρυθμισμένα εκ των προτέρων προγράμματα από τρίτους. Σε ορισμένες περιπτώσεις όπου υπάρχει τυφλή εμπιστοσύνη προς την ορθότητα των ρυθμίσεων αυτών σε συνδυασμό με έλλειψη γνώσης των εργαλείων, ενδέχεται λόγω ανθρώπινου λάθους ή στοχευμένα, να μένει ο τελικός χρήστης ευάλωτος σε ρίσκα ασφαλείας, όπως μη εξουσιοδοτημένη πρόσβαση, μόλυνση με κακόβουλο λογισμικό και υποκλοπές ευαίσθητων προσωπικών δεδομένων. Στην περίπτωση που και ο πάροχος νέφους έχει παραλείψει να εφαρμόσει τις απαραίτητες ενημερώσεις ασφαλείας στις τεχνολογίες εικονικοποίησης που χρησιμοποιεί, είναι πιθανό κάποιο από τα προγράμματα αυτά να μπορέσει να πραγματοποιήσει μια επίθεση hyperjacking [6] και να αποκτήσει έλεγχο του υπερ-επόπτη (hypervisor) με αποτέλεσμα να είναι σε θέση να προκαλέσει ζημιές είτε σε διαφορετικές εικονικές μηχανές που ελέγχονται από τον υπερ-επόπτη είτε στον (φυσικό) διακομιστή (host machine ή απλώς host) στον οποίο αυτός εκτελείται.

1.2 Εικονικοποίηση και τεχνολογίες υλοποίησής της

Η εικονικοποίηση αποτελεί μια τεχνολογία που επιτρέπει την διαμέριση πόρων ενός φυσικού μηχανήματος σε πολλά μικρότερα υποσύνολα αυτών. Με αυτόν τον τρόπο, καθίσταται ευκολότερη η διαχείρισή τους και αυξάνεται η απόδοση του υποκείμενου υλικού, αφού μπορεί να χρησιμοποιείται στο έπακρον. Υπάρχουν πολλά είδη εικονικοποίησης που αναλύονται πιο λεπτομερώς στο 2.2.1 αλλά οι δύο βασικότερες υλοποιήσεις της είναι η εικονικοποίηση διακομιστών και λειτουργικών συστημάτων (δοχειοποίηση).

Στην εικονικοποίηση διακομιστών (host virtualization), ένας φυσικός διακομιστής χωρίζει με την βοήθεια ενός υπερ-επόπτη τους πόρους του, όπως είναι η μνήμη, ο αποθηκευτικός χώρος και ο επεξεργαστής του, σε πολλές μικρότερες εικονικές μηχανές. Κάθε εικονική μηχανή μπορεί να εκτελεί διαφορετικό λειτουργικό σύστημα και να έχει διαφορετικές ρυθμίσεις από τις υπόλοιπες. Η διάθεση μιας εικονικής μηχανής δεν περιορίζεται μονάχα σε τοπικούς υπολογιστές και έτσι πολλές φορές συνηθίζεται η ενοικίαση τέτοιων μηχανών (σε περιβάλλοντα νέφους) προκειμένου να μπορέσουν να χρησιμοποιηθούν από τρίτα πρόσωπα για την επίτευξη των δικών τους διαδικασιών.

Κατά την δοχειοποίηση (containerization), μιας ειδικής περίπτωσης εικονικοποίησης, αντί για το υλικό, εικονικοποιείται το λειτουργικό σύστημα. Επομένως, τα δοχεία είναι ένα προϊόν της δοχειοποίησης και αποτελούν το αποτέλεσμα της εικονικοποίησης σε επίπεδο λειτουργικού συστήματος. Με την βοήθεια του λειτουργικού συστήματος, δημιουργούνται εικονικά

περιβάλλοντα στα οποία μπορούν αυτά να εκτελεστούν. Ουσιαστικά, μια μηχανή δοχείων αιτείται από το λειτουργικό σύστημα την διάθεση ενός εικονικού περιβάλλοντος για την εκτέλεση των διεργασιών ενός προγράμματος. Το πρόγραμμα αυτό, καθώς και όλες οι βιβλιοθήκες που χρειάζεται, καθορίζονται σε μια εικόνα δοχείου (container image), την οποία η μηχανή δοχείων θα πρέπει να ανακτήσει, να αποθηκεύσει και με βάση αυτήν να δημιουργήσει το αντιστοιχούμενο δοχείο που αυτή περιγράφει. Η πιο συνηθισμένη υλοποίηση της δοχειοποίησης είναι η δοχειοποίηση εφαρμογών μέσω της μηχανής δοχείων Docker. Αυτό συμβαίνει διότι μέσω της πλατφόρμας που αυτό παρέχει, η διαχείριση και χρήση δοχείων επιτυγχάνεται εύκολα από ομάδες ανάπτυξης λογισμικού κάθε επιπέδου.

Η δοχειοποίηση αποτελεί μια ελαφρύτερη εναλλακτική λύση σε σύγκριση με την εικονικοποίηση και είθισται να προτιμάται για την ανάπτυξη και διάθεση εφαρμογών. Αυτό οφείλεται στο γεγονός ότι η μεταφορά και αναδημιουργία των δοχείων μπορεί να πραγματοποιηθεί σε κάθε περιβάλλον και νέφος, καθώς και στο γεγονός ότι αυτά απαιτούν λιγότερους πόρους σε σχέση με τις εικονικές μηχανές, αφού οι υποκείμενοι πόροι του συστήματος μπορούν να μοιραστούν μεταξύ των διαφορετικών δοχείων. Αυτό, καθιστά την δοχειοποίηση πιο αποδοτική από την εικονικοποίηση διακομιστών, καθώς δεν χρειάζεται να εκτελεστεί ένας υπερ-επόπτης για την διαχείριση υπολογιστικών πόρων. Επιπλέον, η δοχειοποίηση επιτρέπει την εκτέλεση περισσότερων εικονικών περιβαλλόντων στον ίδιο φυσικό διακομιστή και έτσι, η απόδοση του υλικού αυξάνεται ακόμα περισσότερο. Συνήθως χρησιμοποιείται για την διάθεση εφαρμογών στο ευρύ κοινό, οι οποίες ακολουθούν την αρχιτεκτονική μικρο-υπηρεσιών (microservices). Δηλαδή, της διαμέρισης των λειτουργιών τους σε δοχεία. Λόγω του ότι τα παράγωγα της δοχειοποίησης περιέχουν όλες τις απαραίτητες βιβλιοθήκες για την εκτέλεσή τους, χωρίς να βασίζονται στο υποκείμενο νέφος στο οποίο στεγάζονται, η δοχειοποίηση αποτελεί αρκετά δημοφιλή επιλογή τεχνολογίας διάθεσης υπηρεσιών.

1.3 Ασφάλεια συστήματος κατά τη χρήση του Docker

Μία από τις πιο σημαντικές προκλήσεις των επιχειρήσεων κατά την διάθεση υπηρεσιών είναι η επίτευξη και διατήρηση της ασφάλειας. Παραδοσιακά, κατά την επιλογή χρήσης τεχνολογιών εικονικοποίησης, ανάμεσα στις επιλογές εικονικοποίησης υλικού και εικονικοποίησης ΛΣ είθισται να προτιμάται η δεύτερη. Μια λογική απόφαση εάν αναλογιστεί κανείς τα πλεονεκτήματα που προσφέρει τόσο στην απόδοση πόρων του συστήματος όσο και στην αποδοτική αλλά και εύκολη διαχείριση των υπηρεσιών όταν αυτές διατίθενται σε μορφή δοχείων. Παρ' όλα αυτά, η επιλογή αυτή είναι και λιγότερο ασφαλής στην περίπτωση που αφεθεί στις αρχικές ρυθμίσεις.

Για τον λόγο αυτό, εάν η μέγιστη δυνατή απόδοση δεν είναι μια από τις κύριες προτεραιότητες μιας επιχείρησης, προκειμένου να αυξηθεί η ασφάλεια του συστήματός της, διατηρώντας τα πλεονεκτήματα του Docker όσον αφορά την μεταφερσιμότητα και την απαλλαγή από τις εξαρτήσεις του εκάστοτε προγράμματος ακόμα και χωρίς την περαιτέρω διαμόρφωση των ρυθμίσεών του, η προτεινόμενη χρήση είναι η εκτέλεσή του μέσω μιας εικονικής μηχανής. Σε περιπτώσεις ιδιόκτητης υποδομής, η στρώση απομόνωσης μέσω της εικονικοποίησης υλικού ανάμεσα στα προγράμματα και το μηχάνημα στο οποίο εκτελούνται αποτελεί ένα παραπάνω εμπόδιο που θα πρέπει να ξεπεράσει ένας επιτιθέμενος για να προκαλέσει ζημιές στο κύριο σύστημα, αφού θα πρέπει πρώτα να περάσει από τον εικονικό πυρήνα (πυρήνα ΛΣ της εικονικής μηχανής) και έπειτα από τον υπερ-επόπτη. Παράλληλα, ο συνδυασμός με την αρχιτεκτονική δοχείων παρέχει ένα επιπρόσθετο επίπεδο απομόνωσης εφόσον στην προκειμένη περίπτωση η άμεση επικοινωνία με τον πυρήνα του συστήματος αφορά το φιλοξενούμενο ΛΣ και όχι το κύριο σύστημα. Συνεπώς, ένας επιτιθέμενος πρέπει να πραγματοποιήσει δύο αποδράσεις. Μια από την μηχανή ενορχήστρωσης δοχείων ή περιβάλλον δοχειοποίησης και έπειτα μια επιπλέον από το εικονικοποιημένο περιβάλλον.

Παρ' όλα αυτά, υπάρχουν περιπτώσεις όπου η απόδοση του συστήματος δεν δύναται να θυσιαστεί για χάρη της ασφάλειας. Σε αυτές τις περιπτώσεις, επιβάλλεται η τροποποίηση των ρυθμίσεων και του τρόπου λειτουργίας του Docker ώστε να μπορέσει να επιτευχθεί μια ικανοποιητική ισορροπία μεταξύ της ασφάλειας και της απόδοσης του συστήματος. Με βάση μια έρευνα που πραγματοποιήθηκε από την IBM σχετικά με την ασφάλεια των δοχείων [7], βρέθηκε πως ένα ορθώς ασφαλισμένο δοχείο μπορεί να παρέχει το ίδιο επίπεδο ασφάλειας με μια εικονική μηχανή ή ακόμα και μεγαλύτερο. Συγκεκριμένα, στην έρευνα αυτή αναφέρεται πως προκειμένου να ποσοτικοποιηθεί η ασφάλεια των δοχείων θα πρέπει να ληφθεί υπόψιν το ποσοστό των υποδομών που βρίσκεται υπό την ευθύνη του χρήστη και να θεωρηθεί δεδομένο πως οι πάροχοι νέφους θα ακολουθήσουν όλες τις ορθές πρακτικές ασφαλείας για να προστατεύσουν το κομμάτι των υποδομών που τους αντιστοιχεί. Σε αυτήν την περίπτωση, εάν ο χρήστης χρησιμοποιεί υπηρεσίες CaaS (Container as a Service) κατά τις οποίες ο πάροχος νέφους προσφέρει υποδομές προσαρμοσμένες συγκεκριμένα για την εκτέλεση δοχείων, τότε αυτός είναι υπεύθυνος για την ασφάλιση πολύ μικρότερου ποσοστού επιφάνειας επίθεσης συγκριτικά με τον πάροχο και επωφελείται άμεσα από τις προσπάθειες ασφάλισης του παρόχου για το ποσοστό του. Επομένως, συμπεραίνεται πως από την οπτική γωνία του τελικού χρήστη είναι πιο ασφαλές να χρησιμοποιήσει τεχνολογίες εικονικοποίησης ΛΣ μέσω ενός παρόχου νέφους για την παροχή των υπηρεσιών του [8].

1.3.1 Πλεονεκτήματα ασφαλείας με τη χρήση του Docker

Με τη χρήση της αρχιτεκτονικής δοχείων και ειδικότερα του Docker, έρχονται αρκετά εγγενή οφέλη ασφαλείας [9]. Ένα βασικό όφελος αποτελεί η διαφάνεια. Λόγω της φύσης τους, τα δοχεία επιτρέπουν την ακριβή κατανόηση του κώδικα που εκτελείται μέσα σε αυτά, σε αντίθεση με την περίπτωση χρήσης αποκλειστικά εικονικών μηχανών. Επιπρόσθετα, κατά την εμφάνιση προβλημάτων σε μία υπηρεσία με αρχιτεκτονική μικρο-υπηρεσιών που κάνει χρήση δοχείων, είναι διακριτή η διευκόλυνση στον εντοπισμό της πηγής τους.

Ένα εξίσου σημαντικό όφελος αποτελεί το επιπρόσθετο επίπεδο ασφάλειας που παρέχεται. Σε περιβάλλον εικονικής μηχανής θα πρέπει να σκληρυνθεί το ΛΣ φιλοξενίας (αν υπάρχει), ο υπερ-επόπτης, το φιλοξενούμενο ΛΣ και η εκτελούμενη υπηρεσία. Σε περιβάλλον δοχείου, πρέπει να σκληρυνθεί το ΛΣ που φιλοξενεί τη μηχανή δοχείων, η μηχανή δοχείων και η υπηρεσία. Πράγμα που σημαίνει πως σε ένα εικονικό περιβάλλον, με την χρήση του Docker έχουμε επιπρόσθετα επίπεδα που χρειάζονται σκλήρυνση. Συνεπώς, σε εικονικά περιβάλλοντα, η χρήση δοχείων έρχεται με την σκλήρυνση ενός επιπλέον συστατικού, της μηχανής δοχείων, οπότε αυτή προσφέρει καλύτερο επίπεδο ασφάλειας.

Δύο ακόμα σημαντικά πλεονεκτήματα είναι η ευκολία εφαρμογής ενημερώσεων ασφαλείας στα δοχεία και η σταθερότητα του περιβάλλοντος που προσφέρεται μέσω αυτών στις εφαρμογές. Ειδικότερα, η ενημέρωση ενός δοχείου είναι μια διαδικασία δύο μόνο εντολών, ενώ το προσφερόμενο περιβάλλον είναι σταθερό με την λογική πως η μηχανή δοχείων παρέχει ένα επίπεδο πάνω από το ΛΣ, και επομένως το περιβάλλον αυτό δεν είναι ξεχωριστά ευμετάβλητο από την εκάστοτε εφαρμογή που εκτελείται μέσα σε αυτό. Άρα, δεν κινδυνεύει από νέα προβλήματα ασφαλείας που θα μπορούσαν να επιφέρουν τυχόν ενημερώσεις των εξαρτήσεων της εφαρμογής χωρίς την ταυτόχρονη ενημέρωση της εφαρμογής της ίδιας.

1.3.2 Μειονεκτήματα ασφαλείας με τη χρήση του Docker

Παρ' όλα τα προαναφερόμενα πλεονεκτήματα του Docker, όπως η δυνατότητα απαλλαγής από εξαρτήσεις του εκάστοτε συστήματος και η ευελιξία διαχείρισης των δοχείων του, αυτό εμφανίζει αρκετές τρωτότητες σε σχέση με την ασφάλεια. Οι πιο αξιοσημείωτες από αυτές είναι η ανάγκη εκτέλεσης του Docker με διαχειριστικά δικαιώματα και η αρχικά ελαστικότερη απ' ό,τι χρειάζεται απομόνωσή του από τον πυρήνα του συστήματος. Ο άμεσος αντίκτυπος των

παραπάνω είναι πως τα δοχεία είναι πιο ευάλωτα από άλλες τεχνολογίες σε επιθέσεις τύπου άρνησης υπηρεσιών και σε επιθέσεις που εκμεταλλεύονται ευπάθειες του πυρήνα με σκοπό να ξεφύγουν από το δοχείο και να αποκτήσουν πρόσβαση στο κύριο σύστημα μέσω αυτού (Container Escape [10]).

Το γεγονός που αυξάνει τον κίνδυνο κατά την διάρκεια μιας επίθεσης τύπου άρνησης υπηρεσιών είναι πως σε αντίθεση με μια εικονική μηχανή, όπου επιλέγεται το εύρος πρόσβασης στους πόρους του συστήματος κατά τη δημιουργία της, η αρχική ρύθμιση του Docker επιτρέπει στα δοχεία να χρησιμοποιήσουν το ίδιο ποσοστό πόρων με το κύριο σύστημα, δηλαδή δεν υπάρχουν περιορισμοί. Επομένως, ανεξαρτήτως του αν ένα δοχείο βρεθεί υπό τον έλεγχο ενός επιτιθέμενου ή αυτός απλά κατευθύνει πολλά αιτήματα προς αυτό εξωτερικά, αυτό μπορεί δυνητικά να εμποδίσει την εκτέλεση άλλων δοχείων ή ακόμα και την εκτέλεση άλλων εφαρμογών που εκτελούνται στο σύστημα.

Σχετικά με τις επιθέσεις που αποσκοπούν στην απόκτηση πρόσβασης στο κύριο σύστημα μέσω ενός δοχείου, ο κίνδυνος οφείλεται στον τρόπο λειτουργίας των δοχείων σε συνδυασμό με τις αρχικές ρυθμίσεις ασφαλείας του Docker. Λόγω της απευθείας επικοινωνίας τους με τον πυρήνα του συστήματος, τα δοχεία είναι άμεσα ευεπηρέαστα από ευπάθειες του πυρήνα. Συνεπώς, ένας επιτιθέμενος που στοχεύει ένα δοχείο μπορεί να εκμεταλλευτεί μια ευπάθεια του πυρήνα προκειμένου να αποκτήσει πρόσβαση στο κύριο σύστημα - εφόσον η εκτέλεση του Docker γίνεται με διαχειριστικά δικαιώματα, μετά από μια επιτυχημένη επίθεση Container Escape θα έχει διαχειριστικά δικαιώματα και ο ίδιος [11].

1.4 Συνεισφορά Διπλωματικής

Οι πιο συνήθεις τρόποι αντιμετώπισης της ανεπαρκούς προκαθορισμένης ασφάλειας του Docker (δηλ. της ελαστικής απομόνωσης) είναι η ρύθμιση καλύτερων προφίλ AppArmor ή κανόνων SELinux προκειμένου να απομονωθεί περισσότερο από το κύριο σύστημα. Στην πραγματικότητα, αυτές οι πρακτικές λαμβάνουν υπόψιν τους κυρίως τα δοχεία και όχι τη μηχανή δοχείων καθ' αυτού. Γι' αυτό τον λόγο, πολλές φορές πρέπει να ακολουθούνται και καλύτερες πρακτικές κατά τη λειτουργία/χρήση του Docker, όπως η σκλήρυνσή του, η αποφυγή χρήσης του διαχειριστικού λογαριασμού (σε διεργασίες) όσον αφορά τα δοχεία και η αποφυγή λήψης δοχείων από μη έμπιστες πηγές. Υπάρχει επομένως ανάγκη δημιουργίας ενός εργαλείου που αυτοματοποιημένα μπορεί να δημιουργεί ασφαλή εικονικοποιημένα περιβάλλοντα, καθώς και να εγκαθιστά σε αυτά, με βάση τις προαναφερόμενες οδηγίες προστασίας του Docker και των δοχείων του, μια σκληρυμένη έκδοση του Docker.

Η ανάγκη αυτή ικανοποιήθηκε με την ανάπτυξη του εργαλείου που αναπτύχθηκε στα πλαίσια της παρούσας διπλωματικής εργασίας, ονόματι SecDep (Secure Deployment), το οποίο θα περιγράψουμε παρακάτω στο Κεφάλαιο 5. Με την χρήση του, θα επιτευχθεί η ασφάλιση του Docker και του συστήματος με αυτοματοποιημένο τρόπο ακολουθώντας ορθές πρακτικές, χρησιμοποιώντας ένα ασφαλέστερο από το αρχικό Container Runtime και εκτελώντας το Docker εξ ολοκλήρου χωρίς την ανάγκη διαχειριστικών δικαιωμάτων. Το εργαλείο αυτό εστιάζει στην χρήση εικονικοποιημένων περιβαλλόντων, μιας και παρέχουν περισσότερα επίπεδα προς διείσδυση για έναν επιτιθέμενο, εμποδίζοντάς τον στην επίτευξη του στόχου αυτού. Επικοινωνεί με πολλούς παρόχους νέφους στέλνοντάς τους παραμέτρους για τις προδιαγραφές εικονικών μηχανών προκειμένου να μπορέσουν αυτές να δημιουργηθούν αυτόματα. Με αυτόν τον τρόπο, επιτρέπει την δημιουργία πολλαπλών ασφαλών περιβαλλόντων ώστε να μπορεί ένας χρήστης να εγκαθιστά εκεί τα δοχεία της εφαρμογής του. Με την εφαρμογή των κατάλληλων μέτρων και πρακτικών ασφαλείας σε κάθε επίπεδο, τα περιβάλλοντα αυτά σκληρύνονται, μικραίνοντας έτσι την πιθανότητα διείσδυσης. Η σκλήρυνση του ΛΣ επιτυγχάνεται με την εκτέλεση πολλών βημάτων στα οποία μεταξύ άλλων περιλαμβάνεται η σκλήρυνση του SSH, ο εντοπισμός, η εγκατάσταση, η ρύθμιση και η ενεργοποίηση των κατάλληλων για την εκάστοτε διανομή, προγραμμάτων για ανάχωμα ασφαλείας και για παροχή MAC (Mandatory Access Control) καθώς και η δημιουργία και ενεργοποίηση περιοδικά εκτελέσιμων προγραμμάτων για την ενημέρωση του συστήματος και το άνοιγμα/κλείσιμο θυρών προγραμμάτων. Η απεξάρτηση από τον διαχειριστικό λογαριασμό επιτυγχάνεται χάρη στη δουλειά του Akihiro Suda πάνω στο rootlesskit[3], επιτρέποντας έτσι την χρήση ενός πλαστού διαχειριστικού λογαριασμού προκειμένου να μπορέσει η μηχανή δοχείων να εκτελεστεί υπό την κυριότητα οποιουδήποτε χρήστη του συστήματος. Τέλος, η αντικατάσταση του αρχικού Container Runtime με το αντίστοιχο του gVisor[4], εισάγει ένα πιο συμπαγές τείχος προστασίας απέναντι σε συνηθισμένες επιθέσεις κατά των δοχείων, καθιστώντας το σύστημα μας πιο ασφαλές συγκριτικά με τις αρχικές/προκαθορισμένες ρυθμίσεις.

Η χρήση του εργαλείου έρχεται με πολλά πλεονεκτήματα για τον χρήστη. Κατ' αρχήν, το εργαλείο είναι εύκολο στη χρήση και στην εγκατάσταση, καθώς παρέχει πλούσια τεκμηρίωση και οδηγίες εγκατάστασης στο αποθετήριό του, ενώ οι γνώσεις που απαιτούνται στον τομέα των τεχνολογιών νέφους για την χρήση του είναι ελάχιστες. Επιπλέον, είναι ευέλικτο και επεκτάσιμο διότι αποτελείται από δύο εκτελέσιμα προγράμματα, τα οποία μπορεί κάθε χρήστης

[3]A. Suda. rootlesskit. 2020. URL: `https://github.com/rootless-containers/rootlesskit`.
[4]Google. The Container Security Platform. URL: `https://gvisor.dev/`.

να τροποποιήσει ώστε να προσθέσει τις δικές του λειτουργίες αν το επιθυμεί. Τέλος, οι παράμετροι του είναι εύκολα κατανοητές και προσαρμόσιμες στις ανάγκες του χρήστη. Επομένως, το εργαλείο αυτό αποτελεί ένα κατάλληλο μέσο για την δημιουργία, διαχείριση και ασφάλιση περιβαλλόντων δοχειοποιημένων εφαρμογών (containerized applications), δηλ. εφαρμογών που παίρνουν την μορφή δοχείων.

1.5 Δομή Εργασίας

Η υπόλοιπη δομή της αναφοράς είναι η εξής. Στο Κεφάλαιο 2, θα αναλύσουμε τις έννοιες της νεφο-υπολογιστικής και εικονικοποίησης, τις διάφορες τεχνολογίες εικονικοποίησης και θα εμβαθύνουμε στην τεχνολογία των δοχείων με επίκεντρο την ασφάλεια του Docker. Στο επόμενο κεφάλαιο (δηλαδή το Κεφάλαιο 3), θα παρουσιάσουμε εργασίες σχετικές με την παρούσα και θα πραγματοποιηθεί ανάλυση και σύγκριση αυτών με την προτεινόμενη εργασία της διπλωματικής. Αμέσως μετά, στο Κεφάλαιο 4, αναφερόμαστε στην διαδικασία ανάπτυξης του προτεινόμενου εργαλείου και τα παράγωγά της (απαιτήσεις, σχεδιαστικά μοντέλα, κώδικας υλοποίησης κα.). Προχωρώντας στο Κεφάλαιο 5, θα αναφερθούμε στις οδηγίες εγκατάστασης και λειτουργίας του εργαλείου με βάση στοχευμένα σενάρια χρήσης. Έπειτα, στο Κεφάλαιο 6 αποτιμάται η αποδοτικότητα του εργαλείου κατά την πραγματική χρήση του, προκειμένου να εξετασθεί με πρακτικό τρόπο η ικανότητα εξασφάλισης των στόχων του. Τέλος, στο Κεφάλαιο 7, αναφέρονται δυνητικές επεκτάσεις και βελτιώσεις του προτεινόμενου εργαλείου προκειμένου αυτό να μπορέσει να πάρει μια θέση στην αγορά.

Κεφάλαιο 2. Υπόβαθρο

Προκειμένου να κατανοήσουμε πλήρως το πρόβλημα που επιλύει η παρούσα εργασία πρέπει να αναλύσουμε μερικές βασικές έννοιες. Αρχικά, θα δούμε τι είναι το υπολογιστικό νέφος, τι μας προσφέρει, ποια είναι τα μοντέλα ανάπτυξής του και ποια μοντέλα παράδοσης παρέχονται μέσω αυτού. Έπειτα, θα συζητήσουμε την έννοια της εικονικοποίησης και τις διάφορες υποκατηγορίες της που χρησιμοποιούνται στις μέρες μας. Αμέσως μετά, θα αναλυθούν τα πλεονεκτήματα της εικονικοποίησης και θα αναδειχθεί ο λόγος που προτιμάται η εικονικοποίηση του λειτουργικού συστήματος (δηλ. η χρήση δοχειοποιημένων περιβαλλόντων). Τέλος, θα εμβαθύνουμε στην τεχνολογία δοχείων, θα δούμε διάφορες υλοποιήσεις της και θα πραγματοποιήσουμε μια ανάλυση της ασφάλειας του Docker, ως το de facto βιομηχανικό πρότυπο για την διαχείριση των δοχείων και των εικόνων τους στα πλαίσια δοχειοποιημένων εφαρμογών.

2.1 Νεφο-υπολογιστική

Με τον όρο νεφο-υπολογιστική (Cloud Computing), αναφερόμαστε σε ένα μοντέλο παράδοσης υπολογιστικών πόρων κατά παραγγελία από μια επιχείρηση προς τους καταναλωτές της. Οι υπηρεσίες που προσφέρει ένα υπολογιστικό νέφος χωρίζονται σε τρεις κατηγορίες - αυτές οι κατηγορίες αναφέρονται και ως μοντέλα παράδοσης του νέφους. Η πρώτη, SaaS (Software as a Service) (Λογισμικό ως Υπηρεσία), αναφέρεται στην απομακρυσμένη διάθεση λογισμικού, του οποίου η συμμόρφωση με τις λειτουργικές και μη λειτουργικές του ικανότητες που διαφημίζονται προς τους πελάτες αποτελεί ευθύνη του παρόχου του. Η κατηγορία PaaS (Platform as a Service) (Πλατφόρμα ως Υπηρεσία) ορίζεται ως η διάθεση απομακρυσμένης πλατφόρμας με την οποία μια ομάδα έργου μπορεί να αναπτύξει συνεργατικά και να εκτελέσει λογισμικό. Τέλος, η κατηγορία IaaS (Infrastructure as a Service) μεταφράζεται ως η προσφορά απομακρυσμένων (εικονικών και φυσικών) διακομιστών τους οποίους μια επιχείρηση μπορεί να αξιοποιήσει αναλόγως τις ανάγκες της (π.χ. ως προς την φιλοξενία κατάλληλων φόρτων εργασίας) ακολουθώντας φυσικά τους όρους και προϋποθέσεις του παρόχου. Τα πλεονεκτήματα που παρέχει η νεφο-υπολογιστική σε σχέση με την παραδοσιακή μέθοδο διάθεσης υπηρεσιών είναι αρκετά αλλά αυτά που ξεχωρίζουν από μεριάς των πελατών είναι η απόλυτη απαλλαγή ευθύνης των υποδομών νέφους, η απαράμιλλη ταχύτητα διάθεσης και κλιμάκωσης των υπηρεσιών και η εξάλειψη περιττού κόστους λόγω του ευέλικτου μοντέλου χρέωσης όπου προσμετρώνται μόνο οι πόροι που χρησιμοποιήθηκαν.

Σημαντικό ρόλο στην ευρεία αποδοχή των υπηρεσιών που προσφέρονται μέσω της νεφο-υπολογιστικής έχει η ευκολία αλλά και ευελιξία των μεθόδων διάθεσης και μετέπειτα διαχείρισής τους. Σε κάθε περίπτωση γίνεται χρήση ενός API (Application Programming Interface) (Προγραμματιστική Διεπαφή Εφαρμογής), το οποίο είναι προσπελάσιμο έμμεσα μέσω ενός γραφικού περιβάλλοντος (self-service portal) ή ενός εργαλείου γραμμής εντολών (command line tool) ή άμεσα με προγραμματιστικό τρόπο (π.χ. με τη χρήση SDKs (Software Development Kits) (Κιτ Ανάπτυξης Λογισμικού)).

2.1.1 Ορισμός Νεφο-Υπολογιστικής

Σύμφωνα με το "The NIST Definition of Cloud Computing" [14], η νεφο-υπολογιστική είναι ένα μοντέλο που επιτρέπει την ανά πάσα στιγμή διαδικτυακή πρόσβαση από οπουδήποτε σε μια κοινή δεξαμενή ρυθμιζόμενων υπολογιστικών πόρων που μπορούν να παρέχονται και να απελευθερώνονται γρήγορα και με ελάχιστη προσπάθεια διαχείρισης ή αλληλεπίδρασης με τον πάροχο υπηρεσιών. Στους υπολογιστικούς αυτούς πόρους περιλαμβάνονται δίκτυα, διακομιστές, χώρος αποθήκευσης, εφαρμογές και υπηρεσίες. Αυτό το μοντέλο νέφους αποτελείται από πέντε βασικά χαρακτηριστικά, τρία μοντέλα υπηρεσιών (ή μοντέλα παράδοσης) και τέσσερα μοντέλα ανάπτυξης.

2.1.2 Χαρακτηριστικά

Τα πέντε βασικά χαρακτηριστικά της νεφο-υπολογιστικής με βάση τους Mell και Grance [14] είναι τα εξής:

- **Αυτοεξυπηρέτηση κατά παραγγελία (On-demand Self-service)**:

  Ένας καταναλωτής μπορεί να προμηθευτεί υπολογιστικούς πόρους, όπως αποθηκευτικό χώρο και ισχυρούς επεξεργαστές, κατά απαίτηση δίχως την ανάγκη αλληλεπίδρασης με τον αντίστοιχο πάροχο νέφους. Μόλις συγκροτηθούν, η χρήση των πόρων αυτών μπορεί να αυτοματοποιηθεί, οδηγώντας σε ένα περιβάλλον αυτοεξυπηρέτησης κατ' απαίτηση.

- **Πανταχού παρούσα πρόσβαση (Ubiquitous Access)**:

  Οι υπηρεσίες που παρέχονται είναι διαθέσιμες μέσω του διαδικτύου και είναι προσβάσιμες μέσω πρότυπων μηχανισμών από ετερογενείς πλατφόρμες λεπτών και πυκνών πελατών (thin & thick clients).

- **Πολλαπλή Μίσθωση (Multi-Tenancy)**:

  Οι υπολογιστικοί πόροι του παρόχου είναι σχεδιασμένοι με τέτοιο τρόπο ώστε να μπορούν να εξυπηρετήσουν πολλούς καταναλωτές έκαστος, χρησιμοποιώντας ένα μοντέλο πολλαπλής μίσθωσης (multi-tenant model) με διαφορετικούς φυσικούς και εικονικούς πόρους που εκχωρούνται και ανακατανέμονται ανάλογα με τη ζήτηση των καταναλωτών. Το μοντέλο αυτό διακατέχεται και από μια αίσθηση ανεξαρτησίας θέσης διότι ανεξάρτητα από το που βρίσκεται ένας τελικός χρήστης μπορεί να χρησιμοποιήσει πόρους από οποιοδήποτε κέντρο δεδομένων επιθυμεί. Παραδείγματα πόρων που παρέχονται αποτελούν μεταξύ άλλων το εύρος ζώνης δικτύου (network bandwidth), ο αποθηκευτικός χώρος και οι εικονικές μηχανές.

- **Ελαστικότητα (Elasticity)**:

  Οι υπολογιστικές, δικτυακές και αποθηκευτικές δυνατότητες μπορούν να παρέχονται και να απελευθερώνονται ελαστικά (αυτόματα) με σκοπό την οριζόντια ή κάθετη κλιμάκωση υπηρεσιών αναλόγως την παρούσα ζήτηση. Από την οπτική γωνία του καταναλωτή, οι παρεχόμενες δυνατότητες μοιάζουν απεριόριστες και μπορούν να κατανεμηθούν ανά πάσα στιγμή σε οποιαδήποτε ποσότητα.

- **Μετρούμενη υπηρεσία (Measured Service)**:

  Τα συστήματα νέφους ελέγχουν και βελτιστοποιούν αυτόματα τη χρήση των πόρων, αξιοποιώντας δυνατότητες μέτρησης/παρακολούθησης κατάλληλες για τον τύπο της υπηρεσίας (π.χ. αποθήκευση, επεξεργασία, εύρος ζώνης). Η χρήση των πόρων μπορεί να παρακολουθείται, να ελέγχεται και να καταγράφεται, παρέχοντας διαφάνεια τόσο στον πάροχο όσο και στον καταναλωτή της υπηρεσίας που χρησιμοποιείται. Η καταγραφόμενη χρήση έπειτα χρησιμοποιείται για την χρέωση του καταναλωτή ανάλογα με το μοντέλο χρέωσης που σχετίζεται με την χρησιμοποιούμενη υπηρεσία.

2.1.3 Μοντέλα Υπηρεσιών/Παράδοσης

Τα τρία μοντέλα υπηρεσιών ή παράδοσης της νεφο-υπολογιστικής, όπως αυτά αναγράφονται στο [14], είναι τα παρακάτω:

- **Software as a Service (SaaS) (Λογισμικό ως Υπηρεσία)**:

  Παρέχεται στον καταναλωτή η δυνατότητα χρήσης εφαρμογών εκτελούμενων σε μια υποδομή νέφους, οι οποίες προσφέρονται είτε από τον πάροχο της υποδομής είτε από τρίτο μέρος. Οι εφαρμογές αυτές είναι προσβάσιμες, από διάφορες συσκευές ικανές να συνδεθούν στο διαδίκτυο, μέσω φυλλομετρητή ή προγραμματιστικής διεπαφής. Δεν προσφέρεται έλεγχος ή δυνατότητα διαχείρισης της υποκείμενης υποδομής νέφους ή των δυνατοτήτων της υπηρεσίας (λογισμικού), με εξαίρεση περιορισμένη παραμετροποίηση κάποιων ρυθμίσεων διαμόρφωσης της εφαρμογής. Το μοντέλο χρέωσης είθισται να είναι της μορφής μιας σταθερής μηνιαίας ή ετήσιας συνδρομής χρησιμοποιώντας βαθμίδες με διαφορετικά επίπεδα παροχής υπηρεσιών του λογισμικού [15].

- **Platform as a Service (PaaS) (Πλατφόρμα ως Υπηρεσία)**:

  Παρέχεται η δυνατότητα ανάπτυξης και εκτέλεσης εφαρμογών σε ένα κατάλληλο περιβάλλον παρεχόμενο από μια πλατφόρμα που υποστηρίζεται από πόρους του υπολογιστικού νέφους. Οι εφαρμογές αυτές αναπτύσσονται από τον καταναλωτή μέσω της πλατφόρμας χρησιμοποιώντας ένα ολοκληρωμένο περιβάλλον ανάπτυξης και εκτέλεσης αποτελούμενο από runtimes γλωσσών προγραμματισμού, βιβλιοθήκες, υπηρεσίες και εργαλεία ανάπτυξης. Ο καταναλωτής δεν έχει τον έλεγχο της υποκείμενης υποδομής νέφους, αλλά έχει τον έλεγχο των εφαρμογών που εκτελούνται σε αυτήν, των δεδομένων τους, καθώς και των ρυθμίσεων διαμόρφωσής τους και του περιβάλλοντος ανάπτυξης/εκτέλεσής τους. Συνήθως, τα περιβάλλοντα είναι προκαθορισμένα ως προς το περιεχόμενο τους. Όμως, γίνεται προσπάθεια από τους παρόχους των υπηρεσιών PaaS να καλύψουν τις ανάγκες όλων των πιθανών ομάδων έργων λογισμικού παρέχοντας μια μεγάλη ποικιλία από διαφορετικά περιβάλλοντα. Το μοντέλο χρέωσης υπηρεσιών PaaS συνήθως περιλαμβάνει μια σταθερή χρέωση ανά χρονική περίοδο για κάθε είδος πόρου που χρειάστηκε να χρησιμοποιηθεί από τον πάροχο για την επίτευξη των απαιτήσεων της εφαρμογής του καταναλωτή μέσω της παρεχόμενης πλατφόρμας [16]. Ουσιαστικά, ο καταναλωτής χρεώνεται με βάση την χρήση των πόρων του παρόχου.

- **Infrastructure as a Service (IaaS) (Υποδομή ως Υπηρεσία)**:

  Παρέχεται η δυνατότητα χρήσης επεξεργαστικών, αποθηκευτικών, δικτυακών και άλλων ειδών υπολογιστικών πόρων. Συνήθως, οι πόροι αυτοί συγκροτούνται στην μορφή μιας εικονικής μηχανής, δηλ. ενός απογυμνωμένου περιβάλλοντος στο οποίο ο καταναλωτής μπορεί να εγκαταστήσει και να εκτελέσει το λογισμικό της επιλογής του, συμπεριλαμβανομένων λειτουργικών συστημάτων και εφαρμογών. Ο καταναλωτής δεν έχει τον έλεγχο της υποκείμενης υποδομής νέφους, αλλά έχει τον έλεγχο των λειτουργικών συστημάτων, του αποθηκευτικού χώρου, των περιβαλλόντων ανάπτυξης/εκτέλεσης, των εγκατεστημένων εφαρμογών, των δεδομένων τους και των ρυθμίσεων διαμόρφωσής τους. Το μοντέλο χρέωσης υπηρεσιών IaaS συνήθως αποτελείται από μια συνεχόμενη χρέωση ανά χρονική περίοδο λόγω της ανάθεσης των πόρων στον καταναλωτή, η οποία αυξάνεται μετά την υπέρβαση ενός ορίου χρήσης για ορισμένους πόρους, όπως το εύρος ζώνης δικτύου.

2.1.4 Μοντέλα Ανάπτυξης

Τα τέσσερα μοντέλα ανάπτυξης του υπολογιστικού νέφους σύμφωνα με το [14] και την Intel [17] είναι τα εξής:

- **Ιδιωτικό νέφος (Private Cloud)**:

  Το ιδιωτικό νέφος είναι αποκλειστικά αφιερωμένο σε έναν μόνο οργανισμό αποτελούμενο από πολλαπλούς καταναλωτές (π.χ. επιχειρησιακές μονάδες ή τμήματα). Ενδεχομένως να ανήκει, να διαχειρίζεται και να λειτουργεί από τον ίδιο τον οργανισμό, από μια τρίτη οντότητα, ή έναν συνδυασμό των δύο. Το νέφος αυτό μπορεί να βρίσκεται εντός ή εκτός του οργανισμού (π.χ. στην περίπτωση που λειτουργεί από τρίτη οντότητα). Παρέχει πλήρη έλεγχο στον τρόπο με τον οποίο μοιράζονται και αποθηκεύονται τα δεδομένα και διασφαλίζει την συμμόρφωση με τυχόν κανονισμούς, τους οποίους καλείται ένας οργανισμός να ακολουθήσει. Επιπλέον, λόγω της αποκλειστικής αφιέρωσής του σε έναν μόνο οργανισμό, εξασφαλίζεται η διαθεσιμότητα των δεδομένων κατά παραγγελία, όπως επίσης και η αξιοπιστία του για κρίσιμους φόρτους εργασίας. Τέλος, λόγω του πλήρους ελέγχου, μπορεί να εγκαθιδρυθεί ένα υψηλό επίπεδο ασφαλείας, υψηλότερο σε σχέση με αυτό που μπορεί να επιτευχθεί από άλλα μοντέλα ανάπτυξης (νέφους).

- **Κοινοτικό νέφος (Community Cloud)**:

  Είναι διαθέσιμο για μια κοινότητα καταναλωτών ή οργανισμών με κοινές ανησυχίες, όπως οι απαιτήσεις ασφαλείας και ζητήματα πολιτικής και συμμόρφωσης. Μπορεί να ανήκει, να διαχειρίζεται και να λειτουργεί από έναν ή περισσότερους οργανισμούς της κοινότητας, μια τρίτη οντότητα ή έναν συνδυασμό των δύο. Το νέφος αυτό μπορεί να βρίσκεται εντός ή εκτός των οργανισμών της κοινότητας (εφόσον λειτουργεί από ένα τρίτο μέρος). Το κόστος κτήσης, λειτουργίας και συντήρησης του νέφους συνήθως διαμοιράζεται μεταξύ των μελών/οργανισμών της κοινότητας. Επιπροσθέτως, το νέφος αυτού του είδους είναι προσβάσιμο συνήθως μόνο από τα μέλη της κοινότητας. Οπότε, μπορεί να θεωρηθεί ένα ιδιωτικό νέφος, όμως όχι μόνο για έναν αλλά για πολλαπλούς οργανισμούς. Το επίπεδο ασφάλειας που μπορεί να επιτευχθεί σε αυτό το είδος νέφους είναι υψηλότερο από το δημόσιο νέφος και χαμηλότερο από το ιδιωτικό.

- **Δημόσιο νέφος (Public Cloud)**:

  Εδώ, το νέφος υποδομής είναι διαθέσιμο για το γενικό κοινό. Μπορεί να ανήκει και να διαχειρίζεται από μια επιχείρηση, έναν ακαδημαϊκό ή κυβερνητικό οργανισμό ή έναν συνδυασμό των παραπάνω. Λειτουργεί εντός των υποδομών του παρόχου νέφους. Το δημόσιο νέφος προσφέρει είτε εικονικές μηχανές που εκτελούνται σε (φυσικούς) διακομιστές του παρόχου νέφους, των οποίων οι υπολογιστικοί πόροι χρησιμοποιούνται από πολλούς καταναλωτές ταυτόχρονα, είτε τους φυσικούς του διακομιστές για αποκλειστική χρήση. Αποτελεί το πιο δημοφιλές μοντέλο ανάπτυξης που επιλέγουν εταιρείες για την παροχή των υπηρεσιών τους, λόγω της απουσίας απαίτησης μεγάλου αρχικού κόστους επένδυσης για τον απαραίτητο εξοπλισμό και της ευελιξίας που παρέχεται μέσω της αυτοεξυπηρέτησης κατά παραγγελία. Αποτελεί κατάλληλη επιλογή για μεγάλους φόρτους εργασίας μικρής διάρκειας λόγω του μοντέλου χρέωσης ανά χρήση, ενώ διευκολύνει μια επιχείρηση στην μετέπειτα διαχείριση του κόστους με βάση τις προβλέψεις της ζήτησης της υπηρεσίας που αυτή προσφέρει και των δυνατοτήτων κλιμάκωσης υπηρεσιών που το νέφος παρέχει σε αυτήν. Όμως, το επίπεδο ασφάλειας του δημοσίου νέφους είναι το χαμηλότερο σε σχέση με τα άλλα 3 είδη νέφους.

- **Υβριδικό νέφος (Hybrid Cloud)**:

  Είναι συνδυασμός δύο ή περισσότερων νεφών (ιδιωτικού, κοινοτικού ή δημοσίου) που διατηρούνται ως ξεχωριστές οντότητες αλλά συνδέονται μεταξύ τους με πρότυπες ή ιδιόκτητες τεχνολογίες που επιτρέπουν τη φορητότητα δεδομένων και εφαρμογών. Η πιο συνηθισμένη μορφή ενός υβριδικού νέφους αντιστοιχεί στη χρήση ενός ιδιωτικού νέφους, το οποίο μπορεί να αντλεί επιπλέον πόρους από το δημόσιο νέφος, όταν φθάνει στα όρια της χωρητικότητάς του.

2.2 Εικονικοποίηση

Η εικονικοποίηση (virtualization) είναι ένας όρος που χρησιμοποιούμε όταν θέλουμε να αναφερθούμε στην αναπαράσταση πόρων σε εικονική μορφή με σκοπό την αναδιαμόρφωσή τους ούτως ώστε να ικανοποιούνται οι ανάγκες ενός συστήματος (όπως η αύξηση της χρήσης των φυσικών του πόρων). Εφαρμόζεται σε μια πληθώρα πόρων, όπως είναι οι διακομιστές, το λειτουργικό σύστημα, ακόμα και τα δεδομένα. Μπορεί επομένως να χωριστεί σε εικονικοποίηση φυσικών (υπολογιστικών ή μη) πόρων ή λογισμικού.

Η πιο συνηθισμένη χρήση εικονικοποίησης είναι αυτή των διακομιστών η οποία αποτελεί και τον πυλώνα της νεφο-υπολογιστικής. Προκειμένου να επιτευχθεί, απαιτείται η χρήση ενός υπερ-επόπτη. Δηλαδή, ενός λογισμικού υπεύθυνου για την κατάτμηση των φυσικών πόρων ενός διακομιστή σε μια ή περισσότερες εικονικές αναπαραστάσεις ενός συνόλου αυτών με σκοπό να χρησιμοποιηθούν ως ξεχωριστοί εικονικοί διακομιστές (virtual hosts/servers) (οι κοινώς λεγόμενες εικονικές μηχανές).

Ένας υπερ-επόπτης μπορεί να ανήκει σε δύο αποκλειστικές κατηγορίες. Είτε πρόκειται για υπερ-επόπτη τύπου 1 στην περίπτωση απευθείας πρόσβασης με το υλικό, είτε τύπου 2 εάν υπάρχει ήδη λειτουργικό σύστημα εγκατεστημένο στον υποκείμενο (φυσικό) διακομιστή προς κατάτμηση. Η επιλογή τύπου υπερ-επόπτη είναι άμεσα εξαρτώμενη από τις ανάγκες του κάθε χρήστη. Στην περίπτωση που η υψηλή ταχύτητα και απόδοση είναι σημαντικές μη λειτουργικές απαιτήσεις, η άμεση πρόσβαση του υπερ-επόπτη τύπου 1 στο υλικό συμβάλλει στην επίτευξη της ικανοποίησης αυτών των δύο απαιτήσεων. Από την άλλη, ένας υπερ-επόπτης τύπου 2 δεν έρχεται σε αντιπαράθεση με το υποκείμενο ΛΣ και επιτρέπει την χρήση προγραμμάτων και οδηγών που το ΛΣ αυτό πιθανόν να μην είναι σε θέση να εκτελέσει.

Η εικονικοποίηση διακομιστών χωρίζεται σε δύο κατηγορίες βάσει της μεθόδου επίτευξής της: την πλήρη εικονικοποίηση (full virtualization) και την παρα-εικονικοποίηση [18] (paravirtualization). Στην πρώτη περίπτωση το λειτουργικό σύστημα που εκτελείται στον εικονικό διακομιστή συμπεριφέρεται όπως θα συμπεριφερόταν σε έναν φυσικό διακομιστή. Αυτό συμβαίνει διότι από μεριάς του λειτουργικού συστήματος δεν υπάρχει επίγνωση της εικονικοποίησης που έχει λάβει μέρος για να δημιουργηθούν οι εικονικοί πόροι στους οποίους στεγάζεται. Η πλήρης εικονικοποίηση μπορεί να είναι δύο ειδών: υποβοηθούμενη από το λογισμικό (software-assisted) ή από το υλικό (hardware assisted). Στο πρώτο είδος, πραγματοποιείται

εικονικοποίηση ολόκληρου του υλικού μέσω του υπερ-επόπτη (τύπου 2) που εκτελείται στο ΛΣ. Στο δεύτερο είδος, όπου δεν υπάρχει ΛΣ, δύναται το λειτουργικό σύστημα του εικονικού διακομιστή να κάνει χρήση της φυσικής κεντρικής μονάδας επεξεργασίας του διακομιστή φιλοξενίας εάν αυτή το υποστηρίζει [19].

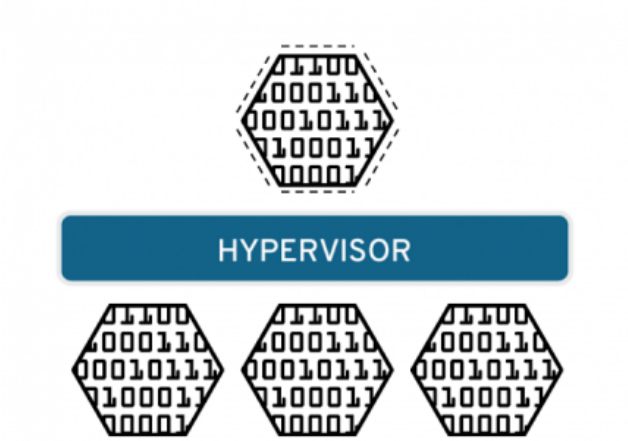
Στην περίπτωση της παρα-εικονικοποίησης το φιλοξενούμενο λειτουργικό σύστημα αναγνωρίζει την ύπαρξη του υπερ-επόπτη και απαιτείται η τροποποίηση και των δύο ώστε να μπορούν να επικοινωνούν με χρήση υπερ-κλήσεων. Αν και το γεγονός αυτό μειώνει την συμβατότητα σε σχέση με την πλήρη εικονικοποίηση, η άμεση πρόσβαση στο υποκείμενο φυσικό υλικό συμβάλλει στην αύξηση της αποδοτικότητας - αυτό επιτυγχάνεται κυρίως για ορισμένα είδη πόρων.

Σχετικά με την εικονικοποίηση λογισμικού, η πιο συνηθισμένη χρήση της είναι η εικονικοποίηση ΛΣ όπου τότε αναφερόμαστε στην δοχειοποίηση. Κατά την δοχειοποίηση, ενθυλακώνεται ένα πρόγραμμα εξ ολοκλήρου σε ένα εικονικό περιβάλλον που ονομάζεται δοχείο. Έπειτα, το δοχείο αυτό εκτελείται ως διεργασία του ΛΣ και μοιράζεται τους υποκείμενους πόρους του συστήματος με τα υπόλοιπα δοχεία και τα λοιπά προγράμματα που μπορεί να εκτελούνται στο σύστημα αυτό. Περισσότερες λεπτομέρειες για την δοχειοποίηση, αναφέρονται και στην Ενότητα 2.4.

2.2.1 Είδη εικονικοποίησης

Υπάρχουν πολλά είδη εικονικοποίησης. Πέντε βασικά αυτών, όπως αναφέρονται από την Red Hat [20], συνδυαστικά με τρία ακόμα που χρησιμοποιούνται συχνά, είναι τα παρακάτω:

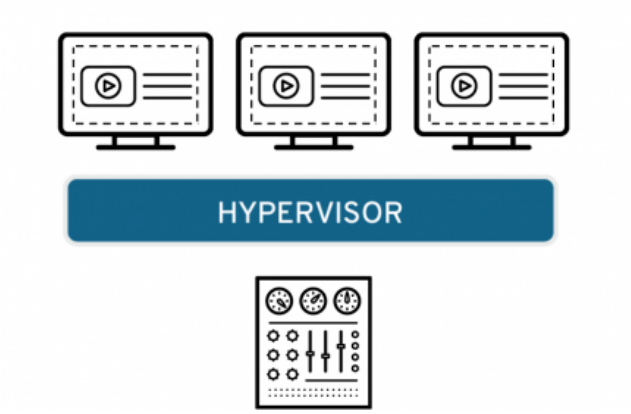
- **Εικονικοποίηση Δεδομένων**:



Σχήμα 2.1: Εικονικοποίηση Δεδομένων [20]

Η εικονικοποίηση δεδομένων είναι μια προσέγγιση ενσωμάτωσης δεδομένων από διαφορετικές πηγές, σε μια ολιστική, λογική προβολή δίχως την ανάγκη της φυσικής μετακίνησής τους [21]. Δηλαδή, διασκορπισμένα ετερογενή δεδομένα παρεχόμενα από πηγές διαφόρων τοποθεσιών δύναται να συσσωματωθούν σε μοναδικά, λογικά τεμάχια μιας ενιαίας εικονικής πηγής. Με αυτόν τον τρόπο, οι εταιρείες μπορούν από ένα μόνο μοντέλο διαχείρισης δεδομένων να οργανώσουν και να επεξεργαστούν διασκορπισμένες πληροφορίες με γνώμονα τις ανάγκες των χρηστών με μεγαλύτερη ευκολία και αποδοτικότητα, χωρίς την ανάγκη να γνωρίζουν τεχνικές λεπτομέρειες (όπως γλώσσες πρόσβασης, δομές αποθήκευσης κα.). Τομείς οι οποίοι επωφελούνται από την εικονικοποίηση δεδομένων είναι η λήψη αποφάσεων, η επιχειρηματική αναλυτική και η αξιολόγηση των κινδύνων.

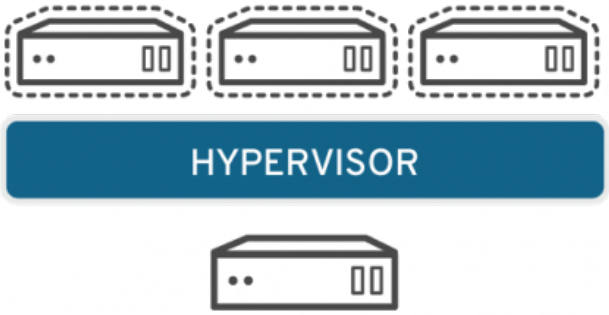
- **Εικονικοποίηση Επιφάνειας Εργασίας**:



Σχήμα 2.2: Εικονικοποίηση Επιφάνειας Εργασίας [20]

Με την εικονικοποίηση επιφάνειας εργασίας, δίνεται σε έναν κεντρικό διαχειριστή η ικανότητα διαμοιρασμού προσομοιωμένων περιβαλλόντων εργασίας σε εκατοντάδες φυσικές μηχανές ή συσκευές ταυτοχρόνως. Εν αντιθέσει με τα παραδοσιακά περιβάλλοντα εργασίας που χρήζουν εγκατάστασης, διαμόρφωσης και ενημέρωσης σε κάθε υπολογιστή, η εικονικοποίηση επιφάνειας εργασίας καθιστά δυνατή τη μαζική διαμόρφωση, ενημέρωση και έλεγχο ασφαλείας σε όλα τα εικονικά περιβάλλοντα εργασίας που παρέχονται από έναν μόνο διακομιστή. Καθ' αυτόν τον τρόπο, οι επιχειρήσεις επιτρέπουν στους χρήστες να μπορούν να εργαστούν από οπουδήποτε και με κάθε συσκευή ανεξαρτήτως του είδους ή του λειτουργικού συστήματός τους [22].

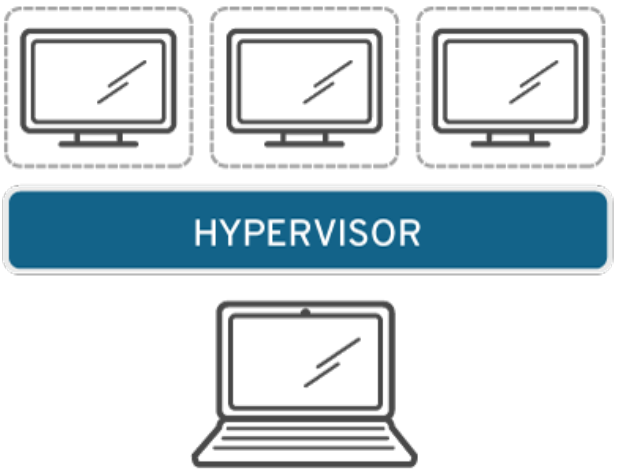
- **Εικονικοποίηση Διακομιστών**:



Σχήμα 2.3: Εικονικοποίηση Διακομιστών [20]

Οι διακομιστές είναι υπολογιστές σχεδιασμένοι με σκοπό να επεξεργάζονται πολύ καλά έναν μεγάλο όγκο συγκεκριμένων διεργασιών, ώστε οι κύριοι υπολογιστές μιας επιχείρησης να μπορούν να δίνουν προτεραιότητα σε άλλες εργασίες. Με την εικονικοποίηση διακομιστών αναφερόμαστε στην διαδικασία κατά την οποία ένας φυσικός διακομιστής χωρίζεται σε πολλούς μικρότερους εικονικούς διακομιστές, με απώτερο σκοπό την αποτελεσματικότερη αξιοποίηση των πόρων του. Αυτό είναι απαραίτητο διότι προτιμάται για λόγους ευκολίας της διαχείρισής τους, κάθε διακομιστής να είναι υπεύθυνος για μια μόνο διεργασία την φορά. Μετά την κατάτμησή του, ενώ μπορεί να ακολουθείται η ίδια πρακτική, παύει το φυσικό μηχάνημα να μένει με αχρησιμοποίητους πόρους και πρακτικά μπορεί το σύνολο τον πόρων του να χρησιμοποιηθεί για την εξυπηρέτηση πολλαπλών λειτουργιών. Ο ορισμός της εικονικοποίησης διακομιστών αναλύεται πιο λεπτομερώς και στην Ενότητα 2.2.4.

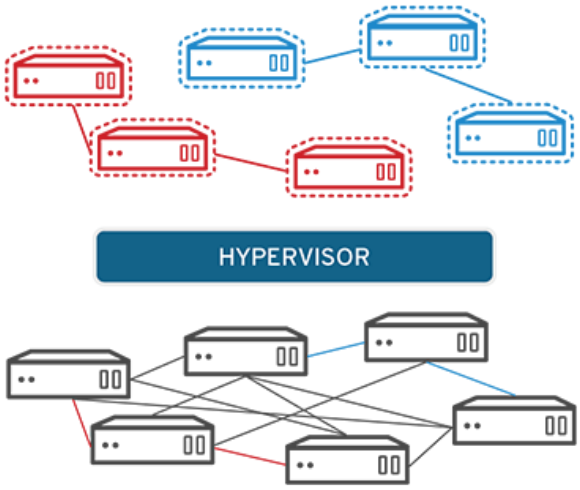
- **Εικονικοποίηση Λειτουργικού Συστήματος**:



Σχήμα 2.4: Εικονικοποίηση Λειτουργικού Συστήματος [20]

Η εικονικοποίηση λειτουργικού συστήματος είναι κάτι που συμβαίνει στον πυρήνα (ΛΣ). Ουσιαστικά, αναφερόμαστε στην διαδικασία της δοχειοποίησης λειτουργικών συστημάτων. Κατά την εφαρμογή της, μπορούν να εκτελεστούν ταυτοχρόνως πολλαπλά λειτουργικά συστήματα μέσα σε απομονωμένα εικονικά περιβάλλοντα, όπου το κάθε ένα από αυτά μοιράζεται τον ίδιο πυρήνα με το λειτουργικό σύστημα φιλοξενίας. Μεγαλύτερη ανάλυση της εικονικοποίησης σε επίπεδο λειτουργικού συστήματος πραγματοποιείται στην Ενότητα 2.4.1.

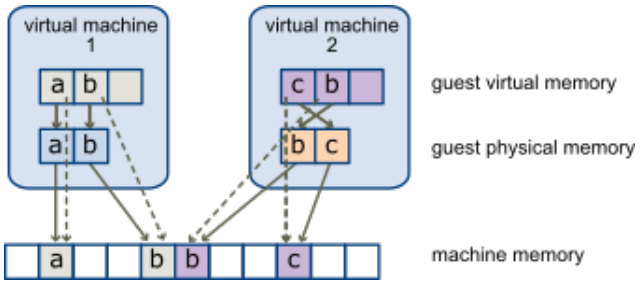
- **Εικονικοποίηση Λειτουργιών Δικτύου**:



Σχήμα 2.5: Εικονικοποίηση Λειτουργιών Δικτύου [20]

Η εικονικοποίηση λειτουργιών δικτύου (Network Functions Virtualization - NFV) αφορά τον διαχωρισμό των βασικών λειτουργιών ενός δικτύου (όπως ο διαμοιρασμός αρχείων, και η διαμόρφωση IP) από το φυσικό υλικό (που συνήθως χρησιμοποιούνταν για την εκτέλεσή τους), ώστε να μπορούν να διανεμηθούν σε διάφορα περιβάλλοντα. Παραδοσιακά, οι λειτουργίες δικτύου εκτελούνταν σε ιδιόκτητο υλικό συγκεκριμένου σκοπού και επομένως ήταν απαραίτητο να πραγματοποιηθεί αγορά, ρύθμιση και συντήρηση του κάθε φυσικού εξαρτήματος. Με την αύξηση όμως της δημοτικότητας των τεχνολογιών εικονικοποίησης, άρχισε να γίνεται πιο δημοφιλής και η πρακτική πακεταρίσματος των λειτουργιών των εξαρτημάτων αυτών σε διακομιστές κοινής χρήσης (commodity servers). Το αποτέλεσμα αυτού, ήταν η απόκτηση της δυνατότητας εκτέλεσης των λειτουργιών δικτύου μιας επιχείρησης σε τυπικούς διακομιστές γενικού σκοπού και κατά προέκταση, η αντικατάσταση κάθε ξεχωριστού φυσικού μηχανήματος με ένα αντίστοιχο εικονικό εκτελούμενο μέσα από μια εικονική μηχανή [23]. Η εικονικοποίηση των λειτουργιών δικτύων μειώνει τον αριθμό των φυσικών εξαρτημάτων, όπως οι μεταγωγείς, δρομολογητές, διακομιστές, καλώδια και κόμβοι, που απαιτούνται για τη δημιουργία πολλαπλών, ανεξάρτητων δικτύων και είναι ιδιαίτερα δημοφιλής στον κλάδο των τηλεπικοινωνιών.

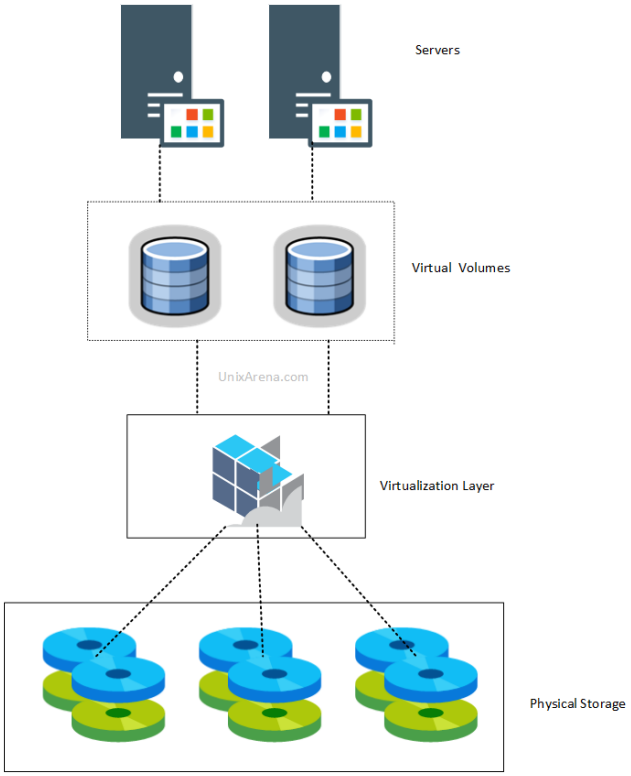
- **Εικονικοποίηση Μνήμης**:



Σχήμα 2.6: Εικονικοποίηση Μνήμης [24]

Η εικονικοποίηση μνήμης αποτελεί ένα κομμάτι της ευρύτερης έννοιας της εικονικοποίησης πόρων [25]. Συγκεκριμένα, είναι μια τεχνική κατά την οποία δύναται να διαχειριστούμε με έναν πιο αποδοτικό τρόπο την φυσική μνήμη (RAM) που χρησιμοποιείται στα υπολογιστικά μας συστήματα. Αυτό συμβαίνει διότι στην βασικότερη μορφή της, η εικονικοποίηση μνήμης εμφανίζεται ως εικονική μνήμη ή όπως η μνήμη swap σε διακομιστές και σταθμούς εργασίας [26]. Δηλαδή, ως επιπρόσθετη μνήμη την οποία το σύστημα εκλαμβάνει ως πραγματική και μπορεί να την χρησιμοποιήσει. Για να επιτευχθεί αυτό, μέσω του υπερ-επόπτη πραγματοποιείται αντιστοίχιση σελίδων φυσικής μνήμης του φιλοξενούμενου λειτουργικού συστήματος στις σελίδες φυσικής μνήμης της υποκείμενης μηχανής. Καθ' αυτόν τον τρόπο, κάθε εικονική μηχανή βλέπει έναν συνεχόμενο χώρο διευθύνσεων μνήμης που δύναται να χρησιμοποιήσει [24]. Ως αποτέλεσμα, επιτυγχάνεται εν γένει υψηλότερη αξιοποίηση της μνήμης και η δυνατότητα διαμοιρασμού μιας κοινής δεξαμενής μνήμης ακόμα και σε κατανεμημένα συστήματα, παρακάμπτοντας τους περιορισμούς της φυσικής μνήμης [27].

- **Εικονικοποίηση Αποθήκευσης**:

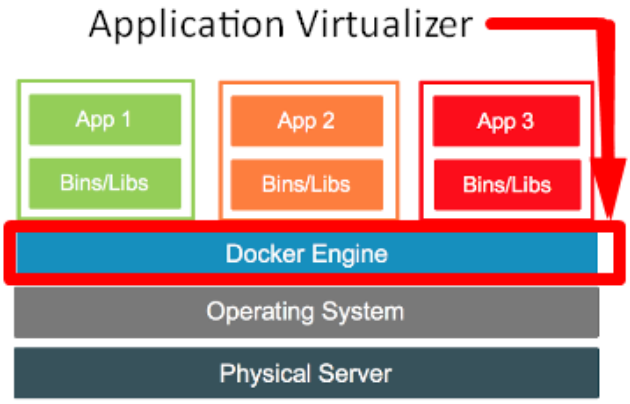


Σχήμα 2.7: Εικονικοποίηση Αποθήκευσης [28]

Ένα ακόμα κομμάτι της ευρύτερης έννοιας της εικονικοποίησης πόρων είναι και η εικονικοποίηση αποθήκευσης. Συγκεκριμένα, ο όρος εικονικοποίηση αποθήκευσης αναφέρεται στην πρακτική της συγκέντρωσης φυσικού αποθηκευτικού χώρου από πολλαπλές συσκευές αποθήκευσης σε μια φαινομενικά ενιαία, εικονική συσκευή [29]. Παρομοίως με την εικονικοποίηση μνήμης, αυτό είναι κάτι που θα επιτρέψει την υψηλότερη αξιοποίηση ενός πόρου. Συγκεκριμένα, του αποθηκευτικού χώρου (π.χ. ενός δίσκου). Με την χρήση της έρχονται πολλά πλεονεκτήματα. Αρχικά, επιφέρει μεγαλύτερη ευελιξία στον τομέα της αποθήκευσης. Επιπλέον, εγγυάται υψηλή διαθεσιμότητα και ευκολία στην δημιουργία αντιγράφων ασφαλείας. Χρήσιμη λειτουργία που παρέχεται μέσω της εικονικοποίησης αποθήκευσης αποτελεί και η αφαίρεση ή το κρύψιμο ετερογένειας αποθηκευτικών συσκευών που εικονικοποιούνται, οι οποίες μπορεί να προέρχονται από διαφορετικούς πωλητές ή από τις εφαρμογές που τις διαχειρίζονται. Τέλος, προσφέρει έναν ενιαίο τρόπο διαχείρισης των συσκευών μέσω κεντρικοποιημένων κονσόλων ή APIs και επομένως, αποτελεί ένα πιο διαχειρίσιμο μοντέλο χώρου αποθήκευσης σε σχέση με το παραδοσιακό, όπου κάθε υπολογιστής έχει πρόσβαση μονάχα στον δικό του δίσκο.

Προκειμένου να γίνει πράξη, απαιτείται αναλόγως την μέθοδο εικονικοποίησης και τον τύπο της, είτε να χρησιμοποιηθεί ένας αλγόριθμος για να εντοπίσει δυναμικά τα δεδομένα είτε να δημιουργηθεί ένας χάρτης αντιστοίχισής τους χρησιμοποιώντας μεταδεδομένα [30]. Αφότου γίνει αυτό, τα δεδομένα πλέον αποθηκεύονται σε ένα αρχείο και οι συστοιχίες δίσκων τοποθετούνται μέσα σε μια εικονική δεξαμενή. Έπειτα, κάθε αίτημα ανάγνωσης και εγγραφής από τις εφαρμογές φιλτράρεται και δύναται, π.χ. έμμεσα μέσω των αντιστοιχιών, η δυνατότητα εύρεσης και αποθήκευσης δεδομένων.

- **Εικονικοποίηση Εφαρμογών**:



Σχήμα 2.8: Εικονικοποίηση Εφαρμογών [31]

Η εικονικοποίηση εφαρμογών είναι μια τεχνολογία λογισμικού που ενθυλακώνει τα προγράμματα από το υποκείμενο λειτουργικό σύστημα στο οποίο εκτελούνται. Υπάγεται στην διαδικασία της δοχειοποίησης και επομένως, τα προγράμματα που ενθυλακώνονται, συμπεριλαμβανομένων των βιβλιοθηκών και εξαρτήσεών τους, παίρνουν την μορφή δοχείων. Στα δοχεία αυτά, εικονικοποιούνται οι απαραίτητοι υπολογιστικοί πόροι όπως το λειτουργικό σύστημα, η μνήμη και η κεντρική μονάδα επεξεργασίας [32], κάνοντας χρήση των διαθέσιμων μηχανισμών απομόνωσης του πυρήνα του λειτουργικού συστήματος. Περισσότερες λεπτομέρειες για τον τρόπο κατά τον οποίο επιτυγχάνεται αυτό, παρουσιάζονται στην Ενότητα 2.4.1.

2.2.2 Ιστορική αναδρομή της εικονικοποίησης

Όπως αναφέρει μια θυγατρική της IBM [20], ενώ η τεχνολογία εικονικοποίησης χρονολογείται από τη δεκαετία του 1960, δεν υιοθετήθηκε ευρέως μέχρι τις αρχές της δεκαετίας του 2000. Οι τεχνολογίες που την έκαναν πραγματικότητα, όπως οι υπερ-επόπτες, αναπτύχθηκαν πριν από δεκαετίες για να δώσουν σε πολλούς χρήστες ταυτόχρονη πρόσβαση σε υπολογιστές που επεξεργαζόντουσαν πολλά δεδομένα ταυτόχρονα. Κάτι ιδιαίτερα δημοφιλές στον τομέα των επιχειρήσεων για καθήκοντα ρουτίνας (routine tasks) που έπρεπε να εκτελεστούν χιλιάδες φορές και πολύ γρήγορα όπως η μισθοδοσία υπαλλήλων.

Ωστόσο, μέσα στις επόμενες δεκαετίες, ήρθαν στο προσκήνιο άλλες λύσεις στο πρόβλημα διαμοιρασμού ενός μηχανήματος σε πολλούς χρήστες, μειώνοντας έτσι το ενδιαφέρον για την τεχνολογία εικονικοποίησης. Μία από αυτές ήταν ο διαμοιρασμός χρόνου (time-sharing), όπου ένας χρήστης μπορούσε να χρησιμοποιεί το λειτουργικό σύστημα απομονωμένα από τους υπόλοιπους. Κάτι που οδήγησε στη δημιουργία λειτουργικών συστημάτων όπως το UNIX, το οποίο με τη σειρά του άνοιξε το δρόμο για την άφιξη του Linux. Καθ' όλη τη διάρκεια αυτή, η εικονικοποίηση παρέμεινε σε μεγάλο βαθμό μη διαδεδομένη.

Προχωρώντας στη δεκαετία του 1990, οι περισσότερες επιχειρήσεις διέθεταν φυσικούς διακομιστές και στοίβες μηχανημάτων ενός προμηθευτή, οι οποίες δεν είχαν τη δυνατότητα εκτέλεσης εφαρμογών σε υλικό διαφορετικού προμηθευτή. Καθώς οι εταιρείες αναβάθμιζαν τα περιβάλλοντα πληροφορικής τους με λιγότερο δαπανηρούς διακομιστές, λειτουργικά συστήματα και εφαρμογές από διάφορους προμηθευτές, ήταν υποχρεωμένες να υπολειτουργούν το φυσικό υλικό, αφού κάθε διακομιστής μπορούσε συνήθως να εκτελέσει μόνο μια εργασία/εφαρμογή που είχε υλοποιηθεί με βάση το υλικό του συγκεκριμένου προμηθευτή (του διακομιστή).

Από εκείνο το σημείο και έπειτα, άρχισε να γίνεται εμφανής η ανάγκη της εικονικοποίησης και να ανεβαίνει η δημοτικότητά της. Οι εταιρείες μπορούσαν πλέον να διαμερίσουν τους διακομιστές τους και να εκτελούν ακόμα και τις παλαιές τους εφαρμογές σε πολλούς τύπους και εκδόσεις λειτουργικών συστημάτων. Οι διακομιστές άρχισαν να χρησιμοποιούνται πιο αποδοτικά ή και καθόλου (όταν η ζήτηση ήταν ελάχιστη), μειώνοντας το απαιτούμενο κόστος αγοράς, εγκατάστασης, συντήρησης και ψύξης τους.

Η ευρεία εφαρμογή της εικονικοποίησης συνέβαλε στη μείωση του εγκλωβισμού σε έναν μόνο προμηθευτή και την κατέστησε το θεμέλιο του υπολογιστικού νέφους. Σήμερα είναι τόσο διαδεδομένη σε όλες τις επιχειρήσεις που συχνά απαιτείται εξειδικευμένο λογισμικό διαχείρισης των εικονικών πόρων [33] για να μπορέσει κανείς να παρακολουθεί τα δρώμενα μιας επιχείρησης. Πρόκειται για ένα λογισμικό το οποίο διασυνδέεται με εικονικά περιβάλλοντα και το υποκείμενο

φυσικό υλικό τους με απώτερο σκοπό την απλοποίηση της διαχείρισης πόρων, τη βελτίωση της ανάλυσης δεδομένων και τον εξορθολογισμό των λειτουργιών τους. Ουσιαστικά, αποτελεί ένα λογισμικό που συνδέεται απομακρυσμένα με υπερ-επόπτες, προσφέροντας μια φιλική προς τον χρήση διεπαφή και επιπρόσθετες λειτουργίες, όπως η συγκρότηση αναφορών χρήσης, η αυτοματοποίηση επιβολής κανόνων και η παρακολούθηση χρήσης εικονικών περιβαλλόντων.

2.2.3 Τι είναι ένας υπερ-επόπτης

Προτού οι υπερ-επόπτες έρθουν στο προσκήνιο, οι περισσότεροι φυσικοί υπολογιστές μπορούσαν να εκτελέσουν ένα λειτουργικό σύστημα τη φορά. Αυτό συνέβαλε στη σταθερότητα τους μιας και δε χρειαζόταν να διαχειριστούν αιτήματα από άλλα λειτουργικά συστήματα. Αυτή η προσέγγιση όμως είχε ένα μειονέκτημα. Μεγάλο κομμάτι των πόρων του συστήματος έμενε ανεκμετάλλευτο.

Τη λύση σε αυτό το πρόβλημα την έφερε η εισαγωγή των υπερ-εποπτών. Πρόκειται για μια στρώση λογισμικού που καθιστά δυνατή την εκτέλεση πολλαπλών λειτουργικών συστημάτων, το ένα δίπλα στο άλλο, μοιράζοντας τους ίδιους φυσικούς πόρους σε κάθε ένα από αυτά. Η πράξη αυτή ονομάζεται εικονικοποίηση και τα στιγμιότυπα των λειτουργικών συστημάτων λέγονται εικονικές μηχανές και αντιπροσωπεύουν προσομοιώσεις φυσικών υπολογιστών.

Οι υπερ-επόπτες είναι υπεύθυνοι για τη διαχείριση των εικονικών μηχανών χωρίζοντάς τις και αναθέτοντας σε κάθε μια ένα κομμάτι της διαθέσιμης υπολογιστικής ισχύος, μνήμης και χώρου αποθήκευσης. Αυτή η διαδικασία τις αποτρέπει από την αλληλεπίδραση μεταξύ τους. Μάλιστα, στην περίπτωση κατάρρευσης μιας εικονικής μηχανής, οι υπόλοιπες παραμένουν ανεπηρέαστες.

2.2.3.1 Είδη υπερ-εποπτών

Οι υπερ-επόπτες χωρίζονται σε δύο κατηγορίες ανάλογα με το περιβάλλον στο οποίο εκτελούνται. Με βάση την IBM [34], αυτές είναι:

- **Τύπου 1 (Bare Metal)**:

  Ένας υπερ-επόπτης τύπου 1 εκτελείται απευθείας στο φυσικό υλικό του υποκείμενου υπολογιστή, αλληλεπιδρώντας άμεσα με την κεντρική μονάδα επεξεργασίας, τη μνήμη και το φυσικό αποθηκευτικό χώρο. Για το λόγο αυτό, οι υπερ-επόπτες τύπου 1 αναφέρονται επίσης ως bare-metal υπερ-επόπτες και αντικαθιστούν το λειτουργικό σύστημα του

κεντρικού υπολογιστή. Η άμεση πρόσβαση στο φυσικό υλικό, τους καθιστά ιδιαίτερα αποδοτικούς. Παρ' όλα αυτά, επειδή αντικαθιστούν το ΛΣ, προκειμένου να μπορούν να αξιοποιηθούν απομακρυσμένα, συχνά απαιτείται μια ξεχωριστή μηχανή στην οποία θα εκτελείται λογισμικό διαχείρισης εικονικών πόρων [35], μέσω του οποίου προσφέρεται μια διεπαφή για τον έλεγχο των εικονικών μηχανών και του υλικού του κεντρικού υπολογιστή.

- **Τύπου 2 (Hosted)**:

  Ένας υπερ-επόπτης τύπου 2 δεν εκτελείται απευθείας στο υποκείμενο υλικό. Αντ' αυτού, εκτελείται ως εφαρμογή σε ένα υπάρχον λειτουργικό σύστημα. Η χρήση τους δε συνηθίζεται σε περιβάλλοντα με πολλούς διακομιστές λόγω της καθυστέρησης που εισάγεται εξαιτίας της συνεχούς επικοινωνίας του με το ΛΣ φιλοξενίας και επειδή το υποκείμενο αυτό ΛΣ βάζει σε προτεραιότητα τις δικές του εφαρμογές και λειτουργίες έναντι αυτών του υπερ-επόπτη [36].

Σε κάθε τύπο υπερ-επόπτη, όταν το φιλοξενούμενο ΛΣ αιτηθεί πρόσβαση στους πόρους υπολογισμού, μνήμης και δικτύου του φυσικού υλικού, όλες οι προσβάσεις περνάνε πρώτα από αυτόν. Στην περίπτωση όμως υπερ-επόπτη τύπου 2, επειδή εκτελείται ως εφαρμογή του ΛΣ φιλοξενίας, οι προσβάσεις αυτές χρειάζεται να μεταφραστούν προτού περάσουν στο φιλοξενούμενο ΛΣ και τους υποκείμενους πόρους του. Επομένως, σε αντίθεση με τους υπερ-επόπτες τύπου 1 όπου η πρόσβαση γίνεται άμεσα, η χρήση υπερ-εποπτών τύπου 2 εισάγει προβλήματα καθυστέρησης που μπορεί να επηρεάσουν την απόδοση. Κατά την χρήση υπερ-επόπτη τύπου 2 παρέχεται μεγαλύτερη συμβατότητα/γκάμα υλικού διότι αυτό διαχειρίζεται από το υποκείμενο ΛΣ φιλοξενίας. Επιπροσθέτως, εισάγεται πιθανός κίνδυνος ασφαλείας εάν ένας εισβολέας παραβιάσει το κεντρικό λειτουργικό σύστημα, επειδή θα μπορούσε στη συνέχεια να χειραγωγήσει οποιοδήποτε φιλοξενούμενο λειτουργικό σύστημα εκτελείται σε αυτό.

Από την άλλη μεριά, οι υπερ-επόπτες τύπου 2 είναι καταλληλότεροι για μεμονωμένους τελικούς χρήστες υπολογιστών που έχουν την ανάγκη να εκτελέσουν πολλαπλά λειτουργικά συστήματα (σε έναν υπολογιστή). Παραδείγματα τέτοιων χρηστών είναι μηχανικοί, επαγγελματίες ασφαλείας που αναλύουν κακόβουλο λογισμικό και υπάλληλοι επιχειρήσεων που χρειάζονται πρόσβαση σε εφαρμογές που είναι διαθέσιμες αποκλειστικά σε διαφορετικές πλατφόρμες λογισμικού από τη δική τους. Διατίθενται συχνά πρόσθετες εργαλειοθήκες για τους χρήστες, οι οποίες μπορούν να εγκατασταθούν στο υποκείμενο λειτουργικό σύστημα προκειμένου να παρέχουν βελτιωμένες συνδέσεις μεταξύ του υποκείμενου λειτουργικού συστήματος και εκείνου της εικονικής μηχανής. Οι πρόσθετες δυνατότητες που υποστηρίζονται μετά την παραπάνω διαδικασία μπορεί να είναι η αποκοπή και επικόλληση μεταξύ των δύο συστημάτων

ή η κοινή πρόσβαση στον αποθηκευτικό χώρο. Η τρέχουσα προσέγγιση επιτρέπει τη γρήγορη εναλλαγή σε διαφορετικά λειτουργικά συστήματα πέραν του ήδη υπάρχοντος, πράγμα που αυξάνει την παραγωγικότητα του τελικού χρήστη, αφού μπορεί να έχει πρόσβαση σε εργαλεία που δεν υποστηρίζονται στο δικό του (αρχικό/υπάρχον σύστημα).

Εξαίρεση στον κανόνα απουσίας λειτουργικού συστήματος αποτελεί η χρήση ενός υπερεπόπτη ανοιχτού κώδικα βασισμένου στο KVM (Kernel-based Virtual Machine), που επιτρέπει στο Linux να συμπεριφέρεται ως ένας υπερ-επόπτης. Αυτό συμβαίνει διότι το KVM αποτελεί κομμάτι του πυρήνα του Linux από την έκδοση 2.6.20 και έπειτα, επιτρέποντάς του να επωφεληθεί από τους διαθέσιμους μηχανισμούς απομόνωσης μέσω αυτού. Επομένως, του προσφέρεται η ικανότητα να λάβει αποκλειστικούς πόρους από το φυσικό μηχάνημα [37], κάτι που εξαλείφει το μειονέκτημα έλλειψης προτεραιότητας των διεργασιών ενός υπερ-επόπτη τύπου 2 έναντι αυτών του λειτουργικού συστήματος φιλοξενίας.

2.2.3.2 Χαρακτηριστικά ενός υπερ-επόπτη

Αν και υπάρχουν διαφορετικά είδη υπερ-εποπτών, όλοι έχουν κάποια χαρακτηριστικά που πρέπει να λάβει κανείς υπόψιν όταν επιλέγει ποιον θα χρησιμοποιήσει. Μερικά σημαντικά αυτών, όπως αναφέρονται από την IBM [34], είναι:

- **Απόδοση**:

  Βασικό χαρακτηριστικό ενός υπερ-επόπτη είναι η απόδοσή του. Αυτή διαφέρει από τον ένα υπερ-επόπτη στον άλλο αναλόγως την κατασκευή και τον τύπο του. Όμως, εν γένει, οι υπερ-επόπτες τύπου 1 θα πρέπει να παρέχουν απόδοση κοντά στην εγγενή λόγω της απουσίας ανάγκης μετάφρασης των αιτημάτων του φιλοξενούμενου ΛΣ.

- **Εργαλεία διαχείρισης**:

  Η εκτέλεση εικονικών μηχανών δεν αποτελεί το μοναδικό καθήκον ενός διαχειριστή κατά τη χρήση ενός υπερ-επόπτη. Απαραίτητες πρόσθετες ενέργειες είναι η συντήρηση και η ανάλυσή τους, καθώς και η διαγραφή όσων δε χρησιμοποιούνται πλέον. Επομένως, η ύπαρξη εργαλείων που να καθιστούν δυνατές αυτές τις ενέργειες αποτελεί σημαντικό παράγοντα κατά την επιλογή λογισμικού υπερ-επόπτη.

- **Οικοσύστημα**:

  Για τη διαχείριση ενός υπερ-επόπτη σε διάφορες κλίμακες πρέπει να υπάρχει καλή τεκμηρίωση και διάφορα εργαλεία διαχείρισης είτε επίσημα είτε από την κοινότητα που να επιτρέπουν δυνατότητες, όπως δημιουργία αντιγράφων ασφαλείας, ανάλυση χωρητικότητας και διαχείριση εναλλαγής εικονικών μηχανών [38] με αντίγραφά τους σε περιπτώσεις σφάλματος του λειτουργικού συστήματος της εκτελούμενης (εικονικής μηχανής).

- **Μεταφορά κατά τη λειτουργία**:

  Πρέπει να υπάρχει η δυνατότητα μεταφοράς εικονικών μηχανών από έναν υπερ-επόπτη σε έναν δεύτερο σε διαφορετική φυσική μηχανή, ιδανικά χωρίς την ανάγκη διακοπής της λειτουργίας τους. Ένα χαρακτηριστικό που χρησιμεύει τόσο για την αποτροπή αποτυχίας παροχής υπηρεσιών της εκάστοτε επιχείρησης όσο και για την εξισορρόπηση του φόρτου εργασίας.

- **Κόστος**:

  Το κόστος είναι ένας παράγοντας που πρέπει να ληφθεί υπόψιν κατά την επιλογή ενός υπερ-επόπτη. Οι περισσότεροι είναι δωρεάν αλλά υπάρχουν και εμπορικές εκδόσεις που προσφέρουν περισσότερες δυνατότητες. Για παράδειγμα, μπορεί να προσφέρεται λογισμικό διαχείρισης των λειτουργιών τους που να επιτρέπει την εύκολη κλιμάκωση με βάση τις απαιτήσεις της επιχείρησης.

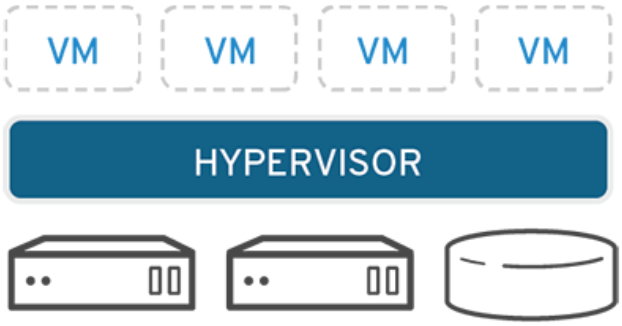
2.2.3.3 Επιλογή υπερ-επόπτη

Σε κάθε περίπτωση, η ασφάλεια αποτελεί έναν σημαντικό παράγοντα επιλογής υπερ-επόπτη διότι στην περίπτωση παραβίασής του, η χειραγώγηση όλων των εικονικών μηχανών που αυτός διαχειρίζεται θα μπορεί να πραγματοποιηθεί χωρίς μεγάλη δυσκολία.

Η επιλογή του τύπου υπερ-επόπτη που θα χρησιμοποιηθεί είναι άμεσα εξαρτώμενη από τις ανάγκες του κάθε τελικού χρήση ή επιχείρησης ζυγίζοντας τα πλεονεκτήματα και μειονεκτήματα που έρχονται με την κάθε επιλογή. Παραδείγματος χάριν, αν η ταχύτητα και η απόδοση θεωρούνται από τον χρήστη χαρακτηριστικά υψίστης σημασίας, τότε η επιλογή υπερ-επόπτη τύπου 1 αποτελεί μονόδρομο. Επιπλέον, ο μεγαλύτερος βαθμός απομόνωσης που προσφέρουν, παρέχει και μεγαλύτερη ασφάλεια. Από την άλλη, αν η προτεραιότητα είναι η ευκολία διαχείρισης και η ευελιξία ταχείας εναλλαγής σε διαφορετικά λειτουργικά συστήματα, η πιο ταιριαστή επιλογή είναι ένας υπερ-επόπτης τύπου 2.

2.2.4 Εικονικοποίηση Διακομιστών

Σύμφωνα με τον ορισμό της Red Hat [39], η εικονικοποίηση είναι μια τεχνολογία που μας επιτρέπει να δημιουργήσουμε πολλαπλά εικονικά περιβάλλοντα ή αποκλειστικούς πόρους από ένα μόνο, φυσικό σύστημα υλικού. Ένα λογισμικό ονόματι υπερ-επόπτης (hypervisor) συνδέεται στο υλικό αυτό[1] και δίνει τη δυνατότητα διαμερισμού ενός συστήματος σε ξεχωριστά, διακριτά και ασφαλή περιβάλλοντα, γνωστά και ως εικονικές μηχανές (Virtual Machines - VMs). Επομένως, αυτές οι εικονικές μηχανές βασίζονται στην ικανότητα του υπερ-επόπτη να διαχωρίζει τους πόρους της μηχανής και να τους κατανέμει κατάλληλα. Οι εικονικές μηχανές έχουν τη μορφή ενός ενιαίου αρχείου, πράγμα που καθιστά εύκολη τη μεταφορά και ανάγνωσή τους από οποιονδήποτε υπολογιστή αναμένοντας τον ίδιο τρόπο λειτουργίας.



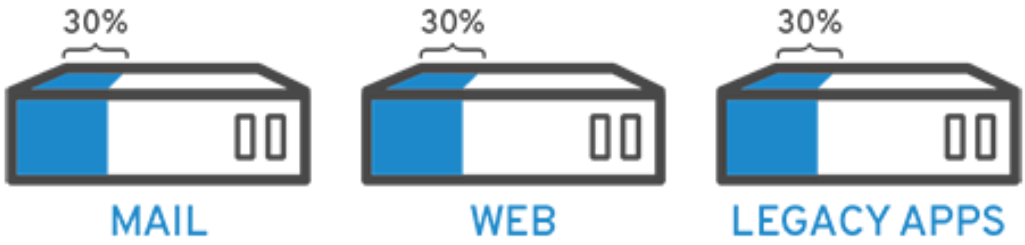
Σχήμα 2.9: Υπερ-επόπτης πάνω σε διακομιστές [20]

Το φυσικό υλικό, εξοπλισμένο με έναν υπερ-επόπτη, ονομάζεται ξενιστής (host) ή μηχάνημα φιλοξενίας (hosting machine), ενώ οι πολλές εικονικές μηχανές που χρησιμοποιούν τους πόρους του είναι οι επισκέπτες (guests) ή αλλιώς τα φιλοξενούμενα μηχανήματα (hosted machines). Οι επισκέπτες, αντιμετωπίζουν τους υπολογιστικούς πόρους, όπως είναι η κεντρική μονάδα επεξεργασίας, η μνήμη και ο αποθηκευτικός χώρος, ως μια δεξαμενή πόρων που μπορεί εύκολα να ανακατανεμηθεί. Οι χειριστές (operators) μπορούν να ελέγχουν εικονικά στιγμιότυπα αυτών των πόρων, ούτως ώστε να πραγματοποιούν οριζόντια ή κάθετη κλιμάκωση. Δηλαδή είτε να δημιουργούν περισσότερες εικονικές μηχανές, είτε να αναδημιουργούν την ίδια με επιπλέον πόρους (εφόσον αυτοί δεν έχουν δεσμευτεί από άλλες εικονικές μηχανές του ίδιου φυσικού μηχανήματος) όταν αυτό είναι απαραίτητο.

[1] Απευθείας στην εικονικοποίηση υποβοηθούμενη από το υλικό και έμμεσα στην εικονικοποίηση υποβοηθούμενη από το λογισμικό.

Η εικονικοποίηση καθιστά δυνατή τη δημιουργία χρήσιμων υπηρεσιών ΤΠ (Τεχνολογίας Πληροφοριών) χρησιμοποιώντας πόρους στους οποίους παραδοσιακά μπορούσαμε να έχουμε πρόσβαση μονάχα με την ιδιοκτησία φυσικών μηχανημάτων. Μας επιτρέπει να αξιοποιήσουμε όλες τις δυνατότητες ενός φυσικού μηχανήματος διανέμοντάς τις σε πολλούς χρήστες και περιβάλλοντα. Με άλλα λόγια, υποστηρίζεται η πολλαπλή μίσθωση ανά φυσικό μηχάνημα με τη μορφή εικονικών μηχανών καθώς και η αυξημένη χρήση πόρων των φυσικών μηχανών (στα κέντρα δεδομένων του νέφους).

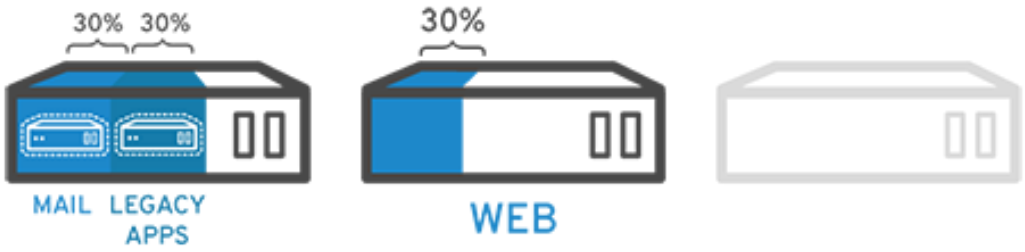
Με βάση ένα παράδειγμα της Red Hat [20], ας φανταστούμε πως έχουμε τρεις φυσικούς διακομιστές (servers) (δηλ. φυσικά μηχανήματα), στον καθένα από τους οποίους φιλοξενείται ένα συγκεκριμένο συστατικό (component) (λογισμικού): ένας διακομιστής ηλεκτρονικού ταχυδρομείου (email server), ένας διακομιστής ιστού (web server) και ένας διακομιστής που εκτελεί εσωτερικές εταιρικές εφαρμογές κληρονομιάς (οι οποίες σταμάτησαν να διατηρούνται αλλά είναι ακόμα λειτουργικές), αντίστοιχα. Στο συγκεκριμένο παράδειγμα κάθε ένας από τους (φυσικούς) διακομιστές χρησιμοποιεί μονάχα το 30% των δυνατοτήτων του (ως προς τους πόρους που μπορεί να διαθέσει).



Σχήμα 2.10: Χρήση διακομιστών χωρίς εικονικοποίηση [20]

Παραδοσιακά, αυτή η αρχιτεκτονική, όπου εκτελούνται μεμονωμένες εργασίες έκαστη αποκλειστικά σε συγκεκριμένο διακομιστή, ήταν ευκολότερη και πιο αξιόπιστη αλλά δεν παύει να είναι και η λιγότερο αποδοτική λύση. Με την άφιξη της τεχνολογίας της εικονικοποίησης όμως, είναι πλέον εφικτό να χωριστεί ένας διακομιστής σε περισσότερα μέρη, έχοντας δύο ή παραπάνω εικονικές μηχανές με τη χρήση μιας φυσικής.

Εφόσον ένας διακομιστής χωρίζεται σε δύο ή παραπάνω εικονικά μέρη, μπορεί να αυξηθεί ραγδαία η αξιοποίηση των δυνατοτήτων του. Με βάση το προηγούμενο παράδειγμα, αν μια εικονική μηχανή λαμβάνει το 30% των πόρων ενός διακομιστή/φυσικού μηχανήματος, τότε ορισμένες από τις προαναφερόμενες λειτουργικότητες/συστατικά λογισμικού (παροχής υπηρεσιών ιστού, ηλεκτρονικού ταχυδρομείου και εφαρμογών) θα μπορούσαν να εγκατασταθούν στον ίδιο διακομιστή με την μορφή εικονικών μηχανών.



Σχήμα 2.11: Χρήση διακομιστών με εικονικοποίηση [20]

Αφού η δημιουργία και καταστροφή των εικονικών μηχανών σε ένα μηχάνημα πραγματοποιείται δυναμικά ανάλογα με τη ζήτηση, αυτό σημαίνει πως ένας διακομιστής μπορεί να συνεχίσει να χρησιμοποιείται για νέους σκοπούς σε σχέση με τους αρχικούς ή να αποσυρθεί τελείως σταματώντας την λειτουργία του (switch off). Το τελευταίο είναι χρήσιμο κυρίως όταν η ζήτηση σε ένα κέντρο δεδομένων είναι μικρή και επομένως υπάρχει οικονομικό συμφέρον (λόγω ενεργειακού κόστους) ως προς το κλείσιμο των διακομιστών που δεν απαιτούνται για την κάλυψη της παρούσας ζήτησης.

2.2.4.1 Παρα-εικονικοποίηση

Όταν αναφερόμαστε στην εικονικοποίηση συνήθως μιλάμε για την πιο συνηθισμένη μορφή της, η οποία είναι η πλήρης εικονικοποίηση. Με την πάροδο του χρόνου και την αύξηση της δημοτικότητας της εικονικοποίησης, αναπτύχθηκαν πολλοί υπερ-επόπτες που μπορεί να διαφέρουν όχι μόνο στα χαρακτηριστικά τους αλλά και στις διάφορες τεχνικές που χρησιμοποιούν για να κάνουν την εικονικοποίηση πραγματικότητα. Μια από αυτές ονομάζεται παρα-εικονικοποίηση (paravirtualization) [40].

Κάθε εικονική μηχανή απαιτεί ένα συγκεκριμένο ποσοστό χρήσης υπολογιστικών πόρων του φυσικού διακομιστή για να εκτελείται. Μέσα σε αυτό το ποσοστό, συμπεριλαμβάνεται και η επιβάρυνση που εισάγει η συνεχής μετάφραση εικονικών σε φυσικούς πόρους. Έχοντας υπόψιν πως κάθε φυσικός διακομιστής διαθέτει πεπερασμένους πόρους, με την πλήρη εικονικοποίηση επιβάλλεται ένα όριο στον αριθμό των εικονικών μηχανών που μπορεί αυτός να υποστηρίξει. Επιπροσθέτως, προκύπτουν και προβλήματα καθυστέρησης αφού η διαδικασία της μετάφρασης απαιτεί κάποιο χρόνο για να διεκπεραιωθεί.

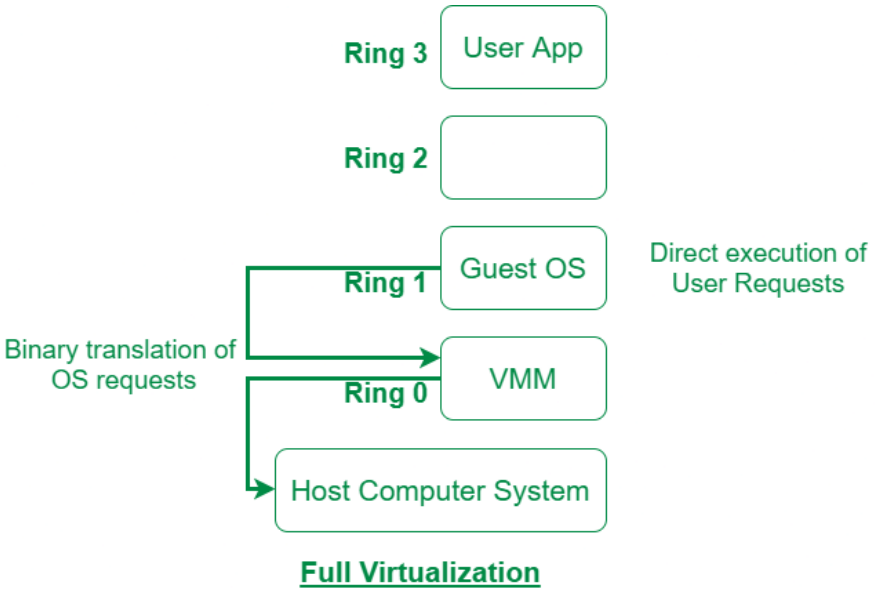
Επομένως, η πλήρης εικονικοποίηση περιορίζει σημαντικά τον αριθμό των εικονικών μηχανών που δύναται ένας διακομιστής να εκτελέσει παράλληλα, όπως επίσης και τα είδη εφαρμογών που μπορούν να εκτελεστούν σε μια εικονική μηχανή (εφόσον η ταχύτητα εκτέλεσης τους παίζει σημαντικό ρόλο στην ευχρηστία τους).

Η παρα-εικονικοποίηση είναι μια τεχνική εικονικοποίησης που αναπτύχθηκε προκειμένου να ξεπεραστούν τα προαναφερόμενα προβλήματα επιδόσεων που έρχονται με την χρήση της πλήρους εικονικοποίησης. Κατά την χρήση της, το φιλοξενούμενο λειτουργικό σύστημα δεν είναι πλήρως απομονωμένο από το υλικό αλλά απομονώνεται μερικώς [41] και έχει άμεση επικοινωνία με αυτό. Υπάρχει δηλαδή επίγνωση της εικονικοποίησης από μεριάς του ΛΣ των εικονικών μηχανών και αξιοποίηση της, μέσω της χρήσης υπερ-κλήσεων προς τον υπερ-επόπτη. Οι κλήσεις αυτές, επιτρέπουν σε κάθε ΛΣ να ζητάει πόρους κατά παραγγελία και να αναθέτει την εκτέλεση διεργασιών του απευθείας στο υλικό. Για να μπορέσει να επιτευχθεί αυτό, απαιτείται η τροποποίηση του φιλοξενούμενου λειτουργικού συστήματος, η οποία θα του επιτρέπει την υλοποίηση ενός ειδικού API, ώστε να μπορεί να κάνει χρήση των υπερ-κλήσεων [42], ενώ επιβάλλεται να υποστηρίζεται και από τον υπερ-επόπτη η κατανόησή τους.

Η παρα-εικονικοποίηση παρέχει ορισμένα πλεονεκτήματα συγκριτικά με την πλήρη εικονικοποίηση. Καταρχάς, η απόδοση των εικονικών μηχανών είναι συγκρίσιμη με αυτή των φυσικών μηχανών, αφού η παρα-εικονικοποίηση δεν επιβάλλει την επιβάρυνση της μετάφρασης εικονικών σε φυσικούς πόρους. Επιπροσθέτως, το γεγονός πως το φιλοξενούμενο λειτουργικό σύστημα γνωρίζει πως εκτελείται σε εικονικό περιβάλλον, του δίνει την δυνατότητα να αποφύγει την χρήση περιττών προγραμμάτων που μπορεί να αποτελέσουν ευπαθή σημεία για επιθέσεις, όπως για παράδειγμα το BIOS. Επομένως, η χρήση της παρα-εικονικοποίσης παρέχει μεγαλύτερη ασφάλεια στο σύστημα. Μερικοί τομείς που επωφελούνται από την παρα-εικονικοποίηση είναι η χρήση λογισμικών που επιτρέπουν την ανάκαμψη από καταστροφές, την μετανάστευση δεδομένων μεταξύ λειτουργικών συστημάτων [43] ή ακόμα και λογισμικά ενσωματωμένων συστημάτων αυτοκινήτων [44].

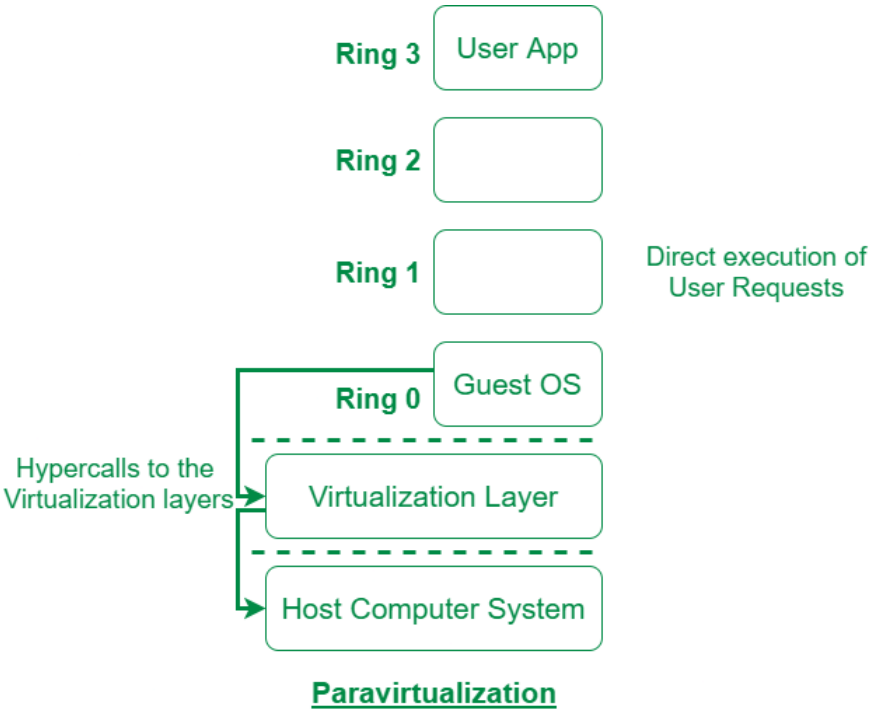
Παρ' όλα τα πλεονεκτήματα που παρέχει η χρήση της παρα-εικονικοποίησης, υπάρχουν και μερικά ζητήματα που πρέπει να ληφθούν υπόψιν. Εξαιτίας της ανάγκης τροποποίησης του λειτουργικού συστήματος των φιλοξενούμενων λειτουργικών συστημάτων, κάτι που δεν είναι πάντοτε εφικτό, αυτά καθίστανται λιγότερο φορητά σε σχέση με την πλήρη εικονικοποίηση, αφού τροποποιούνται για την υποστήριξη συγκεκριμένου υλικού αντί ενός υπερ-επόπτη [44]. Ταυτοχρόνως, η στενή εξάρτηση μεταξύ του υπερ-επόπτη και των φιλοξενούμενων ΛΣ μπορεί να διακοπεί από τις ενημερώσεις του λειτουργικού συστήματος [43], κάνοντας την παρα-εικονικοποίηση λιγότερο αξιόπιστη.

Στο Σχήμα 2.12 [45] παρουσιάζεται η αρχιτεκτονική της πλήρους εικονικοποίησης όπου το φιλοξενούμενο λειτουργικό σύστημα (της εικονικής μηχανής) επιβάλλεται να περάσει τα αιτήματά του (πρόσβασης πόρων) μέσω του υπερ-επόπτη.



Σχήμα 2.12: Πλήρης εικονικοποίηση [45]

Αντιθέτως, στο Σχήμα 2.13 όπου και φαίνεται η αρχιτεκτονική της παρα-εικονικοποίησης, βλέπουμε πως μέσω των υπερ-κλήσεων, όλα τα αιτήματα προορίζονται στη στρώση εικονικοποίησης και από εκεί στο κύριο σύστημα (χωρίς την ανάγκη κάποιας επεξεργασίας ή μετάφρασης).



Σχήμα 2.13: Παρα-εικονικοποίηση [45]

Οι διαφορές της πλήρους εικονικοποίησης με την παρα-εικονικοποίηση με βάση τον οργανισμό GeeksforGeeks [45] είναι οι εξής:

Πίνακας 2.1: Διαφορές πλήρους εικονικοποίησης και παρα-εικονικοποίησης

Κριτήρια	Πλήρης εικονικοποίηση	Παρα-εικονικοποίηση
Βαθμός απομόνωσης	Πλήρης απομόνωση της εικονικής μηχανής.	Μερική απομόνωση και χρήση API για αμεσότερη επικοινωνία.
Επίπεδο ασφάλειας	Λιγότερο ασφαλής.	Υπάρχει επίγνωση του εικονικού περιβάλλοντος και δε γίνεται εκκίνηση του BIOS [46], πράγμα που την καθιστά ασφαλέστερη.
Τρόπος λειτουργίας	Χρήση δυαδικής μετάφρασης[2].	Χρήση υπερ-κλήσεων κατά την εκτέλεση.
Απόδοση	Πιο αργές ταχύτητες[3] [47].	Γρηγορότερη εκτέλεση.
Μεταφερσιμότητα	Μεγαλύτερη συμβατότητα και μεταφερσιμότητα.	Λόγω της αρχιτεκτονικής της είναι δυσκολότερη η μεταφορά εικονικών μηχανών (από έναν φυσικό διακομιστή σε έναν άλλο)
Ευκολία χρήσης	Υποστήριξη όλων των συστημάτων χωρίς την απαίτηση τροποποιήσεων.	Απαιτείται τροποποίηση του φιλοξενούμενου λειτουργικού συστήματος για να κάνει χρήση υπερ-κλήσεων.

2.2.5 Πλεονεκτήματα της εικονικοποίησης

Η εικονικοποίηση προσφέρει πολλά πλεονεκτήματα στις επιχειρήσεις. Τα πιο αξιοσημείωτα αυτών με βάση την IBM [48] είναι τα εξής:

[2]Αυτό ισχύει στην περίπτωση εικονικοποίησης υποβοηθούμενη από το λογισμικό.

[3]Με βάση την VMware, η απόδοση της παρα-εικονικοποίησης είναι καλύτερη υπό ορισμένες περιπτώσεις, ενώ η πλήρης εικονικοποίηση με χρήση δυαδικής μετάφρασης παρέχει καλύτερη απόδοση από την πρώτη γενιά εικονικοποίησης υποβοηθούμενη από το υλικό.

- **Αποδοτικότητα πόρων**:

  Η χρήση εικονικοποίησης συνεπάγεται την μείωση του αριθμού των φυσικών μηχανημάτων που απαιτούνται για την εκτέλεση των εφαρμογών. Αυτό συμβαίνει διότι εφόσον ένας φυσικός διακομιστής μπορεί να φιλοξενήσει πολλαπλές εικονικές μηχανές, αυτές με την σειρά τους δύναται να αντικαταστήσουν άλλους φυσικούς διακομιστές που θα ήταν απαραίτητοι για την εκτέλεση διαφορετικών λειτουργιών [49]. Με αυτόν τον τρόπο, εξοικονομείται η ενέργεια που θα απαιτούσε η διάθεση των (έξτρα) υπολογιστικών πόρων των φυσικών μηχανημάτων που αποσύρθηκαν, ενώ ταυτόχρονα αξιοποιούνται σε μεγαλύτερο βαθμό οι υπολογιστικοί πόροι του μηχανήματος που φιλοξενεί τις εικονικές μηχανές.

- **Ευκολότερη διαχείριση**:

  Αντικαθιστώντας φυσικούς υπολογιστές με προγραμματιστικά καθορισμένες εικονικές μηχανές δύναται η χρήση αυτοματοποιημένων ροών διαχειριστικών εργασιών. Οι διαχειριστές συστημάτων μπορούν να χρησιμοποιούν εργαλεία για τον καθορισμό εικονικών μηχανών χρησιμοποιώντας πρότυπα κατάλληλα για την υποδομή κάθε επιχείρησης. Με αυτόν τον τρόπο, η εγκατάσταση και η ρύθμιση των εικονικών μηχανών μπορεί να γίνεται επανειλημμένα με αυτοματοποιημένο τρόπο δίχως το ρίσκο ανθρώπινου λάθους και γλιτώνοντας τον χρόνο εγκατάστασης και ρύθμισής τους χειροκίνητα. Ένας συνδυασμός εργαλείων που κάνει αυτή τη διαδικασία πραγματικότητα είναι τα Ansible[4] και Terraform[5].

- **Ελάχιστος χρόνος διακοπής λειτουργίας**:

  Οι καταρρεύσεις λειτουργικών συστημάτων και εφαρμογών μπορεί να προκαλέσουν διακοπή λειτουργίας και να διαταράξουν την παραγωγικότητα των χρηστών. Οι διαχειριστές έχουν την δυνατότητα εκτέλεσης πλεοναζουσών εικονικών μηχανών, με σκοπό την ταχεία εναλλαγή σε αυτές στην περίπτωση που προκύψουν προβλήματα στις αρχικές (που έχουν διατεθεί). Κάτι τέτοιο, δεν θα ήταν αποδοτικό για την επιχείρηση διαθέτοντας αποκλειστικά μη εικονικοποιημένα φυσικά μηχανήματα.

- **Ταχύτερη παροχή**:

  Η αγορά, εγκατάσταση και διαμόρφωση του υλικού για κάθε εφαρμογή είναι χρονοβόρα. Εφόσον το υλικό είναι ήδη στη θέση του και μπορεί να εντοπίζεται και απομακρυσμένα (όπως στην περίπτωση περιβαλλόντων νέφους), η παροχή εικονικών μηχανών για την εγκατάσταση και εκτέλεση όλων των εφαρμογών είναι σημαντικά ταχύτερη.

[4]R. Hat. Ansible. URL: `https://www.ansible.com/`.
[5]HashiCorp. Terraform. URL: `https://www.terraform.io/`.

2.3 Ασφάλεια στην εικονικοποίηση

Η χρήση της εικονικοποίησης παρέχει αρκετά εγγενή οφέλη ασφαλείας με την μορφή μέτρων ανάκαμψης από επιθέσεις. Ένα σημαντικό εξ αυτών, είναι η ικανότητα επαναφοράς εικονικών μηχανών που έχουν μολυνθεί με κακόβουλο λογισμικό σε μια χρονική περίοδο πριν τη μόλυνση τους. Αυτό επιτυγχάνεται μέσω της δυνατότητας δημιουργίας στιγμιοτύπων εικονικών μηχανών [52], η οποία παρέχεται από τον υπερ-επόπτη. Επιπρόσθετα, μπορεί εύκολα να πραγματοποιηθεί διαγραφή και αναδημιουργία τους με έναν αυτοματοποιημένο τρόπο, σε περίπτωση που η επαναφορά σε προηγούμενη χρονική περίοδο για οποιονδήποτε λόγο δεν είναι εφικτή. Αυτή η λύση μπορεί να εφαρμοστεί χωρίς αρνητικές επιπτώσεις εάν τα δεδομένα της εικονικής μηχανής είτε δεν μας ενδιαφέρουν, είτε βρίσκονται σε διαφορετική τοποθεσία επειδή γίνεται χρήση εικονικοποίησης αποθήκευσης, είτε υπάρχουν αντίγραφα ασφαλείας τους. Αυτές οι λειτουργίες είναι αρκετές φορές αδύνατο να εφαρμοστούν σε ένα φυσικό μηχάνημα διότι το κακόβουλο λογισμικό συχνά μπορεί να είναι βαθιά ριζωμένο στα βασικά συστατικά του συστήματος [48]. Επιπλέον, ακόμα και αν το κακόβουλο λογισμικό ήταν σε θέση να εξαφανιστεί με μια επαναφορά, κάτι τέτοιο θα χρειαζόταν σημαντικά περισσότερο χρόνο για να διεκπεραιωθεί σε ένα φυσικό μηχάνημα συγκριτικά με μια εικονική μηχανή.

Παρ' όλα αυτά, η εικονικοποίηση δεν είναι απαλλαγμένη από κινδύνους καθώς παραβιάζοντας τον υπερ-επόπτη, ένας επιτιθέμενος έχει πρόσβαση σε όλες τις εικονικές μηχανές που διαχειρίζονται μέσω αυτού. Επίσης, αυτό είναι δυσκολότερο να εντοπιστεί λόγω της ικανότητας του υπερ-επόπτη να επιτρέπει στις εικονικές μηχανές τη μεταξύ τους επικοινωνία χωρίς την αλληλεπίδραση με το φυσικό δίκτυο. Τέλος, η ακεραιότητα του συστήματος εξαρτάται άμεσα και από τον τύπο του υπερ-επόπτη. Αυτό συμβαίνει διότι ειδικότερα για υπερ-επόπτες τύπου 2, υπάρχει πολλές φορές η ανάγκη διαμοιρασμού δεδομένων μεταξύ των εικονικών μηχανών και του ΛΣ φιλοξενίας. Επομένως, η παραβίαση ενός υπερ-επόπτη τύπου 2 μπορεί δυνητικά να οδηγήσει στην εξάπλωση κακόβουλου λογισμικού και να κινδυνεύσει όλο το σύστημα [53].

2.3.1 Απειλές στην εικονικοποίηση

Όλες οι μορφές εικονικοποίησης είναι ευάλωτες σε επιθέσεις. Όπως αναφέρεται και στο [54] μέσω του [55], πολλές φορές δε δύναται ο υπερ-επόπτης, το λειτουργικό σύστημα ή ακόμα και η υπηρεσία ελέγχου πρόσβασης (Mandatory Access Control) του Linux να ανταπεξέλθουν στις απαιτήσεις ασφαλείας όλων των εφαρμογών. Το παράδειγμα που παρουσιάζεται στο [55] αναφέρεται στην εικονικοποίηση χώρου αποθήκευσης μέσω δικτύου αλλά πολλές από τις απειλές δεν περιορίζονται μονάχα εκεί.

Πολλές από τις απειλές που θα αναφερθούν παρακάτω στο 2.3.1.3, μπορούν να κατηγοριοποιηθούν και ως εξής:

Πίνακας 2.2: Πηγές απειλών στην εικονικοποίηση

Πηγή απειλής	Περιγραφή
Δίκτυο ⇒ Υπερ-επόπτη	Απειλές που προέρχονται από το δίκτυο και στοχεύουν τον υπερ-επόπτη.
Δίκτυο ⇒ Εικονική Μηχανή	Απειλές που προέρχονται από το δίκτυο και στοχεύουν εικονικές μηχανές.
Υπερ-επόπτη ⇒ Εικονική Μηχανή	Απειλές που προέρχονται από τον υπερ-επόπτη και στοχεύουν εικονικές μηχανές.
Εικονική μηχανή ⇒ Εικονική Μηχανή	Απειλές που προέρχονται από εικονικές μηχανές και στοχεύουν άλλες εικονικές μηχανές.
Εικονική μηχανή ⇒ Υπερ-επόπτη	Απειλές που προέρχονται από εικονικές μηχανές και στοχεύουν τον υπερ-επόπτη.
Διαχειριστή ⇒ Υπερ-επόπτη	Απειλές που προέρχονται από τον διαχειριστή εικονικών μηχανών και στοχεύουν τον υπερ-επόπτη.
Διαχειριστή ⇒ Εικονική Μηχανή	Απειλές που προέρχονται από τον διαχειριστή εικονικών μηχανών και στοχεύουν εικονικές μηχανές.

2.3.1.1 Απειλές για τον πάροχο νέφους μέσω δικτύου

Σήμερα όλο και περισσότερες επιχειρήσεις θα προτιμήσουν να βασιστούν σε έναν πάροχο νέφους για την απόκτηση υποδομών προκειμένου να εξυπηρετούν τους δυνητικούς πελάτες τους έναντι της παραδοσιακής διαδικασίας αγοράς, ρύθμισης και διαχείρισης φυσικών διακομιστών. Η ταχύτερη εκκίνηση παροχής υπηρεσιών, το μικρότερο κόστος και η ευκολία διαχείρισης της υποδομής τους, δεν αφήνουν περιθώρια αμφιβολίας της ορθότητας αυτής της απόφασης. Για να μπορούν όμως να παρέχουν τις υπηρεσίες τους στους τελικούς χρήστες, είναι απαραίτητη η μεταφορά δεδομένων από την επιχείρηση προς τον πάροχο νέφους, στις υποδομές του οποίου στεγάζονται οι εφαρμογές τους. Όπως αναφέραμε όμως στο 1.1, εισάγεται έτσι ένα αναγκαίο

μοντέλο εμπιστοσύνης ανάμεσα στον πάροχο νέφους και τις επιχειρήσεις. Δηλαδή, ταυτόχρονη εμπιστοσύνη ως προς την απουσία μη εξουσιοδοτημένης πρόσβασης στα δεδομένα των επιχειρήσεων από τον πάροχο και ως προς την ικανότητα του παρόχου να λάβει τα απαραίτητα μέτρα ασφαλείας για την προστασία τους από εξωτερικούς κακόβουλους χρήστες.

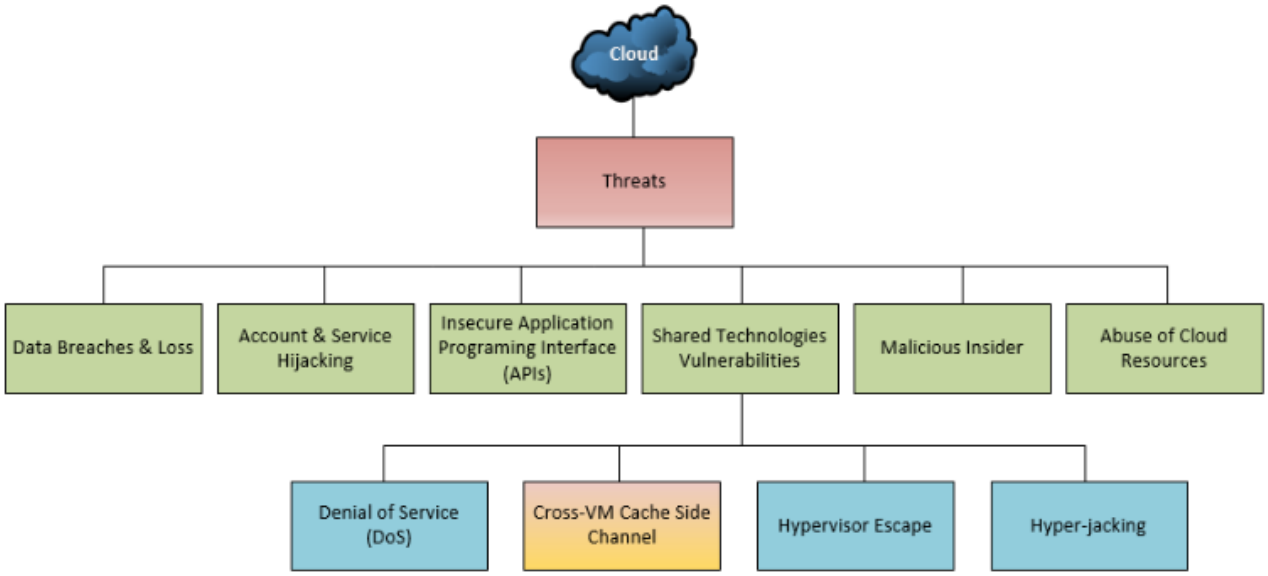
Ένα από τα σημαντικότερα ζητήματα που απασχολεί μια επιχείρηση είναι η ασφάλεια. Επιλέγοντας να χρησιμοποιήσουν τις υπηρεσίες ενός παρόχου νέφους, όμως, παραχωρούν ουσιαστικά πρόσβαση στον πάροχο αυτό στις εφαρμογές τους και στα ευαίσθητα δεδομένα αυτών των εφαρμογών, διότι η ευθύνη προστασίας των υποδομών ανήκει στον ιδιοκτήτη των υποδομών αυτών και στην προκειμένη περίπτωση αυτός είναι ο πάροχος νέφους. Έτσι, κακόβουλοι εισβολείς θα προσπαθήσουν να βρουν τρωτότητες στη διαδικασία παράδοσης των υπηρεσιών του παρόχου, στις υπηρεσίες τις ίδιες ή και στις διεπαφές με τις οποίες παρέχονται. Ένας συνηθισμένος τρόπος για να γίνει αυτό είναι εκτελώντας επιθέσεις τύπου Cross-site scripting (XSS), έγχυσης SQL (SQL injection), χειραγώγησης cookies ή εκμετάλλευσης μη ασφαλούς ρύθμισης, θέτοντας έτσι σε κίνδυνο ευαίσθητες πληροφορίες και δεδομένα των επιχειρήσεων. Επιπλέον, επειδή όλα τα δίκτυα είναι επιρρεπή σε επιθέσεις αν δεν έχουν ληφθεί τα κατάλληλα μέτρα προστασίας, ένας πάροχος πρέπει να μπορεί να προστατευτεί από επιθέσεις, όπως η κατασκοπεία (surveillance) και διείσδυση δικτύου (network penetration) [55].

2.3.1.2 Απειλές για τον πάροχο νέφους μέσω εικονικών μηχανών

Όπως μια επιχείρηση πρέπει να εμπιστεύεται τον πάροχο νέφους ότι θα προστατεύσει τα δεδομένα της, έτσι και ο πάροχος πρέπει να εμπιστεύεται την επιχείρηση ότι δε θα προσπαθήσει να προκαλέσει ζημιά στις υπηρεσίες του. Αυτό μπορεί να πραγματοποιηθεί με την εκτέλεση κακόβουλου λογισμικού στις εικονικές μηχανές του παρόχου είτε από την επιχείρηση την ίδια, είτε από έναν επιτιθέμενο που παραβίασε τις εικονικές μηχανές της.

Ουσιαστικά, κάθε εικονική μηχανή που έχει πρόσβαση στο διαδίκτυο είναι ευάλωτη σε απειλές όπως δούρειοι ίπποι (Trojans), ιοί και κακόβουλα λογισμικά που μπορεί να εξαπλωθούν στο σύστημα μεταπηδώντας από μια εικονική μηχανή στον υπερ-επόπτη και από εκεί είτε να συνεχίσουν στο σύστημα είτε να μολύνουν και τις υπόλοιπες εικονικές μηχανές που αυτός διαχειρίζεται. Επομένως, ο πάροχος απαιτείται να έχει λάβει τα κατάλληλα μέτρα προστασίας έναντι μολυσμένων ή κακόβουλων εικονικών μηχανών.

Με βάση την ανάλυση που έγινε στο [56], υπάρχουν διάφορες απειλές που εκμεταλλεύονται ευπάθειες των εικονικών μηχανών ή του υποκείμενου υπερ-επόπτη τους. Από αυτές, οι πιο συνηθισμένες είναι η επίθεση πλευρικού καναλιού (cross VM side channel attack), οι επιθέσεις βάσει χρονοπρογραμματιστή (Scheduler based attacks) και οι επιθέσεις που στοχεύουν τρωτότητες της διαδικασίας μετανάστευσης και επαναφοράς εικονικών μηχανών (VM migration and rollback attacks). Από την άλλη, όσον αφορά τις απειλές από εικονικές μηχανές προς τον υπερ-επόπτη, ανάμεσα στις πιο σημαντικές βρίσκονται η μεταπήδηση εικονικής μηχανής (VM Hopping) και η απόδραση εικονικής μηχανής (VM Escape). Τέλος, μερικές επιπρόσθετες απειλές σύμφωνα με τους Aalam, Kumar και Gour [57], είναι η κλοπή εικονικών μηχανών και η τροποποίηση του υπερ-επόπτη.



Σχήμα 2.14: Απειλές στην εικονικοποίηση [58]

Ακολουθεί συνοπτική επεξήγηση των παραπάνω απειλών:

- **Επίθεση πλευρικού καναλιού**:

  Κατά την επίθεση πλευρικού καναλιού, ο επιτιθέμενος, έχοντας πρόσβαση σε μια εικονική μηχανή, προσπαθεί να ανακτήσει πληροφορίες από άλλες εικονικές μηχανές του ίδιου φυσικού διακομιστή. Αυτό μπορεί να επιτευχθεί με την παρακολούθηση και ανάλυση της χρήσης των πόρων του φυσικού διακομιστή, όπως της μνήμης και του επεξεργαστή. Μέσα στις πληροφορίες που μπορεί να ανακτηθούν μπορεί να βρίσκονται και ιδιωτικά κλειδιά κρυπτογράφησης, τα οποία μπορούν αργότερα να χρησιμοποιηθούν για πραγματοποίηση επιθέσεων ενδιάμεσου και παρακολούθησης δικτύου, όπως αναφέρεται στο [58] μέσω του [59].

- **Επίθεση βάσει χρονοπρογραμματιστή**:

  Σε μια επίθεση βάσει χρονοπρογραμματιστή, ο επιτιθέμενος προσπαθεί να επηρεάσει τον χρονοπρογραμματιστή του υπερ-επόπτη προκειμένου να καταναλώσει περισσότερο χρόνο επεξεργασίας στον επεξεργαστή του φυσικού μηχανήματος, εισάγοντας διεργασίες με μεγαλύτερη προτεραιότητα. Έτσι, μπορεί να επιτύχει την αποκλειστική χρήση του επεξεργαστή και να προκαλέσει απώλεια απόδοσης στις υπόλοιπες εικονικές μηχανές.

- **Επιθέσεις κατά της διαδικασίας μετανάστευσης και επαναφοράς**:

  Στις επιθέσεις που στοχεύουν τη διαδικασία μετανάστευσης και επαναφοράς εικονικών μηχανών, ο επιτιθέμενος προσπαθεί να επηρεάσει την εμπιστευτικότητα των δεδομένων μιας εικονικής μηχανής. Ανά πάσα στιγμή, ένας υπερ-επόπτης μπορεί να δημιουργεί στιγμιότυπα της κατάστασης μιας εικονικής μηχανής προκειμένου να συνεχίσει την εκτέλεση σε αυτά, χωρίς να το γνωρίζει ο χρήστης. Έτσι, δεδομένου ότι ένα κομμάτι του ιστορικού εκτέλεσής της χάνεται, ο επιτιθέμενος προσπαθεί να εκμεταλλευτεί αυτό το γεγονός για να αποκτήσει πρόσβαση σε ευαίσθητες πληροφορίες μιας εικονικής μηχανής ενός παραβιασμένου υπερ-επόπτη (η οποία δεν εκτελείται επί της παρούσης). Μπορεί για παράδειγμα να παρακάμψει κάποιους ελέγχους ασφαλείας που κανονικά θα ήταν προγραμματισμένοι για την εκάστοτε εικονική μηχανή, ή ακόμα και να αναιρέσει ορισμένες ενημερώσεις ασφαλείας της [60].

- **Μεταπήδηση εικονικής μηχανής**:

  Η μεταπήδηση εικονικής μηχανής, είναι μια μέθοδος επίθεσης κατά την οποία ένας επιτιθέμενος, έχοντας παραβιάσει μια εικονική μηχανή μπορεί να μεταπηδήσει από εκείνη σε μια άλλη [61]. Αυτό ευνοείται από το γεγονός ότι πολλές φορές, οι εικονικές μηχανές συνυπάρχουν με άλλες στον ίδιο υπολογιστή. Έτσι, ένας επιτιθέμενος μπορεί να εκμεταλλευτεί αδυναμίες στην δυνατότητα του υπερ-επόπτη να επιτρέπει την πρόσβαση σε μια εικονική μηχανή από μια άλλη. Η επίθεση αυτή βασίζεται στην γνώση πληροφοριών, όπως η διεύθυνση IP μιας εικονικής μηχανής. Έπειτα, αφού μια εικονική μηχανή παραβιαστεί, δύναται ο επιτιθέμενος να διαχειριστεί την ροή της δικτυακής κυκλοφορίας της, να δημιουργήσει άρνηση υπηρεσιών και να τροποποιήσει τα αρχεία της, αλλοιώνοντας το αρχείο διαμόρφωσής της.

- **Απόδραση εικονικής μηχανής**:

  Στις επιθέσεις απόδρασης εικονικής μηχανής, ένας επιτιθέμενος επιχειρεί να χρησιμοποιήσει κακόβουλες εφαρμογές για να αποκτήσει πρόσβαση στον υπερ-επόπτη. Αυτό μπορεί να επιτευχθεί με την εκμετάλλευση αδυναμιών της απομόνωσης που προσφέρει ο υπερ-επόπτης στις εικονικές μηχανές και ως αποτέλεσμα, να μπορέσει να ξεφύγει από τα όρια της εικονικής μηχανής και να επηρεάσει όχι μόνο τις υπόλοιπες εικονικές μηχανές αλλά και το κύριο σύστημα στο οποίο αυτές στεγάζονται [62].

- **Κλοπή εικονικής μηχανής**:

  Η κλοπή εικονικών μηχανών επιτυγχάνεται μετά από παραβίαση του υπερ-επόπτη. Ένα γεγονός που μπορεί να πραγματοποιηθεί δίχως την ανάγκη φυσικής πρόσβασης στο υποκείμενο μηχάνημα. Τα δεδομένα μιας εικονικής μηχανής αποθηκεύονται σε έναν εικονικό δίσκο με την μορφή μιας εικόνας μηχανής. Αυτό συμβαίνει για να μπορεί ένας υπερ-επόπτης να δημιουργεί αντίγραφα ασφαλείας των εικονικών μηχανών και να μπορούν αυτά έπειτα να μεταφερθούν σε ένα διαφορετικό φυσικό μηχάνημα. Ένας επιτιθέμενος, έχοντας πρόσβαση στον υπερ-επόπτη, μπορεί να αντιγράψει αυτές τις εικόνες εικονικών μηχανών στο δικό του μηχάνημα μέσω του διαδικτύου και έχοντας αρκετό χρόνο στην διάθεσή του, να καταφέρει αργότερα να ανακτήσει ευαίσθητες πληροφορίες μέσα από αυτές.

- **Τροποποίηση του υπερ-επόπτη**:

  Η τροποποίηση του υπερ-επόπτη πραγματοποιείται με την χρήση rootkit που βασίζονται στις εικονικές μηχανές (VMBR - Virtual Machine Based Rootkits). Πρόκειται για ένα κακόβουλο λογισμικό, το οποίο δύσκολα ανιχνεύεται. Με την χρήση ενός VMBR μπορεί ένας κακόβουλος χρήστης να παραβιάσει το φιλοξενούμενο ΛΣ της εικονικής μηχανής και έπειτα να συνεχίσει στην παραβίαση του υπερ-επόπτη ή να παραβιάσει κατευθείαν τον υπερ-επόπτη με σκοπό να αποκτήσει απόλυτο έλεγχο του συστήματος στο οποίο αυτός εκτελείται.

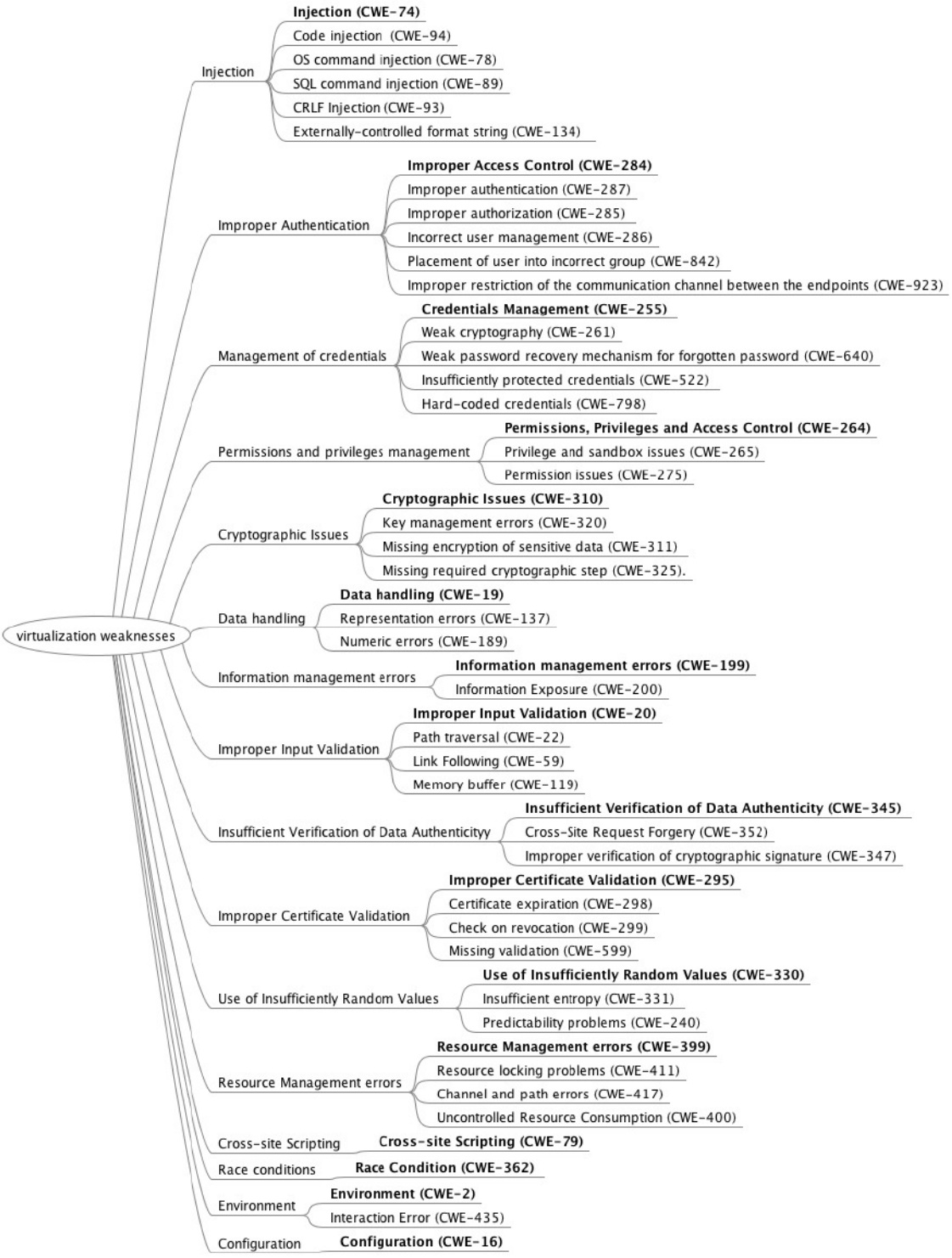
2.3.1.3 Κατηγοριοποίηση απειλών στην εικονικοποίηση

Οι προαναφερόμενες απειλές σε συνδυασμό με μερικές πιθανές ευπάθειες, έχουν κατηγοριοποιηθεί από τους Sane, Niang και Fall στο [56] ως εξής:

Πίνακας 2.3: Κατηγοριοποίηση απειλών και στην εικονικοποίηση

Κατηγορία	Απειλές/Επιθέσεις και πιθανά ευπαθή σημεία
Απειλές/Επιθέσεις Δικτύου	XML Signature Wrapping Attacks
	Flooding Attacks (DDoS)
	Metadata Spoofing Attacks
	Insecure Web Apps & APIs
	Cross Site Scripting Attacks (XSS)
	Port Scanning
	Botnets
	Spoofing Attacks
	DNS Attacks
	Sniffer Attacks
	Denial of Service (DoS)
Απειλές/Επιθέσεις για τον Πάροχο	Sniffing Attacks
	Spoofing Attacks
	Denial of Service (DoS)
	Cross VM Side Channel Attacks
	VM Scheduler Based Attacks
	VM Hopping
	VM Escape
	Dynamic Data Updates
	Data Scavenging
Απειλές/Επιθέσεις για τις Εφαρμογές	Malware Injection
	Steganography Attacks
	Web Services Attacks
	Protocol Based Attacks
	Security Misconfigurations
	SQL Injection Attacks

Μια πιο εκτενής κατηγοριοποίηση για την κατηγορία των απειλών σε εφαρμογές, πραγματοποιήθηκε από τον οργανισμό ENISA στο [63] όπου και απεικονίζεται στο ακόλουθο σχήμα:



Σχήμα 2.15: Πιθανά σημεία εμφάνισης τρωτοτήτων και οι απειλές που τους αντιστοιχούν [63]

2.3.2 Η τριάδα της ασφάλειας

Εν γένει, ο λόγος που μια επιχείρηση ενδιαφέρεται για την ασφάλεια, είναι προκειμένου να διασφαλίσει την ακεραιότητα, την εμπιστευτικότητα και τη διαθεσιμότητα των περιουσιακών στοιχείων (assets) που μεταφέρει στο νέφος. Αυτά τα τρία στοιχεία αποτελούν την τριάδα της ασφάλειας [64] και η απώλεια οποιουδήποτε από αυτά μπορεί να έχει σοβαρές επιπτώσεις για την επιχείρηση. Η σημασία του καθενός, καθώς και γενικές ορθές πρακτικές διατήρησής τους περιγράφονται ως εξής:

- **Ακεραιότητα δεδομένων**

  Η ακεραιότητα των δεδομένων είναι ένα από τα τρία βασικά στοιχεία της ασφάλειας, δίχως την οποία οι επιχειρήσεις δε θα μπορούσαν να παραμείνουν λειτουργικές. Αναφέρεται στην προστασία των δεδομένων από μη εξουσιοδοτημένη αλλοίωση καθ' όλη τη διάρκεια της ύπαρξής τους, είτε αυτά βρίσκονται στο στάδιο της μεταφοράς, την επεξεργασίας ή της αποθήκευσης. Για κάθε επιχείρηση, απαιτείται μεγάλη προσοχή κατά τον σχεδιασμό των βάσεων δεδομένων και της συντήρησής τους σε περιβάλλοντα νέφους αλλά και η χρήση ορθών πρακτικών, όπως ο περιοδικός έλεγχος των δεδομένων για την ανίχνευση πιθανών αλλοιώσεων και η χρήση μηχανισμών αναγνώρισης και αποκατάστασης σφαλμάτων. Επιπρόσθετα, απαιτείται να υπάρχει και κατάλληλος έλεγχος πρόσβασης, αφού η πρόσβαση από μη εξουσιοδοτημένους φορείς αποτελεί απειλή για την ακεραιότητα των δεδομένων.

- **Εμπιστευτικότητα δεδομένων**

  Όπως η ακεραιότητα, έτσι και η εμπιστευτικότητα των δεδομένων είναι κάτι για το οποίο μια επιχείρηση πρέπει να φροντίσει εκ των προτέρων. Αναφέρεται στην προστασία των δεδομένων από μη εξουσιοδοτημένη πρόσβαση κατά τη μεταφορά, την αποθήκευση και την επεξεργασία τους. Ένα κενό ασφαλείας στην τελική εφαρμογή ενός χρήστη μπορεί να δώσει πρόσβαση σε έναν εισβολέα παραβιάζοντας έτσι τις βάσεις δεδομένων της επιχείρησης. Επειδή σε ένα περιβάλλον νέφους δεν έχει η ίδια η επιχείρηση πρόσβαση στις υποδομές που χρησιμοποιεί, πέρα από μέτρα προστασίας που θα πρέπει να λάβει η ίδια, όπως η διασφάλιση ύπαρξης ρυθμίσεων ασφαλείας και μηχανισμών περιορισμού πρόσβασης στα δεδομένα της, είναι απαραίτητη και η επικοινωνία με τον πάροχο νέφους προκειμένου να διαπιστωθεί ο τρόπος εξασφάλισης της εμπιστευτικότητας των δεδομένων από μεριάς του.

- **Διαθεσιμότητα δεδομένων**

  Η διαθεσιμότητα των δεδομένων, που ολοκληρώνει την τριάδα της ασφάλειας, είναι το στοιχείο που εξασφαλίζει πως μια επιχείρηση θα μπορεί να παρέχει τις υπηρεσίες της στους τελικούς της χρήστες. Αναφέρεται στην αποφυγή της διακοπής πρόσβασης στα δεδομένα της επιχείρησης από εξουσιοδοτημένους φορείς και εξαρτάται άμεσα από τη συνεχή παροχή υπηρεσιών υποδομών προς την επιχείρηση. Η απώλεια της διαθεσιμότητας θα είχε ως αποτέλεσμα την διακοπή σημαντικών λειτουργιών της επιχείρησης και δυνητικά την μείωση της αξιοπιστίας της. Για να μπορέσει να διασφαλιστεί, πρέπει μια επιχείρηση να έχει προβλέψει για ένα σχέδιο ανάκτησης εφεδρικών αντιγράφων προς αποφυγή της απώλειας σημαντικών δεδομένων της, καθώς και για ένα σχέδιο επαναφοράς των διαδικασιών παροχής τους ώστε να μειώσει στο ελάχιστο την οποιαδήποτε διάρκεια διακοπής των υπηρεσιών της. Τέλος, πρέπει να υπάρχει εμπιστοσύνη προς τον πάροχο νέφους πως δεν θα υπάρξει από μεριάς του απρόσμενη διακοπή της λειτουργίας υποδομών που μπορεί να είναι απαραίτητες για την επιχείρηση.

2.3.3 Μέτρα ασφαλείας

Προκειμένου να προστατευτεί μια επιχείρηση από τις απειλές που αναφέρθηκαν παραπάνω, θα πρέπει αυτή να έχει λάβει τα κατάλληλα μέτρα ασφαλείας. Μερικές ορθές πρακτικές με βάση τους οργανισμούς GeeksforGeeks [65] και ENISA [63] είναι οι παρακάτω:

- **Συχνή ενημέρωση του υπερ-επόπτη**:

  Ο υπερ-επόπτης είναι ο πυρήνας του συστήματος εικονικοποίησης και επομένως η ασφάλειά του είναι ζωτικής σημασίας. Οι εταιρείες που αναπτύσσουν το λογισμικό του, τον ενημερώνουν συχνά για να διορθώσουν τυχόν ευπάθειες που έχουν ανακαλυφθεί. Επομένως, οι επιχειρήσεις πρέπει να εφαρμόζουν τις ενημερώσεις αυτές το συντομότερο δυνατόν από την στιγμή που θα είναι διαθέσιμες.

- **Περιορισμός πρόσβασης στο διαχειριστικό πάνελ του υπερ-επόπτη**:

  Η πρόσβαση στον υπερ-επόπτη παρέχει πλήρη έλεγχο στις λειτουργίες του και επομένως, πρέπει να περιορίζεται μόνο σε ένα πολύ μικρό αριθμό από εξουσιοδοτημένα άτομα. Επιπλέον, οι επιχειρήσεις πρέπει να επιβάλλουν την χρήση πολύπλοκων κωδικών πρόσβασης και να τους αλλάζουν τακτικά, καθώς και την χρήση αυθεντικοποίησης πολλαπλών παραγόντων.

- **Έλεγχος των μηχανημάτων που έχουν πρόσβαση στον υπερ-επόπτη**:

  Οι επιχειρήσεις πρέπει να ελέγχουν συχνά ποια μηχανήματα έχουν πρόσβαση στον υπερ-επόπτη και να προσθέτουν ή να αφαιρούν μηχανήματα από την σχετική λίστα εξουσιοδότησης. Ο συχνός έλεγχος πρόσβασης θα μπορέσει επίσης να αναδείξει προσπάθειες μη εξουσιοδοτημένης πρόσβασης ώστε να ληφθούν τα κατάλληλα μέτρα περιορισμού των ατόμων που επιχειρούν να εισέλθουν στον υπερ-επόπτη.

- **Περιορισμός δικτυακής πρόσβασης στο διαχειριστικό πάνελ του υπερ-επόπτη**:

  Η πρόσβαση στο διαχειριστικό πάνελ του υπερ-επόπτη πρέπει να πραγματοποιείται μόνο από ασφαλή δίκτυα. Επομένως, για την επιβολή περιορισμών θα πρέπει να γίνεται χρήση αναχωμάτων ασφαλείας και άλλων εργαλείων ασφαλείας του δικτύου.

- **Χρήση κρυπτογράφησης εικονικών μηχανών**:

  Η κρυπτογράφηση των εικονικών μηχανών προστατεύει τα δεδομένα τους από μη εξουσιοδοτημένη πρόσβαση. Επιπλέον, στην περίπτωση κλοπής εικονικής μηχανής μετά από παραβίαση του υπερ-επόπτη, ο επιτιθέμενος δεν θα είναι σε θέση να αποκτήσει πρόσβαση στα δεδομένα της.

- **Απομόνωση των εικονικών μηχανών μεταξύ τους**:

  Ένας υπερ-επόπτης πρέπει να επιτρέπει την αλληλεπίδραση μεταξύ εικονικών μηχανών μόνο όταν αυτό είναι απαραίτητο (παραδείγματος χάριν, για τον διαμοιρασμό αποθηκευτικού χώρου). Επιβάλλεται να εφαρμοστούν πολιτικές που να διαχειρίζονται την φυσική και λογική κατάτμηση πόρων. Αυτό θα αποτρέψει την μη εξουσιοδοτημένη πρόσβαση, θα μειώσει τις επιθέσεις έγχυσης κώδικα από μια εικονική μηχανή σε μια άλλη, τις επιθέσεις πλευρικού καναλιού, καθώς και το ρίσκο επίθεσης τύπου άρνησης υπηρεσίας.

- **Παρακολούθηση των πόρων**:

  Η δικτυακή κίνηση, η μνήμη και οι διεργασίες των εκτελούμενων εικονικών μηχανών πρέπει να παρακολουθούνται διαρκώς προκειμένου να ξεχωρίζουν οι εικονικές μηχανές που παρουσιάζουν ασυνήθιστες συμπεριφορές. Αυτό θα βοηθήσει στην ορθότερη εκτέλεση υπαρχόντων μηχανισμών ασφαλείας και ανίχνευσης εισβολών.

- **Ορθή διαχείριση στιγμιοτύπων εικονικών μηχανών**:

  Κάθε στιγμιότυπο μιας εικονικής μηχανής, δύναται να περιέχει ευαίσθητα δεδομένα, όπως κωδικοί και προσωπικά δεδομένα χρηστών. Συνεπώς, πρέπει αυτά να προστατεύονται κατά την αποθήκευσή τους έναντι μη εξουσιοδοτημένης πρόσβασης, τροποποίησης και αντικατάστασης. Αυτό περιλαμβάνει την ορθή κρυπτογράφησή τους και την διαγραφή όσων στιγμιοτύπων δεν χρειάζονται πλέον.

- **Ασφάλιση του μηχανήματος φιλοξενίας**:

  Το μηχάνημα φιλοξενίας αποτελεί ένα από τα πιο σημαντικά σημεία που πρέπει να προστατευτούν. Αν ένας εισβολέας καταφέρει να αποκτήσει πρόσβαση σε αυτό, θα μπορεί να αποκτήσει πρόσβαση σε όλες τις εικονικές μηχανές που αυτό φιλοξενεί. Επομένως, οι επιχειρήσεις πρέπει να εφαρμόζουν τακτικά ενημερώσεις λογισμικού, να περιορίζουν την πρόσβαση στο μηχάνημα, να εφαρμόζουν πολύπλοκους κωδικούς πρόσβασης και να παρακολουθούν την πρόσβαση σε αυτό.

- **Σκλήρυνση των εικονικών μηχανών**:

  Οι εικονικές μηχανές πρέπει να σκληρύνουν προκειμένου να περιοριστεί η πρόσβαση σε αυτές, διότι πολύ συχνά αποτελούν σημείο εισόδου για την εξάπλωση κακόβουλου λογισμικού. Αυτό περιλαμβάνει μεταξύ άλλων την απενεργοποίηση υπηρεσιών που δεν χρειάζονται και την εφαρμογή πολιτικών πρόσβασης και ασφαλείας.

2.4 Δοχεία και δοχειοποίηση (containerization)

Τα δοχεία αποτελούν και αυτά κομμάτι των τεχνολογιών εικονικοποίησης αλλά σε επίπεδο λειτουργικού συστήματος. Κατά την περιγραφή της διαδικασίας δημιουργίας δοχείων δεν χρησιμοποιείται πλέον ο όρος εικονικοποίηση αλλά δοχειοποίηση (containerization). Σε αντίθεση με την εικονικοποίηση διακομιστών που αναφέρθηκε στην Ενότητα 1.2, για την δοχειοποίηση δεν είναι αναγκαία η ύπαρξη υπερ-επόπτη αλλά έναν παρόμοιο ρόλο αναλαμβάνει η μηχανή δοχείων.

Τα απαραίτητα συστατικά για την επίτευξη της δοχειοποίησης είναι μια εικόνα δοχείου και μια μηχανή δοχείων. Στην εικόνα δοχείου εμπεριέχονται τα συστατικά μέρη ενός λογισμικού και οδηγίες συναρμολόγησής του προς ανάγνωση από την μηχανή δοχείων. Η συναρμολόγηση αυτή πραγματοποιείται σε στρώσεις. Δηλαδή, για να φτάσουμε στο τελικό αποτέλεσμα, κάθε αυτοτελές κομμάτι του λογισμικού εισάγεται πάνω από το προηγούμενό του. Από εκεί και πέρα, για την δημιουργία δοχείων από τις αντίστοιχες εικόνες τους υπάρχει μια αλληλουχία βημάτων

που πρέπει να περαιωθεί. Αρχικά, η μηχανή δοχείων θα προσπελάσει την εικόνα δοχείου προκειμένου να εξακριβώσει τις προδιαγραφές του. Έπειτα, με την βοήθεια των μεθόδων απομόνωσης που έχει στην διάθεσή της μέσω των μηχανισμών εικονικοποίησης του πυρήνα του λειτουργικού συστήματος φιλοξενίας, θα δημιουργήσει ένα εικονικό περιβάλλον απομονωμένο από το ήδη υπάρχον φυσικό και τέλος, θα πραγματοποιήσει εκκίνηση του δοχείου ως διεργασία του συστήματος.

Ορισμένα από τα βήματα δημιουργίας των δοχείων περιλαμβάνουν και άλλες διαδικασίες για τις οποίες είναι υπεύθυνα συγκεκριμένα προγράμματα με τα οποία συνεργάζεται η μηχανή δοχείων. Σε κάθε περίπτωση, θα χρησιμοποιηθεί ένα πρόγραμμα για τον ρόλο ενός Container Runtime χαμηλού επιπέδου, όπως είναι το runC. Αυτού του είδους τα προγράμματα είναι υπεύθυνα για την επικοινωνία με τον πυρήνα του ΛΣ φιλοξενίας ώστε να ξεκινήσουν οι διαδικασίες χαμηλού επιπέδου, όπως η ανάθεση χώρων ονομάτων και ομάδων ελέγχου σε δοχεία. Υπάρχουν και μηχανές δοχείων χαμηλού επιπέδου έναντι των πιο γνωστών υψηλού επιπέδου όπως το Docker, στις οποίες λείπουν ορισμένες λειτουργίες, όπως η απόκτηση εικόνων δοχείων από ένα κεντρικό αποθετήριο και ο έλεγχος εγκυρότητας των ρυθμίσεων των εικόνων αυτών. Οι μηχανές δοχείων υψηλού επιπέδου είναι ουσιαστικά μια συλλογή εργαλείων που διευκολύνουν την δοχειοποίηση απαλλάσσοντας τον χρήστη από την ανάγκη της χειροκίνητης εκτέλεσης των βημάτων της [66]. Αυτές αναλαμβάνουν πολλές φορές ονομάζονται και Container Runtimes υψηλού επιπέδου [67].

2.4.1 Μηχανές δοχείων και εικονικοποίηση σε επίπεδο ΛΣ

Πολλά λειτουργικά συστήματα, ειδικά αυτά που κάνουν χρήση του πυρήνα Linux, διαθέτουν μηχανισμούς απομόνωσης διεργασιών και δυνατότητες εικονικοποίησης (λειτουργικού συστήματος), όπως είναι οι ομάδες ελέγχου (control groups) με τις οποίες μπορεί να επιτευχθεί ο περιορισμός πόρων σε ένα υποσύνολο διεργασιών, καθώς και οι χώροι ονομάτων χρηστών (user namespaces) που προσφέρουν την δυνατότητα περιορισμού των δικαιωμάτων που έχει μια διεργασία στο σύστημα αρχείων. Με την βοήθεια των εργαλείων αυτών, δύναται να πραγματοποιηθεί εικονικοποίηση σε επίπεδο ΛΣ. Μια μέθοδος εικονικοποίησης λογισμικού όπου ο πυρήνας ενός ΛΣ επιτρέπει την ύπαρξη πολλαπλών απομονωμένων περιβαλλόντων [68], τα οποία έπειτα θα συμπεριφέρονται ως ξεχωριστά συστατικά λογισμικού [69].

Μια από τις υποχρεώσεις της μηχανής δοχείων είναι η επικοινωνία με τον πυρήνα του λειτουργικού συστήματος φιλοξενίας για να επιτευχθεί αυτή η εικονικοποίηση. Ουσιαστικά, η μηχανή δοχείων λειτουργεί ως διαμεσολαβητής των δοχείων για να μοιράζονται το ίδιο λειτουργικό σύστημα και κατά προέκταση τους υποκείμενους πόρους του μεταξύ τους. Αυτό συμβαίνει

διότι αυτή αιτείται από το λειτουργικό σύστημα να επιτρέψει την απομόνωση των διεργασιών που εκτελούνται σε χώρους ονομάτων και να αναθέσει αποκλειστικούς πόρους συστήματος σε αυτές. Με αυτό τον τρόπο, οι διεργασίες είναι ανεξάρτητες μεταξύ τους και απομονωμένες ενώ διαμοιράζονται με ασφαλή τρόπο (σε ένα βαθμό) τους πόρους του συστήματος. Ένα δοχείο μπορεί να θεωρηθεί πως αντιστοιχεί σε μια τέτοια απομονωμένη διεργασία.

Το τελικό επιθυμητό αποτέλεσμα της δοχειοποίησης είναι η δημιουργία ενός απομονωμένου περιβάλλοντος (δηλ. το εκτελέσιμο "δοχείο") στο οποίο πραγματοποιείται η εκτέλεση ενός λογισμικού με βιβλιοθήκες και εξαρτήσεις διαφορετικών εκδόσεων συγκριτικά με αυτές που μπορεί να προϋπάρχουν στο σύστημα. Το περιβάλλον αυτό είναι υποχρεωμένο να χρησιμοποιήσει ένα υποσύνολο των πόρων του συστήματος που του απονεμήθηκε μέσω της μηχανής δοχείων και αναμένεται να εκτελείται με την ίδια συμπεριφορά ανεξαρτήτως υποδομής. Τα παραπάνω είναι εφικτά και με την χρήση εικονικών μηχανών αλλά με κόστος την επιβάρυνση του συστήματος φιλοξενίας λόγω της αδυναμίας διαμοιρασμού των πόρων με τον τρόπο που το επιτυγχάνουν τα δοχεία. Το γεγονός ότι κάθε δοχείο μοιράζεται τον ίδιο πυρήνα του λειτουργικού συστήματος φιλοξενίας με τα υπόλοιπα δοχεία, έχει ως αποτέλεσμα να μην χρειάζεται να απονεμηθούν πόροι για εικονικοποίηση μηχανών με σκοπό την στέγαση ενός ολόκληρου ξεχωριστού λειτουργικού συστήματος (όπως γίνεται στις εικονικές μηχανές). Επιπλέον, κάθε στρώση μιας εικόνας δοχείου αντιστοιχουμένη σε ένα στιγμιότυπο λογισμικού, δύναται να χρησιμοποιηθεί ταυτοχρόνως από περισσότερα του ενός δοχεία προς αποφυγή διπλής αποθήκευσης [70]. Αυτό το χαρακτηριστικό συμβάλλει στην εξάλειψη περιττής υπολογιστικής ισχύος. Τέλος, κάθε εικόνα δοχείου μπορεί να χρησιμοποιηθεί ως πρότυπο για την δημιουργία μιας καινούριας, καθιστώντας έτσι ευκολότερη την επεκτασιμότητα ενός λογισμικού σε αντίθεση με τις εικονικές μηχανές, όπου κάθε νέο στιγμιότυπό τους απαιτεί την επαναληπτική εκτέλεση των βημάτων δημιουργίας τους.

2.4.2 Εργαλεία διαχείρισης δοχείων και έλευση του Docker

Στις μέρες μας, η δημοτικότητα του Docker έχει συνταυτίσει τους όρους Docker και Container (δοχείο) αν και είναι διαφορετικοί. Παρ' όλα αυτά, η ιδέα της δημιουργίας απομονωμένων περιβαλλόντων εκτέλεσης λογισμικού υπήρχε προτού βγει το Docker στην αγορά. Ιστορικά, οι πρώτες τεχνολογίες περί δοχείων έκαναν την είσοδό τους από το 1979, όταν εισήχθη το chroot [71] στην έβδομη έκδοση του Unix [72]. Πρόκειται για μια εντολή που περιορίζει την πρόσβαση αρχείων που διαθέτει μια εφαρμογή, σε ένα συγκεκριμένο φάκελο, ο οποίος ορίζεται ως ο καινούριος αρχικός (root) (για την εφαρμογή αυτή). Ο κύριος σκοπός του chroot ήταν η ενίσχυση της ασφάλειας ούτως ώστε στην περίπτωση εκμετάλλευσης μιας εσωτερικής

ευπάθειας της εφαρμογής, να παραμένουν ανεπηρέαστα τα μέρη του συστήματος εκτός του φακέλου στον οποίο αυτή είχε πρόσβαση. Εκείνη την εποχή αυτό ήταν αρκετό, αλλά οι απαιτήσεις ασφαλείας με τον καιρό αυξάνονταν και υπήρχε η ανάγκη διαφορετικών μεθόδων απομόνωσης μιας και από κατασκευής του, το chroot δεν περιόριζε έναν χρήστη ή μια διεργασία με διαχειριστικά δικαιώματα [73]. Μερικά χρόνια αργότερα, το 2004 δημιουργήθηκαν και ενσωματώθηκαν στον πυρήνα του Linux οι ομάδες ελέγχου, με την βοήθεια των οποίων ήταν πλέον δυνατή η απομόνωση πόρων του συστήματος σε ένα υποσύνολο διεργασιών.

Αυτές οι τεχνολογίες σήμαναν της έναρξη της ιδέας της δοχειοποίησης και έτσι το 2008 ήρθε στο προσκήνιο το LXC (Linux Containers)[6]. Το πρώτο εργαλείο που χρησιμοποιούσε τεχνολογίες δοχείων για την δημιουργία και εκτέλεση πολλαπλών λειτουργικών συστημάτων Linux στο ίδιο μηχάνημα. Χρησιμοποιώντας μηχανισμούς που προσέφερε το λειτουργικό σύστημα, επέτρεπε την πλήρη εικονικοποίηση ενός στιγμιότυπου Linux σε μορφή δοχείου και παρείχε αρκετές λειτουργίες, όπως η δημιουργία, η εκκίνηση και η διαγραφή LXC δοχείων. Παρ' όλα αυτά, επικεντρωνόταν στην δοχειοποίηση λειτουργικών συστημάτων και όχι εφαρμογών [75], καθιστώντας δύσκολη και περίπλοκη την χρήση του όταν η κύρια ανάγκη ήταν η απομόνωση εφαρμογών.

Ενώ λοιπόν προϋπήρχαν εργαλεία διαχείρισης δοχείων, τα οποία χρησιμοποιούνται ακόμα και στις μέρες μας, λόγω του συνδυασμού της δυσκολίας στην χρήση τους και του εξειδικευμένου σκοπού ύπαρξής τους δεν είχαν υιοθετηθεί ευρέως. Όλα τα παραπάνω οδήγησαν στην δημιουργία του Docker το 2013, με την έλευση του οποίου η τεχνολογία των δοχείων εκτοξεύτηκε. Το Docker είναι ένα σύνολο προϊόντων PaaS (Platform as a Service) (Πλατφόρμα ως Υπηρεσία) και μέσω αυτού, παρέχεται μια πλατφόρμα με μηχανισμούς για συναρμολόγηση, θέση σε λειτουργία, εκτέλεση, ενημέρωση και διαχείριση προγραμμάτων σε μορφή δοχείων. Σε αντίθεση με το LXC, το Docker αποτελεί μια μηχανή δοχείων υψηλού επιπέδου με κύριο στόχο την δοχειοποίηση εφαρμογών. Εκτός από τον διαχωρισμό ανάμεσα στον πηγαίο κώδικα, τις βιβλιοθήκες και εξαρτήσεις ενός λογισμικού από το κύριο σύστημα (φιλοξενίας), παρέχει και δυνατότητες επικοινωνίας με αποθετήρια εικόνων δοχείων, όπως είναι το Docker Hub[7], το επίσημο αποθετήριο του Docker, το Quay[8], ένα εναλλακτικό αποθετήριο της Red Hat και οποιοδήποτε άλλο αποθετήριο συμβατό με τις προδιαγραφές που έχει ορίσει η ανοικτή δομή διοίκησης OCI (Open Container Initiative)[9]. Μέσω υπηρεσιών αυτού του είδους, οι χρήστες έχουν πρόσβαση και διαμοιράζονται χιλιάδες εικόνες δοχείων που τους επιτρέπουν

[6]L. Containers. What's LXC?. URL: `https://linuxcontainers.org/lxc/introduction/`.
[7]Docker. Build and Ship any Application Anywhere. URL: `https://hub.docker.com/`.
[8]R. Hat. Quay builds, analyzes, distributes your container images. URL: `https://quay.io/`.
[9]T. L. Foundation. Open Container Initiative. URL: `https://opencontainers.org/`.

να ολοκληρώσουν διάφορα μέρη μιας υπάρχουσας ή νέας εφαρμογής. Επιπλέον, όντας μια μηχανή δοχείων υψηλού επιπέδου, όλες οι λειτουργίες που θα απαιτούσαν εξειδικευμένες γνώσεις προκειμένου να πραγματοποιηθούν, έχουν συμπυκνωθεί σε απλές εντολές. Έτσι, καθίσταται εύκολη η χρήση του Docker για προγραμματιστές κάθε επιπέδου και απλοποιείται κατά πολύ την διαδικασία ανάπτυξης και παράδοσης κατανεμημένων εφαρμογών. Προσφέροντας μια φιλική προς τον χρήστη διεπαφή, έλεγχο εκδόσεων (version control) για δοχεία [79] και ένα οικοσύστημα γύρω από όλα αυτά, είναι εμφανής ο λόγος που κατάφερε να επισκιάσει το LXC.

2.4.3 Χρήση δοχείων στην ανάπτυξη και παράδοση εφαρμογών

Ο λόγος που πολλές επιχειρήσεις στρέφονται στην χρήση της δοχειοποίησης είναι διότι με την παραδοσιακή μέθοδο ανάπτυξης λογισμικού, υπήρχε πάντα το ρίσκο ένα πρόγραμμα που αναπτύχθηκε σε ένα συγκεκριμένο περιβάλλον να μη λειτουργεί με τον αναμενόμενο τρόπο κατά τη μεταφορά του σε ένα άλλο, εκτός εάν έχει ελεγχθεί ότι υπάρχουν όλες οι εξαρτήσεις που χρειάζεται, στις εκδόσεις που τις χρειάζεται. Ακόμα και σε αυτήν την περίπτωση όμως, πέραν του κόπου για τον έλεγχο και τον παράγοντα ανθρώπινου λάθους, είναι αρκετά πιθανό ένα άλλο πρόγραμμα να χρειάζεται διαφορετικές εκδόσεις των ίδιων εξαρτήσεων. Αυτό πρακτικά σήμαινε πως το έτερο αυτό πρόγραμμα θα έπρεπε να στεγαστεί σε διαφορετικό διακομιστή, αυξάνοντας το σχετικό κόστος.

Τα παραπάνω προβλήματα έρχεται να λύσει η τεχνολογία της δοχειοποίησης, αφού αυτή καθιστά δυνατή την συστέγαση δοχείων, δηλ. διαφορετικών προγραμμάτων ή συστατικών προγραμμάτων στο ίδιο μηχάνημα (είτε αυτό είναι φυσικό είτε εικονικό) και μάλιστα με λιγότερη επιβάρυνση συγκριτικά με την χρήση εικονικών μηχανών. Όπως αναφέραμε και στην Ενότητα 2.4.2, από το 2013 και έπειτα, η άφιξη του Docker επιτάχυνε κατά πολύ την υιοθέτηση της τεχνολογίας αυτής. Σε τέτοιο βαθμό, που σε μια έρευνα της IBM [80] βρέθηκε πως το 61% όσων ξεκίνησαν να χρησιμοποιούν δοχεία, το κάνουν στο 50% ή παραπάνω των εφαρμογών που δημιούργησαν τα τελευταία δύο χρόνια, ενώ 64% αυτών, αναμένουν στο 50% των υπαρχουσών εφαρμογών τους να κάνουν χρήση δοχείων στα επόμενα δύο χρόνια.

Ένας ακόμα λόγος που επιτάχυνε την υιοθέτηση της τεχνολογίας της δοχειοποίησης, είναι πως τα τελευταία χρόνια έχουν αλλάξει δραματικά και οι μέθοδοι ανάπτυξης και παράδοσης εφαρμογών. Από τις παραδοσιακές μεθόδους, όπως το μοντέλο καταρράκτη (waterfall) [81] βάσει του οποίου υπήρχε ένα συγκεκριμένο αμετάβλητο σχέδιο ανάπτυξης που καλούνταν να ακολουθήσει μια ομάδα ανάπτυξης λογισμικού, έχουμε φτάσει σε μια εποχή όπου οι απαιτήσεις της αγοράς αλλάζουν συνεχώς, με αποτέλεσμα να υπάρχει ανάγκη για καινούριες μεθόδους που να ανταποκρίνονται τάχιστα στις αλλαγές αυτές. Έτσι, έχουν δημιουργηθεί μεθοδολογίες

όπως η Agile [82] κατά την οποία η ανάπτυξη και η παράδοση λογισμικού πραγματοποιείται σε πολλές μικρές και ευμετάβλητες φάσεις προκειμένου να προσαρμόζεται στις αλλαγές που ενδέχεται να υπάρξουν στον αρχικό σχεδιασμό. Η μεθοδολογία αυτή συνεργάζεται με πρακτικές όπως το DevOps [83] όπου οι ομάδες υπεύθυνες για την ανάπτυξη και λειτουργία μιας εφαρμογής επικοινωνούν στενά με σκοπό να υπάρχει μια συνεχής ροή παραγωγής και παράδοσης λογισμικού. Αυτό επιτυγχάνεται με μια πρακτική του DevOps, το CI/CD (Continuous Integration/Continuous Delivery) (Συνεχής Ενοποίηση/Συνεχής Παράδοση) [84]. Κατά το μοντέλο αυτό, δημιουργούνται αυτοματοποιημένες διαδικασίες που εκτελούνται κατά την διάρκεια της ανάπτυξης και παράδοσης μιας εφαρμογής προκειμένου να πραγματοποιείται έλεγχος της ποιότητας του κώδικα, να εντοπίζονται σφάλματα και να παράγονται εκτελέσιμα πακέτα τα οποία έπειτα παραδίδονται στους πελάτες. Μάλιστα, μπορεί να υποστηρίζεται και η Συνεχής Διάταξη (Continuous Deployment - CD) όπου τα εκτελέσιμα πακέτα των εφαρμογών διατάσσονται σε περιβάλλοντα παραγωγής και γίνονται άμεσα διαθέσιμα προς απομακρυσμένη κατανάλωση από τους πελάτες τους.

Τα δοχεία αποτελούν ιδανική επιλογή για την εφαρμογή των παραπάνω μεθοδολογιών και πρακτικών, καθώς επιτρέπουν το γρήγορο και αποτελεσματικό πακετάρισμα εφαρμογών (ή συστατικών (components) εφαρμογής) και την εκτέλεσή τους σε οποιαδήποτε πλατφόρμα ή περιβάλλον νέφους. Λόγω των χαρακτηριστικών τους, εξαλείφουν τυχόν προβλήματα ασυμβατότητας μεταξύ των περιβαλλόντων εκτέλεσής τους και προσφέρουν μεγαλύτερη αποδοτικότητα πόρων διότι μπορεί κανείς να εκτελέσει πολλά περισσότερα αντίγραφα ενός προγράμματος για συγκεκριμένη ποσότητα πόρων σε σχέση με τον αριθμό που θα μπορούσε να εκτελέσει χρησιμοποιώντας εικονικές μηχανές πάνω από αυτούς τους πόρους. Το τελευταίο μάλιστα μειώνει ιδιαίτερα το κόστος λειτουργίας και αυξάνει την κλιμακωσιμότητα. Τα δοχεία επίσης, λόγω της ταχείας διάθεσης και εκτέλεσής τους, συμβαδίζουν πλήρως και με τον ρυθμό ανάπτυξης και παράδοσης που υποστηρίζουν οι παραπάνω μεθοδολογίες. Τέλος, η ανεξαρτησία των δοχείων μεταξύ τους, καθιστά πιο σταθερή την εφαρμογή, αφού σε περίπτωση που κάποιο από αυτά αποτύχει, η υπόλοιπη εφαρμογή θα συνεχίσει να λειτουργεί κανονικά, ενώ η ανακατασκευή του δοχείου που απέτυχε μπορεί να επιτευχθεί εύκολα και γρήγορα με μια απλή τροποποίηση της εκάστοτε εικόνας δοχείου. Έχοντας αυτά υπόψιν, γίνεται εμφανής και ο λόγος που τα δοχεία και εν γένει η δοχειοποίηση εφαρμογών είναι η προτιμότερη επιλογή για περιπτώσεις ανάπτυξης και παράδοσης εφαρμογών που ακολουθούν την αρχιτεκτονική μικρο-υπηρεσιών, καθώς και περιπτώσεις εφαρμογής μεθόδων DevOps [85].

2.4.4 Χρήσεις του Docker στο νέφος

Οι δυνατότητες που προσφέρει το Docker το καθιστούν την καταλληλότερη επιλογή τόσο για επιχειρήσεις που λειτουργούν αποκλειστικά με ενοικιαζόμενους υπολογιστικούς πόρους όσο και για αυτές που επιλέγουν να λειτουργούν με έναν συνδυασμό ενοικιαζόμενων και ιδιόκτητων φυσικών εγκαταστάσεων. Υπάρχουν δύο βασικές περιπτώσεις χρήσης του Docker στο νέφος. Συγκεκριμένα αυτές είναι: η εγκατάσταση δοχείων σε εικονικές μηχανές και η εγκατάσταση δοχείων έμμεσα σε πόρους χωρίς την ανάγκη δημιουργίας εικονικής μηχανής. Η δεύτερη περίπτωση χρήσης εντάσσεται στην κατηγορία υπηρεσιών CaaS [86] (Container as a Service) (Δοχείο ως Υπηρεσία), όπως η ECS (Elastic Container Service) της Amazon. Μια υπηρεσία όπου ένας πάροχος νέφους αντί να παρέχει πρόσβαση σε υπολογιστικούς πόρους γενικού σκοπού, παρέχει μια ευέλικτη υποδομή, κατάλληλα ρυθμισμένη για την εκτέλεση δοχείων [87].

Από τις δύο αυτές επιλογές, η πρώτη προσφέρει μια ευελιξία ως προς την διαμόρφωση του περιβάλλοντος εκτέλεσης των δοχείων. Οι χρήστες έχουν την δυνατότητα να ακολουθήσουν ορισμένες ορθές πρακτικές ασφαλείας που μπορεί να έχουν θεσπιστεί στην εταιρεία τους, να εγκαταστήσουν επιπλέον λογισμικό παρακολούθησης και ελέγχου ποιότητας των δοχείων και να προσαρμόσουν το περιβάλλον εκτέλεσης των δοχείων στις ανάγκες τους. Επιπλέον, η ύπαρξη μιας επιπλέον στρώσης απομόνωσης μεταξύ των δοχείων και του κύριου συστήματος αποτελεί ένα επιπρόσθετο εμπόδιο σε περιπτώσεις που ένα δοχείο βρεθεί σε κατάσταση παραβίασης. Το Docker επιτρέποντας μέσω των δοχείων την περιορισμένη έκθεση των πόρων του συστήματος στον έξω κόσμο, καθιστά ευκολότερη την ανίχνευση και απομόνωση των προβλημάτων ασφαλείας, καθώς επίσης και την αποκατάστασή τους.

Από την άλλη, η δεύτερη περίπτωση χρήσης επιτρέπει στις επιχειρήσεις/πελάτες να επικεντρωθούν στην ανάπτυξη των δοχειοποιημένων εφαρμογών τους και να αφήσουν την διαχείριση της υποδομής στον πάροχο νέφους. Αυτό είναι ιδιαίτερα χρήσιμο σε περιπτώσεις που οι επιχειρήσεις δεν έχουν υπαλλήλους με την απαραίτητη εμπειρία και ικανότητες σε αυτόν τον τομέα ή δεν έχουν τον απαραίτητο προϋπολογισμό για αυτό. Η χρήση υπηρεσιών CaaS αυτομάτως εξασφαλίζει στις επιχειρήσεις τα πλεονεκτήματα των υπηρεσιών IaaS, όπως η κλιμάκωση, η αντοχή σε αποτυχίες και η ευελιξία, ενώ ταυτόχρονα προσφέρει και τα πλεονεκτήματα των δοχείων, όπως η ταχεία διάθεση και απόσυρσή τους αλλά και η υψηλή τους απόδοση κατά την εκτέλεση. Συγκεκριμένα, μέσω των υπηρεσιών CaaS, οι χρήστες διαθέτουν δυνατότητες ενορχήστρωσης δοχείων [88] χωρίς την ανάγκη χειροκίνητου στησίματος πλατφορμών, όπως

το Kubernetes [89] και το Docker Swarm [90], που παρέχουν αυτή την δυνατότητα. Επιπρόσθετα, οι πάροχοι υπηρεσιών CaaS έχουν την κατάλληλη τεχνογνωσία για να προστατεύσουν τις υπηρεσίες τους, εξασφαλίζοντας ένα υψηλό επίπεδο ασφάλειας για τις εφαρμογές που τις χρησιμοποιούν.

Σε κάθε περίπτωση, λόγω των πλεονεκτημάτων που παρέχει η χρήση δοχείων, είναι πολύ συνήθης η θέση σε λειτουργία, εφαρμογών μέσω Docker σε περιβάλλοντα νέφους επειδή αυτό απομονώνει τις αναγκαίες βιβλιοθήκες και εξαρτήσεις τους από το υπόλοιπο σύστημα και επιτρέπει την αποδοτικότερη διαχείριση των εφαρμογών αυτών μέσω της διεπαφής του με εντολές για εκκίνηση, επιτήρηση πόρων, παύση και άλλες λειτουργίες. Τα προβλήματα που μπορεί να προέκυπταν σε ένα περιβάλλον ανάπτυξης, όπως μη συμβατές εκδόσεις προγραμμάτων και η δυσκολία διαχείρισής τους, εξαλείφονται με την χρήση δοχείων αφήνοντας το Docker να εγκαθιδρύσει έναν συστημικό τρόπο διανομής και ελέγχου εφαρμογών. Επιπροσθέτως, καθιστά πολύ εύκολη τη μεταφορά τους σε οποιοδήποτε μηχάνημα (εφόσον αυτό υποστηρίζει την εκτέλεση της μηχανής δοχείων του Docker) ή ακόμα και νέφος, θέτοντας το κορυφαία επιλογή για επιχειρήσεις που επιλέγουν να ακολουθήσουν την στρατηγική πολλαπλών νεφών (multi-cloud computing) κατά την κατασκευή εφαρμογών. Δηλαδή να μην βασίζονται αποκλειστικά σε έναν πάροχο νέφους για όλες τις λειτουργίες μιας εφαρμογής [91] αλλά να εκμεταλλεύονται τα οφέλη (π.χ. περισσότερη ασφάλεια, ποιότητα και αυξημένη διαθεσιμότητα) χρήσης υπηρεσιών από πολλούς παρόχους με γνώμονα τις ανάγκες τους.

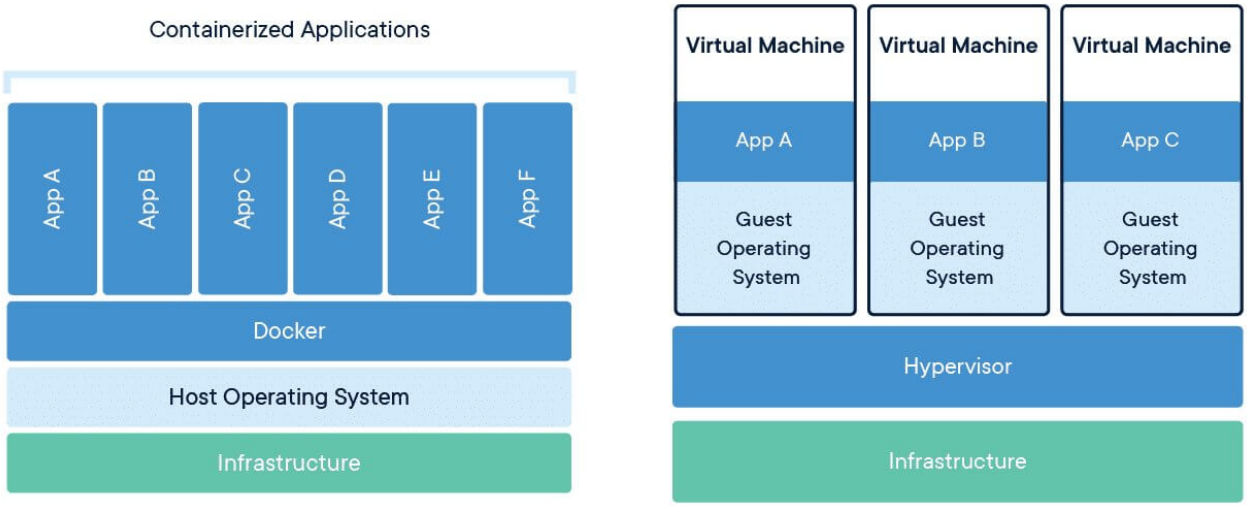
2.4.5 Πλεονεκτήματα δοχείων έναντι εικονικών μηχανών

Παρ' όλο που πολλές φορές τα δοχεία συγχέονται με τις εικονικές μηχανές, οι δύο αυτές έννοιες έχουν αρκετές διαφορές στην αρχιτεκτονική τους. Στην παραδοσιακή εικονικοποίηση είτε αυτή γίνεται στις υπάρχουσες υποδομές μιας επιχείρησης είτε σε ένα περιβάλλον νέφους, ένας υπερεπόπτης πρέπει να χρησιμοποιηθεί για να εικονικοποιήσει φυσικό υλικό. Αυτό συμβαίνει διότι μια εικονική μηχανή είναι μια πλήρης αναπαράσταση ενός φυσικού διακομιστή. Από την άλλη, ένα δοχείο αντί να εικονικοποιήσει το υλικό, εικονικοποιεί το λειτουργικό σύστημα, ή αλλιώς εξομοιώνει μονάχα το περιβάλλον εκτέλεσης ενός λογισμικού. Άρα, εάν θεωρηθεί ως τελικός στόχος η εκτέλεση ενός λογισμικού απομονωμένο από το υπόλοιπο σύστημα, αυτό επιτυγχάνεται με δύο διαφορετικούς τρόπους, αφού οι εικονικές μηχανές και τα δοχεία χρησιμοποιούν διαφορετικού είδους εικονικοποίηση. Στην περίπτωση των δοχείων δεν έχουμε απομόνωση μηχανών αλλά διεργασιών. Γεγονός που συμβάλλει στην αποφυγή της επιβάρυνσης του συστήματος που θα επιβάλλονταν από τις διεργασίες του υπερ-επόπτη και της καθυστέρησης που θα υπήρχε για την εκκίνηση ενός ολόκληρου λειτουργικού συστήματος. Επιπλέον, η μη

χρήση τεχνολογιών εικονικοποίησης σε επίπεδο υλικού αυξάνει την μεταφερσιμότητα των δοχείων, καθώς σε μια εικόνα δοχείου μπορούν να ορισθούν όλες οι απαραίτητες εξαρτήσεις ενός λογισμικού και τα δοχεία μπορούν να αναδημιουργηθούν και να εκτελεστούν σε οποιοδήποτε περιβάλλον νέφους είναι ήδη εγκατεστημένη μια μηχανή δοχείων (όπως το Docker) όπου μάλιστα η ταχύτητα δημιουργίας τους είναι πολύ πιο γρήγορη σε σχέση με αυτή των εικονικών μηχανών λόγω του μικρού μεγέθους και των ελάχιστων μη λειτουργικών απαιτήσεών τους.

Ένας από τους χαρακτηρισμούς των δοχείων είναι η "ελαφρότητά" τους σε σχέση με μια εικονική μηχανή λόγω της ικανότητάς τους να μοιράζονται τον πυρήνα του ίδιου λειτουργικού συστήματος (ΛΣ). Η απαίτηση μιας εικονικής μηχανής να χρειάζεται, εικονικό αντίγραφο του υλικού, δικό της ΛΣ και εγκατεστημένα πακέτα προς υποστήριξη ενός λογισμικού την καθιστά μεγαλύτερη σε μέγεθος και λιγότερο αποδοτική στη χρήση πόρων του συστήματος. Απεναντίας, τα δοχεία περιέχουν μονάχα το λογισμικό προς εκτέλεση και τις εξαρτήσεις (βιβλιοθήκες και προγράμματα) που χρειάζεται προκειμένου αυτό να είναι λειτουργικό [92]. Επομένως, είναι εγγενώς μικρότερα σε μέγεθος και έχουν και μικρότερο χρόνο εκκίνησης. Πράγμα που τραβάει το ενδιαφέρον των επιχειρήσεων διότι αυτό μεταφράζεται σε υψηλότερη αποδοτικότητα & βαθμό χρήσης των διακομιστών και επομένως μειωμένα κόστη λειτουργίας. Επίσης, τα δοχεία κλιμακώνονται αρκετά πιο γρήγορα σε σχέση με τις εικονικές μηχανές, επιτρέποντας την αντίδραση σε οποιαδήποτε διακύμανση του φόρτου εργασίας.

Το Σχήμα 2.16 παρουσιάζει τις διαφορές ανάμεσα στην αρχιτεκτονική της εικονικοποίησης (δεξιά) και των δοχείων (αριστερά).



Σχήμα 2.16: Χρήση εικονικοποίησης έναντι δοχείων [93]

Η απουσία του εικονικού λειτουργικού υλικού είναι το χαρακτηριστικό που καθιστά τα δοχεία γρήγορα, φορητά και μικρότερα σε μέγεθος. Για να γίνει πιο εμφανής η διαφορά, σύμφωνα με τη Red Hat [94], το μέγεθος των εικονικών μηχανών είναι της τάξεως των gigabyte ενώ των δοχείων είναι της τάξεως των megabyte. Πολλές φορές, όπως αναφέρεται και σε μια δημοσίευση [92] της IBM, τα δοχεία χρησιμοποιούνται για εφαρμογές αρχιτεκτονικής μικρο-υπηρεσιών (microservices), όπου κάθε ξεχωριστό κομμάτι (δηλ. μικρο-υπηρεσία) αντιπροσωπεύει ένα συστατικό της εφαρμογής που παίρνει την μορφή ενός δοχείου. Με αυτόν τον τρόπο, είναι ευκολότερη η κλιμάκωση μιας εφαρμογής απ' ότι θα ήταν με μια μονολιθική αρχιτεκτονική. Αυτό συμβαίνει διότι μπορεί να κλιμακώνεται μόνο το μέρος ή τα μέρη της που έχουν αύξηση του φόρτου εργασίας τους, μέσω της χρήσης περισσότερων δοχείων που αντιστοιχούν σε αυτά τα μέρη/μικρο-υπηρεσίες. Αντιθέτως, η κλιμάκωση μιας μονολιθικής εφαρμογής θα απαιτούσε την δημιουργία πολλαπλών δοχείων μεγάλου μεγέθους ή πολλαπλών εικονικών μηχανών, οδηγώντας σε μεγάλη και αχρείαστη σπατάλη πόρων. Λόγω του γεγονότος πως μια εφαρμογή αποτελείται από πολλαπλές μικρο-υπηρεσίες, οι οποίες μπορεί να έχουν πολλαπλά στιγμιότυπα (δηλ. μεγάλο αριθμό δοχείων) ανά πάσα χρονική στιγμή, απαιτείται η χρήση μιας πλατφόρμας ενορχήστρωσης δοχείων για την αυτοματοποίηση της εγκατάστασης, εκτέλεσης και κλιμάκωσης της εφαρμογής κατά μήκος πολλαπλών πόρων (δηλ. εικονικών ή φυσικών μηχανών), όπως είναι το Kubernetes ή το Docker Swarm. Από την άλλη, αν είναι επιθυμητή η ενορχήστρωση των δοχείων μιας εφαρμογής σε έναν πόρο (φυσικό ή εικονικό μηχάνημα), τότε είναι δυνατή η χρήση εργαλείων όπως το Docker Compose[10].

Το μόνο μειονέκτημα της τεχνολογίας των δοχείων, το οποίο δεν επηρεάζει κατά πολύ τη χρήση τους, είναι το γεγονός ότι δοχεία που δημιουργήθηκαν για να εκτελούν προγράμματα που απαιτούν την ύπαρξη του λειτουργικού συστήματος Windows δε μπορούν να εκτελεστούν σε ένα περιβάλλον με τον πυρήνα του Linux και το ίδιο ισχύει και αντιστρόφως [96].

Τα δοχεία προσφέρουν μια σειρά από πλεονεκτήματα σε σχέση με τις παραδοσιακές εικονικές μηχανές. Αυτά, σύμφωνα με την IBM [97], μπορούν να συνοψιστούν ως εξής:

- **Μεταφερσιμότητα**:

  Τα δοχεία είναι ανεξάρτητα από το λειτουργικό σύστημα και το περιβάλλον εκτέλεσής τους και ως εκ τούτου μπορούν να εκτελεστούν σε οποιαδήποτε πλατφόρμα ή περιβάλλον νέφους ομοιόμορφα και με συνέπεια. Εφόσον τα κομμάτια ενός προγράμματος μπορούν να ορισθούν σε ένα μονάχα αρχείο κειμένου και να κατασκευαστούν με αυτοματοποιημένο τρόπο οπουδήποτε, αυτό δίνει στα δοχεία πολύ μεγαλύτερο προβάδισμα στον τομέα της μεταφερσιμότητας σε σχέση με τις εικονικές μηχανές.

[10]Docker. Docker Compose. URL: `https://github.com/docker/compose`.

- **Εύκολη και συνεπής χρήση**:

  Το Docker Engine ξεκίνησε το βιομηχανικό πρότυπο Docker για τη χρήση δοχείων προσφέροντας απλά εργαλεία ανάπτυξης και μια καθολική προσέγγιση πακεταρίσματος που λειτουργεί εξίσου καλά σε όλες τις πλατφόρμες. Το οικοσύστημα των δοχείων έχει μετατοπιστεί σε μηχανές δοχείων που ακολουθούν τα πρότυπα της Open Container Initiative (OCI) και οι προγραμματιστές μπορούν εύκολα και γρήγορα να πακετάρουν τις εφαρμογές τους ως δοχεία, τα οποία μπορούν να παρέχονται χωρίς την έγνοια υποστήριξης πολλών διαφορετικών πλατφορμών.

- **Ταχύτητα**:

  Τα δοχεία είναι ελαφρύτερα από τις παραδοσιακές εικονικές μηχανές λόγω του διαμοιρασμού του ίδιου ΛΣ και γι' αυτό έχουν μικρότερο χρόνο εκκίνησης. Αυτό τα καθιστά ιδανικά για την ανάπτυξη και την εκτέλεση εφαρμογών σε περιβάλλοντα νέφους όπου η αποδοτικότητα αξιοποίησης των υπολογιστικών πόρων αντανακλάται σε άμεση εξοικονόμηση κόστους.

- **Απομόνωση σφαλμάτων**:

  Εφόσον κάθε δοχείο είναι απομονωμένο και λειτουργεί ανεξάρτητα από τα υπόλοιπα, η αποτυχία ενός δοχείου στο ίδιο ΛΣ δε θα επηρεάσει τη συνεχή λειτουργία των υπόλοιπων δοχείων. Με αυτόν τον τρόπο, οι ομάδες ανάπτυξης λογισμικού μπορούν να εντοπίζουν και να διορθώνουν τυχόν τεχνικά προβλήματα χωρίς να υπάρχει διακοπή λειτουργίας σε άλλα δοχεία. Επιπρόσθετα, ο εντοπισμός του προβλήματος είναι εύκολος διότι εστιάζει σε ένα μόνο δοχείο και όχι σε περισσότερα. Αυτό οδηγεί σε πιο γρήγορη αποσφαλμάτωση και εν τέλει στην πιο γρήγορη ανάπτυξη και συντήρηση προγραμμάτων.

- **Μέγιστη χρηστικότητα πόρων**:

  Το γεγονός ότι τα δοχεία μοιράζονται το ίδιο ΛΣ και κοινά κομμάτια μιας στρώσης εικόνας δοχείου μπορούν να χρησιμοποιηθούν ξανά για την δημιουργία πολλών δοχείων, τα καθιστά αρκετά μικρά σε μέγεθος ώστε να είναι δυνατόν να εκτελεστούν πολλά περισσότερα δοχεία σε έναν διακομιστή απ' ότι θα μπορούσαν εικονικές μηχανές. Επομένως, αξιοποιούνται καλύτερα οι πόροι του συστήματος.

- **Ευκολία διαχείρισης**:

  Με την χρήση των δοχείων έχουμε και την δυνατότητα αυτοματοποίησης της κλιμάκωσης & διαχείρισης δοχειοποιημένων εφαρμογών κατά μήκος πολλαπλών διακομιστών (εικονικών ή φυσικών) μέσω πλατφορμών ενορχήστρωσης δοχείων. Δίχως την υποστήριξη τέτοιου είδους πλατφορμών, ο φόρτος διαχείρισης θα ήταν αρκετά υψηλός και η

διαχείριση του φόρτου εργασίας των δοχειοποιημένων εφαρμογών σε περιβάλλοντα με πολλούς διακομιστές θα απαιτούσε ειδικές γνώσεις για να πραγματοποιηθεί. Παράλληλα, η διαχείριση των υποκείμενων φυσικών διακομιστών μπορεί να αυτοματοποιηθεί μέσω της χρήσης διαχειριζόμενων εκδόσεων των πλατφορμών (managed container orchestration platforms) αυτών που προσφέρονται από παρόχους νέφους. Επιπρόσθετα, η διαχείριση δοχείων σε ένα μόνο σύστημα είναι εξίσου εύκολη και απλή. Οι διαδικασίες ενημέρωσης, εκκίνησης, τερματισμού και εν γένει διαχείρισης των δοχείων πραγματοποιούνται με απλές εντολές, καθιστώντας την χρήση δοχείων προσιτή σε κάθε επιπέδου προγραμματιστή.

- **Ασφάλεια**:

  Η απομόνωση των εφαρμογών ως δοχεία, όταν εφαρμόζεται ακολουθώντας ορθές πρακτικές, εγγενώς αποτρέπει την εισβολή κακόβουλου λογισμικού από το να επηρεάσει τα υπόλοιπα δοχεία ή το σύστημα στο οποίο εκτελούνται. Συγκεκριμένα, χρησιμοποιώντας Kernel Security Modules, όπως είναι το AppArmor ή το SELinux, μπορούν να ορισθούν άδειες ασφαλείας με σκοπό τον περιορισμό του εύρους πρόσβασης του Docker στο σύστημα, ενώ με access authorization plugins [98], περιορίζεται η πρόσβαση στον δαίμονα του Docker. Επιπροσθέτως, πολύ εύκολα μπορεί να περιοριστεί και η επικοινωνία μεταξύ δοχείων, καθώς και με το δίκτυο του συστήματος.

2.4.6 Διαφορές σε υλοποιήσεις της δοχειοποίησης

Στην Ενότητα 2.4 αναφέραμε την έννοια της δοχειοποίησης. Η ραγδαία αύξηση του ενδιαφέροντος και της χρήσης δοχείων οδήγησε στην ανάγκη για πρότυπα γύρω από την τεχνολογία αυτή. Έτσι, η Open Container Initiative (OCI), η οποία ιδρύθηκε τον Ιούνιο του 2015 από την Docker και άλλους ηγέτες του κλάδου, προωθεί κοινά, ανοικτά πρότυπα και προδιαγραφές γύρω από την τεχνολογία δοχείων. Εξαιτίας αυτού, η OCI συμβάλλει στη διεύρυνση των επιλογών για μηχανές δοχείων ανοιχτού κώδικα και συνεπώς, οι χρήστες δε θα είναι εγκλωβισμένοι στην τεχνολογία ενός συγκεκριμένου προμηθευτή. Απεναντίας, θα μπορούν να επωφεληθούν από τις πιστοποιημένες από την OCI τεχνολογίες που θα τους επιτρέπουν να δημιουργούν εφαρμογές σε δοχεία χρησιμοποιώντας ένα ευρύ σύνολο εργαλείων DevOps, τις οποίες θα μπορούν να εκτελούν με τον ίδιο τρόπο σε οποιεσδήποτε υποδομές της επιλογής τους.

Σήμερα, αν και το Docker αποτελεί μία από τις πιο γνωστές και ευρέως χρησιμοποιούμενες μηχανές δοχείων, υπάρχουν πολλές άλλες εναλλακτικές υλοποιήσεις. Για παράδειγμα, το Podman [99] είναι μια γνωστή εναλλακτική μηχανή δοχείων ενώ το LXC και το containerd είναι εναλλακτικά Container Runtimes - μάλιστα το τελευταίο ήταν για καιρό η προεπιλογή του Docker για Container Runtime προτού υιοθετήσει το runC. Οι προαναφερόμενες εναλλακτικές υλοποιήσεις μπορεί να έχουν διαφορετικά χαρακτηριστικά και προεπιλογές αλλά η υιοθέτηση και η αξιοποίηση των προδιαγραφών της OCI καθώς αυτές εξελίσσονται θα διασφαλίσει ότι οι εναλλακτικές αυτές παραμένουν ανεξάρτητες από τους προμηθευτές, πιστοποιημένες για να τρέχουν σε πολλαπλά λειτουργικά συστήματα και χρησιμοποιήσιμες σε πολλαπλά περιβάλλοντα [97].

Παρότι οι προδιαγραφές της OCI έχουν ως στόχο να διασφαλίσουν την ομοιόμορφη λειτουργία της τεχνολογίας αυτής, υπάρχουν αρκετές διαφορές στην υλοποίηση της. Παραδείγματος χάριν, το Podman, ενώ συμμορφώνεται στις προδιαγραφές της OCI όπως το Docker, δουλεύει χωρίς δαίμονα από πίσω, πράγμα που επιδιώκει να κατευνάσει τις ανησυχίες γύρω από τον τρόπο λειτουργίας του Docker. Αυτό συμβαίνει διότι το Docker χρειάζεται διαχειριστικές άδειες για την λειτουργία του δαίμονα του, γεγονός που αφήνει το σύστημα ευάλωτο σε επιθέσεις σε περίπτωση που αυτός παραβιαστεί. Παρότι και οι δύο αυτές μηχανές δοχείων χρησιμοποιούν το runC ως Container Runtime, το Podman χρησιμοποιεί το systemd για την διαχείριση των δοχείων του, το οποίο μπορεί να χρησιμοποιηθεί από χρήστες χωρίς διαχειριστικές άδειες. Συνεπώς, εξαιτίας του τρόπου λειτουργίας του, αν και από το 2021, χάρη στη δουλειά του Akihiro Suda[11], η αποφυγή χρήσης δαίμονα δεν ισχύει μόνο για αυτό πλέον, το Podman δύναται να εκτελεστεί από έναν χρήστη πέραν του διαχειριστή/ριζικού (root).

Ένα ακόμα εργαλείο που έχει παρόμοια αρχιτεκτονική με το Podman είναι το rkt[12] το οποίο προσπαθούσε να κρατήσει μια προσέγγιση ασφαλούς σχεδιασμού εξ αρχής (Secure-by-design). Μπορούσε να ενσωματώσει χαρακτηριστικά ασφαλείας, όπως υποστήριξη SELinux, TPM measurement και εκτέλεση δοχείων σε απομονωμένες από το υλικό εικονικές μηχανές. Παρ' όλα αυτά, σύμφωνα με τον Slingerland [101], έπαψε να έχει ενεργή ανάπτυξη και επομένως δεν θα συνεχίζει να ανταγωνίζεται παρόμοια εργαλεία στον κλάδο των δοχείων.

Σχετικά με τα Container Runtimes runC και containerd, το πρώτο είναι ένα Container Runtime χαμηλού επιπέδου ενώ το δεύτερο υψηλού επιπέδου. Δημιουργήθηκαν και τα δύο από το Docker σε διαφορετικές χρονικές περιόδους. Από τις αρχικές του εκδόσεις, το Docker χρησιμοποιούσε το containerd από προεπιλογή. Μετέπειτα, αποφάσισε να το καταστήσει ένα αυτόνομο Container Runtime και τον ρόλο του πήρε, το runC, προκειμένου να μπορέσουν τα

[11] A. Suda. rootlesskit. 2020. URL: https://github.com/rootless-containers/rootlesskit.
[12] rkt. rkt. URL: https://github.com/rkt/rkt.

δοχεία του Docker να δουλεύουν ευκολότερα σε διαφορετικές πλατφόρμες όπως το Kubernetes [102]. Επιπροσθέτως, η απόφαση αυτή κατέστησε το Docker πιο διαχειρίσιμο διότι πλέον αποτελούνταν από πολλά μικρότερα εργαλεία, όπου το καθένα από αυτά είχε συγκεκριμένους ρόλους. Αναλυτικότερα, το containerd πλέον είναι υπεύθυνο για την απόκτηση εικόνων δοχείων και την διαχείρισή τους, προτού τις μεταβιβάσει στο runC, το οποίο είναι το εργαλείο που θα αλληλεπιδράσει με τον πυρήνα του Linux προκειμένου να χρησιμοποιήσει χαρακτηριστικά όπως, οι ομάδες ελέγχου (control groups), για να δημιουργήσει δοχεία.

2.5 Ασφάλεια στο Docker

Οι εφαρμογές σε δοχεία έχουν ένα εγγενές επίπεδο ασφαλείας, αφού μπορούν να εκτελούνται ως απομονωμένες διεργασίες και να λειτουργούν ανεξάρτητα από τα υπόλοιπα δοχεία. Αν εξασφαλιζόταν πλήρης απομόνωση, θα μπορούσε να αποτραπεί, στην περίπτωση μόλυνσης από κακόβουλο λογισμικό, ο κίνδυνος να επηρεαστούν άλλα δοχεία ή το ίδιο το σύστημα. Ωστόσο, η απομόνωση τους δεν είναι πλήρης. Επομένως, ενώ ο διαμοιρασμός κομματιών μιας εφαρμογής σε δοχεία βοηθάει στην αποδοτικότητα του συστήματος, ανοίγει και ένα παράθυρο ευκαιρίας για επιθέσεις λόγω αυτής της τρωτότητας. Το γεγονός, επίσης, πως μοιράζονται τον ίδιο πυρήνα σημαίνει πως μια επίθεση με στόχο αυτόν, μπορεί δυνητικά να επηρεάσει όλα τα δοχεία.

Σχετικά με τις εικόνες δοχείων που αναφέρθηκαν στο 1.2 και το 2.4, τα κομμάτια δηλαδή από τα οποία μια εφαρμογή σε μορφή δοχείου αποτελείται και αντιστοιχούν σε καλούπια μέσω των οποίων παράγονται τα δοχεία της εφαρμογής, η ασφάλεια δεν είναι πάντα εγγυημένη. Αυτό είναι κάτι που συμβαίνει διότι ο καθένας έχει την δυνατότητα να ανεβάσει μια εικόνα δοχείου προς χρήση από τρίτους. Σε περίπτωση που δεν εξετασθεί το περιεχόμενό της μπορεί είτε να περιέχει κακόβουλο λογισμικό, είτε να μην ακολουθούνται ορθές πρακτικές ασφαλείας με αποτέλεσμα να μένει το σύστημα που την χρησιμοποιεί ευάλωτο σε επιθέσεις. Συνεπώς, πρέπει να ληφθούν μέτρα προστασίας όπως η χρήση εικόνων προερχόμενες μόνο από εγκεκριμένες πηγές, δηλαδή να υπάρχει εμπιστοσύνη ανάμεσα στον προμηθευτή μιας εικόνας δοχείου και τον τελικό χρήστη. Επιπροσθέτως, πριν την χρήση μιας εικόνας δοχείου, πρέπει αυτή να εξετάζεται με εργαλεία ανίχνευσης τρωτοτήτων (όπως το Snyk[13] ή το Trivy[14]), καθώς και να έχει πραγματοποιηθεί επαρκώς σκλήρυνση του Docker ώστε να μειωθούν οι επιπτώσεις κατά την χρήση της εαν περιέχει κακόβουλο λογισμικό.

[13]Snyk. Snyk. URL: `https://snyk.io/`.
[14]A. Security. Trivy. URL: `https://aquasecurity.github.io/trivy/v0.49/`.

Οι πάροχοι τεχνολογίας δοχείων έχουν αποκτήσει μια προσέγγιση ασφαλούς σχεδιασμού ώστε πολλά από τα απαραίτητα μέτρα να είναι ενεργοποιημένα χωρίς την απαίτηση επιπρόσθετης αλληλεπίδρασης από τον χρήστη. Πλέον, η μηχανή δοχείων υποστηρίζει όλες τις ιδιότητες απομόνωσης που υποστηρίζει και το λειτουργικό σύστημα στο οποίο εκτελείται, ενώ άδειες ασφαλείας μπορούν να δοθούν στα δοχεία και τον δαίμονα του Docker μέσω του πυρήνα του ΛΣ, με σκοπό τον επιπρόσθετο περιορισμό χρήσης πόρων και του εύρους του συστήματος στο οποίο έχει πρόσβαση η μηχανή δοχείων [97].

2.5.1 Μετριασμός επιθέσεων στο Docker με χρήση χώρων ονομάτων

Με βάση το μοντέλο που περιγράφει το [105], όπως αναφέρεται στο [106], έχουμε ένα μηχάνημα στο οποίο εκτελούνται διάφορα δοχεία, από τα οποία ένα υποσύνολο έχει τεθεί υπό τον έλεγχο ενός κακόβουλου χρήστη. Για να προστατευτούμε από επιθέσεις, όπως άρνηση υπηρεσίας και κλιμάκωση δικαιωμάτων, πρέπει να πραγματοποιηθεί απομόνωση των διεργασιών, των αρχείων της συσκευής, του IPC, του δικτύου και των πόρων.

Αυτά επιτυγχάνονται κατά σειρά με τη χρήση:

- **Χώρων ονομάτων (namespaces)** όπου οριοθετείται η ορατότητα των διεργασιών ανάμεσα στα δοχεία και το σύστημα, όπως επίσης και τα δικαιώματά τους.
- **Mount namespaces**, με τα οποία οι διεργασίες κάθε δοχείου βλέπουν με διαφορετικό τρόπο τη διάταξη των αρχείων του συστήματος. Όλες οι δράσεις mount, που γίνονται στο εκάστοτε δοχείο, περιορίζονται σε αυτό. Επίσης, οριοθετούνται τα δικαιώματα των αρχείων του πυρήνα σε μονάχα ανάγνωσης και αποτρέπονται περαιτέρω remountings.
- **Device Whitelist Controller**, ενός χαρακτηριστικού [107] των cgroups για περιορισμό συσκευών, με τη βοήθεια του οποίου περιορίζεται το σύνολο τον συσκευών στις οποίες έχει πρόσβαση ένα δοχείο και το αποτρέπει από το να δημιουργήσει καινούριες αναπαραστάσεις συσκευών. Επιπροσθέτως, επειδή τα mount γίνονται με τη χρήση της παραμέτρου nodev[15], στην περίπτωση που μια αναπαράσταση συσκευής είχε δημιουργηθεί σε προηγούμενο χρόνο μέσα στην εικόνα δοχείου που χρησιμοποιήθηκε για να

[15]Μια παράμετρος της εντολής mount κατά την οποία αποτρέπεται η δημιουργία και η πρόσβαση αναπαραστάσεων συσκευών που βρίσκονται στον φάκελο /dev.

κατασκευαστεί το δοχείο, οι διεργασίες του δοχείου αυτού δε δύνανται να τη χρησιμοποιήσουν για να επικοινωνήσουν με τον πυρήνα. Επιπλέον, επειδή η προεπιλεγμένη συνθήκη είναι να μη δίνονται εκτεταμένα προνόμια σε ένα δοχείο, δεν υπάρχει πρόσβαση σε καμία συσκευή παρά μόνο εάν γίνει εκτέλεση δοχείου ως χρήστης με ανώτατα δικαιώματα, όπου τότε υπάρχει πρόσβαση σε όλες (τις συσκευές).

- **IPC namespaces** για περιορισμό IPC, δηλαδή της επικοινωνίας των διεργασιών μεταξύ τους. Τα IPC namespaces καθιστούν δυνατή τη δημιουργία ξεχωριστών συνόλων των διεργασιών που επικοινωνούν μεταξύ τους, αλλά όχι με άλλες διεργασίες πέραν του υποσυνόλου.

- **Network namespaces**. Μία από τις σημαντικότερες απομονώσεις είναι αυτή του δικτύου. Χωρίς δικτυακή απομόνωση υπάρχει κίνδυνος επιθέσεων, όπως ενδιάμεσου (Man in the middle), ARP, DNS πλαστογράφησης (spoofing) και άλλες. Για να απομονωθεί η κίνηση δικτύου που λαμβάνει μέρος σε ένα δοχείο από αυτήν των υπολοίπων και του συστήματος πρέπει να γίνει χρήση των network namespaces. Κάθε δοχείο θα έχει δικές του διευθύνσεις IP, συσκευές και ό,τι χρειάζεται, προκειμένου να γίνεται αλληλεπίδραση μεταξύ των δοχείων μέσω της διεπαφής δικτύου του καθενός σαν να είναι εξωτερικές οντότητες.

- **Ομάδες ελέγχου (cgroups)**. Επιβάλλεται η οριοθέτηση των υπολογιστικών πόρων προκειμένου να αποφευχθεί μια επίθεση τύπου άρνησης υπηρεσίας, όπου μια διεργασία ή ένα σύνολο αυτών προσπαθεί να καταναλώσει όλους τους πόρους του συστήματος. Με τη βοήθεια των ομάδων ελέγχου, επιτυγχάνεται ο έλεγχος του (μέγιστου) ποσοστού πόρων όπως CPU, μνήμης και χωρητικότητας, που μπορεί να έχει κάθε δοχείο στη διάθεση του.

Αυτές οι δυνατότητες υποστηρίζονται από το Docker και μπορεί κανείς να τις εκμεταλλευτεί για να προστατεύσει το περιβάλλον του από επιθέσεις. Επιπλέον, υπάρχει και η δυνατότητα υποστήριξης Kernel Security Modules, όπως τα SELinux [108] και AppArmor[16] αλλά και του seccomp [110] (στην περίπτωση χρήσης LXC), καθώς επίσης και συμβατότητα με Linux capabilities (ικανότητες), που θα μπορούσαν να εισάγουν ένα ακόμα επίπεδο ασφαλείας, αν χρησιμοποιηθούν σωστά, περιορίζοντας τα δικαιώματα των διεργασιών των δοχείων σε μονάχα όσα χρειάζονται. Το Docker παρέχει αρκετά υπάρχοντα μέσα άμυνας προκειμένου να προστατευτεί από επιθέσεις ακόμα και χωρίς επιπρόσθετες ρυθμίσεις. Παρ' όλα αυτά, οι αρχικές

[16]AppArmor. AppArmor. URL: `https://apparmor.net/`.

ρυθμίσεις ασφαλείας του είναι πιο ελαστικές απ' όσο χρειάζεται προκειμένου να συνεχίζει να λειτουργεί κανονικά για όλους τους χρήστες, αφήνοντας έτσι τους διαχειριστές ασφαλείας υπεύθυνους να χρησιμοποιήσουν όσες δυνατότητες είναι απαραίτητες προκειμένου να ανταπεξέλθουν σε κάθε επίθεση ανάλογα με το περιβάλλον και τις ανάγκες τους.

2.5.2 Συχνά είδη επιθέσεων σε δοχεία και μέθοδοι πρόληψής τους

Μερικά είδη επιθέσεων σε δοχεία με τους τρόπους αντιμετώπισής τους, όπως αναφέρονται στο [111], είναι τα εξής:

- **Εκμεταλλεύσεις πυρήνα (Kernel Exploits)**:

  Ο πυρήνας είναι υπεύθυνος για τη διαχείριση των λειτουργιών και διεργασιών των δοχείων. Λόγω του διαμοιρασμού του πυρήνα του συστήματος με κάθε δοχείο το οποίο εκτελείται σε αυτό, εάν οποιοδήποτε από αυτά τεθεί υπό τον έλεγχο ενός κακόβουλου χρήστη, αυτός μπορεί δυνητικά να πάρει τον έλεγχο του συστήματος και όλων των δοχείων αυτού. Ορισμένοι τρόποι με τους οποίους αυτό έχει επιτευχθεί στο παρελθόν, ήταν μέσω εκμετάλλευσης μιας ευπάθειας των ομάδων ελέγχου [112] και μέσω μιας ευπάθειας ονόματι "Dirty Pipe" [113]. Η μέθοδος αντιμετώπισης που προτείνεται είναι η χρήση SELinux ή AppArmor κατά την εκτέλεση δοχείων σε συνδυασμό με εκμετάλλευση των user namespaces. Επιπλέον, συνιστάται να μην εκτελούνται εφαρμογές με δικαιώματα διαχειριστικού λογαριασμού μέσα σε δοχεία. Υπάρχουν και άλλα μέτρα προστασίας που μπορεί να παρθούν αλλά πιθανόν να μην είναι εφικτά διότι ενδέχεται να πηγαίνουν ενάντια στις λειτουργίες του εκάστοτε δοχείου. Αυτά είναι, να τεθεί το σύστημα του σε επίπεδο πρόσβασης που επιτρέπει μόνο την ανάγνωση των αρχείων, να απενεργοποιηθεί η επικοινωνία μεταξύ δοχείων και να αποφευχθεί η εγκατάσταση περιττών προγραμμάτων σε αυτά.

- **Άρνηση υπηρεσίας**:

  Μια από τις πιο συνηθισμένες επιθέσεις σε πόρους διαθέσιμους μέσω δικτύου είναι η άρνηση υπηρεσίας. Κατά τη διάρκεια μιας τέτοιας επίθεσης, μια διεργασία ή ένα σύνολο διεργασιών επιχειρεί να καταναλώσει όλους τους πόρους του συστήματος προκειμένου να μην μπορεί να εξυπηρετήσει άλλους χρήστες. Αυτό μπορεί να συμβεί εάν ένα δοχείο βρεθεί υπό τον έλεγχο ενός επιτιθέμενου και επιχειρήσει να διεκδικήσει πόρους που κανονικά δε χρειάζεται. Για να ανταπεξέλθει ένα σύστημα σε μια επίθεση άρνησης υπηρεσίας, πρέπει να γίνει χρήση των δυνατοτήτων που αναφέραμε στο 2.5.1 με σκοπό

τον αυστηρότερο έλεγχο διαμοιρασμού των πόρων του συστήματος. Με τον καθορισμό μέγιστων διαθέσιμων πόρων για κάθε δοχείο εξ αρχής, δεν υπάρχει κίνδυνος να προσπαθήσει κάποιο δοχείο να διεκδικήσει περισσότερους από αυτούς που μπορούν να του ανατεθούν.

- **Αποδράσεις Δοχείων (Container Breakouts)**:

  Σε τέτοιου είδους επιθέσεις, ένας επιτιθέμενος προσπαθεί αφότου απέκτησε πρόσβαση σε ένα δοχείο, να καταφέρει μέσω αυτού να έχει πρόσβαση στα αρχεία του κύριου συστήματος. Αυτό μπορεί να συμβεί με τη χρήση της συνάρτησης "open_by_handle_at()" που απαιτεί δικαιώματα διαχειριστικού λογαριασμού μέσα από το δοχείο, προκειμένου να κάνει κλήση της ικανότητας (capability) "CAP_DAC_READ_SEARCH" στην οποία είχε πρόσβαση εξ αρχής [111]. Σύμφωνα με το Docker, η μόνη έκδοσή του που μπορούσε να επηρεαστεί από αυτή την ευπάθεια ήταν η 0.11 και στην επόμενη διορθώθηκε. Για την πρόληψη μελλοντικών μεθόδων διεκπεραίωσης τέτοιου είδους επίθεσης, συνιστάται να τίθενται τα δοχεία και οι αποθηκευτικοί τους χώροι σε κατάσταση μονάχα ανάγνωσης, να αποφεύγεται η χρήση διαχειριστικού λογαριασμού μέσα τα δοχεία, καθώς και να αποφεύγεται η χρήση της παραμέτρου "privileged".

- **Δηλητηριασμένες εικόνες δοχείων**:

  Οι εικόνες δοχείων μπορεί να περιέχουν κακόβουλο λογισμικό ή λογισμικό για το οποίο έχουν βρεθεί (πλέον) ευπάθειες. Ο τωρινός τρόπος ελέγχου εγκυρότητάς τους βασίζεται μονάχα στην παρουσία ενός υπογεγραμμένου manifest αλλά δε γίνεται ποτέ αυθεντικοποίηση του αθροίσματος ελέγχου (checksum) της κάθε εικόνας από το σύστημα. Αυτό αφήνει ανοιχτό το ενδεχόμενο ένας επιτιθέμενος να διαδώσει οποιαδήποτε εικόνα μαζί με το υπογεγραμμένο manifest της. Επιβάλλεται λοιπόν, οι χρήστες να κατεβάζουν εικόνες από εγκεκριμένους προμηθευτές και επιπρόσθετα να τις ελέγχουν με κατάλληλα εργαλεία ανίχνευσης τρωτοτήτων προτού τις χρησιμοποιήσουν. Τέλος, πρέπει οι ίδιοι να ελέγχουν το checksum ώστε να εξακριβώνεται πριν την χρήση αν είναι όντως η εικόνα που ζητήθηκε.

- **Απόκτηση μυστικών κωδικών/κλειδιών**:

  Η απόκτηση επιχειρησιακών ή προσωπικών μυστικών ελλοχεύει πολλούς κινδύνους για μια επιχείρηση. Σε περίπτωση που κάτι τέτοιο συμβεί, ένας επιτιθέμενος μπορεί να έχει πρόσβαση σε βάσεις δεδομένων ή άλλα κομμάτια του συστήματος απειλώντας έτσι την ακεραιότητα, την εμπιστευτικότητα και διαθεσιμότητα των δεδομένων. Προκειμένου να

αποφευχθεί η εισβολή σε κάποιο δοχείο που θα προκαλούσε την απόκτηση τέτοιων μυστικών, καθίσταται επιτακτική η χρήση δοχείων σε κατάσταση ανάγνωσης και όχι εγγραφής αλλά και η αποφυγή χρήσης μεταβλητών για την αποθήκευση κωδικών. Χρήσιμη επίσης θα ήταν και η εσκεμμένη παράλειψη της παραμέτρου "privileged".

- **Ενδιάμεσου (Man-in-the-Middle - MitM)**:

  Σε μια τέτοια επίθεση ένας κακόβουλος χρήστης προσπαθεί να μπει ανάμεσα στην επικοινωνία δύο οντοτήτων με σκοπό να αλλοιώσει, να υποκλέψει ή να παρακολουθεί πληροφορίες. Με βάση το CVE-2020-13401 που αναφέρεται στο [114], μετά από παραβίαση ενός δοχείου, εάν αυτό εκτελείται με την ικανότητα "CAP_NET_RAW", τότε μπορεί ένας επιτιθέμενος να δημιουργήσει διαφημίσεις δρομολογητών IPv6 και κατά συνέπεια, να παραποιήσει εξωτερικούς κεντρικούς υπολογιστές IPv6, ώστε να αποκτήσει ευαίσθητες πληροφορίες ή να προκαλέσει άρνηση παροχής υπηρεσιών. Το καλύτερο αντίμετρο αυτής της επίθεσης είναι η απομόνωση δικτύου. Πρέπει να γίνει ρύθμιση των δοχείων με τέτοιο τρόπο ώστε αυτά να μην έχουν πρόσβαση στη δικτυακή επικοινωνία του κύριου μηχανήματος ή άλλων δοχείων. Αυτά επιτυγχάνονται με χρήση network namespaces, καθώς και με την ορθότερη ρύθμιση των API που χρησιμοποιεί το Docker για την επικοινωνία μέσω δικτύου. Τέλος, πρέπει να χρησιμοποιούνται ασφαλή πρωτόκολλα επικοινωνίας με αμφίδρομη αυθεντικοποίηση.

- **Πλαστογράφηση ARP (ARP spoofing)**:

  Σε μια επίθεση πλαστογράφησης Address Resource Protocol (ARP), ο επιτιθέμενος επιχειρεί να στείλει ψεύτικα ARP μηνύματα σε ένα δίκτυο τοπικής περιοχής (Local Area Network - LAN). Με αυτόν τον τρόπο, είναι πιθανό να προσποιηθεί πως η συσκευή του είναι μια από τις συσκευές της επιχείρησης, αφού πλέον η MAC διεύθυνσή του, αντιστοιχεί στην MAC μιας συσκευής της οποίας η IP αναγνωρίζεται από το σύστημα [115]. Επομένως, το σύστημα της επιχείρησης, θα συμπεριφέρεται στην συσκευή του επιτιθέμενου με τον ίδιο τρόπο που θα συμπεριφερόταν στην αυθεντική συσκευή. Δηλαδή, στέλνοντας σε αυτόν όλα τα πακέτα και τα δεδομένα που προορίζονταν για εκείνη. Το Docker χρησιμοποιεί το ARP για να κάνει την αντιστοίχιση IPv4 σε MAC διευθύνσεις, οι οποίες χρησιμοποιούνται από την εικονική γέφυρα δικτύου για να διανέμουν σωστά τα πλαίσια (frames) Ethernet, καθώς δεν υπάρχει φίλτρο για τα πακέτα ARP και επομένως κανένας μηχανισμός άμυνας. Γι' αυτό τον λόγο ένα δοχείο μπορεί να προσποιηθεί ότι είναι ένα άλλο ή ακόμα και το κύριο μηχάνημα. Ακόμα και χωρίς την παραβίαση ενός δοχείου, αλλά έχοντας πρόσβαση στο τοπικό δίκτυο, υπάρχει κίνδυνος ο επιτιθέμενος να υποκλέψει μυστικά της επιχείρησης ή των τελικών χρηστών της υπηρεσίας που η επιχείρηση προσφέρει. Ένας από τους τρόπους αποφυγής μιας τέτοιας επίθεσης, είναι η

εκτέλεση δοχείων δίχως την ικανότητα “NET_RAW”, αφού έτσι τα προγράμματα μέσα σε ένα δοχείο δε θα μπορούν να δημιουργήσουν PF_PACKET sockets και θα ήταν αδύνατη η διεκπεραίωση της επίθεσης. Βέβαια, αυτή η μέθοδος έχει και μειονεκτήματα διότι αυτή η ικανότητα μπορεί να ήταν άκρως απαραίτητη για την ορθή λειτουργία της υπηρεσίας. Επομένως, μια εναλλακτική μέθοδος προστασίας είναι η χρήση “ebtables” για φιλτράρισμα πλαισίων Ethernet, ούτως ώστε ARP πακέτα με λάθος πρωτόκολλο αποστολέα ή διεύθυνση υλικού να ανιχνεύονται εγκαίρως και να απορρίπτονται.

Κεφάλαιο 3. Σχετικές Εργασίες

Με την πάροδο των χρόνων και τη ραγδαία αύξηση ενδιαφέροντος προς τον κλάδο της υπολογιστικής νέφους, άρχισε να γίνεται όλο και πιο επιτακτική η ανάγκη για αυτοματοποιημένο στήσιμο και προστασία των υποδομών των επιχειρήσεων. Η χρήση τεχνολογιών εικονικοποίησης και η άνοδος των υπηρεσιών IaaS, έχει ελαφρύνει τον φόρτο εργασίας όσον αφορά το στήσιμο των υποδομών αυτών, αφού πλέον υπάρχουν πολλά εργαλεία στα οποία γίνεται καθορισμός τους. Πέραν από εργαλεία στα οποία δηλώνονται οι προδιαγραφές μιας εικονικής μηχανής σε πραγματικό χρόνο με την χρήση παραμέτρων, υπάρχουν και άλλα που ανήκουν στην οικογένεια εργαλείων IaC (Infrastructure as Code) (Υποδομή ως Κώδικας). Σε αυτού του είδους τα εργαλεία, όλα τα χαρακτηριστικά μιας εικονικής μηχανής δηλώνονται σε ένα αρχείο κειμένου και με βάση αυτό, ξεκινάει η διαδικασία δημιουργίας τους. Η αυτοματοποίηση της ασφάλισης αυτών των εικονικών μηχανών όμως, είναι ακόμα σε πρώιμο στάδιο. Οι περισσότερες υπηρεσίες που διατίθενται στους χρήστες κάνουν χρήση τεχνολογιών εικονικοποίησης έμμεσα από τον πάροχο νέφους που χρησιμοποιούν και τεχνολογίες δοχείων άμεσα από επιλογή τους λόγω της υπεροχής που προσφέρουν στη διαχείριση και τη διασφάλιση των εφαρμογών τους. Όλες οι υπηρεσίες όμως, είναι εν δυνάμει ευάλωτες σε επιθέσεις εάν δε ληφθούν τα απαραίτητα μέτρα προστασίας και αφήσουν ανοιχτά σημεία εισόδου στα συστήματά τους.

Η παρούσα εργασία έχει ως στόχο την ανάπτυξη ενός εργαλείου που επιτυγχάνει την αυτοματοποίηση τριών τομέων. Την επικοινωνία με παρόχους νέφους για την αίτηση διάθεσης υπολογιστικών πόρων, την ασφάλιση των πόρων αυτών και τέλος, την εγκατάσταση, ασφάλιση και χρήση του Docker για τη διάθεση εφαρμογών που βρίσκονται στο Docker Hub, δηλ. το επίσημο αποθετήριο εικόνων δοχείων του Docker. Στην ενότητα αυτή θα γίνει αναφορά σε εργαλεία που έχουν αναπτυχθεί για έναν ή παραπάνω από αυτούς τους τρεις τομείς και θα πραγματοποιηθεί σύγκρισή τους ανάλογα με τον τρόπο λειτουργίας τους και τις δυνατότητες που παρέχουν.

3.1 Αυτοματοποίηση δημιουργίας εικονικών μηχανών

Σε σχέση με τα εργαλεία αυτοματοποίησης δημιουργίας εικονικών μηχανών, τα κριτήρια που θα ληφθούν υπόψιν για την σύγκριση με το εργαλείο που προτείνει η διπλωματική εργασία είναι τα εξής:

- **Ευελιξία κατά την χρήση**:

  Το εργαλείο πρέπει να μπορεί να προσαρμόζεται στις ανάγκες του χρήστη όταν αυτός χρειάζεται να πραγματοποιήσει ορισμένες παραμετροποιήσεις κατά την χρήση του.

- **Ευκολία κατά την χρήση**:

  Το εργαλείο πρέπει να μπορεί να χρησιμοποιηθεί εύκολα από έναν χρήστη που έχει χρησιμοποιήσει στο παρελθόν εργαλεία γραμμής εντολών. Οι παράμετροί του πρέπει να ακολουθούν ένα μοτίβο που θα διευκολύνει την κατανόηση της σημασιολογίας τους και θα επιτρέπει τον συνδυασμό τους για ένα επιθυμητό αποτέλεσμα.

- **Υποστήριξη μεγάλων ονομάτων στον κλάδο**:

  Το εργαλείο πρέπει να υποστηρίζει την επικοινωνία με τα νέφη μεγάλων εταιρειών στον κλάδο της νεφο-υπολογιστικής, τα οποία κατέχουν μεγάλο μερίδιο της σχετικής αγοράς, ώστε να είναι εύκολο για να έναν χρήστη να παραμείνει στον πάροχο της επιλογής του.

- **Ευκολία στην επέκταση**:

  Το εργαλείο πρέπει να είναι εύκολο στην επέκτασή του, ώστε να μπορεί να αυτοματοποιηθεί περισσότερο η διαδικασία δημιουργίας εικονικών μηχανών.

3.1.1 Εργαλεία δημιουργίας εικονικών μηχανών

Τα εργαλεία με τα οποία θα γίνει σύγκριση στον τομέα δημιουργίας εικονικών μηχανών, περιγράφονται στην συνέχεια ως εξής:

- **Terraform**:

  Το Terraform είναι ένα δημοφιλές εργαλείο ανοιχτού κώδικα που αναπτύχθηκε από την εταιρεία HashiCorp. Αποτελεί ένα εργαλείο, κατά την χρήση, του οποίου ο χρήστης καθορίζει τα χαρακτηριστικά της υποδομής που θέλει να δημιουργήσει σε ένα αρχείο που ακολουθεί συγκεκριμένη σύνταξη. Αυτό το χαρακτηριστικό, εντάσσει το Terraform στην κατηγορία εργαλείων "Υποδομή ως Κώδικας (Infrastructure as Code - IaC)" και χρησιμοποιείται για τον καθορισμό των υποδομών που θα χρησιμοποιηθούν για την στέγαση εφαρμογών. Το Terraform υποστηρίζει πολλούς παρόχους νέφους και επιτρέπει την αυτοματοποίηση της διαδικασίας δημιουργίας υποδομών σε αυτούς. Ο τρόπος λειτουργίας του είναι αρκετά απλός. Αφού ο χρήστης έχει δημιουργήσει το αρχείο που

περιγράφει την υποδομή, το εκτελεί για να γίνουν οι απαραίτητες ενέργειες. Έπειτα, δημιουργούνται οι εικονικές μηχανές που περιγράφονται στο αρχείο. Ο χρήστης μπορεί να αποδεσμεύσει τους πόρους που δημιουργήθηκαν όταν δεν τους χρειάζεται πλέον και η διαδικασία της αποδέσμευσης, είναι όσο απλή όσο και η δημιουργία τους.

- **Libcloud CLI**[1]:

  Το Libcloud CLI, αποτελεί την δεύτερη τροποποιημένη έκδοση από ένα τρίτο πρόσωπο, ενός προγράμματος που επιχειρούσε να επιτύχει την ανάπτυξη ενός εργαλείου γραμμής εντολών που χρησιμοποιεί την βιβλιοθήκη libcloud για την δημιουργία εικονικών μηχανών κατά μήκος διαφόρων παρόχων νέφους. Είναι ένα εργαλείο γραμμής εντολών που απαιτεί αρχείο ρυθμίσεων μονάχα για την αυθεντικοποίηση με τους παρόχους νέφους.

3.1.2 Σύγκριση με το SecDep

Σχετικά με τα κριτήρια που έχουν οριστεί για την σύγκριση με εργαλεία που ως κύριο στόχο έχουν την δημιουργία εικονικών μηχανών, το εργαλείο που προτείνει η διπλωματική εργασία παρέχει όλα τα απαραίτητα χαρακτηριστικά.

- **Σύγκριση με Terraform**:

  Χρειάζεται την δημιουργία ενός αρχείου πριν την χρήση του, ενώ κάθε αλλαγή απαιτεί την τροποποίηση του αρχείου αυτού. Αυτό το καθιστά λιγότερο ευέλικτο σε σχέση με άλλα εργαλεία. Είναι σχετικά εύκολο στην χρήση όπως τα υπόλοιπα και δύναται να χρησιμοποιήσει τα νέφη όλων των μεγάλων παρόχων. Παρ' όλα αυτά δεν αποτελεί εργαλείο που μπορεί να επεκταθεί εύκολα καθώς ο τρόπος λειτουργίας του το καθιστά αυτοτελές.

- **Σύγκριση με Libcloud CLI**:

  Είναι αρκετά ευέλικτο και εύκολο στην χρήση, παρ' όλα αυτά πλέον υποστηρίζει μονάχα έναν πάροχο νέφους, ο οποίος δεν αποτελεί δημοφιλή επιλογή γενικότερα. Θα μπορούσε να επεκταθεί η λειτουργικότητά του, αλλά έχει να ανανεωθεί από το 2018.

Ο χρήστης δύναται κατά την χρήση του SecDep (μέσω της εκτέλεσης του secdep.py) να επιλέξει τις ενέργειες και τα χαρακτηριστικά που επιθυμεί για την εικονική μηχανή του. Όλες οι παράμετροι είναι εύκολα κατανοητές και συνδυάζονται μεταξύ τους για να επιτευχθεί το επιθυμητό αποτέλεσμα. Υποστηρίζει όλους τους δημοφιλείς παρόχους νέφους και όλες

[1]Terradue. libcloud-cli. URL: `https://github.com/Terradue/libcloud-cli`.

τις καλές πρακτικές ενός εργαλείου γραμμής εντολών, όπως ο έλεγχος σφαλμάτων κατά την εισαγωγή παραμέτρων. Επιπλέον, μπορεί εύκολα να επεκταθεί η χρήση του, είτε εισάγοντας έτοιμες εντολές σε αρχεία bash είτε προσθέτοντας νέες ενέργειες στον κώδικά του, αφού πρόκειται για ένα αρχείο python.

Πίνακας 3.1: Σύγκριση εργαλείων δημιουργίας εικονικών μηχανών

Κριτήρια	Terraform	Libcloud CLI	SecDep
Ευελιξία κατά την χρήση	Όλες οι προδιαγραφές δηλώνονται σε αρχείο κειμένου.	Οι παράμετροι επιλέγονται κατά την δημιουργία της εντολής.	Οι παράμετροι επιλέγονται κατά την δημιουργία της εντολής.
Ευκολία χρήσης	Κάθε αλλαγή απαιτεί τροποποίηση του αρχείου προδιαγραφών.	Οι παράμετροι είναι εύκολες στην κατανόηση και την χρήση.	Οι παράμετροι είναι εύκολες στην κατανόηση και την χρήση.
Εύρος υποστήριξης παρόχων	Όλοι οι μεγάλοι και μικροί πάροχοι υποστηρίζονται.	Μονάχα ένας πάροχος υποστηρίζεται.	Υποστηρίζονται όλοι οι μεγάλοι πάροχοι.
Επεκτασιμότητα	Λόγω της αρχιτεκτονικής του είναι δυσκολότερη η επέκτασή του.	Επεκτείνεται εύκολα με τροποποίηση του εκτελέσιμου αρχείου.	Επεκτείνεται εύκολα με τροποποίηση του εκτελέσιμου αρχείου.

3.2 Αυτοματοποίηση σκλήρυνσης εικονικών μηχανών

Τα κριτήρια που θα ληφθούν υπόψιν για εργαλεία αυτοματοποίησης σκλήρυνσης εικονικών μηχανών είναι τα παρακάτω:

- **Υποστήριξη πολλών διανομών**:

  Το εργαλείο πρέπει να υποστηρίζει διάφορες διανομές Linux και να είναι αρκετά ευέλικτο ώστε να μπορεί να ξεχωρίσει την διανομή στην οποία εκτελείται.

- **Ευκολία κατά την χρήση**:

  Το εργαλείο πρέπει να μπορεί να χρησιμοποιηθεί εύκολα από έναν χρήστη. Ιδανικά, να αρκεί η εκτέλεσή του σε μια εικονική μηχανή χωρίς την ανάγκη εγκατάστασης εξαρτήσεων.

- **Εύρος σκλήρυνσης**:

  Το εργαλείο πρέπει να μπορεί να σκληρύνει την εικονική μηχανή σε ικανοποιητικό βαθμό και να μην περιορίζεται σε λίγες μονάχα δυνατότητες.

- **Υποστήριξη διαρκούς σκλήρυνσης**:

  Το εργαλείο πρέπει να παρέχει την δυνατότητα διαρκούς σκλήρυνσης, προσφέροντας μηχανισμούς που θα εκτελούνται περιοδικά με σκοπό την διατήρησή της.

3.2.1 Εργαλεία σκλήρυνσης εικονικών μηχανών

Τα παρακάτω εργαλεία θα συγκριθούν με το harden. Με άλλα λόγια, το δεύτερο εκτελέσιμο αρχείο του εργαλείου SecDep, το οποίο μετά από επιλογή του χρήστη εκτελείται σε μια εικονική μηχανή αφότου αυτή δημιουργηθεί.

- **JShielder**[2]:

  Το JShielder είναι ένα εργαλείο ανοιχτού κώδικα που αναπτύχθηκε από τον Jason Soto με σκοπό την αυτοματοποίηση σκλήρυνσης λειτουργικών συστημάτων Linux. Ο πηγαίος κώδικάς του βρίσκεται στο GitHub και ο τρόπος λειτουργίας του είναι η εκτέλεσή του ως χρήστης με διαχειριστικά δικαιώματα στο σύστημα.

- **nixarmor**[3]:

  Το nixarmor, αποτελεί και αυτό ένα εργαλείο ανοιχτού κώδικα, το οποίο στεγάζεται στο GitHub και αναπτύχθηκε από τον Emir Ozer. Περιέχει διαφορετικά εκτελέσιμα αρχεία για την σκλήρυνση διάφορων διανομών Linux, τα οποία μπορούν να εκτελεστούν ως αυτόνομα προγράμματα από τον χρήστη.

3.2.2 Σύγκριση με τον τομέα σκλήρυνσης VM του harden

Σχετικά με τα δηλωθέντα κριτήρια, το SecDep (μέσω του harden) ικανοποιεί όλους τους παραπάνω στόχους.

[2]Jsitech. JShielder. URL: `https://github.com/Jsitech/JShielder`.
[3]E. Ozer. nixarmor. URL: `https://github.com/emirozer/nixarmor`.

- **Σύγκριση με JShielder**:

  Το JShielder παρέχει ένα ικανοποιητικό εύρος σκλήρυνσης του συστήματος και η ευκολία στην χρήση του το καθιστά ένα εργαλείο που μπορεί να χρησιμοποιηθεί από έναν χρήστη με μικρή εμπειρία στον κλάδο. Παρ' όλα αυτά, δεν υποστηρίζει πολλές διανομές και οι εκδόσεις των διανομών που υποστηρίζονται μέσω αυτού, είναι αρκετό καιρό ξεπερασμένες. Θα μπορούσε να βελτιωθεί με μια ανανέωση του πηγαίου του κώδικα αλλά η ανάπτυξή του φαίνεται να έχει σταματήσει το 2019.

- **Σύγκριση με το nixarmor**:

  Σε αντίθεση με το JShielder, το nixarmor, για να χρησιμοποιηθεί με τον ίδιο τρόπο που χρησιμοποιούνται τα υπόλοιπα εργαλεία, θα πρέπει να επεκταθεί με την προσθήκη κώδικα που να του επιτρέπει να ξεχωρίζει ποιο είναι το κατάλληλο εκτελέσιμο για την κάθε διανομή. Αν αγνοήσουμε το μειονέκτημα αυτό, μπορούμε να χαρακτηρίσουμε τη χρήση του εύκολη για χρήστες που έχουν την εμπειρία να διακρίνουν με κάποιο τρόπο την τρέχουσα διανομή και άρα είναι σε θέση να επιλέξουν και να εκτελέσουν χειροκίνητα ένα από τα εκτελέσιμα αρχεία που το nixarmor περιέχει. Η σκλήρυνση που παρέχει καλύπτει πολλούς τομείς μιας διανομής και μπορούν εύκολα να προστεθούν επιπλέον συναρτήσεις στον κώδικά του. Παρ' όλα αυτά, η ανάπτυξή του έχει παύσει από το 2015 και οι υποστηριζόμενες διανομές του έχουν περιοριστεί σε μονάχα 4.

Συγκριτικά με τα παραπάνω εργαλεία, το harden υποστηρίζει την ικανότητα αναγνώρισης της διανομής στην οποία εκτελείται και μπορεί δυναμικά να διαμορφώσει την συμπεριφορά του αναλόγως. Οι υποστηριζόμενες διανομές είναι 6 και οι λειτουργίες του, αν και σύμφωνα με τα αποτελέσματα του Κεφαλαίου 6 είναι ικανοποιητικές, θα μπορούσαν εύκολα να επεκταθούν με την προσθήκη νέων συναρτήσεων. Επιπλέον, αν και το harden αποτελεί ένα εκτελέσιμο, παράγει κατά την εκτέλεσή του δύο ακόμα εκτελέσιμα, τα οποία μπορούν να εκτελούνται περιοδικά με σκοπό την αυτόματη ενημέρωση του συστήματος και το κλείσιμο αχρησιμοποίητων θυρών. Αυτό σημαίνει πως η σκλήρυνση μπορεί να είναι διαρκής και όχι στατική, εκτελούμενη μόνο μια φορά.

Πίνακας 3.2: Σύγκριση εργαλείων σκλήρυνσης εικονικών μηχανών

Κριτήρια	JShielder	nixarmor	SecDep
Εύρος υποστήριξης διανομών	Υποστηρίζεται μια διανομή με δύο εκδόσεις.	Υποστηρίζονται 4 διανομές.	Υποστηρίζονται 6 διανομές με πολλαπλές εκδόσεις τους.
Ευκολία χρήσης	Αρκεί η εκτέλεσή του με διαχειριστικά δικαιώματα.	Αρκεί η εκτέλεση ενός εκτελέσιμου αρχείου του με διαχειριστικά δικαιώματα.	Αρκεί η εκτέλεσή του με διαχειριστικά δικαιώματα.
Εύρος σκλήρυνσης	Παρέχεται ικανοποιητικό εύρος σκλήρυνσης, καλύπτοντας πολλούς τομείς.	Το εύρος σκλήρυνσης του συστήματος είναι μικρότερο από τα άλλα εργαλεία.	Η σκλήρυνση του συστήματος επιτυγχάνεται σε ικανοποιητικό βαθμό.
Υποστήριξη διαρκούς σκλήρυνσης	Υποστηρίζεται η διαρκής ενημέρωση πακέτων.	Υποστηρίζονται οι διαρκείς ενημερώσεις ασφαλείας σε μια διανομή.	Υποστηρίζεται η δημιουργία δύο περιοδικά εκτελέσιμων αρχείων για την ενημέρωση των πακέτων κάθε υποστηριζόμενης διανομής και το κλείσιμο των αχρησιμοποίητων θυρών.

3.3 Αυτοματοποίηση εγκατάστασης/σκλήρυνσης του Docker

Για τα εργαλεία αυτοματοποίησης της διαδικασίας εγκατάστασης/σκλήρυνσης του Docker, τα κριτήρια που θα πρέπει να ικανοποιούνται είναι τα παρακάτω:

- **Ευκολία κατά την χρήση**:

  Ιδανικά, θα πρέπει να αρκεί η εκτέλεση του εργαλείου με διαχειριστικά δικαιώματα, δίχως την ανάγκη περαιτέρω ενεργειών από τον χρήστη.

- **Εύρος σκλήρυνσης**:

  Αναμένεται η σκλήρυνση του Docker να επιτευχθεί σε ικανοποιητικό βαθμό, πραγματοποιώντας τις απαραίτητες ρυθμίσεις στον δαίμονά του.

- **Υποστήριξη διαρκούς σκλήρυνσης**:

  Το εργαλείο πρέπει να πραγματοποιεί βήματα που θα διασφαλίσουν την διαρκή σκλήρυνση του Docker ή/και των δοχείων του.

- **Επεκτασιμότητα**:

  Θα πρέπει να μπορεί ένας έμπειρος χρήστης να προσθέσει νέες λειτουργίες με σκοπό την επέκταση των δυνατοτήτων του.

3.3.1 Εργαλεία εγκατάστασης/σκλήρυνσης του Docker

Μια από τις λειτουργίες του εκτελέσιμου harden πέρα από την σκλήρυνση του συστήματος, είναι και η αυτόματη εγκατάσταση και σκλήρυνση του Docker. Τα εργαλεία που αποσκοπούν στην αυτοματοποίηση της διαδικασίας αυτής, τα οποία θα συγκριθούν με το harden σε αυτόν τον τομέα, είναι τα παρακάτω:

- **docker-rootless-setup**[4]:

  Το docker-rootless-setup είναι ένα εργαλείο ανοιχτού κώδικα, το οποίο στεγάζεται στο GitHub. Θεωρείται πως ανήκει στην κατηγορία εργαλείων σκλήρυνσης του Docker διότι αποσκοπεί στην αυτοματοποίηση της εγκατάστασης του Rootless Docker. Το αποτέλεσμα μετά την εγκατάστασή αυτού, είναι να μπορεί πλέον το Docker να λειτουργεί χωρίς

[4]zerint. Docker-Rootless Full Setup. URL: `https://github.com/zerint/docker-rootless-setup`.

την ανάγκη διαχειριστικών δικαιωμάτων, αυξάνοντας έτσι την ασφάλεια του συστήματος. Για την χρήση του αρκεί μονάχα η εκτέλεσή του στο σύστημα που θέλουμε να εγκαταστήσουμε το Rootless Docker. Η επέκτασή του είναι εύκολα εφικτή, μιας και πρόκειται για ένα εκτελέσιμο αρχείο bash.

- **docksec**[5]:

  Το docksec είναι ένα ακόμα εργαλείο ανοιχτού κώδικα, το οποίο επιχειρεί να ασφαλίσει τον δαίμονα του Docker. Ο τρόπος με τον οποίο γίνεται αυτό, είναι με την διασφάλιση ορισμένων ενεργειών που συστήνει το docker-bench-security[6]. Το docker-bench-security είναι ένα εργαλείο που στεγάζεται στο GitHub υπό τον επίσημο λογαριασμό της ομάδας ανάπτυξης του Docker. Χρησιμοποιείται για τον έλεγχο των ρυθμίσεων ασφαλείας μιας εγκατάστασης του Docker και παρέχει οδηγίες για την βελτίωσή τους. Το docksec πραγματοποιεί παρόμοιους ελέγχους για την ύπαρξη ορισμένων ρυθμίσεων που προτείνει το docker-bench-security και προσφέρει την δυνατότητα αυτόματης εφαρμογής τους. Αρκεί να εκτελεστεί στο σύστημα που είναι εγκατεστημένο το Docker, ενώ η επέκταση των δυνατοτήτων του πραγματοποιείται το ίδιο εύκολα με τα υπόλοιπα εργαλεία.

3.3.2 Σύγκριση με τον τομέα σκλήρυνσης του Docker μέσω του harden

Σε σχέση με τα κριτήρια που πρέπει να ικανοποιούνται, το harden μπορεί και πάλι να τα καλύψει όλα.

- **Σύγκριση με docker-rootless-setup**:

  Όπως το docker-rootless-setup, έτσι και το harden εκτελείται στο σύστημα για το οποίο θέλει να πραγματοποιήσει την εγκατάσταση/σκλήρυνση του Docker. Αυτό που συμβαίνει και στις δύο περιπτώσεις είναι η εγκατάσταση μιας τροποποιημένης έκδοσης του Docker, η οποία δύναται να εκτελεστεί από οποιονδήποτε χρήστη του συστήματος, δίχως την ανάγκη διαχειριστικών δικαιωμάτων. Ωστόσο, το harden υποστηρίζει περισσότερες διανομές από το docker-rootless-setup (που περιορίζεται μόνο σε διανομές βασισμένες στο Debian) και παρέχει περισσότερες δυνατότητες σκλήρυνσης. Πέρα από την εγκατάσταση του Rootless Docker, με το harden πραγματοποιείται αντικατάσταση του αρχικού Container Runtime (runC) με το runsc, από το gVisor. Αυτό, αναπτύχθηκε από την Google και αποτελεί μια ασφαλέστερη επιλογή από το αρχικό (runC), διότι περιορίζει

[5]T. LeRoy. docksec. URL: `https://github.com/TedLeRoy/docksec`.
[6]Docker. Docker Bench for Security. URL: `https://github.com/docker/docker-bench-security`.

τις κλήσεις του συστήματος (system calls) στις άκρως απαραίτητες για την λειτουργία του Docker. Ο συνδυασμός αυτών των δύο βημάτων, όχι μόνο περιορίζει την επιφάνεια επίθεσης σε περίπτωση απόδρασης από ένα δοχείο, αλλά μειώνει και την πιθανότητα κάτι τέτοιο να συμβεί.

- **Σύγκριση με docksec**:

  Το docksec, ενώ επιτυγχάνει την σκλήρυνση του δαίμονα του Docker, βασιζόμενο σε οδηγίες του docker-bench-security, περιορίζεται μονάχα σε αυτό. Δεν πραγματοποιεί ούτε την εγκατάσταση του Rootless Docker, ούτε την αντικατάσταση του runC με το runsc. Επιπλέον, δεν υποστηρίζει την αυτόματη ενημέρωση των δοχείων που εκτελούνται στο σύστημα, όπως κάνει το harden. Με την εκτέλεση του harden, πέρα από τα παραπάνω βήματα, ρυθμίζεται ο δαίμονας του Docker ώστε να εκτελείται με μεγαλύτερη ασφάλεια. Αυτό επιτυγχάνεται χρησιμοποιώντας παραμέτρους όπως "no-new-privileges" για να μην μπορεί ένα δοχείο να λάβει περισσότερα δικαιώματα στην συνέχεια της εκτέλεσής του, καθώς και ενεργοποιώντας τις δυνατότητες του SELinux εάν αυτό υποστηρίζεται από την εκάστοτε υποκείμενη διανομή του συστήματος. Τέλος, πραγματοποιείται εγκατάσταση της υπηρεσίας σε μορφή δοχείου ονόματι "watchtower", η οποία έχει ως στόχο την αυτόματη ενημέρωση των δοχείων που εκτελούνται στο σύστημα.

Επομένως, το harden παρέχει μια ολοκληρωμένη λύση για την εγκατάσταση και σκλήρυνση του Docker, η οποία εκτελεί όλα τα παραπάνω βήματα δίχως την ανάγκη αλληλεπίδρασης ή περαιτέρω ρυθμίσεων από τον χρήστη. Υποστηρίζει πολλές διανομές για τις οποίες μπορεί να κατεβάσει τα απαραίτητα πακέτα για την λειτουργία του Rootless Docker και πέρα από την εγκατάσταση αυτού, επιτυγχάνει να σκληρύνει τον δαίμονα του Docker με κατάλληλες ρυθμίσεις και αντικατάσταση του προκαθορισμένου Container Runtime.

Πίνακας 3.3: Σύγκριση εργαλείων σκλήρυνσης του Docker

Κριτήρια	docker-rootless-setup	docksec	SecDep
Ευκολία χρήσης	Αρκεί η εκτέλεσή του με διαχειριστικά δικαιώματα.	Αρκεί να εκτελεστεί με διαχειριστικά δικαιώματα.	Αρκεί η εκτέλεσή του με διαχειριστικά δικαιώματα.
Εύρος σκλήρυνσης	Περιορίζεται στην εγκατάσταση του Rootless Docker.	Περιορίζεται στη σκλήρυνση του Docker μέσω καλύτερων ρυθμίσεων.	Καταφέρνει να ρυθμίσει καλύτερα τον δαίμονα, να εγκαταστήσει το Rootless Docker και να αντικαταστήσει το προκαθορισμένο Container Runtime με ένα ασφαλέστερο.
Υποστήριξη διαρκούς σκλήρυνσης	Δεν λαμβάνονται βήματα για την διασφάλιση διαρκούς σκλήρυνσης.	Μετά την εκτέλεσή του δεν πραγματοποιεί παραπάνω βήματα.	Πέρα από την σκλήρυνση του Docker, εγκαθιστά και το watchtower για να διασφαλίσει πως τα δοχεία του συστήματος θα ενημερώνονται διαρκώς.
Επεκτασιμότητα	Αρκεί η προσθήκη νέων λειτουργιών στο εκτελέσιμο αρχείο του.	Αρκεί η προσθήκη νέων λειτουργιών στο εκτελέσιμο αρχείο του.	Αρκεί η προσθήκη νέων λειτουργιών στο εκτελέσιμο αρχείο του.

3.4 Συμπεράσματα συγκρίσεων

Μετά από τις συγκρίσεις του SecDep με τα προαναφερθέντα εργαλεία, ακολουθώντας τα κριτήρια που έχουν οριστεί για τον κύριο σκοπό του καθενός, μπορούμε να συμπεράνουμε πως το SecDep όχι μόνο υπερτερεί σε σχέση με τα υπόλοιπα, αλλά προσφέρει μια ολοκληρωμένη λύση με τις δυνατότητες όλων των παραπάνω. Παρέχει δηλαδή, λειτουργίες αυτοματοποίησης για την δημιουργία και σκλήρυνση ενός συστήματος νέφους, ειδικά ρυθμισμένο για την ασφαλή εκτέλεση εφαρμογών που προσφέρονται σε μορφή δοχείων.

Στον τομέα της αυτοματοποίησης δημιουργίας εικονικών μηχανών, με την χρήση της βιβλιοθήκης libcloud, προσφέρεται μια πληθώρα επιλογών για την χρήση νεφών όλων των μεγάλων παρόχων νέφους. Υποστηρίζονται πολλές διανομές, διάφορα μεγέθη εικονικών μηχανών και αρκετές τοποθεσίες. Επιπροσθέτως, ακολουθούνται όλες οι ορθές πρακτικές εργαλείων γραμμής εντολών, όπως ο έλεγχος εγκυρότητας των παραμέτρων και η ευκολία στην κατανόηση της σημασιολογίας και της χρήσης τους. Τέλος, επιτρέπει και την αποδοτική διαχείριση όλων των εικονικών μηχανών που δημιουργήθηκαν μέσω αυτού, κατά μήκος όλων των παρόχων νέφους που υποστηρίζει.

Σχετικά με την σκλήρυνση των εικονικών μηχανών, το SecDep υποστηρίζει πολλές διανομές, σκληραίνει τα συστήματα σε ικανοποιητικό βαθμό και επιπρόσθετα, προσφέρει μηχανισμούς για την υποστήριξη διαρκούς σκλήρυνσης.

Για την σκλήρυνση του Docker, δεδομένου ότι πολλές από τις τεχνολογίες που χρησιμοποιεί είναι σχετικά νέες, δεν υπάρχουν στην αγορά εργαλεία που να έχουν ως κύριο στόχο την αυτόματη εφαρμογή τους. Τα περισσότερα επικεντρώνονται στην σκλήρυνση των δοχείων και όχι του δαίμονα, ενώ όσα επιχειρούν να σκληρύνουν τον δαίμονα έχουν να ανανεωθούν πολύ προτού έρθουν στο προσκήνιο οι τεχνολογίες που χρησιμοποιεί το harden.

Το SecDep παρέχει ένα σύνολο δυνατοτήτων που δεν προσφέρονται από κανένα άλλο αυτοτελές εργαλείο και επιτυγχάνει τον σκοπό του με ευκολία και αποτελεσματικότητα. Για την απόκτηση ίδιου επιπέδου λειτουργιών και αποτελεσμάτων, θα απαιτούνταν η εκτέλεση και η τροποποίηση πολλών ξεχωριστών εργαλείων. Επομένως, μπορεί να θεωρηθεί πως είναι ένα πιο ολοκληρωμένο εργαλείο που όμοιό του δεν υπάρχει ακόμα διαθέσιμο.

Κεφάλαιο 4. Ανάπτυξη Συστήματος

Στην παρούσα ενότητα θα αναλυθούν οι τεχνολογίες που χρησιμοποιήθηκαν για την υλοποίηση του συστήματος, οι αποφάσεις που πάρθηκαν κατά την ανάπτυξή του και οι σχεδιαστικές επιλογές που πραγματοποιήθηκαν. Επίσης, θα αναλυθεί η αρχιτεκτονική του συστήματος και θα επεξηγηθούν οι λειτουργίες των συστατικών του.

4.1 Μοντέλο ανάπτυξης

Για την ανάπτυξη του SecDep, ακολουθήθηκε το μοντέλο καταρράκτη. Στο μοντέλο αυτό, κάθε στάδιο της δραστηριότητας ανάπτυξης εκτελείται σειριακά. Δηλαδή, για την μετάβαση σε κάθε επόμενο στάδιο, απαιτείται η επιτυχής ολοκλήρωση του προηγούμενου. Έτσι και εδώ, ξεκινάμε με την ανάλυση απαιτήσεων, έπειτα προχωράμε στη σχεδίαση του συστήματος και τέλος φτάνουμε στην υλοποίησή του. Η επίδειξη της τελικής μορφής του εργαλείου καλύπτεται στο Κεφάλαιο 5, ενώ η αποτίμηση των αποτελεσμάτων εκτέλεσής του, στο Κεφάλαιο 6.

4.2 Απαιτήσεις από το εργαλείο

Κατά την ανάπτυξη του εργαλείου, έγινε μια προσπάθεια να καλυφθούν ορισμένες βασικές απαιτήσεις που θα έπρεπε αυτό να ικανοποιεί. Όπως όλα τα εργαλεία, έτσι και αυτό έχει έναν σκοπό να εκπληρώσει. Ο σκοπός αυτός είναι η διευκόλυνση του χρήστη ή ενός οργανισμού, στην εγκατάσταση και διαμόρφωση με αυτοματοποιημένο τρόπο ενός ασφαλούς, κατανεμημένου περιβάλλοντος για την εγκατάσταση και λειτουργία μιας εφαρμογής μικρο-υπηρεσιών. Για να γίνει αυτό πραγματικότητα, πρέπει το εργαλείο να ικανοποιεί τις παρακάτω απαιτήσεις.

4.2.1 Λειτουργικές απαιτήσεις

- Το εργαλείο πρέπει για κάθε πάροχο να υποστηρίζει την λειτουργία εμφάνισης λίστας των διαθέσιμων μεγεθών εικονικής μηχανής

- Το εργαλείο πρέπει για κάθε πάροχο να υποστηρίζει την λειτουργία εμφάνισης λίστας των διαθέσιμων τοποθεσιών
- Το εργαλείο πρέπει για κάθε πάροχο να υποστηρίζει την λειτουργία εμφάνισης λίστας των διαθέσιμων διανομών
- Το εργαλείο πρέπει για κάθε εντολή που επιδέχεται παραμέτρους, να ρωτάει τον χρήστη για κάθε παράμετρο που δεν δόθηκε
- Το εργαλείο πρέπει να ελέγχει την εγκυρότητα κάθε παραμέτρου που δίνεται από τον χρήστη
- Το εργαλείο πρέπει να υποστηρίζει την λειτουργία δημιουργίας εικονικής μηχανής
- Το εργαλείο πρέπει να υποστηρίζει την λειτουργία παύσης εικονικής μηχανής
- Το εργαλείο πρέπει να υποστηρίζει την λειτουργία εκκίνησης εικονικής μηχανής
- Το εργαλείο πρέπει να υποστηρίζει την λειτουργία επανεκκίνησης εικονικής μηχανής
- Το εργαλείο πρέπει να υποστηρίζει την λειτουργία διαγραφής εικονικής μηχανής
- Το εργαλείο πρέπει να υποστηρίζει την σύνδεση με SSH ακόμα και χωρίς να διαθέτει ο χρήστης πελάτη SSH
- Το εργαλείο πρέπει να υποστηρίζει την λειτουργία εμφάνισης όλων των εικονικών μηχανών που διαχειρίζεται
- Το εργαλείο πρέπει να εμφανίζει μηνύματα λάθους σε περίπτωση που κάτι πάει στραβά κατά την εκτέλεση
- Το εργαλείο πρέπει να υποστηρίζει την διαδραστική διαμόρφωση του αρχείου ρυθμίσεών του
- Το εργαλείο πρέπει να υποστηρίζει την εκτέλεση εξωτερικού εκτελέσιμου αρχείου στις εικονικές μηχανές με σκοπό την σκλήρυνσή τους
- Το εργαλείο πρέπει να εγκαθιστά και να σκληραίνει και το Docker πέρα από το λειτουργικό σύστημα
- Το εργαλείο θα ενημερώνει περιοδικά τα πακέτα της εικονικής μηχανής και θα κλείνει αχρησιμοποίητες θύρες

- Το εργαλείο πρέπει να υποστηρίζει την εγκατάσταση δοχείων στις εικονικές μηχανές που δημιουργεί, κατά την διάρκεια της σκλήρυνσής τους, μέσω ενός αρχείου docker-compose.yml που θα βρίσκεται στον ίδιο φάκελο με το εκτελέσιμο αρχείο του
- Το εργαλείο πρέπει να μπορεί να εγκαθιστά δοχεία στις εικονικές μηχανές, κατά την διάρκεια σκλήρυνσής τους, μέσω παραμέτρων που έχει εισάγει ο χρήστης
- Το εργαλείο πρέπει να διαθέτει λειτουργία εμφάνισης όλων των διαθέσιμων εντολών του
- Το εργαλείο πρέπει να διαθέτει λειτουργία εμφάνισης συμπλήρωσης εντολών και για τα 3 πιο δημοφιλή κελύφη εντολών (Bash, Zsh, tcsh)
- Το εργαλείο πρέπει να διαθέτει λειτουργία εμφάνισης της έκδοσής του
- Το εργαλείο πρέπει να διαθέτει παράμετρο για την προσπέραση (bypassing) του βήματος επιβεβαίωσης κατά την δημιουργία εικονικών μηχανών
- Το εργαλείο πρέπει να διαθέτει μηχανισμό συγκεκριμενοποίησης περιοχής για τον πάροχο AWS
- Το εργαλείο πρέπει να διαθέτει μηχανισμό επιλογής θύρας για την λειτουργία σύνδεσης μέσω SSH
- Το εργαλείο πρέπει να υποστηρίζει την αρχικοποίηση μονάχα ενός παρόχου δίχως την συμπλήρωση κενών πεδίων για τους υπόλοιπους από τον χρήστη

4.2.2 Μη λειτουργικές απαιτήσεις

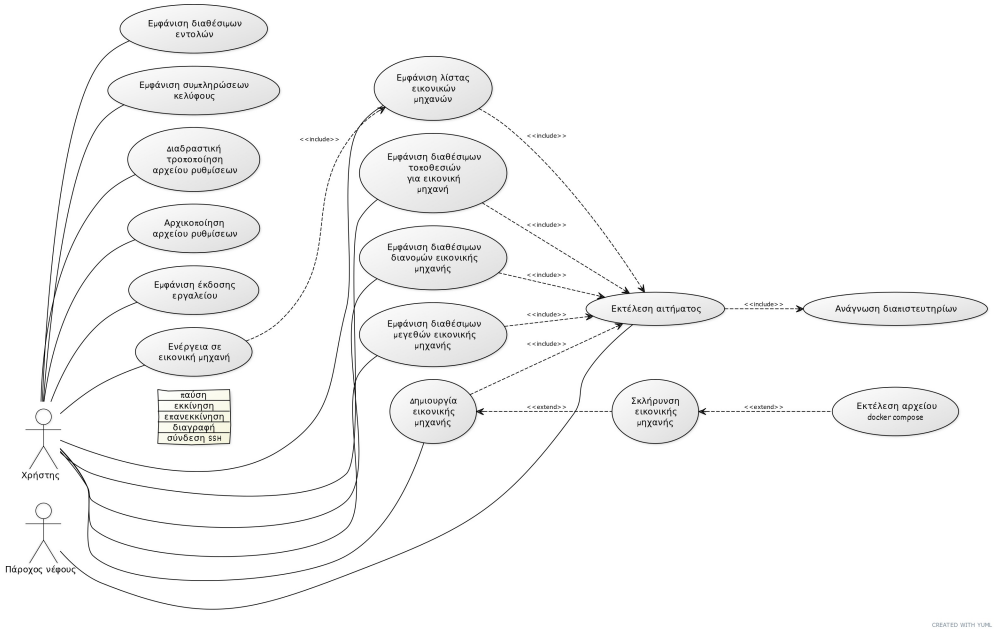
- Το εργαλείο πρέπει να υποστηρίζει την χρήση των 3 μεγαλύτερων ονομάτων στον κλάδο της νεφο-υπολογιστικής (Amazon, Google, Microsoft)
- Το εργαλείο πρέπει να υποστηρίζει την χρήση των 6 δημοφιλέστερων διανομών για περιβάλλον διακομιστή (Debian, Ubuntu, Red Hat Enterprise Linux, Fedora, CentOS, openSUSE Leap)
- Το εργαλείο πρέπει να λειτουργεί μέσω της γραμμής εντολών ώστε να μπορεί να αυτοματοποιηθεί περαιτέρω η εκτέλεση των εντολών του
- Το εργαλείο πρέπει να είναι εύκολο στην χρήση για έναν χρήστη που έχει χρησιμοποιήσει προγράμματα γραμμής εντολών στο παρελθόν

- Το εργαλείο θα πρέπει να είναι αξιόπιστο, δηλαδή κάθε εκτέλεση να επιφέρει τα αναμενόμενα αποτελέσματα
- Το εργαλείο πρέπει να δημιουργεί ξεχωριστό αρχείο για τις ρυθμίσεις του στον ίδιο φάκελο που βρίσκεται και το εκτελέσιμο αρχείο του
- Το εργαλείο πρέπει να κρατάει αρχείο των διευθύνσεων IP των εικονικών μηχανών που δημιουργεί ώστε να μπορούν να διαμορφωθούν περαιτέρω εάν ο χρήστης επιθυμεί να χρησιμοποιήσει άλλα προγράμματα, όπως το Ansible
- Το εργαλείο πρέπει να διαθέτει ένα μοτίβο εντολών που να μπορεί ο χρήστης να καταλαβαίνει και να διαμορφώνει ανάλογα με τις ανάγκες του
- Το εργαλείο πρέπει να δημιουργεί κλειδιά SSH σε περίπτωση που δεν υπάρχουν, στον ίδιο φάκελο με το εκτελέσιμο αρχείο του
- Το εργαλείο θα πρέπει να εγκαθιστά τις υπηρεσίες watchtower και portainer[1] για την περαιτέρω σκλήρυνση του Docker και την διευκόλυνση του χρήστη κατά την εγκατάσταση δοχείων αντίστοιχα

[1]Portainer. Portainer. URL: `https://www.portainer.io/`.

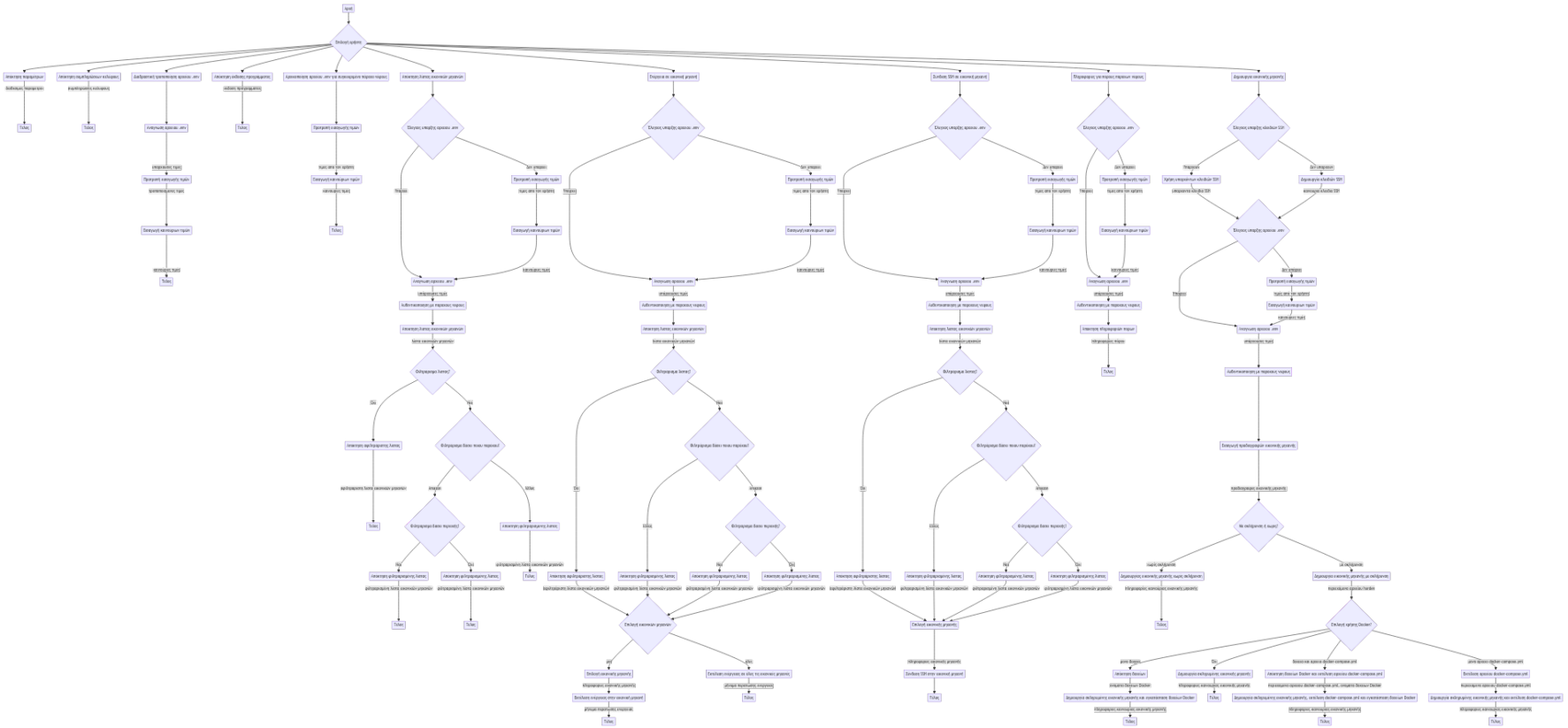
4.3 Διαγραμματική Μοντελοποίηση

Οι παραπάνω απαιτήσεις και ο τρόπος λειτουργίας του εργαλείου, μεταφράζονται καλύτερα στον χρήστη μέσω ενός σχεδιαγράμματος περίπτωσης χρήσης. Το Σχήμα 4.1 δημιουργήθηκε με την βοήθεια του εργαλείου "yuml"[2]. Επιπλέον, με το εργαλείο "mermaid"[3], δημιουργήθηκε ένα διάγραμμα ροής (flowchart) για το SecDep που απεικονίζεται στο Σχήμα 4.2. Το διάγραμμα ροής αυτό, απεικονίζει την ροή των εντολών που μπορεί να επιλεγούν από τον χρήστη κατά την εκτέλεση του προγράμματος.



Σχήμα 4.1: Διάγραμμα περιπτώσεων χρήσης

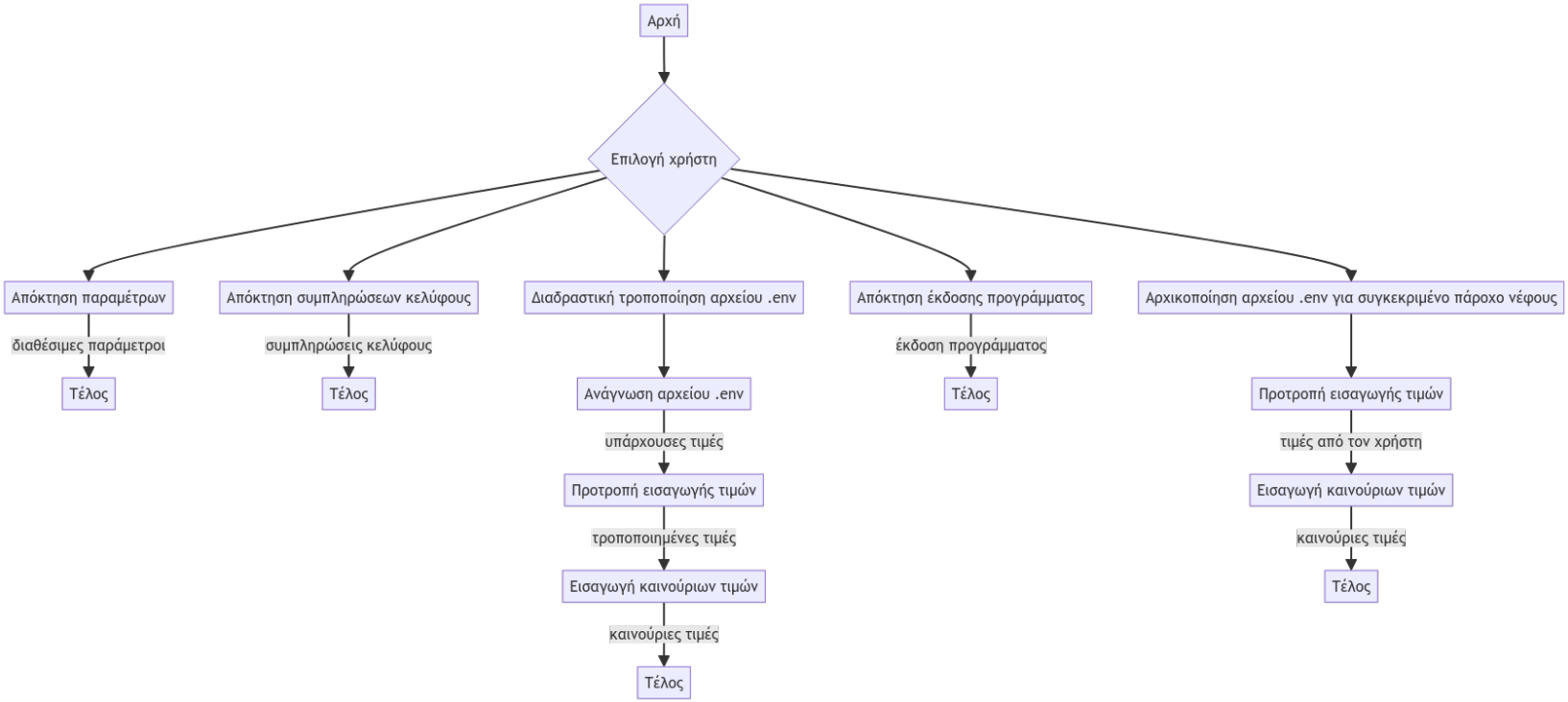
[2]yUML. yUML. URL: `https://yuml.me/`.
[3]Mermaid. Mermaid. URL: `https://mermaid.live/`.



Σχήμα 4.2: Διάγραμμα ροής - Γενικό

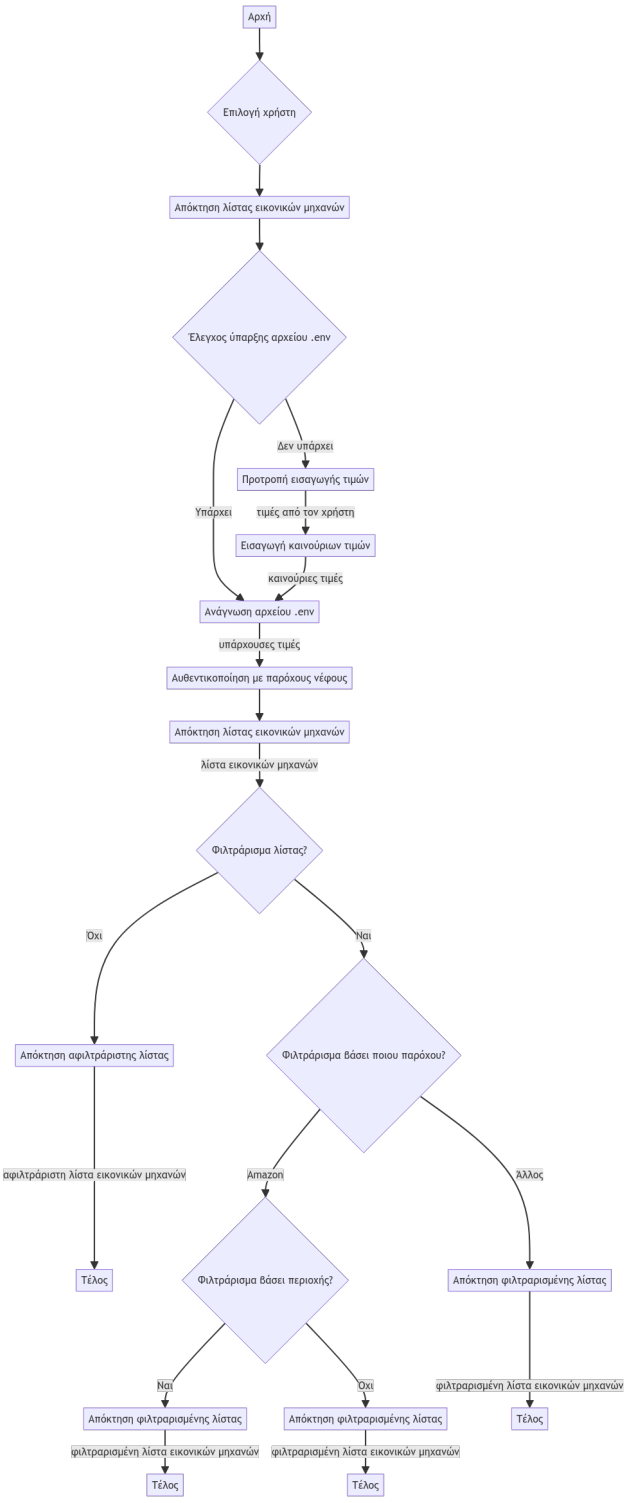
Λόγω της πολυπλοκότητας του Σχήματος 4.2, αυτό διασπάστηκε σε 6 ξεχωριστά διαγράμματα ροής, για τις εξής λειτουργίες του SecDep:

- Εκτέλεση απλών μεθόδων



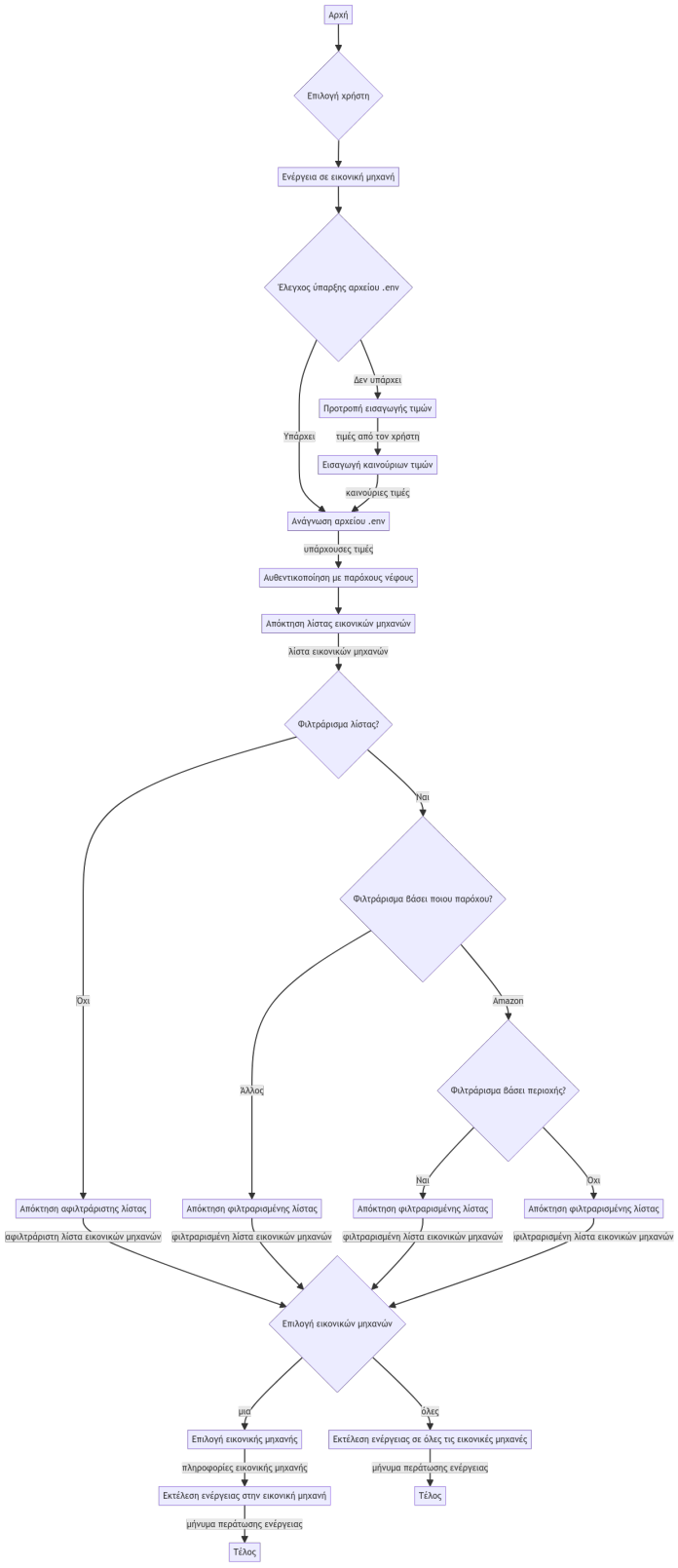
Σχήμα 4.3: Διάγραμμα ροής - Απλές λειτουργίες

- Απόκτηση λίστας εικονικών μηχανών



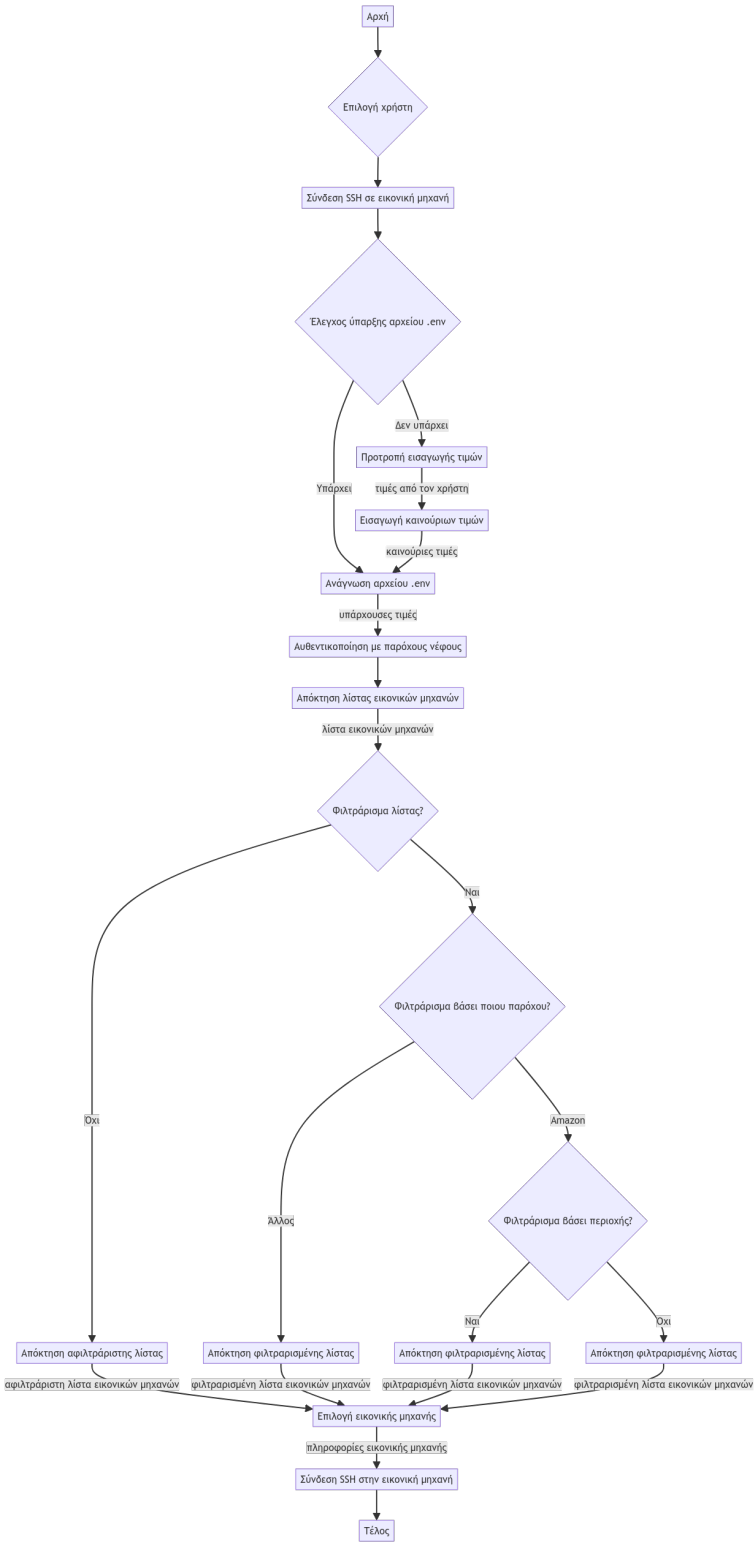
Σχήμα 4.4: Διάγραμμα ροής - Λίστα εικονικών μηχανών

- Εκτέλεση ενέργειας σε εικονική μηχανή



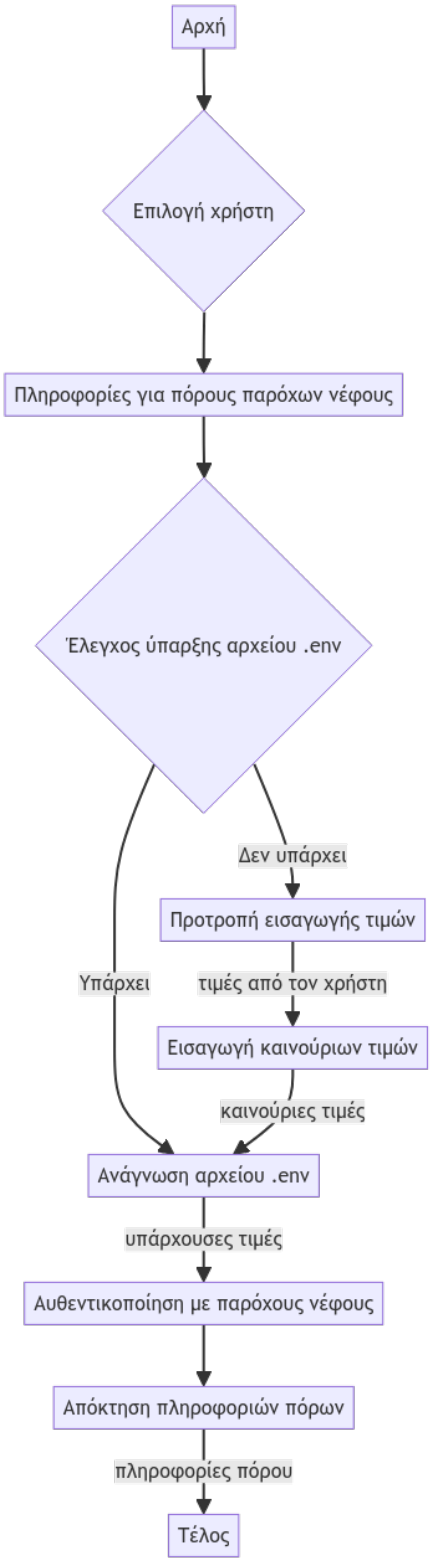
Σχήμα 4.5: Διάγραμμα ροής - Ενέργεια σε εικονική μηχανή

- Σύνδεση SSH σε εικονική μηχανή



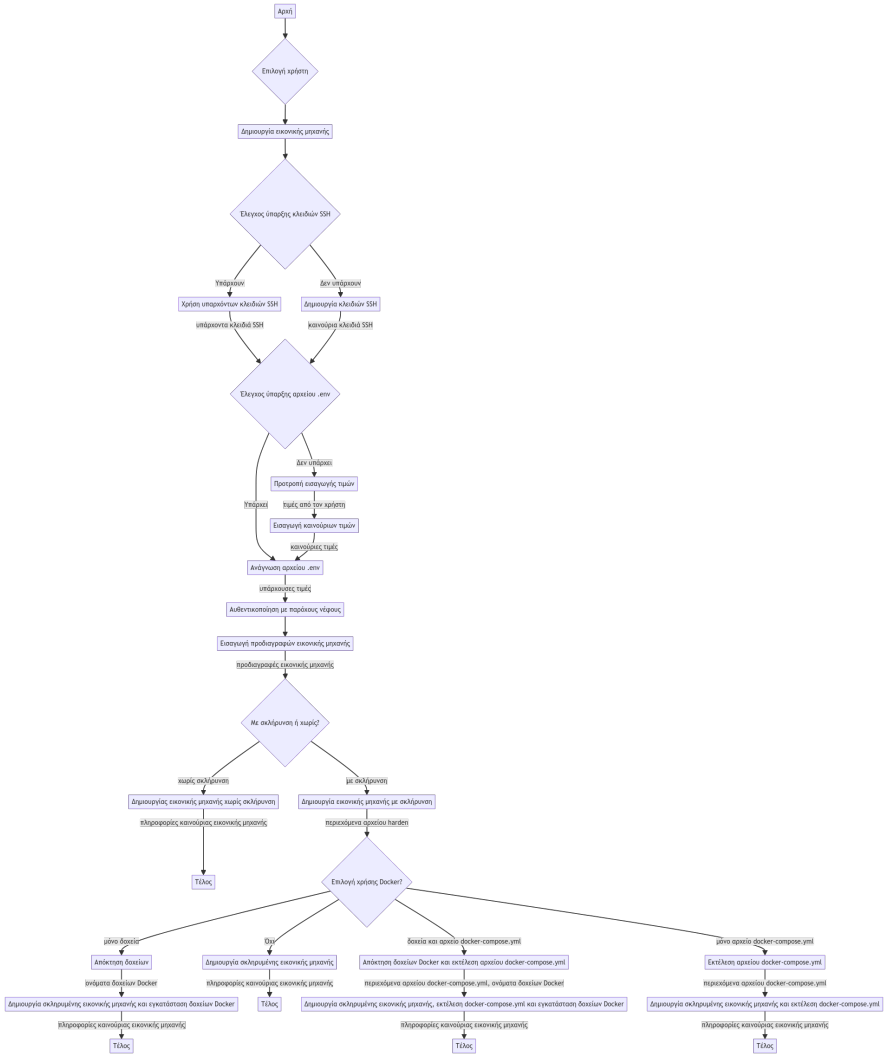
Σχήμα 4.6: Διάγραμμα ροής - Σύνδεση SSH σε εικονική μηχανή

- Απόκτηση πληροφοριών πόρων παρόχου



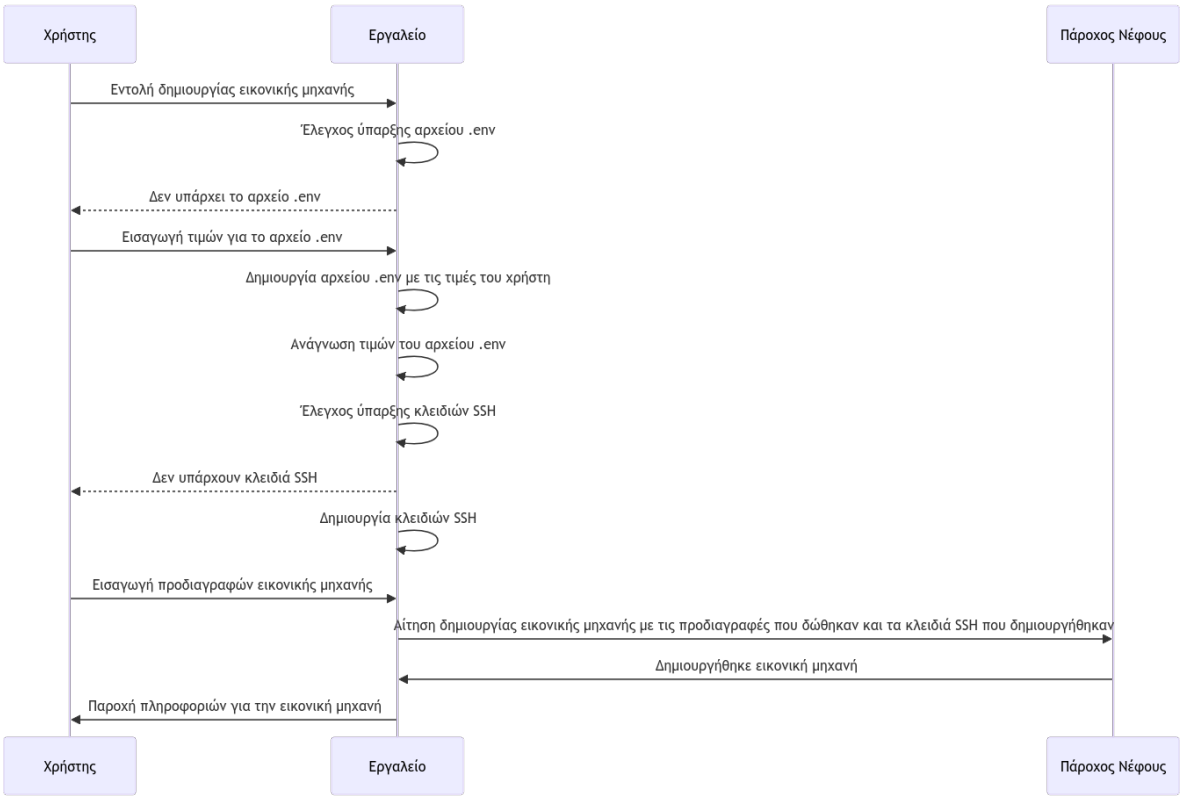
Σχήμα 4.7: Διάγραμμα ροής - Απόκτηση πληροφοριών πόρων παρόχου

- Δημιουργία εικονικής μηχανής



Σχήμα 4.8: Διάγραμμα ροής - Δημιουργία εικονικής μηχανής

Ουσιαστικά, κατά την πραγματική χρήση του SecDep υπάρχουν πάντοτε δύο οντότητες που αλληλεπιδρούν μεταξύ τους. Ο χρήστης και ο πάροχος νέφους. Ο χρήστης μπορεί να ζητήσει πληροφορίες για διαθέσιμους πόρους, όπως οι εικονικές του μηχανές, ή τα συστατικά που έχει στην διάθεσή του για την δημιουργία τους, όπως οι διανομές, τα μεγέθη και οι τοποθεσίες τους. Αυτή η αλληλεπίδραση μεταξύ του χρήστη και του παρόχου νέφους, απεικονίζεται καλύτερα στο Σχήμα 4.9, το οποίο δημιουργήθηκε με το εργαλείο "mermaid". Πρόκειται για ένα διάγραμμα ακολουθίας (sequence diagram), στο οποίο απεικονίζεται η αλληλεπίδραση του χρήστη με τον πάροχο νέφους μέσω του SecDep, κατά την δημιουργία μιας εικονικής μηχανής.



Σχήμα 4.9: Διάγραμμα ακολουθίας για την δημιουργία εικονικής μηχανής

4.4 Αποφάσεις που πάρθηκαν κατά την ανάπτυξη

Κατά την ανάπτυξη του συστήματος, προκειμένου να επιτευχθούν οι στόχοι που θέσαμε στην Ενότητα 1.4 και συγκεκριμένα για την επικοινωνία και διασύνδεση με διάφορους παρόχους νέφους, επιλέχθηκε να γίνει χρήση της βιβλιοθήκης libcloud[4]. Μια από τις δύο επιλογές βιβλιοθηκών της Apache[5] για την επικοινωνία με παρόχους νέφους στα πλαίσια της υλοποίησης

[4]T. A. S. Foundation. Apache Libcloud. URL: `https://libcloud.apache.org/`.
[5]T. A. S. Foundation. Apache. URL: `https://www.apache.org/`.

multi-cloud εφαρμογών, πλατφορμών και εργαλείων. Συγκριτικά με την βιβλιοθήκη jclouds[6], η οποία σχεδιάστηκε για την γλώσσα Java[7], η libcloud είναι πιο απλή στην χρήση, αφού είναι σχεδιασμένη για την γλώσσα Python[8]. Επιπροσθέτως, η libcloud είναι πιο διαδεδομένη, ενώ η γλώσσα python αποτελεί μια πιο ευέλικτη επιλογή για τις ανάγκες της ανάπτυξης του εργαλείου μας. Αυτό συμβαίνει διότι σε σχέση με την Java, η python είναι απλούστερη στην σύνταξή της, προ εγκατεστημένη σε πολλά λειτουργικά συστήματα και διαθέτει πολλές επιβοηθητικές βιβλιοθήκες, επίσημες ή από τρίτους, οι οποίες δύναται να εμπλουτίσουν τα προγράμματα σε python, με παραπάνω λειτουργίες.

Η ανάγκη για μια βιβλιοθήκη που θα λειτουργεί ως μεσάζοντας μεταξύ του εργαλείου και των παρόχων νέφους, οφείλεται στο γεγονός ότι οι πάροχοι νέφους δεν διαθέτουν ένα κοινό API για την διαχείριση των υποδομών τους. Απεναντίας, ο καθένας από αυτούς διαθέτει το δικό του API, το οποίο χρησιμοποιείται έμμεσα μέσω του πίνακα ελέγχου του, που βρίσκεται στην ιστοσελίδα του, ενός δικού του εργαλείου γραμμής εντολών, ή άμεσα μέσω της βιβλιοθήκης που παρέχει ο ίδιος. Η libcloud αποτελεί μια προσπάθεια δημιουργίας ενός ενιαίου API, το οποίο χρησιμοποιώντας τις ίδιες μεθόδους, επιχειρεί να προσκομίσει πανομοιότυπα αποτελέσματα (π.χ. δημιουργία εικονικής μηχανής με συγκεκριμένα χαρακτηριστικά) ανεξαρτήτως του παρόχου νέφους που στοχεύει.

Παρ' όλη την προσπάθεια που καταβλήθηκε από την libcloud όμως, για την ενοποίηση αυτή, οι διαφορές στους τρόπους λειτουργίας του κάθε παρόχου αρχίζουν να εμφανίζονται σύντομα κατά την ανάπτυξη. Για το εργαλείο SecDep που προτείνει η παρούσα διπλωματική εργασία, χρειάζεται να γίνει χρήση των λειτουργιών "Compute" της βιβλιοθήκης libcloud. Στην αρχική σελίδα της βιβλιοθήκης, αναφέρεται υποστήριξη για πάνω από 50 παρόχους. Κατά την σελίδα τεκμηρίωσης [130] για τις λειτουργίες "Compute", αναφέρονται 48 πάροχοι σε έναν πίνακα, ο οποίος αναγράφει την υποστήριξη ή απουσία αυτής, για διάφορες λειτουργίες/μεθόδους (διαχείρισης) εικονικών μηχανών. Στις λειτουργίες αυτές περιλαμβάνονται η δημιουργία εικονικών μηχανών, η εκκίνηση, η παύση, η διαγραφή τους και εν γένει όσες λειτουργίες είναι απαραίτητες για την ολοκληρωμένη λειτουργία του προτεινόμενού μας εργαλείου. Από αυτούς τους 48 παρόχους, οι 25 υποστηρίζουν και τις 9 λειτουργίες διαχείρισης. Ωστόσο, από αυτούς τους 25 έπρεπε να αφαιρεθούν οι 20 για λόγους, όπως η μη ύπαρξη τεκμηρίωσης χρήσης από την libcloud, η μη συνέχιση της λειτουργίας τους, η έλλειψη πληροφοριών τιμής ή δοκιμαστικής περιόδου για τις υπηρεσίες τους και η αδυναμία δημιουργίας λογαριασμού στην επίσημη σελίδα τους. Επομένως, υπάρχουν εν τέλει 5 πάροχοι νέφους που δύναται να χρησιμοποιηθούν, υποστηρίζοντας όλες τις πιθανές λειτουργίες διαχείρισης εικονικών μηχανών. Τέλος, από

[6]T. A. S. Foundation. Apache jclouds. URL: https://jclouds.apache.org/.
[7]Oracle. Java. URL: https://www.java.com/en/.
[8]P. S. Foundation. Python. URL: https://www.python.org/.

αυτούς τους 5, επιλέχθηκαν για επίσημη υποστήριξη από το εργαλείο που προτείνεται στην παρούσα διπλωματική εργασία, τα τρία μεγαλύτερα ονόματα του χώρου, οι Amazon, Google και Microsoft. Οι άλλοι δύο είτε δεν παρείχαν δοκιμαστική περίοδο των υπηρεσιών τους, είτε δεν υπήρχε εμπιστοσύνη ως προς την ποιότητα των υπηρεσιών αυτών.

Παρακάτω θα διαπιστώσουμε πως ακόμα και με την χρήση της βιβλιοθήκης libcloud και την επίσημη υποστήριξη όλων των λειτουργιών για κάθε πάροχο, υπήρχε η ανάγκη χρήσης της επίσημης βιβλιοθήκης της Azure (Microsoft Azure SDK) για την δημιουργία Resource Group[9] και Virtual Network[10] διότι αυτές οι λειτουργίες δεν υποστηρίζονται μέσω της libcloud. Επιπροσθέτως, δίχως την υποστήριξη των λειτουργιών αυτών, ο χρήστης θα ήταν υποχρεωμένος να δημιουργεί αυτούς τους πόρους χειροκίνητα ή να είναι περιορισμένος μονάχα σε μια γεωγραφική τοποθεσία. Οι δύο τελευταίες επιλογές πάνε ενάντια σε έναν από τους βασικότερους στόχους του εργαλείου. Την ομοιόμορφη λειτουργία για κάθε πάροχο νέφους που υποστηρίζεται.

Πέρα από την απόφαση για την χρήση της libcloud και την γλώσσα προγραμματισμού Python, πάρθηκε επίσης η απόφαση για την υποστήριξη των 6 δημοφιλέστερων επιλογών για λειτουργικά συστήματα Linux σε περιβάλλοντα διακομιστών. Οι διανομές αυτές υποστηρίζονται και για τους 3 παρόχους νέφους και κάθε μια από αυτές περιλαμβάνει διαφορετικές εκδόσεις της κάθε διανομής. Αυτές είναι οι εξής για κάθε πάροχο:

Πίνακας 4.1: Υποστηριζόμενες εκδόσεις διανομών για κάθε πάροχο

		Πάροχοι		
		AWS	Azure	GCE
Διανομές	Ubuntu[11]	22.04, 22.10	22.04, 22.10	Όλες
	Debian[12]	10, 11	10, 11	Όλες
	CentOS[13]	7, 8, 9	8.4, 8.5	Όλες
	Fedora[14]	37	36, 37	Όλες
	Red Hat Enterprise Linux[15]	7.9, 8.6, 9	8.6, 9.1	Όλες
	openSUSE Leap[16]	15.3, 15.4	15.3, 15.4	Όλες

[9]Microsoft. Microsoft Azure SDK for Python. URL: https://pypi.org/project/azure-mgmt-resource/.

[10]Microsoft. Microsoft Azure SDK for Python. URL: https://pypi.org/project/azure-mgmt-network/.

[11]Canonical. Ubuntu. URL: https://ubuntu.com/.

[12]Debian. Debian. URL: https://www.debian.org/.

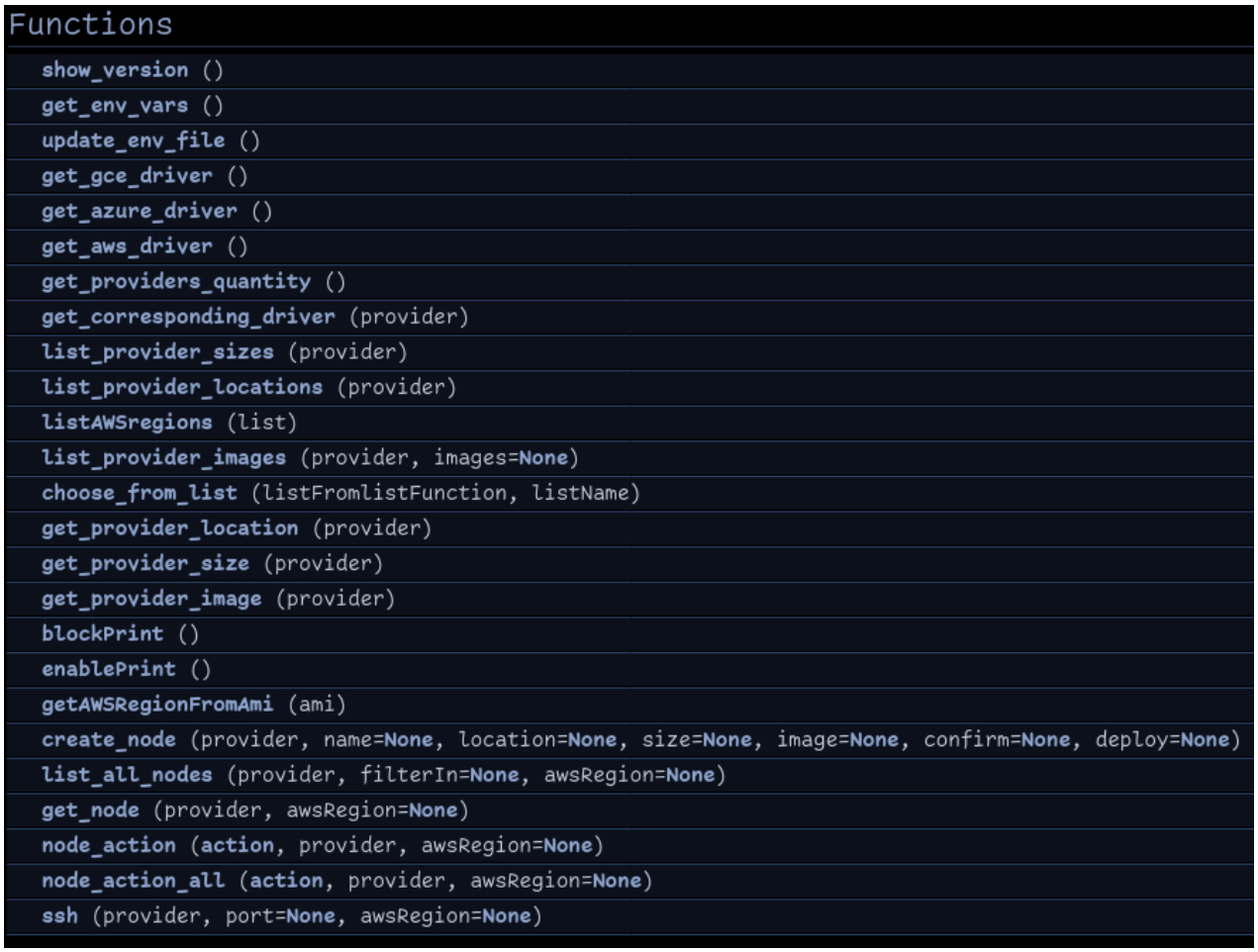
[13]CentOS. CentOS. URL: https://www.centos.org/.

[14]Fedora. Fedora. URL: https://fedoraproject.org/.

Ο λόγος που οι εκδόσεις είναι συγκεκριμένες για τους παρόχους AWS και Azure, είναι διότι ο αριθμός των διαθέσιμων διανομών και εκδόσεών τους είναι υπερβολικά μεγάλος. Αυτό θα είχε ως αποτέλεσμα, κατά την εκτέλεση της λειτουργίας δημιουργίας λίστας τους ή κατά τον έλεγχο της ορθότητας των παραμέτρων που εισάγει ο χρήστης, η αντίστοιχη αιτούμενη εκτέλεση του εργαλείου από τον χρήση, να καθυστερούσε σε τεράστιο βαθμό να ολοκληρωθεί. Επομένως, θα υπήρχε αρνητική επίδραση στην εμπειρία του χρήστη.

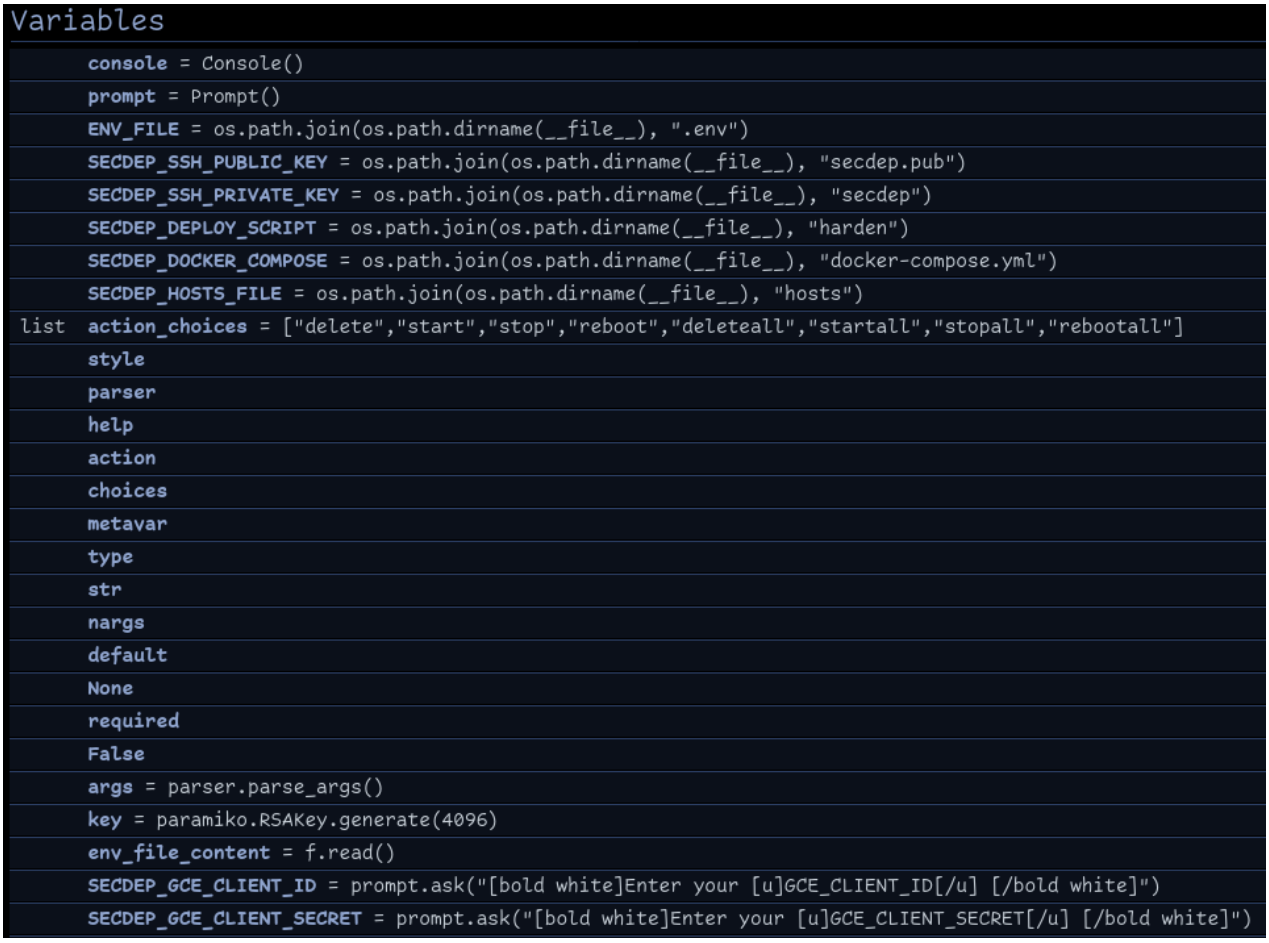
4.5 Αρχιτεκτονική Εργαλείου

Οι διαθέσιμες συναρτήσεις και μεταβλητές του secdep.py, απεικονίζονται στα Σχήματα 4.10, 4.11 και 4.12 αντίστοιχα. Η δημιουργία τους πραγματοποιήθηκε με το εργαλείο "doxygen"[17].



Σχήμα 4.10: Συναρτήσεις του secdep.py

[15]R. Hat. Red Hat. URL: https://www.redhat.com/.
[16]openSUSE. openSUSE. URL: https://www.opensuse.org/.
[17]D. van Heesch. Doxygen. URL: https://github.com/doxygen/doxygen.



Σχήμα 4.11: Μεταβλητές του secdep.py

```
    SECDEP_GCE_PROJECT_ID = prompt.ask("[bold white]Enter your [u]GCE_PROJECT_ID[/u] [/bold white]")
    SECDEP_AZURE_TENANT_ID = prompt.ask("[bold white]Enter your [u]AZURE_TENANT_ID[/u] [/bold white]")
    SECDEP_AZURE_SUB_ID = prompt.ask("[bold white]Enter your [u]AZURE_SUB_ID[/u] [/bold white]")
    SECDEP_AZURE_APP_ID = prompt.ask("[bold white]Enter your [u]AZURE_APP_ID[/u] [/bold white]")
    SECDEP_AZURE_PASSWORD = prompt.ask("[bold white]Enter your [u]AZURE_PASSWORD[/u] [/bold white]")
    SECDEP_AWS_ACCESS_KEY = prompt.ask("[bold white]Enter your [u]AWS_ACCESS_KEY[/u] [/bold white]")
    SECDEP_AWS_SECRET_KEY = prompt.ask("[bold white]Enter your [u]AWS_SECRET_KEY[/u] [/bold white]")
dict  AWS_ubuntu22_04_images
dict  AWS_ubuntu22_10_images
dict  AWS_debian_10_images
dict  AWS_debian_11_images
dict  AWS_centos7_images
dict  AWS_centos8_images
dict  AWS_centos9_images
dict  AWS_fedora37_images
dict  AWS_redhat9_images
dict  AWS_redhat8_6_images
dict  AWS_redhat7_9_images
dict  AWS_opensuseLeap15_3_images
dict  AWS_opensuseLeap15_4_images
dict  AWS_images
dict  AZURE_images
    providers_quantity = get_providers_quantity()
    gce_driver = get_gce_driver()
    azure_driver = get_azure_driver()
    aws_driver = get_aws_driver()
```

Σχήμα 4.12: Μεταβλητές του secdep.py (συνέχεια)

Προκειμένου να κατανοήσουμε καλύτερα πως αλληλεπιδρούν ορισμένες από τις συναρτήσεις μεταξύ τους, δημιουργήθηκαν δύο διαγράμματα συστατικών (component diagrams) για τις εξής λειτουργίες:

- Απόκτηση λίστας εικονικών μηχανών συγκεκριμένης περιοχής του παρόχου νέφους Amazon (Σχήμα 4.13)
- Απόκτηση πληροφοριών AMI (Amazon Machine Image) (Σχήμα 4.14)

Τα διαγράμματα αυτά δημιουργήθηκαν με την χρήση της βιβλιοθήκης "pydoctrace"[18] και του εργαλείου ανοιχτού κώδικα "plantuml"[19]. Με την εκτέλεση της Εντολής 1, εγκαθίσταται η βιβλιοθήκη στο σύστημα. Μετά την περαίωσή της και την εγκατάσταση του πακέτου plantuml, προκειμένου να αποκτήσουμε τα διαγράμματα συστατικών χρειάστηκε να πραγματοποιηθούν ορισμένες προσθέσεις στον κώδικα του εργαλείου.

```
pip install pydoctrace --break-system-packages
```

Εντολή 1: Εγκατάσταση της βιβλιοθήκης pydoctrace

Οι προσθέσεις αυτές αφορούν την προσθήκη των εξής γραμμών κώδικα του secdep.py στο σημείο που ορίζονται οι βιβλιοθήκες:

```
from pydoctrace.doctrace import trace_to_component_puml
from pydoctrace.callfilter.presets import (
    EXCLUDE_BUILTINS_PRESET,
    EXCLUDE_CALL_DEPTH_PRESET_FACTORY,
    EXCLUDE_DEPTH_BELOW_5_PRESET,
    EXCLUDE_STDLIB_PRESET,
    EXCLUDE_TESTS_PRESET,
    TRACE_ALL_PRESET,
)
```

Έπειτα, προσθέτουμε και τις παρακάτω γραμμές κώδικα, ακριβώς πριν την δήλωση των συναρτήσεων που θέλουμε να καλέσουμε:

[18] L. Sorel-Giffo. pydoctrace. URL: https://github.com/lucsorel/pydoctrace.
[19] PlantUML. PlantUML. URL: https://github.com/plantuml/plantuml.

```
ABOVE_1_PRESET = EXCLUDE_CALL_DEPTH_PRESET_FACTORY(1)
@trace_to_component_puml(filter_presets=[EXCLUDE_BUILTINS_PRESET,
↪  ABOVE_1_PRESET, EXCLUDE_DEPTH_BELOW_5_PRESET, EXCLUDE_STDLIB_PRESET,
↪  EXCLUDE_TESTS_PRESET, TRACE_ALL_PRESET])
```

Με την εκτέλεση των Εντολών 2 και 3, παράγονται δύο αρχεία με κατάληξη ".puml", τα οποία περιέχουν οδηγίες για την απεικόνιση των διαγραμμάτων συστατικών στα Σχήματα 4.13 και 4.14 αντίστοιχα.

```
python3 secdep.py --provider aws --listimages --print
```

Εντολή 2: Απόκτηση πληροφοριών AMI (Amazon Machine Image)

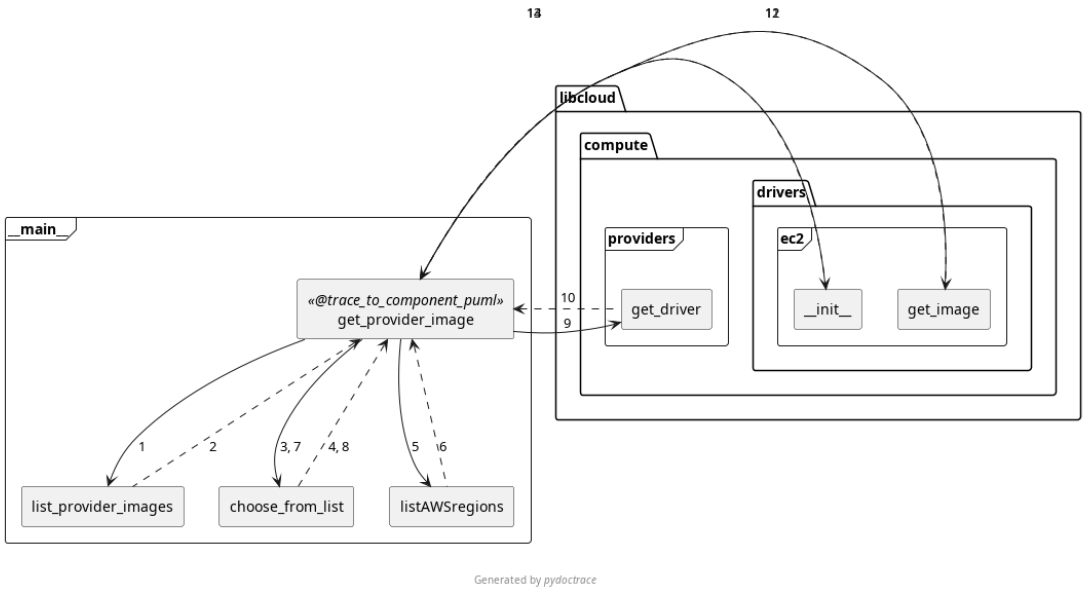
```
python3 secdep.py --provider aws --list --awsregion eu-north-1
```

Εντολή 3: Απόκτηση λίστας εικονικών μηχανών συγκεκριμένης περιοχής του παρόχου νέφους Amazon

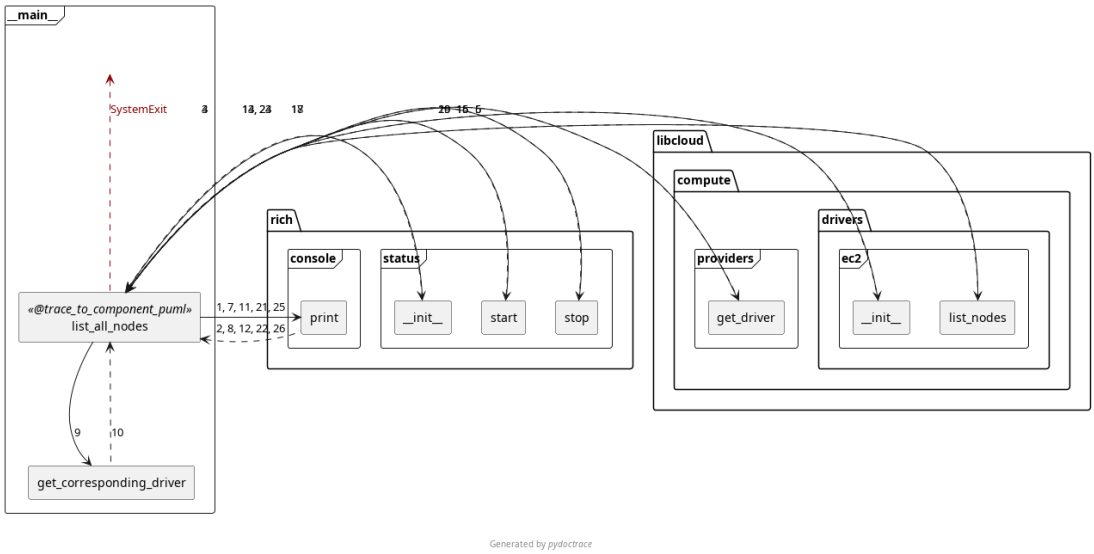
Τέλος, η μετατροπή των αρχείων κειμένου σε σχήματα πραγματοποιείται με την εκτέλεση της εντολής 4.

```
plantuml -progress -DPLANTUML_LIMIT_SIZE=8192 <όνομα αρχείου>.puml
```

Εντολή 4: Μετατροπή αρχείων ".puml" σε αρχεία ".png" χρησιμοποιώντας το plantuml



Σχήμα 4.13: Διάγραμμα συστατικών για την Εντολή 2



Σχήμα 4.14: Διάγραμμα συστατικών για την Εντολή 3

Τα παραπάνω διαγράμματα αποτελούν απλουστευμένες εκδοχές των πραγματικών. Αυτό συμβαίνει, διότι έχουμε ρυθμίσει το pydoctrace να παραλείπει τις εμφανίσεις ορισμένων εσωτερικών κλήσεων συναρτήσεων, καθώς και αυτών που βρίσκονται σε επίπεδο callstack μεγαλύτερο από ένα. Στην αντίθετη περίπτωση, τα διαγράμματα θα ήταν αρκετά δυσνόητα και δυσανάγνωστα.

Παρατηρώντας το Σχήμα 4.13, αποκτάμε μεγαλύτερη κατανόηση σχετικά με την εξάρτηση των συναρτήσεων μεταξύ τους. Προκειμένου να φτάσουμε στην απόκτηση πληροφοριών για το AMI που θα διαλέξουμε, αρχικά πρέπει να λάβουμε την λίστα των διαθέσιμων διανομών. Έπειτα, επιλέγουμε μια από αυτές και αποκτάμε την λίστα των διαθέσιμων περιοχών της Amazon. Αυτό το βήμα είναι απαραίτητο, διότι κάθε AMI έχει ξεχωριστό αναγνωριστικό ανά περιοχή. Αφότου επιλέξουμε την περιοχή που επιθυμούμε, το εργαλείο επικοινωνεί με τον πάροχο νέφους εκτελώντας την συνάρτηση απόκτησης της εικόνας και μας επιστρέφει τις πληροφορίες της.

Στο Σχήμα 4.14, βλέπουμε πως γίνεται κλήση της συνάρτησης απόκτησης του οδηγού για την αυθεντικοποίηση με την Amazon και έπειτα λαμβάνουμε την λίστα των εικονικών μηχανών για την περιοχή που ορίσαμε κατά την σύνταξη της εντολής, από την αντίστοιχη συνάρτηση. Οι μέθοδοι start και stop αφορούν την εκκίνηση και την παύση ενός μηχανισμού, ο οποίος έχει προστεθεί στον τρόπο εμφάνισης των εντολών, ώστε να γνωρίζει ο χρήστης πως η εκτέλεση του προγράμματος συνεχίζεται. Αυτό αποτελεί ένα χαρακτηριστικό που αυξάνει την εμπειρία χρήσης, ιδίως για εντολές με μεγάλης διαρκείας εκτέλεση.

Το SecDep αποτελείται από δύο εκτελέσιμα αρχεία. Το πρώτο αρχείο, το "secdep.py", είναι το κύριο αρχείο του προγράμματος. Αυτό το αρχείο είναι υπεύθυνο για την επικοινωνία με τους παρόχους νέφους και την διαχείριση των εικονικών μηχανών. Το δεύτερο αρχείο, το "harden", είναι υπεύθυνο για την σκλήρυνση των εικονικών μηχανών και την εγκατάσταση/σκλήρυνση του Docker. Το harden, εκτελείται μόνο στις εικονικές μηχανές κατά την δημιουργία τους και μόνο εάν ο χρήστης το επιθυμεί.

Οι βασικότερες συναρτήσεις του secdep.py είναι οι εξής:

- **create_node**:

  Η συνάρτηση "create_node", είναι αυτή που δημιουργεί τις εικονικές μηχανές λαμβάνοντας ή ζητώντας από τον χρήστη τις κατάλληλες παραμέτρους. Οι βασικές παράμετροι που μπορεί να λάβει είναι:

  - **provider**: Ο πάροχος νέφους στο νέφος του οποίου θα δημιουργηθεί η εικονική μηχανή (aws, gce, azure).
  - **name**: Το όνομα της εικονικής μηχανής.
  - **location**: Η περιοχή στην οποία θα δημιουργηθεί η εικονική μηχανή.
  - **size**: Το μέγεθος της εικονικής μηχανής.
  - **image**: Η εικόνα από την οποία θα δημιουργηθεί η εικονική μηχανή.
  - **confirm**: Χρησιμοποιείται για επιβεβαίωση της δημιουργίας της εικονικής μηχανής χωρίς να χρειαστεί χειροκίνητη επιβεβαίωση από τον χρήστη.
  - **deploy**: Για την εκτέλεση του harden (δηλ. την εφαρμογή της σκλήρυνσης).

- **node_action**:

  Η συνάρτηση "node_action", είναι αυτή που εκτελεί τις ενέργειες πάνω στις εικονικές μηχανές. Στις ενέργειες αυτές περιλαμβάνονται:

  - **start** (εκκίνηση της εικονικής μηχανής)
  - **stop** (παύση της εικονικής μηχανής)
  - **reboot** (επανεκκίνηση της εικονικής μηχανής)
  - **delete** (διαγραφή της εικονικής μηχανής)

- **list_all_nodes**: Η συνάρτηση "list_all_nodes", είναι αυτή που εμφανίζει τις εικονικές μηχανές που διαχειρίζεται το SecDep. Χωρίς παραμέτρους θα εμφάνιζε όλες τις εικονικές μηχανές από όλους τους παρόχους. Επειδή, όμως, δεν είναι πάντοτε απαραίτητο να εμφανίζονται όλες οι εικονικές μηχανές, ο χρήστης μπορεί να δώσει τις παρακάτω παραμέτρους:
  - **provider**
  - **filterIn**: Χρησιμοποιείται για το φιλτράρισμα της λίστας κατά την χρήση της συνάρτησης "node_action". Με βάση την τιμή της ενέργειας που θέλει να πραγματοποιήσει ο χρήστης, δεν περιλαμβάνονται στην λίστα, οι εικονικές μηχανές για τις οποίες δεν είναι δυνατή η εκτέλεση της ενέργειας αυτής.
  - **awsRegion**: Μόνο για τον πάροχο AWS. Χρησιμοποιείται για την επιλογή της περιοχής. Δίχως την παράμετρο αυτή, θα γινόταν έρευνα για εικονικές μηχανές σε όλες τις περιοχές. Κάτι που θα έκανε την εκτέλεση της εντολής πολύ αργή, ειδικά εάν ο χρήστης γνωρίζει πως χρησιμοποιεί μονάχα μια περιοχή.

Το αρχείο harden, αποτελείται από πολλές συναρτήσεις, οι οποίες εκτελούνται με την σειρά προκειμένου να ασφαλιστεί το σύστημα στο οποίο εκτελείται. Οι πιο αξιοσημείωτες αυτών είναι:

- **hardenSSH**:

  Η συνάρτηση υπεύθυνη για την σκλήρυνση του SSH.

- **dockerInit**:

  Η συνάρτηση υπεύθυνη για την εγκατάσταση/σκλήρυνση του Docker.

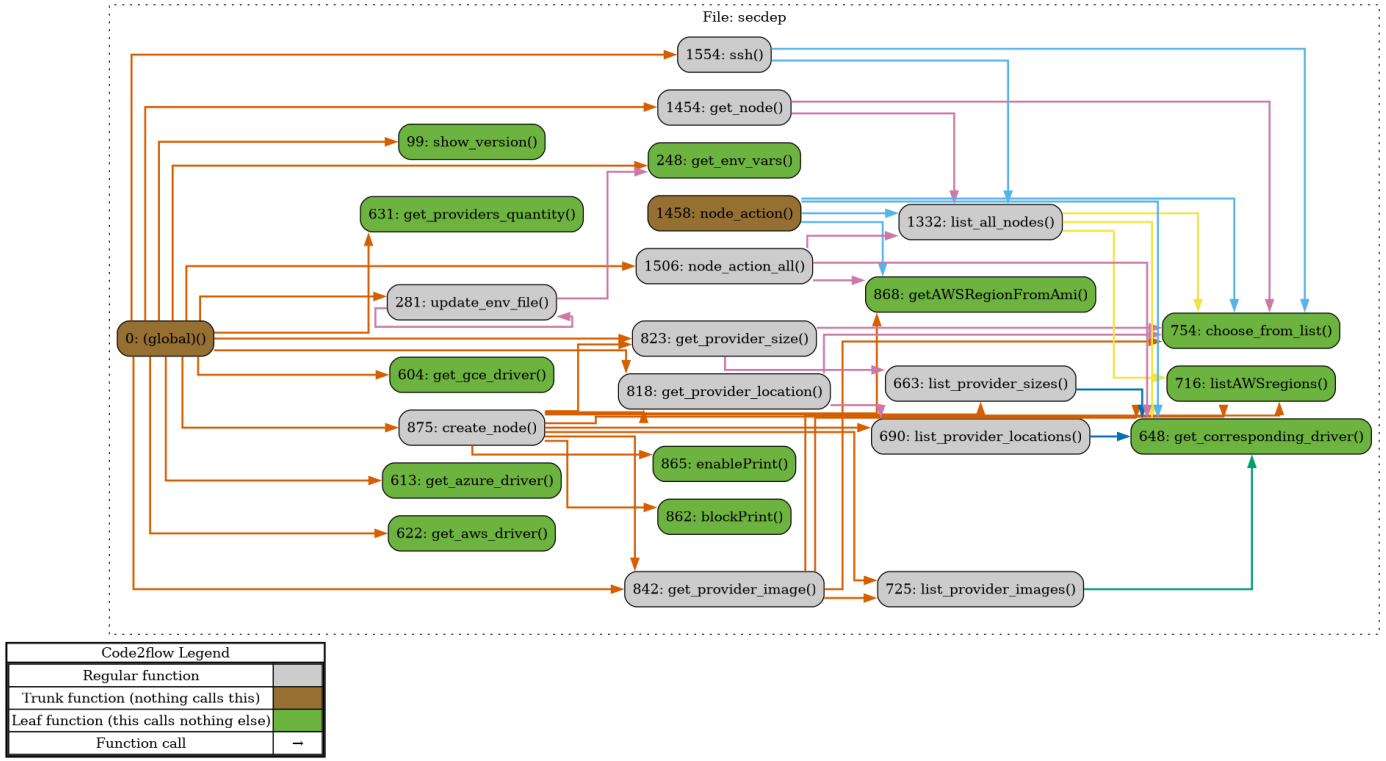
- **kernelSecurityModuleInit**:

  Η συνάρτηση υπεύθυνη για την εγκατάσταση/ρύθμιση του κατάλληλου για την διανομή (της εικονικής μηχανής) kernel security module.

- **configureFail2ban**:

  Αφότου έχει προηγηθεί η εγκατάσταση του κατάλληλου για την διανομή προγράμματος αναχώματος ασφαλείας, η συνάρτηση αυτή ρυθμίζει το fail2ban για την προστασία από επιθέσεις brute force.

Με το “code2flow”[20] δημιουργήθηκε ένα διάγραμμα που απεικονίζει τις κλήσεις συναρτήσεων που γίνονται κατά την εκτέλεση του αρχείου secdep.py. Παράλληλα, το ίδιο αποτέλεσμα επιτεύχθηκε για το αρχείο harden με την βοήθεια του “callGraph”[21]. Τα παραπάνω απεικονίζονται στα Σχήματα 4.15 και 4.16 αντίστοιχα.



Σχήμα 4.15: Διάγραμμα κλήσεων συναρτήσεων του secdep.py

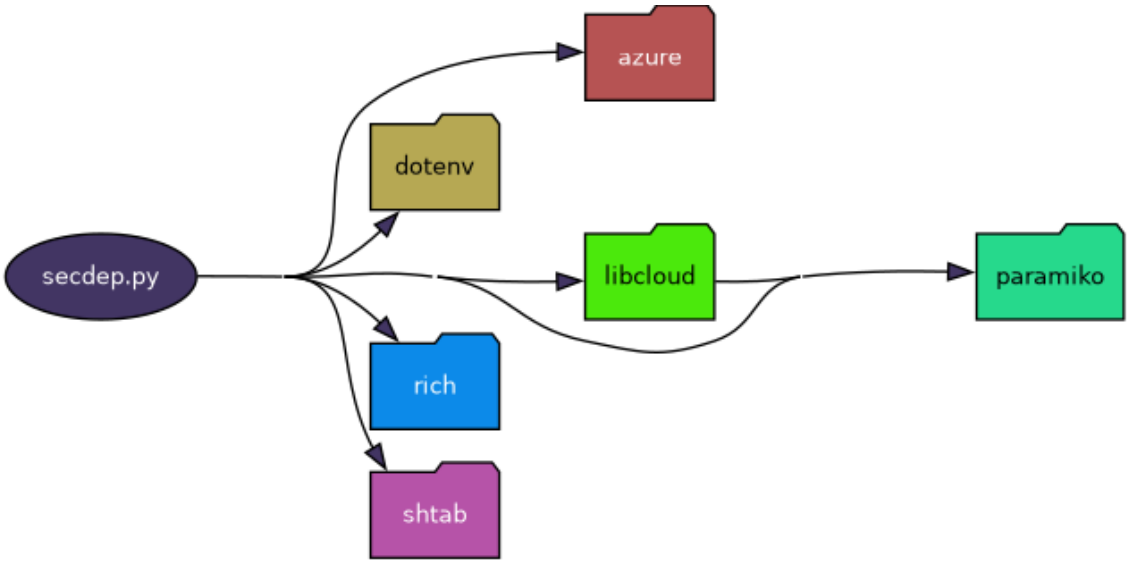
[20] S. Rogowski. code2flow. URL: `https://github.com/scottrogowski/code2flow`.
[21] C. Koknat. callGraph. URL: `https://github.com/koknat/callGraph`.

Σχήμα 4.16: Διάγραμμα κλήσεων συναρτήσεων του harden

Επιπλέον, ένας χάρτης των σημαντικών εξαρτήσεων του secdep.py δημιουργήθηκε με την βοήθεια του "pydeps"[22]. Αυτό έγινε με την εκτέλεση της παρακάτω εντολής:

```
pydeps -T png --cluster --include-missing --max-bacon=1 --noshow --reverse
↪  --rankdir RL -o secdep-module-dependencies-organized.png secdep.py
```

Εντολή 5: Εντολή δημιουργίας χάρτη εξαρτήσεων του secdep.py



Σχήμα 4.17: Διάγραμμα σημαντικών εξαρτήσεων του secdep.py

Οι παραπάνω βιβλιοθήκες, αποτελούν τους πυλώνες του SecDep. Κάθε μια από αυτές, εξειδικεύεται σε διαφορετικούς τομείς και χρησιμοποιείται για την κάλυψη διαφόρων λειτουργιών του SecDep. Με την σειρά που απεικονίζονται, έχουν τις εξής αρμοδιότητες:

- **azure**:

  Η βιβλιοθήκη της Azure, χρησιμοποιείται για την επικοινωνία και αυθεντικοποίηση με τον πάροχο νέφους Azure. Ο λόγος που χρειάζεται να χρησιμοποιηθεί συνδυαστικά με την libcloud, είναι για επιτύχουμε την ομοιόμορφη συμπεριφορά του εργαλείου κατά μήκος όλων των παρόχων που αυτό υποστηρίζει. Συγκεκριμένα, την χρειαζόμαστε για την επίτευξη των λειτουργιών δημιουργίας Resource Group και Virtual Network, καθώς αυτές λειτουργίες δεν υποστηρίζονται από την libcloud την παρούσα στιγμή.

[22]Bjorn. pydeps. URL: https://github.com/thebjorn/pydeps.

- **dotenv**:

  Η βιβλιοθήκη dotenv[23] χρησιμοποιείται για την ανάγνωση αρχείων ".env". Το SecDep κάνει χρήση ενός αρχείου ".env", στο οποίο δηλώνονται οι τιμές που χρειάζεται για την αυθεντικοποίηση με τους παρόχους νέφους.

- **libcloud**:

  Η βιβλιοθήκη libcloud είναι υπεύθυνη για την αυθεντικοποίηση με τους παρόχους νέφους, με σκοπό την εκτέλεση διαφόρων λειτουργιών σχετικά με τις εικονικές μηχανές του καθενός. Μας επιτρέπει να δημιουργούμε και να διαχειριζόμαστε εικονικές μηχανές, καθώς και να λαμβάνουμε πληροφορίες για πόρους του κάθε παρόχου νέφους.

- **rich**:

  Η βιβλιοθήκη rich[24] εμπλουτίζει την εμφάνιση του SecDep. Βοηθάει στην καλύτερη διαμόρφωση της εμφάνισης των αποτελεσμάτων των εντολών του, με σκοπό την θετική αύξηση της εμπειρίας χρήσης του εργαλείου.

- **shtab**:

  Η βιβλιοθήκη shtab[25] χρησιμοποιείται για παραγωγή συμπληρώσεων κελύφους. Αυτό αποτελεί βασικό χαρακτηριστικό για την ευχρηστία του εργαλείου, καθώς διευκολύνει τον χρήστη στην εύρεση των διαθέσιμων παραμέτρων και στην ταχύτερη σύνταξη εντολών.

- **paramiko**:

  Η βιβλιοθήκη paramiko[26] αποτελεί εξάρτηση της libcloud για την επικοινωνία με τις εικονικές μηχανές, κάνοντας χρήση του πρωτοκόλλου SSH. Το SecDep, χρησιμοποιεί την βιβλιοθήκη αυτή για την διαδραστική σύνδεση σε εικονικές μηχανές μέσω SSH σε περίπτωση απουσίας πελάτη που να προσφέρει αυτή τη δυνατότητα. Επιπλέον, χρησιμοποιείται για την δημιουργία κλειδιών SSH.

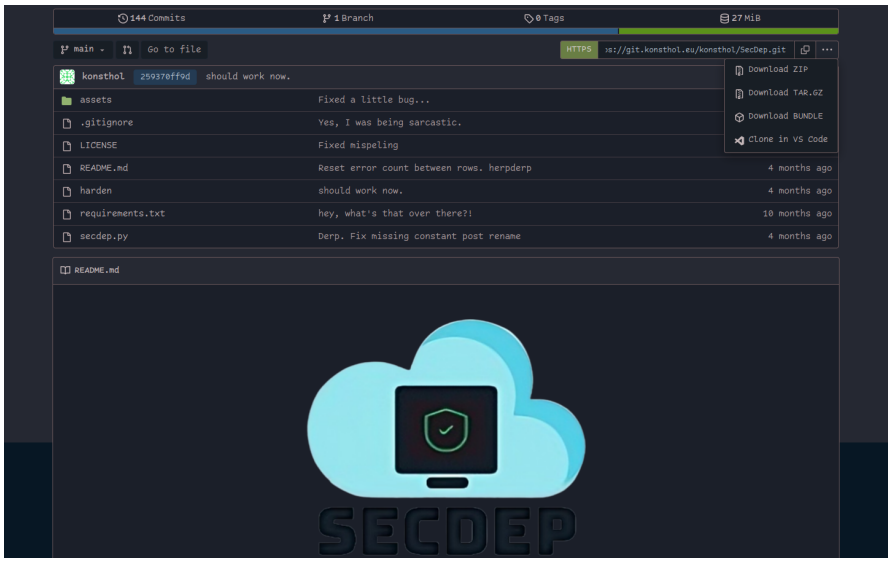
[23] S. Kumar. python-dotenv. URL: `https://github.com/theskumar/python-dotenv`.
[24] Textualize. rich. URL: `https://github.com/Textualize/rich`.
[25] Iterative. shtab. URL: `https://github.com/iterative/shtab`.
[26] paramiko. paramiko. URL: `https://github.com/paramiko/paramiko`.

Κεφάλαιο 5. Εγκατάσταση & Επίδειξη του εργαλείου SecDep

Στην παρούσα ενότητα θα παρουσιαστεί η διαδικασία εγκατάστασης του εργαλείου SecDep, καθώς και η ρύθμισή του για την χρήση των υποδομών νέφους της Amazon. Μετά την ολοκλήρωση των παραπάνω διαδικασιών θα πραγματοποιηθεί επίδειξη της λειτουργίας του με βάση ορισμένα σενάρια χρήσης.

5.1 Εγκατάσταση του πηγαίου κώδικα του SecDep

Για την εκτέλεση του SecDep είναι απαραίτητη προϋπόθεση να υπάρχει εγκατεστημένη η python με έκδοση μεγαλύτερη ή ίση της 3.7. Για την εγκατάσταση του πηγαίου κώδικα του SecDep απαιτείται η χρήση του προγράμματος git ή η πρόσβαση σε έναν φυλλομετρητή. Στην περίπτωση που ο χρήστης δεν έχει εγκατεστημένο το git, πρέπει να μεταβεί στην ιστοσελίδα του αποθετηρίου[1] του κώδικα και να επιλέξει είτε την επιλογή αποθήκευσης ως αρχείο zip, είτε ως tar.gz όπως απεικονίζεται παρακάτω. Έπειτα, το εκφορτωμένο συμπιεσμένο αρχείο πρέπει να αποσυμπιεστεί με το κατάλληλο για τον τύπο αρχείου πρόγραμμα.



Σχήμα 5.1: Αποθήκευση του πηγαίου κώδικα του SecDep ως αρχείο

[1]konsthol. SecDep. 2023. URL: `https://git.konsthol.eu/konsthol/SecDep`.

Η πιο εύκολη μέθοδος για την εγκατάσταση του SecDep είναι η χρήση του git η οποία απαιτεί μονάχα την εκτέλεση της παρακάτω εντολής.

```
git clone https://git.konsthol.eu/konsthol/SecDep.git
```

Εντολή 6: Εγκατάσταση του SecDep με την χρήση του git

5.1.1 Εγκατάσταση των απαιτούμενων εξαρτήσεων

Μετά την ολοκλήρωση της εγκατάστασης του πηγαίου κώδικα του SecDep, αφού ο χρήστης μεταβεί στον φάκελο με τα περιεχόμενα (δηλ. του ριζικού φακέλου του έργου λογισμικού που εκφορτώθηκε) που απεικονίζονται στο Σχήμα 5.1, επιβάλλεται να γίνει χρήση του προγράμματος pip[2] για την εγκατάσταση των βιβλιοθηκών που απαιτούνται για την λειτουργία του εργαλείου. Αυτό επιτυγχάνεται με την εκτέλεση της παρακάτω εντολής.

```
pip install -r requirements.txt
```

Εντολή 7: Εγκατάσταση των απαιτούμενων βιβλιοθηκών για την λειτουργία του SecDep

Στην περίπτωση που η παραπάνω εντολή αποτύχει λόγω πρόσφατων τροποποιήσεων στις τελευταίες εκδόσεις του pip, η εντολή πρέπει να μετατραπεί στην παρακάτω.

```
pip install -r requirements.txt --break-system-packages
```

Εντολή 8: Εξαναγκασμένη μορφή της εντολής εγκατάστασης εξαρτήσεων του SecDep

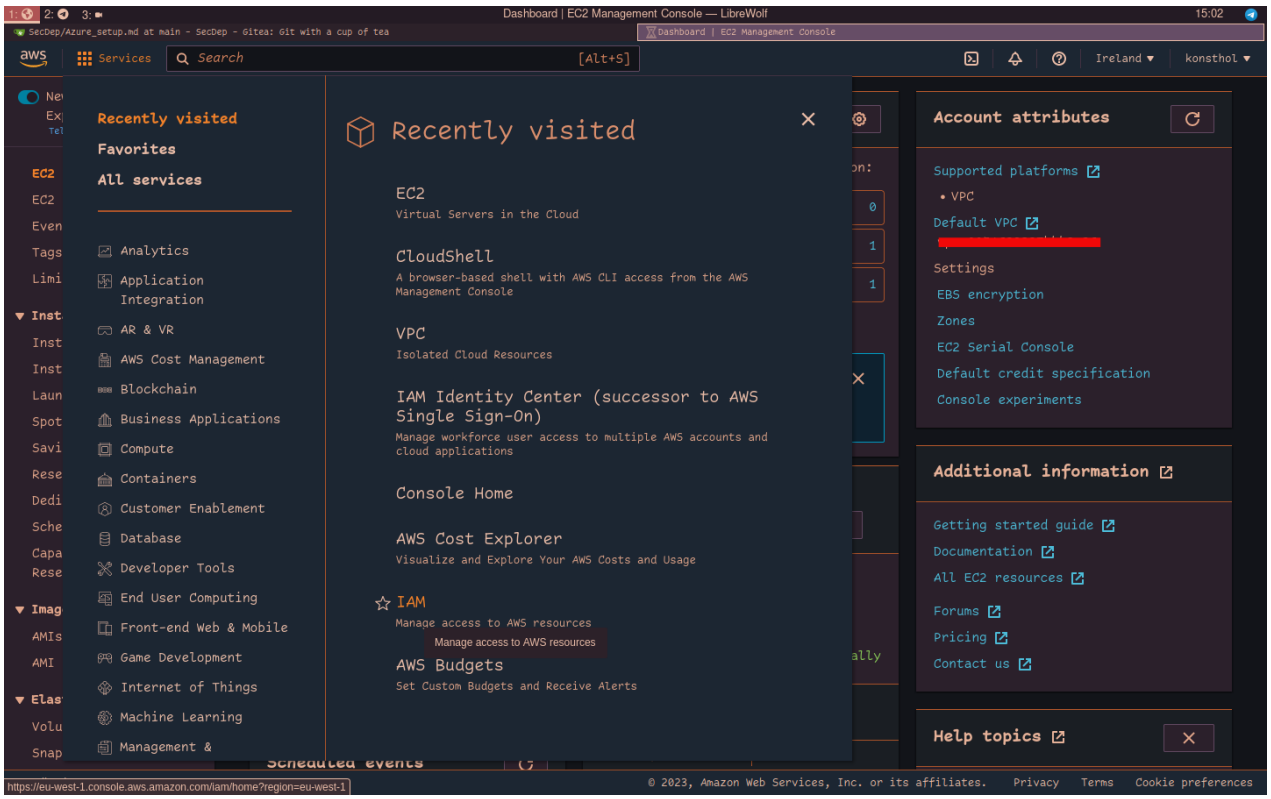
5.2 Ρύθμιση του SecDep

Αφού έχει ολοκληρωθεί η εγκατάσταση των απαιτούμενων εξαρτήσεων, ο χρήστης πρέπει να δώσει στο εργαλείο τα απαραίτητα διαπιστευτήρια για την χρήση των υποδομών νέφους της επιλογής του. Ενώ υποστηρίζονται και οι τρεις μεγαλύτεροι πάροχοι υπηρεσιών IaaS, επιλέχθηκε για την παρούσα ενότητα η κάλυψη της χρήσης των υπηρεσιών της Amazon έναντι της Azure και της GCE, λόγω της ευκολότερης και απλούστερης διαδικασίας αυθεντικοποίησης που παρέχει. Συγκεκριμένα, απαιτεί την συμπλήρωση μονάχα δύο πεδίων. Αυτά στην περίπτωση του εργαλείου μας είναι, τα "SECDEP_AWS_ACCESS_KEY" και "SECDEP_AWS_SECRET_KEY".

[2]pypa. The Python package installer. URL: https://github.com/pypa/pip.

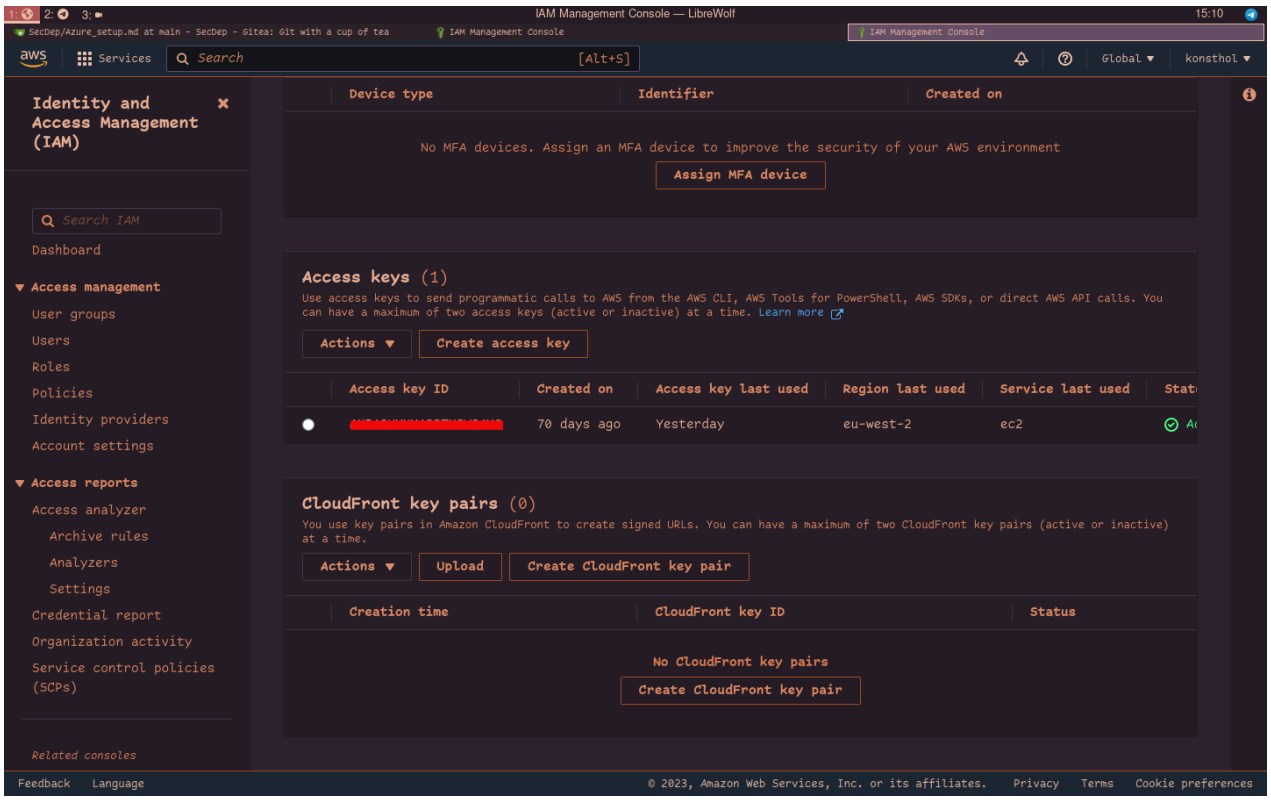
5.2.1 Δημιουργία κλειδιού πρόσβασης με την AWS

Για την απόκτηση των παραπάνω διαπιστευτηρίων, ο χρήστης πρέπει να επισκεφθεί την ιστοσελίδα διαχείρισης του λογαριασμού που έχει δημιουργήσει με την AWS και να μεταβεί στην ενότητα IAM (Identity and Access Management) προκειμένου να δημιουργήσει ένα κλειδί πρόσβασης. Αυτό απεικονίζεται ως εξής:



Σχήμα 5.2: Μετάβαση στην ενότητα IAM της AWS

Έπειτα, πρέπει να εισέλθει στην ενότητα “Manage Access Keys” και να δημιουργήσει ένα νέο κλειδί πρόσβασης όπως φαίνεται παρακάτω:



Σχήμα 5.3: Δημιουργία νέου κλειδιού πρόσβασης

Η αντιστοιχία των απαιτούμενων πεδίων με το κλειδί πρόσβασης που δημιουργήθηκε είναι η εξής:

Πίνακας 5.1: Αντιστοιχία των πεδίων του SecDep με το κλειδί πρόσβασης της AWS

Μεταβλητή του SecDep	Αντιστοιχία
SECDEP_AWS_ACCESS_KEY	Το αναγνωριστικό του κλειδιού
SECDEP_AWS_SECRET_KEY	Το περιεχόμενο του κλειδιού

5.2.2 Αντιστοίχιση των πεδίων του SecDep με το κλειδί πρόσβασης της AWS

Μετά το πέρας της δημιουργίας του κλειδιού πρόσβασης, ο χρήστης πρέπει να αρχικοποιήσει τις μεταβλητές του SecDep με τις τιμές που αντιστοιχούν στην κάθε μια. Υπάρχουν πολλοί τρόποι για την επίτευξη αυτού του σκοπού. Στην προκειμένη περίπτωση που θα γίνει χρήση ενός μονάχα παρόχου, αρκεί η εκτέλεση της παρακάτω εντολής:

```
python3 secdep.py --init aws
```

Εντολή 9: Αρχικοποίηση των μεταβλητών του SecDep για την χρήση της AWS

Έπειτα, θα ζητηθούν από τον χρήστη τα αντίστοιχα διαπιστευτήρια. Οι μεταβλητές αυτές θα αποθηκευτούν στο αρχείο ".env" που θα δημιουργηθεί στον ριζικό φάκελο του έργου. Εάν το εκτελέσιμο μεταφερθεί κάποτε σε νέο φάκελο, δίχως το αρχείο ".env", τότε αυτό θα αναδημιουργηθεί στην νέα τοποθεσία. Στην περίπτωση που ο χρήστης μελλοντικά θέλει να αλλάξει τις τιμές των μεταβλητών ή να προσθέσει νέες, μπορεί να το κάνει αλλάζοντας τα περιεχόμενα του αρχείου ".env" χειροκίνητα με έναν επεξεργαστή κειμένου ή με την εκτέλεση της εντολής:

```
python3 secdep.py --edit
```

Εντολή 10: Επεξεργασία των μεταβλητών του SecDep

Η εντολή αυτή θα προτρέψει τον χρήστη να επιλέξει ποια μεταβλητή να αλλάξει και θα αναγράφει την προηγούμενη τιμή της ώστε να γίνεται ευδιάκριτη η αλλαγή.

5.3 Επίδειξη του SecDep

Ανά πάσα στιγμή, ο χρήστης μπορεί να εκτελέσει την παρακάτω εντολή για να λάβει πληροφορίες σχετικά με τις διαθέσιμες παραμέτρους:

```
python3 secdep.py --help
```

Εντολή 11: Εκτέλεση της παραμέτρου help του SecDep

5.3.1 Δημιουργία εικονικής μηχανής

Σχετικά με την δημιουργία εικονικών μηχανών, που είναι και μια από τις βασικές λειτουργίες του SecDep, ο χρήστης μπορεί να εκτελέσει την παρακάτω εντολή για να δημιουργήσει μια εικονική μηχανή στο νέφος της Amazon, η οποία θα έχει με λειτουργικό σύστημα Debian 11 και θα βρίσκεται στην περιοχή eu-north-1. Το μέγεθος που χρησιμοποιείται είναι το t3.micro, το οποίο είναι διαθέσιμο για χρήστες εντός της δωρεάν δοκιμαστικής περιόδου ενός χρόνου.

Μετά την δημιουργία της εικονικής μηχανής, επιστρέφεται στον χρήση η διεύθυνση IP της, καθώς και η εντολή που μπορεί να εκτελέσει προκειμένου να συνδεθεί σε αυτήν χρησιμοποιώντας το πρωτόκολλο SSH. Η εντολή αυτή σχηματίζεται δυναμικά από το εργαλείο λαμβάνοντας υπόψιν πολλές παραμέτρους. Αρχικά, σε περίπτωση που η εικονική μηχανή έχει σκληρύνει,

```
python3 secdep.py --provider aws --create --name test-node --size t3.micro
↪  --image ami-08869bacfa1188ec9
```

Εντολή 12: Εκτέλεση της παραμέτρου create του SecDep

η θύρα που θα χρησιμοποιηθεί δεν θα είναι η προκαθορισμένη. Έπειτα, το πλήρες μονοπάτι μέχρι το κλειδί SSH, το οποίο δημιουργήθηκε μαζί με την εικονική μηχανή στο ριζικό φάκελο του έργου, θα αποκτηθεί με βάση την τοποθεσία του εκτελέσιμου. Τέλος, η IP διεύθυνση της εικονικής μηχανής θα είναι αυτή που επιστρέφεται στον χρήστη. Επομένως, μόλις ολοκληρωθεί η δημιουργία της εικονικής μηχανής, ο χρήστης είναι σε θέση να αντιγράψει την εντολή αυτή σε ένα τερματικό και να συνδεθεί στην καινούρια του εικονική μηχανή.

Εάν ο χρήστης δεν διαθέτει πρόγραμμα ικανό να συνδεθεί μέσω SSH σε έναν απομακρυσμένο διακομιστή, υποστηρίζεται και η παράμετρος "--ssh". Η συγκεκριμένη παράμετρος θα επιστρέψει στον χρήστη μια λίστα με τις διαθέσιμες εικονικές μηχανές του, ώστε να επιλέξει διαδραστικά μια από αυτές για να συνδεθεί. Μπορεί προαιρετικά να συνδυαστεί με την παράμετρο "--provider" για το φιλτράρισμα της λίστας και με την παράμετρο "--port" για επιλογή θύρας. Το μειονέκτημα που θα έχει ο χρήστης συνδέοντας με αυτόν τον τρόπο, είναι η αδυναμία του έπειτα να εκτελέσει διαδραστικά προγράμματα τύπου TUI (Text-based User Interface) [151], όπως είναι το htop[3] και το vim[4]. Μια περίπτωση εκτέλεσής της είναι η εξής:

```
python3 secdep.py --provider aws --ssh
```

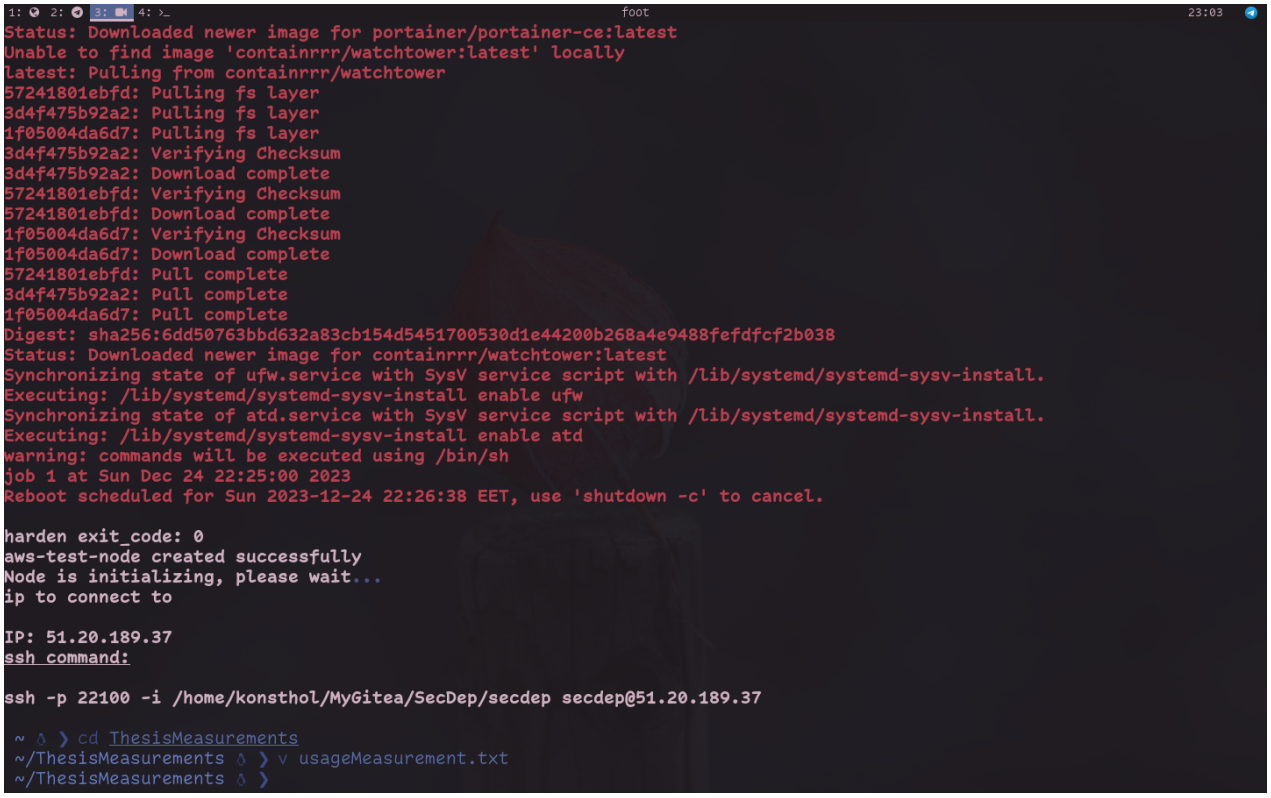
Εντολή 13: Εκτέλεση της παραμέτρου ssh του SecDep

Σχετικά με την εντολή δημιουργίας εικονικής μηχανής, χωρίς την παράμετρο "--yes", θα ζητηθεί από τον χρήστη να επιβεβαιώσει την δημιουργία της εικονικής μηχανής, παρέχοντας πληροφορίες σχετικά με τις επιλογές του. Η συγκεκριμένη εντολή, μπορεί επιπλέον να συνδυαστεί με την παράμετρο "--deploy". Με την προσθήκη της παραμέτρου αυτής, θα εκτελεστεί η ίδια διαδικασία αλλά με την διαφορά πως θα πραγματοποιηθεί εκτέλεση και του αρχείου harden στην εικονική μηχανή, το οποίο θα σκληρύνει το λειτουργικό της σύστημα και θα εγκαταστήσει την μηχανή δοχείων Docker, την οποία επίσης θα σκληρύνει. Ένα πλήρες παράδειγμα όπου γίνεται χρήση της παραμέτρου αυτής και εγκαθίστανται στην εικονική μηχανή δύο δοχεία Docker, αποτελεί η εκτέλεση της Εντολής 14. Παράλληλα, το αποτέλεσμα μετά την εκτέλεσή της απεικονίζεται στο Σχήμα 5.4.

[3]H. Muhammad. htop. URL: https://htop.dev/.
[4]B. Moolenaar. Vim. URL: https://github.com/vim/vim.

```
python3 secdep.py --provider aws --create --name test-node --size t3.micro
↪  --image ami-08869bacfa1188ec9 --yes --deploy node mysql
```

Εντολή 14: Εκτέλεση της παραμέτρου create του SecDep, με την παράμετρο deploy για σκλήρυνση



Σχήμα 5.4: Δημιουργία εικονικής μηχανής με παράμετρο για σκλήρυνση

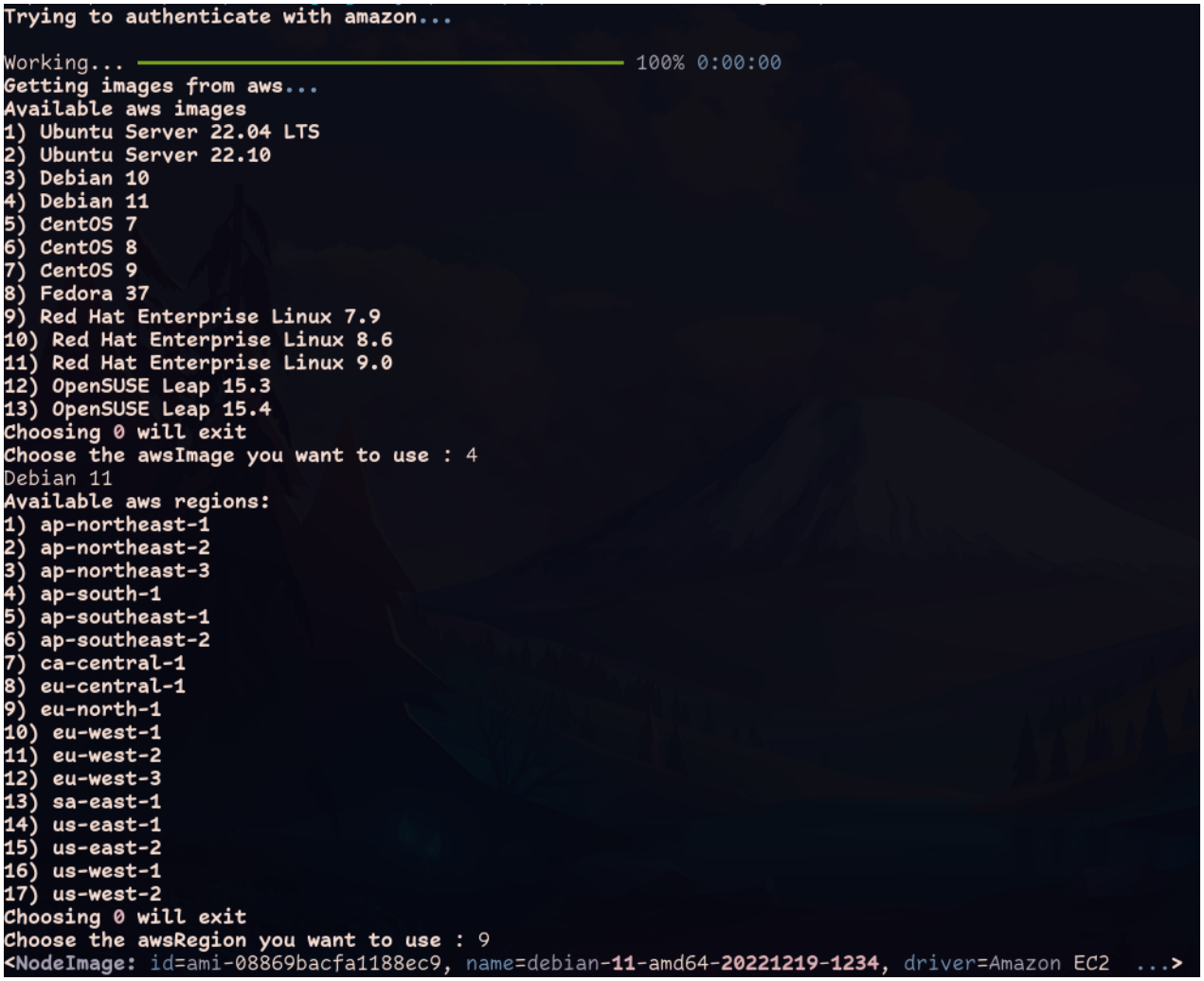
Με μια ακόμα προσθήκη παραμέτρου, συγκεκριμένα της "--docker_compose", θα εκτελεστεί στην εικονική μηχανή και ένα αρχείο docker-compose.yml που θα πρέπει ο χρήστης να έχει μεταφέρει στον ριζικό φάκελο του SecDep. Για οποιαδήποτε απαραίτητη παράμετρο δεν έχει δοθεί, ή έχει δοθεί λανθασμένα, θα προτρέπεται ο χρήστης να την δώσει ξανά έως ότου ολοκληρωθεί η διαδικασία ή ακυρωθεί από τον χρήστη.

5.3.2 Εύρεση πληροφοριών πόρων

Εάν ο χρήστης δεν γνωρίζει τις διαθέσιμες επιλογές για τις παραμέτρους που εισάγει, υποστηρίζονται ξεχωριστές εντολές, οι οποίες ακολουθούν παρόμοιο μοτίβο μεταξύ τους. Παραδείγματα εντολών που υποστηρίζονται, είναι της μορφής που παρατηρείται στην Εντολή 15. Η οθόνη επιλογής πόρου που εμφανίζεται στον χρήστη μετά την εκτέλεση αυτού του είδους εντολών, απεικονίζεται στο Σχήμα 5.5.

```
python3 secdep.py --provider aws --listimages --print
```

Εντολή 15: Εκτέλεση της παραμέτρου listimages του SecDep



Σχήμα 5.5: Οθόνη επιλογής πόρου

Το αποτέλεσμα της Εντολής 15, είναι να εμφανιστεί μια λίστα με τις διαθέσιμες εικόνες που μπορεί να χρησιμοποιήσει ο χρήστης για την δημιουργία εικονικών μηχανών. Έπειτα, με την διαδραστική επιλογή μιας από αυτές, θα εμφανιστούν πληροφορίες σχετικά με την εικόνα που επιλέχθηκε, οι οποίες θα είναι της μορφής που απεικονίζεται παρακάτω.

```
<NodeImage: id=ami-0eb2c4104acb437b2, name=debian-10-amd64-20221224-1239,
↪  driver=Amazon EC2  ...>
```

Από την παραπάνω πληροφορία, ο χρήστης θα πρέπει να καταγράψει το αναγνωριστικό (id) της εικόνας. Εάν χρειάζεται πληροφορίες σχετικά με τα διαθέσιμα μεγέθη των εικονικών μηχανών, αυτό επιτυγχάνεται με την εντολή:

```
python3 secdep.py --provider aws --listsizes --print
```

Εντολή 16: Εκτέλεση της παραμέτρου listsizes του SecDep

Η διαδικασία επιλογής του μεγέθους είναι παρόμοια με την επιλογή της εικόνας, ενώ το αποτέλεσμα είναι της μορφής:

```
<NodeSize: id=t3.micro, name=t3.micro, ram=1024, disk=0, bandwidth=0,
↪  price=0.0204, driver=Amazon EC2  ...>
```

Από την παραπάνω πληροφορία, το σημαντικό πάλι είναι το αναγνωριστικό (id) του μεγέθους.

5.3.3 Λίστα εικονικών μηχανών

Μερικές από τις εντολές που δύναται να χρησιμοποιήσει ένας χρήστης συχνά μπορεί να είναι για την εμφάνιση εικονικών μηχανών ή την διαγραφή τους. Η εμφάνιση εικονικών μηχανών επιτυγχάνεται με την εκτέλεση της Εντολής 17. Το αποτέλεσμά της μπορεί να απεικονίζεται όπως στο Σχήμα 5.6.

```
python3 secdep.py --provider aws --awsregion us-east-2 --list
```

Εντολή 17: Εκτέλεση της παραμέτρου list του SecDep

Στο Σχήμα 5.6, βλέπουμε πως υπήρχε στην περιοχή "us-east-2" μια εικονική μηχανή, στην οποία είχε δοθεί το όνομα "test-node". Για κάθε πάροχο, συνδυάζεται το όνομά του στο όνομα της εικονικής μηχανής που δημιουργείται μέσω αυτού, με σκοπό την ταχύτερη διάκρισή της όταν ο χρήστης ζητάει λίστα με εικονικές μηχανές κατά μήκος πολλών παρόχων. Η συγκεκριμένη είχε ήδη διαγραφεί και για τον λόγο αυτό, η κατάστασή της δηλώνεται ως "terminated". Σε αντίθετη περίπτωση, θα βλέπαμε την τρέχουσα κατάστασή της και τις διάφορες πληροφορίες της. Στις πληροφορίες αυτές περιλαμβάνονται, η δημόσια και ιδιωτική διεύθυνση IP της, το μέγεθός της, η διανομή που χρησιμοποιεί, καθώς και λοιπές πληροφορίες σε μορφή JSON, τα περιεχόμενα των οποίων διαφέρουν από πάροχο σε πάροχο.

Η παράμετρος "--awsregion" είναι προαιρετική και χρησιμοποιείται αποκλειστικά για την Amazon. Η παράλειψή της θα είχε ως αποτέλεσμα να γίνει έρευνα σε όλες τις διαθέσιμες περιοχές. Γεγονός που ενδεχομένως να αποτελούσε σπατάλη χρόνου.

Σχήμα 5.6: Λίστα με τις εικονικές μηχανές που έχουν δημιουργηθεί

5.3.4 Διαγραφή εικονικής μηχανής

Με την παρακάτω εντολή, ο χρήστης λαμβάνει μια λίστα με τις διαθέσιμες εικονικές μηχανές του παρόχου Amazon που έχουν δημιουργηθεί στην περιοχή eu-north-1 και επιλέγει μια από αυτές για διαγραφή. Μετά από επιτυχή διαγραφή, λαμβάνει επιβεβαίωση στην οθόνη του.

```
python3 secdep.py --provider aws --action delete --awsregion eu-north-1
```

Εντολή 18: Εκτέλεση της παραμέτρου delete του SecDep

Εάν ήθελε να διαγράψει όλες τις εικονικές μηχανές της περιοχής, η τιμή της παραμέτρου "--action", από "delete" θα έπρεπε να αντικατασταθεί σε "deleteall". Με παρόμοιο τρόπο υποστηρίζονται και εντολές για εκκίνηση, επανεκκίνηση ή παύση εικονικών μηχανών.

5.4 Περισσότερες πληροφορίες

Το SecDep, ως ένα ολοκληρωμένο εργαλείο, παρέχει αρκετές λειτουργίες για την δημιουργία και διαχείριση εικονικών μηχανών κατά μήκος πολλών παρόχους νέφους. Τα παραδείγματα εντολών που καλύψαμε στην παρούσα ενότητα, περιορίστηκαν σε έναν μόνο πάροχο νέφους. Ωστόσο, ορισμένες από τις διαθέσιμες εντολές του διαφοροποιούνται για τους υπόλοιπους.

Πιο λεπτομερείς οδηγίες χρήσης και εγκατάστασης περιέχονται και στην σελίδα του αποθετηρίου του SecDep[5] μέσω του αρχείου "README.md". Στην αρχική του σελίδα, εμφανίζονται τα περιεχόμενα του αρχείου οδηγιών. Εκεί, ο χρήστης μπορεί να βρει πληροφορίες σχετικά με την παραμετροποίηση και των υπόλοιπων παρόχων νέφους, χρήσιμες ρυθμίσεις που καθιστούν την εκτέλεση του SecDep πιο εύκολη, καθώς και περισσότερες οθόνες εκτέλεσης.

[5]konsthol. SecDep. 2023. URL: `https://git.konsthol.eu/konsthol/SecDep`.

Κεφάλαιο 6. Πειραματική Αποτίμηση Εργαλείου

Το εργαλείο που αναπτύχθηκε στα πλαίσια της διπλωματικής αυτής εργασίας, έχει τρεις βασικούς στόχους. Την δημιουργία εικονικών μηχανών κατά μήκος πολλών παρόχων νέφους, την σκλήρυνση του λειτουργικού συστήματός τους και την σκλήρυνση της μηχανής δοχείων Docker που εγκαθιστά σε αυτές. Στην παρούσα ενότητα θα παρουσιαστούν τα αποτελέσματα της αξιολόγησης της ασφάλειας των εικονικών μηχανών που δημιουργήθηκαν, χρησιμοποιώντας τρία διαφορετικά εργαλεία αξιολόγησης. Τα εργαλεία αυτά, με την σειρά που θα αναλυθούν είναι το Vuls, το Lynis και το LUNAR.

6.1 Εργαλεία Αξιολόγησης

Όλα τα εργαλεία αξιολόγησης που χρησιμοποιήθηκαν στην παρούσα εργασία, είναι εργαλεία ανοιχτού κώδικα και διατίθενται δωρεάν στο κοινό. Τα εργαλεία αυτά επιλέχθηκαν με βάση την δημοτικότητά τους, την ευκολία εγκατάστασης και χρήσης τους και την ποικιλία των ελέγχων που πραγματοποιούν. Στην συνέχεια θα παρουσιαστούν τα εργαλεία αυτά και ο τρόπος με τον οποίο χρησιμοποιήθηκαν.

6.1.1 Vuls: VULnerability Scanner

Το Vuls[1] είναι ένα εργαλείο ανοιχτού κώδικα, το οποίο αναπτύχθηκε από την Future Corp χρησιμοποιώντας την γλώσσα προγραμματισμού Go. Πρόκειται για ένα εργαλείο αξιολόγησης ασφάλειας, το οποίο είναι σε θέση να εντοπίσει ευπάθειες σε τοπικούς ή και απομακρυσμένους διακομιστές χρησιμοποιώντας δεδομένα από πολλές διαφορετικές πηγές. Με βάση την σελίδα του στο GitHub[2], στις πηγές αυτές περιλαμβάνονται:

[1] K. Kanbe. Vuls. URL: `https://vuls.io/`.
[2] future-architect. Vuls. URL: `https://github.com/future-architect/vuls`.

- **Το NVD (National Vulnerability Database)**
- **Τα αρχεία ορισμού OVAL (Open Vulnerability and Assessment Language)** που διατίθενται από τις διανομές Linux
- **Συμβουλευτικές πηγές ασφαλείας**
  - Alpine-secdb
  - Red Hat Security Advisories
  - Debian Security Bug Tracker
  - Ubuntu CVE Tracker
  - Microsoft CVRF
- **Πηγές με PoC (Proof of Concept) εκμετάλλευσης ευπαθειών**
  - Exploit Database
  - Metasploit-Framework modules
- **Η CERT (Computer Emergency Readiness Team)**
- **Η CISA (Cybersecurity & Infrastructure Security Agency)**
- **Οι Cyber Threat Intelligence(MITRE ATT&CK and CAPEC)**
- **Βιβλιοθήκες ευπαθειών όπως αυτή της Aqua Security**

Ο χρήστης του εργαλείου απαιτείται να έχει εγκατεστημένο το Vuls σε έναν υπολογιστή και να έχει αποθηκεύσει βάσει των οδηγιών εγκατάστασης, τα δεδομένα των πηγών που θα χρησιμοποιηθούν. Έπειτα σε ένα αρχείο toml ρυθμίζονται τα διαπιστευτήρια των διακομιστών που θα αξιολογηθούν διότι η απομακρυσμένη σύνδεση θα πραγματοποιηθεί με χρήση SSH. Υπάρχει η δυνατότητα δύο επιπέδων αξιολόγησης. Αυτά είναι τα παρακάτω:

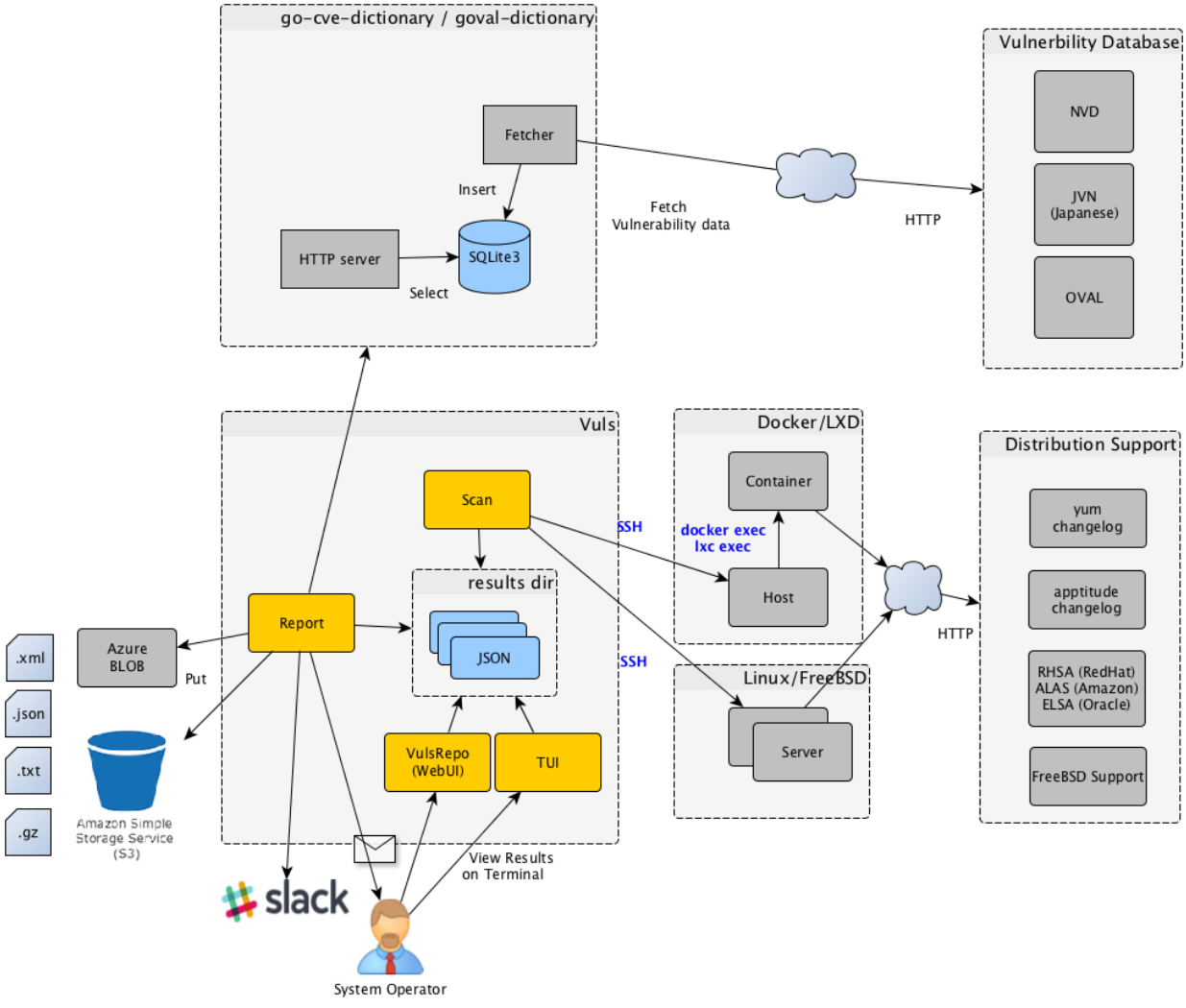
- **Γρήγορη σάρωση**:

  Σε αυτό το επίπεδο αξιολόγησης, το Vuls δεν απαιτεί την σύνδεση με χρήστη διαχειριστικών δικαιωμάτων στον απομακρυσμένο διακομιστή αλλά ούτε και την ύπαρξη πακέτων λογισμικού για την υποβοήθηση της διαδικασίας σάρωσης. Επιπροσθέτως, επιβάλλει ελάχιστο φόρτο εργασιών στον διακομιστή που αξιολογείται.

- **Εκτεταμένη σάρωση**:

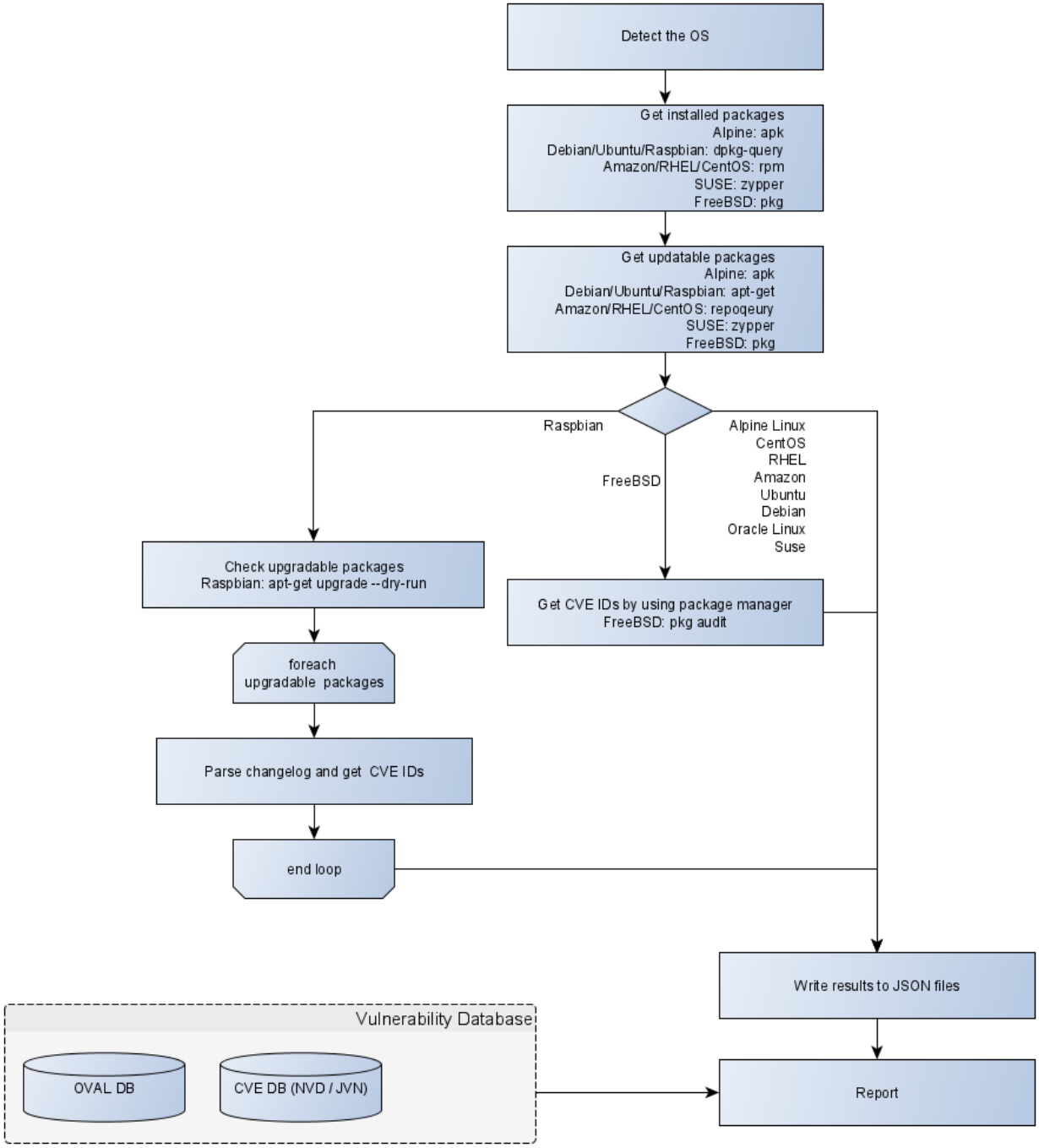
  Στο δεύτερο επίπεδο αξιολόγησης, το Vuls χρειάζεται να συνδεθεί ως ένας χρήστης ικανός να εκτελέσει εντολές διαχειριστικού επιπέδου. Η σάρωση που πραγματοποιείται είναι πιο εκτεταμένη και περιλαμβάνει ελέγχους που απαιτούν την ύπαρξη πακέτων λογισμικού στον απομακρυσμένο διακομιστή όπως τα lsof, debian-goodies και reboot-notifier.

Στο Σχήμα 6.1, όπως αυτό πάρθηκε από την επίσημη ιστοσελίδα του εργαλείου, απεικονίζεται η αρχιτεκτονική του Vuls κατά την διαδικασία σάρωσης απομακρυσμένου διακομιστή.



Σχήμα 6.1: Αρχιτεκτονική του Vuls [156]

Στο Σχήμα 6.2, το οποίο παρομοίως αποκτήθηκε από την επίσημη σελίδα του, απεικονίζεται η διαδικασία που ακολουθείται κατά την εκτέλεση της σάρωσης ενός συστήματος.



Σχήμα 6.2: Διαδικασία εκτεταμένης σάρωσης του Vuls [157]

6.1.2 Lynis - Security auditing and hardening tool, for UNIX-based systems.

Το Lynis[3] είναι το δεύτερο εργαλείο ανοιχτού κώδικα που χρησιμοποιήθηκε για την αξιολόγηση της ασφάλειας των εικονικών μηχανών. Σε αντίθεση με το Vuls, το Lynis εκτελείται αποκλειστικά στον απομακρυσμένο διακομιστή. Πραγματοποιεί μια εκτενή ανάλυση του συστήματος, εντοπίζοντας απροσεξίες ή ελλείψεις ρυθμίσεων που μπορεί να αποτελέσουν σημεία εισόδου για επιτιθέμενους με βάση τις βέλτιστες πρακτικές ασφαλείας. Έπειτα, επιστρέφει στον χρήστη μια λεπτομερή αναφορά με τα ευρήματά του και προτάσεις για την επίλυσή τους. Στο τελευταίο κομμάτι της αναφοράς, παρέχεται και μια βαθμολογία της ασφάλειας του συστήματος, η οποία κυμαίνεται από το 0 έως το 100. Εκτελείται εύκολα σε κάθε σύστημα Linux διότι πρόκειται για ένα απλό εκτελέσιμο αρχείο bash.

Στο Σχήμα 6.3 απεικονίζεται ένα κομμάτι της αναφοράς που επιστρέφει.

```
[+] Users, Groups and Authentication
------------------------------------
  - Search administrator accounts...                [ OK ]
  - Checking UIDs...                                [ OK ]
  - Checking chkgrp tool...                         [ FOUND ]
  - Consistency check /etc/group file...            [ OK ]
  - Test group files (grpck)...                     [ OK ]
  - Checking login shells...                        [ WARNING ]
  - Checking non unique group ID's...               [ OK ]
  - Checking non unique group names...              [ OK ]
  - Checking LDAP authentication support            [ NOT ENABLED ]
  - Check /etc/sudoers file                         [ NOT FOUND ]

[ Press [ENTER] to continue, or [CTRL]+C to stop ]


[+] Shells
------------------------------------
  - Checking console TTYs...                        [ WARNING ]
  - Checking shells from /etc/shells...
    Result: found 6 shells (valid shells: 6).

[ Press [ENTER] to continue, or [CTRL]+C to stop ]


[+] File systems
------------------------------------
  - [FreeBSD] Querying UFS mount points (fstab)...  [ OK ]
  - Query swap partitions (fstab)...                [ OK ]
  - Testing swap partitions...                      [ OK ]
  - Checking for old files in /tmp...               [ WARNING ]
  - Checking /tmp sticky bit...                     [ OK ]
```

Σχήμα 6.3: Κομμάτι της αναφοράς του Lynis [158]

[3]CISOfy. Lynis. URL: https://cisofy.com/lynis/.

6.1.3 LUNAR - Lockdown UNix Auditing and Reporting

Το τελευταίο εργαλείο αξιολόγησης που χρησιμοποιήθηκε είναι το LUNAR[4]. Πρόκειται για ένα εκτελέσιμο αρχείο bash, το οποίο όπως και το Lynis, εκτελείται στον διακομιστή του οποίου την ασφάλεια αξιολογεί. Ο τρόπος λειτουργίας του είναι προσεγγίσιμος και απλός. Εκτελεί μια σειρά ελέγχων στο σύστημα χρησιμοποιώντας το κριτήριο αναφοράς CIS (Center for Internet Security)[5], καθώς και άλλα πλαίσια και επιστρέφει στον χρήστη μια αναφορά με τα ευρήματά του. Παρομοίως με το Lynis, στο τέλος της αναφοράς του περιέχεται ο αριθμός των ελέγχων που εκτελέστηκαν, σε συνδυασμό με τον αριθμό αποτυχιών και επιτυχιών τους.

6.2 Προετοιμασία περιβαλλόντων προς αξιολόγηση

Για την αποτίμηση της ασφάλισης του συστήματος δημιουργήθηκαν δύο εικονικές μηχανές, χρησιμοποιώντας τον πάροχο νέφους AWS. Από αυτές, μονάχα για την δεύτερη επιλέχθηκε η επιλογή σκλήρυνσης. Για το λειτουργικό σύστημα τους, επιλέχθηκε το AMI με αναγνωριστικό κωδικό ami-08869bacfa1188ec9, το οποίο αντιστοιχεί στην διανομή Debian 11 και στην περιοχή eu-north-1, η οποία μεταφράζεται ως Europe (Stockholm). Για τα τεχνικά χαρακτηριστικά των εικονικών μηχανών χρησιμοποιήθηκε το t3.micro με τα προνόμια της δοκιμαστικής περιόδου ενός χρόνου που παρέχει η Amazon για νέους χρήστες. Το t3.micro υπάγεται στην σειρά υπολογιστών γενικού σκοπού με τα εξής χαρακτηριστικά, όπως αυτά αναφέρονται στην επίσημη ιστοσελίδα της Amazon[6] και στην σελίδα του Vantage[7]:

- **Τιμή On Demand**: 0.01$ ανά ώρα (για συστήματα Linux)
- **Μνήμη**: 1 GiB
- **vCPU**: 2
- **Μνήμη ανά vCPU**: 0.5 GiB
- **Επεξεργαστής**: Intel Skylake E5 2686 v5
- **Ταχύτητα ρολογιού**: 3.1 GHz
- **Ταχύτητα δικτύου**: 5 Gbps

[4]L. Blast. Lunar. URL: https://github.com/lateralblast/lunar.

[5]CIS. CIS. URL: https://www.cisecurity.org/.

[6]A. W. Services. Amazon EC2 T3 Instances. URL: https://aws.amazon.com/ec2/instance-types/t3/.

[7]Vantage. t3.micro. URL: https://instances.vantage.sh/aws/ec2/t3.micro.

- **Αποθηκευτικός χώρος**: 30 GiB (EBS)

Οι εντολές που χρησιμοποιήθηκαν για την δημιουργία των εικονικών μηχανών, έχουν την παρακάτω μορφή (π.χ. δείτε Εντολή 19) διότι έχει εισαχθεί το εκτελέσιμο αρχείο secdep.py στο PATH του συστήματος και έχει δηλωθεί ως alias του το secdep. Στην περίπτωση που αυτά τα βήματα δεν έχουν ληφθεί θα πρέπει οι εντολές να είναι της μορφής:

```
python3 secdep.py <παράμετροι χρήστη>
```

Χρησιμοποιήθηκε η εξής εντολή για την δημιουργία της πρώτης εικονικής μηχανής (VM_1) χωρίς[8] σκλήρυνση λειτουργικού συστήματος (χρήση του secdep.py):

```
secdep -P aws -c -n test-node -s t3.micro -i ami-08869bacfa1188ec9 --yes
```

Εντολή 19: SecDep - Δημιουργία εικονικής μηχανής χωρίς σκλήρυνση συστήματος

Για την δημιουργία εικονικής μηχανής με σκλήρυνση[9] λειτουργικού συστήματος (VM_2), εγκατάσταση και σκλήρυνση της μηχανής δοχείων Docker και εισαγωγής ενός αρχείου docker-compose.yml που βρίσκεται στον ίδιο φάκελο με το secdep.py, χρησιμοποιήθηκε η εντολή (χρήση του secdep.py και έπειτα του harden):

```
secdep -P aws -c -n test-node -s t3.micro -i ami-08869bacfa1188ec9 --yes
↪  --docker_compose --deploy
```

Εντολή 20: SecDep - Δημιουργία εικονικής μηχανής με σκλήρυνση συστήματος

6.3 Εγκατάσταση/Χρήση των εργαλείων αξιολόγησης

Αφότου έχουν δημιουργηθεί οι εικονικές μηχανές, μπορεί να ξεκινήσει η διαδικασία αξιολόγησης της ασφάλειάς τους. Αρχικά, θα παρουσιαστεί η διαδικασία εγκατάστασης του κάθε εργαλείου και έπειτα θα παρουσιαστεί η διαδικασία χρήσης τους.

[8] Ο χρόνος εκτέλεσης της είναι κατά μέσο όρο 37 δευτερόλεπτα.

[9] Η εντολή αυτή έχει χρόνο εκτέλεσης 2 λεπτά και 55 δευτερόλεπτα. Επομένως, φαίνεται πως η σκλήρυνση του συστήματος (δηλ. μιας εικονικής μηχανής) έχει ένα επίβαρο επίδοσης.

6.3.1 Εγκατάσταση και προετοιμασία του Vuls

Για την εγκατάσταση του Vuls, αρκεί σε έναν υπολογιστή της επιλογής μας να εκτελεστεί η παρακάτω εντολή, η οποία λαμβάνει το αρχείο εντολών εγκατάστασης και το εκτελεί με το κέλυφος bash:

```
bash <( curl -s \
https://raw.githubusercontent.com\
/vulsio/vulsctl/master/install-host/install.sh )
```

Εντολή 21: Εγκατάσταση του Vuls

Ορθή πρακτική θα ήταν να γίνει έλεγχος του περιεχομένου του αρχείου πριν την εκτέλεση της εντολής αυτής για να βεβαιωθεί κανείς ότι δεν περιέχει κακόβουλο κώδικα. Στην συνέχεια, θα πρέπει να δημιουργηθεί ο φάκελος που θα περιέχει τα δεδομένα των πηγών από όπου και θα αντλεί τις πληροφορίες ευπαθειών, καθώς και ένα αρχείο config.toml στο οποίο θα δηλωθούν τα διαπιστευτήρια των διακομιστών που θα αξιολογηθούν.

Τα παραπάνω επιτυγχάνονται με τις Εντολές 22 και 23 που ακολουθούν:

```
mkdir -p /usr/share/vuls-data
```

Εντολή 22: Δημιουργία φακέλου δεδομένων του Vuls

Η Εντολή 23 θα μπορούσε να παραληφθεί αν ο χρήστης επιθυμεί να δημιουργήσει το αρχείο ρυθμίσεων του Vuls εξ ολοκλήρου με χειροκίνητο τρόπο. Μετά την εκτέλεσή της, ο χρήστης θα πρέπει σε κάθε περίπτωση να συμπληρώσει τις κατάλληλες τιμές στα πεδία που αφορούν τους διακομιστές που θα αξιολογηθούν. Παρ' όλα αυτά, έχει συμπεριληφθεί για λόγους διευκόλυνσής του.

```
cat <<EOF > /usr/share/vuls-data/config.toml
[cveDict]
type = "sqlite3"
SQLite3Path = "/usr/share/vuls-data/cve.sqlite3"

[ovalDict]
type = "sqlite3"
SQLite3Path = "/usr/share/vuls-data/oval.sqlite3"

[gost]
type = "sqlite3"
SQLite3Path = "/usr/share/vuls-data/gost.sqlite3"

[metasploit]
type = "sqlite3"
SQLite3Path = "/usr/share/vuls-data/go-msfdb.sqlite3"

[servers]

[servers.secdepawsFresh]
host = "<ip του διακομιστή>"
port = "22"
user = "secdep"
keyPath = "<πλήρες μονοπάτι ιδιωτικού κλειδιού SSH>"
scanMode = [ "deep" ]

[servers.secdepawsHardened]
host = "<ip του διακομιστή>"
port = "22100"
user = "secdep"
keyPath = "<πλήρες μονοπάτι ιδιωτικού κλειδιού SSH>"
scanMode = [ "deep" ]
EOF
```

Εντολή 23: Δημιουργία και αρχικοποίηση του αρχείου ρυθμίσεων του Vuls

Για να μπορέσει ο διακομιστής στον οποίο είναι εγκατεστημένο το Vuls να συνδεθεί στους διακομιστές της Amazon, απαιτείται η εκτέλεση δύο εντολών της μορφής:

```
ssh-keyscan -H -p <θύρα SSH> <ip του διακομιστή> >> ~/.ssh/known_hosts
```

Εντολή 24: Εισαγωγή του διακομιστή της Amazon στο known_hosts

Επιπροσθέτως, θα πρέπει να μεταφερθεί στον υπολογιστή που είναι εγκατεστημένο το Vuls, το ιδιωτικό κλειδί του διακομιστή (ένα αρχείο ονόματι secdep), το οποίο δημιουργείται στον ίδιο φάκελο που βρίσκεται το secdep.py. Για την αρχικοποίηση της βάσης δεδομένων ευπαθειών του Vuls, εκτελούνται με σειρά οι εντολές:

```
cd /usr/share/vuls-data
```

Εντολή 25: Μεταφορά στον φάκελο δεδομένων του Vuls

```
gost fetch debian --dbpath /usr/share/vuls-data/gost.sqlite3
```

Εντολή 26: Απόκτηση δεδομένων μέσω της εντολής gost

```
go-cve-dictionary fetch nvd --dbpath /usr/share/vuls-data/cve.sqlite3
```

Εντολή 27: Απόκτηση δεδομένων μέσω της εντολής go-cve-dictionary

```
goval-dictionary fetch debian 11 --dbpath /usr/share/vuls-data/oval.sqlite3
```

Εντολή 28: Απόκτηση δεδομένων μέσω της εντολής goval-dictionary

```
go-msfdb fetch msfdb --dbpath /usr/share/vuls-data/go-msfdb.sqlite3
```

Εντολή 29: Απόκτηση δεδομένων μέσω της εντολής go-msfdb

Έπειτα, στο σύστημα του διακομιστή, πρέπει να εγκατασταθούν ορισμένα πακέτα λογισμικού και να εισαχθεί ο χρήστης secdep στο αρχείο sudoers με το περιεχόμενο "secdep ALL=(ALL) NOPASSWD:ALL" ώστε να εκτελεί εντολές που απαιτούν διαχειριστικά δικαιώματα δίχως την ανάγκη εισαγωγής κωδικού. Αυτά επιτυγχάνονται με τις παρακάτω εντολές:

```
sudo visudo
```

Εντολή 30: Άνοιγμα αρχείου sudoers

```
sudo apt update
```

Εντολή 31: Ενημέρωση αποθετηρίων λογισμικού

```
sudo apt install -y debian-goodies reboot-notifier aptitude lsof
```

Εντολή 32: Εγκατάσταση πακέτων λογισμικού

Για τον έλεγχο συνδεσιμότητας στους διακομιστές χρειάζεται να εκτελεστεί η εντολή:

```
vuls configtest
```

Εντολή 33: Έλεγχος συνδεσιμότητας του Vuls

Αφότου έχουν εκτελεστεί όλα τα παραπάνω βήματα και το αποτέλεσμα της εντολής 33 περιλαμβάνει τα ονόματα των διακομιστών, όπως αυτά ορίσθηκαν στο αρχείο ρυθμίσεων, ο διακομιστής είναι έτοιμος για αξιολόγηση. Αυτό επιτυγχάνεται με την εντολή:

```
vuls scan <όνομα διακομιστή>
```

Εντολή 34: Εκκίνηση αξιολόγησης μέσω του Vuls

Μετά το πέρας της διαδικασίας αξιολόγησης, μπορούμε να εκτελέσουμε την παρακάτω εντολή προκειμένου να παραχθεί η αναφορά σε μορφή json:

```
vuls report -format-json
```

Εντολή 35: Παραγωγή αναφοράς σε μορφή json

Για να μπορέσουμε να δούμε τα αποτελέσματα σε μια ευανάγνωστη μορφή, μπορούμε να κατεβάσουμε στον διακομιστή μας το εργαλείο VulsRepo[10]. Θα πρέπει να μεταφερθούμε σε έναν φάκελο της επιλογής μας και έπειτα να το κατεβάσουμε μέσω της εντολής git και να αρχικοποιήσουμε το αρχείο ρυθμίσεών του. Αυτά τα βήματα επιτυγχάνονται με τις παρακάτω εντολές:

```
cd /opt
```

Εντολή 36: Μεταφορά στον φάκελο εγκατάστασης του VulsRepo

[10]ishiDACo. VulsRepo. URL: https://github.com/ishiDACo/vulsrepo.

```
git clone https://github.com/ishiDACo/vulsrepo.git
```

Εντολή 37: Εγκατάσταση του VulsRepo

```
cd /opt/vulsrepo/server/
```

Εντολή 38: Μεταφορά στον φάκελο του εκτελέσιμου αρχείου του VulsRepo

Η Εντολή 40 μπορεί να παραληφθεί αν ο χρήστης αρχικοποίησε το αρχείο χειροκίνητα μετά την Εντολή 39. Αλλιώς, θα πρέπει να εκτελεστεί η Εντολή 40 και έπειτα να συμπληρωθούν οι απαραίτητες τιμές στο αρχείο ρυθμίσεων.

```
cp vulsrepo-config.toml.sample vulsrepo-config.toml
```

Εντολή 39: Αρχικοποίηση του αρχείου ρυθμίσεων του VulsRepo

```
cat <<EOF > /opt/vulsrepo/server/vulsrepo-config.toml
[server]
rootPath = "/opt/vulsrepo"
resultsPath  = "/usr/share/vuls-data/results"
serverPort  = "<θύρα του VulsRepo>"
serverIP = "<ip του διακομιστή του Vuls>"
EOF
```

Εντολή 40: Προσαρμογή του αρχείου ρυθμίσεων του VulsRepo

Τέλος, για να εκκινήσει το VulsRepo, εκτελούμε την εντολή:

```
./vulsrepo-server
```

Εντολή 41: Εκκίνηση του VulsRepo

6.3.2 Εγκατάσταση και προετοιμασία του Lynis

Σε αντίθεση με το Vuls, η εγκατάσταση του Lynis και η λειτουργία του είναι πιο απλή. Εδώ τα βήματα που πρέπει να ακολουθηθούν είναι η σύνδεση στους διακομιστές της Amazon μέσω SSH (για την οποία η εντολή είναι έτοιμη μετά την δημιουργία κάθε διακομιστή), η εγκατάσταση του εργαλείου μέσω git και η εκτέλεσή του. Αυτά πραγματοποιούνται ως εξής:

```
ssh -p <θύρα του διακομιστή> -i <πλήρες μονοπάτι ιδιωτικού κλειδιού SSH>
↪  secdep@<ip του διακομιστή>
```

Εντολή 42: Σύνδεση στο διακομιστή της Amazon

```
sudo su
```

Εντολή 43: Μετάβαση σε χρήστη διαχειριστικών δικαιωμάτων

```
git clone https://github.com/CISOfy/lynis
```

Εντολή 44: Εγκατάσταση του Lynis

```
cd lynis
```

Εντολή 45: Μεταφορά στον φάκελο του Lynis

```
./lynis audit system > ~/lynis.log
```

Εντολή 46: Εκκίνηση αξιολόγησης (και αποθήκευσή της) μέσω του Lynis

6.3.3 Εγκατάσταση και προετοιμασία του LUNAR

Όπως και το Lynis, το LUNAR εγκαθίσταται και εκτελείται με τον ίδιο τρόπο. Αφότου έχει πραγματοποιηθεί η σύνδεση στο διακομιστή και έχουμε μεταβεί σε χρήστη με διαχειριστικά δικαιώματα, πρέπει να εκτελεστούν οι εξής εντολές:

```
git clone https://github.com/lateralblast/lunar
```

Εντολή 47: Εγκατάσταση του LUNAR

```
cd lunar
```

Εντολή 48: Μεταφορά στον φάκελο του LUNAR

```
./lunar.sh -a > ~/lunar.log
```

Εντολή 49: Εκκίνηση αξιολόγησης (και αποθήκευσή της) μέσω του LUNAR

6.4 Αποτελέσματα αξιολόγησης

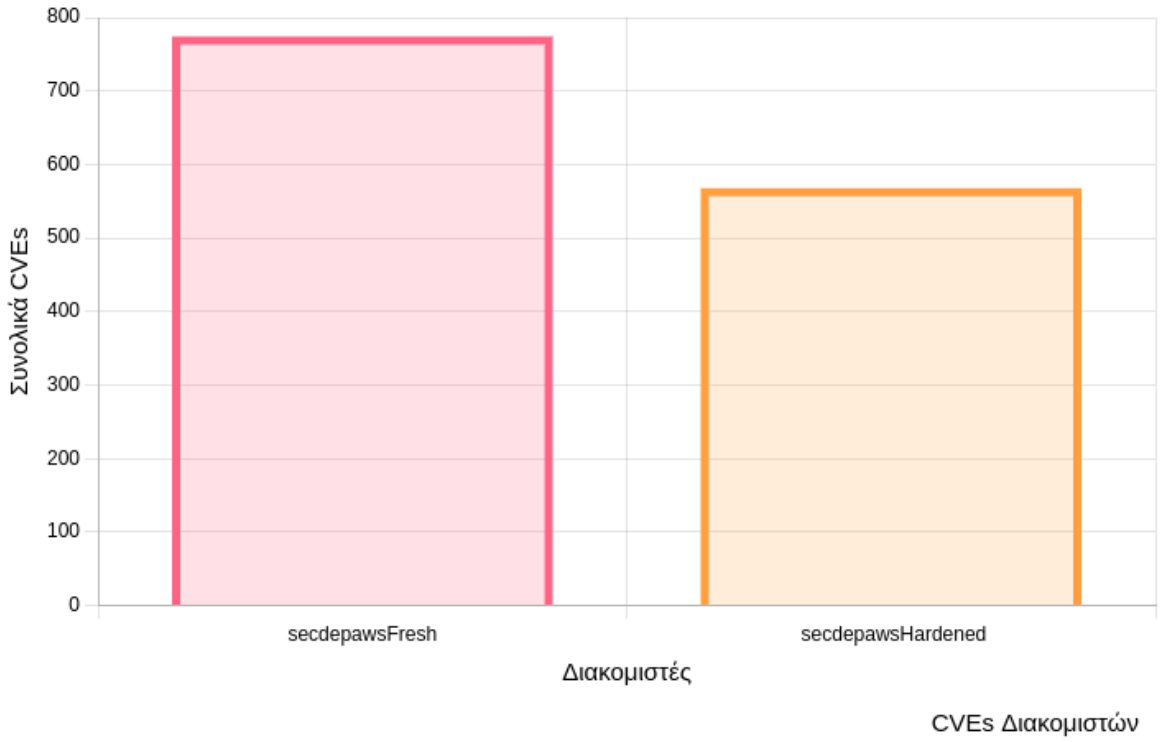
Μετά την ολοκλήρωση της αξιολόγησης των δύο διακομιστών, τα αποτελέσματα που παρήχθησαν από τα εργαλεία αξιολόγησης αναδεικνύουν τις διαφορές στην ασφάλεια των δύο διακομιστών. Στην παρούσα ενότητα θα αναλυθούν τα αποτελέσματα του κάθε εργαλείου αξιολόγησης ξεχωριστά και έπειτα θα βγει μια συμπερασματική αξιολόγησή τους.

6.4.1 Αποτελέσματα αξιολόγησης με το Vuls

Μετά την εκτέλεση του Vuls και στους δύο διακομιστές, φαίνεται να υπάρχει μια μείωση των CVEs από τον μη σκληρυμένο διακομιστή προς τον σκληρυμένο. Συγκεκριμένα βλέπουμε τα εξής αποτελέσματα:

- **secdepawsFresh** (VM_1):
  - **Αριθμός CVEs που εντοπίσθηκαν με το OVAL**: **0**
  - **Αριθμός CVEs που εντοπίσθηκαν με το gost**: **774**
  - **Αριθμός CVEs που εντοπίσθηκαν με το CPE**: **0**
  - **Αριθμός PoC**: **0**
  - **Αριθμός εκμεταλλεύσεων**: **1**
- **secdepawsHardened** (VM_2):
  - **Αριθμός CVEs που εντοπίσθηκαν με το OVAL**: **0**
  - **Αριθμός CVEs που εντοπίσθηκαν με το gost**: **568**
  - **Αριθμός CVEs που εντοπίσθηκαν με το CPE**: **0**
  - **Αριθμός PoC**: **0**
  - **Αριθμός εκμεταλλεύσεων**: **1**

Γραφικά, αυτό μεταφράζεται ως εξής:

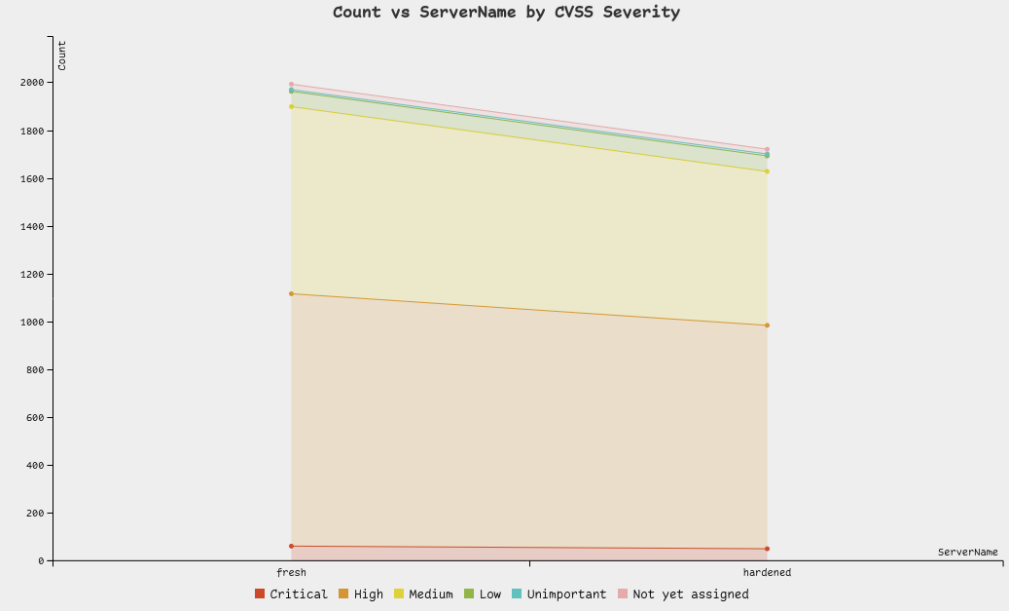


Σχήμα 6.4: CVEs διακομιστών

Από τα παραπάνω μέχρι στιγμής αποτελέσματα, παρατηρούμε ότι από τους δύο διακομιστές μας, ο διακομιστής secdepawsHardened, σχετικά με τα CVEs που εντοπίστηκαν, έχει ασφαλιστεί κατά 26.62%. Επιπροσθέτως, με βάση τα γραφήματα που παρήχθησαν από το VulsRepo, μπορούμε να διακρίνουμε μια άμεση πτώση των ευπαθειών με βάση την κατηγορία σοβαρότητάς τους στο Σχήμα 6.5. Στα Σχήματα 6.5 έως και 6.10, ο αριθμός των CVEs που παρουσιάζεται είναι μεγαλύτερος συγκριτικά με τα παραπάνω δεδομένα. Αυτό οφείλεται στο γεγονός ότι ενώ το Vuls συγκεντρώνει καθαρά τον αριθμό των CVEs που περιλαμβάνονται σε έναν διακομιστή, το VulsRepo υπολογίζει κάθε πακέτο που σχετίζεται με ένα CVE ως ξεχωριστή περίπτωση ευπάθειας. Επομένως, για κάθε CVE που εντοπίζεται, ο αριθμός των ευπαθειών αυξάνεται όχι μόνο κατά ένα, αλλά και κατά τον αριθμό των πακέτων που σχετίζονται με αυτό.

CVSS Severity	ServerName	fresh	hardened	Totals
Critical		61	50	**111**
High		1,055	934	**1,989**
Medium		782	643	**1,425**
Low		64	65	**129**
Unimportant		7	8	**15**
Not yet assigned		23	20	**43**
Totals		**1,992**	**1,720**	**3,712**

Σχήμα 6.5: Πίνακας συνολικών ευπαθειών ανά κατηγορία σοβαρότητας

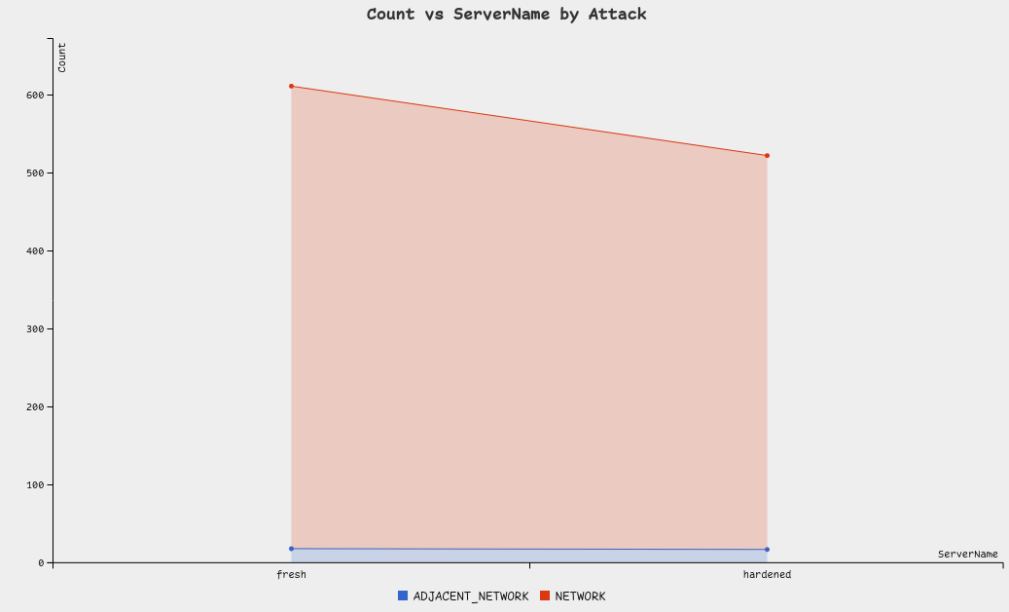


Σχήμα 6.6: Γράφημα συνολικών ευπαθειών ανά κατηγορία σοβαρότητας

Πιο συγκεκριμένα, παρατηρήθηκε 14.57% μείωση των ευπαθειών δικτύου και 13.58% των φυσικών ευπαθειών. Αυτό απεικονίζεται ορθότερα στα παρακάτω σχήματα:

Attack	ServerName	fresh	hardened	Totals
ADJACENT_NETWORK		18	17	35
NETWORK		593	505	1,098
Totals		611	522	1,133

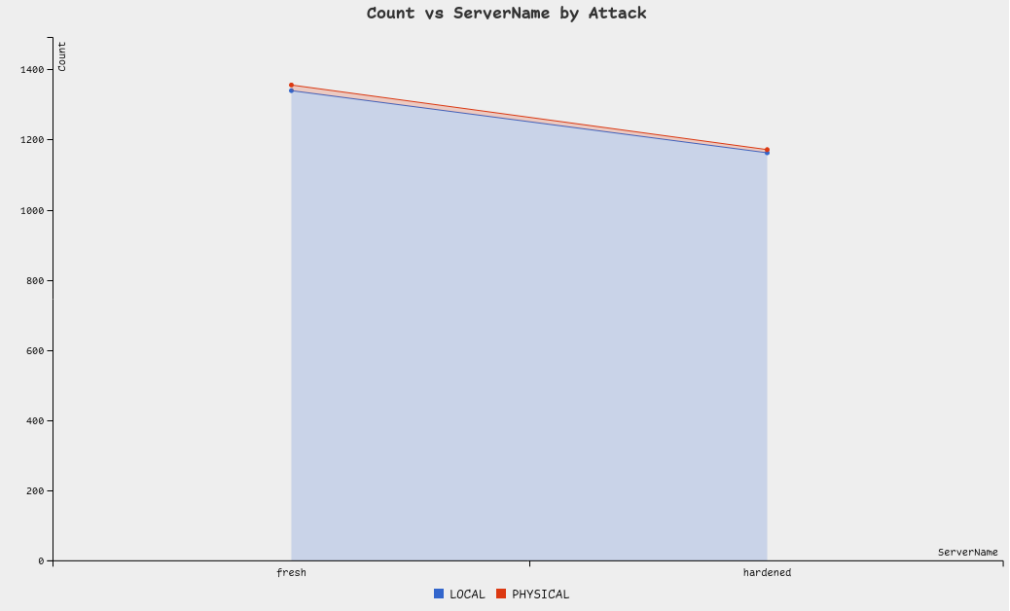
Σχήμα 6.7: Ευπάθειες δικτύου



Σχήμα 6.8: Μείωση ευπαθειών δικτύου

Attack	ServerName	fresh	hardened	Totals
LOCAL		1,339	1,162	2,501
PHYSICAL		16	9	25
Totals		1,355	1,171	2,526

Σχήμα 6.9: Φυσικές ευπάθειες



Σχήμα 6.10: Μείωση φυσικών ευπαθειών

Στην περίπτωση που υπάρχουν απορίες σχετικά με την συνολική αύξηση κατά δύο, των CVEs μικρού και ασήμαντου βαθμού κρισιμότητας αντίστοιχα, η αύξηση αυτή μπορεί να εξηγηθεί ή και να επιλυθεί. Ο λόγος που τα CVEs ασήμαντου βαθμού κατόπιν σκλήρυνσης μεταβάλλονται από 7 σε 8, είναι διότι στην σκληρυμένη εικονική μηχανή (VM_2) εγκαθίσταται το πακέτο "uidmap", το οποίο αποτελεί εξάρτηση του rootlesskit (για την λειτουργία του Rootless Docker). Το πακέτο αυτό, σχετίζεται με το πακέτο "shadow", το οποίο πάσχει από μια τρωτότητα ασήμαντου βαθμού που επηρεάζει μονάχα την χρήση της εντολής "su". Επομένως, εάν ο χρήστης χρησιμοποιεί όπως συνιστάται για λόγους ασφαλείας, το "sudo", τότε αυτός δεν επηρεάζεται καθόλου.

Η αύξηση κατά ένα των CVEs μικρού βαθμού κρισιμότητας, οφείλεται στο γεγονός ότι στην σκληρυμένη εικονική μηχανή (VM_2) εγκαθίσταται το πακέτο "vim". Ο λόγος εγκατάστασής του είναι για την διευκόλυνση του χρήστη στην ρύθμιση του συστήματος μέσω απομακρυσμένης σύνδεσης SSH. Το πακέτο αυτό, πάσχει από μια τρωτότητα υπερχείλισης buffer με βάση το σωρό (Heap-based Buffer Overflow), για την εκμετάλλευση της οποίας απαιτείται όχι μόνο απομακρυσμένη πρόσβαση στο σύστημα, αλλά και ύπαρξη διαχειριστικών δικαιωμάτων. Δύο προαπαιτούμενα για τα οποία μέσω της σκλήρυνσης του συστήματος, έχουμε προνοήσει να μην δύναται να αποκτηθούν.

Τέλος, οι κριτικές τρωτότητες που παρατηρούνται στον σκληρυμένο διακομιστή (VM_2), έχουν μειωθεί από 61 σε 50. Ο λόγος που η μείωση δεν είναι ακόμη μεγαλύτερη, είναι διότι αυτή σχετίζεται άμεσα με τα επηρεαζόμενα πακέτα. Πρόκειται για βασικά πακέτα του συστήματος, όπως τα "qemu-utils" και "linux-image-5.10.0-20-cloud-amd64", για τις ευπάθειες των οποίων δεν έχει δημοσιευθεί ακόμη επίσημη ενημέρωση ασφαλείας. Επιπρόσθετα, τα πακέτα αυτά δεν βρίσκονται υπό τον έλεγχο του χρήστη αλλά του παρόχου νέφους Amazon, ο οποίος είναι υπεύθυνος για την ενημέρωση τους. Συνεπώς, το μόνο που μπορεί να πράξει ο χρήστης, είναι να περιμένει την επίσημη ενημέρωση ασφαλείας, η οποία θα εφαρμοστεί αυτόματα, λόγω των ρυθμίσεων του SecDep.

6.4.2 Αποτελέσματα αξιολόγησης με το Lynis

Αφότου ολοκληρώθηκε η εκτέλεση του Lynis στους δύο διακομιστές, ήταν πλέον δυνατή η σύγκριση του δείκτη ασφαλείας τους με την εξής εντολή:

```
grep -i "index" lynis.log | sed 's/^ *//g' | cut -d' ' -f4
```

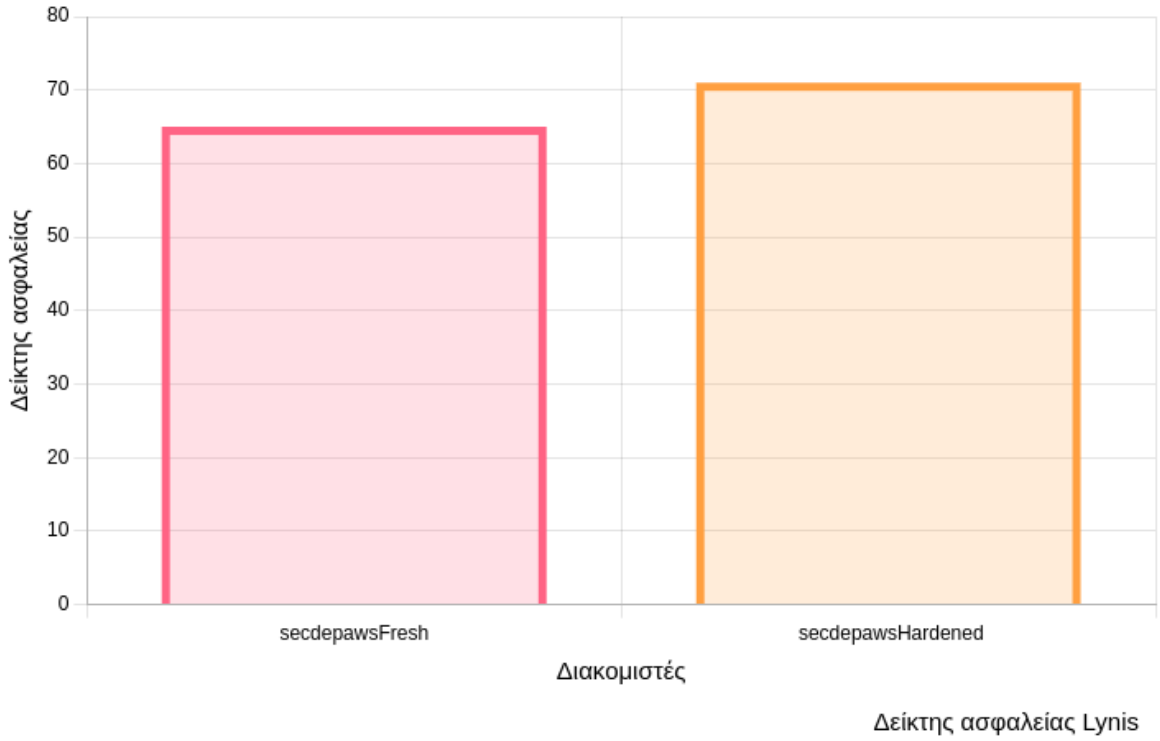
Εντολή 50: Εκτύπωση δείκτη ασφαλείας του Lynis

Το αποτέλεσμα της εντολής 50 διαφέρει προς το καλύτερο όταν το SecDep χρησιμοποιείται με την παράμετρο σκλήρυνσης, όπως φαίνεται και στον Πίνακα 6.1:

Πίνακας 6.1: Δείκτης ασφαλείας του Lynis

Διακομιστής	Δείκτης Ασφαλείας
Προ σκλήρυνσης	65
Μετά σκλήρυνσης	71

Η αύξηση ασφάλειας του διακομιστή ανέρχεται στο 9.23% και απεικονίζεται γραφικά στο Σχήμα 6.11.



Σχήμα 6.11: Αύξηση δείκτη ασφαλείας του Lynis

Παρ' όλο που υπήρξε αύξηση του δείκτη ασφαλείας, παρήχθησαν ορισμένες προτάσεις για την περαιτέρω βελτίωσή του. Μερικές από αυτές, όπως η ενημέρωση του Lynis εάν αυτό αποκτήθηκε μέσω των αποθετηρίων της διανομής, ή ο έλεγχος διεργασιών, οι οποίες εκτελούνται χρησιμοποιώντας διαγραμμένα αρχεία, δεν έχουν ιδιαίτερη σημασία. Ιδίως το δεύτερο, εφόσον κατόπιν σκλήρυνσης πραγματοποιείται και επανεκκίνηση του συστήματος. Άλλες προτάσεις, όπως η κατάτμηση του δίσκου, με σκοπό τον διαχωρισμό ορισμένων φακέλων από το

υπόλοιπο σύστημα, αφορούν κυρίως συστήματα ή εικονικές μηχανές που βρίσκονται εξ ολοκλήρου υπό τον έλεγχο του χρήστη, έναντι της περίπτωσης χρήσης υποδομών ενός παρόχου νέφους. Ωστόσο, υπήρχαν κάποιες χρήσιμες προτάσεις, από τις οποίες οι πιο σημαντικές που ενδεχομένως να άξιζε να υλοποιηθούν σε μεταγενέστερη έκδοση του SecDep, είναι οι εξής:

- Ρύθμιση ημερομηνίας λήξης λογαριασμών χρηστών
- Ρύθμιση ελάχιστης και μέγιστης ηλικίας κωδικών πρόσβασης
- Περαιτέρω σκλήρυνση του SSH
- Εγκατάσταση και ρύθμιση προγράμματος ελέγχου ακεραιότητας ευαίσθητων αρχείων
- Εγκατάσταση προγράμματος σάρωσης κακόβουλου λογισμικού

6.4.3 Αποτελέσματα αξιολόγησης με το LUNAR

Παρόμοια με το Lynis, εφόσον το LUNAR ολοκληρώσει την αξιολόγηση του διακομιστή παράγεται μια αναφορά. Στο τελικό μέρος της αναφοράς παρουσιάζονται, ο αριθμός των ελέγχων που πραγματοποιήθηκαν, των θετικών αποτελεσμάτων και των προειδοποιήσεων για σημεία που ενδέχεται να χρήζουν βελτίωσης.

Προκειμένου να εξάγουμε μια βαθμολογία, θα λάβουμε υπόψιν τον αριθμό των προειδοποιήσεων, τον οποίο θα αποκτήσουμε με την εκτέλεση της παρακάτω εντολής:

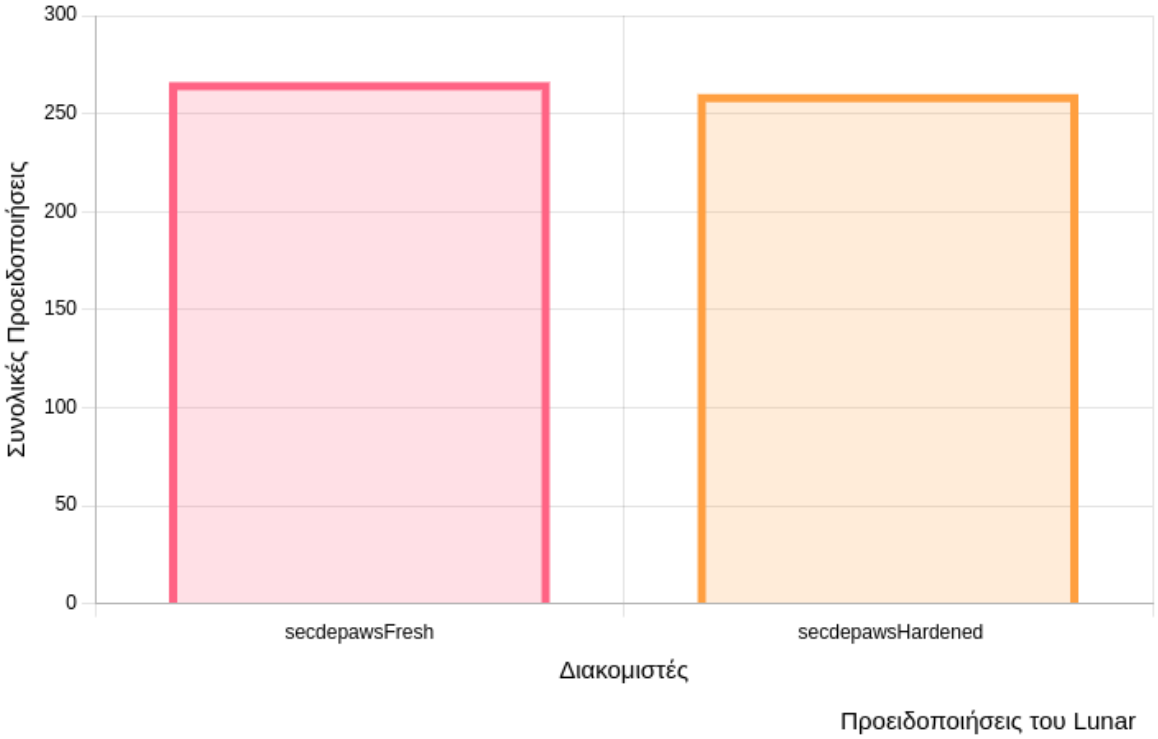
```
grep -i "warnings:" lunar.log | awk '{print $2}'
```

Εντολή 51: Εκτύπωση συνολικών προειδοποιήσεων του Lunar

Πίνακας 6.2: Συνολικές προειδοποιήσεις του Lunar

Διακομιστής	Αριθμός Προειδοποιήσεων
Προ σκλήρυνσης	266
Μετά σκλήρυνσης	260

Από τον Πίνακα 6.2 διαπιστώνεται πως έχουμε 2.26% αύξηση της ασφάλειας, το οποίο απεικονίζεται και ως εξής:



Σχήμα 6.12: Συνολικές προειδοποιήσεις του Lunar

Στην περίπτωση του LUNAR, η αύξηση της ασφάλειας φαίνεται να είναι μικρότερη. Αυτό οφείλεται κατά έναν βαθμό και στο γεγονός πως αυτό πραγματοποιεί υπερβολικά αυστηρούς ελέγχους, οι οποίοι ενδέχεται να μην είναι πάντα απαραίτητοι. Παρ' όλα αυτά, οι προτάσεις που παράγονται από το LUNAR για βελτίωση της ασφάλειας παραμένουν χρήσιμες και μπορούν να ληφθούν υπόψιν για μελλοντικές εκδόσεις του SecDep. Οι πιο αξιοσημείωτες από αυτές είναι:

- Περαιτέρω σκλήρυνση του SSH
- Εισαγωγή παραπάνω περιορισμών στον πυρήνα
- Αυστηρότερες άδειες πρόσβασης σε αρχεία και φακέλους

6.5 Συμπεράσματα αποτελεσμάτων αξιολόγησης

Από τα αποτελέσματα των τριών εργαλείων αξιολόγησης, προκύπτει ότι μετά την εφαρμογή της παραμέτρου σκλήρυνσης, ένας διακομιστής μπορεί να ασφαλιστεί σε ικανοποιητικό βαθμό. Η συγκεκριμένη παράμετρος σκληραίνει τον διακομιστή και με μεθόδους που δεν είναι μετρήσιμες, όπως η δημιουργία εκτελέσιμων αρχείων και η εφαρμογή περιοδικής τους εκτέλεσης με στόχους την ενημέρωση των πακέτων λογισμικού και το κλείσιμο θυρών που δεν χρησιμοποιούνται αντίστοιχα.

Επιπρόσθετα, η σκλήρυνση της μηχανής δοχείων Docker δεν απεικονίζεται σε κανένα από τα παραπάνω γραφήματα. Αυτό συμβαίνει διότι δεν υπάρχει εργαλείο που να μπορεί να αξιολογήσει την ασφάλεια του Docker όταν αυτό χρησιμοποιείται χωρίς διαχειριστικά δικαιώματα. Ωστόσο, η σκλήρυνσή του έχει πετύχει διότι η απαλλαγή από τον περιορισμό των διαχειριστικών δικαιωμάτων έχει ως αποτέλεσμα την μείωση της επιφάνειας επίθεσης σε περίπτωση παραβίασής του, ενώ η αντικατάσταση του runC με το runsc μειώνει την πιθανότητα παραβίασης. Για την ενίσχυση της ασφάλειας του Docker, εκτελούνται και βήματα όπως η ρύθμιση της παραμέτρου του δαίμονα "no-new-privileges" που αποτρέπει τα δοχεία από το να λαμβάνουν επιπρόσθετα δικαιώματα και η παράμετρος "selinux-enabled" εάν υποστηρίζεται από την διανομή. Τέλος, το SecDep ασφαλίζει το Docker επιπρόσθετα με έμμεσο τρόπο, εκτελώντας την υπηρεσία watchtower[11], η οποία είναι υπεύθυνη για την αυτόματη ενημέρωση των εικόνων δοχείων του συστήματος.

[11]Containrrr. Watchtower. URL: `https://containrrr.dev/watchtower/`.

Επομένως, το συμπέρασμα που προκύπτει είναι πως η χρήση του SecDep μπορεί να αυξήσει την ασφάλεια ενός συστήματος. Ωστόσο, ενώ τα εργαλεία αξιολόγησης παρουσιάζουν ένα ποσοστό αύξησης της ασφάλειας, η πραγματική αύξηση είναι μεγαλύτερη διότι πολλές από τις αλλαγές που πραγματοποιούνται δεν αντικατοπτρίζονται στα αποτελέσματα των αποτιμήσεων.

Κεφάλαιο 7. Συμπεράσματα & Προτάσεις για Μελλοντική Έρευνα

Σήμερα, η τεχνολογία της εικονικοποίησης χρησιμοποιείται ευρέως σε πολλούς τομείς της πληροφορικής. Πρακτικά, όλες οι μεγάλες εταιρείες παροχής υπηρεσιών, σε παγκόσμιο επίπεδο, χρησιμοποιούν εικονικές μηχανές για την παροχή των υπηρεσιών τους. Επιπλέον, η τεχνολογία των δοχείων, έχει αποτελέσει μια από τις μεγαλύτερες επαναστατικές εξελίξεις της τελευταίας δεκαετίας, δίνοντας την δυνατότητα στους προγραμματιστές να αναπτύσσουν και να εκτελούν εφαρμογές σε οποιοδήποτε περιβάλλον, χωρίς να χρειάζεται να ανησυχούν για τα χαρακτηριστικά του συστήματος στο οποίο θα εκτελούνται. Ωστόσο, η χρήση των παραπάνω τεχνολογιών, όπως και οποιασδήποτε άλλης τεχνολογίας, προϋποθέτει την υπακοή σε κάποιους κανόνες και πρωτόκολλα, προκειμένου να επιτευχθεί η αποδοτικότερη και ασφαλέστερη λειτουργία τους.

Στην παρούσα διπλωματική εργασία, παρουσιάστηκαν διάφορα ζητήματα ασφαλείας που αφορούν περιβάλλοντα εικονικών μηχανών και δοχείων. Με γνώμονα τα ζητήματα αυτά, πραγματοποιήθηκε η ανάπτυξη ενός εργαλείου, το οποίο ικανοποιώντας όλες τις απαιτήσεις που ορίσθηκαν στην Ενότητα 4.2, μπορεί με αυτοματοποιημένο τρόπο να δημιουργεί εικονικές μηχανές στις οποίες θα εγκατασταθούν δοχεία Docker, αντιμετωπίζοντας ταυτόχρονα τα προαναφερθέντα ζητήματα ασφαλείας. Το εργαλείο αυτό, ονομάζεται SecDep και αποτελείται από δύο εκτελέσιμα αρχεία, τα οποία μπορούν να εκτελεστούν και ως αυτοτελείς εφαρμογές. Το πρώτο, υπεύθυνο για την δημιουργία και διαχείριση των εικονικών μηχανών και το δεύτερο, για την σκλήρυνση του συστήματος και την εγκατάσταση/σκλήρυνση του δαίμονα Docker.

7.1 Συμπεράσματα

Με βάση τα αποτελέσματα των πειραμάτων που πραγματοποιήθηκαν στο Κεφάλαιο 6, μπορούμε να συμπεράνουμε ότι το εργαλείο SecDep, είναι ικανό να δημιουργήσει εικονικές μηχανές που έχουν σκληρύνει σε ικανοποιητικό βαθμό, χωρίς περαιτέρω παρέμβαση από τον χρήστη. Μπορεί λοιπόν να αποτελέσει ένα χρήσιμο εργαλείο για την αυτοματοποιημένη δημιουργία και διαμόρφωση ενός ασφαλούς κατανεμημένου περιβάλλοντος, έτοιμο να υποδεχτεί δοχεία Docker για την στέγαση εφαρμογών μικρο-υπηρεσιών. Μιας αρχιτεκτονικής που έχει αρχίσει να αποκτά ιδιαίτερη δημοτικότητα τα τελευταία χρόνια.

Επιπλέον, το SecDep μπορεί να αποτελέσει ένα χρήσιμο εργαλείο για την εκπαίδευση σε θέματα ασφάλειας, καθώς ο χρήστης έχει την δυνατότητα να μελετήσει τον πηγαίο κώδικά του και να προσθέσει τις δικές του λειτουργίες, προκειμένου να επεκτείνει την λειτουργικότητά του. Το εργαλείο διανέμεται με την άδεια χρήσης GPLv3, η οποία παρέχει το δικαίωμα στον χρήστη να το χρησιμοποιήσει, να το μελετήσει, να το τροποποιήσει και να το διανείμει ελεύθερα.

7.2 Προτάσεις για Μελλοντική Έρευνα

Παρ' όλο που το SecDep, είναι ένα πλήρες και λειτουργικό εργαλείο, υπάρχουν πολλές ευκαιρίες για την επέκτασή του. Αρχικά, θα μπορούσε να επεκταθεί η λειτουργικότητά του, ώστε να υποστηρίζει και άλλους παρόχους νέφους. Αυτό βέβαια εξαρτάται άμεσα και από το εύρος της υποστήριξης συγκεκριμένων συναρτήσεων από την βιβλιοθήκη libcloud. Επιπλέον, θα μπορούσε να επεκταθεί με σκοπό την υποστήριξη παραπάνω μεθόδων της βιβλιοθήκης, δίνοντας στον χρήστη δυνατότητες όπως η ρύθμιση DNS, εξωτερικής αποθήκευσης δεδομένων και εξισορρόπησης φόρτου εργασιών.

Τα παραπάνω αφορούν την επέκταση ενός από τα δύο εκτελέσιμα αρχεία του SecDep (secdep.py), αυτού για την διαχείριση και δημιουργία εικονικών μηχανών. Το δεύτερο (harden), το οποίο επικεντρώνεται στην σκλήρυνση του συστήματος και του Docker, θα μπορούσε και αυτό να επεκταθεί, προσθέτοντάς του νέες λειτουργίες. Μια από αυτές θα μπορούσε να είναι η επιλογή του χρήστη για εφαρμογή συγκεκριμένων ρυθμίσεων ασφαλείας. Επίσης, θα μπορούσε να χωριστεί σε δύο ανεξάρτητα εκτελέσιμα αρχεία, το ένα υπεύθυνο για την σκλήρυνση του συστήματος και το άλλο για την εγκατάσταση και σκλήρυνση του δαίμονα του Docker. Με αυτόν τον τρόπο, ο χρήστης θα μπορούσε να επιλέξει να εγκαταστήσει μια σκληρυμένη έκδοση του Docker σε ένα σύστημα που έχει ήδη σκληρύνει, χωρίς να χρειάζεται να χρησιμοποιήσει το πρώτο εκτελέσιμο αρχείο. Τέλος, θα μπορούσε να βελτιωθεί η διαδικασία σκλήρυνσης ακολουθώντας τις συμβουλές που αναφέρθηκαν στις Ενότητες 6.4.2 και 6.4.3. Τα αποτελέσματα είναι ήδη ικανοποιητικά, ωστόσο πάντα υπάρχουν περιθώρια βελτίωσης.

Βιβλιογραφία

[1] AWS. What are microservices? (Αναφορά: σελίδες iii, v)
url: https://aws.amazon.com/microservices/
(Επίσκεψη 19/11/2023).

[2] C. Slingerland. Horizontal vs. vertical scaling: how do they compare? 2023 (Αναφορά: σελίδα 1)
url: https://www.cloudzero.com/blog/horizontal-vs-vertical-scaling/
(Επίσκεψη 08/05/2023).

[3] Oracle. Mysql (Αναφορά: σελίδα 2)
url: https://www.mysql.com/.

[4] NGINX. Nginx (Αναφορά: σελίδα 2)
url: https://nginx.org/en/.

[5] M. Balduzzi, J. Zaddach, D. Balzarotti, E. Kirda και S. Loureiro. A security analysis of amazon's elastic compute cloud service. Στο *Proceedings of the 27th Annual ACM Symposium on Applied Computing*, SAC '12, 1427–1434, Trento, Italy. Association for Computing Machinery, 2012. isbn: 9781450308571. doi: 10.1145/2245276.2232005 (Αναφορά: σελίδα 3)
url: https://doi.org/10.1145/2245276.2232005
(Επίσκεψη 04/12/2023).

[6] A. J. Monteclaro. What is hyperjacking? how to prevent hyperjacking on a vm. 2023 (Αναφορά: σελίδα 3)
url: https://www.serverwatch.com/virtualization/hyperjacking/
(Επίσκεψη 20/08/2023).

[7] S. Vaughan-Nichols. Containers or virtual machines: which is more secure? the answer will surprise you. 2018 (Αναφορά: σελίδα 5)
url: https://www.zdnet.com/article/which-is-more-secure-containers-or-virtual-machines-the-answer-will-surprise-you/
(Επίσκεψη 25/08/2023).

[8] J. Bottomley. Containers and cloud security. 2018 (Αναφορά: σελίδα 5)
url: https://blog.hansenpartnership.com/containers-and-cloud-security/
(Επίσκεψη 27/07/2023).

[9] C. Tozzi. The inherent security benefits of docker containers. 2017 (Αναφορά: σελίδα 6)
url: https://cloudnativenow.com/features/security-benefits-docker-containers/
(Επίσκεψη 10/12/2023).

[10] O. Abargil. 7 ways to escape a container. 2023 (Αναφορά: σελίδα 7)
url: https://www.panoptica.app/research/7-ways-to-escape-a-container
(Επίσκεψη 23/10/2023).

[11] A. Mouat. 5 security concerns when using docker. 2016 (Αναφορά: σελίδα 7)
url: https://www.oreilly.com/content/five-security-concerns-when-using-docker/
(Επίσκεψη 06/12/2023).

[12] A. Suda. Rootlesskit. 2020 (Αναφορά: σελίδες 8, 61)
url: https://github.com/rootless-containers/rootlesskit.

[13] Google. The container security platform (Αναφορά: σελίδα 8)
url: https://gvisor.dev/.

[14] P. Mell και T. Grance. The nist definition of cloud computing. en, Σεπτ. 2011. doi: https://doi.org/10.6028/NIST.SP.800-145 (Αναφορά: σελίδες 11, 13, 14)
(Επίσκεψη 12/11/2023).

[15] vendr. Our guide to every saas pricing model. 2022 (Αναφορά: σελίδα 13)
url: https://www.vendr.com/blog/saas-pricing-model#value-based-saas-pricing-models
(Επίσκεψη 03/11/2023).

[16] S. Logic. Paas (platform-as-a-service) - definition & overview (Αναφορά: σελίδα 13)
url: https://www.sumologic.com/glossary/paas/
(Επίσκεψη 08/04/2023).

[17] Intel. An overview of cloud deployment models (Αναφορά: σελίδα 14)
url: https://www.intel.com/content/www/us/en/cloud-computing/deployment-models.html
(Επίσκεψη 08/07/2023).

[18] Abacus. Three types of server virtualization explained (Αναφορά: σελίδα 16)
url: https://goabacus.com/three-types-of-server-virtualization-explained/
(Επίσκεψη 29/11/2023).

[19] GeeksforGeeks. Hardware based virtualization (Αναφορά: σελίδα 17)
url: https://www.geeksforgeeks.org/hardware-based-virtualization/
(Επίσκεψη 29/03/2023).

[20] R. Hat. What is virtualization? 2018 (Αναφορά: σελίδες 18–22, 26, 31–33)
url: https://www.redhat.com/en/topics/virtualization/what-is-virtualization
(Επίσκεψη 21/05/2023).

[21] altexsoft. Data virtualization: process, components, benefits, and available tools. 2021 (Αναφορά: σελίδα 18)
url: https://www.altexsoft.com/blog/data-virtualization/
(Επίσκεψη 07/10/2023).

[22] VMware. What is desktop virtualization? (Αναφορά: σελίδα 19)
url: https://www.vmware.com/topics/glossary/content/desktop-virtualization.html
(Επίσκεψη 11/11/2023).

[23] R. Hat. What is nfv? 2019 (Αναφορά: σελίδα 22)
url: https://www.redhat.com/en/topics/virtualization/what-is-nfv
(Επίσκεψη 18/08/2023).

[24] VMware. Memory virtualization. 2019 (Αναφορά: σελίδα 23)
url: https://docs.vmware.com/en/VMware-vSphere/7.0/com.vmware.vsphere.resmgmt.doc/GUID-6E85F6DE-7365-4C28-B902-725D3C76C2E6.html
(Επίσκεψη 24/03/2023).

[25] H. I. Smart. Memory virtualization in cloud computing (Αναφορά: σελίδα 23)
url: https://www.hostitsmart.com/blog/memory-virtualization-in-cloud-computing/
(Επίσκεψη 02/08/2023).

[26] B. Hill. Intro to virtualization: hardware, software, memory, storage, data and network virtualization defined. 2012 (Αναφορά: σελίδα 23)
url: https://petri.com/intro-to-virtualization/
(Επίσκεψη 13/01/2023).

[27] R. Agrawal. Processor and memory virtualization. 2023 (Αναφορά: σελίδα 23)
url: https://www.codingninjas.com/studio/library/processor-and-memory-virtualization
(Επίσκεψη 10/12/2023).

[28] LINGESH. Virtualization & hypervisor – basic interview questions. 2019 (Αναφορά: σελίδα 24)
url: https://www.unixarena.com/2019/08/virtualization-hypervisor-basic-interview-questions.html/
(Επίσκεψη 01/09/2023).

[29] Crystal. What is storage virtualization | introduction and implementation. 2022 (Αναφορά: σελίδα 24)
url: https://www.ubackup.com/enterprise-backup/storage-virtualization-jkzbj.html
(Επίσκεψη 13/11/2023).

[30] D. Muvaa. Storage virtualization in cloud computing – how it works (use cases) (Αναφορά: σελίδα 25)
url: https://cloudinfrastructureservices.co.uk/storage-virtualization-in-cloud-computing-how-it-works-use-cases/
(Επίσκεψη 17/08/2023).

[31] T. Point. Virtualization 2.0 - overview (Αναφορά: σελίδα 25)
url: https://www.tutorialspoint.com/virtualization2.0/virtualization2.0_overview.htm
(Επίσκεψη 28/02/2023).

[32] GeeksforGeeks. Virtualisation with docker containers. 2023 (Αναφορά: σελίδα 25)
url: https://www.geeksforgeeks.org/virtualisation-with-docker-containers/
(Επίσκεψη 02/04/2023).

[33] R. Hat. What is virtualization management? 2018 (Αναφορά: σελίδα 26)
url: https://www.redhat.com/en/topics/virtualization/what-is-virtualization-management
(Επίσκεψη 11/01/2023).

[34] IBM. What are hypervisors? (Αναφορά: σελίδες 27, 29)
url: https://www.ibm.com/topics/hypervisors
(Επίσκεψη 31/03/2023).

[35] S. Simic. What is a hypervisor? types of hypervisors 1 & 2. 2022 (Αναφορά: σελίδα 28)
url: https://phoenixnap.com/kb/what-is-hypervisor-type-1-2
(Επίσκεψη 26/06/2023).

[36] A. W. Services. What's the difference between type 1 and type 2 hypervisors? (Αναφορά: σελίδα 28)
url: https://aws.amazon.com/compare/the-difference-between-type-1-and-type-2-hypervisors/
(Επίσκεψη 29/01/2023).

[37] R. Hat. What is kvm? 2022 (Αναφορά: σελίδα 29)
url: https://www.redhat.com/en/topics/virtualization/what-is-KVM
(Επίσκεψη 27/01/2023).

[38] N. Team. What is a failover? clustering and replication use cases. 2023 (Αναφορά: σελίδα 30)
url: https://www.nakivo.com/blog/vm-failover-guide/
(Επίσκεψη 23/10/2023).

[39] R. Hat. Understanding virtualization. 2018 (Αναφορά: σελίδα 31)
url: https://www.redhat.com/en/topics/virtualization
(Επίσκεψη 11/08/2023).

[40] SUSE. Paravirtualization (Αναφορά: σελίδα 33)
url: https://www.suse.com/suse-defines/definition/paravirtualization/
(Επίσκεψη 09/06/2023).

[41] R. Fernandez. What is paravirtualization? definition and uses. 2023 (Αναφορά: σελίδα 34)
url: https://www.serverwatch.com/virtualization/what-is-paravirtualization/
(Επίσκεψη 13/11/2023).

[42] M. Karimyar. What is paravirtualization in cloud computing? 2023 (Αναφορά: σελίδα 34)
url: https://blog.servermania.com/what-is-paravirtualization
(Επίσκεψη 09/12/2023).

[43] I. for Professionals. Paravirtualization vs. full virtualization: pros and cons. 2022 (Αναφορά: σελίδα 34)
url: https://www.insightsforprofessionals.com/it/data-center/paravirtualization-alternative-full-virtualization
(Επίσκεψη 23/08/2023).

[44] BlackBerry. Paravirtualization (Αναφορά: σελίδα 34)
url: https://blackberry.qnx.com/en/ultimate-guides/automotive-hypervisor/paravirtualization
(Επίσκεψη 30/11/2023).

[45] GeeksforGeeks. Difference between full virtualization and paravirtualization. 2022 (Αναφορά: σελίδες 34–36)
url: https://www.geeksforgeeks.org/difference-between-full-virtualization-and-paravirtualization/
(Επίσκεψη 25/04/2023).

[46] A. Mallett. Why your virtual servers may be more secure than their physical counterparts. 2019 (Αναφορά: σελίδα 36)
url: https://ine.com/blog/why-your-virtual-servers-may-be-more-secure-than-their-physical-counterparts
(Επίσκεψη 11/03/2023).

[47] VMware. Understanding full virtualization, paravirtualization and hardware assisted virtualization. 2007 (Αναφορά: σελίδα 36)
url: https://www.vmware.com/content/dam/digitalmarketing/vmware/en/pdf/techpaper/VMware_paravirtualization.pdf
(Επίσκεψη 23/03/2023).

[48] IBM. What is virtualization? (Αναφορά: σελίδες 36, 38)
url: https://www.ibm.com/topics/virtualization
(Επίσκεψη 20/02/2023).

[49] TechClaw. Virtualization in cloud computing: bridging the gap between resources and efficiency. 2023 (Αναφορά: σελίδα 37)
url: https://medium.com/@techclaw/virtualization-in-cloud-computing-bridging-the-gap-between-resources-and-efficiency-3c5a9c65981e
(Επίσκεψη 30/11/2023).

[50] R. Hat. Ansible (Αναφορά: σελίδα 37)
url: https://www.ansible.com/.

[51] HashiCorp. Terraform (Αναφορά: σελίδα 37)
url: https://www.terraform.io/.

[52] N. Carklin. Understanding the correct use of vm snapshots. 2021 (Αναφορά: σελίδα 38)
url: https://www.parallels.com/blogs/ras/vm-snapshot/
(Επίσκεψη 25/03/2023).

[53] S. J. Bigelow. Virtual security tactics for type 1 and type 2 hypervisors. 2013 (Αναφορά: σελίδα 38)
url: https://www.techtarget.com/searchitoperations/answer/Virtual-security-tactics-for-Type-1-and-Type-2-hypervisors
(Επίσκεψη 31/03/2023).

[54] C. Wen-Zhi, Z. Hong-Wei και H. Wei. Sevmm: vmm-based security control model. Στο *2008 International Conference on Cyberworlds*, 820–823. IEEE, 2008 (Αναφορά: σελίδα 38)
(Επίσκεψη 14/02/2023).

[55] M. Arif και H. Shakeel. Virtualization security: analysis and open challenges. *International Journal of Hybrid Information Technology*, 8(2):237–246, 2015 (Αναφορά: σελίδες 38, 40)
(Επίσκεψη 30/09/2023).

[56] B. Sane, I. Niang και D. Fall. A review of virtualization, hypervisor and vm allocation security: threats, vulnerabilities, and countermeasures. Στο 1317–1322, Δεκ. 2018. doi: 10.1109/CSCI46756.2018.00255 (Αναφορά: σελίδες 41, 44)
(Επίσκεψη 13/06/2023).

[57] Z. Aalam, V. Kumar και S. Gour. A review paper on hypervisor and virtual machine security. *Journal of Physics: Conference Series*, 1950(1):012027, Αύγ. 2021. doi: 10.1088/1742-6596/1950/1/012027 (Αναφορά: σελίδα 41)
url: https://dx.doi.org/10.1088/1742-6596/1950/1/012027
(Επίσκεψη 20/04/2023).

[58] A. Litchfield και A. Shahzad. Virtualization technology: cross-vm cache side channel attacks make it vulnerable. *arXiv preprint arXiv:1606.01356*, 2016 (Αναφορά: σελίδα 41)
(Επίσκεψη 04/02/2024).

[59] Y. Zhang, A. Juels, M. K. Reiter και T. Ristenpart. Cross-vm side channels and their use to extract private keys. Στο *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, CCS '12, 305–316, Raleigh, North Carolina, USA. Association for Computing Machinery, 2012. isbn: 9781450316514. doi: 10.1145/2382196.2382230 (Αναφορά: σελίδα 41)
url: https://doi.org/10.1145/2382196.2382230
(Επίσκεψη 04/02/2024).

[60] Y. Xia, Y. Liu, H. Chen και B. Zang. Defending against vm rollback attack. Στο 1–5, Ιούν. 2012. isbn: 978-1-4673-2264-5. doi: 10.1109/DSNW.2012.6264690 (Αναφορά: σελίδα 42)
(Επίσκεψη 04/02/2024).

[61] M. Rouse. Virtual machine hyper jumping. 2015 (Αναφορά: σελίδα 42)
url: https://www.techopedia.com/definition/30921/virtual-machine-hyper-jumping-vm-jumping
(Επίσκεψη 04/02/2024).

[62] H. Abusaimeh. Virtual machine escape in cloud computing services. *International Journal of Advanced Computer Science and Applications*, 11(7), 2020 (Αναφορά: σελίδα 43)
(Επίσκεψη 04/02/2024).

[63] ENISA. Security aspects of virtualization. 2017 (Αναφορά: σελίδες 45, 47)
url: https://www.enisa.europa.eu/publications/security-aspects-of-virtualization
(Επίσκεψη 14/10/2023).

[64] W. Chai. What is the cia triad (confidentiality, integrity and availability)? 2023 (Αναφορά: σελίδα 46)
url: https://www.techtarget.com/whatis/definition/Confidentiality-integrity-and-availability-CIA
(Επίσκεψη 11/11/2023).

[65] GeeksforGeeks. Hypervisor security in cloud computing. 2023 (Αναφορά: σελίδα 47)
url: https://www.geeksforgeeks.org/hypervisor-security-in-cloud-computing/
(Επίσκεψη 07/07/2023).

[66] Sysdig. What are container runtimes? (Αναφορά: σελίδα 50)
url: https://sysdig.com/learn-cloud-native/container-security/what-are-container-runtimes/
(Επίσκεψη 25/01/2023).

[67] N. Velayudhan. What are container runtimes? 2021 (Αναφορά: σελίδα 50)
url: https://opensource.com/article/21/9/container-runtimes
(Επίσκεψη 11/02/2024).

[68] D. Teimouri. Operating-system-level virtualization. 2017 (Αναφορά: σελίδα 50)
url: https://www.teimouri.net/operating-system-level-virtualization/
(Επίσκεψη 06/01/2023).

[69] V. Beal. Operating system-level virtualization. 2021 (Αναφορά: σελίδα 50)
url: https://www.webopedia.com/definitions/operating-system-level-virtualization/
(Επίσκεψη 24/09/2023).

[70] G. D. Maayan. Container images: technical refresher and security best practices. 2023 (Αναφορά: σελίδα 51)
url: https://www.codemotion.com/magazine/cybersecurity/container-images-technical-refresher-and-security-best-practices/
(Επίσκεψη 28/10/2023).

[71] D. Santra. The chroot command in linux – beginners introduction. 2021 (Αναφορά: σελίδα 51)
url: https://www.linuxfordevices.com/tutorials/linux/chroot-command-in-linux
(Επίσκεψη 07/04/2023).

[72] E. Mell. The evolution of containers: docker, kubernetes and the future. 2023 (Αναφορά: σελίδα 51)
url: https://www.techtarget.com/searchitoperations/feature/Dive-into-the-decades-long-history-of-container-technology
(Επίσκεψη 31/10/2023).

[73] 2. B. March 27. Is chroot a security feature? 2023 (Αναφορά: σελίδα 52)
url: https://www.redhat.com/en/blog/chroot-security-feature
(Επίσκεψη 07/08/2023).

[74] L. Containers. What's lxc? (Αναφορά: σελίδα 52)
url: https://linuxcontainers.org/lxc/introduction/.

[75] E. Kahuha. Lxc vs docker: which container platform is right for you? 2023 (Αναφορά: σελίδα 52)
url: https://earthly.dev/blog/lxc-vs-docker/
(Επίσκεψη 05/08/2023).

[76] Docker. Build and ship any application anywhere (Αναφορά: σελίδα 52)
url: https://hub.docker.com/.

[77] R. Hat. Quay builds, analyzes, distributes your container images (Αναφορά: σελίδα 52)
url: https://quay.io/.

[78] T. L. Foundation. Open container initiative (Αναφορά: σελίδα 52)
url: https://opencontainers.org/.

[79] D. Patil. The untold story: containers before docker's rise - the lxc revolution. 2023 (Αναφορά: σελίδα 53)
url: https://www.linkedin.com/pulse/untold-story-containers-before-dockers-rise-lxc-revolution-patil
(Επίσκεψη 20/09/2023).

[80] IBM. Containers in the enterprise. 2020 (Αναφορά: σελίδα 53)
url: https://www.ibm.com/downloads/cas/VG8KRPRM
(Επίσκεψη 08/12/2023).

[81] ATLASSIAN. Waterfall methodology: a comprehensive guide (Αναφορά: σελίδα 53)
url: https://www.atlassian.com/agile/project-management/waterfall-methodology
(Επίσκεψη 26/03/2023).

[82] L. Davis. What is agile project management? the ultimate guide. 2022 (Αναφορά: σελίδα 54)
url: https://www.forbes.com/advisor/business/what-is-agile-project-management/
(Επίσκεψη 20/06/2023).

[83] Synopsys. Devops (Αναφορά: σελίδα 54)
url: https://www.synopsys.com/glossary/what-is-devops.html
(Επίσκεψη 06/02/2023).

[84] GitLab. What is ci/cd? (Αναφορά: σελίδα 54)
url: https://about.gitlab.com/topics/ci-cd/
(Επίσκεψη 19/03/2023).

[85] K. Center. Containerized applications overview (Αναφορά: σελίδα 54)
url: https://www.datadoghq.com/knowledge-center/containerized-applications/
(Επίσκεψη 29/10/2023).

[86] S. Logic. What is caas? (Αναφορά: σελίδα 55)
url: https://www.sumologic.com/glossary/caas/
(Επίσκεψη 24/05/2023).

[87] aquasec. Container as a service: the basics and top 4 providers. 2023 (Αναφορά: σελίδα 55)
url: https://www.aquasec.com/cloud-native-academy/container-platforms/container-as-a-service/
(Επίσκεψη 30/04/2023).

[88] K. Traiaia. The guide to containers-as-a-service (caas). 2023 (Αναφορά: σελίδα 55)
url: https://www.kerno.io/blog/containers-as-a-service-caas
(Επίσκεψη 24/07/2023).

[89] J. Ellingwood. What is kubernetes? 2018 (Αναφορά: σελίδα 56)
url: https://www.digitalocean.com/community/tutorials/an-introduction-to-kubernetes
(Επίσκεψη 21/01/2023).

[90] Simplilearn. What is docker swarm: modes, example and working. 2023 (Αναφορά: σελίδα 56)
url: https://www.simplilearn.com/tutorials/docker-tutorial/docker-swarm
(Επίσκεψη 14/10/2023).

[91] S. Javaid. What is multi-cloud? features, architecture, pros & cons. 2023 (Αναφορά: σελίδα 56)
url: https://www.cloudways.com/blog/what-is-multi-cloud/
(Επίσκεψη 05/03/2023).

[92] IBM. Containers vs. virtual machines (vms): what's the difference? 2021 (Αναφορά: σελίδες 57, 58)
url: https://www.ibm.com/blog/containers-vs-vms/
(Επίσκεψη 17/11/2023).

[93] W. Ondara. A complete overview of docker architecture. 2022 (Αναφορά: σελίδα 57)
url: https://www.cherryservers.com/blog/a-complete-overview-of-docker-architecture
(Επίσκεψη 08/02/2024).

[94] R. Hat. Containers vs vms. 2020 (Αναφορά: σελίδα 58)
url: https://www.redhat.com/en/topics/containers/containers-vs-vms
(Επίσκεψη 02/06/2023).

[95] Docker. Docker compose (Αναφορά: σελίδα 58)
url: https://github.com/docker/compose.

[96] L. Web. Virtualization vs. containerization — comparing differences. 2023 (Αναφορά: σελίδα 58)
url: https://www.liquidweb.com/kb/virtualization-vs-containerization/
(Επίσκεψη 10/12/2023).

[97] IBM. What is containerization? (Αναφορά: σελίδες 58, 61, 63)
url: https://www.ibm.com/topics/containerization
(Επίσκεψη 29/11/2023).

[98] Docker. Access authorization plugin (Αναφορά: σελίδα 60)
url: https://docs.docker.com/engine/extend/plugins_authorization/#access-authorization-plugin
(Επίσκεψη 25/05/2023).

[99] R. Hat. What is podman? 2022 (Αναφορά: σελίδα 61)
url: https://www.redhat.com/en/topics/containers/what-is-podman
(Επίσκεψη 01/05/2023).

[100] rkt. Rkt (Αναφορά: σελίδα 61)
url: https://github.com/rkt/rkt.

[101] C. Slingerland. What are the best docker alternatives in 2022? 2022 (Αναφορά: σελίδα 61)
url: https://www.cloudzero.com/blog/docker-alternatives/
(Επίσκεψη 07/06/2023).

[102] T. Donohue. The differences between docker, containerd, cri-o and runc. 2023 (Αναφορά: σελίδα 62)
url: https://www.tutorialworks.com/difference-docker-containerd-runc-crio-oci/
(Επίσκεψη 13/05/2023).

[103] Snyk. Snyk (Αναφορά: σελίδα 62)
url: https://snyk.io/.

[104] A. Security. Trivy (Αναφορά: σελίδα 62)
url: https://aquasecurity.github.io/trivy/v0.49/.

[105] E. Reshetova, J. Karhunen, T. Nyman και N. Asokan. Security of os-level virtualization technologies. Στο *Nordic Conference on Secure IT Systems*, 77–93. Springer, 2014 (Αναφορά: σελίδα 63)
(Επίσκεψη 01/08/2023).

[106] T. Bui. Analysis of docker security. 2015. doi: 10.48550/ARXIV.1501.02967 (Αναφορά: σελίδα 63)
url: https://arxiv.org/abs/1501.02967
(Επίσκεψη 02/06/2023).

[107] T. L. kernel user's και administrator's guide. Device whitelist controller (Αναφορά: σελίδα 63)
url: https://www.kernel.org/doc/html/latest/admin-guide/cgroup-v1/devices.html
(Επίσκεψη 26/03/2023).

[108] R. Hat. What is selinux? 2019 (Αναφορά: σελίδα 64)
url: https://www.redhat.com/en/topics/linux/what-is-selinux
(Επίσκεψη 08/04/2023).

[109] AppArmor. Apparmor (Αναφορά: σελίδα 64)
url: https://apparmor.net/.

[110] V. Rothberg. Improving linux container security with seccomp. 2020 (Αναφορά: σελίδα 64)
url: https://www.redhat.com/sysadmin/container-security-seccomp
(Επίσκεψη 11/07/2023).

[111] R. Yasrab. Mitigating docker security issues. *arXiv preprint arXiv:1804.05039*, 2023 (Αναφορά: σελίδες 65, 66)
(Επίσκεψη 29/07/2023).

[112] R. Lakshmanan. New linux kernel cgroups vulnerability could let attackers escape container. 2022 (Αναφορά: σελίδα 65)
url: https://thehackernews.com/2022/03/new-linux-kernel-cgroups-vulnerability.html
(Επίσκεψη 11/02/2024).

[113] E. Mountain, T. McCormick, C. Tafani-Dereeper και F. Baguelin. Escaping containers using the dirty pipe vulnerability. 2022 (Αναφορά: σελίδα 65)
url: https://securitylabs.datadoghq.com/articles/dirty-pipe-container-escape-poc/
(Επίσκεψη 11/02/2024).

[114] F. A. Logic. Top docker security vulnerabilities, best practices, & insights. 2020 (Αναφορά: σελίδα 67)
url: https://www.alertlogic.com/blog/top-docker-security-vulnerabilities-best-practices-insights/
(Επίσκεψη 11/02/2024).

[115] Imperva. Arp spoofing (Αναφορά: σελίδα 67)
url: https://www.imperva.com/learn/application-security/arp-spoofing/
(Επίσκεψη 12/02/2024).

[116] Terradue. Libcloud-cli (Αναφορά: σελίδα 71)
url: https://github.com/Terradue/libcloud-cli.

[117] Jsitech. Jshielder (Αναφορά: σελίδα 73)
url: https://github.com/Jsitech/JShielder.

[118] E. Ozer. Nixarmor (Αναφορά: σελίδα 73)
url: https://github.com/emirozer/nixarmor.

[119] zerint. Docker-rootless full setup (Αναφορά: σελίδα 76)
url: https://github.com/zerint/docker-rootless-setup.

[120] T. LeRoy. Docksec (Αναφορά: σελίδα 77)
url: https://github.com/TedLeRoy/docksec.

[121] Docker. Docker bench for security (Αναφορά: σελίδα 77)
url: https://github.com/docker/docker-bench-security.

[122] Portainer. Portainer (Αναφορά: σελίδα 84)
url: https://www.portainer.io/.

[123] yUML. Yuml (Αναφορά: σελίδα 85)
url: https://yuml.me/.

[124] Mermaid. Mermaid (Αναφορά: σελίδα 85)
url: https://mermaid.live/.

[125] T. A. S. Foundation. Apache libcloud (Αναφορά: σελίδα 93)
url: https://libcloud.apache.org/.

[126] T. A. S. Foundation. Apache (Αναφορά: σελίδα 93)
url: https://www.apache.org/.

[127] T. A. S. Foundation. Apache jclouds (Αναφορά: σελίδα 94)
url: https://jclouds.apache.org/.

[128] Oracle. Java (Αναφορά: σελίδα 94)
url: https://www.java.com/en/.

[129] P. S. Foundation. Python (Αναφορά: σελίδα 94)
url: https://www.python.org/.

[130] T. A. S. Foundation. Apache libcloud - supported providers (Αναφορά: σελίδα 94)
url: https://libcloud.readthedocs.io/en/stable/compute/supported_providers.html
(Επίσκεψη 07/08/2023).

[131] Microsoft. Microsoft azure sdk for python (Αναφορά: σελίδα 95)
url: https://pypi.org/project/azure-mgmt-resource/.

[132] Microsoft. Microsoft azure sdk for python (Αναφορά: σελίδα 95)
url: https://pypi.org/project/azure-mgmt-network/.

[133] Canonical. Ubuntu (Αναφορά: σελίδα 95)
url: https://ubuntu.com/.

[134] Debian. Debian (Αναφορά: σελίδα 95)
url: https://www.debian.org/.

[135] CentOS. Centos (Αναφορά: σελίδα 95)
url: https://www.centos.org/.

[136] Fedora. Fedora (Αναφορά: σελίδα 95)
url: https://fedoraproject.org/.

[137] R. Hat. Red hat (Αναφορά: σελίδα 96)
url: https://www.redhat.com/.

[138] openSUSE. Opensuse (Αναφορά: σελίδα 96)
url: https://www.opensuse.org/.

[139] D. van Heesch. Doxygen (Αναφορά: σελίδα 96)
url: https://github.com/doxygen/doxygen.

[140] L. Sorel-Giffo. Pydoctrace (Αναφορά: σελίδα 99)
url: https://github.com/lucsorel/pydoctrace.

[141] PlantUML. Plantuml (Αναφορά: σελίδα 99)
url: https://github.com/plantuml/plantuml.

[142] S. Rogowski. Code2flow (Αναφορά: σελίδα 104)
url: https://github.com/scottrogowski/code2flow.

[143] C. Koknat. Callgraph (Αναφορά: σελίδα 104)
url: https://github.com/koknat/callGraph.

[144] Bjorn. Pydeps (Αναφορά: σελίδα 106)
url: https://github.com/thebjorn/pydeps.

[145] S. Kumar. Python-dotenv (Αναφορά: σελίδα 107)
url: https://github.com/theskumar/python-dotenv.

[146] Textualize. Rich (Αναφορά: σελίδα 107)
url: https://github.com/Textualize/rich.

[147] Iterative. Shtab (Αναφορά: σελίδα 107)
url: https://github.com/iterative/shtab.

[148] paramiko. Paramiko (Αναφορά: σελίδα 107)
url: https://github.com/paramiko/paramiko.

[149] konsthol. Secdep. 2023 (Αναφορά: σελίδες 108, 118)
url: https://git.konsthol.eu/konsthol/SecDep.

[150] pypa. The python package installer (Αναφορά: σελίδα 109)
url: https://github.com/pypa/pip.

[151] A. Prakash. Linux jargon buster: what are gui, cli and tui in linux? 2024 (Αναφορά: σελίδα 113)
url: https://itsfoss.com/gui-cli-tui/
(Επίσκεψη 28/02/2024).

[152] H. Muhammad. Htop (Αναφορά: σελίδα 113)
url: https://htop.dev/.

[153] B. Moolenaar. Vim (Αναφορά: σελίδα 113)
url: https://github.com/vim/vim.

[154] K. Kanbe. Vuls (Αναφορά: σελίδα 119)
url: https://vuls.io/.

[155] future-architect. Vuls (Αναφορά: σελίδα 119)
url: https://github.com/future-architect/vuls.

[156] future-architect. Vuls architecture (Αναφορά: σελίδα 121)
url: https://vuls.io/docs/en/architecture-remote-scan.html
(Επίσκεψη 05/12/2023).

[157] Vuls. Fast-root scan (Αναφορά: σελίδα 122)
url: https://vuls.io/docs/en/architecture-fast-root-scan.html
(Επίσκεψη 05/12/2023).

[158] CISOfy. Lynis (Αναφορά: σελίδα 123)
url: https://cisofy.com/lynis/.

[159] L. Blast. Lunar (Αναφορά: σελίδα 124)
url: https://github.com/lateralblast/lunar.

[160] CIS. Cis (Αναφορά: σελίδα 124)
url: https://www.cisecurity.org/.

[161] A. W. Services. Amazon ec2 t3 instances (Αναφορά: σελίδα 124)
url: https://aws.amazon.com/ec2/instance-types/t3/.

[162] Vantage. T3.micro (Αναφορά: σελίδα 124)
url: https://instances.vantage.sh/aws/ec2/t3.micro.

[163] ishiDACo. Vulsrepo (Αναφορά: σελίδα 129)
url: https://github.com/ishiDACo/vulsrepo.

[164] Containrrr. Watchtower (Αναφορά: σελίδα 141)
url: https://containrrr.dev/watchtower/.