2023

ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΙΓΑΙΟΥ – ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ
Τμήμα Μηχανικών Πληροφοριακών και Επικοινωνιακών Συστημάτων

Διπλωματική Εργασία

με θέμα:

ΑΝΙΧΝΕΥΣΗ ΚΑΚΟΒΟΥΛΩΝ ΕΠΙΘΕΣΕΩΝ ΜΕ ΤΗ ΧΡΗΣΗ ΤΕΧΝΙΚΩΝ ΜΗΧΑΝΙΚΗΣ ΜΑΘΗΣΗΣ ΚΑΙ ΝΕΥΡΩΝΙΚΩΝ ΔΙΚΤΥΩΝ

Των φοιτητών:

Βλάσση Νικόλαο ΑΜ: 3232019005

Καμάρα Βλάσιο ΑΜ: 3232019014

Επιβλέπουσα Καθηγήτρια:

Κωνσταντία Μπαρμπάτσαλου

Περιεχόμενα

Α. Εισαγωγή

Η παρούσα διπλωματική εργασία αποτελεί μια πρώτη προσπάθεια που κάνουμε να ερευνήσουμε το πως μπορούμε να βελτιώσουμε την υπάρχουσα ασφάλεια δικτύου χρησιμοποιώντας μεθόδους τεχνητής νοημοσύνης. Πιο συγκεκριμένα, επικεντρωθήκαμε στο κομμάτι των Intrusion Detection Systems, δηλαδή των συστημάτων που είναι υπεύθυνα για την ανίχνευση κακόβουλης κίνησης σε ένα δίκτυο. Στα πλαίσια της εργασίας, δημιουργήσαμε δυο διαφορετικά μοντέλα τεχνητής νοημοσύνης τα οποία λειτουργούν σαν IDS, τα οποία εκπαιδεύσαμε με βάση δεδομένα που είναι διαθέσιμα στο διαδίκτυο ενώ στο τέλος μετρήσαμε την αποτελεσματικότητα τους. Δημιουργήσαμε επίσης, χρησιμοποιώντας υπάρχοντα εργαλεία, ένα περιβάλλον το οποίο θα μπορεί να συλλέγει και να επεξεργάζεται δικτυακή κίνηση σε πραγματικό χρόνο την οποία θα μεταφέρει στο IDS για κατηγοριοποίηση.

Για να γίνουν εφικτά τα παραπάνω, αξιοποιήσαμε τις γνώσεις που αποκτήσαμε σχετικά με τα IDS από το Μάθημα της κυρίας Μπαρμπάτσαλου «Ψηφιακή Εγκληματολογία», τις οποίες και ενισχύσαμε διαβάζοντας επιπλέον βιβλιογραφία. Διαβάσαμε επίσης σχετική βιβλιογραφία με την τεχνητή νοημοσύνη εστιάζοντας σε προσπάθειες που έχουν γίνει για την αξιοποίηση της στον κλάδο της ασφάλειας πληροφοριακών και επικοινωνιακών συστημάτων. Τέλος, προσπαθήσαμε να βελτιώσουμε τις γνώσεις και τις ικανότητες μας πάνω στα εργαλεία τα οποία χρησιμοποιήσαμε για να υλοποιήσουμε την εργασία. Ορισμένα από αυτά είναι το elasticsearch, το Proxmox VE και η Python, τις βιβλιοθήκες της οποίας αξιοποιήσαμε για να γράψουμε τον κώδικα για το κομμάτι της τεχνητής νοημοσύνης. Αυτή τη πορεία της μελέτης μας ακολουθεί και η δομή του κειμένου.

Στο κεφάλαιο Β παρουσιάζονται γενικές πληροφορίες για τα IDS, τα χαρακτηριστικά τους και τους διαφορετικούς τύπους που υπάρχουν. Παρουσιάζουμε επίσης τα πλεονεκτήματα και τα μειονεκτήματα του κάθε τύπου. Στο κεφάλαιο Γ γίνεται μια εισαγωγή στην τεχνητή νοημοσύνη, στα βήματα που έχουν γίνει τα τελευταία χρόνια, στις διάφορες χρήσεις τις. Αναφέρονται επίσης οι βασικοί τύποι αλγορίθμων μηχανικής μάθησης. Στο κεφάλαιο Δ επικεντρώνουμε στην σημαντικότερη βιβλιογραφία που μελετήσαμε και αφορά την χρήση της μηχανικής μάθησης σε συστήματα IDS. Στο κεφάλαιο Ε αναλύουμε την υλοποίηση μας. Αρχικά αναλύουμε το real time περιβάλλον που δημιουργήσαμε και στη συνέχεια περιγράφουμε την διαδικασία και το σκεπτικό με το οποίο δημιουργήσαμε τα δύο μοντέλα τεχνητής νοημοσύνης. Παρουσιάζονται επίσης τα αποτελέσματα του κάθε μοντέλου. Τέλος, επειδή η προσπάθεια που κάναμε δε θέλουμε να περιοριστεί μόνο στα πλαίσια αυτής της διπλωματικής εργασίας, παρουσιάζουμε στο κεφάλαιο ΣΤ τα βήματα που θέλουμε να κάνουμε στο μέλλον για να βελτιώσουμε την υπάρχουσα υποδομή μαζί με τα συμπεράσματα που βγάλαμε.

B. Intrusion Detection Systems

Θα ξεκινήσουμε αυτό το κεφάλαιο περιγράφοντας τις μεθόδους Intrusion Detection.

B.1.1. Μέθοδοι Intrusion Detection

Signature Based Intrusion Detection: Τα SIDS λειτουργούν με βάση "signatures" γνωστών επιθέσεων, τα οποία διατηρούνται σε μία βάση δεδομένων και αν εντοπιστεί κάποιο από αυτά στο δίκτυο μας τότε παράγεται alarm. Σε αυτού του είδους το intrusion detection, το βασικό κομμάτι της διαδικασίας είναι η εισαγωγή των νέων "signatures", ώστε να μην εμφανίζονται κενά ασφαλείας στα συστήματα μας. Το κύριο πρόβλημα με τα SIDS είναι το γεγονός πως δεν μπορούν να εντοπίσουν τα λεγόμενα "zero days" αφού δεν υπάρχουν στην βάση τους τα αντίστοιχα "signatures".

Anomaly Based Intrusion Detection: Αυτά τα IDS λειτουργούν συνήθως με βάση κάποιο machine learning αλγόριθμο. Ο αλγόριθμος μαθαίνει από τη φυσιολογική κίνηση σε ένα δίκτυο και έτσι μπορεί να παράγει alarm κάθε φορά που εντοπίζεται μία ασυνήθιστη δραστηριότητα στο δίκτυο. Η χρήση τους είναι σε πιο αρχικό στάδιο σε σχέση με τα signature based, όμως διάφοροι ερευνητές προσδοκούν αρκετά από αυτά με σκοπό την αντιμετώπιση zero day attacks.

Hybrid Intrusion Detection: Πρόκειται για τον συνδυασμό των δυο παραπάνω μεθόδων.

Τα IDS διαχωρίζονται επίσης και από το σημείο που είναι τοποθετημένα στο δίκτυο αλλά και το εύρος της κίνησης που ελέγχουν. Με βάση αυτά διακρίνουμε 5 κατηγορίες.

B.1.2. Κατηγορίες IDS

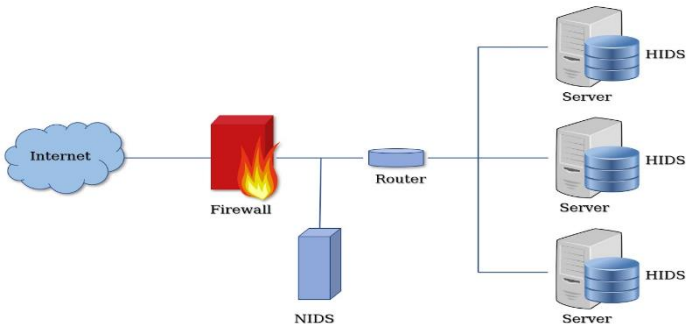
Network Intrusion Detection System (NIDS): Τα NIDS είναι εγκατεστημένα μέσα στο δίκτυο με τέτοιο τρόπο που μπορούν να παρακολουθούν το σύνολο του traffic, χωρίς αυτό να επηρεάζει τις επιδόσεις του δικτύου. Αυτό το καταφέρνουν αφού τα NIDS δεν προσθέτουν traffic στο δίκτυο, απλά ακούν το traffic που έχει το κανάλι και έτσι μπορούν να ελέγχουν τα πακέτα που εισέρχονται και εξέρχονται από το δίκτυο και να τα φιλτράρουν για κακόβουλο traffic.

<u>Τα πλεονεκτήματα των NIDS είναι τα παρακάτω:</u>

- Ένα NIDS που έχει τοποθετηθεί σωστά, μπορεί να παρακολουθήσει ένα πολύ μεγάλο δίκτυο
- Απλά ακούει το δίκτυο, αυτό σημαίνει ότι δεν το επιβαρύνει με επιπλέον φόρτο.
- Παρακολουθεί σε πραγματικό χρόνο τα δεδομένα στο δίκτυο, έτσι έχει την δυνατότητα να εντοπίζει τις επιθέσεις την στιγμή που γίνονται.

<u>Παρουσιάζουν και ορισμένα μειονεκτήματα</u>

- Όσο αυξάνεται το traffic η διαδικασία της ανάλυσης δυσκολεύει.
- Δεν μπορεί να εντοπίσει επιθέσεις αν τα πακέτα είναι κρυπτογραφημένα.



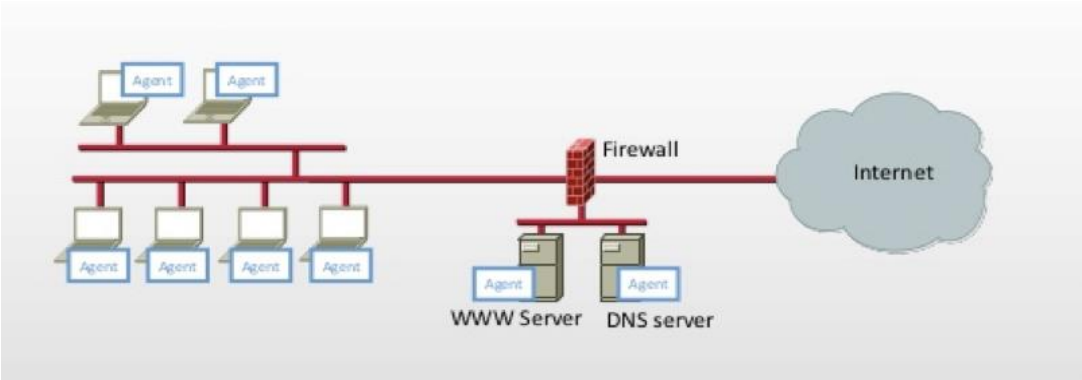
Host Intrusion Detection System (HIDS): Είναι ένα σύστημα που είναι εγκατεστημένο στον υπολογιστή που προστατεύει και είναι ρυθμισμένο έτσι ώστε να ανιχνεύει κακόβουλο traffic που σχετίζεται με system calls, log files καθώς και αλλαγές που γίνονται στο ίδιο το σύστημα. Για παράδειγμα μπορεί να ανιχνεύσει ένα brute force στον χρήστη του συστήματος, κάποιο privilege escalation, ή ένα malware που προσπαθεί να αλλάξει την registry. Το γεγονός πως παρακολουθούν γεγονότα που συμβαίνουν στο μηχάνημα που είναι εγκατεστημένα, του δίνει την δυνατότητα να ανακαλύπτει επιθέσεις που μπορεί να περνούν απαρατήρητες από το NIDS. Τα HIDS συνήθως χρησιμοποιούνται για να προστατευτούν συστήματα με ευαίσθητα δεδομένα.

Τα πλεονεκτήματα των HIDS είναι τα παρακάτω:

- Ανιχνεύουν επιθέσεις που δεν μπορούν να ανιχνεύσουν τα NIDS (π.χ. malware).
- Μικρό κόστος
- Μπορούν να εντοπίσουν επιθέσεις ακόμα και αν στο δίκτυο το payload είναι κρυπτογραφημένο.

Παρουσιάζουν και ορισμένα μειονεκτήματα:

- Είναι δύσκολα στην διαχείριση, αφού αν θέλουμε να προστατέψουμε πολλούς υπολογιστές, πρέπει να το εγκαταστήσουμε σε όλους.
- Δεν μπορούν να εντοπίσουν τα network scan.



Protocol Based Intrusion Detection System (PIDS): Το PIDS κάνει monitor ένα πρωτόκολλο, όπως για παράδειγμα όλα τα http και https πακέτα που έρχονται και φεύγουν από έναν server. Στις περισσότερες περιπτώσεις τοποθετείται μπροστά από έναν server (π.χ. apache) και παρακολουθεί το εισερχόμενο και εξερχόμενο traffic.

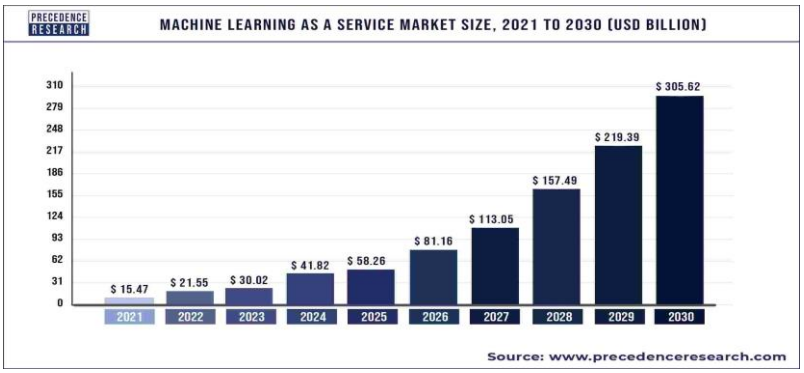
Application Protocol Based Intrusion Detection System (APIDS): Πρόκειται για ένα Intrusion detection system για συγκεκριμένες εφαρμογές. Συνήθως σχετίζεται άμεσα με κάποιο host based Intrusion detection system. Ουσιαστικά το APIDS κάνει monitor τις επικοινωνίες ανάμεσα στο application και τον server. Ένα καλό παράδειγμα APIDS είναι ένας μηχανισμός που παρακολουθεί τις επικοινωνίες του web server με την βάση δεδομένων.

Hybrid Intrusion Detection System: Αποτελεί έναν συνδυασμό δύο η παραπάνω τεχνικών. Σε αυτή τη περίπτωση συνδυάζονται δεδομένα από τον host και από το δίκτυο και δίνουν στον αμυνόμενο μία πλήρη εικόνα του δικτύου, πράγμα που καθιστά το Hybrid Intrusion Detection System πιο αποτελεσματικό σε σχέση με τα υπόλοιπα.

Γ. Machine Learning

Η μηχανική μάθηση αποτελεί ένα υπό-πεδίο της τεχνητής νοημοσύνης. Μελετά την ανάπτυξη συστημάτων τα οποία έχουν την ικανότητα να «μαθαίνουν» από τα δεδομένα ώστε να διεκπεραιώνουν και να βελτιώνουν την λειτουργία τους, αντί αυτή να είναι ρητά προγραμματισμένη από πριν, όπως στα παραδοσιακά συστήματα τεχνητής νοημοσύνης. Οι αλγόριθμοι machine learning είναι ιδιαίτερα ικανοί στην αναγνώριση μοτίβων, στην εξαγωγή χρήσιμων πληροφοριών και στην πραγματοποίηση προβλέψεων με μεγάλη ακρίβεια.

Ως αποτέλεσμα, αυτός ο τομέας έχει γνωρίσει άνθιση τα τελευταία χρόνια και εφαρμογές του χρησιμοποιούνται σε ολοένα και περισσότερους κλάδους αντικαθιστώντας πολλές φορές ανθρώπινη εργασία. Σύμφωνα με στοιχεία του Statista[1] η αξία του κλάδου αναμένεται να εκτοξευθεί από τα 22.6 δις δολάρια που βρισκόταν το 2022 στα 126 δις δολάρια μέχρι το 2025. Έδαφος κερδίζει επίσης και η παροχή Machine Learning as a Service (MLaaS) η οποία σύμφωνα με το Precedence Research[2] εκτιμάται ότι θα φτάσει τα 58 δις δολάρια μέχρι το 2025 και τα 305 δις δολάρια μέχρι το 2030.



Από τα στοιχεία που συγκεντρώσαμε φαίνεται ότι αυτή τη στιγμή το machine learning αξιοποιείται από επιχειρήσεις και οργανισμούς κυρίως για risk management, για performance analysis, για συναλλαγές αλλά και για την αυτοματοποίηση διαφόρων διαδικασιών κ.α. Άλλοι τομείς στους οποίους χρησιμοποιείται το machine learning είναι η αναγνώριση εικόνων, φωνής, η ανίχνευση απατών σε online συναλλαγές, οι προσωπικοί βοηθοί, η ανίχνευση κινδύνων σε υπολογιστικά περιβάλλοντα.

Η τεράστια αύξηση του όγκου των διαθέσιμων δεδομένων και των μεταδεδομένων (metadata) που χρησιμοποιούνται για την εκπαίδευση αλγορίθμων μηχανικής μάθησης σε συνδυασμό με την αύξηση της υπολογιστικής ισχύος τις τελευταίες δεκαετίες έχουν παίξει σημαντικό ρόλο στην βελτίωση τους.

Υπάρχουν 2 βασικές μορφές μηχανικής μάθησης. Η επιβλεπόμενη μάθηση (supervised learning) κατά την οποία ο αλγόριθμος εκπαιδεύεται για να καταλάβει τη σχέση μεταξύ των δεδομένων που δίνουμε και ενός επιθυμητού αποτελέσματος. Δηλαδή έχουμε προκαθορισμένη είσοδο (input – δεδομένα) και έξοδο (output – αποτέλεσμα). Στην περίπτωση της μη-επιβλεπόμενης (unsupervised) μάθησης γνωστό είναι μόνο το κομμάτι της εισόδου (input) των δεδομένων και ο υπολογιστής καλείται να αναγνωρίσει τα μοτίβα που μπορεί να υπάρχουν.

Τέλος, υποσύνολο της μηχανικής μάθησης είναι και τα τεχνητά νευρωνικά δίκτυα (Artificial Neural Networks) τα οποία αναλύουμε στο κεφάλαιο Ε.2.4.2.

Δ. Related Work

Κατά τη διάρκεια της διπλωματικής μας εργασίας, πραγματοποιήσαμε όσο το δυνατόν πιο ενδελεχή έρευνα γινόταν, σχετικά με την χρήση αλγορίθμων μηχανικής μάθησης σε IDS. Από αυτήν προκύπτει ότι τέτοια συστήματα βρίσκονται ακόμα σε ερευνητικό, πειραματικό στάδιο. Βρήκαμε παρ' όλα αυτά πολλούς ερευνητές από πολλές χώρες του κόσμου που καταπιάνονται με αυτό το ζήτημα. Σε αυτό το κεφάλαιο θα προσπαθήσουμε να αναφερθούμε σε βασικές πλευρές της βιβλιογραφίας που διαβάσαμε αναφέροντας ενδεικτικά ορισμένες μελέτες, αφού δεν είναι εφικτό να αναφερθούμε αναλυτικά σε όλες.

Μεγάλος όγκος ερευνητών εστιάζει την μελέτη στην χρήση πιο σύγχρονων μεθόδων μηχανικής μάθησης όπως τα νευρωνικά δίκτυα. Κύριοι λόγοι για αυτό είναι η ικανότητα τους να διαχειρίζονται πολύ μεγάλους όγκους δεδομένων και κυρίως πολύπλοκων δεδομένων που έχουν πολλά χαρακτηριστικά όπως η δικτυακή κίνηση. Οι περισσότερες μελέτες επίσης εστίασαν στην χρήση Network IDS. Αναφέρουμε ενδεικτικά την μελέτη τους Ashiku κ.α.[3] οι οποίοι δημιούργησαν ένα NIDS με την χρήση νευρωνικού δικτύου το οποίο εκπαιδεύτηκε στο dataset UNSW-NB15 καταφέρνοντας να πετύχουν accuracy περίπου 94%. Αντίστοιχο παράδειγμα είναι οι Mebawondu κ.α.[4] που δημιούργησαν ένα νευρωνικό δίκτυο το οποίο εκπαιδεύτηκε στο UNSW-NB15 και πέτυχε accuracy περίπου 80%. Βρήκαμε και μελέτες που αξιοποιήθηκαν παραδοσιακοί αλγόριθμοι μηχανικής μάθησης όπως αυτή των Jairu κ.α.[5] που χρησιμοποίησαν Support Vector Machines (SVM), Random Forest κ.α. στο dataset CIC-IDSI-2017 πετυχαίνοντας καλά αποτελέσματα.

Ενδιαφέρον έχουν και οι μελέτες για την ανάπτυξη Host IDS. Ενδεικτικά αναφέρουμε τον G.Creech[6] ο οποίος δημιούργησε ένα Anomaly Detection Host IDS με χρήση νευρωνικού δικτύου στηριζόμενος κυρίως σε dataset όπως τα ADFA -WD, ADFA-LD που περιέχουν logs για Windows και Linux αντίστοιχα. Σκοπός του ήταν να εντοπίσει zero day attacks. Αντίστοιχη μελέτη πραγματοποίησαν και οι Moskovich κ.α.[7] μελέτησαν την χρήση πολλών διαφορετικών αλγορίθμων μηχανικής μάθησης για τη δημιουργία Anomaly Detection Host IDS, με τη χρήση dataset που δημιούργησαν οι ίδιοι.

Υπάρχουν επίσης και μελέτες για Hybrid IDS όπως αυτή των Rao κ.α.[8] οι οποίοι στηρίχτηκαν σε δύο διαφορετικά μοντέλα μηχανικής μάθησης, ένα unsupervised και ένα supervised για να δημιουργήσουν ένα Hybrid IDS. Η εκπαίδευση έγινε σε τρία dataset, τα KDDCup99, NSL-KDD, UNSW-NB15.

Όλα τα παραπάνω προτεινόμενα IDS, ανήκουν στην κατηγορία του Anomaly Detection. Πολύ ενδιαφέρον έχουν όμως και οι περιπτώσεις ερευνητών που προσπάθησαν να δημιουργήσουν η να βελτιώσουν τα ήδη υπάρχοντα Signature Based IDS. Χαρακτηριστικά αναφέρουμε τους Kim κ.α.[9] οι οποίοι δημιούργησαν ένα Signature Based IDS με βάση νευρωνικά δίκτυα, το οποίο εκπαιδεύτηκε για να αναγνωρίζει τα malicious payloads σε HTTP κίνηση με βάση διάφορα patterns. Το σημαντικό σε αυτή τη μελέτη είναι ότι εκτός των διαθέσιμων datasets που χρησιμοποιήθηκαν, όπως το CICIDS2017, οι ερευνητές είχαν στη διάθεση τους τεράστιο όγκο real-time κίνησης που συλλέχθηκε από SOC υπεύθυνο για μεγάλες εταιρίες στην Κορέα. Αξίζει επίσης να σημειωθεί η μελέτη των Sohi κ.α.[10] χρησιμοποίησαν μεθόδους μηχανικής μάθησης και συγκεκριμένα νευρωνικά δίκτυα για να παράξουν παραλλαγές των ήδη υπαρχόντων malware με σκοπό να ανανεώσουν τους κανόνες των signature based IDS και να βελτιώσουν την επίδοση τους. Ισχυρίζονται ότι παρατήρησαν βελτίωση μέχρι 16%.

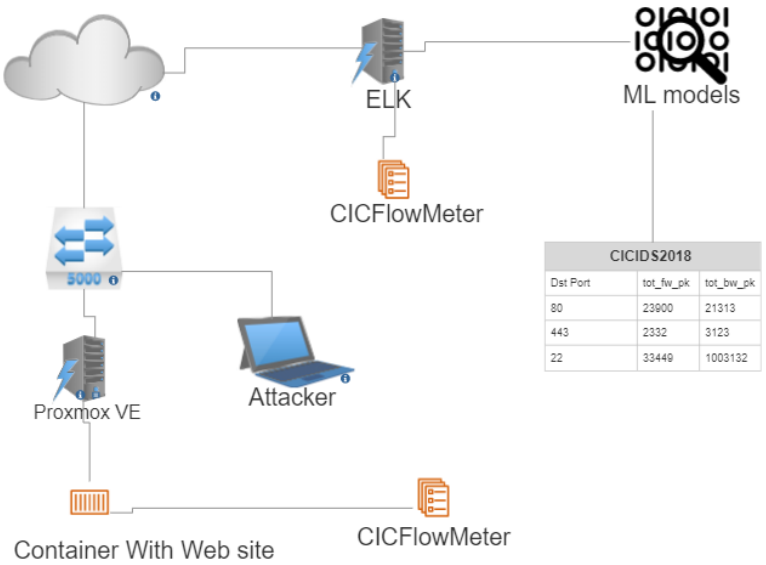
Συνολικά από την έρευνα μας φαίνεται η ικανότητα που έχουν τα μοντέλα της μηχανικής μάθησης να εξάγουν χρήσιμα χαρακτηριστικά δικτυακής κίνησης και να κάνουν κατηγοριοποίηση της κίνησης αυτής με καλή ακρίβεια. Οι περισσότεροι ερευνητές χρησιμοποίησαν έναν τύπο αλγόριθμου αλλά υπήρχαν και ορισμένοι που συνδύασαν 2 η και περισσότερους με ίδια ή και καλύτερα αποτελέσματα.

Σημαντικό είναι επίσης να αναφέρουμε ότι υπάρχει έλλειψη διαθέσιμων δεδομένων για την εκπαίδευση τέτοιων αλγορίθμων. Οι περισσότερες μελέτες που διαβάσαμε χρησιμοποίησαν παλιά dataset τα οποία δεν αντικατοπτρίζουν τις σημερινές συνθήκες και ανάγκες που υπάρχουν στην κυβερνοασφάλεια. Η διαθεσιμότητα real time δεδομένων για εκπαίδευση αλλά και για τελικό έλεγχο έπαιξε πολύ θετικό ρόλο στους αλγορίθμους των ερευνητών που είχαν αυτή τη δυνατότητα. Τέλος αξίζει να σημειώσουμε ότι οι περισσότεροι ερευνητές ασχολήθηκαν με το ίδιο το μοντέλο μηχανικής μάθησης που κάνει την πρόβλεψη και όχι με όλο το «οικοσύστημα» από την συλλογή της πληροφορίας, την επεξεργασία της, μέχρι την τελική κατηγοριοποίηση. Όλες οι πληροφορίες που προκύπτουν από την παραπάνω μελέτη, έπαιξαν ρόλο στο να σκεφτούμε και να σχεδιάσουμε την υλοποίηση μας.

Ε. Υλοποίηση

Η υλοποίηση μας αποτελείται από δυο επιμέρους τμήματα, ένα real-time περιβάλλον στο οποίο παράγουμε και συλλέγουμε κίνηση και το IDS το οποίο χρησιμοποιεί μεθόδους τεχνητής νοημοσύνης για να κατηγοριοποιήσει την κίνηση αυτή. Έχουμε δοκιμάσει δυο αλγόριθμους τεχνητής νοημοσύνης για υποψήφια IDS, οι οποίοι εκπαιδεύτηκαν για την αναγνώριση της κίνησης χρησιμοποιώντας το dataset CICIDS 2018. Επιλέξαμε να δημιουργήσουμε και ένα real-time περιβάλλον πλάι στο IDS για να ώστε να δημιουργήσουμε ένα περιβάλλον πλήρως λειτουργικό, από την αρχή ως το τέλος, αλλά και για να δούμε πως συμπεριφέρεται το IDS εκτός του dataset, σε πραγματικό περιβάλλον που βέβαια είναι πολύ απλουστευμένο και δε προσομοιάζει σε καμία περίπτωση την πολυπλοκότητα και την συνθετότητα ενός σύγχρονου εταιρικού περιβάλλοντος.

Πιο συγκεκριμένα, το περιβάλλον μας αποτελείται από ένα Proxmox VE, πάνω στο οποίο έχουμε φτιάξει ένα container που έχουμε εγκατεστημένο το DVWA (Damn Vulnerable Web App) και στο οποίο κάνουμε επιθέσεις που προσομοιάζουν τα σενάρια που καλύπτει το dataset. Τις επιθέσεις τις πραγματοποιήσαμε με Kali Linux. Για να συλλέξουμε την κίνηση χρησιμοποιήσαμε το elasticsearch και για να την μεταμορφώσουμε σε μορφή κατάλληλη για το σύστημα τεχνητής νοημοσύνης χρησιμοποιήσαμε το CICFlowMeter. Η δομή του περιβάλλοντος μας φαίνεται στην παρακάτω εικόνα:



E.1. Real time

Σε αυτό το κεφάλαιο θα εξηγήσουμε τα βασικά μέρη που αποτελούν το real time περιβάλλον μας.

E.1.1. ELK Stack

Το ELK stack είναι ένα αρκτικόλεξο που χρησιμοποιείται για να περιγράψει ένα stack από 3 γνωστά projects : Elasticsearch, Logstash, and Kibana. Το ELK stack μας δίνει την δυνατότητα να συγκεντρώνουμε logs από όλα τα συστήματα και τα applications, να τα αναλύσουμε και να φτιάξουμε διαγράμματα έτσι ώστε να μπορέσουμε να κάνουμε καλύτερη ανάλυση σε επίπεδο ασφάλειας και όχι μόνο.

E = Elasticsearch

Είναι ένα κατανεμημένο search και analytics engine η καρδιά του Elastic Stack. Ο ρόλος του είναι να αποθηκεύει τα δεδομένα, έτσι ώστε να είναι εύκολη και γρήγορη η αναζήτηση. Μπορούμε να χρησιμοποιήσουμε το elasticsearch για να κάνουμε αναζήτηση σε δεδομένα πολλών τύπων, λόγω της δυνατότητας που έχει να αποθηκεύει δεδομένα από διάφορες πηγές, τα οποία στην συνέχεια τα ταξινομεί με βάση αυτόματο η και manual τρόπο(μέσα από το configuration). Τέλος το elasticsearch έχει την δυνατότητα να τρέχει παράληλα σε πολλά nodes και έτσι προσφέρει στον χρήστη την δυνατότητα να κάνει αναζίτηση σε μεγάλο όγκο δεδομένων, ανάλογα και με τους πόρους που του έχουμε εκχωρήσει.

Συνοψίζοντας, το elasticseach χρησιμοποιεί τους shipping agents (filebeat,packetbeat etc.) οι οποίοι είναι υπεύθυνοι να φέρνουν τα δεδομένα στο σύστημα. Μετά είναι υπεύθυνο το ίδιο να προετοιμάσει τα δεδομένα του με τέτοιο τρόπο που θα μπορεί να γίνουν index και να τρέξουμε σε αυτά τα query μας. **Παρακάτω θα παρουσιάσουμε την αρχιτεκτονική του elasticsearch**

Elasticsearch Cluster: Το Elasticsearch cluster απαρτίζεται από το σύνολο των nodes που χρησιμοποιούμε για να αποθηκεύσουμε τα δεδομένα. Μπορούμε να ρυθμίσουμε τόσο των αριθμό του όσο και τις ip που χρησιμοποιούν στο αρχείο config/elasticsearch.yml.

Elasticsearch Node: Σαν Node ορίζεται ένας server που είναι μέρος του cluster. Το node είναι ένα instance και όχι μηχάνημα που πρακτικά σημαίνει ότι μπορούμε να έχουμε παραπάνω από ένα node σε ένα μηχάνημα. Τα είδη των nodes είναι 3.

- Elasticsearch master node: Είναι υπεύθυνο για τις λειτουργίες του cluster (π.χ. δημιουργία διαγραφή index)
- Elasticsearch data node: Έχει δεδομένα και ανεστραμμένο το index.
- Elasticsearch client node: Λειτουργεί σαν load balancer.

Το Elasticsearch χρησιμοποιεί 2 βασικές πόρτες για επικοινωνία σε δικτυακό επίπεδο:

- Port 9200 – χρησιμοποιείτε για requests που έρχονται προς το cluster, ένα καλό παράδειγμα είναι το RestApi,με το οποίο μπορεί κάποιος να τραβήξει δεδομένα από κάποιο index.
- Port 9400 – χρησιμοποιείτε για εσωτερική επικοινωνία στο cluster.

Elasticsearch Shards: Δεν υπάρχει κάποιο συγκεκριμένο όριο στα documents που αποθηκεύουμε ανά index, όμως υπάρχει ο κίνδυνος το index να ξεπεράσει το storage limit και να δημιουργήσει πρόβλημα στο elasticsearch Για να το αποφύγουμε αυτό το κόβουμε σε

μικρότερα κομμάτια που λέγονται shards. Τα shards είναι μικρά και ευέλικτα indexes που είναι τα Building blocks τις αρχιτεκτονικής του elasticsearch. Πρακτικά είναι αυτά που δίνουν την δυνατότητα στο elasticsearch να λειτουργεί κατανεμημένα, αφού μπορούν να βρίσκονται οπουδήποτε μέσα στο cluster.

Elasticsearch Replicas: Είναι αντίγραφα των index shards και χρησιμοποιούνται σαν μηχανισμός ασφάλειας για να δεδομένα (backup). Δεν βρίσκονται ποτέ στον ίδιο Node με τα αρχικά δεδομένα και καλό είναι να βρίσκονται και σε διαφορετική περιοχή γεωγραφικά.

Elasticsearch Analyzers: Είναι υπεύθυνος να αναλύσει τα δεδομένα και να τα χωρίσει στα πεδία που πρέπει. Πρακτικά κάθε analyzer έχει ένα tokenizer και πολλά token φίλτρα. Το tokenizer είναι υπεύθυνο να κόψει τα δεδομένα στα προκαθορισμένα φίλτρα.

Elasticsearch Documents: Το elasticsearch είναι σχεδιασμένο έτσι ώστε να υποστηρίζει την λήψη εγγράφων που είναι αποθηκευμένα σαν Json. Υποστηρίζει επίσης πολύ επίπεδες δομές δεδομένων, πράγμα που μας δίνει την δυνατότητα να έχουμε πολύπλοκα δεδομένα καθώς και να κάνουμε πολύπλοκα ερωτήματα.

L = Logstash

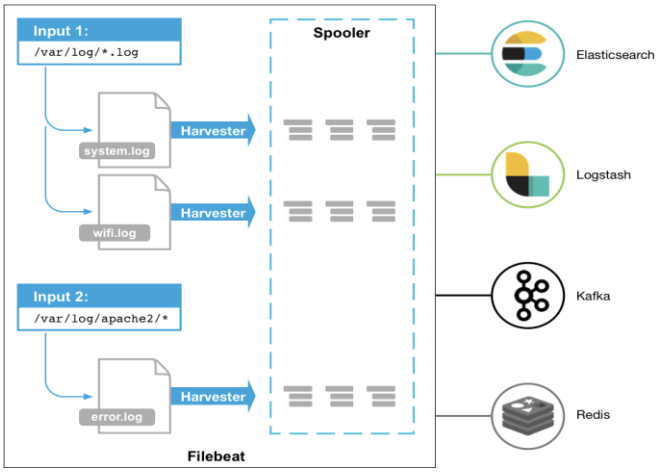
Το logstash είναι ένα εργαλείο το οποίο μας επιτρέπει να συλλέγουμε δεδομένα από διαφορετικές πηγές, να τα μετατρέπουμε και να τα στέλνουμε στο σύστημα που θέλουμε.

K = Kibana

Το Kibana είναι ένα εργαλείο για data visualization, ενώ επιπλέον μας παρέχει έναν πιο εύχρηστο τρόπο για γρήγορη αναζήτηση μέσα στα δεδομένα μας.

Filebeat

Το Filebeat είναι η εφαρμογή με την οποία στέλνουμε τα logs από τον client στον server. Με το filebeat κάνουμε monitor τα log αρχεία, στο path που του έχουμε ορίσει. Σε κάθε εκκίνηση του, το filebeat ανοίγει μία ή παραπάνω εισόδους οι οποίες κοιτάζουν στα αρχεία που έχουμε ορίσει και στην συνέχεια ξεκινάει έναν harvester για κάθε αρχείο και στέλνει τα νέα data που έχουν μέσα τα logs στο libbeat. Αυτό συγκεντρώνει τα data και τα στέλνει στο σημείο που έχει οριστεί στο filebeat.



Το configuration αρχείο του filebeat είναι το /etc/filebeat/ filebeat.yml

```
-rw-------. 1 root root   10031 Jan 25 18:47 filebeat.yml
[root@localhost filebeat]# pwd
/etc/filebeat
```

Το έχουμε ρυθμίσει ώστε να στέλνει μόνο το αρχείο /root/TCPDUMP_and_CICFlowMeter/csv/cicids2018.dataset.csv το οποίο δημιουργούμε από το cicflowmeter σε συνδυασμό με ένα custom script.

```
# Paths that should be crawled and fetched. Glob based paths.
paths:
  #- /var/log/*
  #- /root/packet.log
  - /root/TCPDUMP_and_CICFlowMeter/csv/cicids2018.dataset.csv
  #- c:\programdata\elasticsearch\logs\*
```

Παρακάτω φαίνεται ότι τα στέλνουμε στο elasticsearch που είναι εγκατεστημένο στο ίδιο μηχάνημα.

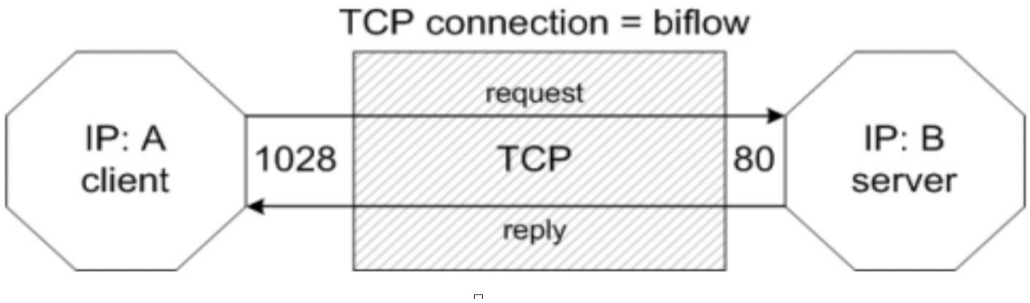
```
# ---------------------------- Elasticsearch Output ----------------------------
output.elasticsearch:
  # Array of hosts to connect to.
  hosts: ["localhost:9200"]
```

E.1.2. CICFlowMeter

Το CICFlowMeter είναι ένα εργαλείο το οποίο δημιουργήθηκε από το Canadian Institute of Cybersecurity (CIC) και με το οποίο φτιάχνουμε και αναλύουμε τα δικτυακά flows. Χρησιμοποιείται για να παράξει αμφίδρομα flows, όπου το πρώτο πακέτο καθορίζει την κατεύθυνση forward (από source στο destination) και backward (από το destination στο source). Μπορεί να συλλέξει περισσότερα από 80 χαρακτηριστικά της κίνησης που σχετίζονται με την διάρκεια του session, τον αριθμό των πακέτων , τον αριθμό των bytes κ.α.

Με το CICFlowMeter δημιουργούνται Biflows από pcap αρχεία. Σαν flow ορίζεται η ακολουθία πακέτων που μοιράζεται κοινές ιδιότητες. Κατά κύριο λόγο σε μία ακολουθία τα χαρακτηριστικά που είναι κοινά είναι τα: Source IP, Destination IP, Protocol, Source port, Destination port.

Ο Στόχος του bidirectional flows είναι η διασύνδεση των request με τις απαντήσεις που αυτά λαμβάνουν. Όπως φαίνεται στο παρακάτω παράδειγμα, ο client με IP A μέσω της πόρτας 1028 κάνει ένα request, στον server με IP B στην πόρτα 80 και αυτός με την σειρά του απαντά από την πόρτα 80 στην πόρτα 1028. Σε αυτό το παράδειγμα υπάρχουν και τα 5 χαρακτηριστικά που αναφέρθηκαν πιο πάνω.



Στον παρακάτω πίνακα φαίνονται τα features που κάνει extract το CICFlowMeter:

Feature Name	Description
fl_dur	Flow duration
tot_fw_pk	Total packets in the forward direction
tot_bw_pk	Total packets in the backward direction
tot_l_fw_pkt	Total size of packet in forward direction
fw_pkt_l_max	Maximum size of packet in forward direction
fw_pkt_l_min	Minimum size of packet in forward direction
fw_pkt_l_avg	Average size of packet in forward direction
fw_pkt_l_std	Standard deviation size of packet in forward direction
Bw_pkt_l_max	Maximum size of packet in backward direction
Bw_pkt_l_min	Minimum size of packet in backward direction
Bw_pkt_l_avg	Mean size of packet in backward direction
Bw_pkt_l_std	Standard deviation size of packet in backward direction
fl_byt_s	flow byte rate that is number of packets transferred per second
fl_pkt_s	flow packets rate that is number of packets transferred per second
fl_iat_avg	Average time between two flows
fl_iat_std	Standard deviation time two flows
fl_iat_max	Maximum time between two flows
fl_iat_min	Minimum time between two flows
fw_iat_tot	Total time between two packets sent in the forward direction
fw_iat_avg	Mean time between two packets sent in the forward direction
fw_iat_std	Standard deviation time between two packets sent in the forward direction
fw_iat_max	Maximum time between two packets sent in the forward direction
fw_iat_min	Minimum time between two packets sent in the forward direction
bw_iat_tot	Total time between two packets sent in the backward direction
bw_iat_avg	Mean time between two packets sent in the backward direction
bw_iat_std	Standard deviation time between two packets sent in the backward direction
bw_iat_max	Maximum time between two packets sent in the backward direction
bw_iat_min	Minimum time between two packets sent in the backward direction
fw_psh_flag	Number of times the PSH flag was set in packets travelling in the forward direction (0 for UDP)
bw_psh_flag	Number of times the PSH flag was set in packets travelling in the backward direction (0 for UDP)
fw_urg_flag	Number of times the URG flag was set in packets travelling in the forward direction (0 for UDP)
bw_urg_flag	Number of times the URG flag was set in packets travelling in the backward direction (0 for UDP)
fw_hdr_len	Total bytes used for headers in the forward direction
bw_hdr_len	Total bytes used for headers in the forward direction
fw_pkt_s	Number of forward packets per second
bw_pkt_s	Number of backward packets per second
pkt_len_min	Minimum length of a flow

pkt_len_max	Maximum length of a flow
pkt_len_avg	Mean length of a flow
pkt_len_std	Standard deviation length of a flow
pkt_len_va	Minimum inter-arrival time of packet
fin_cnt	Number of packets with FIN
syn_cnt	Number of packets with SYN
rst_cnt	Number of packets with RST
pst_cnt	Number of packets with PUSH
ack_cnt	Number of packets with ACK
urg_cnt	Number of packets with URG
cwe_cnt	Number of packets with CWE
ece_cnt	Number of packets with ECE
down_up_ratio	Download and upload ratio
pkt_size_avg	Average size of packet
fw_seg_avg	Average size observed in the forward direction
bw_seg_avg	Average size observed in the backward direction
fw_byt_blk_avg	Average number of bytes bulk rate in the forward direction
fw_pkt_blk_avg	Average number of packets bulk rate in the forward direction
fw_blk_rate_avg	Average number of bulk rate in the forward direction
bw_byt_blk_avg	Average number of bytes bulk rate in the backward direction
bw_pkt_blk_avg	Average number of packets bulk rate in the backward direction
bw_blk_rate_avg	Average number of bulk rate in the backward direction
subfl_fw_pk	The average number of packets in a sub flow in the forward direction
subfl_fw_byt	The average number of bytes in a sub flow in the forward direction
subfl_bw_pkt	The average number of packets in a sub flow in the backward direction
subfl_bw_byt	The average number of bytes in a sub flow in the backward direction
fw_win_byt	Number of bytes sent in initial window in the forward direction
bw_win_byt	# of bytes sent in initial window in the backward direction
Fw_act_pkt	# of packets with at least 1 byte of TCP data payload in the forward direction
fw_seg_min	Minimum segment size observed in the forward direction
atv_avg	Mean time a flow was active before becoming idle
atv_std	Standard deviation time a flow was active before becoming idle
atv_max	Maximum time a flow was active before becoming idle
atv_min	Minimum time a flow was active before becoming idle
idl_avg	Mean time a flow was idle before becoming active
idl_std	Standard deviation time a flow was idle before becoming active
idl_max	Maximum time a flow was idle before becoming active
idl_min	Minimum time a flow was idle before becoming active

E.1.3. Δημιουργία csv αρχείων για το Filebeat

Με το παρακάτω script φτιάχνουμε το log που φτάνει στο elasticsearch, τρέχοντας το με nohup για να μην κλείνει μαζί με το session του ssh από το οποίο το τρέξαμε. Έχουμε έναν daemon, ο οποίος κάθε φορά που εμφανίζεται αρχείο .csv μέσα στον φάκελο /root/TCPDUMP_and_CICFlowMeter/csv/, τον προσθέτει στο τέλος του αρχείου cicids2018.dataset.csv, από το οποίο μέσω του filebeat πάνε στο elk.

```
if [ -f /root/TCPDUMP_and_CICFlowMeter/csv/ctl/cicids2018.dataset.ctl ]
then
 echo "the deamon is running" >> /root/TCPDUMP_and_CICFlowMeter/csv/log/cicids2018.dataset.log
exit 1
else
touch /root/TCPDUMP_and_CICFlowMeter/csv/ctl/cicids2018.dataset.ctl
 echo "the deamon is starting" >> /root/TCPDUMP_and_CICFlowMeter/csv/log/cicids2018.dataset.log
fi
while [ 5 = 5 ]
do
if [ -f /root/TCPDUMP_and_CICFlowMeter/csv/ctl/cicids2018.dataset.ctl ]
then
array=($(ls /root/TCPDUMP_and_CICFlowMeter/csv|grep csv|grep -v cicids2018))
for i in "${array[@]}"
do
        cat $i >> cicids2018.dataset.csv
        mv $i ./archive/
        echo "procesing $i" >> /root/TCPDUMP_and_CICFlowMeter/csv/log/cicids2018.dataset.log
done
else
exit 0
fi
sleep 10
done
```

Ταυτόχρονα για να μαζεύουμε τα δεδομένα , τρέχουμε την παρακάτω εντολή :

Nohup /root/TCPDUMP_and_CICFlowMeter/capture_interface_pcap.sh eth0 . root

E.1.4. Εξαγωγή δεδομένων από το elasticsearch

Η εξαγωγή δεδομένων από το elasticsearch στο μοντέλο γίνεται με το παρακάτω script:

```
curl -o /tmp/data.csv -v -XPOST "http://localhost:9200/.ds-filebeat-8.5.3-2023.01.06-000001/_search " -H 'Content-Type: application/json' -d '
{
"size": 5000,
  "query": {
    "match": {
      "log.file.path": {
        "query" : "/root/TCPDUMP_and_CICFlowMeter/csv/cicids2018.dataset.csv"
      }
    }
  },
  "fields": [
    "message"
  ],
  "_source": false
}
'
awk '{ gsub("{", "\n") } 9' /tmp/data.csv|awk '/1/ -F "/n" {print $1$2$3}'|grep -v _index|grep -v took|grep -v total|grep -v value > tmp_export.csv
sed -i "s/\"message\"\:\[\"//" tmp_export.csv
sed -i "s/\"\]\}\}\,//" tmp_export.csv
cat headers.csv > export.csv
cat tmp_export.csv >> export.csv
rm tmp_export.csv
sed -i '/^$/d' export.csv
```

Εδώ χρησιμοποιώντας την curl, σε συνδυασμό με το rest api του elasticsearch συλλέγουμε τα δεδομένα. Στην συνέχεια γίνεται μία μικρή επεξεργασία ώστε να αφαιρεθούν τα περιττά δεδομένα του elasticsearch.

E.1.5. Proxmox VE

Το Proxmox VE είναι μία open source virtualization πλατφόρμα, η οποία έχει δύο βασικές τεχνολογίες, KVM για τα VM και LXC για τα containers. Το συγκεκριμένο εργαλείο το χρησιμοποιήσαμε για να στήσουμε σε τοπικό δίκτυο τις υπηρεσίες στις οποίες κάναμε τις επιθέσεις. Για λόγους οικονομίας πόρων προτιμήσαμε τα containers, έτσι ώστε να μην εξομοιώνουμε τον kernel αλλά να χρησιμοποιούμε τον kernel που έχει hypervisor. Από την μία με αυτόν τον τρόπο εξοικονομήσαμε πόρους, από την άλλη όμως μας προστέθηκαν περιορισμοί όπως πχ ότι δεν μπορούμε να έχουμε container με windows μηχάνημα ενώ σε

άλλες εκδόσεις linux δημιουργήθηκαν προβλήματα με τα repos. Παρακάτω φαίνεται το linux container στο GUI

E.2. Classification with AI Models

Με βάση την έρευνα που κάναμε και αναφέραμε σε προηγούμενο κεφάλαιο, αποφασίσαμε να δοκιμάσουμε δυο διαφορετικούς τύπους αλγορίθμων μηχανικής μάθησης τα οποία και αναφέρουμε παρακάτω, με σκοπό να κάνουμε σύγκριση των αποτελεσμάτων. Για την εκπαίδευση των μοντέλων τεχνητής νοημοσύνης χρησιμοποιήθηκε το Jupyter Notebook σε τοπικά μηχανήματα αλλά και το Google Colab Pro+ αφού απαιτήθηκε μεγάλη υπολογιστική ισχύς. Ο κώδικας γράφτηκε σε Python. Σχετικά με τα δεδομένα που χρησιμοποιήσαμε για την εκπαίδευση, μας απασχόλησε το πως αυτά θα είναι όσο πιο up-to-date γίνεται και θα αντικατοπτρίζουν σύγχρονες τάσεις. Αξιοποιήσαμε το MITRE ATT&CK με σκοπό να βρούμε τέτοιες πληροφορίες. Έχοντας σαν δεδομένο ότι εμείς δεν έχουμε την υποδομή να παράξουμε τέτοια κακόβουλη κίνηση και σε τέτοιο εύρος που να αντικατοπτρίζει ένα ρεαλιστικό περιβάλλον, καταλήξαμε στην χρήση κάποιων από τα δημόσια διαθέσιμα datasets. Με βάση και παραπάνω και λόγω και της μεγαλύτερης διαθεσιμότητας σχετικών dataset καταλήξαμε στη δημιουργία Anomaly Detection Network IDS. Το dataset που χρησιμοποιήσαμε είναι το CICIDS2018

E.2.1. CICIDS 2018

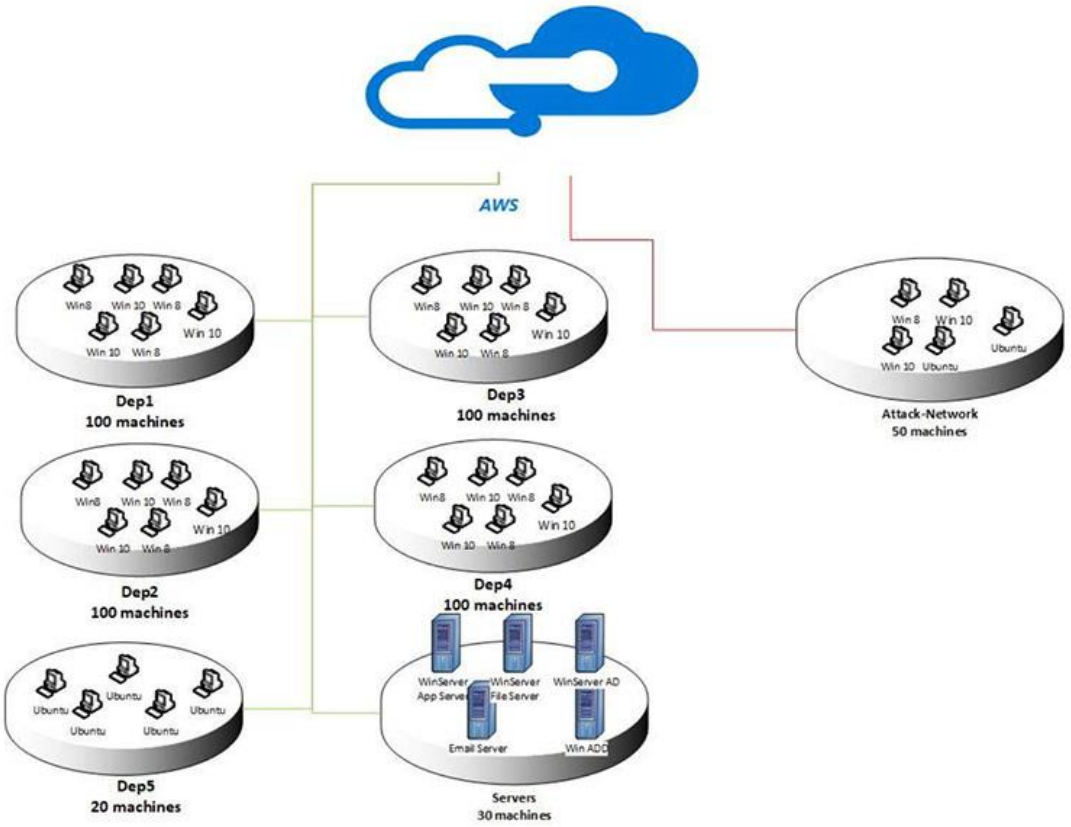
Το CICIDS 2018 είναι ένα dataset που έχει δημιουργηθεί από τον οργανισμό Communications Security Establishment (CSE) του Καναδά σε συνεργασία με το Canadian Institute for Cybersecurity (CIC)[11], με σκοπό να δώσει ώθηση στο να αναπτυχθούν μοντέλα τεχνητής νοημοσύνης για anomaly detection. Εξ όσων γνωρίζουμε, πρόκειται για ένα από τα πιο σύγχρονα datasets για IDS που είναι διαθέσιμα για το ευρύ κοινό. Οι ίδιοι οργανισμοί έχουν δημιουργήσει αντίστοιχα dataset παλαιότερα, όπως το CICIDS 2017, αλλά και πιο εξειδικευμένα, όπως dataset για IoT Συσκευές (CIC IoT Dataset 2022), για DDoS επιθέσεις (CIC-DDoS2019), για Malware (CIC-MalMem-2022), για Malware σε Android συσκευές (CICMalDroid 2020). Εκτιμάμε πως για μελλοντική έρευνα, μπορούν να χρησιμοποιηθούν συνδυαστικά αρκετά από τα παραπάνω. Στην παρούσα εργασία χρησιμοποιήσαμε το CICIDS 2018 επειδή καλύπτει ένα ευρύτερο φάσμα επιθέσεων και είναι ιδανικό για τη δημιουργία machine learning μοντέλων IDS.

Πριν καταλήξουμε σε αυτό το dataset, κάναμε μια έρευνα για όλα τα dataset που είναι διαθέσιμα για IDS. Ορισμένα από τα πιο γνωστά είναι το KDD Cup99[12], το NSL-KDD[13], τα AFDA LD και WD[14]. Σε σχέση με τα παραπάνω, το CICIDS 2018 καλύπτει μεγαλύτερο εύρος επιθέσεων και έχει πολύ μεγαλύτερο αριθμό δειγμάτων, γεγονός που κάνει την κίνηση πιο αντιπροσωπευτική.

Οι δημιουργοί του dataset προσπάθησαν να δώσουν λύσεις στα παρακάτω προβλήματα. Στη δημιουργία ενός dataset το οποίο από την μια θα αντικατοπτρίζει τη δομή ενός σύγχρονου και πολύπλοκου εταιρικού περιβάλλοντος το οποίο όμως δε θα περιέχει πολλή «ιδιωτική» πληροφορία ώστε να μπορεί να γενικευτεί και να αξιοποιηθεί και σε άλλες περιπτώσεις.

Προσπάθησαν επίσης να δημιουργήσουν κακόβουλη κίνηση η οποία θα καλύπτει τις σύγχρονες τάσεις που υπάρχουν στις κυβερνοεπιθέσεις.

Για να πετύχουν τα παραπάνω, στράφηκαν στη λύση ενός dataset για Anomaly Detection IDS. Η αρχιτεκτονική δικτύου που επέλεξαν και που φαίνεται στην παρακάτω εικόνα προσομοιάζει αυτή ενός κανονικού δικτύου και περιέχει 5 subnet τα: (R&D Department (Dep1), Management Department (Dep2), Technician Department (Dep3), Secretary and operation Department (Dep4), IT Department (Dep5) και ένα Server Room).



Για να παραχθεί η κίνηση, δημιουργήθηκαν δυο προφίλ χρηστών με τη χρήση μεθόδων τεχνητής νοημοσύνης τα οποία αντιπροσωπεύουν πιο γενικές τάσεις και συμπεριφορές που παρατηρούνται μέσα σε ένα δίκτυο.

E.2.1.1. B-Profile (Benign)

Χρησιμοποιώντας αλγόριθμους τεχνητής νοημοσύνης και μεθόδους στατιστικής ανάλυσης όπως K-Means, Random Forest, SVM, δημιουργείται κίνηση η οποία προσομοιάζει την κανονική κίνηση όσον αφορά γνωρίσματα όπως το μέγεθος των πακέτων, ο αριθμός των πακέτων ανά flow, το μέγεθος του payload κ.α. Τα πρωτόκολλα που αναπαρίστανται στην benign κίνηση περιλαμβάνουν τα HTTPS, HTTP, SMTP, POP3, IMAP, SSH, and FTP. Σύμφωνα με τους δημιουργούς, η παραπάνω λύση είναι πολύ «ευέλικτη» αφού μόλις εξαχθούν τα patterns μπορεί να δημιουργηθεί «συνθετική» κίνηση μέσω ενός agent (CIC-BenignGenerator) ή και χειροκίνητα, χωρίς να χρειάζεται να γίνει anonymized κάποια πληροφορία.

E.2.1.2. M-Profile (Malicious)

Μέσω του M-Profile, γίνεται αναπαράσταση διαφορετικών επιθέσεων. Για κάθε επίθεση χρησιμοποιούνται ένα ή περισσότερα μηχανήματα που βρίσκονται εκτός δικτύου. Η δημιουργία αυτής της κίνησης γίνεται είτε χειροκίνητα, είτε με την χρήση αυτοματοποιημένων scripts. Αναφέρεται από τους δημιουργούς ότι περιλαμβάνονται 7 διαφορετικά σενάρια επιθέσεων τα οποία φαίνονται στον παρακάτω πίνακα. Αξίζει να σημειώσουμε ότι δε βρήκαμε κίνηση η οποία προσομοιάζει στην επίθεση Heartleech.

Attack	Tools	Duration	Attacker	Victim
Bruteforce attack	FTP – Patator SSH – Patator	One day	Kali linux	Ubuntu 16.4 (Web Server)
DoS attack	Hulk, GoldenEye, Slowloris, Slowhttptest	One day	Kali linux	Ubuntu 16.4 (Apache)
DoS attack	Heartleech	One day	Kali linux	Ubuntu 12.04 (Open SSL)
Web attack	• Damn Vulnerable Web App (DVWA) • In-house selenium framework (XSS and Brute-force)	Two days	Kali linux	Ubuntu 16.4 (Web Server)
Infiltration attack	• First level: Dropbox download in a windows machine • Second Level: Nmap and portscan	Two days	Kali linux	Windows Vista and Macintosh
Botnet attack	• Ares (developed by Python): remote shell, file upload/download, capturing • screenshots and key logging	One day	Kali linux	Windows Vista, 7, 8.1, 10 (32-bit) and 10 (64-bit)
DDoS+PortScan	Low Orbit Ion Canon (LOIC) for UDP, TCP, or HTTP requests	Two days	Kali linux	Windows Vista, 7, 8.1, 10 (32-bit) and 10 (64-bit)

E.2.1.2.1. Brute Force

Για την πραγματοποίηση της επίθεσης χρησιμοποιείται το Patator, ένα tool το οποίο είναι γραμμένο σε Python. Χρησιμοποιούνται δυο modules ένα για FTP και ένα για SSH. Ο επιτιθέμενος χρησιμοποιεί Kali Linux ενώ ο στόχος είναι ένα Ubuntu 14.0. Η λίστα κωδικών που χρησιμοποιήθηκε περιείχε 90εκ λέξεις

E.2.1.2.2. Botnet

Σε αυτό το σενάριο χρησιμοποιείται το Zeus, το οποίο είναι ένα Trojan horse malware που τρέχει σε διάφορες εκδόσεις των Windows. Χρησιμοποιείται συνήθως για κακόβουλες ενέργειες όπως κλοπή στοιχείων e-banking, εγκατάσταση ransomware

και συνήθως εγκαθίσταται μέσω phishing επικοινωνίας. Συμπληρωματικά χρησιμοποιήθηκε και το Ares Botnet το οποίο είναι open source και έχει τις παρακάτω δυνατότητες: remote cmd.exe shell, persistence, file upload/download, screenshot, key logging. Στο σενάριο μολύνθηκαν υπολογιστές και με τα δυο botnets και κάθε 400 δευτερόλεπτα ζητούνταν από τα zombies screenshots.

E.2.1.2.3. DoS/DDoS

Σε αυτό το σενάριο χρησιμοποιούνται διαφορετικά εργαλεία. Το πρώτο εργαλείο είναι το Slowloris το οποίο χρησιμοποιείται για επιθέσεις σε web servers και ακολουθεί τα παρακάτω βήματα: Ξεκινάει κάνοντας πολλά request, στέλνει περιοδικά headers ώστε να κρατάει ανοιχτά τα connections, δε κλείνει ποτέ τα connections και αν ο server τα κλείσει ανοίγει νέα. Χρησιμοποιήθηκαν επίσης τα εργαλεία Hulk, GoldenEye, Slowhttptest χωρίς όμως να παρατίθενται περισσότερες πληροφορίες. Για την DDoS επίθεση χρησιμοποιήθηκαν το Low Orbit Ion Cannon (LOIC) και το High Orbit Ion Cannon (HOIC) το οποίο αποτελεί μια μετεξέλιξη του LOIC και μπορεί να επιτεθεί μέχρι και σε 256 URLs ταυτόχρονα. Τα παραπάνω εργαλεία χρησιμοποιήθηκαν από 4 υπολογιστές για να κάνουν τις επιθέσεις.

E.2.1.2.4. Web Attacks

Αυτό το σενάριο περιλαμβάνει διαφορετικά στάδια, που όλα στοχεύουν στο Damn Vulnerable Web App (DVWA) το οποίο έχει στηθεί σε server. Η επίθεση περιλαμβάνει XSS, Brute Force και SQL Injection.

E.2.1.2.5. Infiltration

Αυτό το σενάριο αποτελείται από δυο στάδια. Αρχικά μέσω mail, μεταφορτώνεται από τον στόχο ένα malicious αρχείο το οποίο εκμεταλλεύεται ευπάθεια κάποιας εφαρμογής. Μέσω του Metasploit δημιουργείται backdoor στον υπολογιστή. Το δεύτερο στάδιο περιλαμβάνει ενέργειες όπως IP Sweep, full port scan με τη χρήση του nmap.

Τέλος, χρησιμοποιείται το CICFlowMeter Tool για να γίνουν extract περίπου 80 features της κίνησης.

E.2.2. Preprocessing

Το CICIDS2018 αποτελείται από 10 αρχεία σε μορφή csv, καθένα από τα οποία περιέχει την κίνηση μιας μέρας. Μέσα σε κάθε αρχείο περιέχεται φυσιολογική κίνηση του δικτύου αλλά και κακόβουλη κίνηση. Επεξεργαστήκαμε κάθε αρχείο ξεχωριστά χρησιμοποιώντας τις βιβλιοθήκες pandas και numpy της python. Όλα τα αρχεία περιείχαν λάθος τιμές και περιττές πληροφορίες τις οποίες και διαχειριστήκαμε. Πιο συγκεκριμένα:

<u>Σχετικά με τα δείγματα (κάθε δείγμα είναι ένα row στο dataframe):</u>

Βρήκαμε 95.760 infinite και NaN τιμές. Η συντριπτική πλειοψηφία των τιμών άνηκε στο Benign Label, οπότε θεωρήσαμε πως δεν θα χαθεί σημαντική πληροφορία και τις διαγράψαμε. Υπήρχαν επίσης περίπου 100 δείγματα τα οποία διαγράφτηκαν γιατί αντί για αριθμητική τιμή περιείχαν τον τίτλο του κάθε column (πχ. Protocol, Label κτλ.) Τέλος βρέθηκαν περίπου 4εκ διπλότυπα (duplicate) δείγματα τα οποία επίσης διαγράφηκαν. Μετά από αυτές τις ενέργειες, παρέμειναν 11.657.532 δείγματα στο dataset

<u>Σχετικά με τα features (κάθε feature είναι ένα column στο dataframe):</u>

Βρέθηκαν 4 columns, τα Flow ID, Source IP, Source Port, Destination IP, τα οποία περιέχονται μόνο σε ένα csv από τα δέκα. Παρ' ότι θεωρούμε ότι περιέχουν πολύ σημαντική πληροφορία, ειδικά για ορισμένες επιθέσεις όπως το Infiltration, δε μπορούσαμε να τα κρατήσουμε αφού αυτή η πληροφορία υπάρχει για ένα πολύ μικρό δείγμα του dataset, οπότε και τα διαγράψαμε. Βρέθηκαν επίσης 8 columns, τα: Bwd PSH Flags, Bwd URG Flags, Fwd Byts/b Avg, Fwd Pkts/b Avg, Fwd Blk Rate Avg, Bwd Byts/b Avg, Bwd Pkts/b Avg, Bwd Blk Rate Avg, που σε όλα τα αρχεία είχαν μόνο μηδενικές τιμές και για το λόγο αυτό διαγράφηκαν.

Αποφασίσαμε επίσης να διαγράψουμε το Timestamp και το Protocol. Ως προς το timestamp, προχωρήσαμε σε αυτή την απόφαση επειδή δεν θέλαμε να γίνει correlated κάποια επίθεση με την ώρα που αυτή πραγματοποιήθηκε. Το feature Protocol, περιείχε μόνο 3 τιμές, τις 0,6,17 οι οποίες δεν ήταν ξεκάθαρο τι αντιπροσωπεύουν. Θεωρήσαμε πως αυτή η πληροφορία είναι περιττή αφού καλύπτεται από το Destination Port.

Από όλα τα features που παρέμειναν στο dataset, το Destination Port είναι το μόνο που δεν είναι αριθμητικό. Παρότι τα ports είναι νούμερα, δεν πρόκειται για συνεχής τιμές και δεν μπορούν να τοποθετηθούν σε κάποια κλίμακα με βάση τα νούμερα. Πρόκειται για τιμές που αντιπροσωπεύουν ξεχωριστές κατηγορίες. Για έναν αλγόριθμο τεχνητής νοημοσύνης, θα ήταν πρόβλημα να αφήσουμε έτσι αυτό το feature γιατί θα το αντιμετώπιζε ως αριθμητικό και έτσι θα χάνονταν πολύ πληροφορία. Για τον λόγο αυτό χωρίσαμε τα Destination Ports σε κατηγορίες, ανάλογα με την γενική χρήση τους και ύστερα εφαρμόσαμε την τεχνική One Hot Encoding. Αντικαταστήσαμε δηλαδή το Destination Port με παραπάνω features καθένα από τα οποία αντιπροσωπεύει μια από την κάθε κατηγορία που επιλέξαμε και παίρνει τις τιμές 1 ή 0 ανάλογα με τον αν το Port ανήκει ή όχι σε αυτή την κατηγορία. Με αυτόν τον τρόπο δώσαμε στον αλγόριθμο μας ένα input που μπορεί να γίνει αντιληπτό από αυτό.

Οι κατηγορίες που επιλέξαμε είναι οι:

- Πόρτες που σχετίζονται με το HTTP – HTTPS: (80, 81, 280, 443, 444, 8080, 8545)
- Πόρτες που σχετίζονται με information gathering σε επίπεδο δικτύου: (53, 1900, 79,15, 84, 23, 92, 123, 135, 137, 138, 139, 5355, 3478)
- Πόρτες που σχετίζονται με ενέργειες σε επίπεδο δικτύου: (0, 1, 21, 22, 67, 68, 139,445, 500, 3389, 3398, 3544, 1433, 3306)

Το σκεπτικό μας για αυτό το διαχωρισμό είναι ότι τις επιθέσεις που έχουμε να αντιμετωπίσουμε στο dataset, μπορούμε να τις χωρίσουμε σε 2 μεγάλες κατηγορίες με βάση το layer στο μοντέλο OSI που θέλουμε να τις εντοπίσουμε. Δηλαδή τις επιθέσεις που σχετίζονται άμεσα με το web (xss,sql injection, web brute force attack) και τις επιθέσεις που στοχεύουμε να τις εντοπίσουμε με βάση το δικτυακό traffic (DOS attack, επικοινωνία με botnet).

Μετά τις παραπάνω ενέργειες, έμειναν 70 features στο dataset.

Τέλος παρατηρήσαμε ότι πολλές τιμές είχαν σαν τύπο «object» παρότι ήταν αριθμητικές, τις οποίες και μετατρέψαμε σε integer ή float. Μετά την παραπάνω διαδικασία τα δείγματα από κάθε είδος κίνησης και το ποσοστό τους επί του συνόλου της κίνησης που περιέχονται όλες τις μέρες είναι:

ΔΕΙΓΜΑΤΑ	ΑΡΙΘΜΟΣ	ΠΟΣΟΣΤΟ
Benign	*10.317.691*	*88,5066%*
DDoS attacks-LOIC-HTTP	*575.364*	*4,9356%*
DDOS attack-HOIC	*198.861*	*1,7059%*
DoS attacks-Hulk	*145.199*	*1,2455%*
Bot	*144.535*	*1,2398%*
Infiltration	*127.844*	*1,0967%*
SSH-Bruteforce	*94.041*	*0,8067%*
DoS attacks-GoldenEye	*41.406*	*0,3552%*
DoS attacks-Slowloris	*9.908*	*0,0850%*
DDOS attack-LOIC-UDP	*1.730*	*0,0148%*
Brute Force -Web	*551*	*0,0047%*
Brute Force -XSS	*228*	*0,0020%*
SQL Injection	*78*	*0,0007%*
DoS attacks-SlowHTTPTest	*55*	*0,0005%*
FTP-BruteForce	*41*	*0,0004%*
ΣΥΝΟΛΟ	***11.657.532***	***100%***

E.2.3. Class Imbalance

Από τον παραπάνω πίνακα φαίνεται ότι το dataset έχει πολύ μεγάλο class imbalance. Η φυσιολογική κίνηση του δικτύου αντιπροσωπεύει το 88% της συνολικής κίνησης ενώ όλες οι επιθέσεις μαζί το 12%. Ορισμένες από αυτές μάλιστα έχουν διψήφιο αριθμό δειγμάτων τη στιγμή που τα συνολικά δείγματα ξεπερνάνε τα 10εκ.

Με τέτοιες αναλογίες είναι πολύ δύσκολο να εκπαιδευτεί ο αλγόριθμος τεχνητής νοημοσύνης ώστε να μπορεί να αναγνωρίζει τις κατηγορίες με πολύ μικρή εκπροσώπηση. Στην βιβλιογραφία που διαβάσαμε, έχουν δοκιμαστεί πολλές πρακτικές για να αντιμετωπιστεί αυτό το πρόβλημα. Όλες αυτές τις πρακτικές μπορούμε να τις συνοψίσουμε σε δυο μεγάλες κατηγορίες. Το Resampling και τις προσαρμογές στον αλγόριθμο τεχνητής νοημοσύνης. Τέλος για να εξαχθούν τα σωστά συμπεράσματα, πρέπει να χρησιμοποιηθούν οι κατάλληλες μετρήσεις.

E.2.3.1. Resampling

Στη συντριπτική πλειοψηφία της βιβλιογραφίας που διαβάσαμε χρησιμοποιούνται διάφορες τεχνικές resampling. Resampling σημαίνει η αλλαγή των αναλογιών των κατηγοριών του dataset με σκοπό να υπάρχει μια πιο ίση κατανομή. Αυτό γίνεται είτε αφαιρώντας δείγματα από τις πλειοψηφούσες κατηγορίες (Under Sampling), είτε παράγοντας συνθετικά δείγματα από τις μειοψηφούσες κατηγορίες (Over Sampling). Μπορεί να υπάρξει και συνδυασμός των δύο αυτών μεθόδων. Η python έχει συγκεκριμένες βιβλιοθήκες που κάνουν αυτή τη δουλειά. Η πιο διαδεδομένη είναι η imblearn (Imbalanced Learning)[15]. Για το under sampling περιέχει μεθόδους όπως η RandomUnderSampler η οποία αφαιρεί τυχαία δείγματα από κάθε κατηγορία, η NearMiss η οποία αφαιρεί δείγματα που είναι παρόμοια με άλλες κατηγορίες. Για over sampling περιέχει μεθόδους όπως η RandomOverSampler, η SMOTE κ.α.

Η χρήση τέτοιων τεχνικών μπορεί να βελτιώσει πολύ τον αλγόριθμο όμως χρειάζεται προσοχή. Καταρχάς για το Under Sampling πρέπει να έχουμε υπόψη ότι μπορεί να χαθεί πολλή πληροφορία ειδικά αν προβούμε σε μεγάλη μείωση. Ιδιαίτερα ορισμένοι αλγόριθμοι όπως ο NearMiss, ο οποίος αφαιρεί δείγματα τα οποία έχουν μεγάλη ομοιότητα με άλλες κατηγορίες, ενδέχεται να αλλοιώσουν πολύ το dataset, να

απλοποιήσουν το πρόβλημα και όποια θετικά αποτελέσματα παρατηρήσουμε να μην είναι ρεαλιστικά. Αντίστοιχα οι αλγόριθμοι over sampling μπορούν να αυξήσουν το δείγμα με τέτοιο τρόπο που να μην είναι αντιπροσωπευτικός και ρεαλιστικός.

Σε αυτό το σημείο χρειάζεται να επισημάνουμε το εξής. Ένα από τα πρώτα βήματα που περιλαμβάνει η εκπαίδευση ενός μοντέλου τεχνητής νοημοσύνης είναι να χωρίζεται το dataset σε train και test κομμάτια. Το train αξιοποιείται για την εκπαίδευση του και το test για τον τελικό έλεγχο. Ότι resampling κάνουμε, πρέπει να είναι μόνο στο train κομμάτι και όχι στο test. Το test πρέπει να μείνει ανέγγιχτο ώστε να αντιπροσωπεύει όσο το δυνατόν καλύτερα τις πραγματικές συνθήκες. Στο μεγαλύτερο κομμάτι της βιβλιογραφίας που διαβάσαμε, χρησιμοποιείται resampling σε ολόκληρο το dataset, πράγμα που νομίζουμε ότι οδηγεί στην αλλοίωση του προβλήματος και άρα σε μη ρεαλιστικά αποτελέσματα.

Στην υλοποίηση μας χρησιμοποιήσαμε τους αλγορίθμους RandomUnderSampler και RandomOverSampler για να διαμορφώσουμε το train κομμάτι του dataset. Θα παρουσιάσουμε τις αναλογίες σε κάθε μοντέλο τεχνητής νοημοσύνης ξεχωριστά.

E.2.3.2. Προσαρμογές στον αλγόριθμο τεχνητής νοημοσύνης

Μπορούμε καταρχάς να προσαρμόσουμε ορισμένους αλγόριθμους κατάλληλα ώστε να διαχειρίζονται διαφορετικά την κάθε κατηγορία. Ένας τρόπος να το κάνουμε αυτό είναι να ορίσουμε διαφορετικά βάρη για την κάθε κατηγορία, τα οποία συνήθως είναι αντιστρόφως ανάλογα με την συχνότητα τους στο dataset. Αυτό γίνεται συνήθως μέσω της παραμέτρου class_weights η οποία υπάρχει στους περισσότερους αλγόριθμους. Ένας άλλος τρόπος είναι να χρησιμοποιήσουμε συναρτήσεις λάθους που δίνουν μεγαλύτερη ποινή σε λάθος κατηγοριοποίηση «δύσκολων» δειγμάτων όπως αυτών που ανήκουν στις μειοψηφούσες κατηγορίες. Μια τέτοια συνάρτηση που δοκιμάσαμε για το νευρωνικό δίκτυο είναι η focal loss. Τέλος γίνεται να χρησιμοποιήσουμε πολλούς αλγορίθμους μαζί καθένας από τους οποίους εκπαιδεύεται σε μόνο μια κατηγορία και στο τέλος συνυπολογίζονται όλες οι προβλέψεις. Χρησιμοποιήσαμε μια τέτοια μέθοδο για να ενισχύσουμε τον XGBoost και πιο συγκεκριμένα τη One vs the Rest (OVR) από την βιβλιοθήκη sklearn την οποία και θα αναλύσουμε παρακάτω.

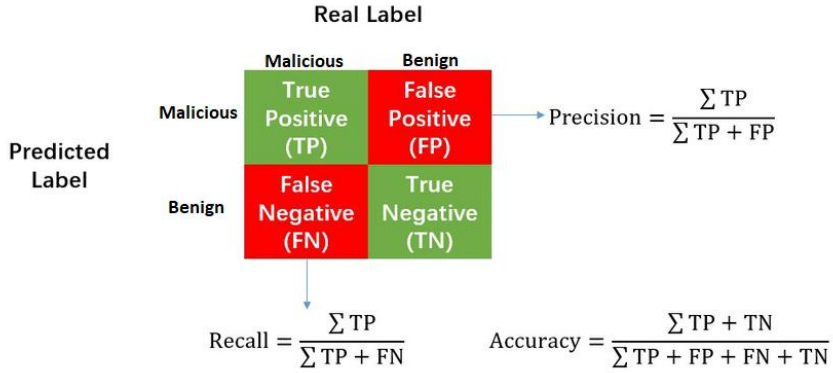
Σε συνδυασμό με τις παραπάνω τεχνικές, προχωρήσαμε και σε «ομαδοποίηση» των επιθέσεων. Μετά την ομαδοποίηση οι κατηγορίες που έμειναν είναι 6 οι οποίες παρουσιάζονται στον παρακάτω πίνακα μαζί με τις αντίστοιχες αναλογίες τους πριν κάνουμε resampling. Θεωρούμε ότι παρόλο που υπάρχει σε ένα βαθμό γενίκευση και απλοποίηση του προβλήματος με την ομαδοποίηση ορισμένων επιθέσεων, αυτή είναι σε αποδεκτό βαθμό και δεν αποτρέπει το σύστημα μας από το να εκπληρώνει επαρκώς τον ρόλο του σαν IDS. Πιο συγκεκριμένα ομαδοποιήσαμε όλες τις Denial of Service επιθέσεις ανεξάρτητα από το πιο εργαλείο χρησιμοποιήθηκε γιατί έχουν παρόμοια χαρακτηριστικά και οδηγούν στο ίδιο αποτέλεσμα. Ομαδοποιήσαμε επίσης τις δυο Brute Force επιθέσεις (FTP, SSH). Τέλος προχωρήσαμε σε ομαδοποίηση των SQL Injection, XSS και WEB Brute Force καθώς αποτελούν βήματα της ίδιας επίθεσης στο DVWA (Damn Vulnerable Web App). Αντίστοιχη ομαδοποίηση είχαν αναφέρει και οι δημιουργοί του dataset στο κομμάτι που παρουσίαζαν τις επιθέσεις.

E.2.3.3. Metrics

Τέλος, σε dataset με μεγάλο class imbalance, έχει σημασία οι μετρήσεις (metrics) που θα χρησιμοποιήσουμε να είναι οι κατάλληλες. Για παράδειγμα στην δική μας περίπτωση η

πιο παραδοσιακή μέτρηση που είναι το accuracy δεν είναι αντιπροσωπευτική. Μπορεί το μοντέλο τεχνητής νοημοσύνης μας να έχει κατηγοριοποιήσει σωστά το 98% των δειγμάτων που φαντάζει καλό, αλλά να μην έχει εντοπίσει σωστά καμία από τις επιθέσεις παρά μόνο την φυσιολογική κίνηση. Για τον λόγο αυτό χρησιμοποιήσαμε και διαφορετικές μετρήσεις που θα παρουσιάσουμε παρακάτω. Πριν προχωρήσουμε όμως χρειάζεται να ορίσουμε ορισμένες μεταβλητές με βάση ένα binary classification πρόβλημα που αφορά την κατηγοριοποίηση κίνησης δικτύου σε Benign (0) και Malicious (1). Ως Negative ορίζουμε την Benign κίνηση και Positive την Malicious. True positives (TP) είναι τα δείγματα τα οποία είναι Malicious και κατηγοριοποιήθηκαν ως Malicious. Τα δείγματα τα οποία είναι Malicious αλλά κατηγοριοποιήθηκαν ως Benign ονομάζονται False Negatives (FN). Τα δείγματα τα οποία είναι Benign και κατηγοριοποιήθηκαν ως τέτοια ονομάζονται True Negatives (TN) ενώ αυτά που κατηγοριοποιήθηκαν ως Malicious False Positives (FP).

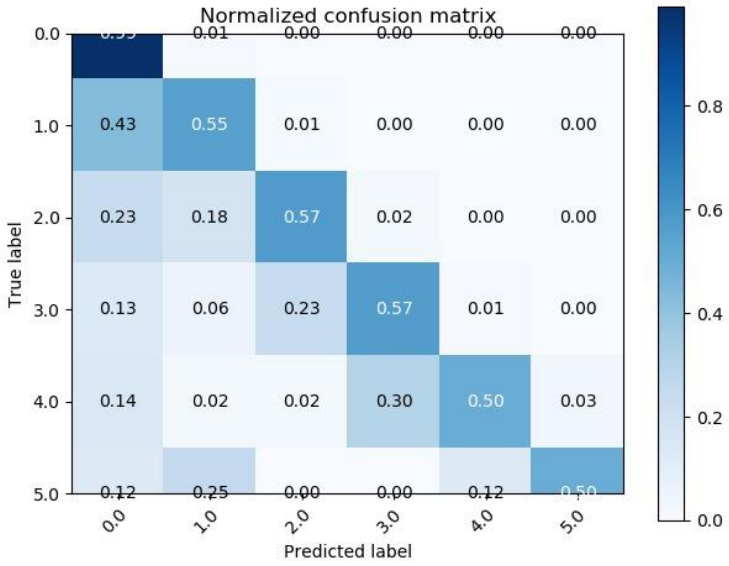
Με βάση τα παραπάνω ορίζουμε ως Precision, το κλάσμα των TP προς το σύνολο των Positives δηλαδή TP + FP. Ως Recall ορίζουμε το κλάσμα των TP προς το σύνολο των TP + FN. Αυτές οι μετρήσεις μας βοηθάνε να δούμε πιο συγκεκριμένα χαρακτηριστικά για την κάθε κατηγορία ενώ μπορούν να γενικευτούν και για ολόκληρη την πρόβλεψη. Παρουσιάζουμε αυτές τις μετρήσεις αναλυτικά στην παρακάτω εικόνα.



Χρησιμοποιήσαμε επίσης την μέτρηση F1 Score η οποία είναι ένας αρμονικός συνδυασμός Precision και Recall και είναι πολύ αντιπροσωπευτική και για dataset με μεγάλο class imbalance. Και αυτή η μέτρηση μπορεί να υπολογιστεί ξεχωριστά για κάθε κατηγορία αλλά και για το σύνολο της πρόβλεψης. Ο τύπος του F1 Score είναι ο παρακάτω:

$$F_1 = \frac{2}{\frac{1}{\text{recall}} \times \frac{1}{\text{precision}}} = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$
$$= \frac{\text{tp}}{\text{tp} + \frac{1}{2}(\text{fp} + \text{fn})}$$

Τέλος χρησιμοποιήσαμε το confusion matrix το οποίο αποτελεί έναν αναλυτικό πίνακα με όλες τις προβλέψεις ανά κατηγορία και έχει τη μορφή που φαίνεται στην παρακάτω εικόνα:



E.2.4. Classification

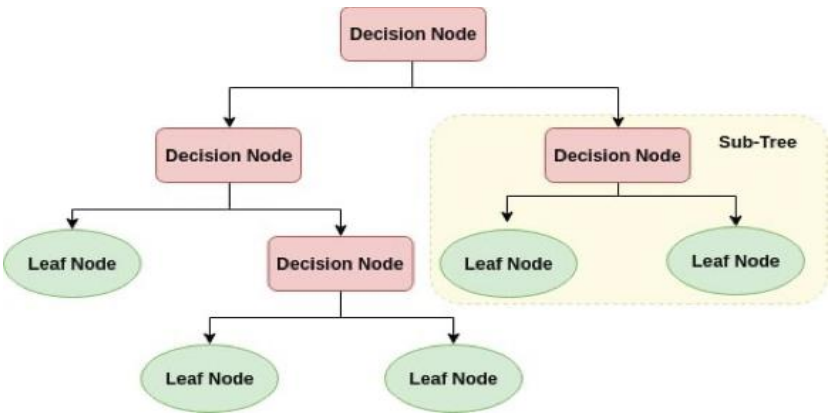
Για την πρόβλεψη των κατηγοριών της κίνησης χρησιμοποιήσαμε δυο διαφορετικά μοντέλα μηχανικής μάθησης. Το πρώτο ονομάζεται XGBoost και ανήκει στην κατηγορία των Decision Trees (Δέντρα αποφάσεων). Το δεύτερο είναι ένα νευρωνικό δίκτυο που συνδυάζει τρείς διαφορετικές μορφές, το Convolutional Neural Network (CNN), το Long Short-Term Memory και το Artificial Neural Network. Για την υλοποίηση τους χρησιμοποιήθηκαν βιβλιοθήκες και modules της Python, όπως το XGBoost, το sklearn, το Tensorflow και το Keras. Παρακάτω θα παρουσιάσουμε γενικές πληροφορίες για τον κάθε αλγόριθμο, τα βήματα που ακολουθήσαμε καθώς και τα αποτελέσματα που πετύχαμε.

E.2.4.1. XGBoost

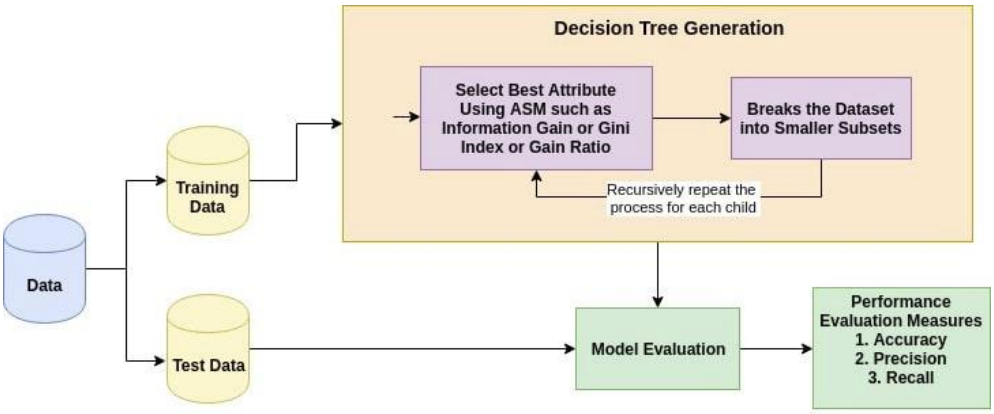
Αρχικά θα αναφερθούμε στη δομή και τον τρόπο λειτουργίας του XGBoost. Στη συνέχεια θα αναλύσουμε τα βήματα που κάναμε στην υλοποίηση μας.

E.2.4.1.1. Δομή και τρόπος λειτουργίας:

Decision Trees: Ο XGBoost ανήκει στην οικογένεια των decision trees. Τα decision trees είναι supervised αλγόριθμοι που χρησιμοποιούνται για classification και regression προβλήματα. Ο στόχος είναι το μοντέλο να προβλέπει την τιμή μιας target μεταβλητής, μαθαίνοντας από κανόνες που δημιούργησε κατά την εκπαίδευση του σε συνδυασμό με τα features.



Τα decision trees μπορούμε να τα φανταστούμε σαν ένα διάγραμμα ροής που μας βοηθάει να πάρουμε αποφάσεις με βάση τις εισόδους (features) που του δίνουμε. Το πρώτο node ονομάζεται και root node. Στην συνέχεια διαχωρίζει να φύλλα με βάση την τιμή που έχει η παράμετρος που χρησιμοποιεί.



Η διαδικασία που ακολουθείται είναι η εξής: Επιλέγεται το καλύτερο attribute για να γίνουν split οι εγγραφές μέσω του Attribute Selection Measures (ASM). Στη συνέχεια γίνεται το attribute ένα decision node και «σπάει» σε μικρότερα subsets. Αυτή η διαδικασία επαναλαμβάνεται μέχρι να ισχύουν τα παρακάτω:

- Όλες οι εγγραφές έχουν το ίδιο attribute value.
- Να μην υπάρχουν άλλα attributes.
- Να έχουν επεξεργαστεί όλα τα δεδομένα

Ορισμένα πλεονεκτήματα που παρουσιάζουν τα decision trees:

- Είναι εύκολο να κατανοηθούν γιατί μπορούν να οπτικοποιηθούν.
- Δεν χρειάζονται μεγάλη προετοιμασία των δεδομένων (preprocessing).
- Το κόστος εκπαίδευσης είναι λογαριθμικό σε συνάρτηση με τα δεδομένα που χρησιμοποιούνται για την εκπαίδευση.
- Μπορούν να χειριστούν αριθμούς καθώς και δεδομένα που είναι χωρισμένα σε κατηγορίες.
- Μπορούν να χειριστούν multi-class προβλήματα.
- Το μοντέλο τους είναι white box, δηλαδή αν μια κατάσταση είναι παρατηρήσιμη, τότε είναι πιο εύκολο να εξηγηθεί με Boolean λογική.

Στα μειονεκτήματα τους συγκαταλέγονται τα παρακάτω:

- Υπάρχει ο κίνδυνος να δημιουργήσουν πολύπλοκα δένδρα και αυτό να έχει σαν αποτέλεσμα overfitting.
- Είναι ασταθή αφού μικρές παραλλαγές στα data δημιουργούν νέα decision trees.
- Υπάρχουν προβλήματα που είναι δύσκολο να λυθούν με αυτούς τους αλγόριθμους, αφού υπάρχουν συναρτήσεις που δεν μπορούν να τις εκφράσουν (π.χ. XOR, parity bit).
- Αν κυριαρχεί μία κλάση στα δεδομένα τα decision trees, είναι πολύ πιθανό στην συνέχεια να χαρακτηρίζει ψευδώς τα δεδομένα με αυτό το label.

Boosting: Ο βασικός αλγόριθμος των boosting μοντέλων είναι σχετικά απλός. Πρακτικά φτιάχνουμε ένα μοντέλο στο training dataset και μετά φτιάχνουμε ένα δεύτερο μοντέλο με το οποίο προσπαθούμε να λύσουμε τα προβλήματα που είχε το πρώτο. Ας υποθέσουμε πως έχουμε δεδομένα με δύο καταστάσεις 0 και 1 και θέλουμε να φτιάξουμε ένα μοντέλο το οποίο θα προσδιορίζει τις κλάσεις στα training δεδομένα. Έτσι φτιάχνουμε το πρώτο μοντέλο και του δίνουμε τα δεδομένα μας στην είσοδο (τα οποία έχουν ένα βάρος που καθορίζει και την πιθανότητα να επιλεχθούν από το μοντέλο) και στην συνέχεια φτιάχνουμε το δεύτερο μοντέλο και του δίνουμε τα δεδομένα που είχαμε στην έξοδο του πρώτου. Έχουμε όμως αυξήσει το βάρος στα δεδομένα που είχαμε λάθος πρόβλεψη, έτσι ώστε να έχουμε περισσότερες πιθανότητες να επιλεχθούν.

Gradient Boosting: Η τεχνική gradient boosting αποτελεί συνέχεια τις τεχνικής boosting. Έχει πάρει το όνομα τις επειδή συνδυάζει 2 τεχνικές, την Gradient Descent και το boosting. Η Gradient Descent είναι μία τεχνική που χρησιμοποιείται για να βρούμε τις παραμέτρους με τις οποίες θα έχουμε την μικρότερη απώλεια στην πρόβλεψη του μοντέλου μας.

XGBoost: Το Xgboost(Extreme Gradient Boosting) χρησιμοποιείται για supervised learning προβλήματα, όπου χρησιμοποιούμε τα δεδομένα(features) για να προβλέψουμε μία target μεταβλητή (y). Όπως ήδη έχει αναφερθεί το boosting προσπαθεί στα νέα δέντρα να βελτιώσει τα λάθη που έκανε στα προηγούμενα, το πετυχαίνει χρησιμοποιώντας μία loss function. Για παράδειγμα στο παρακάτω πρόβλημα:

Dst Port	RST Flag Cnt	bw_psh_flag	PSH Flag Cnt	Label
80	1	0	1	Benign
22	0	0	1	Brute Force -Web
500	1	1	1	Brute Force -XSS
443	1	1	0	Sql injection

Το πρώτο βήμα είναι να φτιάξουμε το πρώτο decision tree. Ας υποθέσουμε ότι το φτιάχνουμε με βάση το feature RST flag Cnt. Αξιολογούμε τις επιδόσεις του δέντρου με την χρήση της μία loss function, για παράδειγμα την cross entropy. Το loss είναι μεγάλο όταν το label και το prediction δεν είναι ίδια και χαμηλό όταν έχουμε κάνει σωστή πρόβλεψη. Φτιάχνοντας το δεύτερο δέντρο, θέλουμε να μειώσουμε τις απώλειες που είχαμε στο πρώτο και έτσι προσπαθούμε να βρούμε που μειώνεται η τιμή στην loss function έτσι ώστε να χρησιμοποιήσουμε εκ νέου αυτές τις εισόδους και να τα διορθώσουμε.

E.2.4.1.2. Υλοποίηση

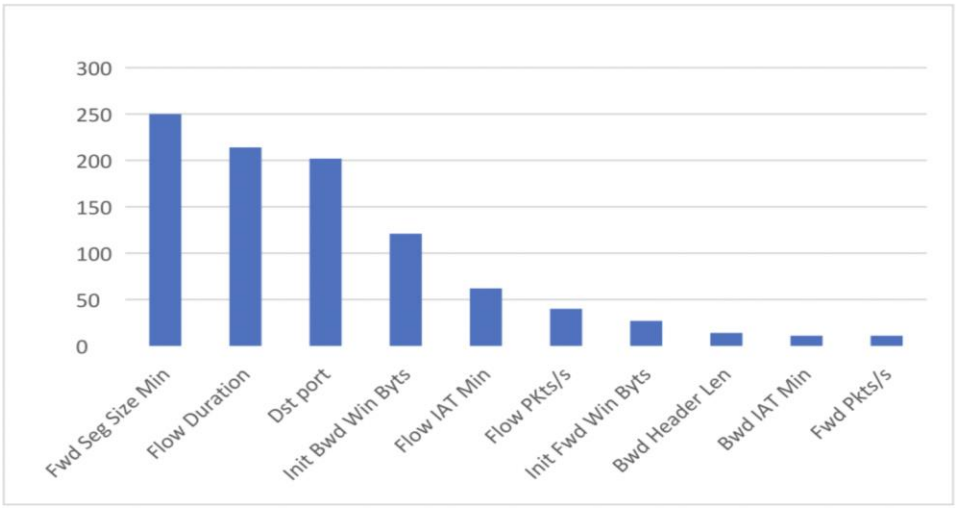
Στην υλοποίηση μας ακολουθήσαμε τα παρακάτω βήματα:

E.2.4.1.2.1. Feature Selection

Feature selection είναι η διαδικασία με την οποία κρατάμε τα features που έχουν την μεγαλύτερη βαρύτητα. Αυτή η διαδικασία βοηθάει στο να αποφύγουμε το overfitting του μοντέλου, αλλά και να αυξήσουμε την απόδοση του, μειώνοντας

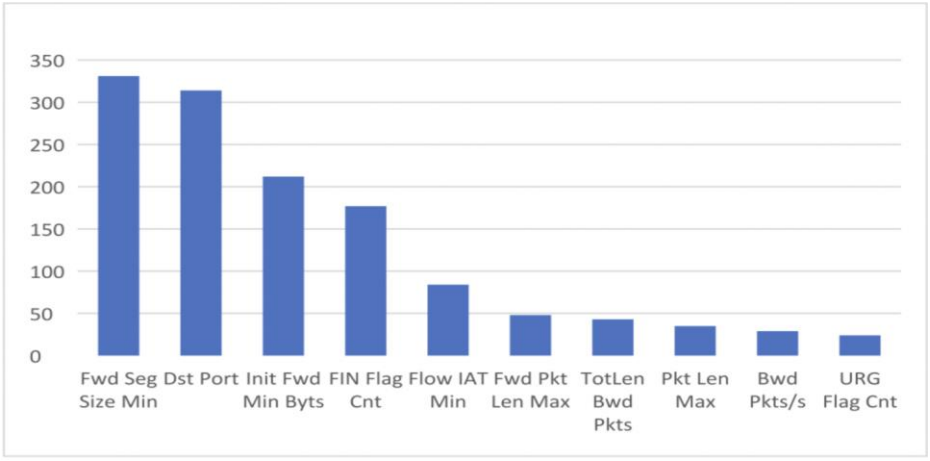
δεδομένα που δεν παίζουν τόσο μεγάλο ρόλο στο αποτέλεσμα. Χρειάζεται προσοχή όμως γιατί με λάθος παραμέτρους μπορεί να χαθεί χρήσιμη πληροφορία. Πραγματοποιήσαμε feature selection χρησιμοποιώντας τον αλγόριθμο random forest. Παρακάτω παραθέτουμε τα σημαντικότερα features για κάθε επίθεση:

Brute Force



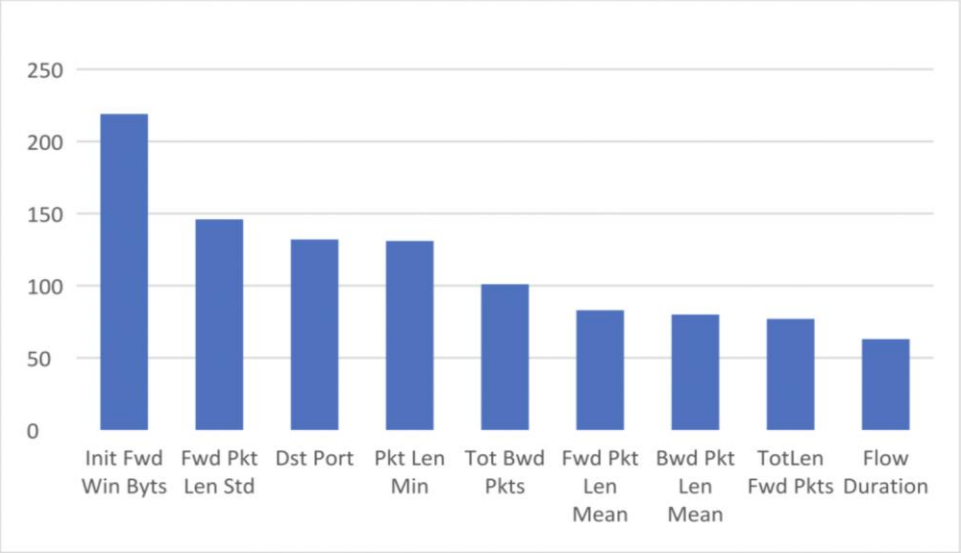
Τα σημαντικότερα features είναι και αυτά που αναδεικνύουν την φύση της επίθεσης, από την στιγμή που ο επιτιθέμενος δοκιμάζει πολλές φορές κωδικούς η δοκιμάζει πρόσβαση σε τυχαίους πόρους, είναι κατανοητό πως τα σημαντικότερα features θα σχετίζονται με την διάρκεια του flow(Flow duration), την διάρκεια του segment (FWD Seg Size Min),τον όγκο των δεδομένων που στέλνονται και λαμβάνονται στην αρχή τις επικοινωνίας (init Bwd Win Byts, init Fwd Win Byts) καθώς και τον χρόνο που περνάει ανάμεσα σε δύο flows.

Dos Attack



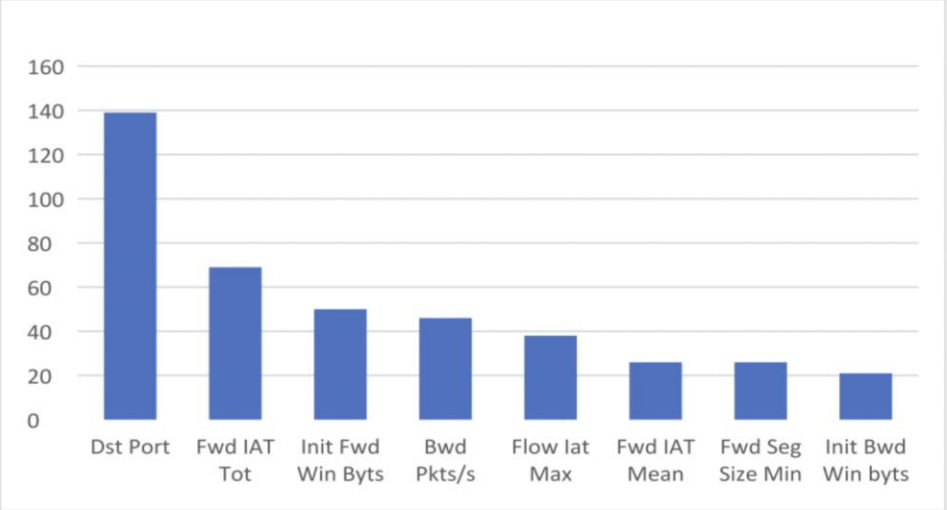
Στόχος της συγκεκριμένης επίθεσης είναι να αυξήσει την χρήση των πόρων για να κάνει την υπηρεσία μη διαθέσιμη. Όπως είναι κατανοητό τα feature που παίζουν σημαντικό ρόλο είναι τα feature που σχετίζονται με το μέγεθος των πακέτων (fw_seg_min, tot_l_fw_pkt), τα bytes που στέλνονται προς τα πίσω σαν απάντηση (init Bwd Win Byts), ο αριθμός των πακέτων που έχουν το flag fin_cnt, καθώς και ο ελάχιστος χρόνος που απέχουν τα 2 flow μεταξύ τους(fl_iat_min).

Web Attack



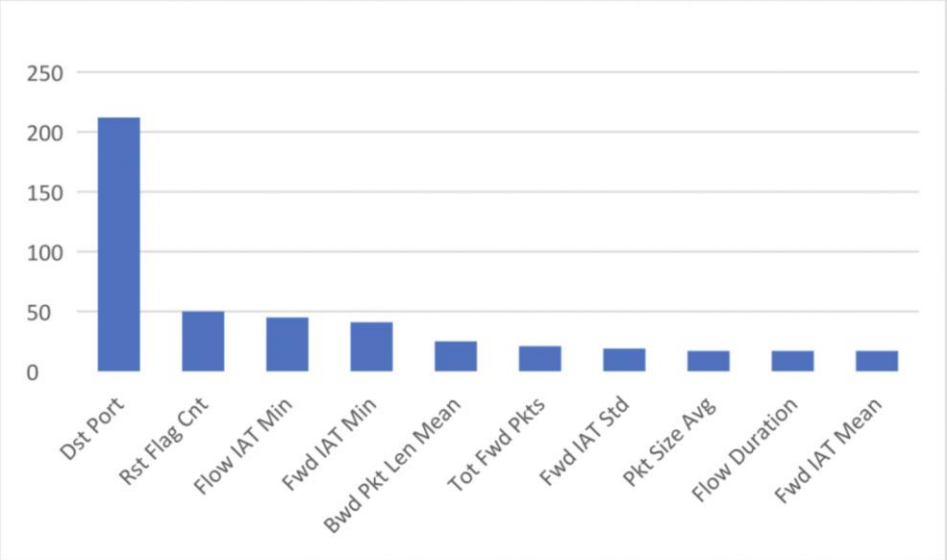
Με βάση και το feature importance τα features που παίζουν τον σημαντικότερο ρόλο για αυτήν την επίθεση είναι ο αριθμός των byte που στάλθηκαν στο αρχικό window προς την destination ip(Init Fwd Win Byts), το destination port, ο αριθμός των πακέτων προς τα εμπρός(Fwd Pkt Len Std), επίσης σημαντικό ρόλο παίζει το μέγεθος των πακέτων που φτάνουν στο destination (TotLen Fwd Pkts, Flow Pkts Len Max, Bwd Pkt Len Max) καθός και το flow duration.

Infiltration



Τα σημαντικά features εδώ είναι το destination port, features που σχετίζονται με την συχνότητα των πακέτων (Fwd IAT Tot, Fwd IAT Mean, Fwd IAT Max), ο όγκος των δεδομένων που στέλνονται και λαμβάνονται στην αρχή τις επικοινωνίας (init Bwd Win Byts, init Fwd Win Byts), ο όγκος των πακέτων που στέλνονται σαν απάντηση(Bwd Pkts/s).

Botnet



Όπως βλέπουμε και από το γράφημα τα σημαντικότερα features είναι αυτά που σχετίζονται με την συχνότητα των πακέτων(Flow IAT Min, Fwd IAT Min, Flow IAT Mean, Fwd IAT std), ο αριθμός των πακέτων που έχουν rst flag (Rst Flag Cnt) το μέγεθος των πακέτων (Bwd Pkt Len Mean, Pkt Size Avg) αλλά το σημαντικότερο και λόγο της φύσης των επιθέσεων είναι το destination port αφού όλα τα bot μιλάνε μέσω τις ίδιας πόρτας στον "orchestrator".

E.2.4.1.2.2. Training and parameter tuning

Στη συνέχεια κάναμε under sample με τέτοιο τρόπο που να κρατάμε τις αναλογίες και να μειώνουμε τα δεδομένα. Η αναλογία φαίνεται στην επόμενη εικόνα:

```
Total occurences of "0" in array:  6913
Total occurences of "1" in array:  630
Total occurences of "2" in array:  651
Total occurences of "3" in array:  590
Total occurences of "4" in array:  850
Total occurences of "5" in array:  969
```

Έπειτα κάνουμε grid search για να βρούμε τις καλύτερες παραμέτρους του μοντέλου χρησιμοποιώντας τον παρακάτω κώδικα:

```
params_new={'max_depth':[4], 'n_estimators': [300,400],
                    'learning_rate':  [0.05],  'subsample':[0.9],  'n_classes':[4],
                    'colsample_bytree':[0.5,0.6,0.7],'colsample_bylevel':0.5
                    'colsample_bynode':0.5}
model_2 = GridSearchCV(XGBClassifier(),params_new,refit=True,verbose=3)
model_2.fit(Xres,yres)
```

Οι παράμετροι με τους οποίους κάνουμε το search στο μοντέλο μας είναι οι παρακάτω:

- max_depth: ορίζει το μέγιστο βάθος του decision tree, εδώ δοκιμάσαμε τις τιμές 0.3, 0.4, 0.5 .

- n_estimators: ο αριθμός των decision trees που θα φτιάξει ο αλγόριθμος εδώ δοκιμάσαμε 300 και 400.
- learning_rate: Ορίζεται το step size που χρησιμοποιείτε για να αποφύγουμε το overfitting.
- Subsample: Ορίζεται την αναλογία των training instances , όταν είναι 0,5 τότε το XGBOOST διαλέγει τυχαίο δείγμα από α data που προορίζονται για το training, πράγμα που αποτρέπει το overfitting.
- n_classes: Ορίζει τον αριθμό των κλάσεων.
- max_depth: Ορίζει το μέγιστο βάθος των decision trees.

Οι καλύτερες παράμετροι φαίνονται παρακάτω

```
Fitting 5 folds for each of 3 candidates, totalling 15 fits
[CV 1/5] END learning_rate=0.05, max_depth=4, n_estimators=300, sampling_method=0.5, subsample=0.7;, score=0.928 total time=  39.6s
[CV 2/5] END learning_rate=0.05, max_depth=4, n_estimators=300, sampling_method=0.5, subsample=0.7;, score=0.931 total time=  39.2s
[CV 3/5] END learning_rate=0.05, max_depth=4, n_estimators=300, sampling_method=0.5, subsample=0.7;, score=0.928 total time=  39.2s
[CV 4/5] END learning_rate=0.05, max_depth=4, n_estimators=300, sampling_method=0.5, subsample=0.7;, score=0.933 total time=  39.0s
[CV 5/5] END learning_rate=0.05, max_depth=4, n_estimators=300, sampling_method=0.5, subsample=0.7;, score=0.924 total time=  39.2s
[CV 1/5] END learning_rate=0.05, max_depth=4, n_estimators=300, sampling_method=0.6, subsample=0.7;, score=0.928 total time=  39.2s
[CV 2/5] END learning_rate=0.05, max_depth=4, n_estimators=300, sampling_method=0.6, subsample=0.7;, score=0.931 total time=  38.8s
[CV 3/5] END learning_rate=0.05, max_depth=4, n_estimators=300, sampling_method=0.6, subsample=0.7;, score=0.928 total time=  38.9s
[CV 4/5] END learning_rate=0.05, max_depth=4, n_estimators=300, sampling_method=0.6, subsample=0.7;, score=0.933 total time=  39.1s
[CV 5/5] END learning_rate=0.05, max_depth=4, n_estimators=300, sampling_method=0.6, subsample=0.7;, score=0.924 total time=  38.9s
[CV 1/5] END learning_rate=0.05, max_depth=4, n_estimators=300, sampling_method=0.7, subsample=0.7;, score=0.928 total time=  39.5s
[CV 2/5] END learning_rate=0.05, max_depth=4, n_estimators=300, sampling_method=0.7, subsample=0.7;, score=0.931 total time=  39.1s
[CV 3/5] END learning_rate=0.05, max_depth=4, n_estimators=300, sampling_method=0.7, subsample=0.7;, score=0.928 total time=  38.9s
[CV 4/5] END learning_rate=0.05, max_depth=4, n_estimators=300, sampling_method=0.7, subsample=0.7;, score=0.933 total time=  38.8s
[CV 5/5] END learning_rate=0.05, max_depth=4, n_estimators=300, sampling_method=0.7, subsample=0.7;, score=0.924 total time=  38.9s
GridSearchCV(estimator=XGBClassifier(),
             param_grid={'learning_rate': [0.05], 'max_depth': [4],
                         'n_estimators': [300],
                         'sampling_method': [0.5, 0.6, 0.7],
                         'subsample': [0.7]},
             verbose=3)
```

Στη συνέχεια επαναλαμβάνουμε όλα τα βήματα αλλά με μεγαλύτερο δείγμα:

```
Total occurences of "0" in array:  691313
Total occurences of "1" in array:  63087
Total occurences of "2" in array:  65140
Total occurences of "3" in array:  40000
Total occurences of "4" in array:  85415
Total occurences of "5" in array:  80000
```

Επιπλέον ορίζουμε τις καλύτερες παραμέτρους που βρέθηκαν παραπάνω:

model_1 = XGBClassifier(random_state = 42, verbosity = 3,scoring = 'f1', learning_rate = 0.05, n_estimators = 300, subsample = 0.7, n_classes= 6, max_depth = 4)
model_1.fit(Xres, yres)

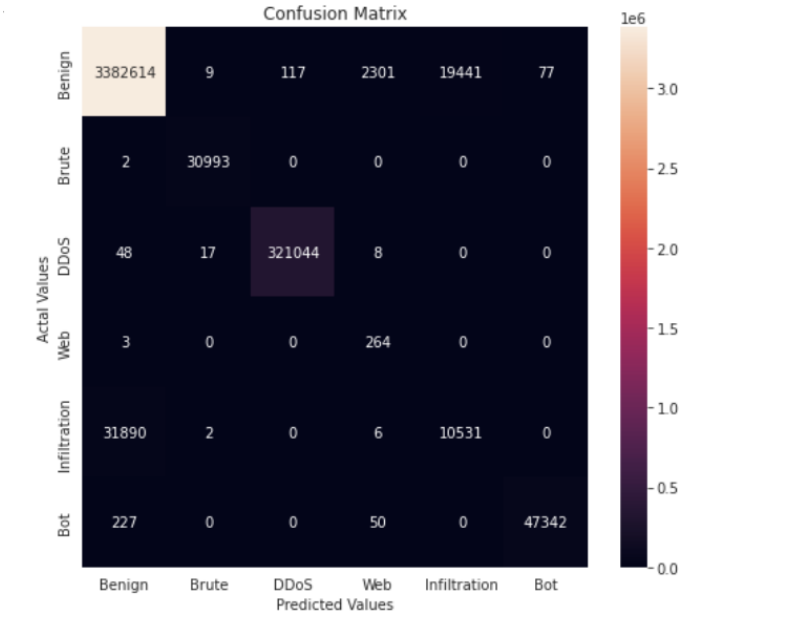
E.2.4.1.3. One Vs The Rest

Πρόκειται για μια μέθοδο που μετατρέπει ένα multiclass classification πρόβλημα σε πολλά binary classification προβλήματα. Το χρησιμοποιήσαμε στην προσπάθεια μας να βελτιώσουμε τα αποτελέσματα σε συγκεκριμένες κατηγορίες όπως το infiltration. Θα εξηγήσουμε πως λειτουργεί με ένα παράδειγμα από το dataset που έχουμε χωρίσει τις κατηγορίες σε Benign, Brute Force, Dos Attack, Web Attack, Infiltration, Botnet. Με βάση την παραπάνω μέθοδο το πρόβλημα μετατρέπεται σε:

- Benign vs [Brute force attack, Dos attack, Web attack, Infiltration, Botnet]
- Brute force attack vs [Benign, Dos attack, Web attack, Infiltration, Botnet]
- Dos attack vs [Benign, Brute force attack, Web attack, Infiltration, Botnet]
- Web attack vs [Benign, Brute force attack, Dos attack, Infiltration, Botnet]
- Infiltration vs [Benign, Brute force attack, Dos attack, Web attack, Botnet]
- Botnet vs [Benign, Brute force attack, Dos attack, Web attack, Infiltration]

E.2.4.1.4. Αποτελέσματα

Το μοντέλο έχει Accuracy 0.9851041828589966. Αυτό όμως και με βάση τα όσα είπαμε ποιο πριν, δεν αρκεί για να βγάλουμε ολοκληρωμένη εικόνα για το μοντέλο μας, αφού όπως φαίνεται και παρακάτω τα δεδομένα μας δεν έχουν ίση κατανομή.



Με την βοήθεια του confusion matrix μπορούμε να βγάλουμε ολοκληρωμένα συμπεράσματα για την απόδοση του μοντέλου μας. Φαίνεται πως έχουμε πρόβλημα σε 2 κλάσεις. Για αρχή στο infiltration όπου 31890 εγγραφές χαρακτηρίστηκαν λανθασμένα ως benign (False negative), ενώ και 19441 εγγραφές από benign κίνηση χαρακτηρίστηκαν εσφαλμένα ως infiltration (False positive). Η δεύτερη κλάση στην οποία έχουμε πρόβλημα είναι οι web επιθέσεις. Αυτό πρακτικά συνέβη γιατί κάναμε μεγάλο σε ποσοστό RandomOverSampler, με αποτέλεσμα να θεωρεί το μοντέλο μας την μη κακόβουλη κίνηση σαν κακόβουλη (2301 False positive). Παρόλο που έχουμε πολύ καλά αποτελέσματα σε πρόβλεψη των επιθέσεων, δημιουργούμε μεγάλο θόρυβο στους αναλυτές πράγμα που δεν είναι αποδεχτό.

Οι υπόλοιπες κλάσεις έχουν πολύ καλά ποσοστά πρόβλεψης πράγμα που από τις μετρήσεις που εξηγήσαμε ποιο πριν. Στο δικό μας μοντέλο το macro π.χ. του precision υπολογίζεται ως (0.99+0.99+0.99+0.06+0.39+0.99)/6 =0.73.

	precision	recall	f1-score	support
0	0.99	0.99	0.99	3404559
1	0.99	1.00	0.99	30995
2	0.99	0.99	0.99	321117
3	0.06	0.98	0.11	267
4	0.39	0.16	0.23	42429
5	0.99	0.99	0.99	47619
accuracy			0.99	3846986
macro avg	0.73	0.85	0.72	3846986
weighted avg	0.98	0.99	0.98	3846986

E.2.4.2. Neural Network

Τα νευρωνικά δίκτυα είναι ένας τύπος αλγορίθμων μηχανικής μάθησης που έχει εμπνευστεί από την λειτουργία του ανθρώπινου εγκεφάλου. Είναι σχεδιασμένα για να μπορούν να επεξεργάζονται και να αναλύουν τεράστιους όγκους δεδομένων και με βάση αυτά να κάνουν προβλέψεις, να παίρνουν αποφάσεις. Διαφέρουν από τους κλασικούς αλγόριθμους μηχανικής μάθησης και στην αρχιτεκτονική και στον τρόπο με τον οποίο επεξεργάζονται τα δεδομένα και μαθαίνουν από αυτά.

Από την μια, οι παραδοσιακοί αλγόριθμοι τεχνητής νοημοσύνης, έχουν μια καθορισμένη δομή, ξεκάθαρους κανόνες με βάση τους οποίους μαθαίνουν και κάνουν προβλέψεις. Βασίζονται σε μαθηματικά μοντέλα όπως η γραμμική παλινδρόμηση (linear regression), τα δέντρα αποφάσεων (decision trees) κ.α.

Από την άλλη, τα νευρωνικά δίκτυα έχουν πιο «black box» αρχιτεκτονική. Αποτελούνται από τεχνητούς νευρώνες (artificial neurons) οι οποίοι επεξεργάζονται τα δεδομένα, είναι κατανεμημένοι σε στρώματα (layers) και διασυνδέονται μεταξύ τους με συσχετίσεις που ονομάζονται βάρη (weights).

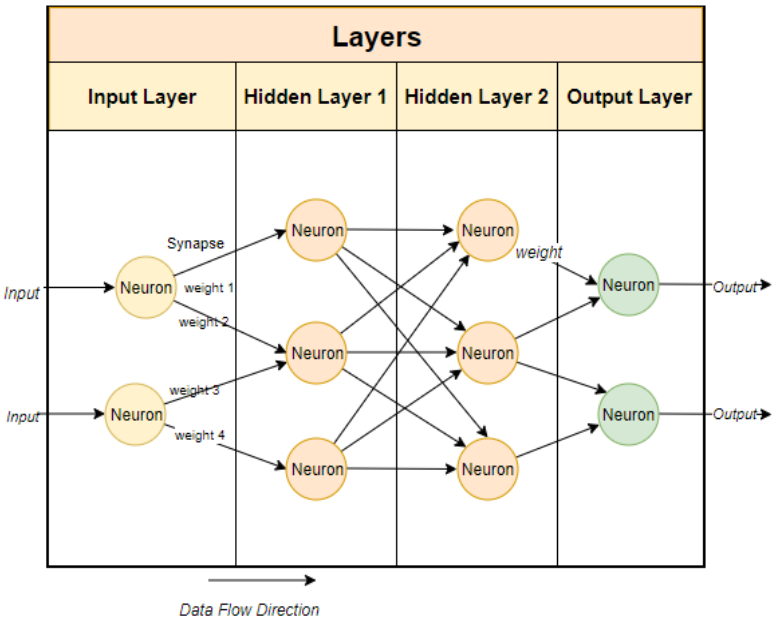
E.2.4.2.1. Δομή

Πιο συγκεκριμένα κάθε νευρωνικό δίκτυο αποτελείται από τρία βασικά μέρη:

Το input layer το οποίο είναι το πρώτο στρώμα του νευρωνικού δικτύου και το οποίο δέχεται τα δεδομένα. Εδώ, κάθε νευρώνας αντιστοιχεί σε ένα γνώρισμα (feature) των δεδομένων που εισάγονται. Για παράδειγμα σε ένα πρόβλημα αναγνώρισης εικόνας, ένα feature μπορεί να είναι ένα pixel της εικόνας και άρα ένας νευρώνας να αντιστοιχεί σε ένα pixel. Σε αυτό το στρώμα, οι νευρώνες δεν κάνουν κανέναν υπολογισμό, δεν επεξεργάζονται την πληροφορία και απλά την μεταφέρουν στο επόμενο στρώμα.

Έπειτα από το input layer, ακολουθεί ένα η περισσότερα **hidden layers**. Εδώ γίνεται όλη η επεξεργασία της πληροφορίας. Κάθε νευρώνας δέχεται είσοδο από τους νευρώνες του προηγούμενου layer, υπολογίζει το σταθμισμένο άθροισμα (weighted sum), στο οποίο εφαρμόζει μια συνάρτηση ενεργοποίησης (activation function) και παράγει ένα αποτέλεσμα το οποίο μεταφέρει στους νευρώνες του επόμενου layer.

Αυτή η διαδικασία, επιτρέπει στο νευρωνικό δίκτυο να μαθαίνει πολύπλοκες συσχετίσεις μεταξύ εισόδου και εξόδου που ξεφεύγουν από την απλή γραμμική συσχέτιση. Ο αριθμός των στρωμάτων και των νευρώνων εξαρτάται από τον τύπο και την πολυπλοκότητα του προβλήματος που καλούμαστε να λύσουμε. Δεν υπάρχει παρ' όλα αυτά κάποια έτοιμη λύση για το πως πρέπει να παραμετροποιηθεί το νευρωνικό δίκτυο για κάθε συγκεκριμένο πρόβλημα, παρά αυτή η παραμετροποίηση προκύπτει μέσα από συνεχείς δοκιμές. Σε γενικές γραμμές, πολλοί νευρώνες και πολλά στρώματα είναι ικανά να λύσουν πιο πολύπλοκα προβλήματα, όμως μετά από ένα σημείο μπορεί να οδηγήσουν το νευρωνικό δίκτυο να απομνημονεύει απλά την είσοδο που δέχεται και να μην είναι σε ικανότητα να γενικεύσει τα μοτίβα και την πληροφορία που δέχεται.

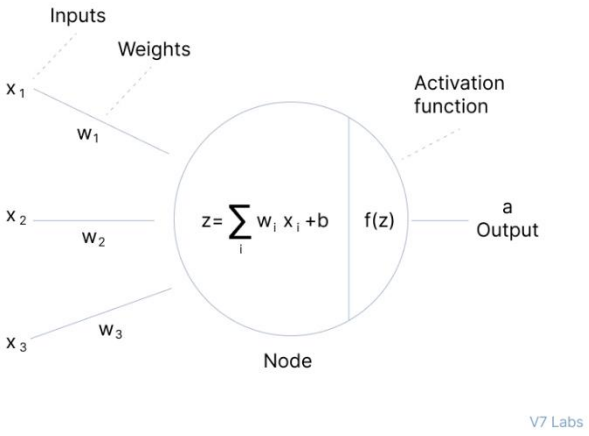


Στο τέλος του κάθε νευρωνικού δικτύου βρίσκεται το **output layer** το οποίο κάνει και την τελική πρόβλεψη με βάση την είσοδο από τα προηγούμενα στρώματα. Και εδώ, ο αριθμός των νευρώνων εξαρτάται από το πρόβλημα που θέλουμε να λύσουμε. Σε ένα πρόβλημα δυαδικής ταξινόμησης (binary classification), αρκεί ένας νευρώνας ο οποίος σαν έξοδο θα έχει μια τιμή μεταξύ 0 και 1 η οποία θα αντιπροσωπεύει την πιθανότητα η είσοδος μας να ανήκει σε μια από τις δυο κατηγορίες. Η έξοδος του νευρώνα περνάει από μια συνάρτηση ενεργοποίησης για να παραχθεί το τελικό αποτέλεσμα.

Άλλα χαρακτηριστικά του νευρωνικού δικτύου:

Συναρτήσεις ενεργοποίησης (Activation Functions): Είναι οι συναρτήσεις που εφαρμόζονται στην έξοδο του κάθε νευρώνα. Καθορίζουν το αν ο νευρώνας πρέπει να ενεργοποιηθεί και να παράξει αποτέλεσμα στηριζόμενες σε ένα κατώφλι (threshold). Ορισμένες γνωστές συναρτήσεις είναι η Sigmoid, η Relu, η Tanh, η Softmax.

Βάρη (Weights): Είναι οι παράμετροι που καθορίζουν το πόσο ισχυρή είναι η σύνδεση μεταξύ των νευρώνων. Αρχικοποιούνται με κάποια τυχαία τιμή και στη συνέχεια ανανεώνονται με σκοπό την βελτίωση της ακρίβειας του δικτύου,



Biases (Προκαταλήψεις): Προστίθενται στην έξοδο του νευρώνα με σκοπό να μετατοπιστεί την συνάρτηση ενεργοποίησης προς μια κατεύθυνση. Βοηθάνε στο να δημιουργηθούν προβλέψεις που δεν καθορίζονται μόνο στην είσοδο

Συνάρτηση απόκλισης (Loss Function): Χρησιμοποιείται για να μετρηθεί το λάθος μεταξύ της πρόβλεψης του δικτύου και της πραγματικής κατάστασης. Αποτελεί τον “οδηγό" στη διαδικασία της εκπαίδευσης με βάση του οποίου ανανεώνονται τα βάρη με σκοπό την μείωση του λάθους. Ορισμένες γνωστές συναρτήσεις απόκλισης είναι η binary cross-entropy, η categorical cross-entropy, η mean squared error κτλ

E.2.4.2.2. Εκπαίδευση

Σε αυτό το κεφάλαιο θα παρουσιάσουμε το πως «μαθαίνει» ένα νευρωνικό δίκτυο. Σκοπός της διαδικασίας της εκπαίδευσης είναι η παραμετροποίηση των βαρών και των “προκαταλήψεων” ώστε το δίκτυο να είναι σε θέση να κάνει καλύτερες προβλέψεις. Σε γενικές γραμμές, η εκπαίδευση περιλαμβάνει τα παρακάτω βήματα:

Αρχικοποίηση: Τα βάρη και τα biases αρχικοποιούνται τυχαία και δίνεται στο δίκτυο η πρώτη δέσμη δεδομένων εκπαίδευσης.

Μετάδραση (feedforward): Τα δεδομένα εκπαίδευσης επεξεργάζονται από το δίκτυο και παράγεται το πρώτο αποτέλεσμα.

Υπολογισμός απόκλισης (loss calculation): Η τελική πρόβλεψη συγκρίνεται με το πραγματικό αποτέλεσμα και μια συνάρτηση (loss function) χρησιμοποιείται για να υπολογιστεί η διαφορά μεταξύ των δυο τιμών. Σκοπός της διαδικασίας είναι να ελαχιστοποιηθεί αυτή η διαφορά.

Οπισθοδιάδοση (backpropagation): Το λάθος στην πρόβλεψη “διαδίδεται” προς τα πίσω στο δίκτυο, τα βάρη και τα biases ανανεώνονται για να μειωθεί το λάθος. Ο τρόπος με τον οποίο γίνεται η ανανέωση είναι μέσω μιας συνάντησης “κλίσης καθόδου” (gradient descent)

Επανάληψη: Όλη αυτή η διαδικασία επαναλαμβάνεται αρκετές φορές, ώστε να φτάσει το λάθος σε αποδεκτά επίπεδα.

Δοκιμή: Η απόδοση του δικτύου εξετάζεται σε διαφορετικά δεδομένα από αυτά που χρησιμοποιήθηκαν στην εκπαίδευση, έτσι ώστε να βγει ένα πιο ασφαλές συμπέρασμα για την ικανότητα και την ακρίβεια του.

Η όλη διαδικασία της εκπαίδευσης είναι χρονοβόρα και συνήθως απαιτεί πολλούς υπολογιστικούς πόρους. Είναι όμως απαραίτητη για να μπορεί το νευρωνικό δίκτυο να κάνει σωστές προβλέψεις. Η επιλογή παραμέτρων με τις οποίες θα γίνει η εκπαίδευση παίζουν σημαντικό ρόλο στο τελικό αποτέλεσμα.

Τα παραπάνω χαρακτηριστικά των νευρωνικών δικτύων, η δομή τους, η διαδικασία με την οποία μαθαίνουν από τις πληροφορίες τα κάνουν πολύ πιο ευέλικτα, ισχυρά και ικανά να λύνουν πολλούς διαφορετικούς τύπους προβλημάτων. Σήμερα που η υπολογιστική ισχύς έχει αυξηθεί σε πολύ μεγάλο βαθμό, όλο και συχνότερα βλέπουμε νευρωνικά δίκτυα να χρησιμοποιούνται για να λύσουν πολλά διαφορετικά προβλήματα. Ορισμένα από αυτά είναι η αναγνώριση εικόνων (Image recognition), η αναγνώριση ομιλίας (Speech recognition), η Επεξεργασία φυσικής γλώσσας (Natural language processing), η Πρόβλεψη χρονικών σειρών (Time series forecasting), η Ανίχνευση ανωμαλιών (Anomaly detection) και πολλά άλλα.

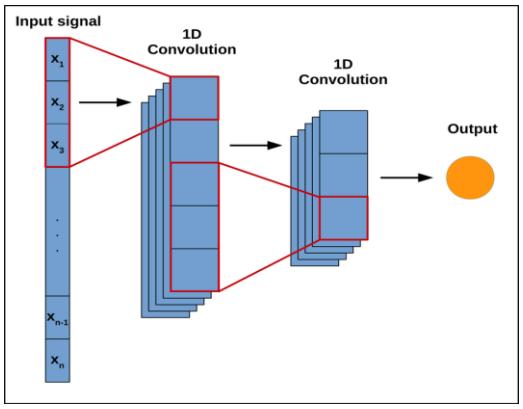
Οδηγούμενοι από αυτές τις δυνατότητες, επιλέξαμε να δημιουργήσουμε ένα απλό μοντέλο νευρωνικού δικτύου προκειμένου να δούμε τις δυνατότητες του σε ένα δύσκολο και πολυδιάστατο πρόβλημα, όπως αυτό της ανίχνευσης απειλών σε ένα δίκτυο. Στην προσπάθεια μας αυτή, ανακαλύψαμε πολλούς διαφορετικούς τύπους νευρωνικών δικτύων καθένα από τα οποία “ειδικεύεται” στο να λύνει διαφορετικού τύπου προβλήματα. Μετά από έρευνα, καταλήξαμε σε δομή η οποία συνδυάζει τρεις διαφορετικούς τύπους νευρωνικού δικτύου τους οποίους και θα παρουσιάσουμε παρακάτω.

E.2.4.2.3. Convolutional Neural Networks

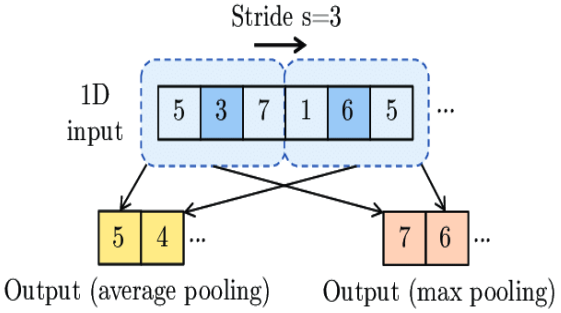
Πρόκειται για τύπο νευρωνικού δικτύου ο οποίος είναι εμπνευσμένος από τη λειτουργία του οπτικού φλοιού, του μέρους του εγκεφάλου που είναι υπεύθυνο για την επεξεργασία οπτικής πληροφορίας και εξειδικεύεται στην επεξεργασία πληροφορίας ακίνητων ή κινούμενων αντικειμένων καθώς και στην αναγνώριση προτύπων. Όπως γίνεται αντιληπτό, τα CNN χρησιμοποιούνται για ανάλογους σκοπούς, όπως για παράδειγμα την κατηγοριοποίηση εικόνων, την αναγνώριση προσώπων κτλ. Για αυτούς τους σκοπούς χρησιμοποιούνται συνήθως δυσδιάστατα (2D) CNN. Υπάρχουν όμως και μονοδιάστατα (1D) CNN, είδος που χρησιμοποιήσαμε στην εργασία μας, τα οποία χρησιμοποιούνται για να εξάγουν χαρακτηριστικά και μοτίβα σε αριθμητικά δεδομένα, χρονοσειρές ή κείμενο. Στην γενική τους μορφή, τα CNN αποτελούνται από πολλά στρώματα συστροφής (convolution layers) και ομαδοποίησης (pooling layers).

Σκοπός των convolution layers είναι να αναγνωρίζουν μοτίβα και να εξάγουν σημαντικά γνωρίσματα της πληροφορίας που δέχονται μέσω μιας μαθηματικής πράξης που ονομάζεται συστροφή (convolution). Κάθε στρώμα περιέχει “φίλτρα”, καθένα από τα οποία έχει ένα μικρό δεκτικό πεδίο και λειτουργεί σε ένα συγκεκριμένο υποπεδίο του input. Τα φίλτρα κινούνται διαδοχικά πάνω στο input και υπολογίζουν το εσωτερικό γινόμενο των βαρών (weights) και των γνωρισμάτων

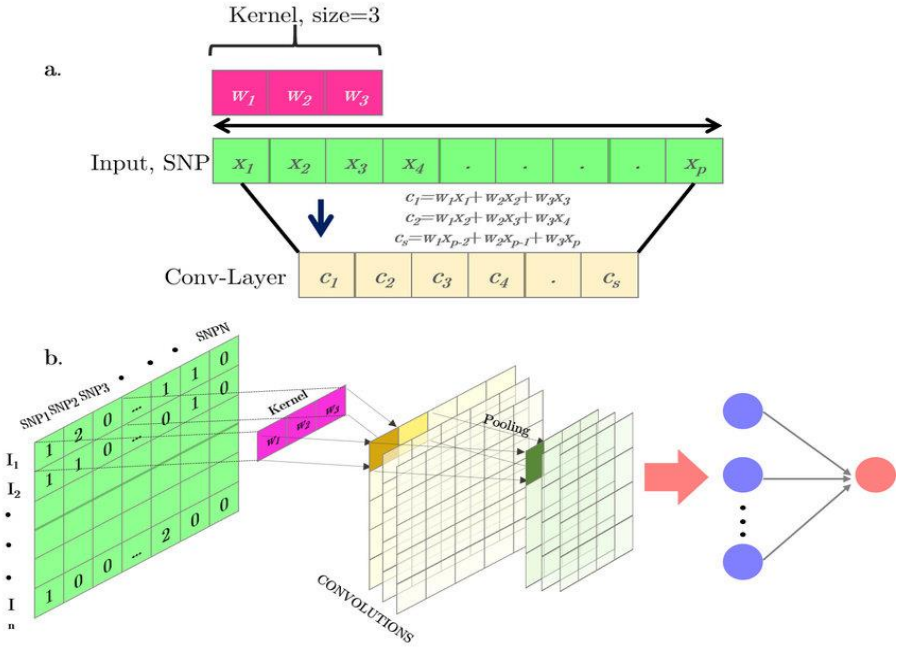
(features) που περιλαμβάνονται στο δεκτικό τους πεδίο την κάθε φορά. Έτσι παράγουν τα feature maps τα οποία αντιπροσωπεύουν συγκεκριμένα γνωρίσματα του input όπως σχήματα, άκρες, γωνίες αν πρόκειται για εικόνες, ή τάσεις, περιοδικά μοτίβα, επαναλήψεις, αν πρόκειται για αριθμητικά δεδομένα, χρονοσειρές ή κείμενο. Στην διπλανή εικόνα φαίνεται η διαδικασία αυτή σε ένα 1D CNN.



Μετά τα convolution layers ακολουθούν συνήθως τα pooling layers τα οποία διαιρούν τα feature maps σε κομμάτια και κρατάνε συνήθως μια τιμή από αυτά (είτε τη μέγιστη είτε την μέση). Με αυτόν τον τρόπο κρατιέται η πιο σημαντική πληροφορία, μειώνεται ο "θόρυβος" και γίνεται λιγότερο πολύπλοκη η πληροφορία. Η διαδικασία αυτή φαίνεται στην διπλανή φωτογραφία.

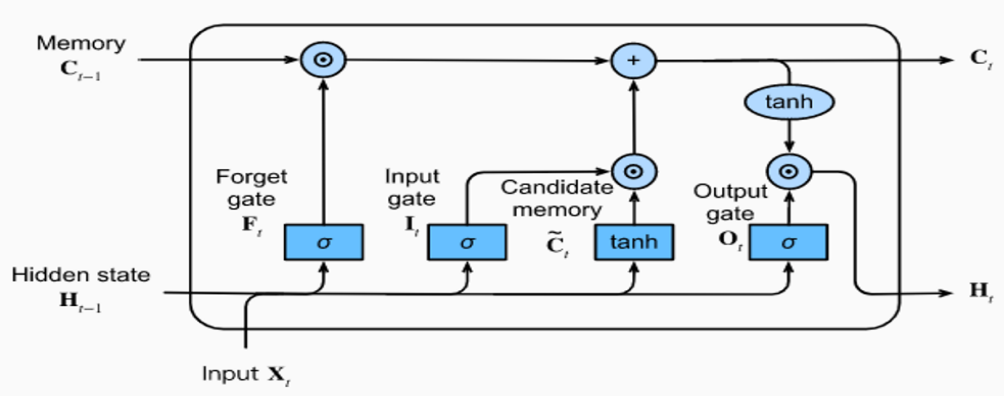


Συνολικά η δομή ενός ConvNet φαίνεται παρακάτω:



E.2.4.2.4. Long Short-Term Memory (LSTM)

Τα LSTM ανήκουν στην κατηγορία των Recurrent Neural Networks (RNN) και ειδικεύονται στην ανάλυση δεδομένων που περιέχουν ακολουθίες, όπως χρονοσειρές, ομιλία ή κείμενο. Οι νευρώνες LSTM, περιέχουν "μνήμη" η οποία επιτρέπει να διατηρείται μόνο η χρήσιμη πληροφορία από μια προηγούμενη κατάσταση και να απορρίπτεται οποιαδήποτε άλλη. Έτσι τα LSTM είναι ικανά να ανιχνεύουν πιο μακροπρόθεσμα μοτίβα και μεταβολές στα δεδομένα. Σε συνδυασμό με άλλες μεθόδους, μπορούν να βοηθήσουν και στην ανίχνευση ανωμαλιών (anomaly detection) όπως αυτή που κάνουμε εμείς στην παρούσα εργασία. Η δομή των LSTM cells φαίνεται παρακάτω:



E.2.4.2.5. Artificial (Feed-forward) Neural Networks

Πρόκειται για τον κλασικό τύπος νευρωνικού δικτύου που έχει την δομή που περιγράφηκε στην εισαγωγή του κεφαλαίου. Αυτό το κομμάτι του δικτύου είναι υπεύθυνο για την κατηγοριοποίηση (classification) της κίνησης.

E.2.4.2.6. Υλοποίηση

Η δομή του νευρωνικού μας δικτύου φάινεται στην παρακάτω εικόνα.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv1d (Conv1D)             (None, 68, 16)            64

 batch_normalization (BatchN  (None, 68, 16)           64
 ormalization)

 conv1d_1 (Conv1D)           (None, 66, 32)            1568

 batch_normalization_1 (Batc  (None, 66, 32)           128
 hNormalization)

 max_pooling1d (MaxPooling1D  (None, 33, 32)           0
 )

 conv1d_2 (Conv1D)           (None, 31, 64)            6208

 batch_normalization_2 (Batc  (None, 31, 64)           256
 hNormalization)

 conv1d_3 (Conv1D)           (None, 29, 128)           24704

 batch_normalization_3 (Batc  (None, 29, 128)          512
 hNormalization)

 max_pooling1d_1 (MaxPooling  (None, 14, 128)          0
 1D)

 lstm (LSTM)                 (None, 256)               394240

 dense (Dense)               (None, 512)               131584

 dropout (Dropout)           (None, 512)               0

 dense_1 (Dense)             (None, 512)               262656

 dropout_1 (Dropout)         (None, 512)               0

 dense_2 (Dense)             (None, 6)                 3078

=================================================================
Total params: 825,062
Trainable params: 824,582
Non-trainable params: 480
_________________________________________________________________
```

Δέχεται το input που αποτελείται από 70 features και το προωθεί στα Convolution Layers με σκοπό να εξάγουν τα features. Το κομμάτι αυτό του δικτύου αποτελείται από 2 συνεχόμενα Conv1D layers τα οποία ακολουθούνται από ένα batch normalisation layer και ένα MaxPooling layer. Η διαδικασία αυτή επαναλαμβάνεται δυο φορές. Μετά το τέλος της διαδικασίας τα features έχουν μειωθεί από 70 σε 14, ενώ έχουν παραχθεί 128 feature maps. Το αποτέλεσμα δίνεται ως input σε ένα layer LSTM με 256 νευρώνες το οποίο και επεξεργάζεται με σκοπό να αναγνωρίσει μοτίβα. Τέλος, το input προωθείται σε 3 Dense layers, τα οποία κάνουν το classification και αποτελούνται από 512, 512 και 6 νευρώνες. Το τελευταίο dense layer έχει 6 νευρώνες, όσες και οι κατηγορίες κίνησης που έχουμε επιλέξει. Η συνάρτηση ενεργοποίησης που έχουμε επιλέξει εδώ είναι η softmax, η οποία παράγει ένα διάνυσμα που αντιπροσωπεύει πιθανότητες το input μας να ανήκει σε κάθε μια από τις 6 κατηγορίες. Για παράδειγμα στο παρακάτω δείγμα από inputs του test dataset μας βλέπουμε τα εξής:

Test	Predictions
[0 0 0 0 1 0]	[0.184 0. 0. 0. 0.816 0.]
[1 0 0 0 0 0]	[0.909 0. 0. 0. 0.091 0.]
[1 0 0 0 0 0]	[0.964 0. 0. 0. 0.036 0.]
[1 0 0 0 0 0]	[0.863 0. 0. 0. 0.137 0.]
[1 0 0 0 0 0]	[0.932 0. 0. 0. 0.068 0.]
[0 0 1 0 0 0]	[0. 0. 1. 0. 0. 0.]

Για το πρώτο δείγμα που ανήκει στην κατηγορία Infiltration (4), το δίκτυο μας κάνει την πρόβλεψη ότι κατά 81% ανήκει σε αυτήν την κατηγορία, ενώ εκτιμάει ότι υπάρχει και 18% πιθανότητα να είναι καλόβουλη κίνηση (Benign 0). Τα επόμενα 4 δείγματα που αποτελούν καλόβουλη κίνηση τα κατηγοριοποιεί ως τέτοια με πολύ υψηλές πιθανότητες ενώ είναι απόλυτα σίγουρο (100% πιθανότητα) ότι το τελευταίο δείγμα ανήκει στην κατηγορία Dos όπως και είναι στην πραγματικότητα.

Σε αυτό το κομμάτι, θα εξηγήσουμε πιο αναλυτικά και συγκεκριμένα την διαδικασία εκπαίδευσης του νευρωνικού μας δικτύου, έχοντας ως βάση τους γενικότερους άξονες που αναλύσαμε παραπάνω.

Αναφέραμε παραπάνω ότι η εκπαίδευση είναι μια διαδικασία που απαιτεί υπολογιστικούς πόρους και χρόνο, αφού περιλαμβάνει την βελτιστοποίηση πάρα πολλών παραμέτρων του νευρωνικού δικτύου. Το δίκτυο μας συγκεκριμένα, περιλαμβάνει 824.582 εκπαιδεύσιμες παραμέτρους (weights, biases). Εκτός όμως από αυτές τις παραμέτρους, υπάρχουν και παράμετροι οι οποίες καθορίζουν την "κατεύθυνση" τον ρυθμό, τον τρόπο της ίδιας της εκπαίδευσης, τις περισσότερες από τις οποίες τις καθορίζουμε πριν την έναρξη της διαδικασίας. Οι παράμετροι αυτοί ονομάζονται hyper-parameters. Η εύρεση των κατάλληλων hyper-parameters, είναι και αυτή μια χρονοβόρα διαδικασία. Παρότι υπάρχουν ορισμένες γενικές πρακτικές, δεν είναι πάντα βέβαιο ότι θα βοηθήσουν στο συγκεκριμένο πρόβλημα που καλούμαστε να λύσουμε κάθε φορά. Ως αποτέλεσμα ακολουθείται μια διαδικασία trial and error, όπου δοκιμάζονται πολλές διαφορετικές λύσεις ώσπου να πετύχουμε το καλύτερο αποτέλεσμα. Ορισμένες από τις hyper parameters που διαλέξαμε εμείς για το νευρωνικό μας δίκτυο είναι οι παρακάτω:

Loss function: Για προβλήματα classification με πολλές κατηγορίες όπως το δικό μας (multiclass classification) η πιο σύνηθης συνάρτηση απόκλισης είναι η categorical cross-entropy. Μελετάει την απόκλιση μεταξύ της κατανομής πιθανοτήτων των προβλέψεων (predicted labels) και των πραγματικών κατηγοριών (true labels). Ο υπολογισμός της απόκλισης γίνεται αθροίζοντας όλα τα δείγματα όλων των κατηγοριών που υπάρχουν σε ένα batch (δέσμη) και δίνεται από τον παρακάτω τύπο:

$$-\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{C}\mathbf{1}_{y_i \in C_c}\log\left(p_{\text{model}}[y_i \in C_c]\right)$$

Ν είναι ο αριθμός των δειγμάτων,
C ο αριθμός των κατηγοριών,
Ο αριθμός 1 δείχνει αν το δείγμα ανήκει στην συγκεκριμένη κατηγορία (αν δεν ανήκει θα είναι 0),
Το p_model [yi in Cc] είναι η πιθανότητα το y να ανήκει στην C

Για παράδειγμα, για ένα από τα δείγματα που δώσαμε παραπάνω η απόκλιση υπολογίζεται ως εξής:

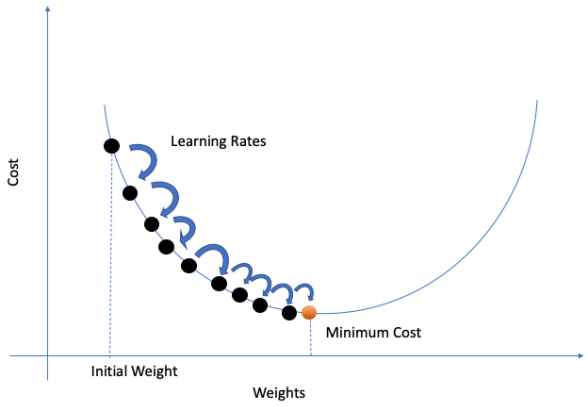
Test	Predictions
[0 0 0 0 1 0]	[0.184 0. 0. 0. 0.816 0.]

*Loss = -[0 * $\log_2$(0.184) + 0 * $\log_2$(0) + 0 * $\log_2$(0) + 0 * $\log_2$(0) + 0 * $\log_2$(0.816) + 0 * $\log_2$(0)]*

Loss = -0.293

Optimizer: Είναι ο αλγόριθμος που ανανεώνει τις παραμέτρους (weights, biases) του νευρωνικού δικτύου με σκοπό να ελαχιστοποιηθεί η συνάρτηση απόκλισης. Χρησιμοποιήσαμε έναν από τους πιο διαδεδομένους αλγόριθμους που ονομάζεται Adam

Learning rate: Ο ρυθμός με τον οποίο ανανεώνονται οι παράμετροι. Μετά από πολλές δοκιμές καταλήξαμε σε αρχικό learning rate 1e-04 το οποίο θα μειώνεται αυτόματα όταν βλέπουμε ότι δε μειώνεται το loss μετά από κάποιο διάστημα. Για να το υλοποιήσουμε αυτό χρησιμοποιήσαμε τη συνάρτηση ReduceLROnPlateau από την βιβλιοθήκη Callbacks του Keras.



Epochs: Η παράμετρος αυτή δείχνει πόσες φορές θα περάσει όλο το dataset για να εκπαιδευτεί το νευρωνικό μας δίκτυο. Λίγα epochs οδηγούν σε ελλειπή εκπαίδευση,

ενώ πάρα πολλά μπορεί να οδηγήσουν σε overfitting. Επιλέξαμε η διάρκεια της εκπαίδευσης να είναι 100 epochs. Επίσης χρησιμοποιήσαμε δυο συναρτήσεις από την βιβλιοθήκη callbacks του Keras, την EarlyStopping, με την οποία σταματάμε αυτόματα την εκπαίδευση αν βλέπουμε ότι το νευρωνικό δίκτυο δε βελτιώνεται σε ορισμένες μετρήσεις και την ModelCheckpoint η οποία αποθηκεύει τις παραμέτρους με το καλύτερο αποτέλεσμα απ όλα τα epochs. Τ

Batch size: Το νευρωνικό δίκτυο δεν εκπαιδεύεται χρησιμοποιώντας όλο το dataset μονομιάς. Αντιθέτως, δέχεται το dataset κομμάτι-κομμάτι το μέγεθος του οποίου το καθορίζουμε με αυτή την παράμετρο. Συνήθως οι τιμές για το batch size κυμαίνονται μεταξύ 32 και 256. Στην επιλογή τους παίζει ρόλο το μέγεθος του dataset, ο αριθμός των κατηγοριών που υπάρχουν αλλά και η κατανομή τους. Στο δικό μας dataset για παράδειγμα που περιλαμβάνει εκατομμύρια δείγματα και έχει άνιση κατανομή ανάμεσα στις διαφορετικές κατηγορίες, αν ορίσουμε ένα πολύ μικρό batch size, μπορεί να χάσουμε πολλή πληροφορία, αφού μπορεί να μην περιλαμβάνει καθόλου μια συγκεκριμένη κατηγορία που αντιπροσωπεύει ένα πολύ μικρό δείγμα του συνόλου. Μετά από πολλές δοκιμές, είδαμε ότι τα καλύτερα αποτελέσματα τα πετύχαμε με batch size 128.

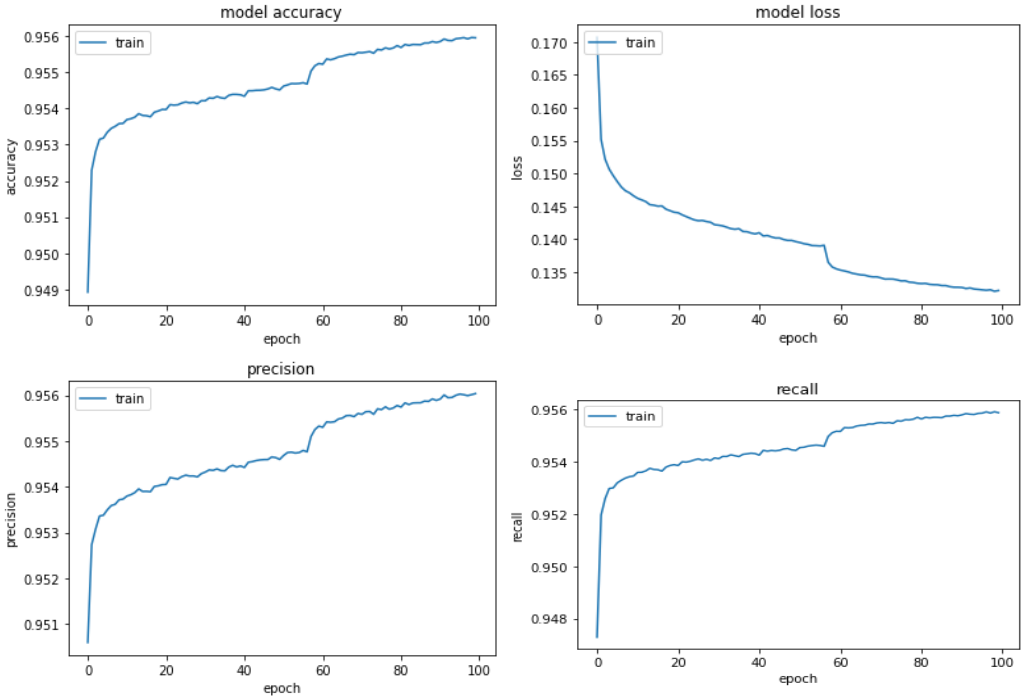
Train/Test split ratio: Λόγω της άνισης κατανομής σε κατηγορίες, χρειάστηκε να πειραματιστούμε αρκετά με τις αναλογίες των train, validation και test dataset. Η αναλογία που καταλήξαμε για το νευρωνικό δίκτυο είναι Train 60%, Validation 20%, Test 20%. Θεωρήσαμε ότι όσο πιο μεγάλα είναι τα validation και test κομμάτια, τοσο περισσότερες είναι οι πιθανότητες να αντιπροσωπεύουν όλες τις κατηγορίες ικανοποιητικά. Πειραματιστήκαμε επίσης με το training κομμάτι. Εκπαιδεύοντας το δίκτυο με το train dataset χωρίς να κάνουμε κάποια αλλαγή, δεν παρατηρήσαμε καλά αποτελέσματα. Το δίκτυο μας είχε δυσκολία να εντοπίσει κατηγορίες με πολύ μικρότερη αναλογία όπως τα Web Attacks και γενικότερα είχε μια ροπή προς την Benign κατηγορία που ήταν η συντριπτική πλειοψηφία. Χρησιμοποιήσαμε αλγόριθμους UnderSampling και OverSampling για να διαμορφώσουμε διαφορετικές αναλογίες. Μετά από αρκετές δοκιμές, η αναλογία με την οποία πετύχαμε τα καλύτερα αποτελέσματα ήταν η παρακάτω:

Class	Resampled train dataset samples	Resampled train dataset ratio	Original train dataset samples	Original train dataset ratio
Benign	1.000.000	70%	6.190.615	88%
Brute	56.449	3%	56.449	0.8%
Dos	200.000	14%	583.514	8%
Web	5.000	0.3%	514	0.008%
Infiltration	76.706	5%	76.706	1%
Bot	86.721	6%	86.721	1%

Η ίδια η αρχιτεκτονική του δικτύου, οι συναρτήσεις ενεργοποίησης για κάθε layer, αποτελούν hyper parameters. Αναλύσαμε συγκεκριμένα την δομή και το σκεπτικό μας παραπάνω. Αξίζει να σημειώσουμε ότι πέρα από γενικές κατευθύνσεις για το τι είδους layers θα χρησιμοποιήσουμε, τις υπόλοιπες παραμέτρους, δηλαδή πόσα layers, πόσους νευρώνες το καθένα ποιές συναρτήσεις ενεργοποίησης, τις επιλέξαμε και αυτές μέσω διαδικασίας trial and error, δηλαδή κάναμε πάρα πολλές δοκιμές και επιλέξαμε εκείνες με το καλύτερο αποτέλεσμα.

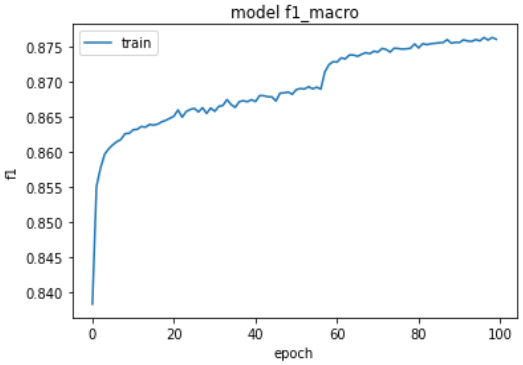
E.2.4.2.7. Αποτελέσματα

Παρακάτω θα παρουσιάσουμε γραφήματα από τη διαδικασία της εκπαίδευσης καθώς και τα αποτελέσματα του μοντέλου στο test dataset.



Βλέπουμε ότι το μοντέλο μας αποδίδει συνολικά καλά στο training set έχοντας accuracy, precision και recall περίπου 96%. Το f1 score είναι πιο χαμηλά περίπου στο 87%.

Στο test set πέτυχε accuracy 98.7%, ενώ εδώ το f1 είναι 80%, όπως φαίνεται στο παρακάτω print.

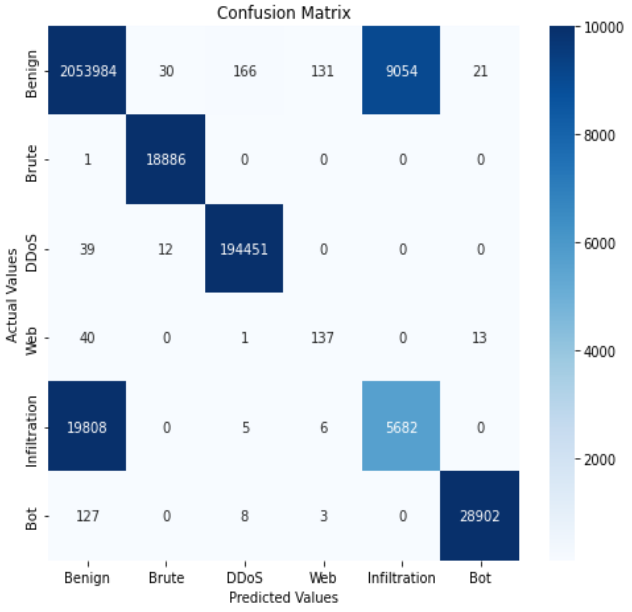



```
model.evaluate(x_test,y_test, batch_size=512)

WARNING:tensorflow:From /usr/local/lib/python3.8/dist-packages/tensorflow/python/autograph/pyct/static_analysis/liveness.py:83: Analyzer.lamba_check (from tensorflo
Instructions for updating:
Lambda fuctions will be no more assumed to be used in the statement where they are used, or at least in the same block. https://github.com/tensorflow/tensorflow/iss
4554/4554 [==============================] - 36s 6ms/step - loss: 0.0788 - acc: 0.9874 - roc-auc: 0.9992 - pr-auc: 0.9961 - f1_macro: 0.8093 - f1_weighted: 0.9857
```

Το γεγονός ότι στο test set έχουμε μεγαλύτερο accuracy από το training set, δείχνει ότι δεν έχουμε κάνει overfitting και ότι θα μπορούσαμε να συνεχίσουμε το training και παραπάνω. Σταματήσαμε στα 100 epochs για τα οποία χρειάστηκαν περίπου 6 ώρες. Το μοντέλο μας φαίνεται να βελτιώνεται σταθερά κατά την πάροδο των epochs, ενώ μια πιο μεγάλη άνοδος περίπου στην epoch 60, οφείλεται στο ότι εκείνη τη στιγμή μειώθηκε το learning rate αυτόματα με τη συνάρτηση ReduceLROnPlateau, πράγμα που σημαίνει ότι το μοντέλο μας βρήκε πιο εύκολα κάποιο τοπικό ελάχιστο της loss function. Το f1 score διαφέρει αρκετά διότι είχαμε τελείως διαφορετικές αναλογίες στο training και test set όπως εξηγήσαμε και παραπάνω. Μετά από πολλές δοκιμές αυτές οι αναλογίες απέδωσαν το καλύτερο αποτέλεσμα.

Το confusion matrix που παραθέτουμε, μπορεί να μας οδηγήσει σε ασφαλέστερα και πιο συγκεκριμένα συμπεράσματα για την ακρίβεια του μοντέλου μας σε κάθε κατηγορία ξεχωριστά. Ερμηνεύει επίσης γιατί το f1 score είναι χαμηλό. Φαίνεται ότι το μοντέλο μας πήγε πολύ καλά στις περισσότερες κατηγορίες. Αντιμετώπισε δυσκολία στις ίδιες κατηγορίες με το XGBoost, στις οποίες βέβαια τα πήγε λίγο καλύτερα. Πιο συγκεκριμένα είχε καλύτερο f1 score στα Web attacks (0.59 έναντι 0.11) και στο Infiltration (0.28 έναντι 0.23). Στα συμπεράσματα θα παρουσιάσουμε γιατί θεωρούμε πως είχαμε αυτά τα αποτελέσματα



Class	Precision	Recall	F1
Benign	*0.99*	*0.99*	*0.99*
Brute Force	*0.99*	*0.99*	*1.00*
Dos Attack	*0.99*	*0.99*	*1.00*
Web Attack	*0.49*	*0.71*	*0.59*
Infiltration	*0.38*	*0.22*	*0.28*
Botnet	*0.99*	*0.99*	*1.00*

ΣΤ. Συμπεράσματα – Future Work

Από την έρευνα που πραγματοποιήσαμε αλλά και από την υλοποίηση μας, φαίνονται οι δυνατότητες που υπάρχουν στην χρήση Anomaly Based IDS. Σε ένα σύγχρονο και ρεαλιστικό dataset και τα δυο μοντέλα τεχνητής νοημοσύνης κατάφεραν και αναγνώρισαν την πλειοψηφία των επιθέσεων χωρίς να παρουσιάσουν μεγάλες διαφορές μεταξύ τους. Γενικά θεωρούμε πως και τα δύο, έχουν θετικές και αρνητικές πλευρές, τις οποίες πρέπει να λάβουν υπόψη, όσοι επιδιώξουν να δημιουργήσουν αντίστοιχα συστήματα σε αληθινό περιβάλλον. Από την μια οι παραδοσιακοί αλγόριθμοι μηχανικής μάθησης είναι αρκετά ερμηνεύσιμοι. Αν κατανοηθεί το πρόβλημα που καλούνται να λύσουν και εκπαιδευτούν σωστά, είναι αρκετά εύκολο να ερμηνευτεί γιατί πήραν την μια ή την άλλη απόφαση και πως κατέληξαν στο τελικό αποτέλεσμα. Αυτή η ερμηνευσιμότητα είναι πολύ σημαντική για τον ρόλο ενός security engineer, ο οποίος πρέπει να γνωρίζει γιατί παράχθηκε ένα false positive στο δίκτυο, ή γιατί έγινε μια επίθεση που πέρασε απαρατήρητη. Ταυτόχρονα με βάση τα συμπεράσματα του, να βελτιώνει το υπάρχον σύστημα. Η black box αρχιτεκτονική που έχουν απ' την άλλη τα νευρωνικά δίκτυα, καθιστά πολύ πιο δύσκολο το παραπάνω έργο. Αυτό πρέπει να ληφθεί υπόψη και να «ζυγιστεί» μαζί με τις τεράστιες δυνατότητες που έχουν τα νευρωνικά δίκτυα. Οι διαφορετικοί τύποι που υπάρχουν, οι συνδυασμοί που μπορούν να προκύψουν από αυτούς δημιουργούν πολλές δυνατότητες να λυθούν πολύπλοκα προβλήματα που δεν είναι εφικτό να λυθούν από τους παραδοσιακούς αλγόριθμους. Η ικανότητα επίσης να διαχειρίζονται τεράστιους όγκους δεδομένων αλλά και η καλύτερη αξιοποίηση από τα νευρωνικά δίκτυα του σύγχρονου hardware (GPUs) πρέπει

να ληφθεί υπόψη. Ιδιαίτερα στο κομμάτι των IDS, ο όγκος των δεδομένων που πρέπει να διαχειριστούν είναι τεράστιος.

Σχετικά με τη δική μας υλοποίηση, παρατηρήσαμε παρόμοια αποτελέσματα και στα δυο μοντέλα. Πετύχαμε πολύ καλά αποτελέσματα στις κατηγορίες Benign, Brute Force, Dos Attack και Botnet. Και τα δυο μοντέλα προέβλεψαν σωστά σε μεγάλο βαθμό την κατηγορία Web Attacks, είχαν όμως πολλά False Positives. Σε αυτό ρόλο έπαιξε ο πολύ μικρός αριθμός δειγμάτων που ήταν διαθέσιμος στο αρχικό dataset (περίπου 800). Η δημιουργία πολλών συνθετικών δειγμάτων φαίνεται ότι επηρέασε την κατηγοριοποίηση εις βάρος του Benign σε διαφορετικό βαθμό βέβαια στις δυο περιπτώσεις.

Σχετικά με το Infiltration αυτό που φαίνεται αρχικά είναι ότι δεν έχουμε αρκετή πληροφορία διαθέσιμη στο dataset ώστε να το διαχωρίσουμε από το Benign. Γι' αυτό υπάρχουν χαμηλά score στα δυο μοντέλα και σε οποιαδήποτε άλλα δοκιμάσαμε. Θεωρούμε πως τα πεδία που λείπουν και αναφέραμε στο preprocessing, καθώς και όσα πεδία είχαν μηδενικές τιμές, πιθανό να μας έδιναν την παραπάνω πληροφορία που χρειαζόμασταν. Είναι κάτι το οποίο πρέπει να το εξετάσουμε σε μελλοντικό χρόνο, πραγματοποιώντας εμείς οι ίδιοι τις επιθέσεις. Αντίστοιχα πρέπει να δοκιμάσουμε μια διαφορετική δομή στο One Hot Encoding που κάναμε στο Destination Port, ίσως και σε συνδυασμό με τις υπόλοιπες πληροφορίες που έλειπαν (Src IP, Dst IP, Src Port). Σχετικά με αντίστοιχες μελέτες στο ίδιο dataset που βρήκαμε στο διαδίκτυο, παρατηρήσαμε ότι όσοι πέτυχαν πολύ υψηλά score στο infiltration, είχαν κάνει undersample κυρίως τα Benign δείγματα (ορισμένοι και τα Infiltration) και στο train set και στο test. Αυτό σημαίνει ότι κατά τη διαδικασία αυτή, πολλά από τα δείγματα που ήταν δύσκολο να κατηγοριοποιηθούν πιθανώς διαγράφτηκαν και έτσι εμφανίστηκαν καλύτερα αποτελέσματα που δεν αντικατοπτρίζουν κατά την γνώμη μας την πραγματικότητα.

Επιπρόσθετα, όσον αφορά τις δυο παραπάνω κατηγορίες (Web Attacks, Infiltration), παρατηρήσαμε ότι πρόκειται για σενάρια επιθέσεων που αποτελούνται από πολλά βήματα. Τις Web attacks τις ομαδοποιήσαμε εμείς με βάση και την ομαδοποίηση που είχαν κάνει οι κατασκευαστές του dataset και αναφέρεται στην ιστοσελίδα. Θεωρήσαμε ότι ήταν ένας συμβιβασμός που μπορούσαμε να κάνουμε και λόγω του πολύ μικρού αριθμού δειγμάτων. Και το Infiltration όμως είναι μια επίθεση που αποτελείται από δυο στάδια διαφορετικά μεταξύ τους. Δεδομένου ότι το CicFlowMeter είναι ένα εργαλείο που φτιάχνει Bidirectional flows τα οποία είναι αρκετά διαφορετικά στα δυο αυτά στάδια, θεωρούμε ότι σε μελλοντική έρευνα θα πρέπει να διαχωριστεί και να υπάρχει ξεχωριστά το reverse shell από το scan του δικτύου.

Σχετικά με το real time περιβάλλον που στήσαμε, μια πρόκληση που είχαμε να αντιμετωπίσουμε είναι το γεγονός πως το dataset δημιουργήθηκε σε ένα περιβάλλον με 470 προσωπικούς υπολογιστές και 30 servers, το οποίο δεν έχει σχέση με το δικό μας. Αυτό σημαίνει ότι και ο φόρτος του δικτύου αλλά και το εύρος των επιθέσεων ήταν τελείως διαφορετικά στα δύο δίκτυα. Παρατηρήσαμε επίσης ότι τα στατιστικά των επιθέσεων στο dataset με τις αντίστοιχες που δοκιμάσαμε ήταν πολύ διαφορετικά πράγμα που οδήγησε τα μοντέλα μας να μην μπορούν να προβλέψουν καλά επιθέσεις στις οποίες είχαμε πολύ καλό score στο dataset όπως το Dos.

Τέλος, εντοπίζουμε σαν πρόβλημα το ότι υπάρχουν πολύ λίγα διαθέσιμα δημόσια δεδομένα για να εκπαιδευτούν τέτοια συστήματα. Το dataset που χρησιμοποιήσαμε έχει δημιουργηθεί 5 χρόνια πριν διάστημα το οποίο έχουν αλλάξει αρκετά οι τάσεις και οι τεχνικές που χρησιμοποιούν οι επιτιθέμενοι. Θεωρούμε ότι η έλλειψη κατάλληλων και σωστά διαμορφωμένων δεδομένων είναι ο κύριος λόγος που εμποδίζει την ανάπτυξη Anomaly Based IDS.

Πρόκληση για εμάς το επόμενο διάστημα είναι η εξεύρεση τέτοιων δεδομένων, ή και η δημιουργία τους σε πειραματικό περιβάλλον. Είναι επίσης η βελτιστοποίηση των μοντέλων μηχανικής μάθησης που δημιουργήσαμε, στηριζόμενοι στην γνώση που αποκτήσαμε όλο αυτό το διάστημα, αλλά και η δημιουργία νέων όπως Signature Based, Host Based, έτσι ώστε να επιτυγχάνεται σε μεγαλύτερο βαθμό στο δίκτυο η άμυνα σε βάθος.

[1] https://www.statista.com/topics/9583/machine-learning/#topicOverview
[2] https://www.precedenceresearch.com/machine-learning-as-a-service-market
[3] L. Ashiku C. Dagli, Network Intrusion Detection System using Deep Learning, Complex Adaptive Systems Conference Theme: Big Data, IoT, and AI for a Smarter Future, Malvern, Pennsylvania, June 16-18, 2021
[4] J.O.Mebawondu, O.D.Alowolodu, J.O.Mebawondu, A.O.Adetunmbi, Network intrusion detection system using supervised learning paradigm, Scientific African, 2020
[5] P. Jairu, A.B.Mailewa ,Network Anomaly Uncovering on CICIDS-2017 Dataset: A Supervised Artificial Intelligence Approach, Minnesota State University Mankato, MN, USA, 2022
[6] G.Creech ,Developing a high-accuracy cross platform Host-Based Intrusion Detection System capable of reliably detecting zero-day attacks, 2013
[7] R. Moskovitch, S.Pluderman, I.Gus, D.Stopel, C.Feher, Y. Parmet, Y.Shahar, Y.Elovici, Host Based Intrusion Detection using Machine Learning, Deutsche Telekom Laboratories at Ben-Gurion University, 2007
[8] K. N. Rao, K.V.Rao, P. Reddy, A hybrid Intrusion Detection System based on Sparse autoencoder and Deep Neural Network, Computer Communications, 2021
[9] A. Kim, M. Park, A.D.H. Lee, AI-IDS: Application of Deep Learning to Real-Time Web Intrusion Detection, IEEE Access, 2020
[10] S.M.Sohi, J.Seifert, F.Ganji, RNNIDS: Enhancing network intrusion detection systems through deep learning, 2020
[11] https://www.unb.ca/cic/datasets/ids-2018.html
[12] https://www.kdd.org/kdd-cup/view/kdd-cup-1999/Data
[13] https://www.unb.ca/cic/datasets/nsl.html
[14] https://research.unsw.edu.au/projects/adfa-ids-datasets
[15] https://imbalanced-learn.org/stable/#