Ανοικτό Πανεπιστήμιο Κύπρου
Σχολή Θετικών και Εφαρμοσμένων Επιστημών

Μεταπτυχιακή Διατριβή
στα Πληροφοριακά και Επικοινωνιακά Συστήματα

Performance Evaluation of Extrusion Detection Systems

Δημήτριος Καραμούζης

Επιβλέπων Καθηγητής
Δρ. Ηλίας Αθανασόπουλος

Ιούνιος 2017

Ανοικτό Πανεπιστήμιο Κύπρου

Σχολή Θετικών και Εφαρμοσμένων Επιστημών

Performance Evaluation of Extrusion Detection Systems

Δημήτριος Καραμούζης

Επιβλέπων Καθηγητής
Δρ. Ηλίας Αθανασόπουλος

Η παρούσα μεταπτυχιακή διατριβή υποβλήθηκε
προς μερική εκπλήρωση των απαιτήσεων για απόκτηση

μεταπτυχιακού τίτλου σπουδών
στα Πληροφοριακά Συστήματα

από τη Σχολή Θετικών και Εφαρμοσμένων Επιστημών
του Ανοικτού Πανεπιστημίου Κύπρου

Ιούνιος 2017

Περίληψη

Η αύξηση της ανάπτυξης κακόβουλου λογισμικού και επιθέσεων και η συνεχής ανάγκη για προστασία της πληροφορίας οδήγησε την βιομηχανία της πληροφορικής στην ανάπτυξη συστημάτων και λογισμικού , ικανών για την ανίχνευση εισβολών.

Ενώ πρωταρχικός σκοπός των συστημάτων ανίχνευσης εισβολών είναι η προστασία του δικτύου από εξωτερικές επιθέσεις , είναι κοινώς αποδεκτό ότι οι εισβολές δεν πραγματοποιούνται μόνο σε υπηρεσίες που είναι εκτεθειμένες στο διαδίκτυο , αλλά έχουν επεκταθεί και σε επιθέσεις που ξεκινάνε από το εσωτερικό δίκτυο. Τα συστήματα ανίχνευσης εισβολών έρχονται είτε με την μορφή συσκευής είτε με τη μορφή λογισμικού , το οποίο εγκαθίσταται σε υπάρχων λειτουργικό σύστημα.

Το σύστημα ανίχνευσης εισβολών που επιλέχτηκε να εξεταστεί σε αυτήν την μεταπτυχιακή διατριβή είναι το Suricata , έκδοση 3.1.3. Το Suricata εγκαταστάθηκε σε εικονική μηχανή, χρησιμοποιώντας την πλατφόρμα VMware Workstation 12 με λειτουργικό σύστημα Ubuntu Server 16.10.

Με σκοπό την μέτρηση της απόδοσης του Suricata στο εσωτερικό δίκτυο αναπτύχθηκαν 4 διαφορετικά σενάρια σε γλώσσα python, με τα οποία αναπαράχθηκε ένας ικανός αριθμός επιτυχημένων και αποτυχημένων αιτήσεων TCP κίνησης , χρησιμοποιώντας τα πρωτόκολλα HTTP, FTP & MySQL.

Η διατριβή ανέδειξε ότι το Suricata δεν επιβαρύνει την απόδοση του δικτύου ενώ η ανάγκη του σε υπολογιστική ισχύ περιορίστηκε σε χαμηλά έως μέτρια επίπεδα.

Μικρή καθυστέρηση και αυξημένη ανάγκη σε επεξεργαστική ισχύ παρατηρήθηκε μόνο σε περιπτώσεις μεγάλου αριθμού αποτυχημένων αιτήσεων , οι οποίες με την σειρά τους δημιούργησαν και μεγαλύτερο αριθμό εντοπισμένων κινδύνων. Το Suricata λειτούργησε αποτελεσματικά και εντόπισε σημαντικούς κινδύνους όπως επίθεση ωμής βίας , σάρωση εφαρμογής ιστού , παραβίαση εταιρικής πολιτικής κ.α.

Μεγάλος αριθμός κινδύνων που εντοπίστηκαν και καταγράφηκαν από το Suricata , αναφέρονταν στα μηνύματα 3-way TCP handshake και οφείλονταν στην τεχνολογία Receive side scaling των σύγχρονων καρτών δικτύου.

Η εγκατάσταση και παραμετροποίηση ενός συστήματος IDS τύπου Suricata που έρχεται με μορφή λογισμικού δεν είναι εύκολη και απαιτεί εξειδικευμένη γνώση σε επίπεδο δικτύων , λειτουργικών συστημάτων , πρωτοκόλλων και τεχνολογιών.

Το Suricata αποδείχτηκε ικανό στην εποπτεία της κίνησης που υποβλήθηκε , παρόλα αυτά οι πολλές λειτουργίες και παραμέτροι που υποστηρίζει ενδέχεται να αποτελέσουν αντικείμενο μελλοντικών ερευνών.

Summary

Increase in developing malicious software , attacks and the constant need for data protection has lead the computer industry to start developing systems and software capable for detecting intrusions.

While the primary goal of intrusion detection systems is to protect the network of attacks coming from the outside , it is widely acceptable that intrusions are not only performed on services exposed to the Internet , but have also expanded to attacks originating from the internal network. Intrusion detection systems can be either appliances or software that can be installed on an existing operating system.

Intrusion detection system chosen to be examined in this thesis is Suricata version 3.1.3. It is an open-source software capable of monitoring the network traffic while its detection engine is based on signature matching. Suricata was installed on a virtual machine running Ubuntu Server 16.10 as operating system using the VMware Worstation 12 platform.

Aiming to measure Suricata performance on the internal network 4 different scripts were developed using Python, that reproduce a capable amount of successful and unsuccessful requests of TCP traffic using HTTP, FTP and MySQL protocols.

Thesis shown that Suricata does not cause a burden on the networks' performance while its need in processing power was limited to low up to medium level.

Small delay and increased need in processing power was observed only in cases of large number of failed requests which alongside created a large number of detected alerts. Suricata operated effectively and detected important alerts like bruteforce attack, web application scan, corporate policy violation etc.

A large number of alerts that were detected and recorded by Suricata , referred to 3-way TCP handshake messages and were caused by RSS technology of modern network adapters.

Installing and configuring a software IDS system like Suricata is not easy and requires specialized knowledge of networks , operating systems , protocols and technologies.

Suricata proved efficient in monitoring the traffic that was subjected to it , nevertheless the large number of functions and parameters that is supports may be a subject of future researches.

Ευχαριστίες

Περιεχόμενα

Ευρετήριο Εικόνων-Πινάκων

Κεφάλαιο 1
Εισαγωγή

Η σημασία της προστασίας των δεδομένων μιας υπηρεσίας, ενός πληροφοριακού συστήματος αλλά και ολόκληρης της υποδομής ενός οργανισμού (δημόσιου ή και ιδιωτικού) κρίνεται επιτακτική αν λάβει κανείς υπόψη την έξαρση που παρουσιάζουν οι απειλές που εμφανίζονται από την ύπαρξη κακόβουλου κώδικα και κατ' επέκταση λογισμικού. Παρά το γεγονός ότι οι οργανισμοί και οι χρήστες καταβάλλουν μεγάλες προσπάθειες για την προστασία των δικτύων τους και των υποδομών τους περιορίζοντας την έκθεση απόρρητων δεδομένων, οι κίνδυνοι κακόβουλων επιθέσεων και υποκλοπής στοιχείων είναι πλέον μεγαλύτεροι από ποτέ.

Η ανάγκη προστασίας και παρακολούθησης των εσωτερικών μηχανών της υποδομής ενός οργανισμού για ύποπτη δραστηριότητα δημιούργησε την έννοια της ανίχνευσης εξερχόμενης κίνησης (extrusion detection). Η ανίχνευση εξερχόμενης κίνησης στοχεύει στην ανάπτυξη μηχανισμών που είναι σε θέση να εντοπίσουν τόσο επιτυχημένες όσο και αποτυχημένες προσπάθειες χρήσης των πόρων ενός συστήματος με σκοπό την παραβίαση άλλων συστημάτων μιας υποδομής. Οι τεχνικές ανίχνευσης εστιάζουν στην ανάλυση της δραστηριότητας ενός συστήματος όσον αφορά την εξερχόμενη κίνηση με σκοπό την ανίχνευση κακόβουλων χρηστών ή και λογισμικού που μπορεί να θέσουν σε κίνδυνο την ασφάλεια των γειτονικών συστημάτων μιας υποδομής.

Στην προσπάθεια προστασίας και εποπτείας της εξερχόμενης κίνησης του δικτύου, υφίστανται διάφορα εργαλεία που μπορούν να προσφέρουν πολύτιμες υπηρεσίες όπως τα τείχη προστασίας (firewalls), τα προγράμματα προστασίας και ανίχνευσης ιών (antivirus), αλλά και τεχνικές όπως η κρυπτογραφία (cryptography). Μεταξύ εκείνων που ξεχωρίζουν είναι τα Συστήματα Ανίχνευσης Εισβολών (Intrusion Detection Systems, IDS) και τα Συστήματα Αποτροπής Εισβολών (Intrusion Protection Systems, IPS).

Σκοπός της διατριβής είναι η αποτίμηση (evaluation) εργαλείων για την εποπτεία της εξερχόμενης κίνησης δικτύου (extrusion detection) εστιάζοντας στο Suricata. Το Suricata έχει χαρακτηριστεί ως ένα εκ των καταλληλότερων λογισμικών στην εποπτεία της εξερχόμενης κίνησης ενός δικτύου. Το συγκεκριμένο λογισμικό ανοιχτού κώδικα εποπτεύει την κυκλοφορία του δικτύου χρησιμοποιώντας ισχυρούς και εκτεταμένους κανόνες για την δημιουργία υπογραφών καθώς και Lua scripting με σκοπό την ανίχνευση πολυσύνθετων απειλών.

Για την επίτευξη του σκοπού της διατριβής, παρατίθενται σε επιμέρους ενότητες τα βιβλιογραφικά και ερευνητικά δεδομένα. Ειδικότερα, στην 1η ενότητα παρουσιάζονται εισαγωγικά στοιχεία επί του θέματος, ο σκοπός της διατριβής, η αναγκαιότητα ερευνητικής προσέγγισης ενώ διατυπώνονται και τα ερευνητικά ερωτήματα. Η 2η ενότητα εστιάζει στην ασφάλεια των δικτύων και συγκεκριμένα στην αναγκαιότητά της αλλά στους μηχανισμούς με τους οποίους μπορεί αν επιτευχθεί.

Η 3η ενότητα της διατριβής αναφέρεται στα Συστήματα Ανίχνευσης Εισβολών (Intrusion Detection System, IDS) και συγκεκριμένα μέσω βιβλιογραφικών και αρθρογραφικών δεδομένων παρατίθενται τα βασικά χαρακτηριστικά τους, τα πλεονεκτήματα και μειονεκτήματά τους, η αρχιτεκτονική τους, οι μηχανισμοί λειτουργίας τους αλλά και οι τεχνικές ανίχνευσης των εισβολών. Η 4η ενότητα εστιάζει στα Συστήματα Αποτροπής Εισβολών (Intrusion Protection Systems, IPS) παραθέτοντας στοιχεία αναφορικά με την έννοιά τους, την αρχιτεκτονική, τα πλεονεκτήματα και μειονεκτήματα που παρουσιάζουν, τον τρόπο ταξινόμησής τους αλλά και τις τεχνικές αποτροπής των εισβολών.

Στις επόμενες ενότητες, παρουσιάζονται στοιχεία που σχετίζονται με την διενέργεια της έρευνας. Ειδικότερα, στην 5η ενότητα παρουσιάζεται το εργαλείο ανίχνευσης εισβολών Suricata το οποίο θα χρησιμοποιηθεί στην πραγματοποίηση της έρευνας, η 6η ενότητα εστιάζει στα στοιχεία της έρευνας (μεθοδολογία, αποτελέσματα, συστάσεις) ενώ στην 7η ενότητα της διατριβής παρατίθενται στοιχεία σχετικών εργασιών αναφορικά με την αξιολόγηση Συστημάτων Ανίχνευσης Εισβολών (IDS). Η διατριβή ολοκληρώνεται στην 8η ενότητα με την παράθεση των συμπερασμάτων που εξήχθησαν.

1.1 Σκοπός

Ο σκοπός της παρούσας μεταπτυχιακής διατριβής είναι η αποτίμηση (evaluation) εργαλείων για την εποπτεία της εξερχόμενης κίνησης δικτύου (extrusion detection) εστιάζοντας στο Suricata. Πλέον, η δραστηριότητα των δικτύων είναι τέτοια ώστε να καθίσταται απαραίτητη η ύπαρξη κατάλληλων μηχανισμών και εργαλείων αξιολόγησης ώστε να ανιχνεύονται και να αποτρέπονται κακόβουλες επιθέσεις. Το σύνολο των δεδομένων που διακινούνται στο δίκτυο, στα πληροφοριακά συστήματα αλλά και στις υποδομές των οργανισμών (δημόσιων και ιδιωτικών), επιτάσσουν τη διασφάλιση του απορρήτου, την προστασία των δεδομένων και τη σταθερότητα του δικτύου.

1.2 Αναγκαιότητα και Σπουδαιότητα Έρευνας

Η εποπτεία εξερχόμενης κίνησης στο δίκτυο είναι μια βασική τεχνική για τον περιορισμό κινδύνων και τον εντοπισμό προβλημάτων ασφαλείας. Ο μηχανισμός εποπτείας, ανάλογα με την πολυπλοκότητα των υπογραφών που χρησιμοποιούνται, μπορεί να επιφέρει σημαντικές καθυστερήσεις. Η εποπτεία της κίνησης στο δίκτυο από τη στιγμή καταβολής του απασχολεί τους απανταχού χρήστες, οργανισμούς και επιχειρήσεις. Οι εξελίξεις που έχουν σημειωθεί σε επίπεδο ταχύτητας δικτύων αλλά και όγκου δεδομένων και πληροφοριών που διαχειρίζονται, έχουν καταστήσει το ρόλο συστημάτων ανίχνευσης και αποτροπής κακόβουλων επιθέσεων ιδιαίτερα σημαντικό. Τα συστήματα ανίχνευσης και αποτροπής εισβολών έχουν μελετηθεί ενδελεχώς σε επίπεδο βιβλιογραφίας καθώς έχουν κεντρίσει το ενδιαφέρον της κοινωνίας της πληροφορικής αφού αγγίζουν το ευαίσθητο σημείο της ασφάλειας των συστημάτων αυτής.

Ενδιαφέρον σχετικά με τα συστήματα ανίχνευσης και αποτροπής των κακόβουλων επιθέσεων έχει εκφράσει σημαντικό τμήμα ερευνητών κυρίως λόγω της ταχείας και διευρυμένης ανάπτυξης και αποδοχής των δικτύων σε πολλούς τομείς της καθημερινής ζωής. Πλέον, η αρχιτεκτονική των δικτύων και τα επίπεδα ασφάλειας απασχολούν έντονα σε ερευνητικό επίπεδο ώστε να διαμορφωθεί ένα όσο το δυνατόν πληρέστερο εργαλείο εποπτείας. Οι υφιστάμενες έρευνες έχουν εστιάσει στην αξιολόγηση της

αποτελεσματικότητας και των επιπέδων ασφαλείας σε επιμέρους συστήματα ανίχνευσης κακόβουλων επιθέσεων προσφέροντας πολύτιμη πληροφόρηση.

Ωστόσο, αν και η αποτίμηση (evaluation) εργαλείων για την εποπτεία της εξερχόμενης κίνησης του δικτύου (extrusion detection) δεν βρίσκεται πλέον σε πρώιμο στάδιο αλλά έχει τύχει πανθομολογούμενης αναγνώρισης, υπάρχουν εργαλεία τα οποία δεν έχουν αξιολογηθεί επαρκώς σε πρακτικό επίπεδο όπως το Suricata. Το συγκεκριμένο εργαλείο αν και διακρίνεται μεταξύ αυτών που μπορούν να προσφέρουν υψηλά επίπεδα ασφάλειας μέσω της ανίχνευσης εισβολών και κακόβουλων επιθέσεων, εντούτοις σε πειραματικό επίπεδο δεν έχει επαρκώς αξιολογηθεί. Επομένως, διαμορφώνεται ένα αυξημένο ενδιαφέρον αποτίμησης λόγω της έλλειψης αρκετών εμπειρικών μελετών.

Η σπουδαιότητα της έρευνας είναι ακόμη μεγαλύτερη αν αναλογιστεί κανείς ότι πέραν του ερευνητικού κενού που καλείται να συμπληρώσει, μέσω της παράθεσης σχετικών εργασιών που εστίασαν στην αξιολόγηση συστημάτων ανίχνευσης εισβολών, θα μπορέσει ο αναγνώστης να κατανοήσει τον ιδιαίτερο ρόλο τους στην ομαλή λειτουργία των δικτύων αλλά και των μελλοντικών προοπτικών τους. Επιπλέον η αξιολόγηση του Suricata θα ενισχύσει ανάλογες ερευνητικές προσπάθειες στο μέλλον αλλά και την υπάρχουσα βιβλιογραφία.

Ολοκληρώνοντας, η ερευνητική προσέγγιση του θέματος της αποτίμησης του εργαλείου Suricata στην εποπτεία της εξερχόμενης κίνησης του δικτύου σε συνδυασμό με τα διαθέσιμα βιβλιογραφικά δεδομένα, καθιστούν τη συμβολή της παρούσας μεταπτυχιακής διατριβής στην ακαδημαϊκή κοινότητα εξόχως σημαντική. Αφενός, θα συμβάλλει στην κατανόηση του ρόλου τους στη σύγχρονη τεχνολογία και αφετέρου στη βελτίωση των εργαλείων ανίχνευσης και αποτροπής των εισβολών.

1.3. Ερευνητικοί Στόχοι/Ερευνητικά Ερωτήματα

Οι ερευνητικοί στόχοι της μεταπτυχιακής διατριβής εστιάζουν στα εξής:

1. **Ερευνητικός Στόχος 1:** Να διαπιστωθεί με ποια εργαλεία μπορεί να εποπτευθεί η εξερχόμενη κίνηση του δικτύου.
2. **Ερευνητικός Στόχος 2:** Να αποσαφηνιστεί ο τρόπος λειτουργίας του Suricata.
3. **Ερευνητικός Στόχος 3:** Να εντοπιστούν οι υπογραφές που εμποδίζουν την απόδοσή του.
4. **Ερευνητικός Στόχος 4:** Να διερευνηθεί ο αριθμός των πακέτων που εποπτεύονται στη μονάδα του χρόνου για ένα συγκεκριμένο σύνολο υπογραφών.

Τα ερευνητικά ερωτήματα προέρχονται και ενσωματώνουν ένα σημαντικό θεωρητικό υπόβαθρο ώστε να αποτελέσουν μια «επαλήθευση» ή μια «διερεύνηση» των όσων θεωρητικά έχουν συζητηθεί και δημοσιευτεί κατά καιρούς από πολλούς ερευνητές. Στην παρούσα μεταπτυχιακή διατριβή, η αποτίμηση (evaluation) εργαλείων για την εποπτεία της εξερχόμενης κίνησης δικτύου (extrusion detection) εστιάζοντας στο Suricata, αποτέλεσε ένα ενδιαφέρον πεδίο προς διερεύνηση. Η μελέτη και η αποτίμηση εργαλείων για την εποπτεία της εξερχόμενης κίνησης δικτύου εξαρτάται από την επίδραση επιμέρους μεταβλητών όπως είναι η διαθεσιμότητα των εργαλείων, ο τρόπος λειτουργίας τους, ο ρόλος των πρωτοκόλλων και υπογραφών. Τα ερευνητικά ερωτήματα που προκύπτουν από τους ερευνητικούς στόχους είναι τα εξής:

1. **Ερευνητικό Ερώτημα 1:** Με ποια εργαλεία μπορούμε να εποπτεύουμε την εξερχόμενη κίνηση δικτύου;
2. **Ερευνητικό Ερώτημα 2:** Πώς λειτουργεί το Suricata;
3. **Ερευνητικό Ερώτημα 3:** Με ποιες «υπογραφές» έχει προβλήματα στην απόδοσή του;
4. **Ερευνητικό Ερώτημα 4:** Πόσα πακέτα εποπτεύονται στη μονάδα του χρόνου, για ένα συγκεκριμένο σύνολο υπογραφών;

Κεφάλαιο 2
Ασφάλεια Δικτύων

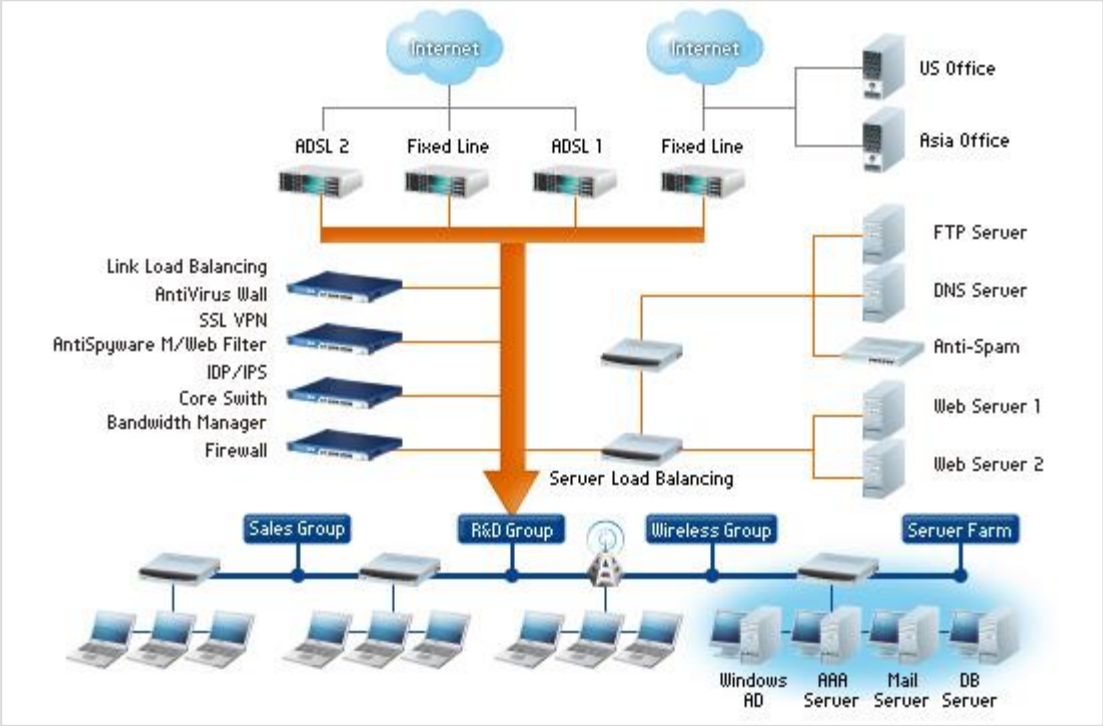
2.1 Εννοιολογική Προσέγγιση Ασφάλειας Δικτύων

Η ταχεία εξάπλωση της χρήσης των ηλεκτρονικών υπολογιστών σε καθημερινό επίπεδο ικανοποιώντας πληθώρα αναγκών, έχει δημιουργήσει έναν νέο περιβάλλον αναφορικά με τα επίπεδα ασφάλειας των χρησιμοποιούμενων δικτύων. Η εξασφάλιση υψηλών επιπέδων ασφάλειας στα δίκτυα είναι προϋπόθεση τόσο για την ομαλή λειτουργία τους όσο και για τη δημιουργία κλίματος εμπιστοσύνης από τους χρήστες. Η τεχνολογία του δικτύου αναπτύσσεται με ταχείς ρυθμούς γνωστοποιώντας παράλληλα στους χρήστες τη σημασία της ασφάλειας του δικτύου [01,02].

Τα δίκτυα υπολογιστών έχουν εξελιχθεί σε τέτοιο βαθμό διασυνδέοντας πληθώρα υπολογιστών και δημιουργώντας ταυτόχρονα σύνθετα περιβάλλοντα που απαιτούν κατάλληλη υποδομή ασφάλειας. Στην πραγματικότητα, οι περισσότεροι χρήστες δεν συνειδητοποιούν πολλές φορές το βαθμό των κινδύνων στους οποίους είναι εκτεθειμένοι κατά τη συνήθη πορεία μιας ημέρας και κατά πόσο η ορθή λειτουργία των εν λόγω δικτύων είναι ζωτικής σημασίας για την ευημερία και την επιβίωση ενός οργανισμού/μιας επιχείρησης[03].

Η ασφάλεια των δικτύων αποτελεί βασικό σημείο αναφοράς για την πληροφορική διότι τόσο ποσοτικά όσο και σε είδος αυξάνονται καθημερινά οι δικτυακές επιθέσεις. Η προστασία των υπολογιστών και η ασφάλεια των δικτύων είναι κρίσιμα ζητήματα καθώς οι κακόβουλες επιθέσεις δημιουργούν προβλήματα στο δίκτυο. Στην έννοια της ασφάλειας δικτύων συμπεριλαμβάνονται στοιχεία που εστιάζουν στη διατήρηση της εμπιστευτικότητας και ακεραιότητας του μηνύματος στο δίκτυο, στον έλεγχο ταυτότητας των χρηστών και των υπηρεσιών και στην προστασία από κακόβουλες επιθέσεις. Η ασφάλεια δικτύου εστιάζει στην προστασία παρεμβολών και

παρεμβάσεων τόσο στη λειτουργία του δικτύου όσο και σε δεδομένα και πληροφορίες χρηστών[04].



Εικόνα 2.1: Συσκευή ασφάλειας δικτύων Πηγή: [05]

Η ασφάλεια του δικτύου περιλαμβάνει πολιτικές και πρακτικές που υιοθετούνται για την πρόληψη και τον έλεγχο της μη εξουσιοδοτημένης πρόσβασης, κατάχρησης και τροποποίησης αλλά και της άρνησης ενός δικτύου ηλεκτρονικών υπολογιστών ή του ίδιου του δικτύου να παρέχει πρόσβαση σε συγκεκριμένους πόρους [06]. Στην ασφάλεια δικτύου εντάσσεται και η άδεια πρόσβασης σε δεδομένα τα οποία ελέγχονται από το διαχειριστή του δικτύου. Η ασφάλεια του δικτύου καλύπτει ένα ευρύτερο φάσμα δικτύων, τόσο δημόσια όσο και ιδιωτικά, που χρησιμοποιούνται σε καθημερινό επίπεδο για διενέργεια συναλλαγών και επικοινωνίας μεταξύ επιχειρήσεων, κυβερνητικών οργανισμών και ιδιωτών. Τα δίκτυα μπορούν να έχουν ιδιωτικό χαρακτήρα όπως για παράδειγμα εντός μιας επιχείρησης άλλα και δημόσιο όπου η πρόσβαση είναι ανοικτή στο σύνολο των χρηστών [07].

Η έννοια της ασφάλειας δικτύων αναφέρεται στις δυνατότητες προστασίας των πληροφοριών που διακινούνται αλλά και της ομαλής λειτουργίας του δικτύου μέσω ενεργειών που διασφαλίζουν τη μη εξουσιοδοτημένη χρήση των πόρων του. Στην ασφάλεια των δικτύων συμπεριλαμβάνονται ενέργειες, δράσεις και δυνατότητες

αντίστασης έναντι κακόβουλων παρεμβολών και επιθέσεων, ηθελημένων ή μη, τη διατήρηση αξιοπιστίας, την επαλήθευση της ταυτότητας των χρηστών και την ακεραιότητα και τήρηση του απορρήτου των δεδομένων [07]. Ο σχεδιασμός, η λειτουργία και η συντήρηση του δικτύου διαμορφώνουν ένα σημαντικό μέρος του έργου της διαχείρισης της ασφάλειας. Η ασφάλεια του δικτύου υποστηρίζεται από ένα σύνολο υπηρεσιών που υλοποιούνται μέσω των κατάλληλων μηχανισμών [01,08].

2.2 Η Αναγκαιότητα της Ασφάλειας Δικτύων

Η όλο και αυξανόμενη χρήση των δικτύων ηλεκτρονικών υπολογιστών έχει συνακόλουθα επηρεάσει και το βαθμό επιθέσεων σε επίπεδο επικοινωνίας. Η συντριπτική πλειοψηφία των οργανισμών, επιχειρήσεων και χρηστών έχει γίνει στόχος διαδικτυακών επιθέσεων επιτυχημένων ή όχι. Οι στόχοι των συγκεκριμένων επιθέσεων είναι πολύπλευροι και μπορεί να εστιάζουν από απλή πρόσβαση στο δίκτυο τρίτου, στην παραβίαση προσωπικών δεδομένων, στην άντληση απόρρητων πληροφοριών, για λόγους προβολής κλπ[09].

Τα περιστατικά επιθέσεων στην ασφάλεια των δικτύων αυξάνονται με ανησυχητικό ρυθμό κάθε χρόνο. Καθώς η πολυπλοκότητα των απειλών αυξάνει, το ίδιο αυξάνεται και η ανάγκη για ασφάλεια στο δίκτυο. Πλέον, οι χρήστες, τα κέντρα δεδομένων, αλλά και οι διαχειριστές πρέπει να είναι σε θέση αφενός μεν οι πρώτοι να κατανοούν τα βασικά για την ασφάλεια δικτύου και οι δε να διαμορφώνουν συνθήκες ασφαλούς πλοήγησης σε αυτά[10].

Οι συγκεκριμένες συνθήκες έχουν καταστήσει την ασφάλεια των δικτύων σε ύψιστη προτεραιότητα καθώς σε αντίθεση περίπτωση οι συνέπειες μιας παραβίασης δικτύου μπορεί να είναι ιδιαίτερα βλαπτικές όπως η απώλεια προσωπικών δεδομένων, η οικονομική ζημιά, η βλάβη στο κύρος και την αξιοπιστία ενός οργανισμού / μιας επιχείρησης. Ειδικότερα, για κάθε χρήστη (ιδιώτη, επιχείρησης) ορισμένες πληροφορίες και δεδομένα ταυτίζονται με την ανάγκη τήρησης του απορρήτου σε οποιονδήποτε τρίτο ή ανταγωνιστή [09,11].

Η αναγκαιότητα της ασφάλειας των δικτύων θεωρείται σημαντική καθώς αποτελεί τη βάση σταθερότητας ολόκληρου του δικτύου συναλλαγών και επικοινωνιών μεταξύ

των διαφόρων χρηστών. Οι εξελίξεις στην τεχνολογία είναι τέτοιες ώστε παρέχονται όλο και μεγαλύτερες δυνατότητες ασφάλειας των δικτύων βελτιώνοντας τα επίπεδα ασφαλούς επικοινωνίας. Για την προστασία των χρηστών, προτεραιότητα πριν τη οποιαδήποτε επικοινωνία στο διαδίκτυο αποτελεί η ασφάλεια. Η επίτευξη ασφάλειας στο δίκτυο βασίζεται στην αξιοποίηση κατάλληλων μηχανισμών και εργαλείων (π.χ. firewalls, antivirus κλπ) αλλά και στη λήψη μέτρων προστασίας από την πλευρά των χρηστών [11].

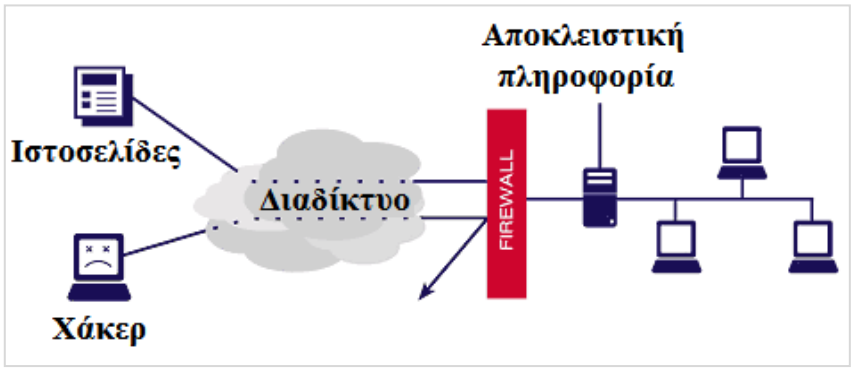
2.3 Προσέγγιση Μηχανισμών Ασφάλειας Δικτύων

Η ασφάλεια των δικτύων είναι μια σημαντική πτυχή των καθημερινών συναλλαγών και επικοινωνιών η οποία επιτυγχάνεται μέσω νέων μεθόδων και εργαλείων. Τα δίκτυα υπολογιστών καλούνται να διαχειριστούν και να αντιμετωπίσουν πολυάριθμες απειλές μέσω της ασφάλειας η οποία επιτυγχάνεται σε τρία (3) διαφορετικά επίπεδα: πρόληψη, ανίχνευση, αντιμετώπιση. Οι μηχανισμοί ασφάλειας του δικτύου σε επίπεδο πρόληψης εστιάζουν σε αποτροπή των επιθέσεων επιθέσεις πριν από την είσοδό τους, σε επίπεδο ανίχνευσης στον εντοπισμό των επιθέσεων αφού πραγματοποιηθούν ενώ η αντιμετώπιση εστιάζει στην επίλυση των προβλημάτων που δημιουργήθηκαν τα οποία δεν μπόρεσαν να προβλεφθούν [12].

Η πρόληψη των επιθέσεων μέσω των σχετικών μηχανισμών είναι η ιδανική λύση, σε σύγκριση με την ανίχνευση και την αντιμετώπιση αλλά είναι πρακτικά αδύνατο να αποτραπεί το 100% των επιθέσεων. Οι τεχνικές ανίχνευσης παρέχουν αποτελέσματα που μπορούν να χρησιμοποιηθούν για την πρόληψη περαιτέρω επιθέσεων ενώ η αντιμετώπιση δημιουργεί νέες δυνατότητες αλλά και νέες ανάγκες στο δίκτυο. Έτσι, και τα τρία στάδια ασφάλειας συνδυαστικά προσφέρουν μια αποτελεσματική προσέγγιση για την επίτευξη της ασφάλειας του δικτύου [13]. Μεταξύ των σημαντικότερων μηχανισμών ασφάλειας των δικτύων αναφέρονται τα τείχη προστασίας (firewalls), τα προγράμματα προστασίας και ανίχνευσης ιών (antivirus) και τα συστήματα πρόληψης και ανίχνευσης των εισβολών [14].

1. **Τείχη προστασίας (firewalls).** Για να εξασφαλίσει ένα δίκτυο ή υποδίκτυο την ασφάλεια συνήθως φιλτράρει την επικοινωνία με συγκεκριμένα κριτήρια όπως

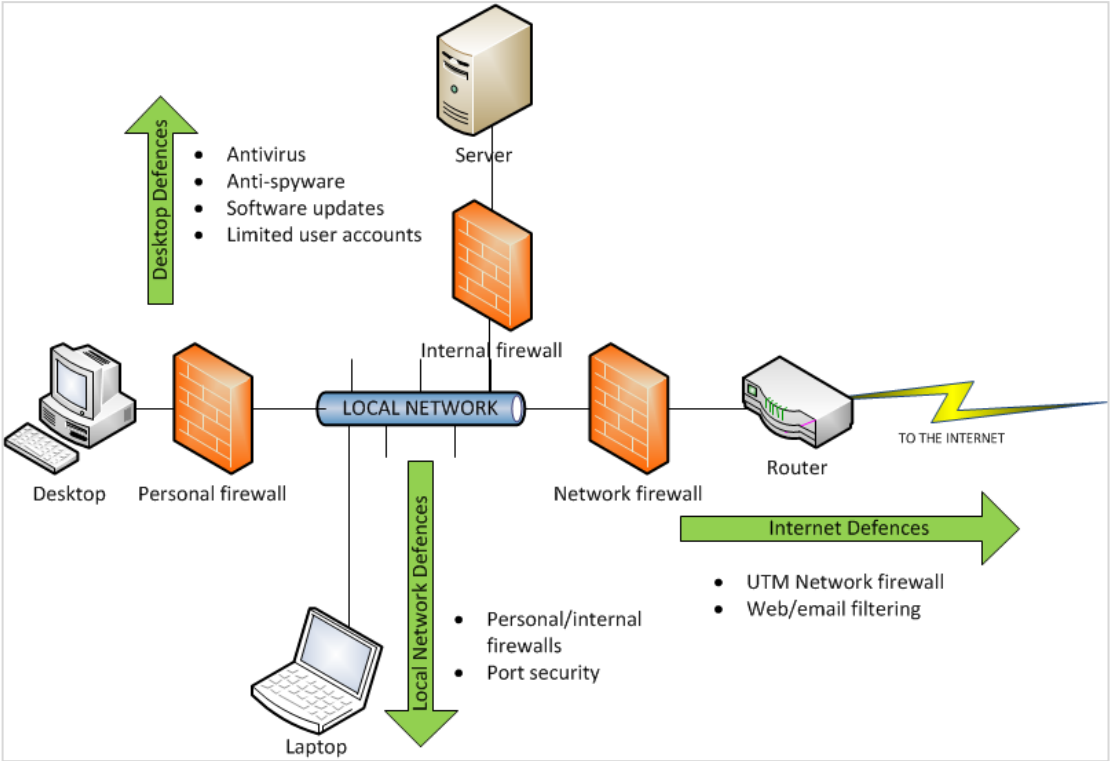
η προέλευση, ο προορισμός και το πρωτόκολλο. Τα firewall αποτελούν μια κοινή άμυνα ασφάλειας των δικτύων και σήμερα χαρακτηρίζονται ως αναπόσπαστο μέρος κάθε δικτύου [15]. Ένα τείχος προστασίας μπορεί να είναι υλικό (hardware) ή λογισμικό (software) ενώ η λειτουργικότητά του βασίζεται σε μηχανισμούς φιλτραρίσματος που προστατεύουν έναντι ανεπιθύμητων επιθέσεων [16]. Η θεμελιώδης λειτουργία ενός firewall εστιάζει στην αξιολόγηση της πρόσβασης βάσει των κριτηρίων που έχουν προκαθοριστεί. Το μειονέκτημα των firewalls είναι ότι δεν μπορούν να προστατεύσουν πλήρως ένα εσωτερικό δίκτυο από τις εσωτερικές επιθέσεις. Το τείχος προστασίας παρέχει το πλεονέκτημα της πρόσθετης ασφάλειας έναντι κακόβουλων επιθέσεων [17].



Εικόνα 2.2: Τείχος προστασίας (firewall) Πηγή: [18]

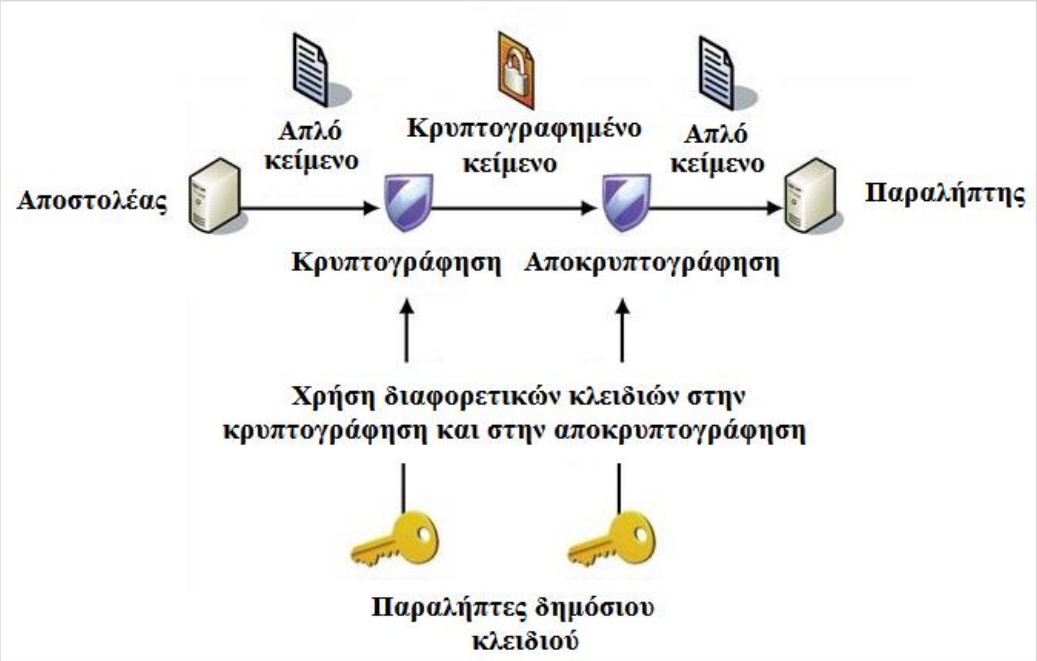
2. **Προγράμματα προστασίας και ανίχνευσης ιών (antivirus).** Οι ιοί των υπολογιστών είναι προγράμματα τα οποία πλήττουν τα δεδομένα του υπολογιστή ή προκαλούν δυσλειτουργίες σε αυτό [19]. Στο περιβάλλον του δικτύου, ένας ιός αποτελεί ανυπολόγιστη απειλή και μπορεί να είναι καταστροφικός. Τα προγράμματα προστασίας από ιούς (antivirus) αποτελούν λογισμικό που μπορεί να εγκατασταθεί σε έναν υπολογιστή με σκοπό τον εντοπισμό, την πρόληψη και τη λήψη αποφάσεων σχετικά με τη διαγραφή ή την απομόνωση κακόβουλων προγραμμάτων, ιών και worms [20]. Αν και τα προγράμματα προστασίας από ιούς παρακολουθούν την αρτιότητα των αρχείων και τυχόν προσπάθειες τροποποιήσεών τους, δεν είναι σε θέση να μπλοκάρουν την ελεύθερη κυκλοφορία ιών ή worms στο δίκτυο. Τα προγράμματα προστασίας και ανίχνευσης ιών εγκαθίστανται σε συγκεκριμένα

σημεία του δικτύου και ειδικότερα στο σημείο επαφής με το περιβάλλον του χρήστη [21].



Εικόνα 2.3: Αναλυτική προσέγγιση ασφάλειας δικτύου Πηγή: [22]

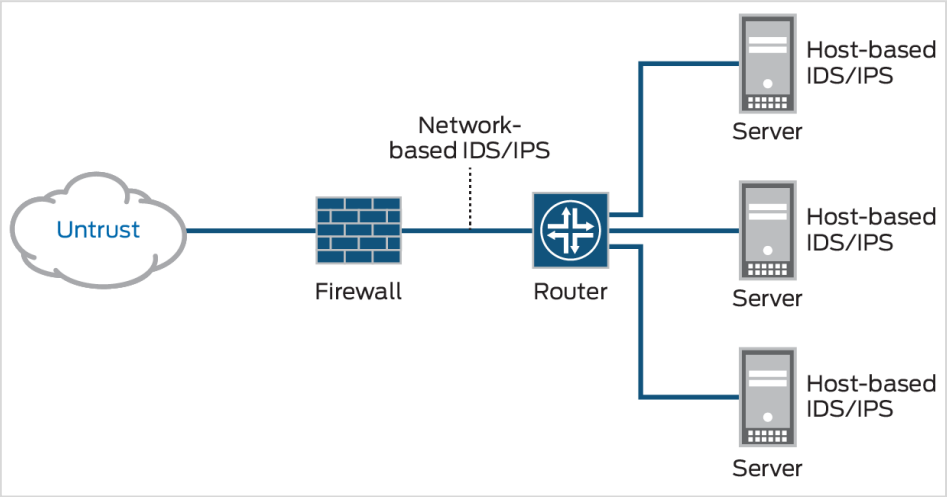
3. **Κρυπτογραφία (cryptography).** Η κρυπτογραφία στην ασφάλεια των δικτύων βασίζεται στη δημιουργία κωδικοποιημένων πληροφοριών με τη χρήση κατάλληλου αλγορίθμου ο οποίος δημιουργεί το κρυπτογράφημα [23]. Σε επίπεδο δικτύου, οι πληροφορίες μεταξύ των χρηστών κωδικοποιούνται κατά τέτοιο τρόπο ώστε η κρυπτογράφηση και η αποκρυπτογράφησή τους να είναι δυνατή με τη χρήση σχετικών κλειδιών [24]. Στην κλασσική ή αλλιώς συμμετρική κρυπτογραφία η αποκρυπτογράφηση θεωρείται εξαιρετικά εύκολη όταν είναι γνωστό το σχετικό κλειδί ενώ στην περίπτωση κρυπτογραφίας δημοσίου κλειδιού ή αλλιώς ασύμμετρη, το δημόσιο κλειδί μπορεί να είναι γνωστό χωρίς να εγκυμονούνται κίνδυνοι για τις πληροφορίες αρκεί το ιδιωτικό κλειδί να παραμένει κρυφό. [25].



Εικόνα 2.4: Διαδικασία κρυπτογραφίας Πηγή: [26]

4. **Συστήματα Ανίχνευσης & Αποτροπής Εισβολών (Intrusion Detection & Prevention Systems, IDPS).** Τα IDPS αποτελούν μηχανισμό ασφάλειας όπου ανιχνεύουν και ενεργούν σε περιπτώσεις μη εξουσιοδοτημένης πρόσβασης στα συστήματα του δικτύου, παρέχοντας σε πραγματικό χρόνο, παρακολούθηση της κίνησής του. Τα IDPSs μπορεί να αποτελούνται από υλικό (hardware), λογισμικό (software) ή συνδυασμό και των δύο [27]. Τα IDPSs είναι ιδιαίτερα αποτελεσματικά για την ασφάλεια των δικτύων αλλά παρουσιάζουν αυξημένο κόστος και απαιτήσεις λειτουργίας. Μια διάκριση των IDPS είναι σε Συστήματα Ανίχνευσης Εισβολής (Intrusion Detection Systems, IDS) και σε Συστήματα Αποτροπής Εισβολών (Intrusion Prevention Systems, IPS)[28]. Η συγκεκριμένη διάκριση βασίζεται στο ότι τα η IDS εντοπίζουν εισβολές και τις αναφέρουν ενώ τα IPS τις ανιχνεύουν και δεν επιτρέπουν την πρόσβαση μέσω αποκλεισμών. Ως εκ τούτου, τα IPS μπορούν να θεωρηθούν ως επέκταση των IDS. Ωστόσο, οι τεχνολογίες πλέον έχουν συγκλίνει σε τέτοιο βαθμό ώστε τα περισσότερα IDS συστήματα επιτελούν και λειτουργίες πρόληψης προστασίας και ανίχνευσης. Ο τρόπος λειτουργίας μεταξύ ανίχνευσης και πρόληψης μπορούν επιλεχθεί ανάλογα με τη ρύθμιση επιμέρους παραμέτρων. Η διαφορά ανάμεσα σε ένα

firewall και σε ένα σύστημα IDPS μπορεί να είναι δυσδιάκριτη ωστόσο τα τελευταία παρέχουν ένα υψηλότερο επίπεδο ασφάλειας σε τεχνικό επίπεδο [13].



Εικόνα 2.5: Σύστημα Ανίχνευσης & Αποτροπής Εισβολών (IDPS) Πηγή: [29]

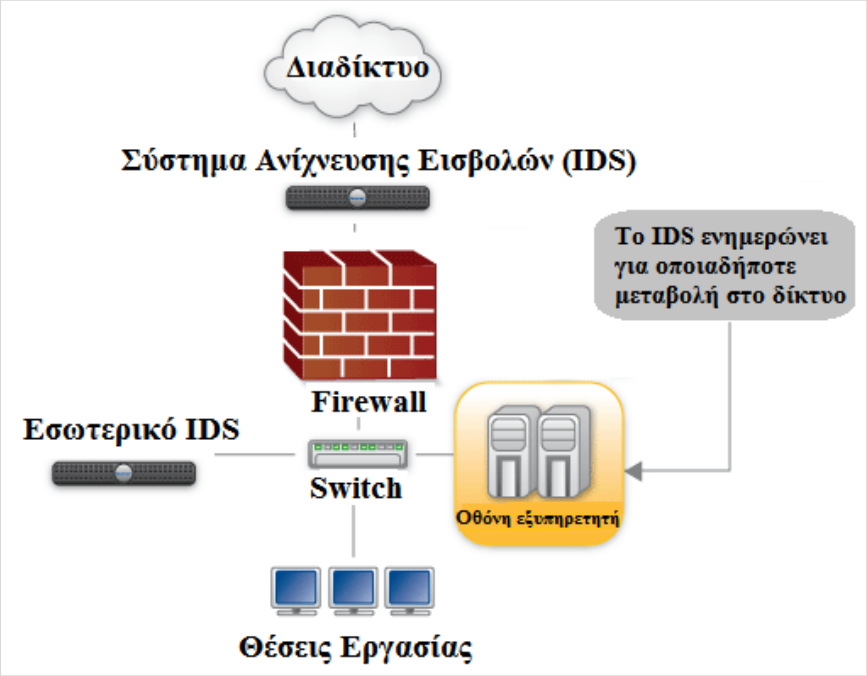
Κεφάλαιο 3

Συστήματα Ανίχνευσης Εισβολών (Intrusion Detection System, IDS)

3.1 Έννοια & Σχεδιασμός Συστημάτων Ανίχνευσης Εισβολών (Intrusion Detection System, IDS)

Η ανίχνευση εισβολών στα δίκτυα τις τελευταίες δεκαετίες έχει συγκεντρώσει το ενδιαφέρον της πληροφορικής καθώς έχουν αυξηθεί σημαντικά οι προσπάθειες κακόβουλων επιθέσεων. Ο εντοπισμός της κακόβουλης δραστηριότητας και των εισβολών είναι μια αρκετά πολύπλοκη διαδικασία που σε ορισμένες περιπτώσεις δεν στέφεται με επιτυχία δημιουργώντας προβλήματα στη λειτουργία των δικτύων [30]. Μέχρι πριν από μερικά χρόνια, τα διαθέσιμα συστήματα ανίχνευσης εισβολών δεν μπορούσαν να αντικρούσουν το σύνολο των κακόβουλων επιθέσεων. Οι εξελίξεις ωστόσο σε επίπεδο τεχνολογίας διαμόρφωσαν τις προϋποθέσεις δημιουργίας σύγχρονων συστημάτων όπως τα Συστήματα Ανίχνευσης Εισβολών (Intrusion Detection System, IDS) [31].

Τα Συστήματα Ανίχνευσης Εισβολών (Intrusion Detection System, IDS) επιτελούν τη διαδικασία ανίχνευσης και εντοπισμού μη εξουσιοδοτημένης ή ασυνήθιστης δραστηριότητας στο σύστημα. Επίσης, συμβάλλουν στην αξιολόγηση της δραστηριότητας ως ύποπτης ή μη στο εταιρικό δίκτυο. Τα IDS ελέγχοντας το δίκτυο είναι σε θέση να εντοπίζουν εισβολές και επιθέσεις αλλά και ανεπιθύμητη δραστηριότητα [32].



Εικόνα 3.1: Λειτουργία Συστήματος Ανίχνευσης (IDS) Πηγή: [33]

Ο στόχος ενός IDS είναι να παρέχει τις αναγκαίες ενδείξεις μιας πιθανής ή μιας πραγματικής επίθεσης. Τα IDS ενημερώνουν για μια επίθεση στο δίκτυο ή μια κακόβουλη εισβολή και δεν αναφέρουν τα τυχόν τρωτά σημεία του δικτύου [34]. Η διαφορά μεταξύ της ανίχνευσης της εισβολής και της τρωτότητας του δικτύου είναι ότι όταν πραγματοποιείται μια κακόβουλη επίθεση σε συγκεκριμένη χρονική στιγμή, το IDS είναι σε θέση να εντοπίσει την εισβολή αλλά όχι πάντα από ποιο σημείο του δικτύου προήλθε. Η ανίχνευση της εισβολής βοηθά στη μετέπειτα αντιμετώπισής της με την ευπάθεια του δικτύου να εξακολουθεί να υφίσταται.

Το σύστημα IDS είναι μια ολοκληρωμένη μέθοδος ανίχνευσης τυχόν επιθέσεων από ανάλυση και παρακολούθηση των δραστηριοτήτων του δικτύου. Μπορεί να έχει τη μορφή συσκευής ή να είναι εφαρμογή λογισμικού που παρακολουθεί το δίκτυο ή συστήματα για κακόβουλες παραβιάσεις [35]. Ένα σύστημα IDS παρακολουθεί την κίνηση του δικτύου για ύποπτη δραστηριότητα ενημερώνοντας σχετικά το διαχειριστή του συστήματος ή του δικτύου και στη συνέχεια φιλτράρεται για να αξιολογηθεί ως αληθής ή ψευδής.. Σε ορισμένες περιπτώσεις, το IDS μπορεί επίσης να ενεργήσει έναντι των κακόβουλων εισβολών με τη λήψη μέτρων, όπως η παρεμπόδιση της συγκεκριμένης IP να έχει πρόσβαση στο δίκτυο. Υπάρχει ένα ευρύ φάσμα IDS, που

κυμαίνεται από λογισμικό προστασίας από ιούς μέχρι συστημάτων που μπορούν να ελέγχουν τη δραστηριότητα ενός ολόκληρου δικτύου [36].

Ο σχεδιασμός της ανίχνευσης εισβολής αποτελείται από δύο βασικά βήματα: το πρώτο βήμα είναι η δημιουργία των αρχείων ελέγχου και το δεύτερο ο έλεγχος ώστε να εντοπιστούν οι τυχόν εισβολές. Το δίκτυο ελέγχεται σε τακτά χρονικά διαστήματα για τυχόν ύποπτη δραστηριότητα σε περιπτώσεις υπέρβασης καθορισμένων κριτηρίων [32]. Τα IDS σχεδιάζονται με τέτοιο τρόπο ώστε να παρακολουθούν όλη την εισερχόμενη και εξερχόμενη δραστηριότητα του δικτύου εντοπίζοντας τυχόν ύποπτη δραστηριότητα που μπορεί να υποδεικνύει κακόβουλη δραστηριότητα ή προσπάθεια υποβιβασμού της ασφάλειας. Θεωρούνται παθητικά συστήματα παρακολούθησης, δεδομένου ότι η κύρια λειτουργία τους είναι η ανίχνευση και προειδοποίηση για ύποπτη δραστηριότητα. Ο εντοπισμός των κακόβουλων προσπαθειών σε ένα σύστημα IDS γίνεται με την αξιοποίηση των κατάλληλων μηχανισμών. Ο όρος IDS καλύπτει στην πραγματικότητα μια μεγάλη ποικιλία συστημάτων τα οποία παράγουν το τελικό αποτέλεσμα της ανίχνευσης εισβολών [37].

3.2 Χαρακτηριστικά των IDS

Η επιλογή και εφαρμογή του συστήματος IDS βασίζεται στα ιδιαίτερα χαρακτηριστικά και στους στόχους που καλείται να επιτύχει. Αν και υπάρχουν κάποια κοινά χαρακτηριστικά στοιχεία μεταξύ των IDS, εντούτοις η διαφοροποίηση μεταξύ των επιθέσεων θέτει και συγκεκριμένα χαρακτηριστικά στοιχεία που θα πρέπει να διαθέτει ένα IDS. Στα IDS υπάρχουν επιμέρους τμήματα τα οποία αλληλοσυνεργάζονται για την αποτροπή μιας εισβολής. Στα τυπικά χαρακτηριστικά ενός συστήματος IDS περιλαμβάνονται τα εξής στοιχεία:

1. **Πρωτογενής συλλογή πληροφοριών.** Η αξιοποίηση συγκεκριμένων μηχανισμών επιτρέπει την ευκολότερη ανίχνευση τυχόν εισβολών και τη διαμόρφωση συγκεκριμένων προτύπων αντιμετώπισης μελλοντικών.
2. **Επεξεργασία πληροφοριών και δεδομένων.** Κάθε IDS διαθέτει υποσύστημα επεξεργασίας και αξιολόγησης των εισβολών κατευθύνοντας και επιλέγοντας τα μέτρα αντιμετώπισής τους [32].

3. **Ύπαρξη αντιμέτρων.** Η ανίχνευση των εισβολών ενεργοποιεί μηχανισμούς αντίμετρων και μέτρων αποτροπής ενώ ενημερώνεται και η σχετική βάση πληροφοριών [38].
4. **Αποθήκευση και ανάκληση πληροφοριών εισβολών.** Τα IDS διαθέτουν μηχανισμούς συλλογής και αποθήκευσης πληροφοριών και δεδομένων που σχετίζονται με προσπάθειες κακόβουλης εισβολής. Η ύπαρξη βάσεων δεδομένων ισχυροποιεί το σύστημα έναντι εισβολών.
5. **Διασύνδεση με υποσυστήματα.** Ορισμένα IDS μπορούν να διασυνδεθούν υπό την προϋπόθεση ότι αποτελούν ξεχωριστό τμήμα ενός συνολικότερου συστήματος [39].

Τα επιμέρους χαρακτηριστικά για κάθε IDS δημιουργούν μια συνολική εικόνα για το σύστημα ενώ η επάρκεια και αξιοπιστία του βασίζεται στην ύπαρξη κατ' ελάχιστο συγκεκριμένων χαρακτηριστικών. Ειδικότερα, τα IDS επιτυγχάνουν μέγιστη αξιοπιστία και αποτελεσματικότητα στη βάση χαρακτηριστικών όπως μεγάλο εύρος ανιχνεύσεων κακόβουλων εισβολών, ανίχνευση ή αποτροπή κακόβουλων επιθέσεων σε πραγματικό χρόνο, διάκριση πραγματικών ή εικονικών επιθέσεων (false positive/negative alarms), αυτόματη διαχείριση εισβολής και επαναφοράς σε κανονική λειτουργία μετά από μια επίθεση χωρίς παρεμβάσεις και αξιοποίηση των λιγότερων δυνατών πόρων του συστήματος. Τα IDS στην πλειοψηφία τους διαθέτουν τα χαρακτηριστικά που προαναφέρθηκαν ενώ η αξιολόγηση της αποδοτικότητας και η αξιοπιστία τους βασίζεται στο κατά πόσο είναι σε θέση να συνδυάσουν τα συγκεκριμένα χαρακτηριστικά [40].

3.3. Προτυποποίηση IDS

Η προτυποποίηση των ανιχνεύσεων στα IDS είναι ένα εξειδικευμένο πρόβλημα το οποίο απαιτεί την αξιολόγηση πολλών επιμέρους παραγόντων που σχετίζονται με τις αναζητήσεις. Υπάρχουν μια σειρά ερωτημάτων στα οποία θα πρέπει να δοθούν απαντήσεις όπως η επιλογή του αλγόριθμου ανιχνεύσεων, ο βαθμός ασφάλειας και αντίδρασης στις μεταβολές δραστηριότητας, η συχνότητα ανιχνεύσεων κ.ά. Σε πραγματικό χρόνο , η αναγκαιότητα ανίχνευσης των εισβολών απαιτεί την ανάπτυξη ενός μοντέλου "μηχανής αναζήτησης" που ταυτόχρονα μπορεί να συμβαδίσει με τις ταχύτητες των σύγχρονων δικτύων. Η διαμόρφωση patterns στα IDS επιτρέπει όχι μόνο την αντιμετώπιση των εισβολών σε συγκεκριμένη χρονική περίοδο αλλά και τη διαμόρφωση προτύπων αντιμετώπισης μελλοντικών εισβολών. Η προτυποποίηση μέσω εκτενούς παρακολούθησης και αξιολόγησης της δραστηριότητας επιτρέπει στα IDS να διαμορφώσουν ένα ισχυρό πλαίσιο προστασίας των συστημάτων [41].

Υπάρχουν δύο τύποι δεδομένων που χρησιμοποιούνται: τα πρότυπα (προκαθορισμένα και στατικά) που επιτρέπουν την προεπεξεργασία των δεδομένων και το κείμενο αναζήτησης που είναι δυναμικό και μεταβαλλόμενο για κάθε IDS [41]. Στα IDS χρησιμοποιείται πρότυπο αλγορίθμου που συμβάλλει στην ανίχνευση των εισβολών. Κατά τη διάρκεια της διαδικασίας εντοπισμού, η αποτελεσματικότητα του αλγορίθμου καθορίζει και την απόδοση του συστήματος IDS. Η χαμηλή αποδοτικότητα σχετίζεται με βασικές ελλείψεις στο πρότυπο του αλγόριθμου. Το μέγεθος των προτύπων που χρησιμοποιούνται σε μια ανίχνευση εισβολής μπορεί να επηρεάσει σημαντικά τα χαρακτηριστικά απόδοσης του αλγορίθμου.Ωστόσο, υπάρχουν αλγόριθμοι οι οποίοι δεν επηρεάζονται από το μέγεθος του προτύπου παρακάμπτοντας χαρακτηριστικά και τμήματα επιταχύνοντας σημαντικά την απόδοση των IDS. Τα πρότυπα αναζήτησης στην ανίχνευση εισβολών αντιπροσωπεύουν τμήματα των γνωστών τεχνικών επίθεσης και μπορούν να ποικίλουν σε μέγεθος από 1 έως 30 ή περισσότερους χαρακτήρες, αλλά είναι συνήθως μικρά [42].

Στην προτυποποίηση των IDS ιδιαίτερα σημαντικές είναι οι δυνατότητες αποθήκευσης των δεδομένων και η προσωρινή μνήμη. Πρότυπα που απαιτούν μικρό μέγεθος μνήμης αποθήκευσης επωφελούνται περισσότερο των δυνατοτήτων των IDS έναντι προτύπων όπου αυξημένα επίπεδα μνήμης μπορούν να οδηγήσουν σε αστοχίες ανίχνευσης εισβολών. Το μέγεθος του προτύπου επηρεάζει συνήθως τις επιδόσεις του

IDS επειδή το πρόβλημα αναζήτησης σχετίζεται με την προσωρινή αποθήκευση μνήμης του επεξεργαστή [43].

Το μέγεθος του αλφαβήτου που χρησιμοποιούνται στα IDS ορίζεται στο 1 byte. Το μέγεθος του αλφαβήτου έχει σημαντικό αντίκτυπο στον αλγόριθμο ανίχνευσης καθιστώντας τον πιο αποτελεσματικό και πιο γρήγορο. Το μέγεθος του κειμένου αναζήτησης σε συστήματα IDS είναι συνήθως λιγότερο από μερικές χιλιάδες bytes [42].

3.4 Πλεονεκτήματα & Μειονεκτήματα IDS

Η ύπαρξη IDS σε οποιοδήποτε δίκτυο σε συνδυασμό με firewall αποτελεί τη βάση ανίχνευσης και αποτροπής εισβολών στο δίκτυο. Τα συστήματα IDS ανάλογα με τον τύπο τους (Network Intrusion Detection System – NIDS και Host Intrusion Detection System, HIDS), μπορούν να προσφέρουν σε διαφορετικά επίπεδα δημιουργώντας ένα συγκεκριμένο πλαίσιο λειτουργίας. Ειδικότερα, για τα IDS που βασίζονται και λειτουργούν στο δίκτυο, σημειώνονται τα ακόλουθα πλεονεκτήματα:

1. **Χαμηλό επίπεδο κόστους χρήσης.** Τα NIDS μπορούν να χρησιμοποιηθούν σε οποιοδήποτε τμήμα του δικτύου μη απαιτώντας την ύπαρξη σχετικού λογισμικού μειώνοντας τα πάγια έξοδα διαχείρισης [44].
2. **Είναι ευκολότερα στη χρήση.** Τα NIDS είναι πιο εύκολο να αξιοποιηθούν χωρίς να επηρεάζουν υφιστάμενα συστήματα ή υποδομές [45].
3. **Ευκολότερος εντοπισμός επιθέσεων στο δίκτυο.** Τα NIDS ανιχνεύουν ευκολότερα τις επιθέσεις στο δίκτυο έναντι των IDS που εγκαθίστανται σε ένα συγκεκριμένο σύστημα [46].
4. **Ανίχνευση εισβολών σε πραγματικό χρόνο.** Οι κακόβουλες επιθέσεις στα NIDS εντοπίζονται σε πραγματικό χρόνο και ως εκ τούτου, ο εισβολέας δεν μπορεί να αποκρύψει οποιοδήποτε στοιχείο σχετίζεται με την επίθεση [45].
5. **Ανίχνευση αποτυχημένων επιθέσεων.** Ένας αισθητήρας IDS με βάση το δίκτυο που αναπτύσσεται εκτός του τείχους προστασίας (firewall), μπορεί να εντοπίσει κακόβουλες επιθέσεις ακόμα και πριν από αυτό προειδοποιώντας για τις συγκεκριμένες επιθέσεις.

Τα NIDS αποτελούν ένα ισχυρό μηχανισμό προστασίας έναντι των κακόβουλων επιθέσεων, ωστόσο παρουσιάζουν μειονεκτήματα και περιορισμούς που θα μπορούσαν

να αποδειχτούν σημαντικά στην ομαλή λειτουργία του δικτύου. Συγκεκριμένα, μειονεκτήματα των NIDS αποτελούν τα εξής:

1. **Δυσκολία ελέγχου μεγάλου όγκου δεδομένων.** Στο δίκτυο η ύπαρξη τεράστιου όγκου δεδομένων απαιτούν από τα NIDS αρκετό χρόνο αλλά και δυνατότητες αξιολόγησης πληθώρας δεδομένων καθιστώντας τον έλεγχο αρκετά πολύπλοκη και χρονοβόρα διαδικασία [47].
2. **Ευάλωτα σε DoS.** Σε ορισμένα NIDS εντοπίζονται τρωτά σημεία που καθιστούν εύκολη την παραβίασή τους [45].
3. **Αδυναμία ελέγχου κρυπτογραφημένων πακέτων.** Τα NIDS εμφανίζουν σημαντικές δυσκολίες ελέγχου και αξιολόγησης πακέτων δεδομένων που έχουν κρυπτογραφηθεί. Η κρυπτογράφηση δεδομένων δυσκολεύει σημαντικά τον έλεγχό τους από τα NIDS [46].
4. **Δυσκολίες εγκατάστασης.** Τις περισσότερες φορές ο εντοπισμός εκείνου του σημείου που θα πρέπει να τοποθετηθεί το NIDS είναι μια αρκετά πολύπλοκη διαδικασία χωρίς πάντοτε να στέφεται με επιτυχία μειώνοντας την αποδοτικότητα του συστήματος και καθιστώντας το ευάλωτο [45]..

Μια σημαντική κατηγορία συστημάτων IDS είναι τα Host Intrusion Detection System (HIDS) τα οποία εγκαθίστανται σε εκάστοτε υπολογιστή του δικτύου. Η χρήση των HIDS συνοδεύεται από πλεονεκτήματα και μειονεκτήματα. Συγκεκριμένα, μεταξύ των πλεονεκτημάτων των HIDS περιλαμβάνονται:

1. **Επαλήθευση επιτυχίας ή αποτυχίας μιας κακόβουλης επίθεσης.** Τα HIDS χρησιμοποιώντας αρχεία καταγραφής προηγούμενων επιθέσεων, μπορούν να αξιολογήσουν την επιτυχία ή αποτυχία μιας επίθεσης [48].
2. **Ανίχνευση επιθέσεων που τα NIDS δεν μπορεί να εντοπίσουν.** Τα HIDS μπορούν να ανιχνεύουν επιθέσεις που τα δικτυακά IDS δεν είναι σε θέση να εντοπίσουν προλαμβάνοντας τυχόν βλάβες του συστήματος [47].
3. **Χαμηλό κόστος.** Οι αισθητήρες που χρησιμοποιούν παρουσιάζουν χαμηλότερο κόστος έναντι των αντίστοιχων των NIDS [49].
4. **Ανίχνευση τροποποιήσεων αρχείων.** Τα HIDS παρέχουν άμεση πληροφόρηση για τυχόν προσπάθειες αλλοίωσης ή τροποποίησης αρχείων δεδομένων [47].
5. **Δυνατότητες ελέγχου κρυπτογραφημένων πακέτων.** Τα HIDS έναντι των NIDS δύναται να αξιολογούν και να ελέγχουν και κρυπτογραφημένα δεδομένα.

6. **Ανίχνευση εισβολών σε πραγματικό χρόνο.** Αν και σε ορισμένες περιπτώσεις η άμεση ανίχνευση δεν είναι δυνατή, εντούτοις με κατάλληλη ρύθμιση τα HIDS μπορούν να ανιχνεύουν εισβολές σε πραγματικό χρόνο [50].

Η χρήση των HIDS δεν συνεπάγεται και την εξασφάλιση της ιδανικής λύσης προστασίας έναντι των κακόβουλων επιθέσεων. Αντίστοιχα με τα NIDS τα HIDS παρουσιάζουν μια σειρά μειονεκτημάτων που τα καθιστούν ευάλωτα. Ειδικότερα, μειονεκτήματα των HIDS είναι τα εξής:

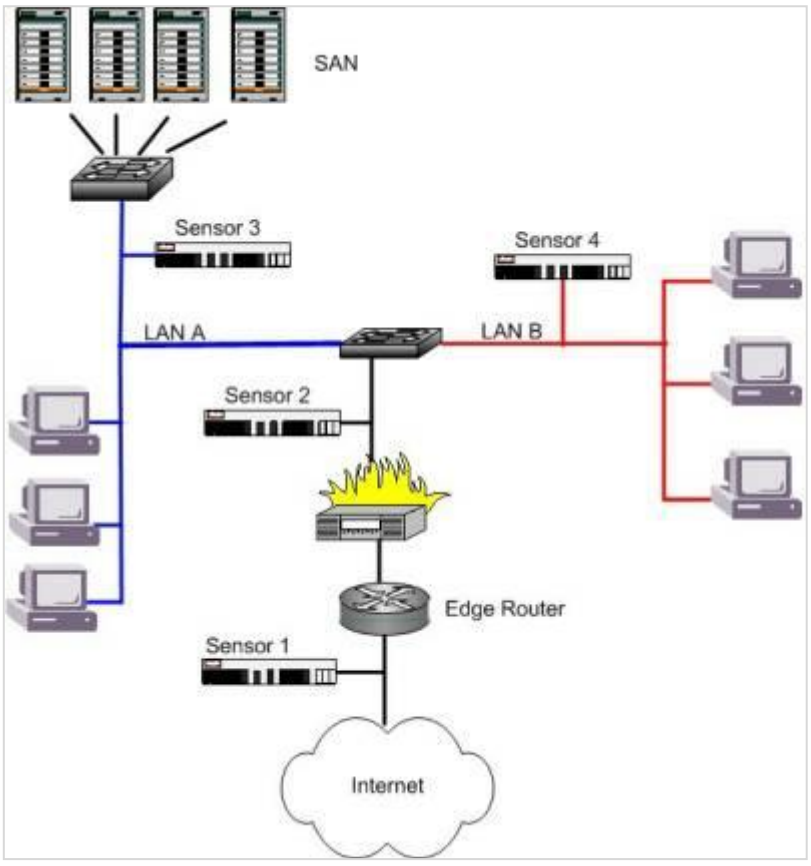
1. **Ανίχνευση επιθέσεων με τη χρήση έμμεσων πηγών πληροφόρησης.** Η αξιοποίηση αρχείων καταγραφής προηγούμενων επιθέσεων πολλές φορές απαιτεί χρόνο που μπορεί να είναι πολύτιμος στην αντιμετώπιση της εισβολής.
2. **Αδυναμία ελέγχου απομακρυσμένων συστημάτων.** Η εγκατάσταση σε συγκεκριμένο υπολογιστή του δικτύου δεν επιτρέπει την ανίχνευση εισβολών στο σύνολο των υπολογιστών του δικτύου [51].
7. **Αδυναμία προστασίας μεγάλων δικτύων.** Ο χαρακτήρας της τοπικής προστασίας ενός υπολογιστή από κακόβουλες επιθέσεις, δεν επιτρέπει την αξιοποίηση των NIDS σε δίκτυα μεγάλου μεγέθους[47].
3. **Δυσκολία διαχείρισης.** Τα HIDS παρουσιάζουν δυσκολίες διαχείρισης κυρίως λόγω της ανάγκης ρύθμισης των αναγκών του εκάστοτε υπολογιστή του δικτύου [32,49].
4. **Απενεργοποίηση από ορισμένες επιθέσεις.** Τα HIDS μπορούν να απενεργοποιηθούν από ορισμένες κακόβουλες επιθέσεις που δεν μπορούν να διαχειριστούν [49].

3.5 Αρχιτεκτονική συστήματος IDS

Η αρχιτεκτονική των IDS συναρτάται άμεσα από τις λειτουργίες και τους ελέγχους που θα επιτελούνται. Ειδικότερα, η αξιοποίηση των IDS σε επίπεδο υπολογιστή ή δικτύου καθορίζει και την αρχιτεκτονική στην οποία οικοδομείται ολόκληρο το σύστημα ανίχνευσης των εισβολών [52]. Η χρήση συστημάτων στον ίδιο υπολογιστή ή η επιμέρους διάκρισης των συστημάτων διαμορφώνει και το πλαίσιο λειτουργίας των IDS [30]. Η «συστέγαση» ή όχι των συστημάτων ανίχνευσης εισβολής επηρεάζει την αρχιτεκτονική των IDS καθώς αφενός μεν μέσω της «συστέγασης» μπορεί νε επιτευχθεί μείωση του κόστους αφετέρου σε περίπτωση επιτυχούς κακόβουλης

εισβολής απενεργοποιείται συνολικά το IDS γεγονός που δεν μπορεί να συμβεί σε περιπτώσεις τοποθέτησής του σε ξεχωριστό σύστημα [53].

Ανάλογα με το μηχανισμό συλλογής δεδομένων ο οποίος επηρεάζει και την αρχιτεκτονική τους, τα IDS διακρίνονται σε Network Intrusion Detection System (NIDS) και Host Intrusion Detection System (HIDS). Η συνύπαρξη συστημάτων εντοπισμού των εισβολών η μεμονωμένων συστημάτων αλλά και τα σημεία ελέγχου των επιθέσεων επηρεάζουν την αρχιτεκτονική των IDS [54].



Εικόνα 3.2: Αρχιτεκτονική δομή NIDS με αισθητήρες Πηγή: [54]

Στα NIDS, η αρχιτεκτονική δομή τους περιλαμβάνει τα ακόλουθα:

1. **Αισθητήρας (sensor).** Επιτελούν το ρόλο της παρακολούθησης της δραστηριότητας σε ένα ή περισσότερα τμήματα του δικτύου. Ο έλεγχος και η αξιολόγηση πραγματοποιείται στο σύνολο των δεδομένων ανεξάρτητα από την κατεύθυνσή τους. Σε ένα IDS μπορούν να τοποθετηθούν περισσότεροι του ενός αισθητήρες ανάλογα με τις ανάγκες προστασίας ενώ διακρίνονται σε αισθητήρες υλικού και λογισμικού και σε αισθητήρες λογισμικού [53,55].

2. **Διακομιστής διαχείρισης (management servers).** Αποτελούν κεντρικό σύστημα συλλογής των πληροφοριών των αισθητήρων [55]. Έχουν δυνατότητα ανάλυσης και αξιολόγησης των συλλεχθέντων πληροφοριών από τους αισθητήρες εντοπίζοντας κακόβουλες επιθέσεις που δεν είναι σε θέση ο καθένας ξεχωριστά να εντοπίσει. Ο αριθμός τους (ένας ή περισσότεροι) συναρτώνται από τις ανάγκες προστασίας.
3. **Διακομιστής βάσεων δεδομένων (database servers).** Αποτελεί το χώρο αποθήκευσης των πληροφοριών που συγκεντρώνονται από τους αισθητήρες και το διακομιστή διαχείρισης [52,55].
4. **Κονσόλα (console).** Αποτελεί πρόγραμμα το οποίο επιτρέπει στο χρήστη τη διαχείριση του IDS και την αξιολόγηση της δραστηριότητας που συλλέγεται.

Στα HIDS, η αρχιτεκτονική δομή είναι απλούστερη με συγκεκριμένο λογισμικό να επιτελεί το ρόλο ανίχνευσης των εισβολών. Ο πράκτορας (agent), όπως καλείται το συγκεκριμένο λογισμικό, θέτει το σύστημα υπό παρακολούθηση αξιοποιώντας το δίκτυο [55]. Η προστασία των δεδομένων επιτυγχάνεται μέσω της κρυπτογράφησης ενώ στην περίπτωση που ο πράκτορας έχει τη μορφή συσκευής (hardware), η τοποθέτησή τους γίνεται πριν την είσοδο των πληροφοριών στον υπολογιστή. Τα στοιχεία που συλλέγει ο πράκτορας μεταβιβάζονται στο διακομιστή διαχείρισης ο οποίος αποθηκεύει τις πληροφορίες στην βάση δεδομένων [55].

3.6 Ταξινόμηση IDS

Η διαθέσιμη τεχνολογία έχει επιτρέψει τη δημιουργία πολλών τύπων IDS. Ανάλογα με την περιοχή στην οποία λειτουργούν τα IDS και το είδος των εισβολών που μπορούν να ανιχνεύσουν, ταξινομούνται σε επιμέρους κατηγορίες [57,58]. Ειδικότερα, τα IDS διακρίνονται στα εξής:

1. **Μεμονωμένα IDS (Host Intrusion Detection System).** Τα HIDS παρακολουθούν και συλλέγουν πληροφορίες εισβολών οι οποίες αφορούν ένα μεμονωμένο υπολογιστή [55]. Τα HIDS ελέγχουν τα αρχεία καταγραφών, τις διεργασίες που επιτελεί ο υπολογιστής καθώς και σημαντικούς πόρους του (π.χ. hard disk, memory, network). Σε περίπτωση μη εξουσιοδοτημένης χρήσης ή εντοπισμού κακόβουλης δραστηριότητας προκύπτει ενημέρωση για το

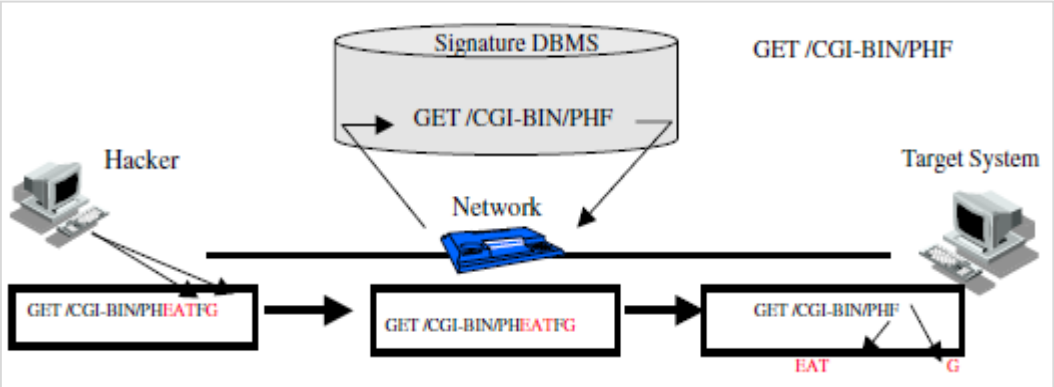
διαχειριστή μέσω σχετικών μηνυμάτων [27]. Η λειτουργία των συγκεκριμένων συστημάτων βασίζεται στην ανίχνευση της ύποπτης δραστηριότητας μετά την σχετική επίθεση [40].

2. **Δικτυακά IDS (Network Intrusion Detection System)**. Το NIDS ελέγχει την κίνηση του δικτύου σε συγκεκριμένα τμήματα μέσω αισθητήρων και στη συνέχεια αναλύει τις δραστηριότητες μέσω πρωτοκόλλων για την αναγνώριση ύποπτων περιστατικών [27]. Τα NIDS ανιχνεύουν την προσπάθεια εισβολής πριν πραγματοποιηθεί και βασίζονται είτε σε προϋπάρχουσες επιθέσεις λόγω κακής χρήσης (misuse detection) με δυνατότητα ειδοποίησης και τερματισμού της σύνδεσης ή μέσω εντοπισμού ανώμαλης δραστηριότητας (anomaly detection) συγκρίνοντας δεδομένα λειτουργίας του συγκεκριμένου τμήματος του δικτύου [59,60].
3. **Ασύρματα IDS (Wireless Intrusion Detection System).** Τα WIDS έχουν παρόμοια λειτουργία με τα NIDS με τη διαφορά ότι ελέγχουν την κίνηση στο ασύρματο δίκτυο όπως ad hoc δίκτυα, ασύρματα δίκτυα αισθητήρων, ασύρματα δίκτυα πλέγματος [32,55].
4. **Υβριδικά IDS (Hybrid Intrusion Detection System).** Αποτελούν συνδυασμό συστημάτων ανίχνευσης εισβολών υιοθετώντας πολλαπλές τεχνολογίες και μπορούν να εκπληρώσουν το στόχο μιας πιο πλήρους και ακριβής ανίχνευσης. Τα υβριδικά IDS έχουν τη δυνατότητα ελέγχου της δραστηριότητας του δικτύου τόσο μεμονωμένα στον υπολογιστή όσο και συνολικά. Η λειτουργία τους μοιάζει με εκείνη των NIDS συγκρίνοντας τις διακινούμενες πληροφορίες με καταχωρημένες υπογραφές επιθέσεων με βασική διαφορά το γεγονός ότι η συγκεκριμένη διαδικασία πραγματοποιείται μόνο για το συγκεκριμένο σημείο του δικτύου[55].
5. **Ανάλυσης Συμπεριφοράς Δικτύου (Network Behavior Analysis).** Το σύστημα NBA επιθεωρεί την κίνηση στο δίκτυο και αναγνωρίζει τις επιθέσεις σε περιπτώσεις μη αναμενόμενης δραστηριότητας. [55].

3.7 Κατηγοριοποίηση Επιθέσεων Ενάντια Στα IDS

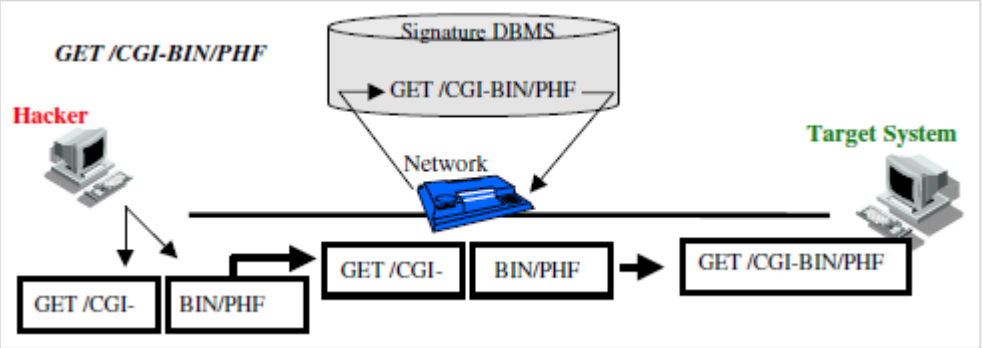
Ένας εισβολέας με στόχο ένα δίκτυο IDS έχει πολλούς πιθανούς στόχους. Ο πιο συνήθης είναι η εισβολή σε ένα σύστημα ή δίκτυο χωρίς η συγκεκριμένη δραστηριότητα να γίνει αντιληπτή από το IDS. Οι επιθέσεις στα IDS ταξινομούνται σε επιμέρους κατηγορίες ανάλογα με το σκοπό μεταξύ των οποίων περιλαμβάνεται η υπονόμευση του δικτύου, η μείωση των επιπέδων ασφάλειας, η απενεργοποίηση του συστήματος κλπ. Ειδικότερα, οι επιθέσεις ενάντια στα IDS κατηγοριοποιούνται ως εξής:

1. **Επίθεση εισαγωγής (insertion).** Η συγκεκριμένη επίθεση συμβαίνει κάθε φορά που ένα IDS δέχεται ένα πακέτο που δεν είναι αποδεκτό από τον ξενιστή. Η εισαγωγή του εισβολέα στοχεύει σε λανθασμένες διαβιβάσεις ειδοποιήσεων του IDS δημιουργώντας μόνιμη «σύγχυση» στο σύστημα ή στο δίκτυο. Τα IDS ενώ σε φυσιολογική λειτουργία εντοπίζουν τις επιθέσεις στην κίνηση του δικτύου με τη χρήση αλγορίθμων, μια επίθεση εισαγωγής δημιουργεί παρεμβολές με σκοπό την έκδοση λανθασμένων ειδοποιήσεων [32].



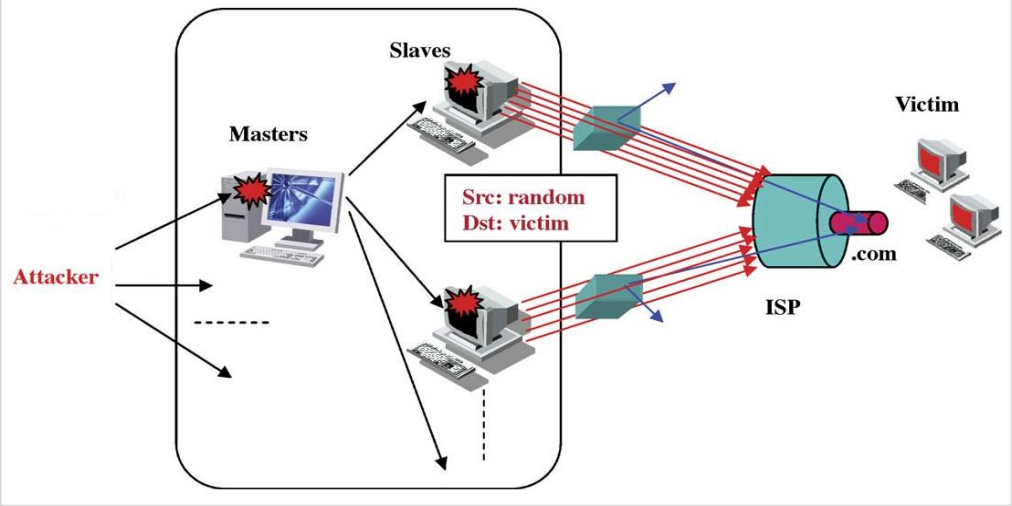
Εικόνα 3.3: Επίθεση εισαγωγής Πηγή: [32]

2. **Επίθεση υπεκφυγής (evasion).** Μια επίθεση υπεκφυγής συμβαίνει κάθε φορά που ένα IDS απορρίπτει ένα πακέτο που είναι αποδεκτό από τον ξενιστή [61]. Ουσιαστικά, ο εισβολέας επιτυγχάνει να μειώνει την ασφάλεια του IDS σε τέτοιο βαθμό ώστε ο εισβολέας να τροποποιεί δεδομένα χωρίς το σύστημα να μπορεί να το αντιληφθεί [62,63].



Εικόνα 3.4: Επίθεση υπεκφυγής Πηγή: [32]

3. **Πολύπλοκες επιθέσεις (complex).** Στις πολύπλοκες επιθέσεις δεν μπορεί να εντοπιστεί με ακρίβεια ο σκοπός καθώς μπορεί να χρησιμοποιηθούν τεχνικές εισαγωγής ή υπεκφυγής του συστήματος. Ο εισβολέας μέσω διαφορετικών τεχνικών επίθεσης αποσκοπεί να θέσει το IDS στα όρια αντοχών του επωφελούμενος από τις αδυναμίες του εκάστοτε συστήματος [32].
4. **Επιθέσεις υπερδιέγερσης (overstimulation).** Ο εισβολέας αποσκοπεί στη δημιουργία πολυάριθμων ψευδών ειδοποιήσεων στο IDS βρίσκοντας περιθώρια εισβολής στο δίκτυο ή στο σύστημα [64,65,66].
5. **Επιθέσεις μόλυνσης (poisoning).** Με τις επιθέσεις μόλυνσης στοχεύεται η εισαγωγή δεδομένων που θα αλλοιώσουν τις δυνατότητες ανίχνευσης απειλών για το IDS. Ο στόχος είναι να παραπλανηθεί ο αλγόριθμος του συστήματος επηρεάζοντας αρνητικά και την απόδοση του IDS (π.χ. χαμηλότερη ακρίβεια ανίχνευσης). Οι επιθέσεις μόλυνσης αποτελούν μια νέα και περίπλοκη και έχουν αποκτήσει μεγάλο ενδιαφέρον αναφορικά με την ασφάλεια των IDS [67,68).
6. **Επιθέσεις άρνησης εξυπηρέτησης (Denial Of Service, DoS).** Η επίθεση DoS στοχεύει στην αναστολή ή υπολειτουργία του IDS μέσω της δημιουργίας δραστηριότητας υπερφόρτωσης του αισθητήρα. Οι επιθέσεις DoS χρησιμοποιούνται υποστηρικτικά των επιθέσεων υπεκφυγής του συστήματος [69,70].



Εικόνα 3.5: Επίθεση DoS Πηγή: [32]

3.8 Πρωτόκολλο Επικοινωνίας IDS

Η επικοινωνία στο δίκτυο βασίζεται στην ύπαρξη συγκεκριμένων πρωτοκόλλων τα οποία συμβάλλουν στην επίτευξη ασφάλειας. Διαχρονικά, έχουν πραγματοποιηθεί προσπάθειες δημιουργίας και καθιέρωσης ενός πρωτοκόλλου επικοινωνίας για τα IDS. Προς αυτήν την κατεύθυνση κινήθηκαν και οι προσπάθειες ομάδων εργασίας του Internet Engineering Task Force (IETF) αρχικά με τη δημιουργία Πρωτοκόλλου Ειδοποίησης Εισβολής (Intrusion Alert Protocol, IAP)[71,72]. Ο σχεδιασμός του IAP έγινε με βάση το πρωτόκολλο HTTP [73] ενώ χρησιμοποιεί Transport Layer Security (TLS) για να εξασφαλιστεί η εμπιστευτικότητα, η ακεραιότητα και ο αμοιβαίος έλεγχος ταυτότητας. Η ανάπτυξη του IAP παρά τις πολλές αναθεωρήσεις των προδιαγραφών παρουσίασε αρκετές ατέλειες μεταξύ των οποίων η δυνατότητα παρεμβολών γεγονός ανεπιθύμητο από την άποψη της ασφάλειας [74].

Τα σύγχρονα IDS θα πρέπει να βασίζονται σε πρωτόκολλα εισβολών ενισχύοντας τη χρήση υπαρχόντων πρωτοκόλλων (π.χ. HTTP, TCP), προκειμένου να αποφεύγονται εισβολές από κακόβουλη δραστηριότητα. Η δημιουργία του Blocks Extensible Exchange Protocol (BEEP), που αποτελεί ένα ευρύτερο πλαίσιο για πρωτόκολλα ασύγχρονης επικοινωνίας με σύνδεση, οδήγησε σε ασφαλέστερες συνθήκες μεταφοράς και διασφάλισης των δεδομένων [75]. Το BEEP αποτέλεσε και τη βάση δημιουργίας μετέπειτα πρωτοκόλλων για τα IDS με βασικό στόχο την επίτευξη ασφάλειας στο

δίκτυο. Ουσιαστικά, η δημιουργία ενός ενιαίου πρωτοκόλλου επικοινωνίας για τα IDS το οποίο καλείται Intrusion Detection Exchange Protocol (IDXP) παρέχει πιστοποίηση στην επικοινωνία, εξασφαλίζοντας ακεραιότητα στα μεταφερόμενα δεδομένα και διασφαλίζοντας ταυτόχρονα την προστασία από τυχόν κακόβουλες παρεμβάσεις ή επιθέσεις [76].

Το πρωτόκολλο IDXP βασίζεται στην ύπαρξη συνδεδεμένου πρωτοκόλλου όπως για παράδειγμα το TCP. Η δομική οργάνωση του IDXP βασίζεται στο πλαίσιο λειτουργίας του Blocks Extensible Exchange Protocol (BEEP) εντοπίζοντας σημαντικές ομοιότητες [76]. Το πλεονέκτημα του IDXP εντοπίζεται στη δυνατότητα ανταλλαγής μηνυμάτων μεταξύ συστημάτων IDS υπό μορφή απλού κειμένου, δυαδικών δεδομένων ή μηνυμάτων μορφής IDMEF (Intrusion Detection Message Exchange Format). Η εγκατάσταση του πρωτοκόλλου IDXP στηρίζεται στη διαμόρφωση επικοινωνίας υπό το πλαίσιο ασφάλειας του BEEP και αφού ολοκληρωθούν οι διαδικασίες ταυτοποίησης και πιστοποίησης. Μεταξύ των συστημάτων που επικοινωνούν αξιοποιώντας τις δυνατότητες του Intrusion Detection Exchange Protocol (IDXP) δεν υπάρχει περιορισμός ως προς τα proxies τα οποία επιτελούν διαχειριστικό και όχι μόνο ρόλο. Η έναρξη επικοινωνίας σηματοδοτείται από την ολοκλήρωση του σχετικού πλαισίου βάσει του BEEP [71].

3.9 Μηχανισμοί Λειτουργίας IDS

Ο μηχανισμός λειτουργίας των IDS βασίζεται στην ύπαρξη σχετικών πρωτοκόλλων και υπογραφών τα οποία διαμορφώνουν και το πλαίσιο των ανιχνεύσεων. Μεταξύ των υπογραφών και των αντίστοιχων μηχανισμών ανίχνευσης των κακόβουλων εισβολών συγκαταλέγονται:

1. Σύνδεση σε «ύποπτη» διεύθυνση IP (Internet Protocol Address) η οποία εντοπίζεται στη βάση δεδομένων της [55].
2. Ύποπτο Πρωτόκολλο Ελέγχου Μεταφοράς (Transmission Control Protocol, TCP) [32].
3. E –mail που συνοδεύεται από συννημένο κακόβουλο αρχείο (worm, virus).
4. Απόπειρα DoS πολλαπλής εκτέλεσης εντολών ώστε να πληγεί η λειτουργία του IDS [37].

5. Επίθεση εισβολής σε FTP server για διαχείριση αρχείων χωρίς την αναγκαία ταυτοποίηση του χρήστη [77].

Οι υπογραφές στο μηχανισμό λειτουργίας των IDS είναι ιδιαίτερα καθοριστικές ώστε να επιτυγχάνονται οι αναγκαίες ταυτοποιήσεις. Ανάλογα με τη λειτουργία τους στα IDS οι υπογραφές κατηγοριοποιούνται από πολύ απλές μέχρι κρίσιμης σημασίας και σπουδαιότητας στη βάση απλών ελέγχων ή καθορισμού της σύνδεσης με επιμέρους πρωτόκολλα [27]. Ο σκοπός μιας υπογραφής ανίχνευσης εισβολής διαφοροποιείται ανάλογα με το στόχο που έχει τεθεί στο σύστημα IDS. Ουσιαστικά, οι υπογραφές ως μηχανισμός λειτουργίας των IDS αποσκοπούν στην ενημέρωση για τυχόν απόπειρες εισβολής. Οι υπογραφές ανάλογα με τη λειτουργία τους μπορούν να ενημερώνουν για την ευπάθεια του συστήματος ή για περιπτώσεις όπου εισβολείς επιθυμούν να βλάψουν το σύστημα ενώ άλλες υπογραφές μπορούν να αναφέρουν απλά μια ασυνήθιστη δραστηριότητα χωρίς να προσδιορίζεται μια συγκεκριμένη επίθεση [78].

3.10 Τεχνικές Ανίχνευσης Εισβολών στα IDS

Οι τεχνικές ανίχνευσης εισβολής αποτελούν ουσιαστικά τον τρόπο με τον οποίο ανιχνεύεται η κακόβουλη δραστηριότητα στα IDS ενώ διακρίνονται στις εξής κατηγορίες: ανίχνευσης ανωμαλιών (anomaly detection), ανίχνευσης κακής χρήσης (misuse detection), υβριδικές (hybrid detection) ανίχνευσης ανωμαλιών πρωτοκόλλων (protocol anomaly detection). Ειδικότερα, ως προς τις τεχνικές ανίχνευσης εισβολών στα IDS σημειώνονται τα εξής:

1. **Τεχνική ανίχνευσης ανωμαλιών (anomaly detection).** Η συγκεκριμένη τεχνική βασίζεται στην αξιολόγηση αποθηκευμένων χαρακτηριστικών της συνήθους συμπεριφοράς των χρηστών στη βάση δεδομένων συγκρίνοντας στη συνέχεια την τρέχουσα με τη συγκεκριμένη βάση πληροφοριών [32]. Στην περίπτωση που εντοπιστούν αξιόλογες αποκλίσεις, αναφέρεται ύποπτη δραστηριότατα [79]. Ουσιαστικά, με τη τεχνική ανίχνευσης ανωμαλιών αποσκοπείται ο εντοπισμός δραστηριότητας που ξεφεύγει της συνηθισμένης για το δίκτυο ή για το σύστημα διαμορφώνοντας τεκμήριο κακόβουλης παρέμβασης ή εισβολής [80]. Η βάση της τεχνικής ανίχνευσης ανωμαλιών είναι ο εντοπισμός των επιθέσεων αξιολογώντας πρότυπα (patterns) που αντιπροσωπεύουν τη

«φυσιολογική» συμπεριφορά χρηστών, δικτύου ή συστήματος [81]. Τα πρότυπα συμπεριφοράς (patterns) διαμορφώνονται από τη συλλογή και επεξεργασία των δεδομένων σε καθημερινή βάση αποτελώντας ουσιαστικά και τη βάση σύγκρισης [83]. Ωστόσο, οι διακυμάνσεις που παρατηρούνται στα δίκτυα και τους χρήστες είναι τέτοιες ώστε πολλές φορές να καθιστούν ιδιαίτερα δύσκολη την προτυποποίηση [84]. Ανεξάρτητα ωστόσο από τις όποιες δυσκολίες, η ανίχνευση ανωμαλιών μέσω προτυποποίησης μπορεί να αποτρέψει κακόβουλες επιθέσεις πριν αυτές πραγματοποιηθούν [60].

2. **Τεχνική ανίχνευσης κακής χρήσης (misuse detection).** Η συγκεκριμένη τεχνική αξιοποιείται από τα περισσότερα IDS [84]. Η φιλοσοφία της βασίζεται στον εντοπισμό ύποπτης δραστηριότητας στη βάση προϋπαρχόντων κακόβουλων επιθέσεων. Οι τυχόν ομοιότητες με προηγούμενες επιθέσεις που έχουν προτυποποιηθεί, ενεργοποιούν το IDS [79]. Η συγκεκριμένη τεχνική χαρακτηρίζεται από υψηλές ταχύτητες ανιχνεύσεων κακόβουλων επιθέσεων και χαμηλό ποσοστό ψευδών ειδοποιήσεων. Μειονέκτημα ωστόσο της συγκεκριμένης τεχνικής αποτελεί η δυσκολία εντοπισμού επίθεσης που δεν έχει προτυποποιηθεί στη βάση δεδομένων καθιστώντας δύσκολη την ανίχνευση νέων επιθέσεων [32].
3. **Υβριδική τεχνική ανίχνευσης (hybrid detection).** Αποτελεί τεχνική που συνδυάζει τα χαρακτηριστικά της τεχνικής ανίχνευσης ανωμαλιών και της τεχνικής κακής χρήσης. Η συγκεκριμένη τεχνική ανίχνευσης κακόβουλων επιθέσεων συνδυάζει στη αξιολόγηση της συμπεριφοράς του δικτύου, του συστήματος και του χρήστη όσο και σε πρότυπα επιθέσεων που έχουν προτυποποιηθεί [60].
4. **Τεχνική ανίχνευσης ανωμαλιών πρωτοκόλλου (protocol anomaly detection).** Αποτελεί μια τεχνική που τυγχάνει αναγνώρισης τα τελευταία χρόνια ενώ σε πρακτικό επίπεδο αποτελεί παράλλαξη της τεχνικής ανίχνευσης ανωμαλιών [83]. Η βασική διαφορά έγκειται στο γεγονός ότι η τεχνική ανίχνευσης ανωμαλιών πρωτοκόλλου δεν στηρίζεται στον εντοπισμό ύποπτης δραστηριότητας στη βάση προτύπων συμπεριφοράς δικτύου, συστήματος ή χρήστη αλλά στην αξιολόγηση της ορθής χρήσης των πρωτοκόλλων επικοινωνίας και κυρίως των TCP/IP. Η ύπαρξη όλο και περισσότερο επιθέσεων

μέσω των πρωτοκόλλων επικοινωνίας καθιστούν τη συγκεκριμένη τεχνική όλο και σημαντικότερη στα IDS [80].

3.11 Προκλήσεις των IDS

Τα περισσότερα από τα σημερινά IDS επικεντρώνονται στην ανίχνευση εισβολών σε δίκτυα που λειτουργούν έως τα 100 Mbps χρησιμοποιώντας τις δυνατότητες της τεχνολογίας των τελευταίων ετών. Τα IDS αν και έχουν εξελιχθεί σημαντικά τις τελευταίες δεκαετίες ωστόσο δεν έχουν καταφέρει να συμβαδίσουν με την ταχεία ανάπτυξη και αύξηση της πολυπλοκότητας των επιθέσεων καθώς και του μεγάλου όγκου των δεδομένων. Πολλά από τα τρέχοντα IDS συχνά λειτουργούν μόνο σε καθεστώς ανίχνευσης και μπορούν ν' ανιχνεύσουν επιθέσεις αλλά δεν μπορούν να εμποδίσουν αποτελεσματικά τυχόν κακόβουλες επιθέσεις πριν προκληθεί βλάβη. Πλέον, τα IDS βρίσκονται αντιμέτωπα με μια σειρά προσκλήσεων που εστιάζουν στα ακόλουθα σημεία:

1. **Πρόσθετη Κάλυψη έναντι επιθέσεων:** Τα IDS προϊόντα επικεντρώνονται στην υπογραφή ή στην ανωμαλία ή στην ανίχνευση DoS. Πλέον, η πολυπλοκότητα των επιθέσεων προαπαιτεί ενσωμάτωση λύσεων κάλυψης όλων των πιθανών προσπαθειών εισβολής στο δίκτυο [32].
2. **Εξειδίκευση ανιχνεύσεων:** Τα IDS χαρακτηρίζονται γενικά από ακρίβεια στον προσδιορισμό των επιθέσεων. Ο προσδιορισμός μιας ψευδούς ή αληθούς εισβολής είναι μεταξύ των βασικών δραστηριοτήτων των IDS που ορισμένες φορές όμως δυσκολεύονται με σαφήνεια να διακρίνουν τον αριθμό τους εξαιτίας ορισμένων τύπων επιθέσεων που έχουν ακριβώς αυτό το [85].
3. **Εστίαση στην ανίχνευση και λιγότερο πρόληψη:** Τα IDS θα πρέπει να εστιάσουν περισσότερο στην ανίχνευση επιθέσεων και λιγότερο στην πρόληψή τους η οποία είναι μια αντιδραστική δραστηριότητα και συχνά δεν αποτρέπει την εισβολή [86].
4. **Ανάπτυξη IDS για υψηλές ταχύτητες δικτύων:** Τα IDS θα πρέπει να εστιάσουν την ανάπτυξή τους σε ακρίβεια ανιχνεύσεων στα υψηλής ταχύτητας δίκτυα με ανάλογη βελτίωση των υποδομών και της αρχιτεκτονικής τους [32].
5. **Ανάπτυξη εφαρμογών λογισμικού και υλικού υπολογιστών για υποστήριξη των IDS:** Τα IDS καλούνται να ακολουθήσουν τις τεχνολογικές

εξελίξεις προσαρμοζόμενα στην ανάπτυξη του λογισμικού και των υπολογιστών ώστε οι ανιχνεύσεις εισβολών να παρουσιάζουν ακρίβεια.

6. **Αύξηση επεκτασιμότητας:** Ο σχεδιασμός των IDS εστιάζει σε παροχή ανίχνευσης εισβολών σε οργανισμούς και επιχειρήσεις μικρής ή μεσαίας κλίμακας. Πλέον, σημαντική πρόκληση αποτελούν τα δίκτυα κυβερνήσεων και κρατών και διεύρυνση των τμημάτων του δικτύου όπου μπορούν να ελεγχθούν.
7. **Μείωση απαιτούμενων τεχνολογικών πόρων:** Σημαντική πρόκληση για τα σύγχρονα IDS είναι να λειτουργήσουν αξιοποιώντας όσο το δυνατόν λιγότερους πόρους παρέχοντας ταυτόχρονα υψηλά επίπεδα ακρίβειας ανιχνεύσεων.

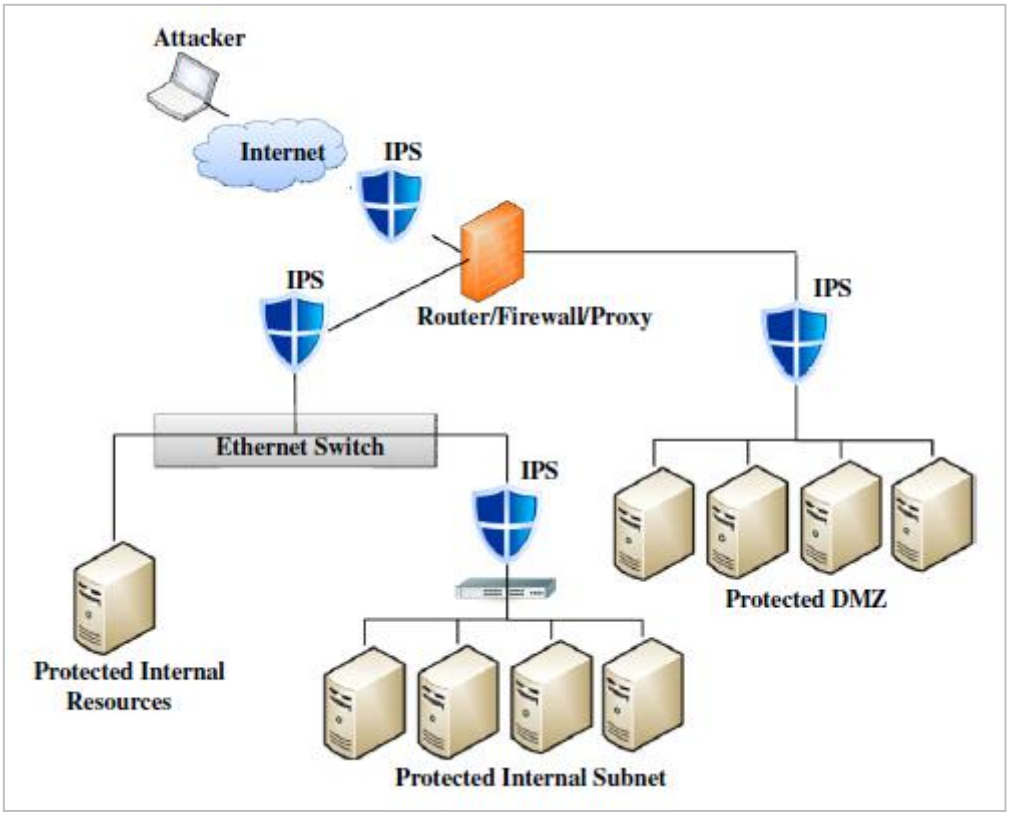
Κεφάλαιο 4

Συστήματα Αποτροπής Εισβολών (Intrusion Protection Systems, IPS)

4.1 Έννοια των Intrusion Protection Systems (IPS)

Τα Συστήματα Αποτροπής Εισβολών (Intrusion Protection Systems, IPS) διαθέτουν τέτοια αρχιτεκτονική ώστε να παρέχουν πιο ακριβείς ανιχνεύσεις επιθέσεων αποτελώντας τη βάση για την ανάπτυξη μηχανισμών αντιμετώπισης επιθέσεων. Ένα IPS χωρίς επαρκή ικανότητα ανταπόκρισης είναι περιορισμένης χρησιμότητας στην ασφάλεια ενός δικτύου. Τα IPS παρακολουθούν τις δραστηριότητες της κίνησης στο δίκτυο και στο σύστημα για την ανίχνευση πιθανών εισβολών με τη βοήθεια του IDS και ανταποκρίνονται δυναμικά σε εισβολές παρεμποδίζοντας ή απομονώνοντας την προσπάθεια. Τα IPS πρέπει να ρυθμίζονται με ακρίβεια ώστε να προσφέρουν τα προσδοκόμενα αποτελέσματα [32].

Για την πρόληψη των εισβολών παράλληλα με τα IPS χρησιμοποιούνται ως επί το πλείστων και firewall προσδιορίζοντας συνολικά την αποδεκτή κίνηση και συμπεριφορά στο δίκτυο. Με βάση τους προκαθορισμένους κανόνες, το IPS ορίζει την τόσο την αποδεκτή όσο και την μη αποδεκτή κίνηση σε ένα δίκτυο. Σε περίπτωση εντοπισμού απειλής, το IPS μπορεί να εμποδίσει μεμονωμένα την επίθεση, μπορεί να αλλάξει το περιεχόμενο της επίθεσης ή να αλλάξει το περιβάλλον ασφαλείας [87].



Εικόνα 4.1: Σύστημα Αποτροπής Εισβολών (Intrusion Protection Systems, IPS) Πηγή: [60]

Τα Συστήματα Πρόληψης Εισβολών (IPS), είναι συσκευές ή προγράμματα που χρησιμοποιούνται για την ανίχνευση εισβολών σε δίκτυα ή συστήματα δρώντας προστατευτικά. Μεταξύ των ενεργειών δράσης των IPS περιλαμβάνονται η δημιουργία σχετικών συναγερμών ή ειδοποιήσεων ή η αυτόματη αποτροπή της εισβολής. Τα IPS όπως και τα firewalls αποτελούν βασικά εργαλεία για την προστασία του δικτύου ή και των συστημάτων αυτού από εισβολές.

Μια αποτελεσματική προσέγγιση ασφάλειας με βάση IPS , δεν απαιτεί την εγκατάσταση IDS σε κάθε κόμβο[88]. Η προσέγγιση αυτή λύνει το πρόβλημα της εμπιστοσύνης και τη μεταφορά ειδοποιήσεων με μήνυμα με αποτέλεσμα τους λιγότερους ψευδής συναγερμούς. Τα IPS ταξινομούνται κυρίως σε δύο κατηγορίες: Host και δικτύου[60,89].

Σε ορισμένα συστήματα ή δίκτυα δεν επαρκούν τα firewalls στην προσπάθεια αποτροπής μια εισβολής καθιστώντας εξόχως σημαντική τη χρησιμότητα των IPS τα οποία εστιάζουν κυρίως στον εντοπισμό πιθανών συμβάντων, την καταγραφή πληροφοριών και τη δημιουργία αναφορών. Επιπλέον, τα IPS χρησιμοποιούνται και για άλλους σκοπούς όπως ο εντοπισμός προβλημάτων αναφορικά με θέματα και πολιτικές

ασφάλειας. Τα IPS συλλέγουν τις πληροφορίες που σχετίζονται με παρατηρούμενα γεγονότα, γνωστοποιώντας στους διαχειριστές ασφάλειας τα σημαντικά παρατηρηθέντα γεγονότα δημιουργώντας σχετικές εκθέσεις. Πολλές φορές ο ρόλος των IPS συγχέεται με τα IDS τα οποία εντοπίζουν την κακόβουλη δραστηριότητα. Οι κύριες λειτουργίες των IPS είναι να προσδιορίσουν την κακόβουλη δραστηριότητα, να συνδέσουν τις πληροφορίες σχετικά με αυτή τη δραστηριότητα, να δημιουργήσουν σχετική αναφορά και να την αποτρέψουν[53].

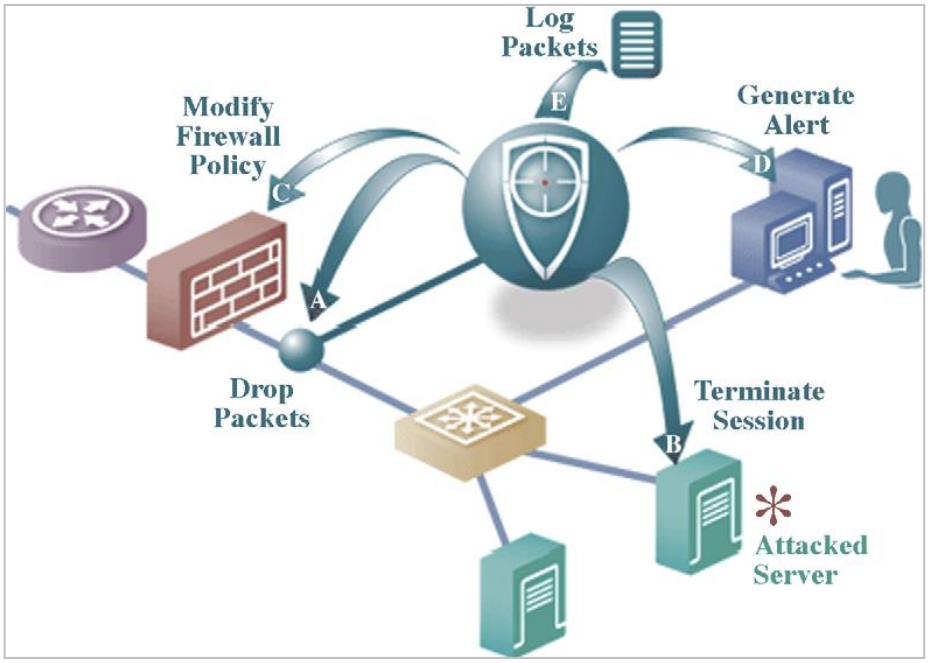
Τα IPS θεωρούνται επεκτάσεις των IDS επειδή οι δραστηριότητες και των δύο αφορούν σε παρακολούθηση της κίνησης στο δίκτυο για κακόβουλη δραστηριότητα. Οι κύριες διαφορές τους, σε αντίθεση με τα IDS είναι ότι τα IPS τοποθετούνται σε σειρά και είναι σε θέση να αποτρέψουν ενεργά ή να μπλοκάρουν εισβολές που ανιχνεύονται, μπορούν να αναλάβουν δράσεις όπως η αποστολή ενός συναγερμού, ο εντοπισμός κακόβουλων δραστηριοτήτων και ο αποκλεισμός παραβατικών IP [90,91].

4.2 Αρχιτεκτονική των IPS

Η αρχιτεκτονική του IPS (εικόνα 4.2) καλύπτει ένα ευρύ φάσμα δραστηριοτήτων και ενεργειών αναφορικά με την αποτροπή των εισβολών σε ένα δίκτυο ή σε ένα σύστημα με αναφορές και ειδοποιήσεις που μπορούν να προστατέψουν ουσιαστικά τη λειτουργία τους. Κατά την ανίχνευση μίας επίθεσης, η αρχιτεκτονική των IPS επιτρέπει στο σύστημα τα εξής [32]:

1. **Αποτροπή εισβολών.** Επειδή η αρχιτεκτονική των IPS , τους επιτρέπει να λειτουργούν σε κατάσταση in - line, είναι σε θέση να αποτρέψουν εισβολές σε πραγματικό χρόνο χωρίς να απαιτείται κάποια επιμέρους ενέργεια από τους χρήστες.
2. **Τερματισμός σύνδεσης.** Η αρχιτεκτονική των IPS επιτρέπει την ομαλή λειτουργία των πρωτοκόλλων επικοινωνίας και τον έλεγχο των πληροφοριών που διακινούνται ενώ σε περίπτωση που η δραστηριότητα αξιολογηθεί ως επικίνδυνη , μπορεί να αποκλειστεί.

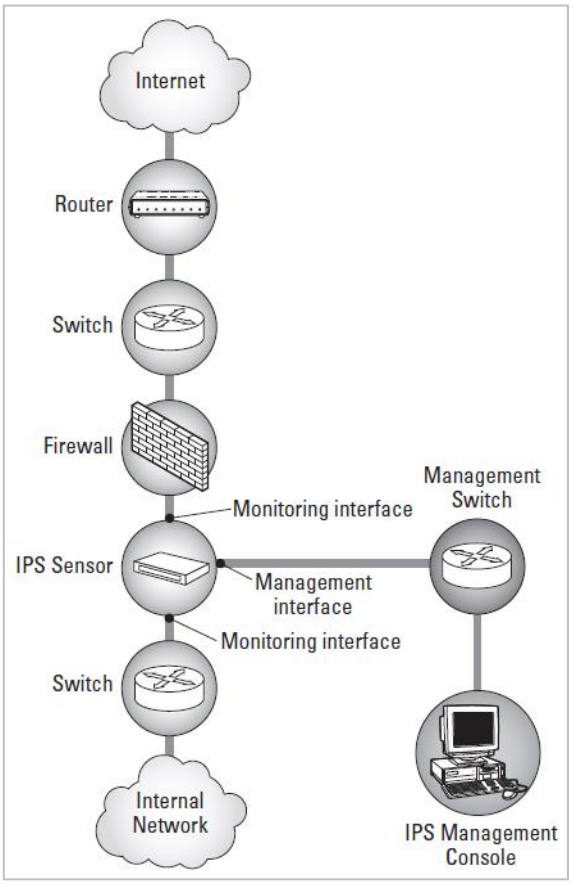
3. **Τροποποίηση τείχους προστασίας.** Η αρχιτεκτονική του IPS επιτρέπει στους χρήστες να αναμορφώνουν τα τείχη προστασίας του δικτύου σε περιπτώσεις επιθέσεων που μεταβάλλουν τις ρυθμίσεις ελέγχου.
4. **Ενημέρωση σε πραγματικό χρόνο.** Η δραστηριότητα που παραβιάζει τις πολιτικές ασφάλειας του συστήματος εντοπίζεται μέσω της αρχιτεκτονικής IPS που δημιουργεί και αποστέλλει ειδοποίηση σε πραγματικό χρόνο σε ένα σύστημα διαχείρισης. Η σωστή διαμόρφωση των ειδοποιήσεων είναι ζωτικής σημασίας για τη διατήρηση της αποτελεσματικής προστασίας.
5. **Επιμέρους έλεγχοι.** Ο εντοπισμός κακόβουλης δραστηριότητας μέσω της αρχιτεκτονικής των IPS είναι σε θέση να εντοπίσει αυτήν την δραστηριότητα σε επιμέρους συστήματα/τμήματα του δικτύου με σκοπό τον ενδελεχή έλεγχο τους.



Εικόνα 4.2: Λειτουργία IPS στο δίκτυο Πηγή: [32]

Στα βασικά συστατικά στοιχεία της αρχιτεκτονικής των IPS περιλαμβάνονται οι αισθητήρες και η κονσόλα διαχείρισης. Στα IPS τις περισσότερες φορές επιλέγεται η τοποθέτηση περισσότερων του ενός αισθητήρων (sensors) που βρίσκονται σε διαφορετικά τμήματα του δικτύου. Μερικά από τα σημεία όπου θα μπορούσε να τοποθετηθεί ένας αισθητήρας IPS είναι περιμετρικά και σε στρατηγικά σημεία εντός του δικτύου έχοντας εποπτικό ρόλο για την προστασία ολόκληρου του συστήματος. Βασικό σημείο της αρχιτεκτονικής των IPS όπως προαναφέρθηκε είναι και η κονσόλα

διαχείρισης που παρέχει την κεντρική διοίκηση και τον έλεγχο όλων των αισθητήρων του συστήματος. Στα τυπικά χαρακτηριστικά μια κονσόλας διαχείρισης IPS εντάσσονται η συγκέντρωση των συμβάντων ασφαλείας, η κεντρική διαχείριση της πολιτικής ανίχνευσης, η λήψη, εισαγωγή και εφαρμογή ενημερώσεων του IPS, οι διεπαφές του χρήστη για συμβάντα ασφαλείας, εκθέσεις, ειδοποιήσεις, και πίνακες καθώς και ενημερώσεις αναφορικά με τη λειτουργία των αισθητήρων και της ίδιας της κονσόλας. Στην εικόνα 4.3 παρουσιάζεται μια τυπική αρχιτεκτονική δομή ενός συστήματος IPS [92,93].



Εικόνα 4.3: Αρχιτεκτονική δομή IPS Πηγή: [93]

4.3 Πλεονεκτήματα & Μειονεκτήματα IPS και Διαφορές από Firewall

Τα συστήματα IPS μπορούν να συμβάλλουν σημαντικά στην αποτροπή των κακόβουλων επιθέσεων παρουσιάζοντας σημαντικά πλεονεκτήματα αλλά και μειονεκτήματα. Παρότι τα συστήματα IPS εμφανίζουν σειρά μειονεκτημάτων που θα μπορούσαν να αποτελούν αιτία αποφυγής της χρήσης του σε εκτεταμένα δίκτυα, παρουσιάζουν μια σειρά πλεονεκτημάτων που τα καθιστούν ιδιαίτερα ελκυστικά. Τα IPS μπορούν να προσφέρουν προστασία αποτροπής εισβολών που οποιαδήποτε άλλη μέθοδο ασφαλείας δεν μπορεί να προσφέρει. Ένα από τα πλεονεκτήματα των IPS είναι η ικανότητά τους να δρουν όπως το λογισμικό προστασίας από ιούς με ταυτόχρονη ανίχνευση αλλά και αποτροπή της κακόβουλης δραστηριότητας , εντοπίζοντας παράλληλα την προέλευσή της. Τα IPS μπορούν να αποτρέψουν τις επιθέσεις που αφορούν σε αλλοίωση των δεδομένων ή υπερφόρτωσης του δικτύου [77,94].

Τα IPS είναι σημαντικά για την ομαλή λειτουργία ενός δικτύου καθώς παρέχουν ενημέρωση για τυχόν κακόβουλη δραστηριότητα σε πραγματικό χρόνο δίνοντας τη δυνατότητα στους χρήστες και τους διαχειριστές να προβούν στις κατάλληλες ενέργειες περιορίζοντας τους όποιους κινδύνους. Στα συστήματα IPS οι ενημερώσεις είναι συχνές εξασφαλίζοντας την προστασία από οποιαδήποτε νεότερη επίθεση ενώ είναι σε θέση να ενημερώνονται για οποιαδήποτε ύποπτη δραστηριότητα για το σύνολο των συσκευών του δικτύου, δημιουργώντας συσχετισμούς και αρχεία καταγραφής μέσω της κονσόλας διαχείρισης [77,94].

Ένα από τα πιο κοινά μειονεκτήματα των IPS είναι η ανίχνευση των ψευδώς θετικών ή ψευδώς αρνητικών εισβολών όταν η κακόβουλη δραστηριότητα λαμβάνει χώρα εκτός του δικτύου ελέγχου. Σε ορισμένες περιπτώσεις το σύστημα δεν είναι σε θέση να εντοπίσει μια πραγματική απειλή δημιουργώντας ψευδής ειδοποιήσεις στους χρήστες. Το συγκεκριμένο πρόβλημα είναι γνωστό ωστόσο θα πρέπει να αποτελεί έναν από τους κύριους στόχους των διαχειριστών δικτύου και τους κατασκευαστές IPS ώστε να ελαχιστοποιηθεί όσο το δυνατόν περισσότερο[77].

Ένα άλλο μειονέκτημα των IPS είναι το αυξημένο κόστος ειδικά στην περίπτωση που η πολυπλοκότητα του δικτύου απαιτεί την εγκατάσταση περισσότερων του ενός συστημάτων. Συγκεκριμένα, η τοποθέτηση πολλών αισθητήρων πέραν του όποιου κόστους δημιουργεί σημαντικές δυσκολίες ως προς τη συνολική απόδοση του δικτύου καθώς αυξάνονται οι πιθανότητες λανθασμένων ειδοποιήσεων. Η διαχείριση ενός

πολύπλοκου συστήματος IPS δημιουργεί προβλήματα που ο διαχειριστής καλείται να επιλύσει ώστε να προστατεύεται πλήρως το δίκτυο αλλά και το σύστημα [94].

Τα firewalls και τα IPS είναι βασικά εργαλεία για την προστασία από εισβολές. Η λειτουργία και των δύο εστιάζει στην αποτροπή εισβολών παρουσιάζοντας ωστόσο ορισμένες διαφοροποιήσεις. Ειδικότερα, τα firewall αποσκοπούν στον έλεγχο του δικτύου εμποδίζοντας το σύνολο της μη αποδεκτής κίνησης ενώ το IPS έχει σχεδιαστεί για να επιτρέπει τα πάντα εκτός από την κακόβουλη δραστηριότητα. Ουσιαστικά αν και τα δύο συστήματα διενεργούν ελέγχους, τα firewall δεν είναι σε θέση να αποτρέψουν όλες τις εισβολές καθώς δεν ελέγχουν το σύνολο του δικτύου σε αντίθεση με τα IPS που παρέχουν μια πιο εξειδικευμένη προστασία. Τέλος, τα firewall έναντι των IPS παρουσιάζουν διαφοροποίηση ως προς τα επίπεδα ελέγχου τα οποία δεν ξεπερνούν τα στενά όρια μιας συσκευής του συστήματος χωρίς τη δυνατότητα επικοινωνίας και ανταλλαγής πληροφοριών, στοιχείο που στα IPS όχι μόνο είναι εφικτό αλλά αποτελεί και τη βάση λειτουργίας τους [77].

4.4 Ταξινόμηση IPS

Οι εξελίξεις σε τεχνολογικό επίπεδο συνέβαλλαν σημαντικά στη διαμόρφωση διαφόρων τύπων IPS τα οποία ανάλογα με τη λειτουργία του ταξινομούνται σε επιμέρους κατηγορίες. Μεταξύ των IPS, διακρίνονται οι εξής κατηγορίες:

1. **Μεμονωμένα IPS (Host Intrusion Prevention System).** Τα HIPS είναι ένας συνδυασμός προσωπικού firewall, IDS, και antivirus [95]. Αποτελούν ένα προ-εγκατεστημένο πακέτο λογισμικού που παρακολουθεί μια συσκευή και το δίκτυο για ύποπτη δραστηριότητα αναλύοντας τα γεγονότα που συμβαίνουν εντός συγκεκριμένων ορίων ελέγχου. Με άλλα λόγια ένα HIPS έχει ως στόχο να αποτρέψει κακόβουλη εισβολή μέσω της παρακολούθησης της συμπεριφοράς του κώδικα. Αυτό καθιστά δυνατή την ασφάλεια του δικτύου χωρίς να εξαρτάται από μια συγκεκριμένη απειλή που πρέπει να προστεθεί σε μια ενημέρωση [60,95, 96].
2. **Δικτυακά IPS (Network Intrusion Prevention System)**. Το IPS με βάση το δίκτυο (NIPS) είναι ένα σύστημα που χρησιμοποιείται για την παρακολούθηση ενός δικτύου αναφορικά με την προστασία, την εμπιστευτικότητα, την

ακεραιότητα και τη διαθεσιμότητα [097]. Οι κύριες λειτουργίες του περιλαμβάνουν την προστασία του δικτύου από εισβολές και απειλές, όπως η άρνηση υπηρεσίας (DoS) και η μη εξουσιοδοτημένη χρήση. Τα NIPS παρακολουθούν το δίκτυο για κακόβουλη δραστηριότητα στη βάση ανάλυσης πρωτοκόλλων. Η εγκατάσταση ενός NIPS δημιουργεί πεδία ασφαλείας έναντι δούρειων ίππων (trojans), ιών (virus) και πολυμορφικών απειλών. Το NIPS βρίσκεται σε σειρά με το δίκτυο και ελέγχει την κίνηση ενώ όταν εντοπίσει κακόβουλη δραστηριότητα ενεργεί με βάση προκαθορισμένους κανόνες [98].

3. **Ασύρματα IPS (Wireless Intrusion Prevention System).** Τα WIPS έχουν παρόμοια λειτουργία με τα NIDS με τη διαφορά ότι ελέγχουν τη δραστηριότητα μέσω της ασύρματης τεχνολογίας. Τα συγκεκριμένα συστήματα μειονεκτούν στο γεγονός ότι απαιτούν ασύρματη διασύνδεση παρουσιάζοντας ταυτόχρονα αυξημένο κόστος [99].
4. **Υβριδικά IPS (Hybrid Intrusion Prevention System).** Το HIPS είναι σχεδιασμένα κατά τέτοιο τρόπο ώστε να συνδυάζουν τις τεχνολογικές δυνατότητες. Η ανίχνευση των εισβολών βασίζεται σε ανωμαλίες δραστηριότητας και σε ανιχνεύσεις βάσει υπογραφών [100]. Οι παράμετροι των ανιχνευτών που ελέγχονται από μια κεντρική μονάδα με σκοπό να ενισχυθεί η συνολική απόδοση της ανίχνευσης επιθέσεων DDoS, να μειωθεί ο χρόνος εντοπισμού των κακόβουλων επιθέσεων και να αυξηθεί η ακρίβεια εντοπισμού [101].

4.5 Τεχνικές Ανίχνευσης Εισβολών στα IPS

Οι τεχνικές ανίχνευσης εισβολών στα IPS δεν διαφοροποιούνται αισθητά έναντι των όσων ισχύουν και για τα IDS. Ειδικότερα, μεταξύ των τεχνικών ανίχνευσης της κακόβουλης δραστηριότητας περιλαμβάνονται η τεχνική της ανίχνευσης ανωμαλιών (anomaly detection), της ανάλυσης κατάστασης των πρωτοκόλλων (protocol anomaly detection), της ανίχνευσης βάσει κανόνων (rule – based), της ανίχνευσης βάση υπογραφής (signature – based). Ειδικότερα:

1. Τεχνική ανίχνευσης ανωμαλιών (anomaly detection). Το IPS βάσει της συγκριμένης τεχνικής μπορεί να ανιχνεύσει περιστατικά με βάση

συγκεκριμένων πρότυπα και να ορίσει τη δραστηριότητα ως φυσιολογική ή μη [102]. Ένα διακριτό πλεονέκτημα της ανίχνευσης ανωμαλιών είναι ότι βασίζεται στην αξιολόγηση καταστάσεων που δεν μπορεί να ορίσει κάποιο πρωτόκολλο. Η ανίχνευση ανωμαλιών είναι μια περίπλοκη διαδικασία που ενώ πολλές φορές οδηγεί σε ψευδείς ειδοποιήσεις, αποτελεί ισχυρό εργαλείο εντοπισμού κακόβουλης δραστηριότητας [93].

2. Τεχνική ανάλυσης πρωτοκόλλου (stateful protocol analysis). Το IPS με τη συγκεκριμένη τεχνική μπορεί να εντοπίσει κακόβουλη δραστηριότητα με τον έλεγχο σε μεμονωμένο δίκτυο χωρίς να βασίζεται σε πρότυπα συμπεριφοράς δικτύου, συστήματος ή χρήστη αλλά στην αξιολόγηση της ορθής χρήσης των πρωτοκόλλων επικοινωνίας και κυρίως του TCP/IP. Η αυξανόμενη παρουσία προσπαθειών εισβολής μέσω πρωτοκόλλων επικοινωνίας καθιστούν τη συγκεκριμένη τεχνική όλο και σημαντικότερη στα IPS [93,103].
3. Τεχνικές ανίχνευσης εισβολής βάσει κανόνων (rule – based). Το IPS μπορεί να ανιχνεύσει προσπάθειες εισβολής συγκρίνοντας με προκαθορισμένες περιπτώσεις και γνωστά τρωτά σημεία. Αυτή η τεχνική ανίχνευσης είναι αρκετά αποτελεσματική στην ανίχνευση γνωστών απειλών. Μερικά παραδείγματα επιθέσεων που αποτρέπονται με τη συγκεκριμένη τεχνική είναι οι επιθέσεις που στοχεύουν σε τρωτά σημεία των λειτουργικών συστημάτων, επιθέσεις DoS, υποκλοπές δεδομένων κλπ. Επειδή οι τύποι επιθέσεων ανανεώνονται συνεχώς, τα IPS πρέπει να ενημερώνουν τακτικά τους κανόνες ελέγχου [93].
4. Τεχνικές ανίχνευσης εισβολής βάσει υπογραφής (signature – based). Η συγκεκριμένη προσέγγιση βασίζεται στον έλεγχο δραστηριότητας βάσει υπογραφών εντοπίζοντας κακόβουλη δραστηριότητα. Ο περιορισμός αποτροπής των εισβολών με τη συγκεκριμένη τεχνική εστιάζει στον έλεγχο βάσει προκαθορισμένων υπογραφών δημιουργώντας αφενός λιγότερες ψευδείς ειδοποιήσεις, ωστόσο το σύστημα είναι περισσότερο ευάλωτο σε νέες επιθέσεις [93].

4.6 Κατηγοριοποίηση Επιθέσεων Ενάντια Στα IPS

Τα Συστήματα Αποτροπής Εισβολών (IPS) έχουν σχεδιαστεί ώστε να αποτρέπουν πολλούς διαφορετικούς τύπους επιθέσεων. Η ανίχνευση και αποτροπή των επιθέσεων διαφοροποιείται ανάλογα με το είδος της επίθεσης στο σύστημα ενώ η κατανόησή τους βοηθά τη συνεχή βελτίωση των IPS συνολικά. Μερικές από τις βασικότερες κατηγορίες επιθέσεων που τα IPS καλούνται να αντιμετωπίσουν είναι οι εξής:

1. Επιθέσεις Denial of Service (DoS). Οι επιθέσεις DoS αποσκοπούν στην παρεμπόδιση παροχής υπηρεσιών δημιουργώντας προβλήματα προσβασιμότητας. Ουσιαστικά, το δίκτυο αδυνατεί να παρέχει κανονικά τις υπηρεσίες του, είτε σε επίπεδα εύρους είτε σε επίπεδα συνδεσιμότητας. Οι συγκεκριμένες επιθέσεις υπερφορτώνουν το δίκτυο με πακέτα αντλώντας τους υπολογιστικούς πόρους του και εμποδίζοντας την πρόσβαση στους χρήστες. Οι επιθέσεις DoS μπορούν να λάβουν έναν πολυδιάστατο χαρακτήρα εντείνοντας το πρόβλημα λειτουργίας του δικτύου [104,105].
2. Επιθέσεις worms. Το συγκεκριμένο είδος επιθέσεων αποσκοπεί μέσω κακόβουλου κώδικα να βλάψει τη λειτουργία των συστημάτων και του δικτύου. Τα worms δικτύου αξιοποιούν τα όποια κενά ασφαλείας των διάφορων εφαρμογών ενώ μέσω του δικτύου μπορούν να εξαπλωθούν ταχύτατα. Οι περισσότερες επιθέσεις τύπου worm μπορούν να αποφευχθούν με τη χρήση προεπιλεγμένων ρυθμίσεων ασφαλείας τόσο στα firewall όσο και στα IPS. Οι επιθέσεις τύπου worm αναζητούν συνεχώς αδυναμίες και κενά ασφαλείας με αποτέλεσμα να εξαπλώνονται σε επόμενα υπολογιστικά συστήματα μέσα σε έναν αέναο κύκλο μέχρι να απομονωθούν από κάποιο σύστημα ασφάλειας [106].
3. Δούρειοι ίπποι (Trojans). Τα Trojans αποτελούν ένα άλλο είδος κακόβουλου λογισμικού. Όπως τα worms έτσι και τα Trojans σχεδιάζονται για την αναζήτηση κενών ασφάλειας ωστόσο απαιτούν την ανθρώπινη παρέμβαση. Συγκεκριμένα, τα Trojans παρουσιάζονται ως συμβατό και αποδεκτό λογισμικό ή αρχείο το οποίο όμως υποκρύπτει απειλή για το σύστημα. Η αποδοχή τους δίνει πρόσβαση σε τρίτους στο σύστημα η οποία μπορεί να κυμαίνεται από απλή παρακολούθηση της δραστηριότητας μέχρι τον πλήρη έλεγχο. Τα IPS διαθέτουν όλες τις απαραίτητες δικλείδες ασφαλείας έναντι των συγκεκριμένων

επιθέσεων προειδοποιώντας άμεσα το χρήστη και απομονώνοντας ταυτόχρονα την εισβολή [107].

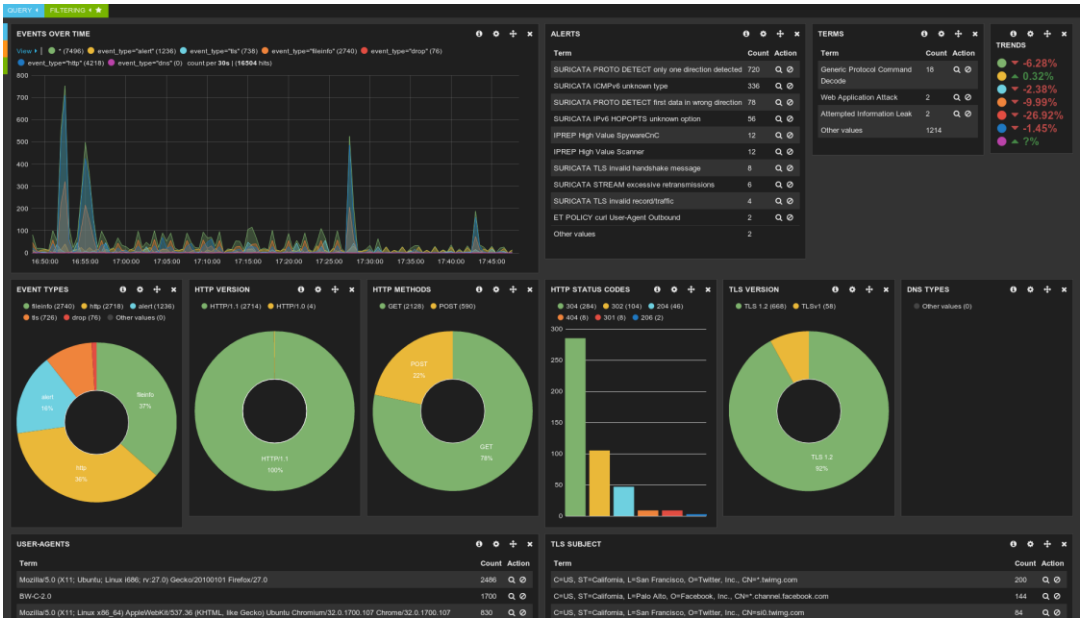
4. Επίθεση υπερφόρτωσης μνήμης (buffer overflow). Μέσω της συγκεκριμένης τεχνικής, γίνεται προσπάθεια ανάληψης του πλήρη έλεγχου του συστήματος από τον εισβολέα. Μια επίθεση υπερφόρτωσης μνήμης εστιάζει στην αποστολή πολλαπλών μηνυμάτων λειτουργίας στην προσωρινή μνήμη με τα περίσσεια δεδομένα να αντικαθιστούν προκαθορισμένες ρυθμίσεις λειτουργίας βλάπτοντας τη λειτουργία του συστήματος. Τα IPS είναι σε θέση να εμποδίσουν τέτοιου είδους επιθέσεις πάντα με τις κατάλληλες επιλογές ρυθμίσεων [108].

Κεφάλαιο 5
Suricata

5.1 Επισκόπηση Suricata

Το Suricata είναι ένα Σύστημα Αποτροπής Εισβολών (IDS) που βασίζεται σε τεχνική ανίχνευσης εισβολών βάση κανόνων (rule – based) παρακολουθώντας την κίνηση στο δίκτυο και ειδοποιώντας για τυχόν ύποπτη δραστηριότητα. Όπως τα περισσότερα IDS έχει σχεδιαστεί για να προσαρμόζεται στο πλαίσιο των υφιστάμενων απαιτήσεων ασφαλείας του δικτύου. Το Suricata μπορεί να παρέχει ασφάλεια στο δίκτυο in – line αλλά και παθητικά μέσω κατάλληλων ρυθμίσεων, μπορεί να διαχειρίζεται πολλαπλά επίπεδα κυκλοφορίας, ενώ λειτουργεί πολυνηματικά (multi thread) [109].

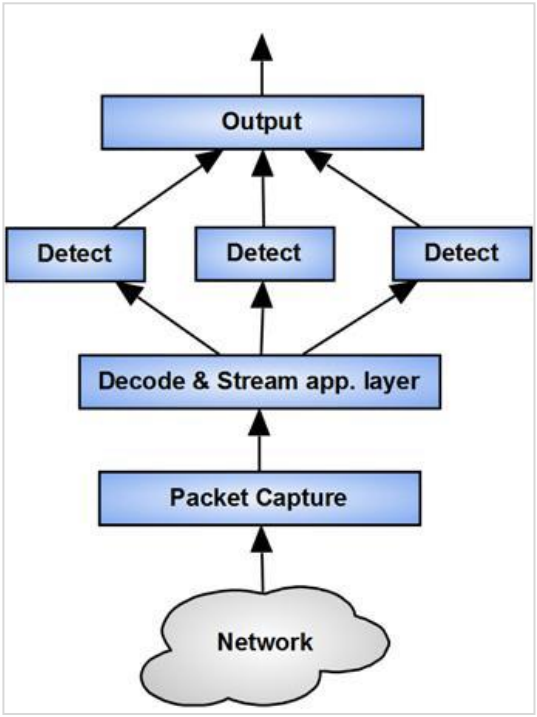
Βασίζει την λειτουργία του σε αρχεία κανόνων , καθένα από τα οποία αντιστοιχεί σε διαφορετικού τύπου κίνηση, tcp ή udp. Τα αρχεία κανόνων περιέχουν με την σειρά τους υπογραφές βάση των οποίων το Suricata εξετάζει τα πακέτα που κυκλοφορούν στο δίκτυο , με σκοπό να διαπιστώσει εάν η αλληλουχία των πακέτων αντιστοιχεί σε κάποια από αυτές τις υπογραφές. Στην περίπτωση που συμβεί το τελευταίο , καταγράφει την “ύποπτη” κίνηση σε αρχείο.



Εικόνα 5.1: Λειτουργία Suricata Πηγή: [110]

Το Suricata μπορεί να λειτουργήσει σαν IDS αλλά και IPS ενώ βασίζεται σε προκαθορισμένο σύνολο κανόνων, η ακρίβεια των οποίων θα καθορίσει το ποσοστό των ψευδώς αρνητικών και ψευδώς θετικών απειλών. Η λειτουργία ως IDS ή IPS εξαρτάται από το είδος του ελέγχου δραστηριότητας στο οποίο υποβάλλεται. Ο σχεδιασμός του βασίζεται σε παρεμφερή λογισμικά όπως το Snort ωστόσο χρησιμοποιεί μια πολυνηματική προσέγγιση ανίχνευσης (multi – threaded) που επιτρέπει την αξιοποίηση όλων των τεχνολογικών δυνατοτήτων και εκτέλεσης ταυτόχρονης ανάλυσης της κίνησης του δικτύου διαμορφώνοντας ένα ευρύτερο λειτουργικό πλαίσιο [111].

Το Suricata αποτελεί ένα IDS επόμενης γενιάς το οποίο διαθέτει χαρακτηριστικά που το καθιστούν ως ιδιαίτερα αξιόπιστο εργαλείο αποτροπής εισβολών. Η λειτουργία του παρουσιάζει σημαντικές βελτιώσεις και λειτουργίες σε σχέση με άλλα παρόμοια λογισμικά. Ως πολυνηματικό εργαλείο, το Suricata προσφέρει αυξημένη ταχύτητα και αποτελεσματικότητα στην ανάλυση της κίνησης του δικτύου. Είναι σχεδιασμένο κατά τέτοιο τρόπο ώστε να αξιοποιεί την αυξημένη επεξεργαστική ισχύ που προσφέρεται από την τεχνολογία των σύγχρονων επεξεργαστών [112].



Εικόνα 5.2: Μηχανισμός λειτουργίας Suricata Πηγή: [113]

5.2 Χαρακτηριστικά Suricata

Το Suricata διαθέτει ορισμένα χαρακτηριστικά που το καθιστούν ως ιδιαίτερα αξιόπιστο εργαλείο αποτροπής εισβολών. Μεταξύ των σημαντικότερων χαρακτηριστικών του στοιχείων αναφέρονται τα εξής [114]:

1. Πολυνηματισμός (multi - threading). Αποτελεί πολυνηματική εφαρμογή με δυνατότητες αξιοποίησης πολλαπλών CPUs και πλήθους πακέτων.
2. Αυτόματη ανίχνευση πρωτοκόλλου. Πρωτόκολλα όπως τα IP, TCP, UDP, ICMP, HTTP, TLS, FTP και SMB αναγνωρίζονται αυτόματα κατά την έναρξη της δραστηριότητας του δικτύου με ανάπτυξη σχετικών υπογραφών οι οποίες ορίζουν το πρωτόκολλο που θα χρησιμοποιηθεί.
3. Αποσυμπίεση GZip. Ο αναλυτής HTTP αποκωδικοποιεί όλα τα συμπιεσμένα δεδομένα του δικτύου παρέχοντας την αναγκαία πληροφόρηση για την ανίχνευση της εισβολής.
4. Ανεξάρτητη βιβλιοθήκη HTTP. Ο αναλυτής HTTP μπορεί να αξιοποιηθεί και από εφαρμογές όπως proxy, http filters κ.τ.λ.
5. Πρότυπες μεθόδους εισαγωγής.

6. Μεταβλητές ροής.
7. Ταχεία ταυτοποίηση IP.
8. Μονάδα καταγραφής HTTP.
9. Χρησιμοποίηση της τεχνολογίας επιτάχυνσης της κάρτας γραφικών.
10. Ταυτοποίηση αρχείων, συνδεσιμότητας, εξαγωγή δεδομένων.
11. Καταγραφικό DNS.
12. Δυνατότητα δημιουργίας αρχείου καταγραφής κίνησης δικτύου cap.

5.3 Πλεονεκτήματα Suricata

Το Suricata χαρακτηρίζεται από μια σειρά πλεονεκτημάτων που το καθιστούν ιδιαίτερα ανταγωνιστικό στον τομέα του. Μεταξύ των σημαντικότερων πλεονεκτημάτων του συγκαταλέγονται τα εξής:

1. Εξαιρετικές δυνατότητες ελέγχων. Η πολυεπίπεδη λειτουργία του κατά πολλαπλών απειλών του επιτρέπουν να ανταποκρίνεται σε όλες τις ανάγκες ελέγχων ενώ και τα αποτελέσματα αξιολογήσεων κινούνται σε πολύ υψηλά επίπεδα [115].
2. Ταχύτατη ταυτοποίηση πρωτοκόλλων. Τα πιο κοινά πρωτόκολλα αναγνωρίζονται αυτόματα από το Suricata κατά τη διαδικασία μεταφοράς αρχείων και πληροφοριών διασφαλίζοντας την ακεραιότητα μεταβίβασης αλλά και αποτρέποντας τις όποιες κακόβουλες επιθέσεις. Το Suricata εντοπίζει αυτόματα πρωτόκολλα όπως το HTTP σε οποιαδήποτε θύρα και εφαρμόζει τη σωστή λογική ανίχνευσης και καταγραφής συμβάλλοντας σε μεγάλο βαθμό στην εύρεση κακόβουλου λογισμικού [116].
3. Διευρυμένη αναγνώριση αρχείων. Το Suricata μπορεί να εντοπίσει χιλιάδες είδη αρχείων κατά τη μεταφορά τους μέσω του δικτύου και να λάβει αποφάσεις αναφορικά με περαιτέρω έλεγχό τους [115].
4. Αυξημένη ταχύτητα και αποτελεσματικότητα στην ανάλυση της κίνησης δικτύου. Ως πολυνηματικό εργαλείο (multi - threaded) προσφέρει λεπτομερή και ακριβή αποτελέσματα αναφορικά με την κίνηση στο δίκτυο. Σε συνδυασμό με τη αυξημένη ισχύ που παρέχουν οι σύγχρονοι επεξεργαστές, αποτελεί πολύτιμο εργαλείο ελέγχου [116].

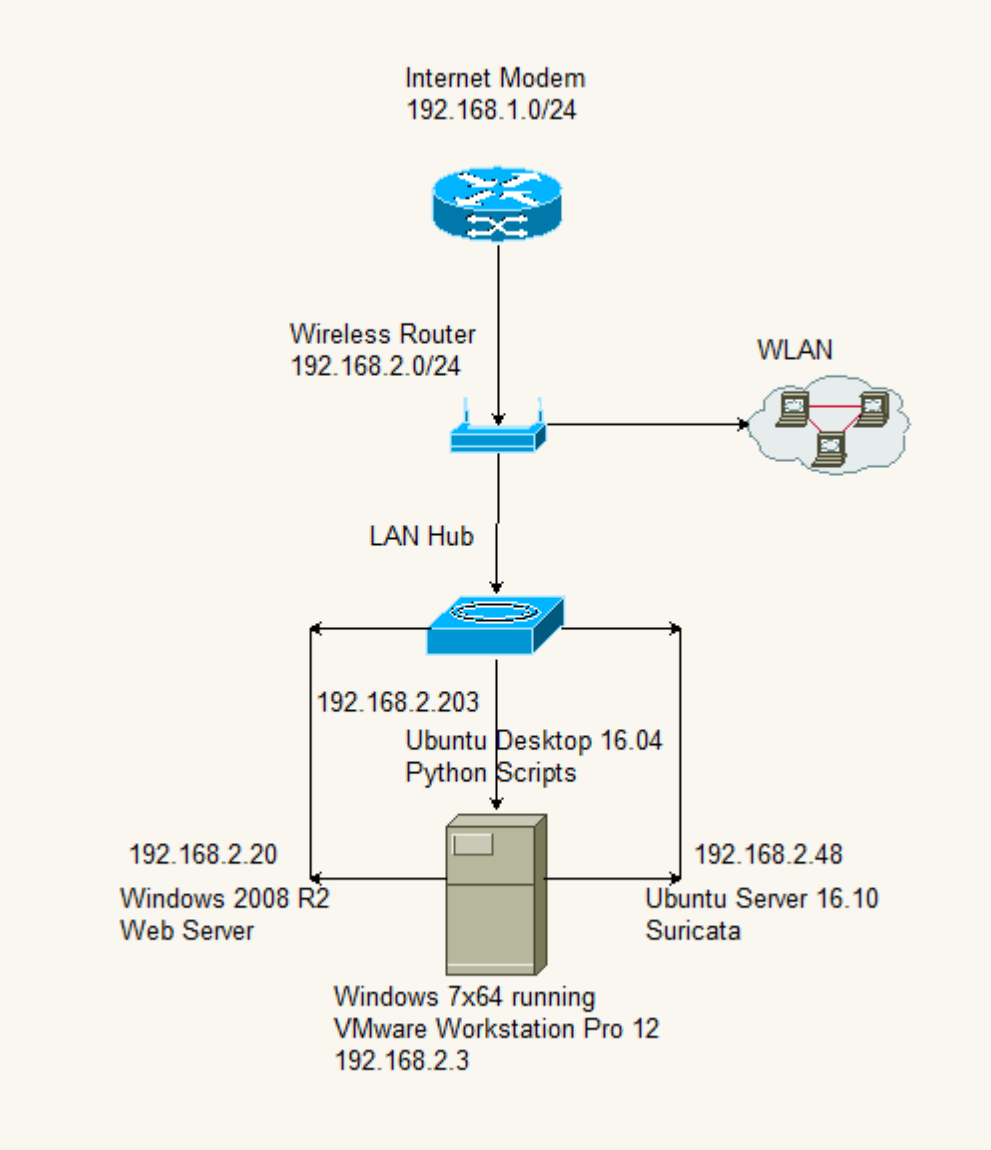
5. Ευκολία χρήσης. Το Suricata συγκαταλέγεται μεταξύ εκείνων των IDS όπου τόσο η εγκατάσταση όσο και η ρύθμιση των επιλογών είναι μια απλή διαδικασία. Η χρήση του δεν επηρεάζει τη λειτουργία των υφιστάμενων υποδομών ενώ μπορεί άμεσα να παρέχει πληροφορίες αναφορικά με κακόβουλη δραστηριότητα.
6. Υποστήριξη όλων των λειτουργικών συστημάτων. Μπορεί να αξιοποιηθεί σε όλα τα λειτουργικά συστήματα (Linux, Free BSD, Open BSD, MacOS/ Mac OS X, Windows) [115].
7. Ταχεία ταυτοποίηση IP. Ο έλεγχος της IP μέσω του Suricata διασφαλίζει τη σύνδεση ενώ σε περιπτώσεις ύποπτης δραστηριότητας δημιουργούνται ειδοποιήσεις [116].

Κεφάλαιο 6
Πειραματική Μελέτη

6.1 Μεθοδολογία

Στα πλαίσια της εργασίας πραγματοποιήθηκαν συγκριτικές μετρήσεις που σχετίζονται με την απόδοση του εργαλείου Suricata σε ένα εσωτερικό δίκτυο, την καθυστέρηση που ενδέχεται να επιφέρει , την ανάγκη σε υπολογιστική ισχύ αλλά και τους κινδύνους που καταγράφονται στα αρχεία καταγραφής του. Λόγω του ότι η αναπαραγωγή δικτυακής κίνησης υψηλού όγκου, αφενός δεν είναι εφικτή σε ένα μικρό τοπικό δίκτυο , αφετέρου απαιτεί ισχυρό και εξειδικευμένο υπολογιστικό και δικτυακό εξοπλισμό , προτιμήθηκε να πραγματοποιηθούν μετρήσεις κάτω από συγκεκριμένες συνθήκες - σενάρια, έτσι ώστε τα αποτελέσματα να είναι όσο τον δυνατόν πιο ακριβή.

Οι δοκιμές πραγματοποιήθηκαν σε ηλεκτρονικό υπολογιστή με επεξεργαστή Intel Core i7 920 2.67 GHz με 12 GB μνήμη RAM και λειτουργικό σύστημα Windows 7 64-bit. Η εγκατάσταση του Suricata καθώς και των διαφόρων εργαλείων που χρησιμοποιήθηκαν για την εκτέλεση των δοκιμών, έγινε σε εικονικές μηχανές που δημιουργήθηκαν σε αυτόν τον υπολογιστή με χρήση της πλατφόρμας Vmware Workstation Pro 12. Η τοπολογία του δικτύου πάνω στο οποίο έγιναν οι δοκιμές περιγράφεται στην εικόνα 6.1 , επισημάνεται δε ότι προτιμήθηκε η χρήση παλαιού τύπου hub (10Mbit/s) , το οποίο επιτρέπει την μετάδοση της κίνησης σε όλες τις πόρτες του. Με αυτόν τον τρόπο το Suricata ήταν σε θέση να εποπτεύσει ολόκληρη την κίνηση του δικτύου. Αξίζει επίσης να σημειωθεί ότι χρησιμοποιήθηκε ξεχωριστή κάρτα δικτύου για κάθε εικονική μηχανή , ενώ όλες ρυθμίστηκαν σε ταχύτητα 10Mbps , και απενεργοποιήθηκε το πρωτόκολλο IPv6. Τέλος , απενεργοποιήθηκε το offloading στην κάρτα δικτύου της εικονικής μηχανής που φιλοξένησε το Suricata όπως προτείνεται και από τον κατασκευαστή[117].



Εικόνα 6.1 : Τοπολογία δικτύου

Οι εικονικές μηχανές που χρησιμοποιήθηκαν είχαν τα εξής χαρακτηριστικά και εξυπηρέτησαν τους παρακάτω σκοπούς στο πείραμα:

• Suricata 3.1.3, σε λειτουργικό σύστημα Ubuntu Server 16.10 Kernel version 4.8.0-22 (2Vcpu's, 2GB Ram)

• Wordpress 4.7.1, σε λειτουργικό σύστημα Windows Server 2008 R2 (2Vcpu's, 4GB Ram)

• Python 2.7.12, σε λειτουργικό σύστημα Ubuntu Desktop 16.04 Kernel version 4.8.0-49 (1Vcpu, 1GB Ram)

Για λόγους αξιοπιστίας της λειτουργίας του τοπικού δικτύου των δοκιμών χρησιμοποιήθηκαν 2 εργαλεία μέτρησης της απόδοσης του , σε επίπεδο TCP. Τα εργαλεία που χρησιμοποιήθηκαν ήταν το D-ITG 2.8.1 [118] και το IPERF 3.1 [119].

Χρησιμοποιώντας τις εντολές ITGSend -a destination_ip -c 1470 -C 850 -t 180000 -T TCP -l Sender.log και iperf3 -c destination_ip -t 180 -p 8999 αντίστοιχα πήραμε τις μετρήσεις που εμφανίζονται στον Πίνακα 6.1.

Network Performance Tool	**Receiver Windows 2008 Server - Sender Ubuntu Desktop (Python Scripts)**	**Receiver Windows 2008 Server - Sender Ubuntu Server (Suricata)**
	Average Bitrate	
D-ITG	8966 Kbit/s	7666 Kbit/s
IPERF	9.10 Mbit/s	8.24 Mbit/s

Πίνακας 6.1 : Μέτρηση απόδοσης δικτύου με D-ITG & IPERF

Στην εικονική μηχανή Windows 2008 R2 εγκαταστάθηκε η πλατφόρμα Wordpress με όλα τα προ-απαιτούμενα , όπως IIS 7.5, PHP 5.6 , MySQL 5.1 με σκοπό την δημιουργία του Web Site (devsite.gr.intranet). Επίσης , στον IIS του Web Server ενεργοποιήθηκε η υπηρεσία FTP (Basic Authentication). Τέλος , η εικονική μηχανή Ubuntu Desktop 16.04 χρησιμοποιήθηκε για την ανάπτυξη και εκτέλεση σεναρίων(scripts) σε γλώσσα python με σκοπό την μέτρηση της απόδοσης του εργαλείου Suricata.

Η εγκατάσταση του εργαλείου Suricata έγινε βάση των ακόλουθων επίσημων οδηγιών του κατασκευαστή [120]. Για την σωστή λειτουργία του Suricata απαραίτητη προϋπόθεση ήταν η εγκατάσταση των παρακάτω πακέτων στο λειτουργικό Ubuntu Server : build-essential, locate, lxterminal , cmake, autoconf, libcanberra-gtk*, automake, libtool libpcap-de, v libnet1-dev , libyaml-0-2, libyaml-dev, zlib1g, zlib1g-dev, libcap-ng-dev, libcap-ng0, make, flex, bison, git, git-core, subversion, libmagic-dev, libgeoip1, libgeoip-dev, libjansson4, libjansson-dev, python-simplejson, libnss3-dev, libnspr4-dev, libnetfilter-queue-dev, libnetfilter-queue1, libnfnetlink-dev, libnfnetlink0. Επίσης , προαπαιτούμενο στοιχείο για την σωστή λειτουργία του Suricata είναι η εγκατάσταση της βιβλιοθήκης PCRE [121] (στην δική μας περίπτωση εγκαταστάθηκε η έκδοση 8.38).

Λόγω του ότι το Suricata στηρίζει την μεθοδολογία του στην ανίχνευση βάση υπογραφών, χρησιμοποιήθηκε η συλλογή υπογραφών της εταιρείας Emerging Threats [122], η οποία είναι διαθέσιμη χωρίς συνδρομή ενώ για την διαχείριση και ανανέωση των υπογραφών χρησιμοποιήθηκε το εργαλείο Oinkmaster [123].

Βασική προϋπόθεση για την σωστή λειτουργία του Suricata, αποτελεί η ενημέρωση του πεδίου HOME_NET με το IP range του δικτύου που εποπτεύεται στο παραμετρικό αρχείο suricata.yaml και παραθέτεται στο Παράρτημα Β (στην δική μας περίπτωση ενημερώθηκε με την τιμή 192.168.2.0/24) ενώ επίσης με την παραμετροποίηση του ίδιου αρχείου ενεργοποιήθηκε η δημιουργία αρχείων καταγραφής για αιτήματα dns, tls, http, καθώς και στατιστικών αναφορικά με το Suricata. Επίσης, στην ενότητα host-os-policy του ίδιου αρχείου, οι IP διευθύνσεις αντιστοιχήθηκαν με τα λειτουργικά συστήματα των υπολογιστών του δικτύου.

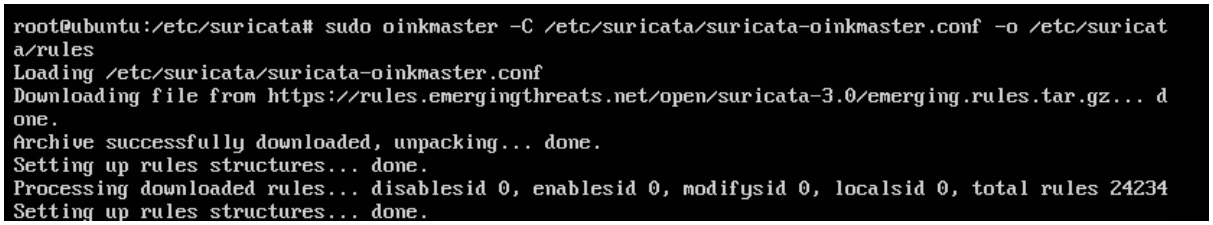
Τα διαθέσιμα αρχεία κανόνων του Suricata που μπορούν να ενεργοποιηθούν-απενεργοποιηθούν, παραμετροποιώντας κατάλληλα το αρχείο suricata.yaml και ταυτόχρονα ενημερώνονται από τα εργαλείο Oinkmaster, παραθέτονται στην εικόνα 6.2.

```
root@ubuntu:/etc/suricata/rules# ls *.rules
app-layer-events.rules          emerging-icmp_info.rules       emerging-telnet.rules
botcc.portgrouped.rules         emerging-icmp.rules            emerging-tftp.rules
botcc.rules                     emerging-imap.rules            emerging-trojan.rules
ciarmy.rules                    emerging-inappropriate.rules   emerging-user_agents.rules
compromised.rules               emerging-info.rules            emerging-voip.rules
decoder-events.rules            emerging-malware.rules         emerging-web_client.rules
dns-events.rules                emerging-misc.rules            emerging-web_server.rules
drop.rules                      emerging-mobile_malware.rules  emerging-web_specific_apps.rules
dshield.rules                   emerging-netbios.rules         emerging-worm.rules
emerging-activex.rules          emerging-p2p.rules             files.rules
emerging-attack_response.rules  emerging-policy.rules          http-events.rules
emerging-chat.rules             emerging-pop3.rules            local.rules
emerging-current_events.rules   emerging-rpc.rules             modbus-events.rules
emerging-deleted.rules          emerging-scada.rules           rbn-malvertisers.rules
emerging-dns.rules              emerging-scan.rules            rbn.rules
emerging-dos.rules              emerging-shellcode.rules       smtp-events.rules
emerging-exploit.rules          emerging-smtp.rules            stream-events.rules
emerging-ftp.rules              emerging-snmp.rules            tls-events.rules
emerging-games.rules            emerging-sql.rules             tor.rules
```

Εικόνα 6.2 : Αρχεία κανόνων Suricata

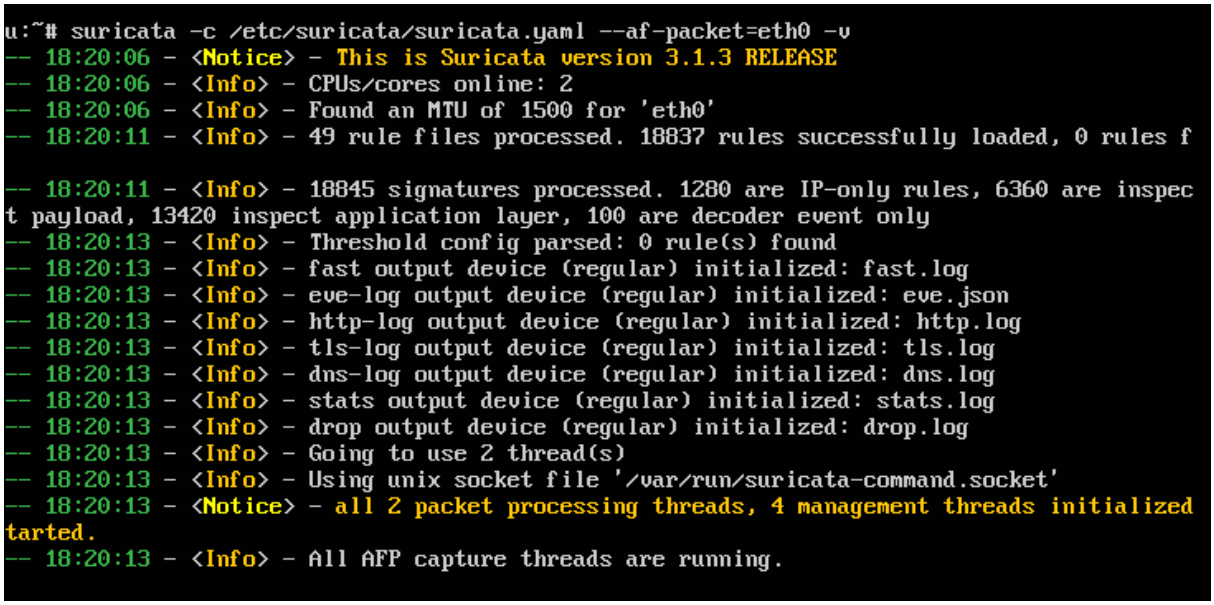
Κατά την διάρκεια των δοκιμών που πραγματοποιήθηκαν ενεργοποιήθηκαν όλα τα παραπάνω αρχεία κανόνων στο παραμετρικό αρχείο suricata.yaml πλην των κανόνων του αρχείου modbus-events.rules. Επίσης, ενεργοποιήθηκαν όλες οι υπογραφές των αρχείωνemerging-attack_response,emerging-dos,emerging-exploit,emerging-ftp,emerging-misc,emerging-policy,emerging-scan,emerging-sql,emerging-web_client,emerging-web_server που σχετίζονται με τα σενάρια που εκτελέσαμε , ο κώδικας των οποίων αναφέρεται στο Παράρτημα Α.

Κατά την ενημέρωση των αρχείων-υπογραφών του Suricata , αναφέρθηκαν 24234 κανόνες σύμφωνα με το εργαλείο Oinkmaster:



Εικόνα 6.3 : Ενημέρωση υπογραφών Suricata με χρήση του εργαλείου Oinkmaster

Όπως φαίνεται στην εικόνα 6.3 κατά την εκκίνηση του Suricata φορτώνονται επιτυχώς 18837 κανόνες.



Εικόνα 6.4 : Εκκίνηση λειτουργίας Suricata

Προκειμένου να επωφεληθούμε από την τεχνολογία multithread, επιλέχθηκε να εκτελέσουμε το Suricata σε περιβάλλον λειτουργίας AF_mode (υποστηρίζεται για πυρήνες Linux > 3.1) , σύμφωνα με το οποίο είναι εφικτό να εποπτευθεί η κίνηση που διοχετεύεται σε μία κάρτα δικτύου από πολλαπλά threads , βελτιώνοντας με αυτόν τον τρόπο την απόδοση του. Παράλληλα , στο παραμετρικό αρχείο suricata.yaml ρυθμίστηκε ο τύπος του cluster να είναι cluster_flow όπου όλα τα πακέτα μιας δεδομένης ροής αποστέλλονται στο ίδιο socket.

6.2 Σενάρια (scripts)

Με σκοπό την μέτρηση της απόδοσης του Suricata αναπτύχθηκαν τα ακόλουθα σενάρια (scripts) σε γλώσσα python. Ο κώδικας των σεναρίων αναφέρεται στο Παράρτημα Α.

Α) Σενάριο προσπέλασης ιστοσελίδας.

Το συγκεκριμένο σενάριο χρησιμοποιώντας την βιβλιοθήκη requests της γλώσσας Python [124] μπορεί να εκτελέσει x προσπάθειες προσπέλασης μιας δοσμένης ιστοσελίδας.

Β) Σενάριο σύνδεσης διαχειριστή σε ιστοσελίδα Wordpress

Το σενάριο αυτό χρησιμοποιώντας πάλι την βιβλιοθήκη requests της γλώσσας Python [124] μπορεί να εκτελέσει x προσπάθειες σύνδεσης στην σελίδα του διαχειριστή μιας δοσμένης ιστοσελίδας Wordpress.

Γ) Σενάριο μεταφόρτωσης αρχείου μέσω FTP

Το εν λόγω σενάριο πραγματοποιήθηκε χρησιμοποιώντας την βιβλιοθήκη ftplib της γλώσσας Python [125] μπορεί να εκτελέσει x προσπάθειες σύνδεσης σε FTP Server , διαγραφή του αρχείου file.txt μετά από επιτυχημένη ανεύρεσή του , μεταφόρτωση εκ νέου του ίδιου αρχείου και εκτύπωση λίστας των αρχείων καταλόγου του FTP Server. Κατά την διάρκεια των δοκιμών το αρχείο file.txt είχε μέγεθος 10MB και έχει φτιαχτεί πριν την εκτέλεση του σεναρίου με την χρήση της εντολής dd if=/dev/urandom of=file.txt bs=1048576 count=10 σε περιβάλλον υπολογιστή με λειτουργικό σύστημα Linux.

Δ) Σενάριο απομακρυσμένης σύνδεσης σε βάση MySQL και απεικόνισης των πινάκων της.

Το παραπάνω σενάριο πραγματοποιήθηκε χρησιμοποιώντας την βιβλιοθήκη MySQLdb της Python [126] και μπορεί να εκτελέσει x προσπάθειες απομακρυσμένης σύνδεσης σε βάση MySQL και απεικόνισης των πινάκων της.

Όλα τα σενάρια αποθηκεύουν σε αρχείο κειμένου την ώρα που πραγματοποιήθηκε το αίτημα , υπολογίζουν τον χρόνο εκτέλεσης κάθε αιτήματος , τον μέσο όρο εκτέλεσης για x αιτήματα , την απόκλιση και το συνολικό χρόνο εκτέλεσης του script. Επίσης, με την κατάλληλη τροποποίηση των παραπάνω script, που προορίζονται για επιτυχημένες προσπάθειες προσπέλασης υπηρεσιών HTTP Browse, HTTP Login, FTP, MySQL

καταφέραμε να δημιουργήσουμε τέσσερα ακόμα script που έχουν σαν σκοπό να παράγουν αποτυχημένες προσπάθειες.

Κατά την διάρκεια των δοκιμών τα scripts εκτελέστηκαν α) μεμονωμένα, β) χωρισμένα σε τετράδες (επιτυχημένες-αποτυχημένες προσπάθειες) γ) όλα τα scripts παράλληλα τόσο με απενεργοποιημένο όσο και με ενεργοποιημένο το Suricata στο τοπικό δίκτυο, με σκοπό να ελέγξουμε τόσο την λειτουργία όσο και την απόδοση του Suricata καθώς και εάν επιφέρει κάποια επιβάρυνση-καθυστέρηση στις υπηρεσίες που προσφέρονται στο τοπικό δίκτυο γενικότερα. Τα στατιστικά των σεναρίων παραθέτονται στους πίνακες του κεφαλαίου 6.3.1.

Παράλληλα με την εκτέλεση των σεναρίων, με την χρήση του εργαλείου tcpdump [127] δημιουργήθηκαν αρχεία καταγραφής cap στην εικονική μηχανή που εκτελέστηκαν τα σενάρια με σκοπό την καταγραφή των πακέτων για περαιτέρω ανάλυση. Για τα στατιστικά των αρχείων καταγραφής cap που παραθέτονται στους πίνακες του κεφαλαίου 6.3.1 χρησιμοποιήθηκε το εργαλείο capinfos [128].

Τα στοιχεία που παραθέτονται στους πίνακες του κεφαλαίου 6.3.2 , περιλαμβάνουν τα σημαντικότερα από το αρχείο καταγραφής stats.log του Suricata. Συγκεκριμένα , αυτά είναι τα παρακάτω : Runtime (χρόνος εκτέλεσης του Suricata) , capture.kernel_packets (αριθμός πακέτων που εποπτεύθηκαν), decoder.pkts (αριθμός πακέτων που αποκωδικοποιήθηκαν), decoder.avg_pkt_size (μέσος όρος του μεγέθους του πακέτου), tcp.invalid_checksum (αριθμός πακέτων που παρουσίασαν λανθασμένο checksum), tcp.sessions (αριθμός των συνεδριών tcp), tcp.reassembly_gap (αριθμός χαμένων πακέτων στην ροή TCP), detect.alert (αριθμός εντοπισμένων κινδύνων). Επίσης , παρουσιάζονται οι εντοπισμένοι κίνδυνοι από το αρχείο καταγραφής (fast.log).

Για τις γραφικές παραστάσεις που αποτυπώθηκαν στους πίνακες του κεφαλαίου 6.3.2 και περιλαμβάνουν τους σημαντικότερους δείκτες των στατιστικών χρησιμοποιήθηκε το εργαλείο ανοικτού κώδικα suri-stats [129].

6.3 Αποτελέσματα Δοκιμών

6.3.1 Στατιστικά χρόνων σεναρίων – Στατιστικά χρόνων αρχείων καταγραφής

α)Εκτέλεση των σεναρίων μεμονωμένα

Suricata Απενεργοποιημένο							
Στατιστικά σεναρίου				Στατιστικά αρχείου καταγραφής cap			
Σενάριο – επιτυχημένες/αποτυχημένες αιτήσεις	Μέσος χρόνος ανά αίτημα (sec)	Απόκλιση	Χρόνος σεναρίου (sec)	Αριθμός πακέτων	Ρυθμός μετάδοσης δεδομένων (bytes/sec)	Μέσος όρος μεγέθους πακέτου (bytes)	Μέσος όρος ρυθμού μετάδοσης πακέτου (packets/sec)
Α- 500 επιτυχημένες αιτήσεις	1,257	0,022	629	12681	1634	81,07	20
Α- 30Κ αποτυχημένες αιτήσεις	0,0076	0,001	229	180001	9000	92,17	787
Β- 150 επιτυχημένες αιτήσεις	2,58	0,18	387	4126	1155	108,52	10
Β- 150 αποτυχημένες αιτήσεις	3,56	0,02	534	1926	581	161,5	3
Δ- 150 επιτυχημένες αιτήσεις	2,6	0,006	391	1650	330	78,26	4
Δ- 150 αποτυχημένες αιτήσεις	2,6	0,005	391	902	181	78,45	2
Γ- 150 επιτυχημένες αιτήσεις	9,3	0,01	1396	1091401	147250	1507,16	781
Γ- 30Κ αποτυχημένες αιτήσεις	0,0037	0,0041	112	210003	16500	70,51	1875
Σύνολα			**4069**	**1502690**	**176631**	**2177,64**	**3482**

Πίνακας 6.2 : Χρόνοι εκτέλεσης , Στατιστικά αρχείου cap – Suricata απενεργοποιημένο – Εκτέλεση σεναρίων μεμονωμένα

Suricata Ενεργοποιημένο							
Στατιστικά σεναρίου				Στατιστικά αρχείου καταγραφής cap			
Σενάριο – επιτυχημένες/αποτυχημένες αιτήσεις	Μέσος χρόνος ανά αίτημα (sec)	Απόκλιση	Χρόνος σεναρίου (sec)	Αριθμός πακέτων	Ρυθμός μετάδοσης δεδομένων (bytes/sec)	Μέσος όρος μεγέθους πακέτου (bytes)	Μέσος όρος ρυθμού μετάδοσης πακέτου (packets/sec)
Α- 500 επιτυχημένες αιτήσεις	1,262	0,022	631,5	12729	1645	81,61	20
Α- 30Κ αποτυχημένες αιτήσεις	0,0077	0,001	232	180001	8875	92,17	776
Β- 150 επιτυχημένες αιτήσεις	2,59	0,07	388	4118	1143	107,76	10
Β- 150 αποτυχημένες αιτήσεις	3,57	0,025	535,5	1909	566	159,09	3
Δ- 150 επιτυχημένες αιτήσεις	2,6	0,008	391	1650	330	78,26	4
Δ- 150 αποτυχημένες αιτήσεις	2,6	0,0014	391	900	180	78,47	2
Γ- 150 επιτυχημένες αιτήσεις	9,32	0,01	1398,4	1091400	147000	1507,17	780
Γ- 30Κ αποτυχημένες αιτήσεις	0,0038	0,0015	116	210011	15875	70,52	1810
Σύνολα			**4083,4**	**1502718**	**175614**	**2175,05**	**3405**

Πίνακας 6.3 : Χρόνοι εκτέλεσης , Στατιστικά αρχείου cap – Suricata Ενεργοποιημένο – Εκτέλεση μεμονωμένων σεναρίων

Συγκρίνοντας τους δύο πίνακες διαπιστώνουμε ότι τόσο ο μέσος χρόνος ανά αίτημα όσο και ο χρόνος του κάθε σεναρίου διαφοροποιείται ελάχιστα ενεργοποιώντας το Suricata. Συγκεκριμένα, μέγιστη διαφορά στον μέσο όρο ανά αίτημα παρουσιάζει το σενάριο Β (150 αποτυχημένες αιτήσεις) με 0,01 sec ενώ μεγαλύτερη διαφορά στον συνολικό χρόνο εκτέλεσης παρουσιάζει το σενάριο Γ (30Κ αποτυχημένες αιτήσεις) με 4 sec.

Παρομοίως, οι ρυθμοί μετάδοσης δεδομένων και ο μέσος όρος ρυθμού μετάδοσης των πακέτων παρουσιάζουν ελάχιστες διαφορές. Μεγαλύτερη διαφορά παρουσιάζει το σενάριο Γ (30Κ αποτυχημένες αιτήσεις) όπου παρουσιάζεται διαφορά 625 bytes/sec στον ρυθμό μετάδοσης των δεδομένων και 65 packets/sec ενεργοποιώντας το Suricata.

β) Εκτέλεση των σεναρίων χωρισμένα σε τετράδες, επιτυχημένες-αποτυχημένες προσπάθειες.

Suricata Απενεργοποιημένο							
Στατιστικά σεναρίου				Στατιστικά αρχείου καταγραφής cap			
Σενάριο – επιτυχημένες/αποτυχημένες αιτήσεις	Μέσος χρόνος ανά αίτημα (sec)	Απόκλιση	Χρόνος σεναρίου (sec)	Αριθμός πακέτων	Ρυθμός μετάδοσης δεδομένων (bytes/sec)	Μέσος όρος μεγέθους πακέτου (bytes)	Μέσος όρος ρυθμού μετάδοσης πακέτου (packets/sec)
Α- 30Κ αποτυχημένες αιτήσεις	0,0078	0,0066	235				
Δ- 150 αποτυχημένες αιτήσεις	2,6	0,012	391				
Γ- 30Κ αποτυχημένες αιτήσεις	0,0041	0,002	125				
Β- 150 αποτυχημένες αιτήσεις	3,59	0,54	539	392809	7250	80,71	725
Δ- 150 επιτυχημένες αιτήσεις	2,7	0,067	406				
Α- 500 επιτυχημένες αιτήσεις	1,34	0,04	670				
Β- 150 επιτυχημένες αιτήσεις	2,72	0,064	408				
Γ- 150 επιτυχημένες αιτήσεις	9,61	0,37	1441	1109846	142500	1483,55	768
Σύνολα			**4215**	**1502655**			

Πίνακας 6.4 : Χρόνοι εκτέλεσης , Στατιστικά αρχείου cap – Suricata απενεργοποιημένο – Εκτέλεση σεναρίων χωρισμένα σε τετράδες

Suricata Ενεργοποιημένο							
Στατιστικά σεναρίου				Στατιστικά αρχείου καταγραφής cap			
Σενάριο – επιτυχημένες/αποτυχημένες αιτήσεις	Μέσος χρόνος ανά αίτημα (sec)	Απόκλιση	Χρόνος σεναρίου (sec)	Αριθμός πακέτων	Ρυθμός μετάδοσης δεδομένων (bytes/sec)	Μέσος όρος μεγέθους πακέτου (bytes)	Μέσος όρος ρυθμού μετάδοσης πακέτου (packets/sec)
Α- 30Κ αποτυχημένες αιτήσεις	0,0078	0,002	235				
Δ- 150 αποτυχημένες αιτήσεις	2,6	0,03	391				
Γ- 30Κ αποτυχημένες αιτήσεις	0,0059	0,003	179				
Β- 150 αποτυχημένες αιτήσεις	3,59	0,09	539	392873	7250	80,89	721
Δ- 150 επιτυχημένες αιτήσεις	2,71	0,06	407				
Α- 500 επιτυχημένες αιτήσεις	1,34	0,03	672				
Β- 150 επιτυχημένες αιτήσεις	2,73	0,08	409				
Γ- 150 επιτυχημένες αιτήσεις	9,62	0,35	1443	1109858	142450	1483,54	767
Σύνολα			**4275**	**1502731**			

Πίνακας 6.5 : Χρόνοι εκτέλεσης , Στατιστικά αρχείου cap – Suricata ενεργοποιημένο – Εκτέλεση σεναρίων χωρισμένα σε τετράδες

Κατά την παραπάνω περίπτωση εκτέλεσης των σεναρίων διακρίνεται εύκολα ότι οι περισσότερες τιμές είναι παραπλήσιες. Διαφοροποίηση στον μέσο όρο ανά αίτημα παρουσιάζεται στο σενάριο Γ (150 επιτυχημένες αιτήσεις) με 0,01 sec ενώ ο συνολικός χρόνος εκτέλεσης του script παρουσιάζει μέγιστη διαφορά της τάξεως των 54 sec για το σενάριο Γ (30Κ αποτυχημένες αιτήσεις) με το Suricata ενεργοποιημένο.

Ο ρυθμός μετάδοσης των δεδομένων εμφανίζεται μικρότερος κατά 50 bytes/sec κατά την εκτέλεση των σεναρίων με επιτυχημένες αιτήσεις ενώ ο μέσος όρος του ρυθμού μετάδοσης των πακέτων παρουσιάζει διαφορά 4 πακέτων/sec κατά την εκτέλεση των σεναρίων με αποτυχημένες αιτήσεις με το Suricata ενεργοποιημένο.

γ) Εκτέλεση των σεναρίων παράλληλα

Suricata Απενεργοποιημένο							
Στατιστικά σεναρίου				Στατιστικά αρχείου καταγραφής cap			
Σενάριο – επιτυχημένες/αποτυχημένες αιτήσεις	Μέσος χρόνος ανά αίτημα (sec)	Απόκλιση	Χρόνος σεναρίου (sec)	Αριθμός πακέτων	Ρυθμός μετάδοσης δεδομένων (bytes/sec)	Μέσος όρος μεγέθους πακέτου (bytes)	Μέσος όρος ρυθμού μετάδοσης πακέτου (packets/sec)
Α- 500 επιτυχημένες αιτήσεις	1,357	0,045	678,6				
Α- 30Κ αποτυχημένες αιτήσεις	0,03	0,012	925				
Β- 150 επιτυχημένες αιτήσεις	2,73	0,08	409				
Β- 150 αποτυχημένες αιτήσεις	3,64	0,08	546				
Δ- 150 επιτυχημένες αιτήσεις	5,23	0,6	784				
Δ- 150 αποτυχημένες αιτήσεις	5,18	0,5	778				
Γ- 150 επιτυχημένες αιτήσεις	10,46	0,83	1569				
Γ- 30Κ αποτυχημένες αιτήσεις	0,049	0,02	1461				
Σύνολα			**7150,6**	**1526162**	**133875**	**1100,8**	**972**

Πίνακας 6.6 : Χρόνοι εκτέλεσης , Στατιστικά αρχείου cap – Suricata απενεργοποιημένο – Εκτέλεση σεναρίων παράλληλα

Suricata Ενεργοποιημένο							
Στατιστικά σεναρίου				Στατιστικά αρχείου καταγραφής cap			
Σενάριο – επιτυχημένες/αποτυχημένες αιτήσεις	Μέσος χρόνος ανά αίτημα (sec)	Απόκλιση	Χρόνος σεναρίου (sec)	Αριθμός πακέτων	Ρυθμός μετάδοσης δεδομένων (bytes/sec)	Μέσος όρος μεγέθους πακέτου (bytes)	Μέσος όρος ρυθμού μετάδοσης πακέτου (packets/sec)
Α- 500 επιτυχημένες αιτήσεις	1,39	0,072	694				
Α- 30Κ αποτυχημένες αιτήσεις	0,03	0,015	938				
Β- 150 επιτυχημένες αιτήσεις	2,78	0,1	417				
Β- 150 αποτυχημένες αιτήσεις	3,7	0,08	556				
Δ- 150 επιτυχημένες αιτήσεις	5,18	0,46	777				
Δ- 150 αποτυχημένες αιτήσεις	5,18	0,48	777				
Γ- 150 επιτυχημένες αιτήσεις	10,73	0,95	1610				
Γ- 30Κ αποτυχημένες αιτήσεις	0,049	0,02	1470				
Σύνολα			**7239**	**1525876**	**129500**	**1101**	**941**

Πίνακας 6.7 : Χρόνοι εκτέλεσης , Στατιστικά αρχείου cap – Suricata ενεργοποιημένο – Εκτέλεση σεναρίων παράλληλα

Εκτελώντας τα σενάρια παράλληλα με το Suricata ενεργοποιημένο, παρατηρείται διαφοροποίηση στα σενάρια που πραγματοποιούν επιτυχημένες αιτήσεις τόσο στον μέσο χρόνο ανά αίτημα όσο και στο συνολικό χρόνο σεναρίου. Συγκεκριμένα,το σενάριο Α (500 αιτήσεις) παρουσιάζει αύξηση 0,033 sec , το σενάριο Β (150 αιτήσεις) 0,05 sec και το σενάριο Γ (150 αιτήσεις) 0,27 sec στον μέσο χρόνο ανά αίτημα ενώ οι συνολικοί χρόνοι σεναρίων διαφοροποιούνται κατά 15,4 sec - σενάριο Α , 8 sec - σενάριο Β & 41 sec – σενάριο Γ. Τέλος , ο ρυθμός μετάδοσης των δεδομένων μειώνεται κατά 4375 bytes/sec ενώ ο μέσος όρος ρυθμού μετάδοσης πακέτων κατά 31 packets / sec με το Suricata ενεργοποιημένο.

6.3.2 Στατιστικά Suricata – Αρχείο καταγραφής fast.log

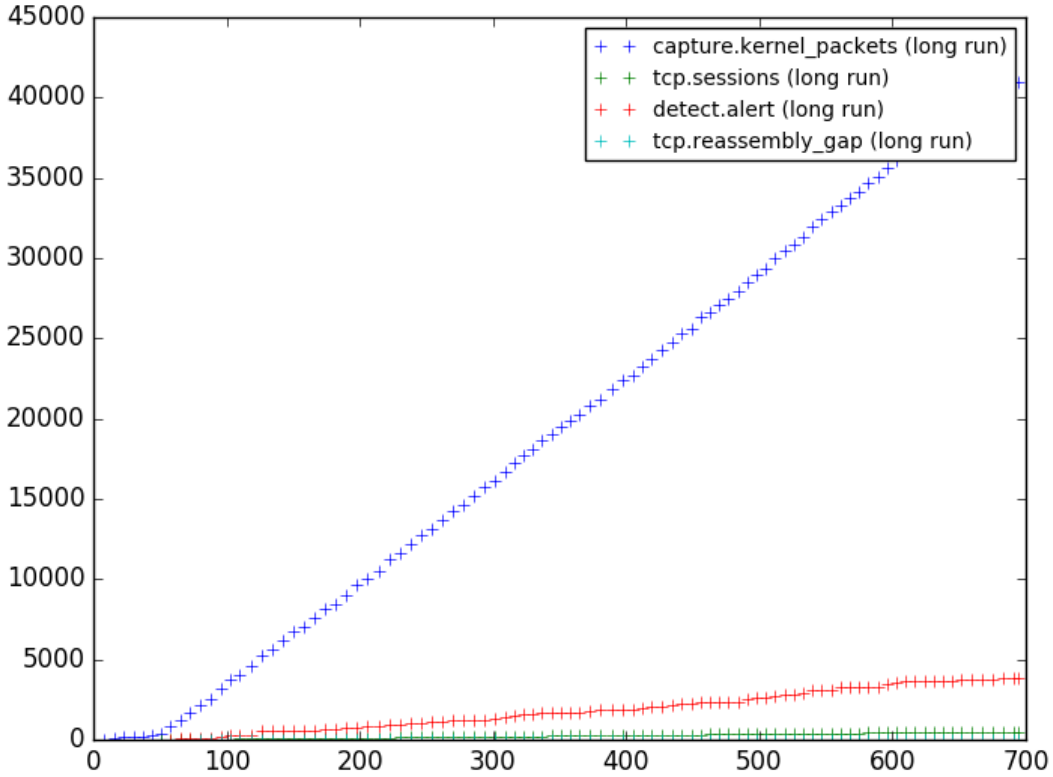
α) Εκτέλεση των σεναρίων μεμονωμένα

Μετρητής	Τιμή
Runtime	**11λ 34δ**
capture.kernel_packets	**40939**
decoder.pkts	**40939**
decoder.avg_pkt_size	**759**
tcp.invalid_checksum	**0**
tcp.sessions	**502**
tcp.reassembly_gap	**43**
detect.alert	**3847**

Πίνακας 6.8 : Στατιστικά εκτέλεσης Suricata – Σενάριο Α 500 επιτυχημένες αιτήσεις

Διαδικασία Suricata	
Μέγιστο % CPU	Μέγιστο % Memory
1.7	**18.6**

Πίνακας 6.9 : Μέγιστη χρήση επεξεργαστή – μνήμες από το Suricata Σενάριο Α 500 επιτυχημένες αιτήσεις



Εικόνα 6.5 : Γραφική παράσταση στατιστικών – Σενάριο Α 500 επιτυχημένες αιτήσεις

Αρχείο καταγραφής fast.log (Πληθέστεροι εντοπισμένοι κίνδυνοι)

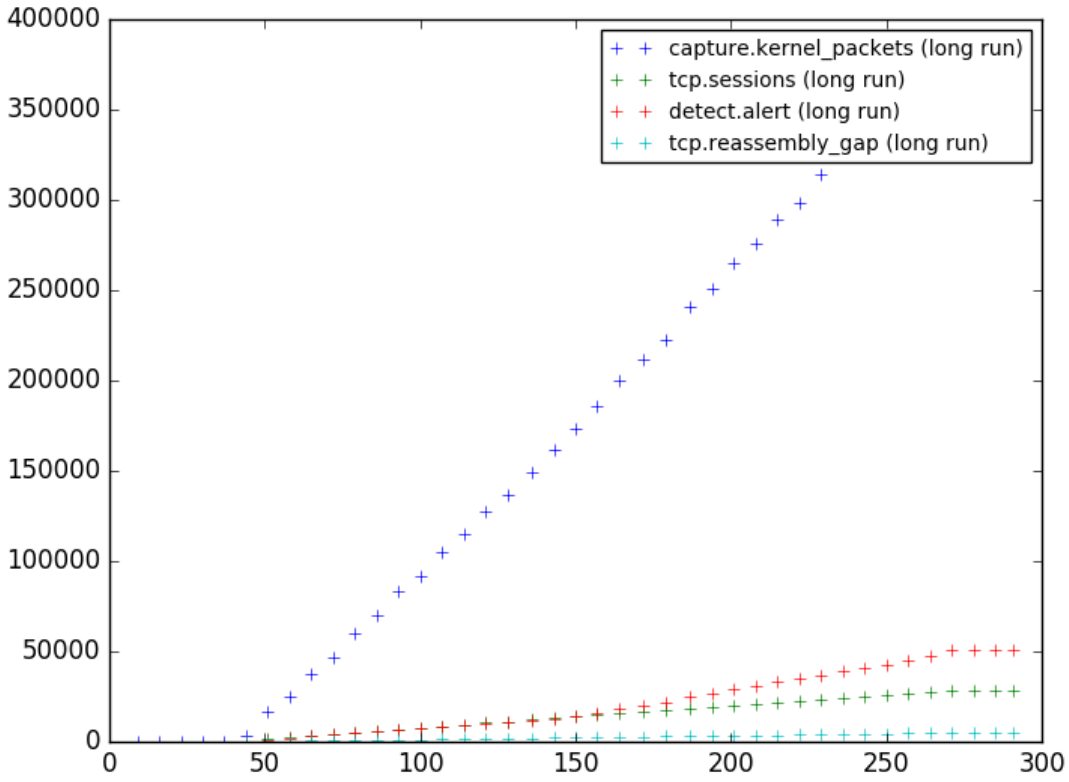
04/29/2017-21:49:31.314509 [**] [1:2210010:2] SURICATA STREAM 3way handshake wrong seq wrong ack [**] [Classification: Generic Protocol Command Decode] [Priority: 3] {TCP} 192.168.2.203:34058 -> 192.168.2.20:80 **Πλήθος 1257**
04/29/2017-21:49:31.315385 [**] [1:2210000:2] SURICATA STREAM 3way handshake with ack in wrong dir [**] [Classification: Generic Protocol Command Decode] [Priority: 3] {TCP} 192.168.2.20:80 -> 192.168.2.203:34058 **Πλήθος 2320**

Όπως φαίνεται από τον Πίνακα 6.7 κατά την εκτέλεση του συγκεκριμένου σεναρίου το Suricata ήταν σε θέση να εποπτεύσει και να αποκωδικοποιήσει το σύνολο των πακέτων 40939 και των συνεδριών TCP 502 , ενώ ο δείκτης tcp.reassembly_gap παρέμεινε σε χαμηλά επίπεδα 43. Η διαδικασία του Suricata κυμάνθηκε σε αρκετά χαμηλά επίπεδα επεξεργαστή-μνήμης ενώ το σύνολο των κίνδυνων που καταγράφηκαν ανήκουν στην κατηγορία stream-events , που περιλαμβάνει υπογραφές για την ροή του

πρωτοκόλλου TCP. Οι 500 επιτυχημένες αιτήσεις HTTP δεν αντιστοιχήθηκαν και δεν ενεργοποίησαν κάποια υπογραφή άλλης κατηγορίας του Suricata π.χ emerging-web_server ή emerging_scan.

Μετρητής	Τιμή
Runtime	**04λ 51δ**
capture.kernel_packets	**387416**
decoder.pkts	**387416**
decoder.avg_pkt_size	**227**
tcp.invalid_checksum	**5**
tcp.sessions	**27996**
tcp.reassembly_gap	**4877**
detect.alert	**50861**

Πίνακας 6.10 : Στατιστικά εκτέλεσης Suricata – Σενάριο Α 30Κ αποτυχημένες αιτήσεις



Εικόνα 6.6 : Γραφική παράσταση στατιστικών – Σενάριο Α 30Κ αποτυχημένες αιτήσεις

Διαδικασία Suricata	
Μέγιστο % CPU	Μέγιστο % Memory
18.6	**23.1**

Πίνακας 6.11 : Μέγιστη χρήση μνήμης – επεξεργαστή από το Suricata Σενάριο Α 30Κ αποτυχημένες αιτήσεις

Αρχείο καταγραφής fast.log (Πληθέστεροι εντοπισμένοι κίνδυνοι)

04/29/2017-22:08:19.909739 [**] [1:2101201:10] GPL WEB_SERVER 403 Forbidden [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 192.168.2.20:80 -> 192.168.2.203:35046 **Πλήθος 19699**
04/29/2017-22:08:19.925670 [**] [1:2210016:2] SURICATA STREAM CLOSEWAIT FIN out of window [**] [Classification: Generic Protocol Command Decode] [Priority: 3] {TCP} 192.168.2.20:80 -> 192.168.2.203:35050 **Πλήθος 3806**
04/29/2017-22:08:19.941909 [**] [1:2210032:2] SURICATA STREAM FIN1 FIN with wrong seq [**] [Classification: Generic Protocol Command Decode] [Priority: 3] {TCP} 192.168.2.203:35052 -> 192.168.2.20:80 **Πλήθος 3136**
04/29/2017-22:08:20.098163 [**] [1:2009749:4] ET SCAN Unusually Fast 403 Error Messages, Possible Web Application Scan [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 192.168.2.20:80 -> 192.168.2.203:35090 **Πλήθος 1188**
04/29/2017-22:10:08.281795 [**] [1:2210045:2] SURICATA STREAM Packet with invalid ack [**] [Classification: Generic Protocol Command Decode] [Priority: 3] {TCP} 192.168.2.203:35052 -> 192.168.2.20:80 Πλήθος **10015**

Όπως φαίνεται από τον Πίνακα 6.9 κατά την εκτέλεση του συγκεκριμένου σεναρίου το Suricata ήταν σε θέση να εποπτεύσει και να αποκωδικοποιήσει το σύνολο των πακέτων 387416 , οι συνεδρίες TCP έφθασαν τις 27996 , ενώ ο δείκτης tcp.reassembly_gap έφθασε την τιμή του 4877.

Η διαδικασία του Suricata κυμάνθηκε σε μέτρια επίπεδα επεξεργαστή-μνήμης ενώ οι κίνδυνοι που καταγράφηκαν ανήκουν α) στην κατηγορία emerging-web_server “GPL WEB SERVER 403” που περιλαμβάνει υπογραφές για αδυναμίες και επιθέσεις Web

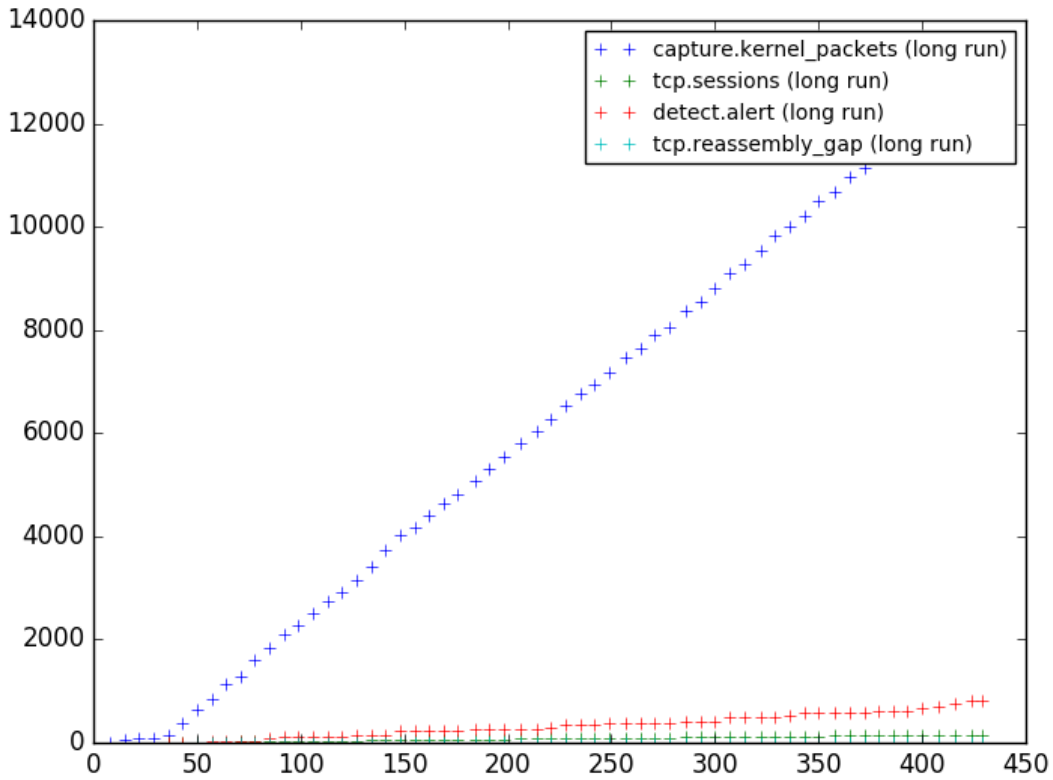
Server β) στην κατηγορία emerging-scan “ET SCAN Unusually Fast 403 Error Messages” που περιλαμβάνει υπογραφές για τον εντοπισμό επαναλαμβανόμενων σαρώσεων πορτών/υπηρεσιών και γ) stream-events “SURICATA STREAM CLOSEWAIT FIN out of window” , “SURICATA STREAM Packet with invalid ack” , “SURICATA STREAM FIN1 FIN with wrong seq” που περιλαμβάνει υπογραφές για την ροή του πρωτοκόλλου TCP.

Μετρητής	Τιμή
Runtime	**07λ 09δ**
capture.kernel_packets	**12937**
decoder.pkts	**12937**
decoder.avg_pkt_size	**759**
tcp.invalid_checksum	**0**
tcp.sessions	**157**
tcp.reassembly_gap	**4**
detect.alert	**821**

Πίνακας 6.12 : Στατιστικά εκτέλεσης Suricata – Σενάριο Β 150 επιτυχημένες αιτήσεις

Διαδικασία Suricata	
Μέγιστο % CPU	Μέγιστο % Memory
1.3	**18.2**

Πίνακας 6.13 : Μέγιστη χρήση μνήμης – επεξεργαστή από το Suricata Σενάριο Β 150 επιτυχημένες αιτήσεις



Εικόνα 6.7 : Γραφική παράσταση στατιστικών – Σενάριο Β 150 επιτυχημένες αιτήσεις

Αρχείο καταγραφής fast.log (Πληθέστεροι εντοπισμένοι κίνδυνοι)

04/29/2017-22:34:24.734348 [**] [1:2012843:3] ET POLICY Cleartext WordPress Login [**] [Classification: Potential Corporate Privacy Violation] [Priority: 1] {TCP} 192.168.2.203:38580 -> 192.168.2.20:80 **Πλήθος 125**

04/29/2017-22:34:24.734348 [**] [1:2012888:3] ET POLICY Http Client Body contains pwd= in cleartext [**] [Classification: Potential Corporate Privacy Violation] [Priority: 1] {TCP} 192.168.2.203:38580 -> 192.168.2.20:80 **Πλήθος 125**

04/29/2017-22:34:40.782247 [**] [1:2014020:4] ET WEB_SERVER Wordpress Login Bruteforcing Detected [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 192.168.2.203:38592 -> 192.168.2.20:80 **Πλήθος 6**

04/29/2017-22:35:10.523193 [**] [1:2210000:2] SURICATA STREAM 3way handshake with ack in wrong dir [**] [Classification: Generic Protocol Command Decode] [Priority: 3] {TCP} 192.168.2.20:80 -> 192.168.2.203:38614 **Πλήθος 324**

04/29/2017-22:35:10.523448 [**] [1:2210010:2] SURICATA STREAM 3way handshake wrong seq wrong ack [**] [Classification: Generic Protocol Command Decode] [Priority: 3] {TCP} 192.168.2.203:38614 -> 192.168.2.20:80 **Πλήθος 186**

Όπως φαίνεται από τον Πίνακα 6.11 κατά την εκτέλεση του συγκεκριμένου σεναρίου το Suricata ήταν σε θέση να εποπτεύσει και να αποκωδικοποιήσει το σύνολο των πακέτων 12937 , οι συνεδρίες TCP έφθασαν τις 157 , ενώ ο δείκτης tcp.reassembly_gap είχε μόλις την τιμή 4.

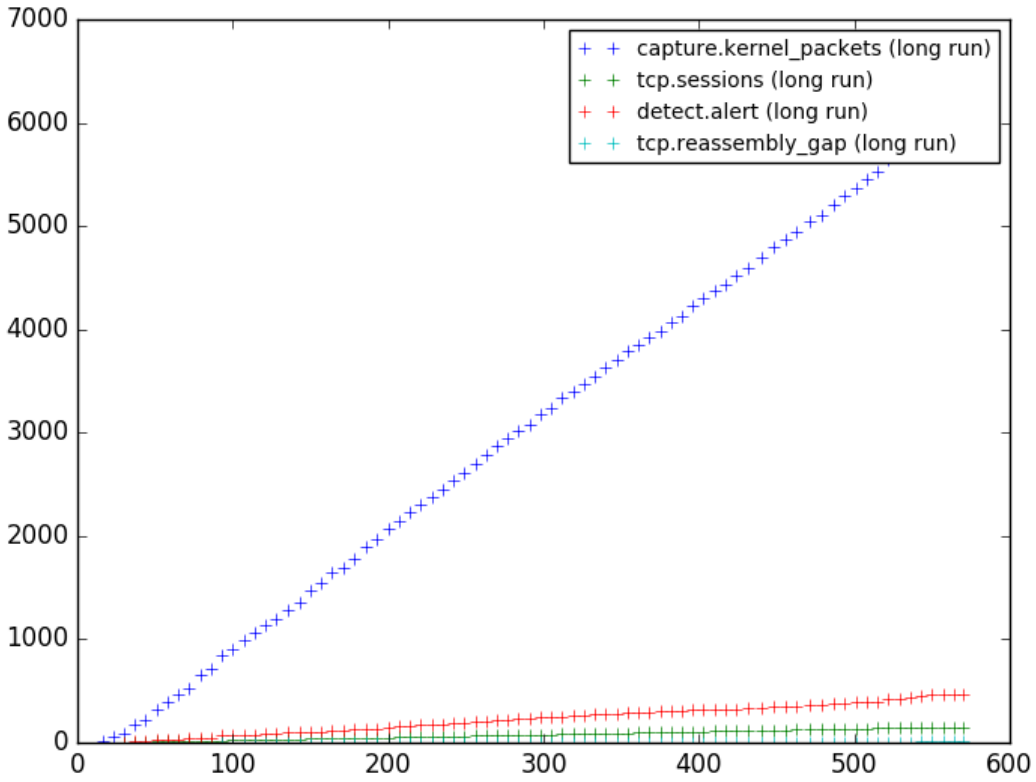
Η διαδικασία του Suricata κυμάνθηκε σε χαμηλά επίπεδα επεξεργαστή-μνήμης ενώ οι κίνδυνοι που καταγράφηκαν ανήκουν α) στην κατηγορία emerging-web_server "ET WEB_SERVER Wordpress Login Bruteforcing Detected" που περιλαμβάνει υπογραφές για αδυναμίες και επιθέσεις Web Server β) στην κατηγορία emerging-policy "ET POLICY Cleartext WordPress Login" , " ET POLICY Http Client Body contains pwd= in cleartext" που περιλαμβάνει υπογραφές σχετικές με την standard πολιτική ασφαλείας και γ) stream-events "SURICATA STREAM 3way handshake wrong seq wrong ack" , "SURICATA STREAM 3way handshake with ack in wrong dir" που περιλαμβάνει υπογραφές για την ροή του πρωτοκόλλου TCP.

Μετρητής	Τιμή
Runtime	**09λ 30δ**
capture.kernel_packets	**6116**
decoder.pkts	**6116**
decoder.avg_pkt_size	**394**
tcp.invalid_checksum	**0**
tcp.sessions	**151**
tcp.reassembly_gap	**8**
detect.alert	**470**

Πίνακας 6.14 : Στατιστικά εκτέλεσης Suricata – Σενάριο Β 150 αποτυχημένες αιτήσεις

Διαδικασία Suricata	
Μέγιστο % CPU	Μέγιστο % Memory
1.0	**17.8**

Πίνακας 6.15 : Μέγιστη χρήση μνήμης – επεξεργαστή από το Suricata Σενάριο Β 150 αποτυχημένες αιτήσεις



Εικόνα 6.8 : Γραφική παράσταση στατιστικών – Σενάριο Β 150 αποτυχημένες αιτήσεις

fast.log (Πληθέστεροι εντοπισμένοι κίνδυνοι)

04/29/2017-22:46:44.790507 [**] [1:2012843:3] ET POLICY Cleartext WordPress Login [**] [Classification: Potential Corporate Privacy Violation] [Priority: 1] {TCP} 192.168.2.203:38880 -> 192.168.2.20:80 **Πλήθος 121**

04/29/2017-22:46:44.790507 [**] [1:2012888:3] ET POLICY Http Client Body contains pwd= in cleartext [**] [Classification: Potential Corporate Privacy Violation] [Priority: 1] {TCP} 192.168.2.203:38880 -> 192.168.2.20:80 **Πλήθος 121**

04/29/2017-22:46:47.172950 [**] [1:2012997:4] ET WEB_SERVER PHP Possible http Remote File Inclusion Attempt [**] [Classification: Web Application Attack] [Priority: 1] {TCP} 192.168.2.203:38880 -> 192.168.2.20:80 **Πλήθος 121**

04/29/2017-22:47:44.503151 [**] [1:2210000:2] SURICATA STREAM 3way handshake with ack in wrong dir [**] [Classification: Generic Protocol Command Decode] [Priority: 3] {TCP} 192.168.2.20:80 -> 192.168.2.203:38914 **Πλήθος 45**

04/29/2017-22:48:14.043608 [**] [1:2014020:4] ET WEB_SERVER Wordpress Login Bruteforcing Detected [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 192.168.2.203:38930 -> 192.168.2.20:80 **Πλήθος 9**

Όπως φαίνεται από τον Πίνακα 6.13 κατά την εκτέλεση του συγκεκριμένου σεναρίου το Suricata ήταν σε θέση να εποπτεύσει και να αποκωδικοποιήσει το σύνολο των πακέτων - 6116 , οι συνεδρίες TCP έφθασαν τις 151 , ενώ ο δείκτης tcp.reassembly_gap είχε μόλις την τιμή 8.

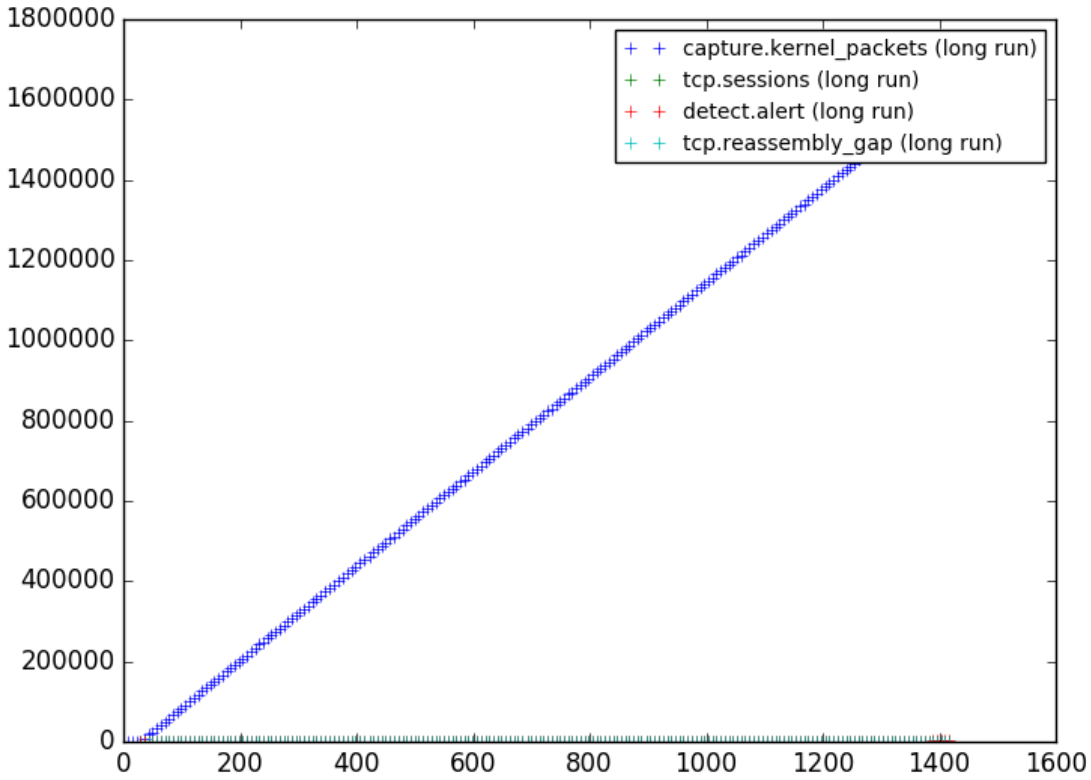
Η διαδικασία του Suricata κυμάνθηκε σε χαμηλά επίπεδα επεξεργαστή-μνήμης ενώ οι κίνδυνοι που καταγράφηκαν ανήκουν α) στην κατηγορία emerging-web_server "ET WEB_SERVER Wordpress Login Bruteforcing Detected" , "ET WEB_SERVER PHP Possible http Remote File Inclusion Attempt" που περιλαμβάνει υπογραφές για αδυναμίες και επιθέσεις Web Server β) στην κατηγορία emerging-policy "ET POLICY Cleartext WordPress Login" , " ET POLICY Http Client Body contains pwd= in cleartext" που περιλαμβάνει υπογραφές σχετικές με την standard πολιτική ασφαλείας και γ) stream-events "SURICATA STREAM 3way handshake with ack in wrong dir" που περιλαμβάνει υπογραφές για την ροή του πρωτοκόλλου TCP.

Μετρητής	Τιμή
Runtime	**23λ 37δ**
capture.kernel_packets	**1634179**
decoder.pkts	**1634179**
decoder.avg_pkt_size	**1022**
tcp.invalid_checksum	**7**
tcp.sessions	**607**
tcp.reassembly_gap	**147**
detect.alert	**1911**

Πίνακας 6.16 : Στατιστικά εκτέλεσης Suricata – Σενάριο Γ 150 επιτυχημένες αιτήσεις

Διαδικασία Suricata	
Μέγιστο % CPU	Μέγιστο % Memory
8.3	**17.9**

Πίνακας 6.17 : Μέγιστη χρήση μνήμης – επεξεργαστή από το Suricata Σενάριο Γ 150 επιτυχημένες αιτήσεις



Εικόνα 6.9 : Γραφική παράσταση στατιστικών – Σενάριο Γ 150 επιτυχημένες αιτήσεις

Fast.log (Πληθέστεροι εντοπισμένοι κίνδυνοι)

04/29/2017-17:41:15.516189 [**] [1:2210002:2] SURICATA STREAM 3way handshake right seq wrong ack evasion [**] [Classification: Generic Protocol Command Decode] [Priority: 3] {TCP} 192.168.2.203:52234 -> 192.168.2.20:21 **Πλήθος 56**

04/29/2017-17:41:15.516401 [**] [1:2210010:2] SURICATA STREAM 3way handshake wrong seq wrong ack [**] [Classification: Generic Protocol Command Decode] [Priority: 3] {TCP} 192.168.2.203:52234 -> 192.168.2.20:21 **Πλήθος 385**

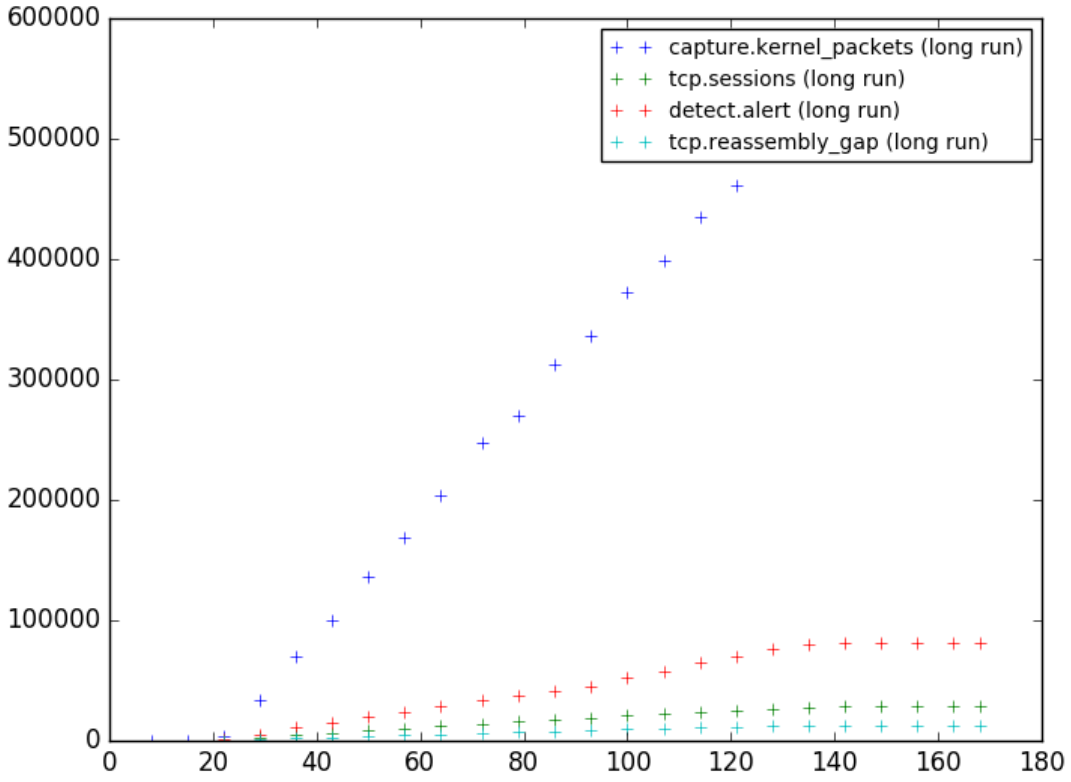
04/29/2017-17:41:15.517143 [**] [1:2010731:4] ET FTP FTP CWD command attempt without login [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 192.168.2.203:52234 -> 192.168.2.20:21 **Πλήθος 149**
04/29/2017-17:41:15.519997 [**] [1:2010737:2] ET FTP FTP NLST command attempt without login [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 192.168.2.203:52234 -> 192.168.2.20:21 **Πλήθος 150**
04/29/2017-17:41:15.524734 [**] [1:2010740:2] ET FTP FTP STOR command attempt without login [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 192.168.2.203:52234 -> 192.168.2.20:21 **Πλήθος 150**
04/29/2017-17:43:35.424989 [**] [1:2210016:2] SURICATA STREAM CLOSEWAIT FIN out of window [**] [Classification: Generic Protocol Command Decode] [Priority: 3] {TCP} 192.168.2.20:2575 -> 192.168.2.203:37074 **Πλήθος 66**
04/29/2017-17:41:15.516917 [**] [1:2210000:2] SURICATA STREAM 3way handshake with ack in wrong dir [**] [Classification: Generic Protocol Command Decode] [Priority: 3] {TCP} 192.168.2.20:21 -> 192.168.2.203:52234 **Πλήθος 461**

Όπως φαίνεται από τον Πίνακα 6.15 κατά την εκτέλεση του συγκεκριμένου σεναρίου το Suricata ήταν σε θέση να αποκωδικοποιήσει όλα τα πακέτα που επόπτευσε 1634179, οι συνεδρίες TCP έφθασαν τις 607 ενώ ο δείκτης tcp.reassembly_gap είχε την τιμή 147.

Η διαδικασία του Suricata κυμάνθηκε σε χαμηλά επίπεδα επεξεργαστή-μνήμης ενώ οι κίνδυνοι που καταγράφηκαν ανήκουν α) στην κατηγορία emerging-ftp “ET FTP FTP CWD command attempt without login” , “ET FTP FTP NLST command attempt without login”, “ET FTP FTP STOR command attempt without login” που περιλαμβάνει υπογραφές για αδυναμίες, επιθέσεις ή δραστηριότητα σχετική με FTP Server και β) στην κατηγορία stream-events “SURICATA STREAM 3way handshake with ack in wrong dir” , “SURICATA STREAM 3way handshake right seq wrong ack evasion” , “SURICATA STREAM 3way handshake wrong seq wrong ack” , “SURICATA STREAM CLOSEWAIT FIN out of window“ που περιλαμβάνει υπογραφές για την ροή του πρωτοκόλλου TCP.

Μετρητής	Τιμή
Runtime	**2λ 48δ**
capture.kernel_packets	**535032**
decoder.pkts	**535032**
decoder.avg_pkt_size	**113**
tcp.invalid_checksum	**9**
tcp.sessions	**28541**
tcp.reassembly_gap	**12974**
detect.alert	**81461**

Πίνακας 6.18 : Στατιστικά εκτέλεσης Suricata – Σενάριο Γ 30Κ αποτυχημένες αιτήσεις



Εικόνα 6.10 : Γραφική παράσταση στατιστικών – Σενάριο Γ 30Κ αποτυχημένες αιτήσεις

Διαδικασία Suricata	
Μέγιστο % CPU	Μέγιστο % Memory
33.2	**18.5**

Πίνακας 6.19 : Μέγιστη χρήση μνήμης – επεξεργαστή από το Suricata Σενάριο Γ 30K αποτυχημένες αιτήσεις

fast.log (Πληθέστεροι εντοπισμένοι κίνδυνοι)

04/29/2017-16:52:35.795937 [**] [1:2260002:1] SURICATA Applayer Detect protocol only one direction [**] [Classification: Generic Protocol Command Decode] [Priority: 3] {TCP} 192.168.2.20:21 -> 192.168.2.203:39490 **Πλήθος 17251**

04/29/2017-16:52:35.811191 [**] [1:2100491:10] GPL FTP FTP Bad login [**] [Classification: Potentially Bad Traffic] [Priority: 2] {TCP} 192.168.2.20:21 -> 192.168.2.203:39490 **Πλήθος 20786**

04/29/2017-16:52:35.839572 [**] [1:2210002:2] SURICATA STREAM 3way handshake right seq wrong ack evasion [**] [Classification: Generic Protocol Command Decode] [Priority: 3] {TCP} 192.168.2.203:39508 -> 192.168.2.20:21 **Πλήθος 5176**

04/29/2017-16:52:35.839864 [**] [1:2210010:2] SURICATA STREAM 3way handshake wrong seq wrong ack [**] [Classification: Generic Protocol Command Decode] [Priority: 3] {TCP} 192.168.2.203:39508 -> 192.168.2.20:21 **Πλήθος 3442**

04/29/2017-16:52:35.841459 [**] [1:2210000:2] SURICATA STREAM 3way handshake with ack in wrong dir [**] [Classification: Generic Protocol Command Decode] [Priority: 3] {TCP} 192.168.2.20:21 -> 192.168.2.203:39508 **Πλήθος 16391**

04/29/2017-16:54:31.732531 [**] [1:2210032:2] SURICATA STREAM FIN1 FIN with wrong seq [**] [Classification: Generic Protocol Command Decode] [Priority: 3] {TCP} 192.168.2.203:59404 -> 192.168.2.20:21 **Πλήθος 2211**

04/29/2017-16:54:31.733505 [**] [1:2210007:2] SURICATA STREAM 3way handshake SYNACK with wrong ack [**] [Classification: Generic Protocol Command Decode] [Priority: 3] {TCP} 192.168.2.20:21 -> 192.168.2.203:59408 **Πλήθος 776**

Όπως φαίνεται από τον Πίνακα 6.17 κατά την εκτέλεση του συγκεκριμένου σεναρίου το Suricata ήταν σε θέση να αποκωδικοποιήσει όλα τα πακέτα που επόπτευσε 535032, οι συνεδρίες TCP ήταν συνολικά 28541 ενώ ο δείκτης tcp.reassembly_gap είχε αυξημένη τιμή 12974.

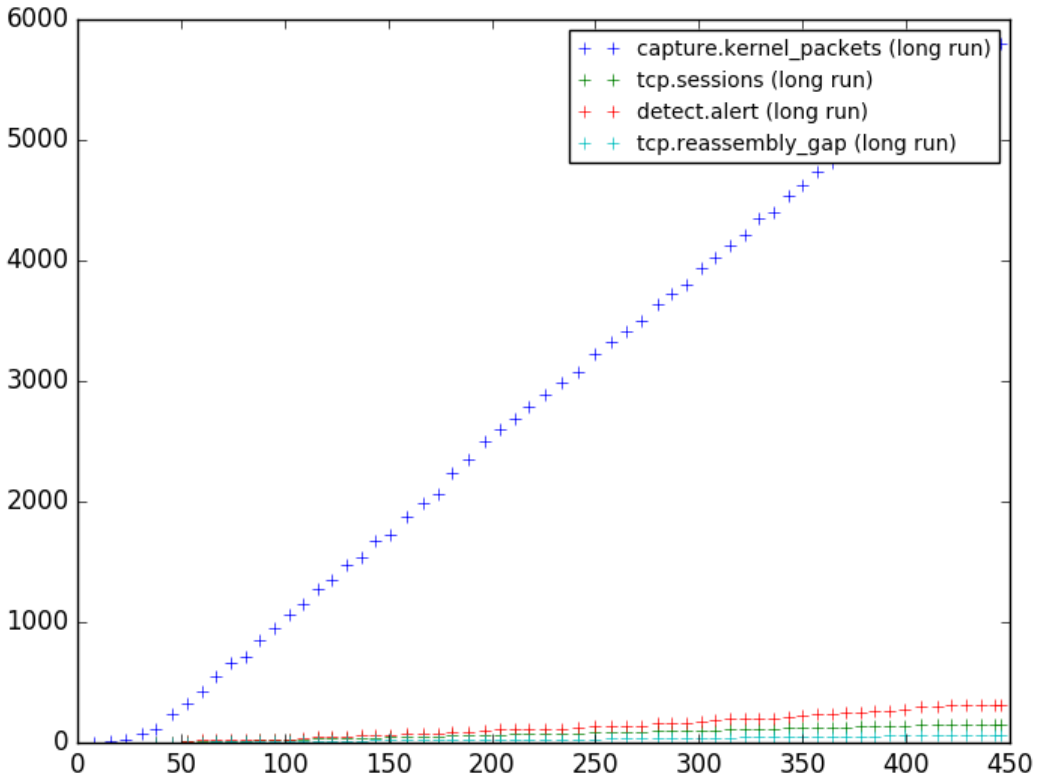
Η διαδικασία του Suricata κυμάνθηκε σε ανεβασμένα επίπεδα επεξεργαστή-μνήμης ενώ οι κίνδυνοι που καταγράφηκαν ανήκουν α) στην κατηγορία emerging-ftp " GPL FTP FTP Bad login" που περιλαμβάνει υπογραφές για αδυναμίες , επιθέσεις ή δραστηριότητα σχετική με FTP Server , β) στην κατηγορία app-layer-events "SURICATA Applayer Detect protocol only one direction" ενώ όλοι οι υπόλοιποι ανήκουν στην κατηγορία stream-events που περιλαμβάνει υπογραφές για την ροή του πρωτοκόλλου TCP.

Μετρητής	Τιμή
Runtime	**7λ 26δ**
capture.kernel_packets	**5795**
decoder.pkts	**5795**
decoder.avg_pkt_size	**167**
tcp.invalid_checksum	**0**
tcp.sessions	**151**
tcp.reassembly_gap	**66**
detect.alert	**317**

Πίνακας 6.20 : Στατιστικά εκτέλεσης Suricata – Σενάριο Δ 150 επιτυχημένες αιτήσεις

Διαδικασία Suricata	
Μέγιστο % CPU	Μέγιστο % Memory
1.0	**18.2**

Πίνακας 6.21 : Μέγιστη χρήση μνήμης – επεξεργαστή από το Suricata Σενάριο Δ 150 επιτυχημένες αιτήσεις



Εικόνα 6.11 : Γραφική παράσταση στατιστικών – Σενάριο Δ 150 επιτυχημένες αιτήσεις

fast.log (Πληθέστεροι εντοπισμένοι κίνδυνοι)

04/29/2017-23:32:22.095348 [**] [1:2010937:2] ET POLICY Suspicious inbound to mySQL port 3306 [**] [Classification: Potentially Bad Traffic] [Priority: 2] {TCP} 192.168.2.203:43228 -> 192.168.2.20:3306 **Πλήθος 35**

04/29/2017-23:32:45.537848 [**] [1:2210000:2] SURICATA STREAM 3way handshake with ack in wrong dir [**] [Classification: Generic Protocol Command Decode] [Priority: 3] {TCP} 192.168.2.20:3306 -> 192.168.2.203:43244 **Πλήθος 100**

2017-23:34:01.086627 [**] [1:2210002:2] SURICATA STREAM 3way handshake right seq wrong ack evasion [**] [Classification: Generic Protocol Command Decode] [Priority: 3] {TCP} 192.168.2.203:43302 -> 192.168.2.20:3306 **Πλήθος 40**

04/29/2017-23:34:01.086792 [**] [1:2210010:2] SURICATA STREAM 3way handshake wrong seq wrong ack [**] [Classification: Generic Protocol Command Decode] [Priority: 3] {TCP} 192.168.2.203:43302 -> 192.168.2.20:3306 **Πλήθος 99**

Όπως φαίνεται από τον Πίνακα 6.19 κατά την εκτέλεση του συγκεκριμένου σεναρίου το Suricata ήταν σε θέση να αποκωδικοποιήσει όλα τα πακέτα που

επόπτευσε - 5795, οι συνεδρίες TCP ήταν συνολικά 151 ενώ ο δείκτης tcp.reassembly_gap είχε τιμή 66.

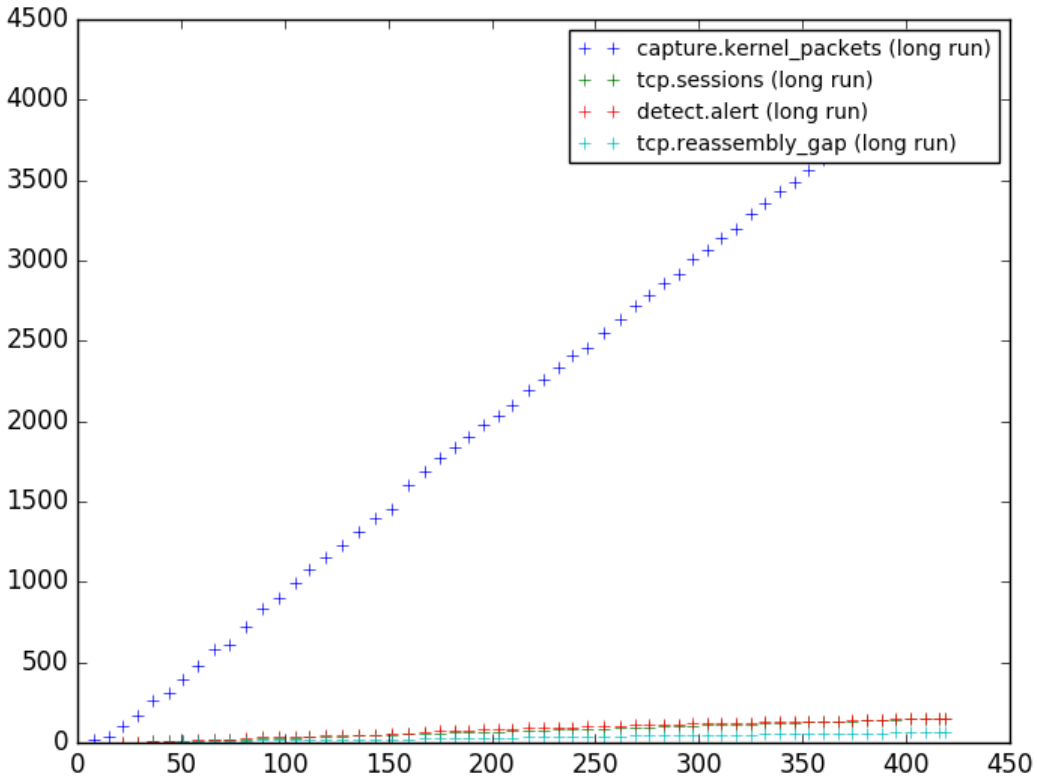
Η διαδικασία του Suricata κυμάνθηκε σε χαμηλά επίπεδα επεξεργαστή-μνήμης ενώ οι κίνδυνοι που καταγράφηκαν ανήκουν α) στην κατηγορία emerging-policy "ET POLICY Suspicious inbound to mySQL port 3306" που περιλαμβάνει που περιλαμβάνει υπογραφές σχετικές με την standard πολιτική ασφαλείας ενώ όλοι οι υπόλοιποι ανήκουν στην κατηγορία stream-events που περιλαμβάνει υπογραφές για την ροή του πρωτοκόλλου TCP.

Μετρητής	Τιμή
Runtime	**6λ 59δ**
capture.kernel_packets	**4220**
decoder.pkts	**4220**
decoder.avg_pkt_size	**174**
tcp.invalid_checksum	**0**
tcp.sessions	**150**
tcp.reassembly_gap	**65**
detect.alert	**154**

Πίνακας 6.22 : Στατιστικά εκτέλεσης Suricata – Σενάριο Δ 150 αποτυχημένες αιτήσεις

Διαδικασία Suricata	
Μέγιστο % CPU	Μέγιστο % Memory
1.0	**18.1**

Πίνακας 6.23 : Μέγιστη χρήση μνήμης – επεξεργαστή από το Suricata Σενάριο Δ αποτυχημένες αιτήσεις



Εικόνα 6.12 : Γραφική παράσταση στατιστικών – Σενάριο Δ 150 αποτυχημένες αιτήσεις

fast.log (Πληθέστεροι εντοπισμένοι κίνδυνοι)

04/29/2017-23:45:49.697447 [**] [1:2010937:2] ET POLICY Suspicious inbound to mySQL port 3306 [**] [Classification: Potentially Bad Traffic] [Priority: 2] {TCP} 192.168.2.203:43526 -> 192.168.2.20:3306 **Πλήθος 35**

04/29/2017-23:46:02.720005 [**] [1:2010494:3] ET SCAN Multiple MySQL Login Failures, Possible Brute Force Attempt [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 192.168.2.20:3306 -> 192.168.2.203:43534 **Πλήθος 29**

04/29/2017-23:46:18.352444 [**] [1:2210000:2] SURICATA STREAM 3way handshake with ack in wrong dir [**] [Classification: Generic Protocol Command Decode] [Priority: 3] {TCP} 192.168.2.20:3306 -> 192.168.2.203:43546 **Πλήθος 30**

04/29/2017-23:46:18.352666 [**] [1:2210002:2] SURICATA STREAM 3way handshake right seq wrong ack evasion [**] [Classification: Generic Protocol Command Decode] [Priority: 3] {TCP} 192.168.2.203:43546 -> 192.168.2.20:3306 **Πλήθος 48**

Όπως φαίνεται από τον Πίνακα 6.21 κατά την εκτέλεση του συγκεκριμένου σεναρίου το Suricata ήταν σε θέση να αποκωδικοποιήσει όλα τα πακέτα που

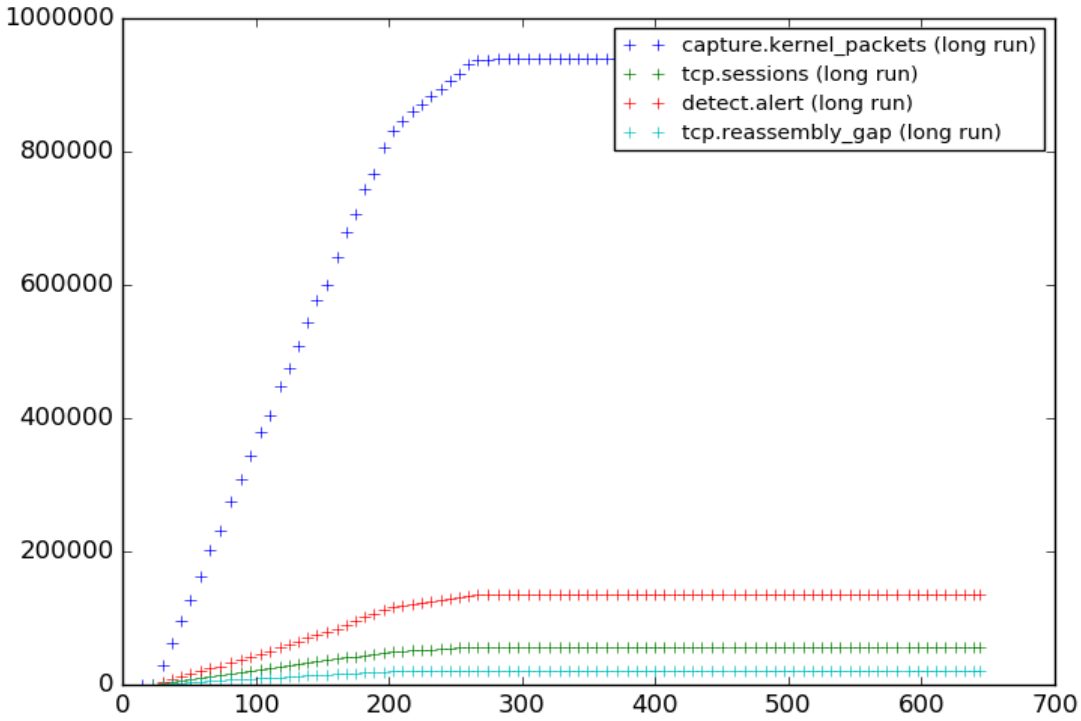
επόπτευσε 4220, οι συνεδρίες TCP ήταν συνολικά 150 ενώ ο δείκτης tcp.reassembly_gap είχε τιμή 65.

Η διαδικασία του Suricata κυμάνθηκε σε χαμηλά επίπεδα επεξεργαστή-μνήμης ενώ οι κίνδυνοι που καταγράφηκαν ανήκουν α) στην κατηγορία emerging-policy “ET POLICY Suspicious inbound to mySQL port 3306” που περιλαμβάνει που περιλαμβάνει υπογραφές σχετικές με την standard πολιτική ασφαλείας β) στην κατηγορία emerging-scan “ET SCAN Multiple MySQL Login Failures, Possible Brute Force Attempt” που περιλαμβάνει υπογραφές για τον εντοπισμό επαναλαμβανόμενων σαρώσεων πορτών/υπηρεσιών ενώ όλοι οι υπόλοιποι ανήκουν στην κατηγορία stream-events που περιλαμβάνει υπογραφές για την ροή του πρωτοκόλλου TCP.

β) Εκτέλεση των σεναρίων χωρισμένα σε τετράδες , επιτυχημένες / αποτυχημένες προσπάθειες

Μετρητής	Τιμή
Runtime	**10λ 44δ**
capture.kernel_packets	**943046**
decoder.pkts	**943046**
decoder.avg_pkt_size	**163**
tcp.invalid_checksum	**14**
tcp.sessions	**56664**
tcp.reassembly_gap	**21556**
detect.alert	**135931**

Πίνακας 6.24 : Στατιστικά εκτέλεσης Suricata – Σενάρια Α,Β,Γ,Δ 60.300 αποτυχημένες αιτήσεις



Εικόνα 6.13 : Γραφική παράσταση στατιστικών – Σενάρια Α,Β,Γ,Δ 60.300 αποτυχημένες αιτήσεις

Διαδικασία Suricata	
Μέγιστο % CPU	Μέγιστο % Memory
39.4	**21.7**

Πίνακας 6.25 : Μέγιστη χρήση μνήμης – επεξεργαστή από το Suricata Σενάρια Α,Β,Γ,Δ 60.300 αποτυχημένες αιτήσεις

fast.log (Πληθέστεροι εντοπισμένοι κίνδυνοι)

04/30/2017-12:08:01.977806 [**] [1:2210016:2] SURICATA STREAM CLOSEWAIT FIN out of window [**] [Classification: Generic Protocol Command Decode] [Priority: 3] {TCP} 178.255.83.2:80 -> 192.168.2.3:1183 Πλήθος 6488

04/30/2017-12:09:12.213269 [**] [1:2210045:2] SURICATA STREAM Packet with invalid ack [**] [Classification: Generic Protocol Command Decode] [Priority: 3] {TCP} 192.168.2.203:60230 -> 192.168.2.20:80 **Πλήθος 15368**

04/30/2017-12:08:06.437729 [**] [1:2010937:2] ET POLICY Suspicious inbound to mySQL port 3306 [**] [Classification: Potentially Bad Traffic] [Priority: 2] {TCP} 192.168.2.203:36174 -> 192.168.2.20:3306 **Πλήθος 35**

04/30/2017-12:08:08.881736 [**] [1:2100491:10] GPL FTP FTP Bad login [**] [Classification: Potentially Bad Traffic] [Priority: 2] {TCP} 192.168.2.20:21 -> 192.168.2.203:53014 **Πλήθος 20759**

04/30/2017-12:08:15.160845 [**] [1:2101201:10] GPL WEB_SERVER 403 Forbidden [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 192.168.2.20:80 -> 192.168.2.203:47152 **Πλήθος 18951**

04/30/2017-12:08:08.880092 [**] [1:2260002:1] SURICATA Applayer Detect protocol only one direction [**] [Classification: Generic Protocol Command Decode] [Priority: 3] {TCP} 192.168.2.20:21 -> 192.168.2.203:53014 **Πλήθος 16348**

04/30/2017-12:08:08.891408 [**] [1:2210000:2] SURICATA STREAM 3way handshake with ack in wrong dir [**] [Classification: Generic Protocol Command Decode] [Priority: 3] {TCP} 192.168.2.20:21 -> 192.168.2.203:53020 **Πλήθος 20088**

04/30/2017-12:08:08.953134 [**] [1:2210010:2] SURICATA STREAM 3way handshake wrong seq wrong ack [**] [Classification: Generic Protocol Command Decode] [Priority: 3] {TCP} 192.168.2.203:53052 -> 192.168.2.20:21 **Πλήθος 6429**

04/30/2017-12:08:19.473246 [**] [1:2010494:3] ET SCAN Multiple MySQL Login Failures, Possible Brute Force Attempt [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 192.168.2.20:3306 -> 192.168.2.203:40510 **Πλήθος 29**

04/30/2017-12:08:35.627258 [**] [1:2014020:4] ET WEB_SERVER Wordpress Login Bruteforcing Detected [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 192.168.2.203:58614 -> 192.168.2.20:80 **Πλήθος 9**

04/30/2017-12:12:35.631809 [**] [1:2012843:3] ET POLICY Cleartext WordPress Login [**] [Classification: Potential Corporate Privacy Violation] [Priority: 1] {TCP} 192.168.2.203:40642 -> 192.168.2.20:80 **Πλήθος 111**

04/30/2017-12:12:35.631809 [**] [1:2012888:3] ET POLICY Http Client Body contains pwd= in cleartext [**] [Classification: Potential Corporate Privacy Violation] [Priority: 1] {TCP} 192.168.2.203:40642 -> 192.168.2.20:80 **Πλήθος 111**

04/30/2017-12:12:38.017714 [**] [1:2012997:4] ET WEB_SERVER PHP Possible http Remote File Inclusion Attempt [**] [Classification: Web Application Attack] [Priority: 1] {TCP} 192.168.2.203:40642 -> 192.168.2.20:80 **Πλήθος 111**

Όπως φαίνεται από τον Πίνακα 6.23 κατά την εκτέλεση του συγκεκριμένου σεναρίου το Suricata ήταν σε θέση να αποκωδικοποιήσει όλα τα πακέτα που επόπτευσε 943046, οι συνεδρίες TCP ήταν συνολικά 56664 ενώ ο δείκτης tcp.reassembly_gap είχε υψηλή τιμή 21556.

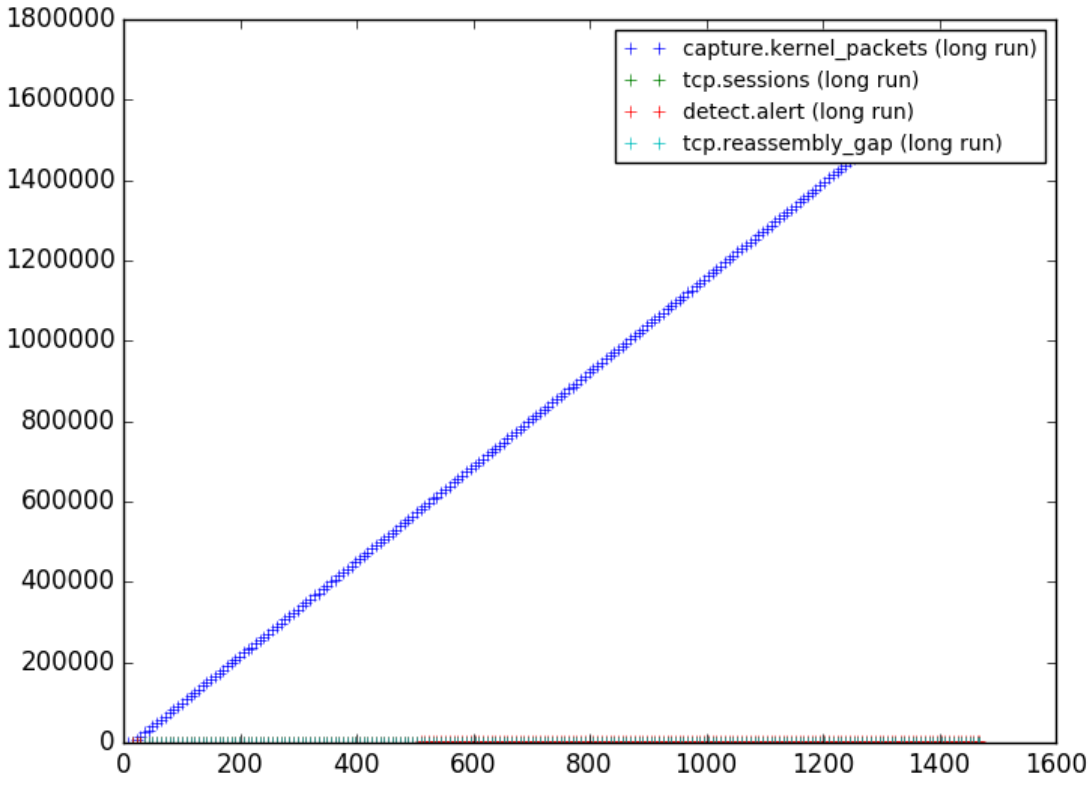
Η διαδικασία του Suricata κυμάνθηκε σε υψηλά επίπεδα επεξεργαστή-μνήμης ενώ οι κίνδυνοι που καταγράφηκαν ανήκουν α) στην κατηγορία emerging-policy που περιλαμβάνει υπογραφές σχετικές με την standard πολιτική ασφαλείας β) στην κατηγορία emerging-scan που περιλαμβάνει υπογραφές για τον εντοπισμό επαναλαμβανόμενων σαρώσεων πορτών/υπηρεσιών γ) στην κατηγορία emerging-ftp που περιλαμβάνει υπογραφές για αδυναμίες , επιθέσεις ή δραστηριότητα σχετική με FTP Server δ)στην κατηγορία emerging-web_server που περιλαμβάνει υπογραφές για αδυναμίες και επιθέσεις Web Server, ε) στην κατηγορία app_layer_events ενώ όλοι οι υπόλοιποι ανήκουν στην κατηγορία stream-events που περιλαμβάνει υπογραφές για την ροή του πρωτοκόλλου TCP.

Μετρητής	Τιμή
Runtime	**24λ 29δ**
capture.kernel_packets	**1698756**
decoder.pkts	**1698756**
decoder.avg_pkt_size	**1014**
tcp.invalid_checksum	**7**
tcp.sessions	**1403**
tcp.reassembly_gap	**179**
detect.alert	**3902**

Πίνακας 6.26 : Στατιστικά εκτέλεσης Suricata – Σενάρια Α,Β,Γ,Δ 950 επιτυχημένες αιτήσεις

Διαδικασία Suricata	
Μέγιστο % CPU	Μέγιστο % Memory
12.3	**18.2**

Πίνακας 6.27 : Μέγιστη χρήση μνήμης – επεξεργαστή από το Suricata Σενάρια Α,Β,Γ,Δ 950 επιτυχημένες αιτήσεις



Εικόνα 6.14 : Γραφική παράσταση στατιστικών – Σενάρια Α,Β,Γ,Δ 950 επιτυχημένες αιτήσεις

fast.log (Πληθέστεροι εντοπισμένοι κίνδυνοι)

04/30/2017-11:34:07.682986 [**] [1:2010937:2] ET POLICY Suspicious inbound to mySQL port 3306 [**] [Classification: Potentially Bad Traffic] [Priority: 2] {TCP} 192.168.2.203:33374 -> 192.168.2.20:3306 **Πλήθος 35**

04/30/2017-11:34:10.502033 [**] [1:2010731:4] ET FTP FTP CWD command attempt without login [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 192.168.2.203:50216 -> 192.168.2.20:21 **Πλήθος 148**

04/30/2017-11:34:10.505054 [**] [1:2010737:2] ET FTP FTP NLST command attempt without login [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 192.168.2.203:50216 -> 192.168.2.20:21 **Πλήθος 150**
04/30/2017-11:34:10.508797 [**] [1:2010740:2] ET FTP FTP STOR command attempt without login [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 192.168.2.203:50216 -> 192.168.2.20:21 **Πλήθος 150**
04/30/2017-11:34:16.745289 [**] [1:2012843:3] ET POLICY Cleartext WordPress Login [**] [Classification: Potential Corporate Privacy Violation] [Priority: 1] {TCP} 192.168.2.203:41030 -> 192.168.2.20:80 **Πλήθος 148**
04/30/2017-11:34:23.354234 [**] [1:2012888:3] ET POLICY Http Client Body contains pwd= in cleartext [**] [Classification: Potential Corporate Privacy Violation] [Priority: 1] {TCP} 192.168.2.203:41054 -> 192.168.2.20:80 **Πλήθος 148**
04/30/2017-11:34:26.090002 [**] [1:2014020:4] ET WEB_SERVER Wordpress Login Bruteforcing Detected [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 192.168.2.203:41062 -> 192.168.2.20:80 **Πλήθος 7**
04/30/2017-11:34:20.130604 [**] [1:2210000:2] SURICATA STREAM 3way handshake with ack in wrong dir [**] [Classification: Generic Protocol Command Decode] [Priority: 3] {TCP} 192.168.2.20:21 -> 192.168.2.203:50244 **Πλήθος 886**
04/30/2017-11:34:20.130897 [**] [1:2210002:2] SURICATA STREAM 3way handshake right seq wrong ack evasion [**] [Classification: Generic Protocol Command Decode] [Priority: 3] {TCP} 192.168.2.203:50244 -> 192.168.2.20:21 **Πλήθος 82**
04/30/2017-11:34:20.131356 [**] [1:2210010:2] SURICATA STREAM 3way handshake wrong seq wrong ack [**] [Classification: Generic Protocol Command Decode] [Priority: 3] {TCP} 192.168.2.203:50244 -> 192.168.2.20:21 **Πλήθος 704**
04/30/2017-11:34:21.996308 [**] [1:2210020:2] SURICATA STREAM ESTABLISHED packet out of window [**] [Classification: Generic Protocol Command Decode] [Priority: 3] {TCP} 192.168.2.203:58584 -> 192.168.2.20:2868 **Πλήθος 1082**

Όπως φαίνεται από τον Πίνακα 6.25 κατά την εκτέλεση του συγκεκριμένου σεναρίου το Suricata ήταν σε θέση να αποκωδικοποιήσει όλα τα πακέτα που επόπτευσε - 1698756, οι συνεδρίες TCP ήταν συνολικά 1403 ενώ ο δείκτης tcp.reassembly_gap είχε τιμή 179.

Η διαδικασία του Suricata κυμάνθηκε σε χαμηλά επίπεδα επεξεργαστή-μνήμης ενώ οι κίνδυνοι που καταγράφηκαν ανήκουν α) στην κατηγορία emerging-policy που

περιλαμβάνει υπογραφές σχετικές με την standard πολιτική ασφαλείας β) στην κατηγορία emerging-ftp που περιλαμβάνει υπογραφές για αδυναμίες , επιθέσεις ή δραστηριότητα σχετική με FTP Server γ) στην κατηγορία emerging-web_server, που περιλαμβάνει υπογραφές για αδυναμίες και επιθέσεις Web Server, ενώ όλοι οι υπόλοιποι ανήκουν στην κατηγορία stream-events που περιλαμβάνει υπογραφές για την ροή του πρωτοκόλλου TCP.

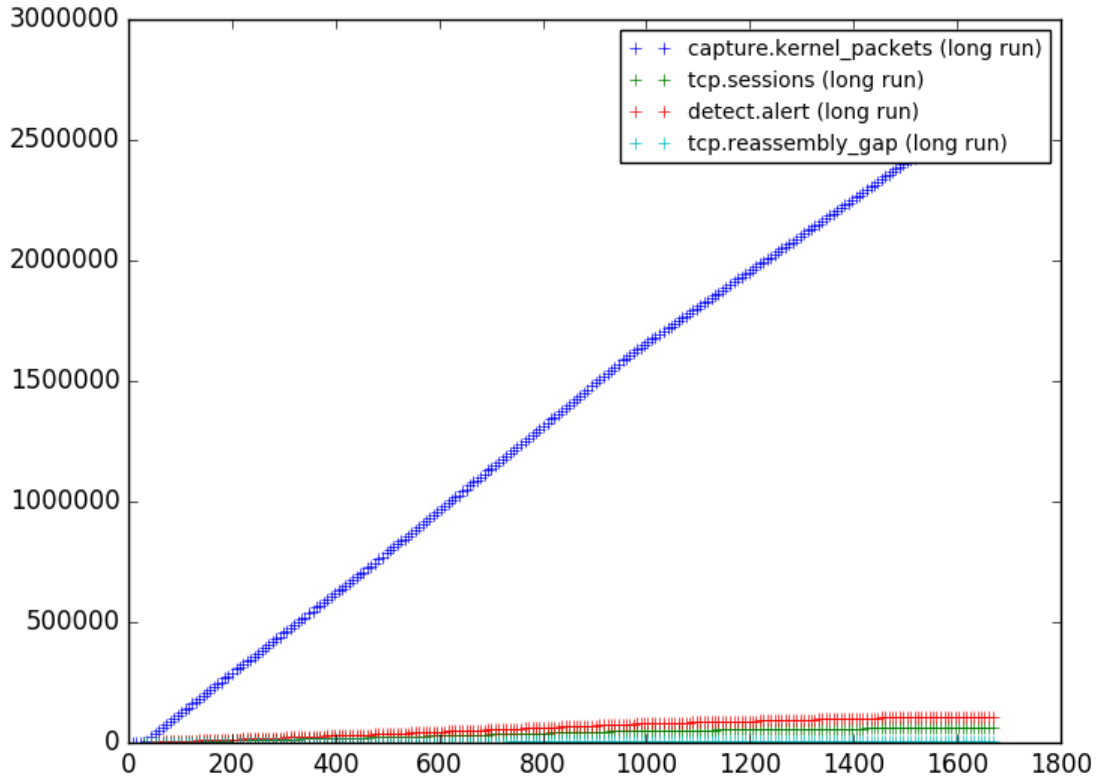
γ)Εκτέλεση των σεναρίων παράλληλα

Μετρητής	Τιμή
Runtime	**27λ 48δ**
capture.kernel_packets	**2582437**
decoder.pkts	**2582437**
decoder.avg_pkt_size	**723**
tcp.invalid_checksum	**19**
tcp.sessions	**61388**
tcp.reassembly_gap	**5302**
detect.alert	**106604**

Πίνακας 6.28 : Στατιστικά εκτέλεσης Suricata – Σενάρια Α,Β,Γ,Δ 61250 επιτυχημένες /αποτυχημένες αιτήσεις

Διαδικασία Suricata	
Μέγιστο % CPU	Μέγιστο % Memory
18.3	**19.2**

Πίνακας 6.29 : Μέγιστη χρήση μνήμης – επεξεργαστή από το Suricata - Σενάρια Α,Β,Γ,Δ 61250 επιτυχημένες /αποτυχημένες αιτήσεις



Εικόνα 6.15 : Γραφική παράσταση στατιστικών – Σενάρια Α,Β,Γ,Δ 61250 επιτυχημένες /αποτυχημένες αιτήσεις

fast.log (Πληθέστεροι εντοπισμένοι κίνδυνοι)

04/30/2017-01:06:54.282232 [**] [1:2012843:3] ET POLICY Cleartext WordPress Login [**] [Classification: Potential Corporate Privacy Violation] [Priority: 1] {TCP} 192.168.2.203:50254 -> 192.168.2.20:80 **Πλήθος 298**

04/30/2017-01:06:54.282232 [**] [1:2012888:3] ET POLICY Http Client Body contains pwd= in cleartext [**] [Classification: Potential Corporate Privacy Violation] [Priority: 1] {TCP} 192.168.2.203:50254 -> 192.168.2.20:80 **Πλήθος 298**

04/30/2017-01:06:55.022922 [**] [1:2010937:2] ET POLICY Suspicious inbound to mySQL port 3306 [**] [Classification: Potentially Bad Traffic] [Priority: 2] {TCP} 192.168.2.203:54302 -> 192.168.2.20:3306 **Πλήθος 67**

04/30/2017-01:06:57.570967 [**] [1:2210000:2] SURICATA STREAM 3way handshake with ack in wrong dir [**] [Classification: Generic Protocol Command Decode] [Priority: 3] {TCP} 192.168.2.20:80 -> 192.168.2.203:50260 **Πλήθος 5002**

04/30/2017-01:06:57.571347 [**] [1:2210010:2] SURICATA STREAM 3way handshake wrong seq wrong ack [**] [Classification: Generic Protocol Command Decode] [Priority: 3] {TCP} 192.168.2.203:50260 -> 192.168.2.20:80 **Πλήθος 2920**

04/30/2017-01:06:57.579423 [**] [1:2210016:2] SURICATA STREAM CLOSEWAIT FIN out of window [**] [Classification: Generic Protocol Command Decode] [Priority: 3] {TCP} 192.168.2.20:80 -> 192.168.2.203:50260 **Πλήθος 2083**

04/30/2017-01:06:57.584057 [**] [1:2101201:10] GPL WEB_SERVER 403 Forbidden [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 192.168.2.20:80 -> 192.168.2.203:50262 **Πλήθος 28129**

04/30/2017-01:06:57.809297 [**] [1:2009749:4] ET SCAN Unusually Fast 403 Error Messages, Possible Web Application Scan [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 192.168.2.20:80 -> 192.168.2.203:50312 **Πλήθος 1623**

04/30/2017-01:07:00.609040 [**] [1:2100491:10] GPL FTP FTP Bad login [**] [Classification: Potentially Bad Traffic] [Priority: 2] {TCP} 192.168.2.20:21 -> 192.168.2.203:38592 **Πλήθος 28414**

04/30/2017-01:07:00.711496 [**] [1:2210002:2] SURICATA STREAM 3way handshake right seq wrong ack evasion [**] [Classification: Generic Protocol Command Decode] [Priority: 3] {TCP} 192.168.2.203:38676 -> 192.168.2.20:21 **Πλήθος 2254**

04/30/2017-01:07:05.071664 [**] [1:2014020:4] ET WEB_SERVER Wordpress Login Bruteforcing Detected [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 192.168.2.203:52844 -> 192.168.2.20:80 **Πλήθος 10**

04/30/2017-01:07:08.123697 [**] [1:2010494:3] ET SCAN Multiple MySQL Login Failures, Possible Brute Force Attempt [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 192.168.2.20:3306 -> 192.168.2.203:57080 **Πλήθος 27**

04/30/2017-01:07:03.087946 [**] [1:2010731:4] ET FTP FTP CWD command attempt without login [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 192.168.2.203:40308 -> 192.168.2.20:21 **Πλήθος 150**

04/30/2017-01:07:03.093557 [**] [1:2010737:2] ET FTP FTP NLST command attempt without login [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 192.168.2.203:40308 -> 192.168.2.20:21 **Πλήθος 150**

04/30/2017-01:07:03.103168 [**] [1:2010740:2] ET FTP FTP STOR command attempt without login [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 192.168.2.203:40308 -> 192.168.2.20:21 **Πλήθος 150**

04/30/2017-01:07:25.103424 [**] [1:2210020:2] SURICATA STREAM ESTABLISHED packet out of window [**] [Classification: Generic Protocol Command Decode] [Priority: 3] {TCP} 192.168.2.203:59802 -> 192.168.2.20:9599 **Πλήθος 1704**

Όπως φαίνεται από τον Πίνακα 6.27 κατά την εκτέλεση του συγκεκριμένου σεναρίου το Suricata ήταν σε θέση να αποκωδικοποιήσει όλα τα πακέτα που επόπτευσε 2582437, οι συνεδρίες TCP ήταν συνολικά 61388 ενώ ο δείκτης tcp.reassembly_gap είχε τιμή 5302.

Η διαδικασία του Suricata κυμάνθηκε σε μέτρια επίπεδα επεξεργαστή-μνήμης ενώ οι κίνδυνοι που καταγράφηκαν ανήκουν α) στην κατηγορία emerging-policy που περιλαμβάνει υπογραφές σχετικές με την standard πολιτική ασφαλείας β) στην κατηγορία emerging-ftp, που περιλαμβάνει υπογραφές για αδυναμίες , επιθέσεις ή δραστηριότητα σχετική με FTP Server γ) στην κατηγορία emerging-web_server, που περιλαμβάνει υπογραφές για αδυναμίες και επιθέσεις Web Server, δ) στην κατηγορία emerging-scan που περιλαμβάνει υπογραφές για τον εντοπισμό επαναλαμβανόμενων σαρώσεων πορτών/υπηρεσιών ενώ όλοι οι υπόλοιποι ανήκουν στην κατηγορία stream-events που περιλαμβάνει υπογραφές για την ροή του πρωτοκόλλου TCP.

6.4 Συγκεντρωτικοί πίνακες

Εκτέλεση σεναρίων μεμονωμένα							
Σενάριο – επιτυχημένες/αποτυχημένες αιτήσεις	Memory %	CPU %	Runtime sec	Captured Kernel Packets	TCP Sessions	Detected Alerts	Tcp_reassembly_gap counter
Α- 500 επιτυχημένες αιτήσεις	18.6	1.7	694	40939	502	3847	43
Α- 30Κ αποτυχημένες αιτήσεις	23.1	18.6	291	387416	27996	50861	4877
Β- 150 επιτυχημένες αιτήσεις	18.2	1.3	429	12937	157	821	4
Β- 150 αποτυχημένες αιτήσεις	17.8	1.0	990	6116	151	470	8
Δ- 150 επιτυχημένες αιτήσεις	18.2	1	446	5795	151	317	66
Δ- 150 αποτυχημένες αιτήσεις	18.1	1	419	4220	150	154	65
Γ- 150 επιτυχημένες αιτήσεις	17.9	8.3	1417	1634179	607	1911	147
Γ- 30Κ αποτυχημένες αιτήσεις	18.5	33.2	168	535032	28541	81461	12974
Σύνολα			4854	2626634	58255	139842	18184

Πίνακας 6.30 : Στατιστικά Μνήμης/Επεξεργαστή/Suricata – Εκτέλεση σεναρίων μεμονωμένα

Από τον παραπάνω πίνακα διακρίνουμε ότι οι μέγιστες τιμές στο ποσοστό της CPU που καταναλώνεται από το Suricata, εμφανίζονται κατά την εκτέλεση των σεναρίων Α (30Κ αποτυχημένες αιτήσεις) και Γ (30Κ αποτυχημένες αιτήσεις) με τιμές 18.6% και 33.2% αντίστοιχα. Τα δύο προαναφερθέντα σενάρια παρουσιάζουν επίσης τον μεγαλύτερο αριθμό ανιχνευμένων κινδύνων (50861 κίνδυνοι για το σενάριο Α και 81461 για το σενάριο Γ) ενώ γι αυτά τα σενάρια ο μετρητής tcp_reassembly_gap παρουσιάζεται ιδιαίτερα αυξημένος , με τιμές 4877 για το σενάριο Α και 12974 για το σενάριο Γ.

Εκτέλεση των σεναρίων χωρισμένα σε τετράδες , επιτυχημένες / αποτυχημένες προσπάθειες							
Σενάριο – επιτυχημένες/αποτυχημένες αιτήσεις	Memory	CPU	Runtime sec	Captured Kernel Packets	TCP Sessions	Detected Alerts	Tcp_reassembly_gap counter
Α- 30Κ αποτυχημένες αιτήσεις	21.7	39.4	644	943046	56664	135931	21556
Δ- 150 αποτυχημένες αιτήσεις							
Γ- 30Κ αποτυχημένες αιτήσεις							
Β- 150 αποτυχημένες αιτήσεις							
Δ- 150 επιτυχημένες αιτήσεις	18.2	12.3	1469	1698756	1403	3902	179
Α- 500 επιτυχημένες αιτήσεις							
Β- 150 επιτυχημένες αιτήσεις							
Γ- 150 επιτυχημένες αιτήσεις							
Σύνολα			2113	2641802	58067	139833	21735

Πίνακας 6.31 : Στατιστικά Μνήμης/Επεξεργαστή/Suricata – Εκτέλεση σεναρίων χωρισμένα σε τετράδες επιτυχημένες / αποτυχημένες αιτήσεις

Από τον παραπάνω πίνακα είναι εμφανές ότι οι μέγιστες τιμές στο ποσοστό της CPU που καταναλώνεται από το Suricata, εμφανίζονται κατά την εκτέλεση των σεναρίων με αποτυχημένες αιτήσεις , με τιμή 39.4%. Τα σενάρια με αποτυχημένες αιτήσεις παρουσιάζουν επίσης τον μεγαλύτερο αριθμό ανιχνευμένων κινδύνων - 135931 ενώ για τα ίδια σενάρια η τιμή του μετρητή tcp_reassembly_gap είναι της τάξεως του 21556.

Εκτέλεση των σεναρίων παράλληλα							
Σενάριο – επιτυχημένες/αποτυχημένες αιτήσεις	Memory	CPU	Runtime sec	Captured Kernel Packets	TCP Sessions	Detected Alerts	Tcp_reassembly_gap counter
Α- 30Κ αποτυχημένες αιτήσεις							
Δ- 150 αποτυχημένες αιτήσεις							
Γ- 30Κ αποτυχημένες αιτήσεις							
Β- 150 αποτυχημένες αιτήσεις							
Δ- 150 επιτυχημένες αιτήσεις							
Α- 500 επιτυχημένες αιτήσεις							
Β- 150 επιτυχημένες αιτήσεις							
Γ- 150 επιτυχημένες αιτήσεις							
Σύνολα	19.2	18.3	1668	2582437	61388	106604	5302

Πίνακας 6.32 : Στατιστικά Μνήμης/Επεξεργαστή/Suricata – Εκτέλεση σεναρίων παράλληλα

Ο παραπάνω πίνακας αποτυπώνει ότι κατά την εκτέλεση και των 8 σεναρίων παράλληλα το ποσοστό της μνήμης και της CPU που καταναλώνεται από το Suricata δεν ξεπερνάει το 19.2 και 18.3% αντίστοιχα.

Εκτέλεση σεναρίων	Runtime sec	Captured Kernel Packets	TCP Sessions	Detected Alerts	Tcp_reassembly_gap counter
Μεμονωμένα	4854	2626634	58255	139842	18184
Χωρισμένα σε τετράδες	2113	2641802	58067	139833	21735
Παράλληλα	1668	2582437	61388	106604	5302

Πίνακας 6.33 : Στατιστικά Εκτέλεσης Σεναρίων Συγκριτικά

Από τον παραπάνω πίνακα διακρίνεται ότι ενώ ο αριθμός των πακέτων που εποπτεύθηκαν από το Suricata αλλά και ο αριθμός των συνεδρίων TCP είναι παραπλήσιοι για τις τρεις περιπτώσεις εκτέλεσης , τόσο ο αριθμός των ανιχνευμένων κινδύνων όσο και η τιμή του μετρητή tcp_reassembly_gap είναι αρκετά μικρότερα κατά την εκτέλεση των σεναρίων παράλληλα.

Εκτελώντας τα σενάρια παράλληλα , είναι πιο ασφαλές να υπολογίσουμε την απόδοση του Suricata στην μονάδα του χρόνου. Σύμφωνα με την τιμή του δείκτη captured kernel packets και του runtime , υπολογίζουμε ότι το Suricata ήταν σε θέση να εποπτεύσει $\sim$ 1548 πακέτα το δευτερόλεπτο για ένα σύνολο 18837 υπογραφών που φορτώθηκαν κατά την έναρξή του.

6.5 Περιορισμοί

Με σκοπό την μέτρηση της απόδοσης του εργαλείου Suricata, όσον αφορά κίνηση που προέρχεται στο εσωτερικό δίκτυο, χρησιμοποιήθηκε παλαιάς τεχνολογίας συσκευή hub ταχύτητας (10Mbit/s) , η οποία επιτρέπει την μετάδοση της κίνησης σε όλες τις διαθέσιμες πόρτες.

Χρησιμοποιήθηκαν 3 εικονικές μηχανές (Windows 2008 R2 , Ubuntu Server 16.10 , Ubuntu Desktop 16.04) σε πλατφόρμα VMware Workstation Pro 12. Παράλληλα, στις εικονικές μηχανές φορτώθηκαν οι οδηγοί και τα εργαλεία VMware tools της συγκεκριμένης πλατφόρμας ενώ παράλληλα εγκαταστάθηκαν όλες οι προτεινόμενες αναβαθμίσεις των λειτουργικών συστημάτων. Οι εικονικές μηχανές διάθεταν ανεξάρτητη κάρτα δικτύου, η ταχύτητα της οποίας ρυθμίστηκε στα 10Mbit/s για λόγους συμβατότητας με την συσκευή hub ενώ απενεργοποιήθηκε το IPv6 σε όλες τις κάρτες δικτύου.

Λόγω του ότι η αναπαραγωγή δικτυακής κίνησης υψηλού όγκου, δεν είναι εφικτή σε ένα μικρό τοπικό δίκτυο, προτιμήθηκε να πραγματοποιηθούν μετρήσεις κάτω από συγκεκριμένες συνθήκες - σενάρια, έτσι ώστε τα αποτελέσματα να είναι όσο τον δυνατόν πιο ακριβή. Συγκεκριμένα, αρχικά αναπτύχθηκαν 4 διαφορετικά σενάρια (scripts) που παρουσιάζονται στο Παράρτημα Α σε γλώσσα python (έκδοση 2.7.12) , τα οποία αναπαράγουν TCP κίνηση και εκτελούν επιτυχημένα αιτήματα HTTP, FTP, MySQL και στην συνέχεια με την κατάλληλη τροποποίηση τους αναπτύχθηκαν 4 ακόμα σενάρια που εκτελούν αποτυχημένα αιτήματα για τα ίδια πρωτόκολλα.

Στο Suricata φορτώθηκε η συλλογή υπογραφών της εταιρείας Emerging Threats η οποία είναι διαθέσιμη χωρίς συνδρομή ενώ το Suricata εκτελέστηκε σε λειτουργία AF_PACKET, ο τύπος του cluster για την συγκεκριμένη λειτουργία ρυθμίστηκε σε cluster_flow ενώ το Offloading στην κάρτα δικτύου που εγκαταστάθηκε το Suricata απενεργοποιήθηκε. Ο οδηγός της κάρτας δικτύου στην ίδια μηχανή δεν υποστήριζε την ρύθμιση των ουρών RSS.

6.6 Συστάσεις Μελλοντικής Έρευνας

Προκειμένου να ελέγξουμε την απόδοση του εργαλείου Suricata στο εσωτερικό δίκτυο αναπαράγαμε TCP κίνηση HTTP,FTP & MySQL εκτελώντας python scripts από μία πηγή προς ένα προορισμό. Μελλοντικές έρευνες θα μπορούσαν να αναπαράγουν μεγαλύτερο όγκο κίνησης, η οποία θα περιλαμβάνει περισσότερα πρωτόκολλα όπως και UDP πακέτα, από περισσότερες πηγές προς περισσότερες του 1 προορισμούς, προκειμένου να ελεγχτεί η λειτουργία και η απόδοση ενός IDS συστήματος.

Επίσης, για τους σκοπούς της παρούσας μεταπτυχιακής διατριβής το Suricata εγκαταστάθηκε σε εικονική μηχανή με λειτουργικό σύστημα Ubuntu Server και πυρήνα Linux 4.8. Θα ήταν άκρως ενδιαφέρον, μελλοντικοί ερευνητές να συγκρίνουν την απόδοση και λειτουργία ενός IDS συστήματος τύπου Suricata τόσο για διαφορετικά λειτουργικά συστήματα όσο και για διαφορετικούς πυρήνες, εάν πρόκειται για συστήματα που βασίζονται σε λειτουργικό τύπου Linux. Παράλληλα , θα μπορούσαν να συγκριθούν παρόμοια συστήματα IDS που έχουν εγκατασταθεί σε εικονικές και φυσικές μηχανές, αφού είναι γνωστό ότι η τεχνολογία της εικονικοποίησης (virtualization) έχει αντίκτυπο στην απόδοση του δικτύου.

Ο υπερβολικός αριθμός εντοπισμένων κινδύνων που αφορούν την ροή δεδομένων και καταγράφηκαν από το Suricata χρήζει περισσότερης έρευνας , αφού όπως αποδείχτηκε οι κίνδυνοι αυτοί ήταν αλληλένδετοι με την τεχνολογία του Receive Side scaling. Όλες οι σύγχρονες κάρτες και οι αντίστοιχοι οδηγοί τους έχουν ενεργοποιημένη την λειτουργία του RSS και το γεγονός αυτό πρέπει να ληφθεί υπόψη και να παραμετροποιηθεί κατάλληλα σε συστήματα IDS. Επίσης , το Suricata εκτελέστηκε σε λειτουργία AF_PACKET, ενώ ο τύπος του cluster για την συγκεκριμένη λειτουργία ρυθμίστηκε σε cluster_flow. Το Suricata υποστηρίζει ακόμα 4 διαφορετικούς τύπους , cluster_round_robin,cluster_cpu,cluster_qm και cluster_random στην λειτουργία AF_PACKET που χρήζουν μελλοντικής μελέτης αναφορικά με την απόδοσή τους.

Τέλος, για τους σκοπούς της παρούσας μεταπτυχιακής διατριβής ενεργοποιήθηκε ένας μεγάλος αριθμός υπογραφών στο Suricata. Μελλοντικές έρευνες θα μπορούσαν να εστιάσουν στο είδος των υπογραφών που επιφέρουν μεγαλύτερο κόστος σε συστήματα IDS αναφορικά με την υπολογιστική ισχύ (επεξεργαστής – μνήμη).

Κεφάλαιο 7

Αξιολόγηση Συστημάτων Ανίχνευσης Εισβολών (IDS)

7.1 Σχετικές Εργασίες

Η αξιολόγηση Συστημάτων Ανίχνευσης Εισβολών (IDS) έχει αποτελέσει αντικείμενο μελέτης σειράς μελετών που εξήγαν ορισμένα σημαντικά συμπεράσματα. Στη συγκεκριμένη ενότητα, θα καταβληθεί προσπάθεια να απεικονιστούν τα στοιχεία ορισμένων εξ αυτών ώστε να διαμορφωθεί μια σαφής εικόνα αναφορικά με το ρόλο των IDS στην προστασία δικτύων και συστημάτων αλλά και οι τυχόν αδυναμίες τους που χρήζουν βελτίωσης.

Οι Cunningham,Lippmann,Fried[130] αξιολόγησαν τη χρησιμότητα των Συστημάτων Ανίχνευσης Εισβολών (IDS) στην πιθανότητα εντοπισμού επιθέσεων (P_d) και την πιθανότητα έκδοσης ψευδούς ειδοποίησης (P_{fa}). Τα δεδομένα της περιόδου δοκιμών των IDS αφορούσαν στην αποστολή αρχείων μέσω του δικτύου τα οποία το σύστημα καλούσε να αξιολογήσει ως επικίνδυνα ή μη. Οι συγκεκριμένοι μελετητές με τις δοκιμές επιθέσεων που διενήργησαν για την αξιολόγηση των IDS κατόρθωσαν ως ένα βαθμό να προσομοιώσουν τη λειτουργία ενός δικτύου και των επιθέσεων που μπορεί να δεχτεί. Ωστόσο, όπως τόνισαν, σε κάθε περίπτωση, η λειτουργικότητα και αποδοτικότητα ενός συστήματος ανίχνευσης εισβολών θα πρέπει να αξιολογείται σε τόσο σε φυσιολογική όσο και σε “επιθετική” κίνηση δικτύου. Τα δεδομένα κατέδειξαν ότι τα IDS αν και εντόπισαν όλες τις πιθανές επιθέσεις ωστόσο εμπεριείχαν ένα μεγάλο ποσοστό ψευδών ειδοποιήσεων. Επίσης, από την αξιολόγηση των δέκα (10) συνολικά IDS που επιλέχθηκαν, προέκυψε ότι εκείνα που είχαν σχεδιαστεί με βάση την τελευταία διαθέσιμη τεχνολογία, παρήγαγαν πιο ασφαλή αποτελέσματα.

Οι Alhomoud,Munira,Pagna[109] με τη σειρά τους αξιολόγησαν τα IDS στην παροχή προστασίας έναντι κακόβουλων επιθέσεων. Για την πραγματοποίηση της μελέτης τους επέλεξαν το Snort (v2.9.0.4) και το Suricata (v1.0.2).Η σειρά και το είδος των ελέγχων που πραγματοποιήθηκε στα συγκεκριμένα IDS ήταν τα ίδια με αντίστοιχες ρυθμίσεις

των συγκεκριμένων λογισμικών. Από τα στοιχεία που συγκέντρωσαν και βάσει των ελέγχων που πραγματοποίησαν προέκυψε ότι και τα δύο λογισμικά ανταποκρίθηκαν στην προστασία του συστήματος με υψηλά επίπεδα απόδοσης και εντοπισμού απειλών αν και χρησιμοποιούσαν διαφορετικές συλλογές υπογραφών. Τα κάθε λογισμικό λειτούργησε αποδοτικότερα ανάλογα με την πλατφόρμα, με την πλατφόρμα Linux να αποτελεί ιδανική επιλογή για το Suricata ανεξάρτητα από την ταχύτητα του δικτύου ενώ ιδανική πλατφόρμα για το Snort αποτελεί το FreeBSD για ταχύτητες έως 750 Mbps και FreeBSD για ταχύτητες μεγαλύτερες του 1,5 Gbps.

Οι Salah,Khiaty,Ahmed[131] διενήργησαν μια πειραματική αξιολόγηση και σύγκριση της απόδοσης του Snort όταν λειτουργεί στο πλαίσιο δύο (2) διαφορετικών λειτουργικών συστημάτων (Windows 7, Windows Server 2008) υπό διαφορετικές διαμορφώσεις του συστήματος αναφορικά με τον επεξεργαστή. Από τα στοιχεία που συγκέντρωσαν προέκυψε αρχικά ότι η απόδοση του Snort στο πλαίσιο των κακόβουλων επιθέσεων κινήθηκε σε υψηλά επίπεδα αλλά παρουσιάστηκαν διαφοροποιήσεις. Ειδικότερα, αποδείχθηκε ότι το Snort σε περιβάλλον SMP (Symmetric Multiprocessor) παρουσιάζει σημαντικά καλύτερη απόδοση από ό,τι σε περιβάλλον UP (Uniprocessor). Το συγκεκριμένο στοιχείο καταδεικνύει ότι η απόδοση ενός IDS εξαρτάται από το λειτουργικό σύστημα αλλά και τα χαρακτηριστικά του επεξεργαστή που επιλέγεται για την εγκατάσταση αυτού.

Ο Μπαλής [132] στα πλαίσια αξιολόγησης των IDS, διενήργησε συγκριτικές μετρήσεις αναφορικά με την ακρίβεια ανίχνευσής τους. Ειδικότερα, οι συγκριτικές μετρήσεις πραγματοποιήθηκαν με τη χρήση δύο αναγνωρισμένων IDS, του Snort και του Suricata. Τα στοιχεία στα οποία βασίστηκε για την αποτίμηση των συγκεκριμένων εργαλείων σε θέματα εποπτείας της εξερχόμενης κίνησης δικτύου ήταν ο βαθμός κάλυψης έναντι γνωστών επιθέσεων με τη χρήση συλλογής υπογραφών, οι δυνατότητες και πιθανότητες ανίχνευσης και αναγνώρισης επιθέσεων αξιοποιώντας το εργαλείο Pytbull, οι πιθανότητες ανίχνευσης επιθέσεων που χρησιμοποιούν τεχνικές αποφυγής ανίχνευσης και η αντοχή έναντι επιθέσεων τύφλωσης με παραγωγή μεγάλου πλήθους ψευδώς θετικών ειδοποιήσεων. Για την αξιολόγηση των συγκεκριμένων IDS σχεδιάστηκαν και πραγματοποιήθηκαν επαναλαμβανόμενες δοκιμές μέσω της δημιουργίας συγκρίσιμων ποσοτικών μετρήσεων. Από τα στοιχεία που συγκεντρώθηκαν, προέκυψε σημαντική λειτουργική ομοιότητα του Snort και του Suricata (παραμετροποίηση, υπογραφές κ.ά.) με βασικό σημείο διαφοροποίησης του

Suricata την αξιοποίηση του πολυνηματικού προγραμματισμού στην ανίχνευση. Ωστόσο, το Suricata υστερεί έναντι του Snort στο επίπεδο αξιοποίησης των υπαρχόντων υπογραφών. Από την πειραματική σύγκριση των στοιχείων ανέκυψε ότι και τα δύο IDS εντοπίζουν άμεσα και με ακρίβεια ενδεχόμενες κακόβουλες επιθέσεις με το Snort να υπερτερεί ελαφρώς σε επίπεδο αποτελεσματικότητας κυρίως λόγω του αυξημένου αριθμού υπογραφών. Σε επίπεδο τεχνικών αποφυγής ανίχνευσης το Snort παρουσίασε αντοχή της τάξης του 90% έναντι 56% του Suricata , έναντι των επιθέσεων τύφλωσης τόσο το Snort όσο και το Suricata ήταν ιδιαίτερα επιρρεπή στην παραγωγή υψηλού αριθμού ψευδώς θετικών ειδοποιήσεων (40.000 – 50.000 ειδοποιήσεις/15 min).

Οι Ridho,Yasin,Sulistyo[133] αξιολόγησαν τρία (3) διαφορετικά IDS και συγκεκριμένα το Snort, το Bro και το Suricata σε περιβάλλον Linux. Για την αξιολόγηση της απόδοσης των συγκεκριμένων IDS χρησιμοποιήθηκαν διαφορετικές IP ενώ από τα στοιχεία που συγκεντρώθηκαν προέκυψε ότι και τα τρία (3) IDS επέδειξαν υψηλή αποδοτικότητα στην αποτροπή εισβολών. Συγκεκριμένα, με βάση σαρώσεις και ελέγχων εισβολής το Snort ανέφερε 926 περιπτώσεις ειδοποιήσεων εκ των οποίων οι 8 υψηλού, 29 μέσου και 889 χαμηλού κινδύνου. Αντίστοιχα, το Suricata επί 1.218 ειδοποιήσεων, οι 22 αξιολογήθηκαν ως επικίνδυνες, οι 44 μέσης επικινδυνότητας και οι 1.152 χαμηλού κινδύνου. Τέλος, το Bro εντόπισε 128 περιπτώσεις κακόβουλης δραστηριότητας στο σύνολό τους χαμηλής επικινδυνότητας.

Ο Zanero [134] στην προσπάθεια αξιολόγησης των συστημάτων IDS/IPS επανεξέτασε δοκιμές εισβολής και πώς μπορούν να οδηγήσουν σε παραπλανητικά αποτελέσματα ή ακόμα και να στεφθούν με επιτυχία. Ο συγκεκριμένος ερευνητής σημείωσε πως μεταξύ των IDS/IPS παρατηρούνται αρκετές διαφορές (τρόποι ελέγχου, πρωτόκολλα, υπογραφές κ.ά.). Βάση ελέγχων που διενήργησε για την αποδοτικότητα και την αξιοπιστία τους έναντι κακόβουλων επιθέσεων, σημειώνει ότι τα IDS και τα IPS ακόμα παρουσιάζουν προβλήματα απόδοσης τέτοιας ώστε να μην αποτελούν την απόλυτη λύση έναντι των επιθέσεων και εισβολών. Ανάλογα στοιχεία εντόπισαν και οι Milenkoski,Payne,Antunes[135] οι οποίοι αξιολόγησαν τα IDS ως εργαλεία αποτροπής κακόβουλων εισβολών και επιθέσεων εστιάζοντας στο Xenini που ακολουθεί τεχνική ανίχνευσης ανωμαλιών.

Αντίστοιχη αξιολόγηση των IDPS πραγματοποίησε και ο Fekolkin [112] επιλέγοντας τη σύγκριση απόδοσης των Snort και Suricata. Από τη συγκριτική αξιολόγηση της

απόδοσής τους προέκυψε ότι ο πολυνηματικός χαρακτήρας του Suricata το καταστεί πιο απαιτητικό σε επίπεδο λειτουργίας έναντι του Snort ενώ του δίνει μεγαλύτερες δυνατότητες βελτιώσεων μέσω της επεκτασιμότητας και των αναβαθμίσεων. Επίσης, σε αντίθεση με το Snort, το Suricata χρησιμοποιεί αυτοματοποιημένη ανίχνευση πρωτόκολλου η οποία αυξάνει τις πιθανότητες ανίχνευσης κακόβουλου λογισμικού και ταυτόχρονα μειώνει τις ψευδώς θετικές ειδοποιήσεις αν και το Snort υπερτερεί ελαφρώς σε επίπεδο αποτελεσματικότητας κυρίως λόγω του αυξημένου αριθμού υπογραφών.

Κεφάλαιο 8
Συμπεράσματα

8.1 Καθυστέρηση στο δίκτυο

Με σκοπό να μετρήσουμε την επίπτωση του Suricata στο δίκτυο , εκτελέσαμε τα σενάρια που αναφέρονται στο Παράρτημα Α τόσο με απενεργοποιημένο όσο και με ενεργοποιημένο το Suricata. Μελετώντας τους πίνακες του Κεφαλαίου 6.3.1 διαπιστώσαμε ότι το Suricata επιφέρει μικρή καθυστέρηση στο δίκτυο και μόνο σε ορισμένες περιπτώσεις. Πιο συγκεκριμένα , εκτελώντας τα σενάρια μεμονωμένα , παρουσιάζεται μικρή καθυστέρηση στον μέσο όρο ανά αίτημα και συνολικό χρόνο εκτέλεσης σεναρίου μόνο σε σενάρια που προκαλούν μεγάλο αριθμό αποτυχημένων αιτήσεων. Παρομοίως , τα ίδια σενάρια παρουσιάζουν μικρή μείωση στον ρυθμό μετάδοσης των δεδομένων και τον μέσο όρο ρυθμού μετάδοσης των πακέτων. Εκτελώντας τα σενάρια χωρισμένα σε τετράδες (αποτυχημένες – επιτυχημένες προσπάθειες) , μικρή καθυστέρηση εμφανίστηκε στον συνολικό χρόνο εκτέλεσης μόνο σε ένα από τα τέσσερα σενάρια που προκαλεί μεγάλο αριθμό αποτυχημένων αιτήσεων ενώ η διαφορά στον ρυθμό μετάδοσης των δεδομένων και τον μέσο όρο ρυθμού μετάδοσης των πακέτων είναι μικρή. Αντιθέτως , εκτελώντας τα σενάρια παράλληλα , παρουσιάζεται μεγαλύτερη μείωση στον μέσο όρο ανά αίτημα και στον συνολικό χρόνο σεναρίου στα σενάρια που προκαλούν επιτυχημένες αιτήσεις ενώ η διαφορά στον ρυθμό μετάδοσης δεδομένων και τον μέσο όρο ρυθμού μετάδοσης πακέτων είναι σχετικά μικρή. Η αντίθεση που εμφανίστηκε σε αυτήν την περίπτωση για τα σενάρια που προκαλούν επιτυχημένες αιτήσεις εμφανίστηκε λόγω του ότι τα 8 σενάρια εκτελέστηκαν από μία εικονική μηχανή , η κάρτα δικτύου της οποίας οποία κλήθηκε να στείλει και να λάβει μεγάλο αριθμό πακέτων με αποτέλεσμα να προκαλέσει καθυστέρηση.

Συμπεραίνουμε λοιπόν ότι η καθυστέρηση ενεργοποιώντας το Suricata ήταν ελάχιστη και προκλήθηκε μόνο στα σενάρια που προκαλούν μεγάλο αριθμό

αποτυχημένων αιτήσεων , οι οποίες με τις σειρά τους παράγουν περισσότερους εντοπισμένους κινδύνους στο Suricata.

8.2 Ανάγκη σε υπολογιστική ισχύ (επεξεργαστής-μνήμη)

Μελετώντας του πίνακες του Κεφαλαίου 6.3.2 κατά σενάριο αλλά και τους συγκεντρωτικούς του Κεφαλαίου 6.4 διαπιστώνουμε ότι η ανάγκη του Suricata σε επεξεργαστή – μνήμη κινήθηκε σε σχετικά μικρά έως μέτρια ποσοστά. Συγκεκριμένα , ο επεξεργαστής κυμάνθηκε σε τιμές Min 1% , Max 39.4% συνολικά για όλες τις εκτελέσεις των σεναρίων , ενώ η μνήμη εμφάνισε τιμές Min 17,8% , Max 23.1%. Αυξημένη ανάγκη σε επεξεργαστική ισχύ προκλήθηκε κατά την εκτέλεση των σεναρίων με μεγάλο αριθμό αποτυχημένων αιτήσεων , οι οποίες προκάλεσαν με την σειρά τους μεγάλο αριθμό εντοπισμένων κινδύνων στο αρχείο καταγραφής fast.log του Suricata.

8.3 Στατιστικά Suricata – εντοπισμένοι κίνδυνοι – υπογραφές

Μελετώντας αρχικά τα στατιστικά που παρουσιάζονται στους πίνακες του κεφαλαίου 6.3.2 και εξήχθησαν από τα αρχεία καταγραφής stats.log του Suricata , εντυπωσιακό είναι το γεγονός ότι δεν καταγράφηκε κάποιο χαμένο πακέτο σε επίπεδο πυρήνα , σε καμία εκτέλεση του Suricata. Αντιθέτως , το Suricata ήταν σε θέση να εποπτεύσει και να αποκωδικοποιήσει το σύνολο των πακέτων σύμφωνα με τις τιμές των capture.kernel_packets και decoder.pkts αντίστοιχα. Ο αριθμός των συνεδριών tcp (tcp.sessions) ήταν αντίστοιχος με την κίνηση που προκαλούσε η εκτέλεση κάθε σεναρίου ενώ ο δείκτης tcp.invalid_checksum παρέμεινα σε πολύ χαμηλά επίπεδα. Διακρίνεται επίσης εύκολα ότι ο δείκτης tcp.reassembly_gap που υποδεικνύει τον αριθμό των χαμένων πακέτων δεδομένων στην ροή δεδομένων tcp παρέμεινε χαμηλός για μικρό αριθμό >=150 τόσο επιτυχημένων όσο και αποτυχημένων αιτήσεων ενώ εμφανίστηκε ιδιαίτερα μεγάλος για μεγάλο αριθμό >=30K αποτυχημένων αιτήσεων. Για τις ίδιες περιπτώσεις μεγάλου αριθμού >=30K αποτυχημένων αιτήσεων, εντοπίζεται ότι ο δείκτης των εντοπισμένων κινδύνων(detect.alert) έχει ανάλογη αύξηση με αυτήν του δείκτη tcp.reassembly_gap.

Σύμφωνα με τον κατασκευαστή[136], η τιμή του συγκεκριμένου δείκτη είναι μηδέν σε ιδανικές συνθήκες , αλλά επηρεάζεται τόσο από τον αριθμό των χαμένων πακέτων ,

των πακέτων με λανθασμένο checksum , και την εξάντληση μνήμης στην μηχανή ροής δεδομένων του Suricata. Καθώς εξάντληση μνήμης δεν παρατηρήθηκε σε μία εκτέλεση του Suricata και τα πακέτα με λανθασμένο checksum ήταν λίγα ενώ ταυτόχρονα δεν καταγράφηκε κάποιο χαμένο πακέτο σε επίπεδο πυρήνα του Suricata, ο συγκεκριμένος δείκτης επηρεάστηκε από τα πακέτα που δέχτηκε η κάρτα δικτύου της εικονικής μηχανής που εγκαταστάθηκε το Suricata.

Από τα αρχεία καταγραφής fast.log που παρουσιάζονται στο κεφάλαιο 6.3.2 , διαπιστώνουμε ότι το Suricata ήταν σε θέση να εντοπίσει και να καταγράψει κινδύνους αντίστοιχους με τα σενάρια που εκτελέστηκαν και αφορούσαν κίνηση tcp , συγκεκριμένα πρωτόκολλα Http (πόρτα 80) , ftp (πόρτα 21) & Mysql (πόρτα 3306). Πιο συγκεκριμένα , ενεργοποιήθηκαν περισσότερες από 15 διαφορετικές υπογραφές από 7 διαφορετικά αρχεία κανόνων του Suricata ενώ εντοπίστηκαν σημαντικοί κίνδυνοι όπως επίθεση ωμής βίας(bruteforce attack), σάρωση εφαρμογής ιστού (web application scan, παραβίαση εταιρικής πολιτικής (corporate policy violation) κ.α.

Αξίζει να σημειωθεί ότι οι 500 επιτυχημένες αιτήσεις HTTP δεν αντιστοιχήθηκαν με κάποια υπογραφή της κατηγορίας web ή scan του Suricata , επομένως δεν εντοπίστηκε κάποιος κίνδυνος γι αυτές , ενώ κατά την εκτέλεση του σεναρίου Γ , όπου εκτελέστηκαν επιτυχημένες αιτήσεις μεταφόρτωσης αρχείου μέσω FTP , οι κίνδυνοι που καταγράφηκαν από το Suricata , υποδείκνυαν την εκτέλεση εντολών FTP CWD , NLST & STOR χωρίς να έχει πραγματοποιηθεί login γεγονός που δεν ίσχυε.

Η πλειοψηφία των κινδύνων που καταγράφηκαν στα αρχεία fast.log του Suricata κατά την εκτέλεση όλων των σεναρίων, σχετίζεται με υπογραφές που ελέγχουν την ροή δεδομένων και αντιστοιχούν στην κατηγορία stream.events.

Οι συγκεκριμένοι κίνδυνοι όπως εμφανίστηκαν στα αρχεία καταγραφής αφορούν στην πλειοψηφία τους το TCP 3-way handshake και τα μηνύματα (SYN,SYN-ACK,ACK). Το γεγονός αυτό οφείλεται στην τεχνολογία RSS(Receive side scaling) , η οποία , χρησιμοποιώντας έναν ασύμετρο αλγόριθμο κατακερματισμού , επιτρέπει στον οδηγό της κάρτας δικτύου να διανέμει την κίνηση σε διαφορετικές ουρές αυτής με σκοπό την βελτίωση της απόδοσης.

Το πρόβλημα αναφέρεται από τον κατασκευαστή [137] και έγκειται στο γεγονός ότι εάν η κίνηση διανέμεται σε διαφορετικές ουρές , η σειρά με την οποία τα πακέτα εποπτεύονται γίνεται απρόβλεπτη και λόγω της διαφοράς χρονισμού της κάρτας δικτύου , του πυρήνα του λειτουργικού και του ίδιου του Suricata , υπάρχει μεγάλη

πιθανότητα τα μηνύματα SYN/ACK να λαμβάνονται από το Suricata αφού έχει ξεκινήσει η αποστολή των δεδομένων από την πηγή προς τον προορισμό , με αποτέλεσμα η κίνηση αυτή να καταγράφεται σαν κίνδυνος. Η λύση που προτείνεται από τον κατασκευαστή είναι είτε η ρύθμιση της κάρτας δικτύου σε συμμετρική λειτουργία , είτε ο περιορισμός των ουρών RSS σε αυτήν σε μία, γεγονός που δεν υποστηρίζεται από όλους τους οδηγούς των καρτών δικτύου, όπως δεν υποστηριζόταν και από τον οδηγό της εικονικής μηχανής που εγκαταστάθηκε το Suricata.

Επίλογος

Η αναμφισβήτητη αύξηση στην ανάπτυξη κακόβουλου λογισμικού καθώς η ύπαρξη κενών ασφαλείας στις εφαρμογές και τα λειτουργικά συστήματα αποτελούν απειλές που μπορούν να θέσουν σε κίνδυνο την εμπιστευτικότητα, την ακεραιότητα, την ιδιωτικότητα και την διαθεσιμότητα των δεδομένων μιας επιχείρησης ή ενός οργανισμού.

Την ανάγκη για διασφάλιση όλων των παραπάνω καλούνται να καλύψουν τα συστήματα ανίχνευσης εισβολών , τόσο για κινδύνους που προέρχονται από το εξωτερικό όσο και από το εσωτερικό δίκτυο. Γι αυτό το λόγο , η χρήση συστήματος IDS θεωρείται best practice σε επιχειρήσεις ή οργανισμούς μεσαίου και μεγάλου μεγέθους.

Στην παρούσα μεταπτυχιακή διατριβή , εξετάστηκε το λογισμικό Suricata που βασίζεται σε ανίχνευση εισβολών βάση αντιστοίχισης υπογραφών. Η τοπολογία του δικτύου της πειραματικής μελέτης ήταν τέτοια έτσι ώστε να επιτρέπει στο Suricata την εποπτεία της κίνησης στο εσωτερικό δίκτυο.

Προκειμένου να ελεγχθεί η απόδοση του Suricata , αναπτύχθηκαν python scripts με σκοπό την αναπαραγωγή κίνησης TCP βασιζόμενων στα πρωτόκολλα HTTP, FTP & MySQL. Πιο συγκεκριμένα , εκτελώντας τα scripts παράλληλα και δημιουργώντας 60.250 επιτυχημένες / αποτυχημένες αιτήσεις , για ένα σύνολο 18837 υπογραφών που φορτώθηκαν κατά την έναρξή του το Suricata ήταν σε θέση να αποκωδικοποιήσει και να εποπτεύσει 2.582.437 πακέτα με ταχύτητα ~ 1548 πακέτα το δευτερόλεπτο.

Το Suricata αποδείχτηκε ότι δεν επιβαρύνει την απόδοση του δικτύου ενώ η ανάγκη του σε υπολογιστική ισχύ περιορίστηκε σε χαμηλά έως μέτρια επίπεδα. Εντοπίστηκαν σημαντικοί κίνδυνοι όπως επίθεση ωμής βίας , σάρωση εφαρμογής ιστού , παραβίαση εταιρικής πολιτικής κ.α. Μεγάλος αριθμός κινδύνων που καταγράφηκαν από το Suricata , οφείλονταν στην τεχνολογία του RSS (Receive side scaling) , το οποίο είναι ενεργοποιημένο σε όλες τις σύγχρονες κάρτες δικτύου.

Συμπερασματικά , για την ορθή λειτουργία συστήματων IDS που έρχονται με την μορφή λογισμικού τύπου Suricata , πρέπει να λαμβάνονται πολλοί παράμετροι υπόψη ξεκινώντας από το λειτουργικό σύστημα και τα πρωτόκολλα και καταλήγοντας μέχρι τις τεχνολογίες της κάρτας δικτύου. Το Suricata αποδείχτηκε ικανό να ελέγξει την κίνηση του εσωτερικού δικτύου που υποβλήθηκε αλλά οι διαφορετικές λειτουργίες και οι παράμετροι που υποστηρίζει χρήζουν μελλοντικής έρευνας.

Βιβλιογραφία

01. F. Barrere, A. Benzekri, F. Grasset, R. Laborde, B. Nasser. "Spidernet : A Security Policy Derivation Tool For Networks". 3rd Ieee Latina America Network Operations And Management Symposium, p. 8, 2003.

02. M. Pawar, J. Anuradha. "Network Security And Types Of Attacks In Network". Procedia Computer Science, 48(1), 503-506, 2015.

03. H. Schumacher, S. Ghosh. "A Fundamental Framework For Network Security". Journal Of Network And Computer Applications, 20(3), 305-322, 1997.

04. M. Prabhu, S. Raghavan, S. "Security In Computer Networks And Distributed Systems". Computer Communications, 19(5), 379-388, 1996.

05. Nexcom. "Network Security Appliance". Available At: Http://Www.Nexcom.Com/Applications/Detailbydivision/Network-Security-Appliance. [Access: 08.03.2017], 2011

06. W. Stalling. «Βασικές Αρχές Ασφάλειας Δικτύων». 3η Αμερικανική Έκδοση. Εκδόσεις Κλειδάριθμος, Αθήνα, σ. 16, 2011.

07. C. Kaufman, R. Perlman, M. Speciner. "Network Security: Private Communication In A Public World". Prentice – Hall, USA, p. 24-25, 2002.

08. R. Laborde, B. Nasser, F. Grasset, F. Barrère, A. Benzekri. "A Formal Approach For The Evaluation Of Network Security Mechanisms Based On Rbac Policies". Electronic Notes In Theoretical Computer Science, 121(1),117-142, 2005.

09. R. Pareek. "Network Security: An Approach Towards Secure Computing". Journal Of Global Research In Computer Science, 2(7),160-163, 2012.

10. M. Curtin. "Introduction To Network Security". Kent Information Services, p. 1-3, 1997.

11. S. Mcclure, J. Scambray, G. Kurz. "Hacking Exposed". 3rd Edition. Mcgraw Hill, USA, p. 8-11, 2001.

12. G. Nazer, A. Selvakumar. "Current Intrusion Detection Techniques In Information Technology - Adetailed Analysis". Eur. J. Sci. Res., 65(1), 611–624, 2011.

13. A. Patcha, J. Park. "An Overview Of Anomaly Detection Techniques: Existing Solutions And Latest Technological Trends". Comput. Netw., 51(12), 3448–3470, 2007.

14. M. Hoque, M. Mukit, M. Bikas. "An Implementation Of Intrusion Detection System Using Genetic Algorithm". Int. J. Netw. Secur. Appl. 4(2), 111–112, 2012.

15. S. Beg, U. Naru, M. Ashraf, S. Moshin. "Feasibility Of Intrusion Detection System With High Performance Computing: A Survey". Int. J. Adv. Comput. Sci., 1(1), 26–35, 2010.

16. Q. Wu. "The Research And Application Of Firewall Based On Netfilter". Physics Procedia, 25(1), 1231-1235, 2012.

17. F. Shiri, B. Shanmugam, N. Idris. "A Parallel Technique For Improving The Performance Of Signature-Based Network Intrusion Detection System". Proceedings Of 3rd International Conference Communication Software And Networks, Iccsn, Ieee, p. 694, 2011.

18. On Time. "Networking". Available At: Http://Ontimems.Com/Networking. . [Access: 10.03.2017], 2017.
19. J. Shukla, G. Singh, P. Shukla, A. Tripathi. "Modeling And Analysis Of The Effects Of Antivirus Software On An Infected Computer Network". Applied Mathematics And Computation, 227(1),11–18, 2014.
20. B. Mishra, S. Pandey. "Effect Of Anti - Virus Software On Infectious Nodes In Computer Network: A Mathematical Model". Physics Letters A, 376(35), 2389–2393, 2012.
21. A. Safarkhanlou, A. Souri, M. Norouzi, H. Sardroud. "Formalizing And Verification Of An Antivirus Protection Service Using Model Checking". Procedia Computer Science, 57(1), 1324-1331, 2015.
22. David. "An Analytical Approach To Securing Your Network". Https://Blog.Zen.Co.Uk/An-Analytical-Approach-To-Securing-Your-Network. [Access: 27.02.2017], 2013.
23. A. Menezes, P. Van Oorschot, S. Vanstone. "Handbook Of Applied Cryptography". Crc Press, USA, p. 67-68, 2001,
24. L. Singh, K. Singh. "Implementation Of Text Encryption Using Elliptic Curve Cryptography". Procedia Computer Science, 54(1), 73-82, 2015
25. Ε. Ζάχος, Α. Παγουρτζής, Π. Γροντάς. «Υπολογιστική Κρυπτογραφία». ΣΕΑΒ, Αθήνα, σελ. 27-28, 2015.
26. Geex. "What Is Data Encryption And Why Is It Needed Over Public Networks?" Available At: Http://Www.Geex.In/What-Is-Data-Encryption-And-Why-Is-It-Needed-Over-Public-Networks. [Access: 22.02.2017], 2016.
27. W. Bul'ajoul, A. James, M. Pannu. "Improving Network Intrusion Detection System Performance Through Quality Of Service Configuration And Parallel Technology". Journal Of Computer And System Sciences, 81(6),981–999, 2015.
28. K. Scarfone, P. Mell. "Guide To Intrusion Detection And Prevention Systems (Idps)". National Institute Of Standards And Technology, USA, Nist Special Publication, p. 65-69, 2012.
29. Juniper. "What is Intrusion Detection and Prevention (IDS/IDP)?". Available at: http://www.juniper.net/us/en/products-services/what-is/ids-ips/[Access: 22.03.2017], 2016.
30. J. Jabez, B. Muthukumar. "Intrusion Detection System (Ids): Anomaly Detection Using Outlier Detection Approach". Procedia Computer Science, 48(1), 338-346, 2015.
31. A. Aburomman, M. Reaz. "A Survey Of Intrusion Detection Systems Based On Ensemble And Hybrid Classifiers". Computers & Security, 65(1), 135-152, 2017.
32. T. Sobh. "Wired And Wireless Intrusion Detection System: Classifications, Good Characteristics And State – Of – The – Art". Computer Standards & Interfaces, 28(6), 670–694, 2006.
33. M. Tiggelaar. "Who Needs An Intrusion Detection System?" Available At: Https://Www.Key4ce.Com/About/Blog/Business-Insights/Needs-Intrusion-Detection-System. [Access: 06.03.2017], 2004.
34. P. Rajendran, B. Muthukumar, G. Nagarajan. "Hybrid Intrusion Detection System For Private Cloud: A Systematic Approach". Procedia Computer Science, 48(1), 325-329, 2015.
35. B. Zarpelão, R. Miani, C. Kawakani, S. Alvarenga. "A Survey Of Intrusion Detection In Internet Of Things". Journal Of Network And Computer Applications, In Press, p. 2-3, 2017.

36. A. Abdullah. "Design Intrusion Detection System Based On Image Block Matching". International Journal Of Computer And Communication Engineering, Iacsit Press, 2(5), 605-607, 2013.
37. P. Mishra, E. Pilli, V. Varadharajan, U. Tupakula. "Intrusion Detection Techniques In Cloud Environment: A Survey". Journal Of Network And Computer Applications, 77(1), 18-47, 2017.
38. R. Kemmerer, G., Vigna. "Intrusion Detection: A Brief History And Overview" University Of California Santa Barbara, Computer Science Department, Supplement To Computer Security And Privacy, USA, p. 28, 2002.
39. Z. Zhang, W. Lee, Y. Huang. "Intrusion Detection Techniques For Mobile Wireless Networks". Acm Wireless Networks, 9(1), 545– 556, 2003.
40. S. Chari, P. Cheng. "Bluebox: A Policy - Driven, Host – Based Intrusion Detection System". Acm Transactions On Information And System Security, 6(2), 1-6, 2003.
41. A. Apostolico, Z. Galil. "Pattern Matching Algorithms". Oxford University Press, USA, p. 66, 1997.
42. M. Norton. "Optimizing Pattern Matching For Intrusion Detection". Available At: Https://S3.Amazonaws.Com/Snort-Org-Site/Production/Document_Files/Files/000/000/085/Original/Optimizingpatternmatchingforids.Pdf?Awsaccesskeyid=Akiaixacied2spmsc7ga&Expires=1489405819&Signature=K3yhi4kckbxcfn%2binsvrjpddstc%3d. [Access: 21.02.2017], 2001.
43. J. Sung, S. Kang, T. Kwon. "A Fast Pattern - Matching Algorithm For Network Intrusion Detection System". International Conference On Research In Networking, p. 1162, 2006.
44. R. Rehman. "Intrusion Detection Systems With Snort. Advanced Ids Techniques Using Snort, Apache, Mysql, Php, And Acid". Upper Saddle River, New Jersey, p. 8, 2003.
45. S. Patil, P. Kulkarni, P. Rane, B. Meshram. "Ids Vs Ips". International Journal Of Computer Networks And Wireless Communications (Ijcnwc), 2(1), 86-90, 2012.
46. M. Su, G. Yu, G. Lin. "A Real - Time Network Intrusion Detection System For Large-Scale Attacks Based On An Incremental Mining Approach". Computers & Security, 28(5), 301–309, 2009.
47. S. Northcutt. "Network Intrusion Detection: An Analysts Handbook". New Riders, Indianapolis, p. 101-112, 2001.
48. S. Gautam, H. Om. "Computational Neural Network Regression Model For Host Based Intrusion Detection System". Perspectives In Science, 8(1), 93–95, 2016.
49. A. Patel, M. Taghavi, K. Bakhtiyari, J. Celestinou. "An Intrusion Detection And Prevention System In Cloud Computing: A Systematic Review". Journal Of Network And Computer Applications, 36(1),25–41, 2013.
50. T. Escamilla. "Intrusion Detection: Network Security Beyond The Firewall". John Wiley, New York, p. 84, 1998.
51. K. Cox. "Managing Security With Snort And Ids Tools". O΄ Reilly, Cambridge, p. 65-66, 2004.
52. E. Amiri, H. Keshavarz, H. Heidaric, E. Mohamadid, H. Moradzadehe. "Intrusion Detection Systems In Manet: A Review". Procedia - Social And Behavioral Sciences, 129(1), 453 – 459, 2014.

53. K. Scarfone, P. Mell. “Guide To Intrusion Detection And Prevention Systems (Idps)". Computer Security Resource Center. National Institute Of Standards And Technology, 1(1), 800–894, 2007.

54. G. Judge. “Fpga Architecture Ups Intrusion Detection Performance”. Available At:Http://Www.Eetimes.Com/Document.Asp?Doc_Id=1271942, [Access: 11. 03.2017], 2003.

55. H. Liao, L. Chun - Hung, L. Ying-Chih, T. Kuang – Yua. “Intrusion Detection System: A Comprehensive Review”. Journal Of Network And Computer Applications, 36(1),16–24, 2013.

56. I. Corona, G. Giacinto, F. Roli. “Adversarial Attacks Against Intrusion Detection Systems: Taxonomy, Solutions And Open Issues”. Information Sciences, 239(1), 201–225, 2013.

57. F. Sabahi, A. Movaghar. “Intrusion Detection: A Survey”. 3rd International, p. 24-25, 2008.

58. P. Stavroulakis, M. Stamp. “Handbook Of Information And Communication Security”. Springer - Verlag, New York, p. 66-71, 2010.

59. F. Maggi, M. Matteucci, S. Zanero. “Reducing False Positives In Anomaly Detectors”. Information Fusion, 10(4), 300-311, 2009.

60. C. Modi, D. Patel, B. Borisaniya, H. Patel, A. Patel, M. Rajarajan. “A Survey Of Intrusion Detection Techniques In Cloud”. Journal Of Network And Computer Applications, 36(1),42–57, 2013.

61. P. Fogla, M. Sharif, R. Perdisci, O. Kolesnikov, W. Lee. “Polymorphic Blending Attacks”. Usenix Security Symposium, p. 245, 2006.

62. H. Kayacik, A. Zincir - Heywood, M. Heywood. “Automatically Evading Ids Using Gp Authored Attacks”. Symposium On Computational Intelligence In Security And Defense Applications, USA, p. 155, 2007.

63. C. Kolbitsch, T. Holz, C. Kruegel, E. Kirda. “Inspector Gadget: Automated Extraction Of Proprietary Gadgets From Malware Binaries”. Ieee Symposium On Security And Privacy,Ieee Computer Society, USA, p. 31, 2010.

64. S. Patton, W. Yurcik, D. Doss. “An Achilles’ Heel Signature - Based Ids: Squealing False Positives In Snort”. Proceedings Of 4th International Symposium On Recent Advances Inintrusion Detection, 10(1),12, 2011.

65. W. Yurcik. “Controlling Intrusion Detection Systems By Generating False Positives: Squealing Proof – Of – Concept”. Proceedings Of The 27th Annual Ieee, Conference On Local Computer Networks (Lcn), USA, p. 134-135, 2002.

66. D. Mutz, G. Vigna, R. Kemmerer. “An Experience Developing An Ids Stimulator For The Black - Box Testing Of Network Intrusion Detection Systems”. Proceedings Of The 19th Annual Computer Security Applications Conference, USA, p. 376, 2003.

67. P. Chan, R. Lippmann. “Machine Learning For Computer Security”. Journal Of Machine Learning Research, 7(1), 2669–2672, 2006.

68. M. Barreno, P. Bartlett, F. Chi, A. Joseph, B. Nelson, B. Rubinstein, U. Saini, D. Tygar. "Open Problems In The Security Of Learning (Position Paper)". Proc. 1st Acm Workshop On Aisec (Aisec 2008). The Association For Computing Machinery, Inc., p. 19-26, 2008.

69. B. Hernacki, J. Bennett, J. Hoagland. “An Overview Of Network Evasion Methods”. Information Security Technical Report, 10(3), 140–149, 2005.

70. A. Carlin, M. Hammoudeh, O. Aldabbas. "Defence For Distributed Denial Of Service Attacks In Cloud Computing". Procedia Computer Science, 73(1), 490-497, 2015.
71. T. Buchheim, B. Feinstein, D. Gupta, G. Matthews, R. Pollock, R. "Iap: Intrusion Alert Protocol". Available at: https://Tools.Ietf.Org/Html/Draft-Ietf-Idwg-Iap-05. [Access: 19.02.2017], 2001,
72. J. Betser, A. Walther, M. Erlinger, T. Buchheim, B. Feinstein, G. Matthews, R. Pollock, K. Levitt. "Global Guard: Creating The Ietf - Idwg Intrusion Alert Protocol (Iap), Darpa Information Survivability Conference, Exposition Ii, 2001. Discex. Proceedings, Iee., p. 6, 2001.
73. R. Fielding, J. Gettys, J. Mogul, H. Nielson, L. Masinter, P. Leach, T. Berners – Lee. "Rfc 2616: Hypertext Transfer Protocol — Http/1.1". Network Working Group, p. 21-22, 1999.
74. A. Elgohary, T. Sobh, M. Zaki. "The Tls Protocol Version". Computers & Security. 25(4), 297–306, 2006.
75. E. Lear. "Using The Netconf Protocol Over The Blocks Extensible Exchange Protocol (Beep), 2006, Network Working Group". Available At: Https://Tools.Ietf.Org/Html/Rfc4744. [Access: 09.03.2017], 2006.
76. B. Feinstein, G. Matthews. "The Intrusion Detection Exchange Protocol (Idxp). Available At: Https://Www.Ietf.Org/Rfc/Rfc4767.Txt. [Access: 12.03.2017], 2007.
77. A. Abdelkarim, H. Nasereddin. "Intrusion Prevention System". International Journal Of Academic Research, 3(1), 432-434, 2011.
78. K. Karen. "Network Intrusion Detection Signatures, Part One". Available At: Https://Www.Symantec.Com/Connect/Articles/Network-Intrusion-Detection-Signatures-Part-One. [Access: 03.03.2017], 2001.
79. M. Mohammad, N. Sulaiman, O. Muhsin. "A Novel Intrusion Detection System By Using Intelligent Data Mining In Weka Environment". Procedia Computer Science, 3(1), 1237–1242, 2011.
80. H. Elshoush, I. Osman. "Alert Correlation In Collaborative Intelligent Intrusion Detection Systems – A Survey". Applied Soft Computing, 11(7), 4349–4365, 2011.
81. M. Aydın, A. Zaim, A. Ceylan. "A Hybrid Intrusion Detection System Design For Computer Network Security". Computers And Electrical Engineering, 35(3), 517–526, 2009.
82. P. Garcıa-Teodoro, J. Dıaz – Verdejo, G. Macia – Fernandez, E. Vazquez. "Anomaly - Based Network Intrusion Detection: Techniques, Systems And Challenges". Computers & Security, 28(1-2), 18–28, 2009.
83. P. Garcıa-Teodoro, J. Dıaz – Verdejo, G. Macia – Fernandez, E. Vazquez. "Evaluation of a low-rate DoS attack against iterative servers". Computer Networks, 51(1), 1013–1030, 2007.
84. A. Raut, K. Singh. "Anomaly Based Intrusion Detection –A Review". International Journal On Network Security, 5(1), 7-12, 2014.
85. J. Mirkovic, G. Prier, P. Reiher. "Attacking Ddos At The Source". Proceedings Of ICNP, Paris, p. 315, 2002.
86. R. Werlinger, K. Hawkey, K. Muldner, P. Jaferian, K. Beznosov. "The Challenges Of Using An Intrusion Detection System: Is It Worth The Effort?" Symposium On Usable Privacy And Security (Soups), USA, p. 3-4, 2008.

87. T. Jia, X. Wang. "The Research And Design Of Intelligent Ips Model Based On Dynamic Cloud Firewall Linkage". International Journal Of Digital Content Technology And Its Applications, 5(3):304-3099, 2011.
88. M. Ahmed, R. Pal, H. Hossain, M. Bikas, M. Hasan. "Nids: A Network Based Approach To Intrusion Detection And Prevention". Computer Science And Information Technology - Spring Conference, 1(1), 141-144, 2009.
89. L. Fagui, S. Xiang, L. Wenqianl. "The Design And Application Of Xen-Based Host System Firewall And Its Extension". The 2009 International Conference On Electronic Computer Technology, p. 393, 2009.
90. H. Tipton, M. Krause. "Information Security Management Handbook". Crc Press, p. 1000, 2007.
91. T. Boyles. "Ccna Security Study Guide: Exam 640-553". John Wiley And Sons, New Jersey, p. 249, 2010.
92. Pmg. "Maximizing The Value Of Network Intrusion Detection". A Corporate White Paper From The Product Management Ngroup Of Intrusion.Com, 2001.
93. S. Piper. "Intrusion Prevention Systems For Dummies". Wiley Publishing, New Jersye, p. 19-21, 2011.
94. N. Ierace, C. Urrution, R. Bassett, R. "Intrusion Prevention System". Available At: Http://Ubiquity.Acm.Org/Article.Cfm?Id=1071927 [Access 02.04.2017], 2005.
95. E. Cole, J. Fossen, S. Northcutt, H. Pomeranz, J. Wright. "Sans Security Essentials". Sans Institute, p. 15, 2006.
96. C. Corman. "Defining The Rules For Preemptive Host Protection". Internet Security Systems, p. 87, 2009.
97. A. Fuchsberger. "Intrusion Detection Systems And Intrusion Prevention Systems". Information Security Technical Report, 10(3), 134-139, 2005.
98. H. Raza, Z. Alli. "Investigation Of Solutions For Intrusion Prevention And Detection". Master's Programme In Computer Network Engineering, p. 16-18, 2003.
99. Z. Yujia, C. Guanlin, W. Wenyong, W. Zebing. "An Overview Of Wireless Intrusion Prevention Systems". Paper Presented At The Communication Systems, Networks And Applications (Iccsna), Second International Conference, 2010.
100. P. Puri, N. Chopde. "Comparative Analysis Of Hybrid Intrusion Detection System And Intrusion Prevention System For Manet." Index Copernicus Value, 4(4), 1966-1971, 2013.
101. O. Cepheli, S. Büyükçorak, G. Kurt. "Hybrid Intrusion Detection System For Ddos Attacks". Journal Of Electrical And Computer Engineering, 1(1), 1-8, 2016.
102. B. Agarwal, N. Mittal. "Hybrid Approach For Detection Of Anomaly Network Traffic Using Data Mining Techniques". Procedia Technology, 6(1), 996-1003, 2012.
103. I. Dubrawsky. "Chapter 7 - Topologies And Ids". Security+, p. 436, 2007.
104. Y. Wang, C. Lin, Q. Li, Y. Fang Y. "A Queueing Analysis For The Denial Of Service (Dos) Attacks In Computer Network". Computer Networks, 51(12), 3564-3573, 2007.
105. A. Prakash, M. Satish, T. Bhargav, N. Bhalaji. "Detection And Mitigation Of Denial Of Service Attacks Using Stratified Architecture". Procedia Computer Science, 87(1), 275-280, 2016.

106. S. Aljawarneh, R. Moftah, A. Maatuk. "Investigations Of Automatic Methods For Detecting The Polymorphic Worms Signatures". Future Generation Computer Systems, 60(1), 67-77, 2016.
107. D. Emm. "Focus On Trojans – Holding Data To Ransom". Network Security, 6(1), 4-7, 2006.
108. J. Foster, V. Osipov, N. Bhalla, N. Heinen, D. Aitel. "Chapter 8 - Windows Buffer Overflows". Buffer Overflow Attacks, p. 322, 2005.
109. A. Alhomoud, R. Munira, J. Pagna, I. Awan, A. Al - Dhelaanb. "Performance Evaluation Study Of Intrusion Detection Systems". Procedia Computer Science, 5(1), 173–180, 2011.
110. Suricata. "Features". Available At: Https://Suricata-Ids.Org/Features/ [Access 22.03.2017], 2016.
111. J. White, T. Fitzsimmons, J. Matthews. "Quantitative Analysis Of Intrusion Detection Systems: Snort And Suricata". In Spie Defense, Security, And Sensing. International Society For Optics And Photonic, 2013.
112. R. Fekolkin. "Intrusion Detection And Prevention Systems: Overview Of Snort And Suricata". Internet Security, p. 2-3, 2015.
113. Suricata User Guide, Performance , Runmodes Available at: http://suricata.readthedocs.io/en/latest/performance/runmodes.html?highlight=runmode. [Access 28.03.2017], 2017
114. Oisf (Open Information Security Foundation). "Suricata User Guide". Available At: Https://Redmine.Openinfosecfoundation.Org/Projects/Suricata/Wiki/Suricata_User_Guide. [Access: 26.03.2017], 2015.
115. Suricata. "All Features". Available At: Https://Suricata-Ids.Org/Features/All-Features/ [Access 28.03.2017], 2017.
116. M. Jonkman. "Suricata Ids Available For Download!". Available At: http://seclists.org/snort/2009/q4/599 [Access 01.04.2017], 2009.
117. Suricata User Guide, Performance , Packet Capture Available at : http://suricata.readthedocs.io/en/latest/performance/packet-capture.html#offloading [Access 20.03.2017], 2016.
118. D-ITG, Distributed Internet Traffic Generator , Available at: http://www.grid.unina.it/software/ITG/ [Access 15.02.17] , 2017.
119. IPERF, The ultimate speed test tool for TCP, UDP and SCTP , Available at: https://iperf.fr/ [Access 15.02.17] , 2017.
120. Suricata User Guide , Installation Available at: http://suricata.readthedocs.io/en/latest/install.html [Access 20.03.2017] , 2016.
121. PCRE- Perl Compatible Regular Expressions Available at: http://www.pcre.org/ [Access 15/02/17] , 2017.
122. Emerging Threats Open Rules Available at: https://rules.emergingthreats.net/open/ [Access 20.03.2017] , 2017.
123. Suricata User Guide, Rule Management with Oinkmaster Available at: http://suricata.readthedocs.io/en/latest/rule-management/oinkmaster.html. [Access 20.03.2017] , 2016.
124. Requests: HTTP for Humans , Available at http://docs.python-requests.org/en/master/ [Access 10/01/17] , 2017.

125. Python Documentation ftplib — FTP protocol client , Available at : https://docs.python.org/2/library/ftplib.html [Access 10/02/17] , 2017.

126. MySQLdb User's Guide, Available at: http://mysql-python.sourceforge.net/MySQLdb.html [Access 01/02/17] , 2017.

127. TCP Dump Man Page, Available at: http://www.tcpdump.org/tcpdump_man.html [Access 01/02/17] , 2017.

128. Linux commands Man Pages, Capinfos, Available at: http://linuxcommand.org/man_pages/capinfos1.html [Access 10/02/17] , 2017.

129. Github- suri-stats , A tool to work on suricata stats.log file, Available at: https://github.com/regit/suri-stats [Access 1/04/17] , 2017.

130. R. Cunningham, R. Lippmann, D. Fried, S. Garfinkel, I. Graf, K. Kendall, S. Webster, D. Wyschogrod, M. Zissman. "Evaluating Intrusion Detection Systems: The 1998 Darpa Off-Line Intrusion Detection Evaluation". IEEE, p. 26-42, 2000.

131. K. Salah, M. Al – Khiaty, R. Ahmed, A. Mahdi. "Performance Evaluation Of Snort Under Windows 7 And Windows Server 2008". Journal Of Universal Computer Science, 17(11), 1605-1622, 2011.

132. Α. Μπαλής. «Εγκατάσταση Και Πειραματική Μελέτη Συστημάτων Ανίχνευσης Εισβολών». Πανεπιστήμιο Πειραιώς, Τμήμα Ψηφιακών Συστημάτων, Πρόγραμμα Μεταπτυχιακών Σπουδών «Τεχνοοικονομική Διοίκηση & Ασφάλεια Ψηφιακών Συστημάτων», Κατεύθυνση «Ασφάλεια Ψηφιακών Συστημάτων» Διπλωματική Εργασία, , Μτε1117, Πειραιάς, σ. 1-215, 2013

133. F. Ridho, F. Yasin, Y. Sulistyo. "Analysis And Evaluation Snort, Bro, And Suricata As Intrusion Detection System Based On Linux Server". Universitas Muhammadiyah Surakarta, Eng, Department Of Informatics, p. 4-6, 2014.

134. S. Zanero. "Flaws And Frauds In The Evaluation Of Ids/Ips Technologies". 21st Conference, 2014.

135. A. Milenkoski, B. Payne, N. Antunes, M. Vieira, S. Kounev, A. Avritzer, M. Luft. "Evaluation Of Intrusion Detection Systems In Virtualized Environments Using Attack Injection". Raid, p. 471–492, 2015.

136. Suricata User Guide, Performance , Statistics Available at : http://suricata.readthedocs.io/en/latest/performance/statistics.html?highlight=drop [Access 20.03.2017] , 2016.

137. Suricata User Guide, Performance , Packet Capture Available at : http://suricata.readthedocs.io/en/latest/performance/packet-capture.html?highlight=RSS [Access 20.03.2017] , 2016.

Παράρτημα Α
Σενάρια-scripts

Α) Σενάριο προσπέλασης ιστοσελίδας

```
import requests # Χρήση βιβλιοθήκης requests για αιτήματα HTTP GET, POST
import time # Χρήση βιβλιοθήκης time για την εύρεση της τρέχουσας ώρας του συστήματος
import math # Χρήση βιβλιοθήκης math για τον υπολογισμό τετραγωνικής ρίζας
from datetime import datetime
ScriptStartTime = time.time() # Έναρξη εκτέλεσης script
count = 0 # Αρχικοποίηση μετρητών
count2 = 0
TempTime = 0
AverageTimeRequest = 0
DeviationSquare = 0
AverageTimeDeviation = 0
RequestTime = []
DeviationTime = []
while (count < x) :
        startTimeRequest = time.time() # Έναρξη εκτέλεσης αιτήματος
        with open("httpaccessrequestlogfile.txt", mode='a') as output_file:
                output_file.write(str(datetime.now())+ '\n')
```

output_file.close() **# Καταγραφή ώρας εκτέλεσης αιτήματος σε αρχείο**

req = requests.request('GET', 'http://devsite.gr.intranet/wordpress')

if req.status_code == 200 : # **Έλεγχος απάντησης αιτήματος**

print "devsite.gr.intranet responded ok"

TempTime = time.time() – startTimeRequest # **Υπολογισμός χρόνου εκτέλεσης αιτήματος**

RequestTime.insert(count,TempTime) # **Αποθήκευση χρόνου εκτέλεσης αιτήματος σε λίστα**

print "Request Time is",TempTime,"Seconds" **# Εκτύπωση χρόνου εκτέλεσης αιτήματος**

AverageTimeRequest = AverageTimeRequest + TempTime # **Υπολογισμός μέσου όρου αιτήματος**

count = count + 1

TotalAverage = AverageTimeRequest/count

print "Average Time per Request",TotalAverage,"Seconds" **# Εκτύπωση μέσου όρου εκτέλεσης αιτήματος**

while (count2 < x) :

DeviationSquare = (abs(RequestTime[count2]-TotalAverage)**2) # **Υπολογισμός απόκλισης**

DeviationTime.insert(count2,DeviationSquare)

AverageTimeDeviation = AverageTimeDeviation + DeviationSquare

count2 = count2 + 1

print "Standard Deviation is",math.sqrt(AverageTimeDeviation/(count2-1)) # **Εκτύπωση απόκλισης**

print ('The script took {0} seconds!'.format(time.time() - ScriptStartTime)) # **Εκτύπωση χρόνου εκτέλεσης σεναρίου**

Β) Σενάριο σύνδεσης διαχειριστή σε ιστοσελίδα Wordpress

```
import requests # Χρήση βιβλιοθήκης requests για αιτήματα HTTP GET, POST
import time # Χρήση βιβλιοθήκης time για την εύρεση της τρέχουσας ώρας του συστήματος
import math # Χρήση βιβλιοθήκης math για τον υπολογισμό τετραγωνικής ρίζας
from datetime import datetime
ScriptStartTime = time.time() # Έναρξη εκτέλεσης script
payload = {  # Απαραίτητα στοιχεία για login στην Admin σελίδα ενός Wordpress site
        'log':'username',
        'pwd':'password',
        'rememberme':'forever',
    'redirect_to':'http://devsite.gr.intranet/wordpress/wp-admin/',
        'wp-submit':'Log In',
        'testcookie':'1'
}
count=0 # Αρχικοποίηση μετρητών
count2 = 0
TempTime = 0
AverageTimeRequest = 0
DeviationSquare = 0
AverageTimeDeviation = 0
RequestTime = []
DeviationTime = []
while (count < x) :
        startTimeRequest = time.time()# Έναρξη εκτέλεσης αιτήματος
```

```
        with open("httpadminpageloginlogfile.txt", mode='a') as output_file:
          output_file.write(str(datetime.now())+ '\n') # Καταγραφή ώρας εκτέλεσης αιτήματος σε αρχείο
        output_file.close()
        sess = requests.session()
        p=sess.post('http://devsite.gr.intranet/wordpress/wp-login.php', data=payload)
        g=sess.get('http://devsite.gr.intranet/wordpress/wp-admin/')
    if 'username' in g.content : # Έλεγχος για επιτυχημένο login
        print "Successfull Login"
    else:
        print "Login Failed"
        TempTime = time.time() – startTimeRequest # Υπολογισμός χρόνου εκτέλεσης αιτήματος
        RequestTime.insert(count,TempTime) # Αποθήκευση χρόνου εκτέλεσης αιτήματος σε λίστα
        print "Request Time is",TempTime,"Seconds" # Εκτύπωση χρόνου εκτέλεσης αιτήματος
        AverageTimeRequest = AverageTimeRequest + TempTime # Υπολογισμός μέσου όρου αιτήματος
        count = count + 1
TotalAverage = AverageTimeRequest/count
print "Average Time per Request",TotalAverage,"Seconds" # Εκτύπωση μέσου όρου εκτέλεσης αιτήματος
while (count2 < x) : # Υπολογισμός απόκλισης
        DeviationSquare = (abs(RequestTime[count2]-TotalAverage)**2)
        DeviationTime.insert(count2,DeviationSquare)
        AverageTimeDeviation = AverageTimeDeviation + DeviationSquare
```

```
        count2 = count2 + 1
print "Standard Deviation is",math.sqrt(AverageTimeDeviation/(count2-1)) # Εκτύπωση απόκλισης
print ('The script took {0} seconds!'.format(time.time() - ScriptStartTime) # Εκτύπωση χρόνου εκτέλεσης σεναρίου
```

Γ) Σενάριο μεταφόρτωσης αρχείου μέσω FTP

```
from ftplib import FTP # Χρήση βιβλιοθήκης FTP για εκτέλεση εντολών FTP
import time # Χρήση βιβλιοθήκης time για την εύρεση της τρέχουσας ώρας του συστήματος
import math # Χρήση βιβλιοθήκης math για τον υπολογισμό τετραγωνικής ρίζας
from datetime import datetime
ScriptStartTime = time.time() # Έναρξη εκτέλεσης script
count = 0 # Αρχικοποίηση μετρητών
count2 = 0
TempTime = 0
AverageTimeRequest = 0
DeviationSquare = 0
AverageTimeDeviation = 0
RequestTime = []
DeviationTime = []
while (count < x) :
        startTimeRequest = time.time() # Έναρξη εκτέλεσης αιτήματος
        with open("ftploginuploadlogfile.txt", mode='a') as output_file:
          output_file.write(str(datetime.now())+ '\n')
          output_file.close() # Καταγραφή ώρας εκτέλεσης αιτήματος σε αρχείο
        ftp = FTP('devsite.gr.intranet','ftpuser','ftpuser') # FTP Login
```

ftp.cwd('wordpress') **# Μετακίνηση στο directory wordpress**

fileName = 'file.txt' **# Έλεγχος για ύπαρξη του αρχείου file.txt**

if fileName in ftp.nlst():

ftp.delete(fileName) **# Διαγραφή αρχείου σε περίπτωση επιτυχημένης εύρεσης**

f = open(fileName, "rb")

ftp.storbinary('STOR ' + fileName, f) **# Upload αρχείου μέσω FTP**

f.close()

print "File List:"

ftp.retrlines('LIST') **# Εκτύπωση αρχείων καταλόγου**

ftp.quit()

TempTime = time.time() – startTimeRequest # **Υπολογισμός χρόνου εκτέλεσης αιτήματος**

RequestTime.insert(count,TempTime) **# Αποθήκευση χρόνου εκτέλεσης αιτήματος σε λίστα**

print "Request Time is",TempTime,"Seconds" **# Εκτύπωση χρόνου εκτέλεσης αιτήματος**

AverageTimeRequest = AverageTimeRequest + TempTime **# Υπολογισμός μέσου όρου αιτήματος**

count = count + 1

TotalAverage = AverageTimeRequest/count

print "Average Time per Request",TotalAverage,"Seconds" **# Εκτύπωση μέσου όρου εκτέλεσης αιτήματος**

while (count2 < x) : # **Υπολογισμός απόκλισης**

DeviationSquare = (abs(RequestTime[count2]-TotalAverage)**2)

DeviationTime.insert(count2,DeviationSquare)

AverageTimeDeviation = AverageTimeDeviation + DeviationSquare

```
        count2 = count2 + 1
print "Standard Deviation is",math.sqrt(AverageTimeDeviation/(count2-1)) # Εκτύπωση απόκλισης
print ('The script took {0} seconds!'.format(time.time() - ScriptStartTime)) # Εκτύπωση χρόνου εκτέλεσης σεναρίου
```

Δ) Σενάριο απομακρυσμένης σύνδεσης σε βάση MySQL και απεικόνισης των πινάκων της.

```
import MySQLdb # Χρήση βιβλιοθήκης MySQLdb για εκτέλεση εντολών MySQL
import time # Χρήση βιβλιοθήκης time για την εύρεση της τρέχουσας ώρας του συστήματος
import math # Χρήση βιβλιοθήκης math για τον υπολογισμό τετραγωνικής ρίζας
from datetime import datetime
ScriptStartTime = time.time()# Έναρξη εκτέλεσης script
count = 0 # Αρχικοποίηση μετρητών
count2 = 0
TempTime = 0
AverageTimeRequest = 0
DeviationSquare = 0
AverageTimeDeviation = 0
RequestTime = []
DeviationTime = []
while (count < x) :
    startTimeRequest = time.time()# Έναρξη εκτέλεσης αιτήματος
    with open("MySQLconnlistlogfile.txt", mode='a') as output_file:
        output_file.write(str(datetime.now())+ '\n')
        output_file.close()   # Καταγραφή ώρας εκτέλεσης αιτήματος σε αρχείο
```

```
connection = MySQLdb.connect(
        host = 'devsite.gr.intranet',
        user = 'REMOTEUSR',
        passwd = 'REMOTEUSR')  # Απομακρυσμένη σύνδεση σε βάση MySQL
cursor = connection.cursor()
cursor.execute("USE wordpress") # Επιλογή βάσης
cursor.execute("SHOW TABLES")
count = count + 1
for (table_name,) in cursor:
        print(table_name)  # Εκτύπωση πινάκων βάσης
TempTime = time.time() – startTimeRequest # Υπολογισμός χρόνου εκτέλεσης αιτήματος
RequestTime.insert(count,TempTime) # Αποθήκευση χρόνου εκτέλεσης αιτήματος σε λίστα
print "Request Time is",TempTime,"Seconds" # Εκτύπωση χρόνου εκτέλεσης αιτήματος
AverageTimeRequest = AverageTimeRequest + TempTime # Υπολογισμός μέσου όρου αιτήματος
TotalAverage = AverageTimeRequest/count
print "Average Time per Request",TotalAverage,"Seconds" # Εκτύπωση μέσου όρου εκτέλεσης αιτήματος
while (count2 < 150) : # Υπολογισμός απόκλισης
        DeviationSquare = (abs(RequestTime[count2]-TotalAverage)**2)
        DeviationTime.insert(count2,DeviationSquare)
        AverageTimeDeviation = AverageTimeDeviation + DeviationSquare
        count2 = count2 + 1
```

print "Standard Deviation is",math.sqrt(AverageTimeDeviation/(count2-1)) # **Εκτύπωση απόκλισης**

print ('The script took {0} seconds!'.format(time.time() - ScriptStartTime)) # **Εκτύπωση χρόνου εκτέλεσης σεναρίου**

Παράρτημα Β

Παραμετρικό αρχείο Suricata.yaml

```
%YAML 1.1
---

# Suricata configuration file. In addition to the comments describing all
# options in this file, full documentation can be found at:
# https://redmine.openinfosecfoundation.org/projects/suricata/wiki/Suricatayaml

##
## Step 1: inform Suricata about your network
##

vars:
  # more specifc is better for alert accuracy and performance
  address-groups:
    HOME_NET: "[192.168.2.0/24]"
    #HOME_NET: "[192.168.0.0/16]"
    #HOME_NET: "[10.0.0.0/8]"
    #HOME_NET: "[172.16.0.0/12]"
    #HOME_NET: "any"

    #EXTERNAL_NET: "!$HOME_NET"
    EXTERNAL_NET: "any"

    HTTP_SERVERS: "$HOME_NET"
    SMTP_SERVERS: "$HOME_NET"
    SQL_SERVERS: "$HOME_NET"
    DNS_SERVERS: "$HOME_NET"
    TELNET_SERVERS: "$HOME_NET"
    AIM_SERVERS: "$EXTERNAL_NET"
    DNP3_SERVER: "$HOME_NET"
    DNP3_CLIENT: "$HOME_NET"
    MODBUS_CLIENT: "$HOME_NET"
    MODBUS_SERVER: "$HOME_NET"
    ENIP_CLIENT: "$HOME_NET"
    ENIP_SERVER: "$HOME_NET"

  port-groups:
```

```
    HTTP_PORTS: "80"
    SHELLCODE_PORTS: "!80"
    ORACLE_PORTS: 1521
    SSH_PORTS: 22
    DNP3_PORTS: 20000
    MODBUS_PORTS: 502


##
## Step 2: select the rules to enable or disable
##

default-rule-path: /etc/suricata/rules
rule-files:
 - botcc.rules
 - ciarmy.rules
 - compromised.rules
 - drop.rules
 - dshield.rules
 - emerging-activex.rules
 - emerging-attack_response.rules
 - emerging-chat.rules
 - emerging-current_events.rules
 - emerging-dns.rules
 - emerging-dos.rules
 - emerging-exploit.rules
 - emerging-ftp.rules
 - emerging-games.rules
 - emerging-icmp_info.rules
 - emerging-icmp.rules
 - emerging-imap.rules
 - emerging-inappropriate.rules
 - emerging-malware.rules
 - emerging-misc.rules
 - emerging-mobile_malware.rules
 - emerging-netbios.rules
 - emerging-p2p.rules
 - emerging-policy.rules
 - emerging-pop3.rules
 - emerging-rpc.rules
 - emerging-scada.rules
 - emerging-scan.rules
 - emerging-shellcode.rules
 - emerging-smtp.rules
 - emerging-snmp.rules
 - emerging-sql.rules
 - emerging-telnet.rules
 - emerging-tftp.rules
 - emerging-trojan.rules
```

```
 - emerging-user_agents.rules
 - emerging-voip.rules
 - emerging-web_client.rules
 - emerging-web_server.rules
 - emerging-web_specific_apps.rules
 - emerging-worm.rules
 - tor.rules
 - decoder-events.rules # available in suricata sources under rules dir
 - stream-events.rules  # available in suricata sources under rules dir
 - http-events.rules    # available in suricata sources under rules dir
 - smtp-events.rules    # available in suricata sources under rules dir
 - dns-events.rules     # available in suricata sources under rules dir
 - tls-events.rules     # available in suricata sources under rules dir
# - modbus-events.rules  # available in suricata sources under rules dir
 - app-layer-events.rules  # available in suricata sources under rules dir

classification-file: /etc/suricata/classification.config
reference-config-file: /etc/suricata/reference.config
# threshold-file: /etc/suricata/threshold.config


##
## Step 3: select outputs to enable
##

# The default logging directory.  Any log or output file will be
# placed here if its not specified with a full path name. This can be
# overridden with the -l command line parameter.
default-log-dir: /var/log/suricata/

# global stats configuration
stats:
  enabled: yes
  # The interval field (in seconds) controls at what interval
  # the loggers are invoked.
  interval: 8

# Configure the type of alert (and other) logging you would like.
outputs:
  # a line based alerts log similar to Snort's fast.log
  - fast:
      enabled: yes
      filename: fast.log
      append: yes
      #filetype: regular # 'regular', 'unix_stream' or 'unix_dgram'

  # Extensible Event Format (nicknamed EVE) event log in JSON format
  - eve-log:
      enabled: yes
```

```
filetype: regular #regular|syslog|unix_dgram|unix_stream|redis
filename: eve.json
#prefix: "@cee: " # prefix to prepend to each log entry
# the following are valid when type: syslog above
#identity: "suricata"
#facility: local5
#level: Info ## possible levels: Emergency, Alert, Critical,
          ## Error, Warning, Notice, Info, Debug
#redis:
#  server: 127.0.0.1
#  port: 6379
#  mode: list ## possible values: list (default), channel
#  key: suricata ## key or channel to use (default to suricata)
# Redis pipelining set up. This will enable to only do a query every
# 'batch-size' events. This should lower the latency induced by network
# connection at the cost of some memory. There is no flushing implemented
# so this setting as to be reserved to high traffic suricata.
#  pipelining:
#    enabled: yes ## set enable to yes to enable query pipelining
#    batch-size: 10 ## number of entry to keep in buffer
types:
  - alert:
      # payload: yes             # enable dumping payload in Base64
      # payload-buffer-size: 4kb # max size of payload buffer to output in eve-log
      # payload-printable: yes   # enable dumping payload in printable (lossy) format
      # packet: yes              # enable dumping of packet (without stream segments)
      http: yes               # enable dumping of http fields
      tls: yes                # enable dumping of tls fields
      ssh: yes                # enable dumping of ssh fields
      smtp: yes               # enable dumping of smtp fields

      # HTTP X-Forwarded-For support by adding an extra field or overwriting
      # the source or destination IP address (depending on flow direction)
      # with the one reported in the X-Forwarded-For HTTP header. This is
      # helpful when reviewing alerts for traffic that is being reverse
      # or forward proxied.
      xff:
        enabled: no
        # Two operation modes are available, "extra-data" and "overwrite".
        mode: extra-data
        # Two proxy deployments are supported, "reverse" and "forward". In
        # a "reverse" deployment the IP address used is the last one, in a
        # "forward" deployment the first IP address is used.
        deployment: reverse
        # Header name where the actual IP address will be reported, if more
        # than one IP address is present, the last IP address will be the
        # one taken into consideration.
        header: X-Forwarded-For
  - http:
```

```
        extended: yes     # enable this for extended logging information
        # custom allows additional http fields to be included in eve-log
        # the example below adds three additional fields when uncommented
        #custom: [Accept-Encoding, Accept-Language, Authorization]
      - dns
      - tls:
        extended: yes     # enable this for extended logging information
      - files:
        force-magic: no   # force logging magic on all logged files
        force-md5: no     # force logging of md5 checksums
      #- drop:
      #    alerts: no      # log alerts that caused drops
      - smtp:
        #extended: yes # enable this for extended logging information
        # this includes: bcc, message-id, subject, x_mailer, user-agent
        # custom fields logging from the list:
        #  reply-to, bcc, message-id, subject, x-mailer, user-agent, received,
        #  x-originating-ip, in-reply-to, references, importance, priority,
        #  sensitivity, organization, content-md5, date
        #custom: [received, x-mailer, x-originating-ip, relays, reply-to, bcc]
        # output md5 of fields: body, subject
        # for the body you need to set app-layer.protocols.smtp.mime.body-md5
        # to yes
        #md5: [body, subject]

      - ssh
      - stats:
        totals: yes       # stats for all threads merged together
        threads: no       # per thread stats
        deltas: no        # include delta values
      # bi-directional flows
      - flow
      # uni-directional flows
      #- netflow

  # alert output for use with Barnyard2
  - unified2-alert:
      enabled: no
      filename: unified2.alert

      # File size limit.  Can be specified in kb, mb, gb.  Just a number
      # is parsed as bytes.
      #limit: 32mb

      # Sensor ID field of unified2 alerts.
      #sensor-id: 0

      # Include payload of packets related to alerts. Defaults to true, set to
      # false if payload is not required.
```

```
    #payload: yes

    # HTTP X-Forwarded-For support by adding the unified2 extra header or
    # overwriting the source or destination IP address (depending on flow
    # direction) with the one reported in the X-Forwarded-For HTTP header.
    # This is helpful when reviewing alerts for traffic that is being reverse
    # or forward proxied.
    xff:
      enabled: no
      # Two operation modes are available, "extra-data" and "overwrite". Note
      # that in the "overwrite" mode, if the reported IP address in the HTTP
      # X-Forwarded-For header is of a different version of the packet
      # received, it will fall-back to "extra-data" mode.
      mode: extra-data
      # Two proxy deployments are supported, "reverse" and "forward". In
      # a "reverse" deployment the IP address used is the last one, in a
      # "forward" deployment the first IP address is used.
      deployment: reverse
      # Header name where the actual IP address will be reported, if more
      # than one IP address is present, the last IP address will be the
      # one taken into consideration.
      header: X-Forwarded-For

  # a line based log of HTTP requests (no alerts)
  - http-log:
      enabled: yes
      filename: http.log
      append: yes
      #extended: yes     # enable this for extended logging information
      #custom: yes       # enabled the custom logging format (defined by customformat)
      #customformat: "%{%D-%H:%M:%S}t.%z %{X-Forwarded-For}i %H %m %h %u
%s %B %a:%p -> %A:%P"
      #filetype: regular # 'regular', 'unix_stream' or 'unix_dgram'

  # a line based log of TLS handshake parameters (no alerts)
  - tls-log:
      enabled: yes  # Log TLS connections.
      filename: tls.log # File to store TLS logs.
      append: yes
      #filetype: regular # 'regular', 'unix_stream' or 'unix_dgram'
      #extended: yes # Log extended information like fingerprint

  # output module to store certificates chain to disk
  - tls-store:
      enabled: no
      #certs-log-dir: certs # directory to store the certificates files

  # a line based log of DNS requests and/or replies (no alerts)
  - dns-log:
```

```
    enabled: yes
    filename: dns.log
    append: yes
    #filetype: regular # 'regular', 'unix_stream' or 'unix_dgram'

  # Packet log... log packets in pcap format. 3 modes of operation: "normal"
  # "multi" and "sguil".
  #
  # In normal mode a pcap file "filename" is created in the default-log-dir,
  # or are as specified by "dir".
  # In multi mode, a file is created per thread. This will perform much
  # better, but will create multiple files where 'normal' would create one.
  # In multi mode the filename takes a few special variables:
  # - %n -- thread number
  # - %i -- thread id
  # - %t -- timestamp (secs or secs.usecs based on 'ts-format'
  # E.g. filename: pcap.%n.%t
  #
  # Note that it's possible to use directories, but the directories are not
  # created by Suricata. E.g. filename: pcaps/%n/log.%s will log into the
  # per thread directory.
  #
  # Also note that the limit and max-files settings are enforced per thread.
  # So the size limit when using 8 threads with 1000mb files and 2000 files
  # is: 8*1000*2000 ~ 16TiB.
  #
  # In Sguil mode "dir" indicates the base directory. In this base dir the
  # pcaps are created in th directory structure Sguil expects:
  #
  # $sguil-base-dir/YYYY-MM-DD/$filename.<timestamp>
  #
  # By default all packets are logged except:
  # - TCP streams beyond stream.reassembly.depth
  # - encrypted streams after the key exchange
  #
  - pcap-log:
      enabled: no
      filename: log.pcap

      # File size limit.  Can be specified in kb, mb, gb.  Just a number
      # is parsed as bytes.
      limit: 5000mb

      # If set to a value will enable ring buffer mode. Will keep Maximum of "max-files" of
size "limit"
      max-files: 2000

      mode: normal # normal, multi or sguil.
      #sguil-base-dir: /nsm_data/
```

```
    #ts-format: usec # sec or usec second format (default) is filename.sec usec is
filename.sec.usec
    use-stream-depth: no #If set to "yes" packets seen after reaching stream inspection
depth are ignored. "no" logs all packets
    honor-pass-rules: no # If set to "yes", flows in which a pass rule matched will
stopped being logged.

  # a full alerts log containing much information for signature writers
  # or for investigating suspected false positives.
  - alert-debug:
      enabled: no
      filename: alert-debug.log
      append: yes
      #filetype: regular # 'regular', 'unix_stream' or 'unix_dgram'

  # alert output to prelude (http://www.prelude-technologies.com/) only
  # available if Suricata has been compiled with --enable-prelude
  - alert-prelude:
      enabled: no
      profile: suricata
      log-packet-content: no
      log-packet-header: yes

  # Stats.log contains data from various counters of the suricata engine.
  - stats:
      enabled: yes
      filename: stats.log
      totals: yes       # stats for all threads merged together
      threads: no       # per thread stats
      #null-values: yes  # print counters that have value 0

  # a line based alerts log similar to fast.log into syslog
  - syslog:
      enabled: no
      # reported identity to syslog. If ommited the program name (usually
      # suricata) will be used.
      #identity: "suricata"
      facility: local5
      #level: Info ## possible levels: Emergency, Alert, Critical,
                  ## Error, Warning, Notice, Info, Debug

  # a line based information for dropped packets in IPS mode
  - drop:
      enabled: yes
      filename: drop.log
      append: yes
      #filetype: regular # 'regular', 'unix_stream' or 'unix_dgram'

  # output module to store extracted files to disk
```

```
#
# The files are stored to the log-dir in a format "file.<id>" where <id> is
# an incrementing number starting at 1. For each file "file.<id>" a meta
# file "file.<id>.meta" is created.
#
# File extraction depends on a lot of things to be fully done:
# - stream reassembly depth. For optimal results, set this to 0 (unlimited)
# - http request / response body sizes. Again set to 0 for optimal results.
# - rules that contain the "filestore" keyword.
- file-store:
    enabled: no       # set to yes to enable
    log-dir: files    # directory to store the files
    force-magic: no   # force logging magic on all stored files
    force-md5: no     # force logging of md5 checksums
    force-filestore: no # force storing of all files
    #waldo: file.waldo # waldo file to store the file_id across runs

# output module to log files tracked in a easily parsable json format
- file-log:
    enabled: no
    filename: files-json.log
    append: yes
    #filetype: regular # 'regular', 'unix_stream' or 'unix_dgram'

    force-magic: no   # force logging magic on all logged files
    force-md5: no     # force logging of md5 checksums

# Log TCP data after stream normalization
# 2 types: file or dir. File logs into a single logfile. Dir creates
# 2 files per TCP session and stores the raw TCP data into them.
# Using 'both' will enable both file and dir modes.
#
# Note: limited by stream.depth
- tcp-data:
    enabled: no
    type: file
    filename: tcp-data.log

# Log HTTP body data after normalization, dechunking and unzipping.
# 2 types: file or dir. File logs into a single logfile. Dir creates
# 2 files per HTTP session and stores the normalized data into them.
# Using 'both' will enable both file and dir modes.
#
# Note: limited by the body limit settings
- http-body-data:
    enabled: no
    type: file
    filename: http-data.log
```

```
  # Lua Output Support - execute lua script to generate alert and event
  # output.
  # Documented at:
  # https://redmine.openinfosecfoundation.org/projects/suricata/wiki/Lua_Output
  - lua:
      enabled: no
      #scripts-dir: /etc/suricata/lua-output/
      scripts:
      #   - script1.lua

# Logging configuration.  This is not about logging IDS alerts/events, but
# output about what Suricata is doing, like startup messages, errors, etc.
logging:
  # The default log level, can be overridden in an output section.
  # Note that debug level logging will only be emitted if Suricata was
  # compiled with the --enable-debug configure option.
  #
  # This value is overriden by the SC_LOG_LEVEL env var.
  default-log-level: notice

  # The default output format.  Optional parameter, should default to
  # something reasonable if not provided.  Can be overriden in an
  # output section.  You can leave this out to get the default.
  #
  # This value is overriden by the SC_LOG_FORMAT env var.
  #default-log-format: "[%i] %t - (%f:%l) <%d> (%n) -- "

  # A regex to filter output.  Can be overridden in an output section.
  # Defaults to empty (no filter).
  #
  # This value is overriden by the SC_LOG_OP_FILTER env var.
  default-output-filter:

  # Define your logging outputs.  If none are defined, or they are all
  # disabled you will get the default - console output.
  outputs:
  - console:
      enabled: yes
      # type: json
  - file:
      enabled: yes
      level: info
      filename: /var/log/suricata/suricata.log
      # type: json
  - syslog:
      enabled: no
      facility: local5
      format: "[%i] <%d> -- "
      # type: json
```

```
##
## Step 4: configure common capture settings
##
## See "Advanced Capture Options" below for more options, including NETMAP
## and PF_RING.
##

# Linux high speed capture support
af-packet:
  - interface: eth0
    # Number of receive threads. "auto" uses the number of cores
    #threads: auto
    # Default clusterid. AF_PACKET will load balance packets based on flow.
    cluster-id: 99
    # Default AF_PACKET cluster type. AF_PACKET can load balance per flow or per hash.
    # This is only supported for Linux kernel > 3.1
    # possible value are:
    #  * cluster_round_robin: round robin load balancing
    #  * cluster_flow: all packets of a given flow are send to the same socket
    #  * cluster_cpu: all packets treated in kernel by a CPU are send to the same socket
    #  * cluster_qm: all packets linked by network card to a RSS queue are sent to the
same
    #  socket. Requires at least Linux 3.14.
    #  * cluster_random: packets are sent randomly to sockets but with an equipartition.
    #  Requires at least Linux 3.14.
    #  * cluster_rollover: kernel rotates between sockets filling each socket before moving
    #  to the next. Requires at least Linux 3.10.
    # Recommended modes are cluster_flow on most boxes and cluster_cpu or cluster_qm
on system
    # with capture card using RSS (require cpu affinity tuning and system irq tuning)
    cluster-type: cluster_flow
    # In some fragmentation case, the hash can not be computed. If "defrag" is set
    # to yes, the kernel will do the needed defragmentation before sending the packets.
    defrag: yes
    # After Linux kernel 3.10 it is possible to activate the rollover option: if a socket is
    # full then kernel will send the packet on the next socket with room available. This
option
    # can minimize packet drop and increase the treated bandwidth on single intensive
flow.
    #rollover: yes
    # To use the ring feature of AF_PACKET, set 'use-mmap' to yes
    #use-mmap: yes
    # Lock memory map to avoid it goes to swap. Be careful that over suscribing could
lock
    # your system
    #mmap-locked: yes
    # Use experimental tpacket_v3 capture mode, only active if use-mmap is true
```

```
    #tpacket-v3: yes
    # Ring size will be computed with respect to max_pending_packets and number
    # of threads. You can set manually the ring size in number of packets by setting
    # the following value. If you are using flow cluster-type and have really network
    # intensive single-flow you could want to set the ring-size independently of the
number
    # of threads:
    #ring-size: 2048
    # Block size is used by tpacket_v3 only. It should set to a value high enough to contain
    # a decent number of packets. Size is in bytes so please consider your MTU. It should
be
    # a power of 2 and it must be multiple of page size (usually 4096).
    #block-size: 32768
    # tpacket_v3 block timeout: an open block is passed to userspace if it is not
    # filled after block-timeout milliseconds.
    #block-timeout: 10
    # On busy system, this could help to set it to yes to recover from a packet drop
    # phase. This will result in some packets (at max a ring flush) being non treated.
    #use-emergency-flush: yes
    # recv buffer size, increase value could improve performance
    # buffer-size: 32768
    # Set to yes to disable promiscuous mode
    # disable-promisc: no
    # Choose checksum verification mode for the interface. At the moment
    # of the capture, some packets may be with an invalid checksum due to
    # offloading to the network card of the checksum computation.
    # Possible values are:
    #  - kernel: use indication sent by kernel for each packet (default)
    #  - yes: checksum validation is forced
    #  - no: checksum validation is disabled
    #  - auto: suricata uses a statistical approach to detect when
    #  checksum off-loading is used.
    # Warning: 'checksum-validation' must be set to yes to have any validation
    #checksum-checks: kernel
    # BPF filter to apply to this interface. The pcap filter syntax apply here.
    #bpf-filter: port 80 or udp
    # You can use the following variables to activate AF_PACKET tap or IPS mode.
    # If copy-mode is set to ips or tap, the traffic coming to the current
    # interface will be copied to the copy-iface interface. If 'tap' is set, the
    # copy is complete. If 'ips' is set, the packet matching a 'drop' action
    # will not be copied.
    #copy-mode: ips
    #copy-iface: eth1

  # Put default values here. These will be used for an interface that is not
  # in the list above.
  - interface: default
    #threads: auto
    #use-mmap: no
```

```
    #rollover: yes
    #tpacket-v3: yes

# Cross platform libpcap capture support
pcap:
  - interface: eth0
    # On Linux, pcap will try to use mmaped capture and will use buffer-size
    # as total of memory used by the ring. So set this to something bigger
    # than 1% of your bandwidth.
    #buffer-size: 16777216
    #bpf-filter: "tcp and port 25"
    # Choose checksum verification mode for the interface. At the moment
    # of the capture, some packets may be with an invalid checksum due to
    # offloading to the network card of the checksum computation.
    # Possible values are:
    #  - yes: checksum validation is forced
    #  - no: checksum validation is disabled
    #  - auto: suricata uses a statistical approach to detect when
    #  checksum off-loading is used. (default)
    # Warning: 'checksum-validation' must be set to yes to have any validation
    #checksum-checks: auto
    # With some accelerator cards using a modified libpcap (like myricom), you
    # may want to have the same number of capture threads as the number of capture
    # rings. In this case, set up the threads variable to N to start N threads
    # listening on the same interface.
    #threads: 16
    # set to no to disable promiscuous mode:
    #promisc: no
    # set snaplen, if not set it defaults to MTU if MTU can be known
    # via ioctl call and to full capture if not.
    #snaplen: 1518
  # Put default values here
  - interface: default
    #checksum-checks: auto

# Settings for reading pcap files
pcap-file:
  # Possible values are:
  #  - yes: checksum validation is forced
  #  - no: checksum validation is disabled
  #  - auto: suricata uses a statistical approach to detect when
  #  checksum off-loading is used. (default)
  # Warning: 'checksum-validation' must be set to yes to have checksum tested
  checksum-checks: auto

# See "Advanced Capture Options" below for more options, including NETMAP
# and PF_RING.
```

```
##
## Step 5: App Layer Protocol Configuration
##

# Configure the app-layer parsers. The protocols section details each
# protocol.
#
# The option "enabled" takes 3 values - "yes", "no", "detection-only".
# "yes" enables both detection and the parser, "no" disables both, and
# "detection-only" enables protocol detection only (parser disabled).
app-layer:
  protocols:
    tls:
      enabled: yes
      detection-ports:
        dp: 443

      #no-reassemble: yes
    dcerpc:
      enabled: yes
    ftp:
      enabled: yes
    ssh:
      enabled: yes
    smtp:
      enabled: yes
      # Configure SMTP-MIME Decoder
      mime:
        # Decode MIME messages from SMTP transactions
        # (may be resource intensive)
        # This field supercedes all others because it turns the entire
        # process on or off
        decode-mime: yes

        # Decode MIME entity bodies (ie. base64, quoted-printable, etc.)
        decode-base64: yes
        decode-quoted-printable: yes

        # Maximum bytes per header data value stored in the data structure
        # (default is 2000)
        header-value-depth: 2000

        # Extract URLs and save in state data structure
        extract-urls: yes
        # Set to yes to compute the md5 of the mail body. You will then
        # be able to journalize it.
        body-md5: no
      # Configure inspected-tracker for file_data keyword
      inspected-tracker:
```

```
    content-limit: 100000
    content-inspect-min-size: 32768
    content-inspect-window: 4096
  imap:
   enabled: detection-only
  msn:
   enabled: detection-only
  smb:
   enabled: yes
   detection-ports:
    dp: 139
  # Note: Modbus probe parser is minimalist due to the poor significant field
  # Only Modbus message length (greater than Modbus header length)
  # And Protocol ID (equal to 0) are checked in probing parser
  # It is important to enable detection port and define Modbus port
  # to avoid false positive
  modbus:
   # How many unreplied Modbus requests are considered a flood.
   # If the limit is reached, app-layer-event:modbus.flooded; will match.
   #request-flood: 500

   enabled: no
   detection-ports:
    dp: 502
   # According to MODBUS Messaging on TCP/IP Implementation Guide V1.0b, it
   # is recommended to keep the TCP connection opened with a remote device
   # and not to open and close it for each MODBUS/TCP transaction. In that
   # case, it is important to set the depth of the stream reassembling as
   # unlimited (stream.reassembly.depth: 0)
  # smb2 detection is disabled internally inside the engine.
  #smb2:
  #  enabled: yes
  dns:
   # memcaps. Globally and per flow/state.
   #global-memcap: 16mb
   #state-memcap: 512kb

   # How many unreplied DNS requests are considered a flood.
   # If the limit is reached, app-layer-event:dns.flooded; will match.
   #request-flood: 500

   tcp:
    enabled: yes
    detection-ports:
     dp: 53
   udp:
    enabled: yes
    detection-ports:
     dp: 53
```

```
http:
  enabled: yes
  # memcap: 64mb

  # default-config:           Used when no server-config matches
  #   personality:            List of personalities used by default
  #   request-body-limit:     Limit reassembly of request body for inspection
  #                           by http_client_body & pcre /P option.
  #   response-body-limit:    Limit reassembly of response body for inspection
  #                           by file_data, http_server_body & pcre /Q option.
  #   double-decode-path:     Double decode path section of the URI
  #   double-decode-query:    Double decode query section of the URI
  #   response-body-decompress-layer-limit:
  #                           Limit to how many layers of compression will be
  #                           decompressed. Defaults to 2.
  #
  # server-config:            List of server configurations to use if address matches
  #   address:                List of ip addresses or networks for this block
  #   personalitiy:           List of personalities used by this block
  #   request-body-limit:     Limit reassembly of request body for inspection
  #                           by http_client_body & pcre /P option.
  #   response-body-limit:    Limit reassembly of response body for inspection
  #                           by file_data, http_server_body & pcre /Q option.
  #   double-decode-path:     Double decode path section of the URI
  #   double-decode-query:    Double decode query section of the URI
  #
  #   uri-include-all:        Include all parts of the URI. By default the
  #                           'scheme', username/password, hostname and port
  #                           are excluded. Setting this option to true adds
  #                           all of them to the normalized uri as inspected
  #                           by http_uri, urilen, pcre with /U and the other
  #                           keywords that inspect the normalized uri.
  #                           Note that this does not affect http_raw_uri.
  #                           Also, note that including all was the default in
  #                           1.4 and 2.0beta1.
  #
  #   meta-field-limit:       Hard size limit for request and response size
  #                           limits. Applies to request line and headers,
  #                           response line and headers. Does not apply to
  #                           request or response bodies. Default is 18k.
  #                           If this limit is reached an event is raised.
  #
  # Currently Available Personalities:
  #   Minimal, Generic, IDS (default), IIS_4_0, IIS_5_0, IIS_5_1, IIS_6_0,
  #   IIS_7_0, IIS_7_5, Apache_2
  libhtp:
    default-config:
      personality: IDS
```

```
  # Can be specified in kb, mb, gb.  Just a number indicates
  # it's in bytes.
  request-body-limit: 100kb
  response-body-limit: 100kb

  # inspection limits
  request-body-minimal-inspect-size: 32kb
  request-body-inspect-window: 4kb
  response-body-minimal-inspect-size: 40kb
  response-body-inspect-window: 16kb

  # response body decompression (0 disables)
  response-body-decompress-layer-limit: 2

  # auto will use http-body-inline mode in IPS mode, yes or no set it statically
  http-body-inline: auto

  # Take a random value for inspection sizes around the specified value.
  # This lower the risk of some evasion technics but could lead
  # detection change between runs. It is set to 'yes' by default.
  #randomize-inspection-sizes: yes
  # If randomize-inspection-sizes is active, the value of various
  # inspection size will be choosen in the [1 - range%, 1 + range%]
  # range
  # Default value of randomize-inspection-range is 10.
  #randomize-inspection-range: 10

  # decoding
  double-decode-path: no
  double-decode-query: no

server-config:

  #- apache:
  #    address: [192.168.1.0/24, 127.0.0.0/8, "::1"]
  #    personality: Apache_2
  #    # Can be specified in kb, mb, gb.  Just a number indicates
  #    # it's in bytes.
  #    request-body-limit: 4096
  #    response-body-limit: 4096
  #    double-decode-path: no
  #    double-decode-query: no

  #- iis7:
  #    address:
  #      - 192.168.0.0/24
  #      - 192.168.10.0/24
  #    personality: IIS_7_0
  #    # Can be specified in kb, mb, gb.  Just a number indicates
```

```
    #   # it's in bytes.
    #   request-body-limit: 4096
    #   response-body-limit: 4096
    #   double-decode-path: no
    #   double-decode-query: no

# Limit for the maximum number of asn1 frames to decode (default 256)
asn1-max-frames: 256


##########################################################################
##################
##
## Advanced settings below
##
##########################################################################
##################

##
## Run Options
##

# Run suricata as user and group.
#run-as:
#  user: suri
#  group: suri

# Some logging module will use that name in event as identifier. The default
# value is the hostname
#sensor-name: suricata

# Default pid file.
# Will use this file if no --pidfile in command options.
#pid-file: /var/run/suricata.pid

# Daemon working directory
# Suricata will change directory to this one if provided
# Default: "/"
#daemon-directory: "/"

# Suricata core dump configuration. Limits the size of the core dump file to
# approximately max-dump. The actual core dump size will be a multiple of the
# page size. Core dumps that would be larger than max-dump are truncated. On
# Linux, the actual core dump size may be a few pages larger than max-dump.
# Setting max-dump to 0 disables core dumping.
# Setting max-dump to 'unlimited' will give the full core dump file.
# On 32-bit Linux, a max-dump value >= ULONG_MAX may cause the core dump size
# to be 'unlimited'.
```

```
coredump:
  max-dump: unlimited

# If suricata box is a router for the sniffed networks, set it to 'router'. If
# it is a pure sniffing setup, set it to 'sniffer-only'.
# If set to auto, the variable is internally switch to 'router' in IPS mode
# and 'sniffer-only' in IDS mode.
# This feature is currently only used by the reject* keywords.
host-mode: auto

# Number of packets preallocated per thread. The default is 1024. A higher number
# will make sure each CPU will be more easily kept busy, but may negatively
# impact caching.
#
# If you are using the CUDA pattern matcher (mpm-algo: ac-cuda), different rules
# apply. In that case try something like 60000 or more. This is because the CUDA
# pattern matcher buffers and scans as many packets as possible in parallel.
#max-pending-packets: 1024

# Runmode the engine should use. Please check --list-runmodes to get the available
# runmodes for each packet acquisition method. Defaults to "autofp" (auto flow pinned
# load balancing).
#runmode: autofp

# Specifies the kind of flow load balancer used by the flow pinned autofp mode.
#
# Supported schedulers are:
#
# round-robin       - Flows assigned to threads in a round robin fashion.
# active-packets    - Flows assigned to threads that have the lowest number of
#                     unprocessed packets (default).
# hash              - Flow alloted usihng the address hash. More of a random
#                     technique. Was the default in Suricata 1.2.1 and older.
#
#autofp-scheduler: active-packets

# Preallocated size for packet. Default is 1514 which is the classical
# size for pcap on ethernet. You should adjust this value to the highest
# packet size (MTU + hardware header) on your system.
#default-packet-size: 1514

# Unix command socket can be used to pass commands to suricata.
# An external tool can then connect to get information from suricata
# or trigger some modifications of the engine. Set enabled to yes
# to activate the feature. You can use the filename variable to set
# the file name of the socket.
unix-command:
  enabled: yes
  filename: /var/run/suricata-command.socket
```

```
# Magic file. The extension .mgc is added to the value here.
#magic-file: /usr/share/file/magic
#magic-file:

legacy:
  uricontent: enabled

##
## Detection settings
##

# Set the order of alerts bassed on actions
# The default order is pass, drop, reject, alert
# action-order:
#   - pass
#   - drop
#   - reject
#   - alert

# IP Reputation
#reputation-categories-file: /etc/suricata/iprep/categories.txt
#default-reputation-path: /etc/suricata/iprep
#reputation-files:
# - reputation.list

# When run with the option --engine-analysis, the engine will read each of
# the parameters below, and print reports for each of the enabled sections
# and exit.  The reports are printed to a file in the default log dir
# given by the parameter "default-log-dir", with engine reporting
# subsection below printing reports in its own report file.
engine-analysis:
  # enables printing reports for fast-pattern for every rule.
  rules-fast-pattern: yes
  # enables printing reports for each rule
  rules: yes

#recursion and match limits for PCRE where supported
pcre:
  match-limit: 3500
  match-limit-recursion: 1500

##
## Advanced Traffic Tracking and Reconstruction Settings
##

# Host specific policies for defragmentation and TCP stream
# reassembly. The host OS lookup is done using a radix tree, just
# like a routing table so the most specific entry matches.
```

```
host-os-policy:
  # Make the default policy windows.
  windows: [192.168.2.20,192.168.2.3]
  bsd: []
  bsd-right: []
  old-linux: []
  linux: [192.168.2.48,192.168.2.203]
  old-solaris: []
  solaris: []
  hpux10: []
  hpux11: []
  irix: []
  macos: []
  vista: []
  windows2k3: []

# Defrag settings:

defrag:
  memcap: 32mb
  hash-size: 65536
  trackers: 65535 # number of defragmented flows to follow
  max-frags: 65535 # number of fragments to keep (higher than trackers)
  prealloc: yes
  timeout: 60

# Enable defrag per host settings
#  host-config:
#
#    - dmz:
#        timeout: 30
#        address: [192.168.1.0/24, 127.0.0.0/8, 1.1.1.0/24, 2.2.2.0/24, "1.1.1.1", "2.2.2.2",
"::1"]
#
#    - lan:
#        timeout: 45
#        address:
#          - 192.168.0.0/24
#          - 192.168.10.0/24
#          - 172.16.14.0/24

# Flow settings:
# By default, the reserved memory (memcap) for flows is 32MB. This is the limit
# for flow allocation inside the engine. You can change this value to allow
# more memory usage for flows.
# The hash-size determine the size of the hash used to identify flows inside
# the engine, and by default the value is 65536.
# At the startup, the engine can preallocate a number of flows, to get a better
# performance. The number of flows preallocated is 10000 by default.
```

```
# emergency-recovery is the percentage of flows that the engine need to
# prune before unsetting the emergency state. The emergency state is activated
# when the memcap limit is reached, allowing to create new flows, but
# prunning them with the emergency timeouts (they are defined below).
# If the memcap is reached, the engine will try to prune flows
# with the default timeouts. If it doens't find a flow to prune, it will set
# the emergency bit and it will try again with more agressive timeouts.
# If that doesn't work, then it will try to kill the last time seen flows
# not in use.
# The memcap can be specified in kb, mb, gb.  Just a number indicates it's
# in bytes.

flow:
  memcap: 128mb
  hash-size: 65536
  prealloc: 10000
  emergency-recovery: 30
  #managers: 1 # default to one flow manager
  #recyclers: 1 # default to one flow recycler thread

# This option controls the use of vlan ids in the flow (and defrag)
# hashing. Normally this should be enabled, but in some (broken)
# setups where both sides of a flow are not tagged with the same vlan
# tag, we can ignore the vlan id's in the flow hashing.
vlan:
  use-for-tracking: true

# Specific timeouts for flows. Here you can specify the timeouts that the
# active flows will wait to transit from the current state to another, on each
# protocol. The value of "new" determine the seconds to wait after a hanshake or
# stream startup before the engine free the data of that flow it doesn't
# change the state to established (usually if we don't receive more packets
# of that flow). The value of "established" is the amount of
# seconds that the engine will wait to free the flow if it spend that amount
# without receiving new packets or closing the connection. "closed" is the
# amount of time to wait after a flow is closed (usually zero).
#
# There's an emergency mode that will become active under attack circumstances,
# making the engine to check flow status faster. This configuration variables
# use the prefix "emergency-" and work similar as the normal ones.
# Some timeouts doesn't apply to all the protocols, like "closed", for udp and
# icmp.

flow-timeouts:

  default:
    new: 30
    established: 300
    closed: 0
```

```
    emergency-new: 10
    emergency-established: 100
    emergency-closed: 0
  tcp:
    new: 60
    established: 600
    closed: 60
    emergency-new: 5
    emergency-established: 100
    emergency-closed: 10
  udp:
    new: 30
    established: 300
    emergency-new: 10
    emergency-established: 100
  icmp:
    new: 30
    established: 300
    emergency-new: 10
    emergency-established: 100

# Stream engine settings. Here the TCP stream tracking and reassembly
# engine is configured.
#
# stream:
#   memcap: 32mb              # Can be specified in kb, mb, gb.  Just a
#                             # number indicates it's in bytes.
#   checksum-validation: yes    # To validate the checksum of received
#                             # packet. If csum validation is specified as
#                             # "yes", then packet with invalid csum will not
#                             # be processed by the engine stream/app layer.
#                             # Warning: locally generated trafic can be
#                             # generated without checksum due to hardware offload
#                             # of checksum. You can control the handling of checksum
#                             # on a per-interface basis via the 'checksum-checks'
#                             # option
#   prealloc-sessions: 2k       # 2k sessions prealloc'd per stream thread
#   midstream: false            # don't allow midstream session pickups
#   async-oneside: false        # don't enable async stream handling
#   inline: no                  # stream inline mode
#    max-synack-queued:  5      # Max different SYN/ACKs to queue
#
#   reassembly:
#     memcap: 64mb              # Can be specified in kb, mb, gb.  Just a number
#                             # indicates it's in bytes.
#     depth: 1mb                # Can be specified in kb, mb, gb.  Just a number
#                             # indicates it's in bytes.
#     toserver-chunk-size: 2560 # inspect raw stream in chunks of at least
#                             # this size.  Can be specified in kb, mb,
```

```
#                   # gb.  Just a number indicates it's in bytes.
#                   # The max acceptable size is 4024 bytes.
#    toclient-chunk-size: 2560 # inspect raw stream in chunks of at least
#                   # this size.  Can be specified in kb, mb,
#                   # gb.  Just a number indicates it's in bytes.
#                   # The max acceptable size is 4024 bytes.
#    randomize-chunk-size: yes # Take a random value for chunk size around the
specified value.
#                   # This lower the risk of some evasion technics but could lead
#                   # detection change between runs. It is set to 'yes' by default.
#    randomize-chunk-range: 10 # If randomize-chunk-size is active, the value of chunk-
size is
#                   # a random value between (1 - randomize-chunk-
range/100)*toserver-chunk-size
#                   # and (1 + randomize-chunk-range/100)*toserver-chunk-size and the
same
#                   # calculation for toclient-chunk-size.
#                   # Default value of randomize-chunk-range is 10.
#
#    raw: yes              # 'Raw' reassembly enabled or disabled.
#                   # raw is for content inspection by detection
#                   # engine.
#
#    chunk-prealloc: 250      # Number of preallocated stream chunks. These
#                   # are used during stream inspection (raw).
#    segments:             # Settings for reassembly segment pool.
#      - size: 4           # Size of the (data)segment for a pool
#        prealloc: 256       # Number of segments to prealloc and keep
#                   # in the pool.
#    zero-copy-size: 128      # This option sets in bytes the value at
#                   # which segment data is passed to the app
#                   # layer API directly. Data sizes equal to
#                   # and higher than the value set are passed
#                   # on directly.
#
stream:
  memcap: 64mb
  checksum-validation: yes     # reject wrong csums
  inline: auto              # auto will use inline mode in IPS mode, yes or no set it statically
  reassembly:
    memcap: 256mb
    depth: 1mb              # reassemble 1mb into a stream
    toserver-chunk-size: 2560
    toclient-chunk-size: 2560
    randomize-chunk-size: yes
    #randomize-chunk-range: 10
    #raw: yes
    #chunk-prealloc: 250
    #segments:
```

```
   #  - size: 4
   #    prealloc: 256
   #  - size: 16
   #    prealloc: 512
   #  - size: 112
   #    prealloc: 512
   #  - size: 248
   #    prealloc: 512
   #  - size: 512
   #    prealloc: 512
   #  - size: 768
   #    prealloc: 1024
   #  - size: 1448
   #    prealloc: 1024
   #  - size: 65535
   #    prealloc: 128
   #zero-copy-size: 128

# Host table:
#
# Host table is used by tagging and per host thresholding subsystems.
#
host:
  hash-size: 4096
  prealloc: 1000
  memcap: 32mb

# IP Pair table:
#
# Used by xbits 'ippair' tracking.
#
#ippair:
#  hash-size: 4096
#  prealloc: 1000
#  memcap: 32mb


##
## Performance tuning and profiling
##

# The detection engine builds internal groups of signatures. The engine
# allow us to specify the profile to use for them, to manage memory on an
# efficient way keeping a good performance. For the profile keyword you
# can use the words "low", "medium", "high" or "custom". If you use custom
# make sure to define the values at "- custom-values" as your convenience.
# Usually you would prefer medium/high/low.
#
# "sgh mpm-context", indicates how the staging should allot mpm contexts for
```

```
# the signature groups.  "single" indicates the use of a single context for
# all the signature group heads.  "full" indicates a mpm-context for each
# group head.  "auto" lets the engine decide the distribution of contexts
# based on the information the engine gathers on the patterns from each
# group head.
#
# The option inspection-recursion-limit is used to limit the recursive calls
# in the content inspection code.  For certain payload-sig combinations, we
# might end up taking too much time in the content inspection code.
# If the argument specified is 0, the engine uses an internally defined
# default limit.  On not specifying a value, we use no limits on the recursion.
detect:
  profile: medium
  custom-values:
    toclient-groups: 3
    toserver-groups: 25
  sgh-mpm-context: auto
  inspection-recursion-limit: 3000
  # If set to yes, the loading of signatures will be made after the capture
  # is started. This will limit the downtime in IPS mode.
  #delayed-detect: yes

  # the grouping values above control how many groups are created per
  # direction. Port whitelisting forces that port to get it's own group.
  # Very common ports will benefit, as well as ports with many expensive
  # rules.
  grouping:
    #tcp-whitelist: 53, 80, 139, 443, 445, 1433, 3306, 3389, 6666, 6667, 8080
    #udp-whitelist: 53, 135, 5060

  profiling:
    # Log the rules that made it past the prefilter stage, per packet
    # default is off. The threshold setting determines how many rules
    # must have made it past pre-filter for that rule to trigger the
    # logging.
    #inspect-logging-threshold: 200
    grouping:
      dump-to-disk: false
      include-rules: false      # very verbose
      include-mpm-stats: false

# Select the multi pattern algorithm you want to run for scan/search the
# in the engine.
#
# The supported algorithms are:
# "ac"      - Aho-Corasick, default implementation
# "ac-bs"   - Aho-Corasick, reduced memory implementation
# "ac-cuda" - Aho-Corasick, CUDA implementation
# "ac-ks"   - Aho-Corasick, "Ken Steele" variant
```

```
# "hs"      - Hyperscan, available when built with Hyperscan support
#
# The default mpm-algo value of "auto" will use "hs" if Hyperscan is
# available, "ac" otherwise.
#
# The mpm you choose also decides the distribution of mpm contexts for
# signature groups, specified by the conf - "detect.sgh-mpm-context".
# Selecting "ac" as the mpm would require "detect.sgh-mpm-context"
# to be set to "single", because of ac's memory requirements, unless the
# ruleset is small enough to fit in one's memory, in which case one can
# use "full" with "ac".  Rest of the mpms can be run in "full" mode.
#
# There is also a CUDA pattern matcher (only available if Suricata was
# compiled with --enable-cuda: b2g_cuda. Make sure to update your
# max-pending-packets setting above as well if you use b2g_cuda.

mpm-algo: auto

# Select the matching algorithm you want to use for single-pattern searches.
#
# Supported algorithms are "bm" (Boyer-Moore) and "hs" (Hyperscan, only
# available if Suricata has been built with Hyperscan support).
#
# The default of "auto" will use "hs" if available, otherwise "bm".

spm-algo: auto

# Suricata is multi-threaded. Here the threading can be influenced.
threading:
  set-cpu-affinity: no
  # Tune cpu affinity of threads. Each family of threads can be bound
  # on specific CPUs.
  #
  # These 2 apply to the all runmodes:
  # management-cpu-set is used for flow timeout handling, counters
  # worker-cpu-set is used for 'worker' threads
  #
  # Additionally, for autofp these apply:
  # receive-cpu-set is used for capture threads
  # verdict-cpu-set is used for IPS verdict threads
  #
  cpu-affinity:
    - management-cpu-set:
        cpu: [ 0 ]  # include only these cpus in affinity settings
    - receive-cpu-set:
        cpu: [ 0 ]  # include only these cpus in affinity settings
    - worker-cpu-set:
        cpu: [ "all" ]
        mode: "exclusive"
```

```
      # Use explicitely 3 threads and don't compute number by using
      # detect-thread-ratio variable:
      # threads: 3
      prio:
        low: [ 0 ]
        medium: [ "1-2" ]
        high: [ 3 ]
        default: "medium"
    #- verdict-cpu-set:
    #    cpu: [ 0 ]
    #    prio:
    #      default: "high"
  #
  # By default Suricata creates one "detect" thread per available CPU/CPU core.
  # This setting allows controlling this behaviour. A ratio setting of 2 will
  # create 2 detect threads for each CPU/CPU core. So for a dual core CPU this
  # will result in 4 detect threads. If values below 1 are used, less threads
  # are created. So on a dual core CPU a setting of 0.5 results in 1 detect
  # thread being created. Regardless of the setting at a minimum 1 detect
  # thread will always be created.
  #
  detect-thread-ratio: 1.0

# Profiling settings. Only effective if Suricata has been built with the
# the --enable-profiling configure flag.
#
profiling:
  # Run profiling for every xth packet. The default is 1, which means we
  # profile every packet. If set to 1000, one packet is profiled for every
  # 1000 received.
  #sample-rate: 1000

  # rule profiling
  rules:

    # Profiling can be disabled here, but it will still have a
    # performance impact if compiled in.
    enabled: yes
    filename: rule_perf.log
    append: yes

    # Sort options: ticks, avgticks, checks, matches, maxticks
    sort: avgticks

    # Limit the number of items printed at exit (ignored for json).
    limit: 100

    # output to json
    json: yes
```

```
 # per keyword profiling
 keywords:
  enabled: yes
  filename: keyword_perf.log
  append: yes

 # per rulegroup profiling
 rulegroups:
  enabled: yes
  filename: rule_group_perf.log
  append: yes

 # packet profiling
 packets:

  # Profiling can be disabled here, but it will still have a
  # performance impact if compiled in.
  enabled: yes
  filename: packet_stats.log
  append: yes

  # per packet csv output
  csv:

    # Output can be disabled here, but it will still have a
    # performance impact if compiled in.
    enabled: no
    filename: packet_stats.csv

 # profiling of locking. Only available when Suricata was built with
 # --enable-profiling-locks.
 locks:
  enabled: no
  filename: lock_stats.log
  append: yes

 pcap-log:
  enabled: no
  filename: pcaplog_stats.log
  append: yes

##
## Netfilter integration
##

# When running in NFQ inline mode, it is possible to use a simulated
# non-terminal NFQUEUE verdict.
# This permit to do send all needed packet to suricata via this a rule:
```

```
#       iptables -I FORWARD -m mark ! --mark $MARK/$MASK -j NFQUEUE
# And below, you can have your standard filtering ruleset. To activate
# this mode, you need to set mode to 'repeat'
# If you want packet to be sent to another queue after an ACCEPT decision
# set mode to 'route' and set next-queue value.
# On linux >= 3.1, you can set batchcount to a value > 1 to improve performance
# by processing several packets before sending a verdict (worker runmode only).
# On linux >= 3.6, you can set the fail-open option to yes to have the kernel
# accept the packet if suricata is not able to keep pace.
nfq:
#  mode: accept
#  repeat-mark: 1
#  repeat-mask: 1
#  route-queue: 2
#  batchcount: 20
#  fail-open: yes

#nflog support
nflog:
   # netlink multicast group
   # (the same as the iptables --nflog-group param)
   # Group 0 is used by the kernel, so you can't use it
 - group: 2
   # netlink buffer size
   buffer-size: 18432
   # put default value here
 - group: default
   # set number of packet to queue inside kernel
   qthreshold: 1
   # set the delay before flushing packet in the queue inside kernel
   qtimeout: 100
   # netlink max buffer size
   max-size: 20000

##
## Advanced Capture Options
##

# Netmap support
#
# Netmap operates with NIC directly in driver, so you need FreeBSD wich have
# built-in netmap support or compile and install netmap module and appropriate
# NIC driver on your Linux system.
# To reach maximum throughput disable all receive-, segmentation-,
# checksum- offloadings on NIC.
# Disabling Tx checksum offloading is *required* for connecting OS endpoint
# with NIC endpoint.
# You can find more information at https://github.com/luigirizzo/netmap
#
```

```
netmap:
  # To specify OS endpoint add plus sign at the end (e.g. "eth0+")
 - interface: eth2
  # Number of receive threads. "auto" uses number of RSS queues on interface.
  #threads: auto
  # You can use the following variables to activate netmap tap or IPS mode.
  # If copy-mode is set to ips or tap, the traffic coming to the current
  # interface will be copied to the copy-iface interface. If 'tap' is set, the
  # copy is complete. If 'ips' is set, the packet matching a 'drop' action
  # will not be copied.
  # To specify the OS as the copy-iface (so the OS can route packets, or forward
  # to a service running on the same machine) add a plus sign at the end
  # (e.g. "copy-iface: eth0+"). Don't forget to set up a symmetrical eth0+ -> eth0
  # for return packets. Hardware checksumming must be *off* on the interface if
  # using an OS endpoint (e.g. 'ifconfig eth0 -rxcsum -txcsum -rxcsum6 -txcsum6' for
FreeBSD
  # or 'ethtool -K eth0 tx off rx off' for Linux).
  #copy-mode: tap
  #copy-iface: eth3
  # Set to yes to disable promiscuous mode
  # disable-promisc: no
  # Choose checksum verification mode for the interface. At the moment
  # of the capture, some packets may be with an invalid checksum due to
  # offloading to the network card of the checksum computation.
  # Possible values are:
  #  - yes: checksum validation is forced
  #  - no: checksum validation is disabled
  #  - auto: suricata uses a statistical approach to detect when
  #  checksum off-loading is used.
  # Warning: 'checksum-validation' must be set to yes to have any validation
  #checksum-checks: auto
  # BPF filter to apply to this interface. The pcap filter syntax apply here.
  #bpf-filter: port 80 or udp
 #- interface: eth3
  #threads: auto
  #copy-mode: tap
  #copy-iface: eth2
  # Put default values here
 - interface: default

# PF_RING configuration. for use with native PF_RING support
# for more info see http://www.ntop.org/products/pf_ring/
pfring:
  - interface: eth0
    # Number of receive threads (>1 will enable experimental flow pinned
    # runmode)
    threads: 1

    # Default clusterid.  PF_RING will load balance packets based on flow.
```

```
    # All threads/processes that will participate need to have the same
    # clusterid.
    cluster-id: 99

    # Default PF_RING cluster type. PF_RING can load balance per flow.
    # Possible values are cluster_flow or cluster_round_robin.
    cluster-type: cluster_flow
    # bpf filter for this interface
    #bpf-filter: tcp
    # Choose checksum verification mode for the interface. At the moment
    # of the capture, some packets may be with an invalid checksum due to
    # offloading to the network card of the checksum computation.
    # Possible values are:
    #  - rxonly: only compute checksum for packets received by network card.
    #  - yes: checksum validation is forced
    #  - no: checksum validation is disabled
    #  - auto: suricata uses a statistical approach to detect when
    #  checksum off-loading is used. (default)
    # Warning: 'checksum-validation' must be set to yes to have any validation
    #checksum-checks: auto
  # Second interface
  #- interface: eth1
  #  threads: 3
  #  cluster-id: 93
  #  cluster-type: cluster_flow
  # Put default values here
  - interface: default
    #threads: 2

# For FreeBSD ipfw(8) divert(4) support.
# Please make sure you have ipfw_load="YES" and ipdivert_load="YES"
# in /etc/loader.conf or kldload'ing the appropriate kernel modules.
# Additionally, you need to have an ipfw rule for the engine to see
# the packets from ipfw.  For Example:
#
#   ipfw add 100 divert 8000 ip from any to any
#
# The 8000 above should be the same number you passed on the command
# line, i.e. -d 8000
#
ipfw:

  # Reinject packets at the specified ipfw rule number.  This config
  # option is the ipfw rule number AT WHICH rule processing continues
  # in the ipfw processing system after the engine has finished
  # inspecting the packet for acceptance.  If no rule number is specified,
  # accepted packets are reinjected at the divert rule which they entered
  # and IPFW rule processing continues.  No check is done to verify
  # this will rule makes sense so care must be taken to avoid loops in ipfw.
```

```
 #
 ## The following example tells the engine to reinject packets
 # back into the ipfw firewall AT rule number 5500:
 #
 # ipfw-reinjection-rule-number: 5500


napatech:
   # The Host Buffer Allowance for all streams
   # (-1 = OFF, 1 - 100 = percentage of the host buffer that can be held back)
   hba: -1

   # use_all_streams set to "yes" will query the Napatech service for all configured
   # streams and listen on all of them. When set to "no" the streams config array
   # will be used.
   use-all-streams: yes

   # The streams to listen on
   streams: [1, 2, 3]

# Tilera mpipe configuration. for use on Tilera TILE-Gx.
mpipe:

 # Load balancing modes: "static", "dynamic", "sticky", or "round-robin".
 load-balance: dynamic

 # Number of Packets in each ingress packet queue. Must be 128, 512, 2028 or 65536
 iqueue-packets: 2048

 # List of interfaces we will listen on.
 inputs:
 - interface: xgbe2
 - interface: xgbe3
 - interface: xgbe4


 # Relative weight of memory for packets of each mPipe buffer size.
 stack:
   size128: 0
   size256: 9
   size512: 0
   size1024: 0
   size1664: 7
   size4096: 0
   size10386: 0
   size16384: 0

##
## Hardware accelaration
```

```
##

# Cuda configuration.
cuda:
  # The "mpm" profile.  On not specifying any of these parameters, the engine's
  # internal default values are used, which are same as the ones specified in
  # in the default conf file.
  mpm:
    # The minimum length required to buffer data to the gpu.
    # Anything below this is MPM'ed on the CPU.
    # Can be specified in kb, mb, gb.  Just a number indicates it's in bytes.
    # A value of 0 indicates there's no limit.
    data-buffer-size-min-limit: 0
    # The maximum length for data that we would buffer to the gpu.
    # Anything over this is MPM'ed on the CPU.
    # Can be specified in kb, mb, gb.  Just a number indicates it's in bytes.
    data-buffer-size-max-limit: 1500
    # The ring buffer size used by the CudaBuffer API to buffer data.
    cudabuffer-buffer-size: 500mb
    # The max chunk size that can be sent to the gpu in a single go.
    gpu-transfer-size: 50mb
    # The timeout limit for batching of packets in microseconds.
    batching-timeout: 2000
    # The device to use for the mpm.  Currently we don't support load balancing
    # on multiple gpus.  In case you have multiple devices on your system, you
    # can specify the device to use, using this conf.  By default we hold 0, to
    # specify the first device cuda sees.  To find out device-id associated with
    # the card(s) on the system run "suricata --list-cuda-cards".
    device-id: 0
    # No of Cuda streams used for asynchronous processing. All values > 0 are valid.
    # For this option you need a device with Compute Capability > 1.0.
    cuda-streams: 2

##
## Include other configs
##

# Includes.  Files included here will be handled as if they were
# inlined in this configuration file.
#include: include1.yaml
#include: include2.yaml
```