ΔΙΕΘΝΕΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΤΗΣ ΕΛΛΑΔΟΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

«Ασφάλεια σε Δίκτυα Καθορισμένα από Λογισμικό και υλοποίηση σεναρίων επίθεσης»

Του φοιτητή
Κακούσιη Γεώργιου
Αρ. Μητρώου: IT174906

Επιβλέπων
Δρ. Αντώνης Σαρηγιαννίδης

Ημερομηνία 06/09/2024

Ασφάλεια σε Δίκτυα Καθορισμένα από Λογισμικό και υλοποίηση σεναρίων επίθεσης
Κωδικός Δ.Ε. 24107
Γεώργιος Κακούσιη
Δρ. Αντώνης Σαρηγιαννίδης
Ημερομηνία ανάληψης Δ.Ε. 18 Ιανουαρίου 2024
Ημερομηνία περάτωσης Δ.Ε. 06 Σεπτεμβρίου 2024

Βεβαιώνω ότι είμαι ο συγγραφέας αυτής της εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, έχω καταγράψει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών, εικόνων και κειμένου, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επιπλέον, βεβαιώνω ότι αυτή η εργασία προετοιμάστηκε από εμένα προσωπικά, ειδικά ως διπλωματική εργασία, στο Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του ΔΙ.ΠΑ.Ε.

Πρόλογος

Επέλεξα το θέμα «Ασφάλεια σε Δίκτυα Καθορισμένα από Λογισμικό και Υλοποίηση Σεναρίων Επίθεσης» στα πλαίσια εκπόνησης της πτυχιακής μου εργασίας καθώς πιστεύω ότι θα με βοηθήσει στην εξέλιξη μου ως Μηχανικός Πληροφορικής και Μηχανικός Δικτύων. Η μελέτη και η συγγραφή της εργασίας με βοήθησε να κατανοήσω και να μάθω περισσότερα για την ασφάλεια στο κομμάτι της δικτύωσης και συγκεκριμένα στα δίκτυα SDN που αποτελούν το μέλλον της δικτύωσης. Επιπλέον η εκτέλεση των προσομοιώσεων με βοήθησε στο να κατανοήσω το τεχνικό κομμάτι στη λειτουργία και την υλοποίηση εφαρμογών που εκτελούνται στους ελεγκτές SDN και τη σημαντικότητα της ασφάλειας για την προστασία του δικτύου και των χρηστών του.

Περίληψη

Στη παρούσα πτυχιακή εξετάζονται τα δίκτυα που καθορίζονται από λογισμικό και οι προκλήσεις ασφαλείας τους. Με δεδομένο ότι τα παραδοσιακά δίκτυα εμφανίζουν εγγενής αδυναμίες λόγω της κατανεμημένης αρχιτεκτονικής τους, η επεκτασιμότητα, συντήρηση και η ασφάλεια είναι θέματα που απασχολούν τους μηχανικούς δικτύων. Αυτές οι προκλήσεις επρόκειτο να επιλυθούν σε μεγάλο βαθμό από δίκτυα SDN που δημιουργήθηκαν με σκοπό τον καλύτερο έλεγχο των δικτύων διαχωρίζοντας το επίπεδο ελέγχου και δεδομένων. Αυτός ο διαχωρισμός επιτρέπει στα δίκτυα να επαναπρογραμματιστούν, όντας πιο ευέλικτα και προσαρμόσιμα. Όπως και στα παραδοσιακά δίκτυα, οι απειλές παραμένουν, όπως οι επιθέσεις DoS/DDoS και οι επιθέσεις σε πρωτόκολλα όπως το ARP και το DHCP. Η προστασία των δικτύων SDN μπορεί να επιτευχθεί με τη χρήση τεχνολογιών όπως τα Firewalls και τα IDS/IPS, παρόμοια με τα παραδοσιακά δίκτυα. Όμως, η υιοθέτηση της αρχιτεκτονικής SDN προσφέρει επιπλέον δυνατότητες, όπως η ενσωμάτωση μεθόδων τεχνητής νοημοσύνης και μηχανικής μάθησης, ενώ ταυτόχρονα παρέχει τη δυνατότητα χρήσης εφαρμογών που εκτελούνται στον controller. Στη παρούσα εργασία, πραγματοποιούνται προσομοιώσεις επιθέσεων, όπως ARP Spoofing, DDoS και Slowloris, ενώ παράλληλα προτείνονται μέθοδοι ανίχνευσης και αντιμετώπισης με βάση τον controller. Τα αποτελέσματα των προσομοιώσεων αποδεικνύουν ότι η αρχιτεκτονική SDN μπορεί να αποτελέσει ένα ισχυρό εργαλείο για την προστασία των δικτύων και των χρηστών τους.

«Security in Software Defined Networks and implementation of attack scenarios»

«Georgios Kakoushi»

Abstract

This thesis examines Software-Defined Networks (SDNs) and their security challenges. Given that traditional networks exhibit inherent weaknesses due to their distributed architecture, scalability, maintenance, and security are issues of concern for network engineers. These challenges were expected to be largely addressed by SDNs, which were designed to provide better network control by separating the control plane from the data plane. This separation allows networks to be reprogrammed, making them more flexible and adaptable. However, as with traditional networks, threats remain, such as DoS/DDoS attacks and protocol attacks like ARP and DHCP. The protection of SDNs can be achieved through technologies such as Firewalls and IDS/IPS, similar to traditional networks. However, the adoption of SDN architecture offers additional capabilities, such as the integration of artificial intelligence and machine learning methods, while also enabling the use of applications running on the controller. In this study, simulations of attacks such as ARP Spoofing, DDoS, and Slowloris are conducted, and detection and mitigation methods based on the controller are proposed. The simulation results demonstrate that the SDN architecture can be a powerful tool for protecting networks and their users.

Ευχαριστίες

Θα ήθελα να εκφράσω τις ειλικρινείς μου ευχαριστίες σε όλους όσους συνέβαλαν στην ολοκλήρωση της πτυχιακής μου εργασίας. Ευχαριστώ θερμά τον κύριο Δρ. Αντώνη Σαρηγιαννίδη για την καθοδήγηση, την υποστήριξη και τις πολύτιμες συμβουλές του κατά τη διάρκεια της εκπόνησης της εργασίας. Η εμπειρία και η γνώση του συνείσφεραν σημαντικά για την επιτυχή ολοκλήρωση της Πτυχιακής εργασίας. Ευχαριστώ επίσης τους συναδέλφους και τους φίλους μου για την υποστήριξή τους και για τις χρήσιμες παρατηρήσεις που συνέβαλαν στην εξέλιξη της εργασίας μου. Τέλος, ευχαριστώ την οικογένειά μου για την ανεξάντλητη υπομονή και ενθάρρυνση καθ' όλη τη διάρκεια της διαδικασίας. Η αμέριστη υποστήριξή τους υπήρξε καθοριστική για την ολοκλήρωση αυτού του έργου.

.

Περιεχόμενα

Κατάλογος Πινάκων

Κατάλογος Εικόνων

Συντομογραφίες

Δ.Ε.	Διπλωματική Εργασία
ΔΙΠΑΕ	Διεθνές Πανεπιστήμιο Ελλάδος
Π.Ε.	Πτυχιακή Εργασία
SDN	Software Defined Networks
IETF	Internet Engineering Task Force
IoT	Internet of Things
ARP	Address Resolution Protocol
DoS	Denial of Service
DDoS	Distributed Denial of Service
DHCP	Dynamic Host Configuration Protocol
IDS	Intrusion Detection System
IPS	Intrusion Prevention System
OSI	Open Systems Interconnection
TCP	Transmission Control Protocol
IP	Internet Protocol
ONF	Open Networking Foundation
QoS	Quality of Service
VLAN	Virtual Local Area Network
Wi-Fi	Wireless Fidelity
IS-IS	Intermediate System-to-Intermediate System
OSPF	Open Shortest Path First
BGP	Border Gateway Protocol
API	Application Programming Interface
MPLS	Multiprotocol Label Switching
RAM	Random Access Memory
ICMP	Internet Control Message Protocol
HTTPS	Hypertext transfer protocol secure
DAI	Dynamic ARP Inspection
SYN	Synchronize
ACK	Acknowledgement

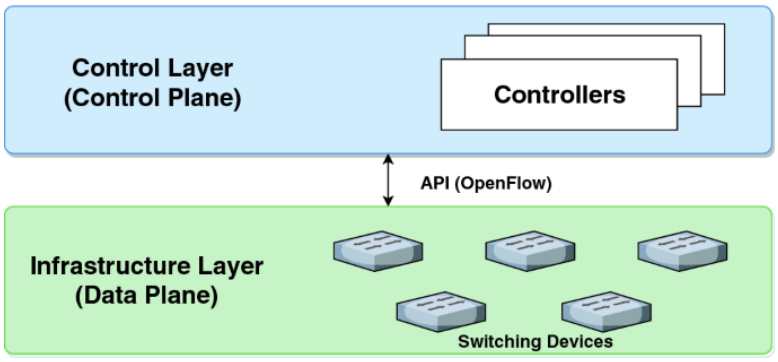
TLS	Transport Layer Security
SSL	Secure Sockets Layer
VPN	Virtual Private Network
VM	Virtual machine

Κεφάλαιο 1ο: Εισαγωγή

Από τα τέλη της δεκαετίας του 1960 που αναπτύχθηκε το πρώτο packet-switching δίκτυο, το ARPANET όπως ονομάστηκε, και υλοποιήθηκε στις Ηνωμένες πολιτείες Αμερικής, σχεδιάστηκε με κριτήρια την κατανομή του φόρτου και την παροχή υπηρεσιών, όπως ανταλλαγής μηνυμάτων, δεδομένων, προγραμμάτων και απομακρυσμένης πρόσβασης [1], [2]. Ένα παραδοσιακό δίκτυο υπολογιστών λειτουργεί με βάση 2 τύπους μοντέλων, OSI και TCP/IP. Στα πρώτα στάδια σχεδιασμού και υλοποίησης των δικτύων μεταγωγής πακέτων οι κατασκευαστές σχεδίαζαν και χρησιμοποιούσαν δικά τους πρωτόκολλα με αποτέλεσμα να μην είναι εφικτή η χρήση εξοπλισμού από διαφορετικούς κατασκευαστές, λόγω ασυμβατότητας, έτσι δημιουργήθηκαν τα δύο αυτά μοντέλα για να αντιμετωπιστεί το πρόβλημα [2].

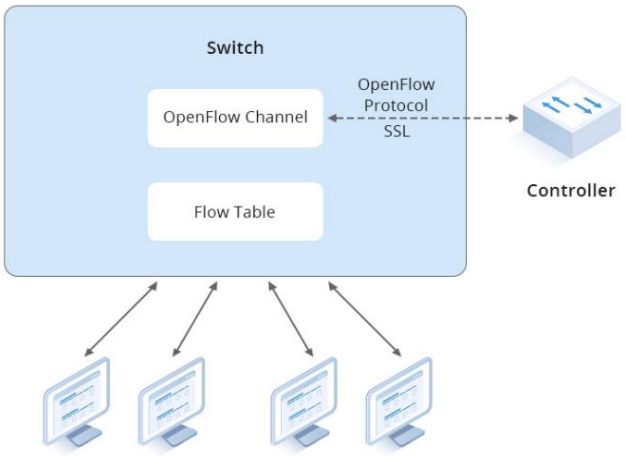
Πριν από την υλοποίηση ενός δικτύου είναι σημαντικό να γίνει κατάλληλος σχεδιασμός με βάση τη χρήση του και τις ανάγκες των χρηστών του. Ο σχεδιασμός του πρέπει να διαθέτει κάποια βασικά χαρακτηριστικά, όπως η χρηστικότητα, η ασφάλεια και η αξιοπιστία με δυνατότητα επέκτασης. Στην παραδοσιακή αρχιτεκτονική δικτύων η λειτουργεία της κάθε συσκευής είναι αυτόνομή και ανεξάρτητη. Αυτό το χαρακτηριστικό κάνει τα παραδοσιακής αρχιτεκτονικής δίκτυα δύσκολα στην τροποποίηση και επέκταση αλλά σε περίπτωση απώλειας μιας συσκευής οι υπόλοιπες συνεχίζουν να εξυπηρετούν. Η αυτόνομη λειτουργία των συσκευών που αποτελούν ένα παραδοσιακής αρχιτεκτονικής δίκτυο, στην περίπτωση που είναι πολύ μεγάλο μπορεί να το κάνει περίπλοκο στην παραμετροποίηση και να δημιουργήσει αρκετά προβλήματα, αφού σε περίπτωση τεχνικής βλάβης ή κάποιου λάθους κατά την παραμετροποίηση καθιστά την επίλυση τους δύσκολη και χρονοβόρα [3].

Για την αντιμετώπιση των προβλημάτων που προκύπτουν από την αρχιτεκτονική των παραδοσιακών δικτύων σχεδιάστηκαν τα δίκτυα SDN που έχουν ως βασική αρχή τον κεντρικοποιημένο έλεγχο. Τα δίκτυα SDN διαχωρίζουν το επίπεδο ελέγχου και το επίπεδο δεδομένων βελτιώνοντας σε μεγάλο βαθμό τη διαδικασία σχεδιασμού και υλοποίησης ενός δικτύου [4]. Στο επίπεδο ελέγχου βρίσκεται ο controller, ο οποίος ελέγχει όλη τη λειτουργία του δικτύου λαμβάνοντας όλες τις αποφάσεις για την διαχείριση και τη δρομολόγηση ενός πακέτου από τον αποστολέα μέχρι και την άφιξη στον παραλήπτη του. Για την λειτουργία του δικτύου μπορούν να χρησιμοποιηθούν παράλληλα περισσότεροι από ένας ελεγκτές ως εφεδρικοί, σε περίπτωση απώλειας του controller, ή σε συνδυασμό, ελέγχοντας ξεχωριστά τμήματα ενός πολύ μεγάλου δικτύου. Επιπλέον, στο επίπεδο δεδομένων βρίσκονται όλες οι συσκευές που μεταφέρουν τα δεδομένα των χρηστών όπως οι μεταγωγείς. Οι συσκευές αυτές με τη λήψη ενός πακέτου μεταφέρουν την πληροφορία στον controller, και ενεργούν σύμφωνα με τις οδηγίες που θα λάβουν από αυτόν [5].



Εικόνα 1.1: Διαχωρισμός επίπεδου ελέγχου και επίπεδου δεδομένων

Για την επικοινωνία μεταξύ επιπέδου ελέγχου και δεδομένων είναι απαραίτητη η χρήση ενός πρωτοκόλλου σχεδιασμένου κατάλληλα για τον σκοπό αυτό. Έχουν σχεδιαστεί αρκετά πρωτόκολλα με το πιο γνωστό να είναι το OpenFlow. Το πρωτόκολλο OpenFlow έχει τυποποιηθεί από τον ONF, με μέλη περισσότερους από 95 μεγάλους κατασκευαστές, μεταξύ αυτών η Cisco, HP και Dell [6], [7]. Με τη χρήση του OpenFlow παρέχεται η δυνατότητα συλλογής δεδομένων που αφορούν τα χαρακτηριστικά των πακέτων, δεδομένα κίνησης, προσθήκη και τροποποίηση καταχωρήσεων στους πίνακες ροής και το σημαντικότερο ασφάλεια στην επικοινωνία μεταξύ controller και μεταγωγέα. Μέχρι σήμερα έχουν υλοποιηθεί αρκετές εκδόσεις του πρωτοκόλλου με την πιο πρόσφατη να είναι η 1.5 ενώ με την υλοποίηση νεότερων εκδόσεων προστίθενται και νέες δυνατότητες [6, 7, 8].



Εικόνα 1.2: Πρωτόκολλο OpenFlow μεταξύ μεταγωγέα και controller

Τα SDN πέρα από τα πλεονεκτήματα τους, εμφανίζουν και πολλές ευπάθειες. Λόγω της κεντρικοποιημένης διαχείρισης τους υπάρχει μεγάλος κίνδυνος να επηρεαστεί η συνολική λειτουργία του δικτύου σε περίπτωση επίθεσης προς τον controller ή σε κάποιο στοιχείο της αρχιτεκτονικής SDN [9].

1.1 Στόχοι και σκοποί της πτυχιακής εργασίας

Η παρούσα πτυχιακή εργασία έχει στόχο την ανάδειξη των απειλών που καλούνται να αντιμετωπίσουν τα δίκτυα SDN και οι διαχειριστές τους. Επιπλέον, με τη χρήση προσομοιώσεων αναδεικνύονται οι επιπτώσεις που μπορούν να έχουν οι επιθέσεις αυτές σε ένα δίκτυο και παρουσιάζονται μέθοδοι εντοπισμού και αντιμετώπισης τους.

1.2 Δομή της πτυχιακής εργασίας

Η παρούσα πτυχιακή εργασία αποτελείται από 7 κεφάλαια.

Στο πρώτο κεφάλαιο γίνεται εισαγωγή στο κομμάτι της δικτύωσης και στους λόγους που έγινε αναγκαία η υλοποίηση μιας νέας αρχιτεκτονικής κατάλληλη να ανταπεξέλθει στις σύγχρονες απαιτήσεις.

Στο δεύτερο κεφάλαιο περιγράφονται οι προκλήσεις που προκύπτουν από τη χρήση των παραδοσιακών τεχνολογιών δικτύωσης.

Στο τρίτο κεφάλαιο γίνεται περιγραφή των εφαρμογών που μπορεί να γίνει χρήση της τεχνολογίας SDN καθώς και των βασικών χαρακτηριστικών που την κάνουν να ξεχωρίζει από τις παραδοσιακές τεχνολογίες που χρησιμοποιούνται. Επιπλέον γίνεται αναφορά στους διάφορους ελεγκτές που είναι διαθέσιμοι σε μορφή ελεύθερου λογισμικού και των χαρακτηριστικών τους.

Στο τέταρτο κεφάλαιο γίνεται εκτενής αναφορά σε επιθέσεις που μπορούν να επηρεάσουν τη λειτουργία ενός δικτύου και των χρηστών του. Μαζί με τις επιθέσεις περιγράφονται μέθοδοι που μπορούν να εφαρμοστούν για την αποτροπή ή την αντιμετώπιση των επιθέσεων.

Στο πέμπτο κεφάλαιο γίνεται εισαγωγή στο κομμάτι των προσομοιώσεων με την περιγραφή των εργαλείων και των μεθόδων υλοποίησης τους καθώς και στο περιβάλλον προσομοίωσης.

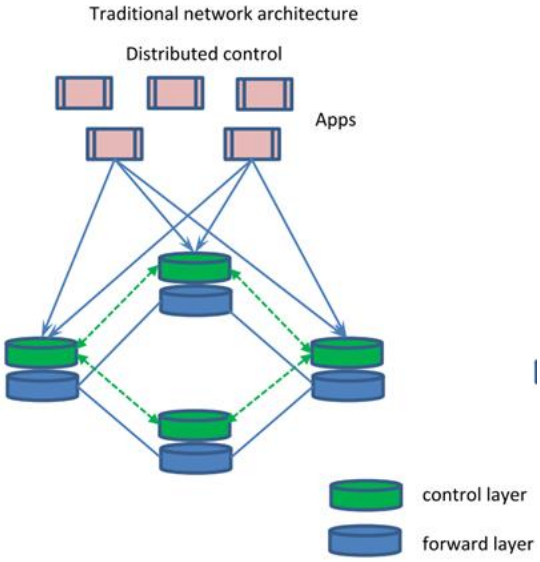
Το έκτο κεφάλαιο αποτελείται από την ανάλυση των επιθέσεων ARP Spoofing, DDoS και Slowloris χωρίς τη λήψη μέτρων προστασίας και στη συνέχεια με την ενσωμάτωση μεθόδου εντοπισμού και αντιμετώπισης για την κάθε επίθεση χωριστά. Η ανάλυση της κάθε προσομοίωσης συνοδεύεται με το αποτέλεσμα της εκτέλεσης της και στις δύο περιπτώσεις.

Στο έβδομο και τελευταίο κεφάλαιο περιγράφονται τα συμπεράσματα που προκύπτουν από την συγγραφή της παρούσας πτυχιακής εργασίας και γίνεται αναφορά στις μελλοντικές επεκτάσεις που μπορούν να γίνουν.

Κεφάλαιο 2ο: Προκλήσεις παραδοσιακών τεχνολογιών δικτύωσης

2.1 Εισαγωγή

Τα παραδοσιακά δίκτυα υιοθετούν αρχιτεκτονική η οποία αναπτύχθηκε σε βάθος δεκαετιών και πλέον έχουν φτάσει σε σημείο που δεν μπορούν να ανταπεξέλθουν στις σύγχρονες απαιτήσεις. Ο σχεδιασμός, η υλοποίηση και η συντήρηση των παραδοσιακών τεχνολογιών δικτύωσης αποτελούν πρόκληση για τους μηχανικούς καθώς όσο μεγαλύτερο γίνεται ένα δίκτυο τόσο δυσκολότερη γίνεται η διαχείριση του [10].



Εικόνα 2.1: Παραδοσιακή αρχιτεκτονική δικτύου

2.2 Προκλήσεις

2.2.1 Διαχείριση

Η αρχιτεκτονική ενός παραδοσιακού δικτύου είναι δομημένη ιεραρχικά όπου στην κορυφή βρίσκεται ο δρομολογητής, στη μέση τα σημεία πρόσβασης, όπως μεταγωγείς, και στο κάτω μέρος οι χρήστες. Η κάθε συσκευή που βρίσκεται στα επίπεδα της αρχιτεκτονικής λειτουργεί αυτόνομα, αφού συνδυάζεται το επίπεδο ελέγχου και δεδομένων. Το χαρακτηριστικό αυτό προσθέτει ακόμα ένα επίπεδο δυσκολίας στους μηχανικούς αφού απαιτείται ακρίβεια χωρίς να δίνει περιθώριο για οποιοδήποτε λάθος ή παράλειψη κατά την παραμετροποίηση του δικτύου. Ο συνδυασμός των δύο επιπέδων μπορεί να αποτελέσει και πλεονέκτημα έναντι των δικτύων SDN αφού σε περίπτωση απώλειας μιας συσκευής η δρομολόγηση μπορεί να γίνει από άλλες, χωρίς να επηρεάζεται σε μεγάλο βαθμό η λειτουργία του. Όπως αναφέρεται στη συνέχεια, στα δίκτυα SDN η διαχείριση γίνεται από τον controller και σε περίπτωση απώλειας του το δίκτυο μπορεί να τεθεί εκτός λειτουργίας, αφού δεν μπορούν να ληφθούν αποφάσεις δρομολόγησης [11], [12].

2.2.2 Επεκτασιμότητα

Η επεκτασιμότητα στα παραδοσιακής αρχιτεκτονικής δίκτυα αποτελεί μια από τις μεγαλύτερες προκλήσεις που αντιμετωπίζουν οι μηχανικοί δικτύων. Τα δίκτυα αυτά έχουν σχεδιαστεί με αρχιτεκτονικές που δεν είναι προσαρμοσμένες για να διαχειρίζονται τον διαρκώς αυξανόμενο όγκο δεδομένων και τον μεγάλο αριθμό συσκευών που συνδέονται καθημερινά, ειδικά με την ανάπτυξη του IoT και άλλων σύγχρονων τεχνολογιών. Η προσθήκη νέων κόμβων και συσκευών αυξάνει τη πολυπλοκότητα της διαχείρισης και παραμετροποίησης τους, δημιουργώντας σημεία συμφόρησης και αυξημένο κίνδυνο σφαλμάτων. Επιπλέον, οι διαδικασίες αναβάθμισης και επέκτασης είναι συχνά χρονοβόρες και απαιτούν υψηλό κόστος σε εξοπλισμό και ανθρώπινο δυναμικό, καθιστώντας δύσκολη τη γρήγορη προσαρμογή στις νέες ανάγκες και τεχνολογικές εξελίξεις [11], [13].

2.2.3 Αυτοματοποίηση

Στη σύγχρονη εποχή, η αυτοματοποίηση διεργασιών έχει αλλάξει τον τρόπο με τον οποίο ο άνθρωπος αλληλεπιδρά στην καθημερινότητά του. Το ίδιο ισχύει και στα δίκτυα, όπου πολλές από τις διεργασίες που εκτελούνται είναι απαραίτητο να αυτοματοποιηθούν για σκοπούς καλύτερης διαχείρισης και παροχής υπηρεσιών. Στα δίκτυα παραδοσιακής αρχιτεκτονικής, η αυτοματοποίηση αποτελεί πρόκληση, καθώς δεν έχουν σχεδιαστεί για να την υποστηρίζουν. Για την οποιαδήποτε αλλαγή στη λειτουργία του δικτύου είναι αναγκαίο να γίνεται χειροκίνητα από τον διαχειριστή, κάτι το οποίο είναι χρονοβόρο και παράλληλα περίπλοκο, καθώς κάθε αλλαγή στην παραμετροποίηση πρέπει να γίνει με κατάλληλο τρόπο, ώστε να μην επηρεαστούν άλλες ρυθμίσεις που έχουν ήδη πραγματοποιηθεί [10].

2.2.4 Ενσωμάτωση με νέες τεχνολογίες

Πέρα από την αυτοματοποίηση που προαναφέρθηκε, έχουν αναπτυχθεί νέες τεχνολογίες, όπως η μηχανική μάθηση και η τεχνητή νοημοσύνη, δύο πολύ υποσχόμενες τεχνολογίες που μπορούν να αλλάξουν τον τρόπο επεξεργασίας των πληροφοριών. Λόγω της κατανεμημένης αρχιτεκτονικής τους, τα παραδοσιακά δίκτυα είναι δύσκολο να αλληλεπιδράσουν με αυτά τα μοντέλα, καθώς δεν διαθέτουν την κατάλληλη μέθοδο επικοινωνίας, με αποτέλεσμα να μην είναι εφικτή ή να διατρέχει κινδύνους, όταν η ενσωμάτωσή τους γίνεται με τις υπάρχουσες μεθόδους διασύνδεσης. Επιπλέον, εξαιτίας της κλειστής δομής των παραδοσιακών συσκευών δικτύωσης, η οποιαδήποτε ενσωμάτωση νέων τεχνολογιών εξαρτάται από τους κατασκευαστές, αφού μόνο αυτοί μπορούν να τις προσθέσουν στις συσκευές τους εφαρμόζοντας αναβαθμίσεις λογισμικού όταν είναι εφικτό.

2.2.5 Κόστος

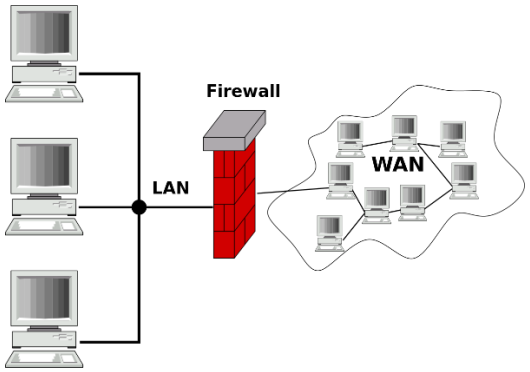
Η πολυπλοκότητα υλοποίησης και συντήρησης ενός δικτύου παραδοσιακής αρχιτεκτονικής αποτελεί πρόκληση για τους μηχανικούς, καθώς απαιτεί χρόνο, εξειδικευμένο προσωπικό και συνεχή επιτήρηση. Κάθε συσκευή ή μέρος του δικτύου είναι αυτόνομο και πρέπει να λειτουργεί συγχρονισμένα με τα υπόλοιπα, κάτι που αυξάνει τη δυσκολία διαχείρισης και συντήρησης. Επιπλέον, η ανάγκη για συνεχείς χειροκίνητες αλλαγές στην παραμετροποίηση προσθέτει επιπλέον πολυπλοκότητα, καθώς οποιαδήποτε αλλαγή ή αναβάθμιση απαιτεί προσοχή για να διασφαλιστεί η ομαλή λειτουργία του δικτύου. Σε σύγκριση με τα δίκτυα SDN, τα παραδοσιακά δίκτυα παρουσιάζουν αυξημένο κόστος και δυσκολία στην προσαρμογή σε νέες απαιτήσεις όπως περιγράφεται πιο κάτω.

1. Κόστος φυσικού εξοπλισμού: Η αγορά εξοπλισμού για σκοπούς αναβάθμισης σε σύγχρονες εκδόσεις, που διαθέτουν επιπλέον λειτουργίες, αυξάνει σημαντικά το κόστος συντήρησης και αναβάθμισης.

2. Κόστος λογισμικού όπως άδειες χρήσης και διαχείρισης: Αρκετά συστήματα πλέον πωλούνται με άδειες χρήσης που έχουν συχνά προκαθορισμένη διάρκεια και με τη λήξη τους απαιτείται ανανέωση για λήψη ενημερώσεων ή τεχνικής υποστήριξης από τον κατασκευαστή.
3. Εξειδικευμένο προσωπικό για παρακολούθηση και επίλυση προβλημάτων: Η διαχείριση και συντήρηση ενός παραδοσιακού δικτύου απαιτεί στις περισσότερες περιπτώσεις προσωπικό με εξειδικευμένες γνώσεις και εκπαιδευμένο κατάλληλα.
4. Κόστος λειτουργίας: Η υψηλή κατανάλωση ενέργειας είναι επίσης ένας σημαντικός παράγοντας κόστους λόγω της απαιτούμενης επεξεργαστικής ισχύς

2.2.6 Ασφάλεια

Κατά τη λειτουργία ενός δικτύου είναι απαραίτητη η ασφαλής επικοινωνία και ανταλλαγή δεδομένων μεταξύ των χρηστών, ανεξάρτητα από το σημείο που βρίσκονται. Για τα παραδοσιακά δίκτυα η ασφάλεια αποτελεί πρόκληση αφού λόγω της κατανεμημένης αρχιτεκτονικής τους δεν είναι εφικτή η παρακολούθηση απειλών στο σύνολο του δικτύου. Η ασφάλεια στα παραδοσιακής αρχιτεκτονικής δίκτυα παρέχεται από συσκευές προστασίας, όπως τα firewall, που είναι εγκατεστημένα στο άκρο ενός τοπικού δικτύου και αδυνατούν να παρέχουν προστασία σε σημεία που δεν έχουν πρόσβαση όπως το εσωτερικό ενός τοπικού δικτύου. Στόχο αποτελούν και τα διάφορα πρωτόκολλα που χρησιμοποιούνται κατά την ανταλλαγή δεδομένων σε ένα δίκτυο, αφού αρκετά δεν έχουν σχεδιαστεί με αυστηρά κριτήρια ασφαλείας, ενώ παράλληλα στόχος μπορεί να γίνει και το λειτουργικό σύστημα της κάθε δικτυακής συσκευής, όπως οι δρομολογητές και οι μεταγωγείς. Ακόμα στόχος γίνονται και οι διάφορες υπηρεσίες που παρέχονται μέσω των δικτύων, όπως οι WEB και mail [14].



Εικόνα 2.2: Μοντέλο ασφάλειας παραδοσιακής αρχιτεκτονικής δικτύων

Η ασφάλεια για τους μηχανικούς δικτύων αποτελεί πρόκληση καθώς η προστασία του δικτύου από διάφορους τύπους απειλών δεν μπορεί να επιτευχθεί από οποιοδήποτε σημείο. Ο εντοπισμός και η αντιμετώπιση μιας επίθεσης σε ένα παραδοσιακό δίκτυο μπορεί να γίνει με δύο τρόπους, με τη χρήση των firewall σε μορφή συσκευής και σε μορφή λογισμικού στη συσκευή του κάθε χρήστη ή διακομιστή. Πέρα από τη χρήση τοίχους προστασίας μπορούν να χρησιμοποιηθούν, σε συνδυασμό, τεχνικές αποτροπής εκτέλεσης μιας επίθεσης όπως περιγράφεται παρακάτω [15]:

1. Τμηματοποίηση: Χωρίζοντας το δίκτυο σε τμήματα μειώνεται η πιθανότητα να επηρεαστεί μεγάλος αριθμός χρηστών σε περίπτωση εκτέλεσης κάποιας επίθεσης.
2. Έλεγχος πρόσβασης: Περιορίζοντας το ποιοι μπορούν να έχουν πρόσβαση μειώνονται οι πιθανότητες κάποιος από τους χρήστες να είναι κακόβουλος ή να παραβιαστεί κάποιο κρίσιμο σύστημα από τοπικό χρήστη.

3. Παρακολούθηση και καταγραφή: Παρατηρώντας το δίκτυο μπορούν να εντοπιστούν πιθανές απόπειρες πρόσβασης σε κρίσιμα συστήματα που μπορούν να επηρεάσουν την ομαλή λειτουργία του.
4. Χρήση εφεδρικών συστημάτων: Σε περίπτωση που κάποιο μέρος του δικτύου αδυνατεί να ανταποκριθεί η χρήση εφεδρικών συστημάτων εξασφαλίζει την άμεση μεταφορά των δεδομένων στον παραλήπτη τους.
5. Συχνή αναβάθμιση λογισμικού: Οι κατασκευαστές ανά τακτικά χρονικά διαστήματα δημοσιεύουν ενημερώσεις στα συστήματα τους με σκοπό την διόρθωση πιθανών ευπαθειών που ίσως κρύβουν και προσθήκη νέων χαρακτηριστικών για προστασία από νέου τύπου επιθέσεις.

2.3 Επίλογος

Οι ανάγκες των χρηστών συνεχώς εξελίσσονται, ενώ τα δίκτυα γίνονται μεγαλύτερα και πιο περίπλοκα, γεγονός που δημιουργεί σημαντικές προκλήσεις για τους διαχειριστές τους. Αυτές οι προκλήσεις καθιστούν δύσκολο το έργο των μηχανικών που έχουν στόχο την καλύτερη και ασφαλέστερη παροχή υπηρεσιών στους τελικούς χρήστες. Παράλληλα, η συνεχής επέκταση και αναβάθμιση των δικτύων φέρνει νέες προκλήσεις στην ασφάλεια αυξάνοντας παράλληλα τους πιθανούς στόχους επιθέσεων.

Κεφάλαιο 3ο: Τεχνολογία SDN

3.1 Εισαγωγή

Από τη δεκαετία του 1990 όταν η χρήση του διαδικτύου άρχισε να εξαπλώνεται σε όλο τον κόσμο, οι μηχανικοί άρχισαν να πειραματίζονται για τη βελτιστοποίηση των δικτύων καθώς η χρήση τους άρχισε να γίνεται όλο και πιο απαιτητική. Αρχικά οι ερευνητές απευθύνθηκαν στον οργανισμό Internet Engineering Task Force (IETF), όπου είναι υπεύθυνος για την τυποποίηση νέων πρωτοκόλλων, με τις ιδέες και τα αποτελέσματα των δοκιμών τους. Αφού ολοκληρώθηκαν οι διαδικασίες τυποποίησης έλαβαν ώθηση για την περεταίρω έρευνα και μελέτη ώστε να ακολουθήσουν το μοντέλο των υπολογιστών όπου παρέχεται η δυνατότητα επαναπρογραμματισμού τους. Οι έρευνες οδήγησαν στον διαχωρισμό του επιπέδου ελέγχου και δεδομένων, ο οποίος αποτελεί και τη σημερινή βασική αρχή των δικτύων καθορισμένων από λογισμικό [16].

3.2 Εφαρμογές χρήσης

Όπως προαναφέρθηκε, η χρήση της τεχνολογίας τα τελευταία χρόνια βελτίωσε σε μεγάλο βαθμό στην καθημερινότητα των ανθρώπων. Πέρα από τους σκοπούς επικοινωνίας και ανταλλαγής πληροφοριών, στην χρήση των δικτύων ενσωματώθηκαν και άλλες εφαρμογές με αυτοματοποιημένες λειτουργίες, με τις οποίες δεν αλληλεπιδρά άμεσα ο άνθρωπος, αλλά καθιστούν την καθημερινότητα του πιο εύκολη και ασφαλή.

3.2.1 Internet of Things

Η χρήση του IoT έδωσε τη δυνατότητα σε πολλές συσκευές να επικοινωνήσουν μέσω των δικτύων από απομακρυσμένα σημεία παρέχοντας πληροφορίες χωρίς την ανθρώπινη παρέμβαση. Αυτές οι συσκευές παρέχουν τη δυνατότητα συλλογής πληροφοριών όπως θερμοκρασία και υγρασία ή ακόμα και τον απομακρυσμένο έλεγχο εργασιών όπως τη διαχείριση φωτισμού σε έξυπνες πόλεις [12].

3.2.2 Κέντρα πληροφοριών (Data centers)

Η συνεχώς αυξανόμενη χρήση υπηρεσιών διαδικτύου αυξάνει την ανάγκη χρήσης διακομιστών. Ένα κέντρο δικτύου, για να καλύψει τις ανάγκες του, μπορεί να φιλοξενεί τεράστιο αριθμό διακομιστών γεγονός που αποτελεί πρόκληση για τους διαχειριστές του, αφού η διασύνδεση των συστημάτων αυτών γίνεται όλο και πιο περίπλοκη. Με τη χρήση των SDN η διαδικασία διαχείρισης τους γίνεται πιο εύκολη λόγω του κεντρικοποιημένου ελέγχου που παρέχουν. Επιπλέον η χρήση των εικονικών μηχανών απαιτεί τη χρήση εικονικών δικτύων, κάτι που επιτυγχάνεται με τα δίκτυα καθοριζόμενα από λογισμικό [17].

3.2.3 Δίκτυα παρόχων

Η ανάπτυξη των δικτύων 5^{ης} γενιάς έκανε τη λειτουργία των τηλεπικοινωνιακών δικτύων πιο περίπλοκη λόγω του μεγάλου αριθμού εφαρμογών και υπηρεσιών που μπορούν να χρησιμοποιηθούν, όπως το IoT. Επιπλέον τα δίκτυα 5G μπορούν να προσφέρουν ταχύτητες πολύ μεγαλύτερες από τις προηγούμενες γενιές αυξάνοντας παράλληλα τον όγκο των δεδομένων που καλούνται να διαχειριστούν. Με την αξιοποίηση των δικτύων SDN οι παρόχοι αποκτούν καλύτερο έλεγχο των δικτύων τους και κατά συνέπεια τους δίνεται η δυνατότητα να παρέχουν καλύτερες υπηρεσίες στους χρήστες [18].

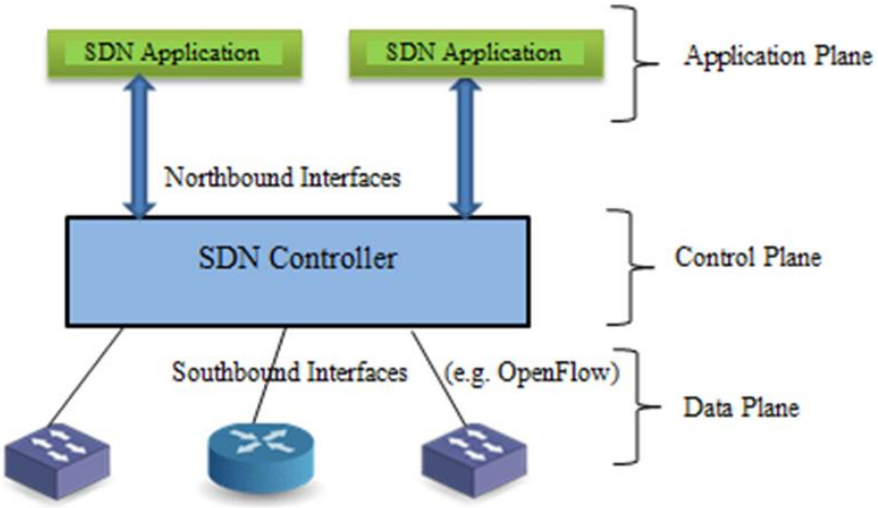
3.3 Οφέλη των SDN

Τα δίκτυα SDN χάρη στην αρχιτεκτονική τους προσφέρουν σημαντικά οφέλη με τη χρήση τους, που τα παραδοσιακά δίκτυα δεν ήταν ικανά. Τα χαρακτηριστικά που περιγράφονται πιο κάτω δίνουν τη δυνατότητα δημιουργίας πολύπλοκων δικτύων χωρίς να απαιτείται χρόνος για την υλοποίηση και την παραμετροποίηση τους [19]:

1. Ευελιξία και έλεγχος: Ο διαχωρισμός του επιπέδου ελέγχου από το επίπεδο δεδομένων παρέχει τη δυνατότητα κεντρικοποιημένης διαχείρισης στο δίκτυο, αφού ο έλεγχος όλων των δικτυακών συσκευών γίνεται από ένα κεντρικό σημείο, τον controller.
2. Παραμετροποίηση: Σε σύγκριση με την κατανεμημένη διαχείριση, η κεντρικοποιημένη παρέχει τη δυνατότητα εύκολης και άμεσης τροποποίησης του δικτύου χωρίς να είναι απαραίτητη η εφαρμογή των αλλαγών κάθε συσκευή χωριστά.
3. Ασφάλεια: Χάρη στην κεντρικοποιημένη διαχείριση παρέχεται η δυνατότητα εφαρμογής μέτρων προστασίας σε κάθε σημείο του δικτύου προσφέροντας γρήγορη και άμεση αντιμετώπιση πιθανών απειλών.
4. QoS: Τα σύγχρονα δίκτυα καλούνται να διαχειριστούν μεγάλο αριθμό διαφορετικών εφαρμογών, όπως οπτικοακουστικό υλικό ή κείμενο. Για την καλύτερη εμπειρία των χρηστών γίνεται χρήση πολιτικών προτεραιότητας σε κάποιες εφαρμογές που απαιτούν συγκεκριμένες συνθήκες λειτουργίας. Η χρήση κάποιων εφαρμογών δεν απαιτείται όλες τις χρονικές περιόδους και για το λόγο αυτό οι πολιτικές QoS μπορούν να εφαρμοστούν δυναμικά, σύμφωνα με την κίνηση και τις απαιτήσεις των χρηστών.
5. Δυναμική δρομολόγηση: Όπως και στις πολιτικές QoS, κάποια δεδομένα χρειάζονται ειδικό χειρισμό σε σύγκριση με άλλα. Χάρη στην δυναμική δρομολόγηση των πακέτων γίνεται επιλογή της πιο κατάλληλης διαδρομής ώστε να φτάσουν στον προορισμό τους με τον καλύτερο δυνατό τρόπο.
6. Ενσωμάτωση με νέες τεχνολογίες: Καθώς η τεχνολογία αναπτύσσεται και δημιουργούνται νέου τύπου υπηρεσίες είναι σημαντικό να μπορούν να ενσωματωθούν με τις υπάρχουσες. Η ανοιχτή δομή των SDN δικτύων επιτρέπει την ενσωμάτωση νέων τεχνολογιών όπως είναι η τεχνητή νοημοσύνη και η μηχανική μάθηση.

3.4 Δομή τεχνολογίας SDN

Η δομή ενός δικτύου SDN χωρίζεται σε 3 βασικά επίπεδα, εφαρμογής, ελέγχου και δεδομένων. Το κάθε ένα από αυτά τα επίπεδα είναι σχεδιασμένο κατάλληλα ώστε να εκτελεί συγκεκριμένες λειτουργίες και να ανταλλάσσει πληροφορίες με τα υπόλοιπα. Τα επίπεδα αυτά για την επικοινωνία τους χρησιμοποιούν δύο διεπαφές, τη Βόρεια που χρησιμοποιείται για τη διασύνδεση του επιπέδου εφαρμογής και του επιπέδου ελέγχου και τη Νότια που χρησιμοποιείται μεταξύ του επιπέδου ελέγχου και επιπέδου δεδομένων [20].



Εικόνα 3.1: Δομή αρχιτεκτονικής SDN

3.4.1 Επίπεδο εφαρμογών

Στο επίπεδο εφαρμογών φιλοξενούνται όλες οι εφαρμογές που αλληλεπιδρούν με τον controller και χρησιμοποιούνται για τη λήψη αποφάσεων. Κατά την διανομή τους οι περισσότεροι ελεγκτές διαθέτουν προεγκατεστημένες εφαρμογές, που μπορούν να χρησιμοποιηθούν μεμονωμένα ή σε συνδυασμό, ενώ παρέχεται επίσης η δυνατότητα στους προγραμματιστές να δημιουργήσουν δικές τους εφαρμογές, βασισμένες σε αυτές ή δημιουργώντας νέες εξ ολοκλήρου. Χάρη στη χρήση ελεγκτών ανοικτού κώδικα η δημιουργία και εκτέλεση πρωτότυπων εφαρμογών είναι σημαντικό πλεονέκτημα για τα SDN δίκτυα, σε σύγκριση με τα παραδοσιακά που οι δυνατότητες τους εξαρτώνται από τον κατασκευαστή [19].

Μια εφαρμογή SDN παρέχει μια μεγάλη γκάμα δυνατοτήτων που παλαιότερα δεν ήταν εφικτή, αφού η παραδοσιακή αρχιτεκτονική δεν το υποστήριζε. Βασική λειτουργία των εφαρμογών και του επιπέδου που τις φιλοξενεί είναι η λήψη αποφάσεων δρομολόγησης. Οι αποφάσεις λαμβάνονται σύμφωνα με την εκτελούμενη εφαρμογή και τις παραμέτρους λειτουργίας της, όπως για παράδειγμα η λειτουργία των VLAN.

Πέρα από την δρομολόγηση πακέτων που προϋπήρχε, το επίπεδο εφαρμογών παρέχει τη δυνατότητα προστασίας του δικτύου στο σύνολο του. Η προστασία του δικτύου μπορεί να επιτευχθεί με την χρήση των δεδομένων που συλλέγονται από τους μεταγωγείς, την μεταφορά τους στον controller με τη χρήση κατάλληλου πρωτοκόλλου, όπως το OpenFlow, και την επεξεργασία τους από εφαρμογές Firewall, IDS και IPS.

Με τη χρήση των σύγχρονων μεθόδων επικοινωνίας και παρακολούθησης περιεχομένου όπως τα μέσα κοινωνικής δικτύωσης και οι πλατφόρμες ζωντανής παρακολούθησης προγραμμάτων, είναι σημαντική η παροχή προτεραιότητας, αφού οι χρήστες τους επηρεάζονται από καθυστερήσεις και διακοπές. Όπως και στα παραδοσιακά δίκτυα η χρήση τεχνολογιών Quality Of Service είναι απαραίτητη. Σε ένα δίκτυο SDN η παροχή QoS μπορεί να γίνει ακόμα πιο αποδοτική, αφού με τη χρήση εφαρμογών δίνεται η δυνατότητα να γίνει δυναμικά και σε συνδυασμό με σύγχρονες μεθόδους, όπως τεχνητή νοημοσύνη και η μηχανική μάθηση για την παροχή έξυπνων λύσεων διαχείρισης.

3.4.2 Βόρεια διεπαφή

Για την ανταλλαγή δεδομένων μεταξύ του controller και του επίπεδου εφαρμογών είναι απαραίτητη η χρήση ενός διαύλου επικοινωνίας. Για τη διασύνδεση τους χρησιμοποιείται η Βόρεια διεπαφή. Η χρήση της διεπαφής επιτρέπει την ανάπτυξη και την εκτέλεση εφαρμογών σε διαφορετικό υπολογιστικό σύστημα από τον controller, ενώ επιτρέπει την επικοινωνία πολλών ελεγκτών με το ίδιο επίπεδο εφαρμογών [5].

3.4.3 Επίπεδο ελέγχου

Το κύριο χαρακτηριστικό της τεχνολογίας SDN είναι το επίπεδο ελέγχου στο οποίο φιλοξενείται ο Controller, και βρίσκεται μεταξύ του επιπέδου εφαρμογών και δεδομένων, ενώ για τη διασύνδεση τους χρησιμοποιείται η Βόρεια και Νότια διεπαφή αντίστοιχα. Στο επίπεδο αυτό λαμβάνονται όλες οι αποφάσεις για τη διαχείριση όχι μόνο των μεταγωγέων αλλά και των πακέτων που καλείται το δίκτυο να μεταφέρει στον παραλήπτη τους. Με την άφιξη ενός πακέτου για το οποίο δεν υπάρχει κανόνας ροής στους μεταγωγείς του δικτύου, η πρώτη ενέργεια χειρισμού του είναι η αποστολή των δεδομένων του στον controller, όπου παράλληλα με την εκτελούμενη εφαρμογή λαμβάνεται η απόφαση για απόρριψη ή προώθηση του από την κατάλληλη διαδρομή. Λόγω της κεντρικοποιημένης διαχείρισης του δικτύου, το επίπεδο ελέγχου βρίσκεται σε ευάλωτη θέση, καθώς όσο μεγαλώνει το δίκτυο καλείται να διαχειριστεί μεγαλύτερο φόρτο εργασίας και πολύ πιθανό να αποτελέσει στόχο διαφόρων επιθέσεων, όπως DDoS και Brute force [19]. Επιπλέον για τη λειτουργία του controller είναι απαραίτητη η χρήση τυποποιημένων πρωτοκόλλων, όπως το OpenFlow, για την επεξεργασία πακέτων και τη διασύνδεση του με τα υπόλοιπα επίπεδα.

3.4.4 Νότια διεπαφή

Από την άλλη μεριά, για την επικοινωνία του επίπεδου ελέγχου και δεδομένων χρησιμοποιείται η Νότια διεπαφή. Η διεπαφή αυτή αντιπροσωπεύεται από τη χρήση πρωτοκόλλων επικοινωνίας, των οποίων ο σκοπός είναι η μεταφορά δεδομένων και εντολών από τις δικτυακές συσκευές και τον controller ή αντίστροφα. Σήμερα, το πιο γνωστό πρωτόκολλο που χρησιμοποιείται είναι το OpenFlow. Άλλα πρωτόκολλα που σχεδιάστηκαν ώστε να χρησιμοποιηθούν στη Νότια διεπαφή είναι NETCONF, OF-Config, Opflex και επιπλέον τα IS-IS, OSPF, BGP, τα οποία σχεδιάστηκαν ώστε να παρέχουν τη δυνατότητα διασύνδεσης με συσκευές δικτύωσης παραδοσιακής αρχιτεκτονικής [21].

3.4.5 Επίπεδο δεδομένων

Ο τελικός χρήστης για την μετάδοση ή τη λήψη δεδομένων είναι συνδεδεμένος στο δίκτυο με μεθόδους όπως τα καλώδια από συνεστραμμένα ζεύγη ή τη χρήση ασύρματων τεχνολογιών, όπως το WiFi. Το επίπεδο που έχει αυτό το ρόλο είναι το επίπεδο δεδομένων. Ο βασικός ρόλος του σε ένα δίκτυο είναι η λήψη, μεταφορά και παράδοση των πακέτων από ένα χρήστη σε άλλον, εντός ή εκτός του τοπικού δικτύου. Στο επίπεδο δεδομένων βρίσκονται οι μεταγωγείς, όπου για την επικοινωνία τους με το επίπεδο ελέγχου χρησιμοποιείται η Νότια διεπαφή, ενώ για την επιτυχή επικοινωνία του με τον controller είναι απαραίτητη η υποστήριξη κοινού πρωτοκόλλου [19].

3.5 OpenFlow

Μέχρι σήμερα η αρχιτεκτονική των δικτύων συνδύαζε το επίπεδο ελέγχου και το επίπεδο δεδομένων σε μια συσκευή, κάτι το οποίο με την ανάπτυξη των SDN άλλαξε. Η αλλαγή αυτή έφερε την ανάγκη υλοποίησης μεθόδων επικοινωνίας των δύο αυτών επιπέδων, αφού διαχωρίστηκαν, και πλέον

επικοινωνούν χρησιμοποιώντας τη Νότια διεπαφή. Για την επικοινωνία τους μέσω της Νότιας διεπαφής είναι απαραίτητη η χρήση ενός πρωτοκόλλου, κοινό και στις δύο πλευρές, ώστε να επιτευχθεί επικοινωνία που να είναι άμεση, αξιόπιστη και ασφαλές, αφού σε περίπτωση μη ορθής λειτουργίας δεν μπορεί να επιτευχθεί η οποιαδήποτε κίνηση εντός του δικτύου. Το πανεπιστήμιο του Στάνφορντ έκανε τη δική του προσπάθεια να δημιουργήσει ένα πρωτόκολλο κατάλληλο να ανταπεξέλθει στις απαιτήσεις ενός δικτύου SDN με αποτέλεσμα να παρουσιάσει για πρώτη φορά το πρωτόκολλο OpenFlow, το 2008, που έχει πλέον τυποποιηθεί από το Open Networking Foundation και σχεδιάστηκε ώστε να επιτρέψει στην ασφαλή επικοινωνία μεταξύ controller και συσκευών που αποτελούν το δίκτυο, χωρίς να επεμβαίνει στα πακέτα [22]. Το OpenFlow χάρη στη χρήση ανοικτού τύπου API παρέχει πολλές δυνατότητες στους μηχανικούς δικτύων με τις πιο σημαντικές να είναι [23], [24]:

1. Επεκτασιμότητα και ευελιξία: Προσφέρεται η δυνατότητα στους μηχανικούς να δημιουργούν εφαρμογές που δεν διανέμονται από τους κατασκευαστές και έχουν τη δυνατότητα να επεξεργάζονται δεδομένα που λαμβάνουν μέσω του OpenFlow και είναι απαραίτητες για την υλοποίηση σύνθετων δικτύων.
2. Μικρό κόστος διαχείρισης και συντήρησης: Παρέχει το βασικό χαρακτηριστικό της λειτουργίας των SDN δικτύων που είναι η κεντρικοποιημένη διαχείριση μειώνοντας σημαντικά το χρόνο υλοποίησης και παραμετροποίησης, αφού η κάθε εργασία πραγματοποιείται μια φορά και εφαρμόζεται σε ολόκληρο το δίκτυο.
3. Ανεξαρτησία από κατασκευαστές: Χάρη στην τυποποίηση του, χρησιμοποιείται από πολλούς κατασκευαστές κάνοντας τις συσκευές συμβατές μεταξύ τους.

Σε ένα δίκτυο SDN, η εντός του δικτύου επικοινωνία είναι σημαντικό να καλύπτει όλες τις απαραίτητες ενέργειες που θα χρειαστεί να εκτελεστούν, και χωρίζονται σε τρεις τύπους μηνυμάτων, συμμετρικά, ασύγχρονα και controller-μεταγωγέα. Τα συμμετρικά μηνύματα χρησιμοποιούνται για την επίτευξη και διατήρηση της επικοινωνίας μεταξύ του controller και των μεταγωγέων, από και προς τις δύο πλευρές, ενώ χρησιμοποιούνται και για τον εντοπισμό πιθανών προβλημάτων που αντιμετωπίζουν οι συσκευές. Τα μηνύματα αυτά έχουν μορφή Hello, Echo, Error και Experimenter. Τα ασύγχρονα μηνύματα είναι αυτά που περιέχουν πληροφορίες σχετικές με την κατάσταση του μεταγωγέα ή των πινάκων flow που διαθέτει και των πακέτων που εισέρχονται σε αυτόν. Τα μηνύματα αυτά αποστέλλονται από τον μεταγωγέα προς τον controller και δεν είναι απαραίτητο να απαντήσει ο δεύτερος σε αυτά εκτός και αν το περιεχόμενο τους αποτελεί κάποιο αίτημα προς τον controller. Ένα παράδειγμα ασύμμετρου μηνύματος είναι το packet-in το οποίο εκπέμπεται σε περίπτωση που εισέλθει στον μεταγωγέα κάποιο πακέτο που δεν ξέρει πως να το διαχειριστεί. Τα μηνύματα controller-μεταγωγέα είναι αυτά που μεταδίδει ο controller ως εντολές προς τον μεταγωγέα. Τα μηνύματα αυτά χρησιμοποιούνται για να ελέγξει ο πρώτος την κατάσταση του δεύτερου και για να του μεταδώσει εντολές που αφορούν τους πίνακες OpenFlow, όπως, την κατεύθυνση που θα μεταφερθεί ένα πακέτο, εντολές παραμετροποίησης και αλλαγής κατάστασης όπως το άνοιγμα ή το κλείσιμο πορτών [25], [26] .

3.5.1 OpenFlow controller

Όπως προαναφέρθηκε η βασική αρχή της λειτουργίας της αρχιτεκτονικής δικτύων SDN είναι ο διαχωρισμός του επιπέδου ελέγχου και δεδομένων. Στο επίπεδο ελέγχου βρίσκεται ο controller και οι βασικές λειτουργίες ενός OpenFlow controller είναι οι ακόλουθες [27]:

1. Διατήρηση της γενικής εικόνας του δικτύου και της τοπολογίας του.
2. Λήψη αποφάσεων και διαχείριση του δικτύου.
3. Διαχείριση συμβάντων (Packet In κλπ).
4. Παροχή API για χρήση με τρίτες εφαρμογές.

3.5.2 OpenFlow switch

Στο φυσικό ή επίπεδο δεδομένων συναντάμε τους OpenFlow συμβατούς μεταγωγείς, οι οποίοι διαθέτουν τρία είδη πορτών. Στην πρώτη κατηγορία βρίσκονται οι φυσικές πόρτες μέσω των οποίων γίνεται η διασύνδεση μεταξύ των συσκευών και των χρηστών με τη χρήση καλωδίων από συνεστραμμένα ζεύγη και καλώδια οπτικών ινών. Στη δεύτερη κατηγορία βρίσκονται οι λογικές πόρτες οι οποίες δεν αντιστοιχούνται σε φυσικές αλλά έχουν να κάνουν με τη λογική επεξεργασία των πακέτων, όπως οι loopback, και στην τελευταία κατηγορία βρίσκονται οι πόρτες που χρησιμοποιούνται αποκλειστικά από το πρωτόκολλο OpenFlow και χωρίζονται σε υποχρεωτικές και προαιρετικές [24].

Οι πόρτες που χρησιμοποιούνται αποκλειστικά από το πρωτόκολλο OpenFlow στους μεταγωγείς χρησιμοποιούνται για την εκτέλεση βασικών ενεργειών. Οι ενέργειες αυτές χωρίζονται σε δύο κατηγορίες ανάλογα με τις δυνατότητες του κάθε μεταγωγέα, τις υποχρεωτικές και τις προαιρετικές. Στην κατηγορία με τις υποχρεωτικές βρίσκονται αυτές που είναι απαραίτητες για τη λειτουργία του μεταγωγέα και την επικοινωνία του με τον controller [24].

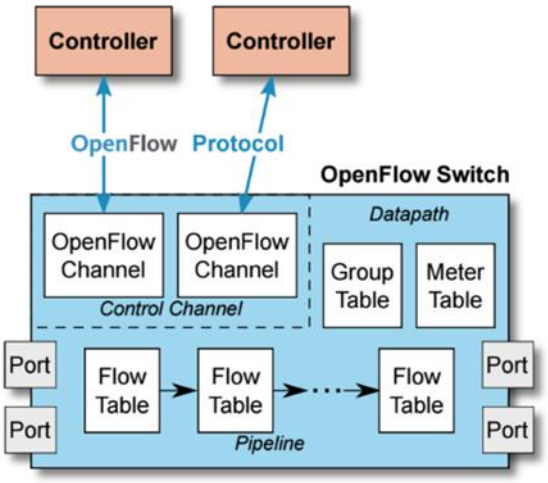
1. ALL: Δίνει στον μεταγωγέα την εντολή να εκπέμπει τα πακέτα προς όλες τις πόρτες του και μπορεί να παρομοιαστεί με τη λειτουργία ενός HUB.
2. CONTROLLER: Καθορίζει ότι μια πόρτα μπορεί μεταφέρει δεδομένα από ή προς τον controller.
3. TABLE: Χρησιμοποιείται κυρίως για την λειτουργία των πινάκων ροής του μεταγωγέα.
4. IN_PORT: είναι η πόρτα από την οποία λαμβάνεται ένα πακέτο.
5. ANY: Αντιπροσωπεύει οποιαδήποτε πόρτα του μεταγωγέα σε περίπτωση που δεν διευκρινίζεται κάποια.
6. UNSET: Αντιπροσωπεύει μια μη ορισμένη πόρτα και μπορεί να είναι εισόδου ή εξόδου.

Από την άλλη στην κατηγορία με τις προαιρετικές πόρτες βρίσκονται οι LOCAL, NORMAL και FLOOD [24]:

1. LOCAL: Πόρτα εντός του μεταγωγέα που χρησιμοποιείται για επικοινωνία με ξένες υπηρεσίες μέσω του OpenFlow δικτύου.
2. NORMAL: Χρησιμοποιείται για μετάδοση πακέτων σε δίκτυο που δεν υποστηρίζει OpenFlow.
3. FLOOD: Εκτελεί την ίδια λειτουργία μετάδοσης των πακέτων που χρησιμοποιούν τα μη OpenFlow δίκτυα, δηλαδή τη μετάδοση των πακέτων σε όλες τις ενεργές πόρτες ενός μεταγωγέα.

3.5.3 Πίνακες ροής

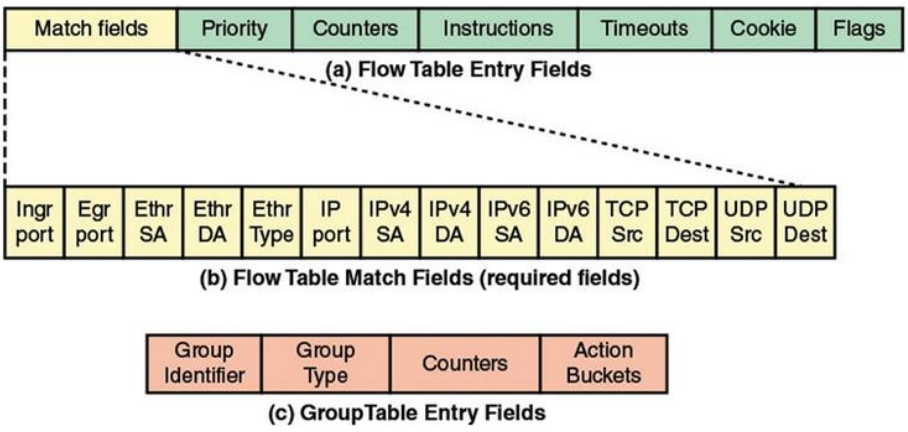
Για την δρομολόγηση των πακέτων ένας OpenFlow μεταγωγέας χρησιμοποιεί τους πίνακες ροής. Κατά την πρώτη εκκίνηση του ο μεταγωγέας δημιουργεί τους πίνακες αυτούς με σκοπό να αποθηκεύσει εκεί τις οδηγίες που θα λάβει από τον controller και αφορούν τα πακέτα που καλείται να μεταδώσει από τον αποστολέα προς τον παραλήπτη τους. Οι καταχωρήσεις των πινάκων αυτών χρησιμοποιούνται ώστε να μην είναι απαραίτητη η μετάδοση των πληροφοριών ενός πακέτου στον controller κάθε φορά για τη λήψη οδηγίας από αυτόν μειώνοντας παράλληλα τον φόρτο προς τον controller και τη μείωση της καθυστέρησης μετάδοσης.



Εικόνα 3.2: Πρωτόκολλο OpenFlow και πίνακες ροής

Η δημιουργία μιας καταχώρησης στον πίνακα ροής περιλαμβάνει πληροφορίες που καθορίζουν τον τρόπο που θα γίνει ο χειρισμός του κάθε πακέτου. Επιπλέον μια καταχώρηση του πίνακα μπορεί να έχει προκαθορισμένη διάρκεια ή προτεραιότητα στην διαδικασία χειρισμού ενός πακέτου. Παρακάτω φαίνονται οι πληροφορίες που μπορεί να διαθέτει μία καταχώρηση του πίνακα ροής ενός OpenFlow μεταγωγέα [24].

1. Match: Το πεδίο match καθορίζει τις προδιαγραφές που πρέπει να διαθέτει ένα πακέτο για να ταυτιστεί με τον κανόνα στον πίνακα.
2. Priority: Αύξον αριθμός που καθορίζει την προτεραιότητα εκτέλεσης της οντότητας.
3. Counters: Μεταβλητή που μετρά τα πακέτα που ταυτίστηκαν με την συγκεκριμένη καταχώρηση.
4. Instructions: Οδηγίες που καθορίζουν τον τρόπο που θα γίνει η διαχείριση του κάθε πακέτου.
5. Timeouts: Χρόνοι κατά τους οποίους θα έχει ισχύ η καταχώρηση του πίνακα.
6. Cookie: Μοναδικό αναγνωριστικό για την ταυτοποίηση του κανόνα.
7. Flags: Πληροφορίες που αφορούν τη διαχείριση του κανόνα στον πίνακα ροής.



Εικόνα 3.3: Δομή καταχώρησης πίνακα ροής

3.6 Controllers

Το βασικό χαρακτηριστικό ενός δικτύου καθορισμένο από λογισμικό είναι ο controller και βρίσκεται στο επίπεδο ελέγχου. Ο controller αναλαμβάνει την διαχείριση ολόκληρου του δικτύου λαμβάνοντας αποφάσεις σύμφωνα με την παραμετροποίηση και τις εφαρμογές που βρίσκονται εγκατεστημένες. Χωρίς τη χρήση του controller δεν μπορεί να υλοποιηθεί ένα δίκτυο SDN αφού ο συμβατός φυσικός εξοπλισμός, που βρίσκεται στο επίπεδο δεδομένων, δεν διαθέτει τη δυνατότητα αυτόνομης λειτουργίας σε σύγκριση με την παραδοσιακή αρχιτεκτονική [28]. Ένας controller είναι σημαντικό να σχεδιαστεί κατάλληλα ώστε να μπορεί να ανταπεξέλθει σε διάφορες συνθήκες ανάλογα με την χρήση του δικτύου και για το λόγο αυτό οι πιο γνωστοί ελεγκτές σχεδιάστηκαν με τις πιο κάτω προδιαγραφές [27]:

1. Συμβατότητα με λειτουργικά συστήματα: Ο controller αποτελεί λογισμικό το οποίο δεν μπορεί να εκτελεστεί απευθείας σε κάποιο μηχάνημα αλλά πρέπει να εγκατασταθεί σε κάποιο λειτουργικό σύστημα. Για το λόγο αυτό είναι αναγκαίο να μπορεί να εγκατασταθεί σε πληθώρα λειτουργικών συστημάτων όπως Linux και Windows.
2. Λογισμικό ανοιχτού κώδικα: Όταν ο controller είναι λογισμικό ανοιχτού κώδικα δίνει τη δυνατότητα στους προγραμματιστές να υλοποιούν ή να τροποποιούν τις εφαρμογές που τρέχουν στον controller ανάλογα με τις ανάγκες τους.
3. Οπτική διεπαφή χρήστη: Για την καλύτερη οργάνωση και διαχείριση του controller είναι σημαντικό να υπάρχει οπτική διεπαφή χρήστη αφού μπορεί να βοηθήσει στην πιο εύκολη εκπαίδευση και τη άμεση παραμετροποίηση του χωρίς να είναι απαραίτητη η χρήση περίπλοκων εντολών σε τερματικό.
4. Χρήση τυποποιημένων πρωτοκόλλων: Λόγω των πολλών κατασκευαστών και διαφορετικών ελεγκτών είναι σημαντικό να γίνεται χρήση πρωτοκόλλων που να υποστηρίζονται από όλους. Επιπλέον είναι σημαντική η χρήση τυποποιημένων πρωτοκόλλων ώστε να είναι εύκολη η παραμετροποίηση από τους χρήστες χωρίς να απαιτούνται επιπλέον γνώσεις για κάθε controller και εξοπλισμό που πιθανόν να χρησιμοποιηθεί.
5. Απόδοση: Ένα πολύ σημαντικό χαρακτηριστικό που πρέπει να έχει ο controller είναι η απόδοση ώστε να μπορεί να ανταπεξέλθει στις ανάγκες του δικτύου και των χρηστών. Για να θεωρείται ένας controller αποδοτικός θα πρέπει να μπορεί να ανταποκρίνεται άμεσα στα αιτήματα των OpenFlow συσκευών και να μπορεί να υποστηρίξει ένα δίκτυο οποιαδήποτε στιγμή.
6. Εγχειρίδιο χρήσης: Για να μπορέσει να γίνει η επιλογή ενός controller είναι απαραίτητο να υπάρχει ένα εγχειρίδιο χρήσης που να διαθέτει όλες τις απαραίτητες πληροφορίες και χαρακτηριστικά που διαθέτει. Ακόμα για να μπορέσει κάποιος να δημιουργήσει μία δική του εφαρμογή χρειάζεται τις απαραίτητες πληροφορίες που αφορούν τον τρόπο λειτουργίας του και τις ενσωματώσεις που ήδη διαθέτει.

Μέχρι σήμερα έχουν υλοποιηθεί αρκετοί ελεγκτές σε διάφορες γλώσσες προγραμματισμού με τις επικρατέστερες να είναι η Java , Python και C++ [27]. Πρώτος controller που σχεδιάστηκε σε γλώσσα Python, στα μέσα του 2000, είναι ο Ethane και αποτελεί τη βασική ιδέα για τη δημιουργία των σύγχρονων εκλεκτών.

3.6.1 POX

Ο controller POX είναι ένας από τους πρώτους και δημοφιλέστερους ελεγκτές που σχεδιάστηκαν για δίκτυα καθοριζόμενα από λογισμικό. Ο controller αυτός είναι η βελτιωμένη εκδοχή του NOX,

υλοποιημένος σε γλώσσα Python, είναι ανοικτού κώδικα, απλά δομημένος και τα βασικά του χαρακτηριστικά είναι τα ακόλουθα [29-31]:

1. Γλώσσα Python: Η Python αποτελεί μία από τις πιο γνωστές γλώσσες προγραμματισμού και αυτό καθιστά την χρήση του εύκολη, γρήγορη και αποτελεσματική.
2. Υποστήριξη OpenFlow: Η επικοινωνία του controller με τους μεταγωγείς και τους δρομολογητές είναι ένα σημαντικό και απαραίτητο κομμάτι σε ένα SDN δίκτυο και αυτό επιτυγχάνεται μέσω της Νότιας διεπαφής. Ο POX για την επικοινωνία του χρησιμοποιεί το πρωτόκολλο OpenFlow και υποστηρίζει την έκδοση 1.0 κάτι το οποίο είναι μειονέκτημα αφού η πιο πρόσφατη έκδοση του πρωτοκόλλου είναι η 1.5.1.
3. Προγραμματισιμότητα: Χάρη στη χρήση της γλώσσας Python οι προγραμματιστές μπορούν να επέμβουν στις λειτουργίες του controller παραμετροποιόντας τον κατάλληλα και σύμφωνα με τις ανάγκες τους. Επιπλέον δίνεται η δυνατότητα δημιουργίας εφαρμογών σε περίπτωση που οι απαιτούμενες δεν συμπεριλαμβάνονται στις εφαρμογές που διανέμονται με τον controller.
4. Προεγκατεστημένες εφαρμογές: Με τη διανομή του ο controller POX παρέχει στους χρήστες του τη δυνατότητα χρήσης έτοιμων εφαρμογών ώστε να μπορούν να τον χρησιμοποιήσουν άμεσα. Οι πιο γνωστές εφαρμογές που συμπεριλαμβάνει είναι HUB, Switch 2ου και 3ου επιπέδου και πολλές άλλες εφαρμογές που μπορούν να συνδυαστούν μεταξύ τους.
5. Εγχειρίδιο χρήσης και Κοινότητα χρηστών: Χάρη στην ύπαρξη εγχειρίδιου χρήσης οι νέοι χρήστες μπορούν να ξεκινήσουν να πειραματίζονται και να χρησιμοποιούν τον controller. Ακόμα χάρη στην κοινότητα που δημιούργησαν οι χρήστες του παρέχεται η δυνατότητα επίλυσης προβλημάτων με την συμβολή άλλων προγραμματιστών και χρηστών.

Πέρα από τη βασική δομή και τις εφαρμογές που διανέμονται με τον controller υπάρχει η δυνατότητα χρήσης τρίτων εφαρμογών που διευκολύνουν τη λειτουργία του όπως η διεπαφή χρήστη POXDesk, συλλέκτη στατιστικών, και Load Balancer.

3.6.2 Ryu

Ο RYU είναι ένας από τους πιο γνωστούς ελεγκτές που έχουν υλοποιηθεί και η διανομή του γίνεται δωρεάν σε μορφή ελεύθερου λογισμικού. Ο σχεδιασμός και η υλοποίηση του είναι σε γλώσσα προγραμματισμού Python. Επιπλέον έχει σχεδιαστεί ώστε να είναι επεκτάσιμος και υποστηρίζει τη χρήση διάφορων πρωτοκόλλων για την επικοινωνία του με τους μεταγωγείς ενώ για τη διαχείριση του οι μηχανικοί μπορούν να ανταλλάξουν πληροφορίες με τη χρήση αιτημάτων HTTP/HTPPS [29], [30]. Τα βασικά χαρακτηριστικά του controller RYU περιγράφονται πιο κάτω:

1. OpenFlow 1.5: Το OpenFlow είναι το βασικό πρωτόκολλο που χρησιμοποιείται για την επικοινωνία των ελεγκτών με τους μεταγωγείς και ο controller RYU υποστηρίζει μέχρι και την πιο πρόσφατη έκδοση του, την 1.5.
2. NETCONF: όπως και το OpenFlow χρησιμοποιείται για τη διαχείριση των δικτυακών συσκευών.
3. OF-Config: Χρησιμοποιείται από τον RYU για παραμετροποίηση των δικτυακών συσκευών όπως οι θύρες και οι διευθύνσεις IP.
4. Border gateway protocol: Το BGP χρησιμοποιείται για την επικοινωνία μεταξύ πολλών δρομολογητών μεταξύ τους και βρίσκονται σε διαφορετικά δίκτυα.

5. Rapid Spanning Tree Protocol: Πρωτόκολλο που χρησιμοποιείται για τη διαχείριση τοπολογιών και την αποφυγή ατέρμονων βρόγχων.

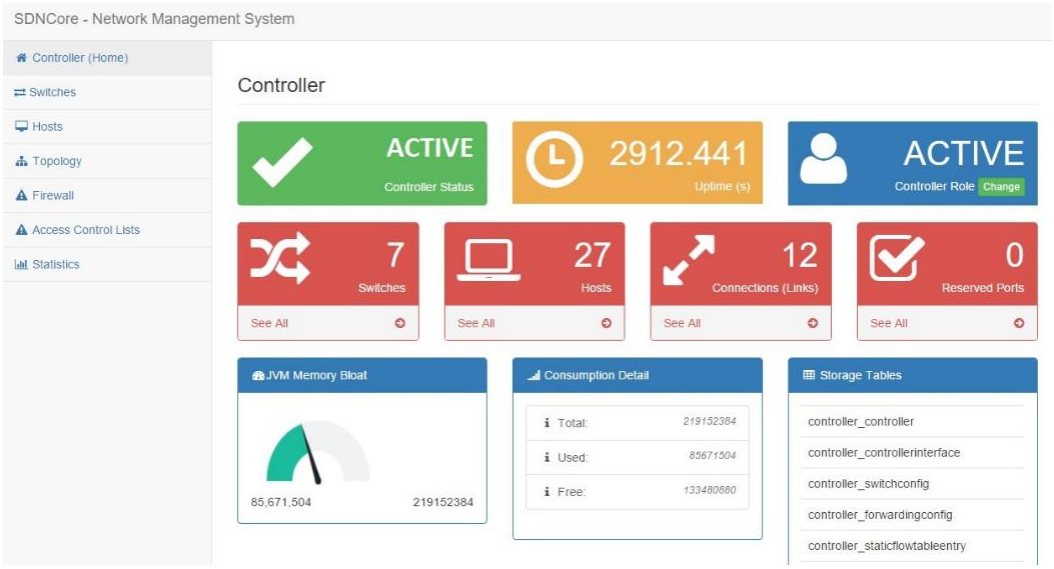
Πέρα από τα πρωτόκολλα που υποστηρίζει, ο RYU διαθέτει βιβλιοθήκες που μπορούν να χρησιμοποιηθούν για την υλοποίηση εφαρμογών όπως firewall, ανάλυσης κίνησης και υπηρεσιών Quality of service. Ακόμα με τη διανομή του παρέχονται εφαρμογές ώστε ο χρήστης να μπορέσει να τον θέσει σε λειτουργία άμεσα μετά την εγκατάσταση του ή ακόμα να τις χρησιμοποιήσει παράλληλα με δικές του, μια από αυτές είναι και η simple_switch_l3 που χρησιμοποιείται στις προσημειώσεις του 6ου Κεφαλαίου [30].

Πέρα από τις δικές του εφαρμογές ο RYU ενσωματώνει και εφαρμογές ανεξάρτητες όπως ο SNORT. Ο SNORT είναι ένας μηχανισμός Intrusion Prevention System (IPS) ο οποίος μπορεί να χρησιμοποιηθεί για τον εντοπισμό διαφόρων απειλών, όπως DDoS και Spoofing, ενώ η λειτουργία του βασίζεται σε κανόνες και για τη χρήση του απαιτείται χωριστή εγκατάσταση καθώς αποτελεί ξεχωριστό κομμάτι από τον controller και μπορεί να χρησιμοποιηθεί ανεξάρτητα [30].

3.6.3 Floodlight

Ο controller Floodlight είναι ένας από τους πιο γνωστούς ελεγκτές. Πρόκειται για ένα controller κατάλληλο για εκπαιδευτικούς σκοπούς αλλά και για επαγγελματικούς και διαθέτει τα εξής χαρακτηριστικά [31], [32]:

1. Γλώσσα Java: Ο controller Floodlight είναι γραμμένος σε γλώσσα Java, χαρακτηριστικό το οποίο τον κάνει πιο εύκολο στη χρήση αφού πρόκειται για μια γλώσσα η οποία είναι κατανοητή και εύκολη στη χρήση.
2. OpenFlow 1.5: Υποστηρίζει μέχρι και την έκδοση του πρωτοκόλλου OpenFlow 1.5 που τον καθιστά κατάλληλο για μεγάλο αριθμό εφαρμογών.
3. Ανοικτού κώδικα: Ο controller διανέμεται δωρεάν και ο χρήστης μπορεί να επέμβει στον κώδικα του ώστε να τον παραμετροποιήσει σύμφωνα με τις ανάγκες του. Επιπλέον ο χρήστης μπορεί να δημιουργήσει δικές του εφαρμογές για να του προσθέσει επιπλέον δυνατότητες.
4. Κοινότητα: όπως και οι υπόλοιπου ελεγκτές που προαναφέρθηκαν διαθέτει μεγάλη κοινότητα χρηστών που ανταλλάσσουν πληροφορίες και απορίες μεταξύ τους.
5. Διεπαφή χρήστη: Πρόκειται για ένα μεγάλο πλεονέκτημα συγκριτικά με τους υπόλοιπους ελεγκτές που προαναφέρθηκαν αφού χάρη στη διεπαφή χρήστη που διαθέτει η διαχείριση του γίνεται ακόμα πιο εύκολη.

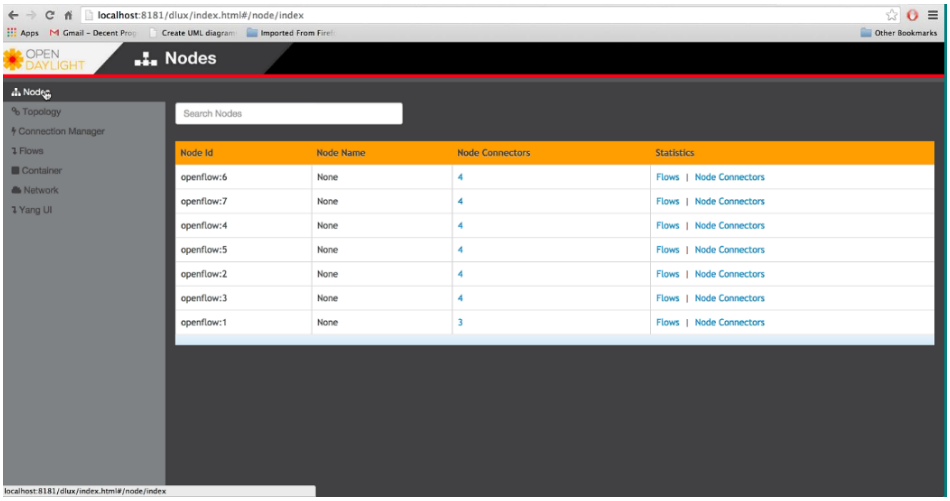


Εικόνα 3.4: Διεπαφή χρήστη Floodlight

3.6.4 OpenDaylight

Όπως οι υπόλοιποι ελεγκτές έτσι και ο OpenDaylight έχει σχεδιαστεί ώστε να μπορεί να χρησιμοποιηθεί σε μεγάλο αριθμό εφαρμογών. Με τα κύρια χαρακτηριστικά του να είναι τα εξής [33], [34], [35]:

1. Βασισμένος σε Java: Όπως και ο Floodlight είναι γραμμένος σε γλώσσα Java και μπορεί να χρησιμοποιηθεί εύκολα από τους περισσότερους χρήστες.
2. Πολλαπλά πρωτόκολλα: Υποστηρίζει αρκετά πρωτόκολλα για δίκτυα SDN χαρακτηριστικό που το δίνει τη δυνατότητα να συνδυαστεί με συσκευές που χρησιμοποιούν και άλλα πρωτόκολλά εκτός από το OpenFlow. Με τα πρωτόκολλα αυτά να είναι NETCONF, BGP και MPLS.
3. Εξειδικευμένες λειτουργίες: Διαθέτει ενσωματωμένες εξειδικευμένες λειτουργίες που τον καθιστούν κατάλληλο για χρήση σε απαιτητικά περιβάλλοντα όπως είναι τα Data centers.
4. Διεπαφή χρήστη: Διαθέτει διεπαφή χρήστη ώστε να γίνεται η διαχείριση του με περισσότερη ευκολία.



Εικόνα 3.5: Διεπαφή χρήστη Opendaylight

3.7 Επίλογος

Τα δίκτυα καθοριζόμενα από λογισμικό είναι μια πολύ υποσχόμενη αρχιτεκτονική αφού παρέχουν τη δυνατότητα στους μηχανικούς δικτύων να προσαρμόζουν το κάθε δίκτυο που υλοποιούν, ανάλογα με τις συνθήκες λειτουργίας του και τις ανάγκες των χρηστών του, χωρίς να απαιτείται παραμετροποίηση της κάθε συσκευής, χάρη στην κεντρικοποιημένη αρχιτεκτονική που υιοθετούν. Συγκριτικά με τα παραδοσιακής αρχιτεκτονικής δίκτυα, προσφέρουν τη δυνατότητα επαναπρογραμματισμού και αναβάθμισης παράλληλα με την ανάπτυξη νέων τεχνολογιών, όπως η τεχνητή νοημοσύνη και η μηχανική μάθηση. Επιπλέον τα πρωτόκολλα που χρησιμοποιούνται, όπως το OpenFlow, δίνουν τη δυνατότητα συλλογής δεδομένων που αφορούν τον τύπο και τον όγκο των δεδομένων. Τέλος, πολλά χαρακτηριστικά που διαθέτουν οι ελεγκτές είναι διαθέσιμα σε όλους, καθιστώντας τους όλους κατάλληλους για τις περισσότερες περιπτώσεις χρήσης. Κατά την υλοποίηση ενός δικτύου SDN ο μηχανικός δικτύων χρειάζεται να αξιολογήσει τις ανάγκες του υπό ανάπτυξη δικτύου, ώστε να επιλέξει τον πιο κατάλληλο controller.

Κεφάλαιο 4ο: Ασφάλεια στο SDN

4.1 Εισαγωγή

Όπως και στα παραδοσιακά δίκτυα οι μηχανικοί καλούνται να αντιμετωπίσουν πληθώρα απειλών που μπορούν να επηρεάσουν την λειτουργία του δικτύου ή των χρηστών του. Με τη χρήση των δικτύων παραδοσιακής αρχιτεκτονικής οι μηχανικοί περιορίζονται στην λήψη μέτρων ασφαλείας στο άκρο του δικτύου ή σε επίπεδο χρήστη, αφού οι μεταγωγείς και οι δρομολογητές δεν διαθέτουν από τη φύση τους προστασία από τις περισσότερες επιθέσεις. Πλέον με την ανάπτυξη των δικτύων SDN παρέχεται η δυνατότητα να εφαρμοστούν μέτρα προστασίας στο σύνολο του δικτύου, μέσω του controller και των εφαρμογών που εκτελούνται. Τα δίκτυα καθοριζόμενα από λογισμικό έρχονται αντιμέτωπα με νέου τύπου επιθέσεις, που δεν υπήρχαν στα παραδοσιακά δίκτυα, καθώς η αρχιτεκτονική τους είναι διαφορετική. Τα διάφορα επίπεδα των SDN δικτύων, και κυρίως ο controller, αποτελούν πλέον στόχο επιθέσεων για την διακοπή ή την παρεμβολή της λειτουργίας τους [36].

4.2 Προκλήσεις στην Ασφάλεια

Οι επιθέσεις δεν επηρεάζουν μόνο τη λειτουργία του δικτύου, αλλά και την ασφάλεια των χρηστών του, θέτοντας σε κίνδυνο τα προσωπικά δεδομένα τους. Σε πολλές περιπτώσεις οι επιθέσεις που έχουν ως στόχο υπηρεσίες ή οργανισμούς προκαλώντας τους προβλήματα τόσο στο τεχνικό αλλά και στο οικονομικό κομμάτι της λειτουργίας τους. Κατά τη χρήση των παραδοσιακών τεχνολογιών δικτύωσης οι επιθέσεις αυτές μπορούν να κατηγοριοποιηθούν σε τρεις βασικές κατηγορίες, πρωτοκόλλων, εξάντλησης πόρων και παραβίασης πρόσβασης.

1. Επιθέσεις πρωτοκόλλων: Η χρήση τυποποιημένων πρωτοκόλλων είναι απαραίτητη για την επικοινωνία μεταξύ συσκευών και υπηρεσιών αφού η ύπαρξη πολλών κατασκευαστών προκαλούσε προβλήματα συμβατότητας μεταξύ των προϊόντων τους. Σε ένα δίκτυο SDN πέρα από τα γνωστά πρωτόκολλα όπως τα ICMP, HTTP/S, DHCP και ARP που χρησιμοποιούνται στη λίστα προστίθεται και πρωτόκολλα που χρησιμοποιούνται αποκλειστικά από τα δίκτυα αυτά όπως είναι το Openflow το οποίο χρησιμοποιείται για την επικοινωνία μεταξύ του επίπεδου ελέγχου και δεδομένων όπως περιγράφεται στο κεφάλαιο 3 [37], [38].
2. Επιθέσεις εξάντλησης πόρων: Όλες οι ψηφιακές συσκευές που χρησιμοποιούνται καθημερινά όπως οι διακομιστές έτσι και τα SDN διαθέτουν συγκεκριμένους πόρους, όπως η μνήμη RAM και ο επεξεργαστής. Οι επιθέσεις εξάντλησης πόρων έχουν στόχο την πρόκληση άρνησης υπηρεσίας δηλαδή την αποτροπή μιας συσκευής να ανταποκριθεί στα αιτήματα που της υποβάλλονται. Οι πιο γνωστές επιθέσεις αυτού του τύπου είναι οι DoS/DDoS και μπορούν να εκτελεστούν με μεγάλη ευκολία αφού υπάρχουν πολλά εργαλεία σχεδιασμένα για αυτό το σκοπό [39].
3. Επιθέσεις παραβίασης πρόσβασης: Η λειτουργία ενός SDN δικτύου γίνεται απομακρυσμένα μέσω της Νότιας και της Βόρειας διεπαφής. Εξαιτίας της απομακρυσμένης πρόσβασης υπάρχει κίνδυνος παραβίασης τους παρέχοντας έτσι στον κακόβουλο χρήστη πλήρη πρόσβαση σε αυτά. Ο controller σε ένα δίκτυο SDN αποτελεί τον μεγαλύτερο στόχο αφού μέσω αυτού παρέχεται πρόσβαση σε ότι αφορά τη λειτουργία του δικτύου όπως οι πίνακες ροής και οι εφαρμογές. Για την παραβίαση ενός συστήματος υπάρχουν πολλοί τρόποι εκτέλεσης με το πιο γνωστό να είναι το Brute Furse το οποίο θα επεξηγηθεί στη συνέχεια.

Όπως προαναφέρθηκε, η ανάπτυξη των δικτύων SDN φέρνει νέες επιθέσεις που τα καθιστούν στόχο. Ο διαχωρισμός των επιπέδων ελέγχου και δεδομένων, καθώς και η προσθήκη του επίπεδου εφαρμογών αυξάνει τους πιθανούς στόχους επιθέσεων. Λόγω της αρχιτεκτονικής των δικτύων SDN, οι επιθέσεις μπορούν να κατηγοριοποιηθούν ανά επίπεδο, δηλαδή στο επίπεδο εφαρμογών, ελέγχου, δεδομένων, καθώς και τις δύο διεπαφές, τη Βόρεια και τη Νότια [40], [41].

1. Επίπεδο εφαρμογών: Στο επίπεδο εφαρμογών φιλοξενούνται όλες οι εφαρμογές που χρησιμοποιούνται για την λειτουργία, διαχείριση και παρακολούθηση του δικτύου. Το επίπεδο αυτό είναι αρκετά ευάλωτο σε επιθέσεις DoS/DDoS, υποκλοπή δεδομένων από μη κρυπτογραφημένη επικοινωνία ή ακόμα και από μολυσμένα πακέτα που πιθανόν να λάβουν.
2. Επίπεδο ελέγχου: Ο controller είναι το βασικό στοιχείο των δικτύων SDN και χωρίς αυτόν το δίκτυο δεν μπορεί να λειτουργήσει. Το επίπεδο ελέγχου είναι και αυτό ευάλωτο σε επιθέσεις όπως το επίπεδο εφαρμογών αφού είναι άμεσα συνδεδεμένα και επιπλέον πιθανή πρόσβαση σε δίνει τη δυνατότητα ελέγχου της συνολικής λειτουργίας του δικτύου.
3. Επίπεδο δεδομένων: Το επίπεδο δεδομένων είναι το πρώτο σημείο στο οποίο πραγματοποιείται μια επίθεση αφού σε αυτό έχουν πρόσβαση όλοι οι χρήστες του δικτύου. Πιθανή επίθεση σε αυτό μπορεί να θέσει σε κίνδυνο τη συνολική λειτουργία του δικτύου και των χρηστών του, καθώς σε αυτό αποθηκεύονται οι πίνακες ροής που χρησιμοποιούνται για πιο άμεση μεταφορά των πακέτων χωρίς την παρέμβαση του controller και πιθανή κακόβουλη τροποποίηση τους να επηρεάσει την ορθή και άμεση μεταφορά στον προορισμό τους.
4. Διεπαφές: Σε ένα δίκτυο SDN οι δύο διεπαφές είναι αναγκαίες για την επικοινωνία των τριών επιπέδων, του επιπέδου εφαρμογών, ελέγχου και δεδομένων. Οι διεπαφές για τη λειτουργία τους χρησιμοποιούν APIs που τους επιτρέπουν να δημιουργούν αιτήματα, να ανταποκρίνονται σε αυτά και να μεταφέρουν πληροφορίες από το ένα επίπεδο σε άλλο. Τα APIs είναι στόχοι εκμετάλλευσης ευπαθειών λογισμικού, DoS/DDoS, παρακολούθησης και υποκλοπής δεδομένων όπως κλειδιών και Tokens.

Με βάση τα παραπάνω, η λήψη μέτρων προστασίας είναι ένα σημαντικό κομμάτι που πρέπει να λάβουν υπόψη οι μηχανικοί δικτύων ώστε να παρέχουν ασφάλεια, εμπιστευτικότητα και διαθεσιμότητα κατά τη χρήση του δικτύου. Η προστασία του δικτύου και των χρηστών του μπορεί να επιτευχθεί με διάφορους τρόπους και τεχνικές, ανάλογα με το τι είναι καλύτερο για κάθε χρήση με τη δυνατότητα συνδυασμού τους. Οι μέθοδοι μπορούν να εφαρμοστούν για αποτροπή ή εντοπισμό των διάφορων απειλών με τη χρήση φυσικού εξοπλισμού, λογισμικού και εφαρμογών στα δίκτυα SDN ή και συνδυασμό τους.

Η αντιμετώπιση ή αποτροπή των διάφορων απειλών με βάση το δίκτυο μπορεί να επιτευχθεί με τη χρήση συστημάτων Firewall και IDS/IPS, που τοποθετούνται συνήθως στο άκρο του τοπικού δικτύου και βρίσκονται σε φυσική συσκευή από την οποία περνάνε όλα τα εισερχόμενα και εξερχόμενα πακέτα από το δίκτυο. Επιπλέον οι διάφοροι μεταγωγείς και δρομολογητές διαθέτουν κάποια ενσωματωμένα μέτρα προστασίας κυρίως από επιθέσεις που έχουν σκοπό να επηρεάσουν την δομή ή τη λειτουργία του δικτύου όπως Port Security, Access Control Lists και Dynamic ARP Inspection. Στα δίκτυα SDN παρέχεται η δυνατότητα εκτέλεσης των πιο πάνω με τη χρήση εφαρμογών είτε αυτές διανέμονται με τους ελεγκτές είτε έχουν υλοποιηθεί για τις ανάγκες της λειτουργίας του δικτύου.

Η χρήση λογισμικού για προστασία περιορίζεται για χρήση από τους χρήστες ή τους διακομιστές που βρίσκονται συνδεδεμένοι στο δίκτυο και οφείλεται στο ότι το λογισμικό εκτελείται τοπικά και μπορεί να προστατέψει μόνο τη συσκευή που βρίσκεται εγκατεστημένο. Αυτός ο περιορισμός ισχύ μόνο για τα παραδοσιακά δίκτυα, αφού η δυνατότητα χρήσης εφαρμογών στον controller του δικτύου επιτρέπει την εκτέλεση τους σε αυτόν, παρέχοντας προστασία στο σύνολο του δικτύου. Ακόμα η χρήση εφαρμογών ταυτοποίησης μπορεί να παρέχει προστασία από επιθέσεις που πραγματοποιούνται με αυτοματοποιημένο τρόπο όπως οι επιθέσεις Spoofing και DDoS.

4.3 Επιθέσεις στο SDN

Όπως αναφέρθηκε στο προηγούμενο κεφάλαιο υπάρχει μεγάλος αριθμός επιθέσεων που μπορούν να γίνουν με στόχο το δίκτυο ή τους χρήστες του. Παράλληλα αρκετές από τις επιθέσεις που μπορεί να στοχεύον κάποιο χρήστη ή διακομιστή υπάρχει πιθανότητα να επηρεάσουν και τη λειτουργία του δικτύου. Στη συνέχεια του κεφαλαίου περιγράφονται επιθέσεις που μπορούν να επηρεάσουν την λειτουργία ενός δικτύου, είτε είναι αυτό ο στόχος είτε όχι. Επιπρόσθετα, για την κάθε επίθεση περιγράφονται τρόποι πρόληψης ή αντιμετώπισης στα δίκτυα καθοριζόμενα από λογισμικό.

4.3.1 Επιθέσεις Denial of Service

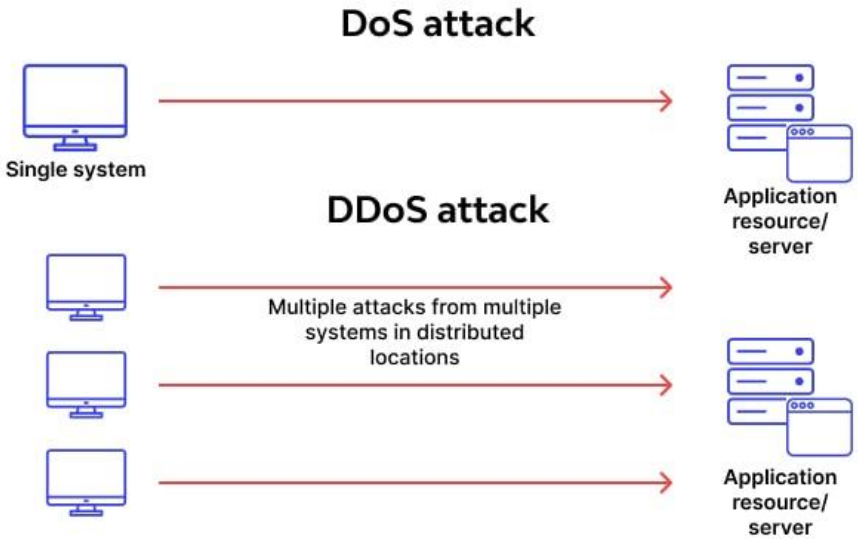
Οι επιθέσεις Denial of Service είναι πιθανότατα οι πιο γνωστές και συχνές που πραγματοποιούνται με στόχο τη διακοπή λειτουργίας ενός διακομιστή ή υπηρεσίας. Για την εκτέλεση των DoS επιθέσεων δεν είναι απαραίτητη η χρήση κάποιου εξειδικευμένου εξοπλισμού, αφού μπορεί να γίνει από οποιαδήποτε συσκευή είτε είναι εις γνώση του χειριστή της είτε όχι, αφού υπάρχει η δυνατότητα να εκτελεστεί με τη χρήση κακόβουλου λογισμικού απομακρυσμένα [42]. Στα SDN δίκτυα πιθανός στόχος γίνεται ο controller και το επίπεδο εφαρμογών, αφού μια πιθανή διακοπή της λειτουργίας τους καθιστά το δίκτυο μη λειτουργικό.

Οι επιθέσεις DoS μπορούν να χωριστούν σε δύο κύριες κατηγορίες, τις γρήγορες και τις αργές. Στις γρήγορες συναντάμε τις DoS και DDoS [42], με βασική διαφορά των δύο αυτών όμοιων επιθέσεων να είναι η μέθοδος εκτέλεσης τους. Από την άλλη, στην κατηγορία των αργών, βρίσκονται η Slowloris, Slow POST και Slow READ [43].

4.3.1.1 Γρήγορες επιθέσεις

Οι επιθέσεις DoS και DDoS μπορούν να επηρεάσουν όλα τα επίπεδα του SDN αφού ο μεγάλος αριθμός πακέτων που εκπέμπονται μπορεί να εξαντλήσει τους πόρους του κάθε επιπέδου ξεχωριστά, ανεξαρτήτως αν ο στόχος είναι άλλος. Η λειτουργία των επιθέσεων αυτών είναι αρκετά απλή αφού το μόνο που χρειάζεται είναι η μετάδοση μεγάλου πλήθους πακέτων, στο θύμα, σε βαθμό που να αδυνατεί να ανταποκριθεί σε αυτά. Σημαντικό να σημειωθεί είναι ότι η διάρκεια και το μέγεθος εκτέλεσης της επίθεσης διαφέρει σε κάθε περίπτωση αφού η δυνατότητα ανταπόκρισης του θύματος εξαρτάται από τους διαθέσιμους πόρους του.

Για την εκτέλεση της επίθεσης δεν απαιτούνται εξειδικευμένες γνώσεις αφού υπάρχουν διαθέσιμα εργαλεία ανοικτού κώδικα, όπως το HPing, που με τη χρήση ορισμάτων δίνονται όλες οι απαραίτητες πληροφορίες, όπως η διεύθυνση δικτύου του θύματος και ο όγκος των μεταδιδόμενων πακέτων. Η διαφορά των DoS και DDoS είναι η διαδικασία εκτέλεσης τους αφού στην πρώτη περίπτωση η εκτέλεση γίνεται από μία συσκευή και στην δεύτερη γίνεται κατανεμημένα, δηλαδή από πολλές διευθύνσεις και χρήστες ταυτόχρονα, που πιθανόν να έχουν μολυνθεί από κακόβουλο λογισμικό χωρίς την συγκατάθεση τους [44], [45].



Εικόνα 4.1: DoS vs DDoS attack

Ένα δίκτυο SDN μπορεί να επηρεαστεί από μια επίθεση άρνησης υπηρεσίας σε όλα του τα επίπεδα με το πρώτο να είναι αυτό των δεδομένων. Όπως και στα παραδοσιακά δίκτυα, όλες οι συσκευές που συνδέονται σε ένα δίκτυο διαθέτουν συγκεκριμένους πόρους, έτσι και στα δίκτυα SDN αν οι πόροι εξαντληθούν είναι δύσκολο να γίνει διαχείριση επιπλέον φόρτου. Όπως περιγράφηκε στο 3ο Κεφάλαιο, όταν ένας μεταγωγέας λάβει κάποιο πακέτο που δεν γνωρίζει πως να το διαχειριστεί το μεταδίδει στον controller ώστε να του δοθεί η απαραίτητη οδηγία για την μεταφορά προς τον προορισμό του και μέχρι να ενημερωθεί για το χειρισμό του τα δεδομένα του πακέτου αποθηκεύονται στην προσωρινή μνήμη του. Η επίθεση DoS/DDoS για την μείωση της πιθανότητας έγκαιρου εντοπισμού της μεταδίδει τα πακέτα με διαφορετικές διευθύνσεις πηγής με αποτέλεσμα ο μεγάλος αριθμός τους να γεμίζει την προσωρινή μνήμη του μεταγωγέα αδυνατώντας να εξυπηρετήσει νέα πιθανόν μη κακόβουλα πακέτα. Εκτός από την προσωρινή μνήμη του μεταγωγέα μία επίθεση άρνησης υπηρεσίας μπορεί να επηρεάσει και τους πίνακες ροής του γεμίζοντας τους αφού δεν διαθέτουν χώρο για απεριόριστες καταχωρήσεις [46].

Μια συνήθης συμπεριφορά ενός μεταγωγέα όταν γεμίσει η προσωρινή του μνήμη είναι να προωθεί όλα τα δεδομένα του πακέτου στον controller. Με αυτό τον τρόπο μία επίθεση άρνησής υπηρεσίας μπορεί να επηρεάσει τη λειτουργία του controller. Στην περίπτωση που το πλήθος αυτών των πακέτων είναι πολύ μεγάλο η επεξεργαστική ισχύς του controller δεσμεύεται για την επεξεργασία του πακέτου με αποτέλεσμα να επηρεάζεται η συνολική λειτουργία του δικτύου ή ακόμα και να διακόπτεται εντελώς [46].

Για την αποτροπή μία επίθεσης άρνησης υπηρεσίας μπορούν να εφαρμοστούν τεχνικές φιλτραρίσματος, επικάλυψης, Honeypots και Load balancing [47].

Οι τεχνικές φιλτραρίσματος βασίζονται στον έλεγχο και την επιβεβαίωση των διευθύνσεων δικτύου και από που προέρχονται μειώνοντας με αυτό τον τρόπο τις πιθανότητες εκτέλεσης μιας επιτυχημένης επίθεσης. Αυτού του είδους οι τεχνικές συνήθως βρίσκονται στους δρομολογητές, δηλαδή στην είσοδο ή την έξοδο του δικτύου. Επιπλέον, μπορεί να γίνει φιλτράρισμα βασισμένο στην διαδρομή που ακολούθησε το πακέτο, μεταξύ λίστας με πιθανές ύποπτες διευθύνσεις, ελέγχοντας τις διευθύνσεις από παλαιότερες επιθέσεις, με τη χρήση πρωτοκόλλων όπως το Source Address Validity Enforcement, με βάση τα Hops που έκανε το πακέτο και με βάση των ανωμαλιών στην κίνηση που παράγεται..

Η εφαρμογή της τεχνικής με αποκάλυψη γίνεται τοποθετώντας ένα δίκτυο μπροστά από το βασικό δημιουργώντας με αυτό τον τρόπο ένα τείχος προστασίας για τον πιθανό επιτιθέμενο αποτρέποντας τον να αποκτήσει πρόσβαση στο πραγματικό δίκτυο. Η τεχνική αυτή συνήθως χρησιμοποιείται για την προστασία του τοπικού δικτύου ενός οργανισμού όπου η πρόσβαση στις υπηρεσίες του γίνεται μόνο τοπικά.

Τα Honeypots λειτουργούν σαν δόλωμα ώστε η επίθεση να επηρεάσει αυτά και όχι την πραγματική υπηρεσία ή διακομιστή και παράλληλα παρέχεται η δυνατότητα συλλογής δεδομένων που παράγονται από την επίθεση.

Η χρήση Load Balancing χρησιμοποιείται για την ομοιόμορφη κατανομή του φόρτου σε δύο ή περισσότερα συστήματα που παρέχουν την ίδια υπηρεσία μειώνοντας έτσι τον φόρτο στους διακομιστές. Η μέθοδος αυτή δεν είναι απαραίτητο ότι χρησιμοποιείται μόνο για προστασία από επιθέσεις αλλά εφαρμόζεται και σε συστήματα με πολύ μεγάλο όγκο επισκεπτών.

Αντίθετα ο εντοπισμός μιας επίθεσης άρνησης υπηρεσίας αποτελεί πρόκληση αφού είναι απαραίτητο να γίνει άμεσα προτού επηρεαστεί η λειτουργία του δικτύου, της υπηρεσίας, ή και του διακομιστή. Μια μέθοδος που εφαρμόζεται είναι αυτή της εντροπίας και ανήκει στην κατηγορία εντοπισμού μέσω ανωμαλιών. Η μέθοδος της εντροπίας βασίζεται στην ανάλυση της εισερχόμενης κίνησης και εντοπισμού των ανωμαλιών σε πραγματικό χρόνο και βασίζεται στο πόσο τυχαίες είναι οι διευθύνσεις που εκπέμπουν τα πακέτα [44], [47]. Για τον υπολογισμό της εντροπίας γίνεται χρήση της φόρμουλας του Shannon όπως φαίνεται στον πιο κάτω μαθηματικό τύπο.

$$H(S_j) = -\sum_{i=1}^{N} p_i \log p_i \qquad (4.1)$$

Αρχικά για τον υπολογισμό της εντροπίας είναι απαραίτητη η συλλογή του συνολικού αριθμού των διευθύνσεων IP από τα εισερχόμενα πακέτα και ο υπολογισμός του πλήθους των μοναδικών διευθύνσεων δικτύου που έστειλαν αυτά τα πακέτα. Στη συνέχεια για τον υπολογισμό της εντροπίας γίνεται διαίρεση του πλήθους των μοναδικών διευθύνσεων IP με το πλήθος των εισερχόμενων πακέτων δηλαδή, ο μέσος όρος του πλήθους των πακέτων που μεταδίδει η κάθε διεύθυνση IP. Σημαντικό να αναφερθεί ότι ο χρήστης μπορεί να ορίσει το πλήθος των εισερχόμενων πακέτων που απαιτείται για την εκτέλεση του ελέγχου. Ακολούθως το αποτέλεσμα της διαίρεσης χρησιμοποιείται από τον τύπο υπολογισμού της εντροπίας που στη γλώσσα Python γίνεται με την ακόλουθη γραμμή κώδικα, όπου p είναι το αποτέλεσμα της διαίρεσης που έγινε νωρίτερα:

```
Entropy -= p * math.log2(p)
```

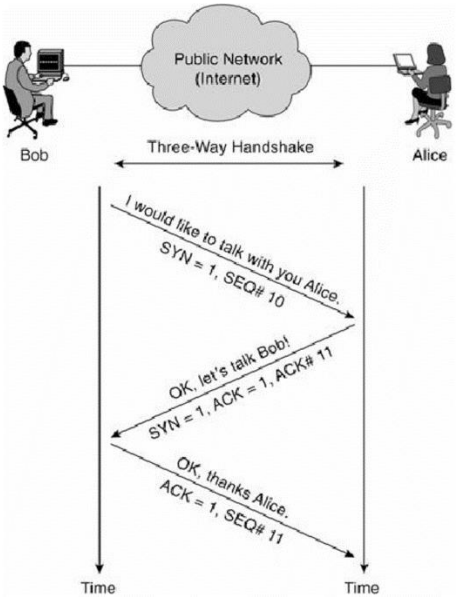
Με την εφαρμογή της μεθόδου με εντροπία ο διαχειριστής καθορίζει το όριο της εντροπίας όπου σε περίπτωση που το αποτέλεσμα της μαθηματικής πράξης είναι μεγαλύτερο από το καθορισμένο θεωρείται ότι εκτελείται επίθεση DDoS.

Τα δίκτυα SDN χάρη στην αρχιτεκτονική και τις δυνατότητες που προσφέρουν παρέχουν την δυνατότητα να αποτρέψουν μια επίθεση άρνησης υπηρεσίας όχι μόνο με τις μεθόδους που προαναφέρθηκαν αλλά και με πολλές άλλες. Οι δυνατότητες δημιουργίας και εκτέλεσης εφαρμογών δίνει τη δυνατότητα στους μηχανικούς και τους προγραμματιστές να υλοποιούν και να πειραματίζονται με πιο αποδοτικές και σύγχρονες μεθόδους όπως η χρήση της τεχνητής νοημοσύνης και της μηχανικής μάθησης [45].

4.3.1.2 Αργές επιθέσεις

Οι αργές επιθέσεις DoS δεν στέλνουν μεγάλο αριθμό πακέτων ταυτόχρονα, όπως οι γρήγορες επιθέσεις. Αντίθετα, κρατούν πολλές συνδέσεις ανοιχτές για μεγάλο χρονικό διάστημα, με σκοπό την εξάντληση των διαθέσιμων πόρων ενός διακομιστή σε επίπεδο πρωτοκόλλου. Οι επιθέσεις αργού τύπου είναι τρεις, η Slowloris, slow POST και slow Read [43].

Η επίθεση Slowloris εκμεταλλεύεται τη διαδικασία χαιρετισμού τριών σταδίων (Three way handshake) που εκτελείται κατά την επικοινωνία ενός χρήστη με κάποιο διακομιστή, όπου κατά την οποία ξεκινά και στη συνέχεια ολοκληρώνει μια σύνδεση όπως φαίνεται στην εικόνα [43].



Εικόνα 4.2: Εκτέλεση Three way handshake

Αρχικά, ο πρώτος χρήστης είναι αυτός που επικοινωνεί αιτούμενος την ανταλλαγή δεδομένων με τη χρήση πακέτου συγχρονισμού, SYN (Synchronize). Εφόσον ο δεύτερος χρήστης είναι διαθέσιμος απαντάει θετικά, με πακέτο τύπου ACK (Acknowledgement), και εφόσον ολοκληρωθεί με επιτυχία η επικοινωνία και η ανταλλαγή δεδομένων μεταξύ τους, ο πρώτος χρήστης ενημερώνει το δεύτερο ότι έλαβε τα δεδομένα με επιτυχία. Με την εκτέλεση της επίθεσης Slowloris ο κακόβουλος χρήστης ανοίγει μεγάλο αριθμό ταυτόχρονων συνδέσεων χωρίς να τις ολοκληρώνει, με αποτέλεσμα να εξαντλούνται οι διαθέσιμες συνδέσεις TCP ενός διακομιστή αποτρέποντας τον να εξυπηρετήσει άλλους χρήστες. Σε ένα δίκτυο SDN, η επίθεση Slowloris δεν μπορεί να το επηρεάσει άμεσα, αφού η λειτουργία του ολοκληρώνεται με την άφιξη των πακέτων στον προορισμό τους. Από την άλλη η επίθεση αυτή μπορεί να επηρεάσει το επίπεδο εφαρμογών του δικτύου, ανοίγοντας μεγάλο αριθμό συνδέσεων στον διακομιστή που το φιλοξενεί, με αποτέλεσμα να θέσει σε κίνδυνο την ορθή και αξιόπιστη λειτουργία του δικτύου.

Όπως και στις Fast DoS επιθέσεις ο εντοπισμός μπορεί να γίνει με βάση το δίκτυο ή και τον διακομιστή. Από πλευράς του διακομιστή μπορεί να γίνει χρήση λογισμικού Firewall το οποίο αναλύει τα εισερχόμενα πακέτα για πιθανά μη ολοκληρωμένες συνδέσεις, ενώ παράλληλα μπορούν να εφαρμοστούν όρια στο πλήθος των αιτημάτων ανά χρήστη ή διεύθυνση δικτύου που επιτρέπονται και ορισμός μέγιστου χρόνου για ανοιχτές συνδέσεις. Παράλληλα παρέχεται η δυνατότητα επικοινωνίας με τον διακομιστή από συγκεκριμένες διευθύνσεις με αποτέλεσμα οι χρήστες που δεν θεωρούνται αξιόπιστοι να μην μπορούν να επικοινωνήσουν με αυτόν, μια τεχνική που μπορεί να χρησιμοποιηθεί

στον controller ενός δικτύου SDN όπου δεν απαιτείται η επικοινωνία απευθείας με τους χρήστες πέρα από τους διαχειριστές του.

Παράλληλα στις επιθέσεις Slow POST και Slow Read γίνεται εκμετάλλευση των ευπαθειών των πρωτοκόλλων στέλνοντας ή λαμβάνοντας δεδομένα με πολύ αργό ρυθμό θέτοντας τον διακομιστή σε κατάσταση αναμονής ολοκλήρωσης της μεταφοράς των δεδομένων. Η αναμονή που προκαλείται οδηγεί σε κατανάλωση διαθέσιμων πόρων του διακομιστή αποτρέποντας τον να επεξεργαστεί νέα εισερχόμενα αιτήματα. Για προστασία από τις επιθέσεις αυτές είναι σημαντικό κατά την υλοποίηση και το σχεδιασμό του διακομιστή να εφαρμοστούν τεχνικές ανάλυσης δεδομένων και κίνησης και παράλληλα περιορισμός μέγιστου χρόνου μετάδοσης ή λήψης [43].

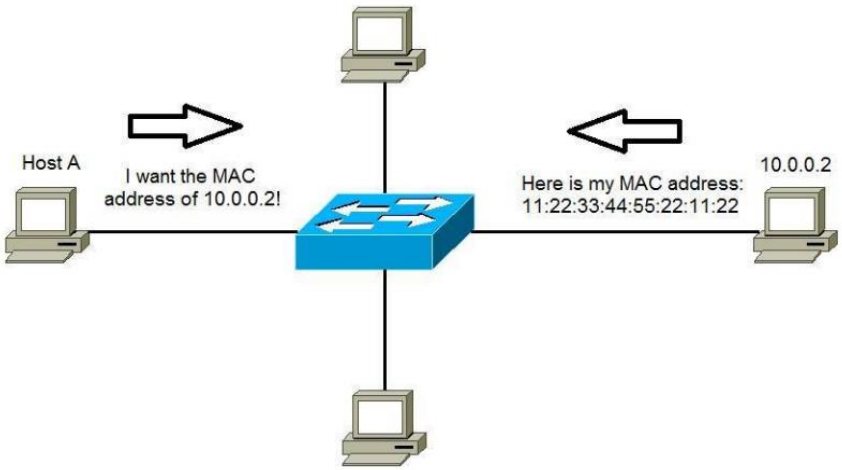
Πέρα από τη χρήση μεθόδων που εφαρμόζονται στο επίπεδο εφαρμογής του controller σε ένα δίκτυο SDN δίνεται η δυνατότητα διασύνδεσης με σύστημα Intrusion Detection System/ Intrusion Prevention System όπου ο controller μεταδίδει σε αυτό μετρήσεις και δεδομένα που αφορούν την κίνηση στο δίκτυο και με βάση κανόνων μπορεί να εντοπίσει πιθανές επιθέσεις τύπου DoS.

4.3.2 Επιθέσεις ARP

Η ανάπτυξη των μεταγωγών δικτύου (network switches) που αντικατέστησαν τους κόμβους (Hubs) διαφοροποιεί τον τρόπο που μεταδίδονται τα πακέτα από τον αποστολέα προς τον παραλήπτη και η χρήση τους βελτίωσε την απόδοση των δικτύων αφού τα πακέτα δεν μεταδίδονται προς όλες τις ενεργές πόρτες αλλά μόνο προς την πόρτα που βρίσκεται ο παραλήπτης, και έφερε την ανάγκη δημιουργίας του πρωτοκόλλου ARP καθώς οι χρήστες του δικτύου χρησιμοποιούν τις διευθύνσεις IP σε αντίθεση με τους μεταγωγείς που χρησιμοποιούν διευθύνσεις MAC.

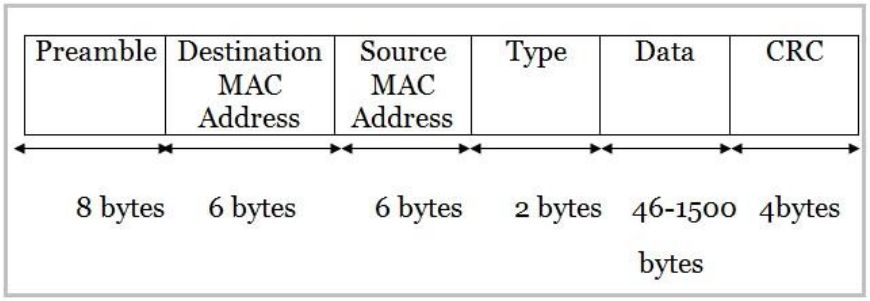
Το πρωτόκολλο ARP σχεδιάστηκε ώστε να γίνεται αντιστοίχιση των διευθύνσεων MAC και IP σε μορφή πίνακα. Με βάση τα παραπάνω κάποιος μπορεί να καταλάβει ότι επιθέσεις αυτού του είδους έχουν στόχο το επίπεδο δεδομένων ενός δικτύου SDN και τους συνδεδεμένους σε αυτό χρήστες. Ο όρος ARP αντιπροσωπεύει το Address Resolution Protocol και η λειτουργία του πρωτοκόλλου αυτού είναι ανταλλαγή πληροφοριών μεταξύ δύο συσκευών για την γνωστοποίηση της φυσικής διεύθυνσης (MAC Address) και η αποθήκευση της σε πίνακα με αντιστοίχιση της διεύθυνσης δικτύου (IP) [48], [49].

Η λειτουργία του ARP είναι πολύ απλή αλλά σημαντική για ένα δίκτυο καθώς η χρήση του μείωσε σημαντικά την κίνηση μέσα στο δίκτυο εξαλείφοντας για την ανάγκη την μετάδοση των πακέτων σε όλους τους χρήστες (broadcast) για την επιτυχή αποστολή τους προς τον παραλήπτη. Πέρα από την μείωση της κίνησης στο δίκτυο προστατεύει τα προσωπικά δεδομένα των χρηστών από πιθανούς κακόβουλους χρήστες, δηλαδή την παρακολούθηση των πακέτων (Packet sniffing) αφού μόνο ο πραγματικός αποδέκτης του πακέτου το λαμβάνει.



Εικόνα 4.3: Εκτέλεση πρωτοκόλλου ARP

Για την αποστολή ενός πακέτου από ένα χρήστη προς ένα άλλο είναι απαραίτητο να συμπεριληφθεί στις πληροφορίες που το συνοδεύουν η διεύθυνση MAC του παραλήπτη ώστε να μπορέσει ο μεταγωγέας να το δρομολογήσει. Αν ο αποστολέας του πακέτου δεν γνωρίζει τη φυσική διεύθυνση του παραλήπτη εκπέμπει ένα πακέτο τύπου ARP request. Το πακέτο αιτήματος μεταδίδεται σε μορφή broadcast ώστε να φτάσει σε όλους τους χρήστες που βρίσκονται συνδεδεμένοι στο δίκτυο και ανταποκρίνεται μόνο ο χρήστης που του ανήκει η διεύθυνση δικτύου, που περιλαμβάνει το αίτημα. Η διαδικασία εκτέλεσης του πρωτοκόλλου ARP κρύβει κινδύνους αφού μπορεί να το μεταδώσει και να το ακούσει οποιοσδήποτε χρήστης [48].

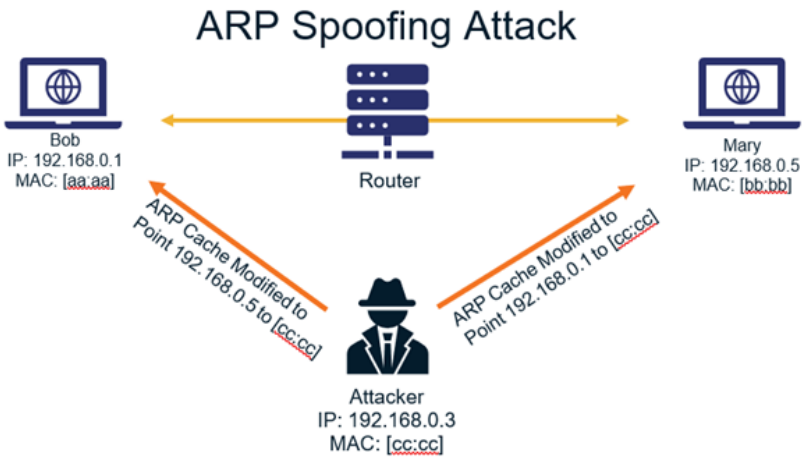


Εικόνα 4.4: Δομή πακέτου ARP

Το πρωτόκολλο ARP λόγω του απλού σχεδιασμού του δεν διαθέτει μέτρα προστασίας, κάτι το οποίο θέτει τους χρήστες σε κίνδυνο από διάφορες επιθέσεις. Για την προστασία των χρηστών είναι απαραίτητο να λαμβάνονται κάποια μέτρα προστασίας, όχι μόνο από τους ίδιους αλλά και από πλευράς του δικτύου. Στα παραδοσιακά δίκτυα η μόνη μέθοδος προστασίας είναι η χρήστη του DAI (Dynamic ARP Inspection). Το DAI σχεδιάστηκε με σκοπό την προστασία των χρηστών επιβεβαιώνοντας την εγκυρότητα των ARP πακέτων [50].

4.3.3 ARP Spoofing

Η επίθεση ARP Spoofing είναι γνωστή και ως ARP Poisoning, ενώ σκοπός της είναι η τροποποίηση του πίνακα ARP του θύματος. Κατά την υλοποίηση της επίθεσης ο επιτιθέμενος μεταδίδει ψευδή πακέτα τύπου ARP reply ώστε να τροποποιήσει τις καταχωρήσεις που χρησιμοποιεί το θύμα με αποτέλεσμα να μην φτάνουν στον πραγματικό παραλήπτη τους [49], [50], [51]. Η εκτέλεση της επίθεσης μπορεί να πραγματοποιηθεί με εργαλεία που βρίσκονται διαθέσιμα σε μορφή ελεύθερου λογισμικού και είναι απλά στη χρήση τους όπως το arpspoof.



Εικόνα 4.5: Επίθεση ARP Spoofing

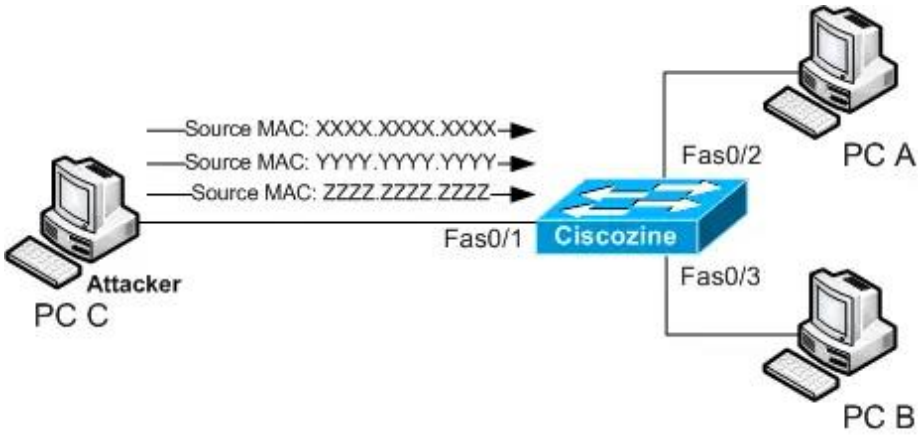
Για την προστασία από αυτού το είδους επιθέσεις υπάρχουν δύο τεχνικές που μπορούν να εφαρμοστούν, η αποτροπή (ARP Poisoning prevention) και ο εντοπισμός (ARP Poisoning detection).

Η αποτροπή μιας επίθεσης ARP Spoofing μπορεί να επιτευχθεί με την κατάλληλη τροποποίηση του πρωτοκόλλου ώστε να μην επιτρέπει την παραποίηση των πληροφοριών του, μέθοδος που απαιτεί την εξ ολοκλήρου ανάπτυξη του [52], [53], [54]. Από την άλλη στα δίκτυα SDN παρέχεται η δυνατότητα ελέγχου των πακέτων προτού φτάσουν στον προορισμό τους με τη χρήση εφαρμογής όπως παρουσιάζεται στις προσομοιώσεις του 6ου Κεφαλαίου.

Αντίθετα ο εντοπισμός της επίθεσης κρύβει κάποιους κινδύνους αφού τα πακέτα ARP μπορεί να φτάσουν στον προορισμό τους προτού ληφθούν μέτρα για την αντιμετώπισης της. Όπως και με την μέθοδο της αποτροπής έτσι και σε αυτή την περίπτωση σε ένα δίκτυο SDN μπορεί να γίνει υλοποίηση εφαρμογών που να εντοπίζουν τις επιθέσεις [53], [54].

4.3.4 MAC Flooding

Εκτός από την επίθεση προς κάποιο χρήστη που βρίσκεται εντός του δικτύου μπορεί να γίνει και εναντίων ολόκληρου του δικτύου, πλημμυρίζοντας το με ψευδής καταχωρήσεις. Αλλιώς η επίθεση αυτή ονομάζεται και ARP Cache Poisoning. Σκοπός της επίθεσης MAC flooding είναι να γεμίσει ο πίνακας που βρίσκονται οι αντιστοιχίσεις των φυσικών διευθύνσεων με τις πόρτες του μεταγωγέα. Πολλές δικτυακές συσκευές στην περίπτωση που ο πίνακας με τις ARP καταχωρήσεις γεμίσει μπαίνουν σε κατάσταση κόμβου (hub) αφού αδυνατούν πλέον να ανταποκριθούν στον μεγάλο αριθμό τους. Στο σημείο αυτό καλό είναι να σημειωθεί ότι σε περίπτωση που ο πίνακας MAC φτάσει στο μέγιστο της χωρητικότητας του αντικαθιστά τις παλαιότερες καταχωρήσεις με τις νεότερες. Στην περίπτωση που ο μεταγωγέας μπει σε κατάσταση κόμβου μεταδίδει πλέον όλα τα πακέτα που λαμβάνει προς όλες τις ενεργές πόρτες του με αποτέλεσμα ο επιτιθέμενος να λαμβάνει όλα τα πακέτα που μεταδίδονται μέσα στο δίκτυο δίνοντας του τη δυνατότητα να εντοπίσει πιθανές μη κρυπτογραφημένες πληροφορίες όπως κωδικούς πρόσβασης.



Εικόνα 4.6: Επίθεση MAC Flooding

Σε ένα δημόσιο δίκτυο είναι σημαντικό να λαμβάνονται μέτρα προστασίας από τέτοιου είδους επιθέσεις, αφού δεν είναι γνωστό ποιοι χρήστες θα το χρησιμοποιήσουν. Στα δημόσια παραδοσιακά δίκτυα η λύση για την προστασία των χρηστών είναι το Port Security, το οποίο εφαρμόζει περιορισμούς στον αριθμό των χρηστών που μπορούν να συνδεθούν στο δίκτυο από μία πόρτα και αποτρέπει την επικοινωνία των χρηστών με αυτούς που είναι συνδεδεμένοι από άλλες. Στα SDN δίκτυα μπορεί να υλοποιηθεί αντίστοιχη λύση με το Port Security, που να διαθέτει περισσότερες δυνατότητες και να διαθέτει μηχανισμούς ταυτοποίησης των χρηστών που επιχειρούν να αποκτήσουν πρόσβαση. Επιπλέον με τη χρήση μεθόδων μηχανικής μάθησης και τεχνητής νοημοσύνης παρέχεται η δυνατότητα ο έλεγχος να γίνεται με δυναμικό τρόπο όπως για παράδειγμα τα πακέτα ARP Reply που μεταδίδονται να γίνονται αποδεκτά μόνο μετά από αίτημα ARP Request.

4.3.5 ARP Denial of Service (ARP DDoS)

Αντίθετα από την επίθεση DDoS που περιγράφηκε σε μία επίθεση ARP Denial of Service ο επιτιθέμενος εκπέμπει στο θύμα μεγάλο αριθμό πακέτων ARP Request δημιουργώντας μεγάλη κίνηση στο δίκτυο και στο θύμα με αποτέλεσμα να μην μπορεί πλέον να ανταποκριθεί σε οποιαδήποτε άλλη διεργασία καταναλώνοντας όλους τους πόρους του, προσπαθώντας να ανταποκριθεί στα αιτήματα αυτά ή ακόμα να τον καταστήσει μη λειτουργικό. Κατά την εκτέλεση της επίθεσης σε ένα δίκτυο SDN υπάρχει και μεγάλος κίνδυνος να επηρεαστεί η συνολική λειτουργία του δικτύου, αφού αν το πλήθος των πακέτων είναι πολύ μεγάλο από αυτό που μπορεί να εξυπηρετήσει ο controller υπερφορτώνεται και δεν μπορεί πλέον να ανταποκριθεί ειδικά σε περιπτώσεις που εκτελούνται εφαρμογές που διαχειρίζονται δεδομένα του πρωτοκόλλου ARP [55], [56].

Για την αντιμετώπιση της σε ένα SDN δίκτυο πιθανός τρόπος είναι να περιορίζεται το πλήθος των πακέτων ARP request σε βαθμό που να βρίσκεται σε πλαίσια φυσιολογικής κίνησης, καθώς η χρήση του πρωτοκόλλου γενικότερα γίνεται μόνο όταν απαιτείται για μικρό χρονικό διάστημα με μικρό αριθμό πακέτων. Ακόμα για την αντιμετώπιση μιας επίθεσης DDoS ARP μπορεί να χρησιμοποιηθεί εντοπισμός με εντροπία όπως και στην κλασική χρήση της επίθεσης άρνησης υπηρεσίας όπου με βάσει την συνολική κίνηση και το προκαθορισμένο όριο μέγιστου πλήθους πακέτων γίνεται εντοπισμός της επίθεσης. Αφού γίνει εντοπισμός της επίθεσης η αντιμετώπιση μπορεί να γίνει είτε με αποκλεισμό του χρήστη ή με περιορισμό της κίνησης που του επιτρέπεται να μεταδώσει [44], [57].

4.3.6 ARP Broadcast storm

Η επίθεση ARP Broadcast storm εκτελείται με στόχο την υπερφόρτωση ενός δικτύου δημιουργώντας μεγάλο όγκο πακέτων τύπου ARP, οδηγώντας ακόμα και στην κατάρρευση του. Κατά την εκτέλεση της

επίθεσης ο κακόβουλος χρήστης εκπέμπει πακέτα τύπου ARP προς όλους τους χρήστες του δικτύου, με broadcast μορφή, δηλαδή στην φυσική διεύθυνση του παραλήπτη χρησιμοποιεί τη MAC FF:FF:FF:FF:FF:FF η οποία είναι τυποποιημένη για την αποστολή πακέτων προς όλους τους ενεργούς χρήστες του δικτύου. Στην περίπτωση που σε ένα δίκτυο SDN τα πακέτα αυτά τυγχάνουν επεξεργασίας από τον controller μπορεί να δημιουργήσει αρκετά προβλήματα, όπως η υπερφόρτωση και η μείωση της απόδοσης του, καθυστερώντας την επεξεργασία άλλων πληροφοριών ή ακόμα και τη διακοπή λειτουργίας του. Επιπλέον σε κίνδυνο βρίσκονται και οι μεταγωγείς αφού σε πολλές περιπτώσεις διαθέτουν πίνακες ARP για την δρομολόγηση των πακέτων γεμίζοντας τους πίνακες με αποτέλεσμα να καθυστερεί η μεταφορά των πακέτων μεταξύ των χρηστών [58], [54].

Για εντοπισμό μιας οποιασδήποτε απόθεσης τύπου Broadcast Storm είναι απαραίτητη η συλλογή δεδομένων όπως το πλήθος των μεταδιδόμενων πακέτων σε μορφή broadcast που μεταδίδουν οι χρήστες. Αφού εντοπιστεί μία επίθεση είναι σημαντικό να αντιμετωπιστεί άμεσα προτού προκαλέσει προβλήματα στο δίκτυο ή τους χρήστες του. Η αντιμετώπιση της μπορεί να γίνει με τον ορισμό μέγιστου πλήθους επιτρεπτών πακέτων ARP στους μεταγωγείς ανά χρήστη ή ενεργοποίηση του Dynamic ARP Inspection (DAI) [58] . Ακόμα με τη χρήση των VLAN το δίκτυο μπορεί να χωριστεί σε τμήματα ώστε να μην υπάρχει πρόσβαση μεταξύ των χρηστών που ανήκουν σε διαφορετικό υποδίκτυο μειώνοντας έτσι το πλήθος των πακέτων Broadcast.

4.3.7 Επίθεση παραβίασης

Μια επίθεση σε ένα δίκτυο δεν είναι απαραίτητο να έχει ως στόχο μόνο τους χρήστες του αλλά τον εξοπλισμό που το απαρτίζει με σκοπό την απόκτηση πρόσβασης σε αυτόν. Επιθέσεις για την απόκτηση μη εξουσιοδοτημένης πρόσβασης μπορούν να γίνουν σε όλα τα επίπεδα ενός δικτύου SDN. Μία τέτοιου είδους επίθεση μπορεί να γίνει σε ένα δίκτυο μεγάλης σημασίας όπως ενός πάροχου ή εκπαιδευτικού ιδρύματος όπου μπορεί να προκαλέσει σοβαρά προβλήματα [59].

Αποκτώντας πρόσβαση στο επίπεδο εφαρμογών ενός SDN δικτύου δίνεται η δυνατότητα στον επιτιθέμενο να διακόψει, να τροποποιήσει ή να εκτελέσει δικές του εφαρμογές. Με τη διακοπή των εφαρμογών η λειτουργία του δικτύου μπορεί να σταματήσει εντελώς και κατά συνέπεια να διακοπεί η παροχή υπηρεσιών προς τους χρήστες του. Ακόμα με την τροποποίηση των παραμέτρων που εξαρτώνται οι εφαρμογές ή την εκτέλεση ξένων εφαρμογών δίνεται η δυνατότητα υποκλοπής ευαίσθητων πληροφορίες που αφορούν τους χρήστες και τα δεδομένα που ανταλλάσσουν μεταξύ τους.

Ο controller αποτελεί τον κυριότερο στόχο για την εκτέλεση μιας επίθεσης καθώς από αυτόν εξαρτάται η λειτουργία του δικτύου και με την πρόσβαση στον controller δίνεται η δυνατότητα διαχείρισης ολόκληρου του δικτύου.

Το πρώτο στοιχείο ενός δικτύου που καλείται να αντιμετωπίσει μια επίθεση είναι αυτό στο οποίο είναι συνδεδεμένοι οι χρήστες. Το στοιχείο αυτό είναι οι μεταγωγείς όπου πιθανή απόκτηση πρόσβασης σε αυτούς δίνεται η δυνατότητα επέκτασης μιας επίθεσης στα ανώτερα μέρη του δικτύου όπως ο controller.

Από την άλλη οι διεπαφές είναι αυτές που καλούνται να μεταφέρουν πληροφορίες μεταξύ των επιπέδων του δικτύου. Η Βόρεια διεπαφή καλείται να μεταφέρει πληροφορίες μεταξύ των επιπέδων εφαρμογής και ελέγχου ενώ πιθανή παραβίαση της δίνει τη δυνατότητα δημιουργίας ψεύτικων πληροφοριών ή τροποποίηση των πραγματικών. Παράλληλα η Νότια διεπαφή αποτελεί δίαυλο επικοινωνίας μεταξύ των επίπεδων ελέγχου και δεδομένων. Με την παραβίαση τα Νότιας διεπαφής δίνεται η δυνατότητα παρακολούθησης των πακέτων που δεν υπάρχει κανόνας ροής, δημιουργία κακόβουλων κανόνων ροής και τροποποίηση των υφιστάμενων.

Μία επίθεση παραβίασης μπορεί να γίνει με τις ακόλουθες τεχνικές:

1. Brute Force: Κατά την τεχνική αυτή ο θύτης με τη χρήση κατάλληλων εργαλείων μπορεί να αποκτήσει πρόσβαση σε ένα controller ¨μαντεύοντας¨ τους κωδικούς πρόσβασης δοκιμάζοντας διάφορους πιθανούς συνδυασμούς. Η διαδικασία αυτή μπορεί να διαρκέσει μερικά λεπτά μέχρι και χρόνια ανάλογα με το μέγεθος και την πολυπλοκότητα του κωδικού.
2. Μη κρυπτογραφημένοι κωδικοί: Σε περίπτωση Man in the Middle επίθεσης υπάρχει μεγάλη πιθανότητα υποκλοπής κωδικών πρόσβασης μεταξύ αυτών και του controller εάν δεν είναι κρυπτογραφημένοι και εισάγονται σαν απλό κείμενο.

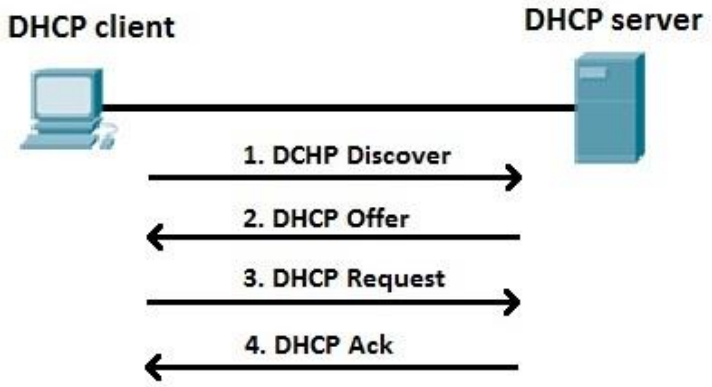
Για την προστασία από μη εξουσιοδοτημένη πρόσβαση είναι απαραίτητο η επικοινωνία μεταξύ των επιπέδων να γίνεται με μηχανισμούς κρυπτογράφησης και χρήση ισχυρών κωδικών. Για την κρυπτογράφηση μπορούν να χρησιμοποιηθούν μηχανισμοί όπως TLS, SSL και IPSec που είναι σχεδιασμένοι ώστε να εξασφαλίζουν ασφαλή επικοινωνία μέσα σε ένα δίκτυο. Επιπλέον είναι σημαντικό να διασφαλιστεί η επικοινωνία μεταξύ των στοιχείων που αποτελούν ένα δίκτυο SDN όπως ο controller και οι μεταγωγείς αφού η επικοινωνία τους γίνεται απομακρυσμένα χρησιμοποιώντας πρωτόκολλα όπως το OpenFlow και αυτό μπορεί να επιτευχθεί με τη χρήση μηχανισμών κρυπτογράφησης κυκλώματος. Ακόμα με τη χρήση δημόσιων και ιδιωτικών κλειδιών μπορεί να γίνει ταυτοποίηση της προέλευσης των δεδομένων απορρίπτοντας με αυτό τον τρόπο δεδομένα άγνωστης προέλευσης. Κατά την πρώτη παραμετροποίηση μιας συσκευής βασικό βήμα είναι η τροποποίηση του εργοστασιακού κωδικού πρόσβασης. Για την επιλογή ενός ασφαλούς κωδικού πρόσβασης είναι σημαντικό να ληφθούν υπόψη τα εξής χαρακτηριστικά:

1. Χρήση πέραν των 8 χαρακτήρων
2. Συνδυασμός αριθμών, πεζών - κεφαλαίων χαρακτήρων και συμβόλων
3. Μη χρήση απλών κωδικών όπως όνομα, ημερομηνίες ή αριθμό τηλεφώνου

Η διασφάλιση επιλογής ενός δυνατού κωδικού πρόσβασης δεν εξαρτάται μόνο από τον χρήστη αλλά και από τον κατασκευαστή αφού πολλές φορές δεν πραγματοποιείται φιλτράρισμα του υποψήφιου κωδικού επιτρέποντας τη χρήση του. Ο κατασκευαστής έχει τη δυνατότητα κατά το σχεδιασμό μιας συσκευής να ορίσει ελάχιστες απαιτήσεις για την αποδοχή χρήσης ενός κωδικού όπως πλέον γίνεται σε πολλές περιπτώσεις.

4.3.8 Επίθεση DHCP

Για την απόκτηση πρόσβασης στο διαδίκτυο μια συσκευή χρειάζεται να αποκτήσει μία διεύθυνση IP όμως δεν είναι εφικτό κάθε φορά που συνδέεται μια νέα συσκευή σε ένα διαφορετικό δίκτυο να γίνεται καταχώρηση χειροκίνητα. Το πρόβλημα αυτό λύνει ο DHCP server όπου η δουλειά του είναι να προμηθεύει τις συσκευές που αποκτούν πρόσβαση στο δίκτυο με μία αποκλειστική διεύθυνση IP, η οποία αποδεσμεύεται όταν ο χρήστης αποσυνδεθεί από το δίκτυο. Η λειτουργία μιας DHCP υπηρεσίας είναι απλή, όταν ένας καινούριος χρήστης συνδεθεί σε ένα δίκτυο εκπέμπει ένα αίτημα DHCP και αφού ο διακομιστής λάβει το αίτημα αυτό ανταποκρίνεται με τις απαραίτητες πληροφορίες που χρειάζεται ο χρήστης. Οι πληροφορίες ενός DHCP reply πακέτου περιλαμβάνει τη διεύθυνση IP που αντιστοιχείται στον νέο χρήστη, τη default gateway, την οποία χρησιμοποιεί ο χρήστης για να επικοινωνήσει εκτός του τοπικού δικτύου και τη subnet mask του δικτύου [60].



Εικόνα 4.7: Εκτέλεση πρωτοκόλλου DHCP

4.3.8.1 DHCP Spoofing

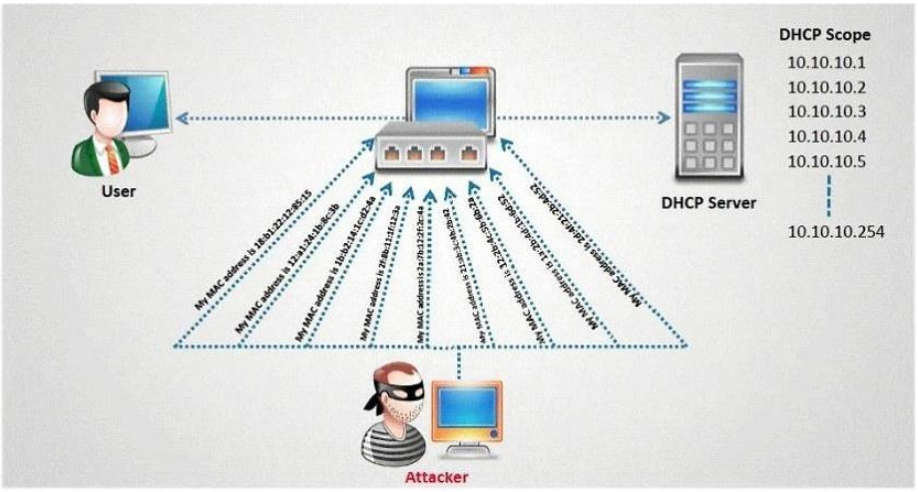
Κατά το DHCP spoofing ο κακόβουλος χρήστης εκτελεί ένα εικονικό DHCP server στον υπολογιστή του. Όταν μία συσκευή αιτηθεί να της ανατεθεί μία διεύθυνση IP το DHCP request πακέτο που εκπέμπει είναι broadcast, δηλαδή οι μεταγωγείς το στέλνουν προς όλες τις ενεργές πόρτες τους, με αποτέλεσμα να λαμβάνει και ο κακόβουλος χρήστης το αίτημα αυτό. Ακολούθως ο εικονικός DHCP διακομιστής ανταποκρίνεται στο αίτημα και αναθέτει μία διεύθυνση δικτύου που δεν είναι ορθή. Ο κύριος σκοπός της επίθεσης αυτής είναι να παραπλανήσει με ψευδή στοιχεία τον χρήστη, όπως default gateway, ώστε τα πακέτα να περνάνε πρώτα από τη συσκευή του θύτη. Το DHCP Spoofing μπορεί να παρομοιαστεί και με το ARP Poisoning αφού έχουν παρόμοιο σκοπό, την υποκλοπή δεδομένων και χρησιμοποιείται για την επίτευξη επίθεσης Man in the Middle [60], [61].

Ο εντοπισμός της επίθεσης σε περίπτωση που δεν ληφθούν μέτρα προστασίας είναι δύσκολο να επιτευχθεί αφού τα πακέτα ανάθεσης εκπέμπονται μόνο στην φυσική διεύθυνση που προήλθε το αίτημα. Αυτό ίσχυε για τα παραδοσιακά δίκτυα που χρησιμοποιούνται μέχρι και σήμερα. Στα δίκτυα SDN ο εντοπισμός μπορεί να γίνει με σχετική ευκολία αφού παρέχεται η δυνατότητα παρακολούθησης των πακέτων. Για παράδειγμα ένας τρόπος είναι η παρακολούθηση των πακέτων DHCP Offer και από που προέρχονται. Αν προέρχονται από αξιόπιστη πηγή τότε επιτρέπεται να φτάσουν τον προορισμό τους, διαφορετικά απορρίπτονται [61].

Από την άλλη η πρόληψη τους, που είναι η καλύτερη επιλογή, μπορεί να γίνει με υπηρεσίες όπως DHCP Snooping και Dynamic ARP Inspection (DAI) που είναι διαθέσιμα και σε παραδοσιακής αρχιτεκτονικής μεταγωγείς. Με παρόμοιο τρόπο μπορεί να γίνει και στα δίκτυα SDN.

4.3.9 DHCP Starvation

Ο σκοπός μιας επίθεσης τύπου DHCP Starvation είναι η εξάντληση των διαθέσιμων διευθύνσεων IP που διαθέτει ένα υποδίκτυο. Η διαδικασία εκτέλεσης της επίθεσης είναι απλή αφού το μόνο που έχει να κάνει ο θύτης είναι να μεταδίδει ψευδή αιτήματα DHCP παραπλανώντας τον διακομιστή ότι κάποια νέα συσκευή έχει συνδεθεί στο δίκτυο. Ο αντίκτυπος της επίθεσης στο δίκτυο είναι η μη δυνατότητα εξυπηρέτησης νέων αιτημάτων για παραχώρηση IP διευθύνσεων αφού έχουν εξαντληθεί όλες οι διαθέσιμες. Για την εκτέλεση μιας τέτοιας επίθεσης υπάρχουν πολλά διαθέσιμα εργαλεία σε μορφή ελεύθερου λογισμικού όπως είναι το dhcpstarv [60].



Εικόνα 4.8: Επίθεση DHCP Starvation

Ο εντοπισμός της επίθεσης όπως και στο DHCP Spoofing αποτελεί πρόκληση αφού αν δεν ληφθούν προληπτικά μέτρα δεν μπορεί ο διακομιστής DHCP να γνωρίζει αν είναι πραγματικά τα αιτήματα ή όχι. Μία μέθοδος εντοπισμού της επίθεσης είναι με την καταμέτρηση του πλήθους των αιτημάτων που εισέρχονται σε ένα δίκτυο όπου θα υπάρχει προκαθορισμένο όριο στον συνολικό αριθμό των πακέτων παρατηρώντας και την πόρτα εισόδου τους. Σε περίπτωση που η επίθεση προέρχεται από μια πόρτα είναι ευκολότερο να εντοπιστεί η επίθεση αφού ανάλογα με την χρήση της πόρτας μπορεί να επιτρέπει συγκεκριμένο αριθμό αιτημάτων [62].

Και πάλι στην περίπτωση της επίθεσης αυτής είναι προτιμότερο να ληφθούν προληπτικά μέτρα. Πολλές περιπτώσεις είναι προτιμότερο να προλαμβάνονται παρά να εντοπίζονται. Υπάρχουν διάφοροι τρόποι πρόληψης αυτού του είδους επιθέσεων όπως το DHCP Lease Time, όπου καθορίζεται μέγιστος χρόνος που μπορεί να διατηρεί δεσμευμένη IP κάποιος χρήστης, δηλαδή ο κάθε χρήστης ή συσκευή μπορεί να κρατήσει μία διεύθυνση IP για προκαθορισμένο χρονικό διάστημα και όταν ο χρόνος αυτός λήξει απαιτείται η αίτηση διεύθυνσης εκ νέου. Επιπλέον μπορεί να γίνει χρήση του Port Security που με την ενεργοποίηση του στον μεταγωγέα σε κάθε φυσική πόρτα του μπορεί να ανατεθεί μία διεύθυνση δικτύου.

4.4 Επίθεση τοπολογίας

Τα δίκτυα καθοριζόμενα από λογισμικό υλοποιούνται χρησιμοποιώντας τοπολογίες όπως και τα παραδοσιακά. Μια επίθεση κατά της τοπολογίας του δικτύου απειλεί τη συνολική λειτουργία του δικτύου αφού μπορεί να παραπλανήσει τη συνολική εικόνα του controller για την τοπολογία του δικτύου επηρεάζοντας με αυτό τον τρόπο την διαχείριση των πακέτων που διακινούνται ή ακόμα μπορεί να οδηγήσει σε διαρροή ευαίσθητων δεδομένων [63]. Οι επιθέσεις αυτές μπορούν να χωριστούν σε 3 βασικές κατηγορίες, Διαταραχή ανίχνευσης, Λανθασμένη απεικόνιση και Τροποποίηση τοπολογίας.

1. Διαταραχή ανίχνευσης: Η επίθεση για διαταραχή ανίχνευσης έχει σκοπό να εμποδίσει τον controller να ανιχνεύσει την τοπολογία του δικτύου επηρεάζοντας παράλληλα τις διαδικασίες και αποφάσεις δρομολόγησης.
2. Λανθασμένη απεικόνιση: Ο επιτιθέμενος χρησιμοποιώντας διάφορα εργαλεία μπορεί να δημιουργήσει την αντίληψη ότι η τοπολογία είναι διαφορετική από την πραγματική εμποδίζοντας τον controller να σχηματίσει μια ορθή εικόνα του δικτύου.
3. Τροποποίηση τοπολογίας: Κατά την απόπειρα τροποποίησης της τοπολογίας ο κακόβουλος χρήστης δημιουργεί εικονικούς μεταγωγείς και δρομολογητές

τροποποιώντας την τοπολογία του δικτύου και παράλληλα την συνολική εικόνα που έχει ο controller οδηγώντας τα πακέτα προς λανθασμένες κατευθύνσεις.

Για την προστασία του δικτύου από επιθέσεις κατά της τοπολογίας είναι σημαντικό να ληφθούν μέτρα όπως η αυθεντικοποίηση των συσκευών που επιχειρούν να επηρεάσουν την τοπολογία και η εφαρμογή μηχανισμών εντοπισμού της και μη εξουσιοδοτημένων συσκευών [64].

4.5 Επιθέσεις OpenFlow

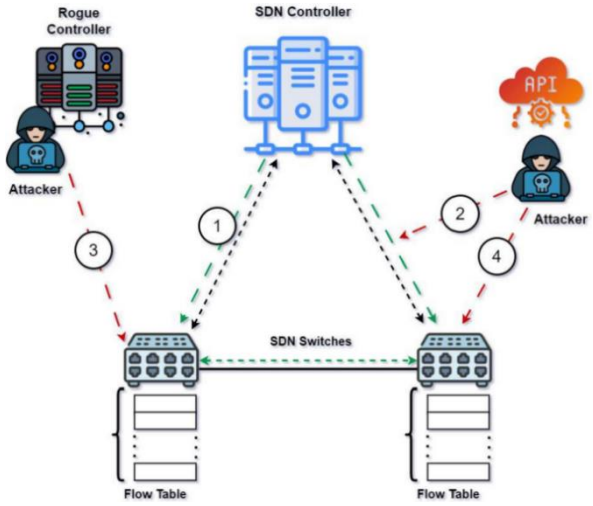
Το πρωτόκολλο OpenFlow χρησιμοποιείται από τα δίκτυα SDN για την επικοινωνία controller και μεταγωγών. Πολλές από τις επιθέσεις που εκτελούνται μπορούν να το επηρεάσουν χωρίς να είναι αυτό ο στόχος αλλά εξαιτίας της λειτουργίας του ίσως επηρεαστεί. Πέρα από τις επιθέσεις που μπορούν να το επηρεάσουν χωρίς να είναι αυτό ο στόχος υπάρχουν και αυτές που το αφορούν αποκλειστικά. Επιθέσεις στο πρωτόκολλο OpenFlow έχουν ως στόχο να επηρεάσουν την επικοινωνία του controller με τους μεταγωγείς, τους πίνακες ροής και τις ενέργειες που εκτελούνται για τη διαχείριση των δεδομένων [65], [66].

1. Υπερχείλιση πίνακα ροής: Κατά την εκτέλεση της ο κακόβουλος χρήστης εκπέμπει σε μεγάλο αριθμό δεδομένα με σκοπό την εξάντληση των πόρων του μεταγωγέα λόγω της μικρής χωρητικότητας των πινάκων ροής.
2. Παραποίηση κανόνων ροής: Οι μεταγωγείς OpenFlow για την διαχείριση των εισερχόμενων πακέτων αποθηκεύουν προηγούμενες οδηγίες δρομολόγησης από τον controller σε πίνακα. Όταν δεν λαμβάνετε κάποιο μέτρο προστασίας για την πηγή των οδηγιών αυτών υπάρχει κίνδυνος να λάβουν οι μεταγωγείς οδηγίες από μη εξουσιοδοτημένους χρήστες με αποτέλεσμα να επηρεαστεί η δρομολόγηση των πακέτων. Σε περίπτωση επίθεσης παραποίησης του πίνακα ροής ο κακόβουλος χρήστης μπορεί να καταχωρήσει οδηγία ώστε όλα τα πακέτα να μεταδίδονται σε αυτό με σκοπό την παρακολούθηση της κίνησης.
3. Πλαστογράφηση controller: Ο controller αποτελεί το βασικό κομμάτι ενός SDN δικτύου και η απώλεια του μπορεί να προκαλέσει σοβαρά προβλήματα στο δίκτυο. Το πρωτόκολλο OpenFlow διατηρεί την επικοινωνία αυτή μέσω της Νότιας διεπαφής. Όταν δεν γίνει κατάλληλη παραμετροποίηση τους μεταγωγείς υπάρχει κίνδυνος η διαχείριση του δικτύου να μεταβιβαστεί σε κακόβουλο ελεγκτή.
4. Παραβίαση ασφαλούς καναλιού: Το κανάλι αυτό παρέχει την επικοινωνία μεταξύ του ελεγκτή και των μεταγωγέων και σε περίπτωση παραβίασης του ο κακόβουλος χρήστης μπορεί να αποκόψει ή να τροποποιήσει τα δεδομένα που μεταφέρονται σε αυτό.

4.6 Εισαγωγή κακόβουλου ελεγκτή

Στην επίθεση Rogue Controller Attack, ο κακόβουλος χρήστης επιχειρεί να εισάγει ένα κακόβουλο ελεγκτή ώστε να αποκτήσει τον έλεγχο του δικτύου εκτελώντας δικές του εφαρμογές ή εισάγοντας κανόνες στους OpenFlow μεταγωγείς [67].

Η ενσωμάτωση του κακόβουλου ελεγκτή σε ένα δίκτυο μπορεί να επιτευχθεί παραβιάζοντας το δίκτυο και εντοπίζοντας πιθανές ευπάθειες στην παραμετροποίηση ή το λογισμικό του ελεγκτή. Ακολούθως για την επίτευξη επικοινωνίας με τις OpenFlow συσκευές ο κακόβουλος controller επιχειρεί να παρουσιάσει τον εαυτό του ως τον πραγματικό παρεμβάλλοντας στην επικοινωνία μέσω της Νότιας διεπαφής.



Εικόνα 4.9: Επίθεση με κακόβουλο ελεγκτή

Αφού εκτελεστεί με επιτυχία η επίθεση ο κακόβουλος χρήστης μπορεί να πετύχει τα ακόλουθα:

1. Τροποποίηση των πινάκων ροής.
2. Παρακολούθηση της κίνησης του δικτύου.
3. Υποκλοπή και τροποποίηση δεδομένων (Man in the middle attack).
4. Μετάδοση κακόβουλου λογισμικού όπως malware.

Για την προστασία του δικτύου από μια επίθεση με κακόβουλο ελεγκτή είναι σημαντικό να ληφθούν τα απαραίτητα μέτρα για την αποτροπή της. Μερικά από αυτά τα μέτρα είναι η κρυπτογράφηση της επικοινωνίας μεταξύ ελεγκτή και OpenFlow συσκευών, η τμηματοποίηση του δικτύου ώστε οι ζωτικής σημασίας συσκευές να μην είναι εύκολα προσβάσιμες και η χρήση μηχανισμών ταυτοποίησης.

4.7 Μέθοδοι προστασίας

Για την αντιμετώπιση πιθανών επιθέσεων υπάρχουν διάφοροι τρόποι. Για την επιλογή μιας μεθόδου προστασίας είναι απαραίτητο να ληφθεί υπόψιν ποιες είναι οι πιθανές επιθέσεις που μπορούν να πραγματοποιηθούν στο δίκτυο. Επιπλέον είναι σημαντικό να είναι εις γνώση του μηχανικού από που μπορούν να πραγματοποιηθούν, μέσα στο τοπικό δίκτυο ή έξω από αυτό.

Για να πραγματοποιηθεί μία επίθεση μέσα από το δίκτυο είναι απαραίτητο επιτιθέμενος με κάποιο τρόπο να αποκτήσει πρόσβαση σε αυτό. Πολλές φορές ένα δίκτυο είναι προσβάσιμο από πολλούς άγνωστους χρήστες όπως σε ένα στάδιο ή εμπορικό κέντρο που διατίθεται δωρεάν πρόσβασή στο διαδίκτυο και αυτό το καθιστά ευάλωτο σε επιθέσεις όπως DDoS, DHCP starvation και MAC flooding.

Από την άλλη, σε ένα ιδιωτικό δίκτυο είναι δυσκολότερο να αποκτήσει πρόσβαση ένας κακόβουλος χρήστης αφού η πρόσβαση είναι ελεγχόμενη με διάφορους τρόπους όπως κωδικοί πρόσβασης και χρήση Network access Control όπου πρόσβαση έχουν μόνο όσοι είναι καταχωρημένοι ως χρήστες. Πλέον πολλές εταιρείες εφαρμόζουν την πολιτική του Bring your own device, δηλαδή οι εργαζόμενοι μπορούν να χρησιμοποιούν τις δικές τους συσκευές για να εργαστούν, ή εάν πρόκειται για απομακρυσμένη εργασία η χρήση VPN. Αυτές οι πολιτικές εργασίας βάζουν ένα δίκτυο σε μεγάλο κίνδυνο αφού οι χρήστες χρησιμοποιούν την ίδια συσκευή και εκτός εργασίας και υπάρχει μεγάλη πιθανότητα εάν έχει προσληφθεί από κάποιο κακόβουλο λογισμικό να αποκτήσει κάποιος μη εξουσιοδοτημένος χρήστης πρόσβαση. Εάν κάποιος κακόβουλος χρήστης αποκτήσει πρόσβαση σε ένα ιδιωτικό δίκτυο είναι πολύ πιθανό να πραγματοποιήσει επιθέσεις τύπου Man in the middle και επιθέσεις Spoofing ώστε να

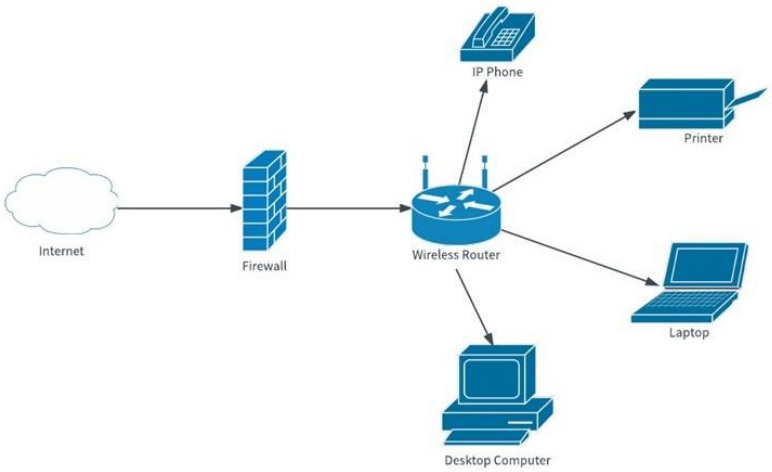
εξυπηρετήσει τους δικούς του σκοπούς όπως διαρροή ευαίσθητων πληροφοριών ή να ζητήσει αμοιβή για την απελευθέρωση του δικτύου ή υπολογιστικών συστημάτων.

Η προστασία ενός δικτύου και των χρηστών του μπορεί να επιτευχθεί με διάφορους τρόπους με τον πιο γνωστό τη χρήση Firewall. Στα σύγχρονα δίκτυα γίνεται και χρήση συστημάτων Intrusion Detection System και Intrusion Prevention System τα οποία παρακολουθώντας το δίκτυο εντοπίζουν ή αποτρέπουν διάφορες επιθέσεις. Τα δίκτυα καθοριζόμενα από λογισμικό έρχονται να φέρουν μια νέα μέθοδο προστασίας που βασίζεται στην χρήση εφαρμογών που εκτελούνται στο επίπεδο εφαρμογών και στον ελεγκτή.

Σημαντικό να σημειωθεί ότι κάθε μέθοδος προστασίας χρησιμοποιείται σύμφωνα με τις ανάγκες λειτουργίας του δικτύου χωρίς να αποκλείεται ο συνδυασμός τους.

4.7.1 Firewall

Σε ένα δίκτυο το Firewall αποτελεί σημαντικό κομμάτι της υλοποίησης του αφού προστατεύει τους χρήστες που είναι συνδεδεμένοι στο τοπικό δίκτυο από πιθανές εξωτερικές απειλές όπως διάφορες μορφές κακόβουλου λογισμικού ή από την εκτέλεση κάποιας επίθεσης. Σε ένα οικιακό δίκτυο τα σύγχρονα modem/router που χρησιμοποιούνται διαθέτουν ενσωματωμένο firewall που εκτελεί βασικές λειτουργίες όμως η προστασία που προσφέρει δεν είναι 100% εγγυημένη αφού δεν διαθέτει την απαραίτητη επεξεργαστική ισχύ. Από την άλλη ένα εταιρικό ή μεγάλης κλίμακας δίκτυο απαιτεί μία συσκευή που να εκτελεί αποκλειστικά λειτουργίες Firewall [68].



Εικόνα 4.10: Firewall εντός δικτύου

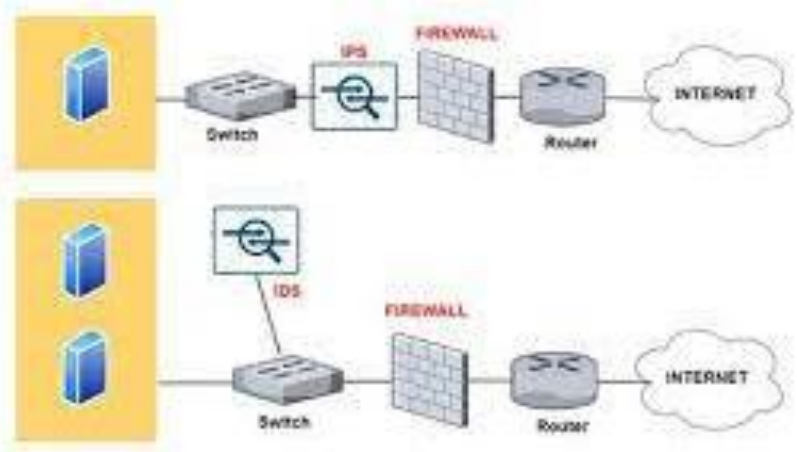
Οι λειτουργία ενός Firewall βασίζεται σε προκαθορισμένους από τον διαχειριστή του δικτύου κανόνες. Οι κανόνες αυτοί μπορεί να είναι η απόρριψη πρόσβασης σε ορισμένες σελίδες στο διαδίκτυο και η απόρριψη πακέτων από ή προς συγκεκριμένες διευθύνσεις. Επιπλέον ένα Firewall μπορεί να αναλύσει το περιεχόμενο των μεταδιδόμενων πακέτων για πιθανό κακόβουλο λογισμικό που ίσως εμπεριέχεται. Ακόμα με τη χρήση μηχανισμών μπορούν να εντοπίσουν και να αποτρέψουν διάφορες επιθέσεις όπως DDoS. Οι περισσότερες από τις δυνατότητες ενός παραδοσιακού Firewall είναι καθορισμένες από τον εκάστοτε κατασκευαστή και για την προσθήκη νέων μεθόδων προστασίας γίνεται με τη λήψη ενημερώσεων λογισμικού ή από την επικοινωνία με κάποια βάση δεδομένων.

4.7.2 IPS/IDS

Οι συντομογραφίες IPS και IDS αντιπροσωπεύουν τους όρους Intrusion Prevention System και Intrusion Detection System αντίστοιχα. Πρόκειται για δύο τεχνολογίες πολύ υποσχόμενες στον τομέα της δικτύωσης αφού σε πραγματικό χρόνο μπορούν να εντοπίσουν και να προστατεύσουν πιθανές επιθέσεις.

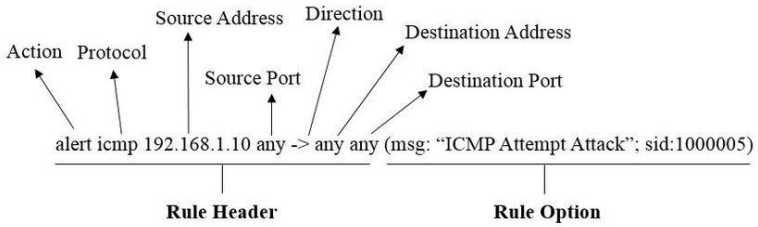
Ένα σύστημα IPS χρησιμοποιείται για την αποτροπή μιας επίθεσης προτού αυτή επηρεάσει τις λειτουργίες του δικτύου. Για τη λειτουργία αναλύει την συμπεριφορά του δικτύου για τυχόν μη φυσιολογική κίνηση όπως μία επίθεση DDoS και με βάση την υπογραφή των πακέτων μπορεί να εντοπίσει πιθανό μολυσμένο αρχείο. Μέχρι στιγμής τα χαρακτηριστικά του είναι παρόμοια με αυτά ενός Firewall. Το χαρακτηριστικό το οποίο κάνει ένα IPS να ξεχωρίζει από ένα Firewall είναι η δυνατότητα να αποτρέπει μόνο την επίθεση χωρίς να επηρεάζει την υπόλοιπη κίνηση του δικτύου. Για την ορθή λειτουργία του πρέπει να ληφθούν υπόψη η χρήση του δικτύου και το μέγεθος του αφού μέσα από αυτό περνάει όλη η κίνηση από και προς τους χρήστες με αποτέλεσμα σε περίπτωση υπερβολικού φόρτου να μην μπορεί να ανταποκριθεί επηρεάζοντας σημαντικά την απόδοση του δικτύου.

Από την άλλη ένα σύστημα IDS είναι σχεδιασμένο να εντοπίζει μία επίθεση αφότου αυτή εκτελείται. Η λειτουργία του βασίζεται στον έλεγχο και σύγκριση του περιεχομένου των πακέτων με μία βάση δεδομένων που διαθέτει μοτίβα τα οποία αποδεικνύουν πιθανές επιθέσεις. Τα IDS από τη φύση τους δεν εκτελούν κάποια διαδικασία διακοπής της επίθεσης αλλά ενημερώνουν τους διαχειριστές του δικτύου.



Εικόνα 4.11: IPS vs IDS

Πολύ γνωστό IPS στα δίκτυα καθοριζόμενα από λογισμικό είναι ο SNORT. Ο SNORT αρχικά σχεδιάστηκε ώστε να παρακολουθεί τα πακέτα που κινούνται σε ένα δίκτυο ενώ στην συνέχεια απέκτησε περισσότερες δυνατότητες που τον κατέστησαν IPS. Η λειτουργία του βασίζεται σε κανόνες που εκτελούνται κατά προτεραιότητα και μπορεί εύκολα να ενσωματωθεί σε ένα δίκτυο SDN. Κατά την εκτέλεση του τα πακέτα που εισέρχονται φιλτράρονται και συγκρίνονται με τους κανόνες που είναι καθορισμένοι και σε περίπτωση αντιστοίχισης δημιουργείται ένα προειδοποιητικό μήνυμα με τις απαραίτητες πληροφορίες και μεταφέρεται στον ελεγκτή. Με βάση το μνήμα που θα λάβει ο controller θα εκτελέσει τις προκαθορισμένες ενέργειες όπως η απόρριψη των πακέτων που προέρχονται από τον ίδιο αποστολέα [69].



Εικόνα 4.12: Δομή κανόνα σε IDS Snort

4.7.3 Προστασία με βάση τον SDN ελεγκτή

Τα SDN δίκτυα πέρα από τις παραδοσιακές μεθόδους προστασίας που χρησιμοποιούνται, όπως και στα παραδοσιακής αρχιτεκτονικής δίκτυα, μπορούν να προστατευτούν χρησιμοποιώντας σύγχρονες μεθόδους. Με τη χρήση των εφαρμογών παρέχεται η δυνατότητα στους μηχανικούς και τους προγραμματιστές να δημιουργούν δικές τους λύσεις προστασίας. Χάρη στην κεντρικοποιημένη λειτουργία των SDN δικτύων και του επιπέδου εφαρμογών η παρακολούθηση της κίνησης που εισέρχεται και εξέρχεται του δικτύου ή ακόμα και της κίνησης που υπάρχει μεταξύ των συσκευών γίνεται πιο εύκολη προσφέροντας ακόμα καλύτερη προστασία σε σύγκριση με ένα παραδοσιακό firewall που προστατεύει μόνο το άκρο του δικτύου χωρίς την απαίτηση επιπλέον υλικού ή λογισμικού. Επιπλέον οι εφαρμογές που τρέχουν σε SDN δίκτυα λειτουργούν και συνδικάστηκα ως firewall, IDS και IPS με επιπλέον δυνατότητες που πιθανόν να μην περιέχονται με τα παραδοσιακά συστήματα προστασίας. Η προστασία με βάση τις εφαρμογές που εκτελούνται παράλληλα με τον ελεγκτή είναι κυρίως εφικτή χάρη στο πρωτόκολλο επικοινωνίας μεταξύ των μεταγωγέων και του ελεγκτή με το πιο γνωστό να είναι το Openflow που παρέχει τις απαραίτητες μεθόδους για τη συλλογή των δεδομένων που απαιτούνται όπως κίνηση και όγκος δεδομένων [70], [71].

4.7.4 Μηχανική μάθηση και Τεχνητή Νοημοσύνη

Η ανάπτυξη των τεχνολογιών Μηχανικής μάθησης και Τεχνητής νοημοσύνης έφερε τη επανάσταση στον τομέα της πληροφορικής. Η χρήση τους συμπεριλαμβάνει την αυτοματοποίηση διάφορων εργασιών, ανέπαφη επικοινωνία του ανθρώπου και την ανάπτυξη μοντέλων για την έγκαιρη πρόβλεψη φυσικών καταστροφών. Πέρα από τις γνωστές χρήσεις τους οι τεχνολογίες αυτές μπορούν να χρησιμοποιηθούν για τον εντοπισμό και αντιμετώπιση επιθέσεων στα δίκτυα. Η χρήση τους μπορεί να γίνει δυναμικά αναλύοντας σε βάθος χρόνου την συνηθισμένη κίνηση σε ένα δίκτυο και την σύγκριση της όταν εκτελείται μια επίθεση. Επιπλέον με τη χρήση διάφορων πληροφοριών που καθορίζονται από τους διαχειριστές ενός δικτύου μια επίθεση μπορεί να γίνει αντιληπτή πρώτού προκαλέσει κάποιο πρόβλημα στο δίκτυο ή τους χρήστες του [72], [73].

4.8 Επίλογος

Όπως φαίνεται από τα πιο πάνω ακόμα και τα δίκτυα SDN που είναι μια πολύ υποσχόμενη αρχιτεκτονική καλούνται να αντιμετωπίσουν μεγάλο αριθμό επιθέσεων όμοιο με τις επιθέσεις που μπορούν να εκτελεστούν και στα παραδοσιακά δίκτυα. Πέρα από την εκτέλεση επιθέσεων στους χρήστες του δικτύου, σε ένα SDN δίκτυο στόχος γίνεται και το επίπεδο ελέγχου μαζί με τα επίπεδα εφαρμογών και δεδομένων. Λόγω της κεντρικοποιημένης αρχιτεκτονικής τους τα δίκτυα SDN σε περίπτωση επίθεσης στον ελεγκτή είναι πολύ πιθανό να διακοπεί η λειτουργία ολόκληρου του δικτύου και για το λόγο αυτό είναι απαραίτητο να λαμβάνονται μέτρα προστασίας που να προστατεύουν και αυτόν.

Κεφάλαιο 5ο: Περιβάλλον προσομοίωσης

5.1 Εισαγωγή

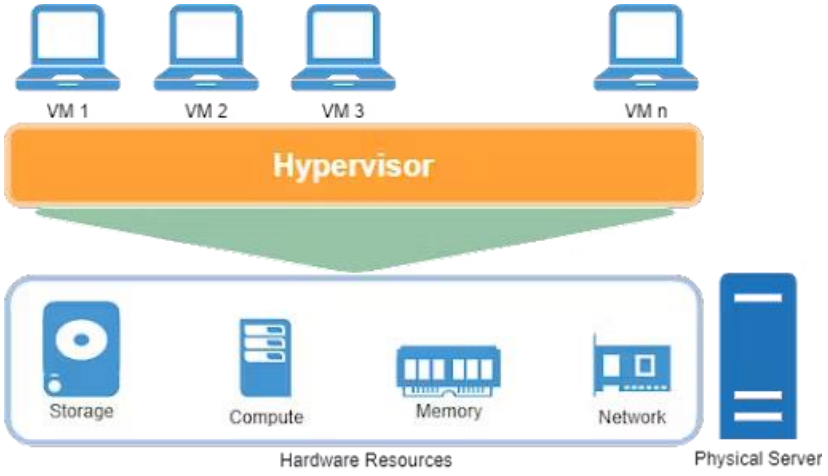
Για την εκτέλεση προσομοιώσεων είναι απαραίτητη η χρήση κατάλληλων εργαλείων που θα μπορέσουν να ανταπεξέλθουν στις απαιτήσεις τους. Η επιλογή των εργαλείων αυτών είναι σημαντικό να γίνει με προσοχή αφού υπάρχουν πολλά εργαλεία που μπορούν να χρησιμοποιηθούν για τον ίδιο σκοπό. Για την εκτέλεση των προσομοιώσεων του Κεφαλαίου 6 τα εργαλεία επιλέχθηκαν με βάση τα ακόλουθα χαρακτηριστικά:

1. Ευχρηστία: Αν διαθέτει οπτικό περιβάλλον είναι σημαντικό να είναι κατάλληλα διαμορφωμένο και κατηγοριοποιημένο. Επιπλέον να διαθέτει εγχειρίδιο χρήστη για την καλύτερη χρήση των δυνατοτήτων του.
2. Αποδοτικότητα: Η λειτουργία του είναι αποδοτική και να αντεπεξέρχεται στις απαιτήσεις της προσομοίωσης.
3. Ευελιξία: να παρέχει τη δυνατότητα τροποποίησης και ορισμού χαρακτηριστικών για την εκτέλεση του
4. Συμβατότητα: να μπορεί να λειτουργήσει παράλληλα με άλλα εργαλεία και σε διάφορα περιβάλλοντα.
5. Αξιοπιστία: το αποτέλεσμα του να είναι αξιόπιστο και με μικρή πιθανότητα λάθους.

5.2 Εργαλεία προσομοιώσεων

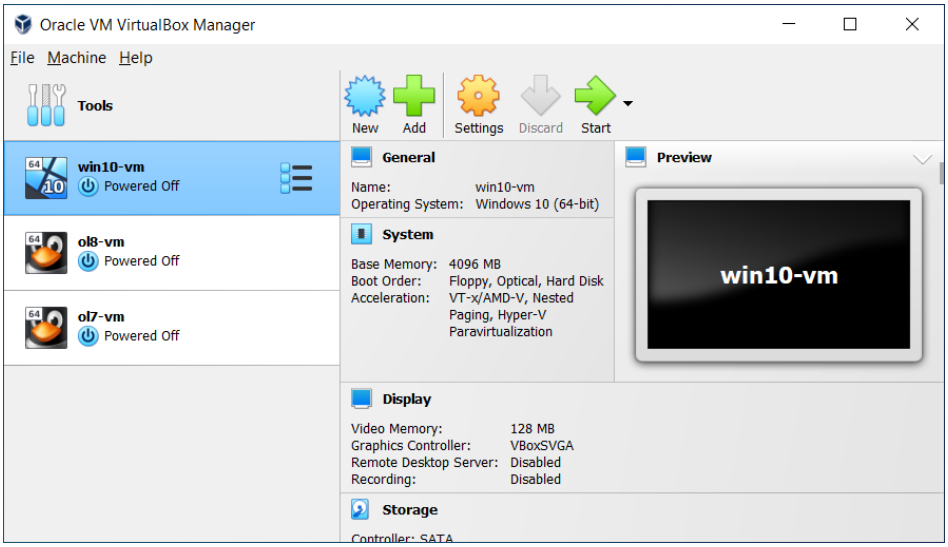
5.2.1 Oracle Virtual Box

Για την εκτέλεση των προσομοιώσεων γίνεται χρήση εικονικών μηχανών. Το virtualization χρησιμοποιείται σε περιπτώσεις που απαιτείται η χρήση λειτουργικών συστημάτων χωρίς την παρουσία επιπλέον φυσικής συσκευής ή ακόμα και σε κέντρα διαχείρισης δεδομένων όπου πολλοί διακομιστές φιλοξενούνται σε ένα φυσικό μηχάνημα [74]. Στην παρούσα πτυχιακή εργασία επιλέχθηκε το Virtual Box της Oracle.



Εικόνα 5.1: Virtualization

Το Virtual Box βρίσκεται στην 7η έκδοση του και μπορεί να εγκατασταθεί στα περισσότερα λειτουργικά συστήματα ενώ υποστηρίζει επεξεργαστές με αρχιτεκτονική x86 και AMD64/Intel64. Η λήψη του γίνεται από την επίσημη ιστοσελίδα της εφαρμογής και μπορεί να εγκατασταθεί σε λίγα μόνο λεπτά [75].



Εικόνα 5.2: Διεπαφή Χρήστη Oracle Virtual Box

Η χρήση του είναι αρκετά απλή αφού χάρη στην εύχρηστη διεπαφή που διαθέτει δεν απαιτούνται εξειδικευμένες γνώσεις. Μέσα από την διεπαφή ο χρήστης έχει τη δυνατότητα να δημιουργήσει, να διαχειριστεί και να τροποποιήσει τις εικονικές μηχανές. Κατά τη δημιουργία μιας μηχανής ο χρήστης ακολουθεί μερικά βασικά βήματα όπως ο ορισμός ονόματος, ο τύπος του λειτουργικού που επιθυμεί να εγκαταστήσει και τα διάφορα χαρακτηριστικά όπως οι πυρήνες επεξεργαστή, η μνήμη RAM και το μέγεθος του αποθηκευτικού χώρου [75]. Επιπλέον οι παράμετροι αυτοί μπορούν να τροποποιηθούν οποιαδήποτε στιγμή όπως μπορεί να γίνει και σε ένα φυσικό σύστημα.

5.2.2 Mininet Simulator

Για την εκτέλεση των σεναρίων, χωρίς τη χρήση φυσικού εξοπλισμού, είναι απαραίτητη η χρήση ενός προγράμματος προσομοίωσης. Για την εκτέλεση των σεναρίων και της τοπολογίας έχει επιλεγεί η χρήση του προσομοιωτή Mininet [76].

Ο προσομοιωτής Mininet είναι λογισμικό ανοικτού κώδικα και η διανομή του γίνεται δωρεάν. Ο Mininet είναι σχεδιασμένος κατάλληλα ώστε να μπορεί να χρησιμοποιηθεί για εκπαιδευτικούς και πειραματικούς σκοπούς και είναι ιδιαίτερα γνωστός για τη χρήση σε δίκτυα καθορισμένα από λογισμικό καθώς διαθέτει τις απαιτούμενες δυνατότητες.

Η εγκατάσταση του γίνεται σε λειτουργικό Linux είναι αρκετά εύκολη για κάποιο χρήστη που διαθέτει βασικές γνώσεις για τα λειτουργικά αυτά και μπορεί να γίνει με τους εξής τρόπους:

1. Εικονική μηχανή: Εάν ο χρήστης δεν διαθέτει προ εγκατεστημένο λειτουργικό Linux μπορεί να κάνει λήψη μιας εικονικής μηχανής και είναι ο πιο γρήγορος τρόπος

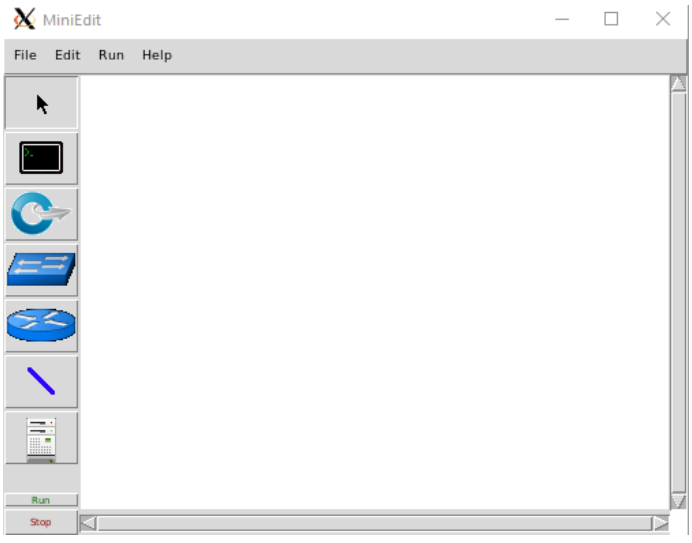
έναρξης χρήσης του προσομοιωτή αφού όλα τα απαραίτητα στοιχεία που χρειάζεται για τη λειτουργία του είναι προ εγκατεστημένα.

2. Εγκατάσταση από Git: Όπως προαναφέρθηκε ο προσομοιωτής Mininet είναι εφαρμογή ανοικτού κώδικα και ο καθένας μπορεί να επέμβει σε αυτή, για το λόγο αυτό η εφαρμογή είναι διαθέσιμη για λήψη μέσα από το GitHub.
3. Εγκατάσταση με χρήση πακέτων: Η εγκατάσταση μιας εφαρμογής σε ένα λειτουργικό σύστημα Linux μπορεί να γίνει μέσω των εντολών apt και apt-get εφόσον είναι διαθέσιμη.

Για τη δημιουργία μιας τοπολογίας, ο προσομοιωτής Mininet μας παρέχει τρεις μεθόδους, με τη χρήση εντολών στο τερματικό, με τη χρήση διεπαφής χρήστη και τη χρήση script.

Η πρώτη μέθοδος είναι η δημιουργία της τοπολογίας με εντολές στο τερματικό. Η μέθοδος αυτή είναι η γρηγορότερη αφού με τη χρήση απλών εντολών μπορεί ο χρήστης να δημιουργήσει την τοπολογία σε μικρό χρονικό διάστημα παραθέτοντας μόνο τις απαραίτητες πληροφορίες για την επιθυμητή τοπολογία όπως το πλήθος των χρηστών, τη δομή του δικτύου και την διεύθυνση του ελεγκτή.

Η δεύτερη μέθοδος είναι με τη χρήση της διεπαφής χρήστη. Η διεπαφή αυτή παρέχει στο χρήστη τα απαραίτητα στοιχεία (controller, switches, links, hosts) για τη δημιουργία της τοπολογίας με μορφή drag and drop. Ο χρήστης πατώντας με το δεξί βελάκι του ποντικιού σε κάποιο από τα στοιχεία αυτά μπορεί να το παραμετροποιήσει σύμφωνα με τις ανάγκες του όπως ο καθορισμός διευθύνσεων IP, έκδοση πρωτοκόλλου OpenFlow και άλλων διαθέσιμων επιλογών.



Εικόνα 5.3: Στιγμιότυπο διεπαφής χρήστη Mininet

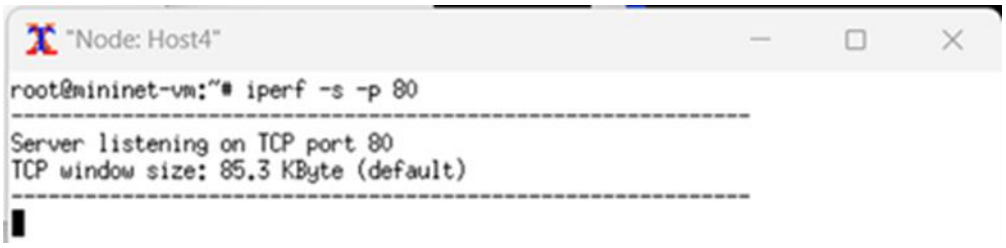
Η τρίτη και σημαντικότερη μέθοδος είναι με τη χρήση script. Με τη μέθοδο αυτή ο χρήστης έχει τη δυνατότητα να δημιουργήσει περίπλοκες τοπολογίες με περισσότερα χαρακτηριστικά. Η χρήση script απαιτεί περισσότερες γνώσεις από τις προηγούμενες δύο μεθόδους αφού ο χρήστης πρέπει με συγκεκριμένη δομή και κώδικα να περιγράψει την τοπολογία. Για την συγγραφή του γίνεται χρήση της γλώσσας python στην οποία βασίζεται και ο προσομοιωτής.

5.2.3 Εργαλείο IPERF

Σε μία προσομοίωση είναι απαραίτητη η χρήση εργαλείων, όπως το IPERF, για την πραγματοποίηση δοκιμών ώστε να δοκιμαστεί η απόδοση του δικτύου. Παράλληλα πέρα από την επιτυχία επικοινωνίας μεταξύ δύο χρηστών είναι σημαντική η καταγραφή των επιδόσεων κατά την επικοινωνία αυτή και η καταγραφή πιθανών προβλημάτων [77].

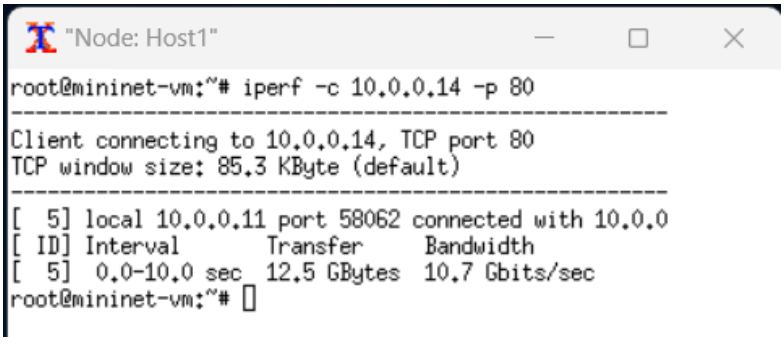
Το εργαλείο IPERF απαιτεί να δηλωθούν κάποιες παραμέτροι με τη χρήση διαφόρων ορισμάτων, δηλαδή πληροφορίες για την εκτέλεση της δοκιμής. Αρχικά χρειάζεται να ορίσουμε μέσα από το τερματικό αν πρόκειται για διακομιστή ή χρήστη, χρησιμοποιώντας το όρισμα -s ή -c αντίστοιχα. Αυτό είναι απαραίτητο αφού η συσκευή πρέπει να μπει σε μία από τις δύο καταστάσεις.

Όσον αφορά το κομμάτι του διακομιστή (-s) δεν είναι υποχρεωτική η χρήση κάποιου άλλου ορίσματος αφού σε περίπτωση που δεν δηλωθούν χρησιμοποιούνται οι προεπιλεγμένες επιλογές που είναι η χρήση της πόρτας 5001 (-p), χρήση του TCP πρωτοκόλλου, αναφορά στατιστικών ανά δευτερόλεπτο (-i) και με απεριόριστο χρόνο λειτουργίας. Επιπλέον ορίσματα που μπορεί να δεχτεί όταν πρόκειται να μπει σε κατάσταση διακομιστή είναι ¨-D¨ για εκκίνηση στο παρασκήνιο και ¨-I¨ για αποθήκευση των αποτελεσμάτων σε αρχείο.



Εικόνα 5.4: Έναρξη εικονικού διακομιστή με τη χρήση του εργαλείου iperf

Όταν πρόκειται ο χρήστης να ενεργήσει ως client (-c) διαθέτει περισσότερα αποκλειστικά ορίσματα παρέχοντας τη δυνατότητα στον χρήστη να επιλέξει τη μέθοδο επικοινωνίας όπως TCP και UDP ενώ μπορεί να επιλέξει μεταξύ του όγκου των δεδομένων και της διάρκειας της επικοινωνίας.



Εικόνα 5.5: Έναρξη χρήστη ως πελάτη σε μια επικοινωνία με τη χρήση του εργαλείου iperf

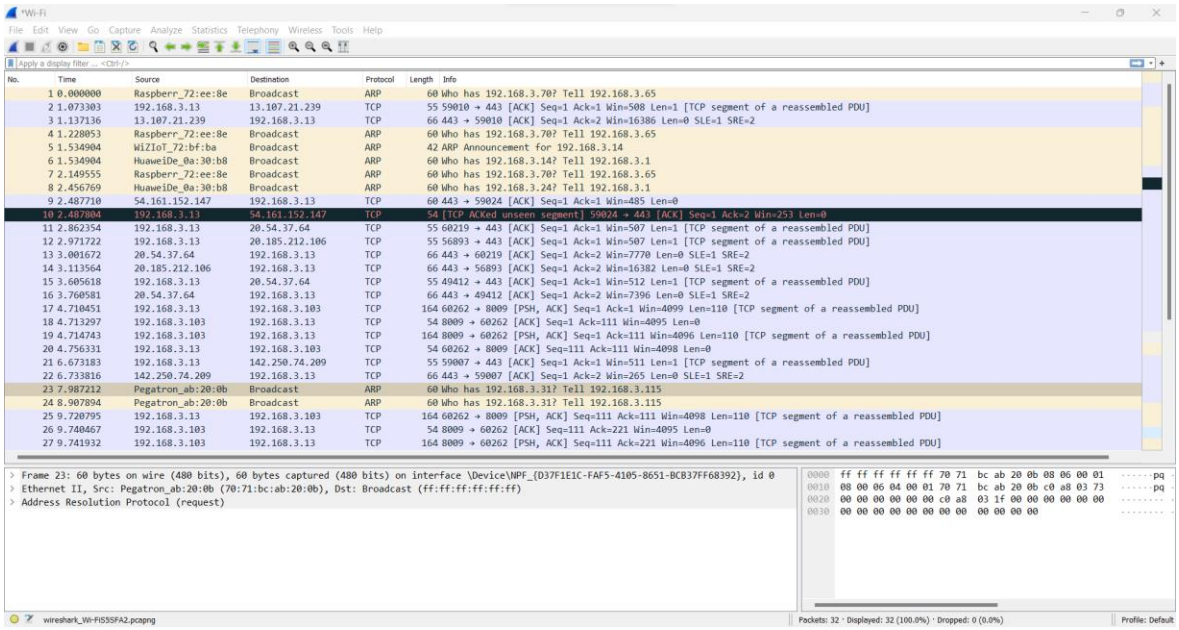
5.2.4 Εργαλείο arpspoof

Για την εκτέλεση της επίθεσης ARP Spoofing του επόμενου κεφαλαίου γίνεται χρήση του εργαλείου ¨arpspoof¨. Το εργαλείο αυτό δεν διανέμεται αυτόνομα αλλά ανήκει σε μία ομάδα εργαλείων, την dsniff και γίνεται την εντολή «apt install dsniff» [78]. Στο πακέτο αυτό συμπεριλαμβάνονται πληθώρα εργαλείων που αφορούν spoofing επιθέσεις. Το εργαλείο ¨arpspoof¨ για την λειτουργία του απαιτεί κάποιες πληροφορίες σε μορφή ορισμάτων. Τα ορίσματα αυτά είναι τα ακόλουθα:

1. -i: Το όνομα της κάρτας δικτύου από την οποία θα εκτελεστεί η επίθεση
2. -c: διεύθυνση ip από την οποία θέλουμε να υποκλέψουμε τα πακέτα που προορίζονται προς αυτή.
3. -t: Η διεύθυνση ip του θύματος που θέλουμε τροποποιήσουμε τον πίνακα ARP
4. -r: Καθορισμός του εύρους διευθύνσεων ip που θέλουμε να εκτελέσουμε την επίθεση

5.2.5 Wireshark

Η εφαρμογή Wireshark είναι πολύ γνωστή στον κομμάτι της δικτύωσης, αφού επιτρέπει την παρακολούθηση των πακέτων σε πραγματικό χρόνο, όταν εξέρχονται ή εισέρχονται σε ένα υπολογιστικό σύστημα και είναι διαθέσιμη για λειτουργικά συστήματα Linux και Windows. Πέρα από την παρακολούθηση των πακέτων η εφαρμογή αυτή δίνει τη δυνατότητα ανάλυσης ενός πακέτου εξάγοντας πληροφορίες από αυτό, όπως οι διευθύνσεις IP και MAC και ο τύπους του πακέτου [79].



Εικόνα 5.6: Διεπαφή χρήστη Wireshark

Η διεπαφή της εφαρμογής Wireshark παρέχει στο χρήστη τις ακόλουθες δυνατότητες:

1. Χρήση φίλτρων ώστε να εντοπίζει ο χρήστης με περισσότερη ευκολία τα δεδομένα που χρειάζεται όπως τύπο πακέτου, διεύθυνση αποστολέα και πολλά άλλα.

2. Με την επιλογή ενός πακέτου στο κάτω μέρος της οθόνης εμφανίζει με λεπτομέρεια όπως πληροφορίες πλαισίου, στοιχεία αποστολέα και παραλήπτη, το πρωτόκολλο επικοινωνίας, και πολλές άλλες πληροφορίες που διαφοροποιούνται σύμφωνα με τον τύπο του κάθε πακέτου.
3. Εμφάνιση δεδομένων πακέτου σε δεκαεξαδική ή δυαδική μορφή
4. Εξαγωγή στατιστικών: μας δίνει τη δυνατότητα να εξάγουμε πληροφορίες για τα πακέτα που διακινήθηκαν συνολικά όπως το πλήθος ενός τύπου πακέτου.
5. Αποθήκευση των εγγραφών που κατέγραψε για μελλοντική ανάλυση

5.3 Περιβάλλον προσομοίωσης

5.3.1 Εικονικές μηχανές

Η εκτέλεση των προσομοιώσεων γίνεται με τη χρήση δύο εικονικών μηχανών στο περιβάλλον Oracle VM Virtualbox. Η πρώτη μηχανή εκτελεί το ρόλο του ελεγκτή και διαθέτει τα ακόλουθα χαρακτηριστικά:

1. 2Gb μνήμη RAM,
2. 2 επεξεργαστικούς πυρήνες και
3. 32Gb αποθηκευτικό χώρο.

Στη δεύτερη μηχανή εκτελείται ο προσομοιωτής Mininet και διαθέτει:

4. 4Gb μνήμη RAM,
5. 2 επεξεργαστικούς πυρήνες και
6. 8Gb αποθηκευτικό χώρο

Και πρόκειται για έτοιμη εικονική μηχανή που διατίθεται από την επίσημη σελίδα του προσομοιωτή Mininet διασφαλίζοντας με αυτό τον τρόπο την ορθή λειτουργία του.

5.3.2 Πιθανά προβλήματα

Για την εκτέλεση των προσομοιώσεων επιλέχθηκε να χρησιμοποιηθεί ο controller Ryu. Λόγω της έκδοσης Python που υποστηρίζει, υπάρχει πιθανότητα να μην εκτελείται επιτυχώς. Για την επίλυση του προβλήματος χρησιμοποιείται το εργαλείο Conda, το οποίο επιτρέπει τη χρήση περιβάλλοντος με καθορισμένες από το χρήση προδιαγραφές, όπως η χρήση συγκεκριμένης έκδοσης της Python.

5.3.3 Εκκίνηση εικονικών μηχανών

Αφού γίνει εκτέλεση των εικονικών μηχανών στο περιβάλλον του Virtual Box σειρά έχουν ο προσομοιωτής Mininet και ο controller.

Για το περιβάλλον προσομοίωσης Mininet οι βασικές εντολές που χρησιμοποιούνται είναι οι ακόλουθες:

1. sudo ~/mininet/examples/miniedit.py: χρησιμοποιείται σε περίπτωση που η τοπολογία θα υλοποιηθεί με τη χρήση του γραφικού περιβάλλοντος που διαθέτει το Mininet.

2. sudo python3 topo.py: εκτέλεση δημιουργίας τοπολογίας με τη χρήση script
3. pingall: ελέγχει αν οι host επικοινωνούν μεταξύ τους
4. xterm: εντολή με την οποία ανοίγει τερματικό για συγκεκριμένο χρήστη
5. quit: τερματίζει τον προσομοιωτή Mininet

Για την εκκίνηση του ελεγκτή Ryu οι βασικές εντολές είναι οι ακόλουθες:

1. conda activate py38: ενεργοποιεί το περιβάλλον Conda για την εκτέλεση του ελεγκτή με την έκδοση python που απαιτεί
2. ryu-manager $CONDA_PREFIX/lib/python3.8/site-packages/ryu/app/app_name.py: εκκινεί τον ελεγκτή με την εφαρμογή της επιλογής του χρήστη (χρήση του simple_switch_l3.py ως βάση για την εκτέλεση των script αντιμετώπισης των επιθέσεων)
3. ^c: τερματίζει την λειτουργία του ελεγκτή

5.3.4 Τοπολογία προσομοιώσεων

Για τις ανάγκες των σεναρίων που ακολουθούν στα επόμενα κεφάλαια γίνεται χρήση script. Αρχικά για την δημιουργία της τοπολογίας είναι απαραίτητο να γίνει εισαγωγή κάποιων βιβλιοθηκών που αφορούν τον προσομοιωτή Mininet.

```
3  from mininet.net import Mininet
4  from mininet.node import Controller, RemoteController, OVSController
5  from mininet.node import CPULimitedHost, Host, Node
6  from mininet.node import OVSKernelSwitch, UserSwitch
7  from mininet.node import IVSSwitch
8  from mininet.cli import CLI
9  from mininet.log import setLogLevel, info
10 from mininet.link import TCLink, Intf
11 from subprocess import call
```

Εικόνα 5.7: Βιβλιοθήκες Mininet

Ακολούθως αφού γίνει η εισαγωγή των απαραίτητων βιβλιοθηκών ακολουθεί η υλοποίηση της τοπολογίας ορίζοντας τα μέρη του δικτύου όπως ο controller, οι μεταγωγείς και οι χρήστες. Το επόμενο βήμα είναι η δημιουργία της μεταβλητής ¨net¨ η οποία δηλώνει το δίκτυο Mininet με ορίσματα τον τύπο της τοπολογίας, αν η τοπολογία θα δημιουργηθεί άμεσα και το υποδίκτυο που χρειάζονται για την δημιουργία της βάσης του δικτύου.

```
13 def myNetwork():
14
15     net = Mininet( topo=None,
16                    build=False,
17                    ipBase='10.0.0.0/24')
```

Εικόνα 5.8: Δήλωση δικτύου

Σειρά έχει η καταχώρηση των συσκευών που θα αποτελούν το δίκτυο. Ένα δίκτυο καθοριζόμενο από λογισμικό απαιτεί την ύπαρξη ενός ελεγκτή. Στην τοπολογία των προσομοιώσεων ο controller έχει το όνομα ¨c0¨. Με το όνομα αυτό δημιουργείται μία μεταβλητή τύπου ¨net¨ και

καλώντας τη μέθοδο addController() δημιουργείται το αντικείμενο του ελεγκτή. Για να επικοινωνήσουν οι OpenFlow συσκευές με τον ελεγκτή απαιτούνται τα εξής ορίσματα:

1. name: το όνομα του ελεγκτή
2. controller: καθορίζει αν ο controller είναι απομακρυσμένος
3. ip: η διεύθυνση IP στην οποία βρίσκεται
4. protocol: το πρωτόκολλο επικοινωνίας μπορεί να είναι TCP ή UDP. Είναι προτιμότερη η χρήση του tcp καθώς είναι πιο ασφαλές και εγγυημένη η επικοινωνία με τις OpenFlow συσκευές.
5. port: η πόρτα από την οποία δίνεται πρόσβαση στον ελεγκτή.

```
18
19      info( '*** Adding controller\n' )
20      c0=net.addController(name='c0',
21                           controller=RemoteController,
22                           ip='192.168.3.148',
23                           protocol='tcp',
24                           port=6633)
```

Εικόνα 5.9: Εισαγωγή ελεγκτή

Για την δημιουργία των μεταγωγέων και των χρηστών ακολουθείται παρόμοια διαδικασία. Για τη δημιουργία ενός μεταγωγέα καλείται η μέθοδος ¨addSwitch¨ δίνοντας ως παραμέτρους τα ακόλουθα:

1. το όνομα του μεταγωγέα
2. cls: η κλάση του μεταγωγέα

Όσον αφορά τη δημιουργία των χρηστών καλείται η μέθοδος ¨add Host¨ με τα ακόλουθα ορίσματα:

1. το όνομα του χρήστη
2. cls: την κλάση του
3. ip: την διεύθυνση ip του αν γίνει στατικά
4. mac: την φυσική διεύθυνση του αν δεν θέλουμε να μπει τυχαία

```
25
26      info( '*** Add switches\n')
27      s1 = net.addSwitch('s1', cls=OVSKernelSwitch)
28      s2 = net.addSwitch('s2', cls=OVSKernelSwitch)
29      s3 = net.addSwitch('s3', cls=OVSKernelSwitch)
30
31      info( '*** Add hosts\n')
32      Host1 = net.addHost('Host1', cls=Host, ip='10.0.0.11', mac='00:00:00:00:00:01', defaultRoute=None)
33      Host2 = net.addHost('Host2', cls=Host, ip='10.0.0.12', mac='00:00:00:00:00:02', defaultRoute=None)
34      Host3 = net.addHost('Host3', cls=Host, ip='10.0.0.13', mac='00:00:00:00:00:03', defaultRoute=None)
35      Host4 = net.addHost('Host4', cls=Host, ip='10.0.0.14', mac='00:00:00:00:00:04', defaultRoute=None)
```

Εικόνα 5.10: Δήλωση μεταγωγέων και χρηστών

Η παραπάνω διαδικασία επαναλαμβάνεται μέχρι να δημιουργηθούν όλοι οι απαιτούμενοι μεταγωγείς και χρήστες. Το Mininet από μόνο του δεν μπορεί να γνωρίζει τον τρόπο που συνδέονται μεταξύ τους οι συσκευές πέρα από την σύνδεση των μεταγωγέων με τον ελεγκτή που είναι προκαθορισμένος. Για την σύνδεση των συσκευών μεταξύ τους χρησιμοποιείται η μέθοδος

¨addLink¨ του αντικειμένου ¨net¨ που δημιουργήθηκε αρχικά δίνοντας σαν παραμέτρους τις δύο συσκευές που θα συνδέει ο σύνδεσμος αυτός. Στην περίπτωση που ο χρήστης επιθυμεί να ορίσει χωρητικότητα στον σύνδεσμο αυτό χρησιμοποιείται επιπλέον το όρισμα ¨bw¨ και τη χρηστικότητα σε Mbps.

```
36
37      info( '*** Add links\n')
38      net.addLink(s1, s2)
39      net.addLink(s1, s3)
40      net.addLink(Host1, s2)
41      net.addLink(s2, Host2)
42      net.addLink(Host3, s3)
43      net.addLink(s3, Host4)
44
```

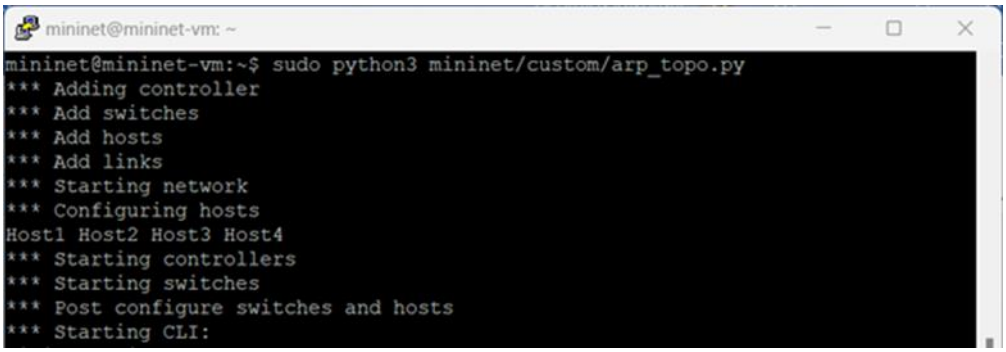
Εικόνα 5.11: Δήλωση συνδέσμων μεταξύ μεταγωγέων και χρηστών

Στο σημείο αυτό έχουν οριστεί όλες οι συσκευές του επιθυμητού δικτύου. Για να χτίσει ο προσομοιωτής το δίκτυο χρησιμοποιείται η μεταβλητή ¨net.build¨. Στη συνέχεια εκτελείται έλεγχος αν οι καθορισμένοι ελεγκτές είναι διαθέσιμοι και γίνεται εκκίνηση τους. Τέλος γίνεται εκκίνηση όλων των συσκευών και ενεργοποιείται το τερματικό του προσομοιωτή ώστε να μπορούν να εκτελεστούν ενέργειες μέσα από αυτόν.

```
44
45      info( '*** Starting network\n')
46      net.build()
47      info( '*** Starting controllers\n')
48      for controller in net.controllers:
49          controller.start()
50
51      info( '*** Starting switches\n')
52      net.get('s1').start([c0])
53      net.get('s2').start([c0])
54      net.get('s3').start([c0])
55
56      info( '*** Post configure switches and hosts\n')
57
58      CLI(net)
59      net.stop()
60
61  if __name__ == '__main__':
62      setLogLevel( 'info' )
63      myNetwork()
```

Εικόνα 5.12: Έλεγχος και εκκίνηση τοπολογίας

Αφού δοθεί η εντολή έναρξης του ελεγκτή εμφανίζονται στο τερματικό τα περιεχόμενα του στιγμιότυπου που ακολουθεί και σε περίπτωση που το αποτέλεσμα είναι διαφορετικό και εμφανίζονται μνήματα σφάλματος υποδεικνύοντας ότι κάποιο πρόβλημα υπάρχει είτε με την διαδικασία εκκίνησης του ελεγκτή ή με κάποιο σημείο στων κώδικα της εκτελούμενης εφαρμογής. Το αποτέλεσμα της επιτυχούς εκκίνησης του προσομοιωτή περιλαμβάνει πληροφορίες που αφορούν τη δημιουργία και εκκίνηση των μεταγωγών και των χρηστών καθώς προστίθεται και ο controller επικοινωνώντας με την διεύθυνση δικτύου του ελεγκτή που ορίστηκε κατά την συγγραφή του Script και ενεργοποιείται το τερματικό του προσομοιωτή Mininet από το οποίο μπορούν να εκτελεστούν διάφορες ενέργειες όπως το άνοιγμα των τερματικών των μεταγωγών και των χρηστών.



Εικόνα 5.13: Αποτέλεσμα επιτυχής έναρξης προσομοιωτή

Κεφάλαιο 6ο: Προσομοίωση επιθέσεων

6.1 Εισαγωγή

Η χρήση προσομοιώσεων αποτελεί ένα πολύ χρήσιμο εργαλείο για τους μηχανικούς αφού τους δίνεται η δυνατότητα να αναλύσουν τη λειτουργία των δικτύων και να προσομοιώσουν διάφορες καταστάσεις, όπως οι επιθέσεις. Επιπλέον, εκτελώντας προσομοιώσεις στα δίκτυα SDN δίνεται η δυνατότητα υλοποίησης και δοκιμής διάφορων εφαρμογών που μπορούν να φανούν χρήσιμες για τη λειτουργία ενός δικτύου όπως η αντιμετώπιση επιθέσεων. Στο παρόν κεφάλαιο προσομοιώνονται τρεις επιθέσεις, ARP Spoofing, DDoS και Slowloris σε συνθήκες χωρίς προστασία και με προστασία παρουσιάζοντας μία μέθοδο για την αποτροπή ή αντιμετώπιση της κάθε μιας ξεχωριστά.

6.2 Επίθεση ARP Spoofing

Όπως περιγράφεται στο σημείο 4.3.3, μία επίθεση τύπου ARP Spoofing έχει ως κύριο σκοπό την τροποποίηση του πίνακα ARP ενός χρήστη και την υποκλοπή δεδομένων που προορίζονται προς άλλο παραλήπτη. Η παρούσα προσομοίωση έχει στόχο την κατανόηση της εκτέλεσης μιας επίθεσης ARP Spoofing και των συνεπειών της.

Κατά την εκτέλεση της προσομοίωσης ακολουθείται το εξής σενάριο: Ο Χρήστης 2 (Host2) έχει στόχο την υποκλοπή δεδομένων που μεταδίδει ο Χρήστης 1 (Host1) προς τον διακομιστή WEB (Host4). Για την επίτευξη του στόχου του, ο Χρήστης 2, εκτελεί μια επίθεση ARP Spoofing με στόχο να τροποποιήσει τον πίνακα ARP του Χρήστη 1 παραπλανώντας τον ότι η διεύθυνση 10.0.0.14 που ανήκει στο διακομιστή WEB, δηλαδή το Χρήστη 4, αντιστοιχείται με την φυσική διεύθυνση 00:00:00:00:00:02 που στην πραγματικότητα ανήκει σε αυτόν. Σημαντικό στοιχείο για την εκτέλεση της προσομοίωσης είναι ότι όλοι οι χρήστες διαθέτουν σταθερές διευθύνσεις δικτύου και ότι δε χρησιμοποιείται κάποιος DHCP server.

Ο διαχειριστής του δικτύου θέλοντας να προστατέψει τους χρήστες του δικτύου από πιθανή επίθεση ARP Spoofing εφαρμόζει μία μέθοδο εντοπισμού και αποτροπής βελτιστοποιώντας την εφαρμογή simple_switch_l3.py που διανέμεται με τον controller και διαθέτει λειτουργία μεταγωγέα επιπέδου 3.

Η παρούσα προσομοίωση χωρίζεται σε τρία μέρη:

1. Διαδικασία εκτέλεσης
2. Επίθεση χωρίς τη λήψη μέτρου προστασίας.
3. Επίθεση με τη λήψη μέτρων προστασίας.

6.2.1 Διαδικασία εκτέλεσης

Το πρώτο μέρος της προσομοίωσης αφορά τα κοινά μέρη που αφορούν την διαδικασία και τις εντολές που χρησιμοποιούνται για την εκτέλεση μιας επίθεσης ARP Spoofing με προστασία ή χωρίς.

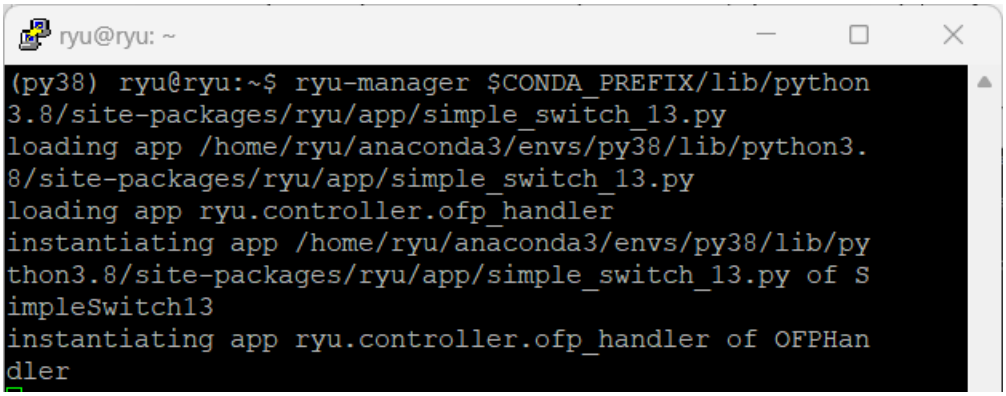
Η προσομοίωση εκτελείται με τη χρήση δύο εικονικών μηχανών όπως αναφέρθηκαν στο σημείο 5.3.1. Αφού γίνει η εκκίνηση των εικονικών μηχανών σειρά έχει η εκκίνηση του controller και του προσομοιωτή Mininet. Για την εκτέλεση του controller χωρίς προστασία γίνεται χρήση της εφαρμογής simple_switch_l3.py η οποία συμπεριλαμβάνεται στην διανομή του controller RYU και για την εκτέλεση της επίθεσης με τη λήψη μέτρου προστασίας γίνεται χρήση αντίστοιχου βελτιστοποιημένου αρχείου με επιπλέον λειτουργίες για την προστασία από επίθεση ARP Spoofing.

Αρχικά είναι απαραίτητο να εκκινήσουν ο controller και ο προσομοιωτής Mininet με επιτυχία, χωρίς την εμφάνιση σφαλμάτων στα τερματικά των εικονικών μηχανών που τα φιλοξενούν.

Για την εκκίνηση του controller γίνεται χρήση της ακόλουθης εντολής που περιέχει ως όρισμα την τοποθεσία και το όνομα της εφαρμογής που θα εκτελεστεί:

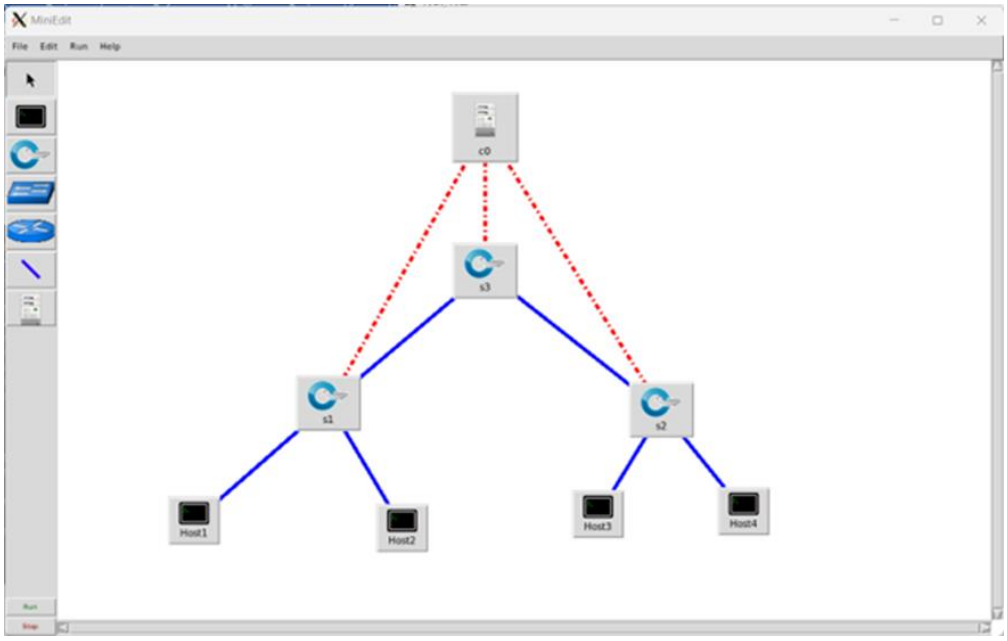
```
$ ryu-manager $CONDA_PREFIX/lib/python3.8/site-packages/ryu/app/simple_switch_13.py
```

Η εκκίνηση του controller επιβεβαιώνεται με την εμφάνιση των πληροφοριών που αφορούν την εφαρμογή που φορτώθηκε και εκτελέστηκε όπως φαίνεται στο ακόλουθο στιγμιότυπο οθόνης.



Εικόνα 6.1: Εκκίνηση controller RYU

Για τους σκοπούς της προσομοίωσης έχει σχεδιαστεί με τη χρήση Script η τοπολογία του δικτύου που εκτελείται με τον προσομοιωτή Mininet, όπως όπως περιγράφηκε στο σημείο 5.3.4. Το προτεινόμενο δίκτυο αποτελείται από ένα controller, τρεις μεταγωγείς και τέσσερις χρήστες, ένας εκ των οποίων εκτελεί το ρόλο του διακομιστή (Host4). Η τοπολογία του δικτύου έχει υλοποιηθεί με τη χρήση script καθώς παρέχει την δυνατότητα να οριστούν χειροκίνητα όλα τα απαραίτητα στοιχεία όπως φυσικές διευθύνσεις MAC και διευθύνσεις δικτύου IP καθώς διευκολύνει την αναγνώριση των χρηστών κατά την εκτέλεση της προσομοίωσης.



Εικόνα 6.2: Τοπολογία δικτύου

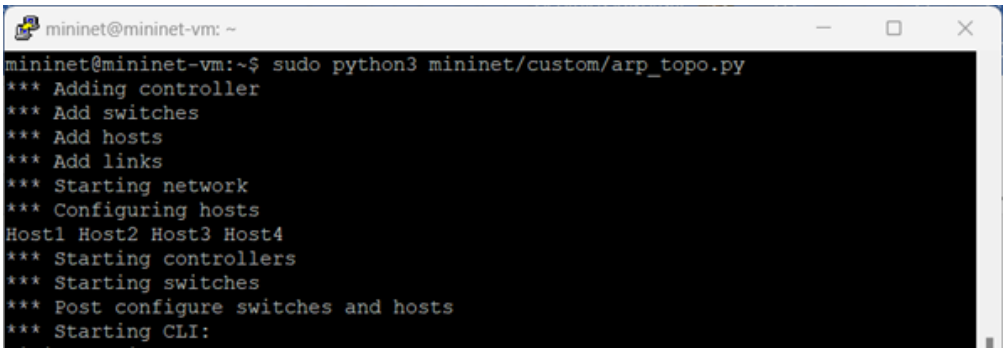
Στον ακόλουθο πίνακα παρουσιάζονται αναλυτικά οι διευθύνσεις MAC και IP που ορίστηκαν χειροκίνητα για κάθε χρήστη ώστε να είναι εύκολα αναγνωρίσιμοι.

Πίνακας 6.1:Αντιστοιχίσεις διευθύνσεων MAC και IP των χρηστών

Χρήστης	MAC	IP
Host1	10.0.0.11	00:00:00:00:00:01
Host2	10.0.0.12	00:00:00:00:00:02
Host3	10.0.0.13	00:00:00:00:00:03
Host4	10.0.0.14	00:00:00:00:00:04

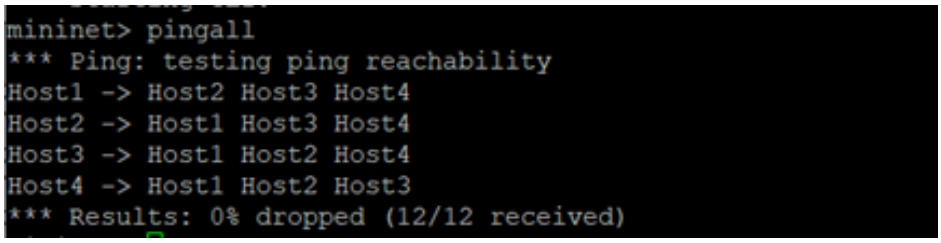
Η εκκίνηση του Mininet γίνεται εκτελώντας το Script που δημιουργήθηκε και περιγράφει την τοπολογία του δικτύου με την ακόλουθη εντολή. Η διαδικασία είναι ίδια με την εκτέλεση οποιουδήποτε Script σε γλώσσα προγραμματισμού Python :

```
$ sudo python3 mininet/custom/simulations_topo.py
```



Εικόνα 6.3: Έναρξη προσομοιωτή Mininet

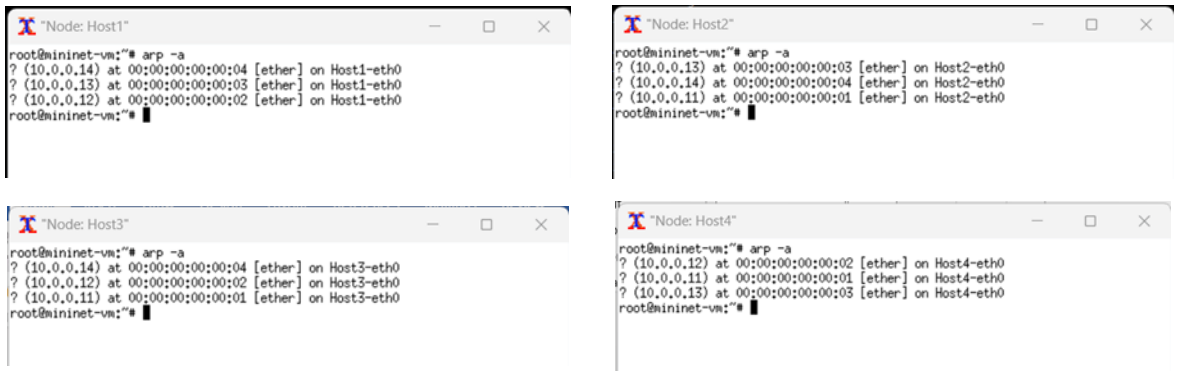
Για την επιβεβαίωση της διασύνδεσης και επικοινωνίας μεταξύ των χρηστών μέσα από το τερματικό του Mininet μπορεί να γίνει χρήση της εντολής «pingall» που εκτελεί ανταλλαγή μηνυμάτων ICMP μεταξύ όλων των χρηστών. Αν το αποτέλεσμα της εντολής εμφανίσει 0% απόρριψη τότε όλα πήγαν καλά και όλοι οι χρήστες επικοινωνούν μεταξύ τους.



Εικόνα 6.4: Εκτέλεση εντολής 'pingall'

Στο σημείο αυτό καλό είναι να επιβεβαιωθεί ότι κατά την επικοινωνία των χρηστών με την προηγούμενη εντολή στο τερματικό του Mininet δημιουργήθηκαν οι πίνακες ARP σε κάθε χρήστη με τις διευθύνσεις MAC και IP, όπως καθορίστηκαν στο script της τοπολογίας. Αυτό μπορεί να επιβεβαιωθεί μέσα από τα τερματικά του κάθε host ξεχωριστά εκτελώντας στον κάθε ένα την ακόλουθη εντολή:

```
# arp -a
```

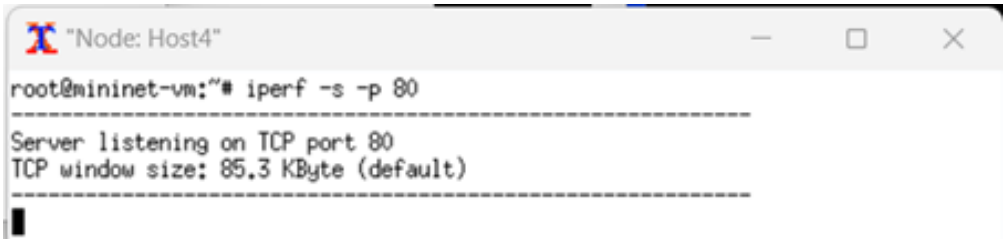


Εικόνα 6.5: Πίνακες ARP χρηστών

Το αποτέλεσμα της εκτέλεσης εμφανίζεται στο τερματικό του κάθε χρήστη και περιλαμβάνει τα δεδομένα του πίνακα ARP που διαθέτει και περιέχουν τη διεύθυνση δικτύου IP, τη φυσική διεύθυνση MAC και την διεπαφή από την οποία μπορεί να γίνει επικοινωνία με τον κάθε χρήστη.

Όπως προαναφέρθηκε κατά την προσομοίωση ο Host4 ενεργεί ως διακομιστής Web με τη χρήση του εργαλείου iperf. Για την εκτέλεση του εργαλείου είναι απαραίτητη η χρήση κάποιων ορισμάτων που αφορούν τη μορφή εκτέλεσης ¨-s¨ που σημαίνει server και ¨-p¨ το οποίο υποδηλώνει την πόρτα στην οποία θα εξυπηρετεί, ακολουθούμενο από τον αριθμό της πόρτας, στην περίπτωση αυτή χρησιμοποιείται η πόρτα με αριθμό 80 στην οποία λειτουργεί ένας Web server. Με τη χρήση της ακόλουθης εντολής εμφανίζεται στην οθόνη του χρήστη που εκτελεί καθήκοντα διακομιστή μήνυμα με την πόρτα που ακούει και το ελάχιστο μέγεθος του παραθύρου TCP πριν λάβει επιβεβαίωση ο αποστολέας.

```
# iperf -s -p 80
```

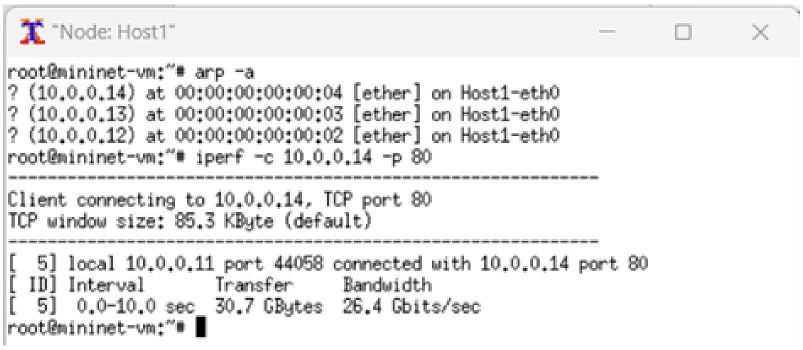


Εικόνα 6.6: Εκκίνηση διακομιστή με τη χρήση του εργαλείου iperf

Στη συνέχεια πρέπει να επιβεβαιωθεί η επικοινωνία client – server από πλευράς του Host 1. Αυτό μπορεί να γίνει εκτελώντας την εντολή:

```
# iperf -c 10.0.0.14 -p 80
```

Η εντολή δέχεται τα ορίσματα ¨-c¨ ως client ακολουθούμενο από την διεύθυνση IP του διακομιστή (Host4) και το όρισμα ¨-p¨ για την πόρτα που επιλέχθηκε, την 80. Αφού γίνει εκτέλεση της εντολής από τον Host1 ξεκινά η διαδικασία επικοινωνίας με τον διακομιστή που βρίσκεται στην διεύθυνση 10.0.0.14 χρησιμοποιώντας την πόρτα 80. Με την εκτέλεση της εντολής αυτής ο host τρέχει ένα εικονικό πρόγραμμα περιήγησης και αφού η επικοινωνία μεταξύ των δύο γίνει με επιτυχία εμφανίζονται στους δύο χρήστες τα αποτελέσματα της ανταλλαγής δεδομένων. Η εκτέλεση της εντολής δίνει σαν αποτέλεσμα την πόρτα που λάμβανε τις απαντήσεις του διακομιστή ο Host 1, τον όγκο των δεδομένων που μεταφέρθηκαν και τη χωρητικότητα του μέσου κατά την επικοινωνία τους.



```
"Node: Host1"
root@mininet-vm:~# arp -a
? (10.0.0.14) at 00:00:00:00:00:04 [ether] on Host1-eth0
? (10.0.0.13) at 00:00:00:00:00:03 [ether] on Host1-eth0
? (10.0.0.12) at 00:00:00:00:00:02 [ether] on Host1-eth0
root@mininet-vm:~# iperf -c 10.0.0.14 -p 80
------------------------------------------------------------
Client connecting to 10.0.0.14, TCP port 80
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  5] local 10.0.0.11 port 44058 connected with 10.0.0.14 port 80
[ ID] Interval       Transfer     Bandwidth
[  5]  0.0-10.0 sec  30.7 GBytes  26.4 Gbits/sec
root@mininet-vm:~#
```

Εικόνα 6.7: Αποτέλεσμα της επικοινωνίας server-client

Με την εκτέλεση και την επιβεβαίωση της επικοινωνίας των χρηστών ακολουθεί η διαδικασία υλοποίησης της επίθεσης ARP Spoofing από τον κακόβουλο host.

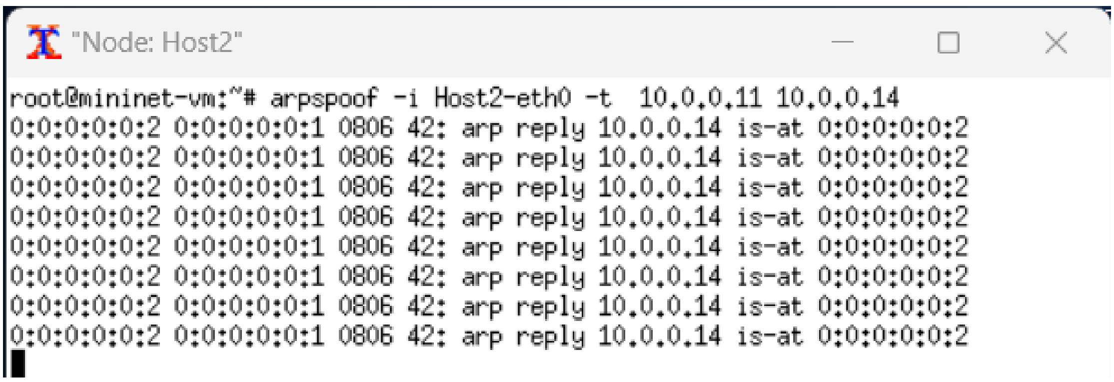
Για την εκτέλεση της επίθεσης χρησιμοποιείται το εργαλείο Arpspoof το οποίο, αφού εκτελεστεί, μεταδίδει πακέτα τύπου ARP Reply ανά μικρά χρονικά διαστήματα ώστε να αποφευχθεί η οποιαδήποτε αλλαγή στον πίνακα ARP του θύματος. Για τη λειτουργία της, η εντολή απαιτεί τη χρήση κάποιων ορισμάτων. Τα ορίσματα αυτά αφορούν:

1. -i: η διεπαφή δικτύου από την οποία θα εκτελεστεί
2. -t: στόχος της επίθεσης
3. Μεταβλητή host: η διεύθυνση δικτύου που θέλει ο κακόβουλος host να υποκλέψει

Αφού ο κακόβουλος host εκτελέσει την επίθεση με την εντολή:

```
# arpspoof -I Host2-eth0 -t 10.0.0.11 10.0.0.14
```

στην οθόνη του εμφανίζονται οι πληροφορίες του πακέτου ARP Reply που μεταδίδονται στο θύμα. Το πακέτο αυτό περιέχει πληροφορίες που αφορούν την φυσική διεύθυνση που διαθέτει ο host που ανταποκρίνεται στη διεύθυνση δικτύου που ζήτησε ο host που μετέδωσε πακέτο ARP Request. Κατά την εκτέλεση της επίθεσης ο host που λαμβάνει το πακέτο δεν έχει ζητήσει τις πληροφορίες αυτές αλλά θεωρεί ότι λαμβάνει ενημέρωση για την τροποποίηση μιας από τις καταχωρήσεις του πίνακα ARP που διαθέτει. Το πακέτο ARP Reply κατά την εκτέλεση της επίθεσης περιέχει πληροφορία που δηλώνει ότι η διεύθυνση δικτύου 10.0.0.14 ανήκει στη φυσική διεύθυνση 00:00:00:00:00:02 παραπλανώντας τον παραλήπτη του.



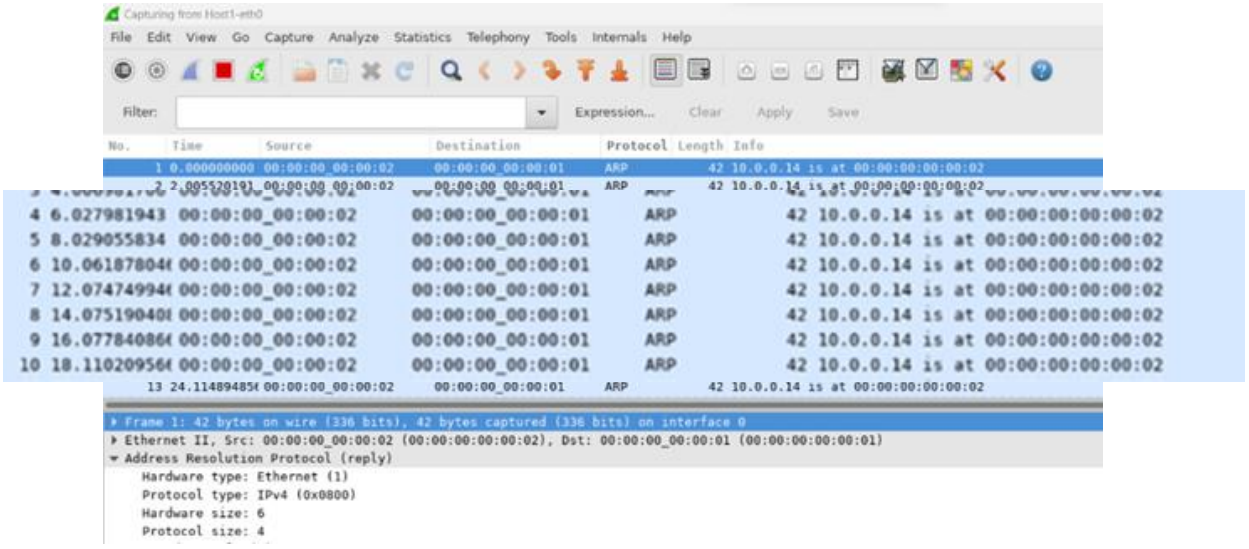
```
"Node: Host2"
root@mininet-vm:~# arpspoof -i Host2-eth0 -t  10.0.0.11 10.0.0.14
0:0:0:0:0:2 0:0:0:0:0:1 0806 42: arp reply 10.0.0.14 is-at 0:0:0:0:0:2
0:0:0:0:0:2 0:0:0:0:0:1 0806 42: arp reply 10.0.0.14 is-at 0:0:0:0:0:2
0:0:0:0:0:2 0:0:0:0:0:1 0806 42: arp reply 10.0.0.14 is-at 0:0:0:0:0:2
0:0:0:0:0:2 0:0:0:0:0:1 0806 42: arp reply 10.0.0.14 is-at 0:0:0:0:0:2
0:0:0:0:0:2 0:0:0:0:0:1 0806 42: arp reply 10.0.0.14 is-at 0:0:0:0:0:2
0:0:0:0:0:2 0:0:0:0:0:1 0806 42: arp reply 10.0.0.14 is-at 0:0:0:0:0:2
0:0:0:0:0:2 0:0:0:0:0:1 0806 42: arp reply 10.0.0.14 is-at 0:0:0:0:0:2
0:0:0:0:0:2 0:0:0:0:0:1 0806 42: arp reply 10.0.0.14 is-at 0:0:0:0:0:2
```

Εικόνα 6.8: Πληροφορίες που εμφανίζονται στον θύτη κατά την επίθεση

6.2.2 Αποτέλεσμα επίθεσης χωρίς προστασία

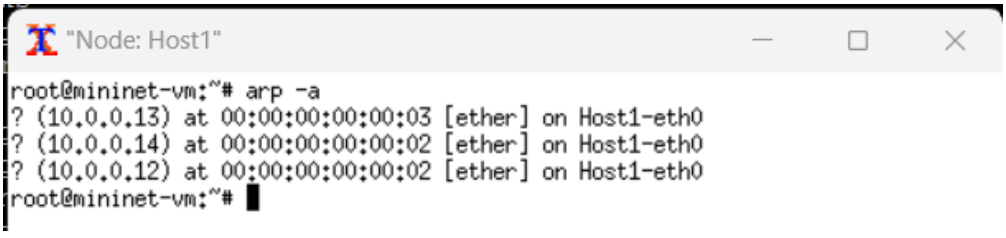
Κατά την εκτέλεση της προσομοίωσης χωρίς κάποιο μέτρο προστασίας ο controller δεν πραγματοποιεί έλεγχο στα πακέτα που λαμβάνει πέρα από τις βασικές λειτουργίες που διαθέτει η εφαρμογή simple_switch_l3.py. Στην περίπτωση αυτή το δίκτυο δεν προσφέρει κανένα μέτρο προστασίας στους χρήστες του καθιστώντας τους ευάλωτους σε οποιαδήποτε πιθανή επίθεση.

Αφού εκτελεστούν τα βήματα που αναφέρθηκαν παραπάνω χωρίς κάποιο πρόβλημα ο Host1 λαμβάνει τα πακέτα τύπου ARP Reply όπως φαίνονται στο ακόλουθο στιγμιότυπο οθόνης του μέσα από την εφαρμογή Wireshark.



Εικόνα 6.9: Στιγμιότυπο των πακέτων ARP Reply που λαμβάνει ο Host1.

Όπως φαίνεται από το πιο πάνω στιγμιότυπο, το θύμα λαμβάνει ανά μικρά χρονικά διαστήματα πακέτο από τον host με φυσική διεύθυνση 00:00:00:00:00:02, που είναι ο Host2. Στη στήλη “Info” του περιβάλλοντος εμφανίζεται η πληροφορία του πακέτου που είναι ότι η IP 10.0.0.14 ανήκει στη MAC 00:00:00:00:00:02 η οποία στην πραγματικότητα είναι λανθασμένη. Με τη λήψη του πακέτου αυτού το θύμα αλλάζει τις καταχωρήσεις στον πίνακα ARP του με τη λανθασμένη στην πραγματικότητα καταχώρηση. Η τροποποίηση αυτή μπορεί να επιβεβαιωθεί μέσα από τις καταχωρήσεις του πίνακα ARP του θύματος κάτι που υποδεικνύει και την επιτυχία της επίθεσης.



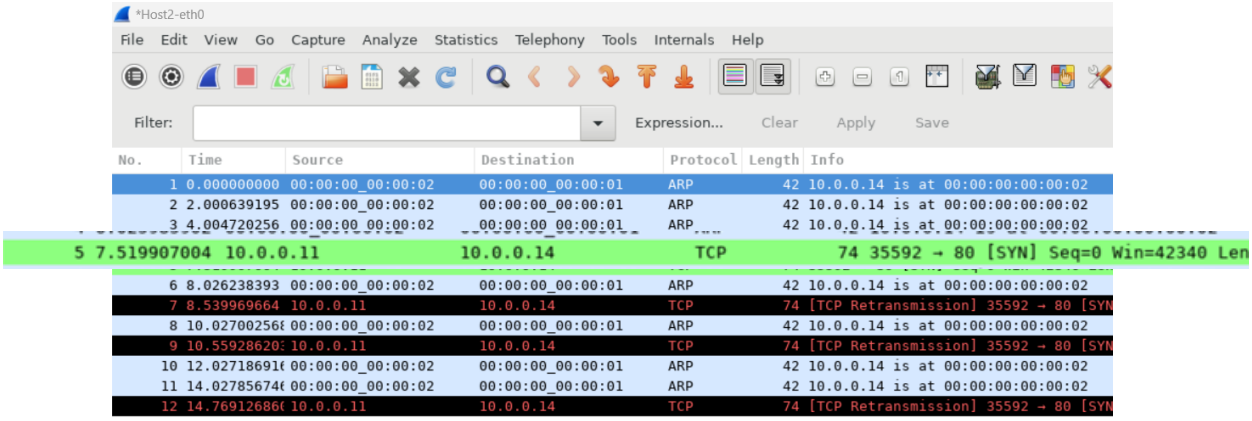
Εικόνα 6.10: Πίνακας ARP Host1 μετά την επίθεση

Η επιτυχία της επίθεσης μπορεί να επιβεβαιωθεί πέρα από τον πίνακα ARP του θύματος και με την απόπειρα επικοινωνίας του με το διακομιστή, δηλαδή τον Host 4. Η απόπειρα επικοινωνίας με το διακομιστή πραγματοποιείται με τον ίδιο τρόπο που έγινε και κατά την δοκιμή ανταλλαγής δεδομένων

μεταξύ τους πριν από την εκτέλεση της επίθεσης από τον Host 2, δηλαδή με την εκτέλεση της ακόλουθης εντολής:

```
# iperf -c 10.0.0.14 -p 80
```

Παρατηρώντας το στιγμιότυπο οθόνης, που ακολουθεί, από την εφαρμογή Wireshark του Host2, φαίνεται ότι σε προσπάθεια του το θύμα να επικοινωνήσει με το διακομιστή στη διεύθυνση 10.0.0.14 αποτυγχάνει αφού τα πακέτα οδηγούνται σε αυτόν. Για να κρύψει τα ίχνη του ο κακόβουλος host και να μην γίνει αντιληπτός μπορεί να χρησιμοποιήσει αντίγραφο της εφαρμογής που επιθυμεί να επικοινωνήσει το θύμα απαντώντας του με ψευδής πληροφορίες ή ακόμα να χρησιμοποιήσει κατάλληλες εφαρμογές που αναμεταδίδουν τα δεδομένα προς τον πραγματικό διακομιστή που αιτείται το θύμα αλλάζοντας τη μορφή της επίθεσης σε τύπου Man in the Middle.



Εικόνα 6.11: Πακέτα που υποκλέπτει ο Host 2 από το Host 1

6.2.3 Επίθεση με προστασία

Αφού έγινε κατανοητή η διαδικασία της επίθεσης σε ένα μη προστατευόμενο δίκτυο σειρά έχει η διαδικασία εντοπισμού και αντιμετώπισης της επίθεσης ARP Spoofing [52].

Ο εντοπισμός της επίθεσης στην παρούσα προσομοίωση γίνεται με τη σύγκριση των δεδομένων που περιέχει ένα πακέτο ARP Reply και των αντιστοιχίσεων που βρίσκονται στον πίνακα ARP_Table όπου δημιουργείται κατά την εκκίνηση των μεταγωγών δικτύου και του controller. Ο πίνακας που δημιουργείται είναι μοναδικός για κάθε μεταγωγέα χρησιμοποιώντας πριν το όνομα του πίνακα ή της μεταβλητής τον όρο self και δηλώνεται εντός της μεθόδου __init__ που χρησιμοποιείται για την αρχικοποίηση μιας κλάσης ή αντικειμένου όπως φαίνεται πιο κάτω.

```
def __init__(self, *args, **kwargs):
    super(SimpleSwitch13, self).__init__(*args, **kwargs)
    self.mac_to_port = {}
    self.ARP_Table = {}
```

Εικόνα 6.12: Μέθοδος αρχικοποίησης κλήσεων και αντικειμένων

Αρχικά, για να πραγματοποιηθεί έλεγχος των πακέτων τύπου ARP Reply κάθε φορά που εισέρχεται στο δίκτυο χρειάζεται να καθοριστεί κανόνας ώστε να γίνεται επεξεργασία των πακέτων αυτών κάθε φορά από τον controller. Ο καθορισμός του κανόνα γίνεται μέσα από την μέθοδο switch_features_handler.

```
def switch_features_handler(self, ev):
    datapath = ev.msg.datapath
    ofproto = datapath.ofproto
    parser = datapath.ofproto_parser

    match = parser.OFPMatch()
    actions = [parser.OFPActionOutput(ofproto.OFPP_CONTROLLER,
                                      ofproto.OFPCML_NO_BUFFER)]
    self.add_flow(datapath, 0, match, actions)

    #Dimiourgia kanonas pou dioxetevei ola ta paketa ARP Reply ston controller
    match = parser.OFPMatch(eth_type=(0x0806), arp_op=2)
    actions = [parser.OFPActionOutput(ofproto.OFPP_CONTROLLER, ofproto.OFPCML_NO_BUFFER)]
    self.add_flow(datapath, 2, match, actions)
```

Εικόνα 6.13: Μέθοδος διαχείρισης γεγονότων στους μεταγωγείς

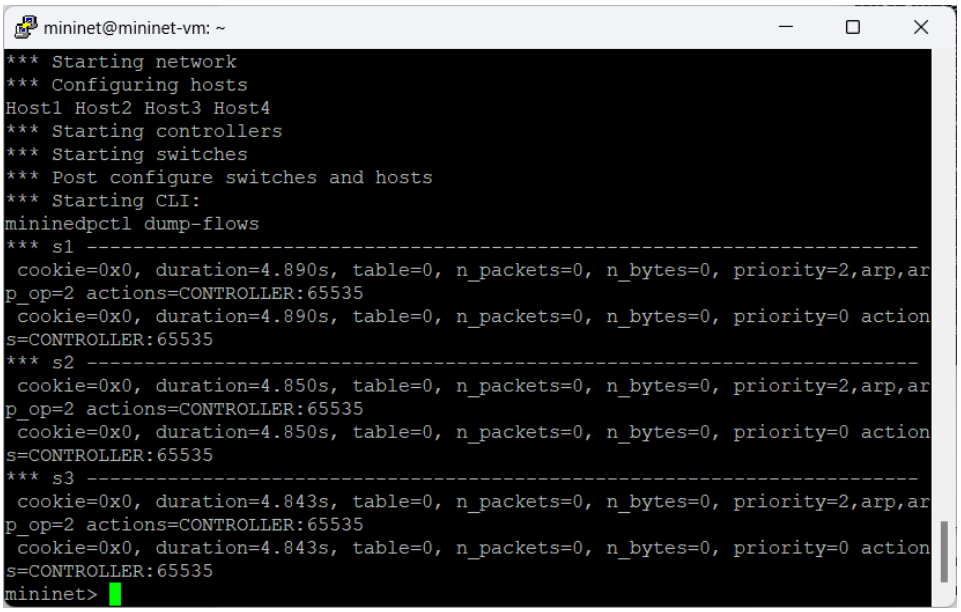
Η μέθοδος switch_features_handler χρησιμοποιείται για την αρχικοποίηση και τη διαχείριση των γεγονότων που δημιουργούνται από τους μεταγωγείς, όπως η άφιξη ενός πακέτου που δεν υπάρχει κάποιος κανόνας διαχείρισης του. Για τη προώθηση των πακέτων ARP Reply στον controller δημιουργούνται οι μεταβλητές match και actions. Αρχικά στη μεταβλητή match ορίζονται όλα τα κριτήρια που χρειάζεται να διαθέτει το εισερχόμενο πακέτο, δηλαδή να είναι πακέτο τύπου ARP, όπου eth_type=(0x0806) είναι το πακέτο ARP και arp_op=2 είναι τύπου reply. Ακολούθως, ορίζεται η μεταβλητή actions στην οποία ενσωματώνονται όλες οι οδηγίες για μεταφορά του πακέτου προς τον controller για ανάλυση. Στην μεταβλητή actions δηλώνεται ότι τα πακέτα αυτά πρέπει να αποστέλλονται πλήρως στον controller χωρίς να αποθηκεύονται στην προσωρινή μνήμη του μεταγωγέα. Στη συνέχεια καταχωρείται ο κανόνας καλώντας τη μέθοδο add_flow που περιλαμβάνεται την εφαρμογή simple_switch_l3.py.

```
def add_flow(self, datapath, priority, match, actions, buffer_id=None, idle_timeout=0, hard_timeout=0):
    ofproto = datapath.ofproto
    parser = datapath.ofproto_parser

    inst = [parser.OFPInstructionActions(ofproto.OFPIT_APPLY_ACTIONS,
                                         actions)]
    if buffer_id:
        mod = parser.OFPFlowMod(datapath=datapath, buffer_id=buffer_id,
                                priority=priority, match=match,
                                instructions=inst, idle_timeout=idle_timeout, hard_timeout=hard_timeout)
    else:
        mod = parser.OFPFlowMod(datapath=datapath, priority=priority,
                                match=match, instructions=inst, idle_timeout=idle_timeout, hard_timeout=hard_timeout)
    datapath.send_msg(mod)
```

Εικόνα 6.14: Μέθοδος add_flow

Καλώντας τη μέθοδο είναι απαραίτητο να δοθούν τα κατάλληλα ορίσματα, το datapath που αντιπροσωπεύει τον μεταγωγέα που θα λάβει τον κανόνα, ένα αριθμό που χαρακτηρίζει την προτεραιότητα (όσο μεγαλύτερος τόσο μεγαλύτερη και η προτεραιότητα) και τις δύο μεταβλητές που δημιουργήθηκαν νωρίτερα, την match και actions. Η δημιουργία του κανόνα είναι σημαντικό να ενσωματωθεί στο κατάλληλο σημείο του κώδικα της εφαρμογής ώστε να γίνει κατά την εκκίνηση του controller. Κατά την εκκίνηση του controller και του Mininet δημιουργείται η καταχώρηση στον πίνακα ροών σε κάθε μεταγωγέα με την ακόλουθη μορφή:



Εικόνα 6.15: Πίνακες ροής μεταγωγέων κατά την εκκίνηση τους

Η κάθε καταχώρηση του πίνακα έχει συγκεκριμένη μορφή και ανάλογα με τα χαρακτηριστικά του κανόνα διαθέτει λιγότερες ή περισσότερες πληροφορίες. Ο κανόνας που δημιουργήθηκε περιέχει πληροφορίες που αφορούν το μοναδικό αναγνωριστικό του κανόνα που αναφέρεται ως cookie, το χρόνο που βρίσκεται σε ισχύ, τον αύξον αριθμό του πίνακα, το πλήθος των πακέτων, τον όγκο των πακέτων που έλαβε μέχρι τη στιγμή εκείνη, την προτεραιότητα, τον τύπο των πακέτων που πρέπει να γίνει αντιστοίχιση και την ενέργεια που θα εκτελεστεί αν εντοπιστεί αντίστοιχο πακέτο με αυτό που ορίζει.

Όπως προαναφέρθηκε, το πρώτο στάδιο για τον εντοπισμό της επίθεσης είναι ο έλεγχος του πακέτου ώστε να εξακριβωθεί αν είναι τύπου ARP Reply. Εφόσον αυτό επιβεβαιωθεί γίνεται αντιστοίχιση των δεδομένων του πακέτου με αυτά του πίνακα ARP που σε περίπτωση μη αντιστοίχισης γίνεται νέα καταχώρηση σε αυτόν. Αν η φυσική διεύθυνση υπάρχει στον πίνακα και η διεύθυνση δικτύου, IP Address, αντιστοιχεί σε άλλη καταχώρηση τότε εντοπίζεται πιθανή επίθεση ARP Spoofing, αν όχι, το πακέτο οδηγείται στον προορισμό του.

Ο έλεγχος πραγματοποιείται με την άφιξη ενός πακέτου ARP και το κάλεσμα της μεθόδου packet_in. Η μέθοδος _packet_in_handler καλείται κάθε φορά εισέρχεται ένα νέο πακέτο στον controller και δεν υπάρχει κάποιος κανόνας στον πίνακα flow του μεταγωγέα ώστε να το διαχειριστεί διαφορετικά.

Αρχικά για τον εντοπισμό της επίθεσης γίνεται έλεγχος αν το πακέτο είναι τύπου ARP και αποθηκεύει το αποτέλεσμα στην μεταβλητή ¨arp_header¨ που σε περίπτωση που το πακέτο είναι τύπου ARP γεμίζει με τις πληροφορίες που περιέχει, διαφορετικά μένει κενή. Στη συνέχεια δημιουργείται η μεταβλητή eth_header στην οποία αποθηκεύονται πληροφορίες που αφορούν το σύνολο του εισερχόμενου πακέτου και θα χρησιμοποιηθούν κατά την απόρριψη κάπου πακέτου. Αφού η μεταβλητή arp_header γεμίσει με δεδομένα και δεν μένει κενή, τότε εκτελείται έλεγχος κατά πόσο είναι τύπου reply χρησιμοποιώντας τον κωδικό που αντιπροσωπεύει αυτού του είδους πακέτο, δηλαδή τον αριθμό 2. Αν το πακέτο είναι ARP Request τότε ο κωδικός έχει τιμή 1. Αν είναι πακέτο ARP Reply τότε καλείται η μέθοδος arp_rep_check συνοδευόμενη από τις πληροφορίες του μεταγωγέα, το μήνυμα, τη μεταβλητή arp_header, την πόρτα εισόδου, την πόρτα εξόδου και την κεφαλίδα ethernet, μεταβλητή eth_header, διαφορετικά το πακέτο συνεχίζει την πορεία του για τον προορισμό του. Στην περίπτωση που το αποτέλεσμα στην μεταβλητή arp_header είναι ψευδές τότε το πακέτο δεν είναι τύπου ARP και δεν

εκτελείται η διαδικασία ελέγχου επιτρέποντας στο πακέτο να προχωρήσει με τα υπόλοιπα βήματα που βρίσκονται στην μέθοδο _packet_in_handler.

```
def _packet_in_handler(self, ev):

    if ev.msg.msg_len < ev.msg.total_len:
        self.logger.debug("packet truncated: only %s of %s bytes",
                          ev.msg.msg_len, ev.msg.total_len)
    msg = ev.msg
    datapath = msg.datapath
    ofproto = datapath.ofproto
    parser = datapath.ofproto_parser
    in_port = msg.match['in_port']

    pkt = packet.Packet(msg.data)
    eth = pkt.get_protocols(ethernet.ethernet)[0]

    arp_header = pkt.get_protocol(arp.arp)
    eth_header = pkt.get_protocol(ethernet.ethernet)

    if arp_header:
        if arp_header.opcode == 2:
            self.arp_rep_check(datapath, msg, arp_header, in_port, out_port, eth_header)
        else:
            self.packet_flood(datapath, msg, in_port, out_port)
    else:
        self.packet_flood(datapath, msg, in_port, out_port)
```

Εικόνα 6.16: Μέθοδος _packet_in_handler και κώδικας ελέγχου πακέτου ARP

Αφού εντοπιστεί πακέτο τύπου ARP Reply και κληθεί η μέθοδος arp_rep_check, εξάγονται από τα δεδομένα που βρίσκονται στη μεταβλητή arp_header οι διευθύνσεις MAC και IP του αποστολέα του πακέτου και αποθηκεύονται στις αντίστοιχες μεταβλητές. Για σκοπούς ενημέρωσης, αν εντοπιστεί κάποιο πακέτο τύπου ARP Reply εμφανίζεται στο τερματικό του controller το κατάλληλο μήνυμα. Ακολούθως εκτελείται έλεγχος αντιστοίχισης της διεύθυνσης IP και MAC του αποστολέα από τον πίνακα ARP του μεταγωγέα. Αν δεν υπάρχει κάποια αντιστοίχιση τότε γίνεται προσθήκη στον πίνακα ARP και το πακέτο προωθείται στον προορισμό του. Στην περίπτωση που υπάρχει κάποια καταχώριση της διεύθυνσης IP στον πίνακα με διαφορετική διεύθυνση MAC από αυτή του πακέτου τότε εντοπίζεται πιθανή επίθεση ARP Spoofing και καλείται η μέθοδος packet_dropper. Αντίθετα αν η αντιστοίχιση των δύο διευθύνσεων είναι αληθές το πακέτο συνεχίζει την πορεία του για τον προορισμό του.

```
def arp_rep_check(self, datapath, msg, arp_header, in_port, out_port, eth_header):

    arp_mac=arp_header.src_mac
    arp_ip=arp_header.src_ip

     self.logger.info("New ARP reply from host: %s", arp_mac)
    if arp_ip not in self.ARP_Table:
        self.logger.info("New Host in ARP table")
        self.ARP_Table[arp_ip] = arp_mac
        self.packet_flood(datapath, msg, in_port, out_port)
    if self.ARP_Table[arp_ip] != arp_mac:
        self.packet_dropper(datapath, eth_header, arp_header, in_port)
        self.logger.info("Warning Host already in ARP table Blocking")
    else:
        self.packet_flood(datapath, msg, in_port, out_port)
```

Εικόνα 6.17: Μέθοδος arp_rep_check

Η μέθοδος packet_dropper χρησιμοποιείται για την προσθήκη κανόνα στον μεταγωγέα ώστε να απορρίπτει τα πακέτα που μεταδίδει η MAC διεύθυνση του θύτη. Η διαδικασία απόρριψης του πακέτου είναι απλή και απαιτεί μερικές ελάχιστες απαιτήσεις που θα καθορίζουν ποια πακέτα θε απορρίπτονται αποθηκεύοντας τα στη μεταβλητή match. Σε αυτή την περίπτωση τα στοιχεία που δίνονται πόρτα του μεταγωγέα από την οποία λήφθηκε το πακέτο, η φυσική διεύθυνση του παραλήπτη, η φυσική διεύθυνση του αποστολέα και ο τύπος του πακέτου. Προαιρετικό αλλά χρήσιμο χαρακτηριστικό είναι οι μεταβλητές hard_timeout και idle_timeout. Η μεταβλητή hard_timeout χρησιμοποιείται ώστε να ακυρωθεί ο κανόνας αναγκαστικά χωρίς προϋποθέσεις ενώ η idle_timeout χρησιμοποιείται για την ακύρωση του κανόνα μόνο αν δεν εντοπιστεί πακέτο με ίδια χαρακτηριστικά που ορίστηκαν στην μεταβλητή match.

```
def packet_dropper(self, datapath, eth_header, arp_header, in_port):
    ofproto = datapath.ofproto
    parser = datapath.ofproto_parser

    match = parser.OFPMatch(in_port=in_port, eth_dst=eth_header.dst,
                            eth_src=eth_header.src, eth_type=eth_header.ethertype)
    self.add_flow(datapath, 3, match, [], hard_timeout=60, idle_timeout=30)
```

Εικόνα 6.18: Μέθοδος packet_dropper

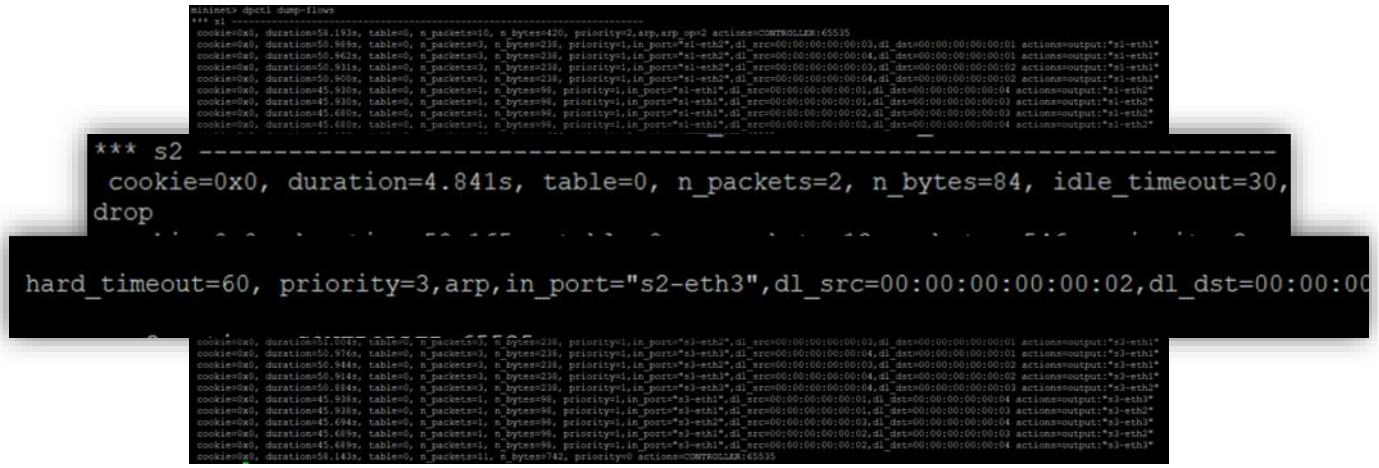
Στην περίπτωση που στο δίκτυο υπάρχει ενεργή εφαρμογή προστασίας η διαδικασία εκτέλεσης της επίθεσης παραμένει ίδια. Αφού εκτελεστεί ο controller RYU και ο προσομοιωτής Mininet εμφανίζεται στο τερματικό τους κατάλληλο μήνυμα. Το αποτέλεσμα της επίθεσης όμως είναι διαφορετικό, εφόσον ο εντοπισμός και η αντιμετώπιση λειτουργούν, ο Host1 δεν επηρεάζεται από την επίθεση και δεν αλλάζει ο πίνακας ARP του, αφού το δίκτυο εντοπίζει την κακόβουλη ενέργεια και την αποτρέπει αποκλείοντας τα κακόβουλα πακέτα που μεταδίδει ο Host 2. Κατά την υλοποίηση της μεθόδου προστασίας έχουν ενταχθεί στον κώδικα διάφορα μηνύματα που ενημερώνουν μέσα από το τερματικό τον διαχειριστή για τον εντοπισμό νέο πακέτου ARP Reply ενώ σε περίπτωση που είναι κακόβουλο ενημερώνει για τον εντοπισμό και την απόρριψη των πακέτων αυτών όπως φαίνεται στο ακόλουθο στιγμιότυπο του τερματικού.

```
ryu@ryu: ~
New ARP reply from host: 00:00:00:00:00:02
Warning Host already in ARP table Blocking
packet in 0000000000000002 00:00:00:00:00:02 00:00:00:00:00:01 3
New ARP reply from host: 00:00:00:00:00:02
Warning Host already in ARP table Blocking
packet in 0000000000000002 00:00:00:00:00:02 00:00:00:00:00:01 3
New ARP reply from host: 00:00:00:00:00:02
Warning Host already in ARP table Blocking
packet in 0000000000000002 00:00:00:00:00:02 00:00:00:00:00:01 3
New ARP reply from host: 00:00:00:00:00:02
Warning Host already in ARP table Blocking
packet in 0000000000000002 00:00:00:00:00:02 00:00:00:00:00:01 3
New ARP reply from host: 00:00:00:00:00:02
Warning Host already in ARP table Blocking
packet in 0000000000000002 00:00:00:00:00:02 00:00:00:00:00:01 3
New ARP reply from host: 00:00:00:00:00:02
Warning Host already in ARP table Blocking
packet in 0000000000000002 00:00:00:00:00:02 00:00:00:00:00:01 3
New ARP reply from host: 00:00:00:00:00:02
Warning Host already in ARP table Blocking
packet in 0000000000000002 00:00:00:00:00:02 00:00:00:00:00:01 3
New ARP reply from host: 00:00:00:00:00:02
Warning Host already in ARP table Blocking
```

Εικόνα 6.19: Εντοπισμός και απόρριψη πακέτων επίθεσης

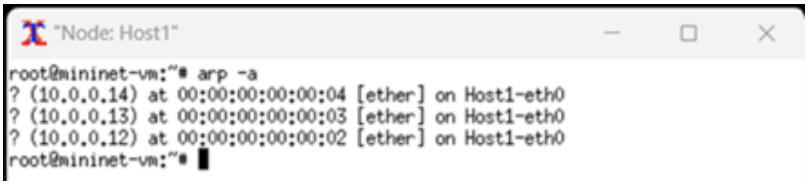
Αφού εντοπιστεί η επίθεση με τη χρήση της μεθόδου add_flow δημιουργείται μια νέα εγγραφή στον πίνακα ροών του μεταγωγέα που εντοπίστηκε η επίθεση. Μια εγγραφή του πίνακα ροών διαθέτει όλες τις πληροφορίες που δόθηκαν κατά την δημιουργία του από την εφαρμογή με κάποιες επιπλέον πληροφορίες που δίνονται κατά την δημιουργία του όπως φαίνονται πιο κάτω.

1. cookie=0x0: Μοναδικός κωδικός αναγνώρισης για κάθε κανόνα.
2. duration=4.841s: Χρόνος που βρίσκεται σε εφαρμογή η εγγραφή.
3. table=0: Αύξον αριθμός του πίνακα ροών.
4. n_packets=2: Πλήθος των πακέτων που αντιστοιχήθηκαν από τον κανόνα.
5. n_bytes=84: Το πλήθος των bytes που αντιστοιχήθηκαν.
6. idle_timeout=30: Χρόνος λήξης του κανόνα σε περίπτωση που δεν υπάρξει άλλη αντιστοίχιση.
7. drop: Τρόπος διαχείρισης των πακέτων που θα αντιστοιχιστούν.
8. hard_timeout=60: Χρόνος λήξης του κανόνα σε οποιεσδήποτε συνθήκες.
9. priority=3: Η προτεραιότητα του κανόνα.
10. arp: Υποδεικνύει ότι αντιστοιχείται με τα πακέτα ARP
11. in_port="s2-eth3": Πόρτα του μεταγωγέα από την οποία εισήλθε η επίθεση
12. dl_src=00:00:00:00:00:02: Η φυσική διεύθυνση του αποστολέα.
13. dl_dst=00:00:00:00:00:01: Η φυσική διεύθυνση του παραλήπτη



Εικόνα 6.20: Εγγραφή στον πίνακα ροής του μεταγωγέα 2.

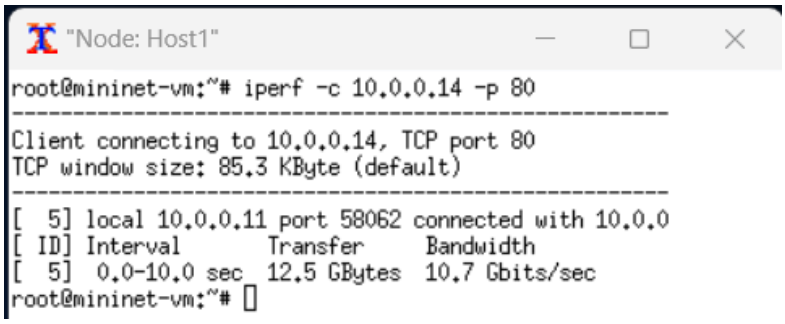
Η αποτυχία της επίθεσης μπορεί να επιβεβαιωθεί με τον έλεγχο των καταχωρήσεων του πίνακα ARP που βρίσκεται στο Host 1.



Εικόνα 6.21: Πίνακας ARP του Host 1 κατά την εκτέλεση της επίθεσης.

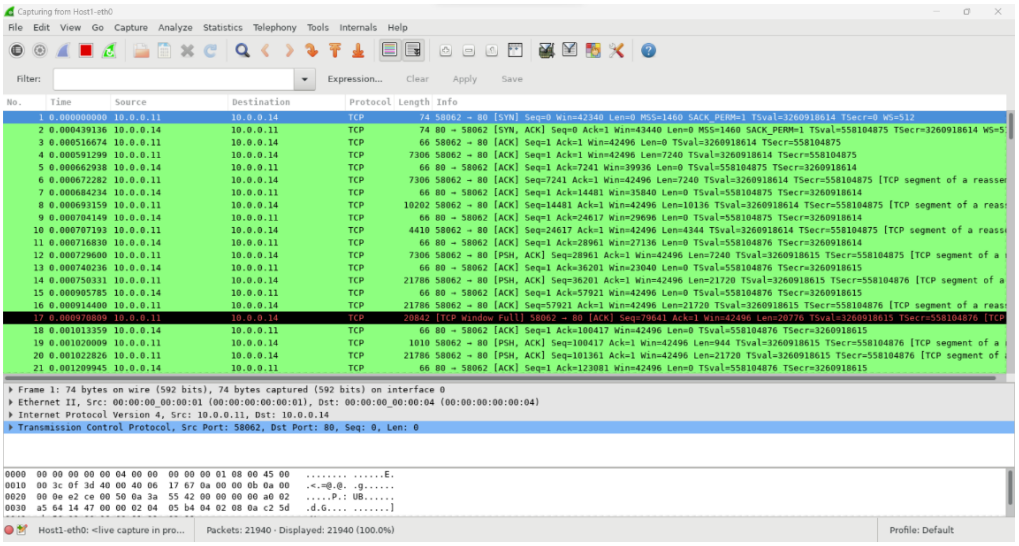
Εφόσον δεν επηρεάστηκε ο Host 1 από την επίθεση πιθανή επικοινωνία του με τον διακομιστή μπορεί να εκτελεστεί με επιτυχία. Για δοκιμαστικούς σκοπούς εκτελείται η εντολή:

```
# iperf -c 10.0.0.14 -p 80
```



Εικόνα 6.22: Επικοινωνία Host 1 με τον Host 4 κατά τη διάρκεια της επίθεσης

Πέρα από την επιτυχή επικοινωνία του Host 1 με τον διακομιστή η αποτυχία εκτέλεσης της επίθεσης μπορεί να επιβεβαιωθεί μέσα από το περιβάλλον του Wireshark. Αρχικά ο Host 1 δεν λαμβάνει κάποιο πακέτο από τον Host 2 που εκτελεί την επίθεση αλλά μόνο τα πακέτα που ανταλλάσσει με το διακομιστή και αυτό φαίνεται από τις εγγραφές source και destination που περιέχουν μόνο πακέτα από τους Χρήστες 1 και 4.



Εικόνα 6.23: Στιγμιότυπο των λαμβανόμενων πακέτων του host 1

6.3 Επιθέσεις DoS

Η ολοένα αυξανόμενη χρήση της τεχνολογίας με πρόσβαση στο διαδίκτυο έφερε την ανάγκη για την ανάπτυξη όλο και περισσότερων υπολογιστικών συστημάτων που προσφέρουν υπηρεσίες ιστού όπως οι Web, Mail και Cloud Storage. Ο μεγάλος αριθμός τέτοιων συστημάτων γίνεται καθημερινά στόχος επιθέσεων για την διακοπή των υπηρεσιών τους προκαλώντας προβλήματα στην καθημερινότητα των χρηστών που εξαρτώνται από αυτά.

Για την διακοπή των υπηρεσιών αυτών οι κακόβουλοι χρήστες εκτελούν επιθέσεις τύπου DoS. Όπως προαναφέρθηκε, οι επιθέσεις DoS χωρίζονται στις γρήγορες και τις αργές. Στο παρόν υποκεφάλαιο εκτελούνται δύο επιθέσεις, μια γρήγορη και μία αργή. Στην πρώτη περίπτωση εκτελείται επίθεση DoS από ένα host του δικτύου με σκοπό την εξάντληση των πόρων του διακομιστή. Στην δεύτερη, εκτελείται επίθεση Slowloris που στοχεύει το πρωτόκολλο HTTP/S και το μέγιστο πλήθος χρηστών που μπορεί να εξυπηρετήσει ο διακομιστής την ίδια στιγμή.

Η εκτέλεση των δυο προσομοιώσεων έχει σκοπό την ανάδειξη των διαφορών που τις κάνουν να ξεχωρίζουν δίνοντας όμως το ίδιο τελικό αποτέλεσμα, την διακοπή μιας υπηρεσίας, και των μεθόδων αντιμετώπισης τους.

6.3.1 DoS/DDoS

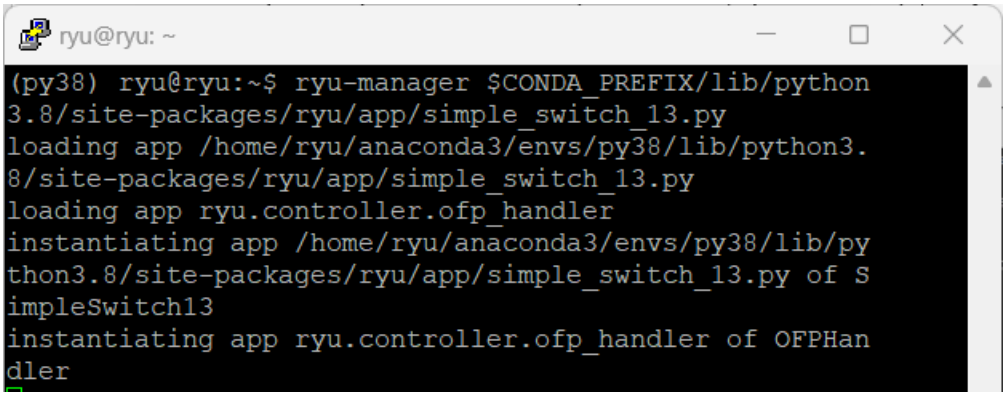
Όπως αναφέρεται στο κεφάλαιο 4 οι επιθέσεις DoS/DDoS έχουν σκοπό την εξάντληση των πόρων μιας συσκευής. Στην ακόλουθη προσομοίωση γίνεται ακριβώς αυτό, ο Host 1 επιχειρεί να εξαντλήσει τους πόρους του Host 4 που ενεργεί ως διακομιστής ιστού. Τα εργαλεία που χρησιμοποιούνται πέρα από τον προσομοιωτή Mininet και τον controller RYU είναι το iperf3 και το hping3 που περιεγράφηκαν στο κεφάλαιο 5.

Για τη μελέτη της επίθεσης η προσομοίωση εκτελείται σε τρεις φάσεις. Αρχικά εκτελείται το κοινό μέρος της επίθεσης που αφορά την χρήση των διάφορων εντολών. Στη συνέχεια παρουσιάζεται το αποτέλεσμα της επίθεσης χωρίς τη λήψη μέτρων προστασίας εκτελώντας το αρχείο simple_switch_l3.py που θα αποτελέσει τη βάση για την υλοποίηση της μεθόδου προστασίας. Στην τελευταία φάση θα επαναληφθεί η εκτέλεση της επίθεσης, αυτή τη φορά με μέτρο προστασίας εκτελώντας το αρχείο simple_switch_l3_ddos_detection.py που αποτελείται από το αρχείο simple_switch_l3.py τροποποιημένο κατάλληλα ώστε να παρέχει τη δυνατότητα εντοπισμού και αντιμετώπισης επιθέσεων DoS/DDoS.

6.3.1.1 Διαδικασία εκτέλεσης

Αρχικά για την εκτέλεση της προσομοίωσης γίνεται εκκίνηση του controller και του προσομοιωτή Mininet. Η εκκίνηση του controller γίνεται με την εντολή:

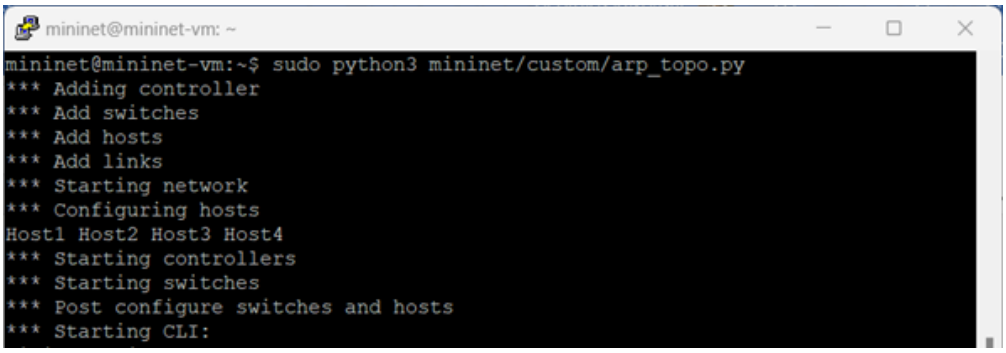
```
$ ryu-manager $CONDA_PREFIX/lib/python3.8/site-packages/ryu/app/simple_switch_13.py
```



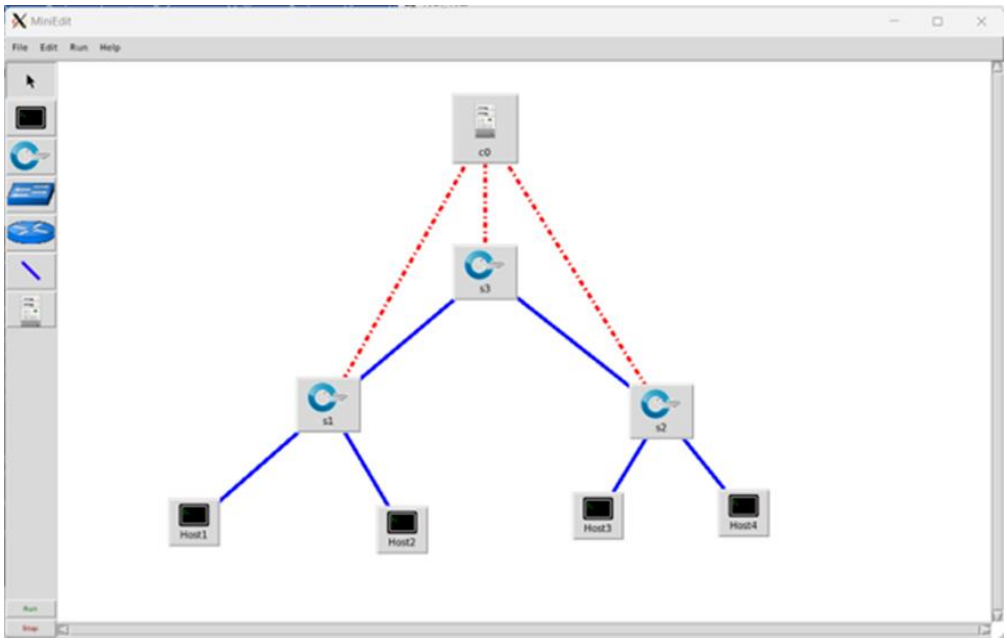
Εικόνα 6.24: Εκκίνηση controller RYU

Και η εκκίνηση του Mininet:

```
$ sudo python3 mininet/custom/simulations_topo.py
```



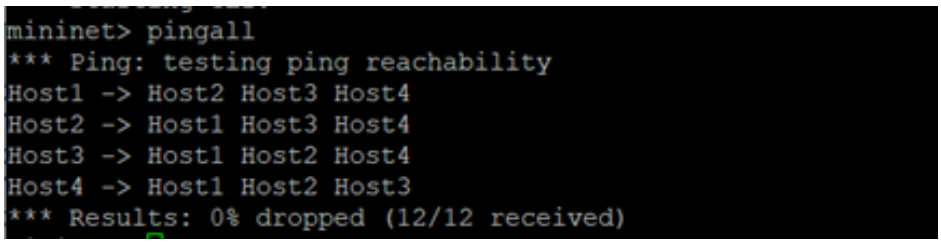
Εικόνα 6.25: Έναρξη προσομοιωτή Mininet



Εικόνα 6.26: Τοπολογία δικτύου

Αφού εκκινήσουν με επιτυχία είναι απαραίτητο να επιβεβαιωθεί η ορθή επικοινωνία μεταξύ των χρηστών και μπορεί να επιτευχθεί από το τερματικό του Mininet εκτελώντας την εντολή:

```
> pingall
```



Εικόνα 6.27: Εκτέλεση εντολής 'pingall'

Η επιτυχής επικοινωνία των χρηστών μπορεί να γίνει μέσα το τερματικό του Mininet καθώς με την ολοκλήρωση της επικοινωνίας των χρηστών εμφανίζονται οι πληροφορίες που αφορούν το ποσοστό αποτυχίας της, στην περίπτωση αυτή 0% αφού όλοι οι χρήστες κατάφεραν να ανταλλάξουν πακέτα ICMP μεταξύ τους.

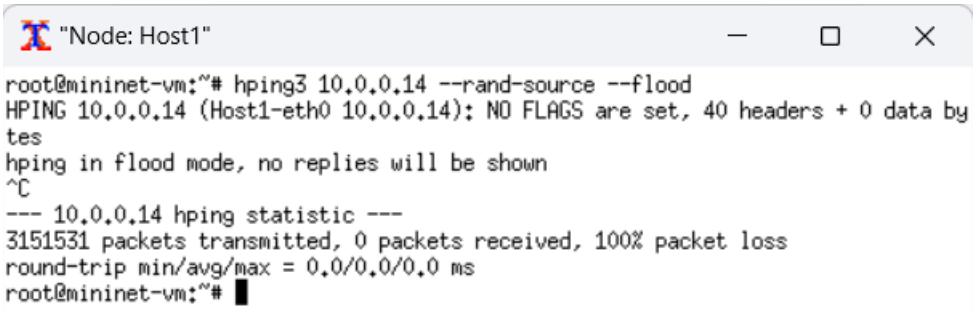
Η επίθεση πραγματοποιείται με τη χρήση του εργαλείου Hping3 και το κάλεσμα του με την ακόλουθη εντολή:

```
# hping3 10.0.0.14 –rand-source –flood
```

Η εντολή hping για να εκτελεστεί χρειάζεται κάποια ορίσματα που καθορίζουν τις παραμέτρους εκτέλεσης της που είναι τα ακόλουθα:

1. 10.0.0.14: Ο στόχος της επίθεσης, το θύμα
2. --rand-source: Θα χρησιμοποιήσει τυχαίες διευθύνσεις δικτύου
3. --flood: θα στείλει πολύ μεγάλο αριθμό πακέτων προς το θύμα

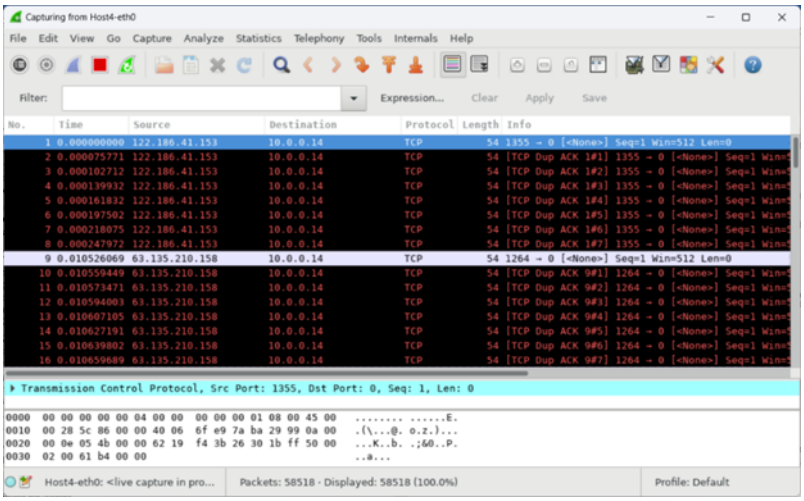
Στο στιγμιότυπο του τερματικού που ανήκει στον Host 1 φαίνεται η εκτέλεση της επίθεσης και παρουσιάζονται κάποια στοιχεία που την αφορούν. Η πρώτη πληροφορία που δίνει επιβεβαιώνει τη χρήση των ορισμάτων που δόθηκαν ενώ η δεύτερη ενημερώνει το host ότι καθώς εκτελείται σε μορφή flood δεν θα εμφανίζονται απαντήσεις από τον host με διεύθυνση δικτύου 10.0.0.14, δηλαδή τον Host 4 και με τη λήξη ή διακοπή της εμφανίζεται το αποτέλεσμα της εκτέλεσης της. Το αποτέλεσμα ενημερώνει τον host ότι μεταδόθηκαν 3.151.531 από τα οποία δεν απαντήθηκε κανένα με 100% απώλεια, πληροφορία που δεν είναι χρήσιμη καθώς ο στόχος δεν είναι να ληφθούν απαντήσεις από το θύμα.



Εικόνα 6.28: Εκτέλεση εντολής hping3

6.3.1.2 Αποτέλεσμα επίθεσης χωρίς προστασία

Για την παρουσίαση του αποτελέσματος της επίθεσης γίνεται χρήση του εργαλείου Wireshark το οποίο καταγράφει όλα τα πακέτα που λαμβάνει ο host στον οποίο εκτελείται. Μέσα από το περιβάλλον του Wireshark στον Host 4 φαίνονται τα πακέτα που έλαβε κατά τη διάρκεια της επίθεσης. Από τα αποτελέσματα της εκτέλεσης του εργαλείου hping3 φαίνεται ότι ο Host 4 δεν διάβασε όλα τα πακέτα που στάλθηκαν αλλά ένα μέρος τους. Αυτό υποδεικνύει ότι δεν διαθέτει τους απαραίτητους πόρους ώστε να λάβει και να επεξεργαστεί όλα τα πακέτα με τον ρυθμό που τα λαμβάνει με αποτέλεσμα να χάνονται όσα περισσεύουν.



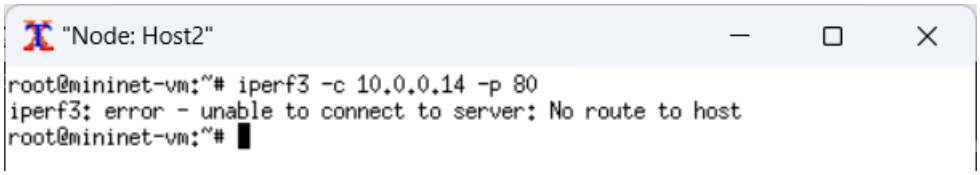
Εικόνα 6.29: Εφαρμογή Wireshark κατά την εκτέλεση της επίθεσης

Παράλληλα με την εκτέλεση της επίθεσης ο Host 2 επιχειρεί να επικοινωνήσει με τον Host 4 χωρίς επιτυχία αφού η επίθεση εξάντλησε τους πόρους του. Αυτό μπορεί να επαληθευτεί με την εκτέλεση του εργαλείου iperf και την ακόλουθη εντολή:

```
#iperf3 -c 10.0.0.14 -p 80
```

Με την εντολή είναι απαραίτητο να δοθούν τα κατάλληλα ορίσματα που καθορίζουν τις παραμέτρους εκτέλεσης και είναι τα πιο κάτω:

1. -c: Εκτέλεση σε λειτουργία client, δηλαδή περιηγητή σε συνδυασμό με την πόρτα
2. 10.0.0.14: Η διεύθυνση δικτύου διακομιστή
3. -p 80: Η πόρτα που δίνει πρόσβαση στον διακομιστή



Εικόνα 6.30: Απόπειρα επικοινωνίας με τον Host 4.

Με την εκτέλεση της εντολής αρχικά δεν εμφανίζεται κάποιο αποτέλεσμα άμεσα αφού γίνονται προσπάθειες επικοινωνίας με τον διακομιστή, όμως με τη λήξη των προσπαθειών επικοινωνίας εμφανίζεται μήνυμα ότι ήταν αδύνατη η επικοινωνία με τον διακομιστή, αποτέλεσμα της μη ανταπόκρισης.

6.3.1.3 Μέθοδος προστασίας και αποτέλεσμα επίθεσης

Για την αντιμετώπιση της επίθεσης DoS/DDoS που ακολουθεί γίνεται χρήση της μεθόδου με Εντροπία του Shannon η οποία με τη χρήση του τύπου που περιγράφηκε στο κεφάλαιο 4.3.1 υπολογίζει την πιθανότητα ύπαρξης μιας επίθεσης DDoS. Όπως προαναφέρθηκε η προτεινόμενη λύση ενσωματώνεται στην εφαρμογή simple_switch_l3.py.

Για την υλοποίηση του εντοπισμού με Εντροπία είναι απαραίτητο να οριστούν κάποιες μεταβλητές από τις οποίες δύο θα είναι σταθερές μεταβλητές και οι δύο θα γεμίζουν δυναμικά με δεδομένα κατά τη λειτουργία του δικτύου. Οι μεταβλητές αυτές είναι απαραίτητο να δημιουργηθούν κατά την εκκίνηση

του controller και των μεταγωγέων μέσα από την μέθοδο __init__ που χρησιμοποιείται για την αρχικοποίηση μεταβλητών.

```
def __init__(self, *args, **kwargs):
    super(SimpleSwitch13, self).__init__(*args, **kwargs)
    self.mac_to_port = {}

    self.window_size = 100
    self.entropy_threshold = 2.5
    self.source_ips = []
    self.ip_to_port = {}
```

Εικόνα 6.31: Μέθοδος __init__.

Οι μεταβλητές που δημιουργούνται είναι οι εξής

1. Self.window_size = 100: Κάθε πόσα εισερχόμενα πακέτα θα πραγματοποιείται ο έλεγχος
2. Self.entropy_threshold = 2.5: Όριο της εντροπίας
3. Self.source_ips: πίνακας με τις διευθύνσεις δικτύου από τα εισερχόμενα πακέτα.
4. Self.ip_to_port: Πίνακας για την αντιστοίχιση διευθύνσεων IP και πόρτας μεταγωγέα.

Αφού οριστικοποιηθούν οι μεταβλητές σειρά έχει η διαδικασία ελέγχου και εντοπισμού της επίθεσης. Στο στάδιο αυτό ο έλεγχος πραγματοποιείται με την είσοδο κάποιου πακέτου σε κάποιο μεταγωγέα όπου καλείται η μέθοδος _packet_in_handler. Σημαντικό δεδομένο σε αυτό το σημείο είναι ότι η διαδικασία εισαγωγής κανόνα για την παράβλεψη της μεθόδου έχει καταργηθεί από την διαδικασία εκτέλεσης του simple_switch_l3.py για τους σκοπούς της προσομοίωσης. Αρχικά αφού κληθεί η μέθοδος δημιουργείται μια μεταβλητή με όνομα ip_pkt στην οποία αποθηκεύονται οι πληροφορίες του πακέτου αν είναι τύπου IPV4. Αφού δημιουργηθεί η μεταβλητή πραγματοποιείται έλεγχος, όπου αν είναι αληθές εξάγεται από αυτήν η διεύθυνση δικτύου του αποστολέα και αποθηκεύεται στην μεταβλητή src_ip. Ακολούθως προστίθεται στον πίνακα source_ips. Στη συνέχεια γίνεται έλεγχος αν η διεύθυνση υπάρχει στον πίνακα ip_to_port ώστε να καταχωρηθεί. Για να μην ξεπεραστεί το όριο του window_size πραγματοποιείται ακόμα ένας έλεγχος προτού γίνει ο οποιοσδήποτε άλλος έλεγχος που αφορά την εντροπία. Ο έλεγχος ελέγχει αν ο πίνακας διαθέτει το πλήθος του window_size και σε περίπτωση που αυτός ξεπεραστεί αφαιρεί τις παλαιότερες καταχωρήσεις από πάνω προς τα κάτω.

Όταν ολοκληρωθεί η καταχώρηση των στοιχείων στους πίνακες ακολουθεί η διαδικασία ελέγχου της εντροπίας. Προτού υπολογιστεί η εντροπία ελέγχεται το μήκος του πίνακα source_ips ώστε να εξακριβωθεί αν διαθέτει το πλήθος τον διευθύνσεων IP που καθορίστηκαν. Αφού συμπληρώθηκε ο αριθμός που απαιτείται καλείται η μέθοδος calculate_entropy στην οποία δίνεται όρισμα ο πίνακας source_ips. Η μέθοδος calculate_entropy, με την ολοκλήρωση εκτέλεσης της, επιστρέφει την τιμή της εντροπίας που χρησιμοποιείται στον επόμενο έλεγχο που καθορίζει την ύπαρξη επίθεσης. Για τον καθορισμό ύπαρξης επίθεσης γίνεται έλεγχος για το κατά πόσο το αποτέλεσμα της εντροπίας είναι μεγαλύτερο από την τιμή της μεταβλητής self.entropy_threshold. Αν κατά τον έλεγχο που πραγματοποιήθηκε η τιμή της εντροπίας είναι μεγαλύτερη καλείται η μέθοδος ddos_defense με ορίσματα το datapath, πληροφορία που δείχνει από που λήφθηκε το πακέτο, τον πίνακα source_ips και την IP του αποστολέα ενημερώνοντας παράλληλα μέσα από το τερματικό για την ύπαρξη επίθεσης.

Όταν ολοκληρωθεί η εκτέλεση της μεθόδου calculate_entropy επιστρέφεται το αποτέλεσμα της εντροπίας και αποθηκεύεται στη μεταβλητή entropy από όπου κλήθηκε. Στη συνέχεια ελέγχεται το αποτέλεσμα της εντροπίας με τη μεταβλητή entropy_threshold που καθορίστηκε στην αρχή της εφαρμογής. Αν η εντροπία είναι μεγαλύτερη από το προκαθορισμένο όριο τότε υπάρχει πιθανή επίθεση DOS/DDoS και καλείται η μέθοδος ddos_defense.

```
@set_ev_cls(ofp_event.EventOFPPacketIn, MAIN_DISPATCHER)
def _packet_in_handler(self, ev):
    if ev.msg.msg_len < ev.msg.total_len:
        self.logger.debug("packet truncated: only %s of %s bytes",
                          ev.msg.msg_len, ev.msg.total_len)
    msg = ev.msg
    datapath = msg.datapath
    ofproto = datapath.ofproto
    parser = datapath.ofproto_parser

    in_port = msg.match['in_port']

    pkt = packet.Packet(msg.data)
    eth = pkt.get_protocols(ethernet.ethernet)[0]

    ip_pkt = pkt.get_protocol(ipv4.ipv4)

    if ip_pkt:

        src_ip = ip_pkt.src

        self.source_ips.append(src_ip)

        if src_ip not in self.ip_to_port:
           self.ip_to_port[src_ip] = in_port

        if len(self.source_ips) > self.window_size:
            self.source_ips.pop(0)

        if len(self.source_ips) == self.window_size:
            entropy = self.calculate_entropy(self.source_ips)
            if entropy > self.entropy_threshold:
                self.ddos_defence(datapath, self.source_ips, src_ip)
                self.logger.warning("Potential DDoS Attack Detected! Entropy: %f", entropy)
```

Εικόνα 6.32: Μέθοδος packet_in_handler.

Η μέθοδος calculate_entropy βασίζεται στο πλήθος των καταχωρήσεων του πίνακα source_ips και τον υπολογισμό του μαθηματικού τύπου της εντροπίας. Στην περίπτωση που το πλήθος είναι λιγότερο ή 1 τότε η εντροπία αυτόματα γίνεται 0. Ο συγκεκριμένος έλεγχος γίνεται για προαιρετικούς λόγους αφού η μέθοδος δεν καλείται αν το αποτέλεσμα είναι μικρότερο από την τιμή της μεταβλητής window_size.

Η διαδικασία υπολογισμού της εντροπίας γίνεται σε τρία βασικά βήματα. Το πρώτο βήμα είναι ο υπολογισμός του πλήθους πραγματικών διευθύνσεων με τη χρήση της μεθόδου values που επιστρέφει τη μεταβλητή count που περιέχει το αποτέλεσμα. Το δεύτερο βήμα εκτελεί διαίρεση του προηγούμενου αποτελέσματος, count, με το σύνολο των καταχωρήσεων του πίνακα source_ips. Το τρίτο και πιο σημαντικό βήμα είναι η εκτέλεση του τύπου της εντροπίας, όπου χρησιμοποιείται η βιβλιοθήκη Maths της Python.

```
def calculate_entropy(self, source_ips):

    total_count = len(source_ips)
    if total_count <= 1:
        return 0

    counts = Counter(source_ips)
    entropy = 0

    for count in counts.values():
        p = count / total_count
        entropy -= p * math.log2(p)

    return entropy
```

Εικόνα 6.33: Μέθοδος calculate_entropy.

Για την αντιμετώπιση της επίθεσης εκτελούνται τέσσερις ενέργειες. Αρχικά δημιουργείται η μεταβλητή counter στην οποία αποθηκεύονται οι διευθύνσεις δικτύου του πίνακα source_ips και εκτελείται για να

υπολογιστεί το πόσες φορές εμφανίζεται μια διεύθυνση. Ακολούθως, στην μεταβλητή bad_ips αποθηκεύονται όλες οι διευθύνσεις που εμφανίζονται λιγότερες από 4 φορές χρησιμοποιώντας την μεταβλητή counter που δημιουργήθηκε στο προηγούμενο βήμα. Στη συνέχεια, γίνεται έλεγχος των διευθύνσεων της μεταβλητής bad_ips σε σύγκριση με αυτές του πίνακα ip_to_port. Από το αποτέλεσμα του ελέγχου εξάγεται η πόρτα του μεταγωγέα από την οποία προέρχεται η επίθεση και καλείται η μέθοδος block_port. Στη συνέχεια ο πίνακας source_ips αδειάζει ώστε να μην επηρεαστούν άλλοι χρήστες που επιχειρούν να επικοινωνήσουν στη συνέχεια.

```
def ddos_defence(self, datapath, source_ips, src_ip):

    counter = Counter(source_ips)
    self.bad_ips  = [value for value, count in counter.items() if count < 4]

    if src_ip in self.bad_ips:
       if src_ip in self.ip_to_port:
          self.port_to_block = self.ip_to_port.get(src_ip, [])
          self.logger.info("DDoS from port %s, Blocking", self.port_to_block)
          self.block_port(datapath, self.port_to_block)
          self.source_ips = []
```

Εικόνα 6.34: Μέθοδος ddos_defense.

Η μέθοδος block_port δέχεται ορίσματα το datapath και την μεταβλητή με τις πόρτες από τις οποίες πρέπει να απορρίπτονται τα πακέτα. Με μέθοδο block_port δημιουργούνται όλες οι μεταβλητές που διαθέτουν τις απαραίτητες πληροφορίες για την καταχώρηση του κανόνα ροής. Οι πληροφορίες αυτές είναι είναι το datapath, που δηλώνει την ταυτότητα τον μεταγωγέα, η πόρτα από την οποία πρέπει να απορρίπτονται τα πακέτα και την προτεραιότητα του κανόνα.

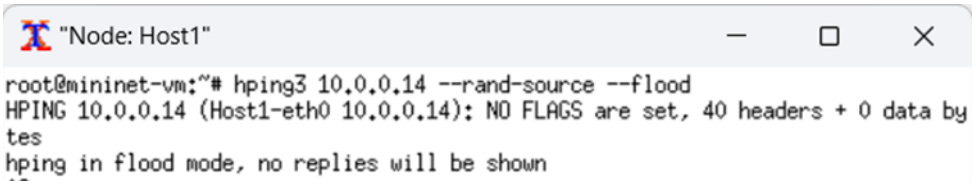
```
def block_port(self, datapath, port_to_block):
    ofproto = datapath.ofproto
    parser = datapath.ofproto_parser

    match = parser.OFPMatch(in_port=port_to_block)
    actions = []

    self.add_flow(datapath, priority=10, match=match, actions=actions)
```

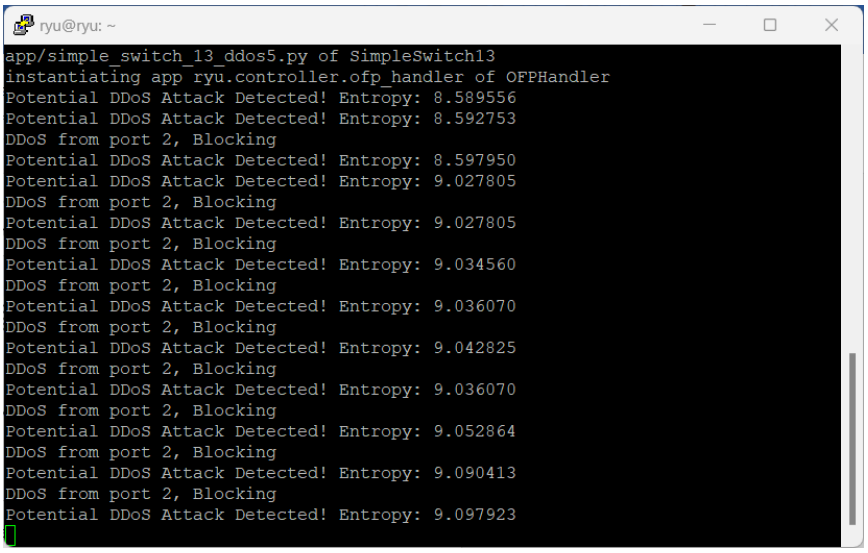
Εικόνα 6.35: Μέθοδος block_port

Κατά την εκτέλεση της επίθεσης εμφανίζονται στο τερματικό του Host 1 πληροφορίες που αφορούν τα πακέτα της επίθεσης καθώς η ενημέρωση για την μη εμφάνιση απαντήσεων στα πακέτα αυτά.



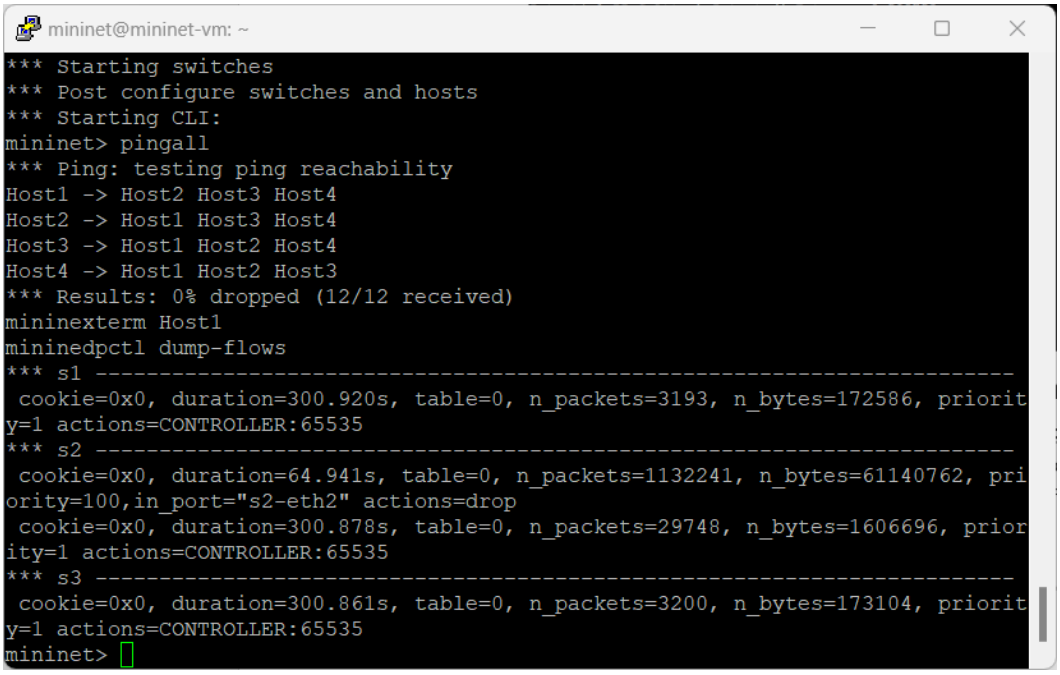
Εικόνα 6.36: Τερματικό Host 1 κατά την επίθεση

Με την συμπλήρωση του πλήθους των πακέτων στον controller πραγματοποιούνται οι ελέγχου και ο εντοπισμός της εντροπίας που περιγράφηκε παραπάνω. Σε περίπτωση εντοπισμού εμφανίζονται στο τερματικό του controller οι πληροφορίες που ορίστηκαν στην εφαρμογή.



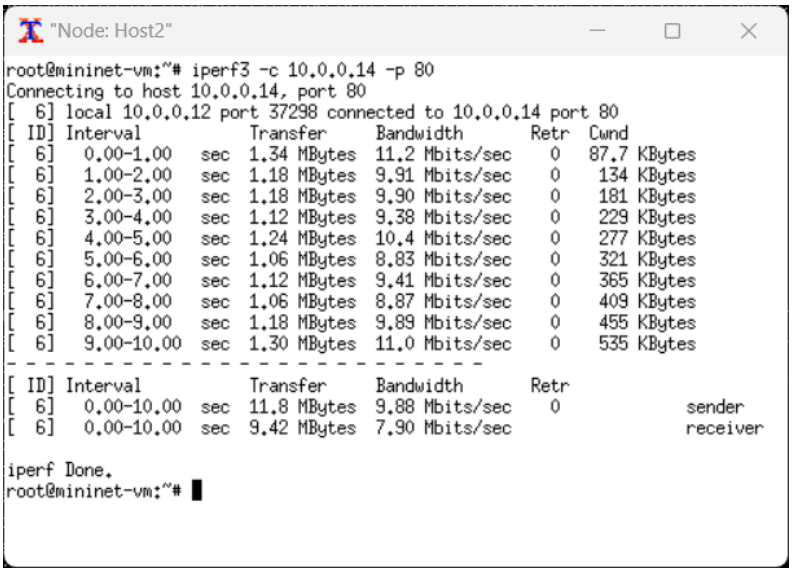
Εικόνα 6.37: Τερματικό controller κατά τον εντοπισμό της επίθεσης.

Επιπλέον μέσα από το τερματικό του Mininet και την εκτέλεση της εντολής <εντολή> φαίνεται ο κανόνας που προστέθηκε για την απόρριψη των πακέτων που προέρχονται από την πόρτα του μεταγωγέα και παράλληλα του host που είναι συνδεδεμένος σε αυτή με τις παραμέτρους που αναφέρθηκαν πιο πάνω. Από το πιο κάτω στιγμιότυπο τερματικού του προσομοιωτή Mininet και τον κανόνα που δημιουργήθηκε φαίνεται ότι η επίθεση εντοπίστηκε στον μεταγωγέα S2 και την πόρτα 2. Από την δομή της τοπολογίας επιβεβαιώνεται ότι ο Host που εκτέλεσε την επίθεση βρίσκεται συνδεδεμένος στο δίκτυο από τον μεταγωγέα S2.



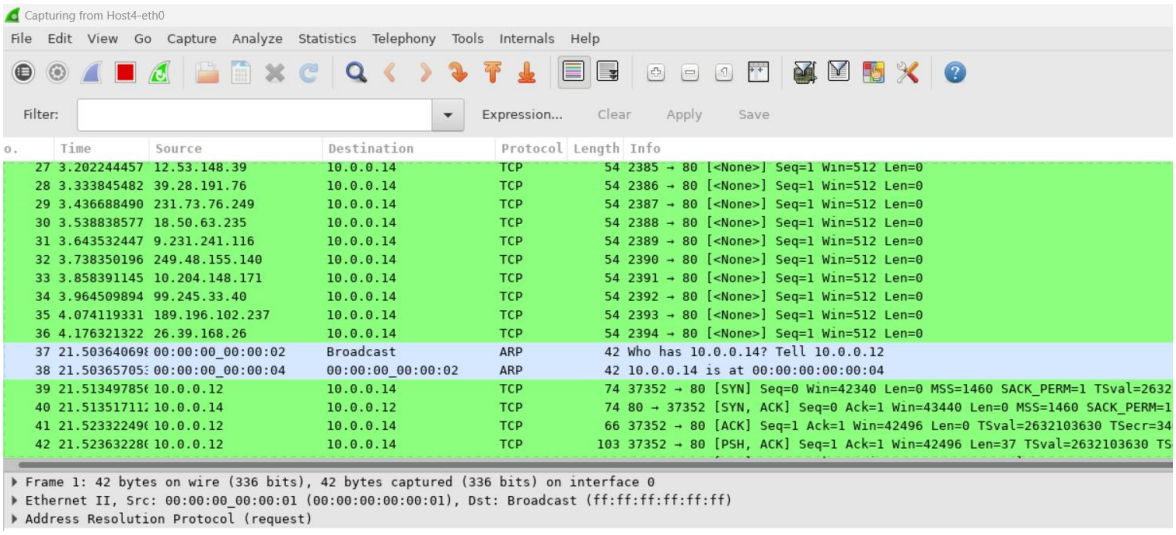
Εικόνα 6.38: Εγγραφή στον πίνακα ροών του μεταγωγέα 2.

Αφού εντοπιστεί η επίθεση και διακοπεί η ροή πακέτων από την πόρτα του μεταγωγέα, ο Host 2 επιχειρεί να επικοινωνήσει με τον διακομιστή που και αυτός είναι συνδεδεμένος από τον ίδιο μεταγωγέα. Παρατηρώντας το αποτέλεσμα της επικοινωνίας τους φαίνεται ότι αυτή έγινε με επιτυχία.



Εικόνα 6.39: Επικοινωνία Host 1 με τον διακομιστή κατά την επίθεση

Επιπλέον η επιτυχία της αντιμετώπισης της επίθεσης μπορεί να επιβεβαιωθεί από την εφαρμογή Wireshark στον Host 4 όπου διακόπηκε η λήψη των πακέτων από τον Host 1 αλλά έγινε επιτυχώς από τον Host 2. Στο στιγμιότυπο οθόνης του Host 4 φαίνονται τα πρώτα πακέτα που μετέδωσε ο Host 1 αφού δεν είχε συμπληρωθεί ο αριθμός των ελάχιστων πακέτων για την εκτέλεση του ελέγχου. Τα πακέτα αυτά είναι ευδιάκριτα κατά την προσομοίωση καθώς οι πηγαίες διευθύνσεις δικτύου τους είναι άγνωστες και δεν ανήκουν στους χρήστες που δημιουργήθηκαν κατά την υλοποίηση της τοπολογίας του δικτύου.



Εικόνα 6.40: Πακέτα που έφτασαν στο θύμα πριν την εφαρμογή του κανόνα ροής

6.3.2 Slowloris

Μια επίθεση Slowloris έχει στόχο την εξάντληση των πόρων ενός πρωτοκόλλου σε σύγκριση με τις γνωστές DoS/DDoS. Στην προσομοίωση εκτέλεσης της επίθεσης Slowloris ο Host 2 επιχειρεί την διακοπή των λειτουργιών του πρωτοκόλλου TCP στον διακομιστή που εκτελείται με τη χρήση του εργαλείου iperf στον Host 4.

Η προσομοίωση χωρίζεται σε 3 μέρη:

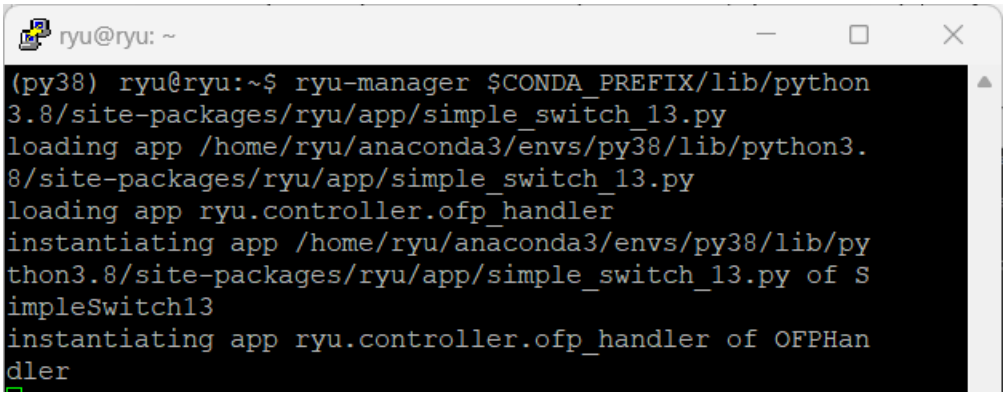
1. Διαδικασία εκτέλεσης.
2. Αποτέλεσμα επίθεσης χωρίς προστασία
3. Εφαρμογή προστασίας και αποτέλεσμα επίθεσης.

6.3.2.1 Διαδικασία εκτέλεσης

Για την εκτέλεση της επίθεσης γίνεται χρήση του controller RYU, του προσομοιωτή Mininet και των εργαλείων iperf και slowhttptest.

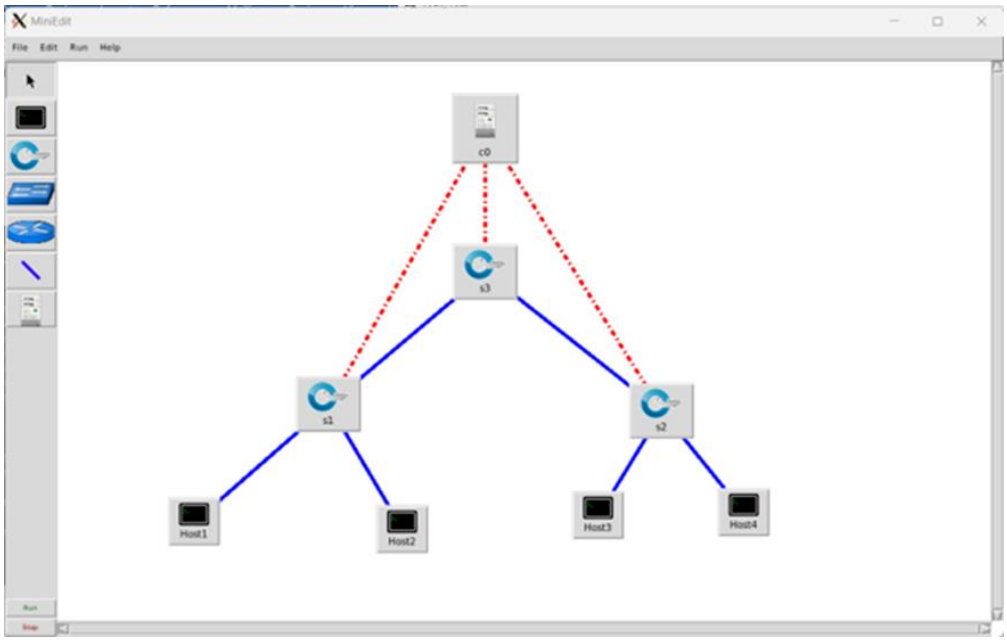
Το πρώτο βήμα είναι η εκκίνηση του controller RYU και του προσομοιωτή Mininet χρησιμοποιώντας τις ακόλουθες εντολές στο τερματικό της κάθε εικονικής μηχανής αντίστοιχα δίνοντας ορίσματα το όνομα και τη θέση του αντίστοιχου αρχείου.

```
$ ryu-manager $CONDA_PREFIX/lib/python3.8/site-packages/ryu/app/simple_switch_13.py
```



Εικόνα 6.41: Εκκίνηση controller RYU

```
$ sudo python3 mininet/custom/simulations_topo.py
```



Εικόνα 6.42: Τοπολογία δικτύου

Αφού εκτελεστούς οι εντολές στην οθόνη των τερματικών εμφανίζεται κατάλληλο μήνυμα που επιβεβαιώνει την εκκίνηση τους, διαφορετικά εμφανίζεται μήνυμα σφάλματος.

Βασικό βήμα για την ορθή εκτέλεση της προσομοίωση είναι η επαλήθευση και επιβεβαίωση της ορθής επικοινωνίας των συσκευών του δικτύου με την εκτέλεση της εντολής pingall μέσα από το τερματικό του Mininet. Το αποτέλεσμα εκτέλεσης της εντολής καθορίζει την επιτυχία επικοινωνίας των χρηστών και αυτό φαίνεται από το ποσοστό αποτυχίας κατά την επικοινωνία τους που στην παρούσα περίπτωση είναι 0%, δηλαδή όλοι οι χρήστες αντάλλαξαν πακέτα μεταξύ τους με επιτυχία.

```
mininet> pingall
*** Ping: testing ping reachability
Host1 -> Host2 Host3 Host4
Host2 -> Host1 Host3 Host4
Host3 -> Host1 Host2 Host4
Host4 -> Host1 Host2 Host3
*** Results: 0% dropped (12/12 received)
mininet>
```

Εικόνα 6.43: Εκτέλεση εντολής pingall.

Το δεύτερο στάδιο εκτέλεσης της προσομοίωσης είναι η εκκίνηση του Web Server στον host 4 που θα εκτελεί το ρόλο του διακομιστή. Για την εκτέλεση του Web Server χρησιμοποιείται το εργαλείο iperf που με τη χρήση των κατάλληλων ορισμάτων ενεργεί ως διακομιστής ή πελάτης. Για να εκκινήσει ως διακομιστής χρησιμοποιούνται τα ακόλουθα ορίσματα:

1. -s: εκτέλεση ως διακομιστής
2. -p: πόρτα στην οποία θα ακούει, συνήθως για διακομιστή ιστού χρησιμοποιείται η 80

```
"Node: Host4"
root@mininet-vm:~# iperf -s -p 80
------------------------------------------------------------
Server listening on TCP port 80
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
```

Εικόνα 6.44: Εκκίνηση διακομιστή με τη χρήση του εργαλείου iperf.

Για την εκτέλεση επικοινωνίας από κάποιο host χρησιμοποιείται και πάλι το εργαλείο iperf με διαφορετικά ορίσματα αυτή τη φορά όπως πιο κάτω:

1. -c: ενεργεί ως πελάτης ακολουθούμενο από τη διεύθυνση δικτύου του διακομιστή
2. -p: η πόρτα στην οποία θα γίνει η απόπειρα επικοινωνίας

```
root@mininet-vm:~# iperf -c 10.0.0.14 -p 80
------------------------------------------------------------
Client connecting to 10.0.0.14, TCP port 80
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  5] local 10.0.0.11 port 44058 connected with 10.0.0.14 port 80
[ ID] Interval       Transfer     Bandwidth
[  5]  0.0-10.0 sec  30.7 GBytes  26.4 Gbits/sec
root@mininet-vm:~#
```

Εικόνα 6.45: Αποτέλεσμα της επικοινωνίας server-client.

Με την εκκίνηση του διακομιστή ιστού και την επιτυχή επικοινωνία από τον host εμφανίζονται πληροφορίες που αφορούν την επικοινωνία των δύο πλευρών, όπως ο όγκος των δεδομένων και η χωρητικότητα του μέσου επικοινωνίας τους.

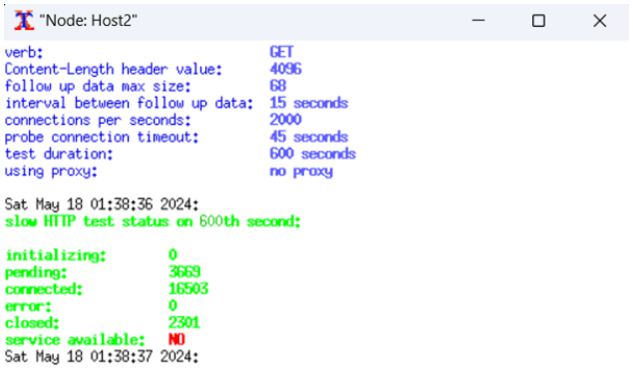
Το τρίτο στάδιο της προσομοίωσης αφορά την εκτέλεση της επίθεσης. Για την εκτέλεση της επίθεσης χρησιμοποιείται το εργαλείο slowhttptest με την εκτέλεση της εντολής:

```
# slowhttptest -H -c 20 -r 5 -u http://10.0.0.14
```

και για τους σκοπούς της προσομοίωσης χρησιμοποιούνται τα ακόλουθα ορίσματα:

1. -H: καθορίζει ότι ο τύπος της επίθεσης είναι Slowloris
2. -c: ορίζει το μέγιστο πλήθος των ταυτόχρονων ανοιχτών συνδέσεων
3. -r: το πλήθος των νέων συνδέσεων ανά δευτερόλεπτο
4. -u: η διεύθυνση δικτύου του θύματος

Κατά την εκτέλεση του εργαλείου στην οθόνη του host εμφανίζονται διάφορες πληροφορίες που αφορούν την διαδικασία και τις παραμέτρους που δόθηκαν μέσω των προαναφερθέντων ορισμάτων και αναγράφονται με μπλε γράμματα. Στη συνέχεια εμφανίζονται πληροφορίες που υποδεικνύουν την πορεία εκτέλεσης της επίθεσης όπως ο χρόνος εκτέλεσης, τα πακέτα που βρίσκονται σε ουρά αναμονής, οι ενεργές και ολοκληρωμένες συνδέσεις ενώ παράλληλα εμφανίζει την διαθεσιμότητα του διακομιστή κατά την διάρκεια της επίθεσης, οι πληροφορίες αυτές αναγράφονται με πράσινα γράμματα για να ξεχωρίζουν από τις πληροφορίες που δόθηκαν από το host που είναι απλά ενημερωτικές.



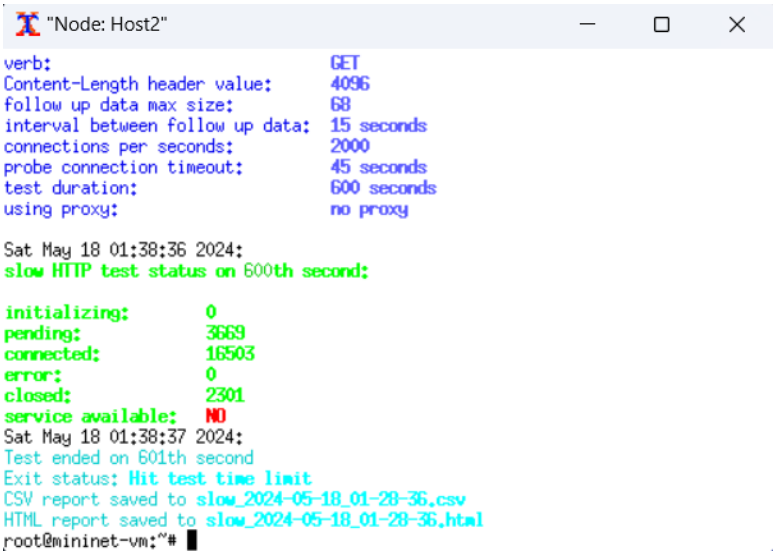
Εικόνα 6.46: Τερματικό Host 2 κατά την εκτέλεση της επίθεσης

6.3.2.2 Αποτέλεσμα επίθεσης χωρίς προστασία

Η επιτυχία της επίθεσης φαίνεται από το στιγμιότυπο τερματικού του Host 1, όπου η απόπειρα επικοινωνίας του με τον διακομιστή ιστού δεν έγινε επιτυχώς εμφανίζοντας το κατάλληλο μήνυμα για κάθε προσπάθεια. Από το μήνυμα που εμφανίστηκε παρουσιάζεται ότι έγιναν 5 αποτυχημένες προσπάθειες καθώς η υπηρεσία δεν είναι διαθέσιμη προσωρινά.

Εικόνα 6.47: Απόπειρα επικοινωνίας Host 1 με το διακομιστή κατά τη διάρκεια της επίθεσης.

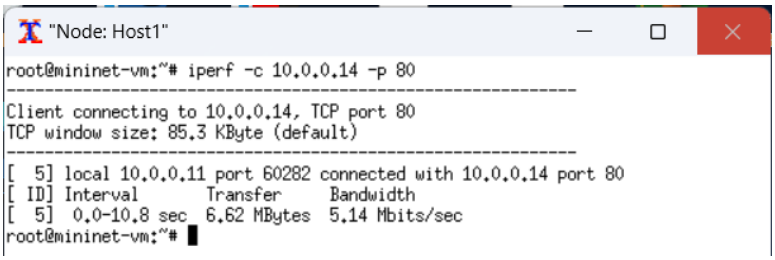
Με την ολοκλήρωση ή τη διακοπή της επίθεσης από το host εμφανίζονται πληροφορίες που ενημερώνουν το host για το αποτέλεσμα της επίθεσης από την αρχή μέχρι το τέλος. Στο ακόλουθο στιγμιότυπο παρουσιάζεται το αποτέλεσμα της ολοκληρωμένης επίθεσης Slowloris που εκτελέστηκε για 600 δευτερόλεπτα, δηλαδή 10 λεπτά. Στην παρούσα εκτέλεση της επίθεσης δημιουργήθηκαν 2000 συνδέσεις το δευτερόλεπτο με μέγιστο χρόνο εκτέλεσης της κάθε σύνδεσης για 45 δευτερόλεπτα.



Εικόνα 6.48: Αποτέλεσμα επίθεσης Slowloris χωρίς προστασία.

Με τη λήξη της επίθεσης παρατηρείται ότι το μέγιστο πλήθος ενεργών συνδέσεων που μπόρεσε να διαχειριστεί ο διακομιστής ιστού είναι 16503 με αποτέλεσμα να μην μπορεί να διαχειριστεί άλλες και να τεθεί εκτός λειτουργίας. Κατά την εκτέλεση ολοκληρώθηκαν 2301 συνδέσεις και δεν εξυπηρετήθηκαν 3669 αιτήματα επικοινωνίας μέχρι και την λήξη του χρόνου εκτέλεσης.

Αφότου ολοκληρώθηκε η επίθεση ο Host 1 επιχείρησε να επικοινωνήσει ξανά με τον διακομιστή, αυτή τη φορά με επιτυχία. Αυτό δείχνει ότι με την λήξη της επίθεσης ο διακομιστής επέστρεψε στα κανονικά επίπεδα λειτουργίας του με αποτέλεσμα να μπορεί να εξυπηρετήσει άλλα αιτήματα.



Εικόνα 6.49: Επιτυχής επικοινωνία Host 1 με το διακομιστή μετά τη λήξη της επίθεσης.

6.3.2.3 Εφαρμογή προστασίας και αποτέλεσμα επίθεσης

Το τρίτο στάδιο της προσομοίωσης εκτελείται με τη χρήση μέσου προστασίας. Για την προστασία των χρηστών του δικτύου εκτελείται η μέθοδος εντοπισμού και αντιμετώπισης. Η μέθοδος που επιλέγηκε για την παρούσα προσομοίωση γίνεται με την παρακολούθηση των ανοιχτών συνδέσεων από την διαδικασία χαιρετισμού τριών σταδίων. Η διαδικασία αυτή αφορά την έναρξη μέχρι και την λήξη της επικοινωνίας όπως περιγράφηκε στο Κεφάλαιο 5.

Η διαδικασία εντοπισμού και αντιμετώπισης της επίθεσης εκτελείται με την εκκίνηση του controller και του Mininet. Αρχικά μέσα από την μέθοδο __init__ που εκτελείται κατά την εκκίνηση του κάθε μεταγωγέα προστίθεται ο πίνακας connection_counter στον οποίο θα αποθηκεύεται το πλήθος ανοιχτών συνδέσεων από κάθε διεύθυνση IP.

```
def __init__(self, *args, **kwargs):
    super(SimpleSwitch13, self).__init__(*args, **kwargs)
    self.mac_to_port = {}
    self.connection_counter = {}
```

Εικόνα 6.50: Μέθοδος __init__.

Με την άφιξη κάποιου πακέτου στον μεταγωγέα και την εκτέλεση της μεθόδου packet_in_handler δημιουργούνται δύο νέες μεταβλητές για την αποθήκευση των απαιτούμενων πληροφοριών που θα χρησιμοποιηθούν στην συνέχεια. Στην μεταβλητή tcp_pkt αποθηκεύονται τα δεδομένα του πακέτου εφόσον είναι τύπου TCP, ενώ στη δεύτερη αποθηκεύονται οι πληροφορίες του πακέτου εφόσον είναι τύπου IPV4. Ακολούθως για την μείωση του φόρτου στον controller το πακέτο ορίστηκε ότι πρέπει να είναι τύπου TCP. Αφού το πακέτο είναι τύπου TCP καλείται η μέθοδος session_check στην οποία δίνονται οι ακόλουθοι παράμετροι:

1. Datapath: Τα στοιχεία της πόρτας και του μεταγωγέα που εισήλθε το πακέτο.
2. Eth_header: Η κεφαλίδα του πακέτου
3. In_port: η πόρτα που εισήλθε το πακέτο
4. Tcp_pkt: Η μεταβλητή tcp_pkt
5. Ipv4_pkt: Η μεταβλητή ipv4_pkt
6. Pkt: το πακέτο

```
def _packet_in_handler(self, ev):
    if ev.msg.msg_len < ev.msg.total_len:
        self.logger.debug("packet truncated: only %s of %s bytes",
                          ev.msg.msg_len, ev.msg.total_len)
    msg = ev.msg
    datapath = msg.datapath
    ofproto = datapath.ofproto
    parser = datapath.ofproto_parser
    in_port = msg.match['in_port']

    pkt = packet.Packet(msg.data)
    eth = pkt.get_protocols(ethernet.ethernet)[0]
    eth_header = pkt.get_protocol(ethernet.ethernet)

    if eth.ethertype == ether_types.ETH_TYPE_LLDP:
        return
    dst = eth.dst
    src = eth.src

    dpid = format(datapath.id, "d").zfill(16)
    self.mac_to_port.setdefault(dpid, {})

    self.mac_to_port[dpid][src] = in_port

    if dst in self.mac_to_port[dpid]:
        out_port = self.mac_to_port[dpid][dst]
    else:
        out_port = ofproto.OFPP_FLOOD

    actions = [parser.OFPActionOutput(out_port)]

    data = None
    if msg.buffer_id == ofproto.OFP_NO_BUFFER:
        data = msg.data

    out = parser.OFPPacketOut(datapath=datapath, buffer_id=msg.buffer_id,
                              in_port=in_port, actions=actions, data=data)
    datapath.send_msg(out)

    tcp_pkt = pkt.get_protocol(tcp.tcp)
    ipv4_pkt = pkt.get_protocol(ipv4.ipv4)

    if tcp_pkt:
        self.session_check(datapath, eth_header, in_port, tcp_pkt, ipv4_pkt)
```

Εικόνα 6.51: Μέθοδος packet_in_handler.

Αφού κληθεί η μέθοδος session_check δημιουργούνται μεταβλητές που θα χρησιμοποιηθούν στη συνέχεια. Οι μεταβλητές αυτές είναι:

1. Max_count: μέγιστος αριθμός συνδέσεων ανά host.
2. Src_ip: διεύθυνση δικτύου του αποστολέα.
3. Dst_ip: διεύθυνση δικτύου του παραλήπτη.
4. Open_con: περιέχει την διεύθυνση δικτύου το αποστολέα και του παραλήπτη.
5. Server_ip: διεύθυνση δικτύου του διακομιστή (μπορεί να περιέχει περισσότερους από ένα διακομιστές).

```
def session_check(self, datapath, eth_header, in_port, tcp_pkt, ipv4_pkt):

    max_count = 3
    src_ip = ipv4_pkt.src
    dst_ip = ipv4_pkt.dst
    open_con = (src_ip, dst_ip)
    server_ip = "10.0.0.14"
```

Εικόνα 6.52: Μεταβλητές για την διενέργεια του ελέγχου εντός της μεθόδου session_check.

Αφού δημιουργηθούν οι μεταβλητές γίνεται έλεγχος για την προέλευση και τον τύπο του πακέτου. Αν το πακέτο δεν προέρχεται από τον διακομιστή και είναι συγχρονισμού τότε πραγματοποιείται έλεγχος για το κατά πόσο υπάρχει ο συνδυασμός διεύθυνσης δικτύου αποστολέα και παραλήπτη στον πίνακα connection_counter.

Στην περίπτωση που υπάρχει καταχώρηση στον πίνακα αυξάνεται ο μετρητής κατά μια μονάδα, διαφορετικά δημιουργείται νέα καταχώρηση με πλήθος 1. Ακολούθως εκτελείται έλεγχος για το πλήθος των ανοιχτών συνδέσεων με το όριο που δηλώθηκε στην μεταβλητή max_count που για τους σκοπούς της προσομοίωσης έχει τιμή 3. Αν το πλήθος των ανοιχτών συνδέσεων είναι μεγαλύτερος του ορίου καλείται η μέθοδος connection_block με ορίσματα τα στοιχεία του μεταγωγέα και της πόρτας του, την κεφαλίδα του πακέτου και τις διευθύνσεις αποστολέα και παραλήπτη.

Εναλλακτικά στην περίπτωση που το πακέτο δεν ανήκει στην διεύθυνση του διακομιστή και είναι τύπου Acknowledge γίνεται μείωση του μετρητή στον πίνακα session_count για τη διεύθυνση IP του αποστολέα με προϋπόθεση να υπάρχει καταχώρηση στον πίνακα. Η διαδικασία πρόσθεσης και αφαίρεσης του πλήθους των ανοιχτών συνδέσεων πραγματοποιείται κάθε φορά που εκπέμπονται πακέτα συγχρονισμού από τους χρήστες.

```
if src_ip != server_ip and tcp_pkt.has_flags(tcp.TCP_SYN):
    if open_con in self.connection_counter:
        self.connection_counter[open_con] += 1
        if self.connection_counter[open_con] > 3:
            self.logger.info("%s OPENS MORE THAN 3 CONNECTIONS WITH %s : BLOCKING", src_ip, dst_ip,)
            self.connection_block(datapath, eth_header, in_port, src_ip, dst_ip)
    else:
        self.connection_counter[open_con] = 1
if src_ip != server_ip and tcp_pkt.has_flags(tcp.TCP_ACK):
    if self.connection_counter[open_con] >= 1:
        self.connection_counter[open_con] -=1
```

Εικόνα 6.53: Έλεγχοι που πραγματοποιούνται μέσα στη μέθοδο session_check.

Το τρίτο σημείο αφορά την αντιμετώπιση μιας επιβεβαιωμένης επίθεσης. Κατά την εκτέλεση της μεθόδου connection_block δημιουργούνται οι απαραίτητές μεταβλητές, ofproto και parser, για την εισαγωγή κανόνα στον πίνακα ροών. Ο κανόνας διακοπής σύνδεσης αποθηκεύεται στη μεταβλητή match και διαθέτει την διεύθυνση δικτύου το αποστολέα.

Για την προσθήκη του κανόνα καλείται η μέθοδος add_flow και είναι απαραίτητο να διαθέτει τα ακόλουθα ορίσματα:

1. Datapath: δηλώνει τον μεταγωγέα openflow.
2. 2: ο αριθμός προτεραιότητας του κανόνα ροής.
3. Match: προέρχεται από τη μεταβλητή που περιέχει την φυσική διεύθυνση πηγής (αποστολέα).
4. []: αντιπροσωπεύει την μεταβλητή action. Είναι κενό ώστε τα εισερχόμενα πακέτα να απορρίπτονται (να μην εκτελεστεί κάποια ενέργεια από τον μεταγωγέα openflow, απόρριψη πακέτου).

```
def connection_block(self, datapath, eth_header, in_port, src_ip, dst_ip):

    ofproto = datapath.ofproto
    parser = datapath.ofproto_parser

    match = parser.OFPMatch(eth_src=eth_header.src)
    self.add_flow(datapath, 2, match, [])
```

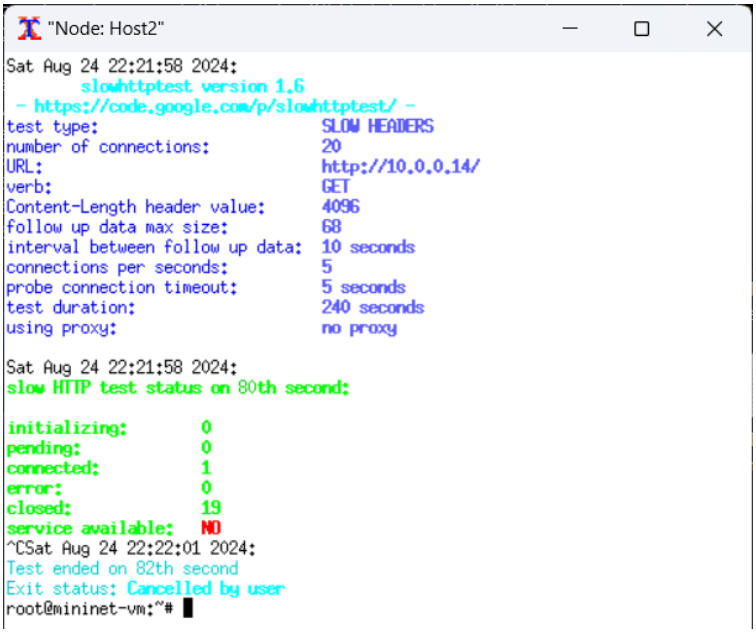
Εικόνα 6.54: Κώδικας αντιμετώπισης επιβεβαιωμένης επίθεσης.

```
def add_flow(self, datapath, priority, match, actions, buffer_id=None):
    ofproto = datapath.ofproto
    parser = datapath.ofproto_parser

    inst = [parser.OFPInstructionActions(ofproto.OFPIT_APPLY_ACTIONS,
                                         actions)]
    if buffer_id:
        mod = parser.OFPFlowMod(datapath=datapath, buffer_id=buffer_id,
                                priority=priority, match=match,
                                instructions=inst)
    else:
        mod = parser.OFPFlowMod(datapath=datapath, priority=priority,
                                match=match, instructions=inst)
    datapath.send_msg(mod)
```

Εικόνα 6.55: Μέθοδος add_flow που βρίσκεται στην εφαρμογή simple_switch_l3.

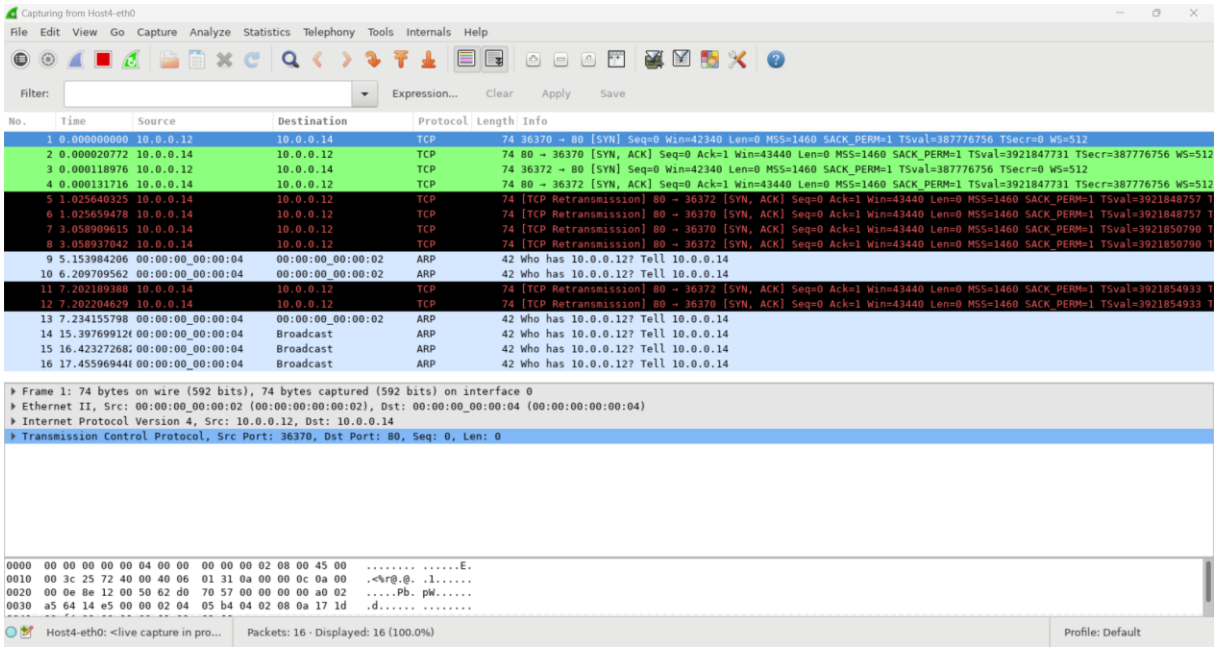
Κατά την εκτέλεση της επίθεσης με προστασία η διαδικασία είναι όμοια με αυτή χωρίς τη λήψη κάποιου μέτρου προστασίας. Το αποτέλεσμα στο τερματικό του κακόβουλου host φαίνεται να είναι όμοιο με αυτό κατά την διαδικασία επίθεσης χωρίς προστασία.



Εικόνα 6.56: Τερματικό Host 2 κατά την διακοπή της επίθεσης.

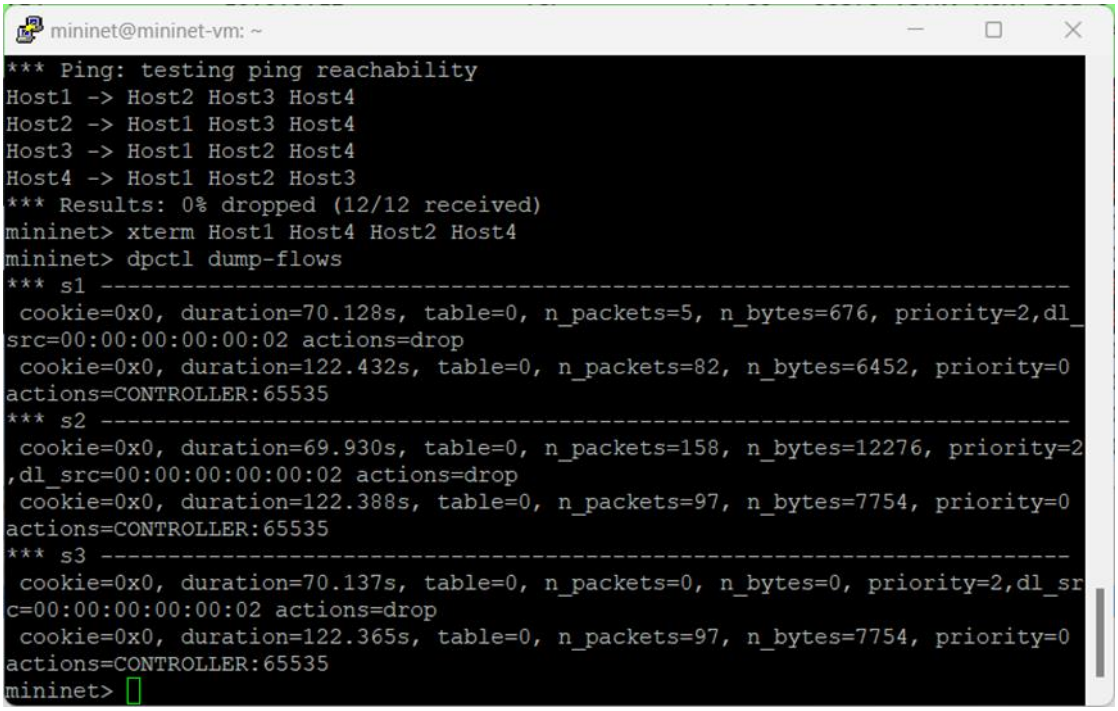
Μέσα από το περιβάλλον του Wireshark στον Host 4 φαίνεται ότι έλαβε κάποια από τα πακέτα προτού εντοπιστεί η επίθεση με ένα από αυτά να είχε ολοκληρωμένη επικοινωνία. Αυτό συμβαίνει καθώς το

εργαλείο ανά κάποια χρονικά διαστήματα ολοκληρώνει κάποιες από τις ανοιχτές συνδέσεις του με το θύμα.



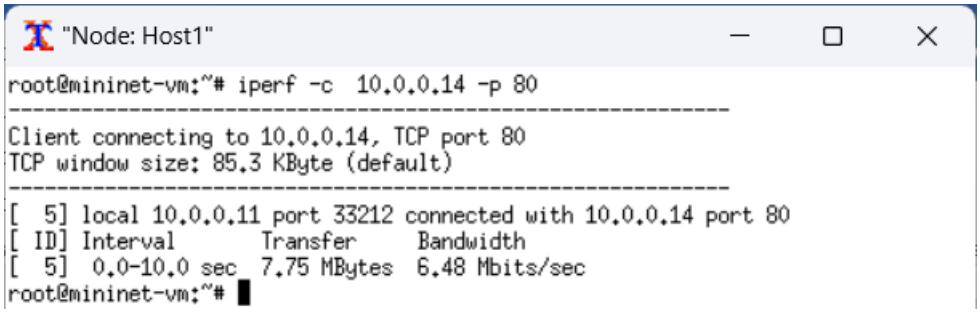
Εικόνα 6.57: Περιβάλλον Wireshark του θύματος κατά την επίθεση.

Η απόρριψη των κακόβουλων συνδέσεων όπως περιγράφηκε πιο πάνω γίνεται με την προσθήκη κανόνα στον πίνακα ροών του μεταγωγέα. Με την εκτέλεση της εντολής dpctl dump-flows στο τερματικό του Mininet εμφανίζονται όλοι οι κανόνες ανά μεταγωγέα όπως φαίνεται το ακόλουθο στιγμιότυπο.



Εικόνα 6.58: Πίνακες ροής μεταγωγέων μετά τον εντοπισμό και την αντιμετώπιση της επίθεσης.

Από πλευράς κάποιου άλλου host που επιχειρεί να επικοινωνήσει με το θύμα το αποτέλεσμα είναι διαφορετικό αφού η επικοινωνία ολοκληρώνεται όπως προβλέπεται από την εκτέλεση του εργαλείου iperf και ολοκληρώνεται με επιτυχία.



Εικόνα 6.59: Εκτέλεση εργαλείου iperf κατά την επίθεση από το Host 1.

Κεφάλαιο 7ο: Συμπεράσματα και μελλοντικές επεκτάσεις

7.1 Συμπεράσματα

Τα δίκτυα καθοριζόμενα από λογισμικό έχουν να προσφέρουν πολλά στον κλάδο της δικτύωσης και με την κατάλληλη ανάπτυξη τους μπορούν να αλλάξουν, προς το καλύτερο, τον τρόπο που γίνεται η ανταλλαγή πληροφοριών μεταξύ δύο απομακρυσμένων σημείων και να αντικαταστήσουν σε μεγάλο βαθμό τα παραδοσιακής αρχιτεκτονικής δίκτυα, παρέχοντας στους χρήστες τους καλύτερη εμπειρία χρήσης. Η χρήση ανοικτού τύπου λογισμικού δίνει πολλές δυνατότητες στους διαχειριστές των δικτύων καθώς τους δίνεται η ευκαιρία να αναπτύσσουν νέες μεθόδους δρομολόγησης και διαχείρισης δεδομένων με βάση τη χρήση του κάθε δικτύου. Ωστόσο για την καλύτερη αξιοποίηση τους είναι απαραίτητη η υλοποίηση νέων πρωτοκόλλων και η βελτιστοποίηση των διαθέσιμων που να ανταποκρίνονται στις σύγχρονες απαιτήσεις.

Η γρήγορη εξάπλωση του διαδικτύου τις τελευταίες δεκαετίες δημιούργησέ προβλήματα στην υλοποίηση και τη διαχείριση των δικτύων. Τα προβλήματα αυτά εμφανίστηκαν λόγο της αρχιτεκτονικής και πολυπλοκότητας ως προς την παραμετροποίηση και τη διαχείριση των δικτύων. Πέρα από τα προβλήματα που προκύπτουν από την αρχιτεκτονική τους δημιουργούνται προβλήματα που αφορούν την ασφάλεια της υποδομής και των χρηστών. Με την ανάπτυξη των δικτύων SDN υλοποιήθηκαν νέου είδους ευπάθειες που απειλούν τα επίπεδα της αρχιτεκτονικής τους. Παράλληλα η αρχιτεκτονική των δικτύων SDN βοήθησε στην ανάπτυξη νέων μεθόδων προστασίας, όπως η χρήση εφαρμογών και μεθόδων όπως η τεχνητή νοημοσύνη που προσφέρουν άμεση και πιο αποδοτική αντιμετώπιση.

Η μέθοδος προστασίας από τις προσομοιώσεις που πραγματοποιήθηκαν είναι πιθανό να μην είναι αποτελεσματικές για κάθε περίπτωση. Αυτό εξαρτάται από τα δεδομένα που αφορούν την κάθε επίθεση ξεχωριστά, όπως η διαδικασία εκτέλεσης και οι παράμετροι της. Πέρα από τις παραμέτρους της κάθε επίθεσης ρόλο παίζει και η δυνατότητα του κάθε host να τις διαχειριστεί. Για τον εντοπισμό της επίθεσης ARP Spoofing η μέθοδος που παραχωρούνται οι διευθύνσεις IP μπορεί να επηρεάσει τη διαδικασία. Κατά την εκτέλεση της επίθεσης DDoS, ανάλογα με την επεξεργαστική ισχύ του, ο host μπορεί να διαχειρίζεται διαφορετικό όγκο δεδομένων, όσο περισσότερους πόρους διαθέτει, τόσο περισσότερα πακέτα μπορεί να διαχειριστεί. Από την άλλη κατά την εκτέλεση της επίθεσης Slowloris οι ταυτόχρονες συνδέσεις δεν επηρεάζονται άμεσα από τους πόρους του host αλλά από της δυνατότητες του πρωτοκόλλου που χρησιμοποιείται.

Η εκτέλεση προσομοιώσεων δείχνει ότι η εκτέλεση επιθέσεων μπορεί να γίνει ευκολά και αποτελεσματικά με τη χρήση των εργαλείων που είναι διαθέσιμα. Ακόμα οι προσομοιώσεις είναι ένα χρήσιμο εργαλείο για την αξιολόγηση επιθέσεων επιτρέποντας την δοκιμή και ανάπτυξη νέων μεθόδων. Η ανάπτυξη της κάθε μεθόδου προστασίας απαιτεί διαφορετικές παραμέτρους ανάλογα με τα χαρακτηριστικά και το μέγεθος του δικτύου. Η ανάπτυξη μεθόδων προστασίας με τη χρήση του controller μπορεί να γίνει αποτελεσματικά χάρη στη χρήση πρωτοκόλλων όπως το OpenFlow που παρέχει πληροφορίες όπως ο τύπος των πακέτων και οι διευθύνσεις αποστολέα και παραλήπτη.

7.2 Μελλοντικές επεκτάσεις

Οι μέθοδοι προστασίας που υλοποιήθηκαν μπορούν να επεκταθούν ώστε να καλύπτουν περισσότερες περιπτώσεις επιθέσεων παρόμοιου ή διαφορετικού τύπου εντοπίζοντας επιθέσεις με διαφορετικές συνθήκες και χαρακτηριστικά. Ακόμα οι υλοποιήσεις αυτές μπορούν να ενσωματωθούν σε μια ενιαία

εφαρμογή και να συνδυαστούν με τεχνητή νοημοσύνη και μοντέλα μηχανικής μάθησης ενώ σε συνδυασμό με τη χρήση βάσης δεδομένων να ενημερώνονται με νέες πιθανές απειλές.

Βιβλιογραφία

[1] L. Roberts, «The ARPANET and Computer Networks,» pp. 141-171.

[2] C. A. Sunshine, «A briff history of Computer Networking,» σε *Computer Network architectures and protocols Second Edition*, New York, Plenum Press, 1989, p. 3.

[3] A. Fogel, S. Fung, L. Pedrosa, M. W. Sullivan, R. Govindan, R. Mahajan και T. Millstein, «AGeneral Approach to Network Configuration Analysis,» USENIX Association, Oakland, 2015.

[4] E. R. Jimson, K. Nisar και M. H. A. Hijazi, «The State of the Art of Software Defined Networking (SDN) Issues in Current Network Architecture and a Solution for Network Management Using the SDN,» *International Journal of Technology Diffusion,* τόμ. 10, pp. 33-48, 2019.

[5] I. H. E. A. K. A. C. Ola Salman, «SDN Controllers: A Comparative Study,» 2016.

[6] A. Lara, A. Kolasani και B. Ramamurhy, «Network Innovation using OpenFlow: A Survey,» 2013.

[7] P. Goranson, C. Black και T. Culiver, «The OPENFLOW Specification,» σε *SOFTWARE DEFINED NETWORKS (SECOND EDITION)*, 2017, pp. 89-136.

[8] A. Mukherjee, S. Dutta, R. A. Saeed και M. K. Naskar, «Fault tracking framework to SOFTWARE-DEFINED NETWORKING (SDN),» σε *Resource Allocation in Next-Generation Broadband Wireless Access Networks*, 2017, pp. 247-272.

[9] S. Scott-Hayward, G. O'Callaghan και S. Sezer, «SDN Security: A Survey,» Belfast, 2013.

[10] «Software-Defined Networking: The New Norm for Networks,» ONF WHITE PAPER , 2012.

[11] D. Serpanos και T. Wolf, Architecture of Network Systems, 2011.

[12] A. B. P. L. I. B. L. L. J. G. V. Angel Leonardo Valdivieso Caraguay, «SDN: Evolution and Opportunities in the Development IoT Applications,» International Journal of Distributed Sensor Networks, 2014.

[13] L. L. Petwrson και B. S. Davie, Computer Networks: A System Approach.

[14] J. M. Kizza, Guide to computer Network Security, 2020.

[15] S. Convery, Network Security Architectures, Indianapolis: Cisco Press, 2005.

[16] N. Feamster, J. Rexford και E. Zegura, «The road to SDN: An intellectual history of programmable networks,» 2014.

[17] M. Jarschel, T. Zinner, T. Hobfeld, P. Tran-Gia και W. Kellerer, «Interfaces, Attributes, and Use Cases: A Compass for SDN,» IEEE Communication Magazine, 2014.

[18] S. Timovic και I. Rdusinovic, «Towards a scalable, robust and QoS-aware vistual-link provisioning in SDN-based ISP networks,» *Transactions on Network and Service Management,* 2019`.

[19] A. G. Rahim Masoudi, «Software Defined Networks: A survey,» *Journal of Network and Computer Applications,* pp. 22-25, 2016.

[20] V. Thirupathi, C. Sandeep, S. N. Kumar και P. P. Kumar, «A Comprehensive Review on SDN Architecture Applications ans Major Benefits of SDN,» *International Journal of Advanced Science and Technology,* τόμ. 28, αρ. 20, pp. 607-614, 2019.

[21] G. LIANG και W. LI, «A novel industrial control architecture based on Software-Defined Network,» 2018.

[22] G. Wright, «Definition OPENFLOW».

[23] D. Pitt, «Key benefits of OEPNFLOW-based SDN,» 2012. [Ηλεκτρονικό]. Available: https://opennetworking.org/news-and-events/blog/key-benefits-of-openflow-based-sdn/.

[24] «OPEN NETWORKING FOUNDATION,» 2015. [Ηλεκτρονικό]. Available: https://opennetworking.org/wp-content/uploads/2014/10/openflow-switch-v1.5.1.pdf.

[25] D. Varnum, «Overlaid,» 2017. [Ηλεκτρονικό]. Available: https://overlaid.net/2017/02/15/openflow-basic-concepts-and-theory/.

[26] «RYU,» 2011-2014. [Ηλεκτρονικό]. Available: https://ryu.readthedocs.io/en/stable/ofproto_v1_2_ref.html.

[27] O. Salman, I. H. Elhajj, A. Kayssi και A. Chehub, «SDN Controllers: A Comparative Study,» Beirut, 2020.

[28] S. H. Haji, S. R. M. Zeebaree, R. H. Saeed, S. Y. Ameen, H. M. Shukur, N. Omar, M. A. M. Sadeeq, Z. S. Ageed, I. M. Ibrahim και H. M. Yasin, «Comparison fo Software Defined Networking with traditional networking,» *Asian Journal of Research in Computer Science,* 2021.

[29] «POX,» [Ηλεκτρονικό]. Available: https://github.com/noxrepo/pox.

[30] V. P. Sarvade και S. Kullkarni, «Modified POX Controller for Enhancing Quality of Experience of Multimedia Streaming in a Realistic Software Defined Vehicular Ad Hoc Networks,» 6th International Conference on Computation System and Information Technology for Sustainable Solutions (CSITSS), 2022.

[31] S. A. Al-Haswawi και M. K. Ibrahem, «Emulation of the POX Controller as a Load Balancer,» Iraqi Journal of Information & Communications Technology, 2021.

[32] S. Asdallahi, B. Goswami και M. Sammer, «RYU Controllers scalability experement on Software Defined Networks,» 2018.

[33] «RYU,» [Ηλεκτρονικό]. Available: https://ryu-sdn.org/.

[34] H. Akeay και D. Yiltas-Kaplan, «Web-Based User Interface for the Floodlight SDN Controller,» *Int. J. Advanced Networking and Applications,* τόμ. 8, αρ. 05, pp. 3175-3180, 2017.

[35] «Floodlight SDN OpenFlow Controller,» [Ηλεκτρονικό]. Available: https://github.com/floodlight/floodlight?tab=readme-ov-file#Documentation-and-Support.

[36] «OPEN DAYLIGHT,» [Ηλεκτρονικό]. Available: https://www.opendaylight.org/.

[37] M. Sisov, «Building a Software-Defined Networking System with OpenDaylight Controller,» Helsinki, 2016.

[38] S. Badotra και J. Singh, «Open Dylight as Controller for Software Defined Networking,» *International Jurnal of Advanced Research in Computer Science,* τόμ. 8, αρ. 5, pp. 1105-1111, 2017.

[39] A. Alhaj και N. Dutta, «Analysis of Security Attacks in SDN Network: A Comprehensive Survey,» σε *Contemporary Issues in Communication, Cloud and Big Data Analytics*, 2022, pp. 27-37.

[40] J. T. Y. X. H. Y. a. Y. Z. Gang Xiong, «A Survey of Network Attacks Based on Protocol Vulnerabilities,» Bejing, China, 2014.

[41] R. Kloti, V. Kotronis και P. Smith, «OpenFlow: A Security Analysis,» 2013.

[42] A. A. L. W. &. T. M. Ghorbani, «Network Attacks,» 2009.

[43] P. K. a. J. S. Najeem, «A review of security, threats and mitigation approaches for SDN architecture».

[44] M. B. Jimenez, D. Fernandez, J. E. Rivaneira, L. Bellido και A. Crdenas, « A Survey of the Main Security Issues and Solutions for the SDN Architecture,» 2021.

[45] W. L. ,. M. T. Ali A. Ghorbani, «Network Attacks,» σε *Network Intrusion Detection and Prevention*, 2009, p. 11.

[46] T. Lukaseder, L. Maile, B. Erb και F. Kargl, «SDN-Assisted Network-Based Mitigation of Slow DDoS Attacks,» 2018.

[47] M. Yue, H. Wang, L. Liu και Z. Wu, «Detecting DoS Attacks Based on Multy-Features in SDN,» 2020.

[48] J.-M. Baek, B.-S. Seo, K.-M. Ko και W.-B. Lee, «Comparative Study of AI-Enabled DDoS Detection Technologies in SDN,» 2023.

[49] R. Kandoi και M. Antikainen, «Denial-of-Service Attacks in OpenFlow SDN,» IFIP/IEEE International Symposium on Integrated Network Management, 2015.

[50] T. Mahjabin, Y. Xiao, G. Sun και W. Jiang, «A survey of distribited denial-of-service attack, prevention, and mitigation techniques,» International Journal of Distributed, 2017.

[51] D. Hercog, «ARP Protocol,» σε *Communication Protocols,* Springer, pp. 321-322.

[52] S. Hajazi και M. S. Obaidat, «Address resolution protocol spoofing attacks and security approaches: A survey,» 2018.

[53] R. J. Clark, H. L. Owen και J. H. Cox Jr,, «Leveraging SDN for ARP Security,» 2016.

[54] T. Alharbi, D. Durando, F. Pakzad και M. Portmann, «Securing ARP in Software Defined Networks,» 2016.

[55] Y. J. a. I. J. Mohammad Z. Masoud, «On Preventing ARP poisoning attack utilizing Software Defined Network (SDN) paradigm,» 2015.

[56] S. Hijazi και M. S. Obaidat, «Address resolution protocol spoofing attacks and security approaches,» 2018.

[57] H. Y. Ibrahim, P. M. Ismael, A. A. Albabawat και A. B. Al-Khalil, «A secure mechanism to prevent ARP Spoofing and ARP broadcasting in SDN,» 2020.

[58] S. Kumar, «Impact of Distributed Denial of Service (DDoS) Attack Due to ARP Storm,» 2005.

[59] C. L. Abad και R. I. Bonilla, «An Analysis on the Schimes for Detecting and Preventing ARP Cache Poisoning Attack,» 2007.

[60] N. Dayal, P. Maity, S. Srivastava και R. Khondoker, «Research Trends in Security and DDoS in SDN,» *Security and communication Networks,* pp. 6386-6411, 2017.

[61] S. Vidya και R. Bhaskaran, «ARP Storm Detection and Prevention Measures,» *IJCSI International Journal of Computer Scienc,* τόμ. 8, αρ. 2, pp. 456-460, 2011.

[62] M. Aldoud, D. Al-Abri, A. A. Maashri και F. Kausar, «DHCP attacking tools: an analysis,» *Computer Virology and Hacking Techniques,* 2021.

[63] S. Akashi και Y. Tong, «Classification of DHCP Spoofing and Effectiveness of DHCP Snooping,» Science and Technology Publications, 2019.

[64] C. Toprak, C. Turker και A. T. Erman, «Detection of DHCP Starvation Attacks in Software Defined Networks: A Case Study,» 3rd International Conference on Computer Science and Engineering, 2018.

[65] T. Bui, M. Antikainen και T. Aura, «Analysis of Topology Poisonign Attacks in Software-Defined Networking,» σε *Secure IT Systems*, 24th Nordic Conference, 2019, pp. 87-102.

[66] N. Kaur, A. K. Singh, N. Kumar και S. Srivastava, «Performance Impact of Topology Poisoning Attack in SDN and its Countermeasure,» 2017.

[67] R. Kloti, V. Kotronis και P. Smith, «OpenFlow: A Security Analysis,» 21st IEEE International Conference on Network Protocols (ICNP), 2013.

[68] M. Brandt, R. Khondoker, R. Marx και K. Bayarou, «Security Analysis of Software Defined Networking Protocols - OpenFlow, OF-Config and OVSDB».

[69] F. I. Khan, A. Sohail, M. S. Saleem, H. B. Javaid, M. F. Timuri και S. Hameed, «Securing Software Defined Network against rogue Controllers,» 2017.

[70] H. Abie, «An Overview of Firewall Technologies,» 2000.

[71] M. Roesch, «Snort,» [Ηλεκτρονικό]. Available: https://www.snort.org/.

[72] S. S. Hayward και G. O'Callaghan, «SDN Security: A Survey,» Queen’s University Belfast.

[73] P.-J. Chen και Y.-W. Chen, «Implementation od SDN Based Network Intrusion Detection and Prevention System,» The 49th Annual IEEE International Conference on Security Technology.

[74] C. Dobre και D. Comaneci, «Securing Networks using SDN and Machine Learnign,» IEEE International Conference on Computational Science and Engineering, 2018.

[75] A. H. Abdi, L. Audah, A. Salah, M. A. Alhartomi, H. Rasheed, S. Ahmed και A. Tahir, «Security Control and Data Planes of SDN: A Comprehensive Review of Traditional, AI, and MTD Approaches to Security Solutions,» 2024.

[76] M. Portnoy, Virtualization Essentials, 2012.

[77] «Virtual Box,» Oracle, [Ηλεκτρονικό]. Available: https://www.virtualbox.org/.

[78] B. Lantz και B. Heller, «Mininet,» Mininet Project Contributors, [Ηλεκτρονικό]. Available: https://mininet.org/.

[79] J. Dugan, S. Elliot, B. A. Mah, J. Poskanzer και K. Prabhu, «Iperf,» [Ηλεκτρονικό]. Available: https://iperf.fr/.

[80] T. Samstag, «Dsniff,» [Ηλεκτρονικό]. Available: https://github.com/tecknicaltom/dsniff.

[81] «Wireshark,» 1998. [Ηλεκτρονικό]. Available: https://www.wireshark.org/.