DIPLOMA THESIS

TECHNICAL UNIVERSITY OF CRETE
SCHOOL OF ELECTRICAL & COMPUTER ENGINEERING

Security Information and Event Management as a Service (SIEM) and Accompanying Android App

Alexandros Antonopoulos

Committee:
Associate Prof. Ioannidis Sotirios
Prof. Dollas Apostolos
Associate Prof. Koutroulis Eftychios

May 2022

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ & ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Πληροφορίες Ασφαλείας και Διαχείριση Συμβάντων ως Υπηρεσία μέσω Android App

Αλέξανδρος Αντωνόπουλος

Εξεταστική επιτροπή:
Αναπ. Καθ. Ιωαννίδης Σωτήριος
Καθ. Δόλλας Απόστολος
Αναπ. Καθ. Κουτρούλης Ευτύχιος

Μάιος 2022

Περίληψη

Το θέμα της διπλωματικής αυτής είναι η δημιουργία μιας εφαρμογής Android για την παρακολούθηση συμβάντων ασφάλειας που ανιχνεύονται μέσω συστήματος ανίχνευσης εισβολών (Intrusion Detection System). Το σύστημα στο οποίο βασίστηκε η εφαρμογή είναι το open source IDS Snort. Θα διαμορφωθεί με βάση τις ανάγκες και τις λειτουργίες που απαιτούνται να χρησιμοποιηθούν.

Ακαδημαϊκά, και στα πλαίσια της επαλήθευσης της προτεινόμενης λύσης, η εγκατάσταση θα πραγματοποιηθεί σε τοπικό δίκτυο με την βοήθεια Virtual Box [1]. Θα δημιουργηθεί ταυτόχρονα Android [2] εφαρμογή, η οποία, όταν λαμβάνεται στο σύστημα ένα Alert, θα ενημερώνει με ειδοποίηση τον χρήστη στο κινητό του τηλέφωνο. Θα δημιουργηθούν ειδικοί κανόνες ανίχνευσης και συγκεκριμένα κριτήρια, για το ποια γεγονότα θα χαρακτηριστούν άξια ειδοποίησης, ανάλογα με τον εργασιακό χώρο που θα τοποθετηθεί (π.χ. Ιατρικό Κέντρο, Λογιστικό Γραφείο, κλπ.)

Abstract

The subject of this diploma is the creation of an Android application for monitoring security incidents detected through intrusion detection system (IDS). The system on which the application was based is the open source IDS Snort. It will be configured based on the needs and required functionality of the proposed solution.

Academically, and in the context of the implementation as a proof of concept the deployment will take place in local network with the help of Virtual Box[1].In parallel, an Android[2] application will be also developed which, when an Alert is received on the system, will notify with a notification the user on his mobile phone. Special detection rules will be created and specific criteria, for which events will be considered noteworthy, depending on the workplace to be placed (eg Medical Center, Accounting Office, etc) will be also applied.

Περιεχόμενα

Κατάλογος Σχημάτων

Γλωσσάρι

UDP	Το πρωτόκολλο User Datagram Protocol (UDP) είναι ένα από τα βασικά πρωτόκολλα που χρησιμοποιούνται στο Διαδίκτυο. Διάφορα προγράμματα χρησιμοποιούν το πρωτόκολλο UDP για την αποστολή σύντομων μηνυμάτων (γνωστών και ως segments) από τον έναν υπολογιστή στον άλλον μέσα σε ένα δίκτυο υπολογιστών. Ένα από τα κύρια χαρακτηριστικά του UDP είναι ότι δεν εγγυάται αξιόπιστη επικοινωνία.
DNS	Το DNS είναι υπεύθυνο να μετατρέπει τα μνημονικά ονόματα (domain names) στις σχετικές IP. Για παράδειγμα, η IP.GR έχει καταχωρήσει το domain ip.gr για να ονομάσει μοναδικά την ύπαρξή της στο Internet.
TCP	Το Πρωτόκολλο Ελέγχου Μετάδοσης/Πρωτόκολλο Διαδικτύου, είναι μια συλλογή πρωτοκόλλων επικοινωνίας στα οποία βασίζεται το Διαδίκτυο αλλά και μεγάλο ποσοστό των εμπορικών δικτύων. Ένα από τα κύρια χαρακτηριστικά του TCP είναι ότι εγγυάται αξιόπιστη επικοινωνία, καθώς κάθε πακέτο ενημερώνει τον αποστολέα ότι λήφθηκε από τον παραλήπτη.
ICMP	Το πρωτόκολλο Internet Control Message Protocol (ICMP) είναι ένα από τα βασικά πρωτόκολλα του διαδικτύου. Χρησιμοποιείται κυρίως από τα λειτουργικά συστήματα των ηλεκτρονικών υπολογιστών ενός δικτύου για την ανταλλαγή μηνυμάτων λάθους, όπως για παράδειγμα την έλλειψη κάποιας υπηρεσίας από έναν server ή την απουσία ενός υπολογιστή από το δίκτυο.
IP	Μία διεύθυνση IP - διεύθυνση διαδικτυακού πρωτοκόλλου είναι ένας μοναδικός αριθμός που χρησιμοποιείται από συσκευές σε ένα δίκτυο υπολογιστών που χρησιμοποιεί το Internet Protocol standard για τη μεταξύ τους αναγνώριση και συνεννόηση.
NMAP	Το Nmap (Network Mapper) είναι ένα ελεύθερο και ανοικτού κώδικα εργαλείο για την εξερεύνηση του δικτύου και τον έλεγχο της ασφάλειας.
PORT	Πύλη δικτύου ονομάζεται το υλικό (υλικό) ή το λογισμικό που χρησιμοποιείται για τη σύνδεση ανάμεσα σε διαφορετικά δικτυακά περιβάλλοντα.
PING	Το ping είναι μέθοδος για τον εντοπισμό της διαθεσιμότητας και της απόδοσης ενός απομακρυσμένου πόρου του δικτύου.
SIEM	Security Information and Event Management, Πληροφορίες Ασφαλείας και Διαχείριση Συμβάντων ως υπηρεσία
SNMP	Το SNMP είναι ένα πρωτόκολλο του επιπέδου εφαρμογών του οποίο διευκολύνει την ανταλλαγή πληροφοριών διαχείρισης μεταξύ των συσκευών του δικτύου.

SMTP	Οι χαρακτήρες SMTP είναι ακρωνύμιο του Simple Mail Transfer Protocol, ένα πρωτόκολλο οποίο χρησιμοποιείται για την αποστολή μηνυμάτων μεταξύ των servers. Τα περισσότερα από τα συστήματα αποστολής email μέσω του Internet χρησιμοποιούν το SMTP για τη δρομολόγηση των μηνυμάτων από έναν server σε έναν άλλο.
DIALOG (Android)	Είναι μια λειτουργία στο λογισμικό Android που ανοίγει ένα παράθυρο σαν αναδυόμενο παράθυρο πάνω στο παρών Activity-οθόνη που προβάλλεται εκείνη την στιγμή.
FTP	(File Transfer Protocol – Πρωτόκολλο Μεταφοράς Αρχείων) είναι το πρωτόκολλο που χρησιμοποιείται για την απομακρυσμένη μεταφορά αρχείων από έναν υπολογιστή σε έναν server και αντίστροφα μέσω ενός FTP client.
IDS	Τα συστήματα ανίχνευσης εισβολών (IDS) είναι υπεύθυνα για την συλλογή δεδομένων από μεμονωμένους υπολογιστές ή από κάποιο δίκτυο υπολογιστών, καθώς και για την ανάλυση των πληροφοριών αυτών.

Κεφάλαιο 1

Εισαγωγή

1.1 Κίνητρα

Ταυτόχρονα με την ανάπτυξη της Πληροφορικής παρουσιάστηκε η αναγκαιότητα να δημιουργηθούν νέα γνωστικά πεδία της επιστήμης, για να εξασφαλιστεί η ασφάλεια των πληροφοριακών συστημάτων, των δικτύων και των δεδομένων των υπολογιστών. Ήταν απολύτως απαραίτητο να καταστεί δυνατή η αποτροπή ή η μη εξουσιοδοτημένη πρόσβαση στα δεδομένα και στη χρήση τους. Κατ' αρχάς όφειλε να συμφωνηθεί ένα θεωρητικό πλαίσιο, εντός του οποίου θα χωρούν οι πολιτικές που σχεδιάζουν την ασφάλεια των συστημάτων. Δεδομένου ότι διακυβεύονται ζητήματα ηθικής, προσωπικών δεδομένων, οικονομικών συμφερόντων, πνευματικής ιδιοκτησίας, κρατικής και διακρατικής ασφάλειας, κ.α., η πολιτική ασφαλείας θα πρέπει να είναι διεθνώς αποδεκτή.

Επιπλέον, οι διαδικασίες του σχεδιασμού και της υλοποίησης των πολιτικών ασφαλείας στα πληροφοριακά συστήματα δεν θα πρέπει να παρεμβαίνουν καθ' οιπονδήποτε τρόπο στην απρόσκοπτη λειτουργία τους. Κομβικό σημείο στη διαδικασία σχεδιασμού είναι η επιλογή των πληροφοριών που πρόκειται να χαρακτηριστούν ως εμπιστευτικές και να προστατευτούν. Ο χρήστης θα πρέπει να γνωρίζει ότι αποτελεσματικές τεχνικές σχεδίασης χρησιμοποιούνται για να προστατεύουν τον ίδιο και την εργασία, ή την παρουσία του στον ψηφιακό κόσμο.

≪Εκτός από τις αρχές της Ακεραιότητας Πληροφοριών, την Εμπιστευτικότητα και τη Διαθεσιμότητα Πληροφοριών οι πολιτικές ασφαλείας θα πρέπει να εμπεριέχουν και τους όρους αυθεντικότητα, εγκυρότητα, μοναδικότητα και μη αποποίηση≫.[4]

Ένα σύστημα ασφαλείας πρέπει να αξιολογεί τους κινδύνους και να δημιουργεί το κατάλληλο πλαίσιο, ώστε τα μέτρα ασφαλείας να αξιολογούν και να περιορίζουν την επικινδυνότητα και να οργανώνεται έτσι που να δικαιολογούνται και να εξασφαλίζονται οι απαιτούμενοι οικονομικοί πόροι για τη δημιουργία τους. Η πολιτική ασφαλείας και το σύνολο των μέτρων προστασίας αποτελούν το security plan, το σχέδιο ασφαλείας, για τα πληροφοριακά συστήματα ενός οργανισμού.

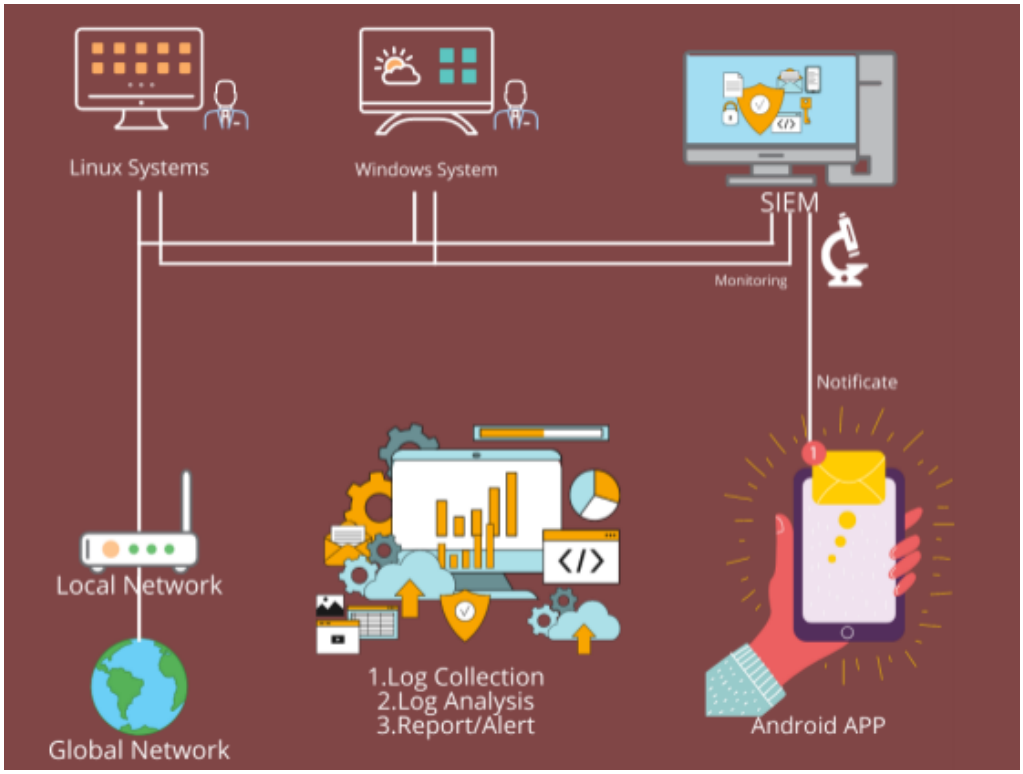
≪Είναι σημαντικό να δημιουργείται η κουλτούρα της αναγκαιότητας ασφάλειας των πληροφοριακών συστημάτων του οργανισμού και μεταξύ των υπαλλήλων και μεταξύ του οργανισμού και των πελατών του, ως νομική και ηθική υποχρέωση και ως παράγοντας εμπιστοσύνης.

Τα είδη των πολιτικών ασφαλείας είναι:

• τα τεχνικά (computer oriented) συστήματα πληροφοριών, λειτουργικά συστήματα και δίκτυα υπολογιστών

• τα οργανωτικά (human oriented) και

• τα ατομικά (individual security policies)



Σχήμα 1.1: Αρχιτεκτονική Συστήματος

Η πολιτική ασφαλείας πρέπει να ικανοποιεί τις απαιτήσεις για την ασφάλεια του πληροφοριακού συστήματος που προέρχονται από όλους τους εμπλεκόμενους στη χρήση και στη λειτουργία του. Δηλαδή τους χρήστες και τους διαχειριστές του πληροφοριακού συστήματος, τη διοίκηση, τους πελάτες του οργανισμού, και να συμφωνεί με τις νομικές και κανονιστικές διατάξεις που διέπουν την λειτουργία του.

Ο καθορισμός της πολιτικής ασφάλειας του πληροφοριακού συστήματος θα πρέπει να καλύπτει τις ακόλουθες κατηγορίες:

- Ζητήματα προσωπικού
- Φυσική ασφάλεια
- Έλεγχο πρόσβασης στο πληροφοριακό σύστημα
- Διαχείριση υλικών και λογισμικών
- Νομικές υποχρεώσεις
- Διαχείριση της πολιτικής ασφάλειας
- Οργανωτική δομή
- Σχέδιο συνέχισης λειτουργίας≫[5]

≪Όταν εφαρμόζεται μια πολιτική ασφαλείας επιδιώκεται:

Α) η πληρότητα (οι οδηγίες και τα μέτρα προστασίας να καλύπτουν το σύνολο των αγαθών και όλες τις λειτουργίες)

Β) η επικαιρότητα (να λαμβάνονται υπόψη οι τρέχουσες τεχνολογικές εξελίξεις)

Γ) η γενικευσιμότητα (να μπορεί η πολιτική ασφαλείας να καλύπτει μικρές αλλαγές ή επεκτάσεις στο πληροφοριακό σύστημα με κάποιες τροποποιήσεις ή προσθήκες).

Δ) η σαφήνεια, την εύκολη κατανόηση, την τεχνολογική ανεξαρτησία και την καταλληλότητα ανάλογα με τον οργανισμό που απευθύνεται.≫[6]

Επομένως, τέτοιες τεχνολογίες μπορούν να αποδειχθούν πολύτιμος σύμμαχος για την ασφάλεια στον κυβερνοχώρο και να βοηθήσουν τα συστήματα παρακολούθησης δικτύων ξεπερνώντας τις τρέχουσες προκλήσεις.Μια τέτοια τεχνολογία συστημάτων ασφαλείας είναι τα Security Incident Event Management (SIEM). Συστήματα τα οποία εποπτεύουν τον χώρο ενός τοπικού δικτύου (στην περίπτωση που περιγράφεται στην παρούσα εργασία) και ενημερώνουν τους διευθύνοντες ή τους υπεύθυνους ασφαλείας για περιπτώσεις παραβιάσεων. Η εξέλιξη τεχνολογιών εντοπισμού απειλών και η βελτίωση της άμεσης και πληρέστερης ενημέρωσης των παραπάνω, είναι μια πρόκληση που καλείται ο δημιουργός των συστημάτων ασφαλείας να αντιμετωπίσει.

Το κόστος ενός τέτοιου συστήματος προφανώς δεν είναι προσιτό απο μια μικρομεσαία επιχείρηση στις μέρες μας, αφήνοντας την έτσι απροστάτευτη απο τους παραπάνω κινδύνους. Δημιουργώντας λοιπόν ενα τέτοιο σύστημα στο "σύννεφο-Cloud", επιτρέπεται σε τέτοιες επιχειρήσεις, χωρίς έξτρα έξοδα όπως αγορά σέρβερ,προσωπικό κλπ. να παρακολουθούνται και να προστατεύονται. Κάθε μια για τις ανάγκες της θα επιτηρείται και θα ενημερώνει τους αρμοδίους αναλόγως.

1.2 Στόχοι

Γενικός στόχος της παρούσας εργασίας είναι να κατασκευαστεί ένα σύστημα, το οποίο θα είναι αποτελεσματικό στο να εντοπίζει τις απειλές και να ενημερώνει άμεσα τους αρμοδίους.

Συγκεκριμένα:

• Θα διαμορφώνεται, με βάση τις ανάγκες, το πρόγραμμα το οποίο θα χρησιμοποιείται για τον εντοπισμό των σφαλμάτων.

• Θα διακρίνονται τα σφάλματα ανάλογα με την επικινδυνότητά τους.

• Θα εξασφαλίζεται μια αξιόπιστη, κρυπτογραφημένη και ασφαλής επικοινωνία μεταξύ των συστημάτων.

• Θα προβάλλεται στον χρήστη λεπτομερώς κάθε σφάλμα και θα ειδοποιείται, κάθε φορά που θα κρίνεται ότι αυτό είναι αναγκαίο.

• Θα υλοποιηθεί η αρχιτεκτονική του συστήματος έτσι ώστε να είναι, όσο γίνεται πιο κατανοητή, και με στόχο την, κατά το δυνατόν, λιγότερη χρήση πόρων.

1.3 Δομή

Παρακάτω αναφέρονται συνοπτικά τα περιεχόμενα κάθε κεφαλαίου:

• Στο Κεφάλαιο 2 αναλύεται η έννοια της Ασφάλειας Συστημάτων και σχετικές τεχνολογίες.

• Στο Κεφάλαιο 3 περιγράφεται η πολυνημάτωση και τα πλεονεκτήματα της στον χώρο της πληροφορικής.

• Στο Κεφάλαιο 4 εξηγείται η διαδικασία της κρυπτογράφησης-αποκρυπτογράφησης και ο τρόπος που χρησιμοποιήθηκε στο σύστημα.

• Στα Κεφάλαια 5 και 6 αναφέρονται αναλυτικά ολες οι λειτουργίες και ο τρόπος κατασκευής των Server και Client αντίστοιχα.

• Στο Κεφάλαιο 7 περιγράφεται η λειτουργία του συστήματος συνολικά και απο τις δύο μεριες, Client και Server.

• Στο Κεφάλαιο 8 αποτυπώνονται τα συμπεράσματα της εργασίας και οι πιθανές μελλοντικές επεκτάσεις της.

Κεφάλαιο 2

Εισαγωγικά για την Ασφάλεια Συστημάτων

2.1 Συστήματα SIEM & IDS

Τα οφέλη των προϊόντων SIEM επιτρέπουν σε έναν οργανισμό να αποκτά μία πολύ καλύτερη εικόνα των events ασφαλείας. Συγκεντρώνοντας δεδομένα των logs ασφαλείας από τα συστήματα ελέγχου ασφαλείας της επιχείρησης, λειτουργικά συστήματα, εφαρμογές και άλλα προγράμματα, το SIEM μπορεί να αναλύει μεγάλο όγκο δεδομένων ασφαλείας ώστε να αναγνωρίζει τις επιθέσεις και τις ευπάθειες που αυτές κρύβουν. Το SIEM συχνά μπορεί να αναγνωρίζει κακόβουλη δραστηριότητα που καμία μεμονωμένη πηγή δεν θα μπορούσε. Αυτό γίνεται γιατί το SIEM είναι το μοναδικό σύστημα ελέγχου ασφαλείας με πεδίο ορατότητας πραγματικά σε όλη την επιχείρηση.[7]

Το IDS αποτελεί σύστημα παρακολούθησης και ανάλυσης των συμβάντων, τα οποία λαμβάνουν χώρα τόσο στους ίδιους τους ηλεκτρονικούς υπολογιστές όσο και στα δίκτυα υπολογιστών. Στόχος είναι ο εντοπισμός ενδείξεων για πιθανές προσπάθειες εισβολής, κατά τις οποίες συχνά εντοπίζονται ίχνη παραβίασης της ακεραιότητας, της εμπιστευτικότητας και της διαθεσιμότητας των πληροφοριακών πόρων. Οι προσπάθειες παράκαμψης των μηχανισμών ασφαλείας μπορεί να προέρχονται από εξωτερικούς χρήστες, προς το εσωτερικό εταιρικό δίκτυο, στους οποίους δεν επιτρέπεται η πρόσβαση στο υπάρχον πληροφοριακό σύστημα. Επίσης, οι προσπάθειες παράκαμψης πιθανόν να προέρχονται από εσωτερικούς χρήστες, με περιορισμένα δικαιώματα πρόσβασης.[8]

Ένα από αυτά τα συστήματα είναι το Snort το οποίο αναλύεται παρακάτω.

2.2 Τι είναι το Snort?

Το Snort[9] είναι ένα open source Network Intrusion Prevention and Detection System. Χρησιμοποιεί μία λογική κανόνων συνδυάζοντας υπογραφές, πρωτόκολλο και μεθόδους εντοπισμού ανώμαλης συμπεριφοράς.

2.2.1 Χρήσεις του SNORT

Το πρόγραμμα SNORT έχει σχεδιαστεί ώστε να είναι σε θέση να κάνει, ανάγνωση πακέτων και εμφάνιση τους στην κονσόλα (Sniffer), αποθήκευση των πακέτων που διαβάζει σε αρχεία, για μελλοντική εξέταση (Packet Logger), παρακολούθηση του δικτύου και ανάλυση της κίνησης, με βάση κανόνες που ορίζει ο χρήστης και εκτέλεση

Σχήμα 2.1: Πρόγραμμα Snort

συγκεκριμένων ενεργειών όταν εντοπιστεί συγκεκριμένη δράση, με επιπλέον δυνατότητα απόρριψης πακέτων[10].

2.2.2 Βασικά χαρακτηριστικά του SNORT

Πρόκειται για ένα πρόγραμμα που διαθέτει, ελαφρύ Network Intrusion Detection System (NIDS), είναι διαθέσιμο σε Linux, Windows, Mac OS X, κ.ά., και είναι σύστημα υψηλών δυνατοτήτων.

Επιπλέον είναι γρήγορο (είναι πιθανόν να μπορεί να εντοπίσει επιθέσεις σε δίκτυα μεγαλύτερα των 1000Mbps), είναι εύκολα παραμετροποιήσιμο, διαθέτει απλή γλώσσα κανόνων και μεγάλη ποικιλία επιλογών για καταγραφή και αναφορά.

2.2.3 Αρχιτεκτονική του SNORT

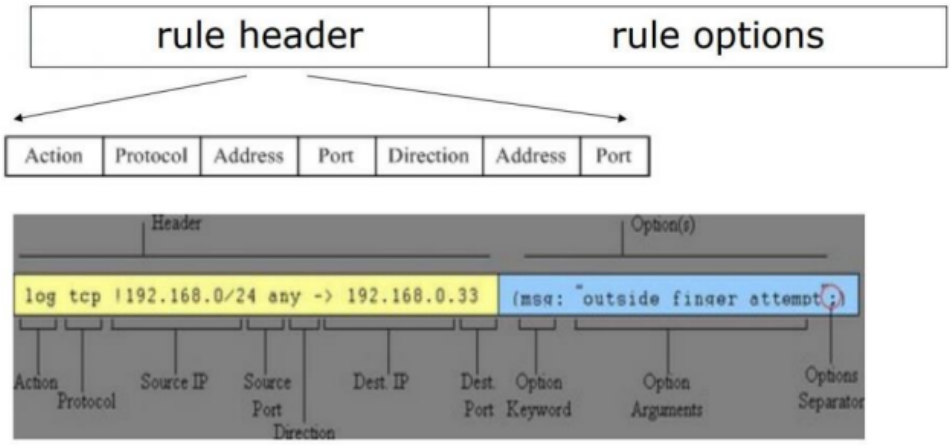
Το πρόγραμμα έχει τη δυνατότητα να λαμβάνει πακέτα από τις δικτυακές διεπαφές (πχ. Ethernet) και να τα προετοιμάζει για επεξεργασία ώστε να τα περάσει από τη μηχανή εντοπισμού, (Preprocessor) Η μηχανή αυτή ανασυγκροτεί τα πακέτα, εντοπίζει ανωμαλίες στις επικεφαλίδες τους, αποκωδικοποιεί το HTTP URL, επανασυνδέει ροές TCP Η μηχανή αναζήτησης (Detection Engine) χρειάζεται τις εξής πληροφορίες: Την IP επικεφαλίδα του πακέτου, την επικεφαλίδα του transport layer. (TCP, UDP, ICMP κ.α.), την επικεφαλίδα του application layer. (DNS, FTP, SNMP, SMTP κ.α.) και το περιεχόμενο (payload) του πακέτου.

Το Snort λειτουργεί με εφαρμογή των κανόνων στα πακέτα εφαρμόζοντας τον αλγόριθμο αντιστοίχισης συμβολοσειρών (string matching) Boyer-Moore.

2.3 Κανόνες SNORT

Οι κανόνες του Snort βρίσκονται σε μία γραμμή και δημιουργούνται από γνωστές υπογραφές επιθέσεων. Συνήθως βρίσκονται στο snort.conf.

Η στήλη του Protocol αφορα τον τύπο του πρωτοκόλλου στον οποίο ανήκει το πακέτο προς εντοπισμό (icmp, ip, udp, tcp). Η στήλη Source IP αφορά την IP του αποστολέα του πακέτου μαζί με την μάσκα δικτύου. (Η τιμή «any», αντιπροσωπεύει οποιαδήποτε διεύθυνση). Η στήλη Source Port αφορά την πύλη από την οποία έγινε η αποστολή του



Σχήμα 2.2: Κανόνες Snort

πακέτου (έχει νόημα για udp και tcp πακέτα).(Η τιμή «any» αντιπροσωπεύει επίσης οποιοδήποτε port).

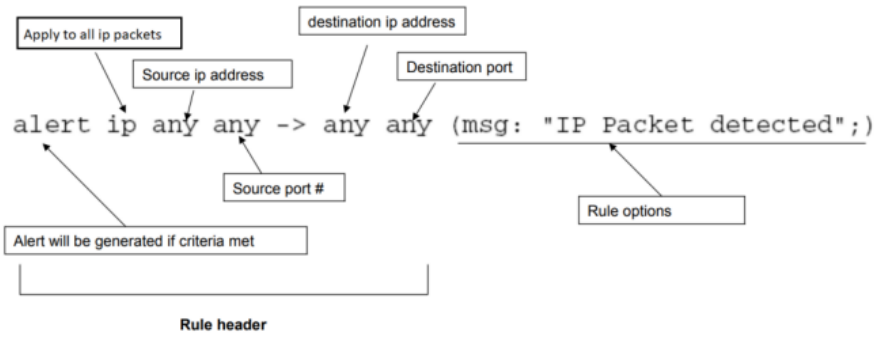
Ακολουθούν οι στηλες Destination IP οπου περιέχει την διεύθυνση του παραλήπτη του πακέτου μαζί με την μάσκα δικτύου. Destination Port την πύλη προορισμού του πακέτου (έχει νόημα για udp και tcp πακέτα). Rule Options οπου εχει τα χαρακτηριστικά ελέγχου πακέτου (υπογραφή επίθεσης, επίπεδο προτεραιότητας και πληροφορίες επίθεσης).

Απαρτίζεται από τα τμήματα:

• Keyword: Δηλώνει το είδος του option (πχ msg, logto, minifrag, ttl, id, dsize, content, flags, seq, ack).
• Argument: Περιέχει παραμέτρους με βάση τις οποίες θα ελεγχθεί το πακέτο.

Σειρά με την οποία το detection engine σκανάρει τους κανόνες: Το Snort δεν ελέγχει τους κανόνες με τη σειρά που βρίσκονται στο αρχείο κανόνων. Από προεπιλογή, η σειρά είναι, πρώτα οι κανόνες ειδοποίησης (alert), έγκρισης (pass) και μετά καταγραφής (log).

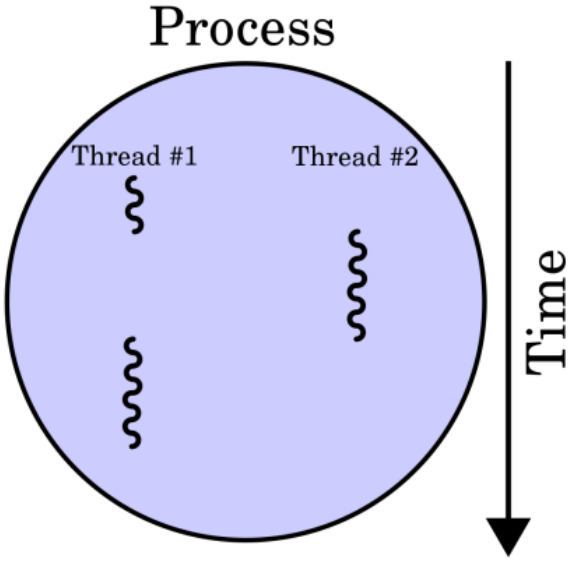
Παρακάτω δίνεται ένα παράδειγμα δείχνοντας τον τρόπο με τον οποίο είναι δομημένος ένας κανόνας.[10] [11] [12]



Σχήμα 2.3: Κανόνες Snort με any επιλογές

Κεφάλαιο 3

Threading



Σχήμα 3.1: Τα νήματα σε ένα Process

≪Ένα νήμα εκτέλεσης (thread) είναι η μικρότερη ακολουθία προγραμματισμένων εντολών που μπορεί να υποστεί διαχείριση ανεξάρτητα από το λειτουργικό σύστημα. Ένα νήμα είναι μια ελαφριά διεργασία. Η υλοποίηση των νημάτων και των διεργασιών διαφέρει από το ένα λειτουργικό σύστημα στο άλλο. Στις περισσότερες όμως περιπτώσεις ένα νήμα εμπεριέχεται σε μια διεργασία. Μπορεί να υπάρχουν πολλαπλά νήματα μέσα στην ίδια διεργασία, τα οποία μπορούν να μοιράζονται πόρους από το σύστημα, όπως μνήμη. Διαφορετικές διεργασίες δεν μπορούν να μοιράζονται τους ίδιους πόρους. Συγκεκριμένα, τα νήματα μιας διεργασίας περιέχουν τις εντολές προς την εκτελούμενη διεργασία (δηλαδή τον κώδικα της) και το εννοιολογικό της πλαίσιο (οι τιμές των μεταβλητών της σε οποιαδήποτε χρονική στιγμή.

Σε έναν απλό επεξεργαστή, η πολυνημάτωση (multithreading) πραγματοποιείται με τη μέθοδο της ≪πολυπλεξίας με διαίρεση χρόνου≫ (όπως στην πολυεπεξεργασία): ο επεξεργαστής μεταπηδάει μεταξύ των διάφορων νημάτων. Αυτή η εναλλαγή μεταξύ των διεργασιών ονομάζεται Μεταγωγή Περιβάλλοντος (context switch)[13], και συμβαίνει σε τόσο σύντομα χρονικά διαστήματα, ώστε ο χρήστης να έχει την εντύπωση ότι τα νήματα εκτελούνται την ίδια στιγμή. Μόνο σε έναν επεξεργαστή με πολλούς επεξεργαστικούς

πυρήνες τα νήματα εκτελούνται πραγματικά ταυτόχρονα και κάθε πυρήνας εκτελεί ένα συγκεκριμένο νήμα ή εργασία. [14] [15]

Ανάμεσα στις γλώσσες προγραμματισμού η υποστήριξη για νήματα ποικίλει: Κάποιες γλώσσες δεν υποστηρίζουν την εκτέλεση περισσοτέρων του ενός νήματος για ένα πρόγραμμα ταυτόχρονα.

Παραδείγματα τέτοιων γλωσσών είναι η Python και η OCaml, οι οποίες, παρόλο που έχουν βιβλιοθήκες για δημιουργία νημάτων, δεν εκτελούνται τα νήματα ταυτόχρονα.[16] [17] Άλλες γλώσσες περιορίζονται στη χρήση νημάτων χρήστη, τα οποία δεν είναι ορατά στο πυρήνα και έτσι δεν μπορούν να εκτελεστούν ταυτόχρονα. Μόνο τα νήματα πυρήνα (που είναι ορατά στον χρονοδρομολογητή του πυρήνα), μπορούν να εκτελεστούν ταυτόχρονα.

Πολλά σύγχρονα λειτουργικά συστήματα υποστηρίζουν τόσο νήματα καταμερισμού χρόνου, όσο και νήματα παράλληλης πολυεπεξεργασίας, στον χρονοδρομολογητή τους. Ο πυρήνας ενός λειτουργικού συστήματος επιτρέπει στους προγραμματιστές να χειρίζονται τα νήματα μέσω της διεπαφής κλήσεων συστήματος.[18] Ορισμένες υλοποιήσεις λέγονται ≪νήμα πυρήνα≫, ενώ ο ειδικός τύπος νήματος πυρήνα που μοιράζεται την ίδια κατάσταση και τις ίδιες πληροφορίες[19], ονομάζεται ≪ελαφρά διεργασία≫. Στα συστήματα unix υπάρχει το POSIX στάνταρντ για δημιουργία POSIX νημάτων.≫

3.1 Διαφορές μεταξύ Νήματος και Ελαφριάς Διεργασίας

≪Ένα νήμα διαφέρει από μια διεργασία ενός πολυεπεξεργαστικού λειτουργικού συστήματος στα εξής:

• Οι διεργασίες είναι τυπικώς ανεξάρτητες, ενώ τα νήματα αποτελούν υποσύνολα μιας διεργασίας.

• Οι διεργασίες περιέχουν σημαντικά περισσότερες πληροφορίες κατάστασης από τα νήματα, ενώ πολλαπλά νήματα μιας διεργασίας μοιράζονται την κατάσταση της διεργασίας όπως επίσης μνήμη και άλλους πόρους.

• Οι διεργασίες έχουν ξεχωριστούς χώρους διευθυνσιοδότησης, ενώ τα νήματα μοιράζονται το σύνολο του χώρου διευθύνσεων που τους παραχωρείται.

• Η εναλλαγή ανάμεσα στα νήματα μιας διεργασίας είναι πολύ γρηγορότερη από την εναλλαγή ανάμεσα σε διαφορετικές διεργασίες.

Ένα πρόγραμμα που μεταγλωττίζεται, δημιουργεί ένα εκτελέσιμο αρχείο. Αυτό όταν εκτελείται, δημιουργείται μια νέα διεργασία μέσα στην οποία τρέχει ο εκτελέσιμος κώδικας. Το λειτουργικό σύστημα φροντίζει να δημιουργήσει χώρο μνήμης-διευθύνσεων για την διεργασία. Στο unix, κάθε εκτέλεση του εκτελέσιμου αρχείου, δημιουργεί μια νέα ξεχωριστή διεργασία, η οποία τρέχει παράλληλα (ή ψευδοπαράλληλα με την τεχνική διαίρεσης χρόνου - όταν έχουμε ένα επεξεργαστή). [20]

Το νήμα είναι μια πολύ πιο "ελαφριά" μορφή διεργασίας, όπου η δημιουργία είναι εκατοντάδες φορές πιο γρήγορη από την δημιουργία μιας διεργασίας. Τα νήματα τρέχουν παράλληλα όπως και οι διεργασίες. Η ιδέα είναι ότι δημιουργούμε μια διεργασία, και μέσα από αυτή δημιουργούμε νήματα τα οποία έχουν πρόσβαση στο χώρο μνήμης της διεργασίας.≫ [21]

3.2 Πολυνημάτωση

≪Η πολυνημάτωση (multithreading) αποτελεί ένα ευρέως διαδεδομένο μοντέλο προγραμματισμού και εκτέλεσης διεργασιών, το οποίο επιτρέπει την ύπαρξη πολλών νημάτων μέσα στα πλαίσια μιας και μόνο διεργασίας. Τα νήματα αυτά μοιράζονται τους πόρους της διεργασίας και μπορούν να εκτελούνται ανεξάρτητα.

Η πολυνημάτωση δεν πρέπει να μπερδεύεται με την πολυδιεργασία (multitasking), η οποία αφορά την εκτέλεση προγραμμάτων, δηλαδή διεργασιών, παράλληλα, σε ένα υπολογιστικό σύστημα με πολλούς επεξεργαστές.

Το πλεονέκτημα ενός πολυνηματικού προγράμματος είναι ότι έχει τη δυνατότητα να εκτελείται γρηγορότερα, σε υπολογιστικά συστήματα που έχουν πολλούς επεξεργαστές, επεξεργαστές με πολλούς πυρήνες, ή κατά μήκος μιας συστοιχίας υπολογιστών.

Για να μπορούν να χειραγωγηθούν σωστά τα δεδομένα, τα νήματα θα πρέπει ορισμένες φορές να συγκεντρωθούν σε ένα ορισμένο χρονικό διάστημα, έτσι ώστε να επεξεργασθούν τα δεδομένα στη σωστή σειρά. Ένα ακόμα πλεονέκτημα της πολυδιεργασίας (και κατ' επέκταση της πολυνημάτωσης), ακόμα και στους απλούς επεξεργαστές, (με έναν πυρήνα επεξεργασίας), είναι η δυνατότητα μιας εφαρμογής να ανταποκρίνεται άμεσα. Έχοντας ένα πρόγραμμα που εκτελείται μόνο σε ένα νήμα, αυτό θα φαίνεται ότι κολλάει ή ότι παγώνει, στις περιπτώσεις που εκτελείται κάποια μεγάλη διεργασία που απαιτεί πολύ χρόνο. Αντίθετα σε ένα πολυνηματικό σύστημα, οι χρονοβόρες διεργασίες μπορούν να εκτελούνται παράλληλα με άλλα νήματα που φέρουν άλλες εντολές για το ίδιο πρόγραμμα, καθιστώντας το άμεσα ανταποκρίσιμο.≫[22])

Τα λειτουργικά συστήματα προγραμματίζουν τα νήματα με έναν από τους δύο παρακάτω τρόπους:

≪1. Πολυδιεργασία προτίμησης (Preemptive multitasking): γενικώς θεωρείται η καλύτερη προσέγγιση. Σύμφωνα με αυτή, το λειτουργικό σύστημα καθορίζει πότε θα γίνεται μια εναλλαγή από το ένα νήμα στο άλλο. Μειονέκτημα αυτής της μεθόδου είναι ότι το σύστημα μπορεί να πραγματοποιήσει μια εναλλαγή σε ακατάλληλη στιγμή με αρνητικές επιπτώσεις για τις εφαρμογές που εκτελούνται.≫[23]

≪2. Πολυνημάτωση συνεργασίας (Cooperative multithreading): Σύμφωνα με αυτή τη προσέγγιση, τα ίδια τα νήματα αναλαμβάνουν τον έλεγχο. Έτσι, όταν ένα νήμα τελειώσει, παραδίδει τους πόρους εκτέλεσης σε άλλο νήμα. Μειονέκτημα αυτής της προσέγγισης είναι ότι ένα νήμα μπορεί να περιμένει επ' αόριστον να ελευθερωθούν πόροι συστήματος≫.[24]

3.3 Διεργασίες, νήματα πυρήνα και νήματα χρήστη

≪Ένα νήμα χρήστη, (user thread), είναι συνδεδεμένο με το εκτελέσιμο κώδικα που τρέχει στην διεργασία και δημιουργείται από το χρήστη χρησιμοποιώντας μια βιβλιοθήκη δημιουργίας νημάτων (όπως τα POSIX νήματα που δημιουργούνται από την βιβλιοθήκη pthread). Η εναλλαγή των νημάτων μιας διεργασίας γίνεται από την ίδια την διεργασία, ενώ ο πυρήνας του λειτουργικού συστήματος αναλαμβάνει μόνο την εναλλαγή των διεργασιών. Κατά την εναλλαγή των νημάτων μέσα στην διεργασία, αναλαμβάνει η διεργασία και δεν γίνονται κλήσεις συστήματος πυρήνα.≫[25]

≪Ένα μειονέκτημα των νημάτων χρήστη είναι το εξής: Όταν ένα νήμα κολλήσει, τότε ο πυρήνας του λειτουργικού συστήματος βλέπει μόνο την διεργασία που τρέχει, κι όχι το προβληματικό νήμα. Ένα νήμα πυρήνα (kernel thread) αποτελεί την ελαφρύτερη μονάδα που μπορεί να δρομολογηθεί, να υλοποιηθεί και να τρέξει μέσα στον πυρήνα του λειτουργικού συστήματος. Η δημιουργία-εκκίνηση αλλά και ο τερματισμός ενός νήματος πυρήνα γίνεται με κατευθείαν κλήσεις συστήματος μέσα στον πυρήνα. Το μοντέλο που ακολουθείται στην υλοποίηση νημάτων διαφέρει ανάλογα με το λειτουργικό σύστημα.≫[26]

≪Παρακάτω δείχνονται τα μοντέλα

• Πολλά σε Ένα: Πολλά νήματα χρήστη αντιστοιχίζονται σε ένα νήμα πυρήνα - τα πράσινα νήματα.[27]

• Ένα προς Ένα: Ένα νήμα χρήστη αντιστοιχίζεται σε ένα νήμα πυρήνα.[28]

• Πολλά προς Πολλά: Πολλά νήματα χρήστη αντιστοιχίζονται σε πολλά νήματα πυρήνα.[29] ≫

≪Τα πράσινα νήματα είναι νήματα χρήστη, που υλοποιούνται σε επίπεδο εφαρμογής και δεν χρησιμοποιούν κλήσεις συστήματος μέσα στο πυρήνα, σε αντίθεση με τα χαμηλού επιπέδου νήματα (naive threads) τα οποία χρησιμοποιούν κλήσεις συναρτήσεων πυρήνα.≫[30]

Στην εφαρμογή (στο κομμάτι του Server όσο και στου Client) χρησιμοποιούνται τα νήματα στον πρώτο (δηλαδή στον server) για να εκτελέστούν ταυτόχρονα τρεις σημαντικές λειτουργίες:

1. Την εκκίνηση του server
2. Την εκκίνηση του snort (προγράμματος ελέγχου)
3. Την δημιουργία, την επεξεργασία και την κρυπτογράφηση αρχείων.

Αυτές οι τρεις λειτουργίες, χάρη στα νήματα, λειτουργούν παράλληλα και απροβλημάτιστα το πρόγραμμα. Γίνεται αντίστοιχη εκκίνηση του κάθε νήματος σε ιεραρχία, και αντίστοιχο κλείσιμο και επαναφορά των νημάτων στο σύστημα, κατά τον τερματισμό του προγράμματος.

Στον Client γίνεται χρήση των Threads, τόσο στα αιτήματα στον Server, προκειμένου, σε περίπτωση καθυστέρησης, να μην επιβαρύνουν τον κεντρικό πυρήνα της εφαρμογής, όσο και σε περιπτώσεις γραφικών μεταβολών του περιβάλλοντος μέσα σε αυτά, όπως, αν πραγματοποιήθηκε η σύνδεση ή όχι, με αντίστοιχα μηνύματα. Τα γραφικά νήματα ονομάζονται runUIThread και εκτελούν το εσωτερικό των εντολών τους ανεξάρτητα απ' την υπόλοιπη λειτουργία.

Σ' ένα αντίστοιχο Thread εκτελείται η διαδικασία της αποκρυπτογράφησης του συνόλου των σφαλμάτων, η οποία, όπως θα αναλυθεί παρακάτω, απαιτεί χρόνο, και στην παρούσα περίπτωση η εφαρμογή θα "κρασάρει" αν δεν εκτελεστεί σε διαφορετικό νήμα η διαδικασία αυτή.

Αποδεικνύεται έτσι πόσο σημαντικά είναι τα νήματα στην σωστή δόμηση και λειτουργία μιας απαιτητικής εφαρμογής σαν αυτή που υλοποιείται. Εκτελούν ένα σύνολο εντολών στο "παρασκήνιο" χωρίς να εμποδίζουν τον χρήστη και καθιστούν την εφαρμογή ικανή να του παρέχει τις υπόλοιπες υπηρεσίες της.

Κεφάλαιο 4

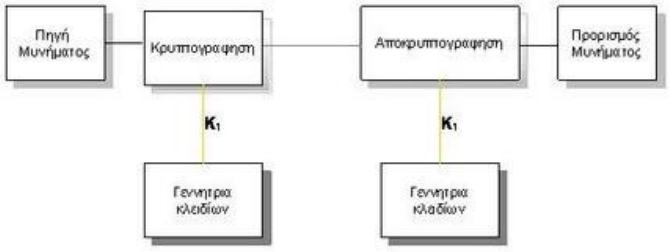
Κρυπτογράφηση

≪Η λέξη **κρυπτογραφία** (cryptography) προέρχεται από τα συνθετικά "κρυπτός+γράφω" και είναι ένα διεπιστημονικό γνωστικό πεδίο που ασχολείται με τη μελέτη, την ανάπτυξη, και τη χρήση τεχνικών κρυπτογράφησης και αποκρυπτογράφησης, με σκοπό την απόκρυψη του περιεχομένου των μηνυμάτων. Η κρυπτογραφία είναι ο ένας από τους δύο κλάδους της κρυπτολογίας (ο άλλος είναι η κρυπτανάλυση), η οποία ασχολείται με τη μελέτη της ασφαλούς επικοινωνίας. Σήμερα η κρυπτολογία θεωρείται ένα διεπιστημονικό γνωστικό πεδίο, το οποίο μπορεί να μελετηθεί ως όψη των εφαρμοσμένων μαθηματικών, της θεωρητικής πληροφορικής ή της επιστήμης ηλεκτρονικού μηχανικού. Παρεμφερείς κλάδοι είναι, αντιστοίχως, η στεγανογραφία και η στεγανοανάλυση.

Η σημασία της κρυπτογραφίας είναι τεράστια στους τομείς της ασφάλειας των υπολογιστικών συστημάτων και των τηλεπικοινωνιών. Ο κύριος στόχος της είναι να παρέχει μηχανισμούς, ώστε, δύο ή περισσότερα άκρα επικοινωνίας (π.χ. άνθρωποι, προγράμματα υπολογιστών κλπ.), να ανταλλάσσουν μηνύματα, χωρίς κανένας τρίτος, εκτός από τα δύο κύρια άκρα, να έχει τη δυνατότητα να διαβάσει την περιεχόμενη πληροφορία.

Ιστορικά, η κρυπτογραφία χρησιμοποιήθηκε για τη μετατροπή της πληροφορίας μηνυμάτων από μια κανονική, κατανοητή μορφή, σε έναν ≪γρίφο≫, που, χωρίς τη γνώση του κρυφού μετασχηματισμού, θα παρέμενε ακατανόητος. Κύριο χαρακτηριστικό των παλαιότερων μορφών κρυπτογράφησης ήταν ότι η επεξεργασία γινόταν πάνω στη γλωσσική δομή του μηνύματος. Στις νεότερες μορφές, η κρυπτογραφία χρησιμοποιεί το αριθμητικό ισοδύναμο, ενώ η έμφαση έχει μεταφερθεί σε διάφορα πεδία των μαθηματικών, όπως διακριτά μαθηματικά, θεωρία αριθμών, θεωρία πληροφορίας, υπολογιστική πολυπλοκότητα, στατιστική και συνδυαστική ανάλυση.

4.1 Συμμετρικά κρυπτοσυστήματα



Σχήμα 4.1: Μοντέλο Συμμετρικού Κρυπτοσυστήματος

Συμμετρικό κρυπτοσύστημα είναι το σύστημα εκείνο το οποίο χρησιμοποιεί, κατά τη διαδικασία της κρυπτογράφησης – αποκρυπτογράφησης, ένα κοινό κλειδί (παραπάνω εικόνα).

Η ασφάλεια αυτών των αλγορίθμων βασίζεται στη μυστικότητα του κλειδιού. Τα συμμετρικά κρυπτοσυστήματα προϋποθέτουν την ανταλλαγή του κλειδιού μέσα από ένα ασφαλές κανάλι επικοινωνίας, ή μέσα από την φυσική παρουσία των προσώπων. Αυτό το χαρακτηριστικό καθιστά δύσκολη την επικοινωνία μεταξύ απομακρυσμένων ατόμων.≫[31]

4.2 Η κρυπτογράφηση που επιλέχθηκε

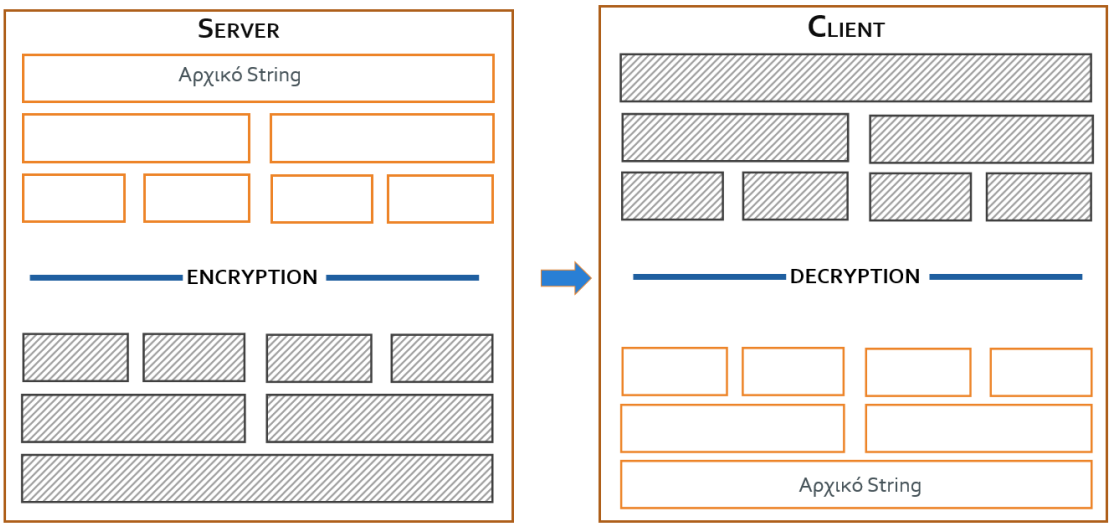
Η συγκεκριμένη κρυπτογράφηση είναι συμμετρική. Χρησιμοποιείται ο αλγόριθμος Vigenere που κάνει κρυπτογράφηση με ένα κλειδί, το οποίο είναι ίδιο και για τις δύο μεριές. Κρυπτογραφείται στη μεριά του server, αποστέλλεται στον Client, κι εκεί, με το ίδιο κλειδί, αποκωδικοποιείται αυτό που κρυπτογραφήθηκε. Έχουν δημιουργηθεί δύο μέθοδοι κρυπτογράφησης: Η μία είναι για μικρά strings και η άλλη είναι για μεγάλα. Αυτή η ανάγκη δημιουργήθηκε λόγω του χρόνου αποκωδικοποίησης που χρειαζόταν για να ολοκληρωθεί, από τον Client: Όταν υπάρχουν μεγάλα strings, δηλαδή πολλά σφάλματα που "συνέλαβε" το Snort, ο χρόνος αποκωδικοποίησης αυξάνεται δραματικά.

Συγκεκριμένα, στα χίλια (1.000) σφάλματα, ο χρόνος αποκωδικοποίησης με την απλή μέθοδο αποκωδικοποίησης χρειαζόταν σχεδόν ένα ολόκληρο λεπτό, (1min.) για να ολοκληρωθεί. Από την μεριά του Server δεν υπήρχε πρόβλημα στον χρόνο κωδικοποίησης, καθώς εκμεταλλευόντουσαν όλοι οι πόροι του συστήματος εκείνη την στιγμή. Όπως είναι γνωστό, στον Client υπάρχει περιορισμός στο ποσοστό πόρων που μπορεί να χρησιμοποιήσει η εφαρμογή λόγω των πολιτικών του Android.

Έπρεπε να βρεθεί ένας τρόπος επίλυσης αυτού του προβλήματος, τροποποιώντας και τις δυο μεριές του συστήματος όταν αυτό ήταν ανάγκη, δηλαδή σ' ένα μεγάλο String (πολλά σφάλματα). Η ιδέα ήταν να διχοτομηθεί το String σε μικρά κομμάτια, ανά 2 κάθε φορά, με όριο, το μέγεθος των κομμένων String να μην φτάνει κάτω από το μέγεθος των 40 χαρακτήρων. (Ο αριθμός 40 χαρακτήρων είναι προσεγγιστικός και αφορά περίπου την πληροφορία ενός σφάλματος).

Αμέσως μετά την διχοτόμηση, ξεκινούσε η κωδικοποίηση κάθε τμήματος με αναδρομή: Μόλις τελειώνει η προηγούμενη να συνεχίζει η επόμενη και να προστίθεται στο τέλος αυτής, δημιουργώντας έτσι το μεγάλο κωδικοποιημένο String σε χρόνο μηδαμινό σε σχέση με τον προηγούμενο τρόπο. Αυτό συνέβη καθώς ο αλγόριθμος κωδικοποίησης έχει περισσότερο χρόνο εκτέλεσης σε μεγαλύτερα String απ' ότι σε μικρότερα. Άρα, κόβοντάς το και μοιράζοντας τις εντολές κωδικοποίησης - αποκωδικοποίησης, σχεδόν εξαφανίζεται την καθυστέρηση.

Παρακάτω θα δειχθεί ένα παράδειγμα διχοτόμησης και ένας πίνακας χρόνου για την αποκωδικοποίηση από την μεριά του Client Android, καθώς, όπως αναφέρθηκε, στον Server δεν υπάρχει καθυστέρηση.



Σχήμα 4.2: Η διαδικασία διχοτόμησης

Η μέθοδος κωδικοποίησης που προτείνεται, λειτουργεί και σε άρτιο αριθμό συμβόλων/σφαλμάτων, και σε περιττό. Επειδή ο τρόπος κατασκευής λειτουργεί με αναδρομές, ήταν αναγκαστική η δημιουργία ενός global String στον οποίο προσετέθη το αποτέλεσμα της κάθε κωδικοποίησης ή αποκωδικοποίησης αντίστοιχα, από την αλλά μεριά.

Στη συνέχεια παρατίθεται ένα παράδειγμα, με όριο χαρακτήρων τον αριθμό 7. Η εκτύπωση εμφανίζει πρώτα το πρώτο μέρος της διχοτόμησης και αμέσως μετά το δεύτερο:

```
Arithmoiapotoenaduotriatesserapenteexieftaoktoenniamhdentelos
Arithmoiapotoenaduotriatessera penteexieftaoktoenniamhdentelos
Arithmoiapotoen aduotriatessera
Arithmo iapotoen
iapo toen
aduotri atessera
ates sera
penteexieftaokt oenniamhdentelos
penteex ieftaokt
ieft aokt
oenniamh dentelos
oenn iamh
dent elos
µ×ÜèÜôâÝÆãâèÔØâÕÉèãè×ÜÕÙØççÊâÕãÊáèÙÊëÝÊÙèÕÔÞèãÊáâÝÆàÜØÊáèÙÑâç
µ×ÜèÜôâÝÆãâèÔØâÕÉèãè×ÜÕÙØççÊâÕ ãÊáèÙÊëÝÊÙèÕÔÞèãÊáâÝÆàÜØÊáèÙÑâç
µ×ÜèÜôâÝÆãâèÔØâ ÕÉèãè×ÜÕÙØççÊâÕ
µ×ÜèÜôâ ÝÆãâèÔØâ
ÝÆãâ èÔØâ
ÕÉèãè×Ü ÕÙØççÊâÕ
ÕÙØç çÊâÕ
ãÊáèÙÊëÝÊÙèÕÔÞè ãÊáâÝÆàÜØÊáèÙÑâç
ãÊáèÙÊë ÝÊÙèÕÔÞè
ÝÊÙè ÕÔÞè
ãÊáâÝÆàÜ ØÊáèÙÑâç
ãÊáâ ÝÆàÜ
ØÊáè ÙÑâç
Arithmoiapotoenaduotriatesserapenteexieftaoktoenniamhdentelos
```

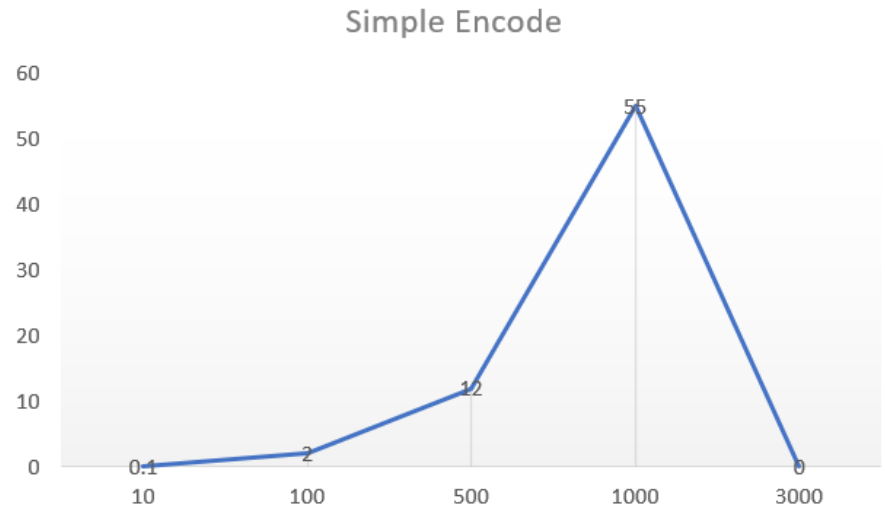
Σχήμα 4.3: Η διαδικασία διχοτόμησης ενός String και η κρυπτογράφηση του

4.3 Σύγκριση χρόνων μεταξύ κωδικοποιήσεων

Ξεκινήσε λοιπόν μια πειραματική διαδικασία για να διαπιστώθει, αν οντώς οι αριθμοί αυτοί, ευννοούν το πρόγραμμά μας, σε πολύ μεγάλο αριθμό σφαλμάτων, όπως των τριών χιλιάδων (3000). Αν και σπάνια περίπτωση, διότι στην παρούσα εργασία, στα ακαδημαϊκά πλαίσια τα οποία υλοποιήθηκε, αυτός ο αριθμός με τις συγκεκριμένες ρυθμίσεις δεν επιτεύχθηκε ποτέ.

4.3.1 Απλή Κωδικοποίηση

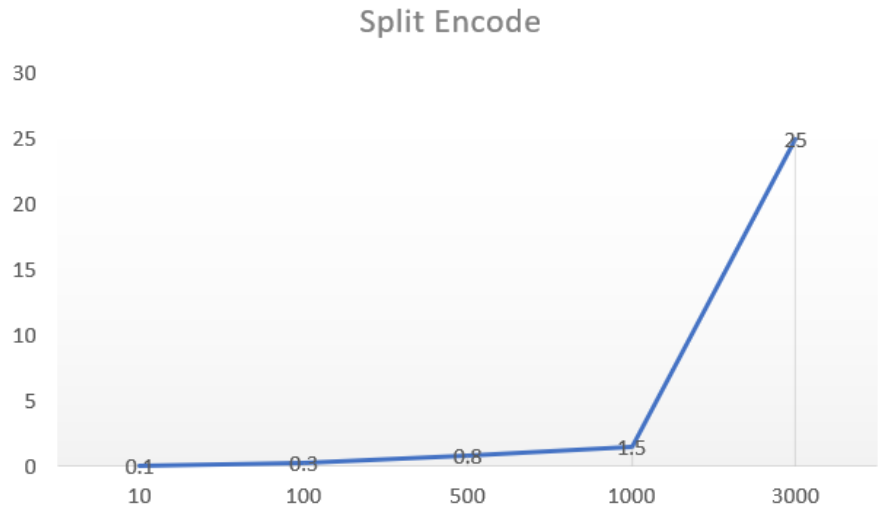
Στο διάγραμμα σφαλμάτων-χρόνου για την απλή αποκωδικοποίηση παρατηρήθηκε οτι σε μικρό αριθμό σφαλμάτων δεν υπάρχει κάποια χρονική καθυστέρηση, οπότε και η συγκεκριμένη κωδικοποίηση, θα επιλέγεται, σ'αυτές τις περιπτώσεις. Παρόλα αυτά μετά τον αριθμό των τριών χιλιάδων σφαλμάτων(3000), υπήρξε σφάλμα εκτέλεσης και στο διάγραμμα τοποθετήθηκε το 0 (μηδεν) για τον αριθμό αυτόν.



Σχήμα 4.4: Διάγραμμα σφαλμάτων-χρόνου απλής αποκωδικοποίησης

4.3.2 Κωδικοποίηση Διχοτόμησης

Στο διάγραμμα σφαλμάτων-χρόνου, για την αποκωδικοποίηση με διχοτόμηση, παρατηρήθηκε ότι, είτε σε μικρό αριθμό σφαλμάτων, ειτε -κυρίως- σε μεγάλο, δεν υπάρχει χρονική καθυστέρηση. Η πειραματική διαδικασία, ολοκληρώθηκε με επιτυχία σε όλα τα στάδια των δοκιμών, σε αντίθεση με την προηγούμενη μέθοδο.

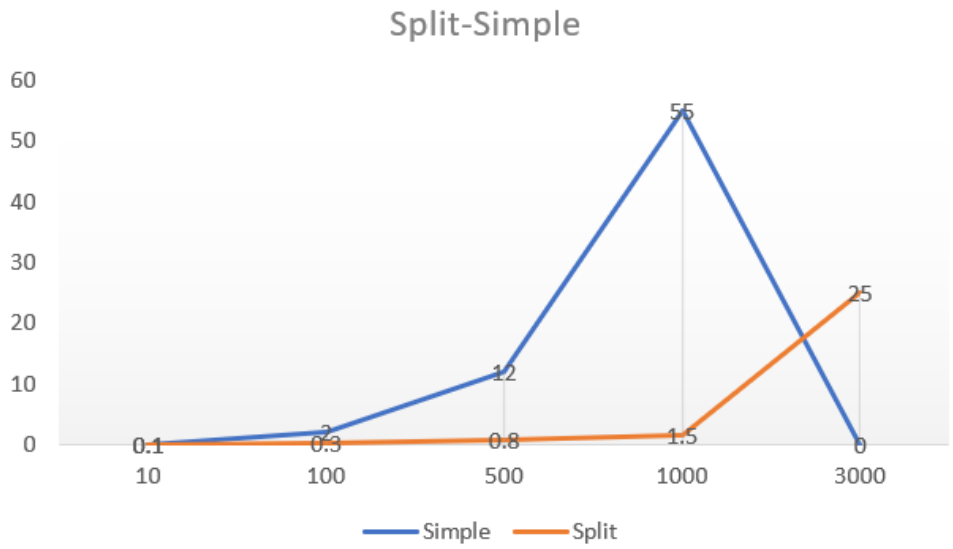


Σχήμα 4.5: Διάγραμμα σφαλμάτων-χρόνου αποκωδικοποίησης με διχοτόμηση

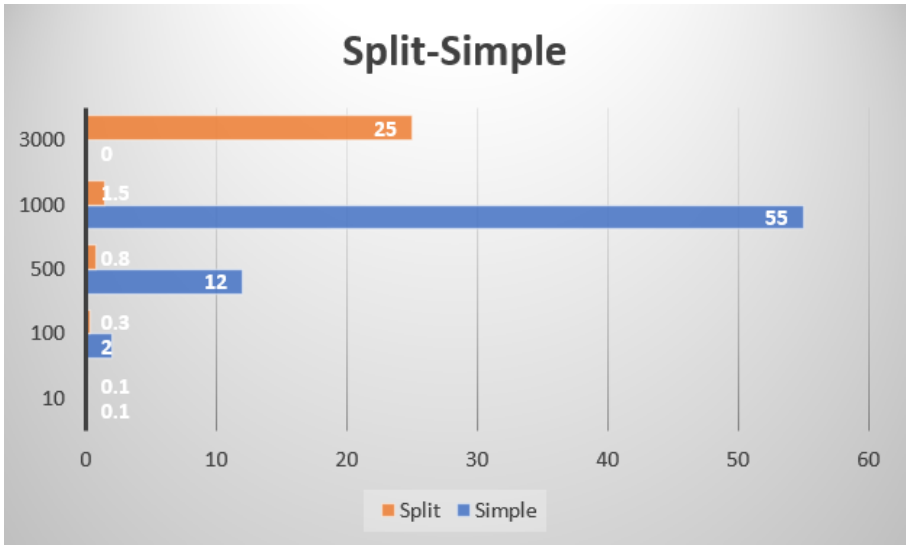
Οι χρόνοι στους οποίους ολοκληρώνεται η διαδικασία μέχρι τον αριθμό των χιλίων (1000) σφαλμάτων, (απο 0.1 sec εως 1.5 sec), θεωρούνται για τον χρήστη που χρησιμοποιεί την εφαρμογή, αμελητέοι.

4.3.3 Συμπέρασμα

Και στα δύο διαγράμματα μαζί παρατηρήθηκε ότι η αποκωδικοποίηση με την μέθοδο της διχοτόμησης στην μέτρηση των χιλίων (1000) σφαλμάτων, είναι έως και 3.600% χρονικά καλύτερη, καθώς κόστισε περίπου ενάμισι δευτερόλεπτο (1,5sec), απ' ότι πενήντα πέντε δευτερόλεπτα (55sec) με την απλή μέθοδο.



Σχήμα 4.6: Διάγραμμα σφαλμάτων-χρόνου απλής αποκωδικοποίησης και με διχοτόμηση



Σχήμα 4.7: Διάγραμμα Μπαρών σφαλμάτων-χρόνου απλής αποκωδικοποίησης και με διχοτόμηση

Αν εξαιρέσουμε την περίπτωση των τριών χιλιάδων (3000) σφαλμάτων, όπου δεν υπάρχουν δεδομένα για σύγκριση από την απλή μεθόδο, στις υπόλοιπες περιπτώσες το ποσοστό αυτό κυμαίνεται απο 0% εώς 1500%.

Ακολουθεί ο πίνακας με τους χρόνους αποκωδικοποίησης ανά αριθμό σφαλμάτων και για τις δύο μεθόδους:

Errors	Simple Encode	Split Encode
10 σφάλματα	~100 ms	~100 ms
100 σφάλματα	~2 seconds	~300ms
500 σφάλματα	~12 seconds	~800ms
1000 σφάλματα	~55 seconds	~1.5 seconds
3000 σφάλματα	~Σφάλμα εκτέλεσης	~25 seconds

Σχήμα 4.8: Πίνακας με τους χρόνους αποκωδικοποίησης

4.4 Βελτιστοποίηση του συστήματος

Παρατηρώντας τον παραπάνω πίνακα, γίνεται κατανοητό πως σε αριθμό σφαλμάτων μεγαλύτερο από τρεις χιλιάδες (3.000) και με την μέθοδο της διχοτόμησης, τα πράγματα για την εφαρμογή γίνονται δύσκολα ,καθώς θα συμπίπτουν το ένα αίτημα στον σέρβερ πάνω στο άλλο, αν ο χρόνος αιτήματος επαναλαμβάνεται σε χρόνο μικρότερο της αποκωδικοποίησης. Αν και γι' αυτό έχουν φροντίσει τα Threads να δουλεύουν παράλληλα, δεν είναι όμως μια σωστή αρχιτεκτονική κατασκευής κώδικα, πόσο μάλλον σε Android που τα restrictions είναι μεγάλα και η εφαρμογή κινδυνεύει να μπει στην BlackList του λογισμικού στο οποίο εκτελείται.

Για τον λόγο αυτόν και διαχωρίστηκε σε αρχεία στον σέρβερ το σύνολο των σφαλμάτων από τον αριθμό τους. Ο αριθμός των σφαλμάτων βρίσκεται σε διαφορετικό αρχείο απ' το περιεχόμενό τους, έτσι ο χρήστης έχει άμεση ανανέωση για τον αριθμό τους και αν αυτός επιθυμεί να δει το συνολικό περιεχόμενό του, μπορεί να ανοίξει ένα καινούριο Activity.

Πριν ανοίξει αυτό το Activity, υπάρχει η λογική αναδυόμενου παραθύρου, το οποίο ενημερώνει τον χρήστη, για την όποια καθυστέρηση. Έχει υπολογιστεί, για την διευκόλυνση του χρήστη, να έχει αυτός τη δυνατότητα να δει στην αρχική οθόνη, τα δύο πιο πρόσφατα σφάλματα. Αυτά τα δύο σφάλματα και το περιεχόμενό τους υπάρχουν στο αρχείο με τον συνολικό αριθμό των σφαλμάτων, προκειμένου να γίνει άμεσα η αποκωδικοποίηση και η ομαλή λειτουργία της εφαρμογής.

Θα γίνει αναλυτική αναφορά στη συνέχεια, για τον συγκεκριμένο τρόπο επικοινωνίας και διαχείρισης του συστήματος.

Κεφάλαιο 5

Server

Σε ότι αφορά το κομμάτι του Server, η προϋπόθεση ήταν να διαβάζει το πρόγραμμα την αντίστοιχη IP ή πύλη του agent που θα ήταν ανάγκη να γίνει "sniff"(ψάξιμο), τα πακέτα, και, στη συνέχεια, να δημιουργείται ένα logfile, το οποίο θα διατίθετο για "τράβηγμα" από τους Client. Η εγκατάσταση του Snort πήρε αρκετό χρονικό διάστημα να επιτευχθεί και να προσαρμοστεί πάνω στις απαιτήσεις της εργασίας. Το ίδιο το Snort, (το logging file του), δεν επέτρεπε να γίνει η διαχείρισή του όπως αρχικά σχεδιάστηκε. Αποφασίστηκε να εκτελείται το Snort σαν service στο παρασκήνιο, προκειμένου αφενός να εξοικονομούνται πόροι απ' το σύστημα, αφετέρου να αποφευχθεί το άνοιγμα τρίτου Thread.

Λειτούργησε όπως είχε προβλεφθεί, αλλά οι πληροφορίες που έδινε το logfile του ήταν ελλιπείς. Υπήρξε ένα πρόβλημα και με την αποκρυπτογράφηση των αρχείων αυτών, καθώς χρησιμοποιούσε άλλη βιβλιοθήκη κωδικοποίησης σαν service απ' ότι αν έτρεχε σε terminal, αλλά, και παρ' όλα αυτά, οι πληροφορίες ήταν μισές για κάθε σφάλμα και δεν εμπεριείχαν ούτε priority κατάταξη στο σφάλμα, δηλαδή πόσο σημαντικό είναι, αλλά ούτε και τα σχόλια απ' τα rulesfiles που ανέφεραν ονομαστικά τον τύπο του σφάλματος.

Παρακάτω φαίνονται η αρχική μορφή εμφάνισης ενός Warning στο αρχείο και οι πληροφορίες που εμπεριέχει:

```
1  06/17-20:13:31.941532  [**] [1:1421:19] PROTOCOL-SNMP AgentX/tcp request
   [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP}
   192.168.1.11:42486 -> 192.168.1.25:705
2  06/17-20:13:31.945300  [**] [1:1418:19] PROTOCOL-SNMP request tcp [**] [
   Classification: Attempted Information Leak] [Priority: 2] {TCP}
   192.168.1.11:34626 -> 192.168.1.25:161
3  06/17-20:13:43.581631  [**] [1:366:11] PROTOCOL-ICMP PING Unix [**] [
   Classification: Misc activity] [Priority: 3] {ICMP} 192.168.1.11 ->
   192.168.1.25
4  06/17-20:13:43.581631  [**] [1:384:8] PROTOCOL-ICMP PING [**] [
   Classification: Misc activity] [Priority: 3] {ICMP} 192.168.1.11 ->
   192.168.1.25
```

Σχήμα 5.1: Ο τρόπος εμφάνισης των σφαλμάτων στο αρχείο του Server

Παρατηρείται λοιπόν, ότι στο πρώτο μέρος αναγράφεται η ημερομηνία και η ώρα που πραγματοποιήθηκε το Warning, με ακρίβεια εκατοστών του δευτερολέπτου, αμέσως μετά αγκύλες με αστερίσκους, οι οποίοι βοηθούν στην διαχείριση της πληροφορίας από την μεριά του Client.

Στη συνέχεια ακολουθεί μια διευθυνσιοδότηση από το σύστημα για το ID του σφάλματος. Ακολουθεί ο τίτλος του σφάλματος και μετά μια κατάταξη για τον τύπο του. Έχουμε ακόμα την Priority κατάταξη, (δηλαδή πόσο σημαντικό είναι ένα σφάλμα), τον τύπο του πακέτου (UDP,ICMP,TCP) και την IP αποστολής και λήψης με τις συγκεκριμένες πύλες.

Έτσι αποφασίστηκε το Snort να εκτελείται σε διαφορετικό Thread από το τέρμιναλ, όπου υπήρχαν όσες πληροφορίες ήταν απαιτητές, όπως IP αποστολής λήψης, τύπος σφάλματος, priority κλπ. Το επόμενο πρόβλημα που παρουσιάστηκε και χρειάστηκε να αντιμετωπιστεί, ήταν το όριο χαρακτήρων που ελαμβάνοντο σε live χρόνο από το τέρμιναλ, με την εντολή γεμίσματος του αποτελέσματος της οθόνης του τέρμιναλ σε αρχείο: (εντολή ¿ όνομα αρχείου). Μετά από έρευνα στο πρόβλημα, πρόεκυψε ότι, με την εντολή /usr/bin/stdbuf –oL, το πρόβλημα αυτό λυνόταν, και στην ουσία το τερματικό δημιουργούσε ένα pipeline με το αρχείο, και το γέμιζε τον ίδιο χρόνο που εκτυπωνόταν το αποτέλεσμα από την εντολή.

5.1 Κλείδωμα IP

Για καλύτερη διαχείριση στο δίκτυο και λήψη της πληροφορίας από τον Client, έγινε η IP του σέρβερ στατική, προκειμένου, σε περίπτωση επανασύνδεσης, να μην χρειάζονται ξανά ρύθμιση οι παράμετροι, και στον σέρβερ, αλλά και στον Client. Η IP που χρησιμοποιήθηκε ήταν η 192.168.1.25 για να αποφευχθεί το IP confliction, σε περίπτωση που ο σέρβερ συνδεθεί στο ρούτερ μας μετά από μια συσκευή η οποία έχει δεσμεύσει ήδη αυτή την διεύθυνση. Συνήθως προτείνεται η IP με νούμερο 200, αλλά σε τοπικό δίκτυο, δεν θα υπήρχε ποτέ τέτοιος αριθμός συνδεδεμένων συσκευών. Σε δεύτερη φάση ανοίχτηκε και πύλη στο ρούτερ για εξαγωγή της πληροφορίας και εκτός του τοπικού δικτύου.

5.2 Priority ιδιότητες

Η Priority είναι μια κατάταξη η οποία διαχωρίζει τα σφάλματα από σημαντικά σε λιγότερο σημαντικά. Η κατάταξη αυτή διαμορφώνεται σε μια κλίμακα από το 1 έως το 4, οι κανόνες της παρούσης εφαρμογής έχουν αριθμό 0. Κοντά στο 1 βρίσκονται σφάλματα σοβαρού επιπέδου όπως προσπάθεια επίθεσης για αναγνώριση προγραμμάτων λογισμικού και κοντά στο 4 βρίσκονται λιγότερο σημαντικά, όπως περιπτώσεις αποσυνδέσεων δικτύου ή εσκεμμένη μεταφορά πακέτων (PING). Στον σέρβερ διαχωρίστηκε η επικινδυνότητα των σφαλμάτων σε πολύ σημαντικά κάτω του αριθμού 2, (συμπεριλαμβάνοντας αυτόν) και σε λιγότερο σημαντικά, πάνω από τον αριθμό αυτόν.

5.3 Rules για ping και nmap και OS nmap

Το snort, εκτός από τα προγράμματα που χρειάζεται να εγκατασταθούν προκειμένου να γίνει compile και να λειτουργήσει, διαθέτει και ένα σύστημα κανόνων (rules), το οποίο εμπεριέχει συγκεκριμένες συνθήκες για κάθε περίπτωση που θα παρατηρηθεί κάποια "ανωμαλία" στο δίκτυο.

Στο παρόν πρόγραμμα έχει επιτραπεί να ανιχνεύεται η διαδικασία επίθεσης nmap, η οποία παίρνει πληροφορίες για μια IP, για τις ανοιχτές πύλες, αλλά και για το λογισμικό το οποίο εκτελείται, και την εντολή PING που είναι λιγότερο σημαντική από την προηγούμενη, προκειμένου να καλυφθεί όλο το εύρος λειτουργίας του συστήματος.

5.4 SSH

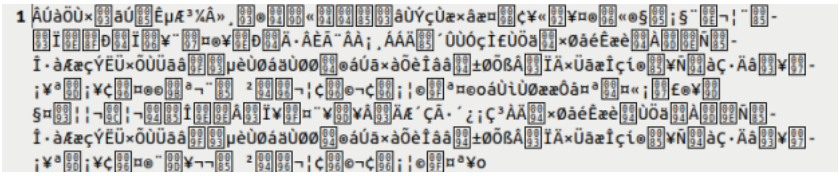
Στον Server έχει ενεργοποιηθεί και η λειτουργία SSH, η οποία αφορά απομακρυσμένο έλεγχο του υπολογιστή από άλλο σύστημα, που βρίσκεται στο ίδιο δίκτυο, ή ακόμα και εκτός, -αν είναι γνωστή η εξωτερική IP και η πύλη που δουλεύει-, προφανώς και open

port στο ρούτερ. Ενεργοποιήθηκε αυτή η λειτουργία για λόγους ασφαλείας, σε περίπτωση που χρειαστεί να απενεργοποιηθεί το σύστημα, ή να επανεκινηθεί, αν υπάρξει κάποιο γενικευμένο πρόβλημα από την εφαρμογή στον Client.

5.5 Αρχική δομή και τελική διάταξη

Κατ' αρχάς ο server παραχωρήθηκε σε ένα συγκεκριμένο IP, σε ένα ενιαίο string με πάνω πάνω τον αριθμό των σφαλμάτων, και από κάτω το περιεχόμενό τους. Ο σέρβερ εκτελείτο από την Python, και τότε το snort εκτελείτο σαν service, αλλά, εκτός από τα θέματα ελλιπούς πληροφορίας, υπήρε και θέμα ασφαλείας: Ο καθένας μπορούσε να διαβάσει τα δεδομένα απ' αυτή την IP. Έτσι δημιουργήθηκε η ανάγκη της κρυπτογράφησης, και, όπως αναφέρθηκε στο αντίστοιχο κεφάλαιο, δημιουργήθηκαν συγκεκριμένα αρχεία για κάθε ανάγκη, προκειμένου να μην σπαταλείται άσκοπα χρόνος από την λειτουργία της εφαρμογής.

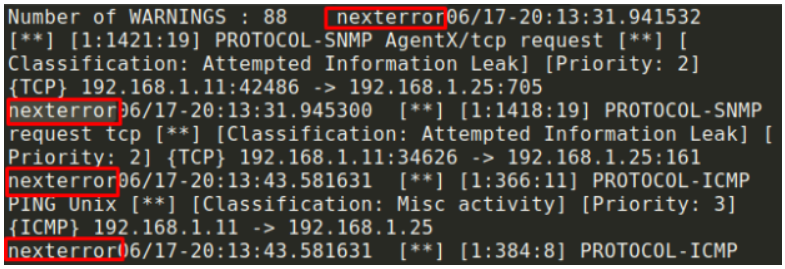
Παρακάτω φαίνεται ένα αρχείο μετά την κρυπτογράφηση, έτοιμο να ≪σερβιριστεί≫ στους Clients:



Σχήμα 5.2: Η πληροφορία μετά την κρυπτογράφηση

Όπως φαίνεται, η κρυπτογράφηση δημιούργησε ένα ενιαίο String, εντός του οποίου, κατά την αποκρυπτογράφηση, ο Client θα βρει την λέξη "Nexterror", την οποία θα χρησιμοποιεί με κατάλληλες εντολές, για να διαχωρίσει τα σφάλματα μεταξύ τους και να τα επεξεργαστεί ευκολότερα.

Στην επόμενη εικόνα φαίνεται το αρχείο πριν την κρυπτογράφηση, με την χρήση του Nexterror μεταξύ σφαλμάτων:



Σχήμα 5.3: Το nexterror σαν μέσο διαχωρισμού μεταξύ των σφαλμάτων

Τα αρχεία, τα οποία απαιτήθηκε να δημιουργήθούν και να διανεμηθούν, είναι τα εξής:

• **Logfilenc.txt** Αρχείο κρυπτογραφημένο, το οποίο περιέχει τον αριθμό των σφαλμάτων, και όλα τα Warnings και το περιεχόμενό τους. Το αρχείο αυτό είναι μεγάλο, το ίδιο και ο χρόνος αποκωδικοποίησης, όταν ο αριθμός των σφαλμάτων είναι και αυτός μεγάλος. Ο χρήστης ανοίγει αυτό το αρχείο σε τοπικό δίκτυο από την πύλη

http://192.168.1.25:4448/logfilenc.txt, μόνο αν ο ίδιος επιθυμεί να δει το σύνολο τον σφαλμάτων αριθμητικά και ταξινομημένα, με βάση ποιο έγινε νωρίτερα σε ένα νέο activity, για το οποίο ενημερώνεται για τον χρόνο που πρέπει να περιμένει.

• **Serfilenc.txt** Αρχείο κρυπτογραφημένο, το οποίο περιέχει μόνο τον αριθμό των σοβαρών σφαλμάτων. Δεν περιέχει κάποια άλλη πληροφορία. Σκοπός του είναι να αποκωδικοποιείται άμεσα και να ενημερώνει το Service της εφαρμογής που δουλεύει στο Background, όταν ενεργοποιηθεί για την πιθανότητα νέων σοβαρών σφαλμάτων ή αύξηση των προηγούμενων.

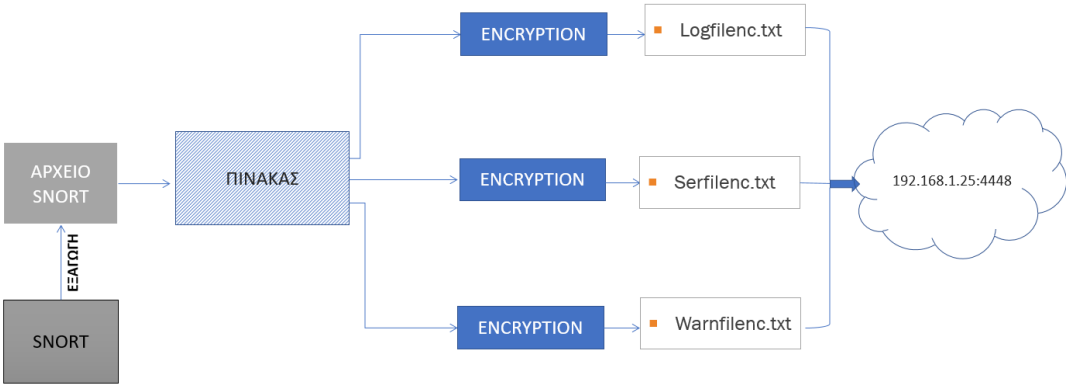
• **Warnfilenc.txt** Αρχείο κρυπτογραφημένο, το οποίο περιέχει τον συνολικό αριθμό των σφαλμάτων και το περιεχόμενο των δύο τελευταίων σφαλμάτων, προκειμένου να εμφανιστούν στην αρχική οθόνη της εφαρμογής από τον χρήστη και να είναι διαθέσιμα για προεπισκόπηση άμεσα, χωρίς να απαιτείται μεγάλος χρόνος αποκωδικοποίησης, καθώς το αρχείο δεν ξεπερνά τους 200 χαρακτήρες.

Παρατηρήθηκε λοιπόν, ότι η βάση, παρέχεται στο τοπικό δίκτυο στην πύλη και IP http://192.168.1.25:4448, εκεί όπου υπάρχουν τα τρία κρυπτογραφημένα αρχεία και MONO. Δεν υπάρχει ούτε η λογική κατασκευής του Server, ούτε κώδικας, ούτε κάποιο αρχείο ακρυπτογράφητο. Κατασκευάστηκε η βάση με τέτοιο τρόπο, έτσι ώστε ο χρήστης να μην έχει πρόσβαση σε κανέναν άλλο χώρο εκτός από αυτά τα τρία κρυπτογραφημένα αρχεία. Το νήμα (Thread) του file_modification είναι μια επαναληπτική διαδικασία, η οποία, ανάλογα με τον χρόνο που θα της δοθεί, θα επαναλαμβάνει [κάθε π.χ. δέκα δευτερόλεπτα (10sec)], την εξής διαδικασία:

1. Θα ανοίγει το αρχείο με το οποίο γεμίζεται, από την εντολή του snort, με τον τρόπο που δείχτηκε παραπάνω,
2. Θα το περνά σε έναν ≪πίνακα≫ με την εντολή readlines,
3. Θα ανοίγει τα κωδικοποιημένα αρχεία και κάποια βοηθητικά, προκειμένου να γράψει κάθε φορά πάνω τους. (Τα βοηθητικά αρχεία χρειάστηκαν για Debuging και κατανόηση).
4. Θα αριθμεί, με επαναληπτική διαδικασία, σύνολο σφαλμάτων και σοβαρότητα αυτών, (θα εχει ρυθμιστεί με τις κατάλληλες συνθήκες),
5. Θα δημιουργεί μια λογική πριν την κωδικοποίηση, για τον τρόπο που θα εξάγεται η πληροφορία σε διάφορες περιπτώσεις,
6. Θα κρυπτογραφεί τα Strings που δημιουργήθηκαν παραπάνω,
7. και θα τα γράφει στο αντίστοιχο αρχείο.

Αυτή η διαδικασία είναι επαναληπτική, υπάρχει η δυνατότητα να επαναλαμβάνεται όσο συχνά χρειάζεται. Συνήθως, για να μην γίνονται περιττά αιτήματα στον Server από τον Client, καλό είναι ο χρόνος ανανέωσης του Server να είναι ο μισός του Client, προκειμένου να συμπίπτουν οι άχρηστες ανανεώσεις, όσο πιο σπάνια γίνεται. Κάθε φορά που εκκινείται ο Server, αντικαθιστά τα παλιά αρχεία, προκειμένου να μην υπάρχουν "σκουπίδια" από παλαιότερες περιόδους, τα οποία θα επιβαρύνουν την διαχείριση των σφαλμάτων.

Παρακάτω παρατίθεται η τελική αρχιτεκτονική του Server, όπως αυτή τελικά υλοποιήθηκε.



Σχήμα 5.4: Η τελική αρχιτεκτονική του Server

Κεφάλαιο 6

Client

Όπως και στον Server, έχει δημιουργηθεί μια αντίστοιχη λογική και στον Client, δηλαδή τον χρήστη ο οποίος εξυπηρετείται από αυτόν. Η εφαρμογή προγραμματίστηκε και δοκιμάστηκε σε περιβάλλον Android, από την πλατφόρμα Android Studio, η οποία βοήθησε στο προγραμματιστικό αλλά και στο σχεδιαστικό κομμάτι της εφαρμογής.

6.1 Android σαν λειτουργικό σύστημα

≪Το Android είναι λειτουργικό σύστημα για συσκευές κινητής τηλεφωνίας το οποίο τρέχει τον πυρήνα του λειτουργικού Linux. Αρχικά αναπτύχθηκε από την Google και αργότερα από την Open Handset Alliance [32] [33]

Επιτρέπει στους κατασκευαστές λογισμικού να συνθέτουν κώδικα με την χρήση της γλώσσας προγραμματισμού Java, ελέγχοντας την συσκευή μέσω βιβλιοθηκών λογισμικού ανεπτυγμένων από την Google.[34][35] [36]

Το Android είναι κατά κύριο λόγο σχεδιασμένο για συσκευές με οθόνη αφής, όπως τα έξυπνα τηλέφωνα και τα τάμπλετ, με διαφορετικό περιβάλλον χρήσης για τηλεοράσεις (Android TV), αυτοκίνητα (Android Auto) και ρολόγια χειρός (Android Wear). Παρόλο που έχει αναπτυχθεί για συσκευές με οθόνη αφής, έχει χρησιμοποιηθεί σε κονσόλες παιχνιδιών, ψηφιακές φωτογραφικές μηχανές, συνηθισμένους Η/Υ (π.χ. το HP Slate 21) και σε άλλες ηλεκτρονικές συσκευές.

Το Android είναι το πιο ευρέως διαδεδομένο λογισμικό στον κόσμο. Οι συσκευές με Android έχουν περισσότερες πωλήσεις από όλες τις συσκευές Windows, iOS και Mac OS X μαζί.≫[37]

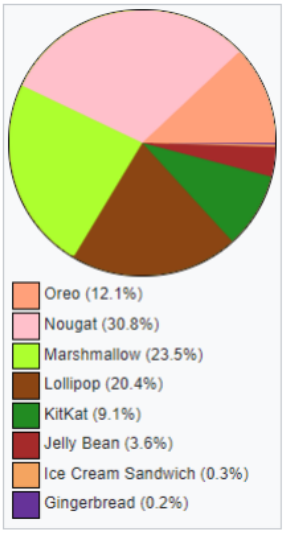
Εδώ παρουσιάζονται οι εκδόσεις Android, από την πρώτη έως και την τελευταία, σήμερα:

Έκδοση :	Κωδική Ονομασία :	Ημερομηνία	API level :
11	11	19 Φεβρουαρίου 2020 (Beta)	30
10	10	3 Σεπτεμβρίου 2019	29
9	*Pie*	6 Αυγούστου 2018	28
8.1	*Oreo*	5 Δεκεμβρίου 2017	27
8.0		21 Αυγούστου 2017	26
7.1	*Nougat*	4 Οκτωβρίου 2016	25
7.0		22 Αυγούστου 2016	24
6.0	*Marshmallow*	5 Οκτωβρίου 2015	23
5.1	*Lollipop*	9 Μαρτίου 2015	22
5.0		3 Νοεμβρίου 2014	21
4.4	*KitKat*	31 Οκτωβρίου 2013	19
4.3	*Jelly Bean*	24 Ιουλίου 2013	18
4.2		13 Νοεμβρίου 2012	17
4.1		9 Ιουλίου 2012	16
4.0	*Ice Cream Sandwich*	16 Δεκεμβρίου 2011	15
3.2	*Honeycomb*	15 Ιουλίου 2011	13
2.3	*Gingerbread*	9 Φεβρουαρίου 2011	10
2.2	*Froyo*	20 Μαΐου 2010	8
2.0	*Eclair*	26 Οκτωβρίου 2009	7
1.6	*Donut*	15 Σεπτεμβρίου 2009	4

Σχήμα 6.1: Εκδόσεις Android Studio [3]

≪Παρόλο που το Android είναι ένα προϊόν ελεύθερου λογισμικού, ένα κομμάτι της ανάπτυξης του λογισμικού συνεχίζεται σε ιδιωτικό παρακλάδι. Για να έρθει αυτό το λογισμικό σε κοινή θέαση, δημιουργήθηκε ένα παρακλάδι του, μόνο για ανάγνωση, ονόματι "Cupcake".

Το Cupcake συνήθως συγχέεται με τον τίτλο μιας ενημέρωσης, σε αντίθεση με όσα δηλώνει η ίδια η Google στην ιστοσελίδα ανάπτυξης του Android: "Το Cupcake αποτελεί ακόμη ένα έργο σε εξέλιξη, όχι μια επίσημη έκδοση." Αξιοσημείωτες αλλαγές στο λειτουργικό Android θα παρουσιαστούν στο cupcake και περιλαμβάνουν αλλαγές στο σύστημα διαχείρισης των μεταφορτώσεων (download manager), το framework, Bluetooth, το λογισμικό συστήματος, το ραδιόφωνο και το σύστημα τηλεφωνίας, εργαλεία προγραμματισμού, το κυρίως σύστημα και διάφορες εφαρμογές, καθώς και πληθώρα διορθώσεων σφαλμάτων.≫



Σχήμα 6.2: Εκδόσεις Android σε ποσοστό [3]

Όπως διαπιστώθηκε και παραπάνω, όσο νεότερο είναι το λογισμικό τού Android, τόσο πιο προσεκτικό είναι και κατασκευαστικά από θέματα ασφάλειας, και από άποψη δικαιωμάτων. Ειδικά από την έκδοση 8 και μετά, (Oreo 8. 0), έγιναν πολλές αλλαγές, τόσο στην ασφάλεια όσο και στην διαχείριση μνήμης.

Μια από αυτές τις αλλαγές παρουσιάστηκε κατά την διάρκεια της υλοποίησης, καθώς προγραμματίστηκε σε λογισμικό Android 10, με όλες τις τελευταίες αλλαγές στον πυρήνα και στα δικαιώματα. Η αλλαγή αυτή αφορούσε τις υπηρεσίες που λειτουργούν στο Background. Άλλαξε ριζικά η έννοια του Service και η λειτουργία αυτών στο παρασκήνιο καθώς απαγορεύτηκε, χωρίς τουλάχιστον να έχει ενημερωθεί ο χρήστης ότι υπάρχει αυτή η υπηρεσία από πίσω που δουλεύει.

Ένας από τους σκοπούς της παρούσας εφαρμογής ήταν, όταν αυτή κλείσει και ο χρήστης ενεργοποιήσει την λειτουργία ειδοποίησης, να ειδοποιείται όταν υπάρχουν, (έστω και ένα), σημαντικά παραπάνω σφάλματα.

Είναι υποχρεωτικό, αφού το λογισμικό το επιτρέπει, να χρησιμοποιείται η υπηρεσία με την μέθοδο Persistent Notification5 . Η ορολογία αυτή σημαίνει Μόνιμη-Επίμονη ειδοποίηση. Στην ουσία, κατά την εκκίνηση της υπηρεσίας, η εφαρμογή, και τοποθετεί στην μπάρα ειδοποιήσεων μια ειδοποίηση, την οποία ουσιαστικά δεν μπορεί ο χρήστης να την σβήσει με τον κλασσικό τρόπο, και μένει μόνιμα και τον ενημερώνει, ότι τρέχει στο παρασκήνιο η συγκεκριμένη υπηρεσία.

Γι' αυτό προγραμματιστικά τοποθετήθηκε, μια επιλογή κλεισίματος, ώστε σε περίπτωση που ο χρήστης θέλει να την απενεργοποιήσει να έχει τη δυνατότητα να το κάνει.

Γενικά, το Android είναι ένα openSource Λογισμικό και κάθε εταιρία έχει περάσει πάνω στον πηγαίο κώδικά της, τις δικές της λεπτομέρειες και προτιμήσεις. Για παράδειγμα, μια από τις υπηρεσίες συστήματος της Huawei για εξοικονόμηση μπαταρίας σε συνδυασμό με τις ρυθμίσεις των τελευταίων εκδόσεων, κάνει hibernate εφαρμογές, την στιγμή που η οθόνη είναι κλειδωμένη. Δηλαδή στην ουσία όλα τρέχουν πιο αργά εκείνη την στιγμή.

Αν ένα Handler έχει σκοπό να ξεκινήσει σε 5 λεπτά θα ανοίξει σε 10' -ίσως και καθόλου- μέχρι να ξεκλειδωθεί η οθόνη. Όποτε απαιτείται πολλή προσοχή, ώστε ότι προγραμματίζεται σε αυτό το λογισμικό, εκτός από τις ενημερώσεις που λαμβάνει κάθε έκδοση του Android, να λαμβάνεται υπόψη και τι ρυθμίσεις έχει συμπεριλάβει ο κάθε κατασκευαστής στο πακέτο του.

Η εφαρμογή (και κάθε εφαρμογή η οποία έχει γραφτεί σε Java), χωρίζεται σε συγκεκριμένες κλάσεις. Η κάθε μια έχει έναν ρόλο στο σύστημα και επικοινωνούν τα στιγμιότυπα μεταξύ τους αν αυτό χρειάζεται.

Στην συγκεκριμένη εφαρμογή οι πιο σημαντικές και χρήσιμες κλάσεις, (οι οποίες αναλύονται παρακάτω μια μια για το περιεχόμενό τους και την λειτουργία τους), είναι οι εξής:

- Main Activity
- Edit Activity
- ServNotificate Activity
- ViewLog Activity

6.2 Main Activity

Αυτή είναι η κλάση που ο κεντρικός πυρήνας καλεί την στιγμή που ανοίγει η εφαρμογή. Όλα ξεκινούν από εκεί και, αν η σχεδίαση είναι τέτοια, συνήθως τελειώνουν και εκεί. Κάθε κλάση ξεκινάει με την αρχικοποίηση ορισμένων μεταβλητών που θα χρησιμοποιηθούν παρακάτω.

Κάθε Activity ξεκινάει με την συνάρτηση onCreate, η οποία φορτώνει την πρώτη λογική και τα Layouts του συγκεκριμένου Activity. Αμέσως μόλις φορτωθεί ο Layout, χρησιμοποιείται η μέθοδος των Shared Preferences και φορτώνονται από την μνήμη του κινητού οι τιμές των μεταβλητών από προηγούμενες φορές που έχει ανοίξει η εφαρμογή, καθώς, κάθε φορά που κλείνει η εφαρμογή, οι τοπικές και οι global τιμές των μεταβλητών χάνονται. Τέτοιες μεταβλητές έχουν τιμές ίδιες με τις περιπτώσεις οδηγιών που προβάλλονται για πρώτη φορά και οι οποίες απορρίπτονται όταν δεν χρειάζεται να εμφανιστούν ξανά. Η εφαρμογή έτσι θυμάται, την επόμενη φορά που θα ανοίξει μέσω αυτής της μεθόδου, να μην ξαναεμφανίσει αντίστοιχο μήνυμα, αν ο χρήστης το έχει επιλέξει. Στο πρόγραμμα κυρίως αποθηκεύονται οι διευθύνσεις και οι πύλες του Server, ssh, username, pass, κλειδί αποκωδικοποίησης κλπ.

Κατά την πρώτη εκκίνηση η εφαρμογή διαβάζει με τον παραπάνω τρόπο αν έχει ξανανοίξει άλλη φορά και ανοίγει ένα αναδυόμενο παράθυρο, όπου ο χρήστης προτρέπεται να καταχωρήσει για τον απομακρυσμένο έλεγχο (SSH) ένα username και password και ένα κλειδί για την κρυπτογράφηση.

Αμέσως μετά εκτελείται η συνάρτηση refreshing () η οποία φροντίζει την επικοινωνία με τον σέρβερ επαναληπτικά, σε προκαθορισμένους χρόνους. (Αυτή θα αναλυθεί λεπτομερώς παρακάτω).

Όταν τελειώσει η εκτέλεση της συνάρτησης onCreate () σειρά παίρνει η συνάρτηση onResume (). Επισημαίνεται εδώ ότι και οι δυο είναι συναρτήσεις@ Override συστήματος, οι οποίες δημιουργούνται αυτόματα για την καλύτερη διαχείριση της λειτουργίας της εφαρμογής.

Στην onResume () ξεκινάει, μετά από χρόνο X, μια επαναληπτική διαδικασία κλήσης της συνάρτησης refreshing (). Ο λόγος που έχει τοποθετηθεί εκεί η επαναληπτική αυτή διαδικασία, είναι επειδή η συνάρτηση αυτή (onResume ()) καλείται, και στο άνοιγμα της εφαρμογής, -όταν ανοίγει από το παρασκήνιο ελαχιστοποιημένη-, αλλά και όταν επιστρέφει από άλλο Activity, μπροστά.

Στη συνέχεια εξετάζεται πώς λειτουργεί η συνάρτηση refreshing () που έχει αναλάβει το κομμάτι της επικοινωνίας και της κωδικοποίησης, όπως θα δειχθεί παρακάτω.

Η συνάρτηση αυτή ξεκινάει με το άνοιγμα ενός Thread μαζί με γραφικό thread το οποίο κρατάει ένα TextView με την λέξη Connecting μέχρι το προηγούμενο thread να καταφέρει να συνδεθεί. Χρησιμοποιείται το Thread (όπως έχει αναφερθεί και παραπάνω), επειδή ο χρόνος μέχρι να γίνει το αίτημα στον σέρβερ είναι άγνωστος, τι μέγεθος θα έχει, αλλά και επειδή η επιτυχία δεν είναι δεδομένη, για τον λόγο αυτό, για να μην δεσμεύεται το κεντρικό Thread από την εφαρμογή και "κρεμαστεί" οποιαδήποτε άλλη λειτουργία της, δημιουργήθηκε ξεχωριστό νήμα (thread).

Μέσα στο νήμα γίνονται ταυτόχρονα δυο αιτήματα στον σέρβερ, μέσω της βιβλιοθήκης URL και ενός Buffer, ο οποίος στην ουσία ανοίγει το σοκετ της επικοινωνίας της IP που δίνεται, δηλαδή του Server. Μόλις αυτή επιτευχθεί, διαβάζει από το url κάθε γραμμή string μέχρι την τελευταία, που είναι NULL, και τα περνά σε μια global μεταβλητή για να τα διαχειρίζεται από διάφορα σημεία του Activity.

Μέσα στην διαδικασία της σύνδεσης έχουν δημιουργηθεί κάποια Flags, τα οποία κατεβαίνουν κατά την διάρκεια της σύνδεσης, και, αν αυτή επιτευχθεί, σηκώνονται. Αν δεν συνδεθεί μένουν κατεβασμένα, (false), μέχρι την επόμενη σύνδεση. Αυτό γίνεται προκείμενου να μπλοκαριστούν σημεία ή κουμπιά της εφαρμογής απ' το να ανοίξουν χωρίς να υπάρχει κάποια σύνδεση-πληροφορία με/από τον σέρβερ.

Τα δυο αιτήματα που προαναφέρθηκαν, γίνονται στα αρχεία του σέρβερ με το σύνολο των σφαλμάτων και τα δυο τελευταία αλλά και τον συνολικό αριθμό σοβαρών σφαλμάτων, προκείμενου να εμφανιστούν οι πληροφορίες αυτές στην κεντρική οθόνη.

Η πληροφορία όμως είναι κρυπτογραφημένη, για τον λόγο αυτό ανοίγεται ένα άλλο εσωτερικό Thread στην refreshing (), το οποίο δουλεύει στο "παρασκήνιο" της εφαρμογής, προκείμενου να αποκρυπτογραφήσει απροβλημάτιστα την πληροφορία. Αμέσως μετά, και αφού τελειώσει αυτή η διαδικασία, ενημερώνονται με γραφικό thread σημεία και Textviews μέσα στην αρχική οθόνη. Επίσης το textview, που είναι ρυθμισμένο να γράφει την λέξη connecting, πλέον δείχνει πληροφορία, τον αριθμό συνολικών σφαλμάτων. Όλα αυτά προφανώς γίνονται σε δευτερόλεπτα , οι διαδικασίες που ακολουθήθηκαν για να λειτουργήσει ομαλά το αίτημα, η άντληση της πληροφορίας, η αποκρυπτογράφησή της και η ομαλή εμφάνισή της, ακόμα και σε περίπτωση λάθους στον χρήστη ήταν, εξαιρετικά λεπτομερείς και τα τεστ και το debugging διαδέχονταν το ένα το άλλο, για να επιτευχθεί το επιθυμητό αποτέλεσμα και να αποφασιστεί η αρχιτεκτονική που επιλέχθηκε.

Τα δυο σφάλματα που εμφανίζονται στην αρχική οθόνη, ξεχωρίζονται με την μέθοδο split, (χάρη στην λέξη nexterror που τοποθετήθηκε σε κάθε επόμενο σφάλμα στον σέρβερ), και τοποθετούνται σε πίνακα δύο στοιχείων, για καλύτερη διαχείριση.

Ανάλογα με το πιο απ' τα δυο (τελευταία) σφάλματα κλικάρονται στην αρχική οθόνη, αντλείται αντίστοιχα το στοιχείο του πίνακα με την εσωτερική πληροφορία του. Αμέσως αναδύεται ένα παράθυρο, το οποίο διαχωρίζει, με την μέθοδο split, την πληροφορία, ανάλογα με το κενό και τα referens τα οποία διευκολύνουν τον διαχωρισμό της συγκεκριμένης πληροφορίας. Σημειώνεται ότι, παρόμοιος τρόπος διαχωρισμού έχει δημιουργηθεί και στο Activity, με το σύνολο των σφαλμάτων, που θα εξηγηθεί παρακάτω.

Σ' αυτό το σημείο θα αναφερθεί η διαδικασία πριν το άνοιγμα των υπολοίπων κλάσεων (Activity/Services) μέσα από την MainActivity:

6.3 View Log Activity

Η συνάρτηση logopen () έχει σκοπό να ανοίξει την ViewLog Activity. Όπως προαναφέρθηκε, το περιεχόμενο ΌΛΩΝ των σφαλμάτων έχει τοποθετηθεί σε διαφορετικό αρχείο, με ειδικευμένη κωδικοποίηση, (ο τρόπος λειτουργίας έχει αναλυθεί στο κεφάλαιο 3).

Όταν πατηθεί το κουμπί που ενεργοποιεί αυτή την συνάρτηση, η συνάρτηση πρώτα ελέγχει αν πραγματοποιείται άλλη σύνδεση στον σέρβερ ή αν έχει αποτύχει η προηγούμενη, προκείμενου να μην μπει στην διαδικασία να δεσμεύσει άσκοπα πόρους, αν δεν κατάφερε ήδη να συνδεθεί το δευτερεύον Thread. Μόλις περάσει από αυτή την λογική, ξεκινάει την διαδικασία σύνδεσης με τον τρόπο που περιγράφηκε και στην refreshing (), με την μόνη διαφορά τώρα, επειδή ο χρόνος αποκωδικοποίησης είναι άγνωστος, ν' ανοίγει ένα αναδυόμενο παράθυρο, το οποίο είναι αδύνατο ο χρήστης να κλείσει, και τον ενημερώνει ότι θα πρέπει να περιμένει μέχρι να τελειώσει η αποκωδικοποίηση. Όταν αυτή τελειώσει, κλείνει το αναδυόμενο και μπαίνει στο Activity των σφαλμάτων.

Μέσω της διαδικασίας putExtra (), η οποία μεταφέρει πληροφορία μεταξύ των Activity, μεταφέρεται από τον χρήστη το String που αντλήθηκε από τον σέρβερ. Όταν το Activity ανοίξει, διαχωρίζεται με την μέθοδο split, μέσω της λέξης nexterror όπως ήδη αναφέρθηκε, και τοποθετείται σε έναν πίνακα, ο οποίος προβάλλεται από πίσω προς τα εμπρός για να έχονται τα τελευταία σφάλματα πρώτα. Έχει δημιουργηθεί ένα ScrollView με Layouts, τα οποία έχουν το καθένα μέσα το περιεχόμενο του σφάλματος.

6.4 Edit Activity

Η συνάρτηση settings () έχει σκοπό να ανοίξει την Edit Activity. Εδώ τα πράγματα είναι πολύ πιο απλά, η Main ανοίγει την συγκεκριμένη Activity και κατά την επιστροφή της από αυτήν με την onResume (), φροντίζει να αποθηκεύσει τις νέες τιμές που έχει αλλάξει ο χρήστης μέσω των SharedPreferences, όπως αναφέρθηκε και παραπάνω.

Εσωτερικά η κλάση έχει και μια διαδικασία σύνδεσης μέσω απομακρυσμένων εντολών (SSH), σε περίπτωση που θέλει ο χρήστης να τερματίσει το σύστημα ή να το επανεκκινήσει. Αξίζει να σημειωθεί ότι στο συγκεκριμένο σημείο έχουν φτιαχτεί οι συναρτήσεις και οι μέθοδοι τις οποίες θα επιλέγει ο ίδιος ο χρήστης, ποια IP θα συνδεθεί, και θα μπορούσε σε μελλοντική έκδοση να εξελιχθεί για σύνδεση σε παραπάνω από έναν Server. Παρόλα αυτά αφήνεται το λογισμικό να επιλεγεί την τοπική IP, αν είναι η συσκευή συνδεδεμένη σε Wifi, και την global, σε αντίθετη κατάσταση.

6.5 ServNotificate Activity

Η συνάρτηση servicestart () έχει σκοπό ν' ανοίξει την ServNotificate Activity. Όπως αναφέρθηκε, κάθε λογισμικό, έτσι και το Android εξελίσσεται. Συγκεκριμένα πάνω από την έκδοση 8., ο Oreo "απαγόρευσε" την ύπαρξη υπηρεσιών στο παρασκήνιο, χωρίς να το γνωρίζει ο χρήστης. Κατά την εκτέλεση λοιπόν της συνάρτησης αυτής ελέγχεται αν η συσκευή διαθέτει λογισμικό μεγαλύτερο ή ίσο της έκδοσης 8. ή όχι. Αν αυτό συμβαίνει, ξεκινάει την λεγόμενη ForegroundService, αλλιώς την κλασσική Service σε εκδόσεις μικρότερες. Ένας τρόπος να μην πατηθεί κατά λάθος αυτό το κουμπί στην αρχική οθόνη, είναι να έχει ταυτόχρονα πράσινο το κουμπί της ενεργοποίησης των ειδοποιήσεων. Σε αντίθετη περίπτωση ενημερώνει τον χρήστη για την ύπαρξη αυτού του κουμπιού.

Ανοίγοντας την υπηρεσία από την κλάση ServNotificate, επανελέγχεται ποια έκδοση Android "τρέχει" ώστε να διαχειρίζεται αναλόγως το σύστημα. Σε περίπτωση που χρησιμοποιείται έκδοση νεότερη της 8, δημιουργείται, όπως μας προτρέπει ο προγραμματιστής της Google, ένα κανάλι-channel πάνω στο οποίο περνιέται ένα ID, το οποίο χρησιμοποιείται για να δημιουργηθεί μια μόνιμη ειδοποίηση στην μπάρα, ώστε να ενημερώνεται ο χρήστης ότι η υπηρεσία είναι ενεργή.

Μέσω των SharedPrefernces πάλι δίνεται η τιμή του κλειδιού της αποκρυπτογράφησης

και το url για την σύνδεση μαζί με την πύλη και το αντίστοιχο αρχείο, περνάει από την Main μέσω της λειτουργίας PutExtra η οποία είναι υπεύθυνη για την μεταφορά πληροφορίας μεταξύ των Activity.

Η διαδικασία που ακολουθείται, είναι παρόμοια μ' αυτή που περιγράφηκε στην αρχική οθόνη. Εδώ υπάρχει μια αντίστοιχη refreshing (), η οποία, ανά συγκεκριμένο χρονικό διάστημα, επαναλαμβάνεται και αντλεί το αρχείο με τον αριθμό τον σοβαρών σφαλμάτων. Ακολούθως τα αποκρυπτογραφεί άμεσα, καθώς έχει προβλεφθεί το μέγεθος του string να είναι μικρό και να περιέχει μόνο τον αριθμό των σφαλμάτων. (Πριν τον αριθμό αυτόν προηγείται ένα string για διευκόλυνση στην αρχική οθόνη).

Μέσω της συνάρτησης substring κρατάει μόνο το νούμερο των σφαλμάτων, το "τριμάρει" σε περίπτωση που έχει κενά, και το συγκρίνει με την προηγούμενη τιμή, αφού το προσθέσει στο σύνολο. Αν για παράδειγμα υπήρχαν 10 σοβαρά σφάλματα και έχουν προκύψει 2 νέα, τότε το σύνολο των σοβαρών σφαλμάτων γίνεται 12 και ο αριθμός αυτός είναι μεγαλύτερος από το 10, όποτε και φροντίζει να σηκώσει Notification το οποίο περιέχει το σύνολο των σφαλμάτων αλλά και τον αριθμό των νέων που προέκυψαν.

Τέλος, προκείμενου ο χρήστης να μπορεί να σταματήσει την διαδικασία ειδοποίησης και να μην χρειάζεται να επανεκκινήσει το κινητό του, προκειμένου αυτή να διακοπεί ή να κλείσει απρόσκοπτα την εφαρμογή, τοποθετήθηκε πάνω στην ειδοποίηση ενημέρωσης για τον χρήστη το κουμπί Exit προκείμενου να την τερματίζει. Δημιουργήσαμε έναν τρόπο εσωτερικά τέτοιον ωστε η υπηρεσία να αναγνωρίζει αν πατήθηκε αυτό το κουμπί από την ειδοποίηση και αυτοτερματίζεται.

Η υπόλοιπη λογική είναι παρά πολύ απλή, για να εξηγηθεί αναλυτικά η αρχιτεκτονική της. Κουμπιά εξόδου τερματίζουν την εφαρμογή ή επιστρέφουν στο προηγούμενο Activity και το refresh button πάνω δεξιά, "καλεί" άμεσα την refreshing (), για να μην περιμένει ο χρήστης το delay που έχει οριστεί για να ξανασυνδεθεί για άντληση πληροφορίας.

6.6 Σχεδιαστικό τμήμα Layouts/drawables

Κάθε Εφαρμογή Android έχει ένα κομμάτι σχεδίασης το οποίο περιλαμβάνει xml αρχεία, τα οποία συνδέονται μεταξύ τους αλλά και με τις Java κλάσεις, ως ContentView για την γραφική απεικόνιση του ανάλογου Activity.

Συνδέονται Buttons, Editexts, Textviews, ImageButtons κλπ των xml αρχείων με τα αρχεία της Java μέσω ενός συστήματος διευθυνσιοδότησης. Το καθένα έχει στο Layout του, το δικό του id, και αναλόγως καλείται μέσα στο πρόγραμμα ώστε να διαχειριστεί η λογική του.

Στο συγκεκριμένο πρόγραμμα έχει κατασκευαστεί η σχεδίαση κατά τέτοιο τρόπο, ώστε η προβολή να είναι ευδιάκριτη και ομαλή σε όλες τις διαφορετικές οθόνες του εμπορίου. Τάμπλετ και έξυπνα τηλέφωνα (smartphones) έχουν διαφορετικές αναλύσεις σε διαφορετικές αναλογίες οθόνης και ίντσες.

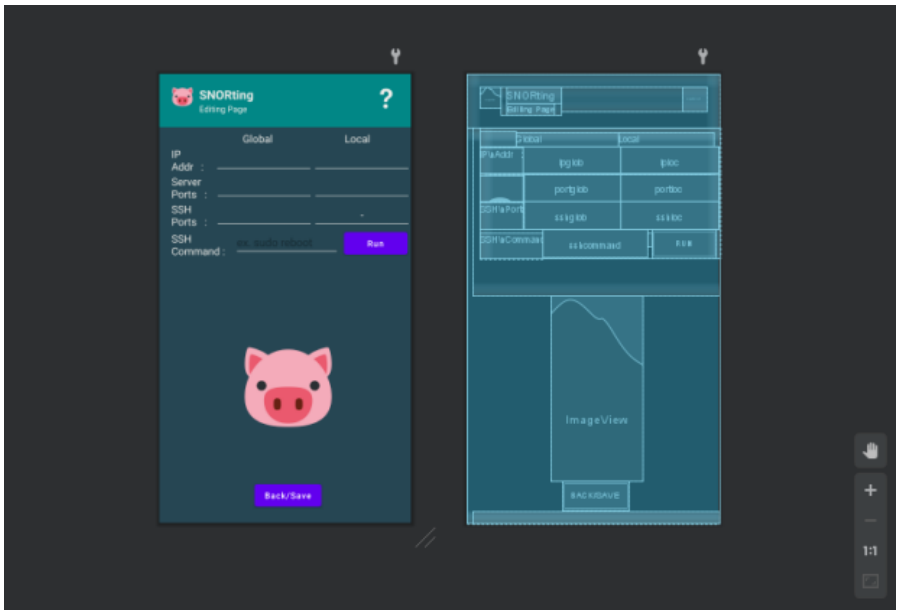
Έχει σχεδιαστεί για κάθε μια από τις παραπάνω κλάσεις, εκτός της υπηρεσίας Service που αφορά την ειδοποίηση, και ένα xml αρχείο σχεδίασης για την προβολή τους, ξεχωριστά της καθεμιάς. Η υπηρεσία δεν χρειάζεται κάποια σχεδίαση καθώς λειτουργεί στο παρασκήνιο και έχει προγραμματιστική λογική.

Έχουν σχεδιαστεί έξτρα αρχεία xml για τον σχεδιασμό αναδυόμενων παράθυρων (Dialogs) και καλούνται σε διάφορα σημεία μέσα στον κώδικα, όποτε αυτό είναι αναγκαίο, για την προβολή τους και την διευκόλυνση του χρήστη.

Εκτός από τα σχεδιαστικά xml αρχεία που βρίσκονται στον φάκελο Layouts, έχει δημιουργηθεί και ο φάκελος Drawable, ο οποίος περιέχει. png και. jpg αρχεία, δηλαδή

εικόνες οι οποίες χρησιμοποιούνται σε διάφορα σημεία του προγράμματος για την προβολή τους, τόσο για γραφιστικούς λόγους εμφάνισης όσο και για την διευκόλυνση του χρήστη.

Παρακάτω δείχνεται ένα παράδειγμα προβολής των Layouts μέσα από το πρόγραμμα Android Studio:



Σχήμα 6.3: Layouts στο Android Studio

Κεφάλαιο 7

Λειτουργία Συστήματος

7.1 Βασική Λειτουργία

Σε αυτό το κεφάλαιο θα περιγραφεί η λειτουργία του συστήματος και από τις δύο μεριές, τόσο του server όσο και του Client. Παρατίθεται ένας πίνακας με τους πόρους που χρησιμοποιήθηκαν:

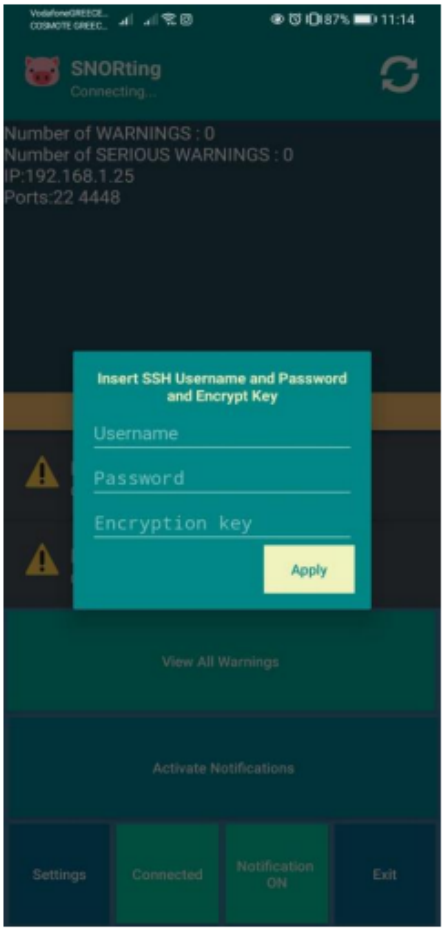
Android Studio	4.1.1
Python	3.8.5
Linux Ubuntu	20.04.1 LTS
Εκδοση Android	10.0
Virtual Box	6.1.16
Snort Version	2.9.17 GRE Build 199

Σχήμα 7.1: Πίνακας με τους πόρους που χρησιμοποιήθηκαν

Η εκκίνηση του server γίνεται σε περιβάλλον Linux.

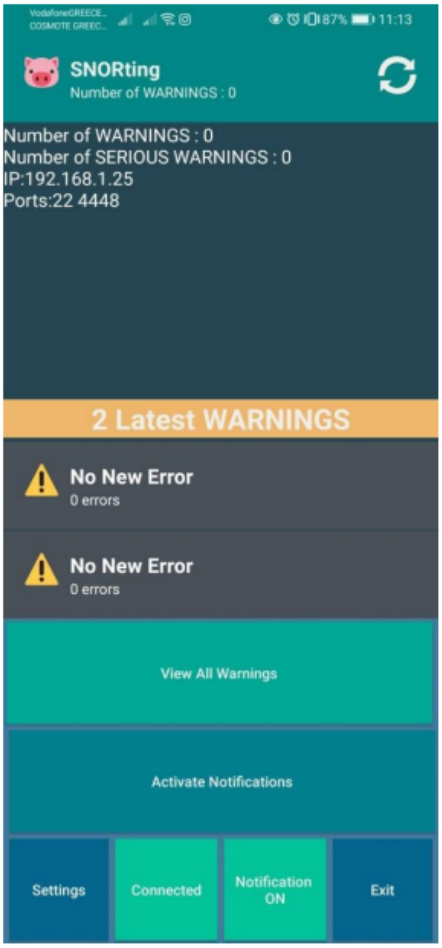
Σχήμα 7.2: Εκκίνηση του Server

Ανοίγοντας η Android εφαρμογή, στο αρχικό παράθυρο, προτρέπει τον χρήστη να εισάγει ένα όνομα χρήστη, έναν κωδικό για την ssh επικοινωνία, και ένα κλειδί για την κρυπτογράφηση, το οποίο είναι κοινό και στις δύο μεριές. Ακολούθως φαίνεται



Σχήμα 7.3: Πρώτο άνοιγμα της εφαρμογής μας

η εφαρμογή στην αρχική της μορφή, όπως είναι συνδεδεμένη στον σέρβερ και δεν υπάρχει κανένα σφάλμα ακόμα για προβολή. Με μία πρώτη ματιά φαίνεται ότι η εφαρμογή σκοπεύει να δείξει στην αρχική οθόνη τον αριθμό των σφαλμάτων, τον αριθμό των σοβαρών σφαλμάτων, μία προεπισκόπηση των δύο τελευταίων σφαλμάτων, κάποια Navigation buttons και ένα Refresh Button, το οποίο κάνει ανανέωση, δηλαδή επιτόπου αίτημα στον server για να αντλήσει πληροφορίες από αυτόν.



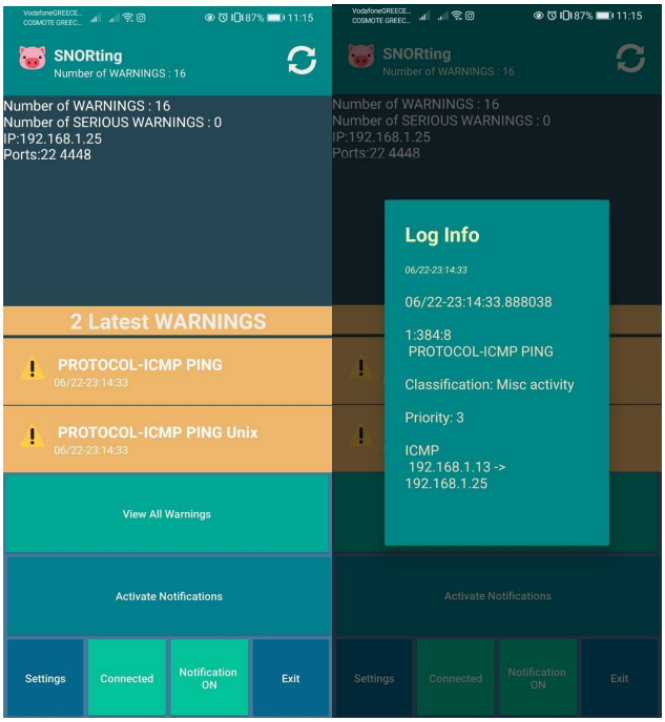
Σχήμα 7.4: Η αρχική οθόνη της εφαρμογής χωρίς σφάλματα

Παρακάτω εκτελείται η εντολή ping στη συγκεκριμένη IP του server από ένα άλλο σύστημα, ώστε να εμφανιστούν τα σφάλματα επιπέδου 3 όπως έχει προαναφερθεί.

```
vm2@vm2-VirtualBox:~$ ping 192.168.1.25
PING 192.168.1.25 (192.168.1.25) 56(84) bytes of data.
64 bytes from 192.168.1.25: icmp_seq=1 ttl=64 time=1.91 ms
64 bytes from 192.168.1.25: icmp_seq=2 ttl=64 time=0.462 ms
64 bytes from 192.168.1.25: icmp_seq=3 ttl=64 time=0.947 ms
64 bytes from 192.168.1.25: icmp_seq=4 ttl=64 time=0.914 ms
64 bytes from 192.168.1.25: icmp_seq=5 ttl=64 time=0.474 ms
64 bytes from 192.168.1.25: icmp_seq=6 ttl=64 time=0.930 ms
64 bytes from 192.168.1.25: icmp_seq=7 ttl=64 time=0.474 ms
64 bytes from 192.168.1.25: icmp_seq=8 ttl=64 time=1.05 ms
^C
--- 192.168.1.25 ping statistics ---
8 packets transmitted, 8 received, 0% packet loss, time 7045ms
rtt min/avg/max/mdev = 0.462/0.894/1.911/0.447 ms
vm2@vm2-VirtualBox:~$
```

Σχήμα 7.5: Εκτέλεση της εντολής Ping

Αμέσως φαίνεται ότι αυτά τα 8 αιτήματα πακέτων εμφανίστηκαν ως 16 στην εφαρμογή, εξαιτίας της χρήσης κι άλλου ενός rule (κανόνα) σχετικά με Ping σε unix. Πατώντας πάνω σε ένα από αυτά, ανοίγει ένα καινούργιο dialog, το οποίο αναφέρει τις πληροφορίες για το σφάλμα, όπως ώρα, ημερομηνία, διευθύνσεις, και πόσο σημαντικό είναι. Ακολου-



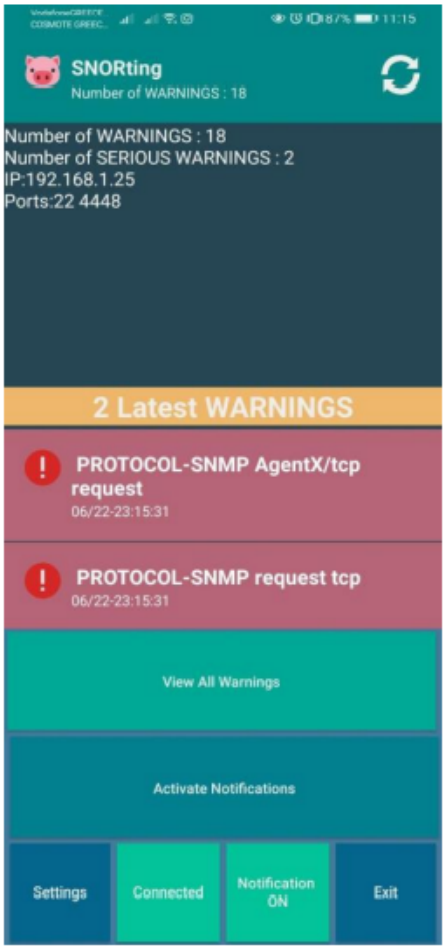
Σχήμα 7.6: Προεπισκόπηση σφαλμάτων στην αρχική οθόνη

θεί η εντολή επίθεσης nmap, η οποία σκοπεύει να εξετάσει ποιες πύλες είναι ανοιχτές στη διεύθυνση που έχει δοθεί.

```
vm2@vm2-VirtualBox:~$ nmap 192.168.1.25
Starting Nmap 7.80 ( https://nmap.org ) at 2021-06-22 23:15 EEST
Nmap scan report for DESKTOP-KHALP22 (192.168.1.25)
Host is up (0.00055s latency).
Not shown: 999 closed ports
PORT   STATE SERVICE
22/tcp open  ssh

Nmap done: 1 IP address (1 host up) scanned in 0.24 seconds
vm2@vm2-VirtualBox:~$
```

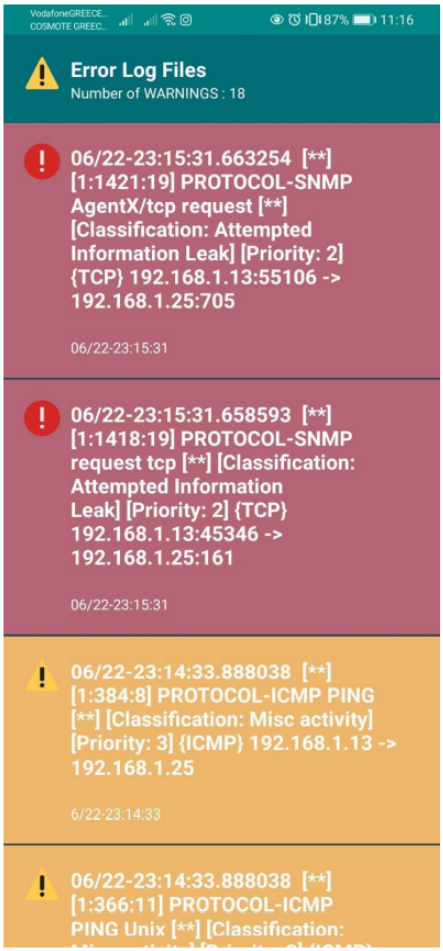
Σχήμα 7.7: Εκτέλεση εντολής NMAP

Στην εφαρμογή φαίνονται δύο νέα σοβαρά σφάλματα, που προστέθηκαν στην αρχική οθόνη ως δύο τελευταία, με χρωματισμό κόκκινο, ο οποίος παραπέμπει σε σημαντικό σφάλμα. Πατώντας πάνω τους, με την ίδια διαδικασία όπως και πριν, υπάρχει η δυνατότητα να αντληθούν περισσότερες πληροφορίες γι' αυτό. Πατώντας το κουμπί view



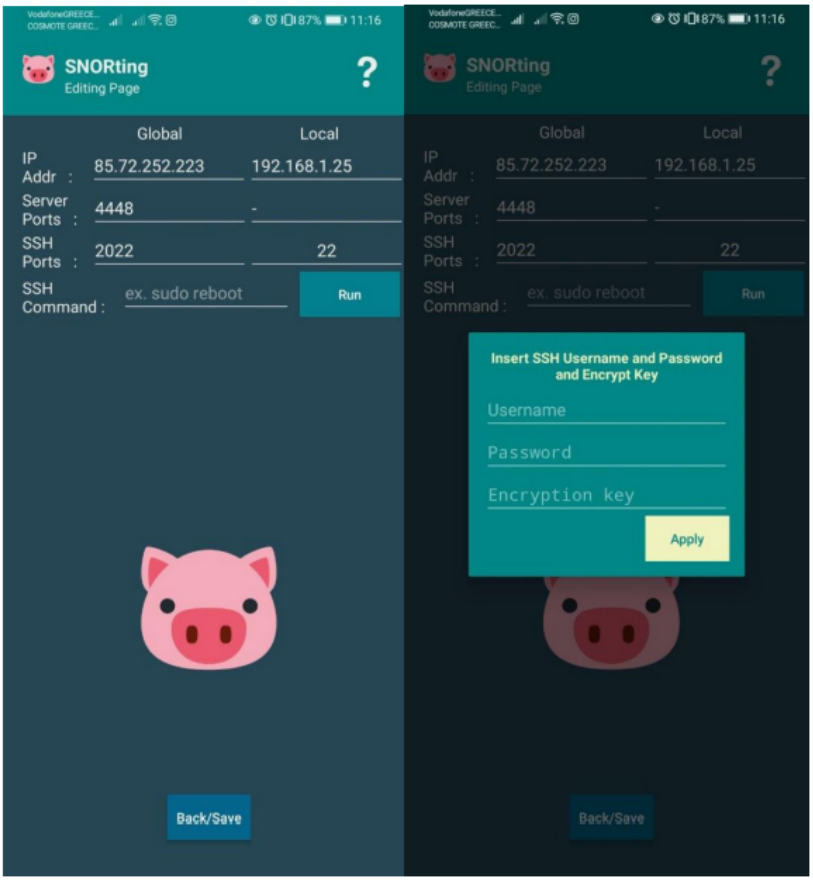
Σχήμα 7.8: Προβολή των σοβαρών σφαλμάτων με κόκκινο χρώμα

all warnings, ανοίγει η εφαρμογή ένα νέο Activity, στο οποίο παρουσιάζονται χρονολογικά, από το πιο πρόσφατο προς το παλαιότερο, τα σφάλματα τα οποία έχουν προκύψει στο συνέδριο που πραγματοποιείται.



Σχήμα 7.9: Προβολή του συνόλου των σφαλμάτων σε καινούργιο Activity

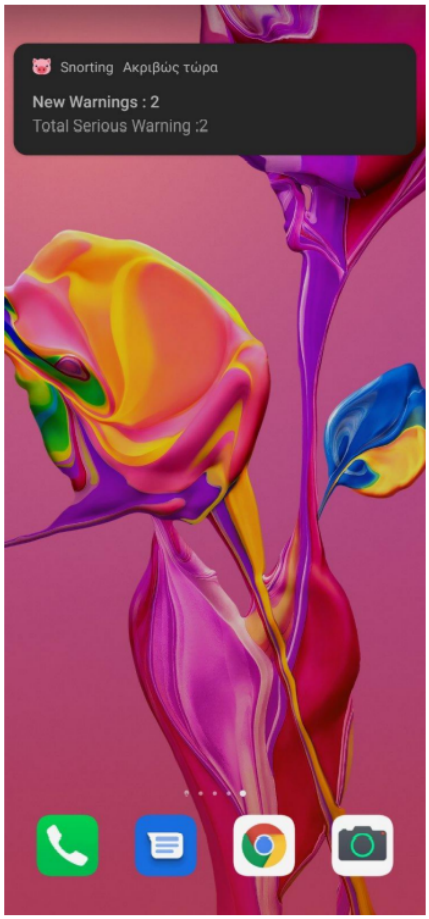
Ένα επόμενο button είναι οι ρυθμίσεις. Εκεί είναι δυνατόν να τροποποιηθούν οι διευθύνσεις σε τοπικό και global επίπεδο, τις Πύλες του ssh και του server, (ακόμα και το username και password του) και το κλειδί κρυπτογράφησης. Το συγκεκριμένο Activity έχει κουμπί βοήθειας πάνω δεξιά. Διαθέτει ssh command, με το οποίο μπορεί να επανεκκινηθεί, ή να τερματιστεί ο σέρβερ εξ αποστάσεως, μέσω του πρωτοκόλλου ssh.



Σχήμα 7.10: Άνοιγμα του Activity των ρυθμίσεων και προβολή της λογικής του

7.2 Λειτουργία ειδοποίησης (notification)

Η λειτουργία αυτή, όπως ήδη αναφέρθηκε, ενεργοποιείται σε persist notification, λόγω των περιορισμών του Android, και εκκινείται μετά το πάτημα του κουμπιού active notification στην αρχική οθόνη. Μας ειδοποιεί για τα σοβαρά σφάλματα, οπότε, αν αυτά υπάρχουν κατά την εκκίνηση, μας βγάζει το σύνολο αυτών που υπάρχουν ήδη. Στην προκειμένη περίπτωση, που έχει τρέξει μία φορά η εντολή nmap, το σύνολο είναι δύο. Εκτελείται πάλι



Σχήμα 7.11: Η ειδοποίηση με την ύπαρξη των ήδη 2 σοβαρών σφαλμάτων

από εξωτερικό μηχάνημα η εντολή nmap στον server: ...και, όπως φαίνεται, ενημερώνει

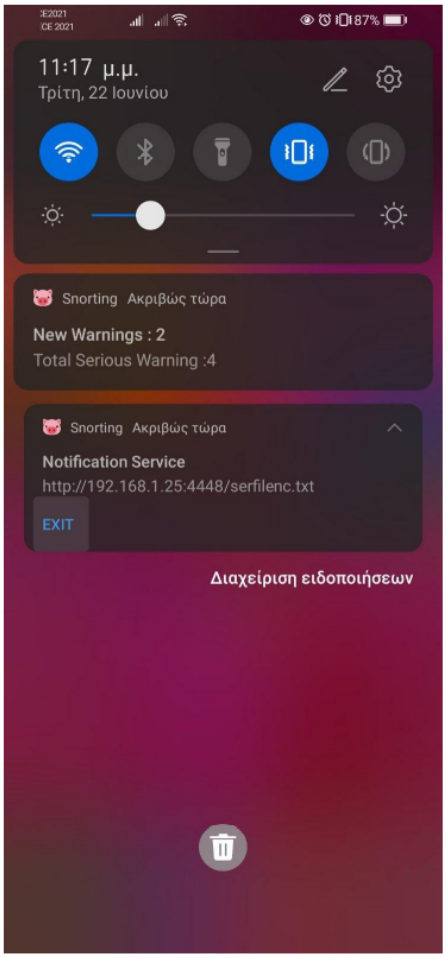
```
vm2@vm2-VirtualBox:~$ nmap 192.168.1.25
Starting Nmap 7.80 ( https://nmap.org ) at 2021-06-22 23:17 EEST
Nmap scan report for DESKTOP-KHALP22.station (192.168.1.25)
Host is up (0.00064s latency).
Not shown: 999 closed ports
PORT   STATE SERVICE
22/tcp open  ssh

Nmap done: 1 IP address (1 host up) scanned in 0.16 seconds
vm2@vm2-VirtualBox:~$
```

Σχήμα 7.12: Εκτέλεση πάλι της εντολής NMAP

ότι υπάρχουν δύο νέα σφάλματα - συνολικά τέσσερα -, μαζί με τα δύο προηγούμενα.

Σχήμα 7.13: Η εμφάνιση αυτών των 2 νέων στην μπάρα ειδοποιήσεων

Τέλος απενεργοποιείται η λειτουργία της ειδοποίησης, πατώντας στο persistent notification το κουμπί exit, τερματίζοντας έτσι το service.



Σχήμα 7.14: Το κουμπί εξόδου για τον τερματισμό της υπηρεσίας ειδοποίησης

Μια αναλυτική επίδειξη της εφαρμογής είναι διαθέσιμη στο βίντεο που είναι διαθέσιμο στη διεύθυνση: https://drive.google.com/file/d/1lFTgOQriTZyl9mu3bmItmMM7j2ZJ6Jgw/view

Κεφάλαιο 8

Επίλογος

8.1 Συμπέρασμα

Στην Εργασία αυτή αναπτύχθηκε μια εφαρμογή Android για την, σε πραγματικό χρόνο, λήψη ενημερώσεων σχετικά με ανιχνευμένες προσπάθειες εισβολής σε ένα πληροφοριακό σύστημα. Μέσω της διαδικασίας δημιουργίας της εφαρμογής, αποκτήθηκε εμπειρία σε πολλούς κλάδους στην πληροφορική, όπως στον προγραμματισμό σε Java[38], Android, Python[39], Linux, στην επικοινωνία μεταξύ δύο ή περισσότερων συσκευών-λογισμικών αλλά και στην επέμβαση στις ρυθμίσεις και στον τρόπο λειτουργίας σε προ-εγκατεστημένα προγράμματα.

Η συγκεκριμένη υλοποίηση -και για ακαδημαϊκούς σκοπούς- έγινε σε τοπικό επίπεδο, όπου ο σέρβερ είναι ταυτόχρονα, και αυτός που ελέγχεται, και αυτός που μοιράζει την πληροφορία των σφαλμάτων που έχουν ανιχνευτεί.

Διαπιστώθηκε η δυσκολία της επικοινωνίας μεταξύ των συστημάτων και οι διαδικασίες που απαιτούνται για να είναι αυτή ασφαλής και αξιόπιστη.

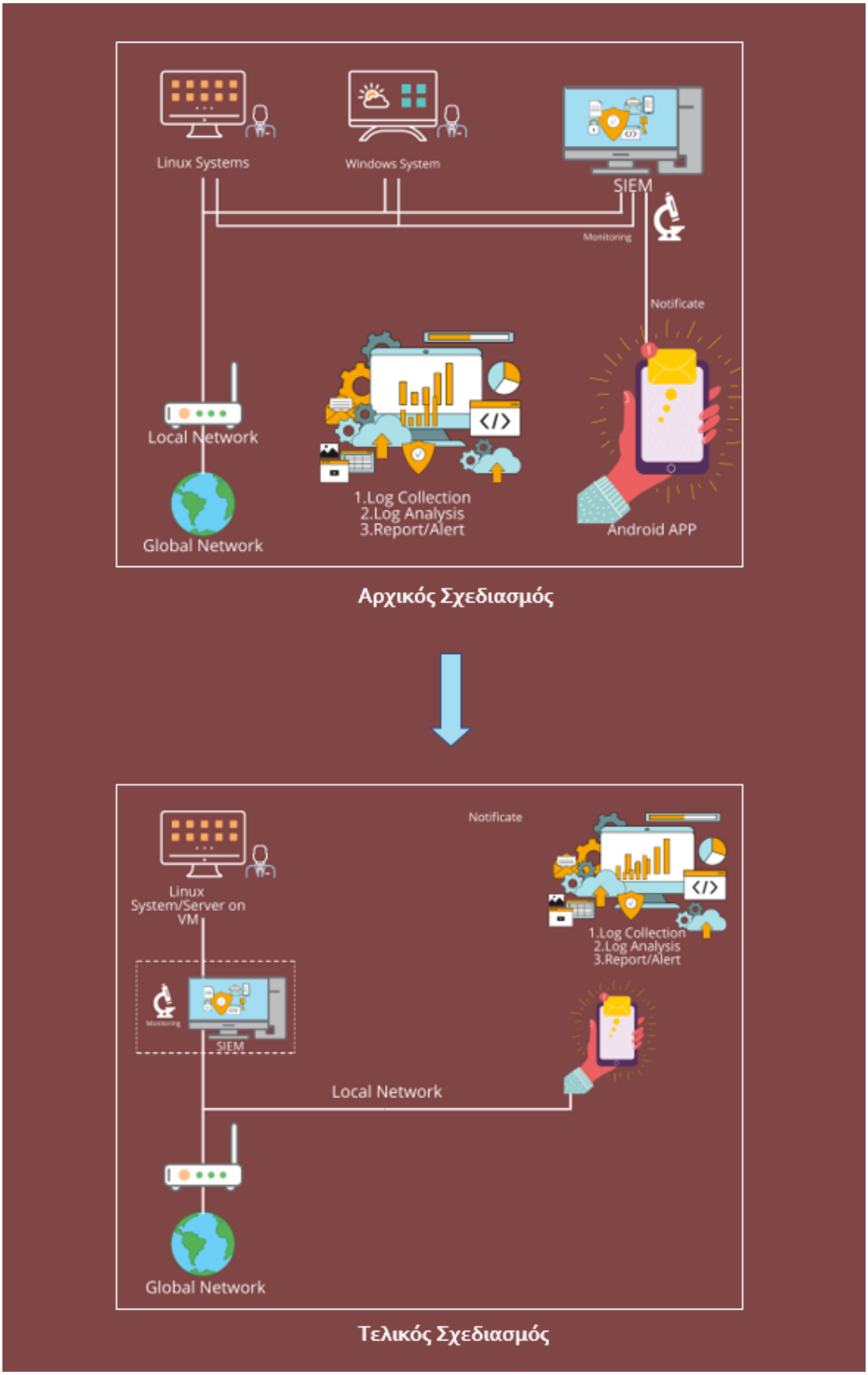
8.2 Βελτίωση-Εξέλιξη στο Μέλλον

Ο τρόπος που έχει κατασκευαστεί ο κώδικας και στα δυο τμήματα, Server και Client, είναι ανοιχτός (GitHub link: https://github.com/AlexAntonopoul/Snort/tree/Client) για να υποδεχτεί μια αναβάθμιση και βελτίωση για κάποιον που θέλει να την εξελίξει περαιτέρω.

Συγκεκριμένα:

- Η διαχείριση και η ταξινόμηση χρονικά των σφαλμάτων και η αποθήκευσή τους για ανάκτηση στο μέλλον.
- Η προσθήκη περισσοτέρων συστημάτων στην εποπτεία του SIEM.
- Η δημιουργία ενός συστήματος ταυτόχρονης ειδοποίησης και στο android app όπου θα βρίσκεται ο τεχνικός ασφαλείας αλλά και στον χρήστη που χρησιμοποιεί το σύστημα που απειλείται, εκείνη τη στιγμή, με σκοπό την ταυτόχρονη ενημέρωση χρήστη-τεχνικού.

Παρακάτω παρατίθεται η τελική τοπολογία της εργασίας, όπως αυτή τελικά υλοποιήθηκε.



Σχήμα 8.1: Τελικό σχήμα της Εργασίας

Βιβλιογραφία

[1] Oracle. Virtual box - https://www.virtualbox.org/.

[2] Android. Google. developer android - https://developer.android.com/docs.

[3] Wikipedia Android. Wikipedia android https://el.wikipedia.org/wiki/android#cite_note-cupcake15.

[4] Γκρίτζαλη Στ. Κάτσικα Σωκρ. Γκρίτζαλη Δημ. *Ασφάλεια Δικτύων Υπολογιστών*. Παπασωτηρίου, Αθήνα 2003.

[5] Tipton F. Harold. *Handbook of Information Security Management, Purposes of Information Security Management*. Krause Micki, 2004.

[6] Μαυρίδης Ι. *Ασφάλεια πληροφοριών στο διαδίκτυο*. ηλεκτρ. βιβλ., Αθήνα 2015.

[7] Dr.Dobb's Journal. *SIEM: A Market Snapshot.* Journal, 5 February 2007.

[8] Mell P. NIST. Scarfone K. *Guide to Intrusion Detection and Prevention Systems (PDF)*. ., 2007.

[9] Snort. Snort faqs and documentation - https://www.snort.org/documents.

[10] Stallings. *Κρυπτογραφία για Ασφάλεια Δικτύων Αρχές και Εφαρμογές*. Stallings, un.

[11] William Stallings. *Βασικές Αρχές Ασφάλειας Δικτύων: Εφαρμογές και Πρότυπα*. William Stallings., un.

[12] Scambray Joel. *Ασφάλεια δικτύων , McClure Stuart*. Kurtz George., 6η Έκδοση.

[13] Γ.Παπακωνσταντίνου Π.Τσανάκας Ν.Κοζύρης Α.Μανουσοπούλου Π.Ματζάκος. *Τεχνολογία Υπολογιστικών Συστημάτων και Λειτουργικά Συστήματα Γ' Ενιαίου Λυκείου Βιβλίο Μαθητή. Αρχειοθετήθηκε από το πρωτότυπο στις 17 Δεκεμβρίου 2014. Ανακτήθηκε στις 29 Μαΐου 2014.* ΟΕΔΒ. σελ. 186., ISBN 9789607251251., 1999.

[14] Sumit Prakash Tayal, Bharat Bhushan Agarwal. *Computer architecture and parallel processing (1st ed. έκδοση).* New Delhi: University Science Press., σελίδες 17–18. ISBN 978-8131804988., 2009.

[15] Sibsankar Haldar. *Operating systems. Upper Saddle River*. NJ: Pearson. σελ. 118., ISBN 9788131730225, 2010.

[16] Alex Martelli. *Python in a nutshell*. (2nd ed. έκδοση). Beijing: O'Reilly. σελ. 341., ISBN 978-0-596-10046-9., 2006.

[17] Charlie. Marsh. *True Parallelism in OCaml Αρχειοθετήθηκε από το πρωτότυπο στις 8 Μαΐου 2014.* Personal home page at princeton.edu, Ανακτήθηκε στις 28 Μαΐου 2014.

[18] Εργαστήριο Υπολογιστικών Συστημάτων. *Χρονοδρομολόγηση ΚΜΕ (PDF),24 Αυγούστου 2014.* Εθνικό Μετσόβιο Πολυτεχνείο., Ανακτήθηκε στις 29 Μαΐου 2014.

[19] M. Naghibzadeh. *Operating system concepts and techniques.* New York: iUniverse. σελίδες 80–81., ISBN 978-0-5953-7597-4., 2005.

[20] Mark Mitchell. *Advanced Linux programming (PDF) (1st ed. έκδοση).* Indianapolis, Ind.: New Riders Pub. σελίδες 61–62., ISBN 978-0735710436., 2001.

[21] David R. Butenhof. *Programming with POSIX threads ISBN 0-201-63392-2.* ([Nachdr.] έκδοση). Reading, Mass.: Addison Wesley Professional., 1997.

[22] National Instruments. *Differences Between Multithreading and Multitasking for Programmers Ανακτήθηκε στις 28 Μαΐου 2014.* National Instruments., 20 Ιανουαρίου 2014.

[23] Doug Lowe. *Networking all-in-one desk reference for dummies (3rd ed. έκδοση).* Hoboken, N.J.: Wiley., ISBN 978-0-470-17915-4, 2008.

[24] Sumit Prakash Tayal, Bharat Bhushan Agarwal. *Computer architecture and parallel processing (1st ed. έκδοση).* New Delhi: University Science Press., ISBN 978-8131804988, 2009.

[25] Dhananjay M. Dhamdhere. *Operating systems : a concept-based approach (2nd ed. έκδοση).* Boston: McGraw-Hill., ISBN 0070611947, 2006.

[26] I. A. Dhotre. *Operating Systems.* Pune - India: Technical Publications Pune., ISBN 9788184316445, 2009.

[27] Oracle. *Many-to-One Model (Green Threads). Oracle.* Oracle, Ανακτήθηκε στις 29 Μαΐου 2014.

[28] Oracle. *One-to-One Model. Oracle.* Oracle, Ανακτήθηκε στις 29 Μαΐου 2014.

[29] Oracle. *Many-to-Many Model.(Java on Solaris–Native Threads).* Oracle, Ανακτήθηκε στις 29 Μαΐου 2014.

[30] Jon Mountjoy. *Weblogic , the definitive guide , development, deployment and maintenance ,covers . Beijing [u.a.]: O'Reilly.* versions 7 and 8.1 (1st ed. έκδοση), ISBN 0-596-00432-X, 2004.

[31] Γ.Χ. Στεφανίδης Β.Α. Κάτος. *Τεχνικές Κρυπτογραφίας και Κρυπτανάλυσης*. ΖΥΓΟΣ, 2003.

[32] Developers Android Kotlin. developer.android.com/kotlin/index.htm.

[33] 25 Οκτωβρίου 2017 openhub. android analyses languages summary. - www..net/p/android/analyses/latest/languagessummary.

[34] W3SCHOOLS. www.w3schools.com/xml/schemaelementsref.asp.

[35] Industry Leaders Announce Open Platform for Mobile Devices. *Open Handset Alliance . . Δελτίο τύπου*. Industry Leaders, (5 Νοεμβρίου 2007).

[36] Stephen Shankland. *Google's Android parts ways with Java industry group.* CNET News., (12 Νοεμβρίου 2007).

[37] Rob Jackson. *Sony Ericsson, HTC Androids Set For Summer 2009.* Android Phone Fans., 10 Δεκεμβρίου 2008.

[38] Java. Java — oracle https://www.java.com/en/.

[39] Python3. Docs Python3. Developers python3 - https://docs.python.org/3/.