Ανοικτό Πανεπιστήμιο Κύπρου

Σχολή Θετικών και Εφαρμοσμένων επιστημών

Μεταπτυχιακό Πρόγραμμα Σπουδών *Ασφάλεια Υπολογιστών και Δικτύων*

Μεταπτυχιακή Διατριβή

Increasing Security of Containerized Blockchain using SDN

Ανάργυρος Χήρας

Επιβλέπουσα Καθηγήτρια
Αδαμαντινή Περατικού

Μάιος 2022

Ανοικτό Πανεπιστήμιο Κύπρου

Σχολή Θετικών και Εφαρμοσμένων Επιστημών

Μεταπτυχιακό Πρόγραμμα Σπουδών *Ασφάλεια Υπολογιστών και Δικτύων*

Μεταπτυχιακή Διατριβή

Increasing Security of containerized blockchain using SDN

Ανάργυρος Χήρας

Επιβλέπουσα Καθηγήτρια
Αδαμαντινή Περατικού

Η παρούσα μεταπτυχιακή διατριβή υποβλήθηκε προς μερική εκπλήρωση των απαιτήσεων για απόκτηση μεταπτυχιακού τίτλου σπουδών στην Ασφάλεια Υπολογιστών και Δικτύων.
από τη Σχολή Θετικών και Εφαρμοσμένων Επιστημών
του Ανοικτού Πανεπιστημίου Κύπρου.

Μάιος 2022

ΛΕΥΚΗ ΣΕΛΙΔΑ

Περίληψη

Στόχος της μεταπτυχιακής διατριβής που παρουσιάζεται στην συνέχεια είναι να μελετηθεί πως με την τεχνολογία SDN: Software Define Networking, μπορούμε να προσφέρουμε μεγαλύτερη ασφάλεια, όσο αφορά την διαθεσιμότητα (availability) των υπηρεσιών, έναντι απειλών που χρησιμοποιούν την τεχνική των επιθέσεων κατανεμημένης άρνησης παροχής υπηρεσιών DDoS: Distributed Denial of Service, σε ένα δυναμικό περιβάλλον που βασίζεται σε containerized permissioned blochchain network.

Για την έρευνα μας ακολουθήθηκε η πειραματική μέθοδος, μέσω της δημιουργίας του κατάλληλου περιβάλλον εργασίας και στην συνέχεια έγινε η ανάπτυξη της προτεινόμενης λύσης.

Τα αποτελέσματα, έδειξαν την επιτυχία όσο αφορά τον εντοπισμό πιθανών επιθέσεων τύπου DDoS που ερευνάτε όπως επίσης και την δυνατότητα που μας δίδεται να αντιμετωπίσουμε επιτυχώς τις επιθέσεις αυτές εκμεταλλευόμενοι τις δυνατότητες για προγραμματισμό που μας παρέχει η τεχνολογία SDN, κάνοντας χρήση των δεδομένων που συλλέγονται μέσω του framework που αναπτύξαμε.

Summary

Main goal of our thesis proposal, as it is present below, is to study how we can use the technology of SDN: Software Defined Networking in order to achieve increased security, in the means of service availability, against threats that use the technic of DDoS: Distributed Denial of Service attack, in an technologically dynamic environment that is based on containerized blockchain network.

For our research we follow the experimental approach, building up a proper testing environment in which we developed our proposed solution.

The outcomes shown a great goal achievement on what has to do with the prompt detection of DDoS attacks as well as the feasibility we have to mitigate these attacks through the programming capabilities that SDN technology offers, using as source data the data are collected on the developed framework.

Ευχαριστίες

Ένα μεγάλο ευχαριστώ στην γυναίκα μου που όλο αυτό τον καιρό στάθηκε δίπλα μου να υπομένει την αγωνία και το άγχος μου και να μου δίνει δύναμη. Επίσης, στην κορούλα μου (μηνών) που κάποιες φορές δεν κατάφερα να παίξω μαζί της, αλλά μου δίνει κουράγιο με το χαμόγελο της.

Τέλος, να ευχαριστήσω ειλικρινά την επιβλέπουσα καθηγήτρια κ. Περατικού Αδαμαντινή για την υπομονή της και την υποστήριξη της για την ολοκλήρωση της μεταπτυχιακής διατριβής.

Περιεχόμενα

ΛΕΥΚΗ ΣΕΛΙΔΑ

Κεφάλαιο 1
Εισαγωγή

Σε αυτή την μεταπτυχιακή διατριβή, θα μελετηθεί πως θα μπορούσε μια νέα τεχνολογία στα δίκτυα υπολογιστών, η τεχνολογία των SDN: Software Defined Networking, να προσφέρει μεγαλύτερη ασφάλεια δικτύου (network security) όσο αφορά την διαθεσιμότητα (availability) των υπηρεσιών, της ακεραιότητας (Integrity) και της εμπιστευτικότητα (confidentiality) των δεδομένων, έναντι των απειλών (threats) και των κινδύνων (risks) που ενδεχομένως παρουσιάζονται σε ένα δυναμικό περιβάλλον όπως είναι ένα blockchain containerized environment σε σχέση με την ασφάλεια που προσφέρει ένα δίκτυο που βασίζεται στην παραδοσιακή αρχιτεκτονική (Traditional Network).

Ως blockchain θα μπορούσε να περιγραφή μία βάση δεδομένων, η οποία είναι γεωγραφικά κατανεμημένη. Η Βάση αυτή, έχει συγκεκριμένη δομή, η οποία αποτελείται από ένα σύνολο blocks συνδεδεμένων το ένα με το άλλο με μία χρονική ακολουθία (αλυσίδα από blocks). Οι χρήστες μπορούν να έχουν πρόσβαση, ελεύθερα προσβάσιμη σε όλους ή σε συγκεκριμένους μόνο χρήστες, στο σύνολο της και να επιβεβαιώνουν την εγκυρότητα των στοιχείων της. Η τεχνολογία των blockchains είναι ιδιαίτερα δημοφιλής σήμερα καθώς προσφέρει ιδιωτικότητα (privacy) των χρηστών και ασφάλεια των συναλλαγών (μέσω μηχανισμών κρυπτογράφησης) με διάφορες εφαρμογές (implementations) όπως είναι τα ψηφιακά νομίσματα (π.χ. bitcoin), πιστοποίηση εγγράφων (π.χ. poex), "έξυπνα" συμβόλαια (smart contract,όπως π.χ. ethereum) και άλλα.

Το container, πρόκειται για ένα απομονωμένο περιβάλλον λειτουργίας εφαρμογής (isolated application runtime environment) που μοιράζεται συγκεκριμένους πόρους (CPU, RAM, Storage και Network) του λειτουργικού συστήματος στο οποίο φιλοξενείται. Η τεχνολογία αυτή, επιτρέπει την γρήγορη ανάπτυξη (deployment), τον εύκολο έλεγχο (testing) και την φορητότητα (portability) των εφαρμογών αυτών. Επιπλέον γίνεται δυνατό πολλαπλοί προγραμματιστές (multiple developers) να δουλεύουν στο ίδιο έργο να

αναπτύσσουν και να βελτιώνουν το παραδοτέο πιο γρήγορα (CI/CD: Continuous Integration/Continuous Deployment).

Κεφάλαιο 2
Βιβλιογραφική Επισκόπηση

Σκοπός του κεφαλαίου είναι να παρουσιαστούν κάποιες από τις σημαντικότερες έρευνες που έχουν γίνει τα τελευταία χρόνια στο χώρο της ασφάλειας των δικτύων με την χρήση της τεχνολογίας Software Define Networking, ξεκινώντας με την περιγραφή της τεχνολογίας, στην συνέχεια παρουσιάζοντας τις νέες προκλήσεις στο χώρο της ασφάλειας και τέλος τα πεδία της τεχνολογίας στα οποία έχει μέχρι σήμερα εφαρμοστεί ώστε να προσφέρει μεγαλύτερη ασφάλεια στα συστήματα αυτά.

2.1 Περιγραφή της Τεχνολογίας SDN

Στην (Shin, Xu, Hong, & Gu, 2016), σκοπός της έρευνας είναι η παρουσίαση της νέας τεχνολογίας δικτύων SDN και συγκεκριμένα τα οφέλη που μπορεί να προσφέρει για την ενίσχυση της ασφάλειας δικτύων και τον περιορισμό .

Τα κύρια χαρακτηριστικά της τεχνολογίας Software Define Networking είναι τα εξής:

Α. Μεγαλύτερος έλεγχος του Δικτύου.

Σε αντίθεση με τα παραδοσιακά δίκτυα, όπου οι διαχειριστές καλούνται να επιβλέπουν/διαχειρίζονται ξεχωριστά τις δικτυακές συσκευές (όπως τους δρομολογητές, switches), έχουμε την εισαγωγή κεντρικού(ων) κόμβου(ων) –ελεγκτών- που ονομάζονται controller(s) και σύνδεση όλων των δικτυακών συσκευών κάνοντας χρήση του OpenFlow (What is OpenFlow?Definition and How it Relates to SDN, 2013) πρωτοκόλλου με τον/τους controller(s), ώστε οι διαχειριστές να έχουν ενιαίο έλεγχο και εικόνα του δικτύου.

Β. Διαχωρισμό του Control Pane και Data Plane

Με τον όρο control plane, εννοούμε την επικοινωνία των δικτυακών συσκευών μεταξύ τους και η εισαγωγή κανόνων δρομολόγησης της κίνησης των δεδομένων, ενώ με τον όρο data plane αναφερόμαστε στην προώθηση των δεδομένων αυτών. Στην περίπτωση

που τα OpenFlow switches/routers δεν έχουν κάποιο κανόνα (flow rule) για την κίνηση των δεδομένων (data plane) τότε στον πίνακα προώθησης τους (flow table), προωθούν τα δεδομένα αυτά στον controller ο οποίος επιβάλλει το ανάλογο κανόνα.

Γ. Δυνατότητα Προγραμματισμού του Δικτύου.

Το σημαντικότερο χαρακτηριστικό της τεχνολογίας αυτής είναι η δυνατότητα του προγραμματισμού των κεντρικών κόμβων controllers, ώστε να συμπεριφέρονται και να λειτουργούν αναλόγως με τις απαιτήσεις ασφαλείας που θέλει να ενισχύσει/καλύψει κάθε οργανισμός.

Εκμεταλλευόμενοι τα παραπάνω χαρακτηριστικά οι ερευνητές στο χώρο της ασφάλειας πληροφορίων, τα τελευταία χρόνια, έχουν ασχοληθεί με την ανάπτυξη αξιόλογων προτάσεων στις οποίες η έρευνα κάνει αναφορά.

Χαρακτηριστικά, στην δυνατότητα παροχής πρόσβασης (access control) υπηρεσιών, σε αντίθεση με την παραδοσιακή αρχιτεκτονική δικτύων οπού απαιτείται η χρήση Firewall για τον έλεγχο πρόσβασης, έχουμε την δυνατότητα να αναπτύξουμε κώδικα ο όποιος θα «τρέχει» πάνω από τον controller και θα επιβάλλει τους απαραίτητους κανόνες αν επιτρέπεται ή όχι κάποια συγκεκριμένη κίνηση δεδομένων.

Επιπλέον, στην περίπτωση που ανιχνευθεί πιθανών ύποπτη κίνηση τότε μπορούμε να κατευθύνουμε την κίνηση αυτή σε ειδικά διαμορφωμένες δικτυακές συσκευές (honeypots) για περαιτέρω ανάλυση, χωρίς να έχει επηρεαστεί η υπόλοιπη κίνηση.

Στην περίπτωση της καλύτερης εποπτείας του δικτύου, αντί για την εγκατάσταση συγκεκριμένων προγραμμάτων (agents) σε όλο το δίκτυο μας, κάτι το οποίο απαιτεί μεγάλο κόπο σε ένα μεγάλο δίκτυο, το control plane περιοδικά ενημερώνετε με την κατάσταση όλων των δικτυακών συσκευών. Οι παραπάνω δικτυακές συσκευές θα μπορούσαν να είναι συσκευές (Hardware) συμβατές με το πρωτόκολλο OpenFlow όπως επίσης και λογισμικό (Software OpenFlow Switch). Το τελευταίο επιτρέπει την εγκατάσταση του σε κάθε συσκευή που έχει πρόσβαση στο δίκτυο που επιβλέπουμε (π.χ. έξυπνα κινητά – smartphones).

Τα στατιστικά που συλλέγονται μπορούν να επεξεργαστούν περισσότερο και με την ανάπτυξη προγραμμάτων οι διαχειριστές του δικτύου να έχουν πλήρη εικόνα του δικτύου που διαχειρίζονται ανεξαρτήτου μεγέθους και τύπου δικτυακών συσκευών.

Οι Shit et al., ανέδειξαν τις αυξημένες δυνατότητες που προσφέρει η τεχνολογία Software Define Networking όσο αφορά την πρόληψη (prevention) και εντοπισμό (detection) κακόβουλων επιθέσεων όπως και την απόκριση (response) του συστήματος για τον περιορισμό των κινδύνων και την μεγαλύτερη διαθεσιμότητα του δικτύου.

2.2 Προκλήσεις Ασφαλείας

Το SDN, αν και είναι σε θέση να παρέχει μεγαλύτερη ασφάλεια στα δίκτυα υπολογιστών μέσω της παρακολούθησης και ελέγχου της κίνησης (packet flow monitoring and control), μέσω ενός κεντρικού κόμβου (central controller) και να εφαρμόζει άμεσα νέους κανόνες που θα επιτρέπουν ή όχι την κίνηση αυτή (network traffic flow), νέες απειλές έναντι του ίδιου του SDN έχουν εντοπιστεί (Dacier, König, Cwalinski, Kargl, & Dietrich, 2017), (Smeliansky, 2014).

Σκοπός της έρευνας των Dacier et al. (Dacier, König, Cwalinski, Kargl, & Dietrich, 2017) ήταν να αξιολογηθούν τα χαρακτηριστικά της αρχιτεκτονικής SDN κατά πόσο τα θετικά στοιχεία που προσφέρουν στην ασφάλεια δικτύων έχουν μεγαλύτερο όφελος σε σχέση με τις νέες ευπάθειες (vulnerabilities) που ενδεχομένως προκύπτουν, έτσι όπως συζητήθηκαν από ειδικούς (ακαδημαϊκούς και μη) στο χώρο της ασφάλειας και μη των δικτύων κατά την διάρκεια του σεμιναρίου που διοργανώθηκε από το Dagstuhl seminars, οι οποίοι τάσσονταν υπέρ και άλλοι κατά στο το ότι η νέα αυτή τεχνολογία μπορεί να προσφέρει προστιθέμενο όφελος στην ασφάλεια δικτύων.

Ένα από τα κυριότερα χαρακτηριστικά της νέας αρχιτεκτονικής είναι ο κεντρικοποιήμενος έλεγχος του δικτύου μέσω της εισαγωγής των λεγόμενων controllers. Οι controllers, σε σύγκριση με την μέχρι σήμερα γνώστη αρχιτεκτονική δικτύων, προσφέρουν την καλύτερη διαχείριση της κίνησης των δεδομένων (data plane) με την επιβολή πολιτικών δρομολόγησης και ασφαλείας από ένα κεντρικό σημείο. Αν και το παραπάνω αυξάνει την πολυπλοκότητα, τελικά είναι δυνατόν να εντοπιστούν πιο εύκολα και γρήγορα ασυμφωνίες είτε στις πολιτικές ασφαλείας είτε στους πίνακες δρομολόγησης και να περιορίσουμε με αυτόν τον τρόπο τον πιθανές ευπάθειες από την παραπάνω λάθος υλοποίηση.

Παρόλα αυτά οι συμμετέχοντες εγείρουν το θέμα του Μοναδικού Σημείου Αποτυχίας (Single Point of Failure: SPoF (Single Point of Failure - an overview | ScienceDirect Topics, 2016)), για το δίκτυο μας. Αυτό συνεπάγεται πως σε περίπτωση αστοχίας των controllers που μπορεί να προέρχεται από την εκδήλωση μιας απειλής από εσωτερικό ή εξωτερικό

εισβολέα, μπορεί να προκαλέσει ολική ή μερική μη διαθεσιμότητα του δικτύου ή οποιαδήποτε άλλη ζημιά που να θέτει τις υπηρεσίες που παρέχει το δίκτυο σε κίνδυνο.

Η δυνατότητα προγραμματισμού που προσφέρει το SDN, είναι ένα σημαντικό όφελος που ώθησε στο ταχύτερο ενδιαφέρον της επιστημονικής κοινότητας στην ασφάλεια δικτύων να ασχοληθεί για την ανάπτυξη λύσεων που θα μπορούσαν να προσφέρουν ακόμη μεγαλύτερη ασφάλεια. Επιπλέον, όλοι οι συμμετέχοντες συμφώνησαν πως αυτό παρέχει μεγαλύτερη ευελιξία και ταχύτητα στην παραμετροποίηση του δικτύου και έλεγχο του μέσω ανάπτυξης φιλικών προς το χρήστη εφαρμογών (User friendly environment).

Όπως τονίζεται όμως, η δυνατότητα προγραμματισμού των controllers και η ανάπτυξη εφαρμογών ασφαλείας από οποιονδήποτε χωρίς να υπάρχει έλεγχος πρόσβασης (Access Control) αποτελεί μια σημαντική ευπάθεια, καθώς είτε από σφάλμα λογισμικού (bug) είτε από ανθρώπινο λάθος (παράληψη κάποιας πολιτικής ασφαλείας) θα μπορούσε να θέσει το δίκτυο σε κίνδυνο.

Όλοι οι συμμετέχοντες στο τέλος της συζήτησης κατέληξαν στο γεγονός η νέα αυτή τεχνολογία μπορεί να προσφέρει καλύτερη ασφάλεια δικτύων εντούτοις θα πρέπει να αντιμετωπιστούν οι παραπάνω ευπάθειες, με τον έλεγχο των εφαρμογών που αναπτύσσονται ώστε να μην υπάρχουν αστοχίες στο κώδικα ή στις πολιτικές ασφάλειας που πρόκειται να εφαρμόσουν όπως επίσης την ύπαρξη έλεγχο προσπέλασης ώστε να υπάρχει ο σωστός διαχωρισμός καθηκόντων (separation of duties) όσο αφορά τους controllers.

Ο Smeliansky (Smeliansky, 2014)στην μελέτη του, ισχυρίζεται πως οι ανάγκες δικτύωσης σήμερα με την νέες αρχιτεκτονικές υπολογιστικών συστημάτων όπως είναι τα Υπολογιστικά συστήματα σε νέφος (Cloud computing) και η εικονικοποίηση υπολογιστικών συστημάτων (virtualization) δεν μπορούν να καλυφθούν πλέον με τις παραδοσιακές αρχιτεκτονικές δικτύου τα λεγόμενα Traditional Access Networks: TANs, οπού κυρίως αποτελούνται από υλικές/φυσικές συσκευές δικτύου (Hardware network devices) όπως είναι τα firewall, routers και switches καθώς δεν προσφέρουν την ευελιξία (flexibility), επεκτασιμότητα (scalability) και την δυνατότητα να παρέχουν διαφορετικού τύπου υπηρεσίες δυναμικά αναλόγως τις ανάγκες που υπάρχουν εκείνη την στιγμή (convergence).

Ένα δίκτυο βασισμένο στο SDN, μπορεί να «χτιστεί» πάνω από τις υπάρχουσες φυσικές δικτυακές συσκευές και να επεκτείνει τις δυνατότητες του ώστε να καλύπτει τις παραπάνω ανάγκες. Κάνοντας χρήση των χαρακτηριστικών του SDN, δίνεται η δυνατότητα δημιουργίας δυναμικών εικονικών δικτύων, ώστε να εξυπηρετήσουν την κίνηση αντίστοιχων εικονικών συστημάτων που μόλις δημιουργήθηκαν, να προσφέρει συγκεκριμένη ποιότητα υπηρεσίας Quality of Service:QoS σε μια για παράδειγμα κινητή συσκευή που αιτείται πρόσβαση στο δίκτυο όπως επίσης να υλοποιήσουμε μηχανισμούς υψηλής διαθεσιμότητας (high Availability) για μια συγκεκριμένη υπηρεσία που φιλοξενείται σε κάποιο διακομιστή (server).

Όσο αφορά την ασφάλεια δικτύων, ο Smeliansky αναλύει τις απειλές (threats) που υφίστανται σε ένα παραδοσιακό δίκτυο TAN και πως αυτές μπορούν να περιοριστούν σε ένα SDN-based network. Αυτές οι απειλές μπορούν να χωριστούν σε τρία (3) πεδία ενδιαφέροντος που είναι η Υποδομή (Infrastructure), Λογισμικό (Software) και Πρωτόκολλα (Protocols).

Κύρια απειλή όσο αφορά την υποδομή σε ένα παραδοσιακό δίκτυο είναι παραβίαση φυσικής ασφάλειας με κίνδυνο την μη εξουσιοδοτημένη πρόσβαση (unauthorized access) όπου κάποιος κακόβουλος θα μπορούσε να προκαλέσει βλάβη και μη διαθεσιμότητα της υποδομή (π.χ. καταστροφή των δικτυακών συσκευών) ή να έχει πρόσβαση στο δίκτυο (π.χ. σύνδεση σε κάποιο switch), επομένως δεν διασφαλίζεται η εμπιστευτικότητα της πληροφορίας. Αντιθέτως σε ένα SDN δίκτυο, λόγω της κεντρικοποιήμενης αρχιτεκτονικής του, θα πρέπει να διασφαλίσουμε την προστασία της υποδομής στην οποία φιλοξενείται ο controller από την μη εξουσιοδοτημένη πρόσβαση.

Στις απειλές όσο αφορά το λογισμικό, κυρίαρχο ρόλο έχουν τα κακόβουλα λογισμικά (malware) όπου στην περίπτωση των TANs μια επίθεση έναντι του control plane μπορεί να πραγματοποιηθεί με τον παραπάνω τρόπο. Στην περίπτωση του SDN ο διαχωρισμός του control plane και data plane προσφέρει αυξημένη ασφάλεια στο control plane. Επιπλέον, κάνοντας χρήση της αρχιτεκτονικής Network Function Virtualization: NFV (Craven, 2020), όπου πλέον όλες οι δικτυακές συσκευές είναι εικονικές (virtualized), παρέχεται η δυνατότητα να γίνεται ανακατεύθυνση ύποπτης κίνησης, μέσω της δημιουργίας εφαρμογής πάνω στον controller (c-application), σε εικονική συσκευή (Virtual Network Function) για περαιτέρω έλεγχο.

Ωστόσο, μια πιθανή ευπάθεια των εφαρμογών αυτών θα μπορούσε να οδηγήσει σε πιθανή εκμετάλλευση της από κάποιον εισβολέα. Οι εφαρμογές θα πρέπει να

αναπτύσσονται με σκοπό να μπορούν να επαναχρησιμοποιηθούν (reusable), να είναι επεκτάσιμες (scalable) και να υπάρχει επαρκή τεκμηρίωση της λειτουργίας τους.

Σημαντική βαρύτητα δίνεται στα πρωτόκολλα ασφαλείας (security protocols) που χρησιμοποιούνται για την επικοινωνία μεταξύ των network elements στο SDN. Ο Smeliasnky τα διακρίνει σε τρείς κατηγορίες

1. Switch – controller πρωτόκολλο ασφαλείας,
2. c-application πρωτόκολλο ασφαλείας,
3. και controller-controller πρωτόκολλο ασφαλείας.

Στην (Shin, Xu, Hong, & Gu, 2016) επικοινωνία το Open Flow πρωτόκολλο που χρησιμοποιείται αν και προσφέρει κρυπτογράφηση Secure Socket Layer: SSL, κρίνεται ανεπαρκές καθώς δεν προσφέρει την δυνατότητα ταυτοποίησης (Authentication, Authorization, Accounting – AAA requirements) και επιπλέον απαιτεί αρκετή υπολογιστική ισχύς η όποια δεν είναι πάντα διαθέσιμη σε ένα εικονικό μηχάνημα που «τρέχει» ως virtual switch.

Στα πρωτοκόλλα ασφαλείας όσο αφορά τις εφαρμογές του controller (c-apps), η λειτουργία τους θα πρέπει να περιορίζεται μόνο στην ανίχνευση για ύποπτη κίνηση ελέγχοντας την επικεφαλίδα του πακέτου (packet header) και για την περαιτέρω ανάλυση του να προωθούνται σε VNF. Με τον τρόπο αυτό, περιορίζουμε τον κίνδυνο να είναι ο controller ο στόχος μιας επίθεσης.

Σε ένα κατανεμημένο περιβάλλον, είναι απαραίτητη η ύπαρξη επιπλέον controllers ώστε να καλύπτει την περίπτωση της μη διαθεσιμότητας κάποιων εξ αυτών ή για την καλύτερη κατανομή τους φορτίου (load balancing) που πρέπει να εξυπηρετήσει. Ως βέλτιστο πρωτόκολλο επικοινωνίας προτείνεται το Transport Layer Security (TLS).

Περαιτέρω έρευνα είναι απαραίτητη, ειδικότερα στην αύξηση της ασφάλειας της επικοινωνίας μεταξύ των controller – switch καθώς και των controllers μεταξύ τους.

Η ανάγκη να αντιμετωπιστούν οι ευπάθειες της αρχιτεκτονικής του SDN είναι ένα από τα ζητήματα που ασχολείται και η έρευνα (Scott-Hayward, O'Callaghan, & Sezer, 2013).

Οι Scott-Hayward et al. , μέσω της έρευνας τους αναλύουν τις προκλήσεις ασφαλείας που φέρνει μαζί της η νέα τεχνολογία SDN σε κάθε ένα από τα επίπεδα (layers) που αποτελείται (application layer, control layer, data layer) όπως επίσης και στα

πρωτόκολλα επικοινωνίας (application–control interface, control–data interface) που χρησιμοποιούνται ανάμεσα στα παραπάνω layers.

Η έρευνα διαπιστώνει πως τα πιο ευάλωτα συστατικά της δομής του SDN αποτελούν το control και data layer. Τα παραπάνω αποτελούν κύριοι στόχοι επιθέσεων όπως της μη εξουσιοδοτημένης πρόσβασης στον controller, άρνηση παροχής υπηρεσιών (D)DoS , τροποποίηση κανόνων δρομολόγησης, παρακολούθηση της κίνησης του δικτυού, τα οποία παραβιάζουν εκείνα τα χαρακτηριστικά ασφαλείας (Confidentiality, Integrity, Availability: CIA) που θα έπρεπε να προσφέρει ένα ασφαλές δίκτυο.

Αν και οι παραπάνω απειλές έχουν διαπιστωθεί από την επιστημονική κοινότητα, οι ερευνητές στην εργασία τους εντοπίζουν ελάχιστη πρόοδο και προτάσεις να περιορίσουν τις συνέπειες από μια πιθανή επίθεση.

Το σημαντικότερο κατά τους ιδίους, έχει να κάνει με την δυνατότητα προγραμματισμού του δικτύου που είναι ένα από τα βασικότερα πλεονεκτήματα του SDN. Είναι απαραίτητο να σχεδιάζονται εφαρμογές, με τέτοιο τρόπο που δεν θα υπάρχει σύγκρουση μεταξύ των κανόνων ασφαλείας που υλοποιούν. Επομένως, η ανάπτυξη διαδικασιών και ενός ολοκληρωμένου πλαισίου (framework) ανάπτυξης εφαρμογών κρίνεται επιβεβλημένο. Οι Scott-Hayward et al, αναφέρουν τις υπάρχουσες λύσεις που έχουν μέχρι σήμερα υλοποιηθεί για τον έλεγχο και εντοπισμό λαθών κατά την παραμετροποίηση και εφαρμογή κανόνων ασφαλείας, χρησιμοποιώντας τεχνικές όπως τα Binary Decision diagrams (Binary decision diagram| wikipedia, 2022).

Περνώντας στην υλοποίηση ενός SDN-based network, πρέπει και εδώ να εφαρμόζονται οι κατάλληλες διαδικασίες ώστε να υπάρχει ο έλεγχος (audit) των διασυνδεμένων δικτυακών συσκευών (network devices). Αυτό θα επιτρέψει τον γρηγορότερο εντοπισμό προβλημάτων, την απομόνωση μόνο της συσκευής/λειτουργίας στην οποία εντοπίστηκε το πρόβλημα και την απρόσκοπτη λειτουργία του συνόλου του δικτύου.

Η ερεύνα, διαπιστώνει πως η επιστημονική κοινότητα έχει δώσει μεγαλύτερη σημασία στην παρουσίαση προτάσεων που αυξάνουν σημαντικά την ασφάλεια δικτύων χρησιμοποιώντας την τεχνολογία SDN, όπως προτάσεις για συστήματα ανίχνευσης και πρόληψης εισβολών (Intrusion Detection Systems/ Intrusion Prevention Systems), ανίχνευσης ύποπτης κίνησης και προώθηση της σε ειδικές λειτουργικές μονάδες (middle-boxes) για περαιτέρω ανάλυση και λιγότερο στην αντιμετώπιση των απειλών που

εμφανίζονται στην αρχιτεκτονική του, όπου σημαντικό κομμάτι της έρευνας επικεντρώνεται στο πρωτόκολλο επικοινωνίας μεταξύ control–data plane layer.

Στην (Hussein, Elhajj, Chehab, & Kayssi, 2016), γίνεται παρουσίαση μιας νέας υλοποίησης ενός SDN-based δικτύου με την εισαγωγή ενός ακόμη επίπεδου (plane) το security plane.

Η ανάγκη που ώθησε τους ερευνητές στο να προχωρήσουν στην εισαγωγή του παραπάνω layer, έχει να κάνει με το γεγονός ότι οι προτεινόμενες αρχιτεκτονικές και λύσεις βασισμένες στο SDN για την καλύτερη ασφάλεια ενός δικτύου, οδηγούσαν είτε στην μεγαλύτερη αύξηση του όγκου της πληροφορίας που θα έπρεπε να επεξεργαστεί (process) ο controller με αποτέλεσμα να τίθεται ευάλωτος για παράδειγμα σε επιθέσεις TCP SYN flooding (Eddy, 2007) ή προωθώντας πακέτα δεδομένων για τα οποία δεν υπήρχε κάποιος κανόνας στο switch, για έλεγχο είτε οι υλοποιήσεις εισήγαγαν επιπλέον καθυστερήσεις στο δίκτυο (delays) εφαρμόζοντας κάποιον μηχανισμό αναμονής (buffering) επεξεργασίας πακέτων στην περίπτωση που ο όγκος είναι μεγάλος έτσι ώστε να περιοριστεί ο κίνδυνος να τεθεί εκτός λειτουργίας. Επιπλέον, όπως διαπιστώνουν, οι περισσότερες μελέτες έχουν να κάνουν με υλοποιήσεις μηχανισμών εντοπισμού (detection) επιθέσεων –κυρίως (D)DoS– και λιγότερο με την αποτροπή/πρόληψη (prevention) επιτυχημένων επιθέσεων.

Οι Hussein et al., στην πρόταση τους το security plane υλοποιείται πάνω από το control plane ως ανεξάρτητο κομμάτι κώδικα (security module) στον controller, το οποίο μπορεί να τρέχει ξεχωριστά σε άλλη υπολογιστική μονάδα εκτός controller ώστε να μειώσει το φόρτο εκτέλεσης επεξεργασίας (overhead).

Η παραπάνω αρχιτεκτονική, περιλαμβάνει την εγκατάσταση προγραμμάτων (security agents) στην μεριά των software switches (Open vSwitch) τα οποία εκτελούν ένα πρώτο έλεγχο (pre-processing) στις επικεφαλίδες των εισερχόμενων πακέτων (data packet headers) και στην συνέχεια προωθείται η κίνηση αυτή μέσω ενός, διαφορετικού από το control plane, καναλιού επικοινωνίας (Southbound interface) στο security module. Εκεί γίνεται η συλλογή και η ανάλυση της κίνησης που έρχεται από τους security agents. Το module περιλαμβάνει ένα μηχανισμό εντοπισμού (detection engine) ύποπτης κίνησης εξετάζοντας τις επικεφαλίδες των πακέτων, χρησιμοποιώντας την γλώσσα προγραμματισμού Pyretic (Agg & Johanyák, 2017). Για την λήψη αποφάσεων χρησιμοποιείται το μοντέλο Finite State Machine: FSM και χρησιμοποιώντας το πλαίσιο (framework) Resonance το security module επικοινωνεί με τον controller για την εφαρμογή νέων κανόνων ασφαλείας. Η επικοινωνία μεταξύ του security module και

controller γίνεται και εδώ μέσω διαφορετικής διεπαφής, μέσω REST API (northbound) interface.

Η παραπάνω υλοποίηση αποσκοπεί στον εντοπισμό (detect) επιθέσεων τύπου, κατανεμημένης άρνησης παροχής υπηρεσιών (D)DoS attacks μέσω της τεχνικής IP spoofing καθώς και στην πρόληψη (prevent) μέσω της άρνησης πρόσβασης στο δίκτυο, αποκλείοντας την πηγή από την οποία εντοπίστηκε η κακόβουλη κίνηση.

Για να το πετύχουν αυτό οι ερευνητές, σε κάθε switch διατηρείται σε μια βάση δεδομένων (Open Vswitch DB: OVDB) πληροφορία σχετικά με τους νόμιμους χρήστες (hosts) ώστε ο controller να είναι ενήμερος. Επιπλέον, εφαρμόζοντας κάποιο όριο (threshold) κίνησης πακέτων από κάποια πηγή και ελέγχοντας την βάση στο switch αν προέρχεται από νόμιμο ή όχι χρήστη, σε περίπτωση που κάποιο ή και τα δύο παραβιάζονται τότε έχουμε ενημέρωση του controller για αποκλεισμό της παραπάνω πηγής.

Αν και οι δοκιμές που έγιναν απέδειξαν πως υπάρχει πολύ καλή απόκριση του συστήματος εντούτοις περαιτέρω μεγαλύτερης κλίμακας δοκιμές πρέπει να γίνουν ώστε να διαπιστωθεί η αποτελεσματικότητα από άποψη επιπλέον φόρτου στο ίδιο το module και στον controller όσο και τυχόν καθυστερήσεις στο δίκτυο (λόγω του packet processing στα switches).

2.3 Πεδία Εφαρμογών

Ένα από τα προβλήματα που καλούνται να λύσουν οι μεγάλες επιχειρήσεις/οργανισμοί έχει να κάνει με την διαχείριση πολλών υπολογιστικών μονάδων οι οποίες τις περισσότερες φορές είναι εγκατεστημένες σε διαφορετικές –γεωγραφικά – τοποθεσίες καθώς επίσης ένα μεγάλο αριθμό εφαρμογών και υπηρεσιών που πρέπει να προσφέρουν στους τελικούς νόμιμους χρήστες. Για να παρέχουν την ζητούμενη ασφάλεια των αγαθών αυτών έναντι εσωτερικών και εξωτερικών απειλών, οι υπεύθυνοι ασφαλείας έπρεπε να εγκαταστήσουν τα απαραίτητα εργαλεία ασφαλείας που μεταξύ άλλων περιλαμβάνουν μηχανισμούς Ελέγχου Πρόσβασης (Access control mechanisms), Firewalls, IDS/IPS στην σωστή τοποθεσία, ώστε να παρέχουν το ζητούμενο όσο αφορά την ασφάλεια. Λόγω της παραπάνω πολυπλοκότητας αυτό ενέχει τον κίνδυνο για παράδειγμα της μη εξουσιοδοτημένης πρόσβασης, λόγω λάθους παραμετροποίησης ενός συστήματος ασφαλείας.

Οι Shukhaman et al. (Shukhman, et al., 2015) θεωρούν πως τα παραπάνω μπορούν να αντιμετωπιστούν καλύτερα και αποτελεσματικότερα σε ένα SDN-based network. Η ερευνητική τους πρόταση αφορά ένα ολοκληρωμένο σύστημα διαχείρισης ασφάλειας πληροφορίων κάτω από ένα SDN-based δίκτυο, κάνοντας χρήση των χαρακτηριστικών που η τεχνολογία αυτή προσφέρει (κεντρικό έλεγχο του δικτύου, προγραμματισμό). Τα βασικά χαρακτηριστικά που πρέπει να προσφέρει η λύση αυτή είναι η ευελιξία και επεκτασιμότητα.

Η πρόταση τους, περιλαμβάνει έναν ελάχιστο αριθμό από εργαλεία που πρέπει να διαθέτει ένα σύστημα διαχείρισης ασφαλείας, υλοποιημένα ως Software modules πάνω από τον controller (POX controller).

Τα modules που αναφέρουν οι ερευνητές είναι:

1. Authentication module, πρόσβαση στο δίκτυο μόνο από νόμιμους χρήστες.
2. Monitoring module, παρέχει πλήρη εικόνα της κατάστασης του δικτύου συλλέγοντας snmp traps από τις δικτυακές συσκευές.
3. Firewall module.
4. Routing & QoS (Quality of Service) module, παρέχει τους κανόνες δρομολόγησης καθώς επίσης εγγυημένες υπηρεσίες δικτύου σε κόμβους που το αιτούνται.

Επιπλέον υπάρχουν modules τα οποία είναι υλοποιημένα έκτος του Controller ως ξεχωριστές οντότητες:

1. Network management module, βασισμένο σε κάποιο monitoring tool όπως είναι το Zabbix, Nagios και παρέχει την διεπάφη για την παραμετροποίηση των πολιτικών ασφαλείας που θέτουμε.
2. Security Element management module, διαχειρίζεται την λειτουργία των security elements (proxy, antivirus, IDS, Data Leak Protection: DLP analyzers) τα οποία έχουν υλοποιηθεί πάνω από Virtual Machines: VMs, χρησιμοποιώντας το OpenNebula (OpenNebula, 2022) σύστημα διαχείρισης, με το οποίο είναι διασυνδεμένο.

Οι πληροφορίες σχετικά με τα δικαιώματα των χρηστών, την τοπολογία του δικτύου και η τρέχουσα κατάσταση του, οι κανόνες δρομολόγησης και κίνησης των πακέτων αποθηκεύονται σε βάσεις δεδομένών.

Τα εισερχόμενα πακέτα ελέγχονται από το firewall module, στην περίπτωση που υπάρχει διαθέσιμος κανόνας να που επιτρέπει την κίνηση τότε τα πακέτα προωθούνται στο Routing module για την βέλτιστη δρομολόγηση της κίνησης και στην συνέχεια ενημερώνει τους πίνακες προώθησης των Open Flow switches. Σε περίπτωση που είτε δεν υπάρχει κάποιος κανόνας είτε υπάρχει κανόνας για άρνηση προώθησης τότε τα πακέτα απορρίπτονται.

Οι ερευνητές κάνοντας χρήση εργαλείων προσομοίωσης (Mininet simulator) και εκτελώντας διαφορετικά σενάρια κάθε φορά –πλήθος των κανόνων που εισήγαγαν στο firewall-, υπολόγισαν την αποτελεσματικότητα του firewall από την άποψη την απόκριση του συστήματος σε σχέση με την επιπλέον καθυστέρηση που εισάγει (network delay) και της διαθέσιμης χωρητικότητας του δικτύου (bandwidth).

Περαιτέρω ασκήσεις θα πρέπει να γίνουν σε περιβάλλον πιο κοντά σε πραγματικό ώστε να υπολογιστούν επιπλέον παράμετροι όπως ενδεικτικά αναφέρουμε το DB convergence time και το resource utilization of VMs.

Οι Xu & Lu (Xu & Liu, 2016) στην έρευνα τους ασχολούνται με μία από τις σημαντικότερες απειλές των δικτύων που αντιμετωπίζουν οι επιχειρήσεις/οργανισμοί που έχει να κάνει με τις επιθέσεις κατανεμημένης επίθεσης άρνησης παροχής υπηρεσιών Distributed Denial of Services: DDoS attacks. Στόχος των επιθέσεων αυτών είναι για παράδειγμα να θέσουν εκτός λειτουργίας είτε συνολικά το δίκτυο με το να δεσμεύουν το διαθέσιμο εύρος (bandwidth) λειτουργίας του είτε κάποιο δικτυακό κόμβο, όπως ένα εξυπηρετητή (server), καταλαμβάνοντας (overwhelming) τους διαθέσιμους υπολογιστικούς πόρους του.

Η έρευνα κάνει χρήση των δυνατότητων που προσφέρει η τεχνολογία SDN, ώστε να αύξηση την ασφάλεια δικτύων μέσω του κεντρικού ελέγχου του δικτύου και την γρήγορη εφαρμογή κανόνων ασφαλείας στις δικτυακές συσκευές που απαγορεύουν ή επιτρέπουν την κίνηση των πακέτων όπως επίσης την δυνατότητα οι κανόνες αυτοί να περιέχουν πεδία όπως τα packet count και bytes count. Η συνήθη διαδικασία περιλαμβάνει πρώτα τον εντοπισμό του στόχου της επίθεσης και στην συνέχεια στην εύρεση της πηγής (source IP address) του εισβολέα και ο αποκλεισμός του.

Η πρόταση των ερευνητών λαμβάνει υπόψη της τους περιορισμούς των OpenFlow switches όσο αναφορά το συνολικό αριθμό κανόνων που μπορεί να διαθέτουν, καθώς οι

κανόνες αυτοί αποθηκεύονται σε ειδικού τύπου μνήμη Ternary Content-Addressable Memory: TCAM, η οποία λόγω κόστους είναι περιορισμένη.

Για να εντοπιστεί ο πιθανός στόχος της επίθεσης (destination IP address), οι ερευνητές επιλέγουν την εισαγωγή κανόνων με την συλλογή στατιστικών στοιχείων σχετικά με τον όγκο της κίνησης των δεδομένων (data flow volume in and out) όπως επίσης τυχόν ασσυμετρία (packets count/bytes count) μεταξύ της εισερχόμενης και εξερχόμενης κίνησης (inbound and outbound data traffic). Λόγω του περιορισμού του συνολικού αριθμού των κανόνων που μπορεί να διαθέτει κάθε το switch, ο παραπάνω κανόνας δεν εφαρμόζεται για όλα τα switch ενός PoP (Point of Presence (POP), 2019) δικτύου αλλά γίνεται διαμοιρασμός της παρακολούθηση της κίνησης, χωρίζοντας το σε επιμέρους μικρότερα σε μέγεθος δίκτυα. Στην συνέχεια χρησιμοποιώντας τον αλγόριθμο Ford – Fulkerson (Ford-Fulkerson Algorithm for Maximum Flow Problem, 2013) ελέγχεται αν μπορεί ο παραπάνω κανόνας να εφαρμοστεί και σε switch που ελέγχουν μικρότερο αριθμό συνδεδεμένων κόμβων. Για τον εντοπισμό του στόχου/θύματος χρησιμοποιείται το μοντέλο Self-Organizing Mapping:SOM (Self-Organizing Map - an overview | ScienceDirect Topics, 2013) για να ταξινομηθούν τα δεδομένα που συλλέγονται. Αφού γίνει ο εντοπισμός του πιθανού στόχου ή το μικρότερο δυνατό μέγεθος IP addresses range, με παρόμοιο τρόπο γίνεται στην συνέχεια ο εντοπισμός του εισβολέα.

Εκτός της παραπάνω μεθόδου, η οποία ονομάζεται διαδοχική (sequentially) οι ερευνητές προχωρήσαν και στην υλοποίηση της ταυτόχρονης (concurrent) μεθόδου οπού στην περίπτωση που εντοπιστεί ένα εύρος πιθανών στόχων (IP victims range) τότε ταυτόχρονα τρέχει και ο αλγόριθμος για τον εντοπισμό της πηγής της επίθεσης.

Με βάση τις προσομοιώσεις που έγιναν οι ερευνητές κατέληξαν πως και οι δύο μέθοδοι μπορούν να προσφέρουν είτε σχεδόν ακριβή εντοπισμό του θύματος (πιο γρήγορα με την διαδοχική μέθοδο) είτε του επιτιθέμενου (πιο γρήγορα με την ταυτόχρονη μέθοδο) με την προϋπόθεση πως η διαθέσιμη TCAM των switches είναι πολύ μεγάλη. Στην αντίθετη περίπτωση μόνο το εύρος μπορεί να εντοπιστεί.

Η (Machidon, Mladin, Sandu, & Bocu, 2014) ασχολείται με τις δυνατότητες που μπορεί να προσφέρει η τεχνολογία SDN σε μία ανερχομένη τεχνολογία εικονικοποίησης (virtualization), αυτής των –Linux- containers.

Η τεχνολογία των linux containers, σε σύγκριση με τις υφιστάμενες τεχνολογίες virtualization όπως είναι οι full virtualization, para-virtualization και Operating

System:OS – level virtualization, επιτρέπει την παράλληλη εκτέλεση περισσότερων του ενός απομονωμένων (container) linux συστημάτων σε κάθε host, ενώ η εκκίνηση και ο τερματισμός είναι κατά πολύ πιο γρήγορη διαδικασία όπως επίσης και η απόδοση τους καθώς εκμεταλλεύονται πλήρως τους διαθέσιμους πόρους που δεσμεύουν. Ο kernel κάθε host συστήματος είναι υπεύθυνος να διαχειρίζεται και να προσφέρει τους απαραίτητους πόρους (resource provisioning) στα container που εξυπηρετεί.

Στην εργασία αυτή, παρουσιάζεται η δυνατότητα που έχουμε κάτω από ένα SDN δίκτυο να παρέχουμε, δυναμικά, πόρους δικτύωσης σε Linux containers, ελέγχοντας την λειτουργία του δικτύου μέσω εφαρμογών υλοποιημένες πάνω στον SDN controller (c-applications).

Η υλοποίηση έγινε με την χρήση 3 Virtual Machines (VMs) να εκτελούνται σε ένα host – Linux OS χρησιμοποιώντας το Kernel Virtualization Module ως hypervisor- στα οποία έχει εγκατασταθεί η ανοιχτού κώδικα πλατφόρμα διαχείρισης containers και πλέον διαδεδομένη Docker (What is a Container?, 2021). Σκοπός των ερευνητών σε κάθε VM να τρέχουν τουλάχιστον 2 container ώστε να δοθεί η δυνατότητα αργότερα να υλοποιηθούν και ξεχωριστά εικονικά δίκτυα (Virtual LANs:VLANs) μέσω των εφαρμογών (c-applications) του controller. Σε κάθε VM, έγινε η εγκατάσταση ενός virtual switch (vSwitch) και στην συνέχεια δημιουργήθηκαν οι διεπαφές (interfaces) για την επικοινωνία των switches, χρησιμοποιώντας General Routing Encapsulation:GRE tunnels. Μετά την επιτυχή ολοκλήρωση των διεπαφών, δημιουργήθηκαν τα containers σε κάθε VM και τα containers αυτά εκχωρήθηκαν κάτω από την διαχείριση των vSwitches. Τέλος, οι ερευνητές εγκατέστησαν τον controller, που στην περίπτωση αυτή χρησιμοποιήθηκε ο Open Daylight SDN controller, και στην συνέχεια ολοκληρώθηκε η παραμετροποίηση του συστήματος με την εκχώρηση της διαχείρισης των VSwitches στον SDN Controller.

Όπως συμπεραίνουν οι ερευνητές, η παραπάνω υλοποίηση θα μπορούσε να καλύψει άμεσα όποιες δικτυακές ανάγκες θα μπορούσαν να προκύψουν, όπως απομόνωση κάποιων containers από τα υπόλοιπα ακόμη και αν ανήκουν στο ίδιο Virtual Machine, με ταχύτητα και ευελιξία μέσω ανάπτυξης εφαρμογών πάνω από τον controller, χρησιμοποιώντας τα Northbound interfaces που διαθέτει.

Σε ένα SDN-based network οι controllers είναι κύριοι στόχοι επιθέσεων. Μία μορφή επίθεσης είναι έναντι των υπολογιστικών συστημάτων στα οποία εκτελείται ως εφαρμογή ο controller, οι λεγόμενες και ως host-based attacks.

Στην (Azab & Fortes, 2017), παρουσιάζεται μία πρόταση για την αντιμετώπιση αυτών των επιθέσεων, κάνοντας χρήση των χαρακτηριστικών που προσφέρουν η τεχνολογία των containers οι εφαρμογές να εκτελούνται σε απομονωμένο περιβάλλον (isolated run-time environment).

Οι ερευνητές υλοποιούν τον μηχανισμό PAFR (Proactive Attack & Failure Resilient), έναν μηχανισμό απομόνωσης του controller (sandboxing) βασισμένο σε linux containers. Για την υλοποίηση του βασίζονται στα Docker containers καθώς προσφέρει γρήγορη παραμετροποίηση των containers, έχοντας το ίδιο μικρό αποτύπωμα.

Ο μηχανισμός PAFR αποτελείται από ένα σύνολο μηχανισμών με σκοπό την αυτοματοποίηση των διαδικασιών της δημιουργίας σημείων αναφοράς (live checkpoints) και την μεταφορά (migration) του ενεργού controller σε άλλο υπολογιστικό σύστημα (host).

Για την ανάπτυξη (deployment) των containerized controllers, μπορεί να χρησιμοποιηθεί είτε έτοιμος κλώνος (docker image file) είτε τροποποιημένο με βάση τις ανάγκες που μπορεί να υπάρχουν, με την χρήση των docker files. Στην συνέχεια, με το ενσωματωμένο εργαλείο RunC εξάγεται ο παραμετροποιήμενος controller σε μορφή αρχείων (dump files), τα οποία αποθηκεύονται σε κοινόχρηστο αποθηκευτικό μέσο (shared storage) για χρήση του σε περίπτωση αποτυχίας ή επίθεσης έναντι του μηχανήματος στο οποίο φιλοξενείται ο controller. Τα αρχεία που εξάγονται ακολουθούν τις προδιαγραφές του Open Controller Protocol , κάτι που επιτρέπει το να μπορούν να χρησιμοποιηθούν για την ανάπτυξη του controller, με γρήγορο τρόπο εκτός της τρέχουσας υποδομής.

Για την επικοινωνία του controller με τα υπόλοιπα network elements, επιλέγεται η χρήση ξεχωριστού εικονικού δικτύου, με την δημιουργία εικονικών διεπαφών (virtual interfaces) για κάθε container που φιλοξενεί κάποιο host και όχι το δίκτυο που δημιουργείται από το docker. Για να το πετύχει αυτό ο μηχανισμός PAFR εκκινεί τους controllers ως μέρος (child process) της RunC διεργασίας.

Για την δημιουργία των checkpoints (snapshots) οι ερευνητές ενσωματώνουν στο PAFR το εργαλείο CRIU (CRIU project, 2022) για την δημιουργία αρχείων (dump files) τα οποία περιέχουν την τρέχουσα κατάσταση του controller (open files, used memory). Τα αρχεία επίσης αποθηκεύονται σε κοινόχρηστο αποθηκευτικό χώρο (shared storage) έτσι ώστε να διαθέσιμα στην περίπτωση που χρειαστεί να γίνει η μεταφορά (migration) του σε άλλο host.

Η διαδικασία του migration, αποτελείται από την δημιουργία ενός πρόσφατου snapshots και η μεταφορά του στο shared storage, στην συνέχεια η δημιουργία του PAFR-ready host με την χρήση του container dump files και την μεταφορά της IP διεύθυνσης (virtual ip address) που χρησιμοποιεί ο controller από τον host στον οποίο εκτελείται και τέλος η ενημέρωση του με το snapshot που είχε δημιουργηθεί.

Με την εκτέλεση διάφορων σεναρίων, οι ερευνητές προχώρησαν στην αξιολόγηση του μηχανισμού τους όσο αφορά την απόδοση του συστήματος κατά την διαδικασία μεταφοράς του controller σε άλλο host, όσο αφορά την καθυστέρηση (latency) και τη διαθέσιμη διαμεταγωγική ικανότητα (throughput) μεταξύ controller – switch. Επιπλέον αξιολογήθηκε η διαθεσιμότητα του συστήματος έναντι επιθέσεων στο host, για να διαπιστωθεί αν η διαδικασία του συχνού migration αυξάνει την ασφάλεια του controller.

Μία ακόμη εφαρμογή των Docker containers και SDN-based network παρουσιάζεται στη (Saldamli, Sanjeeva, Siddalingaapa, Murugesan, & Ertaul, 2019). Σκοπός της έρευνας είναι να ελέγξουν κατά πόσο η νέα αυτή τεχνολογία μπορεί να προσφέρει ενισχύσει την ασφάλεια όσο αφορά την διαθεσιμότητα (availability) μιας υπηρεσίας που εξυπηρετείται στο δίκτυο της.

Οι Saldamli et al. ανέπτυξαν ένα load balancing μηχανισμό (module) ως c-application, σε ένα POX (python) controller ο οποίος εκτελείται ως Docker container. Το παραπάνω module εξυπηρετεί την απρόσκοπτη λειτουργία ενός web service (location based application).

Ο Load balancer, χρησιμοποιεί τον αλγόριθμο Round-Robin (Load balancing (computing), 2022) για να κατανείμει την κίνηση (http request) σε ένα σύνολο από εξυπηρετητές (pool of web servers), με τέτοιο τρόπο ώστε να μην επηρεάζεται η υπηρεσία ανεξαρτήτου των αιτημάτων που δέχεται.

Η λειτουργία του load balancer έχει ως εξής :

Οι clients αιτούνται περιεχόμενο, με το να συνδέονται σε μια Virtual IP address. Η Virtual IP καθώς και οι IP διευθύνσεις των web servers αποδίδονται κατά την εκκίνηση του controller. Στην συνέχεια, εκτελείται ο Round-Robin αλγόριθμος για να επιλεγεί ο server από τον οποίο θα εξυπηρετηθεί το αίτημα και στην συνέχεια ο controller ενημερώνει το

flow table του OpenFlow switch. Τέλος, ένα http reply επιστρέφει στον client με το περιεχόμενο που αιτήθηκε.

Οι ερευνητές, στην υλοποίηση της παραπάνω πρότασης έπρεπε να λάβουν υπόψη και την διασφάλιση της εμπιστευτικότητας των clients και να διασφαλίσουν πως οι πληροφορίες θέσης (client geolocation) που μοιράζονται δεν μπορούν να αντληθούν από κάποιον μη εξουσιοδοτημένο χρήστη. Για το λόγο αυτό τόσο οι πληροφορίες θέσης, όσο και τυχόν άλλες πληροφορίες κρυπτογραφούνται με τον αλγόριθμο Format Preserving Encryption:FPE, πριν αποθηκευτούν στην βάση δεδομένων.

Η παραπάνω αρχιτεκτονική δοκιμάστηκε σε ένα Linux OS περιβάλλον, με την χρήση του MiniNet emulator για την προσομοίωση της κίνησης των client και servers και POX controller σε Docker container. Για τις μετρήσεις απόδοσης του Web service, χρησιμοποιήσαν το ανοιχτού κώδικα εργαλείο Openload.

Τα πειράματα που έγιναν, απέδειξαν πως η απόκριση του συστήματος ήταν πολύ καλύτερη στην περίπτωση που εκτελούταν ο load balancer.

Αν και η παραπάνω αρχιτεκτονική προσφέρει μεγαλύτερη ασφάλεια όσο αφορά την διαθεσιμότητα της υπηρεσίας, εντούτοις περισσότερη έρευνα χρειάζεται ώστε να διασφαλιστεί η διαθεσιμότητα του controller, καθώς με την τρέχουσα υλοποίηση είναι ευάλωτος έναντι επιθέσεων όπως host based attacks.

Η ταχεία υιοθέτηση του Cloud computing τα τελευταία χρόνια, ως αποτέλεσμα της ανάγκης τόσο των χρηστών όσο και των προγραμματιστών (developers) να έχουν πάντα στην διάθεση τους (always available) τις υπηρεσίες που χρειάζονται και τους απαραίτητους πόρους (resources), οδήγησε με την σειρά του στην αντικατάσταση παραδοσιακών δικτυακών συσκευών με εικονικοποιημένες (Virtualized) δικτυακές συσκευές που προσφέρουν χαμηλότερο λειτουργικό κόστος, περισσότερες λειτουργίες και διαλειτουργικότητα (Interoperability).

Η τεχνολογία Network Function Virtualization (NFV), προσφέρει την δυνατότητα να αντικαταστήσουμε τις λειτουργίες των παραδοσιακών –φυσικών- δικτυακών συσκευών (HW network devices components) με προγράμματα (SW components) που τρέχουν σε Virtualized servers. Τα components αυτά ονομάζονται ως Virtual Network Function (VNF).

Η έρευνα των Gedia και Perigo (Gedia & Perigo, 2018), συγκρίνει την απόδοση ενός VNF λειτουργώντας ως ONOS SDN controller (Open Network Operating System (ONOS) SDN Controller for SDN/NFV Solutions, 2022) VNF ανάμεσα στις δύο τεχνολογίες Virtualization που υπάρχουν σήμερα, τα Virtual Machines και τα Containers. Σκοπός των ερευνητών είναι να εντοπίσουν πια από τις δύο τεχνολογίες προσφέρει καλύτερο περιβάλλον λειτουργίας υπηρεσιών VNF.

Η απόδοση του συστήματος γίνεται με την σύγκριση μια σειρά από παράγοντες/χαρακτηριστικά που επηρεάζουν την λειτουργία ενός δικτύου και είναι η διεματαγωγική ικανότητα (throughput), χρόνος εκκίνησης (των δικτυακών συσκευών ή υπηρεσιών (provisioning time), ποσοστό εκμετάλλευσης των διαθέσιμων πόρων του συστήματος (resource utilization).

Για την διεξαγωγή των πειραμάτων και της μέτρησης των διάφορων χαρακτηριστικών οι ερευνητές χρησιμοποίησαν μία σειρά από ανοιχτού κώδικα εργαλεία που προσφέρουν μεγάλη αξιοπιστία και εγκυρότητα. Τα Docker Containers όσο και τα VMs (δημιουργήθηκαν κάτω από την πλατφόρμα OpenStack Cloud computing), φιλοξενήθηκαν στην ίδια υποδομή, ενώ για το λειτουργικό συστήματος (host OS) χρησιμοποιήθηκε Ubuntu Linux.

Οι ερευνητές για να υπολογίσουν την απόδοση των ONOS SDN VNFs ανάμεσα στις δύο πλατφόρμες χρησιμοποίησαν τα εργαλείο Prometheus ως πλατφόρμα ελέγχου (monitoring tool) το οποίο συλλέγει μετρήσεις (metrics) σχετικά με την χρήση των υπολογιστικών μονάδων όπως είναι CPU, Memory και Disk utilization. Για την συλλογή των παραπάνω μετρήσεων χρησιμοποιήθηκαν τα προγράμματα (agents) cAdvisor – στο περιβάλλον των docker containers- και Node exporter –στο περιβάλλον των VMs. Τέλος, οι μετρήσεις που συλλέγονταν από το Prometheus tool χρησιμοποιούνταν και από το Grafana tool για την δημιουργία των ανάλογων μετρήσεων απόδοσης (Key Performance Indicator:KPI). Όλα τα παραπάνω εργαλεία, φιλοξενούνταν σε αντίστοιχα containers/VMs. Για την δημιουργία του αναγκαίας κίνησης δεδομένων (packet traffic) χρησιμοποιήθηκε το MiniNet emulator, ενώ οι μετρήσεις έγιναν με το εργαλείο Iperf3, για τον υπολογισμό του διαθέσιμου bandwidth.

Οι μετρήσεις, έδειξαν πως η απόδοση του ONOS SDN VNF, είναι συγκριτικά πολύ καλύτερη στην περίπτωση των containers σε σχέση με τα VMs.

Η τεχνολογία blockchain, τα χαρακτηριστικά ασφαλείας, από την πρώτη υλοποίηση του ως ψηφιακό νόμισμα (blockchain 1.0), στην συνέχεια στα έξυπνα συμβόλαια (blockchain 2.0) έως σήμερα (blockchain 3.0) όπου οι απαιτήσεις ασφαλείας κυρίως όσο αφορά την ανάγκη για μεγαλύτερη προστασία της ιδιωτικότητας, είναι το αντικείμενο με το οποίο ασχολείται η (Zhang, Xue, & Liu, 2019).

Τα blockchain διακρίνονται σε τρεις (3) κατηγορίες αναλόγως την διαθεσιμότητα τους προς τους χρήστες και το επίπεδο εμπιστοσύνης μεταξύ τους.

Συγκεκριμένα υπάρχουν τα:

- Public blockchain, όπου το blockchain είναι διαθέσιμο σε όλους τους χρήστες. Όλοι έχουν το δικαίωμα να πραγματοποιούν συναλλαγές όπως επίσης και να τις επικυρώνουν.
- Consortium, όπου δικαίωμα να επικυρώνουν τις συναλλαγές έχουν μόνο συγκεκριμένοι χρήστες.
- Private, σε σχέση με το Consortium τα δικαιώματα επικύρωσης των συναλλαγών είναι πιο αυστηρά.

Η αρχιτεκτονική του blockchain, μπορεί να περιγραφεί ως μια κατανεμημένη βάση δεδομένων (distributed ledger) όπου περιέχει ένα σύνολο από αρχεία συναλλαγών, οργανωμένα ιεραρχικά ως μια αλυσίδα από blocks (chain of blocks). Νέα blocks προστίθενται στην αλυσίδα, αφού πρώτα επικυρωθούν από την πλειοψηφία των νόμιμων χρηστών του δικτύου (Peer-to-Peer network). Εφόσον ένα block προστεθεί στο τέλος της αλυσίδας πρακτικά είναι αδύνατο να αλλοιωθεί, καθώς κρυπτογραφείτε με σύγχρονες τεχνικές κρυπτογράφησης (hash functions, Merkle tree, consensus mechanisms).

Σε ένα αποκεντρωμένο σύστημα, η έλλειψη εμπιστοσύνης μεταξύ των χρηστών είναι ένα από τα κυριότερα προβλήματα που θα πρέπει να λυθεί, ώστε να διασφαλιστεί η εγκυρότητα των συναλλαγών. Μέσω των μηχανισμών Ομοφωνίας (consensus algorithms), οι πλειοψηφία των χρηστών φτάνουν σε μια συμφωνία, όποτε επικυρώνεται και η συναλλαγή.

Στόχος των consensus algorithms, είναι να αντιμετωπίσουν μια γνωστή σημαντική απειλή που μπορεί να προκύψει και αφορά την διάδοση ψευδών μηνυμάτων, γνωστό ως Byzantine Generals Problem (BGP) (Byzantine fault, 2022). Για να επιβεβαιωθεί πως το

μήνυμα –ότι μια συναλλαγή είναι έγκυρη- θα πρέπει να αποδειχτεί μέσω μιας διαδικασίας. Οι πιο γνωστοί αλγόριθμοι που χρησιμοποιούνται σήμερα είναι οι

- Proof-of-Work (PoW), χρησιμοποιείται στο πρώτο blockchain που δημιουργήθηκε το bitcoin. Οι χρήστες για να επιβεβαιώσουν την εγκυρότητα της συναλλαγής καλούνται να λύσουν μια δύσκολη συνάρτηση κατακερματισμού. Ονομάζονται miners, και για την συνεισφορά τους αμείβονται με bitcoins.
- Proof-of-State (PoS), οπού οι miners (ονομάζονται validators) επιλέγονται είτε τυχαία είτε με βάση κάποιους περιορισμούς, ενώ κινδυνεύουν με ποινές στην περίπτωση που διαδώσουν ψευδή στοιχεία.

Οι ερευνητές επισημαίνουν πως οι παραπάνω αλγόριθμοι καταναλώνουν αρκετούς υπολογιστικούς πόρους, όποτε απαιτούνται διαφορετικοί αλγόριθμοι ώστε να αναπτυχθούν blockchains εφαρμογές και για άλλες χρήσεις πέρα των ψηφιακών νομισμάτων.

Οι Byzantine Fault Tolerance (BFT algorithms) αλγόριθμοι επιλύουν το πρόβλημα αυτό με πιο διαδεδομένο τον Practical Byzantine Fault Tolerance (PBFT) καθώς μπορεί να εκτελεί γρήγορα ένα μεγάλο αριθμό επικύρωσης συναλλαγών.

Με την χρήση των consensus algorithms, διασφαλίζεται η συνοχή (consistency) του blockchain ότι όλοι οι χρήστες έχουν την ίδια πληροφορία.

Επιπλέον χαρακτηριστικά ασφαλείας που θα πρέπει να διαθέτουν τα blockchains είναι η εμπιστευτικότητα (confidentiality) και ιδιωτικότητα (privacy).

Είναι σημαντικό το περιεχόμενο των συναλλαγών να μην είναι διαθέσιμο στους χρηστές του δικτύου, όπως συμβαίνει για παράδειγμα στο bitcoin (τόσο το περιεχόμενο το συναλλαγών όσο και η διεύθυνση του χρήστη είναι διαθέσιμη σε όλους). Ειδικά, όσο αφορά την ιδιωτικότητα (privacy) κρίνεται ιδιαίτερα σημαντική όταν έχει να κάνει με ευαισθητα προσωπικά δεδομένα , όπως για παράδειγμα συναλλαγές που έχουν σχέση με θέματα υγείας. Η ιδιωτικότητα θα μπορούσε να επιτευχθεί μέσω της ανωνυμίας (anonymity) δηλαδή πέρα της ψευδοανωνυμότητας (pseudoanonymity) που έχουμε σε εφαρμογές όπως το bitcoin θα πρέπει να διασφαλίσουμε πως κάποιος τρίτος δεν θα μπορούσε να συμπεράνει με κανένα διαθέσιμο μέσο την σχέση ανάμεσα σε δύο οντότητες (unlikability).

Η (Steichen, Hommes, & State, 2017) περιγράφει την λειτουργία ενός τείχους προστασίας (firewall) υλοποιημένο κάτω από ένα SDN-based δίκτυο. Σκοπός των ερευνητών είναι να θωρακίσουν ένα blockchain δίκτυο από επιθέσεις DDoS έναντι των κόμβων που είναι υπεύθυνοι για την επικύρωση των συναλλαγών.

Αν και στα public blockchain, όπως το bitcoin επιθέσεις τύπου DDoS είναι πρακτικά αδύνατο να έχουν επιτυχή αποτελέσματα λόγω του τεράστιου αριθμού των nodes που συμμετέχουν στο Proof-of-Work consensus mechanism στην περίπτωση των Cosnortium ή Private blockchains, ο αριθμός των nodes που συμμετέχει στον παραπάνω μηχανισμό εγκυρότητας είναι περιορισμένος.

Υλοποιώντας το ChainGuard ως c-application πάνω από ένα OpenFlow controller, προσφέρουν ασφάλεια έναντι DDoS επιθέσεων όπως και έναντι απειλών μη εξουσιοδοτημένης πρόσβασης σε ένα multichain blockchain.

Τα multichain blockchain, είναι permissioned blockchains που περιέχουν εγγραφές σχετικά με το ποια nodes επιτρέπονται να συνδέονται μαζί τους. Χρησιμοποιούνται κυριώς ως consortium ή private blockchains.

Για την λειτουργία του Access Control, το ChainGuard, διατηρεί τρεις (3) διαφορετικές λίστες με nodes. Στην πρώτη whitelist, έχουμε τη λίστα με τα Nodes που επιτρέπεται (legitimate nodes) να συνδέονται στο blockchain και ήδη υπάρχει κανόνας (flow rule) στο OpenFlow switch. Στην δεύτερη λίστα greylist, υπάρχουν τα Nodes για τα οποία δεν υπάρχει κάποιο rule στο switch, όποτε προωθείτε στον controller για να γίνει έλεγχος αν περιέχεται ή όχι στη λίστα με τα ενημερωμένα nodes που έχουν δικαίωμα πρόσβασης. Τέλος, η τρίτη λίστα blacklist, περιέχει τα nodes τα οποία είναι γνωστό ότι δεν έχουν δικαίωμα πρόσβασης στο blockchain και το αντίστοιχο rule υπάρχει ήδη στο switch.

Η παραπάνω λίστα, ενημερώνεται τακτικά από τα nodes του blockchain , οπότε με την σειρά τους ενημερώνονται τα flow entries στα switches.

Για τον περιορισμό των επιθέσεων (D)DoS, η greylist διαθέτει περιορισμένη χωρητικότητα (fixed capacity). Στην περίπτωση που η παραπάνω λίστα γεμίσει τότε ένας νέος κανόνας για την απόρριψη των νέων –αγνώστων- πακέτων εφαρμόζεται στα switch για ένα μικρό χρονικό διάστημα. Επιλέγεται ένα μικρό χρονικό διάστημα που θα είναι ενεργός ο παραπάνω κανόνας έτσι ώστε να μην γεμίσει με εγγραφές ο πίνακας με τα flows που διατηρούν τα switches.

Τα πειράματα, έδειξαν πως έναντι της DoS επιθέσης η αποτελεσματικότητα του firewall ήταν ικανοποιητική καθώς η λειτουργία του blockchain δεν επηρεάστηκε σχεδόν καθόλου, σε αντίθεση με την DDoS οπού υπάρχει κάποια επιρροή (disruption) στη λειτουργία του blockchain.

Σε ένα SDN-based δίκτυο που εξυπηρετεί εκατοντάδες συσκευές διασυνδεδεμένες μεταξύ τους, είναι απαραίτητο να διασφαλίσουμε την ασφάλεια του δικτύου όσο αφορά την διαθεσιμότητα του όπως επίσης και να περιορίσουμε την πρόσβαση, μόνο σε εξουσιοδοτημένες οντότητες. Σε ένα τέτοιο περιβάλλον είναι απαραίτητο να έχουμε έναν ικανό αριθμό από controllers που να προσφέρουν τις παραπάνω ανάγκες ασφαλείας.

Η (Faizullah, Khan, Alzahrani, & Khan, 2020), συγκρίνει την απόδοση ενός Public blockchain based SDN με ένα permissioned blockchain όσο αφορά την απόδοση του σε επιθέσεις DDoS όπως επίσης και την σύγκλισης (convergence) με τις δικτυακές συσκευές (Open vSwitch).

Οι shao et al. (Shao, Zhu, Chikuvanyanga, & Zhu, 2019) προτείνουν τη χρήση των blockchains για την ασφάλεια των SDN δικτύων χρησιμοποιώντας ένα νέο αλγόριθμο τον Simplified Practical Byzantine Fault Tolerance (SPBFT) για την ανταλλαγή μηνυμάτων (control messages) μεταξύ των controllers και την ανάλυση της αποδοτικότητας του μέσω της θεωρίας των παιγνίων (game theory).

Κεφάλαιο 3
Μεθοδολογία

Η μεταπτυχιακή διατριβή μας, αφορά μια ποσοτική έρευνα καθώς θα προσπαθήσουμε να εξηγήσουμε πως η τεχνολογία SDN, μέσω των χαρακτηριστικών της συνολικής εικόνας του δικτύου (network global view) και της δυνατότητας προγραμματισμού (programmability) που διαθέτει, μπορεί να προσφέρει αυξημένη ασφάλεια σε ένα δίκτυο που φιλοξενεί κόμβους (nodes) υπό την μορφή containers για την ανάγκη ενός ιδιωτικού (private/permissioned) blockchain.

Για τον έλεγχο της παραπάνω ιδέας, θα χρησιμοποιήσουμε την πειραματική διαδικασία, καθώς τα πειράματα κρίνονται ως ο καταλληλότερος ποσοτικός σχεδιασμός όταν θέλουμε να παρατηρήσουμε αν υπάρχει αλλαγή στην εξαρτημένη μεταβλητή (ασφάλεια δικτύου) και μάλιστα θετική, κάνοντας χρήση της τεχνολογίας SDN που μελετάμε (SDN-based network security).

Τα ερωτήματα που καλούμαστε να απαντήσουμε στην έρευνα αυτή, είναι :

i. Πόσο πιο αποδοτικό μπορεί να γίνει το δίκτυο μας κάνοντας χρήση της τεχνολογίας Software Define Networking:SDN;
ii. Πόσο πιο αποτελεσματικό έναντι επιθέσεων τύπου Κατανεμημένης Άρνησης Παροχής Υπηρεσιών (Distributed Denial of Service:DDoS) είναι ένα SDN-based network;

Η ερευνά μας πρωτοπορεί καθώς είναι από τις ελάχιστες μελέτες που χρησιμοποιεί cyber range περιβάλλον για την διεξαγωγή των πειραμάτων και τον μετριασμό τόσο των εξωτερικών όσο και των εσωτερικών παραγόντων που μπορεί να θέσουν σε αμφισβήτηση την εγκυρότητα των αποτελεσμάτων. Στο cyber range περιβάλλον του Ανοιχτού Πανεπιστημίου της Κύπρου θα χρησιμοποιηθούν ανοιχτού κώδικα (Open Source) εργαλεία με μεγάλη αξιοπιστία, για τις μετρήσεις και την ανάλυση των αποτελεσμάτων.

Τα cyber range (Cyber Range, 2018) environments, είναι ειδικά διαμορφωμένα εικονικά περιβάλλοντα λειτουργίας κατάλληλα για την εκπαίδευση των ανθρώπων που εργάζονται στο τομέα της ασφάλειας πληροφορίων όπως επίσης προσφέρουν την δυνατότητα να γίνεται έλεγχος της αποτελεσματικότητας των προτεινόμενων λύσεων.

Τα τελευταία χρόνια είναι ιδιαίτερα δημοφιλή τόσο από την ιδιωτικό όσο και από τον δημόσιο τομέα, για την εκπαίδευση των ειδικών στο τομέα της κυβερνοασφάλειας (cybersecurity) καθώς προσφέρουν πλήθος σεναρίων επιθέσεων εξομοιώνοντας πραγματικά περιβάλλοντα λειτουργίας μιας επιχείρησης ή οργανισμού (Top Online Cybersecurity Trends for 2020, 2020).

3.1 Προτεινόμενη Λύση

Με την πρόταση μας Advanced Monitoring and Response Application: AMRA, επιδιώκουμε να αντιμετωπίσουμε τις (D)DoS επιθέσεις έναντι της υποδομής μας που προκύπτουν από εσωτερικές απειλές, από κόμβους (hosts) εντός του εταιρικού Data Communication Network:DCN δικτύου που έχουν παραβιαστεί (compromised) από εισβολείς (intruders) και λειτουργούν ως zombies (Zombie (computing), 2022).

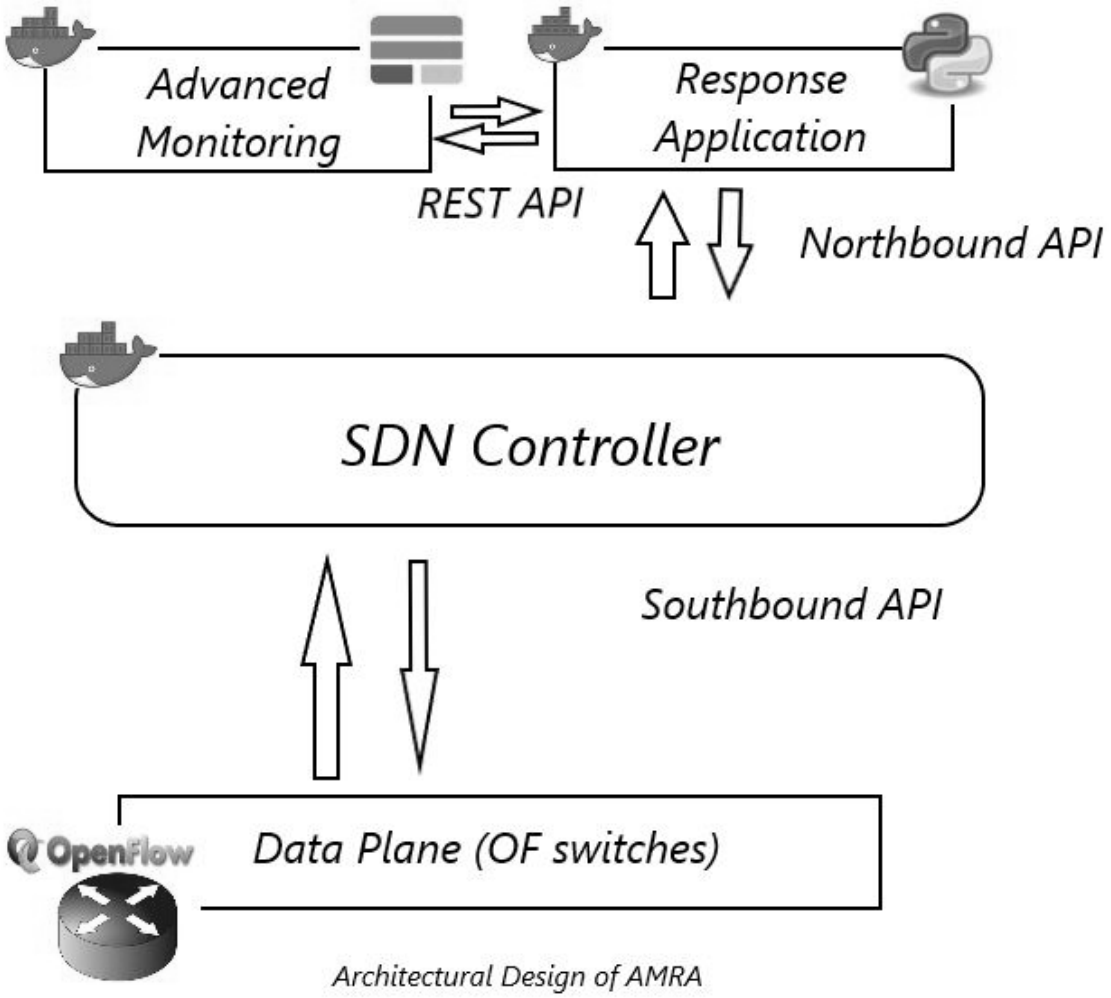
Η πρόταση μας, αποτελείται από ένα ευέλικτο (flexible) και επεκτάσιμο (scalable) πλαίσιο (framework) βασισμένο στην τεχνολογία Software Define Networking: SDN, το οποίο έχει υλοποιηθεί πάνω από τον SDN controller.

Επιπλέον, όπως θα δούμε στην συνέχεια, προσφέρει μικρότερο φόρτο επεξεργασίας (low overhead) στο Data Plane (OpenFlow-enable switches) και καλύτερη διαχείριση της περιορισμένης εσωτερικής μνήμης Ternary Content-Addressable Memory:TCAM, με την καλύτερη εφαρμογή των flow data rules όπως επίσης και προστασία του SDN Controller καθώς τα network flows επεξεργάζονται μέσω ξεχωριστής διασύνδεσης από το AMRA framework.

Αποτελείται από τις παρακάτω λειτουργίες, τα οποία έχουν υλοποιηθεί ως modules στον Controller.

a) Search Engine function
b) Network Monitoring function
c) Attack Detection function
d) Attack Mitigation function

Τα a) και b) αποτελούν το Advanced Monitoring module ενώ τα c) και d) αποτελούν το Response Application module, όπως φαίνεται στο παρακάτω σχεδιάγραμμα (εικ.1)



Εικόνα 1. Αρχιτεκτονική AMRA framework

3.1.1 Advanced Monitoring

Το Module αυτό αποτελείται από τις επιμέρους λειτουργίες την μηχανή αναζήτησης και την επίβλεψη του δικτύου, μέσω των οποίων οι διαχειριστές του δικτύου μπορούν να έχουν σε πραγματικό χρόνο πλήρη εικόνα της κατάστασης του.

Για την υλοποίηση του παραπάνω γίνεται χρήση της πλατφόρμας Elastic Stack. Πρόκειται για ανοιχτού κώδικα πλατφόρμα η όποια αποτελείται από τρία επιμέρους εργαλεία.

i. Elasticsearch Engine.

Η μηχανή αναζήτησης elasticsearch χρησιμοποιείται ευρέως σήμερα από προγραμματιστές (developers) στο χώρο της ανάλυσης δεδομένων (data analytics) καθώς προσφέρει ταχύτητα, ευελιξία στην διαχείριση των δεδομένων που αποθηκεύει όπως επίσης προσφέρει μια σειρά από εργαλεία που την κάνουν συνεχώς επεκτάσιμη

Σύμφωνα με την επίσημη περιγραφή:

“Elasticsearch is a distributed, free and open search and analytics engine for all types of data, including textual, numerical, geospatial, structured, and unstructured. Elasticsearch is built on Apache Lucene and was first released in 2010 by Elasticsearch N.V. (now known as Elastic). Known for its simple REST APIs, distributed nature, speed, and scalability, Elasticsearch is the central component of the Elastic Stack, a set of free and open tools for data ingestion, enrichment, storage, analysis, and visualization (https://www.elastic.co/what-is/elasticsearch)”

Η ερεύνα μας πρωτοπορεί καθώς, αν και η πλατφόρμα προσφέρει λύσεις στον χώρο της ασφάλειας πληροφορίων (Canner, 2021), είναι η πρώτη φορά που θα χρησιμοποιηθεί αποκλειστικά για την συλλογή και ανάλυση network flows. Για την συλλογή των networks flows θα εγκαταστήσουμε στα permissioned blockchain nodes τους packetbeat agents (Packetbeat: Network Analytics Using Elasticsearch, n.d.). Πρόκειται για μικρά σε απαιτήσεις όσο αφορά τους υπολογιστικούς πόρους (lightweight), προγράμματα (agents) με τα οποία έχουμε την δυνατότητα να αποστέλλουμε (pushed) network flows, μέσω ξεχωριστής διασύνδεσης, REST API, αντί του Southbound (Open Flow interface).

Με την παραπάνω υλοποίηση, αποδεσμεύουμε την αποστολή data packet flows μεταξύ data και control plane, στην περίπτωση που δεν υπήρχε κάποιο forwarding rule στα OpenFlow switches για κάποια κίνηση, στους πίνακες προώθησης (forwarding tables) και προστατεύουμε τον controller από πιθανές επιθέσεις (D)DoS.

ii. Kibana Web application

Το kibana, προσφέρει την δυνατότητα της οπτικής αποτύπωσης των δεδομένων που συλλέγονται στην μηχανή αναζήτησης elasticsearch. Υπάρχει δυνατότητα της δημιουργίας πινάκων (dashboards) για πιο φιλική προς το χρήστη παρουσίαση των δεδομένων, όπως επίσης και απευθείας αναζήτηση στα δεδομένα (raw data) με χρήση της γλώσσας Kibana Query Language: KQL.

Δίνεται η δυνατότητα στους διαχειριστές του δικτύου, όπως είναι το Security Operation Center:SOC, να φιλτράρει τα δεδομένα με τέτοιο τρόπο ώστε να έχουν την καλύτερη δυνατή εικόνα της κίνησης του δικτύου (data packet traffic).

3.1.2 Response Application

Το Response application module αποτελεί το βασικό συστατικό (component) της λύσης. Είναι υπεύθυνο για την ανάλυση των δεδομένων που συλλέγονται και αποθηκεύονται στο Advanced Monitoring module.

Αποτελείται από δύο επιμέρους μηχανισμούς.

i. Attack Detection function

Για τον εντοπισμό πιθανής (D)DoS επίθεσης, ο μηχανισμός μας χρησιμοποιεί την τεχνική εντοπισμού ανωμαλιών στα αίτημα σύνδεσης (connection rate-based technique) (Lim, Ha, Kim, Kim, & Yang, 2014) (Shin, et al., 2013), με τον εντοπισμό του αριθμού των προσπαθειών σύνδεσης (number of connections establishment try) (Bawany, Shamsi, & Salah, 2017) προς τους κόμβους του blockchain σε ένα συγκεκριμένο χρονικό παράθυρο.

Η τεχνική βασίζεται στην ιδέα πως, οι εισβολείς –στην περίπτωση μας είναι τα zombie hosts- θα προσπαθήσουν να επιτεθούν ταυτόχρονα σε όσο περισσότερους κόμβους ώστε να επιτύχουν μεγαλύτερο ποσοστό επιτυχίας, μέσα σε ένα χρονικό παράθυρο Δt. Αυτή η ανωμαλία –της ταυτόχρονης προσπάθειας σύνδεσης (connection setup)- δεν παρατηρείται σε κίνηση που προέρχεται από νόμιμα (legitimate) αιτήματα σύνδεσης, που συνήθως αφορά αιτήματα σύνδεσης προς συγκεκριμένο κόμβο (target host) και με πιο σταθερό ρυθμό (lower rate).

Ο αλγόριθμος μας παίρνει ως είσοδο τα Source IP address (possible malicious hosts), target port number και Destination IP address list (Permissioned blockchain network).

Επιπλέον, θέτει ως χρονικό παράθυρο (Δt) το διπλάσιο του sampling time που χρησιμοποιεί ο agent (10 sec).

Στην εικ.2 περιγράφεται ο attack detection algorithm

```
# Detection Algorithm
# Set the interval 2*Δt
threshold.timer = 2*Δt
#Max simultaneously connection
threshold..set = MaxValue
#Destination port to monitor
#Can be customized e.g. TLS/SSL
Destination.port = { TrustedPorts }
# A list of proetected peers
Destination.IPs = { BlockchainNetwork }

if count(Source.IPs) < threshold.set && timer < = threshold.timer :
{
    if dest.port NOT IN Destination.port :
    {
    # Possilble sniffing Drop packet
    # Instruct mitigation mechanism
        Action = 1
    }
    else :
    {
    # Instruct mitigation mechanism
    # Insert allow flow rule
       Action = 0
    }
}
else :
{
# Attack has been identified
# Instruct mitigation mechanism
# Drop packets
Action = 1
}
```

Εικόνα 2. Attack Detection algorithm

ii. Attack Mitigation function

Για την αντιμετώπιση των επιθέσεων, υπάρχουν πολλές τεχνικές που μπορούμε να εφαρμόσουμε σε ένα SDN-based network, μεταξύ των οποίων να απορρίψουμε την κίνηση (drop packets), να μην επιτρέψουμε συγκεκριμένη πόρτα επικοινωνίας (block port), να ανακατευθύνουμε την κίνηση (traffic redirection) ή να απομονώσουμε κάποια κίνηση (traffic isolation) (Bawany, Shamsi, & Salah, 2017).

Στην υλοποίηση μας χρησιμοποιούμε την τεχνική της απόρριψης των πακέτων (drop packets) στην περίπτωση που εντοπιστεί κακόβουλη ενέργεια από τον μηχανισμό εντοπισμού επίθεσης που είδαμε παραπάνω. Σε αντίθετη περίπτωση, η κίνηση

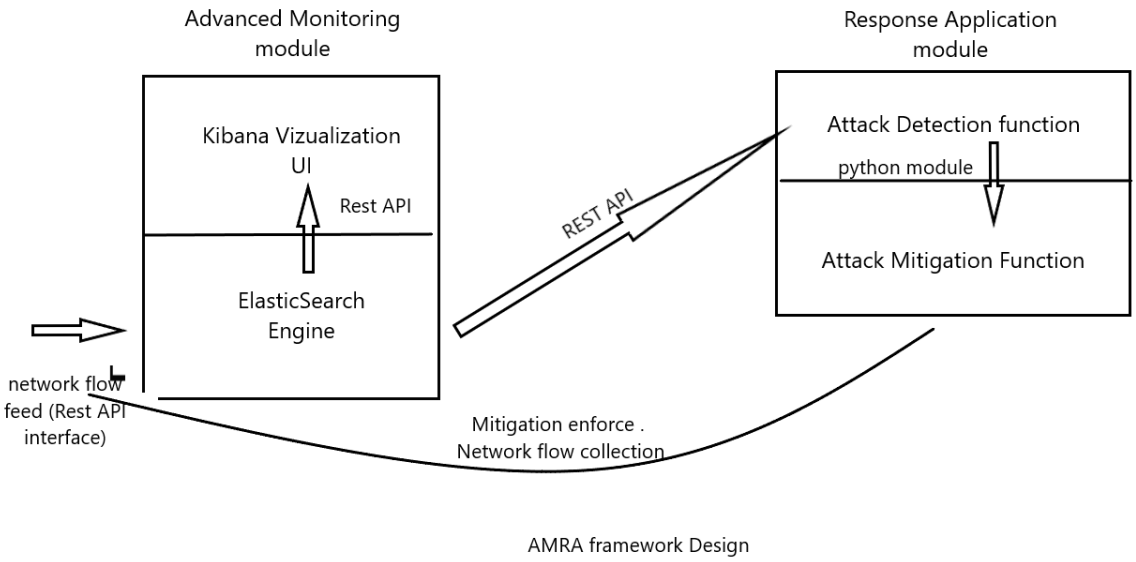
επιτρέπεται και ο αντίστοιχος forwarding κανόνας εγκαθίσταται στα OpenFlow (OF) switches (εικ.3).

Πριν την εισαγωγή ενός νέου κανόνα (flow rule) στα OF switches ο μηχανισμός ελέγχει την κατάσταση των switches, μέσω της ανταλλαγής OpenFlow μηνυμάτων, αν υπάρχει δυνατότητα για την εισαγωγή νέων κανόνων στο flow table που διαθέτουν όπως επίσης αν υπάρχει κάποιος κανόνας που δεν έχει χρησιμοποιηθεί για αρκετό χρόνο, όποτε ο αντίστοιχος κανόνας διαγράφεται.

```
#Attack mitigation algorithm

#Check the OF switch status
# 1. Flow table status
# 2. Remove old (unused) flow rule
{
//Code is skipped
}

# Mitigation run
# input action value of detection algorithm

if detection.action == 1 :
{
# Instruct controller
# to Drop packets
ACTION_SET = DROP_PACKETS
}
else :
ACTION_SET = FORWARD_PACKETS
```

Εικόνα 3. Attack Mitigation Algorithm

Το πλαίσιο (framework) του Advanced Monitoring & Response Application: AMRA παρουσιάζεται παρακάτω (εικ.4):



Εικόνα 4. AMRA framework

Κεφάλαιο 4
Υλοποίηση

Το περιβάλλον στο οποίο θα αναπτυχθεί η λύση και θα πραγματοποιηθούν πειράματα αποτελείται από τρία (3) εικονικά μηχανήματα (VMs: Virtual Machines) με λειτουργικό σύστημα Linux στα οποία τρέχουν, ανάλογα την υπηρεσία που προσφέρουν, τα containers που περιγράφονται στην προηγούμενη ενότητα.

Λόγω των περιορισμένων πόρων που είχαμε στην διάθεση μας, τα παραπάνω VMs φιλοξενούνται σε ένα μόνο μηχάνημα (host) με λειτουργικό σύστημα MS Windows 10 home edition (64bit), με CPU AMD Ryzen 7 3700U – 8 vCPUs- , 16 GB RAM. Για την δημιουργία και διαχείριση των VMs χρησιμοποιήθηκε η λύση του Oracle Virtual Box στην έκδοση 6.1. Το λειτουργικό των VMs είναι Linux Ubuntu 20.04.3 LTS, ενώ για τα containerization επιλέχθηκε το Docker Container στην έκδοση 20.10.7. Τα containers σε κάθε VM θα ανήκουν σε ξεχωριστό δίκτυο (docker bridge) το οποίο θα συνδέεται με το τοπικό virtual switch (Open Virtual Switch) μέσω εικονικής διεπαφής (virtual interface).

Το Open Virtual Switch (OVS), έχει ακριβώς τις ίδιες λειτουργίες με ένα φυσικό L2 - κατά OSI- switch (Open vSwitch 2.5.0 Documentation, n.d.).

Για την επικοινωνία των containers μεταξύ των διαφορετικών VMs, χρησιμοποιήθηκε η μέθοδος των tunnels κάνοντας χρήση του GRE (Generic Routing Encapsulation) πρωτοκόλλου.

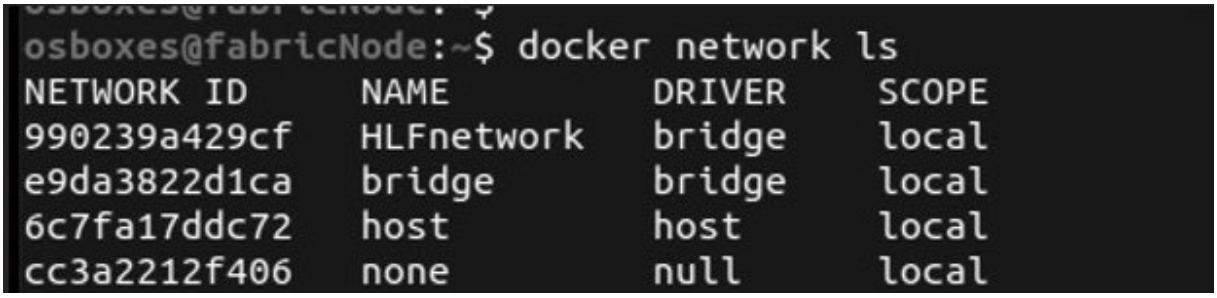
4.1 Εγκατάσταση και Παραμετροποίηση Δικτύων

Ακολουθεί στην συνέχεια η διαστασιοποιήση (dimensioning) κάθε VM.

- VM1
  - Containerized app. HyperLedger Fabric blockchain
  - HW resources: 3vCPUs, 2048 MB RAM
  - eth0 (enp0s3): 192.168.56.103/24
  - docker bridge (HLFnetwork): 172.28.3.0/24
  - switch bridge (br-swHLF): 172.28.30.1/12

Στα παρακάτω στιγμιότυπα οθόνης (εικ. 5 – 10) , παρουσιάζεται η δημιουργία του δικτύου στα οποία διασυνδέονται τα containers.

Εικόνα 5. Δημιουργία docker network



Εικόνα 6. Docker networks

```
osboxes@fabricNode:~$ docker inspect  HLFnetwork
[
    {
        "Name": "HLFnetwork",
        "Id": "990239a429cff91310b69b27826136444ec0c2f52322ba36355dc3f6ae77560b",
        "Created": "2021-04-26T15:00:08.362851492+03:00",
        "Scope": "local",
        "Driver": "bridge",
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": {},
            "Config": [
                {
                    "Subnet": "172.28.3.0/24",
                    "Gateway": "172.28.3.254"
                }
            ]
        },
        "Internal": false,
        "Attachable": false,
        "Ingress": false,
        "ConfigFrom": {
            "Network": ""
        },
        "ConfigOnly": false,
        "Containers": {},
        "Options": {},
        "Labels": {}
    }
]
osboxes@fabricNode:~$
```

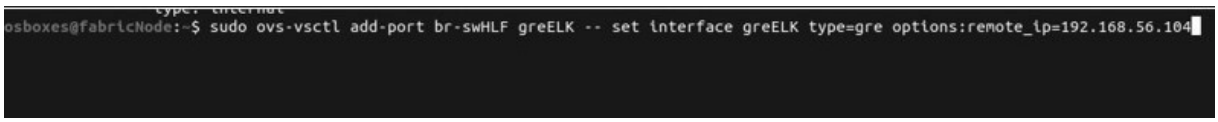
Εικόνα 7. Λεπτομέρειες HLFnetwork

```
osboxes@fabricNode:~$ ip link |grep br
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    link/ether 08:00:27:90:a4:9d brd ff:ff:ff:ff:ff:ff
    link/ether 08:00:27:c0:d0:5f brd ff:ff:ff:ff:ff:ff
    link/ether 96:61:b5:e7:a8:8b brd ff:ff:ff:ff:ff:ff
5: br-swHLF: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UNKNOWN mode DEFAULT group default qlen 1000
    link/ether aa:0e:cd:23:f1:40 brd ff:ff:ff:ff:ff:ff
    link/gre 0.0.0.0 brd 0.0.0.0
    link/ether 00:00:00:00:00:00 brd ff:ff:ff:ff:ff:ff
    link/ether 00:00:00:00:00:00 brd ff:ff:ff:ff:ff:ff
    link/ether 7e:cc:ca:70:c4:4f brd ff:ff:ff:ff:ff:ff
    link/ether 02:42:0f:6b:4b:75 brd ff:ff:ff:ff:ff:ff
11: br-990239a429cf: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP mode DEFAULT group default
    link/ether 02:42:85:b1:3c:f1 brd ff:ff:ff:ff:ff:ff
    link/ether 82:5b:f7:f6:7d:de brd ff:ff:ff:ff:ff:ff
13: veth0@veth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master br-990239a429cf state UP mode DEFAULT group default q
len 1000
    link/ether 6e:55:00:7a:16:e2 brd ff:ff:ff:ff:ff:ff
```

Εικόνα 8. VM1 available interfaces

```
osboxes@fabricNode:~$ ip route
10.19.0.0/16 dev docker0 proto kernel scope link src 10.19.0.1 linkdown
172.16.0.0/12 dev br-swHLF proto kernel scope link src 172.28.30.1
172.28.3.0/24 dev br-990239a429cf proto kernel scope link src 172.28.3.254
192.168.56.0/24 dev enp0s3 proto kernel scope link src 192.168.56.103 metric 100
osboxes@fabricNode:~$
```

Εικόνα 9. VM1 routes



Εικόνα 10. VM1 GRE tunnel

- VM2
  - Containerized application: ELK (Elasticsearch Kibana), pox controller
  - HW resources: 3 vCPUs, 4096 MB RAM
  - eth0 (enp0s3): 192.168.56.104
  - docker bridge (ELKnetwork): 172.28.4.0/24
  - switch bridge (br-swELK): 172.28.30.2/12

Για την δημιουργία των διεπαφών (interfaces), εκτελέστηκαν οι παρακάτω εντολές

sudo ovs-vsctl add-br br-swELK

sudo ip addr add 172.28.30.2/12 dev br-swELK

sudo ip link add veth0 type veth peer name veth1

sudo ovs-vsctl add-port br-swELK veth1

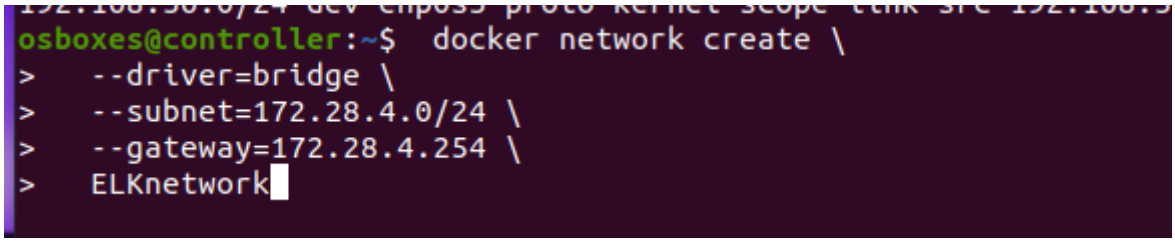
sudo brctl addif br-9ff088496e03 veth0

sudo ip link set veth1 up

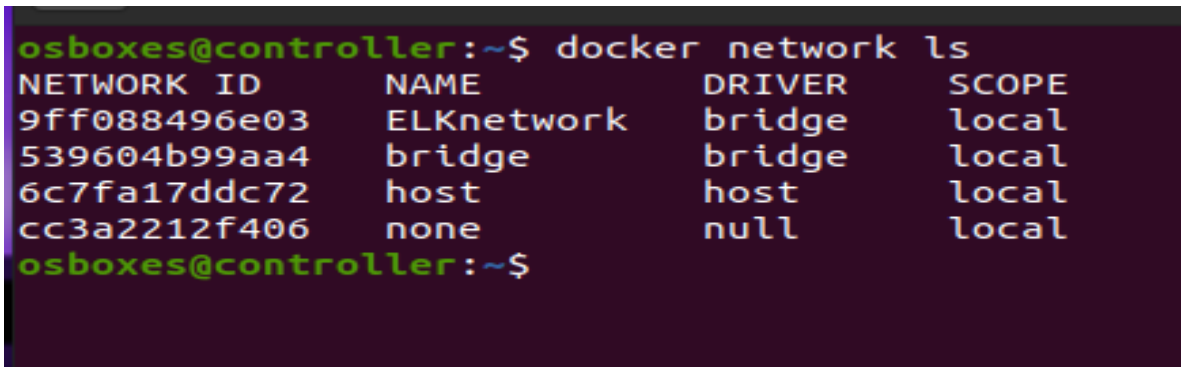
sudo ip link set veth0 up

sudo ip link set dev br-swELK up

Για την δημιουργία του Docker δικτύου, τύπου bridge, χρησιμοποιήθηκε η εντολή docker network create (εικ. 11 -12) .

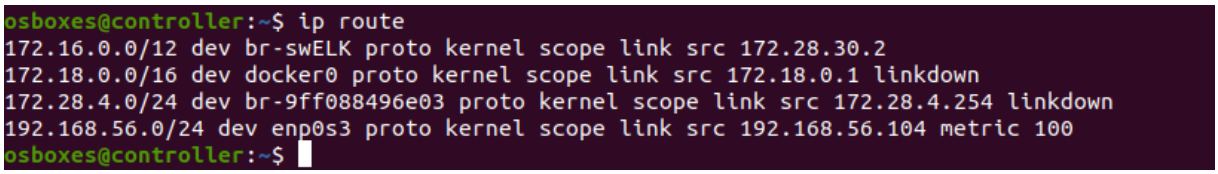


Εικόνα 11. Δημιουργία bridge network



Εικόνα 12. Επιβεβαίωση δημιουργίας

Επιβεβαιώνουμε τις διαθέσιμες δρομολογήσεις που έχουμε δημιουργήσει στο συγκεκριμένο Virtual Machine (εικ.13).



Εικόνα 13. Ip routes

- VM3

Ομοίως για το τρίτο (3) Virtual Machine, το οποίο θα χρησιμοποιηθεί για την προσομοίωση των επιθέσεων έναντι του HyperLedger Fabric blockchain network, έχουμε την εξής διαστασιοποίηση

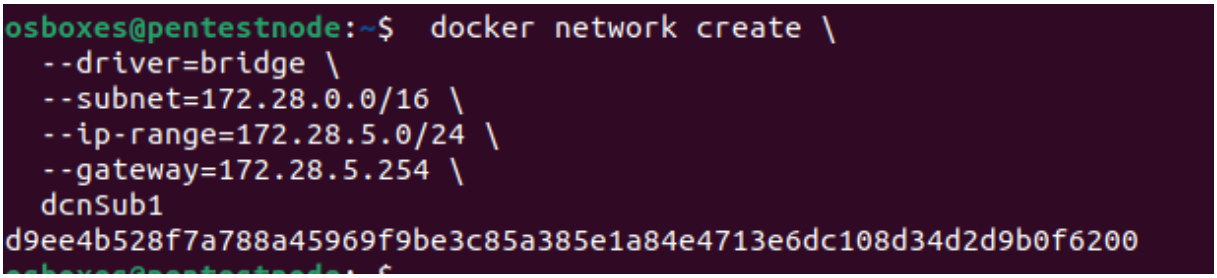
  - HW resources: 2 vCPUs, 2048 MB RAM
  - eth0 (enp0s3): 192.168.56.109
  - docker bridge (dcnSub1): 172.28.5.0/24
  - switch bridge (br-sw1): 172.28.30.3/12

Προχωρούμε στην δημιουργία του Docker bridge network, με την παρακάτω εντολή

```
docker network create \
 --driver=bridge \
 --subnet=172.28.0.0/16 \
 --ip-range=172.28.5.0/24 \
 --gateway=172.28.5.254 \
 dcnSub1
```

Στην παρακάτω εικόνα (εικ.14), φαίνεται η επιτυχής δημιουργία του νέου Docker network.

```
docker network create \
 --driver=bridge \
 --subnet=172.28.0.0/16 \
 --ip-range=172.28.5.0/24 \
 --gateway=172.28.5.254 \
 dcnSub1
```

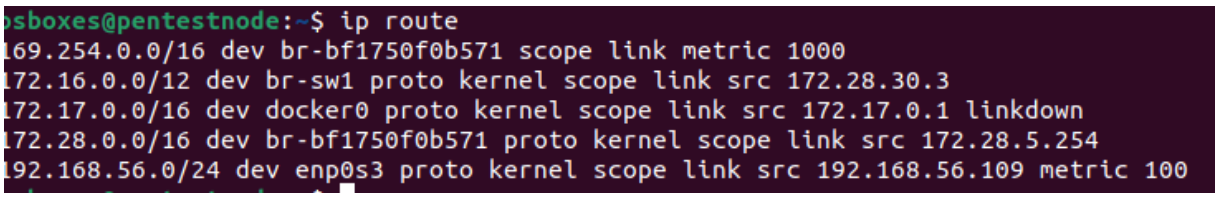


Εικόνα 14. Δημιουργία Docker bridge

Δημιουργία των διεπάφων (interfaces), με τις αντίστοιχες εντολές που χρησιμοποιηθήκαν κατά την δημιουργία των προηγούμενων Virtual Machines (VM1 και VM2).

```
sudo ovs-vsctl add-br br-sw1

sudo ip addr add 172.28.30.3/12 dev br-sw1

sudo ip link add veth0 type veth peer name veth1
sudo ovs-vsctl add-port br-sw1 veth1
sudo brctl addif br-b529db57d6c3 veth0
sudo ip link set veth1 up
sudo ip link set veth0 up
```

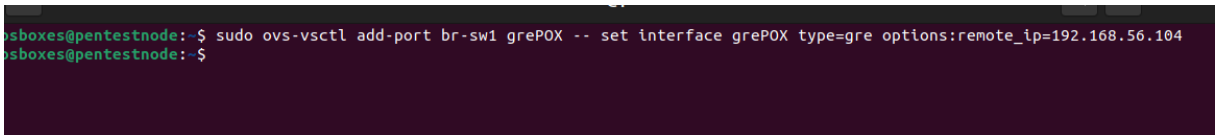
Επιβεβαιώνουμε τα συνολικά δίκτυα (routes) που έχουμε δημιουργήσει (εικ.15).



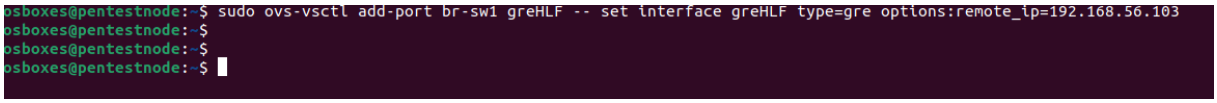
Εικόνα 15. Ip routes VM3

Στην συνέχεια προχωράμε στην διασύνδεση των bridge switches που ανήκουν στα διαφορετικά VMs, χρησιμοποιώντας το General Routing Encapsulation Protocol (GRE tunnels).

Στις παρακάτω εικόνες, φαίνεται η δημιουργία των tunnels με το VM2 (εικ. 16) και το VM1 (εικ. 17), αντίστοιχα.

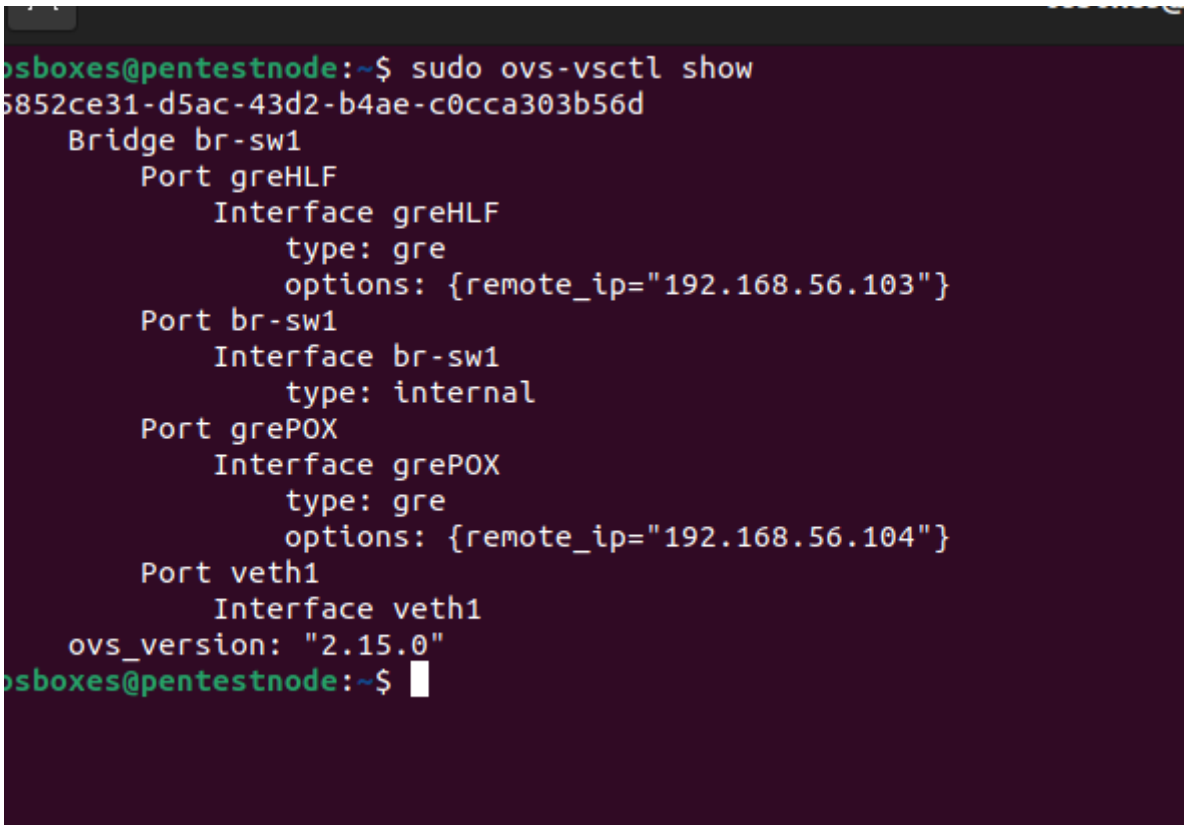


Εικόνα 16. GRE tunnel με το VM2



Εικόνα 17. GRE tunnel με το VM1

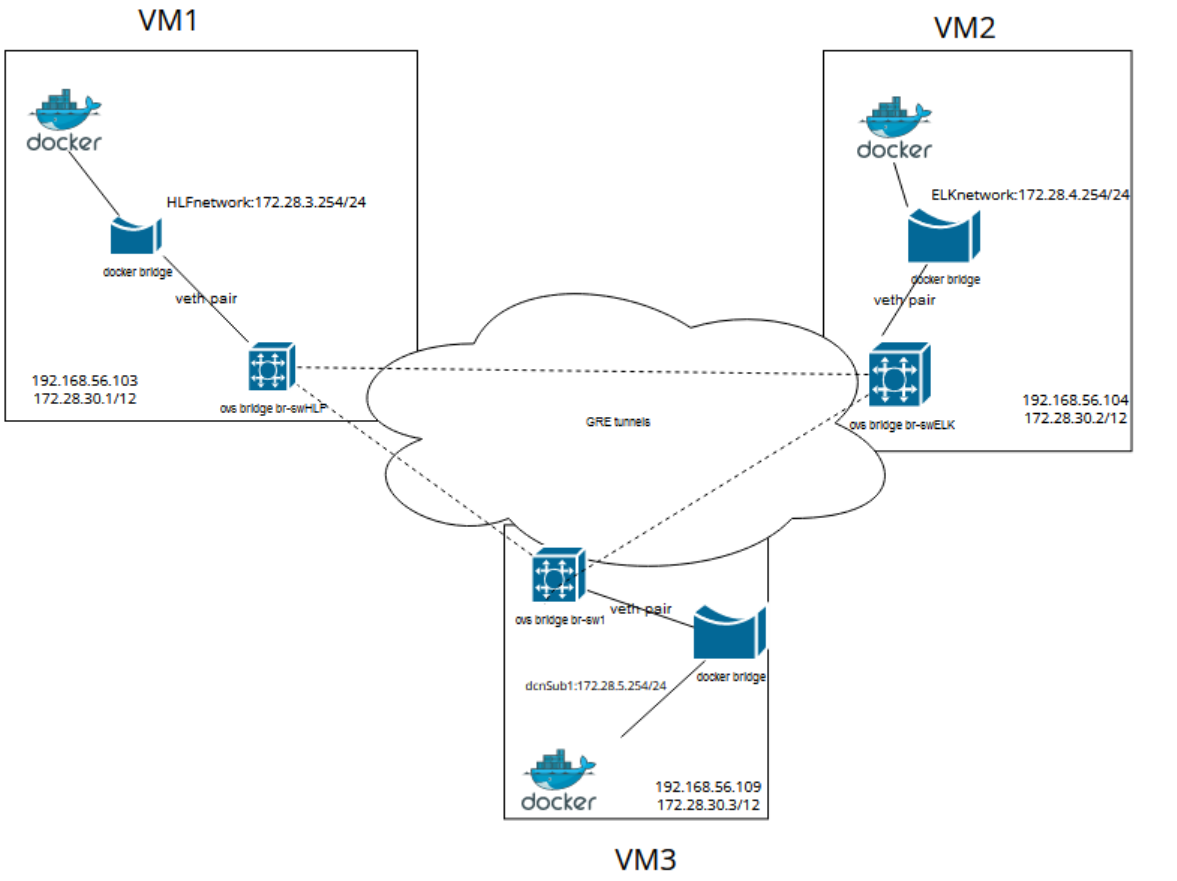
Για τον έλεγχο της δημιουργίας των παραπάνω tunnels, χρησιμοποιούμε την εντολή ovs-vsctl show του Open vSwitch (εικ.18).



Εικόνα 18. Open vSwitch tunnels

Παραμετροποιώντας αντιστοίχως τα υπόλοιπα switches των VM1 και VM2, ολοκληρώνουμε την δημιουργία της διασύνδεσης μεταξύ των VMs της προτεινόμενης λύσης που μελετάμε στην μεταπτυχιακή διατριβή μας.

Στο παρακάτω σχήμα παρουσιάζεται το συνολικό διάγραμμα δικτύου (εικ.19).

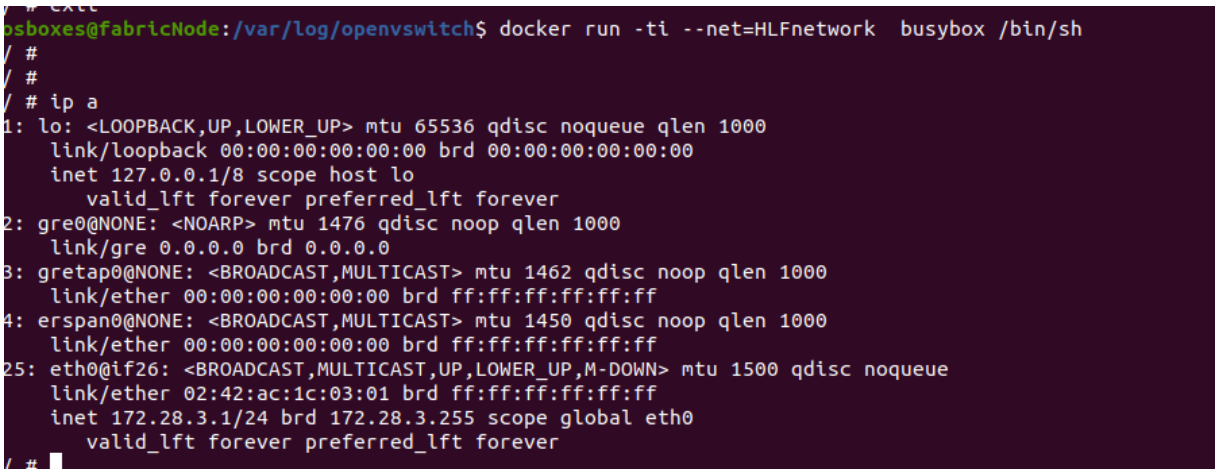


Εικόνα 19. Διάγραμμα Δικτύου

Στην συνέχεια ελέγχουμε την επικοινωνία μεταξύ των containers που ανήκουν σε διαφορετικά VMs, μέσω ICMP echo requests μηνυμάτων.

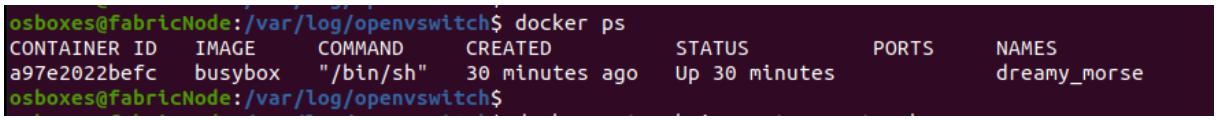
Ενδεικτικά, από το VM1 χρησιμοποιούμε ένα busybox container (Busybox - Official Image | Docker Hub, n.d.), μικρό σε μέγεθος εκτελέσιμο που περιέχει διάφορες εντολές Unix, και ελέγχουμε την επικοινωνία προς ένα άλλο busybox container που ανήκει στο VM2.

Εκτέλεση του busybox container στο VM1 (εικ. 20), με την εντολή docker run –ti –net=HLFnetwork busybox. Με την παραπάνω εντολή, δηλώνουμε πως το container μας θα συνδεθεί στο δικό μας δίκτυο. Επίσης, με τις παραμέτρους ti συνδεόμαστε στο container.



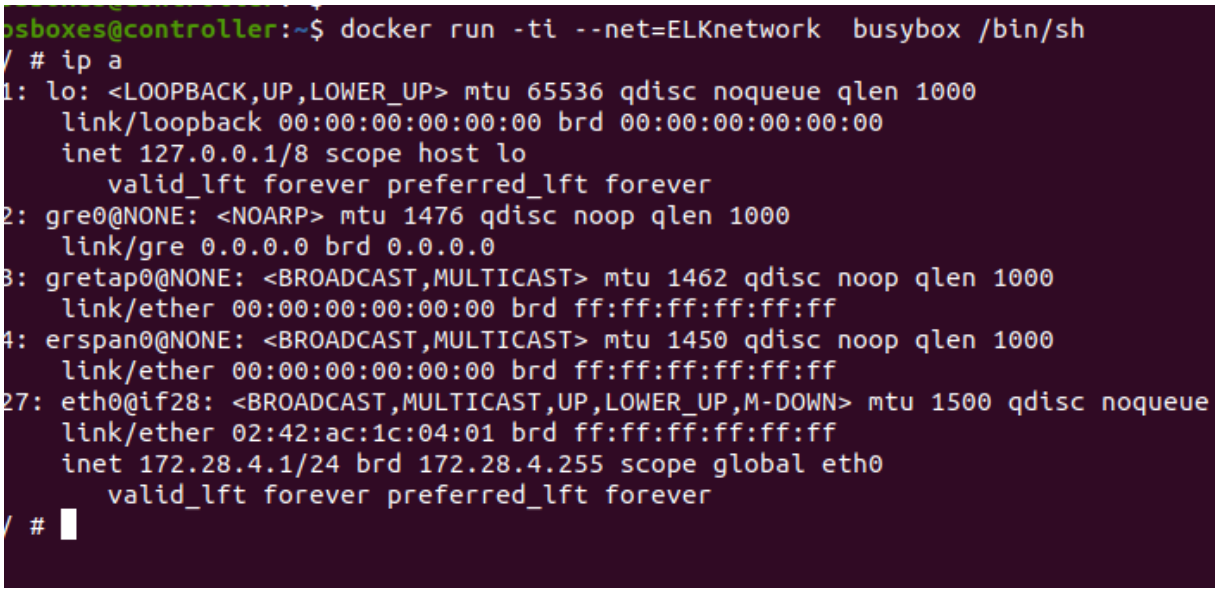
Εικόνα 20. Εκκίνηση του busybox

Σε διαφορετικό shell terminal, ελέγχουμε ότι το docker container εκτελείται (εικ. 21).



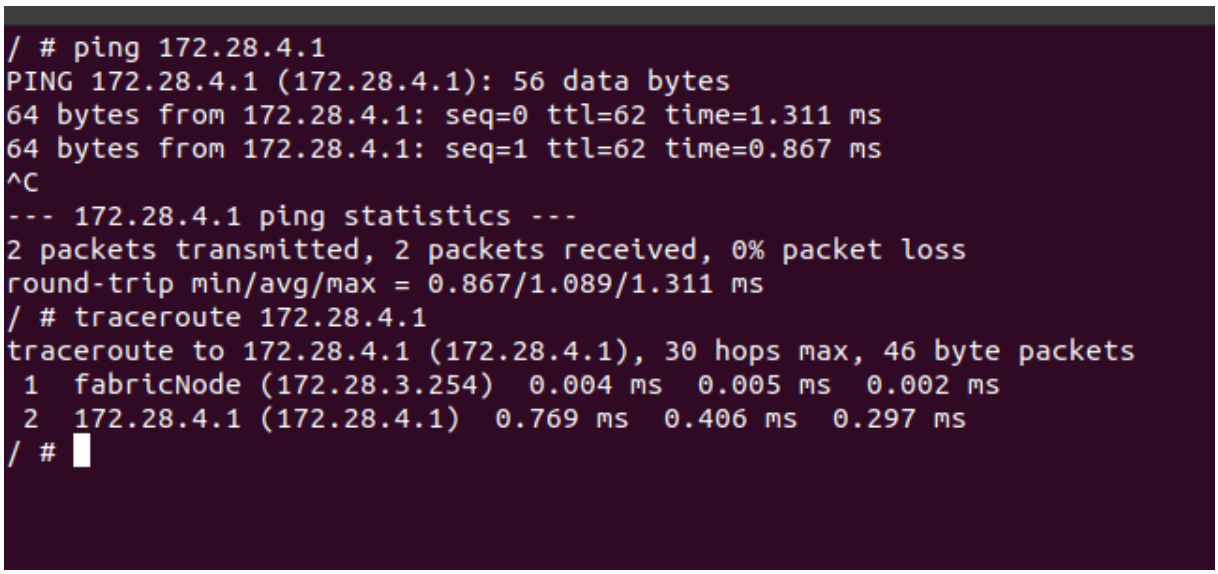
Εικόνα 21. VM1 docker running containers

Ομοίως, εκκινούμε ένα busybox στο VM2, το οποίο θα ανήκει στο bridge network που έχουμε δημιουργήσει (εικ. 22).

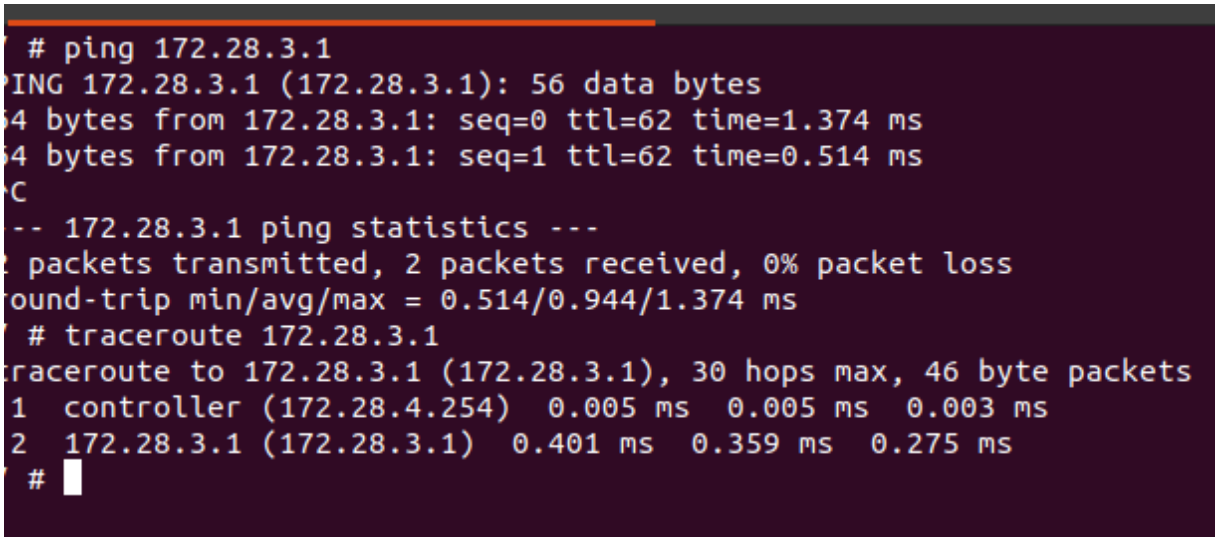


Εικόνα 22. VM2 busybox container

Με την επιτυχή δημιουργία των docker containers, προχωρούμε στο έλεγχο της επικοινωνίας μεταξύ των containers που ανήκουν στα διαφορετικά Virtual Machines, όπως φαίνεται στις παρακάτω εικόνες (εικ. 23 – 24).



Εικόνα 23. VM1 container to VM2



Εικόνα 24. VM2 container to VM1

Στην συνέχεια της διαδικασίας υλοποίησης, προχωρούμε στην εγκατάσταση και παραμετροποίηση των containers.

4.2 Εγκατάσταση και παραμετροποίηση των Containers

Όπως έχουμε αναφέρει στο κεφάλαιο 3 που περιγράφει την μεθοδολογία, στο Virtual Machine 2 θα φιλοξενήσουμε τον SDN controller όπως επίσης το Elastic engine.

4.2.1 POX Controller

Για το POX SDN controller, θα δημιουργήσουμε ένα Dockerfile (εικ.25) για να δημιουργήσουμε το image του containerized pox controller μας. Τα Dockerfiles

(Dockerfile reference, 2022), μας επιτρέπουν την δημιουργία ενός docker image το οποίο περιέχει παραπάνω του ενός service.

```
osboxes@controller:~$ cat Dockerfile
# by Anargyros Chiras
# POX Controller on Alpine
#
# https://github.com/noxrepo/pox
#

FROM alpine:latest

ARG version=eel
ENV POX_VERSION $version

RUN apk upgrade --update --no-cache && \
    apk add --no-cache python2

RUN cd /home/pox && \
    wget https://github.com/noxrepo/pox/archive/$POX_VERSION.zip && \
    unzip $POX_VERSION.zip && rm $POX_VERSION.zip && mv pox-$POX_VERSION pox

WORKDIR /home/pox

EXPOSE 6633
```

Εικόνα 25. POX controller Dockerfile

Για την δημιουργία του image, χρησιμοποιούμε την εντολή docker build.

```
osboxes@controller:~$ docker build -t amra_pox .
Sending build context to Docker daemon  1.953GB
Step 1/8 : FROM alpine:latest
 ---> e9adb5357e84
{....}
Removing intermediate container 40b770d239ff
 ---> fc8dd5cd70d0
Step 7/8 : WORKDIR /home/pox
 ---> Running in c130a326a1ad
Removing intermediate container c130a326a1ad
 ---> 1c5e6b77f8b7
Step 8/8 : EXPOSE 6633
 ---> Running in 4d90b20c285e
Removing intermediate container 4d90b20c285e
 ---> e78deafee731
Successfully built e78deafee731
```

Successfully tagged amra_pox:latest

Ελέγχουμε την επιτυχή δημιουργία του docker image, με την εντολή docker images (εικ. 26).

```
osboxes@controller:~$ docker images |grep -i amra
amra_pox                                          latest    e78deafee731
osboxes@controller:~$
osboxes@controller:~$
```

Εικόνα 26. Docker images

Δημιουργία του container (εικ.27), από το παραπάνω docker image, χρησιμοποιώντας την παρακάτω εντολή.

$docker run --net ELKnetwork --ip 172.28.4.1 -d -it amra_pox sh

Με την παραπάνω εντολή, στο docker container δίνουμε στατική IP διεύθυνση που ανήκει στο υποδίκτυο (subnet) που δημιουργήσαμε και αναφέρεται σε προηγούμενα βήματα.

```
osboxes@controller:~$ docker run --net ELKnetwork --ip 172.28.4.1 -d -it amra_pox sh
e5cdabac16cadc96790e41db2f63e6c23d213d2910fef1df3a0c4b85c71c50ab
osboxes@controller:~$ docker ps
CONTAINER ID   IMAGE      COMMAND   CREATED         STATUS         PORTS      NAMES
e5cdabac16ca   amra_pox   "sh"      6 seconds ago   Up 5 seconds   6633/tcp   epic_bardeen
osboxes@controller:~$
```

Εικόνα 27. POX container run

Για να πάρουμε κονσόλα –να συνδεθούμε στο container- στο pox controller μας, εκτελούμε την εντολή docker exec –it <container_id>, όπως φαίνεται στην παρακάτω εικόνα (εικ. 28).

```
osboxes@controller:~$ docker ps
CONTAINER ID   IMAGE      COMMAND   CREATED         STATUS         PORTS
79bd152f44c8   amra_pox   "sh"      7 minutes ago   Up 7 minutes   6633/tcp
osboxes@controller:~$
osboxes@controller:~$
osboxes@controller:~$
osboxes@controller:~$ docker exec -it 79bd152f44c8 sh
/home/pox #
/home/pox #
/home/pox #
```

Εικόνα 28. Κονσόλα στο POX container

Ξεκινάμε τον controller να λειτουργεί ως OpenFlow controller (εικ. 29).

```
/home/pox # ./pox.py openflow.of_01 --address=172.28.4.1  --port=6633
POX 0.5.0 (eel) / Copyright 2011-2014 James McCauley, et al.
INFO:core:POX 0.5.0 (eel) is up.
INFO:openflow.of_01:[86-61-52-a5-b3-46 1] connected
```

Εικόνα 29. OpenFlow Controller

Ενημερώνουμε το openvswitch να ελέγχεται από τον controller, μέσω της εντολής

ovs-vsctl set-controller br-swELK tcp:<Container_IPaddress:6633> (εικ.30).

```
controller:~$ sudo ovs-vsctl set-controller br-swELK tcp:172.28.4.1:6633
```

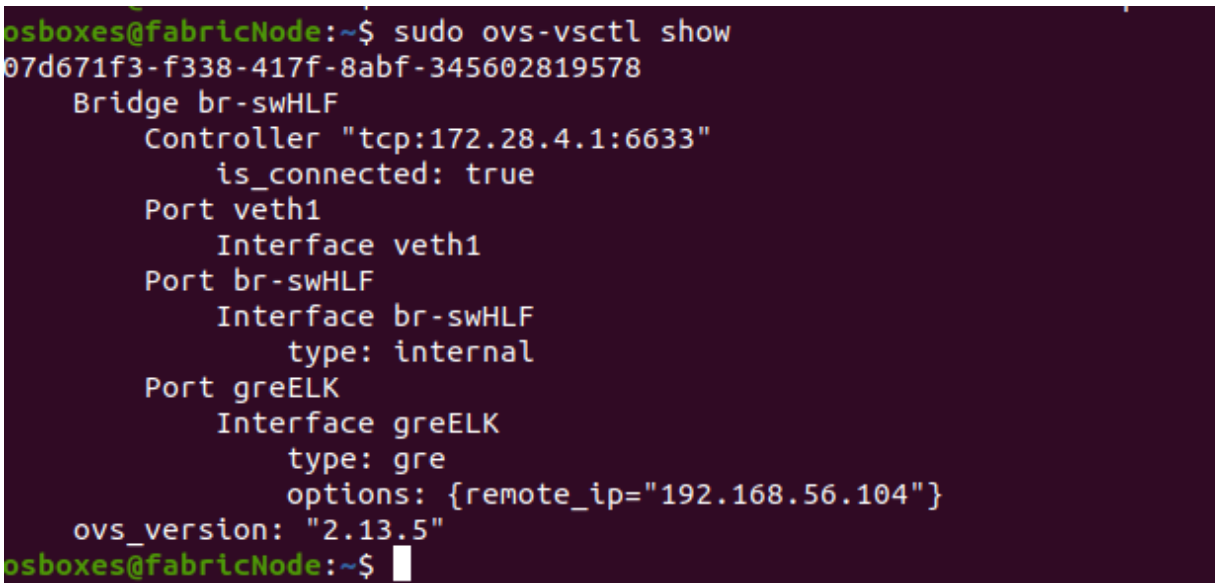
Εικόνα 30. Σύνδεση vSwitch - Controller

Επιβεβαιώνουμε, ότι είναι συνδεδεμένο με τον controller (εικ. 31).

```
osboxes@controller:~$ sudo ovs-vsctl show
[sudo] password for osboxes:
0927aaa4-f84e-49a4-ba22-6fce5f72d2f3
    Bridge br-swELK
        Controller "tcp:172.28.4.1:6633"
            is_connected: true
        Port greHLF
            Interface greHLF
                type: gre
                options: {remote_ip="192.168.56.103"}
        Port veth1
            Interface veth1
        Port br-swELK
            Interface br-swELK
                type: internal
    ovs_version: "2.13.5"
osboxes@controller:~$
```

Εικόνα 31. Κατάσταση bridge Switch

Ομοίως, συνδέουμε τα υπόλοιπα Open vSwitch των Virtual Machines να ελέγχονται (managed) από τον controller (εικ.32).



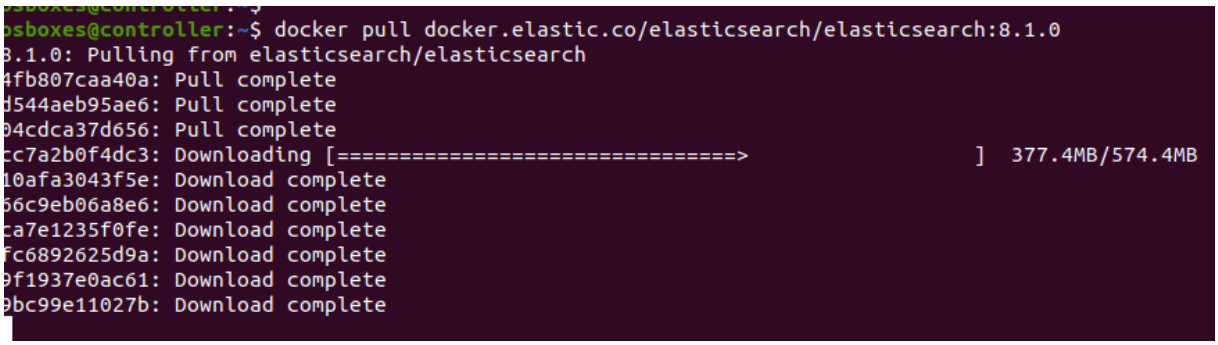
Εικόνα 32. VM1 vSwitch – Controller

4.2.2 Elastic Stack

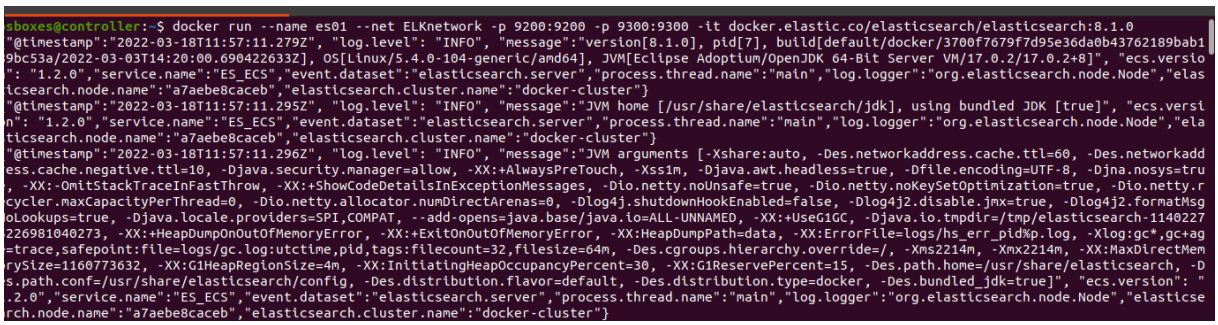
Για την εγκατάσταση του Elastic engine ως Docker container, ακολουθούμε το επίσημο οδηγό (Install Elasticsearch with Docker | Elasticsearch Guide [8.1] | Elastic, 2022), κάνοντας τις απαραίτητες αλλαγές σύμφωνα με τις ανάγκες μας.

Λόγω των περιορισμών που έχουμε στους φυσικούς πόρους του μηχανήματος μας, θα διαστασιοποιήσουμε τον κόμβο μας (docker container), ώστε να χρησιμοποιεί τους ελάχιστους δυνατούς πόρους.

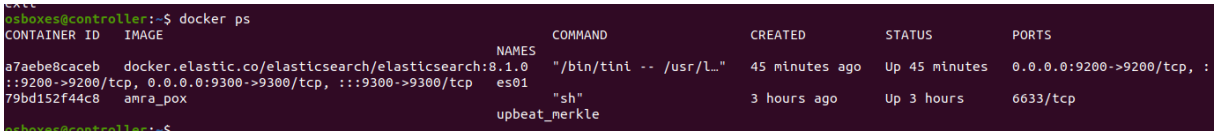
Στα παρακάτω στιγμιότυπα οθόνης (εικ.33-35), περιγράφονται τα βήματα που ακολουθούμε.



Εικόνα 33. Κατέβασμα του Docker image

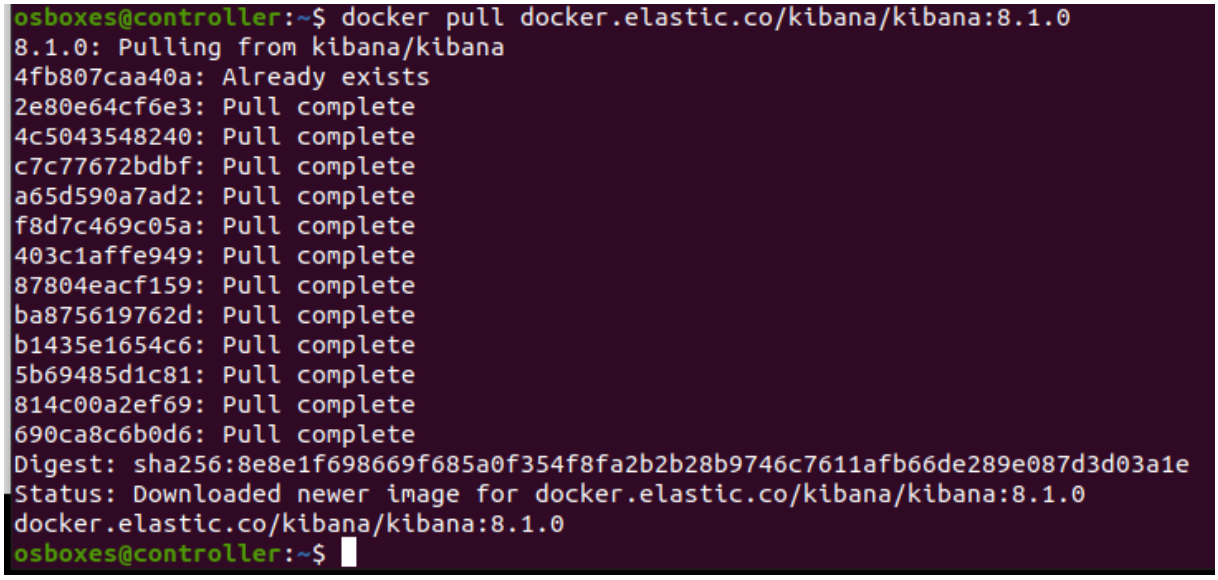


Εικόνα 34. Εκκίνηση Elastic container

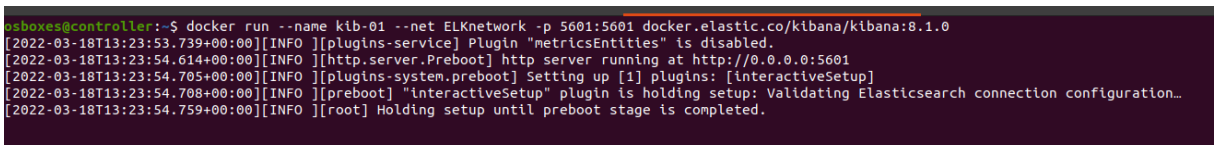


Εικόνα 35. Έλεγχος running containers

Ομοίως, ακολουθώντας τις οδηγίες που παρέχονται από το επίσημο έγγραφο (documentation) (Install Kibana with Docker | Kibana Guide [8.1] | Elastic, 2022), εγκαθιστούμε και το Kibana (εικ.36-37) που θα χρησιμοποιηθεί ως το γραφικό περιβάλλον μέσω του οποίου θα έχουμε οπτική παρακολούθηση της κίνησης που καταγράφεται στο δίκτυο μας (Kibana: Explore, Visualize, Discover Data, 2022) .



Εικόνα 36. Κατέβασμα Kibana docker image



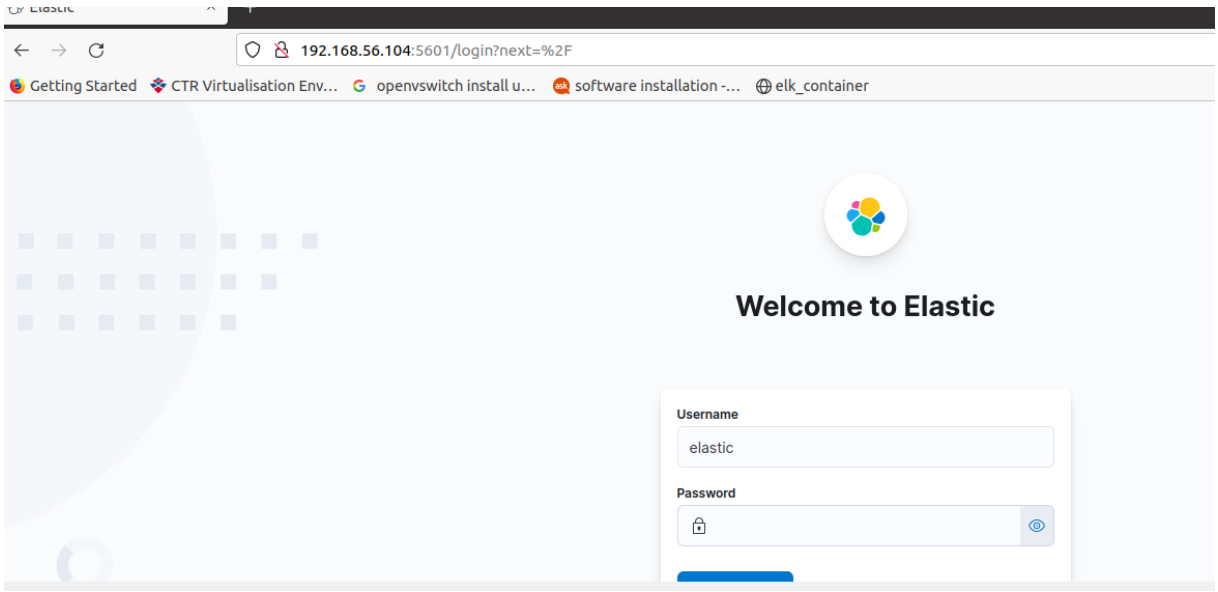
Εικόνα 37. Εκκίνηση του kibana container

Επιβεβαιώνουμε, πως στο node (VM2), τρέχουν ως docker containers τα services των pox controller, kibana και elasticsearch engine (εικ.38).

Εικόνα 38. VM2 running containers

Οι πόρτες (ports) στις οποίες τρέχουν τα services των kibana (port 5601) και elasticsearch (port 9200) είναι διαθέσιμες (exposed) στον κόμβο που φιλοξενούνται (host node).

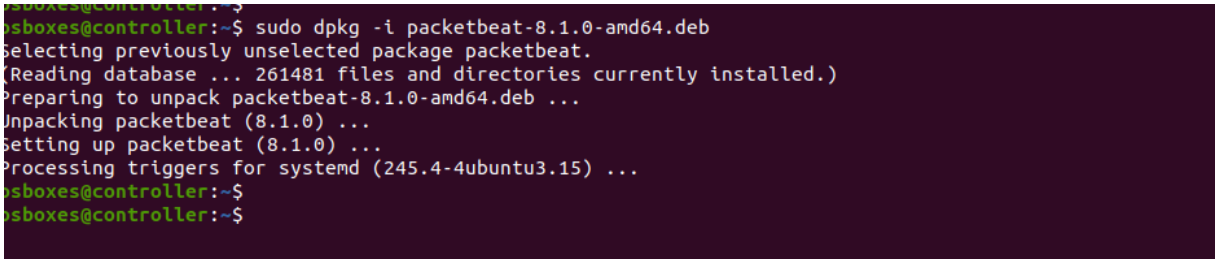
Ενδεικτικά, στην παρακάτω εικόνα (εικ.39), έχουμε ένα στιγμιότυπο οθόνης (screenshot) της διαθεσιμότητας του Kibana User Interface από το VM1.



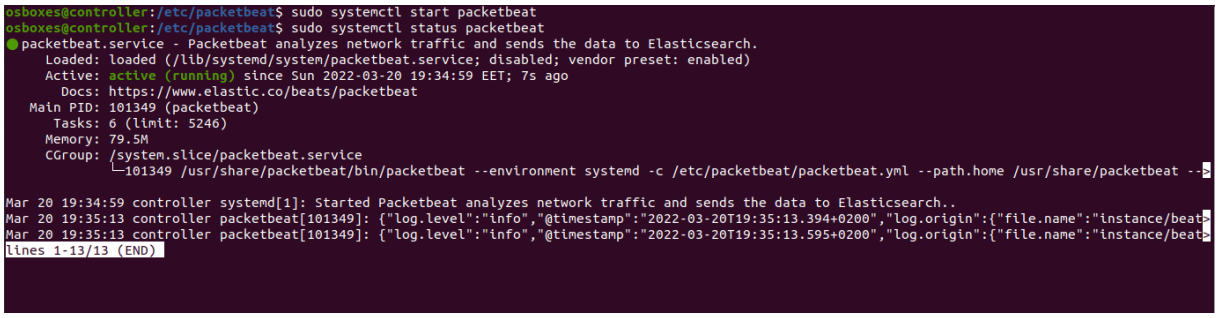
Εικόνα 39. Kibana UI through VM1

Στην συνέχεια της προτεινόμενης υλοποίησης, προχωρούμε στην εγκατάσταση σε όλα τα VMs του packetbeat agent, σύμφωνα με το επίσημο documentation, για την συλλογή των network flows.

Ακολουθεί η εγκατάσταση του agent. (εικ.40-41).

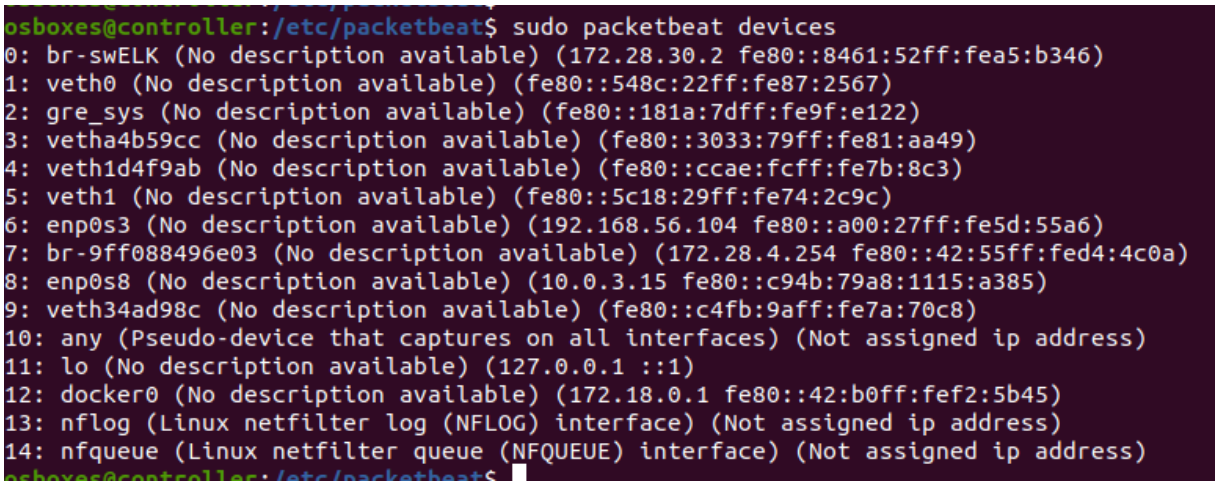


Εικόνα 40. Εγκατάσταση του agent



Εικόνα 41. Εκκίνηση του service

Κατά την παραμετροποίηση του agent, επιλέγουμε την καταγραφή της κίνησης σε όλες τις διαθέσιμες διεπαφές (interfaces) του κόμβου μας (VMs), όπως φαίνεται παρακάτω (εικ.42)



Εικόνα 42. Packetbeat monitor devices

Κατά παρόμοιο τρόπο, προχωρούμε στην εγκατάσταση του packetbeat agent και στα υπόλοιπα Virtual Machines.

4.2.3 Hyperledger Fabric

Με την ολοκλήρωση των docker containers στο Virtual Machine 2 (VM2), προχωρούμε στην εγκατάσταση του containerized permissioned blockchain και συγκεκριμένα του Hyperledger Fabric, όπως έχουμε αναφέρει στο κεφάλαιο 3 της μεθοδολογίας, στο Virtual Machine 1 (VM1).

Όπως έχει ήδη αναφερθεί λόγω των περιορισμών που έχουμε στο περιβάλλον ανάπτυξης όπως προτεινόμενης λύσης, επιλέγουμε να χρησιμοποιήσουμε το έτοιμο περιβάλλον δοκιμών (test network) (Using the Fabric test network — hyperledger-fabricdocs main documentation, 2022) που είναι επίσημα διαθέσιμο, στην έκδοση 2.4 μέχρι την στιγμή που γράφεται η συγκεκριμένη μεταπτυχιακή διατριβή.

Το test network, αποτελείται από τα ελάχιστα στοιχεία (components), ώστε να λειτουργεί το blockchain.

Συγκεκριμένα :

- Δύο (2) peers ως Organization1 και Organization
- Ένα (1) ordering Organization

Καθώς το test network, δημιουργεί ένα ξεχωριστό docker bridge network (test network), προχωρήσαμε στην παραμετροποίηση του κώδικα που παρέχεται, ώστε κατά την εκκίνηση του blockchain network να χρησιμοποιεί (attached) το δικό όπως docker bridge network (HLFnetwork) (εικ.43).

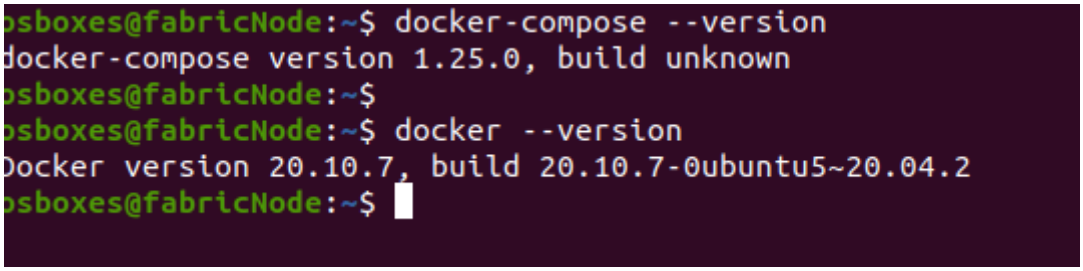
```
osboxes@fabricNode:~/fabric/fabric-samples/test-network/docker$ cat docker-compose-test-net.yaml
# Copyright IBM Corp. All Rights Reserved.
#
# SPDX-License-Identifier: Apache-2.0
#

version: '2.4'

volumes:
  orderer.example.com:
  peer0.org1.example.com:
  peer0.org2.example.com:

#networks:
#  test:
#    name: fabric_test
networks:
    default:
        external:
            name: HLFnetwork
services:
```

Εικόνα 43. Παραμετροποίηση test network

Για την εκκίνηση του blockchain, απαραίτητο είναι η εγκατάσταση του εργαλείου docker-compose (Overview of Docker Compose, 2022), όπως όπως και όπως γλώσσας προγραμματισμού GO

Το docker compose είναι ένα εργαλείο, το οποίο όπως επιτρέπει, μέσω αρχείων παραμέτρων (configuration files – YAML files), να τρέχουμε εύκολα και γρήγορα (με μία εντολή) ένα ή περισσότερα containers με όπως παραμέτρους που θέλουμε έτοιμες (χωρίς να χρειαστεί να συνδεθούμε στα containers για να ανεβάσουμε κάποιο service).

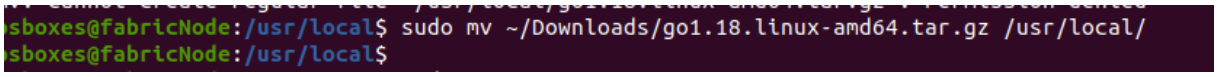
Με την εντολή docker-compose –version ελέγχουμε την έκδοση του εργαλείου (εικ.44).



Εικόνα 44. Docker compose version

Για την εγκατάσταση της GO, ακολουθούμε τις οδηγίες που περιγράφονται στο επίσημο έγγραφο εγκατάστασης (documentation) (Download and install - The Go Programming Language, 2022).

Στα παρακάτω στιγμιότυπα οθόνης (εικ.45-47) ακολουθεί η εγκατάσταση όπως γλώσσας προγραμματισμού, μέσω των binaries files.



Εικόνα 45. Binaries copy

Αφού μεταφέρουμε τα binaries στην συνέχεια προχωράμε στην εξαγωγή όπως (δημιουργείται όπως φάκελος με όνομα go) και στην συνέχεια θέτουμε όπως μεταβλητές του προγράμματος κάτω από το προφίλ όπως.

```
osboxes@fabricNode:/usr/local$ sudo tar -C /usr/local -xzf go1.18.linux-amd64.tar.gz
osboxes@fabricNode:/usr/local$ ls -lrt
total 138424
drwxr-xr-x  2 root    root         4096 Oct 17  2019 src
drwxr-xr-x  2 root    root         4096 Oct 17  2019 sbin
drwxr-xr-x  2 root    root         4096 Oct 17  2019 include
drwxr-xr-x  2 root    root         4096 Oct 17  2019 games
drwxr-xr-x  2 root    root         4096 Oct 17  2019 etc
lrwxrwxrwx  1 root    root            9 Oct 18  2019 man -> share/man
drwxr-xr-x  7 root    root         4096 Dec 16  2020 share
drwxr-xr-x  3 root    root         4096 Dec 16  2020 lib
drwxr-xr-x  2 root    root         4096 Apr  2  2021 bin
drwxr-xr-x 10 root    root         4096 Mar 15 16:08 go
-rw-rw-r--  1 osboxes osboxes 141702072 Mar 20 22:42 go1.18.linux-amd64.tar.gz
osboxes@fabricNode:/usr/local$ vi  $HOME/.profile
osboxes@fabricNode:/usr/local$ export PATH=$PATH:/usr/local/go/bin
```

Εικόνα 46. Go language setup

Επιβεβαιώνουμε την εγκατάσταση.

```
osboxes@fabricNode:/usr/local$ go version
go version go1.18 linux/amd64
osboxes@fabricNode:/usr/local$
```

Εικόνα 47. Go version

Στην συνέχεια προχωράμε στην εκκίνηση του Fabric sample network (εικ.48).

```
drwxrwxr-x 4 osboxes osboxes  4096 Apr 27  2021 organizations
osboxes@fabricNode:~/fabric/fabric-samples/test-network$ ./network.sh up
Starting nodes with CLI timeout of '5' tries and CLI delay of '3' seconds and using database 'leveldb' with crypto from 'cryptogen'
LOCAL_VERSION=2.3.1
DOCKER_IMAGE_VERSION=2.3.1
/home/osboxes/fabric/fabric-samples/test-network/../bin/cryptogen
Generating certificates using cryptogen tool
Creating Org1 Identities
+ cryptogen generate --config=./organizations/cryptogen/crypto-config-org1.yaml --output=organizations
org1.example.com
+ res=0
Creating Org2 Identities
+ cryptogen generate --config=./organizations/cryptogen/crypto-config-org2.yaml --output=organizations
org2.example.com
+ res=0
Creating Orderer Org Identities
+ cryptogen generate --config=./organizations/cryptogen/crypto-config-orderer.yaml --output=organizations
+ res=0
Generating CCP files for Org1 and Org2
Creating volume "docker_orderer.example.com" with default driver
Creating volume "docker_peer0.org1.example.com" with default driver
Creating volume "docker_peer0.org2.example.com" with default driver
Creating peer0.org2.example.com ... done
Creating orderer.example.com    ... done
Creating peer0.org1.example.com ... done
Creating cli                    ... done
CONTAINER ID   IMAGE                               COMMAND             CREATED          STATUS                  PORTS
          NAMES
7cc564b75011   hyperledger/fabric-tools:latest     "/bin/bash"         11 seconds ago   Up 2 seconds
          cli
8f6477606e9a   hyperledger/fabric-peer:latest      "peer node start"   24 seconds ago   Up 11 seconds           0.0.0.0:7051->7051/tcp, :::7051->7051/tcp
          peer0.org1.example.com
bc975977a2ff   hyperledger/fabric-orderer:latest   "orderer"           24 seconds ago   Up 11 seconds           0.0.0.0:7050->7050/tcp, :::7050->7050/tcp, 0.0.0.0:7053->7053/tcp, :::7053-
>7053/tcp   orderer.example.com
65c12338b03f   hyperledger/fabric-peer:latest      "peer node start"   24 seconds ago   Up 11 seconds           7051/tcp, 0.0.0.0:9051->9051/tcp, :::9051->9051/tcp
          peer0.org2.example.com
```

Εικόνα 48. HLF network start

Ελέγχουμε, ότι τα docker containers του Hyperledger Fabric network έχουν ανέβει σωστά (εικ.49).

```
sboxes@fabricNode:~$ docker ps
CONTAINER ID   IMAGE                               COMMAND             CREATED         STATUS
ES
7cc564b75011   hyperledger/fabric-tools:latest     "/bin/bash"         2 minutes ago   Up 2 minutes
3f6477606e9a   hyperledger/fabric-peer:latest      "peer node start"   2 minutes ago   Up 2 minutes
r0.org1.example.com
0c975977a2ff   hyperledger/fabric-orderer:latest   "orderer"           2 minutes ago   Up 2 minutes
erer.example.com
65c12338b03f   hyperledger/fabric-peer:latest      "peer node start"   2 minutes ago   Up 2 minutes
r0.org2.example.com
sboxes@fabricNode:~$
```

Εικόνα 49. HLF Docker containers

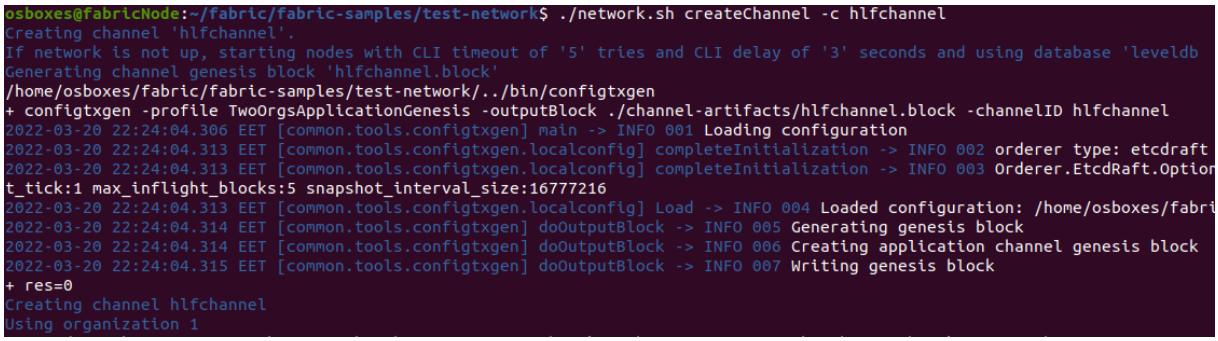
Με την εντολή docker network inspect HLFnetwork, ελέγχουμε αν έχουν αποδοθεί IP διευθύνσεις στα blockchain nodes κάτω από το υποδίκτυο HLFnetwork (εικ.50).

```
sboxes@fabricNode:~$ docker network inspect HLFnetwork
[
    {
        "Name": "HLFnetwork",
        "Id": "990239a429cff91310b69b27826136444ec0c2f52322ba36355dc3f6ae77560b",
        "Created": "2021-04-26T15:00:08.362851492+03:00",
        "Scope": "local",
        "Driver": "bridge",
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": {},
            "Config": [
                {
                    "Subnet": "172.28.3.0/24",
                    "Gateway": "172.28.3.254"
                }
            ]
        },
        "Internal": false,
        "Attachable": false,
        "Ingress": false,
        "ConfigFrom": {
            "Network": ""
        },
        "ConfigOnly": false,
        "Containers": {
            "65c12338b03f3617c8459fc5de0a693b1109681ec184ffbc4d554c5513feceb6": {
                "Name": "peer0.org2.example.com",
                "EndpointID": "06e27f28a03137295a670ead2ec1827f55c931bd0cd7a4e2a7cad8dbb50abeff",
                "MacAddress": "02:42:ac:1c:03:02",
                "IPv4Address": "172.28.3.2/24",
                "IPv6Address": ""
            },
            "7cc564b7501190ad946dbb9bd333d7a3c21ea41b463d1d34f0a5388572457adf": {
                "Name": "cli",
                "EndpointID": "3504d7ce1da623e6be45edf1c60717f692e06b6454cdb7e6572ea3296c6774d6",
                "MacAddress": "02:42:ac:1c:03:04",
                "IPv4Address": "172.28.3.4/24",
                "IPv6Address": ""
            },
```

Εικόνα 50. HLF network

Στην συνέχεια προχωρούμε στην δημιουργία του ιδιωτικού καναλιού επικοινωνίας των κόμβων του blockchain.

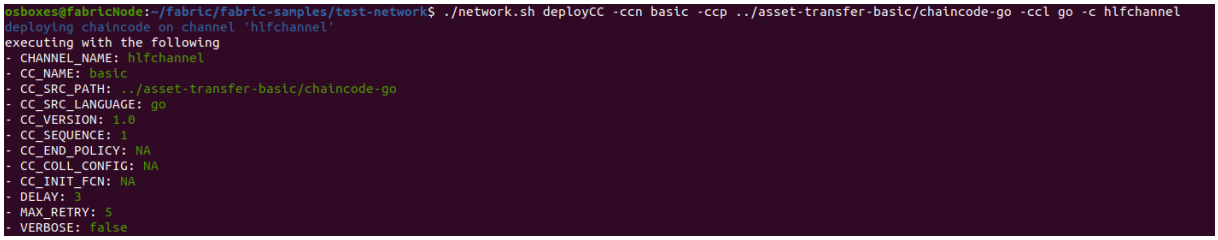
Με την εντολή, network.sh createChannel –c hlfchannel , δημιουργούμε ένα κανάλι με όνομα hlfchannel (εικ.51).



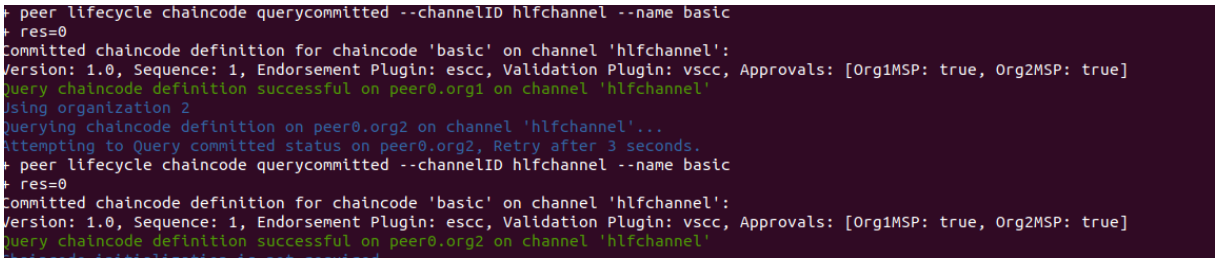
Εικόνα 51. Peer Channel creation

Εφόσον, η επικοινωνία έχει δημιουργηθεί μπορούμε να προχωρήσουμε στην ανάπτυξη του δικού όπως Smart Contract ή όπως αναφέρονται στο Fabric blockchain ως Chaincode.

Για όπως ανάγκες όπως υλοποίησης όπως θα χρησιμοποιήσουμε ένα βασικό Chaincode που περιγράφεται ως Asset-Transfer (εικ.52-53).

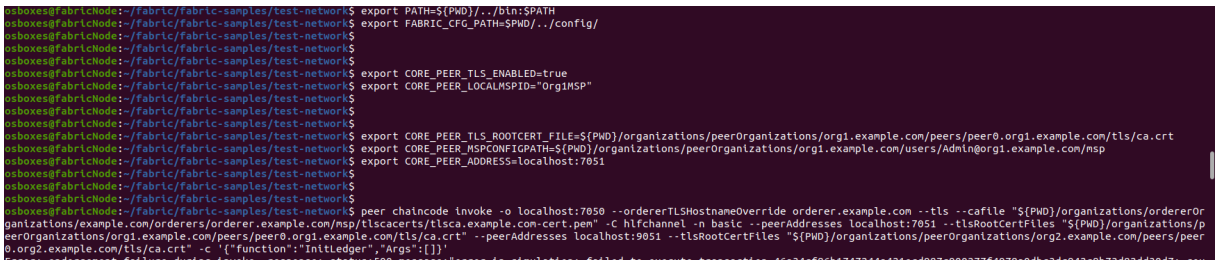


Εικόνα 52. Deploy Chaincode



Εικόνα 53. Chaincode deployment

Με την δημιουργία και του Chaincode, μπορούμε να ξεκινήσουμε όπως συναλλαγές (transactions) μεταξύ των peer nodes (εικ.54).



Εικόνα 54. Peer node transactions

Αφού έχουν δημιουργηθεί οι συναλλαγές στο Chaincode, στην συνέχεια μπορούμε να τρέξουμε ερωτήματα (queries) στον ledger, ώστε να δούμε τις συναλλαγές που έχουν δημιουργηθεί (εικ.55).

Εικόνα 55. Query Ledger

Με την δημιουργία των συναλλαγών στο chaincode –Smart Contract- ολοκληρώσαμε το στάδιο όπως υλοποίησης, που αφορά το περιβάλλον εργασίας που μελετάμε στην μεταπτυχιακή διατριβή μας.

Στο επόμενο κεφάλαιο περιγράφεται ο μηχανισμός AMRA (Advanced Monitoring and Response Application).

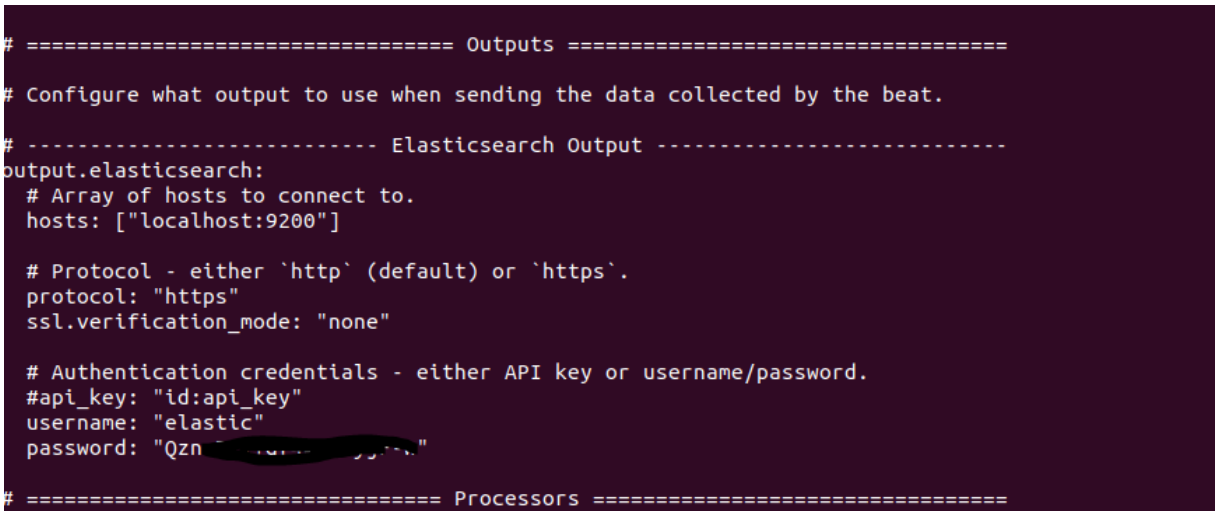
Κεφάλαιο 5
AMRA Framework

Το Advanced Monitoring and Response Application framework, αποτελείται από δύο (2) modules , τα α) Advanced Monitoring και β) Response Application.

Στην συνέχεια περιγράφεται η υλοποίηση των παραπάνω modules.

5.1 Advanced Monitoring

Κατά την εγκατάσταση των network flow agents (packetbeat) προχωρούμε στην απαραίτητη παραμετροποίηση ώστε να επικοινωνεί με το elasticsearch engine (εικ.56 - 57).



Εικόνα 56. Παραμετροποίηση agent

```
root@controller:/etc/packetbeat# packetbeat test output
elasticsearch: https://localhost:9200...
  parse url... OK
  connection...
    parse host... OK
    dns lookup... OK
    addresses: 127.0.0.1
    dial up... OK
  TLS...
    security... WARN server's certificate chain verification is disabled
    handshake... OK
    TLS version: TLSv1.3
    dial up... OK
  talk to server... OK
  version: 8.1.0
root@controller:/etc/packetbeat# packetbeat --help
```

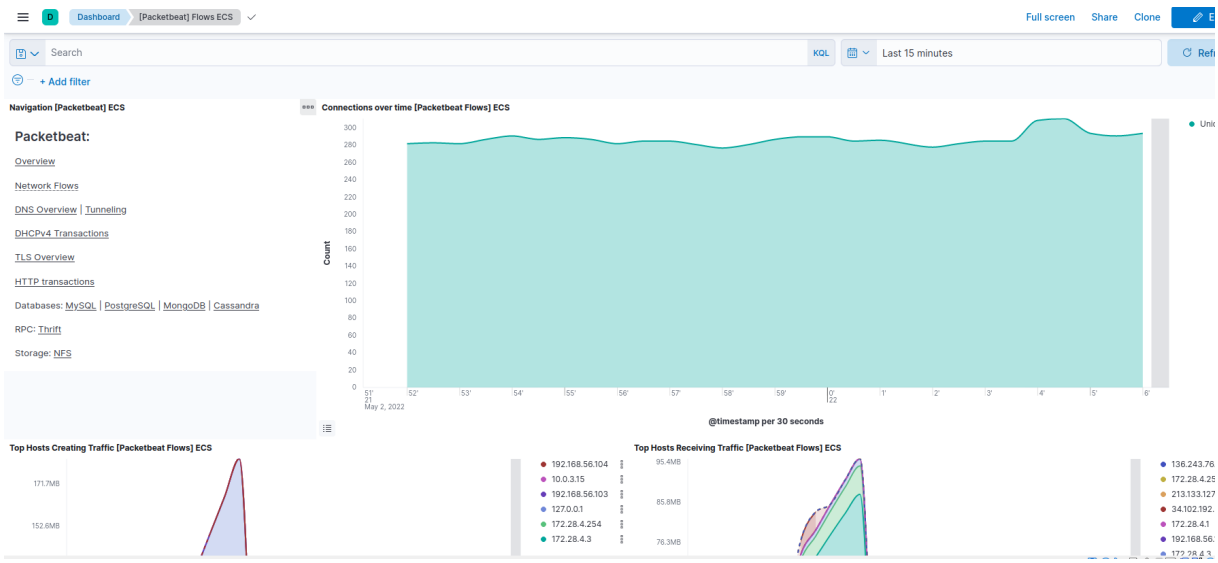
Εικόνα 57. Έλεγχος παραμετροποίησης

Έχουμε την δυνατότητα να φορτώσουμε στο Kibana Web interface, έτοιμο πίνακα γραφικών (dashboard) το όποιο υπάρχει στο πακέτο του packetbeat (εικ.58).

```
exit
osboxes@controller:/etc/packetbeat$ packetbeat setup --dashboards
```

Εικόνα 58. Packetbeat dashboards

Με την ολοκλήρωση της ανάλογης παραμετροποίησης και στα υπόλοιπα Virtual Machines, ελέγχουμε αν στο User Interface του Kibana τα διαθέσιμα δεδομένα που στέλνονται από τους agents (εικ.59).



Εικόνα 59. Packetbeat network flows

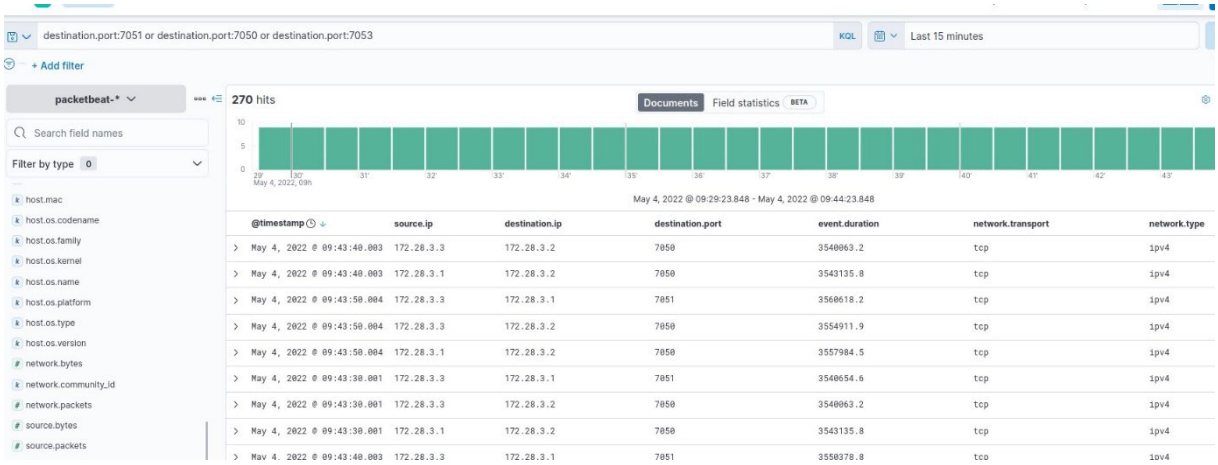
Στόχος του Advanced Monitoring module, είναι η συνεχής παρακολούθηση των πακέτων (network flows) που ανταλλάσσονται μεταξύ του blockchain network με την υπόλοιπη υποδομή.

Για να επιτευχθεί ο παραπάνω στόχος, απαραίτητο είναι να φιλτράρουμε την πληροφορία που αποθηκεύεται στο Elasticsearch, δημιουργώντας τα κατάλληλα ερωτήματα μέσω του Kibana Query Language.

Συγκεκριμένα, όπως φαίνεται και στην παρακάτω εικόνα (εικ.60) μας ενδιαφέρουν τα παρακάτω πεδία (fields).

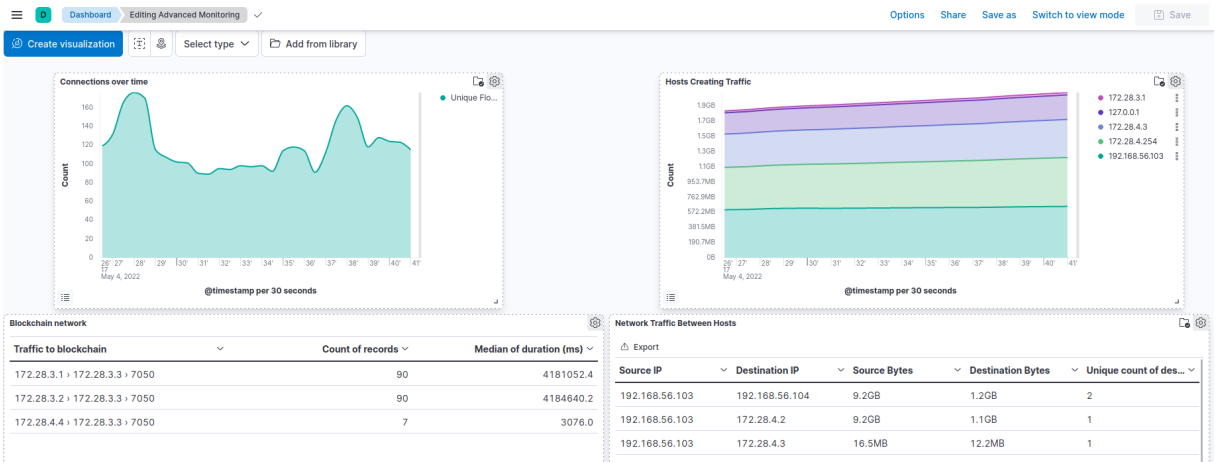
- Timestamp
- Source IP
- Destination IP
- Destination Port
- Packets sent
- Event duration
- Network transport protocol
- Network type

Τα δεδομένα που έχουμε φιλτράρονται ώστε να περιλαμβάνουν μόνο τις πόρτες (destination ports) που αφορούν τα blockchain nodes.



Εικόνα 60. Network flows rule

Στις παρακάτω εικόνες (εικ.61-62), αποτυπώνεται γραφικά (Advanced Monitoring dashboard) συνολικά η κίνηση (network flow) του δικτύου μας.

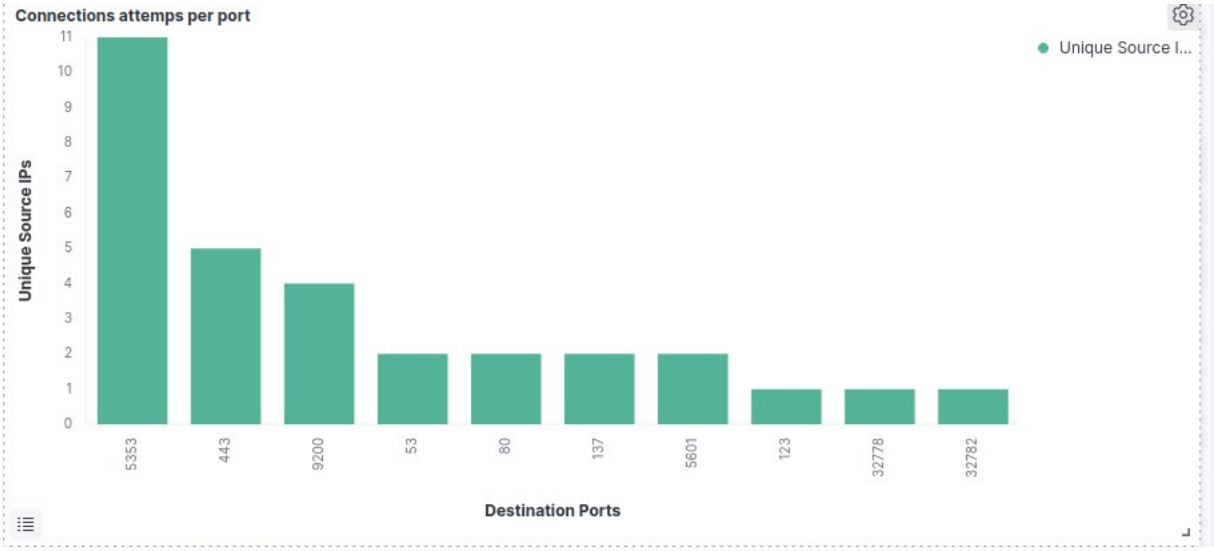


Εικόνα 61. Advanced Monitoring dashboard

Εικόνα 62. Advanced Monitoring

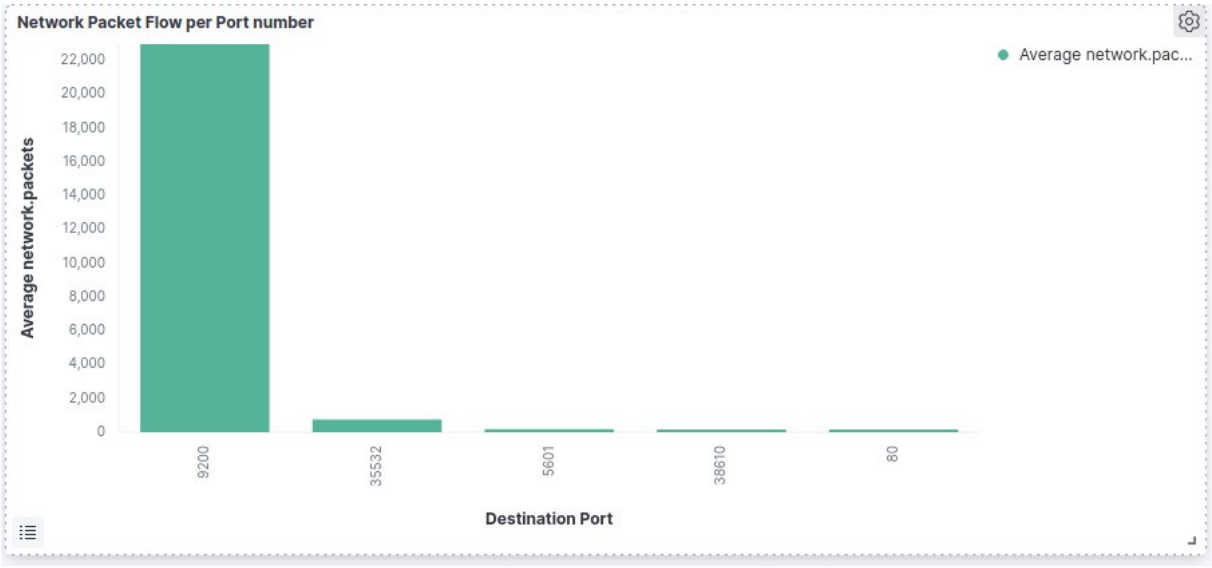
Όπως φαίνεται στην εικ.61, στο γράφημα που αφορά την κίνηση προς το blockchain, έχει γίνει απόπειρα σύνδεσης από κόμβο που δεν ανήκει στο υποδίκτυο του blockchain.

Επιπλέον, με βάση το παρακάτω διάγραμμα (εικ.63), μπορούμε να εντοπίσουμε ενδεχόμενη επίθεση τύπου port scanning, αφού καταγράφεται ο συνολικός αριθμός προσπαθειών προς τις IP πόρτες των αποδεκτών (destination port numbers).

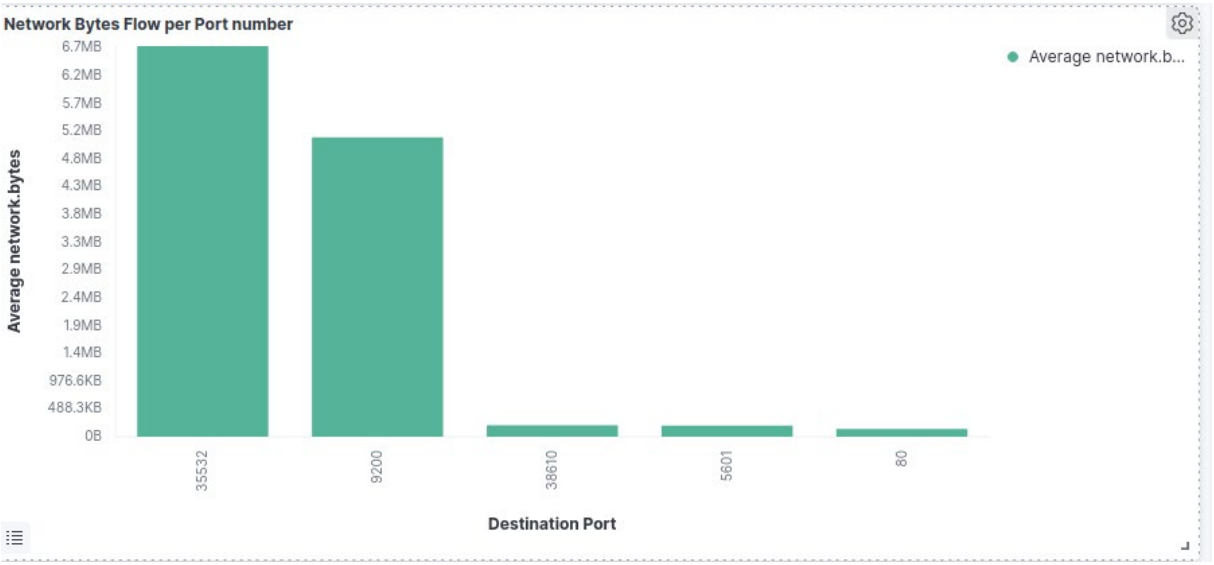


Εικόνα 63. Connections attempts per port

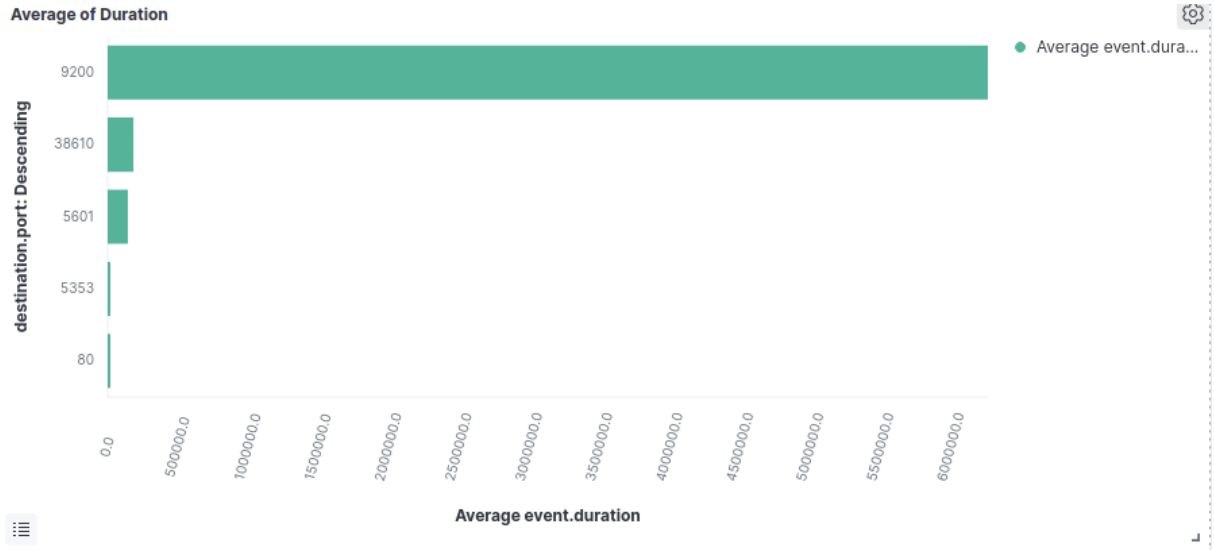
Επόμενα, χρήσιμα διαγράμματα που περιλαμβάνει ο μηχανισμός μας, έχουν να κάνουν με την μέση κίνηση πακέτων (εικ.64), μέσο μέγεθος πακέτου (εικ.65) και διάρκεια του session (εικ.66).



Εικόνα 64. Average of Packets per Flow



Εικόνα 65. Average of Bytes per Flow



Εικόνα 66. Average of Duration

Για τον έλεγχο της λειτουργίας του μηχανισμού, πραγματοποιούμε επίθεση τύπου DoS attack, στέλνοντας πολλαπλά πακέτα connection requests κατά του Ordering service node, το οποίο είναι υπεύθυνο για την συλλογή των συναλλαγών και στην συνέχεια την δημιουργία των blocks (Dabholkar & Saraswat, 2019).

Θα χρησιμοποιήσουμε το ανοιχτού κώδικα εργαλείο (network tool) Hping3, το οποίο μας επιτρέπει την δημιουργία προσαρμοσμένων (customized) πακέτων ώστε να μπορέσουμε να ελέγξουμε την λειτουργία και την επίδοση ενός δικτύου (hping3 | Kali Linux Tools, 2022) .

Δημιουργούμε ένα νέο Linux Ubuntu container κάτω από το ELKnetwork δίκτυο και εγκαθιστούμε το εργαλείο (εικ.67 – 68).

```
osboxes@controller:~$
osboxes@controller:~$ docker run --network ELKnetwork -d -ti ubuntu sh
70838d82a60688ff89582e8b959d7e5544a65506bfbc4a717033a9f82cc5d966
osboxes@controller:~$ docker exec -ti 70838d82a606 sh
#
#
```

Εικόνα 67. Εγκατάσταση νέου container

```
#  apt-get install hping3
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  dbus libapparmor1 libdbus-1-3 libexpat1 libpcap0.8 libtcl8.6 tzdata
Suggested packages:
  default-dbus-session-bus | dbus-session-bus tcl8.6
The following NEW packages will be installed:
  dbus hping3 libapparmor1 libdbus-1-3 libexpat1 libpcap0.8 libtcl8.6 tzdata
0 upgraded, 8 newly installed, 0 to remove and 3 not upgraded.
Need to get 2058 kB of archives.
After this operation, 10.4 MB of additional disk space will be used.
Do you want to continue? [Y/n] Y
```

Εικόνα 68. Εγκατάσταση Hping3 tool

Για την πραγματοποίηση της επίθεσης SYN flood attack, χρησιμοποιούμε την εντολή

#hping3 –c 20000 –d 300 –S –w 64 –p 7050 –flood –rand-source 172.28.3.2

Με την παράμετρο “c” στέλνουμε 20000 πακέτα, με μέγεθος 300 Bytes και windows size 64. Επιπλέον, χρησιμοποιούμε τις παραμέτρους (flags) “flood” με την οποία δηλώνουμε να στέλνουμε τα πακέτα το συντομότερο στην πόρτα “-p”, “-S” για SYN packets και το “—rand-source” ώστε να παράγονται spoofed IP source addresses.

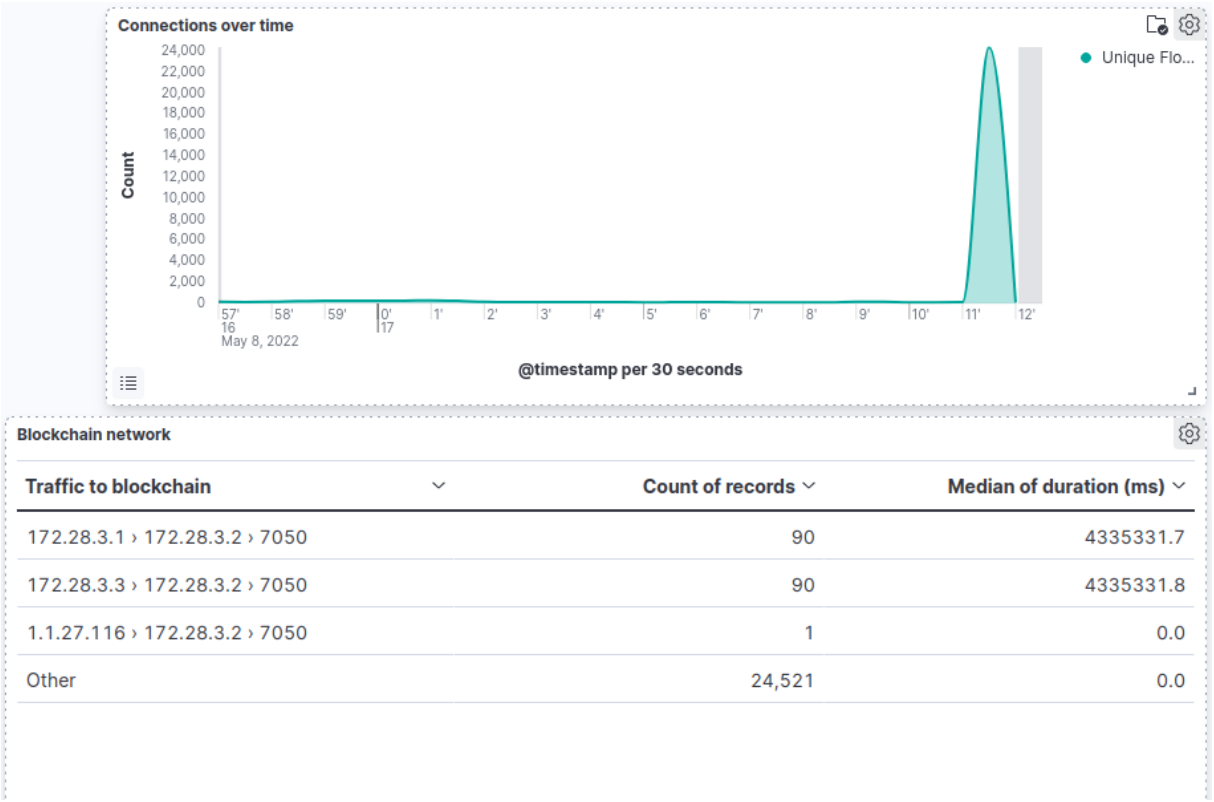
Στην επόμενη εικόνα φαίνεται η εκτέλεση της εντολής (εικ.69).

```
# hping3 -c 20000 -d 200 -S -w 64 -p 7050 --flood --rand-source 172.28.3.2
HPING 172.28.3.2 (eth0 172.28.3.2): S set, 40 headers + 200 data bytes
hping in flood mode, no replies will be shown
```

Εικόνα 69. SYN Flood attack

Όπως φαίνεται στα διαγράμματα που έχουμε αναπτύξει, ο εντοπισμός της επίθεσης καταγράφεται επιτυχώς (εικ.70).

Στο διάγραμμα παρατηρούμε μια μεγάλη καμπύλη (burst) όσο αφορά τις προσπάθειες σύνδεσης μέσα σε ένα πολύ μικρό χρονικό παράθυρο t < 1 sec, ενώ στον πίνακα που καταγράφει την κίνηση προς το blockchain, παρατηρούμε να έχουν πραγματοποιηθεί περισσότερες από 24521 συνδέσεις.



Blockchain network

Traffic to blockchain	Count of records	Median of duration (ms)
172.28.3.1 › 172.28.3.2 › 7050	90	4335331.7
172.28.3.3 › 172.28.3.2 › 7050	90	4335331.8
1.1.27.116 › 172.28.3.2 › 7050	1	0.0
Other	24,521	0.0

Εικόνα 70. AMRA attack detection

Διαπιστώνουμε, πως για την παραπάνω κίνηση που αναφέρετε ως other, προκύπτει λόγω του γεγονότος ότι χρησιμοποιούμε spoofed source ip addresses.

Μπορούμε να πραγματοποιήσουμε μεγαλύτερη ανάλυση της κίνησης αυτής, μέσω του Kibana Discover board, όπως φαίνεται στην παρακάτω εικόνα (εικ.71).

	source.ip	destination.ip	destination.por
@ 17:16:04.307	154.253.63.97	172.28.3.2	7050
@ 17:16:04	179.217.221.47	172.28.3.2	7050
@ 17:16:04.307	48.107.171.200	172.28.3.2	7050
@ 17:16:04.307	63.97.48.63	172.28.3.2	7050
@ 17:16:04.307	85.20.179.245	172.28.3.2	7050
@ 17:16:04.307	107.226.244.246	172.28.3.2	7050
@ 17:16:04.307	207.105.215.39	172.28.3.2	7050
@ 17:16:04.307	245.196.120.63	172.28.3.2	7050
@ 17:16:04.307	185.215.66.148	172.28.3.2	7050
@ 17:16:04.307	179.102.130.209	172.28.3.2	7050

Εικόνα 71. Spoofed IP addresses

Με βάση τα παραπάνω διαπιστώνουμε πως το module Advanced Monitoring που έχουμε υλοποιήσει, είναι σε θέση να καταγράφει την κίνηση που διέρχεται στο δίκτυο μας, όπως επίσης να εντοπίζουμε άμεσα οποιαδήποτε κίνηση η οποία δεν είναι νόμιμη.

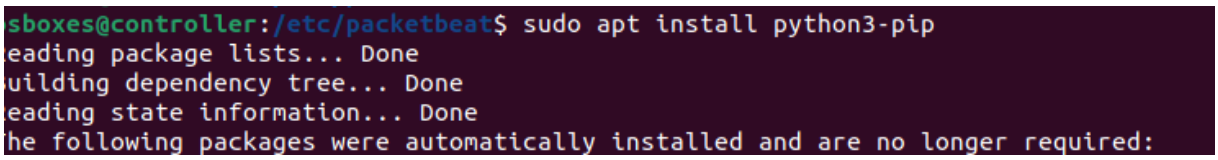
5.2 Response Application

Το δεύτερο module του AMRA framework, αναλαμβάνει την εκτέλεση ερωτημάτων στην elasticsearch database και στην συνέχεια με βάση των συνθηκών που έχουμε ορίσει (connection rate), αντιμετωπίζουμε αυτές τις προσπάθειες σύνδεσης (number of connections established) ως επιθέσεις άρνησης παροχής υπηρεσιών (DDoS attacks) έναντι της υποδομής μας που φιλοξενεί το permissioned blockchain.

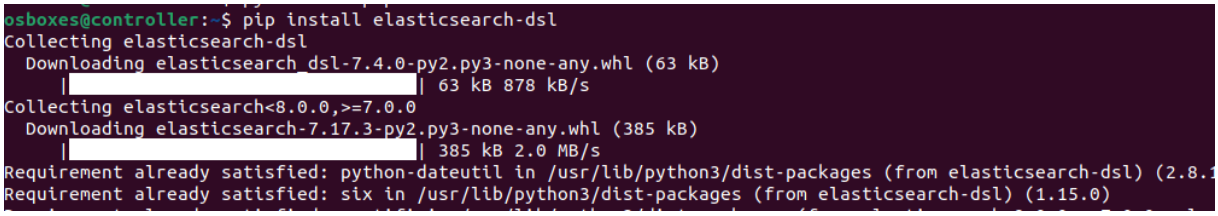
Για την ανάπτυξη του κώδικα, χρησιμοποιούμε την γλώσσα προγραμματισμού python (Python.org, 2022)στην έκδοση 8.3. Ως γλώσσα προγραμματισμού, διακρίνεται για την επεκτασιμότητα της μέσω της δημιουργίας modules και έχει επιτρέψει σήμερα τους προγραμματιστές να δημιουργούν πληθώρα εφαρμογών σε όλους τους τομείς της τεχνολογίας της πληροφορικής.

Χρησιμοποιούμε τα επίσημα python modules (libraries) elasticsearch (Overview | Elasticsearch Python Client [8.1] | Elastic, 2022) και elasticsearch_dsl (elasticsearch-dsl: Python client for Elasticsearch, 2022).

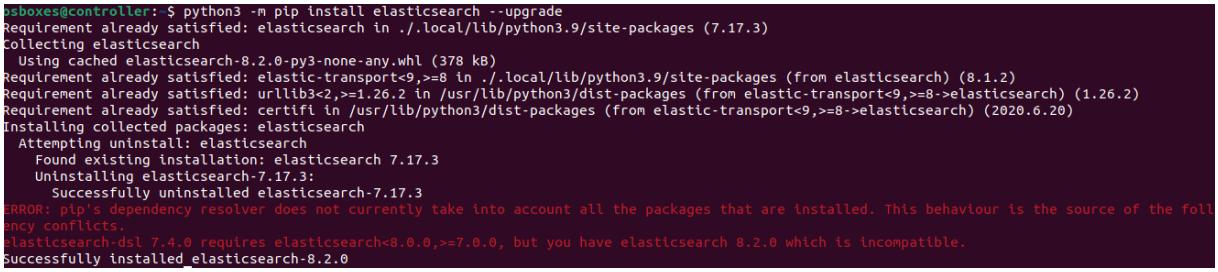
Στις παρακάτω εικόνες (εικ.72-74), παρουσιάζεται η εγκατάσταση του module, χρησιμοποιώντας το επίσημο εργαλείο εγκατάστασης python πακέτων (packages/modules).



Εικόνα 72. Εγκατάσταση PIP package



Εικόνα 73. Elasticsearch-dsl module



Εικόνα 74. Elasticsearch module upgrade

Ελέγχουμε ότι μπορούμε να επικοινωνήσουμε μέσω του Application Programming Interface (API), με το παρακάτω python script.

```
from elasticsearch import Elasticsearch

from elasticsearch_dsl import Search

ES_PASSWD = "NYw*U1h-AotfV3RmjO1T"

client = Elasticsearch("https://localhost:9200",

                ca_certs="/home/osboxes/http_ca.crt",

                basic_auth=("elastic", ES_PASSWD)

                )

client.info()
```

όπως φαίνεται και στην παρακάτω εικόνα (εικ.75), η επικοινωνία ήταν επιτυχής καθώς λάβαμε την απάντηση που περιμέναμε.

```
>>> client.info()
ObjectApiResponse({'name': 'ff96921882fa', 'cluster_name': 'docker-cluster', 'cluster_uuid': '0A0-ewL6T8SqBEQYJpLrXw', 'version': {'number': '8.2.0', 'build_fl
avor': 'default', 'build_type': 'docker', 'build_hash': 'b174af62e8dd9f4ac4d25875e9381ffe2b9282c5', 'build_date': '2022-04-20T10:35:10.180408517Z', 'build_snap
shot': False, 'lucene_version': '9.1.0', 'minimum_wire_compatibility_version': '7.17.0', 'minimum_index_compatibility_version': '7.0.0'}, 'tagline': 'You Know,
 for Search'})
>>>
```

Εικόνα 75. Elasticsearch info

Χρησιμοποιώντας το module elasticsearch-dsl και τις συναρτήσεις (functions) που προσφέρει μπορούμε να δημιουργήσουμε τα κατάλληλα ερωτήματα (queries) ώστε στην συνέχεια μέσω του REST interface του POX controller να αντιμετωπίσουμε τις επιθέσεις του τύπου που είδαμε.

Κεφάλαιο 6
Συμπεράσματα

Η είσοδος νέων τεχνολογιών όπως είναι τα containers και τα permissioned blockchains σταδιακά υιοθετούνται από πολλούς κλάδους επιχειρήσεων αλλά και της κοινωνίας (Lucas , 2021) καθώς προσφέρουν μεγάλη διαθεσιμότητα (availability) των υπηρεσιών που προσφέρει κάθε κλάδος, όπως επίσης έχουν την δυνατότητα να εξασφαλίσουν την εμπιστευτικότητα (confidentiality) και ακεραιότητα (integrity) των δεδομένων που επεξεργάζονται. Παρόλα αυτά η ασφάλεια της υποδομής ως κρίσιμων αγαθών δεν είναι δεδομένη έναντι διάφορων απειλών που σκοπό έχουν να προκαλέσουν βλάβη στα παραπάνω αγαθά.

Με την παρούσα μεταπτυχιακή διατριβή γίνεται μια πρώτη πρόταση το πως θα μπορούσαμε να διασφαλίσουμε την διαθεσιμότητα των παραπάνω υποδομών, χρησιμοποιώντας την τεχνολογία δικτύου SDN: Software Define Network, έναντι επιθέσεων τύπου κατανεμημένης άρνησης παροχής υπηρεσιών DDoS (Distributed Denial of Service).

Παρουσιάστηκαν οι τεχνικές και προτάσεις που έχουν υλοποιηθεί από πλήθος ερευνητών σχετικά με τις τεχνικές εντοπισμού και περιορισμού του κινδύνου (attack detection and mitigation technics), όπως επίσης αναγνωρίστηκαν και οι μελλοντικές προκλήσεις.

Η πρόταση μας, AMRA framework, κάνει χρήση αξιόπιστων εργαλείων που είναι ήδη διαθέσιμα, προσαρμόζοντας τα στο περιβάλλον εργασίας που μελετάμε. Όπως διαπιστώσαμε η λύση που αναπτύξαμε είναι σε θέση να εντοπίζει πιθανές επιθέσεις με επιτυχία. Επιπλέον, δίνει την δυνατότητα της γραφικής απεικονίσεις των επιθέσεων αυτών κάτι που μπορεί να είναι χρήσιμο για τους μηχανικούς ασφαλείας της πρώτης γραμμής (SOC engineers).

Σκοπός του δεύτερου module «Response Application», αξιοποιώντας την δυνατότητα του προγραμματισμού που επιτρέπει ο controller σε ένα SDN based network, να προβαίνουμε στις απαραίτητες ενέργειες ώστε να περιορίσουμε τις επιθέσεις τύπου DDoS αναλύοντας (parsing) τα διαθέσιμα δεδομένα που αποθηκεύονται στην Elasticsearch engine.

Εξωγενείς παράγοντες με κυριότερο την έλλειψη χρόνου και μη διαθεσιμότητα πόρων που προέκυψαν κατά την διάρκεια της έρευνας οδήγησαν στην μη ολοκλήρωση του δεύτερου module.

Ως μελλοντικό βήμα για την έρευνα μας, θα μπορούσε να ολοκληρωθεί ο μηχανισμός που αφορά τον περιορισμό του κινδύνου (mitigation attack). Επιπλέον, η μεταφορά του περιβάλλον εργασίας στο cyber range θα βοηθούσε στην ταχύτερη ανάπτυξη του κώδικα και ενδεχομένως θα ήταν πιο κοντά σε ένα παραγωγικό περιβάλλον.

Βιβλιογραφία

Agg, P., & Johanyák, Z. (2017). Pyretic SDN Programming Language. *ResearchGate*.

Azab, M., & Fortes, J. (2017). Towards proactive SDN-controller attack and failure resilience. *2017 International Conference on Computing, Networking and Communications (ICNC)* (σσ. 442-448). IEEE Xplore.

Bawany, N., Shamsi, J., & Salah, K. (2017). DDoS Attack Detection and Mitigation Using SDN: Methods, Practices, and Solutions. *Arabian Journal for Science and Engineering*, 425-441.

Binary decision diagram| wikipedia. (2022). Ανάκτηση από https://en.wikipedia.org/wiki/Binary_decision_diagram

Busybox - Official Image | Docker Hub. (n.d.). Ανάκτηση από Docker Hub: https://hub.docker.com/_/busybox

Byzantine fault. (2022). Ανάκτηση από Wikipedia: https://en.wikipedia.org/w/index.php?title=Byzantine_fault&oldid=1069251551

Canner, B. (2021). *The 10 Best Open Source SIEM Tools for Businesses*. Ανάκτηση από Best Information Security SIEM Tools, Software, Solutions & Vendors: https://solutionsreview.com/security-information-event-management/the-10-best-open-source-siem-tools-for-businesses/

Craven, C. (2020). *What is NFV (Network Functions Virtualization)? Definition*. Ανάκτηση από SDxCentral: https://www.sdxcentral.com/networking/nfv/definitions/whats-network-functions-virtualization-nfv/

CRIU project. (2022). Ανάκτηση από https://www.criu.org: https://www.criu.org/Main_Page

Cyber Range. (2018). Ανάκτηση από https://www.nist.gov: https://www.nist.gov/system/files/documents/2018/02/13/cyber_ranges.pdf

Dabholkar, A., & Saraswat, V. (2019). Ripping the Fabric: Attacks and Mitigations on Hyperledger Fabric. *Communications in Computer and Information Science* (σσ. 300-311). Springer Link.

Dacier, M., König, H., Cwalinski, R., Kargl, F., & Dietrich, S. (2017). Security Challenges and Opportunities of Software-Defined Networking. *IEEE Security Privacy*, 96-100.

Dockerfile reference. (2022). Ανάκτηση από Docker Documentation: https://docs.docker.com/engine/reference/builder/

Download and install - The Go Programming Language. (2022). Ανάκτηση από https://go.dev/doc/install

Eddy, W. (2007). *TCP SYN Flooding Attacks and Common Mitigations.* Ανάκτηση από https://datatracker.ietf.org/doc/html/rfc4987

elasticsearch-dsl: Python client for Elasticsearch. (2022). Ανάκτηση από elasticsearch-dsl: https://github.com/elasticsearch/elasticsearch-dsl-py

Faizullah, S., Khan, M., Alzahrani, A., & Khan, I. (2020). Permissioned Blockchain-Based Security for SDN in IoT Cloud Networks. *arXiv:2002.00456 [cs]*.

Ford-Fulkerson Algorithm for Maximum Flow Problem. (2013). Ανάκτηση από GeeksforGeeks: https://www.geeksforgeeks.org/ford-fulkerson-algorithm-for-maximum-flow-problem/

Gedia, D., & Perigo, L. (2018). Performance Evaluation of SDN-VNF in Virtual Machine and Container. *2018 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)* (σσ. 1-7). IEEE Xplore.

hping3 | Kali Linux Tools. (2022). Ανάκτηση από Kali Linux: https://www.kali.org/tools/hping3/

Hussein, A., Elhajj, I., Chehab, A., & Kayssi, A. (2016). SDN Security Plane: An Architecture for Resilient Security Services. *2016 IEEE International Conference on Cloud Engineering Workshop (IC2EW)* (σσ. 54-59). IEEE Xplore.

Install Elasticsearch with Docker | Elasticsearch Guide [8.1] | Elastic. (2022). Ανάκτηση από Learn/Docs/Elasticsearch/Reference/8.1: https://www.elastic.co/guide/en/elasticsearch/reference/current/docker.html

Install Kibana with Docker | Kibana Guide [8.1] | Elastic. (2022). Ανάκτηση από Learn/Docs/Kibana/Reference/8.1: https://www.elastic.co/guide/en/kibana/current/docker.html

Kibana: Explore, Visualize, Discover Data. (2022). Ανάκτηση από Elastic: https://www.elastic.co/kibana

Lim, S., Ha, J., Kim, H., Kim, Y., & Yang, S. (2014). A SDN-oriented DDoS blocking scheme for botnet-based attacks. *2014 Sixth International Conference on Ubiquitous and Future Networks (ICUFN)* (σσ. 63-68). IEEE.

Load balancing (computing). (2022). Ανάκτηση από Wikipedia: https://en.wikipedia.org/w/index.php?title=Load_balancing_(computing)&oldid=1073888114

Lucas , S. (2021). *81 of the Top 100 Public Companies are using blockchain technology*. Ανάκτηση από Blockdata: https://www.blockdata.tech/blog/general/81-of-the-top-100-public-companies-are-using-blockchain-technology

Machidon, O., Mladin, A., Sandu, F., & Bocu, R. (2014). Software-defined networking of Linux containers. *2014 RoEduNet Conference 13th Edition: Networking in Education and Research Joint Event RENAM 8th Conference* (σσ. 1-4). IEEE Xplore.

Open Network Operating System (ONOS) SDN Controller for SDN/NFV Solutions. (2022). Ανάκτηση από Open Networking Foundation: https://opennetworking.org/onos/

Open vSwitch 2.5.0 Documentation. (n.d.). Ανάκτηση από Open vSwitch: https://www.openvswitch.org/support/dist-docs-2.5/

OpenNebula. (2022). Ανάκτηση από https://opennebula.io/

Overview | Elasticsearch Python Client [8.1] | Elastic. (2022). Ανάκτηση από https://www.elastic.co/guide/en/elasticsearch/client/python-api/8.1/overview.html

Overview of Docker Compose. (2022). Ανάκτηση από Docker Documentation: https://docs.docker.com/compose/

Packetbeat: Network Analytics Using Elasticsearch. (n.d.). Ανάκτηση από Elastic: https://www.elastic.co/beats/packetbeat

Point of Presence (POP). (2019). Ανάκτηση από Network Encyclopedia: https://networkencyclopedia.com/point-of-presence-pop/

Python.org. (2022). Ανάκτηση από https://www.python.org/

Saldamli, G., Sanjeeva, H., Siddalingaapa, K., Murugesan, R., & Ertaul, L. (2019). A Secure Collaborative Module on Distributed SDN. *2019 10th International Conference on Information and Communication Systems (ICICS)* (σσ. 65-70). IEEE Xplore.

Scott-Hayward, S., O'Callaghan, G., & Sezer, S. (2013). Sdn Security: A Survey. *2013 IEEE SDN for Future Networks and Services (SDN4FNS)* (σσ. 1-7). IEEE Xplore.

Self-Organizing Map - an overview | ScienceDirect Topics. (2013). Ανάκτηση από Sciencedirect: https://www.sciencedirect.com/topics/engineering/self-organizing-map

Shao, Z., Zhu, X., Chikuvanyanga, A., & Zhu, H. (2019). Blockchain-Based SDN Security Guaranteeing Algorithm and Analysis Model. *Wireless and Satellite Systems* (σσ. 348-362). Springer Link.

Shin, S., Porras, P., Yegneswara, V., Fong, M., Gu, G., & Tyson, M. (2013). FRESCO: Modular Composable Security Services for Software-Defined Networks. *20th Annual Network & Distributed System Security Symposium*.

Shin, S., Xu, L., Hong, S., & Gu, G. (2016). Enhancing Network Security through Software Defined Networking (SDN). *2016 25th International Conference on Computer Communication and Networks (ICCCN)* (σσ. 1-9). IEEE Xplore.

Shukhman, A., Polezhaev, P., Ushakov, Y., Legashev, L., Tarasov, V., & Bakhareva, N. (2015). Development of network security tools for enterprise software-defined networks. *Proceedings of the 8th International Conference on Security of Information and Networks* (σσ. 224-228). ACM Digital Library.

Single Point of Failure - an overview | ScienceDirect Topics. (2016). Ανάκτηση από https://www.sciencedirect.com/topics/computer-science/single-point-of-failure

Smeliansky, R. (2014). SDN for network security. *2014 International Science and Technology Conference (Modern Networking Technologies) (MoNeTeC)* (σσ. 1-5). IEEE Xplore.

Steichen, M., Hommes, S., & State, R. (2017). ChainGuard — A firewall for blockchain applications using SDN with OpenFlow. *2017 Principles, Systems and Applications of IP Telecommunications (IPTComm)* (σσ. 1-8). IEEE Xplore.

Top Online Cybersecurity Trends for 2020. (2020). Ανάκτηση από Cybersecurity Blog: https://www.cyberranges.com/cybersecurity-trends-for-2020/

Using the Fabric test network — hyperledger-fabricdocs main documentation. (2022). Ανάκτηση από hypeledger: https://hyperledger-fabric.readthedocs.io/en/latest/test_network.html

What is a Container? (2021). Ανάκτηση από https://www.docker.com/resources/what-container/

What is OpenFlow?Definition and How it Relates to SDN. (2013, August). Ανάκτηση από https://www.sdxcentral.com/: https://www.sdxcentral.com/networking/sdn/definitions/what-is-openflow/

Xu, Y., & Liu, Y. (2016). DDoS attack detection under SDN context. *IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications* (σσ. 1-9). IEEE Xplore.

Zhang, R., Xue, R., & Liu, L. (2019). Security and Privacy on Blockchain. *ACM Computing Surveys*, 51:1-51:34.

Zombie (computing). (2022). Ανάκτηση από Wikipedia: https://en.wikipedia.org/w/index.php?title=Zombie_(computing)&oldid=1072785237