ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ, ΗΛΕΚΤΡΟΝΙΚΗΣ ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΑΝΤΙΜΕΤΩΠΙΣΗ ΔΙΚΤΥΑΚΩΝ ΕΠΙΘΕΣΕΩΝ ΣΕ ΕΥΦΥΗ ΠΡΟΓΡΑΜΜΑΤΙΖΟΜΕΝΑ ΔΙΚΤΥΑ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Δημήτριος Α. Μάρκου

Επιβλέπων: Βασίλειος Μάγκλαρης

Καθηγητής Ε.Μ.Π.

Αθήνα, Μάιος 2016

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ, ΗΛΕΚΤΡΟΝΙΚΗΣ ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΑΝΤΙΜΕΤΩΠΙΣΗ ΔΙΚΤΥΑΚΩΝ ΕΠΙΘΕΣΕΩΝ ΣΕ ΕΥΦΥΗ ΠΡΟΓΡΑΜΜΑΤΙΖΟΜΕΝΑ ΔΙΚΤΥΑ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Δημήτριος Α. Μάρκου

Επιβλέπων: Βασίλειος Μάγκλαρης

Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 27 η Ιουνίου 2016.

.....	.....	.....
Βασίλειος Μάγκλαρης	Ευστάθειος Συκάς	Μιχαήλ Θεολόγου
Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π	Καθηγητής Ε.Μ.Π

Αθήνα, Μάιος 2016

.....

Μάρκου Δημήτριος

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Ευχαριστίες

Η παρούσα διπλωματική εργασία αποτελεί το τελευταίο στάδιο των προπτυχιακών σπουδών μου στο Εθνικό Μετσόβιο Πολυτεχνείο. Αρχικά θα ήθελα να ευχαριστήσω τον καθηγητή μου κ. Βασίλειο Μάγκλαρη για την εμπιστοσύνη που μου έδειξε, την ευκαιρία που μου έδωσε καθώς και για τις πολύτιμες συμβουλές του. Ακόμη, ευχαριστώ όλα τα μέλη του εργαστηρίου NETMODE για την βοήθεια τους, και ιδιαιτέρως τον υπεύθυνο της διπλωματικής μου εργασίας Κωνσταντίνο Γιώτη, για την άριστη συνεργασία μας καθώς και για την πολύτιμη καθοδήγηση του σε όλα τα στάδια εκπόνησης της. Τέλος θα ήθελα να ευχαριστήσω την οικογένεια και τους φίλους μου για την ανεκτίμητη υποστήριξη και συμπαράσταση τους, στη διάρκεια των σπουδών μου.

Περίληψη

Σε μια εποχή που χαρακτηρίζεται από την διακίνηση μεγάλου όγκου πληροφορίας η ανάλυση αυτής και η σωστή κατηγοριοποίησή της αποτελεί πρωταρχικό παράγοντα και βασικό κριτήριο στην ανάπτυξη νέων τεχνολογιών. Παρά ταύτα το σωστό φιλτράρισμα και ο διαχωρισμός της πληροφορίας σε καλόβουλη και κακόβουλη δεν γίνεται τόσο αποτελεσματικά από τα ήδη υπάρχοντα δίκτυα. Άρα η ανάπτυξη δικτύων τα οποία θα παρουσιάζουν μεγαλύτερο βαθμό ευελιξίας και προσαρμοστικότητας είναι απαραίτητη ώστε να υπάρχει σωστή ανταπόκριση στις επιταγές της νέας πραγματικότητας. Η ανάγκη αυτή οδήγησε στην γέννηση των Ευφυών δικτύων(Software Defined Networks aka SDN).Τα ευφυή δίκτυα είναι μια νέα πιο ευέλικτη κατηγορία δικτύων τα οποία ξεπερνούν τα προσχώματα που έθεταν τα μη προγραμματιζόμενα και στατικά υπάρχοντα δίκτυα. Συγκεκριμένα τα ευφυή δίκτυα προάγουν τον διαχωρισμό της διαχείρισης της πληροφορίας σε δύο επίπεδα. Το πρώτο επίπεδο που ονομάζεται πλατφόρμα δεδομένων έχει ως μοναδικό χαρακτηριστικό την διακίνηση και μόνο της πληροφορίας. Το δεύτερο επίπεδο που ονομάζεται πλατφόρμα ελέγχου είναι υπεύθυνο για το έλεγχο της πληροφορίας, των συσκευών και του δικτύου γενικότερα. Σκοπός αυτής της διπλωματικής είναι η εκμετάλλευση αυτής της καινούργιας αρχιτεκτονικής ώστε να αναλύσουμε την διακινούμενη πληροφορία, να την φιλτράρουμε, να την διαχωρίσουμε σε καλόβουλη και κακόβουλη και εν κατακλείδι να αποκόψουμε την μεταφορά της επιβλαβούς για το σύστημα πληροφορίας. Αυτό θα το πετύχουμε μέσω του προγραμματισμού της πλατφόρμας ελέγχου ώστε να μπορούμε ουσιαστικά διαμέσου ενός προγράμματος (Snort) να φιλτράρουμε όλη την διακινουμένη πληροφορία, να διαχωρίζουμε την επιβλαβή πληροφορία και τελικά να την αποκόπτουμε διατηρώντας το σύστημα μας ασφαλές.

Λέξεις κλειδιά: Ευφυή Δίκτυα, πλατφόρμα ελέγχου ευφυών δικτύων Opendaylight,πρόγραμμα ανίχνευσης και καταστολής κακόβουλης πληροφορίας, Ασφάλεια δικτύων

Abstract

In an era characterized by large amounts of data the analysis and the proper classification of those data is a primary factor and basic criterion in developing new technologies. However the proper filtration and separation of the information in good-natured and malicious not done so effectively by existing networks. So the development of networks which will have greater flexibility and adaptability is essential because must be a correct response to the requirements of the new reality. This need led to the birth of Intelligent Networks (Software Defined Networks aka SDN) .The intelligent networks is a new, more flexible class networks which exceed obstacles which raised due to non-programmable and static existing networks. Specifically software defined networks promote the separation for the management of information at two levels. The first level is called data platform which has one and only feature the transportation of the information. The second level is called control platform which is responsible for the management of information, devices and the network in general. The aim of this project is to seize the advantages of this new architecture and try to analyze the transported information, to filter it, to separate the good-natured information from the malicious information and in conclusion to cut off the transfer of harmful for the system information. We will accomplish all the aforementioned with the programming of the control platform which with the help of a specific programm (Snort) will manage to filter all the transported information , to separate the harmful one from the good one , to cut of the harmful information and to keep our system intact eventually.

Key-words: network security,Intrusion Detection System, Intrusion Prevention System,Software Defined Networks,Opendaylight controller

Περιεχόμενα

Κατάλογος Εικόνων

1.Εισαγωγή

Καθημερινά γινόμαστε μάρτυρες διακίνησης τεράστιου όγκου πληροφορίας μέσω των Δικτύων Ηλεκτρονικών Υπολογιστών. Αναμφίβολά η έλευση του Διαδικτύου συνέβαλλε καθοριστικά στην μεταφορά και μεταγωγή όλων αυτών των πληροφοριών. Οι άνθρωποι χρησιμοποιούν όλο και πιο συχνά το Διαδίκτυο για να διεκπεραιώσουν ακόμα και τις πιο απλές εργασίες τους καθώς έχουν πρόσβαση σε τεράστιους όγκους πληροφοριών και υπηρεσιών. Παρόλα αυτά κατά καιρούς έχουμε παρατηρήσει ότι η χρήση του διαδικτύου δεν είναι πάντα αθώα πολλοί κακόβουλοι χρήστες χρησιμοποιούν το Διαδίκτυο για να βλάψουν μεγάλο όγκο χρηστών αλλά και να προσβάλλουν και να υποκλέψουν διαβαθμισμένες πληροφορίες στις οποίες σε διαφορετική περίπτωση δεν θα είχαν πρόσβαση. Άρα αυτομάτως εγείρονται θέματα ασφάλειας όσων αφορά την χρήση του διαδικτύου.

2. Ευφυή Δίκτυα

Τα ευφυή δίκτυα είναι ένα νέο είδος δικτύων τα οποία εισάγουν ένα καινούργιο κεφάλαιο στον τρόπο με τον οποίο σχεδιάζαμε και λειτουργούσαμε τα δίκτυα μέχρι σήμερα. Συγκεκριμένα τα ευφυή δίκτυα προβάλλουν μια νέα αρχιτεκτονική δικτύων η

οποία έχει ως κύριο χαρακτηριστικό της των διαχωρισμό του επιπέδου ελέγχου από το επίπεδο δεδομένων. Η καινοτόμα αυτή αρχιτεκτονική μας επιτρέπει να έχουμε δίκτυα τα οποία είναι λειτουργικά, ευέλικτα, προσαρμοζόμενα σε ξαφνικές αλλαγές, εύκολα προγραμματιζόμενα και κυρίως ελεγχόμενα. Η ανάγκη για την ανάπτυξη τέτοιων δικτύων είναι εμφανής καθώς την συνεχιζόμενη ανάπτυξη των υπολογιστών και του διαδικτύου τα τελευταία τριάντα χρόνια δεν ακολούθησε η ανάπτυξη των δικτύων τα οποία έχουν παραμείνει πολύπλοκα, μη προσβάσιμα και δύσκολα παραμετροποιήσημα. Επίσης τα υπάρχοντα δίκτυα έχουν εξελιχθεί σε τροχοπέδη όσον αφορά την ανάπτυξη νέων καινοτόμων υπηρεσιών και της περαιτέρω επέκτασης του διαδικτύου.

Αναλυτικότερα τα κυριότερα προβλήματα των σημερινών δικτύων είναι:

- **Δυσκολία βελτιστοποίησης**

Οι διαχειριστές των δικτύων αντιμετωπίζουν δυσκολίες στην βελτιστοποίηση μεγάλων εγκαταστάσεων όπως μεγάλα κέντρα δεδομένων, ευρείας γεωγραφικής έκτασης δίκτυα και μεγάλα δίκτυα επιχειρήσεων.

- **Προβλήματα ασφάλειας**

Στα υπάρχοντα δίκτυα έχουν παρατηρηθεί κατά καιρούς προβλήματα ασφάλειας και δυσκολία αναγνώρισης και καταστολής κακόβουλης πληροφορίας που διακινείται διαμέσου αυτών των δικτύων. Με αποτέλεσμα η χρήση τους να καθίσταται επισφαλής.

- **Προβλήματα απόδοσης**

Η μετάδοση μεγάλου όγκου πληροφορίας στα υπάρχοντα δίκτυα αποδοτικά και αξιόπιστα είναι ένα πρόβλημα που έχει παρουσιασθεί κατά καιρούς και έχει ως αποτέλεσμα την μη επιτυχή επικοινωνία των χρηστών και την απώλεια σημαντικών πληροφοριών που πολλές φορές είναι δύσκολα ανακτήσιμες.

- **Προβλήματα ελέγχου και διαχείρισης**

Η διαχείριση και ο έλεγχος των υπαρχόντων δικτύων δεν είναι καινούργιο πρόβλημα. Πολλές φορές έχει παρατηρηθεί η δυσκολία που βιώνουν οι διαχειριστές αυτών των δικτύων στην αντιμετώπιση κάποιων σφαλμάτων που μπορούν να προκύψουν και στην δυσκολία να ελέγξουν τις ροές των πληροφοριών που μεταφέρονται μέσω αυτών των δικτύων.

- **Δυσκολία παραμετροποίησης**

Ακόμα και μεγάλοι πάροχοι δικτυακών υπηρεσιών έχουν εκφράσει την δυσκολία που αντιμετωπίζουν στην παραμετροποίηση των υπαρχόντων δικτύων και στην προσφορά υπηρεσιών ποιότητας στους πελάτες τους. Επίσης αναφέρονται ιδιαίτερα στην δυσκολία παροχής εξατομικευμένων υπηρεσιών μέσω των υπαρχόντων δικτύων ώστε να εξυπηρετήσουν τις ανάγκες των πελατών τους.

Μελετώντας λοιπόν τα προαναφερθέντα προβλήματα των υπαρχόντων δικτύων καταλήγουμε στο συμπέρασμα πως μια ποιοτική και ουσιαστική αναβάθμιση είναι επιτακτική για την καλύτερη λειτουργία των δικτύων και την καλύτερη παροχή υπηρεσιών προς τους τελικούς χρήστες. Μια τέτοια αναβάθμιση προσφέρουν τα ευφυή δίκτυα καθώς όπως αναφέραμε προηγουμένως έχουν αλλάξει τελείως τον τρόπο με τον οποίο διαχειριζόμαστε και ελέγχουμε τα δίκτυα και τις ροές πληροφοριών που διακινούνται με αποτέλεσμα να επιτυγχάνεται καλύτερη ποιότητα υπηρεσιών και πιο ασφαλή επικοινωνία μεταξύ των μελών ενός δικτύου.

Αναλυτικότερα τα πλεονεκτήματα που προκύπτουν από την χρήση των ευφυών δικτύων είναι:

- **Κεντρικοποιημένη δικτυακή επίβλεψη**

Τα ευφυή δίκτυα παρέχουν μια κεντρικοποιημένη επίβλεψη του δικτύου η οποία καθιστά εφικτό τον έλεγχο και την διαχείριση των λειτουργιών που λαμβάνουν χώρα στο

δίκτυο. Με την αποδέσμευση της πλατφόρμας ελέγχου από αυτή των δεδομένων τα ευφυή δίκτυα μπορούν να επιταχύνουν την αποστολή των διαφόρων υπηρεσιών και να επιτύχουν μια πιο αποδοτική επίβλεψη των εικονικών και πραγματικών δικτυακών συσκευών από μια κεντρική τοποθεσία.

- **Ευελιξία**

Πολλές φορές οι επιχειρήσεις έρχονται αντιμέτωπες με αιτήματα πελατών τους που τις υποχρεώνουν να εγκαθιδρύσουν νέες υπηρεσίες στο σύστημά τους για την διεκπεραίωση αυτών των αιτημάτων. Έτσι με την βοήθεια των ευφυών δικτύων οι διαχειριστές των δικτύων της εκάστοτε επιχείρησης μπορούν να αναπτύξουν, να παραμετροποιήσουν και να πειραματιστούν με νέες εφαρμογές και υπηρεσίες χωρίς να επηρεάσουν την λειτουργία του ήδη υπάρχοντος δικτύου της επιχείρησης. Επίσης τα ευφυή δίκτυα παρέχουν ένα σύνολο προγραμματιζόμενων διεπαφών (APIs) οι οποίες βοηθούν στον έλεγχο των πραγματικών αλλά και των εικονικών συσκευών του δικτύου.

- **Αποδοτικότερη ασφάλεια δικτύων**

Ένα από τα μεγάλα πλεονεκτήματα των ευφυών δικτύων το οποίο εκτιμούν ιδιαίτερα οι διαχειριστές τους είναι η κεντρικοποιημένη ασφάλεια. Η τεχνολογία της εικονοποίησης έχει μετατρέψει τον έλεγχο των δικτύων σε μια απαιτητική διαδικασία καθώς εικονικές μηχανές ενεργοποιούνται και απενεργοποιούνται συνεχώς στα πλαίσια μιας πραγματικής υποδομής δικτύου. Έτσι οι διαχειριστές πρέπει συνεχώς να παραμετροποιούν τα τείχη προστασίας και να εφαρμόζουν διαφορετικές πολιτικές φιλτραρίσματος ώστε να καταστήσουν ασφαλές το δίκτυο. Με την βοήθεια των ευφυών δικτύων οι διαχειριστές μπορούν να παρέχουν μια κεντρικοποιημένη και πιο αποδοτική ασφάλεια στο δίκτυο διανέμοντας εύκολα πολίτικες ασφαλείας σε όλο το δίκτυο. Επίσης μπορούν εύκολα να προσαρμοστούν σε κακόβουλη χρήση του δικτύου και να δράσουν αμέσως λόγο της ευκολίας που τους παρέχει ο κεντρικός έλεγχος του δικτύου. Έτσι ο έλεγχος του δικτύου εναποτίθεται σε μια και μόνο κεντρική πλατφόρμα.

- **Μείωση λειτουργικών εξόδων**

Τα ευφυή δίκτυα παρουσιάζουν διαχειριστική αποδοτικότητα, βελτίωση στην λειτουργία πολλών δικτυακών συσκευών και καλύτερο έλεγχο στις εικονοποιημένες

υπηρεσίες που παρέχονται. Έτσι οδηγούν σε μείωση των λειτουργικών εξόδων μια εταιρίας και βοηθούν στην αυτόματη και κεντρικοποιημένη επίλυση προβλημάτων των υπάρχοντών δικτύων που επιβάρυναν οικονομικά την εκάστοτε επιχείρηση.

- **Αποδοτική χρησιμοποίηση υπάρχοντος υλικού(Hardware) και αποφυγή επιπρόσθετων εξόδων**

Η υιοθέτηση των ευφυών δικτύων δίνει νέα πνοή στην ήδη υπάρχουσα υποδομή και στις ήδη υπάρχουσες συσκευές δικτύου. Οι ήδη υπάρχουσες συσκευές δικτύου μπορούν να επαναχρησιμοποιηθούν και να παραμετροποιηθούν ώστε να εκτελούν εντολές τις οποίες λαμβάνουν από την κεντρική μονάδα ελέγχου. Έτσι αφαιρώντας ευφυΐα από τις συσκευές αυτές και αντιμετωπίζοντας αυτές σαν μαύρα κουτιά που θα λαμβάνουν εντολές από την κεντρική πλατφόρμα οδηγούμαστε σε φθηνότερες λύσεις δικτυακού υλικού καθώς έχουμε αφαιρέσει το κόστος δημιουργίας έξυπνων δικτυακών συσκευών.

- **Έλεγχος πόρων συνεφιακών υπηρεσιών(Cloud computing)**

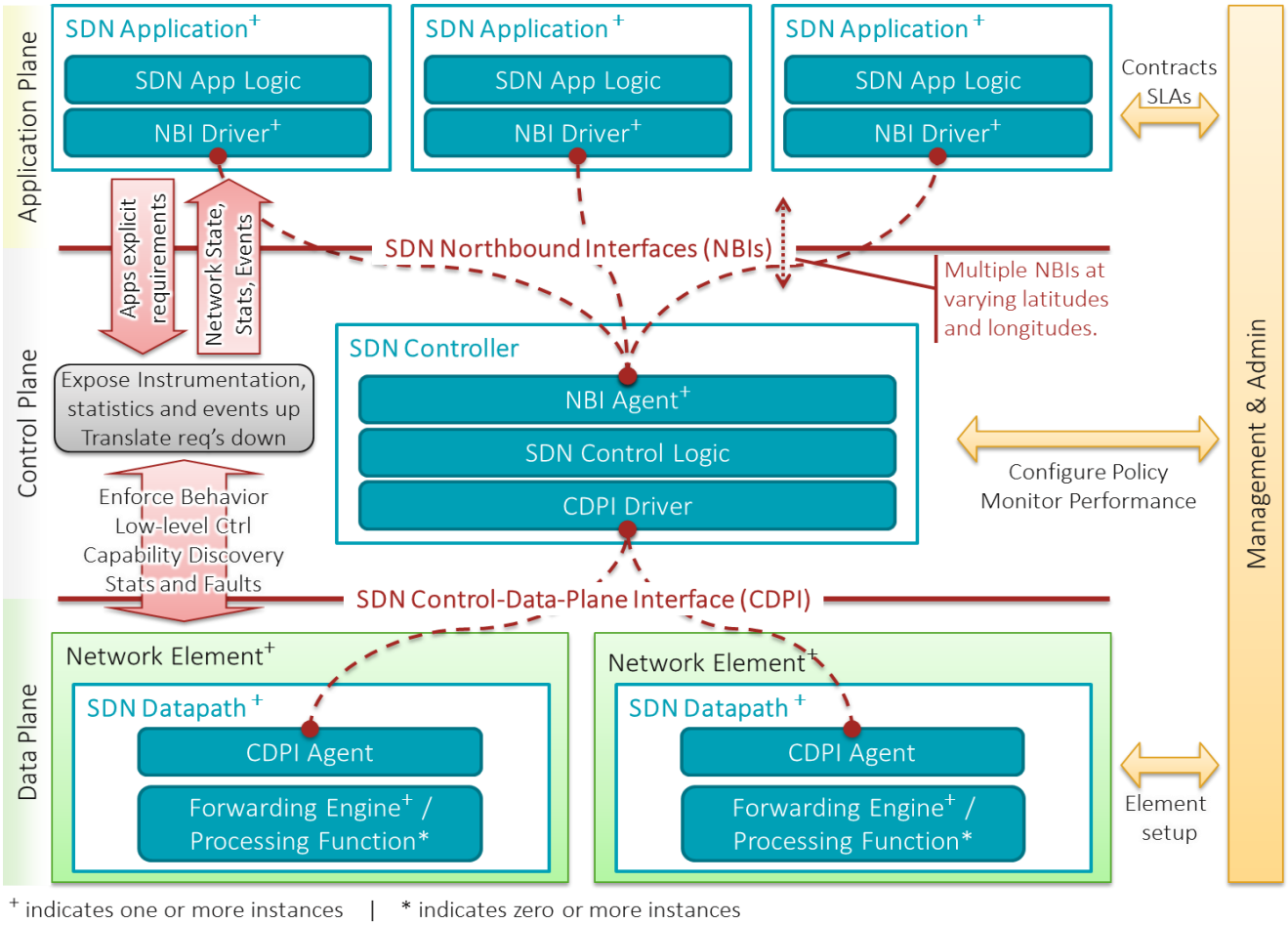
Λόγω της συνεχομένης ανάπτυξης των συνεφιακών υπηρεσιών είναι απαραίτητο να αποσπάσουμε τον έλεγχο των πόρων αυτών των υπηρεσιών και να το αποδώσουμε στη πλατφόρμα ελέγχου των ευφυών δικτύων για να έχουμε μια αποδοτική κατανομή των πόρων που καταναλώνουν αυτές οι υπηρεσίες. Έτσι είναι δυνατή η ενοποίηση των πόρων των υπηρεσιών cloud. Παραδείγματος χάριν μπορούμε να εναποθέσουμε στην κεντρική μονάδα ελέγχου των ευφυών δικτυών τον έλεγχο των δικτυακών οντοτήτων ενός μεγάλου κέντρου δεδομένων.

- **Ποιοτική παροχή υπηρεσιών**

Η ικανότητα της παραμετροποίησης και του ελέγχου ροών δεδομένων μέσω των ευφυών δικτύων είναι ένα από τα μεγάλα πλεονεκτήματα. Τα χαρακτηριστικά των ευφυών δικτύων επιτρέπουν την αποδοτική μεταφορά μεγάλου όγκου δεδομένων με αποτέλεσμα να μπορούν να παρέχουν ποιοτικές υπηρεσίες (QoS)φωνής μέσω του διαύλου του IP πρωτοκόλλου. Επίσης βελτιώνεται η παροχή υπηρεσιών Streaming καθώς τα ευφυή δίκτυα εγγυόνται την αδιάκοπη μεταφορά πληροφορίας και την αποφυγή προβλημάτων που μπορούν να προκύψουν.

2.1 Αρχιτεκτονική Ευφυών Δικτύων

Στην συνέχεια θα αναφερθούμε στην ιδιαίτερη και καινοτόμα αρχιτεκτονική των ευφυών δικτυών και θα αναλύσουμε κάθε ένα ξεχωριστά τα σύνθετα στοιχεία που την απαρτίζουν καθώς και την λειτουργία που το καθένα επιτελεί. Τα στοιχεία και η γενικότερη αρχιτεκτονική των ευφυών δικτύων απεικονίζονται στην Εικόνα 1.



Αρχιτεκτονική SDN

Εικόνα 1

Στην Εικόνα 1. παρατηρούμε την αρχιτεκτονική και τα ξεχωριστά στοιχεία που συνθέτουν την λειτουργία των ευφυών δικτύων. Θα προσπαθήσουμε να αναλύσουμε όλα τα στοιχεία αλλά και τις λειτουργίες τους ακολουθώντας μια από 'πάνω προς τα κάτω ' προσέγγιση η οποία θα μας βοηθήσει να κατανοήσουμε την ιδιαίτερη λειτουργία των ευφυών δικτύων αλλά και πως διαχωρίζονται και επικοινωνούν τα διάφορα επίπεδα μεταξύ τους.

2.1.1 Στοιχεία αρχιτεκτονικής Ευφυών Δικτύων

- **Εφαρμογές SDN**

Στο πάνω μέρος της Εικόνας 1 παρατηρούμε το επίπεδο των εφαρμογών SDN. Οι εφαρμογές SDN είναι προγράμματα που άμεσα και προγραμματιστικά επικοινωνούν με την πλατφόρμα ελέγχου SDN (SDN controller) μέσω των βόρειων διεπαφών (Nourthbound Interfaces aka NBIs) για να ικανοποιήσουν τις όποιες δικτυακές ανάγκες έχουν. Επιπροσθέτως οι εφαρμογές αυτές καταναλώνουν πολλές φορές πόρους του υπάρχοντος δικτύου για να ικανοποιήσουν εσωτερικές λειτουργίες στην κατεύθυνση της λήψης αποφάσεων. Οι εφαρμογές SDN αποτελούνται από δυο μέρη το πρώτο είναι η λογική την οποία εξυπηρετούν και είναι προγραμματισμένες να εκτελούν στο πλαίσιο ενός SDN δικτύου και το δεύτερο μέρος είναι οι οδηγοί των βόρειων διεπαφών που βοηθούν στην αποτελεσματική επικοινωνία με την πλατφόρμα ελέγχου [1].

- **Πλατφόρμα ελέγχου (SDN Control Plane aka Controller)**

Η πλατφόρμα ελέγχου SDN είναι μια λογική κεντροποιημένη πλατφόρμα η οποία είναι υπεύθυνη για την εξυπηρέτηση των απαιτήσεων του επιπέδου εφαρμογής SDN και επιπλέων είναι επιφορτισμένη με την παροχή πληροφοριών προς το επίπεδο εφαρμογών

SDN σχετικά με την κατάσταση του δικτύου. Τέτοιες πληροφορίες είναι παραδείγματος χάριν στατιστικά στοιχεία και γεγονότα που συμβαίνουν.

Μία πλατφόρμα ελέγχου αποτελείται από τα εξής τρία μέρη. Το πρώτο μέρος είναι ένας Northbound Agent οποίος βοηθάει στην σωστή επικοινωνία με το επίπεδο εφαρμογών SDN. Το δεύτερο μέρος είναι η λογική η οποία είναι προγραμματισμένη στην πλατφόρμα ελέγχου και σύμφωνα με αυτήν την λογική η πλατφόρμα εκτελεί τις διάφορες διεργασίες. Τέλος το τρίτο μέρος περιλαμβάνει τους οδηγούς(Drivers) που βοηθούν στην σύνδεση της πλατφόρμας ελέγχου SDN με την πλατφόρμα δεδομένων μέσω διεπαφών.

- **Πλατφόρμα Δεδομένων (SDN Data Plane)**

Η πλατφόρμα ελέγχου είναι μια λογική δικτυακή οντότητα η οποία επιτρέπει παρακολούθηση και έλεγχο σε όλες τις λειτουργίες που αφορούν την μεταφορά και την επεξεργασία δεδομένων που λαμβάνουν χώρα στο εσωτερικό της .

Η πλατφόρμα δεδομένων αποτελείται από τα εξής δύο μέρη. Το πρώτο μέρος είναι ένα πράκτορας (CDPI Agent) που βοηθάει στην σύνδεση και στην επικοινωνία της πλατφόρμας ελέγχου με την πλατφόρμα δεδομένων μέσω διεπαφών. Το δεύτερο μέρος αποτελείται από μια ή περισσότερες μηχανές προώθησης δεδομένων και ίσως όχι πάντα από ρουτίνες οι οποίες εκτελούν κάποιες συγκεκριμένες λειτουργίες όπως επεξεργασία διαφόρων δεδομένων.

Στην συνέχεια θα εμβαθύνουμε την ανάλυση μας όσων αφορά την αρχιτεκτονική των ευφυών δικτύων και θα επεξηγήσουμε περαιτέρω τις λειτουργίες που επιτελούν οι διάφορες διεπαφές που απεικονίζονται στην Εικονα 1 και επιπλέων θα αναφερθούμε στον σημαντικό ρόλο που διαδραματίζουν στην σύνδεση των διαφόρων επιπέδων των ευφυών δικτύων.

- **Διεπαφή σύνδεσης πλατφόρμας ελέγχου και δεδομένων (CDPI)**

Η SDN CDPI διεπαφή είναι μια καθορισμένη διεπαφή η οποία έγκειται ανάμεσα στην πλατφόρμα ελέγχου και στην πλατφόρμα δεδομένων και η οποία παρέχει προγραμματικό έλεγχο σε όλες τις λειτουργίες μεταφοράς που λαμβάνουν χώρα

(forwarding procedures) και επίσης είναι επιφορτισμένη με την μεταφορά στατιστικων δεδομένων και ειδοποιήσεων που αφορούν γεγονότα που συμβαίνουν.

- **Βόρεια Διεπαφή (NorthBound Interfaces aka NBIs)**

Οι SDN NBIs Διεπαφές είναι διεπαφές οι οποίες βρίσκονται ανάμεσα στο επίπεδο των SDN εφαρμογών και στην πλατφόρμα ελέγχου και τυπικα παρέχουν μια γενική και αφαιρετική εικόνα του υπαρχοντος δικτύου όπως επίσης παρέχουν πληροφόρηση σχετικά με τις απιτήσεις και την συμπεριφορά του δικτύου αυτού.

- **Διεπαφές Οδηγών και Πρακτόρων (Drivers and Agents Interfaces)**

Κάθε διεπαφή υλοποιείται από ένα ζευγάρι Οδηγού-Πράκτορα ο πράκτορας αντιπροσωπεύει το Νότιο κομμάτι ή το κομμάτι που συνδέεται με την υποδομή και ο οδηγός αντιπροσωπεύει το Βόρειο κομμάτι ή το κομμάτι που συνδέεται με την εφαρμογή.

3. Openflow και Ευφυή Δίκτυα

Με την λέξη Openflow αναφερόμαστε στο βασικό πρωτόκολλο επικοινωνίας το οποίο λαμβάνει χώρα και αφορά τα ευφυή δίκτυα. Το openflow ουσιαστικά επιτρέπει στην πλατφόρμα ελέγχου ενός ευφυούς δικτύου να επικοινωνήσει και να παραμετροποιήσει την πλατφόρμα δεδομένων και συγκεκριμένα να ελέγξει το κομμάτι που γίνεται η μεταφορά των δεδομένων σε εικονικές ή πραγματικές δικτυακές συσκευές (switches,routers etc.). Σε γενικές γραμμές και απλουστευμένα μπορούμε να πούμε ότι το πρωτόκολλο αυτό επιτρέπει στην πλατφόρμα ελέγχου να παραμετροποιήσει τις συσκευές δικτύου και να δώσει εντολές για το που θα σταλούν οι ροές δεδομένων που καταφτάνουν σ 'αυτές τις συσκευές.

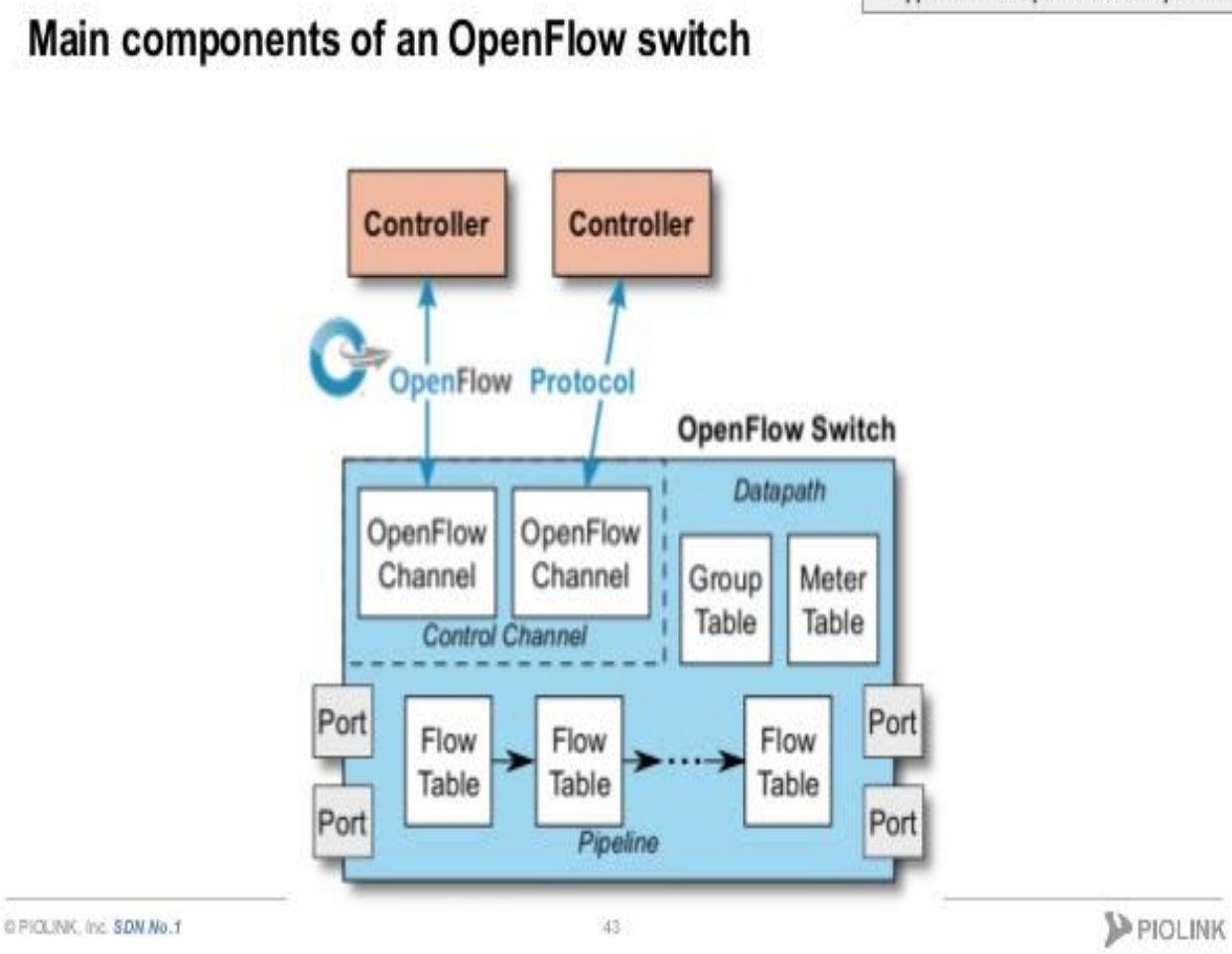
Σε μία συνηθισμένη δικτυακή υποδομή (non-SDN,non-openflow) οι δικτυακές συσκευές που την απαρτίζουν δεν είναι επιφορτισμένες μόνο με την μεταφορά των δεδομένων αλλά είναι προικισμένες με λογισμικό το όποιο βοηθάει τις συσκευές αυτές να ελέγξουν και να επεξεργαστούν τα δεδομένα αυτά. Απ' την άλλη μεριά μια δικτυακή συσκευή η οποία συνεργάζεται με το πρωτόκολλο openflow διαχωρίζει το επίπεδο της μεταφοράς των δεδομένων από το επίπεδο ελέγχου αυτών και έτσι η δικτυακή συσκευή (switch) είναι μόνο επιφορτισμένη με την μεταφορά των δεδομένων και αναθέτει σε μία πλατφόρμα ελέγχου να ελέγξει τα δεδομένα αυτά και να πάρει αποφάσεις επί αυτών των δεδομένων μια τέτοια πλατφόρμα είναι ένας controller.

Άρα η επικοινωνία μεταξύ των δικτυακών συσκευών και της πλατφόρμας ελέγχου γίνεται μέσω του openflow πρωτοκόλλου το οποίο επιτρέπει την πιο αποδοτική χρήση των πόρων του δικτύου και το καλύτερο έλεγχο αυτού πράγμα το οποίο δεν ήταν τόσο εύκολο στα συνηθισμένα δίκτυα [2].

3.1 Ανάλυση Συσκευής Δικτύου Openflow

Σε αυτήν την παράγραφο θα περιγράψουμε τον τρόπο λειτουργίας και τις απαιτήσεις που προκύπτουν σε ένα openflow switch δηλαδή σε μια συσκευή δικτύου η οποία συνεργάζεται με το πρωτόκολλο openflow [3].

Στην Εικόνα 2 βλέπουμε ένα openflow logical Switch. Στην συνέχεια θα περιγράψουμε τα επιμέρους κομμάτια που εμφανίζονται όπως επίσης και τις λειτουργίες που εξυπηρετουν.



Συσκευή δικτύου Openflow

Εικόνα 2

Μια συσκευή δικτύου openflow (openflow logical switch Εικόνα 2) αποτελείται από τα εξής στοιχεία:

- **Flow and Group Tables**

Ένα openflow switch αποτελείται από ένα ή περισσοτέρους flow tables και έναν Group Table οι οποίοι αναλύουν τα πακέτα (δεδομένα) που καταφτάνουν και στην συνέχεια τα προωθούν υπακούοντας σε μία ομάδα κανόνων που περιέχονται σαυτούς τους πίνακες και ονομάζονται flow entries. Όταν ένα πακέτο δεδομένων καταφτάνει στην συσκευή δικτύου τότε επεξεργάζεται και ταυτοποιείται στον πρώτο πίνακα από τους

κανόνες που περιέχονται σαυτόν και στην συνέχεια ίσως προχωρήσει σε κάποιον επόμενο πίνακα σύμφωνα με την σωληνοειδή (pipeline) μορφή που βλέπουμε στην Εικόνα 2 ή θα εκτελέσει κάποια άλλη λειτουργία.

- **Openflow Channels**

Τα Openflow κανάλια (Openflow channels) είναι ουσιαστικά η δίοδος επικοινωνίας της openflow πλατφόρμας ελέγχου (openflow controller) με την openflow συσκευή δικτύου (openflow switch). Μέσω αυτών των καναλιών η πλατφόρμα ελέγχου μπορεί να ελέγξει τα δεδομένα που εισέρχονται μέσα στην συσκευή δικτύου προσθέτοντας, αναβαθμίζοντας, διαγράφοντας και γενικά παραμετροποιόντας τους κανόνες (flow entries) που υπάρχουν στους πίνακες (flow tables) και μέσω των οποίων παίρνονται η αποφάσεις για την μεταφορά των δεδομένων.

- **Flow Entries (Κανόνες)**

Οι κανόνες είναι οντότητες που περιέχονται με την μορφή ομάδας στον εκάστοτε πίνακα και ουσιαστικά ταυτοποιούν ένα πακέτο όταν καταφτάνει και στην συνέχεια εφαρμόζουν ενέργειες η οποίες καθορίζουν την υπόλοιπη πορεία του πακέτου στην συσκευή δικτύου. Βέβαια όταν ένα πακέτο δεδομένων καταφτάνει στην συσκευή δικτύου δεν εκτελεί όλες τις ενέργειες που περιγράφονται στην ομάδα κανόνων του πρώτου πίνακα αλλά εφαρμόζει ενέργειες ενός και μόνο κανόνα που είναι επιφορτισμένος με την επεξεργασία και μεταφορά του συγκεκριμένου πακέτου δεδομένων και μόνο. Θα αναφερθούμε στον τρόπο που τα πακέτα δεδομένων ταυτοποιούνται και μεταφέρονται σε επόμενη ενότητα.

- **Θύρες Openflow (Openflow Ports)**

Όπως μπορούμε να παρατηρήσουμε στην Εικόνα 2 υπάρχουν θύρες οι οποίες είναι προσκολλημένες στην συσκευή δικτύου. Αυτές οι θύρες ονομάζονται openflow θύρες και είναι βασικό κομμάτι μια συσκευής δικτύου. Οι openflow θύρες είναι διεπαφες οι οποίες μεταφέρουν τα πακέτα των δεδομένων μεταξύ της openflow επεξεργασίας και του υπόλοιπου δικτύου. Οι συσκευές δικτύου openflow συνδέονται μεταξύ τους μέσω αυτών των θυρών και ένα πακέτο δεδομένων μπορεί να μεταφερθεί από την μία openflow συσκευή δικτύου (openflow switch) στην άλλη από την θύρα εξόδου της πρώτης και από την θύρα εισόδου της δεύτερης.

3.2 Ανάλυση τρόπου λειτουργίας Openflow

3.2.1 Δεσμευμένες Openflow Θύρες (Reserved Ports)

Οι δεσμευμένες openflow θύρες είναι οι θύρες αυτές οι οποίες περιγράφουν γενικές λειτουργίες (actions) που πρέπει να γίνουν όταν ένα πακέτο καταφθάνει στην δικτυακή συσκευή. Τέτοιες λειτουργίες είναι αποστολή ενός πακέτου στην μονάδα ελέγχου , εκτέλεση πλημμύρας , ομαλή λειτουργία της δικτυακής συσκευής όσον αφορά την μεταγωγή του πακέτου (normal non-openflow L2 Forwarding) και άλλες λειτουργίες τις οποίες θα εξηγήσουμε εκτενώς παρακάτω

- ALL: Με την συγκεκριμένη λειτουργία η openflow συσκευή δικτύου είναι υπεύθυνη να δημιουργήσει αντίγραφα του πακέτου δεδομένων που κατέφθασε και να τα στείλει σε όλες τις διαθέσιμες θύρες για έξοδο από την συσκευή δικτύου εκτός από την θύρα που εισήλθε το αρχικό πακέτο δεδομένων.

- CONTROLLER: Η λειτουργία αυτή δίνει εντολή να χρησιμοποιηθεί το openflow κανάλι που συνδέει την πλατφόρμα ελέγχου openflow με την συσκευή δικτύου. Αυτή η δεσμευμένη θύρα μπορεί να χρησιμοποιηθεί σαν θύρα εξόδου αλλά και σαν θύρα εισόδου. Όταν χρησιμοποιείται σαν θύρα εξόδου τότε το πακέτο δεδομένων μετατρέπεται σε ένα ειδικό πακέτο-μήνυμα (packet-in message) και στέλνετε στην πλατφόρμα ελέγχου για περαιτέρω επεξεργασία. Όταν χρησιμοποιείται σαν θύρα εισόδου τότε σηματοδοτεί ένα πακέτο που στάλθηκε στην δικτυακή συσκευή από την πλατφόρμα ελέγχου.

- TABLE: Αντιπροσωπεύει την αρχή της σωληνοειδούς διαδικασίας του openflow και ουσιαστικά επιβεβαιώνει την ταυτοποίηση του πακέτου στον πρώτο πίνακα κανόνων και την επεξεργασία του πακέτου από τους υπόλοιπους πίνακες σ αυτήν την σωληνοειδούς τύπου επεξεργασία.

- IN-PORT: Αντιπροσωπεύει την πόρτα από την οποία εισήλθε το πακέτο και χρησιμοποιείται ως θύρα εξόδου δηλαδή δίνει εντολή να εξαχθεί το πακέτο από την θύρα από την οποία εισήλθε αρχικά.

- ANY: Συγκεκριμένες τιμές χρησιμοποιούνται σε κάποια openflow αιτήματα όταν δεν έχει χρησιμοποιηθεί καμία συγκεκριμένη πόρτα (π.χ. port is wildcarded) .Κάποια openflow αιτήματα περιέχουν μια αναφορά σε μία συγκεκριμένη θύρα όπου το συγκεκριμένο αίτημα έχει μοναδική εφαρμογή. Χρησιμοποιώντας το ANY σαν αριθμό θύρας σ'αυτά τα συγκεκριμένα αιτήματα επιτρέπουμε σ'αυτά τα αιτήματα να εφαρμοστούν σε οποιεσδήποτε θύρες.

- LOCAL: Αναφέρεται στην τοπική δικτυακή στοίβα της συσκευής και μπορεί να χρησιμοποιηθεί σαν πόρτα εισόδου και σαν πόρτα εξόδου. Η local port μπορεί να χρησιμοποιηθεί για την ενεργοποιήσει απομακρυσμένων οντοτήτων οι οποίες μπορούν να επικοινωνήσουν με την openflow συσκευή και τις λειτουργίες που αυτή κατέχει διαμέσου του openflow δικτύου και όχι μέσω από ένα ξεχωριστό δίκτυο που θα είχε ελεγκτικά χαρακτηριστικά πάνω σε αυτήν την συσκευή δικτύου.

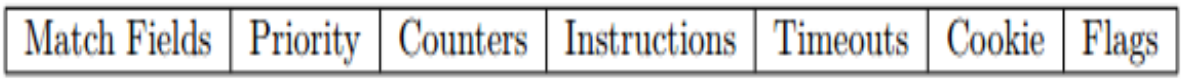
- NORMAL: Αντιπροσωπεύει την παραδοσιακή non-openflow διαδικασία/πόρτα που χρησιμοποιείται από μια non-openflow δικτυακή συσκευή για την μεταφορά και επεξεργασία ενός πακέτου δεδομένων (forwarding).Αυτή η θύρα που αντιπροσωπεύει αυτήν την λειτουργία μπορεί να χρησιμοποιηθεί μόνο σαν θύρα εξόδου και επίσης υπάρχει περίπτωση η λειτουργία αυτή να μην είναι διαθέσιμη σε όλες τις συσκευές δικτύου.

- FLOOD: Αυτή η λειτουργία/πόρτα αντιπροσωπεύει μια πλημμυρίδα πακέτων η οποία είναι παραδοσιακή λειτουργία non-openflow δικτυακών συσκευών. Συγκεκριμένα με αυτήν την λειτουργία το πακέτο θα σταλεί σε όλες τις θύρες εξόδου εκτός από την θύρα πού εισήλθε αρχικά.

  Θα ήθελα επιπλέων να διευκρινίσω ότι οι openflow-only δικτυακές συσκευές δεν υποστηρίζουν τις πόρτες NORMAL και FLOOD ενώ οι υβριδικές δικτυακές συσκευές τις υποστηρίζουν. Με τον όρο υβριδικές δικτυακές συσκευές εννοούμε αυτές τις συσκευές που υποστηρίζουν και το openflow πρωτόκολλο αλλα και την παραδοσιακή διαδικασία μεταγωγής πακέτων από ένα ethernet switch.

3.2.2 Πίνακες και Κανόνες Openflow (Openflow tables and flow entries)

Ένας openflow πίνακας αποτελείται από μία ομάδα από κανόνες (flow entries).Κάθε κανόνας αποτελείται από πεδία που καθορίζουν την ύπαρξη αλλά και την λειτουργία του. Ένα δείγμα ενός κανόνα παρουσιάζεται στην Εικόνα 3.



Openflow κανόνας

Εικόνα 3

Στην παραπάνω εικόνα βλέπουμε τα κύρια πεδία από τα οποία αποτελείται ένα κανόνας σε ένα openflow πίνακα. Τέτοιου τύπου κανόνες καθορίζουν τις λειτουργίες και τις μεταγενέστερες ενέργειες μιας openflow δικτυακής συσκευής κατά την εισαγωγή ενός πακέτου δεδομένων. Στην συνέχεια θα αναλύσουμε διεξοδικότερα τα πεδία αυτά.

Κάθε κανόνας ενός openflow πίνακα (flow table entry) περιέχει:

- Match fields: Τα πεδία ταυτοποίησης (Match fields) ταυτοποιούν τα πακέτα δεδομένων που καταφτάνουν στην δικτυακή συσκευή. Η ταυτοποίηση αυτή γίνεται με βάση την θύρα εισόδου που εισήλθε το πακέτο δεδομένων , τις επικεφαλίδες που φέρει το πακέτο και προαιρετικά με κάποια σωληνοειδούς τύπου πεδία που έχουν διευκρινιστεί από προηγούμενους πίνακες.

- Priority: Αυτό το πεδίο καθορίζει την προτεραιότητα που έχει ένας κανόνας σε σχέση με τους υπόλοιπους κανόνες μιας ομάδας όσον αφορά την χρησιμοποίηση αυτού για την ταυτοποίηση των πακέτων δεδομένων που εισέρχονται.

- Counters: Οι Μετρητές (Counters) είναι τα πεδία αυτά τα οποία ενημερώνονται κάθε φορά που ταυτοποιείται ένα πακέτο δεδομένων από τον συγκεκριμένο κανόνα. Επίσης μέσω των Μετρητών μπορούμε να γνωρίζουμε συνολικά πόσα πακέτα του δικτύου έχουν ταυτοποιηθεί από έναν κανόνα ενός πίνακα το οποίο μπορεί να θεωρηθεί και ένα στατιστικό στοιχείο.

- Instructions: Είναι τα πεδία αυτά τα οποία χρησιμοποιούνται για την μεταβολή των δράσεων που θα εφαρμοσθούν σε ένα πακέτο δεδομένων ή για την μεταβολή της σωληνοειδούς επεξεργασίας.

- Timeouts: Τα πεδία λήξεως χρόνου καθορίζουν την διάρκεια στην οποία είναι ενεργός ένας κανόνας και μπορεί να ταυτοποιεί πακέτα. Μετα το πέρας του χρονικού διαστήματος ο κανόνας αφαιρείται από τον openflow πίνακα.

- Cookies: Αδιαφανή δεδομένα που χρησιμοποιούνται κυρίως από την μονάδα ελέγχου. Μπορούν να χρησιμοποιηθούν από την μονάδα ελέγχου για να φιλτραριστούν κανόνες(flow entries) λόγω αιτήσεων για στατιστικά δεδομένα, για μετατροπή κανόνων και για διαγραφή κανόνων. Τα cookies δεν χρησιμοποιούνται στην επεξεργασία των εισερχόμενων πακέτων.

- Flags: Τα Flags αλλάζουν τον τρόπο με τον οποίο διαχειρίζονται οι κανόνες. Για παράδειγμα το Flag OFPFF_SEND_FLOW_REM ενεργοποιεί μηνύματα απομάκρυνσης κανόνα για έναν συγκεκριμένο κανόνα (flow entrie).

Κάθε κανόνας αναγνωρίζεται από τα πεδία ταυτοποίησης (match fields) και το πεδίο προτεραιότητας του κανόνα (priority). Επίσης το πεδίο του κανόνα που ονομάζεται Instructions περιέχει δράσεις (actions) που θα εφαρμοστούν σε ένα εισερχόμενο πακέτο.

3.2.3 Matching (Πεδίο ταυτοποίησης πακέτων)

Το matching είναι η διαδικασία ταυτοποίησης εισερχομένων δεδομένων(πακέτων).Καθώς καταφτάνει ένα πακέτο σε μια openflow συσκευή δικτύου τότε η συσκευή αυτή ξεκινάει να εξετάζει τους κανόνες (flow entries) που υπάρχουν στον πρώτο πίνακα κανόνων (flow table) και βασιζόμενη στην σωληνοειδή διαδικασία που ακολουθείτε μπορεί να εξετάσει κανόνες και σε επόμενους πίνακες.

Τα πεδία ταυτοποίησης που ανήκουν στο εισερχόμενο πακέτο αφαιρούνται από αυτό και χρησιμοποιούνται για την εξέταση των κανόνων στον πίνακα κανόνων. Τα πεδία αυτά εξαρτώνται από το τύπο του πακέτου και πολλές φορές περιέχουν διάφορες επικεφαλίδες όπως για παράδειγμα Ethernet ή IPv4 διευθύνσεις προέλευσης και προορισμού. Επιπροσθέτως ταυτοποίηση μπορεί να γίνει και με βάση την πόρτα που εισήλθε το πακέτο.

Ένα εισερχόμενο πακέτο ταιριάζει με έναν κανόνα σε έναν πίνακα κανόνων εάν τα πεδία ταυτοποίησης του πακέτου που χρησιμοποιήθηκαν για την ανάλυση των κανόνων σε έναν πίνακα κανόνων ταιριάζουν με τα πεδία ταυτοποίησης ενός από τους κανόνες στον πίνακα. Το πακέτο ταιριάζει μόνο με τον κανόνα που έχει τα ίδια πεδία ταυτοποίησης με αυτό και σε περίπτωση που δύο ή περισσότεροι κανόνες έχουν τα ίδια πεδία ταυτοποίησης με του πακέτου τότε επιλέγεται ο κανόνας με την μεγαλύτερη προτεραιότητα. Στην συνέχεια οι μετρητές που υπάρχουν στον συγκεκριμένο επιλεγμένο κανόνα ενημερώνονται και εφαρμόζονται τα instructions που περιλαμβάνονται στον κανόνα.

3.2.4 Table-Miss (Μη ταυτοποίηση πακέτων)

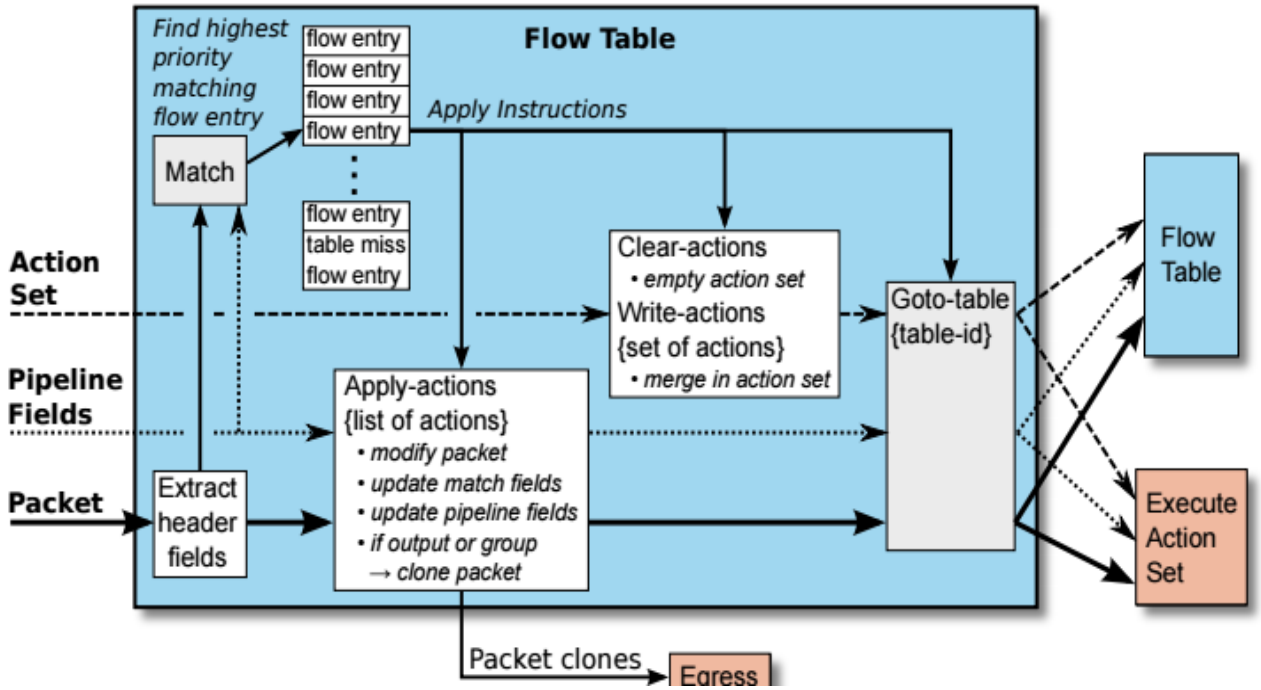
Κάθε πίνακας κανόνων (flow table) πρέπει να υποστηρίζει έναν table-miss κανόνα (flow entrie) σε περίπτωση που ένα εισερχόμενο πακέτο δεν αντιστοιχηθεί με κανέναν κανόνα του πίνακα κανόνων. Ο table miss κανόνας εξηγεί πως θα επεξεργαστούν τα μη αντιστοιχιζόμενα πακέτα για παράδειγμα μπορεί τα πακέτα αυτά να σταλούν στην μονάδα ελέγχου (controlelr) ,να απορριφθούν ή να σταλούν σε επόμενο πίνακα για περαιτερο επεξεργασία.

Ο table-miss κανόνας αναγνωρίζεται από τα πεδία ταυτοποίησης και προτεραιότητας. Συγκεκριμένα όλα τα πεδία ταυτοποίησης έχουν παραλειφθεί και η προτεραιότητα αυτού του κανόνα είναι η χαμηλότερη δυνατή δηλαδή μηδέν. Ο table-miss κανόνας πρέπει τουλάχιστον να υποστηρίζει την δράση αποστολής του μη αντιστοιχιζόμενου πακέτου στην

μονάδα ελέγχου μέσω της πόρτας με το αναγνωριστικό CONTROLLER ή να υποστηρίζει την δράση απόρριψης του πακέτου.

Ο table-miss κανόνας συμπεριφέρεται σε γενικές γραμμές όπως οποιοσδήποτε άλλος κανόνας. Δεν θεωρείται δεδομένη η ύπαρξή του σε ένα πίνακα κανόνων και η μονάδα ελέγχου είναι υπεύθυνη για την τοποθέτηση και την αφαίρεση αυτού του κανόνα. Επίσης ο κανόνας αυτός μπορεί να εκλείψει από τον πίνακα κανόνων καθώς μπορεί να περατωθεί η χρονική διάρκεια ύπαρξης του. Ο table-miss κανόνας επεξεργάζεται τα μη αντιστοιχιζόμενα εισερχόμενα πακέτα και εφαρμόζονται οι οδηγίες (instructions) που υπάρχουν στον συγκεκριμένο κανόνα. Εάν ο συγκεκριμένος κανόνας στείλει το πακέτο στην μονάδα ελέγχου τότε το packet-in μήνυμα θα αναγράφει ως αιτία αδυναμία ταυτοποίησης (table miss).Σε περίπτωση που ένας table-miss κανόνας δεν υπάρχει στον πίνακα τότε τα μη αντιστοιχιζόμενα πακέτα απορρίπτονται.

3.2.5 Instructions (Οδηγίες)



Σωληνοειδής διαδικασία Openflow

Εικόνα 4

Κάθε κανόνας σε έναν πίνακα κανόνων περιέχει ένα σετ από οδηγίες (instructions) οι οποίες εφαρμόζονται όταν ένα εισερχόμενο πακέτο αντιστοιχηθεί σε έναν συγκεκριμένο κανόνα. Αυτές οι οδηγίες έχουν ως αποτέλεσμα μετατροπές στο πακέτο , στο σετ των δράσεων (action set) που θα αναληφθούν και στην σωληνοειδή διαδικασία (δες Εικόνα 4).

Μία συσκευή δικτύου δεν είναι απαραίτητο να υποστηρίζει όλα τα είδη των οδηγιών αλλά πρέπει να υποστηρίζει τις οδηγίες που αναφέρονται ως "Required Instruction" δηλαδή απαραίτητες οδηγίες. Επίσης η μονάδα ελέγχου (controller) μπορεί να ρωτήσει την συσκευή δικτύου ποιες από τις "Optional Instruction" δηλαδή ποιες μη-απαραίτητες οδηγίες υποστηρίζει.

- *Optional Instruction:* **Apply-Actions *action(s):*** Εφαρμόζει συγκεκριμένες δράσεις χωρίς αλλαγές στο σετ των δράσεων (action set). Αυτή η οδηγία μπορεί να χρησιμοποιηθεί για να εκτελέσει μετατροπές στο πακέτο μεταξύ δύο πινάκων ή να εκτελέσει πολλές δράσεις του ίδιου τύπου. Οι δράσεις είναι καθορισμένες σαν λίστα δράσεων (list of actions).

- *Optional Instruction:* **Clear-Actions*:*** Εκκαθαρίζει όλες τις δράσεις από ένα σετ δράσεων.

- *Required Instruction*: **Write-Actions action(s):** Συνδέει ένα σετ δράσεων με το ήδη υπάρχων σετ δράσεων.

- *Optional Instruction*: **Stat-Trigger stat thresholds:** Δημιουργεί ένα γεγονός (event) στην μονάδα ελέγχου όταν κάποιο από τα στατιστικά ενός κανόνα ξεπεράσει ένα συγκεκριμένο ορισμένο κατώφλι τιμών.

- *Required Instruction:* **Goto-Table next-table-id:** Αναφέρει τον επόμενο πίνακα στην σωληνοειδή διαδικασία που ακολουθείται. Οι κανόνες στον τελευταιο πίνακα κανόνων αυτής της διαδρομής δεν πρέπει να περιλαμβάνουν την συγκεκριμένη οδηγία (instruction).

- *Optional Instruction:* **Write-Metadata metadata:** Γράφει τιμές μεταδεδομένων στο πεδίο των μεταδεδομένων (metadata field).

Το σετ των οδηγιών (instruction set) που περιλαμβάνεται σε έναν κανόνα αποτελείται το πολύ από μία οδηγία κάθε τύπου. Η συσκευή δικτύου πρέπει να απορρίπτει κανόνες που δεν εκτελούν τις οδηγίες που περιέχουν. Στην περίπτωση αυτή η συσκευή δικτύου επιστρέφει ένα μήνυμα λάθους (error message).

3.2.6 Actions (Δράσεις)

Μία συσκευή δικτύου δε είναι απαραίτητο να υποστηρίζει όλες τι δράσεις (actions) πρέπει όμως να υποστηρίζει τις λεγόμενες "απαραίτητες δράσεις" δηλαδή τις "Required Actions". Επίσης η μονάδα ελέγχου (Controller) μπορεί να "ρωτήσει" την συσκευή δικτύου ποιες από τις μη-απαραίτητες δράσεις υποστηρίζει δηλαδή ποιες από τις "Optional Actions". Άρα οι απαραίτητες δράσεις που έχουμε και στις οποίες θα αναφερθούμε εκτενέστερα είναι :

- *Required Action*: **Output port no:** Η Δράση αυτή προωθεί ένα εισερχόμενο πακέτο σε μια καθορισμένη openflow θύρα. Οι openflow συσκευες δικτύου πρέπει να υποστηρίζουν προωθήσεις σε φυσικές θύρες (physical ports), σε λογικές θύρες που είναι καθορισμένες από την συσκευή δικτύου (switch-defined logical ports), και σε κατειλημμένες θύρες (reserved ports) όπως η θύρα της μονάδας ελέγχου (CONTROLLER port).

- *Required Action*: **Group group id:** Επεξεργάζετε το εισερχόμενο πακέτο διαμέσου ενός συγκεκριμένου group από δράσεις.

- *Required Action*: **Drop:** Τα εισερχόμενα πακέτα που δεν τους έχει επιβληθεί συγκεκριμένη δράση ή κάποιο συγκεκριμένο group δράσεων απορρίπτονται. Αυτό το αποτέλεσμα μπορεί να προέλθει αν υπάρχει κάποιο άδειο σετ οδηγιών (Instructions) ή αν έχει εφαρμοστεί η Clear-Action οδηγία.

4 SDN Controller (Μονάδα ελέγχου ευφυών δικτύων)

Η μονάδα ελέγχου των ευφυών δικτύων (SDN controller) είναι ουσιαστικά το "μυαλό" που ελέγχει τις συσκευές σε μία αρχιτεκτονική ευφυών δικτύων. Είναι η πλατφόρμα αυτή που ελέγχει όλες τις στρατηγικές λειτουργίες στα ευφυή δίκτυα, πιο συγκεκριμένα διευθύνει και επιβλέπει όλες τις μεταφορές δεδομένων (via Southbound APIs) που γίνονται στις συσκευές δικτύου (switches) και φροντίζει για την εξυπηρέτηση των απαιτήσεων των εφαρμογών (via Northbound APIs) που βρίσκονται σε υψηλότερο επίπεδο.

Η μονάδα ελέγχου τυπικά αποτελείται από αφαιρούμενα στοιχεία(pluggable modules) τα οποία είναι επιφορτισμένα να εκτελούν διάφορες δικτυακές διεργασίες. Κάποιες από τις βασικές διεργασίες είναι για παράδειγμα η καταγραφή και αποθήκευση πληροφοριών για το ποιες δικτυακές συσκευές είναι συνδεμένες και ποιους σκοπούς εξυπηρετούν, η συλλογή στατιστικών δεδομένων κ.α.

Δύο από τα πιο γνωστά πρωτόκολλα που χρησιμοποιούν οι μονάδες ελέγχου ευφυών δικτύων (SDN controllers) για τις επικοινωνία με τις συσκευές δικτύου (switches) είναι τα Openflow και OVSDB. Βέβαια υπάρχουν και άλλα πρωτόκολλα τα οποία χρησιμοποιούνται για την προαναφερθείσα επικοινωνία και είναι εξίσου σημαντικά όπως τα YANG και NETCONF.

4.1 Ιστορικά στοιχεία

Μία συνεχιζόμενη μάχη εξελίσσεται μεταξύ πωλητών δικτυακού εξοπλισμού οι οποίοι θέλουν να αναπτύξουν τις δικές τους μονάδες ελέγχου (SDN Controllers) κάθε ένας ξεχωριστά για τον έλεγχο των δικών τους δικτυακών συσκευών και πιθανών δικτυακών συσκευών άλλων πωλητών.

Η πρώτη μονάδα ελέγχου ευφυών δικτύων (SDN controller) ήταν ο NOX οποίος αναπτύχθηκε από την εταιρεία Nicira Networks το 2008 και θεωρήθηκε η βάση για την

ανάπτυξη και άλλων μονάδων ελέγχου τα επόμενα χρόνια. Στην συνέχεια η Nicira συνεργάστηκε με την NTT και την Google και ανέπτυξαν την μονάδα ελέγχου ONIX οποία θεωρείται ως η βασική μονάδα ελέγχου για το WAN της Google.

Υπάρχει επίσης μια μεγάλη ποικιλία μονάδων ελέγχου οι οποίες είναι Open Source και αναπτυσσόμενές όπως οι POX , Beacon , Trema , Ryu, Floodlight,OpenContrail οι οποίες είναι αρκετά διαδεδομένες στον χώρο των μονάδων ελέγχου για ευφυή δίκτυα.

Μεταγενέστερα σημαντικοί πωλητές δικτυακού εξοπλισμού όπως η HP,Cisco,VMWare και Juniper ενεπλάκησαν στην αγορά των μονάδων ελέγχου για ευφυή δίκτυα και ανέπτυξαν δικές τους μονάδες ελέγχου οι οποίες αρχικά είχαν ως βάση την λογική της μονάδας ελέγχου Beacon όμως στην συνέχεια μετεπήδησαν στην λογική της μονάδας ελέγχου Opendaylight η οποία είναι και η μονάδα ελέγχου που χρησιμοποιήθηκε σε αυτήν την διπλωματική εργασία.

4.2 Είδη Μονάδων Ελέγχου Ευφυών Δικτύων (SDN Controllers)

Στις επόμενες γραμμές θα αναλύσουμε τις σημαντικότερες μονάδες ελέγχου για ευφυή δίκτυα που υπαρχουν:

- NOX: Η NOX είναι μια opensource πλατφόρμα ελέγχου για ευφυή δίκτυα στην οποία αναπτύσσονται εφαρμογές σε C++ γλώσσα παραγραμματισμού. Μία συγγενική πλατφόρμα ελέγχου είναι η POX η οποία εκτελεί ουσιαστικά τις ίδιες λειτουργίες με την NOX αλλά χρησιμοποιεί διαφορετική γλώσσα προγραμματισμού για την ανάπτυξη νέων εφαρμογών και συγκεκριμένα χρησιμοποιεί την Python. Κάποια από τα σημαντικότερα στοιχεία της πλατφόρμας ελέγχου NOX είναι ότι παρέχει ένα C++ Openflow API, παρέχει γρήγορες ασύγχρονες λειτουργίες εισόδου - εξόδου,αναπτύσεται σε συστήματα λειτουργικού Linux και κάποιες από τίς σημαντικότερες λειτουργίες αυτής της πλατφόρμας ελέγχου είναι η διερεύνηση της δικτυακής τοπολογίας (Topology discovery) και η λειτουργία Learning Switch [4].

- Beacon: Η Beacon είναι μια modular,Java-based Openflow μονάδα ελέγχου η οποία βρίσκεται υπό ανάπτυξη από το 2010 και από τότε έχει χρησιμοποιηθεί σε αρκετά

projects και σε πειραματικές εφαρμογές. Η μονάδα ελέγχου Beacon έχει αναπτυχθεί με την βοήθεια της γλώσσας προγραμματισμού java και τρέχει σε πολλές και διαφορετικές πλατφόρμες όπως Linux εξυπηρετητές (servers) πολλών πυρήνων και συσκευές android. Οι διάφορες λειτουργίες που συμβαίνουν στο πλαίσιο της μονάδας ελέγχου beacon μπορούν να χαρακτηριστούν ως δυναμικές και ευέλικτες καθώς μια οποιαδήποτε ανεξάρτητη λειτουργία μπορεί να σταματήσει,να ξεκινήσει και να αποεγκατασταθεί χωρίς να επηρεάζονται οι καταστάσεις των άλλων λειτουργιών. Επίσης η Beacon θεωρείται μονάδα ελέγχου υψηλών επιδόσεων καθώς υποστηρίζει υπηρεσίες πολλών-νημάτων (multithreading) αναπτυχθεί σε Java πλατφόρμες όπως Spring και Equinox (OSGi) [5].

- Floodlight: Η μονάδα ελέγχου floodlight διαχειρίζεται από την εταιρεία Big Switch Networks και παρόλο που αρχικά είχε αναπτυχθεί με βάση την δομή και την αρχιτεκτονική της μονάδας ελέγχου Beacon χρησιμοποιεί για την ανάπτυξη εφαρμογών το Apache Ant tool το οποίο είναι ένα διαδεδομένο εργαλείο για την ανάπτυξη λογισμικού και λόγω αυτού η δομή της floodlight μονάδας ελέγχου είναι αρκετά ευέλικτη. Η Floodlight υποστηρίζει μια αρκετά μεγάλη γκάμα λειτουργιών οι οποίες μπορούν να εξατομικευθούν για να συναντήσουν τις απαιτήσεις του εκάστοτε χρήστη [6].

- Opendaylight: Η μονάδα ελέγχου opendaylight έχει αναπτυχθεί από τον οργανισμό Linux Foundation και υποστηρίζεται πάρα πολύ δυναμικά από μεγάλες εταιρίες δικτυακού εξοπλισμού όπως η Cisco, η Big Switch Networks και άλλες. Η ανάπτυξη της μονάδας ελέγχου αυτής βασίσθηκε στην γλώσσα προγραμματισμού Java και θεωρείται αρκετά δημοφιλής και πολύ καλά υποστηριζόμενη από μία ευρύα κοινότητα χρηστών αλλά και developers. Η δεύτερη έκδοση της Opendaylight πλατφόρμας υποστηρίζει εφαρμογές NV (Network Virtualization) και NFV (Network Functions Virtualization) και προορίζεται για υποστήριξη λειτουργιών δικτύων μεγάλης έκτασης. Η μονάδα ελέγχου αυτή αποτελείται από ένα μεγάλο αριθμό στοιχείων όπως διεπαφές, πρωτόκολλα και εφαρμογές που μπορούν να προσαρμοσθούν στις ανάγκες του κάθε χρήστη. Τέλος η μεγάλη διαφορά αυτής της μονάδας ελέγχου από τις υπόλοιπες είναι ότι συνεργάζεται και με άλλα πρωτόκολλα εκτός από το Openflow πρωτόκολλο [7].

- OpenContrail: Η μονάδα ελέγχου OpenContrail θεωρείται project της Juniper και στοχεύει κυρίως σε NV (Network Virtualization) και NFV (Network Functions Virtualization) λειτουργίες και σε δίκτυα τα οποία είναι περιορισμένης έκτασης. Όπως και η Opendaylight η μονάδα ελέγχου OpenContrail εχει μια διεπαφή REST (REST API) η οποία μπορεί να χρησιμοποιηθεί για την παραμετροποίηση του εκάστοτε συστήματος και για την συλλογή πληροφοριών από αυτό [8].

- ONOS: Η μονάδα ελέγχου ONOS είναι μια open-source μονάδα ελέγχου η οποία στοχεύει κυρίως στην παροχή χαρακτηριστικών υψηλής διαθεσιμότητας και επιδόσεων στο δίκτυο (high availability and performance). Επίσης η μονάδα ελέγχου ONOS εξυπηρετεί κυρίως τις ανάγκες των παρόχων δικτύου και επιλύει θέματα όπως προβλήματα κλιμάκωσης του δικτύου. Τέλος με την μονάδα ελέγχου ONOS είναι εφικτός ο έλεγχος τόσο των Openflow συσκευών δικτύου όσο και αυτόν των συσκευών που συνεργάζονται με παραδοσιακά πρωτόκολλα [9].

- Ryu: Η μονάδα ελέγχου Ryu είναι μια μονάδα ελέγχου ευφυών δικτύων η οποία είναι ανεπτυγμένη σε γλώσσα προγραμματισμού Python και παρέχει στοιχεία τα οποία καθιστούν ένα δίκτυο ευέλικτο και εύκολα παραμετροποιήσιμο. Επιπροσθέτως η υποδομή της μονάδας ελέγχου Ryu εξυπηρετεί την εύκολη ανάπτυξη εφαρμογών ευφυών δικτύων και επίσης ο εκάστοτε χρήστης μπορεί να παραμετροποιήσει τα ήδη υπάρχοντα στοιχεία στην υποδομή της Ryu μονάδας ελέγχου ώστε να εξυπηρετήσει καλύτερα τις ανάγκες του [10].

Σε αυτήν την διπλωματική εργασία αναπτύξαμε εφαρμογές στην μονάδα ελέγχου Opendaylight (Opendaylight Controller) οπότε στις επόμενες σελίδες θα αναφερθούμε και θα αναλύσουμε την μονάδα ελέγχου αυτή αλλά και τα επιμέρους στοιχεία που την απαρτίζουν.

4.3 Opendaylight Controller (μονάδα ελέγχου Opendaylight)

Η μονάδα ελέγχου Opendaylight (Opendaylight controller) θεωρείται ως η βασικότερη open-source πλατφόρμα για τον προγραμματισμό ευφυών δικτύων (Software Defined Networks). Αυτή η μονάδα ελέγχου κατατάσσεται ως η πρώτη επιλογή ανάμεσα στις πλατφόρμες των ευφυών δικτύων οι οποίες χρησιμοποιούνται στην βιομηχανία καθώς υποστηρίζει μεγάλο εύρος εφαρμογών και βοηθάει καταλυτικά στην ανάπτυξη των δικτύων του μέλλοντος.

Μεγάλες εταιρίες όπως εταιρίες παροχής υπηρεσιών χρησιμοποιούν την πλατφόρμα Opendaylight για να αντιμετωπίσουν σημαντικές δικτυακές προκλήσεις όπως η αυτόματη παροχή υπηρεσιών δικτύου (automated service delivery), η βελτιστοποίηση του τρόπου κατανάλωσης των πόρων του δικτύου (Network Resource Optimization),η παροχή υπηρεσιών Cloud και NFV (Network Functions Virtualization) και ο καλύτερος έλεγχος και παρακολούθηση του δικτύου.

Οι χρήστες της μονάδας ελέγχου Opendaylight έχουν παρατηρήσει σημαντικές βελτιώσεις στις επιδόσεις, στην ικανότητα κλιμάκωσης και στην λειτουργία ενός δικτύου. Νέες δικτυακές υπηρεσίες που συνοδεύουν την πλατφόρμα Opendaylight προσφέρουν υψηλό βαθμό διαθεσιμότητας, βελτιωμένη και αποδοτική διαχείριση δεδομένων, μεταφορά σημαντικών μηνυμάτων, και αποτελεσματικό διαχωρισμό της μονάδας ελέγχου από το επίπεδο μεταφοράς δεδομένων ενός δικτύου. Έτσι όλα τα προαναφερθέντα στοιχεία υπογραμμίζουν την καταλληλόλητα της μονάδας ελέγχου Opendaylight όσων αφορά την σωστή διαχείριση ενός δικτύου καθώς προσφέρει μια μεγάλη ποικιλία επιλογών για την παροχή εξατομικευμένων υπηρεσιών σε όλους τους χρήστες ανάλογα με τις ανάγκες τους και επίσης διευκολύνει σε μεγάλο βαθμό την μετάβαση από τα παραδοσιακά δίκτυα στα ευφυή προγραμματιζόμενα δίκτυα (Sdn) καθώς ανοίγει τον δρόμο και προσφέρει νέα εργαλεία για την ανάπτυξη εφαρμογών και υπηρεσιών δικτύου.

Η Opendaylight πλατφόρμα αναπτύσσει υπηρεσίες πάνω σε μία μεγάλη ποικιλία υλικού (Hardware) και οι υπηρεσίες αυτές επιτρέπουν στους χρήστες να έχουν το πλήρη έλεγχο στις εφαρμογές, στα πρωτόκολλα και στα plugins που χρησιμοποιούν. Επίσης λόγω της μη ύπαρξης μίας κοινής δομής Ευφυών δικτύων η μονάδα ελέγχου Opendaylight είναι δομημένη με τέτοιο τρόπο ώστε να μπορεί να αντιμετωπίσει κάθε δικτυακή πρόκληση καθώς παρουσιάζει μεγάλο βαθμό προσαρμοστικότητας. Η μονάδα ελέγχου συνδυάζει εργαλεία όπως open-source και open-APIs ώστε να καταφέρει να παρέχει μια πλατφόρμα ευφυών δικτύων που θα καταστεί τα δίκτυα αυτά πιο έξυπνα και προγραμματιζόμενα.

Η μονάδα ελέγχου Opendaylight χρησιμοποιεί μια model-driven στρατηγική για την περιγραφή του δικτύου αλλά και των λειτουργιών που λαμβάνουν χώρα σε αυτό. Χρησιμοποιώντας YANG δομές δεδομένων σε μια κοινή μονάδα αποθήκευση (data store) και υπηρεσία μετάδοσης μηνυμάτων η πλατφόρμα Opendaylight επιτρέπει την δημιουργία υπηρεσιών υψηλών απαιτήσεων οι οποίες συνδυάζονται μεταξύ τους για την αντιμετώπιση πολύπλοκων προβλημάτων. Στο πυρήνα της πλατφόρμας του Opendaylight υπαρχει το στοιχείο που ονομάζεται Model Driven Service Abstraction Layer (MD-SAL) και λόγω αυτού του στοιχείου κάθε εφαρμογή ή λειτουργία μπορεί να μετατραπεί σε υπηρεσία η οποία στην συνέχεια φορτώνεται στην μονάδα ελέγχου Opendaylight. Οι υπηρεσίες αυτές μπορούν να συνδυαστούν μεταξύ τους για να αντιμετωπίσουν τις αυξανόμενες ανάγκες των χρηστών αλλά και του δικτύου. Επίσης λόγω της σπονδυλωτής δομής που διακρίνει την

πλατφόρμα Opendaylight ο κάθε χρήστης έχει την δυνατότητα να χρησιμοποιήσει και να επεκτείνει υπηρεσίες οι οποίες δημιουργήθηκαν από άλλους.

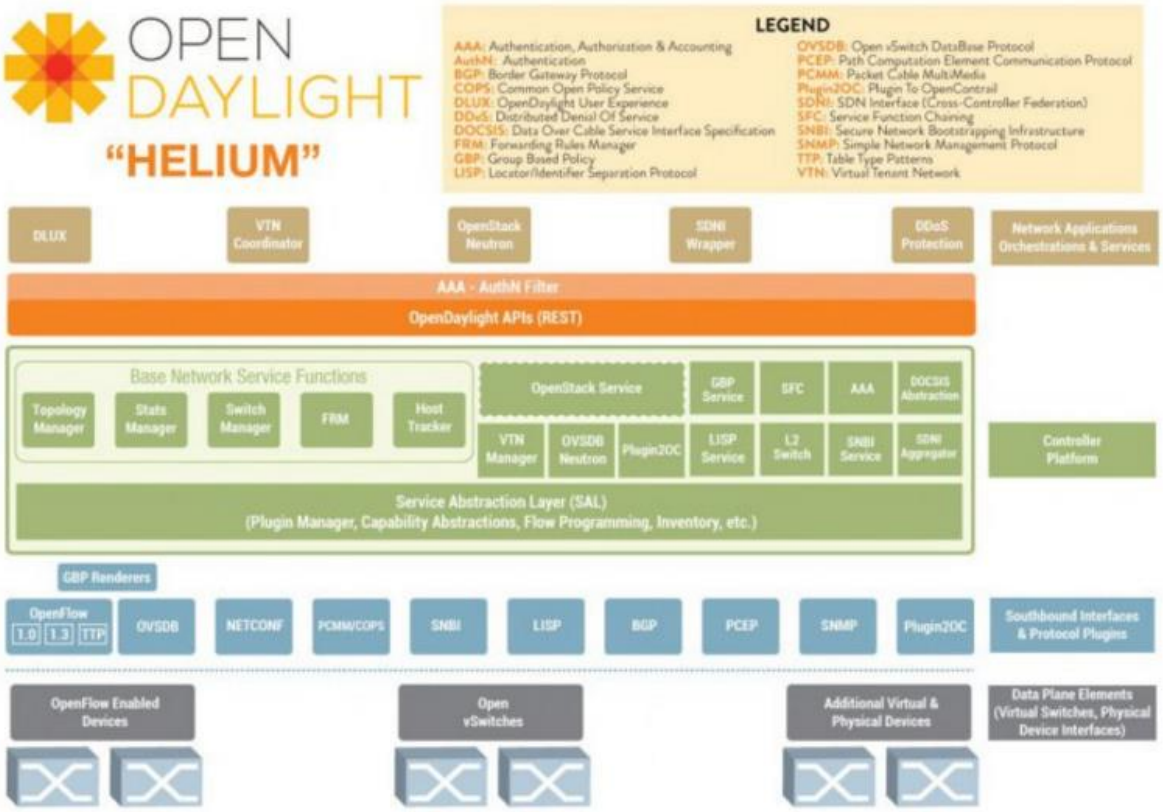
Επίσης η πλατφόρμα Opendaylight υποστηρίζει ένα ευρύ φάσμα δικτυακών πρωτοκόλλων σε μια οποιαδήποτε SDN πλατφόρμα με αποτέλεσμα να επιτυγχάνεται η δυνατότητα προγραμματισμού των νέων δικτύων και η κάλυψη όλων των αναγκων των χρηστών. Για παράδειγμα η Opendaylight μονάδα ελέγχου υποστηρίζει το Openflow πρωτόκολλο αλλά και παραδοσιακά πρωτόκολλα όπως το NETCONF και BGP.

Μέσω των ευφυών δικτύων μπορούμε να επιτύχουμε μεγαλύτερο βαθμό προγραμματισμού και διαχωρισμού στα δίκτυα μας και μέσω της πλατφόρμας ελέγχου Opendaylight μπορούμε και να ελέγξουμε τα δίκτυα αυτά πιο αποδοτικά καθώς η πλατφόρμα αυτή παρέχει προγραμματιζόμενες διεπαφες στους χρήστες (APIs) οι οποίοι μπορούν αποτελεσματικά να αναπτύξουν εφαρμογές στην πλατφόρμα αυτή χωρίς να χρειάζεται να γνωρίζουν τον ακριβή τρόπο λειτουργίας και την εσωτερική οργάνωση κάθε υπηρεσίας και στοιχείου την πλατφόρμας opendaylight. Έπίσης ο αριθμός των διεπαφών αυτών (APIs) αυξάνεται συνεχώς και δημιουργούνται έτσι όλο και περισσότερα εργαλεία στα χέρια των χρηστών για την αντιμετώπιση όλο και περισσότερων δικτυακών προκλήσεων.

Τέλος η σπονδυλοειδή δομή και η ευελιξία που παρουσιάζει η συγκεκριμένη μονάδα ελέγχου επιτρέπει στους χρήστες να επιλέξουν οποιαδήποτε από τις λειτουργίες του για να εξυπηρετήσουν τις ανάγκες τους.

4.3.1 Αρχιτεκτονική Opendaylight Controller

Πριν προχωρήσουμε σε βάθος την ανάλυσή μας σχετικά με την μονάδα ελέγχου Opendaylight θα αναφερθούμε αρχικά στην αρχιτεκτονική και στα διάφορα δομικά στοιχεία που αποτελούν την μονάδα ελέγχου αυτή. Στην παρακάτω εικόνα (Εικόνα 5) γινόμαστε μάρτυρές της αρχιτεκτονικής και της δομής της μονάδας ελέγχου Opendaylight όπως επίσης και όλων των βασικών μερών που απαρτίζουν την μονάδα και λαμβάνουν μέρος σε όλες τις βασικές της λειτουργίες [11].



Αρχιτεκτονική Opendaylight controller

Εικόνα 5

Η μονάδα ελέγχου Opendaylight είναι μια σπονδυλοειδής πλατφόρμα της οποίας τα περισσότερα στοιχεία (modules) χρησιμοποιούν και επαναχρησιμοποιούν κοινές λειτουργίες. Η ιδέα για την κατασκευή και ανάπτυξη εφαρμογών στο εσωτερικό της μονάδας ελέγχου Opendaylight προήλθε από την ανάγκη χρησιμοποίησης των υπηρεσιών που προσφέρουν οι εφαρμογές αυτές μέσω προγραμματιζόμενων διεπαφών (java interfaces). Οι εφαρμογές αυτές είναι κατασκευασμένες στο εσωτερικό της πλατφόρμας ακολουθώντας το μοντέλο παρόχου - καταναλωτή (provider-consumer) και συνεργάζονται αποδοτικά με τον πυρήνα της πλατφόρμας Opendaylight που ονομάζεται MD-SAL (Model Driven Service Abstraction Layer). Η ανάπτυξη εφαρμογών στην πλατφόρμα Opendaylight προϋποθέτει την υιοθέτηση της Model-View-Control προσέγγιση η οποία προάγει :

- Την χρησιμοποίηση του μοντέλου YANG για τον καθορισμό των δεδομένων, των RPC και των ειδοποιήσεων (notifications).

- Την ανάπτυξη REST διεπαφών (REST APIS) οι οποίες είναι προσβάσιμες μέσω RESTconf.

- Την ανάπτυξη Java εφαρμογών για την εξυπηρέτηση ειδοποιήσεων, RPC και μεταβολών στα δεδομένα.

4.3.2 MD-SAL (Model Driven-Service Abstraction Layer)

Το Model Driven-Service Abstraction Layer (MD-SAL) είναι το κέντρο της πλατφόρμας Opendaylight όπου όλα τα διαφορετικά στρώματα και οι εφαρμογές είναι διασυνδεδεμένα μέσω καλά καθορισμένων προγραμματιζόμενων διεπαφών (APIs). Κάθε προγραμματιζόμενη διεπαφη παράγεται μέσω μοντέλων που είναι κατασκευασμένα σε γλώσσα YANG και δημιουργείται κατά την διάρκεια της εγκατάστασης και φόρτωσης της εφαρμογής στην πλατφόρμα Opendaylight.

Στον πυρήνα τής πλατφόρμας υπάρχει μία λογικά κεντρικοποιημένη μονάδα αποθήκευσης δεδομένων (centralized data store) στην οποία αποθηκεύονται η κατάσταση λειτουργίας αλλά και άλλα δεδομένα τα οποία έχουν σχέση με τις εφαρμογές που είναι ενεργές στην πλατφόρμα Opendaylight. Όλες οι κλήσεις που γίνονται και τα δεδομένα που

μεταφέρονται μεταξύ μια εφαρμογής-παρόχου και μιας εφαρμογής-καταναλωτή περνάνε από αυτήν την κεντρικοποιημένη μονάδα αποθήκευσης δεδομένων χρησιμοποιώντας την MD-SAL χαρτογράφηση (mapping).Η κεντρικοποιημένη μονάδα αποθήκευσης δεδομένων διαχωρίζεται λογικά σε δύο επιμέρους μονάδες αποθήκευσης δεδομένων την operational data store και την config data store [12].

- ***Config data store:*** Στην config μονάδα αποθήκευσης δεδομένων αποθηκεύονται δεδομένα τα οποία έχουν σχέση με ενέργειες που οι χρήστες στέλνουν προς την πλατφόρμα Opendaylight. Για παράδειγμα δεδομένα τα οποία στέλνονται στην πλατφόρμα Opendaylight μεσω REST αποθηκεύονται στην config μονάδα αποθήκευσης.

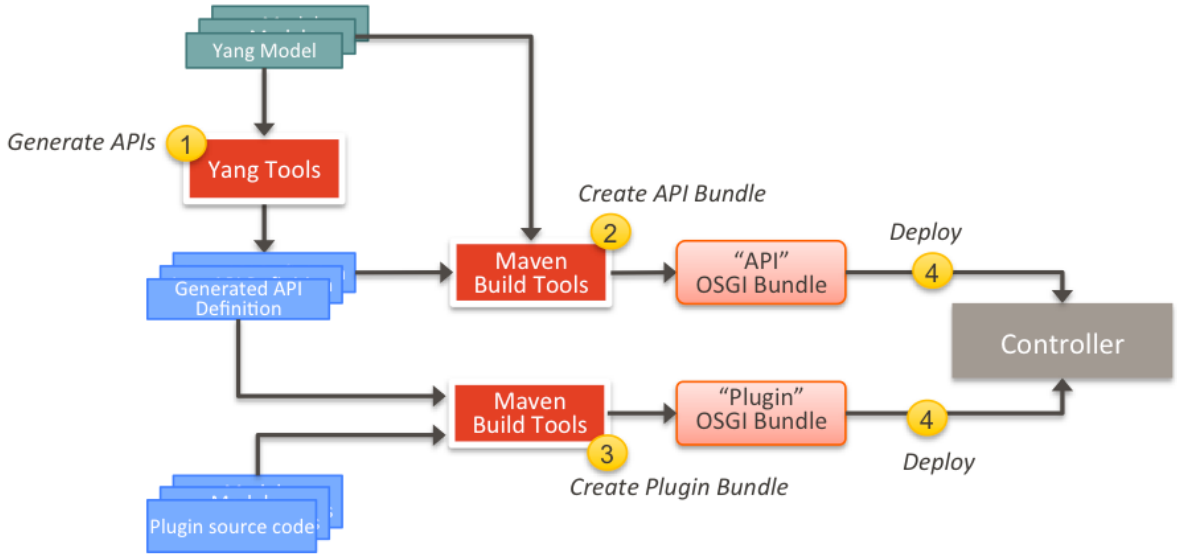
- ***Operational data store:*** Στην operational μονάδα αποθήκευσης δεδομένων αποθηκεύονται δεδομένα τα οποία το ίδιο το σύστημα ανακάλυψε και αποθήκευσε στην μονάδα αυτή. Για παράδειγμα δεδομένα τα οποία είναι σχετικά με το αν μια συσκευή δικτύου βρίσκεται σε λειτουργία ή όχι (switch state) αποθηκεύονται από το ίδιο το σύστημα στην operational μονάδα αποθήκευσης δεδομένων.

Όλες οι εφαρμογές που υπάρχουν στο MD-SAL μπορούν να θεωρηθούν πάροχοι αλλά και καταναλωτές δεδομένων και επίσης το MD-SAL είναι υπεύθυνο να συνδέσει τίς εφαρμογές-καταναλωτές με τις εφαρμογές-παρόχους και να κατασκευάσει ένα δίαυλο επικοινωνίας μεταξύ των εφαρμογών ώστε να γίνει σωστά η εναλλαγή των δεδομένων. Ο κώδικας των προγραμματιζόμενων διεπαφων (APIs) που είναι απαραιτήτως για την επικοινωνία μεταξύ των εφαρμογών (plugins) δημιουργείται από τα Yang μοντέλα κατά την διάρκεια της μετάφρασης (compile time) μιας εφαρμογής. Όταν μια εφαρμογή φορτώνεται στον στην πλατφόρμα Opendaylight μαζί με αυτήν φορτώνεται και ο κώδικας των προγραμματιζόμενων διεπαφών (APIs).

Η λειτουργικότητα η οποία παρέχεται από το MD-SAL είναι η παροχή της "καλωδίωσης" για την σύνδεση των εφαρμογών-παρόχων και των εφαρμογών-καταναλωτών. Μία εφαρμογή καταναλωτής ή πάροχος μπορεί να δηλώσει την ύπαρξη της στο MD-SAL (registration procedure) ,μια εφαρμογή-καταναλωτής μπορεί να βρει μια εφαρμογή-πάροχο αποτελεσματικά , μια εφαρμογή-πάροχος μπορεί να παράγει ειδοποιήσεις (notifications) και να αποθηκεύσει δεδομένα στην μονάδα αποθήκευσης δεδομένων και μια εφαρμογη-καταναλωτης μπορεί να λάβει ειδοποιήσεις και να διαβάσει δεδομένα από την μονάδα αποθήκευσης. Όλα τα παραπάνω δεν θα ήταν εφικτά χωρίς το MD-SAL του οποίου η παρουσία στην πλατφόρμα Opendaylight είναι απαραίτητη για την σωστή και άμεση επικοινωνία μεταξύ διαφόρων εφαρμογών.

Στην συνέχεια θα αναλύσουμε την διαδικασία που ακολουθείται για την κατασκευή μιας εφαρμογής (plugin) στην πλατφόρμα Opendaylight.

Όλες οι εφαρμογές στην πλατφόρμα Opendaylight παρουσιάζουν τον ίδιο «κύκλο ζωής». Η ύπαρξη μιας εφαρμογής βασίζεται σε δύο φάσεις, την φάση σχεδιασμού και την φάση εκτέλεσης της εφαρμογής. Η Εικόνα 6 παρουσιάζει τις δύο αυτές φάσεις.



Σχεδιασμός και εκτέλεση εφαρμογής

Εικόνα 6

Κατά την διάρκεια σχεδίασης μιας εφαρμογής ακολουθούνται οι παρακάτω διαδικασίες:

- Αρχικά πρέπει να αποφασισθεί πια δεδομένα θα καταναλωθούν από την εφαρμογή και για τον λόγω αυτό πρέπει να εισάγουμε όλες τις απαραίτητες προγραμματιζόμενες διεπαφές (APIs) οι οποίες έχουν δημιουργηθεί αυτόματα κατά την διάρκεια μετάφρασης (compile) των YANG μοντέλων που ανήκουν στις εφαρμογές παρόχους.

- Σε δεύτερο στάδιο πρέπει να αποφασισθεί πια δεδομένα θα παρέχονται από την νέα εφαρμογή (plugin) για αυτό το λόγω πρέπει να σχεδιασθεί το YANG μοντέλο της νέας εφαρμογής το οποίο στην συνέχεια επεξεργάζεται από το στάδιο YANG Tools ώστε να δημιουργηθούν οι προγραμματιζόμενες διεπαφες (APIs) του YANG μοντέλου της νέας εφαρμογής.

- Στην συνέχεια η εκτέλεση (implementation) αυτών των προγραμματιζόμενων διεπαφων (APIs) που δημιουργήθηκαν από τα YANG μοντέλα μαζί με άλλα χαρακτηριστικά και λειτουργίες επεξεργάζονται και στην συνέχεια ο καταληκτικώς κώδικας πακετάρεται σε μια OSGI bundle της εφαρμογής (plugins OSGI bundle).

- Τέλος πρέπει να σημειώσουμε πως οι προγραμματιζόμενες διεπαφές (APIs) που δημιουργήθηκαν από τα μοντέλα πακετάρονται και αυτές μαζί με κάποιες βοηθητικές κλάσεις (helper classes) σε μια APIs OSGI bundle.

Μόλις η OSGI bundle της εφαρμογής εγκατασταθεί στην πλατφόρμα Opendaylight και ενεργοποιηθεί τότε ξεκινάει και η φάση της εκτέλεσης.

4.3.3 YANG μοντέλα (YANG models)

Η Yang είναι μια γλώσσα προγραμματισμού η οποία χρησιμοποιείται για τη μοντελοποίηση δεδομένων (data modeling). Το όνομα Yang είναι ακρωνύμιο της πρότασης Yet Another Next Generation και αναπτύχθηκε από το NETMOD το οποίο ήταν ένα group εργασίας στην IETF και δημοσιεύθηκε σαν RFC 6020 τον Οκτώβριο του 2010.

Αυτή η γλώσσα μοντελοποίησης δεδομένων μπορεί να χρησιμοποιηθεί για να μοντελοποιηθούν δεδομένα παραμετροποίησης (configuration data) όπως και επίσης δεδομένα κατάστασης (state data) μια συσκευής δικτύου. Ακόμα η YANG μπορεί να χρησιμοποιηθεί για την μοντελοποίηση ειδοποιήσεων (notifications) και απομακρυσμένων κλήσεων διαδικασιών (RPC) [13].

Όσων αφορά την πλατφόρμα Opendaylight η γλώσσα Yang διαδραματίζει έναν βασικό ρόλο για την λειτουργία της. Καθώς η πλατφόρμα χρησιμοποιεί την Yang για την μοντελοποίηση δεδομένων, ειδοποιήσεων (notifications) και RPCs τα οποία χρησιμοποιούνται από τις διάφορες εφαρμογές της πλατφόρμας Opendaylight. Με την Yang επιτυγχάνουμε να περιγράψουμε την βασική δομή δεδομένων των εφαρμογών της πλατφόρμας τα οποία αποθηκεύονται με δενδροειδή ιεραρχική δομή σε containers.

Στην Εικόνα 7 παρουσιάζεται ένα παράδειγμα του μοντέλου Yang για την αποθήκευση κόμβων (switches) και υποδοχέων-κόμβων (interfaces or ports) το οποίο καθορίστηκε μέσα στο opendaylight-inventory.yang

```
module opendaylight-inventory {
    namespace "urn:opendaylight:inventory";
    container nodes {
        list node {
            key "id";
            leaf id {
                type node-id;
            }
            list "node-connector" {
                key "id";
                leaf id {
                    type node-connector-id;
                }
            }
        }
    }
}
```

Παράδειγμα μοντέλου Yang

Εικόνα 7

Επίσης η γλώσσα Yang μπορεί να χρησιμοποιηθεί και για άλλες λειτουργίες οι οποίες αναφέρονται παρακάτω επιγραμματικά. Λειτουργίες όπως :

- **Augmentations :** Τα Yang μοντέλα μπορούν να επεκταθούν για να προστεθούν επιπλέον δεδομένα σε ένα υπάρχον μοντέλο δεδομένων.

- **Notifications :** Χρησιμοποιούνται για την δημοσίευση μίας ή περισσοτέρων ειδοποιήσεων. Ο καθορισμός μέσω Yang ειδοποιήσεων δημιουργεί κάποιες Java διεπαφές (java interfaces) τις οποίες μια πιθανή εφαρμογή μπορεί να χρησιμοποιήσει για να λαμβάνει ειδοποιήσεις όπως για παράδειγμα ειδοποιήσεις για πιθανή μεταβολή κάποιον στοιχείων σε μία συσκευή δικτύου.

- **RPC :** Το MD-SAL επιτρέπει σε μία εφαρμογή να πραγματοποιήσει κλήση μια διαδικασίας ακόμα και αν η εφαρμογή αυτή δεν γνωρίζει ποιος είναι ο πάροχος αυτής της διαδικασίας.

4.3.4 Instance Identifiers

Στην πλατφόρμα Opendaylight μία εφαρμογή ή ένας εξωτερικός χρήστης μπορεί να εισάγει δεδομένα στην μονάδα αποθήκευσης δεδομένων είτε μέσω μίας MD-SAL συναλλαγής είτε μέσω RESTconf. Τα διάφορα αντικείμενα είναι αποθηκευμένα στην μονάδα αποθήκευσης δεδομένων σε μία ιεραρχική δομή που έχει την μορφή "γονέα-παιδιού" (parent-child hierarchy) και η πρόσβαση σε αυτά τα αντικείμενα γίνεται μέσω Yang Instance Identifiers. Ας υποθέσουμε για παράδειγμα για το Yang μοντέλο που παρουσιάσαμε στην Εικόνα 7 ότι υπάρχει ένας node-connector με το όνομα "opneflow:1:1" ο οποίος είναι αποθηκευμένος στην μονάδα αποθήκευσης δεδομένων. Εάν μία εφαρμογή θέλει να έχει πρόσβαση σε λεπτομέρειες που αφορούν αυτών τον node-connector πρέπει να δημιουργήσει έναν instance identifier όπως παρουσιάζεται στην Εικόνα 8.

```
InstanceIdentifier ncIID = InstanceIdentifier.builder(Nodes.class)
    .child(Node.class, new NodeKey(new NodeId("openflow:1")))
    .child(NodeConnector.class, new NodeConnectorKey(new NodeConnectorId("openflow:1:1")))
    .build();
```

Παράδειγμα Instance Identifiers

Εικόνα 8

Εναλλακτικά οι λεπτομέρειες για τον συγκεκριμένο node-connector μπορούν να είναι προσβάσιμες από έναν χρήστη και μέσω RESTconf χρησιμοποιόντας το URL :"http://localhost:8181/restconf/config/opendaylight-inventory:nodes/node/openflow:1/node-connector/openflow:1:1" όπου το "opendaylight-inventory" καταδεικνύει το όνομα του μοντέλου δεδομένων και το "nodes" καταδεικνύει το όνομα του container που αποτελεί και τη ρίζα του δένδρου [14].

4.3.5 RESTCONF

Υπήρχε πάντοτε η ανάγκη ύπαρξης ενός βασικού μηχανισμού που θα επέτρεπε σε web εφαρμογές να έχουν πρόσβαση σε δεδομένα, ειδοποιήσεις και μοντέλα μιας συσκευής δικτύου με έναν άμεσο τρόπο. Έτσι για την εξυπηρέτηση αυτής της ανάγκης δημιουργήθηκε ένα HTTP πρωτόκολλο που ονομάζεται RESTCONF και επιτρέπει την πρόσβαση σε δεδομένα τα οποία είναι μοντελοποιημένα μέσω της γλώσσας μοντελοποίησης YANG και αποθηκευμένα σε μονάδες αποθήκευσης δεδομένων (datastores) οι οποίες είναι καθορισμένες μέσω NETCONF.

Το NETCONF πρωτόκολλο καθορίζει μονάδες αποθήκευσης δεδομένων και ένα σετ από Create, Retrieve , Update, Delete (CRUD) λειτουργίες οι οποίες μπορούν να χρησιμοποιηθούν για την πρόσβαση σε αυτές τις μονάδες αποθήκευσης δεδομένων [15].

Η γλώσσα μοντελοποίησης YANG καθορίζει το συντακτικό και την σημασιολογία (semantics) των περιεχομένων μίας μονάδας αποθήκευσης δεδομένων, των μοντέλων δεδομένων και των ειδοποιήσεων.

Έτσι το RESTCONF πρωτόκολλο χρησιμοποιεί HTTP λειτουργίες για να εκτελέσει CRUD λειτουργίες σε NETCONF καθορισμένες μονάδες αποθήκευσης δεδομένων οι οποίες περιέχουν δεδομένα μοντελοποιημένα με την γλώσσα μοντελοποίησης YANG. Δεδομένα τα οποία υπόκεινται σε παραμετροποίηση (configuration data) και δεδομένα τα οποία δηλώνουν πληροφορίες κατάστασης (state data) μπορούν να διαβαστούν μέσω της μεθόδου GET του RESTCONF πρωτοκόλλου. Επίσης δεδομένα τα οποία υποκινται σε παραμετροποίηση μπορούν να παραμετροποιηθούν μέσω μεθόδων όπως DELETE , PATCH ,POST και PUT. Επίσης τα δεδομένα τα οποία στέλνονται ή λαμβάνονται μέσω των παραπάνω μεθόδων του RESTCONF πρωτοκόλλου είναι κωδικοποιημένα σε XML ή JSON.

Ακόμα RPC λειτουργίες που είναι μοντελοποιημένες μέσω YANG μπορούν να εκτελεστούν και μέσω του RESTCONF πρωτοκόλλου μέσω της μεθόδου POST για παράδειγμα όπως επίσης μπορεί να υπάρξη πρόσβαση και σε ειδοποιήσεις (notifications) οι οποίες είναι μοντελοποιημένες μέσω YANG.

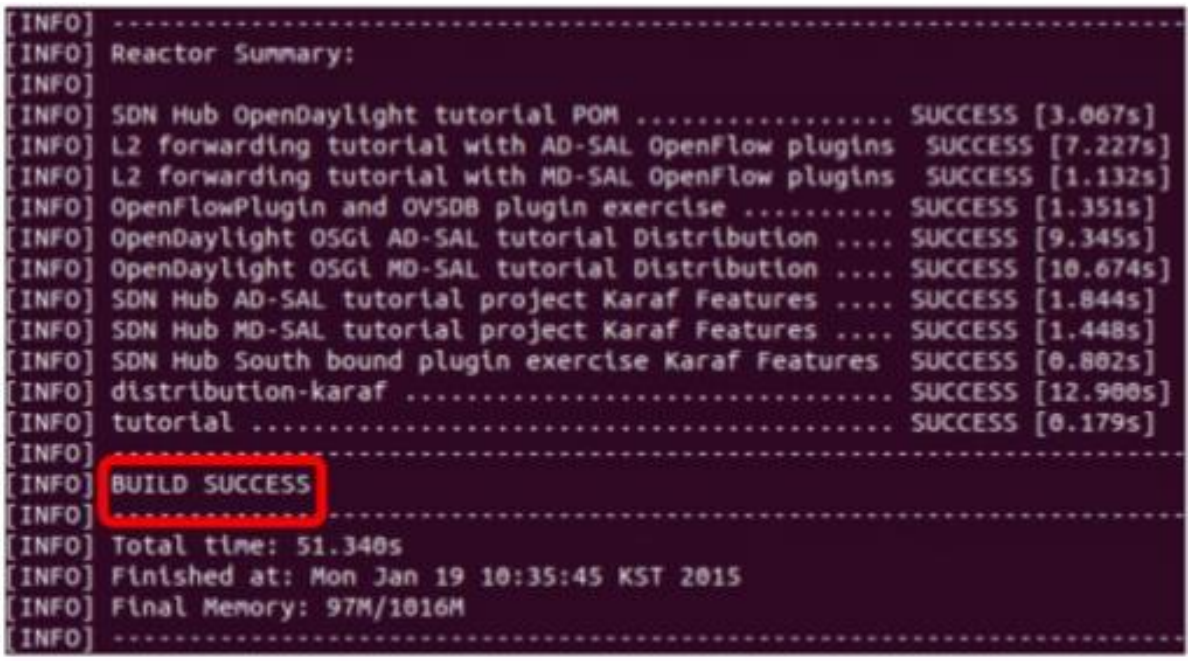
4.3.6 Apache Maven

Η μονάδα ελέγχου Opendaylight χρησιμοποιεί κατά κόρον τον Apache Maven το οποίο είναι ένα εργαλείο που εξυπηρετεί την εύκολη κατασκευή projects. Πιο συγκεκριμένα το Apache Maven χρησιμοποιεί αρχεία POM.xml (Project Object Model) για να χαρτογραφήσει τις εξαρτήσεις μεταξύ των βασικών στοιχείων που αποτελούν μια εφαρμογή. Στην συνέχεια θα αναφέρουμε μερικά στοιχεία για τον Apache Maven.

Ο Apache Maven δημιουργήθηκε για να εξυπηρετήσει την ανάγκη μιας σταθερής και αξιόπιστης λύσης στο πρόβλημα της κατασκευής και διατήρησης μεγάλων projects. Επίσης με την συμβολή του Apache Maven έγινε εφικτός ο καθορισμός των στοιχείων που περιλαμβάνει ένα project , η δημοσίευση πληροφοριών για κάποιο συγκεκριμένο project και επίσης βελτιστοποιήθηκε η διαδικασία διαμοιρασμού JAR αρχείων μεταξύ διαφόρων projects. Οι βασική στόχοι που εξυπηρετούνται με την βοήθεια του Apache Maven είναι :

- Η παροχή διευκολύνσεων στην κατασκευή ενός project.
- Η παροχή ενός ενιαίου συστήματος κατασκευής projects.'
- Η παροχή ποιοτικών πληροφοριών που αφορούν το υπό κατασκευή project.
- Παροχή καθοδήγησης για την βελτιστοποίηση των πρακτικών που ακολουθούνται στην ανάπτυξη εφαρμογών.

Στην συνέχεια θα αναφερθούμε στην λειτουργία του Apache Maven στο πλαίσιο της πλατφόρμας Opendaylight και στο σημαντικό ρόλο που επιτελεί. Παρακάτω παραθέτουμε την Εικόνα 9 η οποία απεικονίζει ένα παράδειγμα μιας επιτυχούς εγκατάστασης μέσω Apache Maven της πλατφόρμας Opendaylight. Η λέξη SUCCESS στο τέλος κάθε module σημαίνει ότι το επιμέρους module έχει εγκατασταθεί με επιτυχία. Στην προκειμένη περίπτωση όλα τα επιμέρους modules έχουν εγκατασταθεί με επιτυχία από τον Apache Maven και για το λόγο αυτό βλέπουμε στο κάτω μέρος της Εικόνας 9 την φράση BUILD SUCCESS. Εάν έστω και ένα επιμέρους module δεν είχε εγκατασταθεί με επιτυχία από τον Apache Maven τότε η φράση BUILD SUCCESS θα είχε αντικατασταθεί με την φράση BUILD FAILURE πράγμα που θα σήμαινε ότι η συνολική εγκατάσταση της πλατφόρμας Opendaylight από τον Apache Maven θα είχε αποτύχει [16].



Apache Maven κατασκευή εφαρμογής

Εικόνα 9

Για να εγκαταστήσουμε την πλατφόρμα Opendaylight χρησιμοποιούμαι την εντολή σε Linux Terminal "mvn" η οποία ενεργοποιεί τον Apach Maven για να ξεκινήσει το χτίσιμο της πλατφόρμας μεταφράζοντας (compile) POM.xml αρχεία.

Ο Apache Maven λειτουργεί με γνώμονα την επίλυση των εξαρτήσεων μεταξύ των διαφόρων πακέτων της πλατφόρμας Opendaylight. Για παράδειγμα μπορεί μια καινούργια εφαρμογή να χρησιμοποιεί κάποια interfaces ή κλάσεις οι οποίες έχουν αναπτυχθεί από κάποια άλλη εφαρμογή άρα ο Apache Maven πρέπει να επιλύσει αυτήν την εξάρτηση που παρουσιάζει η νέα εφαρμογή από την παλαιότερη εφαρμογή. Αυτές οι εξαρτήσεις δηλώνονται στο POM.xml αρχείο της νέας εφαρμογής ώστε ο Apache Maven να καταλάβει ποιες εξαρτήσεις χρειάζεται να επιλύσει για την λειτουργία της νέας εφαρμογής. Επίσης στο αρχειο "~/.m2/repository " ο Apache Maven τοποθετεί όλα τα κατεβασμένα JAR αρχεία πράγμα το οποίο βοηθάει τον χρήστη να κατανοήσει σε περίπτωση αδυναμίας εγκατάστασης της πλατφόρμας Opendaylight αν λείπει κάτι το οποίο οδήγησε την εγκατάσταση σε αποτυχία.

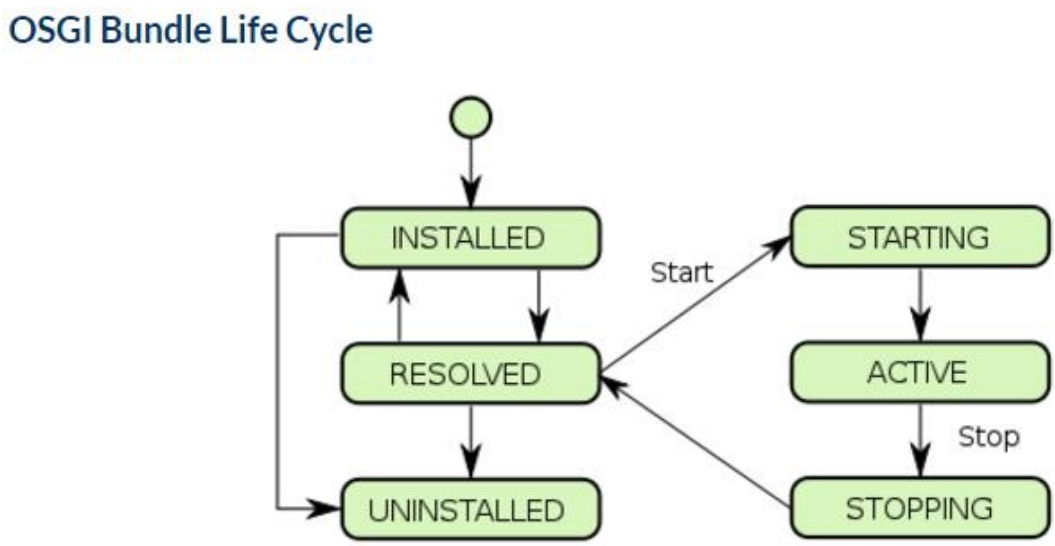
Τα κύρια modules τα οποία εγκαθιστά ο Apache Maven αναφέρονται στο αρχικό POM.xml (root POM.xml). Περαιτέρω κάθε "module-παιδί" έχει το δικό του POM.xml. Όλα αυτά τα POM.xml εγκαθίστανται από τον Apache Maven σαν ξεχωριστές οντότητες και στο τέλος συνδυάζονται για να αποτελέσουν την συνολική δομή της πλατφόρμας Opendaylight.

4.3.7 OSGi framework

Η λέξη OSGi είναι συντομογραφία της έκφρασης Open Service Gateway Initiative.Είναι ένα Java framework που εξυπηρετεί την ανάπτυξη και εκτέλεση σπονδυλωτών (modular) εφαρμογών.

Το OSGi framework ουσιαστικά αποτελείται από δύο μέρη. Το πρώτο μέρος είναι ο καθορισμός των σπονδυλωτών στοιχείων που ονομάζουμε bundles και έχει να κάνει με ένα "χάρτη οδηγιών" για το κύκλο ζωής της κάθε bundle και για το πως αλληλοεπιδρούν με το περιβάλλον οι bundles αυτές και το δεύτερο μέρος είναι η εικονική μηχανή java (java virtual machine aka jvm) την οποία χρησιμοποιούν οι bundles για να εκθέσουν τις λειτουργίες που εξυπηρετούν αλλά και να συνδεθούν με άλλες υπάρχουσες λειτουργίες.

Το OSGi framework σε γενικές γραμμές καθορίζει την αρχιτεκτονική για μια σπονδυλωτή (modular) ανάπτυξη εφαρμογών. Όπως φαίνεται και στην Εικόνα 10 που παρατίθεται το OSGi framework επιτρέπει σε java bundles να φορτώνονται, καταργούνται, ξεκινούν και να σταματούν με ένα δυναμικό χαρακτήρα χωρίς να χρειάζεται να διακόπτεται η λειτουργία της εικονικής μηχανής java κάθε φορά που αλλάζει η κατάσταση λειτουργίας καποιας bundle [17].



Κύκλος ζωής OSGi bundle

Εικόνα 10

Οι bundles που αναφέραμε παραπάνω είναι ουσιαστικα JAR αρχεία και manifest αρχεία που προσδιορίζουν τις λειτουργίες που εξάγει μια bundle για χρησιμοποίηση από άλλες εφαρμογές της πλατφόρμας Opendaylight και επίσης ποιες λειτουργίες εφαρμογών που ήδη υπάρχουν στην πλατφόρμα Opendaylight εισάγει η bundle ως προϋπόθεση για την λειτουργία της.

Μία bundle μπορεί να λειτουργήσει και σαν σημείο παροχής υπηρεσιών στο σύστημα αλλά και σαν σημείο οπού καταναλώνονται υπηρεσίες που ήδη υπάρχουν στο σύστημα. Οι υπηρεσίες αυτές υπάρχουν με την μορφή java διεπαφών (java interfaces) και το μόνο που χρειάζεται μια bundle για να καταναλώσει μια υπηρεσία είναι να υλοποιήσει την αντίστοιχη java διεπαφή της υπηρεσίας.

Το OSGi framework επιτρέπει στην πλατφόρμα Opendaylight την δυναμική φόρτωση των bundles και την αποτελεσματική διασύνδεση τους για να γίνει εφικτή η ανταλλαγή πληροφοριών.

Στην μονάδα ελέγχου Opendaylight χρησιμοποιούμε δύο OSGi containers τα οποία βρίσκονται σε αγαστή συνεργασία ούτως ώστε να δημιουργήσουν το OSGi framework για την δυναμική διαχείριση των bundles. Τα δύο αυτά OSGi containers είναι τα Equinox και Apache Felix.

- **Equinox :** Το Equinox είναι ένα OSGi container [18] το οποίο χρησιμοποιείται για την εφαρμογή και λειτουργία διαφόρων OSGi υπηρεσιών. Στην πλατφόρμα Opendaylight το Equinox χρησιμοποιείται σαν σημείο ενεργοποίησης των Bundles που βρίσκονται στην πλατφόρμα Opendaylight .

- **Apache Felix :** Το Apache Felix είναι ένα έτερο OSGi container [19] το οποίο χρησιμοποιείται στην πλατφόρμα Opendaylight κυρίως κατά την περίοδο κατασκευής της πλατφόρμας μέσω του Apache Maven. Το Apache Felix περιέχει ένα plugin το οποίο βοηθά στον σχηματισμό των δεδομένων που παρέχει ο Apache Maven κατά το χτίσιμο της πλατφόρμας Opendaylight σε OSGi bundles.

4.3.8 Apache Karaf

Το Apache Karaf είναι ένα σύγχρονο και πολυμορφικό container το οποίο διαδραματίζει ένα βασικό ρόλο στην πλατφόρμα Opendaylight. Πιο συγκεκριμένα το Apache Karaf είναι ένα container το οποίο περιβάλλει το OSGi framework (Apache Felix και Equinox) έτσι ώστε να επιτυγχάνεται η σωστή διαχείριση των εφαρμογών της πλατφόρμας Opendaylight και η σωστή λειτουργία τους στο OSGi περιβαλλον [20]. Στην συνέχεια θα αναφέρουμε επιγραμματικα κάποια από τα πλεονεκτήματα του Apache Karaf τα οποία παρ'εχονται στην πλατφόρμα Opendaylight.

- **Ολοκληρωμένη Κονσόλα :** Το Apache Karaf παρέχει μια ολοκληρωμένη κονσόλα τύπου Unix από την οποία ο χρήστης μπορεί να ελέγχει το Apache Karaf container και κατ' επέκταση την πλατφόρμα Opendaylight.

- **Δυναμική παραμετροποίηση :** Το Apache Karaf παρέχει ένα σετ από εντολές οι οποίες είναι κατάλληλες για την παραμετροποίηση του ίδιου του container. Όλα τα αρχεία τα οποία είναι επιφορτισμένα με την παραμετροποίηση του Apache Karaf βρίσκονται στο εσωτερικό του φακέλου *etc.* Οποιαδήποτε αλλαγή σε αυτά τα αρχεία παραμετροποίησης γίνεται αντιληπτή από τον Apache Karaf και εγκαθίσταται.

- **Ανεπτυγμένο σύστημα καταγραφής :** Το Apache Karaf είναι σε θέση να υποστηρίζει όλα τα δημοφιλή εργαλεία καταγραφής πληροφοριών όπως τα slf4j και log4j.

- **Απομακρυσμένη Πρόσβαση :** Το Apache Karaf συνεργάζεται με ένα SSHd Server δίνοντας έτσι την δυνατότητα στον χρήστη να συνδέεται και να ελέγχει το Apache Karaf container από οπουδήποτε θελήσει.

- **Ασφάλεια :** Το Apache Karaf παρέχει ένα ικανοποιητικό πλαισιο ασφαλείας το οποίο είναι βασισμένο στο JAAS (Java Authentication and Authorization Service) και

επίσης παρέχει ένα RBAC (Role-Based Access Control) μηχανισμό για την πρόσβαση του χρήστη στην κονσόλα.

- **OSGi Frameworks :** Όπως αναφέραμε και στην προηγούμενη ενότητα το Apache Karaf δεν είναι στενά συνδεδεμένο με ένα και μόνο OSGi framework ο χρήστης μπορεί με ευκολία να χρησιμοποιήσει οποιοδήποτε OSGi framework θελήσει (Apache Felix,Equinox) παραμετροποιόντας κατάλληλα τα αρχεία παραμετροποίησης που βρίσκονται στον *etc* φάκελο.

Μόλις επιτευχθεί η εγκατάσταση της πλατφόρμας Opendaylight τοτε ο χρήστης μπορεί να προηγηθεί στον φάκελο bin μέσα από τον οποίο μπορεί να εκκινήσει την πλατφόρμα Opendaylight φορτώνοντας το Apache Karaf container. Μόλις γίνει αυτό τότε ο χρήστης θα έχει πρόσβαση στην κονσόλα τύπου Unix του Apache Karaf η οποία θα έχει την μορφή που απεικονίζεται στην Εικόνα 11.

Opendaylight ASCii

Εικόνα 11

4.3.9 Διαδικασία ανάπτυξης εφαρμογής στην Opendayligth πλατφόρμα

Στο σημείο αυτό θα αναφερθούμε και θα εξηγήσουμε τα βασικά σημεία της διαδικασίας ανάπτυξης μιας νέας εφαρμογής στην πλατφόρμα Opendaylight. Η διαδικασία αυτή είναι σημαντικό βήμα για την κατανόηση της εφαρμογής που έχουμε αναπτύξει στα πλαίσια αυτής της διπλωματικής εργασίας.

Θα ξεκινήσουμε και θα κατασκευάσουμε ένα "Hello" project χρησιμοποιώντας τον Apache Maven και ένα αρχέτυπο (archetype) που ονομάζεται "opendaylight-startup-archetype" όπως φαίνεται στην Εικόνα 12.

```
mvn archetype:generate -DarchetypeGroupId=org.opendaylight.controller -DarchetypeArtifactId=opendaylight-startup-archetype \
-DarchetypeRepository=http://nexus.opendaylight.org/content/repositories/opendaylight.snapshot/ \
-DarchetypeCatalog=http://nexus.opendaylight.org/content/repositories/opendaylight.snapshot/archetype-catalog.xml \
-DarchetypeVersion=1.2.0-SNAPSHOT
```

Opendaylight startup archetype

Εικόνα 12

Μόλις εκτελέσουμε την παραπάνω εντολή σε Linux περιβάλλον τότε κατεβαίνουν στον υπολογιστή μας και φορτώνονται όλα τα απαραίτητα αρχεία για την κατασκευή μιας βασικής εφαρμογής στην Opendaylight πλατφόρμα.

Στην συνέχεια μετά την εκτέλεση της εντολής στην εικόνα 12 θα μας ζητηθεί να εισάγουμε κάποια στοιχεία τα οποία είναι απαραίτητα για την κατασκευή του νέου αυτού project. Τα στοιχεία που θα μας ζητηθεί να εισακτούν φαίνονται στην εικόνα 13.

```
Define value for property 'groupId': : org.opendaylight.hello
Define value for property 'artifactId': : hello
Define value for property 'version': 1.0-SNAPSHOT
Define value for property 'package':  org.opendaylight.hello: :
Define value for property 'classPrefix':  hello
Define value for property 'copyright': : Yoyodyne, Inc.
```

Πληροφορίες Opendaylight project

Εικόνα 13

Μόλις εισαγάγουμε τις λεπτομέρειες για την εφαρμογή που θέλουμε να αναπτύξουμε στην Opendaylight πλατφόρμα τότε το αρχέτυπο "opendaylight-startup-archetype" θα δημιουργήσει έναν φάκελο με το όνομα του project δηλαδή στην προκειμένη περίπτωση το όνομα του φακέλου θα είναι "hello". Ο φάκελος αυτός θα περιέχει κάποιους υποφακέλους οι οποίοι φαίνονται στην εικόνα 14 και τους οποίους θα αναλύσουμε στην συνέχεια.

```
cd hello/
ls -1
api
artifacts
features
impl
karaf
pom.xml
```

Περιεχόμενα Hello project

Εικόνα 14

- *Api* : Στον φάκελο αυτόν ο δημιουργός της εφαρμογής μπορεί να κατασκευάσει και να αποθηκεύσει το YANG μοντέλο της εφαρμογής του το οποίο θα καθορίζει τα REST και JAVA APIs τα οποία θα χρησιμοποιηθούν για να καθοριστεί ο τρόπος λειτουργίας της νέας εφαρμογής.

- *Artifacts :* Σε αυτόν τον φάκελο αποθηκεύει ο Apache Maven τις εξαρτήσεις (dependencies) που προκύπτουν από τα διάφορα στοιχεία και δίνει εντολή να επιλυθούν αυτές οι εξαρτήσεις ώστε η λειτουργία της εφαρμογής να μην παρουσιάζει προβλήματα.

- *Features :* Στο επίπεδο του Apache Karaf container κάθε εφαρμογή ορίζεται ως ένα σετ από features (χαρακτηριστικά) τα οποία μπορούν να εγκατασταθούν στο karaf container. Άρα αυτός ο φάκελος περιέχει τους ορισμούς των features της νέας εφαρμογής που αναπτύσσουμε και επίσης περιέχει και τις εξαρτήσεις από άλλα features που πρέπει να εγκατασταθούν στο karaf container ώστε να λειτουργήσει σωστά η νέα εφαρμογή. Ένα παράδειγμα ενός feature αρχείου φαίνεται στην εικόνα 15.

```
<features name="odl-hello-${project.version}" xmlns="http://karaf.apache.org/xmlns/features/v1.2.0"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://karaf.apache.org/xmlns/features/v1.2.0
       http://karaf.apache.org/xmlns/features/v1.2.0">

   <repository>mvn:org.opendaylight.yangtools/features-yangtools/${yangtools.version}/xml/features</repository>
   <repository>mvn:org.opendaylight.controller/features-mdsal/${controller.mdsal.version}/xml/features</repository>

   <feature name='odl-hello-api' version='${project.version}' description='OpenDaylight :: hello :: API'>
       <feature version='${yangtools.version}'>odl-yangtools-common</feature>
       <feature version='${yangtools.version}'>odl-yangtools-binding</feature>
       <bundle>mvn:org.opendaylight.hello/api/${project.version}</bundle>
    </feature>

</features>
```

Παράδειγμα feature.xml

Εικόνα 15

- *Karaf :* Από αυτόν τον φάκελο ο χρήστης μπορεί να εκκινήσει το Karaf container για να φορτώσει την Opendaylight πλατφόρμα και όλες τις εφαρμογές που περιέχονται σε αυτήν όπως για παράδειγμα η νέα Hello εφαρμογή που αναφέρουμε στο παράδειγμά μας.

- *Impl :* Σε αυτόν τον φάκελο περιέχεται ο κώδικας της εφαρμογής που έχει αναπτυχθεί από τον δημιουργό της και καθορίζει τις λειτουργίες της εφαρμογής αυτής.

- Pom.xml : Σε αυτό το xml αρχείο περιέχονται όλα τα επιμέρους στοιχεία που αποτελούν και συνθέτουν το συνολικό project δηλαδή στο project που αναφέρουμε ως παράδειγμα θα περιέχονται το στοιχεία api,karaf,impl ,features και artifacts. Στην εικόνα 16 απεικονίζεται ένα παράδειγμα του Pom.xml αρχείου.

```xml
<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <groupId>org.opendaylight.hello</groupId>
  <artifactId>hello-aggregator</artifactId>
  <version>1.1.0-SNAPSHOT</version>
  <name>${project.artifactId}</name>
  <packaging>pom</packaging>
  <modelVersion>4.0.0</modelVersion>

<modules>
    <module>api</module>
    <module>impl</module>
    <module>karaf</module>
    <module>features</module>
    <module>artifacts</module>
  </modules>
</project>
```

Αρχικό Pom.xml

Εικόνα 16

Στην συνέχεια θα παρουσιάσουμε ένα παράδειγμα ενός Yang μοντέλου που θα αντιστοιχεί στο hello project που κατασκευάζουμε. Πιο συγκεκριμένα θα κατασκευάσουμε το hello.yang αρχείο το οποίο θα περιέχεται στην ακολουθία φακέλων *api/src/main/yang/hello.yang.* Στην συνέχεια θα παραθέσουμε μια εικόνα (εικόνα 17) που θα δείχνει την υλοποίηση αυτού του yang μοντέλου και πιο συγκεκριμένα θα κατασκευάσουμε μια RPC (Hello-World) διαδικασία που θα δέχεται ως είσοδο ένα όνομα και θα παράγει ως έξοδο έναν χαιρετισμό.

```
module hello {
    yang-version 1;
    namespace "urn:opendaylight:params:xml:ns:yang:hello";
    prefix "hello";

   rpc hello-world {
        input {
            leaf name {
                type string;
            }
        }
        output {
            leaf greeting {
                type string;
            }
        }
    }
}
```

hello.yang μοντέλο

Εικόνα 17

Στην εικόνα 18 απεικονίζεται η υλοποίηση του hello-world RPC Api οπου συγκεκριμένα θα υλοποιήσουμε το HelloService interface το οποίο δημιουργήθηκε λόγω του hello.yang αρχείου. Το interface αυτό έχει μία μέθοδο την οποία πρέπει να υλοποιήσουμε στην HelloWorldImpl java class και η μέθοδος αυτή είναι η helloWorld.

```
package org.opendaylight.hello.impl;

import java.util.concurrent.Future;

import org.opendaylight.yang.gen.v1.urn.opendaylight.params.xml.ns.yang.hello.rev150105.HelloService;
import org.opendaylight.yang.gen.v1.urn.opendaylight.params.xml.ns.yang.hello.rev150105.HelloWorldInput;
import org.opendaylight.yang.gen.v1.urn.opendaylight.params.xml.ns.yang.hello.rev150105.HelloWorldOutput;
import org.opendaylight.yang.gen.v1.urn.opendaylight.params.xml.ns.yang.hello.rev150105.HelloWorldOutputBuilder;
import org.opendaylight.yangtools.yang.common.RpcResult;
import org.opendaylight.yangtools.yang.common.RpcResultBuilder;

public class HelloWorldImpl implements HelloService {

    @Override
    public Future<RpcResult<HelloWorldOutput>> helloWorld(HelloWorldInput input) {
        HelloWorldOutputBuilder helloBuilder = new HelloWorldOutputBuilder();
        helloBuilder.setGreeting("Hello " + input.getName());
        return RpcResultBuilder.success(helloBuilder.build()).buildFuture();
    }

}
```

Rpc helloworld implementation

Εικόνα 18

Τέλος μπορούμε να δοκιμάσουμε την λειτουργία του καινούργιου hello-world RPC μέσω Rest οπού θα βάζουμε ως είσοδο το όνομα ενός χρήστη και το πρόγραμμα θα παράγει ως έξοδο το όνομα του χρήστη αυτού μαζί με τον χαιρετισμό "Hello".

5 Snort

Το Snort είναι μία open-source εφαρμογή που έχει ως βασική ευθύνη την αναγνώριση δικτυακών επιθέσεων. Δηλαδή το Snort χρησιμοποιείται κυρίως από τον εκάστοτε χρήστη ή την εκάστοτε επιχείρηση ως IDS (Intrusion Detection System) [21].

Πιο συγκεκριμένα το Snort είναι ένα packet sniffer το οποίο παρακολουθεί όλη την δικτυακή κίνηση που αφορά ένα συγκεκριμένο δίκτυο σε πραγματικό χρόνο. Μόλις αντιληφθεί κάποιο επικίνδυνο πακέτο δεδομένων που έχει προορισμό το δίκτυο που προστατεύει το Snort τότε ειδοποιεί τον διαχειριστή του δικτύου με ένα μήνυμα συναγερμού. Το Snort είναι βασισμένο σε μία "βιβλιοθήκη" που ονομάζεται libcap (library packet capture) η οποία αυτή βιβλιοθήκη είναι ένα εργαλείο που χρησιμοποιείται ευρέως σε εφαρμογές που λειτουργούν ως εξερευνητές και αναλυτές TCP/IP κίνησης. Επιπροσθέτως το Snort μέσα από μια διαδικασία ανάλυσης πρωτοκόλλου μπορεί να αντιληφθεί διαφόρων ειδών δικτυακές επιθέσεις όπως DOS (Denial of Service),DDOS (Distributed Denial of Service) ,PortScan κ.α.

Στην συνέχεια θα περιγράψουμε τις τρεις βασικές λειτουργίες τις οποίες μπορεί να εκμεταλλευτεί ο εκάστοτε χρήστης του Snort [22].

5.1 Sniffer Mode

Όταν το Snort βρίσκεται σε λειτουργία Sniffer τότε η βασική του ευθύνη είναι να παρακολουθεί τα πακέτα τα οποία εισέρχονται στο δίκτυο και στην συνέχεια να τα παρουσιάζει στην κονσόλα του χρήστη σαν συνεχόμενη ροή πληροφοριών. Βέβαια ο κάθε χρήστης μπορεί να επιλέξει πιο κομμάτι της πληροφορίας που ανήκει στα εισερχόμενα πακέτα και την οποία θα του παρουσιάζει το Snort θέλει να παρακολουθεί.

Εάν ο χρήστης επιλέξει να εκκινήσει το Snort με την εντολή *./snort -v* τότε το Snort θα εμφανίσει στην κονσόλα του χρήστη μόνο τις IP,TCP,UDP,ICMP επικεφαλίδες των πακέτων που εισέρχονται στο δίκτυο. Εάν ο χρήστης εκκινήσει το Snort με την εντολη *./snort -vd* τότε

θα εμφανιστούν μαζί με τις επικεφαλίδες και τα δεδομένα που περιέχονται στα εισερχόμενα πακέτα. Τέλος εάν η εκκίνηση του Snort γίνει με την εντολή *./snort -vde* τότε θα εμφανίζονται στον χρήστη μαζί με όλα τα παραπάνω και δεδομένα που αφορούν το επίπεδο διασύνδεσής των εισερχόμενων πακέτων.

5.2 Packet Logger Mode

Η Packet logger λειτουργία του Snort είναι η λειτουργία αυτή η οποία δίνει την δυνατότητα στον χρήστη να αποθηκεύει τα πακέτα δεδομένων που εισέρχονται στο δίκτυο που παρακολουθεί η εφαρμογή Snort. Για παράδειγμα εάν ένας διαχειριστής ενός δικτύου θελήσει να αποθηκεύσει τα δεδομένα που περιέχονται στα πακέτα που υπεισέρχονται στο δίκτυο θα πρέπει να δημιουργήσει έναν *log* φάκελο και στην συνέχεια μπορεί να δώσει εντολή στο Snort να αποθηκεύσει τα δεδομένα των εισερχόμενων πακέτων στο συγκεκριμένο φάκελο. Μια τέτοια εντολή μπορεί να είναι του τύπου *./snort -vde -l ./log.*

Σε περίπτωση που το Snort παρακολουθεί κάποιο δίκτυο το οποίο διαχειρίζεται μεγάλη ροή δεδομένων τότε ο διαχειριστής του δικτύου αυτού έχει την δυνατότητα να δώσει εντολή στο Snort να αποθηκεύσει τα δεδομένα των πακέτων σε δυαδική μορφή. Για να το πετύχει αυτό πρέπει να εκκινήσει το Snort με την εντολή *./snort -l ./log -b.* Αυτό που παρατηρούμε σε αυτήν την εντολή είναι ότι δεν χρειάστηκε να συμπεριλάβουμε τις παραμέτρους αυτές που θα πληροφορήσουν το Snort σχετικά με το ποια κομμάτια του πακέτου δεδομένων θα αποθηκεύσει. Αυτό συνέβη γιατί όταν το Snort λάβει εντολή να αποθηκεύσει τα δεδομένα σε δυαδική μορφή τότε δεν αποθηκεύει κομμάτια του πακέτου δεδομένων όπως επικεφαλίδες ή δεδομένα εφαρμογής αλλά αποθηκεύει όλο το πακέτο. Μόλις τα πακέτα αποθηκευτούν σε δυαδικής μορφής αρχείο τότε μπορούμε να τα διαβάσουμε από αυτό το αρχείο με ένα οποιοδήποτε πρόγραμμα που διαβάζει πακέτα που έχουν αποθηκευτεί σε δυαδική μορφή όπως το Tcpdump ή το Ethereal. Επίσης μπορούμε να διαβάσουμε αυτά τα δυαδικής μορφής αρχεία και με την εφαρμογή Snort χρησιμοποιώντας την εντολή *snort –r* και στην συνέχεια να παραθέσουμε στο τέλος της εντολής το όνομα του δυαδικής μορφής αρχείου. Τέλος εάν ο διαχειριστής του δικτύου θελήσει να ξανατρέξει ένα αρχείο δεδομένων που έχει αποθηκευτεί σε δυαδική μορφή και να εμφανίσει στην κονσόλα κάποια συγκεκριμένα κομμάτια των αποθηκευμένων πακέτων όπως για παράδειγμα TCP/IP επικεφαλίδες μπορεί να το επιτύχει εκτελώντας την εντολή *./snort -v –r binaryfile.* Με αυτήν την εντολή η εφαρμογή Snort ξανατρέχει το

αποθηκευμένο σε δυαδική μορφή αρχείο και εμφανίζει στην οθόνη μόνο την πληροφορία που σχετίζεται με τις IP/TCP επικεφαλίδες των αποθηκευμένων πακέτων.

5.3 Network Intrusion Detection System Mode

Όπως αναφέραμε και προηγουμένως το Snort είναι μια εφαρμογή που έχει ως κύρια λειτουργία την ανίχνευση κακόβουλης για το δίκτυο που προστατεύει κίνησης. Οπότε στις επόμενες γραμμές θα ασχοληθούμε με αυτήν την βασική λειτουργία του Snort.

Επειδή δεν είναι αποδοτικό να αποθηκεύουμε κάθε ένα πακέτο που εισέρχεται στο δίκτυο είναι καλό το Snort να εκκινεί σε λειτουργία ανίχνευσης πιθανής κακόβουλης κίνησης. Για να το επιτύχουμε αυτό πρέπει να εκτελέσουμε μια εντολή που είναι παρόμοια της ./snort -dev –l ./log -c snort.conf. Η εντολή αυτή είναι παρόμοια με τις εντολές που παραθέσαμε στις δύο προηγούμενες παραγράφους που αναλύσαμε τις άλλες λειτουργίες του Snort αυτό όμως που την διαφοροποιεί είναι η παράμετρος -c snort.conf. Όταν το Snort εκκινήσει και διαβάσει αυτήν την παράμετρο καταλαβαίνει πως δεν θα καταγράψει όλα τα πακέτα δεδομένων που εισέρχονται στο δίκτυο αλλά μόνο αυτά που έχουν σχέση με τους κανόνες που έχουν οριστεί και έχουν συμπεριληφθεί σε αυτό το snort.conf αρχείο. Δηλαδή το Snort εξετάζει κάθε ένα πακέτο δεδομένων και αναλόγως των κανόνων που έχουν συμπεριληφθεί στο snort.conf αρχείο αποφασίζει εάν θα αναλάβει κάποια δράση επί του πακέτου.

Υπάρχουν πολύ τρόποι για να καθορίσει ο διαχειριστής ενός δικτύου την μορφή των ειδοποιήσεων που εγείρει το Snort όταν αντιληφθεί κακόβουλη κίνηση δεδομένων προς το υπό προστασία δίκτυο. Η προεπιλογή του Snort σχετικά με την μορφή των ειδοποιήσεων που αποθηκεύει είναι η χρησιμοποίηση κωδικοποίησης ASCii και επίσης η χρησιμοποίηση λειτουργίας full alert που σημαίνει ότι το Snort θα εμφανίζει ένα alert μήνυμα για να ειδοποιήσει τον χρήστη για κακόβουλη κίνηση δεδομένων προς το δίκτυο και επίσης θα εμφανίζει και όλες τις επικεφαλίδες του πακέτου δεδομένων που ήταν η αιτία για την ενεργοποίηση των ειδοποιήσεων αυτών. Υπάρχουν 6 διαφορετικοί τύποι ειδοποιήσεων (alert modes) που μπορούν να χρησιμοποιηθούν κατά την εκκίνηση του Snort και οι οποίοι φαίνονται στο παρακάτω πίνακα (Πίνακας 1).

Option	Description
-A fast	Fast alert mode. Writes the alert in a simple format with a timestamp, alert message, source and destination IPs/ports.
-A full	Full alert mode. This is the default alert mode and will be used automatically if you do not specify a mode.
-A unsock	Sends alerts to a UNIX socket that another program can listen on.
-A none	Turns off alerting.
-A console	Sends "fast-style" alerts to the console (screen).
-A cmg	Generates "cmg style" alerts.

Πίνακας 1

Όταν το Snort παράγει ειδοποιήσεις αυτές θα μοιάζουν με αυτην της Εικόνας 19.

Το πρώτο νούμερο που παρατηρούμε στην παρακάτω εικόνα είναι το Generator ID το οποίο μας πληροφορεί σχετικά με το ποιο στοιχείο της εφαρμογής Snort παρήγαγε αυτήν την ειδοποίηση. Πιο συγκεκριμένα το νούμερο 116 μας καταδεικνύει πως το στοιχείο του Snort το οποίο παρήγαγε αυτήν την ειδοποίηση είναι το decode.

Το δεύτερο νούμερο στην εικόνα 19 ονομάζεται Snort ID και είναι το ID εκείνο το οποίο είναι γραμμένο σε κάθε κανόνα ξεχωριστά και μας πληροφορεί σχετικά με το προτόκολλο που αφορούσε η ειδοποίηση όπου στην συγκεκριμένη περίπτωση το νούμερο 56 αντιστοιχεί σε ένα γεγονός T/TCP.

Ο τρίτος αριθμός ονομάζεται revision ID και χρησιμοποιείται για να πληροφορήσει τον διαχειριστή του δικτύου εάν και πόσες φορές έχει γίνει ενημέρωση (update)στον κανόνα. Έτσι μπορεί εύκολα ο διαχειριστής να καταλάβει ποια εκδοχή του κανόνα παρήγαγε την παρακάτω ειδοποίηση. Στην συγκεκριμένη περίπτωση ο κανόνας έχει ενημερωθεί μία φορά.

```
[**] [116:56:1] (snort_decoder): T/TCP Detected [**]
```

Παράδειγμα περιεχομένου Alert αρχείου

Εικόνα 19

5.3.1 Snort κανόνες (Snort Rules)

Όπως αναφέραμε και στην προηγούμενη παράγραφο η βασικότερη χρήση της εφαρμογής Snort είναι όταν αυτή εκκινεί σε λειτουργία ανίχνευσης κακόβουλης ροής δεδομένων. Όταν το Snort βρίσκεται σε τέτοια λειτουργία το σημαντικότερο ρόλο στην διαδικασία ανίχνευσης τις εκάστοτε κακόβουλης κίνησης διαδραματίζουν οι Snort κανόνες.

Οι Snort κανόνες είναι αυτοί που αποφασίζουν εάν ένα πακέτο που εισέρχεται στο υπό παρακολούθηση από το Snort δίκτυο θα πρέπει να χαρακτηριστεί ως καλόβουλο ή κακόβουλο. Οι κανόνες αυτοί μπορούν να διαχωριστούν σε φακέλους εντός του snort συστήματος με βάση πιο σκοπό εξυπηρετούν. Για παράδειγμα οι κανόνες οι οποίοι ενεργοποιούνται όταν συμβαίνει κάποια DoS επίθεση αποθηκεύονται σε αρχείο με το όνομα DOS ενώ οι κανόνες οι οποίοι ενεργοποιούνται όταν συμβαίνει μία DDoS επίθεση αποθηκεύονται σε αρχείο με το όνομα DDOS. Η εφαρμογή Snort χρησιμοποιεί μια αρκετα ευέλικτη και σταθερή γλώσσα προγραμματισμού κανόνων η οποία συνάμα είναι και αρκετά απλή στην χρήση της. Επίσης παλαιότερες εκδόσεις της Snort εφαρμογής οι κανόνες έπρεπε να γράφονται σε μία και μόνο γραμμή έκαστος ενώ στις καινούργιες εκδόσεις της εφαρμογής Snort οι κανόνες μπορούν να καταλαμβάνουν και περισσότερές από μία γραμμές αρκεί στο τέλος κάθε γραμμής να παραχωρείται ο χαρακτήρας backslash "\".

Οι Snort κανόνες είναι διαχωρισμένοι σε δύο λογικά μέρη το πρώτο μέρος αναφέρεται στις επικεφαλίδες των κανόνων και το δεύτερο μέρος στις επιλογές των κανόνων . Η επικεφαλίδα ενός κανόνα περιέχει την δράση που θα εκτελεστεί αν έχουμε ενεργοποίηση του κανόνα , τα διάφορα πρωτόκολλα που παρακολουθούνται , τις διευθύνσεις πηγής και προορισμού που θα χρησιμοποιηθούν για να ξεχωρίσουμε τα υπό παρακολούθηση πακέτα δεδομένων , τις μάσκες δικτύου καθώς και πληροφορίες για τις θύρες από τίς οποίες εισήλθαν και εξήλθαν τα πακέτα αυτά. Το κομμάτι που αφορά τις επιλογές ενός κανόνα περιέχει μηνύματα ειδοποιήσεων καθώς και πληροφορίες σχετικά με το ποια κομμάτια του εκάστοτε πακέτου δεδομένων πρέπει να εξεταστούν ώστε να ληφθεί η απόφαση εάν πρέπει να ενεργοποιηθεί η δράση του κανόνα. Η εικόνα 20 παρουσιάζει ένα απλό δείγμα ενός Snort κανόνα.

```
alert tcp any any -> 192.168.1.0/24 111 \
    (content:"|00 01 86 a5|"; msg:"mountd access";)
```

Snort κανόνας

Εικόνα 20

5.3.2 Επικεφαλίδες κανόνων (Rule headers)

Η επικεφαλίδα ενός κανόνα περιέχει όλη την πληροφορία η οποία καθορίζει την προέλευση ,το προορισμό ,το πρωτόκολλο αλλά και την δράση που πρέπει να λάβει χώρα σε περίπτωση που κάποιο πακέτο δεδομένων ταυτιστεί με τα πεδία της επικεφαλίδας κάποιου κανόνα. Στην συνέχεια θα αναφέρουμε και θα αναλύσουμε τα πεδία εκείνα που συμπληρώνουν την επικεφαλίδα ενός κανόνα.

- **Δράση κανόνα :** Το πεδίο της επικεφαλίδας ενός κανόνα που αναφέρεται στην δράση είναι εκείνο που καταδεικνύει τις ενέργειες που πρέπει να γίνουν σε περίπτωση που κάποιο πακέτο δεδομένων πληρεί τα κριτήρια κάποιου κανόνα. Υπάρχουν πέντε διαφορετικές εκδοχές δράσεων στην Snort εφαρμογή τις οποίες αναφέρουμε παρακάτω:

1. *Alert* : Ενεργοποιεί μια ειδοποίηση και στην συνέχεια αποθηκεύει το πακέτο δεδομένων.
2. *Log* : αποθηκεύει το πακέτο δεδομένων.
3. *Pass : A*γνοεί το πακέτο δεδομένων.
4. *Activate* : Παράγει μια ειδοποίηση και στην συνέχεια ενεργοποιεί κάποιον άλλον δυναμικό κανόνα.
5. *Dynamic* : Ο κανόνας παραμένει ανενεργός μέχρι να ενεργοποιηθεί από κάποιο activate κανόνα. Μόλις ενεργοποιηθεί τότε αποθηκεύει το πακέτο δεδομένων.

- **Προτόκολλο :** Το επόμενο πεδίο στην επικεφαλίδα ενός κανόνα είναι το πεδίο πρωτοκόλλου. Η εφαρμογή Snort μπορεί και αναλύει τέσσερα διαφορετικά είδη πρωτοκόλλων σύμφωνα με τα οποία εξετάζει αν υπάρχει ύποπτη συμπεριφορά κάποιου πακέτου δεδομένων. Τα πρωτόκολλα αυτά είναι τα TCP,UDP,ICMP και IP.

- **Διευθύνσεις IP :** Το επόμενο πεδίο της επικεφαλίδας ενός κανόνα αφορά την διεύθυνση IP. Η λέξη κλειδί *any* μπορεί να χρησιμοποιηθεί για να καθορίσει με αυτόν τον τρόπο οποιαδήποτε IP διεύθυνση. Στην εφαρμογή Snort οι IP διευθύνσεις αποτυπώνονται με καθαρά αριθμητική μορφή η οποία μορφή αποτελείται από την

IP διεύθυνση και το CIDR πεδίο. Το CIDR πεδίο αποτελεί την μάσκα δικτύου και προστίθεται στην διεύθυνση IP του εκάστοτε κανόνα. Πιο συγκεκριμένα όταν το CIDR πεδίο έχει τιμή /24 τότε αναφερόμαστε σε δίκτυο τάξης C , όταν έχει τιμή /16 τότε αναφερόμαστε σε δίκτυο τάξης B και όταν έχει τιμή /32 τότε αναφερόμαστε σε ένα και μόνο υπολογιστή. Άρα ένας συνδυασμός διεύθυνσης IP/CIDR όπως για παράδειγμα η 192.168.1.0/24 καταδεικνύει ένα εύρος διευθύνσεων από την 192.168.1..1 έως την 192.168.1.255. Άρα οποιοσδήποτε κανόνας που περιέχει τον συνδυασμό του παραδείγματος ως διεύθυνση πηγής ή προορισμού θα ταυτοποιεί εισερχόμενα πακέτα δεδομένων που έχουν ως διεύθυνση πηγής ή προορισμού κάποια IP διεύθυνση από το προαναφερθέν εύρος διευθύνσεων. Επίσης όπως φαίνεται στην Εικόνα 21 υπάρχει ένας τελεστής άρνησης ο οποίος είναι το "!" και χρησιμοποιείται ώστε ο κανόνας να ταυτοποιεί πακέτα δεδομένων τα οποία έχουν οποιαδήποτε IP διεύθυνση εκτός από αυτήν που καταγράφεται μετά τον τελεστή άρνησης στο Snort κανόνα. Μία σημαντική συμβολή αυτού του τελεστή άρνησης είναι όταν ο διαχειριστής δικτύου θέλει να ενεργοποιούνται ειδοποιήσεις όταν εισέρχονται πακέτα δεδομένων τα οποία έχουν διεύθυνση πηγής κάποια διεύθυνση IP η οποία καταδεικνύει ότι το συγκεκριμένο πακέτο δεδομένων προήλθε από κάποιο δίκτυο διαφορετικό από το δίκτυο του διαχειριστή.

```
alert tcp !192.168.1.0/24 any -> 192.168.1.0/24 111 \
    (content:"|00 01 86 a5|"; msg:"external mountd access";)
```

Snort κανόνας

Εικόνα 21

- **Θύρες :** Οι αριθμοί θυρών μπορούν να καθοριστούν σε ένα Snort κανόνα με ποικίλους τρόπους οι οποίοι περιλαμβάνουν τον καθορισμό οποιαδήποτε θύρας με την λέξη κλειδί *any* ,τον καθορισμό στατικών θυρών τον καθορισμό θυρών με τον τελεστή άρνησης και θυρών οι οποίες καλύπτουν κάποιο εύρος. Οι στατικές θύρες ορίζονται με έναν και μόνο αριθμό όπως φαίνεται στην εικόνα 21 στο μέρος που ακολουθεί μετά την IP διεύθυνση προορισμού με το νούμερο 111. Επίσης μέσω του τελεστή εύρους ": " μπορούμε να ορίσουμε εύρος θυρών όπως για παράδειγμα αν ορίσουμε τις θύρες 1:2044 τότε ο κανόνας θα επιθεωρεί τα πακέτα δεδομένων που έχουν τίς σωστές διευθύνσεις IP και πρωτόκολλο και επίσης έχουν θύρες μεταξύ του αριθμού 1 και 2044.Επίσης μπορούμε να εξαιρέσουμε κάποια θύρα μέσω του τελεστή άρνησης όπως για παράδειγμα αν γράψουμε σε κάποιο κανόνα !24 εξαιρείται η θύρα 24.

- **Τελεστής κατεύθυνσης :** Ο τελεστής κατεύθυνσης που τον συμβολίζουμε με ένα βέλος δεξιάς κατεύθυνσης "->" καθορίζει την κατεύθυνση της κίνησης για την οποία είναι υπεύθυνος ένας κανόνας. Η διεύθυνση IP και η θύρα η οποία βρίσκεται στο αριστερό μέρος του τελεστή είναι η διεύθυνση πηγής του διερχόμενου πακέτου δεδομένων και η θύρα από την οποία εισήλθε. Ενώ η διεύθυνση IP και η θύρα η οποία βρίσκεται στο δεξιό μέρος του βέλους είναι διεύθυνση προορισμού και η θύρα από την οποία εξήλθε το πακέτο δεδομένων. Επίσης εκτός από τον τελεστή απλής κατεύθυνσης που προαναφέραμε υπάρχει και ο τελεστής διπλής κατεύθυνσης ο οποίος συμβολίζεται με διπλό βέλος "<>" .Ο τελεστής αυτός δίνει εντολή στην εφαρμογή Snort να θεωρεί τα ζεύγη διευθύνσεων και θυρών ως και διευθύνσεις πηγής αλλά και προορισμού.

5.3.3 Επιλογές Κανόνων (Rule Options)

Οι επιλογές κανόνων μορφοποιούν συνολικά την εφαρμογή Snort όταν αυτή λειτουργεί σε κατάσταση ανίχνευσης κακόβουλης κίνησης δεδομένων προικίζοντας την με ευρωστία και ελαστικότητα. Όλες οι επιλογές κανόνων διαχωρίζονται αναμεταξύ τους χρησιμοποιώντας το ελληνικό ερωτηματικό ";" και επίσης οι λέξεις κλειδιά των επιλογών κανόνων διαχωρίζονται από τίς παραμέτρους τους μέσω της άνω-κάτω τελείας ":". Στην συνέχεια θα αναλύσουμε τις ποιο σημαντικές επιλογές κανόνων της εφαρμογής Snort.

- **Msg :** Η επιλογή msg δίνει εντολή στην μηχανή αποθήκευσης και ειδοποιήσεων να τυπώσει μαζί με την ειδοποίηση και το μήνυμα που έχουμε παραθέσει στο πεδίο του msg. Το μήνυμα που έχει παραθέσει ο διαχειριστής στο πεδίο msg είναι ένα απλό μήνυμα κειμένου. Αρα η μορφή με την οποία θα το συναντήσουμε σε ένα Snort κανόνα είναι *msg : "μήνυμα κειμένου"*.

- **Sid :** Η λέξη κλειδί sid χρησιμοποιείται για να ταυτοποιήσει μοναδικά τους Snort κανόνες . Η πληροφορία αυτή μας βοηθάει να καταλάβουμε ποιος κανόνας ενεργοποίησε κάποια ειδοποίηση. Η λέξη κλειδί sid πρέπει να χρησιμοποιείται μαζί

με την λέξη κλειδί rev την οποία θα αναλύσουμε παρακάτω. Η μορφή της sid επιλογής σε έναν κανόνα είναι *sid : <id snort κανόνα>.*

- **Rev :** Η rev λέξη κλειδί χρησιμοποιείται σε έναν Snort κανόνα ώστε να πληροφορήσει τον διαχειριστή για το πόσες φορές έχει γίνει αναθεώρηση του κανόνα αυτού. Η rev λέξη κλειδί πρέπει να χρησιμοποιείται μαζί με την λέξη κλειδί sid. Η μορφή της rev σε έναν κανόνα είναι *rev : <αριθμός αναθεώρησης>.*

- **Gid :** Η gid λέξη κλειδί χρησιμοποιείται ώστε ό διαχειριστής του δικτύου να μπορεί να ταυτοποιήσει από πιο μέρος της εφαρμογής Snort ενεργοποιήθηκε κάποια ειδοποίηση. Παραδείγματος χάριν τα gid που έχουν τιμή μεγαλύτερη του 100 αναφέρονται σε τμήματα τις εφαρμογής Snort όπως οι προεπεξεργαστές και οι αποκωδικοποιητές. Επίσης πρέπει να σημειωθεί πως η gid λέξη κλειδί είναι προαιρετική και δεν χρειάζεται απαραίτητα σε έναν κανόνα αν όμως δεν έχει οριστεί τότε θα ανατεθεί από την εφαρμογή Snort η προεπιλεγμένη τιμή 1. Η μορφή της gid σε έναν κανόνα είναι *gid:<αριθμος>.*

Στην συνέχεια θα αναλύσουμε τις πληροφορίες που παρέχει το Snort ως έξοδο σχετικά με την κίνηση που έχει επιθεωρήσει μέχρι την στιγμή που σταματάει την λειτουργία του.

5.3.4 Πληροφορίες εξόδου Snort

Η εφαρμογή Snort εκτελεί κατά την διάρκεια λειτουργίας της αρκετές σημαντικές εργασίες για αυτό και όταν περατώσει την λειτουργία της παρέχει σαν έξοδο σημαντικές στατιστικές πληροφορίες σχετικά με την προηγούμενη περίοδο λειτουργία της. Στην συνέχεια θα αναφέρουμε τα βασικά αυτά στατιστικά δεδομένα που παρέχει σαν έξοδο η Snort εφαρμογή.

- **Χρονικά Στατιστικά (Timing Statistics)**

Αυτό το κομμάτι των πληροφοριών εξόδου παρέχει χρονικά στατιστικά. Κάποια από αυτά τα χρονικά στατιστικά είναι ο συνολικός χρόνος που καταναλώθηκε για επεξεργασία πακέτων δεδομένων από την Snort εφαρμογή όπως επίσης και ο μέσος όρος πακέτων που επεξεργάσθηκε η εφαρμογή ανά λεπτό και ανά δευτερόλεπτο. Μία τέτοια στατιστική έξοδος του Snort παρουσιάζεται στην εικόνα 22.

```
===============================================================================
Run time for packet processing was 175.856509 seconds
Snort processed 3716022 packets.
Snort ran for 0 days 0 hours 2 minutes 55 seconds
   Pkts/min:      1858011
   Pkts/sec:        21234
===============================================================================
```

Πληροφορίες εξόδου Snort

Εικόνα 22

- **Είσοδος/Εξοδος Πακέτων δεδομένων**

Στην εικόνα 23 παρατηρουμε τις στατιστικες πληροφορίες εξόδου του Snort σχετικά με το πόσα πακέτα λήφθησαν ,αναλύθηκαν,απορριφθηκαν,φιλτραρίστηκαν και περίμεναν στην αναμονή για να αναλυθουν.

```
===============================================================
Packet I/O Totals:
   Received:      3716022
   Analyzed:      3716022 (100.000%)


       Dropped:            0 (  0.000%)
      Filtered:            0 (  0.000%)
   Outstanding:            0 (  0.000%)
```

Πληροφορίες εξόδου Snort

Εικόνα 23

- **Στατιστικές πληροφορίες προτοκόλλων**

Ένα δείγμα πληροφορίας σχετικά με τα πρωτόκολλα της κίνησης που αποκωδικοποιήθηκε από το Snort απεικονίζεται στην εικόνα 24. Ο συνολικός αριθμός των πακέτων δεδομένων που αποκωδικοποιήθηκε από το Snort μπορεί να είναι μικρότερος από τον αριθμό των πακέτων που φαίνονται από τις στατιστικές πληροφορίες πρωτοκόλλων αυτό συμβαίνει γιατί είναι ενεργοποιημένοι οι προεπεξεργαστες frag3 και stream5 οι οποίοι εγχέουν ψευδό-πακέτα δεδομένων στο σύστημα του Snort.

```
===================================================================
Breakdown by protocol (includes rebuilt packets):
        Eth:      3722347 (100.000%)
       VLAN:            0 (  0.000%)
        IP4:      1782394 ( 47.884%)
       Frag:         3839 (  0.103%)
       ICMP:        38860 (  1.044%)
        UDP:       137162 (  3.685%)
        TCP:      1619621 ( 43.511%)
        IP6:      1781159 ( 47.850%)
```

Πληροφορίες εξόδου Snort

Εικόνα 24

- **Δράσεις και Ετυμηγορίες (Actions and Verdicts)**

Τα στατιστικά στοιχεία σχετικά με τις δράσεις και τις ετυμηγορίες της Snort εφαρμογής εμφανίζονται όταν το Snort βρίσκεται σε λειτουργία ανίχνευσης κακόβουλης κίνησης δηλαδή όταν εκκινεί με την παράμετρο *-c snort.conf*. Στην εικόνα 25 φαίνονται οι δράσεις , αλλά και οι ετυμηγορίες της εφαρμογής Snort που εμφανίζονται μετα την περάτωση της λειτουργίας της.

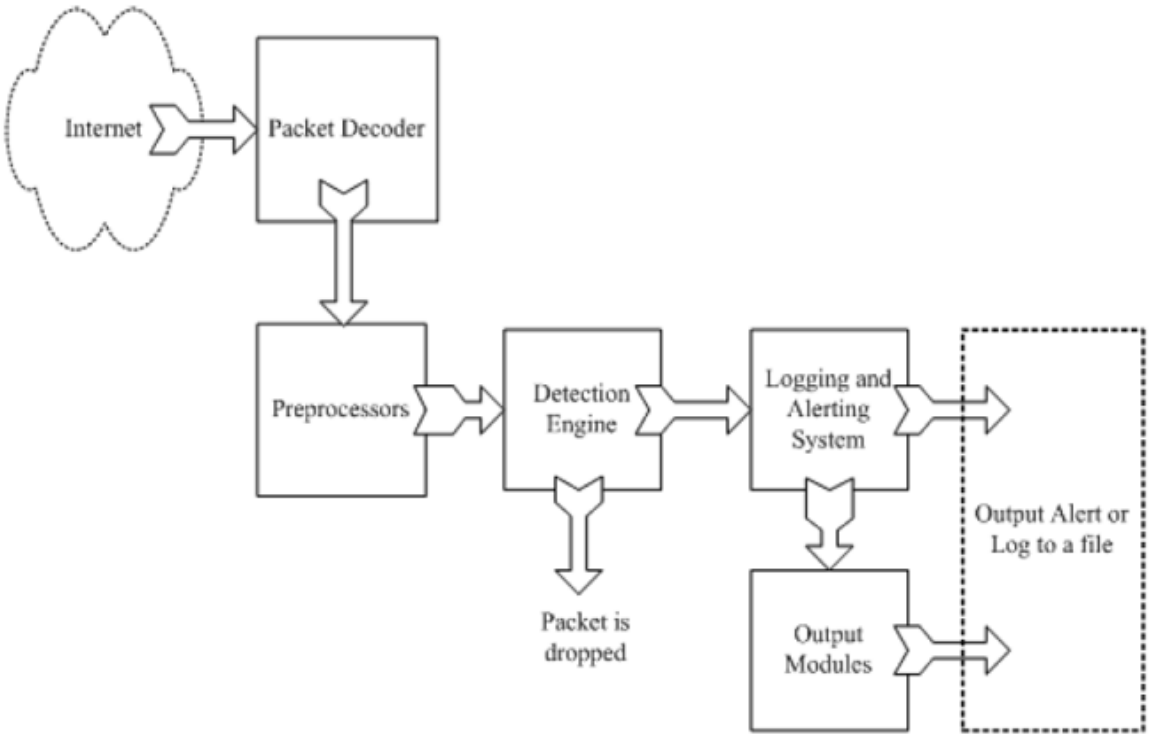
```
===========================================
Action Stats:
     Alerts:            0 (  0.000%)
     Logged:            0 (  0.000%)
     Passed:            0 (  0.000%)
Verdicts:
      Allow:      3716022 (100.000%)
      Block:            0 (  0.000%)
```

Πληροφορίες εξόδου Snort

Εικόνα 25

5.4 Αρχιτεκτονική Snort

Στην συνέχεια θα αναλύσουμε συνοπτικά τα στοιχεία που αποτελούν την ραχοκοκαλιά της εφαρμογής Snort και την μετατρέπουν σε μία εφαρμογή αποτελεσματική στην αντιμετώπιση δικτυακών επιθέσεων. Στην εικόνα 26 απεικονίζονται τα στοιχεία αυτά της αρχιτεκτονικής Snort.



Αρχιτεκτονική Snort

Εικόνα 26

- **Packet Decoder**

Ο Packet Decoder είναι το πρώτο στάδιο στην διαδικασία επεξεργασίας ενός πακέτου δεδομένων. Κύρια λειτουργία του Packet Decoder είναι να αναγνωρίσει ποια πρωτόκολλα

συνδέονται με το υφιστάμενο πακέτο δεδομένων και στην συνέχεια να αποθηκεύσει αυτά τα αναγνωρισθέντα δεδομένα μαζί με το payload του πακέτου (το οποίο δεν αποκωδικοποιεί) ώστε αυτά να μπορούν να χρησιμοποιηθούν για περαιτέρω επεξεργασία από τους προεπεξεργαστες (preprocessors) και την μηχανή ανίχνευσης (detection engine). Επίσης όσο ο Packet Decoder αναλύει τα εισερχόμενα πακέτα παράλληλα ψάχνει για ανωμαλίες στις επικεφαλίδες των πακέτων όπως για παράδειγμα ανωμαλίες στο μέγεθος.

- **Preprocessors (Προεπεξεργαστές)**

Οι προεπεξεργαστές είναι το στοιχείο αυτό στην ραχοκοκαλιά της εφαρμογής Snort που ουσιαστικά προετοιμάζει τα εισερχόμενα πακέτα για να εισαχθούν σε επόμενο στάδιο στην μηχανή ανίχνευσής (detection engine). Ένα από τους σημαντικότερους προεπεξεργαστές που χρησιμοποιούμε και στα πλαίσια αυτής της διπλωματικής είναι ο προεπεξεργαστής sfPortscan.O sfPortscan προεπεξεργαστής έχει σχεδιασθεί για να αναγνωρίζει την πρώτη φάση μιας δικτυακής επίθεσης η οποία πρώτη φάση είναι η διαδικασία που ακολουθεί ο επιτιθέμενος για να καταλάβει ποια πρωτόκολλα ή ποιες υπηρεσίες υποστηρίζει ο δέκτης της επίθεσης. Δηλαδή αυτός ο προεπεξεργαστής αναγνωρίζει μια κλασική δικτυακή επίθεση portscan.

- **Μηχανή Ανίχνευσης (detection engine)**

Η μηχανή ανίχνευσης είναι το ποιο σημαντικό στοιχείο και ο πυρήνας της εφαρμογής Snort καθώς είναι εκείνο το στοιχείο το οποίο θα αναγνωρίσει μια δικτυακή αθέμιτη κίνηση σύμφωνα με τον διαχειριστή του δικτύου. Όταν καταφτάνει ένα πακέτο δεδομένων στην μηχανή ανίχνευσης τότε το πακέτο αυτό συγκρίνεται με τις πληροφορίες που υπάρχουν στους κανόνες που έχουμε ορίσει και αν τα δεδομένα του πακέτου ταιριάζουν με τα δεδομένα κάποιου κανόνα τότε ενεργοποιείται η δράση που είναι ορισμένη σε αυτόν τον κανόνα.

- **Σύστημα καταγραφής και ειδοποιήσεων**

Στο στάδιο αυτό λαμβάνεται η απόφαση για το αν η καταγραφή των δεδομένων και η ενεργοποίηση των ειδοποιήσεων θα εκτελεστεί βάση κάποιου συγκεκριμένου τρόπου που έχει επιλέξει ο διαχειριστής του δικτύου ή θα ακολουθηθεί ο προεπιλεγμένος τρόπος. Αν επιλεγεί κάποιος συγκεκριμένος τρόπος τότε ενεργοποιείται το επόμενο στάδιο στην αλυσίδα του Snort που είναι το Output Module.

- **Output Module**

Η βασική λειτουργία του Output Module είναι ότι παρέχει μεγαλύτερη ευελιξία στην μορφή και στην παρουσίαση των δεδομένων που παράγονται από την εφαρμογή Snort. Στην συνέχεια καταγράφονται οι ειδοποιήσεις και αποθηκεύονται τα δεδομένα για περαιτέρω επεξεργασία από τον διαχειριστή του δικτύου ή για εισαγωγή τους σε κάποια άλλη εφαρμογή ως δεδομένα εισόδου.

6 Ασφάλεια και προστασία σε ευφυή δίκτυα (SDN security)

Ο τομέας των ευφυών δικτύων είναι ένα από τους πιο ενδιαφέροντες και ταχύτατα αναπτυσσόμενους τομείς που αναδύθηκαν παγκοσμίως την τελευταία δεκαετία. Η προοπτική της εγκαθίδρυσης μίας απλοποιημένης εγκατάστασης και η δυνατότητα παρέμβασης μέσω λογισμικού ώστε να μπορείς δυναμικά να παραμετροποιείς τα χαρακτηριστικά των κανόνων (SDN rules) ενός SDN συστήματος δημιουργεί σκεπτικισμό απέναντι στα συνεχώς αυξανόμενα κόστη του δικτυακού υλικού (hardware costs) και επίσης ανοίγει νέους ορίζοντες σχετικά με την ταχύτερη παροχή υπηρεσιών και σχετικά με την βελτιστοποίηση του ελέγχου μεταξύ διαφόρων δικτύων. Είναι αδιαμφισβήτητο γεγονός πως οι τεχνικές που εφαρμόζονται στα παρόντα μη-ευφυή δίκτυα σχετικά με την ασφάλεια των δικτύων αυτών είναι προβληματικές καθώς οι σημερινές λύσεις ασφάλειας σε αυτά τα παρόντα δίκτυα είναι δύσκολο να αναπτυχθούν περαιτέρω, να κλιμακωθούν, και να προγραμματισθούν. Αυτό συμβαίνει γιατί οι πολιτικές που ακολουθούνται στα μη ευφυή δίκτυα είναι άρρηκτα συνδεδεμένες με το υλικό (Hardware) αυτών των δικτύων και με τους διαθέσιμους πόρους αυτού. Οι τεχνικές ασφάλειας δικτύων που εφαρμόζονται σήμερα αντιμετωπίζουν δυσκολίες όσον αφορά τη γρήγορη και αυτόματη καταστολή μιας ενδεχόμενης για το δίκτυο απειλής και επίσης συναντούν εμπόδια όσον αφορά την χορήγηση αυτών των τεχνικών ασφαλείας σε διάφορα στοιχεία του δικτύου όπως σε πολλαπλά κέντρα αποθήκευσης δεδομένων (data centers).

Αυτό που τα ευφυή δίκτυα παρέχουν για την επίλυση των παραπάνω ζητημάτων όσον αφορά την ασφάλεια δικτύων είναι ότι προσθέτουν μια κεντρικοποιημένη ευφυία και έναν μοντέλο ελέγχου που είναι άριστα σχεδιασμένο για να προικίσουν το εκάστοτε δίκτυο με την απαραίτητη ευελιξία που βοηθά στην κατεύθυνση της ανάπτυξης νέων τεχνικών ασφαλείας και που επίσης συνοδεύεται από έναν αριθμό συμπληρωματικών γνωρισμάτων τα οποία είναι χρήσιμα για την κατασκευή ενός συστήματος που θα παρέχει υψηλών προσδοκιών ασφάλεια και επίσης θα είναι ευέλικτο όσον αφορά στην διαχείρισή του και στον προγραμματισμό του.

Στην συνέχεια θα αναφερθούμε στα πλεονεκτήματα που διατρέχουν τα ευφυή δίκτυα όσον αφορά το κομμάτι της ασφάλειας δικτύων και τα οποία πλεονεκτήματα τα διαφοροποιούν από τα παρόντα μη ευφυή δίκτυα.

1. **Καταπολέμηση απειλών μέσω παραμετροποίησης κανόνων.**

Το μοντέλο περιμετρικής ασφάλειας που κυριαρχεί μέχρι και σήμερα στα δίκτυα και το οποίο πρεσβεύει την αρχή "διατήρηση των απειλών εκτός δικτύου" δεν λειτουργεί αποδοτικά όσον αφορά καινούργιες τεχνολογίες όπως για παράδειγμα λειτουργίες δυναμικών cloud τεχνολογιών. Επίσης τα υπάρχοντα δίκτυα και οι εξυπηρετητές (servers) συνεχίζουν να είναι θύματα δικτυακών επιθέσεων παρά το γεγονός ότι χρησιμοποιούνται τεχνολογίες για ασφάλεια δικτύων οι οποίες είναι υψηλών προδιαγραφών και κόστους. Τεχνολογίες όπως ανάλυση περιεχομένου και αδιάκοπης παρακολούθησης (monitoring) ενός δικτύου. Η νέα οπτική στην ασφάλεια δικτύων μέσω της παραμετροποίησης των κανόνων σε ευφυή δίκτυα είναι ιδανική καθώς δεν περιορίζεται από τις παραδοσιακές τεχνολογίες που υπάρχουν μέχρι και σήμερα.

2. **Κεντρικοποιημένος έλεγχος δικτύου.**

Ο λογικά κεντρικοποιημένος έλεγχος επιτρέπει την αποδοτική λειτουργία και αποτελεσματική παρακολούθηση απειλών σε όλο το δίκτυο. Επίσης στην αρχιτεκτονική ευφυών δικτύων η ευφυία όλου του δικτύου είναι κεντρικοποιημένη με αποτέλεσμα οι αποφάσεις να λαμβάνονται με γνώμονα την συνολική εικόνα του δικτύου σε αντίθεση με τις τεχνικές που ακολουθούνται στα σημερινά μη-ευφυή δίκτυα τα οποία έχουν κατασκευαστεί πάνω σε μία λογική αυτονομίας όπου το κάθε στοιχείο του δικτύου καλείται να λάβει αποφάσεις χωρίς να γνωρίζει την συνολική κατάσταση του δικτύου. Έτσι μέσω των ευφυών δικτύων ο διαχειριστής μπορεί να αναγνωρίσει και να καταπολεμήσει πιο αποτελεσματικά μια πιθανή απειλή.

3. **Ικανότητα προγραμματισμού.**

Η ικανότητα προγραμματισμού που προσφέρουν τα ευφυή δίκτυα εγγυάται την παροχή ασφάλειας σε δίκτυα η οποία είναι πολύ υψηλότερων προδιαγραφών από αυτές που υπάρχουν μέχρι σήμερα. Επίσης η δυνατότητα προγραμματισμού αυτή προσφέρει αυτοματισμό και προσαρμογή όσον αφορά την αντιμετώπιση απειλών καθώς επιτρέπει την ύπαρξη μεγάλης ευελιξίας και επανακαθορισμού σε κάθε χρονική στιγμή των πολιτικών ασφαλείας που ακολουθούνται.

Έτσι συνδυάζοντας πληροφορίες σχετικά με την ιστορική αλλά και την παρούσα κατάσταση ενός ευφυούς δικτύου όπως επίσης και δεδομένα σχετικά με την επίδοση του δικτύου αυτού διευκολυνόμαστε σημαντικά στο κομμάτι της λήψης αποφάσεων. Επίσης μέσω των ευφυών δικτύων μπορούμε να επιτύχουμε μεγαλύτερο βαθμό ευελιξίας, λειτουργική

απλοποίηση όπως και βελτιωμένη παροχή ασφάλειας κατά μήκος μιας κοινής υποδομής δικτύου. Ακόμα μέσω μιας σωστής παραμετρικοποίησης μπορούμε να ασφαλίσουμε ικανοποιητικά το περιβάλλον στο οποίο λειτουργούν τα ευφυή δίκτυα χωρίς να χρειαστεί να μεταλλάξουμε την μορφή των δικτύων αυτών και να παρέμβουμε στις ικανότητες τους. Τέλος μέσω της ανάπτυξης της τεχνολογίας των ευφυών δικτύων έχει διευκολυνθεί η διαχείριση και η εφαρμογή πολιτικών ασφαλείας καθώς η ασφάλεια διαδραματίζει ένα σημαντικό ρόλο στον σχεδιασμό ενός ευφυούς δικτύου.

6.1 Μέθοδοι προστασίας ευφυών δικτύων

Η ασφάλεια των ευφυών δικτύων είναι ένα δύσκολο πρόβλημα για τους μηχανικούς δικτύων καθώς πρέπει να εγγυηθούν την σωστή και αδιάκοπη λειτουργία του δικτύου και επίσης να το καταστήσουν απρόσβλητο απέναντι στις πιθανές απειλές που μπορούν να προκύψουν οι οποίες έχουν ως στόχο να παρέμβουν στην λειτουργία αυτού και να το θέσουν εκτός λειτουργίας με σκοπό να μην μπορεί να εξυπηρετήσει τους καλόβουλους χρήστες.

Λόγο της ιδιαίτερης αρχιτεκτονικής που διακρίνει τα ευφυή δίκτυα οι μηχανικοί δικτύου έχουν επινοήσει τρόπους μέσω των οποίων μπορούν να προστατέψουν αποτελεσματικά ένα ευφυές δίκτυο και να το καταστήσουν αξιόπιστο σε μεγάλο βαθμό. Μερικούς από αυτούς τους τρόπους θα αναφέρουμε στις επόμενες γραμμές αυτής της διπλωματικής εργασίας.

Ένα από τους τρόπους μέσω του οποίου μπορεί κάποιος να προστατεύσει επιτυχημένα ένα ευφυές δίκτυο είναι η προστασία της μονάδας ελέγχου του ευφυούς αυτού δικτύου (SDN controller).Πιο συγκεκριμένα η μονάδα ελέγχου ευφυών δικτύων είναι ένα κρίσιμο στοιχείο για το εκάστοτε δίκτυο καθώς η μονάδα αυτή εγγυάται την σωστή λειτουργία του δικτύου. Όλες οι κρίσιμες αποφάσεις για το δίκτυο λαμβάνονται από την μονάδα έλεγχου και όλες οι απαραίτητες ενημερώσεις και τροποποιήσεις γίνονται μέσω αυτής. Άρα η πρόσβαση στην μονάδα αυτή πρέπει να είναι ελεγχόμενη και επίσης πρέπει να έχουν πρόσβαση στην μονάδα χρήστες οι οποίοι είναι καταρτισμένοι. Ακόμα πρέπει η μονάδα ελέγχου αυτή να προστατεύετε πλήρως και να μην εκτίθεται σε απειλές που μπορούν να προκύψουν και έχουν ως στόχο να την πλήξουν. Η μονάδα ελέγχου είναι ουσιαστικά ο πυρήνας του εκάστοτε ευφυούς δικτύου και πρέπει να είναι απροσπέλαστη όσων αφορά κακόβουλους χρήστες. Για παράδειγμα εάν η μονάδα ελέγχου πέσει θύμα μιας DDoS επίθεσης τότε θα σταματήσει να παρέχει τις υπηρεσίες της στο δίκτυο και με την σειρά του το δίκτυο αυτό δεν θα μπορεί αν εξυπηρετήσει τους συνδεμένους σε αυτό χρήστες.

Ένας άλλος τρόπος προστασίας ενός ευφυούς δικτύου είναι η εγκαθίδρυση εμπιστοσύνης σε όλα τα μήκη και πλάτη του δικτύου αυτού. Συγκεκριμένα η απαραίτητη εμπιστοσύνη σχετικά με την ορθότητα που αποπνέουν οι εφαρμογές , οι συνδέσεις και οι συσκευές που αποτελούν το δίκτυο αυτό είναι κρίσιμο κομμάτι για την απρόσκοπτη λειτουργία ενός ευφυούς δικτύου.

Τέλος ένας ακόμη τρόπος για την προστασία ενός ευφυούς δικτύου είναι η άμεση αναγνώριση κάποιας απειλής και η γρήγορη αντίδραση του συστήματος ώστε να την καταστείλει. Ένα κρίσιμο σημείο για την επιτυχή προστασία ενός ευφυούς δικτύου είναι ότι σε περίπτωση που κάποιος κακόβουλος χρήστης καταφέρει να μολύνει το δίκτυο τότε το ίδιο το δίκτυο πρέπει να έχει την δυνατότητα να αναγνωρίσει την κακόβουλη κίνηση αυτή, να την διαχωρίσει από την καλόβουλη κίνηση , να αναγνωρίσει τον κακόβουλο χρήστη και τέλος να τον αποκλείσει από το δίκτυο μέχρι νεοτέρας. Έτσι ένα ευφυές δίκτυο εγγυάται την δυναμική αντιμετώπιση απειλών που μπορεί να προκύψουν.

6.1.1 Σχετικές Εργασίες (Related Work)

Στην συνέχεια θα αναφερθούμε σε σχετικές εργασίες που έχουν δημοσιευτεί στο παρελθόν και μέσω των οποίων έχουν αναπτυχθεί εφαρμογές που έχουν οδηγήσει σε πιο ασφαλή και προστατευμένα ευφυή δίκτυα.

- FortNOX security enforcement Kernel

Το FortNOX είναι ουσιαστικά μία επέκταση της NOX μονάδας ελέγχου το οποίο λειτουργεί σαν μια διαμεσολαβητική μη - παρακαμπτική υπηρεσία η οποία εκτελεί λεπτομερή εξέταση των νέων Openflow κανόνων που εισάγονται από εφαρμογές Openflow ώστε οι κανόνες αυτοί να μην έρχονται σε σύγκρουση με κανόνες που εκτελούν χρέη προστασίας του δικτύου. Πιο συγκεκριμένα ο διαχειριστής ενός δικτύου ή εφαρμογές ασφαλείας του δικτύου εισάγουν Openflow κανόνες ώστε να προικίσουν το εκάστοτε δίκτυο με πολιτικές ασφαλείας οι οποίες καθιστούν το δίκτυο προστατευμένο. Το FortNOX είναι επιφορτισμένο να παρακολουθεί τους Openflow κανόνες οι οποίοι εισάγονται από άλλες ανεξάρτητες εφαρμογές και να αποτρέπει την σύγκρουση αυτών των κανόνων με τους κανόνες που έχουν εισαχθεί από τον διαχειριστή του δικτύου ή από τις εφαρμογές ασφαλείας του δικτύου. Με τον τρόπο αυτό το FortNOX καταφέρνει να μην έχουμε αλλαγή στις πολιτικές ασφαλείας του δικτύου λόγο κανόνων που έχουν εισαγάγει άλλες ανεξάρτητες εφαρμογές και όχι ο διαχειριστής του δικτύου ή κάποια εφαρμογή ασφαλείας του δικτύου αυτού [23].

- FRESCO

Το FRESCO είναι ουσιαστικά ένα πλαίσιο μέσα στο οποίο ο εκάστοτε προγραμματιστής μπορεί να αναπτύξει εφαρμογές οι οποίες θα εκτελούν υπηρεσίες οι οποίες θα καταστούν ένα δίκτυο ασφαλές και θα καταπολεμούν οποιαδήποτε πιθανή απειλή. Πιο συγκεκριμένα το FRESCO παρέχει ένα περιβάλλον που βοηθάει στην ανάπτυξη εφαρμογών ασφαλείας δικτύου. Επίσης αναλαμβάνει την παροχή πόρων του δικτύου που είναι απαραίτητοι για την λειτουργία των εφαρμογών ασφαλείας αυτών και ακόμα συμβάλλει στην εισαγωγή πολιτικών ασφαλείας στο δίκτυο οι οποίες έχουν παραχθεί από τις εφαρμογές ασφαλείας που αναπτύχθηκαν στα πλαίσια του FRESCO. Τέλος με την βοήθεια του FRESCO ο εκάστοτε προγραμματιστής μπορεί να αναπτύξει νέους τρόπους αντιμετώπισης δικτυακών επιθέσεων και να τους εφαρμόσει εύκολα και δυναμικά [24].

- Combining OpenFlow and sFlow for an effective and scalable anomaly detection and mitigation mechanism on SDN environments

Ο παραπάνω τίτλος αναφέρεται σε μια δημοσιευμένη εργασία από το εργαστήριο NETMODE στην σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Η/Υ του Ε.Μ.Π και η οποία πραγματεύεται μία προσέγγιση αντιμετώπισης δικτυακών επιθέσεων στα πλαίσια των ευφυών δικτύων. Πιο συγκεκριμένα μέσω ενός τρόπου συλλογής στατιστικών στοιχείων που αφορούν το δίκτυο μπορούμε να αναγνωρίσουμε κάποια πιθανή κακόβουλη κίνηση δεδομένων που απειλεί την ασφάλεια του δικτύου και στην συνέχεια μέσω παραμετροποίησης των Openflow κανόνων που υπάρχουν σε μια συσκευή δικτύου Openflow να μπορέσουμε να καταστείλουμε την απειλή αυτή και να καταστίσουμε το δίκτυο μας ασφαλές και αξιόπιστο. Πιο συγκεκριμένα μέσω της συλλογής δικτυακών στατιστικών δεδομένων μπορούμε να παρατηρήσουμε μια πιθανή απόκλιση από στην συνήθη κίνηση δεδομένων του δικτύου και στην συνέχεια είμαστε σε θέση να αναγνωρίσουμε την διαφαινόμενη απειλή και να την καταπολεμήσουμε εισάγοντας Openflow κανόνες που αποκόπτουν την απειλή αυτή η οποία έχει ως σκοπό να μολύνει το δίκτυο με κακόβουλη κίνηση [25].

6.2 Ενσωμάτωση συστημάτων ανίχνευσης εισβολών σε ευφυή δίκτυα

Ένα οποιοδήποτε ευφυές δίκτυο για να είναι σε θέση να μπορεί να αναγνωρίσει κάποια πιθανή κακόβουλη ροή δεδομένων και στην συνέχεια να την καταστείλει πρέπει να συνεργάζεται με κάποιο σύστημα το οποίο ανιχνεύει την κακόβουλη κίνηση αυτή και είτε το ίδιο σύστημα στην συνέχεια είναι υπεύθυνο να την καταστείλει είτε δίνει εντολή σε ένα δεύτερο σύστημα να την καταστείλει διατηρώντας το δίκτυο απρόσβλητο και αξιόπιστο.

Τέτοια συστήματα τα οποία είτε αναγνωρίζουν είτε καταστέλλουν είτε αναγνωρίζουν και καταστέλλουν κακόβουλη κίνηση δεδομένων είναι απαραίτητα για ένα οποιοδήποτε δίκτυο και είναι διαθέσιμα στο εμπόριο είτε έναντι πληρωμής είτε δωρεάν.

Τα συστήματα αυτά χρησιμοποιούν διάφορους τρόπους για ανίχνευση της κακόβουλης κίνησης. Κάποια από αυτά χρησιμοποιούν αναγνώριση συγκεκριμένων πρωτοκόλλων ή ακολουθούν εντολές που έχουν λάβει από το διαχειριστή του δικτύου σχετικά με το φιλτράρισμα συγκεκριμένων διευθύνσεων οι οποίες θεωρούνται απειλητικές για το δίκτυο. Άλλα συστήματα αναγνωρίζουν την κακόβουλη κίνηση χρησιμοποιώντας στατιστικά δεδομένα τα οποία είναι τέτοια που απεικονίζουν την κίνηση δεδομένων του δικτύου σε συνθήκες ομαλής λειτουργίας. Έτσι όταν υπάρξει κάποια κακόβουλη κίνηση δεδομένων τα στατιστικά δεδομένα αυτά παρεκκλίνουν από τις συνήθεις τιμές και έτσι αναγνωρίζεται η πιθανή απειλή. Για να είναι σε θέση αυτά τα συστήματα να αναγνωρίσουν κακόβουλη πληροφορία σε ένα ευφυές δίκτυο πρέπει η μονάδα ελέγχου να έχει ορίσει στην συσκευή δικτύου κανόνες οι οποίοι οδηγούν όλη την κίνηση του δικτύου στο μηχάνημα που είναι υπεύθυνο για την αναγνώριση απειλών.

Επίσης υπάρχουν και τα συστήματα τα οποία είναι υπεύθυνα για την καταστολή της κακόβουλης πληροφορίας και τα οποία λειτουργούν με γνώμονα την αδιάκοπη λειτουργία του δικτύου και την συνεχή εξυπηρέτηση των καλόβουλων χρηστών που είναι συνδεμένοι σε αυτό. Τα συστήματα αυτά αναλαμβάνουν να διαχωρίσουν την προσφάτως ανιχνευθείσα κακόβουλη πληροφορία από την καλόβουλη πληροφορία και στην συνέχεια είναι υπεύθυνα να απορρίψουν την κακόβουλη πληροφορία ενώ παράλληλα πρέπει να μπορούν να οδηγήσουν την καλόβουλη πληροφορία στον αρχικό της προορισμό.

Στην επόμενη παράγραφο θα αναφέρουμε επιγραμματικά τα πιο σημαντικά συστήματα ανίχνευσης και καταστολής κακόβουλης πληροφορίας και θα αναλύσουμε τις ιδιότητες τους.

6.2.1 Διαδεδομένα συστήματα ανίχνευσης εισβολών

Όπως αναφέραμε και προηγουμένως υπάρχει ένας μεγάλος αριθμός προϊόντων ανίχνευσης και καταστολής κακόβουλης ροής δεδομένων. Στην συγκεκριμένη παράγραφό θα αναφέρουμε, εκτός του Snort που αναλύσαμε σε προηγούμενη παράγραφο, τέτοια προϊόντα που μας βοηθάνε να αναγνωρίσουμε και να καταπολεμήσουμε δικτυακές απειλές.

- Suricata

Η Suricata είναι μια εφαρμογή η οποία χρησιμοποιώντας κανόνες αλλά και λειτουργία ανίχνευσης ανωμαλιών στα δεδομένα του συστήματος καταφέρνει να ανιχνεύσει πιθανές απειλές στα δεδομένα που κυκλοφορούν στο εσωτερικό του δικτύου. Η εφαρμογή αυτή ανίχνευσης κακόβουλης πληροφορίας υποστηρίζει λειτουργία πολλών νημάτων (multi-threading) και επίσης βοηθάει στην επιτάχυνση της μονάδας επεξεργασίας γραφικών [26].

- BRO IDS

Η εφαρμογή Bro IDS είναι μια εφαρμογή ανίχνευσης κακόβουλης πληροφορίας η οποία βασίζεται στην αναγνώριση ανωμαλιών που προκύπτουν όταν ένα ευφυές δίκτυο δέχεται επίθεση από κάποιον κακόβουλο χρήστη. Η εφαρμογή αυτή έχει σχεδιαστεί για να παρακολουθεί και να επιβάλει πολιτικές ώστε να προστατεύει ιστοσελίδες και συγκεκριμένα σημεία του κυβερνοχώρου και επίσης χαρακτηρίζεται από μεγάλη αποδοτικότητα όταν καλείται να διαχειριστεί και να προστατεύσει μεγάλης έκτασης δίκτυα. Επίσης παρέχει υπηρεσίες αποθήκευσης δεδομένων τα οποία σχετίζονται με πληροφορίες σχετικά με την κίνηση του δικτύου και επίσης παρέχει υπηρεσίες ανάλυσης μεγάλου εύρους πρωτοκόλλων σε όλα τα επίπεδα του δικτύου [27].

- OpenWIPS-ng

Η εφαρμογή αυτή είναι σχεδιασμένη ώστε να προσφέρει στο δίκτυο ασύρματη ασφάλεια (wireless security). Πιο συγκεκριμένα είναι μια εφαρμογή η οποία καταστέλλει δικτυακές επιθέσεις όπου αισθητήρες συλλέγουν ασύρματα δεδομένα του δικτύου που προστατεύουν και στην συνέχεια τα στέλνουν σε έναν εξυπηρετητή (server) για ανάλυση. Ο

εξυπηρετητής αυτός συγκεντρώνει όλα αυτά τα δεδομένα και στην συνέχεια τα αναλύει και αντιδρά σε πιθανές επιθέσεις. Επίσης παρέχει υπηρεσίες ειδοποίησης σε περίπτωση απειλής και υπηρεσίες αποθήκευσης πληροφοριών σχετικά με τις απειλές που προέκυψαν. Τέλος παρέχει στον χρήστη μια γραφική διεπαφή (GUI) η οποία επιτρέπει την διαχείριση του εξυπηρετητή και την παρακολούθηση πληροφοριών σχετικά με τις απειλές που προέκυψαν [28].

6.2.2 Αρχιτεκτονική συστήματος

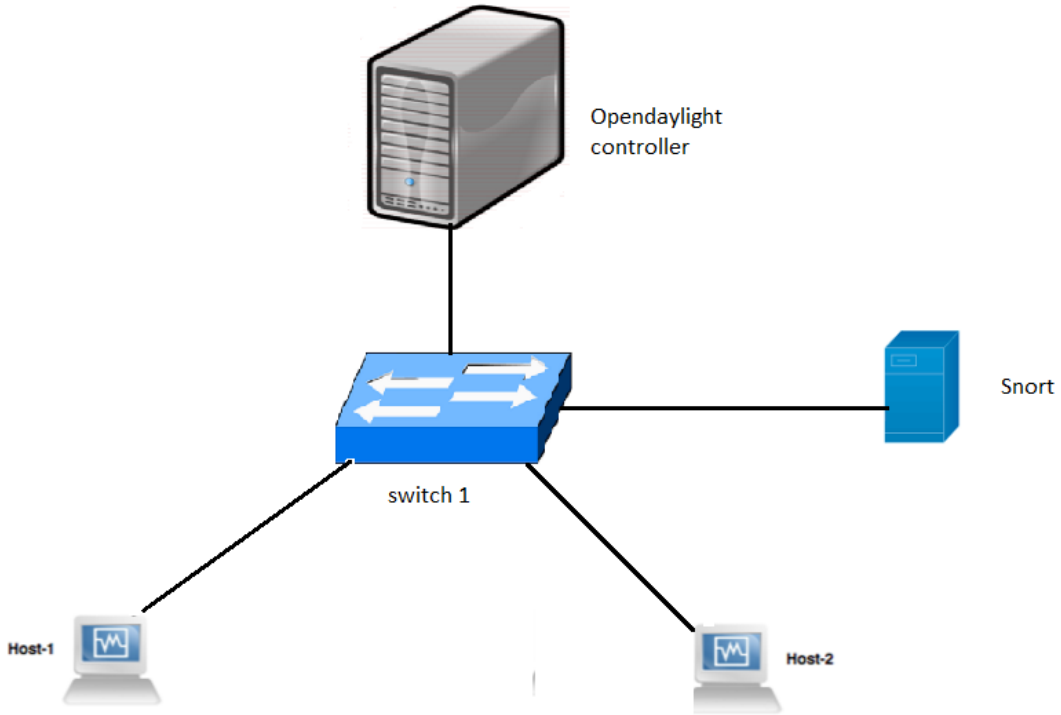
Είναι γεγονός πως τα δίκτυα υπολογιστών και οι συσκευές που είναι συνδεμένες σε αυτά και εξυπηρετούνται από αυτά πολύ συχνά πέφτουν θύματα δικτυακών επιθέσεων που στόχο έχουν να παρέμβουν στην λειτουργία του δικτύου, να υποκλέψουν δεδομένα και να παρεμποδίσουν την εξυπηρέτηση των καλόβουλων χρηστών.

Στόχος αυτή της διπλωματικής εργασίας είναι η ανάπτυξη ενός μοντέλου καταπολέμησης δικτυακών επιθέσεων σε ευφυή προγραμματιζόμενα δίκτυα όπου συνδυάζοντας όλα τα στοιχεία που αναφέρθηκαν στα προηγούμενα κεφάλαια αυτής της διπλωματικής θα επιτύχουμε να δημιουργήσουμε ένα σύστημα το οποίο θα φιλτράρει όλη την διακινούμενη πληροφορία σε ένα δίκτυο, θα διαχωρίζει την καλόβουλη από την κακόβουλη ροή δεδομένων και εν κατακλείδι θα αποκόπτει την κακόβουλη πληροφορία καθιστώντας το σύστημα μας ασφαλές και διαθέσιμο για την παροχή οποιασδήποτε είδους υπηρεσίας.

Όπως αναφέραμε και στο προηγούμενο κεφάλαιο η καταπολέμηση δικτυακών επιθέσεων μπορεί να είναι εξαιρετικά αποδοτική όταν γίνεται μέσω προγραμματισμού σε ευφυή δίκτυα έτσι σε αυτήν την διπλωματική θα εκμεταλλευτούμε την αρχιτεκτονική και την ευελιξία που μας προσφέρουν τα ευφυή προγραμματιζόμενα δίκτυα και μέσω της ανάπτυξης εφαρμογής στην πλατφόρμα ελέγχου Opendaylight θα αποδείξουμε ότι είναι δυνατή η καταπολέμηση οποιασδήποτε δικτυακής επίθεσης αποδοτικά σε δίκτυα ευφυούς τύπου.

Η πλατφόρμα ελέγχου Opendaylight είναι η ιδανική πλατφόρμα για την ανάπτυξη μιας τέτοιας εφαρμογής καθώς η σπονδυλοειδής μορφή της καθώς και η ευκολία με την οποία μπορούμε να προσθέσουμε μια νέα εφαρμογή η οποία μπορεί να συνεργάζεται άριστα και αποδοτικά με ήδη υπάρχουσες εφαρμογές στην πλατφόρμα δίνει την ευελιξία στον προγραμματιστή να αναπτύξει δικά του μοντέλα, να πειραματιστεί και εν τέλει να δημιουργήσει εφαρμογές οι οποίες θα είναι πρωτοποριακές θα επιλύουν προβλήματα στο κομμάτι των ευφυών δικτύων και θα δημιουργούν ισχυρά θεμέλια για την κατασκευή ολοκληρωμένων προϊόντων που θα εξυπηρετούν τις ανάγκες οποιουδήποτε χρήστη. Έτσι η ανάπτυξη του μοντέλου μας για την καταπολέμηση των επιβλαβών δικτυακών επιθέσεων έχει ως πυρήνα την πλατφόρμα ελέγχου Opendaylight και μέσω αυτού του μοντέλου θα αναδείξουμε τα συγκριτικά πλεονεκτήματα που έχουν τα ευφυή προγραμματιζόμενα δίκτυα σε σχέση με τα παρόντα μη ευφυή δίκτυα όσον αφορά το κομμάτι της ασφάλειας δικτύων.

Στην συνέχεια θα αναλύσουμε το μοντέλο το οποίο αναπτύξαμε στα πλαίσια αυτής της διπλωματικής το οποίο έχει ως στόχο την παροχή ασφάλειας σε ευφυή προγραμματιζόμενα δίκτυα μέσω της καταπολέμησης πιθανών απειλών για το δίκτυο. Εδώ είναι χρήσιμο να αναφέρουμε πως όλα τα στοιχεία που απαρτίζουν το μοντέλο που έχει αναπτυχθεί σε αυτήν την διπλωματική εργασία είναι open-source χαρακτηριστικό που μας διευκόλυνε ιδιαίτερα στην εργασία μας. Στην συνέχεια παραθέτουμε την εικόνα 27 η οποία παρουσιάζει την αρχιτεκτονική του μοντέλου μας και την οποία θα αναλύσουμε ευθύς αμέσως.



Τοπολογία Προσομοίωσης

Εικόνα 27

Στην εικόνα 27 παρουσιάζεται το μοντέλο που έχουμε αναπτύξει στα πλαίσια αυτής της διπλωματικής εργασίας. Αυτό το μοντέλο αποτελείται από την πλατφόρμα ελέγχου Opendaylight όπως φαίνεται στην κορυφή του σχήματος ,από την συσκευή στην οποία έχουμε εγκαταστήσει την εφαρμογή Snort για να αντιμετωπίσουμε οποιαδήποτε κακόβουλη κίνηση δεδομένων ,από τους δυο υπολογιστές (host 1 ,host2) οι οποίοι θα διαδραματίσουν τον ρόλο του θύτη και του θύματος στα πλαίσια μια δικτυακής επίθεσης και από μια κεντρική συσκευή δικτύου (switch) η οποία ενώνει όλα τα παραπάνω στοιχεία και καθιστά εφικτή την επικοινωνία μεταξύ αυτών.

Στην συνέχεια θα εξηγήσουμε τις παρεμβάσεις μας σε κάθε ένα στοιχείο της παραπάνω εικόνας καθώς και τις εφαρμογές που έχουμε αναπτύξει σε αυτά.

6.2.3 Επέκταση Opendaylight για ανίχνευση και αντιμετώπιση δικτυακών επιθέσεων

Από την αρχή ακόμα αυτής της διπλωματικής εργασίας είχαμε κατανοήσει ότι για να καταφέρουμε να φιλτράρουμε όλη την κίνηση δεδομένων που κυκλοφορείται σε ένα δίκτυο και στην συγκεκριμένη περίπτωση στο switch 1 πρέπει να δώσουμε εντολή στο switch αυτό να περάσει openflow κανόνες οι οποίοι εκτός των άλλων θα πρέπει να έχουν και ένα επιπλέον action το οποίο θα δίνει εντολή να διοχετεύεται αντίγραφο της κίνησης που υπάρχει στο δίκτυο προς την συσκευή η οποία έχει εγκατεστημένη την εφαρμογή Snort. Με τον τρόπο αυτό θα είμαστε σε θέση να παρακολουθούμε όλη την κίνηση που διακινείται μέσω του switch 1 και θα μπορούμε μέσω του snort να διαχωρίζουμε την καλόβουλη από την κακόβουλη πληροφορία και σε δεύτερο στάδιο θα είμαστε σε θέση να αποκόπτουμε τον χρήστη ο οποίος είναι υπεύθυνος για αυτήν την κακόβουλη πληροφορία που μολύνει το δίκτυό μας.

Έτσι αποφασίσαμε να αναπτύξουμε μια εφαρμογή στην πλατφόρμα ελέγχου Opendaylight η οποία θα παρακολουθεί τους openflow κανόνες οι οποίοι εγκαθίστανται στην συσκευή δικτύου Switch 1 και θα προσθέτει σε αυτούς τους κανόνες ένα επιπλέον

Action δηλαδή μια επιπλέον δράση η οποία θα έχει ως παράμετρο την θύρα στην οποία είναι συνδεμένη η συσκευή που τρέχει η εφαρμογή Snort. Μόλις εγκαταστήσουμε αυτό το επιπλέον action στους υπάρχοντες openflow κανόνες στην συνέχεια ένα αντίγραφο του συνόλου της κίνησης που κυκλοφορείται στην συσκευή δικτύου switch 1 θα διοχετεύεται στην εφαρμογή Snort όπου θα αναλύεται και θα φιλτράρεται.

Το πρώτο βήμα για την ανάπτυξη μιας εφαρμογής που θα εκτελεί τα προαναφερθέντα ήταν να επιλέξουμε το σωστό project της πλατφόρμα ελέγχου Opendaylight το οποίο θα μας παρέχει τα απαραίτητα εφόδια και τις απαραίτητες εφαρμογές με την βοήθεια των οποίον θα μπορέσουμε να αναπτύξουμε την δικιά μας εφαρμογή της οποίας το όνομα για ενεργοποίηση στην πλατφόρμα Opendaylight θα είναι odl-snort. Το project της πλατφόρμας ελέγχου Opendaylight που επιλέξαμε είναι το l2switch project. Ο λόγος ο οποίος επιλέξαμε το συγκεκριμένο project είναι γιατί περιέχει τις εφαρμογές εκείνες οι οποίες επιτρέπουν την εγκατάσταση openflow κανόνων στην συσκευή δικτύου switch 1 οι οποίοι κανόνες καθιστούν εφικτή την επικοινωνία μεταξύ των χρηστών που είναι συνδεμένοι στο switch 1.

Πιο συγκεκριμένα το project αυτό περιέχει εκείνες τις εφαρμογές οι οποίες αναγνωρίζουν πότε ένας καινούργιος χρήστης συνδέεται στο δίκτυο και στην συνέχεια εγκαθίστανται οι κατάλληλοι openflow κανόνες οι οποίοι θα βοηθήσουν τον καινούργιο αυτόν χρήστη να επικοινωνήσει με άλλους χρήστες που είναι ήδη συνδεμένοι στο δίκτυο. Για να το επιτύχουμε αυτό το l2switch project της πλατφόρμας ελέγχου Opendaylight εγκαθιστά openflow κανόνες στην συσκευή δικτύου switch 1 οι οποίοι κανόνες περιέχουν την MAC διεύθυνση του χρήστη που στέλνει ένα πακέτο δεδομένων ,την MAC διεύθυνση του χρήστη για τον οποίον προορίζεται το πακέτο δεδομένων που μόλις εστάλει και επίσης οι κανόνες αυτοί περιέχουν την θύρα από την οποία θα εξέλθει το πακέτο δεδομένων για να προσεγγίσει τον χρήστη για τον οποίο προορίζεται.

Στην συνέχεια θα αναφέρουμε και θα αναλύσουμε συνοπτικά τα στοιχεία που αποτελούν και συνθέτουν το l2switch project ώστε να κατανοήσουμε ακόμη περισσότερο την χρησιμότητά του ως βάση για την ανάπτυξη της δικής μας εφαρμογής.

6.2.3.1 L2switch project (Opendaylight controller)

Το l2switch project αποτελείται από έξι επιμέρους εφαρμογές από τις οποίες κάθε μία βοηθά καταλυτικά για την εγκαθίδρυση openflow κανόνων οι οποίοι επιτρέπουν την επικοινωνία των χρηστών που είναι συνδεμένοι στο δίκτυο [29].

Οι επιμέρους αυτές εφαρμογές είναι:

- **Packet Handler**

Η εφαρμογή Packet Handler είναι επιφορτισμένη με την ευθύνη της αποκωδικοποίησης των πακέτων δεδομένων που καταφτάνουν στην πλατφόρμα ελέγχου Opendaylight κα στην συνέχεια της αποστολής των πακέτων αυτών στους κατάλληλους προορισμούς. Πιο συγκεκριμένα η εφαρμογή αυτή περιέχει μεθόδους οι οποίες αποκωδικοποιούν τα νεοεισερχόμενα πακέτα δεδομένων. Για παράδειγμα υπάρχουν οι αποκωδικοποιητές ArpDecoder,IPv4Decoder και IPv6Decoder οι οποίοι αναλαμβάνουν να αποκωδικοποιήσουν ένα Ethernet πακέτο δεδομένων σε ένα Arp ,είτε σε ένα IPv4 , είτε σε ένα IPv6 πακέτο δεδομένων.

- **Loop Remover**

Η εφαρμογή Loop Remover εγγυάται την αποφυγή δημιουργίας βρόγχων σε ένα οποιοδήποτε δίκτυο. Δηλαδή αυτό που ουσιαστικά κάνει η συγκεκριμένη εφαρμογή είναι να εξερευνάει το υπάρχων δίκτυο και τις συνδέσεις που αυτό φέρει και στην συνέχεια να δημιουργεί και να αποθηκεύει έναν γράφο του δικτύου ο οποίο θα είναι απαλλαγμένος από οποιοδήποτε είδους βρόγχο που θα μπορούσε να προκαλέσει πρόβλημα στην λειτουργία αυτού.

- **Arp Handler**

Ο σκοπός της εφαρμογής Arp Handler είναι η διαχείριση των Arp πακέτων. Πιο συγκεκριμένα η εφαρμογή αυτή διαχειρίζεται και επεξεργάζεται τα Arp πακέτα που καταφτάνουν στην πλατφόρμα ελέγχου Opendaylight καθώς αρχικά έχει εγκαταστήσει έναν Openflow κανόνα που δίνει εντολή να προωθούνται όλα τα arp τύπου πακέτα δεδομένων στην πλατφόρμα ελέγχου Opendaylight. Στην συνέχεια συμβουλεύεται την

λίστα που είναι αποθηκευμένες όλες οι συνδέσεις που υπάρχουν εκείνη την χρονική στιγμή στην συσκευή δικτύου switch 1 ώστε να στείλει τα arp πακέτα δεδομένων στον σωστό Ηπροορισμό.

- **Address Tracker**

Ο σκοπός της εφαρμογής Address Tracker είναι η αναγνώριση των διευθύνσεων των οντοτήτων που είναι συνδεμένες στο δίκτυο. Πιο συγκεκριμένα η εφαρμογή αυτή μαθαίνει τις διευθύνσεις MAC και IP των στοιχείων που υπάρχουν στο δίκτυο και στην συνέχεια αποθηκεύει αυτές τις διευθύνσεις στην μονάδα αποθήκευσης της πλατφόρμας ελέγχου Opendaylight. Οι διευθύνσεις αυτές γίνονται γνωστές στην εφαρμογή μέσω arp και ip πακέτων δεδομένων που καταφτάνουν στην πλατφόρμα και περιέχουν αυτήν την πληροφορία.

- **Host Tracker**

Η Host Tracker εφαρμογή είναι επιφορτισμένη με την ανακάλυψη νέων οντοτήτων που συνδέονται στο δίκτυο. Αναλυτικότερα η εφαρμογή αυτή ανακαλύπτει σε ποια τοποθεσία είναι συνδεμένη μία συσκευή στο δίκτυο και στην συνέχεια ενημερώνει την τοπολογία που είναι αποθηκευμένη στην μονάδα αποθήκευσης της πλατφόρμας ελέγχου Opendaylight για την νέα αυτή συσκευή. Άρα η πλατφόρμα ελέγχου γνωρίζει ανά πάσα στιγμή ποιες οντότητες είναι συνδεμένες στο δίκτυο και ποιες όχι και γνωρίζει επίσης την μορφή του δικτύου ώστε να μπορεί να το διαχειριστεί αποδοτικότερα.

- **L2Switch Main**

Η εφαρμογή L2Switch Main είναι υπεύθυνη για την εγκατάσταση openflow κανόνων ώστε να καταστεί εφικτή η επικοινωνία μεταξύ των διαφόρων οντοτήτων του ευφυούς δικτύου. Πιο συγκεκριμένα η εφαρμογή αυτή αντιδρά στην δικτυακή κίνηση που διακινείται στην συσκευή δικτύου switch 1 και στην συνέχεια εγκαθιστά MAC to MAC openflow κανόνες οι οποίοι καθιστούν εφικτή την μεταφορά πακέτων δεδομένων μεταξύ των διαφόρων χρηστών του δικτύου. Οι κανόνες αυτοί στα πεδία ταυτοποίησης (match fields) έχουν καταχωρίσει την διευθύνσεις MAC πηγής και προορισμού για κάθε ζεύγος οντοτήτων του δικτύου ώστε να είναι δυνατή η μεταγωγή της πληροφορίας μεταξύ αυτών των ζευγών μέσω της ταυτοποίησης των πακέτων δεδομένων από τα πεδία ταυτοποίησης των openflow κανόνων.

6.2.3.2 Διαδικασία κατασκευής odl-snort εφαρμογής

Ο τρόπος με τον οποίο κατασκευάσαμε την εφαρμογή odl-snort είναι ο κοινός τρόπος κατασκευής μιας εφαρμογής στο περιβάλλον της πλατφόρμας Opendaylight τον οποίο επεξηγήσαμε αναλυτικά στην παράγραφο 5.6. Σε αυτήν την παράγραφο θα επικεντρωθούμε στο κομμάτι σύνδεσης της εφαρμογής μας με τα υπόλοιπα στοιχεία του project l2switch καθώς η εφαρμογή μας ουσιαστικά καταναλώνει πληροφορία που παράγεται από τα στοιχεία που αποτελούν το l2switch project. Επίσης θα εξηγήσουμε αναλυτικά τον κώδικα που αναπτύξαμε στην εφαρμογή μας και τις διαδικασίες που ακολουθούνται στον κώδικα αυτόν.

Για να συνδέσει ο προγραμματιστής μια νέα εφαρμογή σε ένα προϋπάρχουν project το οποίο περιέχει σε υπό-φακέλους και άλλες εφαρμογές το πρώτο πράγμα που πρέπει να κάνει είναι να προσθέσει την νέα εφαρμογή στο aggregarion pom.xml του project. Στο aggregation pom.xml υπάρχουν εγγεγραμμένες όλες οι εφαρμογές οι οποίες βρίσκονται σε υπό-φακέλους στο project άρα σε αυτό το pom.xml πρέπει ο προγραμματιστής να προσθέσει και την νέα εφαρμογή.

Αυτή είναι και η διαδικασία που ακολουθήσαμε και με την εφαρμογή odl-snort. Μόλις δημιουργήσαμε την βασική ραχοκοκαλιά της εφαρμογής μας μέσω του τρόπου που εξηγήσαμε στην παράγραφο 5.6 στην συνέχεια την εγγράψαμε στο aggregation pom.xml του l2switch project ώστε ο Apache Maven να γνωρίζει ότι υπάρχει μια νέα εφαρμογή ως υπό-φάκελος στο l2switch project. Στην εικόνα 28 παραθέτουμε αυτό το aggregation pom.xml στο οποίο εκτός των άλλων έχουμε συμπεριλάβει και την εφαρμογή μας.

Όπως παρατηρούμε στην εικόνα 28 στο section modules υπάρχει ένα module που έχει όνομα snort αυτή είναι η νέα εφαρμογή που έχουμε συμπεριλάβει στο l2switch project. Έτσι ο Apache Maven ο οποίος είναι υπεύθυνός για το σωστή εγκατάσταση του l2switch project θα γνωρίζει ότι εκτός τον άλλων εφαρμογών πρέπει να εγκαταστήσει και την νέα εφαρμογή snort. Η εφαρμογή που αναφέρεται στην παρακάτω εικόνα ως module snort είναι η εφαρμογή odl-Snort που αναφέρουμε σε αυτήν την παράγραφο. Ο λόγος που την αναφέρουμε odl-snort είναι για να αποφύγουμε την σύγχυση ανάμεσα στην εφαρμογή snort που έχουμε αναπτύξει στην πλατφόρμα ελέγχου Opendaylight (odl-snort) και στο πρόγραμμα Snort που χρησιμοποιούμε για την ανίχνευση κακόβουλης πληροφορίας.

Αρχικό Pom.xml

Εικόνα 28

Αφού συμπεριλάβαμε την εφαρμογή μας στο aggregation pom.xml του l2switch project στην συνέχεια θα πρέπει να ορίσουμε στην νέα εφαρμογή τις εξαρτήσεις (dependencies) οι οποίες θα δώσουν εντολή στον Apache Maven να προικίσει την νέα μας εφαρμογή με τις απαραίτητες java κλάσεις τις οποίες θα χρειαστεί ώστε να υλοποιήσει την λειτουργία που εξυπηρετεί. Άρα στο φακελο implementation της νέας μας εφαρμογής υπάρχει ένα pom.xml αρχείο στο οποίο θα πρέπει να ορίσουμε όλες τις εξαρτήσεις τις οποίες χρειάζεται η νέα μας εφαρμογή και θα ζητήσει από τον Apache Maven να επιλύσει. Στην εικόνα 29 παρουσιάζουμε αυτό το Pom.xml αρχείο που βρίσκεται στον φάκελο implementation της νέας εφαρμογής μας.

Για να κατανοήσουμε ακόμα περισσότερο την σημασία των εξαρτήσεων για την νέα εφαρμογή μας θα αναφερθούμε στην εξάρτηση με το groupId org.opendaylight.openflowplugin.model. Αυτή η εξάρτηση είναι απαραίτητη να δηλωθεί καθώς περιέχει μια υπηρεσία την SalFlowService την οποία χρειαζόμαστε αν χρησιμοποιήσουμε στην εφαρμογή μας. Άρα για να καταλάβει ο Apache Maven ποιες

εξαρτήσεις έχει η εφαρμογή μας από άλλα υπάρχοντα project πρέπει να δηλώσουμε αυτές τις εξαρτήσεις στο Pom.xml της εφαρμογής μας ώστε ο Apache Maven να μας εφοδιάσει με τις απαραίτητες java classes που χρειαζόμαστε για να υλοποιήσουμε την εφαρμογή μας.

```
<dependency>
  <groupId>${project.groupId}</groupId>
  <artifactId>snort-api</artifactId>
  <version>${project.version}</version>
</dependency>

<dependency>
  <groupId>org.opendaylight.controller</groupId>
  <artifactId>sal-binding-api</artifactId>
</dependency>
<dependency>
  <groupId>org.opendaylight.openflowplugin.model</groupId>
  <artifactId>model-flow-service</artifactId>
</dependency>
<dependency>
  <groupId>org.opendaylight.controller.model</groupId>
  <artifactId>model-topology</artifactId>
</dependency>
<dependency>
  <groupId>org.opendaylight.yangtools</groupId>
  <artifactId>yang-common</artifactId>
</dependency>
<dependency>
  <groupId>org.opendaylight.l2switch.packethandler</groupId>
  <artifactId>packethandler-model</artifactId>
</dependency>
<dependency>
  <groupId>org.opendaylight.l2switch.packethandler</groupId>
  <artifactId>packethandler-impl</artifactId>
</dependency>
<dependency>
  <groupId>org.opendaylight.l2switch.addresstracker</groupId>
  <artifactId>addresstracker-model</artifactId>
</dependency>
<dependency>
  <groupId>org.opendaylight.l2switch.loopremover</groupId>
  <artifactId>loopremover-model</artifactId>
</dependency>
```

Odl-snort-implementation pom.xml

Εικόνα 29

Στην συνέχεια θα αναλύσουμε τον τρόπο με τον οποίο παρέχονται στην εφαρμογή μας οι απαραίτητες υπηρεσίες ώστε να μπορεί για παράδειγμα να έχει πρόσβαση σε δεδομένα τα οποία βρίσκονται στους χώρους αποθήκευσης δεδομένων της Opendaylight πλατφόρμας , να μπορει να εισάγει , να εξάγει και να τροποποιεί τα δεδομένα αυτά και να μπορεί να εκτελεί απομακρυσμένες κλήσεις διαδικασιών (RPCs). Για να είναι σε θέση η εφαρμογή μας να εκτελεί τις προαναφερθέντες λειτουργίες πρέπει εμείς ως κατασκευαστές της εφαρμογής να δηλώσουμε τις προαναφερθείσες MD-SAL υπηρεσίες ώστε το MD-SAL να τις παράσχει στην εφαρμογή μας. Για να γίνει λοιπόν αυτή η παροχή MD-SAL υπηρεσιών στην εφαρμογή μας πρέπει να δηλώσουμε τις υπηρεσίες αυτές σε ένα YANG αρχείο το οποίο βρίσκεται μέσα στο φάκελο implementation της εφαρμογής μας και ονομάζεται στην προκειμένη περίπτωση snort-impl.yang. Το περιεχόμενο αυτού του Yang αρχείου φαίνεται στην εικόνα 30.

```
container broker {
        uses config:service-ref {
           refine type {
              mandatory false;
              config:required-identity mdsal:binding-async-data-broker;
                    }
                 }
            }

container notification-service {
        uses config:service-ref {
          refine type {
            mandatory true;
            config:required-identity mdsal:binding-notification-service;
          }
        }
      }

container rpc-registry {
        uses config:service-ref {
          refine type {
            mandatory true;
            config:required-identity mdsal:binding-rpc-registry;
          }
        }
      }
```

Snort-impl.yang

Εικόνα 30

Όπως φαίνεται στην παραπάνω εικόνα η εφαρμογή για να λειτουργήσει σωστά και για να εκτελέσει τις διεργασίες που επιθυμεί πρέπει να συνδεθεί με τις υπηρεσίες MD-SAL που βρίσκονται στην πλατφόρμα ελέγχου Opendaylight. Οι υπηρεσίες με τις οποίες χρειάζεται να συνδεθεί η νέα μας εφαρμογή είναι οι mdsal data-broker,mdsal notification-service και mdsal rpc-registry. Η πρώτη υπηρεσία δίνει την δυνατότητα στην εφαρμογή μας να εκτελεί συναλλαγές με τις βάσεις αποθήκευσης της πλατφόρμας Opendayight , η δεύτερη

επιτρέπει την παροχή ειδοποιήσεων στην εφαρμογή μας και η τρίτη δίνει το δικαίωμα στην εφαρμογή μας να καταχωρίσει νέες απομακρυσμένες κλήσεις διαδικασιών (RPCs).

Στην συνέχεια θα αναφερθούμε στην java class με το όνομα SnortModule.java .Η συγκεκριμένη κλάση της java είναι ιδιαίτερα σημαντική για την εφαρμογή μας καθώς στο εσωτερικό αυτής της κλάσης βρίσκεται η μέθοδος createInstance(). Η μέθοδός createInstance() είναι ουσιαστικά το σημείο από το οποίο εκκινεί η εφαρμογή μας στην Opendaylight πλατφόρμα ελέγχου. Η μέθοδος αυτή συγκεκριμένα αρχικοποιεί και <<καλωδιώνει >> της νέα εφαρμογή στην πλατφόρμα. Όπως παρατηρούμε από τον κώδικα της SnortModule.createInstance() που παρουσιάζεται στην εικόνα 31 χρησιμοποιούμε τις υπηρεσίες MD-SAL RPCregistry και databroker όπως επίσης χρησιμοποιούμε την υπηρεσία SalFlowService η οποία θα μας χρειαστεί για να τροποποιήσουμε τους Openflow κανόνες της συσκευής δικτύου switch 1 που αναφέραμε στην εικόνα 27.

```
public java.lang.AutoCloseable createInstance() {



    RpcProviderRegistry rpcRegistryDependency = getRpcRegistryDependency();
    SalFlowService salFlowService = rpcRegistryDependency.getRpcService(SalFlowService.class);
    DataBroker dataservice = getBrokerDependency();
```

createInstance μέθοδος

Εικόνα 31

Στην συνέχεια θα αναλύσουμε την κλάση SnortImplementation.java όπου και περιέχεται ο κώδικας που περιγράφει την λειτουργία της odl-snort εφαρμογής μας. Στην κλάση αυτή υλοποιούμε αρχικά την μέθοδο onDataChanged η οποία μέθοδος αυτή παρακολουθεί την μονάδα αποθήκευσης της πλατφόρμας Opendaylight και παρατηρεί για Openflow κανόνες που αποθηκεύονται σε αυτήν την μονάδα αποθήκευσης και οι οποίοι κανόνες είναι τέτοιοι που επιτρέπουν την επικοινωνία μεταξύ χρηστών του δικτύου. Μόλις αντιληφθούμε κάποιον τέτοιο Openflow κανόνα στην συνέχεια αναζητούμε την θύρα της δικτυακής συσκευής που είναι συνδεμένη η Snort IDS εφαρμογή μας και στην συνέχεια παραμετροποιούμε τους Openflow κανόνες προσθέτοντας μια νέα δράση η οποία θα διοχετεύει ένα αντίγραφο της κίνησης που διακινείται στο εσωτερικό της δικτυακής συσκευής. Η παραμετροποίηση αυτών των κανόνων γίνεται μέσω του RPC SalflowService όπου έχουμε την δυνατότητα μέσω αυτού του RPC να παραμετροποιήσουμε προϋπάρχοντάς Openflow κανόνες.

Το επόμενο βήμα στην ανάλυσή μας σχετικά με την odl-snort εφαρμογή είναι η επεξήγηση του τρόπου με τον οποίο η εφαρμογή αυτή γίνεται αντιληπτή από το Karaf Container της πλατφόρμας Opendaylight με την μορφή feature και πως μπορούμε να την ενεργοποιήσουμε.

Όπως αναφέραμε και στην παράγραφο 5.6 για να συμπεριλάβει το Karaf container την εφαρμογή μας και να την ενεργοποιήσει σαν μια OSGi εφαρμογή πρέπει να την συμπεριλάβουμε στο feature.xml. Σε αυτό το feature.xml αρχείο το οποίο βρίσκεται στο υπό-φάκελο features εκτός από την δικιά μας odl-snort εφαρμογή υπάρχουν όλες οι εφαρμογές οι οποίες ενεργοποιούνται σαν feature στο karaf container. Στην Εικόνα 32 φαίνεται η περιγραφή της εφαρμογής μας σε αυτό το feature.xml αρχείο.

```
<feature name='odl-snort' version='${project.version}' description="L2Switch :: Snort">
 <feature version="${project.version}">odl-l2switch-hosttracker</feature>


    <bundle>mvn:org.opendaylight.l2switch.snort/snort-impl/${project.version}</bundle>
```

Δήλωση στο feature.xml

Εικόνα 32

Στην παραπάνω εικόνα (εικόνα 32) παρατηρούμε στην πρώτη γραμμή το όνομα με το οποίο θα απεικονίζεται η εφαρμογή μας στο Karaf container της πλατφόρμας Opendaylight το όνομα αυτό είναι odl-snort. Επίσης στην δεύτερη γραμμή παρατηρούμε ένα δεύτερο feature το οποίο ονομάζεται odl-l2switch-hosttracker. Όταν το Karaf container λάβει εντολή να ενεργοποιήσει το odl-snort το container αυτό πριν ενεργοποιήσει το odl-snort θα ενεργοποιήσει αρχικά το odl-l2switch-hosttracker το οποίο είναι προ-απαιτούμενο για να λειτουργήσει η εφαρμογή μας odl-snort. Στην συνέχεια παρατηρούμε την τρίτη γραμμή με την οποία καταδεικνύουμε στο Karaf container σε ποιο φάκελο θα βρει τον κώδικα της εφαρμογής μας για να τον εκτελέσει. Παρατηρούμε ότι ο κώδικας είναι στο υπό-φάκελο snort – imp που είναι ο implementation φάκελος της εφαρμογής μας. Άρα μόλις εκκινήσουμε εμείς την πλατφόρμα Opendaylight και στην συνέχεια εκτελέσουμε την εντολή feature : install odl-snort τότε το Karaf container θα φορτώσει την εφαρμογή μας στην πλατφόρμα Opendaylight και στην συνέχεια η εφαρμογή μας θα είναι διαθέσιμη για χρήση.

6.2.4 Snort – Opendaylight Agent

Το επόμενο στάδιο στην τοπολογία μας που οδηγεί σε ένα ολοκληρωμένο σύστημα ανίχνευσης και καταστολής κακόβουλης πληροφορίας είναι η Snort IDS εφαρμογή και το Python πρόγραμμα που την συνοδεύει.

Όπως αναφέραμε στην προηγούμενη παράγραφο της διπλωματικής μας η εφαρμογή odl-snort που αναπτύξαμε στην πλατφόρμα Opendaylight έχει ως βασική λειτουργία την καθοδήγηση αντιγράφου κίνησης δεδομένων τα οποία διακινούνται στην συσκευή δικτύου Switch 1 σε μία συγκεκριμένη θύρα της συσκευής δικτύου αυτής στην οποία είναι συνδεμένη η Snort IDS εφαρμογή μας.

Στην συνέχεια η Snort IDS εφαρμογή δέχεται την κίνηση δεδομένων αυτή την οποία ελέγχει με βάση Snort κανόνες τους οποίους έχει ορίσει ο χρήστης με σκοπό την ανακάλυψη τυχόν κακόβουλης πληροφορίας που έχει ως στόχο την μόλυνση του δικτύου.

Σε περίπτωση ανίχνευσης τέτοιας κακόβουλης πληροφορίας η Snort IDS εφαρμογή εισάγει τις πληροφορίες των κακόβουλων πακέτων δεδομένων σε ένα αρχείο του οποίου το όνομα είναι Alert μαζί με τις πληροφορίες του Snort κανόνα που ανακάλυψε την κακόβουλη κίνηση. Οι πληροφορίες των κακόβουλων πακέτων που εμφανίζονται στο Alert αρχείο είναι πληροφορίες όπως διευθύνσεις IP προορισμού και πηγής και επίσης πληροφορίες σχετικά με την θύρα εισόδου του πακέτου και την θύρα εξόδου που προτίθεται να προσεγγίσει. Όσον αφορά τις πληροφορίες που τυπώνονται στο Alert αρχείο και αφορούν τους Snort κανόνες αυτές είναι το μήνυμα του κανόνα και η υπογραφή του τα οποία μας βοηθούν να αναγνωρίσουμε τον κανόνα ο οποίος ανακάλυψε την κακόβουλη πληροφορία.

Στην συνέχεια θα εξηγήσουμε πως χρησιμοποιώντας ένα Python πρόγραμμα καταφέρνουμε να μετατρέψουμε το σύστημα μας από σύστημα ανίχνευσης κακόβουλης πληροφορίας σε σύστημα καταστολής κακόβουλης πληροφορίας δηλαδή σε μια IPS (Intrusion Prevention System) εφαρμογή.

Όπως αναφέραμε προηγουμένως η Snort IDS εφαρμογή τυπώνει την οποιαδήποτε ανίχνευση κακόβουλης πληροφορίας σε ένα Alert αρχείο. Αυτό που εξυπηρετεί στην συνέχεια το Python πρόγραμμα είναι να διαβάζει σε πραγματικό χρόνο την πληροφορία που έχει τυπωθεί στο Alert αρχείο σχετικά με την κακόβουλη κίνηση δεδομένων και στην συνέχεια με την βοήθεια του REST εκτελεί Post http requests ώστε να δώσει εντολή στην Opendaylight πλατφόρμα να τοποθετήσει Openflow κανόνες στην συσκευή δικτύου οι οποίοι κανόνες θα αποκόπτουν την κακόβουλη πληροφορία και θα την εμποδίζουν να φτάσει στον τελικό χρήστη.

Τα Post http requests που εκτελεί το Python πρόγραμμα έχουν ως βάση ένα xml αρχείο στο οποίο εισάγονται οι πληροφορίες που έχουν διαβαστεί από το Alert αρχείο σχετικά με την κακόβουλη κίνηση ώστε οι κανόνες Openflow που θα περαστούν να έχουν τα ίδια χαρακτηριστικά με αυτά των κακόβουλων πακέτων δεδομένων. Δηλαδή εισάγονται πληροφορίες σχετικά με την διεύθυνση IP πηγής και προορισμού της κακόβουλης πληροφορίας.

7 Πειραματικά αποτελέσματα

Στην παράγραφο αυτή θα παρουσιάσουμε τα εργαλεία που χρησιμοποιήσαμε και τα οποία μας βοήθησαν να εκτελέσουμε την προσομοίωση μας ώστε να καταπολεμήσουμε μια ενδεχόμενη DDoS επίθεση όπως επίσης θα εξηγήσουμε κάθε στάδιο της προσομοίωσης αυτής. Ακόμα θα παρουσιάσουμε τα αποτελέσματα που προέκυψαν και θα εξάγουμε ασφαλή συμπεράσματα σχετικά με την αποδοτικότητα του μοντέλου μας και την διαδικασία καταπολέμησης της ενδεχόμενης απειλής. Τέλος τα αποτελέσματα αυτά θα συνοδεύονται από επεξηγηματικές εικόνες και γραφήματα που θα απεικονίζουν την κατάσταση του δικτύου πριν και μετά την καταπολέμηση της DDoS επίθεσης και επίσης θα παρουσιάζουν πληροφορίες που σχετίζονται με το δίκτυο και θα αποδεικνύουν την αποδοτικότητα στην καταπολέμηση δικτυακών επιθέσεων σε ευφυή δίκτυα.

7.1 Εργαλεία προσομοίωσης δικτύου

Το mininet είναι ένα πολύ χρήσιμο εργαλείο το οποίο βοήθησε καταλυτικά σε αυτήν την διπλωματική εργασία. Είναι ουσιαστικά ένας δικτυακός προσομοιωτής μέσω του οποίου ο χρήστης μπορεί να προσομοιώσει, παραμετροποιήσει και να λειτουργήσει ένα εικονικό δίκτυο το οποίο θα έχει την μορφή και τις δυνατότητες που αυτός θα επιλέξει [30].

Το mininet χρησιμοποιεί μια συλλογή από εικονικές δικτυακές συσκευές όπως switches , routers , hosts , links κ.α. τα οποία τα φορτώνει σε έναν Linux πυρήνα (Linux kernel) δημιουργώντας έτσι ένα εικονικό ολοκληρωμένο δίκτυο. Ένας mininet host συμπεριφέρεται όπως και ένα κανονικός υπολογιστής. Ο mininet host έχει την δυνατότητα

να "τρέχει" προγράμματα (τα οποία είναι εγκατεστημένα στο Linux συστημα) όπως για παράδειγμα ένα πρόγραμμα που είναι επιφορτισμένο με την αποστολή πακέτων μέσω μίας Ethernet διεπαφής στα οποία έχουμε χορηγήσει μια συγκεκριμένη ταχύτητα αποστολής. Πιο συγκεκριμένα οι εικονικές συσκευές που παρέχει το mininet δημιουργήθηκαν μόνο με την βοήθεια λογισμικού και στις περισσότερες περιπτώσεις συμπεριφέρονται με τον ίδιο τρόπο όπως και οι πραγματικές δικτυακές συσκευές του εμπορίου. Στην συνέχεια θα αναλύσουμε μερικά από τα πλεονεκτήματα του mininet και θα απεικονίσουμε ένα παράδειγμα τοπολογίας που δημιουργήθηκε μέσω αυτού του προγράμματος.

Ένα από τα βασικά πλεονεκτήματα του mininet είναι η ταχύτητα με την οποία μπορείς να δημιουργήσεις ένα εικονικό δίκτυο. Με το mininet ο χρήστης μπορεί κυριολεκτικά να δημιουργήσει ένα οποιασδήποτε μορφής εικονικό δίκτυο μέσα σε ελάχιστα δευτερόλεπτα. Επίσης ένας χρήστης μπορεί να δημιουργήσει απλά ή σύνθετα python scripts τα οποία θα χρησιμοποιηθούν για την διεξαγωγή πειραμάτων πάνω στην mininet εφαρμογή. Ένα από τα σημαντικότερα πλεονεκτήματα του mininet είναι ότι είναι open-source project πράγμα που σημαίνει ότι ο οποιοσδήποτε χρήστης μπορεί να έχει πρόσβαση και να εξετάσει των πηγαίο κώδικα της εφαρμογής, να τον παραμετροποιήσει κατά το δοκούν , να διορθώσει αστοχίες που μπορεί να υπάρξουν και επίσης μπορεί να προσθέσει επιπλέον κομματια κώδικα που ο ίδιος έχει δημιουργήσει.

Στην συνέχεια θα αναλύσουμε μια mininet εντολή μέσω της οποίας δημιουργείται το εικονικό δίκτυο που απεικονίζεται στην Εικόνα 33 .Η εντολή είναι:

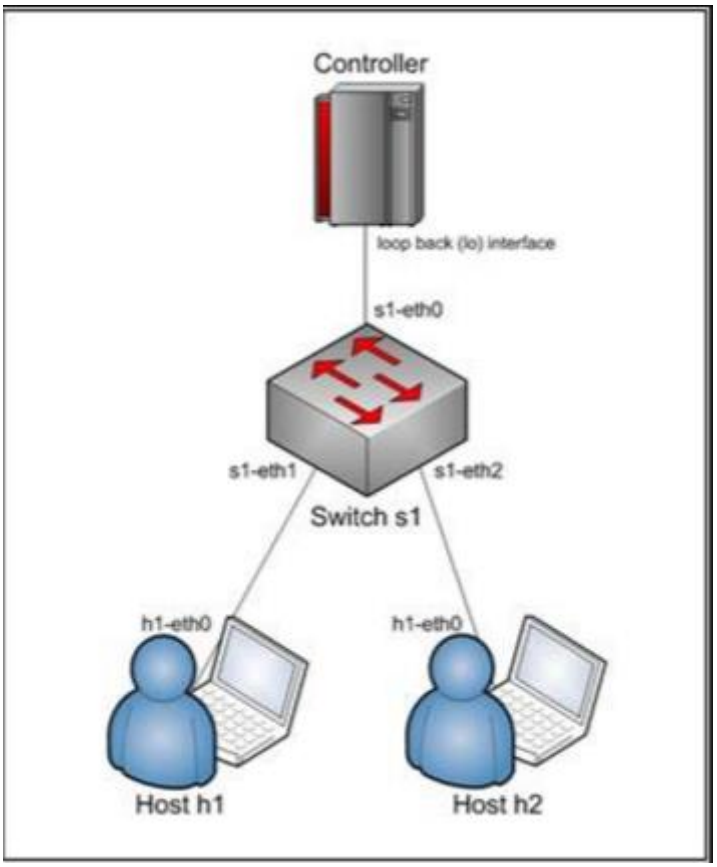
$ sudo mn - - mac - - topo single,2 - -controller remote ,ip=(controller's IP)

- ***- - mac :*** Μέσω αυτης της λεξης-κλειδι αναθέτουμε στο mininet να ορίσει αυτόματα διευθύνσης mac στις δικτυακές συσκευές.

- ***- - topo :*** Με την λέξη topo εξηγούμε στο mininet την μορφή που θέλουμε να έχει η τοπολογία δικτύου. Στην συγκεκριμένη περίπτωση η μορφή της τοπολογίας μας είναι ένα switch (s1) και δύο hosts (h1,h2).

- ***- - controller remote :*** Με αυτήν την λέξη δίνουμε εντολή στο mininet να μην χρησιμοποιήσει την δικιά του μονάδα ελέγχου αλλά να χρησιμοποιήσει μια μονάδα ελέγχου της αρεσκείας του χρήστη την οποία θα την βρει το mininet σε μια

διεύθυνση IP που θα έχει καθορίσει ο χρήστης (π.χ. η μονάδα ελέγχου Opendaylight).



Παράδειγμα τοπολογίας Mininet

Εικόνα 33

7.2 Περιβάλλον εργασίας

Στην παράγραφο αυτή θα αναλύσουμε την προσομοίωση που εκτελέσαμε με σκοπό την δοκιμή του συστήματος που αναπτύξαμε στα πλαίσια αυτής της διπλωματικής εργασίας και στην συνέχεια θα παρουσιάσουμε τα αποτελέσματα που προέκυψαν από την προσομοίωση αυτή.

Η τοπολογία που θα χρησιμοποιήσουμε για την προσομοίωση είναι αυτή που φαίνεται στην εικόνα 28. Αρχικά μέσω mininet προγράμματος θα δημιουργήσουμε την τοπολογία που εμφανίζεται στην εικόνα 28 στην οποία έχουμε τους δύο Hosts και την πλατφόρμα Opendaylight συνδεδεμένα στην συσκευή δικτύου switch 1 .Στην συνέχεια συνδέουμε το Snort IDS-IPS πρόγραμμα στην συσκευή δικτύου switch 1 ώστε να ολοκληρωθεί η τοπολογία μας όπως αυτή φαίνεται στην εικόνα 28.

Στην επόμενη φάση ενεργοποιούμε την odl-snort εφαρμογή μας στην πλατφόρμα Opendaylight η οποία στην συνέχεια εγκαθιστά Openflow κανόνες στην συσκευή δικτύου switch 1 για να καταστήσει δυνατή την επικοινωνία μεταξύ των δύο host . Επίσης προσθέτει σε αυτούς τους Openflow κανόνες την επιπλέον δράση που αντιπροσωπεύει την θύρα της συσκευής δικτύου όπου είναι συνδεδεμένη η Snort IDS-IPS εφαρμογή μας ώστε να στείλουμε αντίγραφο της διακινούμενης εντός της συσκευής δικτύου switch 1 πληροφορία.

Αφού δημιουργήσαμε μια τοπολογία ευφυούς δικτύου η οποία είναι πλήρως λειτουργική συνεχίζουμε στο επόμενο βήμα της προσομοίωσης που είναι η προσπάθεια μόλυνσης του δικτύου μέσω μίας DDoS (Distributed Denial of Service) επίθεσης. Συγκεκριμένα στον Host 1 εκτελούμε μέσω TCPreplay ένα αρχείο το οποίο περιέχει κακόβουλη πληροφορία η οποία έχει εξαχθεί από DDoS επίθεση του παρελθόντος. Στην συνέχεια αυτή η DDoS κακόβουλη πληροφορία προσεγγίζει τον Host 2 και επίσης ένα αντίγραφο αυτής εισάγεται και στο πρόγραμμα Snort IDS-IPS που έχουμε συνδέσει στην συσκευή δικτύου switch 1. Όταν το Snort IDS-IPS πρόγραμμα ανιχνεύσει την κακόβουλη πληροφορία μέσω Snort κανόνων αποθηκεύει τις πληροφορίες σχετικά με την κακόβουλη κίνηση δεδομένων στο Alert αρχείο. Στην συνέχεια σε πραγματικό χρόνο το Python script που συνοδεύει το Snort πρόγραμμα διαβάζει τις πληροφορίες που έχουν αποθηκευτεί στο

Alert αρχείο και εκτελεί Post http requests προς την πλατφόρμα ελέγχου Opendaylight ώστε να δώσει εντολή στην πλατφόρμα να εισάγει Openflow κανόνες που θα αποκόπτουν την κακόβουλη πληροφορία και θα την εμποδίζουν να φτάσει στον Host 2.

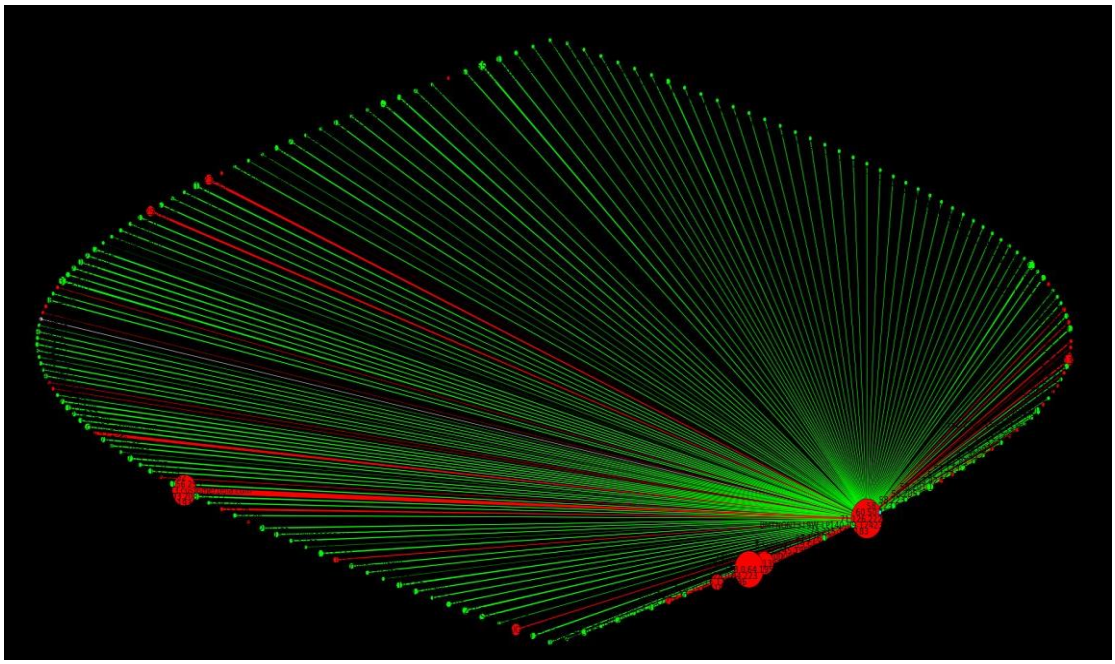
Με τον τρόπο αυτό καταφέρνουμε να διατηρήσουμε τον Host 2 ενεργό και πλήρως λειτουργικό χωρίς να έχει υποστεί τις επιπτώσεις μιας DDoS επίθεσης. Από την άλλη μεριά ο Host 1 οποίος είναι υπεύθυνος για την DDoS επίθεση δεν έχει πλέον την δυνατότητα να επικοινωνήσει με τον Host 2 και συνάμα να τον απειλήσει.

Το πιο σημαντικό εύρημα αυτής της προσομοίωσης είναι ότι μη κακόβουλη χρήστες του δικτύου μπορούν χωρίς προβλήματα να επικοινωνήσουν με τον Host 2 και να ανταλλάξουν δεδομένα αποδοτικά.

7.3 Αποτελέσματα

Σε αυτήν την παράγραφο θα παρουσιάσουμε αναλυτικά τα αποτελέσματα που προέκυψαν από την προσομοίωση που αναφέραμε στην προηγούμενη παράγραφο.

Στην εικόνα 34 παρουσιάζεται η εικόνα που έχει το δίκτυο μας κατά την διάρκεια μιας DDoS επίθεσης που εκτυλίσσεται με βασικό στόχο την διακοπή παροχή υπηρεσιών ενός συγκεκριμένου κόμβου. Η επίθεση έχει ως αποτέλεσμα την αποστολή μεγάλου όγκου δεδομένων προς έναν συγκεκριμένο κόμβο και έτσι ο κόμβος αυτός δεν δύναται να εξυπηρετήσει καλόβουλους χρήστες που σκοπεύουν να χρησιμοποιήσουν τις υπηρεσίες του.



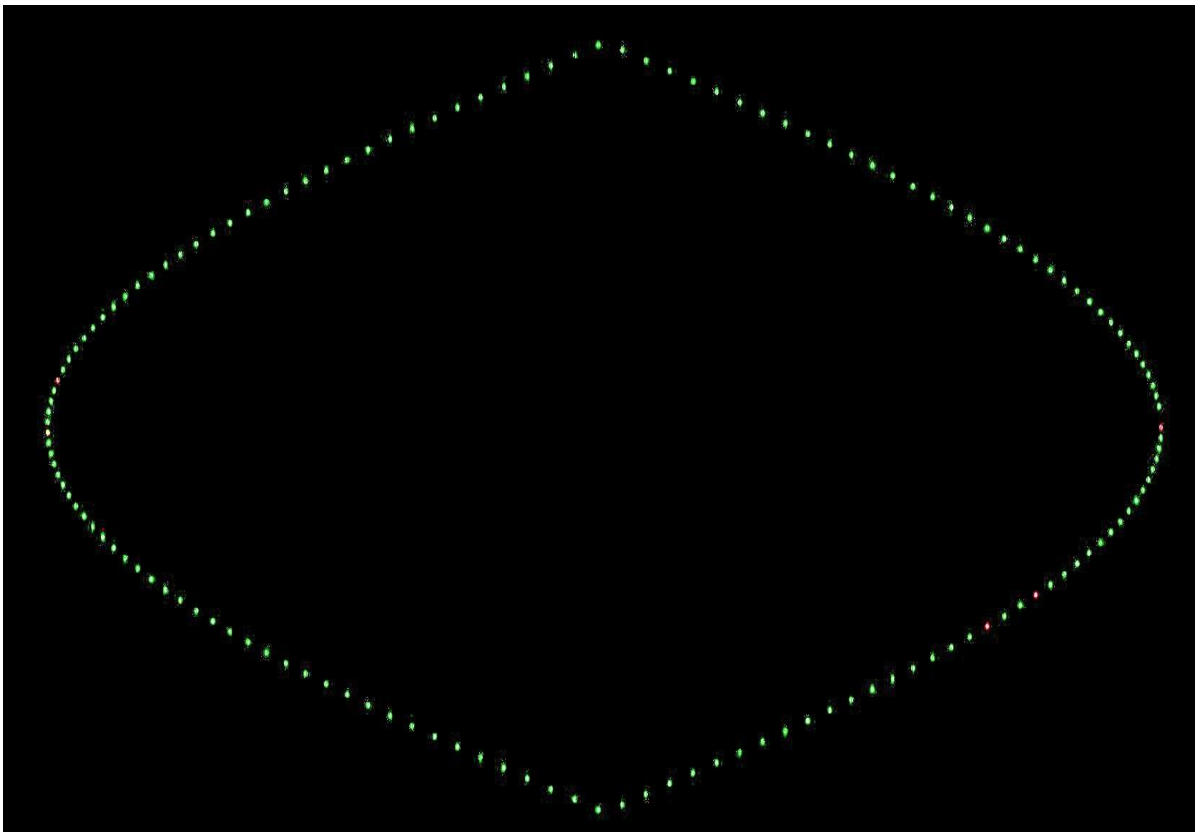
DDoS επίθεση πριν την καταπολέμηση

Εικόνα 19

Όπως παρατηρούμε στην παραπάνω εικόνα (εικόνα 34) ο κόμβος που βρίσκεται κάτω δεξιά στην εικόνα δέχεται τεράστιες ποσότητες δεδομένων με αποτέλεσμα να μην μπορει να εξυπηρετίσει κάποιον καλοβουλο χρήστη που θα αιτηθεί κάποια υπηρεσία του.

Αυτή είναι η εικόνα του δικτύου πριν κάνουμε οποιαδήποτε προσπάθεια καταπολέμισης της απειλής που έχει μολύνει το δίκτυο.

Στην συνέχεια παραθέτουμε την εικόνα 35 η οποία αντιπροσωπεύει την εικόνα του δικτύου μετα το τέλος της προσομοίωσης όπου έχει καταπολεμηθεί αποδοτικά η απειλή που στόχο είχε την μόλυνση με κακόβουλη πληροφορία του δικτύου.



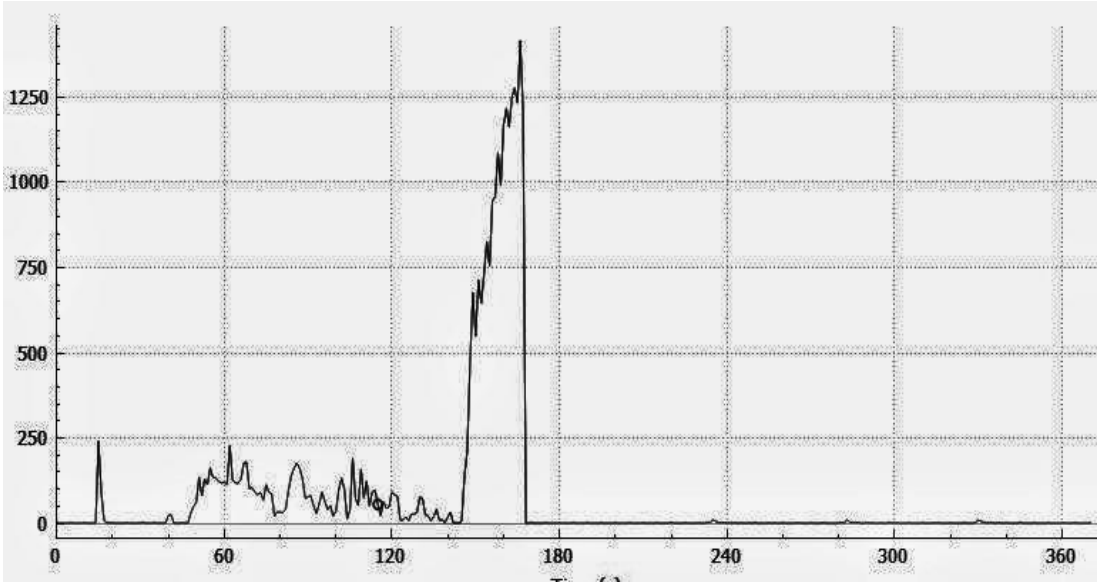
DDoS επίθεση μετά την καταπολέμηση

Εικόνα 20

Όπως παρατηρούμε από την παραπάνω εικόνα η απειλή έχει καταπολεμηθεί πλήρως καθώς έχει διακοπή όλη η κακόβουλη πληροφορία. Συγκρίνοντας εκ των υστέρων μάλιστα

τις εικόνες 34 και 35 παρατηρούμε ότι το σύστημα μας κατέστειλε αποδοτικά την κακόβουλη πληροφορία και διατήρησε το δίκτυο αμόλυντο και λειτουργικό.

Για να κατανοήσουμε ακόμα περισσότερο το αποτέλεσμα της καταστολής της κακόβουλης πληροφορίας από το σύστημα που αναπτύξαμε στα πλαίσια αυτής της διπλωματικής παρουσιάζουμε στην εικόνα 36 το διάγραμμα της κίνησης που προέκυψε μετά την καταστολή της κακόβουλης κίνησης δεδομένων.

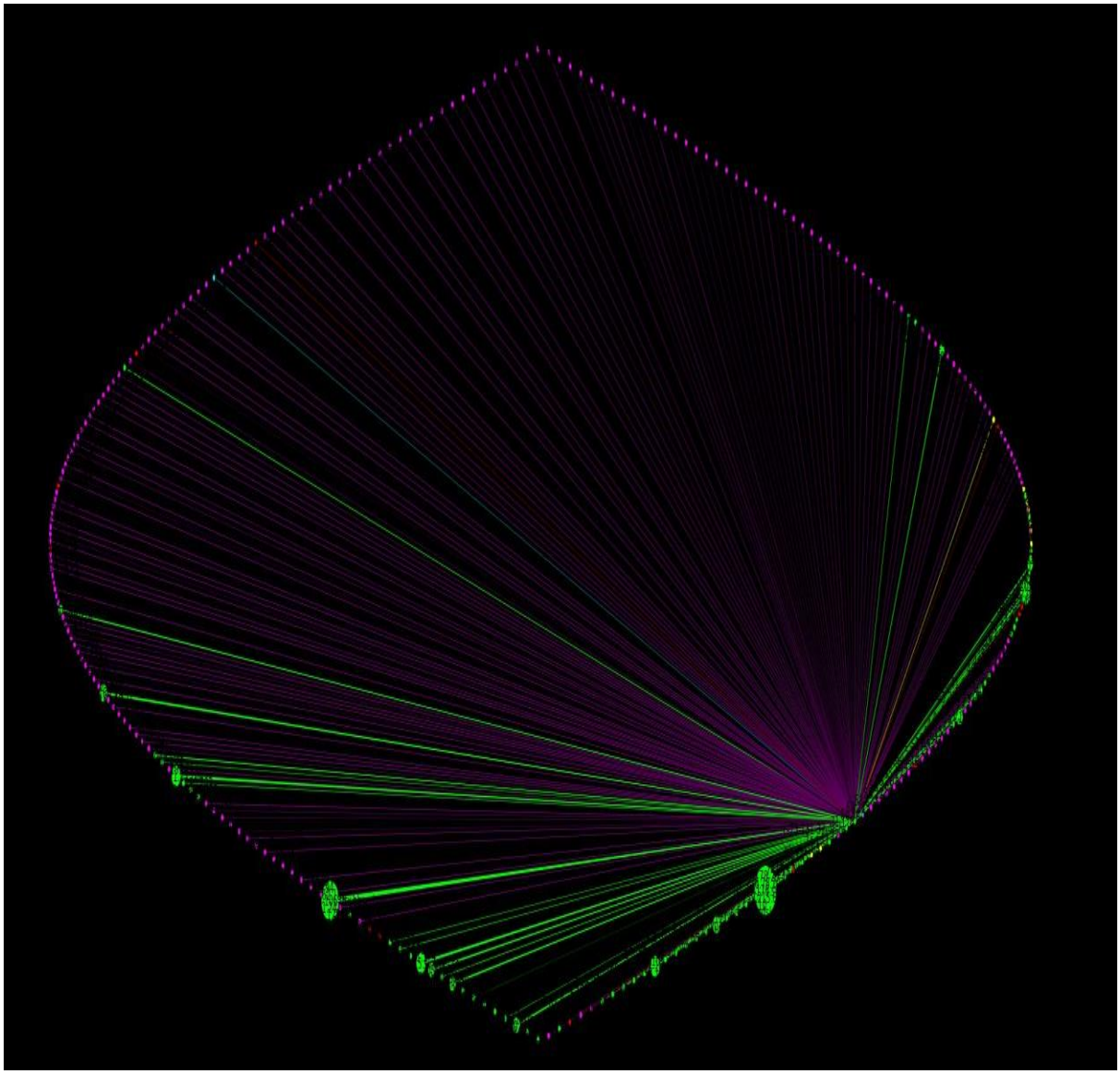


DDoS Επίθεση μετά την καταπολέμηση γράφημα

Εικόνα 21

Παρατηρώντας το διάγραμμα μπορούμε να συμπεράνουμε πως η απόρριψη της κακόβουλης πληροφορίας από το σύστημα μας ήταν άμεση και άκρως αποδοτική καθώς έχουμε μια κατακόρυφη μείωση της κίνησης όταν καταπολεμήθηκε πλήρως η απειλή και διατηρήσαμε το δίκτυο μας ανέπαφο και πλήρως λειτουργικό.

Στην συνέχεια θα παραθέσουμε δύο εικόνες που προκύπτουν από την εκτέλεση του πειράματος και αποδεικνύουν ότι όχι μόνο το σύστημα μας καταπολεμά την κακόβουλη πληροφορία αλλά και επιτρέπει στους καλόβουλους χρήστες να εξυπηρετούνται κανονικά από το δίκτυο χωρίς να θέτει το σύστημα μας εκτός λειτουργίας πράγμα που αποδεικνύει το δυναμικό χαρακτήρα των ευφυών δικτύων.

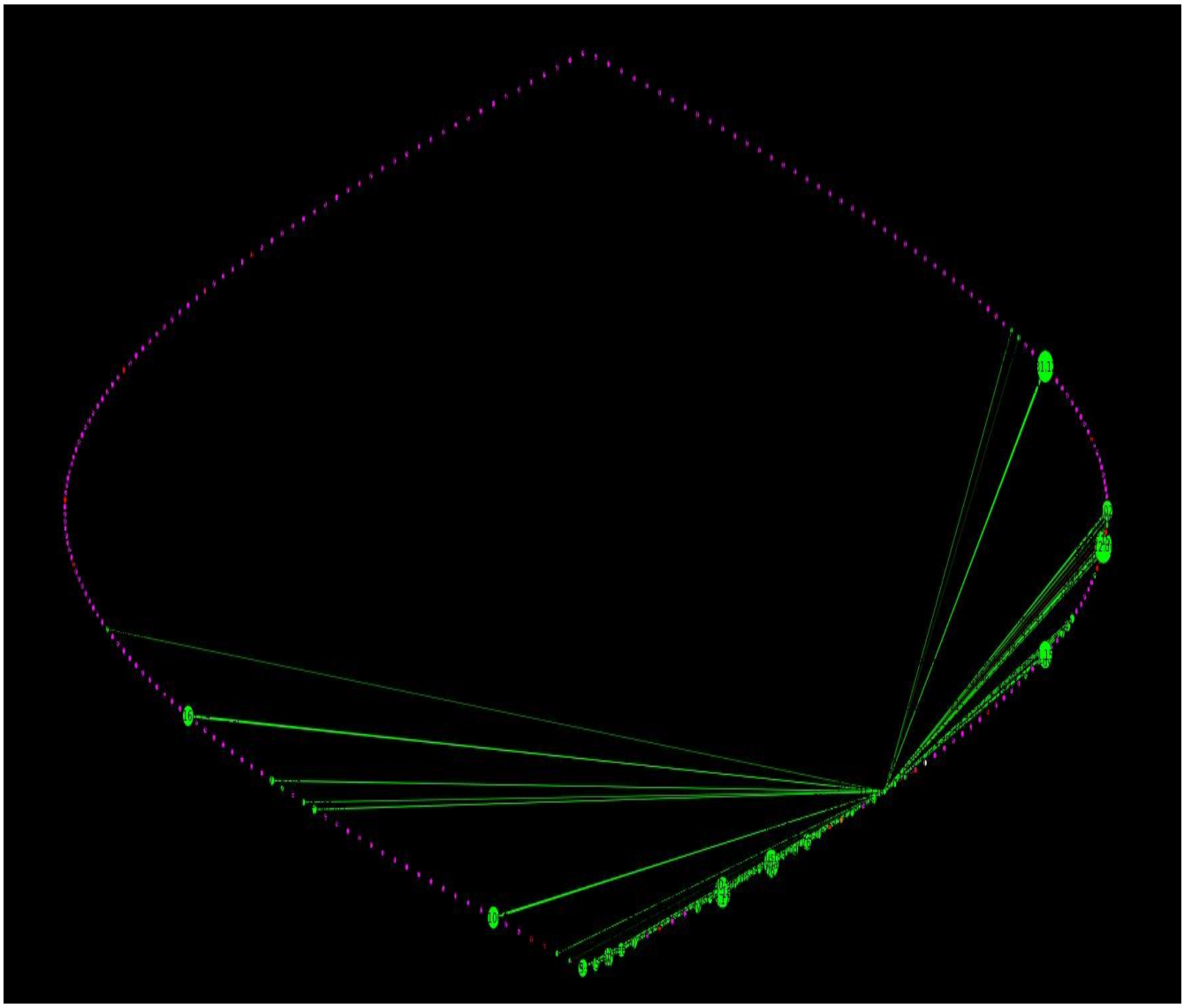


Εικόνα Δικτύου μεικτής κίνησης (καλόβουλη , κακόβουλη) πριν την καταπολέμηση

Εικόνα 37

Στην εικόνα 37 παρουσιάζονται τα δεδομένα που διακινούνται στο εσωτερικό δικτύου πριν την καταπολέμηση μιας οποιαδήποτε απειλής. Το ιδιαίτερο με την παραπάνω εικόνα είναι ότι στο δίκτυο δεν διακινείται μόνο κακόβουλη DDoS κίνηση αλλά και καλόβουλη. Πιο συγκεκριμένα με το μωβ χρώμα απεικονίζεται η κακόβουλη κίνηση δεδομένων ενώ με το πράσινο χρώμα απεικονίζεται η καλόβουλη κίνηση δεδομένων. Στόχος του συστήματος που έχουμε κατασκευάσει είναι να καταφέρει να καταπολεμήσει την μωβ κίνηση ώστε να την αποβάλλει από το δίκτυο καθώς είναι κακόβουλη DDoS κίνηση δεδομένων και παράλληλα να διατηρήσει την πράσινη καλόβουλη κίνηση. Όπως θα παρατηρήσουμε στην εικόνα 38 το σύστημα μας κατάφερε να διατηρήσει την καλόβουλη πράσινη κίνηση και να

καταπολεμήσει αποτελεσματικά την μωβ κακόβουλη κίνηση. Έτσι αποδεικνύεται ότι η με τα ευφυή δίκτυα και το νέο αυτόν τρόπο καταπολέμησης των δικτυακών επιθέσεων οι καλόβουλοι χρήστες μένουν ανεπηρέαστοι και το δίκτυο μας μπορεί να προσφέρει τις υπηρεσίες του χωρίς να παρουσιάζει σημάδια υπολειτουργίας λόγω δικτυακών απειλών.



Εικόνα Δικτύου μεικτής κίνησης (καλόβουλη , κακόβουλη) μετά την καταπολέμηση

Εικόνα 38

Στην συνέχεια θα παρουσιάσουμε την εικόνα στο εσωτερικό της συσκευή δικτύου Switch 1 κατά την διάρκεια της καταπολέμησης της DDoS επίθεσης. Πιο συγκεκριμένα η εικόνα 39 παρουσιάζει τους κανόνες που έχουν περαστεί από την μονάδα ελέγχου Opendaylight και είναι υπεύθυνοι για την αποκοπή της DDoS κακόβουλης κίνησης δεδομένων. Παρατηρούμε ότι οι κανόνες που έχουν περαστεί για να αποκόψουν την κακόβουλη κίνηση έχουν στο πεδίο ταυτοποίησης τις διευθύνσεις πηγής και προορισμού των πακέτων που αποκόπτουν ενώ έχουν μεγαλύτερη προτεραιότητα από κανόνες που επιτρέπουν την διακίνηση των κακόβουλων πακέτων. Επίσης παρατηρούμε ότι στο πεδίο δράσεων αυτών των κανόνων υπάρχει η δράση drop που υποδηλώνει την εντολή απόρριψης κακόβουλων πακέτων. Στην εικόνα 39 παρατηρούμε επίσης ότι το σημείο των κανόνων που υποδηλώνει τον αριθμό των πακέτων που έχουν ταυτοποιηθεί από τους Openflow κανόνες αυτούς είναι μη μηδενικό που σημαίνει ότι αποκόπτει την κακόβουλη κίνηση.

```
cookie=0x37, duration=1.477s, table=0, n_packets=0, n_bytes=0, idle_age=1, priority=99,ip,nw_src=71.126.222.64,nw_dst=202.55.222.13 actions=drop
cookie=0x37, duration=11.758s, table=0, n_packets=3, n_bytes=446, idle_age=11, priority=99,ip,nw_src=194.38.235.134,nw_dst=71.126.222.64 actions=drop
cookie=0x37, duration=9.479s, table=0, n_packets=20, n_bytes=2010, idle_age=1, priority=99,ip,nw_src=71.126.222.64,nw_dst=197.104.126.145 actions=drop
cookie=0x37, duration=6.975s, table=0, n_packets=11, n_bytes=9003, idle_age=4, priority=99,ip,nw_src=194.228.237.162,nw_dst=71.126.222.64 actions=drop
cookie=0x37, duration=9.151s, table=0, n_packets=8, n_bytes=528, idle_age=8, priority=99,ip,nw_src=71.126.222.64,nw_dst=167.88.50.246 actions=drop
cookie=0x37, duration=12.219s, table=0, n_packets=0, n_bytes=0, idle_age=12, priority=99,ip,nw_src=33.181.67.169,nw_dst=71.126.222.64 actions=drop
cookie=0x37, duration=6.355s, table=0, n_packets=0, n_bytes=0, idle_age=6, priority=99,ip,nw_src=71.126.222.64,nw_dst=39.234.88.52 actions=drop
cookie=0x37, duration=3.802s, table=0, n_packets=0, n_bytes=0, idle_age=3, priority=99,ip,nw_src=71.126.222.64,nw_dst=204.69.217.55 actions=drop
cookie=0x37, duration=3.509s, table=0, n_packets=2, n_bytes=132, idle_age=2, priority=99,ip,nw_src=51.76.189.229,nw_dst=71.126.222.64 actions=drop
cookie=0x37, duration=12.869s, table=0, n_packets=25, n_bytes=31340, idle_age=12, priority=99,ip,nw_src=192.173.18.27,nw_dst=71.126.222.64 actions=drop
cookie=0x37, duration=15.17s, table=0, n_packets=7, n_bytes=623, idle_age=15, priority=99,ip,nw_src=71.126.222.64,nw_dst=200.2.135.53 actions=drop
cookie=0x37, duration=48.823s, table=0, n_packets=3151, n_bytes=229622, idle_age=0, priority=99,ip,nw_src=192.95.27.190,nw_dst=71.126.222.64 actions=drop
cookie=0x37, duration=2.881s, table=0, n_packets=0, n_bytes=0, idle_age=2, priority=99,ip,nw_src=202.54.15.22,nw_dst=71.126.222.64 actions=drop
cookie=0x37, duration=13.352s, table=0, n_packets=167, n_bytes=19636, idle_age=3, priority=99,ip,nw_src=71.126.222.64,nw_dst=36.86.181.219 actions=drop
cookie=0x37, duration=47.685s, table=0, n_packets=447, n_bytes=33078, idle_age=0, priority=99,ip,nw_src=51.81.166.201,nw_dst=71.126.222.64 actions=drop
cookie=0x37, duration=48.405s, table=0, n_packets=2237, n_bytes=165538, idle_age=0, priority=99,ip,nw_src=51.173.229.255,nw_dst=71.126.222.64 actions=drop
cookie=0x37, duration=47.858s, table=0, n_packets=2071, n_bytes=153254, idle_age=0, priority=99,ip,nw_src=40.75.89.172,nw_dst=71.126.222.64 actions=drop
cookie=0x37, duration=48.213s, table=0, n_packets=2141, n_bytes=158434, idle_age=0, priority=99,ip,nw_src=202.1.175.252,nw_dst=71.126.222.64 actions=drop
cookie=0x37, duration=8.85s, table=0, n_packets=9, n_bytes=1376, idle_age=5, priority=99,ip,nw_src=51.76.189.155,nw_dst=71.126.222.64 actions=drop
cookie=0x37, duration=8.573s, table=0, n_packets=11, n_bytes=889, idle_age=5, priority=99,ip,nw_src=71.126.222.64,nw_dst=51.76.189.155 actions=drop
cookie=0x37, duration=13.072s, table=0, n_packets=365, n_bytes=381133, idle_age=0, priority=99,ip,nw_src=41.51.84.181,nw_dst=71.126.222.64 actions=drop
cookie=0x37, duration=8.247s, table=0, n_packets=17, n_bytes=11193, idle_age=3, priority=99,ip,nw_src=52.127.3.234,nw_dst=71.126.222.64 actions=drop
cookie=0x37, duration=45.571s, table=0, n_packets=33, n_bytes=3366, idle_age=1, priority=99,ip,nw_src=71.126.222.64,nw_dst=40.75.89.172 actions=drop
cookie=0x37, duration=11.335s, table=0, n_packets=9, n_bytes=1668, idle_age=10, priority=99,ip,nw_src=198.169.186.197,nw_dst=71.126.222.64 actions=drop
cookie=0x37, duration=46.05s, table=0, n_packets=33, n_bytes=3366, idle_age=1, priority=99,ip,nw_src=71.126.222.64,nw_dst=51.81.166.201 actions=drop
cookie=0x37, duration=7.934s, table=0, n_packets=12, n_bytes=1536, idle_age=3, priority=99,ip,nw_src=71.126.222.64,nw_dst=52.127.3.234 actions=drop
cookie=0x37, duration=10.957s, table=0, n_packets=15, n_bytes=1869, idle_age=2, priority=99,ip,nw_src=40.66.79.244,nw_dst=71.126.222.64 actions=drop
cookie=0x37, duration=0.568s, table=0, n_packets=0, n_bytes=0, idle_age=0, priority=99,ip,nw_src=71.126.222.64,nw_dst=193.142.3.52 actions=drop
cookie=0x37, duration=1.796s, table=0, n_packets=0, n_bytes=0, idle_age=1, priority=99,ip,nw_src=71.126.222.64,nw_dst=202.54.15.27 actions=drop
cookie=0x37, duration=15.341s, table=0, n_packets=8, n_bytes=1048, idle_age=15, priority=99,ip,nw_src=200.2.135.53,nw_dst=71.126.222.64 actions=drop
cookie=0x37, duration=32.856s, table=0, n_packets=0, n_bytes=0, idle_age=32, priority=99,ip,nw_src=71.126.222.64,nw_dst=194.47.79.199 actions=drop
cookie=0x37, duration=10.495s, table=0, n_packets=13, n_bytes=1926, idle_age=9, priority=99,ip,nw_src=146.125.144.201,nw_dst=71.126.222.64 actions=drop
cookie=0x37, duration=46.326s, table=0, n_packets=33, n_bytes=3366, idle_age=1, priority=99,ip,nw_src=71.126.222.64,nw_dst=192.120.148.227 actions=drop
cookie=0x37, duration=46.727s, table=0, n_packets=34, n_bytes=3468, idle_age=2, priority=99,ip,nw_src=71.126.222.64,nw_dst=195.198.120.238 actions=drop
cookie=0x37, duration=2.015s, table=0, n_packets=0, n_bytes=0, idle_age=2, priority=99,ip,nw_src=202.54.15.27,nw_dst=71.126.222.64 actions=drop
cookie=0x37, duration=48.684s, table=0, n_packets=2891, n_bytes=213990, idle_age=0, priority=99,ip,nw_src=192.120.148.227,nw_dst=71.126.222.64 actions=drop
cookie=0x37, duration=12.656s, table=0, n_packets=6, n_bytes=1092, idle_age=9, priority=99,ip,nw_src=39.172.224.50,nw_dst=71.126.222.64 actions=drop
cookie=0x37, duration=6.021s, table=0, n_packets=1, n_bytes=66, idle_age=5, priority=99,ip,nw_src=197.11.92.4,nw_dst=71.126.222.64 actions=drop
```

Openflow κανόνες στην συσκευή δικτύου κατά την διάρκεια καταπολέμησης της επίθεσης

Εικόνα 39

Στην συνέχεια θα παρουσιάσουμε και θα εξηγήσουμε το xml αρχείο το οποίο στέλνουμε με post Http request στην Opendaylight μονάδα ελέγχου και περιέχει τις πληροφορίες ώστε να δώσει εντολή στην πλατφόρμα Opendaylight για να περάσει αυτή με την σειρά της τον κατάλληλο Openflow κανόνα στην συσκευή δικτύου switch 1 ώστε να αποκοπεί η κακόβουλη κίνηση δεδομένων που απειλεί το δίκτυο.

```
<?xml version='1.0' encoding='UTF-8' standalone='no'?>
<input xmlns="urn:opendaylight:flow:service">

  <node
xmlns:inv="urn:opendaylight:inventory">/inv:nodes/inv:node[inv:id="openflow:1"]</node>

  <cookie>55</cookie>
  <hard-timeout>0</hard-timeout>
  <idle-timeout>0</idle-timeout>
  <match>
   <ethernet-match>
    <ethernet-type>
     <type>2048</type>
    </ethernet-type>
   </ethernet-match>
   <ipv4-source>71.126.222.64/32</ipv4-source>
   <ipv4-destination>54.210.215.136/32</ipv4-destination>
  </match>
  <priority>99</priority>
  <table_id>0</table_id>
</input>
```

Post Http Request XML

Εικόνα 40

Όπως παρατηρούμε στην παραπάνω εικόνα στο xml περιέχονται στο πεδίο ταυτοποίησης(match) οι διευθύνσεις πηγής και προορισμού της κακόβουλης κίνησης που θέλουμε να αποκόψουμε. Επίσης παρατηρούμε ότι δεν έχει οριστεί κάποια δράση και αυτό γιατί η πλατφόρμα ελέγχου Opendaylight όταν δέχεται αυτό το xml στο οποίο δεν έχει οριστεί κάποια δράση αντιλαμβάνεται ότι θα περάσει Openflow κανόνα στην συσκευή δικτύου switch 1 ο οποίος κανόνας θα έχει ως δράση την παράμετρο drop για να απορρίπτει την κακόβουλη κίνηση που διακινείται στο δίκτυο. Επίσης έχουμε ορίσει κάποιες ενδεικτικές τιμές στα υπόλοιπα πεδία του xml αρχείου για χάριν της προσομοίωσης τα οποία ο εκάστοτε χρήστης μπορεί να τα ορίσει κατά το δοκούν.

Το συγκεκριμένο xml αρχείο το στέλνουμε στην πλατφόρμα ελέγχου Opendaylight μέσω POST Http request στο URL 'http://192.168.1.155:8181/restconf/operations/sal-flow:add-flow' στο οποίο έχουμε ορίσει την IP διεύθυνση και την πόρτα στην οποία ακούει η μονάδα ελέγχου Opendaylight όπως και επίσης την υπηρεσία που είναι υπεύθυνη για να περαστεί ο καινούργιος Openflow κανόνας στην συσκευή δικτύου.

Άρα από όλα τα παραπάνω καταλήγουμε πως η καταπολέμηση απειλών και κακόβουλων δεδομένων γίνεται ιδιαίτερα επιτυχημένα σε ευφυή δίκτυα καθώς η πρωτοποριακή αρχιτεκτονική τους μας δίνει την δυνατότητα να έχουμε μια γενική και ξεκάθαρη εικόνα του δικτύου με αποτέλεσμα να μπορούμε κάθε στιγμή να παίρνουμε αποφάσεις για να αντιμετωπίζουμε καταστάσεις που δημιουργούνται στο δίκτυο έγκαιρα.

Ακόμα πιο σημαντικό είναι ότι για να αντιμετωπίσουμε μια απειλή ή για να παραμετροποιήσουμε κάποια στοιχεία του δικτύου δεν είναι απαραίτητο να θέτουμε συνεχώς το δίκτυο εκτός λειτουργίας αλλά μπορούμε να εφαρμόσουμε οποιαδήποτε αλλαγή θέλουμε δίνοντας εντολές στην εκάστοτε πλατφόρμα ελέγχου και στην προκειμένη περίπτωση στην πλατφόρμα ελέγχου ευφυών δικτύων Opendaylight (Opendaylight controller).

8 Συμπεράσματα και Μελλοντικές επεκτάσεις

Στην συνέχεια θα αναφερθούμε σε πιθανές βελτιώσεις και μελλοντικές προεκτάσεις που μπορούν να γίνουν σαν συνέχεια αυτής της διπλωματικής εργασίας.

Στα πλαίσια περαιτέρω ανάπτυξης του θέματος της συγκεκριμένης διπλωματικής εργασίας μπορούν να γίνουν βελτιώσεις όσον αφορά τα είδη των δικτυακών επιθέσεων που μπορούν να αντιμετωπιστούν με βάση το παραπάνω σύστημα. Επίσης μπορούμε να προσθέσουμε κάποια βάση δεδομένων στην οποία θα αποθηκεύονται οι πληροφορίες που παράγονται από το Snort IDS πρόγραμμα.

Επιπροσθέτως μπορούμε να χρησιμοποιήσουμε εκτός από το Snort και άλλα προγράμματα που υπάρχουν στο εμπόριο και μπορούν να ανιχνεύσουν και να καταπολεμήσουν παράλληλα δικτυακές επιθέσεις οι οποίες περιέχουν μεγάλο όγκο δεδομένων. Επίσης ως μελλοντική έρευνα μπορεί κάποιος να χρησιμοποιήσει και προγράμματα τα οποία υπάρχουν στο εμπόριο και τα οποία δεν ανιχνεύουν κακόβουλη πληροφορία με κανόνες όπως το Snort IDS πρόγραμμα αλλά κάνουν αναγνώριση μέσω στοχαστικών δεδομένων συγκρίνοντας πληροφορίες οι οποίες αντιπροσωπεύουν την εικόνα του δικτύου πριν την επίθεση και πληροφορίες οι οποίες αντιπροσωπεύουν την εικόνα του ίδιου δικτύου μετά από μία επίθεση. Έτσι μέσω αυτής της στοχαστικής ανάλυσης μπορούν τα προγράμματα αυτά να αναγνωρίσουν την πληροφορία που απειλή ένα δίκτυο και στην προκειμένη περίπτωση το δίκτυο που προστατεύουν.

Τέλος μία ακόμα μελλοντική επέκταση της παρούσας διπλωματικής είναι ότι όταν ανιχνεύσουμε και καταπολεμήσουμε μια απειλή στην συνέχεια οι πλατφόρμες ελέγχου των ευφυών δικτύων μπορούν να ανταλλάσσουν μηνύματα μεταξύ τους ώστε να είναι ενήμερες σχετικά με κάποια επίθεση που έγινε και επίσης να πληροφορούνται τα στοιχεία του χρήστη ή των χρηστών που προσπάθησαν να μολύνουν το δίκτυο με κακόβουλη πληροφορία και αυτόματα να λαμβάνουν μέτρα σχετικά με τον αποκλεισμό του κακόβουλου χρήστη ή των κακόβουλων χρηστών ώστε να μην καταφέρουν να μολύνουν και τα δικά τους ευφυή δίκτυα στο μέλλον και να προλάβουν να αποφύγουν την απειλή των δικτύων τους πριν ακόμα αυτή ξεκινήσει.

Βιβλιογραφία

[1]. **sdn.** https://www.sdxcentral.com/sdn/resources/what-the-definition-of-software-defined-networking-sdn/.

[2]. **Openflow.** http://archive.openflow.org/.

[3]. **spec, openflow.** http://archive.openflow.org/wp/documents/.

[4]. **NOX.** http://thenewstack.io/sdn-series-part-iii-nox-the-original-openflow-controller/.

[5]. **Beacon.** https://www.sdxcentral.com/projects/beacon/.

[6]. **Floodlight.** https://www.sdxcentral.com/projects/floodlight/.

[7]. **Opendaylight.** https://www.opendaylight.org/.

[8]. **Opencontrail.** https://www.sdxcentral.com/projects/opencontrail/.

[9]. **ONOS.** http://onosproject.org/.

[10]. **Ryu.** https://osrg.github.io/ryu/.

[11]. **wiki, Opendaylight.** https://wiki.opendaylight.org/view/Main_Page.

[12]. **MD-SAL, Opendaylight.** https://wiki.opendaylight.org/view/OpenDaylight_Controller:MD-SAL:MD-SAL_App_Tutorial.

[13]. **Yang.** https://tools.ietf.org/html/rfc6020.

[14]. **InstanceIdentifiers.** http://sdnhub.org/.

[15]. **RESTCONF.** https://tools.ietf.org/html/draft-ietf-netconf-restconf-05.

[16]. **Maven, Apache.** https://maven.apache.org/.

[17]. **OSGi.** https://www.osgi.org/.

[18]. **Equinox.** http://www.eclipse.org/equinox/.

[19]. **Felix, Apache.** http://felix.apache.org/.

[20]. **Karaf, Apache.** https://karaf.apache.org/manual/latest/overview.html.

[21]. **Snort.** https://www.snort.org/.

[22]. **Manual, Snort.** https://s3.amazonaws.com/snort-org-site/production/document_files/files/000/000/100/original/snort_manual.pdf?AWSAccessKeyId=AKIAIXACIED2SPMSC7GA&Expires=1465666479&Signature=wXFRgWskjve5thXBIfOo%2BYtoVlU%3D.

[23]. **FortNOX.** https://opencourses.uoc.gr/courses/pluginfile.php/13507/mod_resource/content/2/7.%20sdnsec-lec7.pdf.

[24]. **FRESCO.** http://www.internetsociety.org/sites/default/files/Presentation07_2.pdf.

[25]. **NETMODE.** http://www.sciencedirect.com/science/article/pii/S1389128613004003.

[26]. **IPS, Suricata IDS.** https://suricata-ids.org/.

[27]. **IDS, BRO.** https://www.bro.org/.

[28]. **OpenWIPS-ng.** http://www.openwips-ng.org/.

[29]. **project, l2switch.** https://github.com/opendaylight/l2switch.

[30]. **Mininet.** http://mininet.org/.