ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Σχεδιασμός και Ανάπτυξη Μηχανισμού Ανταλλαγής Δικτυακών Συμβάντων σε Συνεργαζόμενα Δίκτυα

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

Ευάγγελου Γιαννόπουλου

Επιβλέπων : Μάγκλαρης Βασίλειος
Καθηγητής Ε.Μ.Π.

Αθήνα, Νοέμβριος 2017

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Σχεδιασμός και ανάπτυξη Μηχανισμού Ανταλλαγής Δικτυακών Συμβάντων σε Συνεργαζόμενα Δίκτυα

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

Ευάγγελου Γιαννόπουλου

Επιβλέπων : Μάγκλαρης Βασίλειος
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 7η Νοεμβρίου 2017.

(Υπογραφή)	*(Υπογραφή)*	*(Υπογραφή)*
.....	.....	.....
Μάγκλαρης Βασίλειος	Νεκταριος Κοζύρης	Ευστάθιος Συκάς
Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.

Αθήνα, Νοέμβριος 2017

(Υπογραφή)

.....

Ευάγγελος Γιαννόπουλος

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

Ευχαριστίες

Η παρούσα διπλωματική εργασία αποτελεί το τελευταίο στάδιο των προπτυχιακών σπουδών μου στο Εθνικό Μετσόβιο Πολυτεχνείο.

Αρχικά θα ήθελα να ευχαριστήσω τον καθηγητή μου κ. Βασίλειο Μάγκλαρη για την εμπιστοσύνη που μου έδειξε, καθ'όλη τη διάρκεια της εκπόνησης της παρούσας εργασίας. Επιπλέον, θα ήθελα να ευχαριστήσω ιδιαιτέρως τον υποψήφιο διδάκτορα Αδάμ Παυλίδη, για την άριστη συνεργασία μας καθώς και για την πολύτιμη καθοδήγηση του σε όλα τα στάδια εκπόνησης της.

Τέλος θα ήθελα να ευχαριστήσω την οικογένεια και τους φίλους μου για την αμέριστη υποστήριξη και συμπαράσταση τους, στη διάρκεια των σπουδών μου.

Περίληψη

Οι διαδικτυακές επιθέσεις έχουν πλέον πάρει τη θέση αυτών που απαιτούν φυσική παρουσία και αυτό διότι για πολλά χρόνια οι αμυντικοί μηχανισμοί σε τράπεζες, συστήματα πληρωμών και άλλες υπηρεσίες οι οποίες μπορούν να αποφέρουν κέρδη στους δράστες, δεν χρησιμοποιούσαν τα κατάλληλα μέτρα ασφαλείας. Με την πάροδο των χρόνων, και την μεταφορά περισσότερων υπηρεσιών στο διαδίκτυο, αυτή η κατάσταση έχει αλλάξει, καθώς τα σύγχρονα δίκτυα τα οποία παρέχουν υπηρεσίες-στόχους προφυλάσσονται πιο αποτελεσματικά.
Η παρούσα διπλωματική στοχεύει στην πρόταση και υλοποίηση ενός μηχανισμού για ανταλλαγή δικτυακών συμβάντων σε συνεργαζόμενες κοινότητες (δημόσιοι οργανισμοί, εκπαιδευτικά ιδρύματα, ακόμη και ιδιωτικές εταιρίες). Παράλληλα, προτείνουμε τη χρήση της έννοιας της «φήμης» των διαφόρων συμμετεχόντων, με σκοπό την δημιουργία κινήτρων για την συνεχή και καλόβουλη χρήση του μηχανισμού.
Επίσης, εκτός από το μηχανισμό ανταλλαγής πληροφορίας, κάνουμε χρήση της διαδιδόμενης πληροφορίας για την ανίχνευση επιθέσεων οι οποίες δεν μπορούν να εντοπιστούν με τη χρήση των περισσότερων δικτυακών και μη λύσεων οι οποίες χρησιμοποιούνται. Η κατηγορία αυτών των επιθέσεων αφορά κυρίως το στρώμα εφαρμογής, όπως για παράδειγμα επιθέσεις brute force σε HTTP φόρμες.

Λέξεις Κλειδιά: <<Συνεργατική αντιμετώπιση και πρόληψη, τεχνικές αντιμετώπισης επιθέσεων, τεχνικές συλλογής αρχείων καταγραφής, ασφάλεια δικτύων>>

Abstract

Cyber-attacks have taken place of attacks that require physical presence, and that is because for a long time defense mechanisms in banks, payment systems and all sorts of security-critical services, were quite immature or inadequately tested before reaching production. As the security field progressed and more security-critical services reached the internet world, the networks that support them have started deploying more effective mechanisms to defend themselves.

This paper proposes a mechanism that facilitates event sharing between peers in collaborating environments (public sector organizations, education institutions, and even private corporations). Furthermore, we propose the use of "reputation" for peer evaluation in order to give incentives to the community to take part in the mechanism in a benign and continuous manner.

In addition, apart from the sharing mechanism, we take advantage of the shared data that we collect in order to look for attacks that cannot be detected by the modern defense mechanisms. The category that all these attacks fall into is the application layer oriented attacks.

Keywords: <<Cooperative Mitigation and Prevention, Mitigation Techniques, Log Management, Network Security>>

Πίνακας περιεχομένων

1

Εισαγωγή

1.1 Αντικείμενο

Με τη ραγδαία και συνεχή ανάπτυξη του Internet, ο ιδιωτικός όσο και ο δημόσιος τομέας έχουν οικοδομήσει έναν διαδικτυακό κόσμο, ο οποίος έχει διεισδύσει στη καθημερινότητά μας. Άμεση συνέπεια αυτού είναι όλο και περισσότερες υπηρεσίες να παρέχονται πλέον μέσω διαδικτύου, γεγονός που έχει διευκολύνει δραστηριότητες που άλλοτε ήταν χρονοβόρες. Απόρροια αυτού όμως είναι και η μετακίνηση διάφορων ειδών δραστών από τον φυσικό κόσμο σε αυτόν του διαδικτύου.

Η μετακίνηση αυτή έχει οδηγήσει σε μια αυξημένη ανάγκη από την κοινότητα παροχής υπηρεσιών κάθε είδους για διαρκή εποπτεία και αναβάθμιση της ασφάλειας. Για το λόγο αυτό όλο και περισσότερες λύσεις προτείνονται για την παρακολούθηση και έπειτα δράση ενάντια στους επίδοξους εισβολείς. Έτσι λοιπόν, όλα τα σύγχρονα δίκτυα χρησιμοποιούν κάποια μορφή εποπτείας, είτε μέσω δεδομένων που διαπερνούν το δίκτυο, όπως για παράδειγμα, ένα σύστημα ανίχνευσης εισβολής (IDS) ή πρωτόκολλα όπως sFlow και NetFlow, είτε μέσω αρχείων καταγραφής, τα οποία βρίσκονται στα διάφορα μηχανήματα και πρέπει να συλλεχθούν σε ένα κεντρικό σημείο.

Στο πλαίσιο της παρούσας διπλωματικής προτείνεται ένα σύστημα για τη συλλογή πληροφορίας (η οποία βρίσκεται σε αρχεία καταγραφής), την περαιτέρω επεξεργασία της πληροφορίας με σκοπό την ομογενοποίηση της, καθώς επίσης και για την ανίχνευση

επιθέσεων στους διάφορους συμμετέχοντες. Έμφαση δίνεται στη σημαντικότητα της συνεργασίας μεταξύ των διάφορων οργανισμών για την αποτροπή ή την αντιμετώπιση τυχόν επιθέσεων που πλήττουν παραπάνω από έναν συμμετέχοντα. Το γεγονός ότι η συνεργατικότητα μπορεί να οδηγήσει σε πολύ πιο αποτελεσματική αντιμετώπιση, ακόμη και πρόληψη κατά επιθέσεων, επιβεβαιώνεται και από τα εξαγόμενα αποτελέσματα.

1.2 Δομή Διπλωματικής

Η παρούσα εργασία αποτελείται από έξι κεφάλαια. Το δεύτερο κεφάλαιο θεμελιώνει το πρόβλημα το οποίο προσπαθούμε να ερευνήσουμε, γιατί άλλες λύσεις έχουν αποτύχει ή μερικώς επιτύχει να το επιλύσουν, καθώς επίσης παρέχει το θεωρητικό υπόβαθρο με το οποίο ο αναγνώστης θα είναι σε θέση να παρακολουθήσει την πορεία της υπόλοιπης εργασίας. Το τρίτο κεφάλαιο αφορά την αρχιτεκτονική της προτεινόμενης λύσης, τα νοητά κομμάτια που την απαρτίζουν καθώς και το ρόλο που εξυπηρετεί το καθένα. Στο επόμενο κεφάλαιο επεξηγείται η υλοποίηση σε μεγαλύτερο βάθος και ως εκ τούτου περιέχει μερικές εισαγωγικές υποενότητες οι οποίες περιλαμβάνουν το απαραίτητο θεωρητικό υπόβαθρο. Στη συνέχεια, το πέμπτο κεφάλαιο αφορά την πειραματική διάταξη η οποία χρησιμοποιήθηκε για την εκτίμηση της χρησιμότητας αλλά και αποδοτικότητας της προτεινόμενης λύσης, καθώς επίσης και κάποιες μετρικές οι οποίες θεμελιώνουν τα παραπάνω. Τέλος, στο έκτο κεφάλαιο παρουσιάζεται η σύνοψη της εργασίας, όπως επίσης και βελτιώσεις που μπορούν να υλοποιηθούν μελλοντικά.

2

Θεωρητικό Υπόβαθρο

2.1 Σύγχρονο Internet και κίνδυνοι

Όπως προαναφέρθηκε και στην εισαγωγή το σύγχρονο Internet έχει παρεισφρήσει στη ζωή μας, με αποτέλεσμα να εγκυμονούν και σοβαροί κίνδυνοι καθώς μοιραζόμαστε εξαιρετικά ευαίσθητα δεδομένα με τον κόσμο του Internet. Τα δεδομένα αυτά περιλαμβάνουν κρατικά δεδομένα τα οποία μας προσδιορίζουν μοναδικά, όπως το ΑΦΜ και το ΑΜΚΑ. Έπειτα επεκτεινόμαστε σε άλλα προσωπικά δεδομένα, όπως φωτογραφίες, ακριβής τοποθεσία στην οποία βρισκόμαστε μέσω των κινητών συσκευών μας, καθώς και πληροφορίες όπως μόνιμη κατοικία, ηλικία κ.α. Τέλος, άξιο αναφοράς είναι το γεγονός ότι οι περισσότερες συναλλαγές στη σύγχρονη αγορά γίνονται μέσω διαδικτύου, με αποτέλεσμα οι διαδικτυακές εφαρμογές των τραπεζών καθώς και οι χρήστες αυτών να γίνονται στόχοι επίδοξων εισβολέων.

Ως αποτέλεσμα των παραπάνω, ο χώρος της ασφάλειας πληροφοριών είναι σε διαρκή αγώνα να βρίσκονται ένα βήμα μπροστά από τους επίδοξους παραβάτες, οι οποίοι έχουν ως δέλεαρ τα χρήματα, κυβερνητικά είτε ακόμη και προσωπικά συμφέροντα.

Το πρώτο στάδιο για την αντιμετώπιση επιθέσεων στο διαδίκτυο είναι η παρακολούθηση των δικτύων, με σκοπό να παράγουμε πληροφορία με τη βοήθεια της οποίας μπορούμε να εντοπίσουν οποιαδήποτε ανωμαλία και στη συνέχεια να προβούμε στην απομόνωση ή στον αποκλεισμό των επιτιθέμενων.

2.2 Συχνότερες επιθέσεις

2.2.1 Δικτυακές επιθέσεις

Μια πρώτη κατηγορία επιθέσεων αφορά τις επιθέσεις οι οποίες μπορούν να ανιχνευθούν στο επίπεδο του δικτύου και δεν απαιτούν τη βαθύτερη εξέταση των πακέτων έως το στρώμα εφαρμογής. Παρακάτω παρατίθενται μερικές από αυτές τις επιθέσεις.

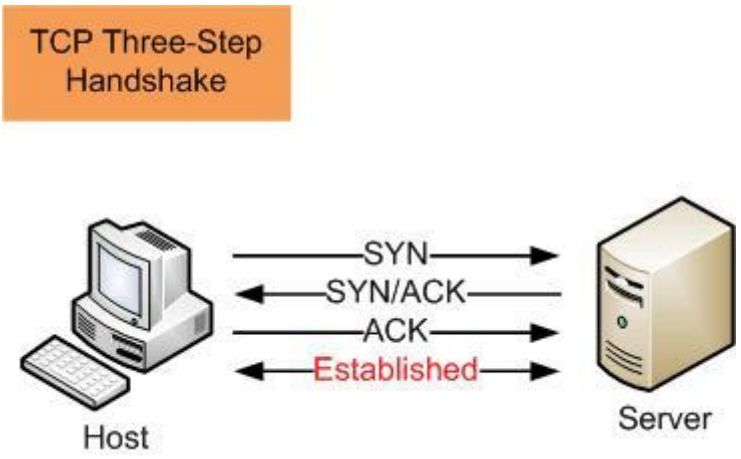
2.2.1.1 DDoS

Στις σύγχρονες εφαρμογές, η διαρκής κάλυψη αποτελεί ιδιότητα εξαιρετικής σημασίας και η απουσία αυτής μπορεί να κοστίσει σημαντικά στον παροχέα μιας υπηρεσίας. Αποτέλεσμα λοιπόν είναι οι επιτιθέμενοι να αναπτύσσουν επιθέσεις άρνησης παροχής υπηρεσίας (DoS). Μια κατηγορία αυτών των επιθέσεων είναι και η κατανεμημένη επίθεση άρνησης παροχής υπηρεσίας (DDoS).

Εμείς αναφερόμαστε σε μια κατηγορία DDoS επιθέσεων, η οποία ονομάζεται ογκομετρική DDoS. Η ονομασία αυτή προκύπτει από το γεγονός ότι ο επιτιθέμενος κάνει χρήση όλου του υπολογιστικού και δικτυακού όγκου που διαθέτει και τον χρησιμοποιεί για να παραλύσει το δίκτυο του παροχέα.

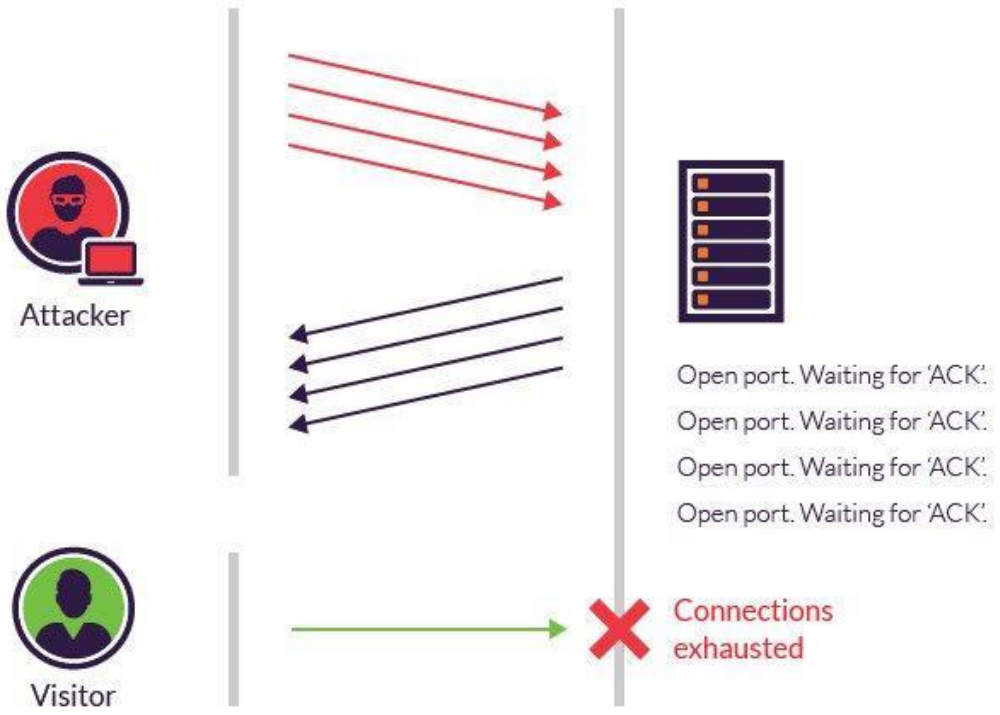
2.2.1.2 SYN Flood

Όπως φαίνεται και στην παρακάτω εικόνα, κατά τη διάρκεια της τριμερούς χειραψίας για την εγκαθίδρυση μιας TCP, ο πελάτης στέλνει ένα αίτημα με το οποίο ζητά μια TCP σύνδεση μέσω ενός TCP πακέτου με τη σημαία SYN. Στη συνέχεια, ο εξυπηρετητής απαντά με ένα πακέτο αναγνώρισης του αιτήματος το οποίο φέρει τις σημαίες SYN και ACK. Σε αυτό το στάδιο ο εξυπηρετητής δεσμεύει μνήμη για την αποθήκευση δεδομένων σχετικών με τη συγκεκριμένη σύνδεση. Η δομή στην οποία αποθηκεύονται οι συνδέσεις, όπως είναι φυσιολογικό, έχει περιορισμένο χώρο. Ως εκ τούτου, οι πόροι του συστήματος εξαντλούνται με τον κατάλληλο όγκο κίνησης, ο οποίος εξαρτάται από το λειτουργικό σύστημα καθώς και από τα τεχνικά χαρακτηριστικά του εξυπηρετητή.



Εικόνα 1 – Three-way Handshake

Όπως μπορεί να φανταστεί κανείς, ο επιτιθέμενος προσπαθεί να εξαντλήσει τους πόρους του συστήματος, και ο τρόπος με τον οποίο θα το επιδιώξει φαίνεται στο παρακάτω σχήμα.

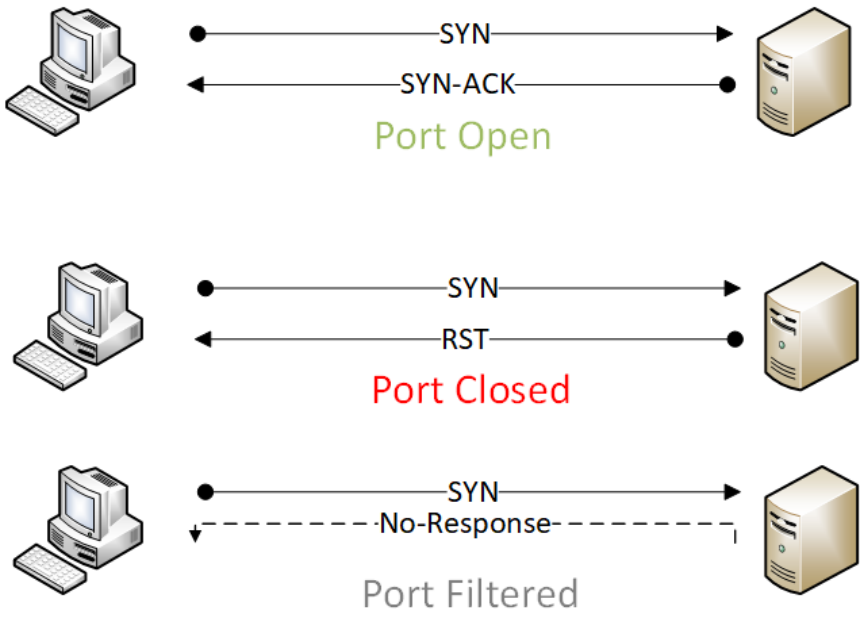


Εικόνα 2 – Επίθεση Syn Flood

Στο παρόν σχήμα βλέπουμε ότι ο επιτιθέμενος στέλνει όσα πακέτα TCP με τη σημαία SYN του επιτρέπει το εύρος ζώνης του δικτύου του. Ως εκ τούτου, δεσμεύεται το μεγαλύτερο μέρος της μνήμης του εξυπηρετητή και έτσι οι καλόβουλοι χρήστες δε μπορούν να χρησιμοποιήσουνε την υπηρεσία την οποία παρέχει ο εξυπηρετητής.

2.2.1.3 Port Scanning

Το σκανάρισμα πόρτας (port scanning) αποτελεί την πρώτη προσέγγιση ενός επιτιθέμενου σε κάποιο μηχάνημα-στόχο. Μέσω της επίθεσης αυτής, μπορούμε να βρούμε ποιές πόρτες είναι ανοιχτές, κλειστές ή φιλτραρισμένες. Στο παρακάτω σχήμα φαίνεται ένα είδος port scanning, το οποίο ονομάζεται TCP Half-Open.



Εικόνα 3 – Port Scanning

Η επίθεση αυτή αποτελεί την πιο απλή αλλά ευρέως χρησιμοποιούμενη μορφή port scanning. Όπως βλέπουμε και παραπάνω, ο επιτιθέμενος στέλνει ένα TCP πακέτο με τη σημαία SYN, ενώ ο εξυπηρετητής απαντά με τρείς διαφορετικούς τρόπους:

A. Με ένα πακέτο TCP με τις σημαίες SYN και ACK, το οποίο σημαίνει ότι ο εξυπηρετητής αποδέχεται το αίτημα εγκαθίδρυσης σύνδεσης, οπότε η πόρτα είναι ανοιχτή.

B. Με ένα πακέτο TCP με τη σημαία RST, το οποίο σημαίνει ότι το αίτημα απορρίπτεται και άρα η πόρτα είναι κλειστή.

C. Εάν για οποιοδήποτε λόγο δεν λάβουμε απάντηση, η πόρτα θεωρείται φιλτραρισμένη από το τοίχος προστασίας.

2.2.2 Επιθέσεις επιπέδου εφαρμογής

2.2.2.1 Επίθεση Brute Force

Όπως προαναφέρθηκε, οι χρήστες διαφόρων υπηρεσιών του διαδικτύου χρησιμοποιούν διαπιστευτήρια της ταυτότητάς τους και έτσι έχουν πρόσβαση σε μια εξατομικευμένη εκδοχή της υπηρεσίας. Τα διαπιστευτήρια αυτά στις περισσότερες περιπτώσεις αποτελούν το μόνο μέσο ταυτοποίησης. Αποτέλεσμα αυτών λοιπόν είναι η χρήση ωμής βίας (brute force) με σκοπό την ανάκτηση των διαπιστευτηρίων του στόχου. Ο επιτιθέμενος κάνει χρήση της υπολογιστικής και δικτυακής δύναμης που διαθέτει για την εξαντλητική δοκιμή όλων των πιθανών συνδυασμών διαπιστευτηρίων με σκοπό την εύρεση του αποδεκτού συνδυασμού. Να σημειωθεί ότι σε περιπτώσεις όπου τα διαπιστευτήρια πρέπει να επαληθευτούν από έναν απομακρυσμένο εξυπηρετητή, ο ρυθμός με τον οποίο ο επιτιθέμενος μπορεί να στέλνει αιτήματα επαλήθευσης είναι αρκετά περιορισμένος εφόσον ο διαχειριστής του εξυπηρετητή έχει μεριμνήσει για αυτό.

2.2.2.1.1 SSH

Το πρωτόκολλο SSH ή αλλιώς Secure Shell, αποτελεί το πιο ευρέως διαδεδομένο πρωτόκολλο για την απομακρυσμένη διαχείριση μηχανημάτων. Το κύριο χαρακτηριστικό το οποίο παρέχει το συγκεκριμένο πρωτόκολλο είναι η ασφάλεια, καθότι όλοι οι προκάτοχοι του (π.χ. telnet, ftp) είτε δεν υποστήριζαν τη δημιουργία ασφαλούς καναλιού για την επικοινωνία είτε τα μέτρα ασφαλείας δεν ήταν επαρκή για την αποφυγή γνωστών επιθέσεων τύπου Man-in-the-middle.

Λόγω της ευρείας υιοθέτησης του πρωτοκόλλου από οργανισμούς και τον ιδιωτικό τομέα, οι χάκερς έχουν αναπτύξει εργαλεία όπου αναζητούν διαπιστευτήρια τα οποία είτε χρησιμοποιούνται συχνά είτε βρίσκονται σε λίστες από παλαιότερες διαρροές διαπιστευτηρίων. Σε αυτό το σημείο πρέπει να αναφερθεί ότι μελέτες δείχνουν πως οι περισσότεροι χρήστες του διδικτύου, χρησιμοποιούν τον ίδιο κωδικό για πολλαπλές υπηρεσίες. Αυτό το γεγονός διευκολύνει τους επιτιθέμενους στην εύρεση των διαπιστευτηρίων [3].

Για το σκοπό αυτό οι επιτιθέμενοι ανιχνεύουν μεγάλες περιοχές IP «χτυπώντας» όλες τις πόρτες των μηχανημάτων που είναι ενεργά, αναζητώντας εξυπηρετητές SSH και έπειτα χρησιμοποιούν επιθέσεις brute force για την ανάκτηση έγκυρων διαπιστευτηρίων. Οι διαχειριστές διαφόρων δικτύων παρατήρησαν αυξημένη κίνηση όταν χρησιμοποιούσαν την καθιερωμένη πόρτα (22) και έτσι, με στόχο να αποφύγουν τυχόν αστάθεια της παρεχόμενης υπηρεσίας, είτε ακόμη και παρείσφρηση στο εσωτερικό του δικτύου, μετέφεραν τον

εξυπηρετητή SSH σε μια μη-συνηθισμένη πόρτα. Για αυτόν ακριβώς το λόγο, οι επιτιθέμενοι αναζητούν SSH εξυπηρετητές σε όλες τις πιθανές πόρτες.

2.2.2.2 Αναγνώριση Στόχου

Η αναγνώριση στόχου (Target Reconnaissance) αποτελεί μία από τις πρώτες προσεγγίσεις ενός επιτιθέμενου με έναν στόχο. Στο στάδιο αυτό, ο επιτιθέμενος προσπαθεί να αναγνωρίσει πιθανές ευάλωτες πτυχές του στόχου, τις οποίες στη συνέχεια θα προσπαθήσει να εκμεταλλευτεί για να διεισδύσει στο εσωτερικό του. Για παράδειγμα, γνωστές ομάδες επιτιθέμενων που συντηρούν τα μεγαλύτερα δίκτυα από μηχανήματα-ζόμπι, με σκοπό να ενσωματώσουν περισσότερα μηχανήματα στο δίκτυο τους, αναζητούν εύκολους στόχους με εκτεθειμένο λογισμικό με γνωστές τρύπες ασφαλείας [2]. Μερικά από αυτά τα εργαλεία παρατίθενται παρακάτω:

- Wpscan
- Joomscan
- WebSploit

Για παράδειγμα, το Wpscan αποτελεί ένα εργαλείο το οποίο στοχεύει στην αναγνώριση ύπαρξης τρωτών σημείων ασφαλείας μέσω αναζήτησης γνωστών URI (Uniform Resource Identifier) τα οποία μπορεί να μην έχουν ενημερωθεί με τις τελευταίες αναβαθμίσεις. Ειδικότερα, το συγκεκριμένο αφορά τις ιστοσελίδες οι οποίες έχουν δημιουργηθεί με το πλαίσιο ανάπτυξης Wordpress, το οποίο χρησιμοποιείται σε περίπου 25% του συνολικού αριθμού συστημάτων διαχείρισης περιεχομένου (CMS, Content Management System). Για αυτό το σκοπό το εργαλείο αναζητά γνωστά URI τα οποία δίνουν πληροφορίες για την έκδοση του Wordpress που χρησιμοποιεί η σελίδα, καθώς επίσης και για όλες τις χρησιμοποιούμενες επεκτάσεις. Ο συγκεκριμένος τρόπος επίθεσης είναι εύκολο να διαπιστωθεί στο επίπεδο του εξυπηρετητή, αλλά λόγω της κρυπτογράφησης της κίνησης από και προς τον εξυπηρετητή, λύσεις όπως τα IDS αποτυγχάνουν να τον ανιχνεύσουν.

2.3 Τρόποι συλλογής και χρήσης πληροφορίας

2.3.1 Παρακολούθηση κίνησης στο δίκτυο

2.3.1.1 sFlow

Το πρωτόκολλο sFlow (sampled flow) πρωτοεμφανίστηκε με σκοπό να καλύψει την ανάγκη για κλιμάκωση (scalability) των αρχιτεκτονικών παρακολούθησης δικτύων, καθώς σε σύγχρονα δίκτυα δεν υπάρχει η δυνατότητα για ανάλυση όλων των πακέτων 1:1. Προς αυτή

την κατεύθυνση, το πρωτόκολλο αναλαμβάνει τη συλλογή πληροφορίας για την κίνηση που διέρχεται από τις συσκευές που το υποστηρίζουν. Το βασικό χαρακτηριστικό του όμως είναι ότι, όπως υποδηλώνει και το όνομα του, πρόκειται για μέθοδο η οποία βασίζεται στη δειγματοληψία. Υπάρχουν δύο μέθοδοι δειγματοληψίας οι οποίες χρησιμοποιούνται. Η πρώτη είναι η δειγματοληψία ανά τακτά χρονικά διαστήματα (π.χ. 1 πακέτο/μsec), ενώ η δεύτερη βασίζεται στην τυχαία λήψη ενός ανά Ν πακέτα. Το μεγάλο μειονέκτημα της δεύτερης μεθόδου είναι ότι το δείγμα μεγαλώνει αναλογικά με το μέγεθος της εισερχόμενης κίνησης.

2.3.1.2 NetFlow

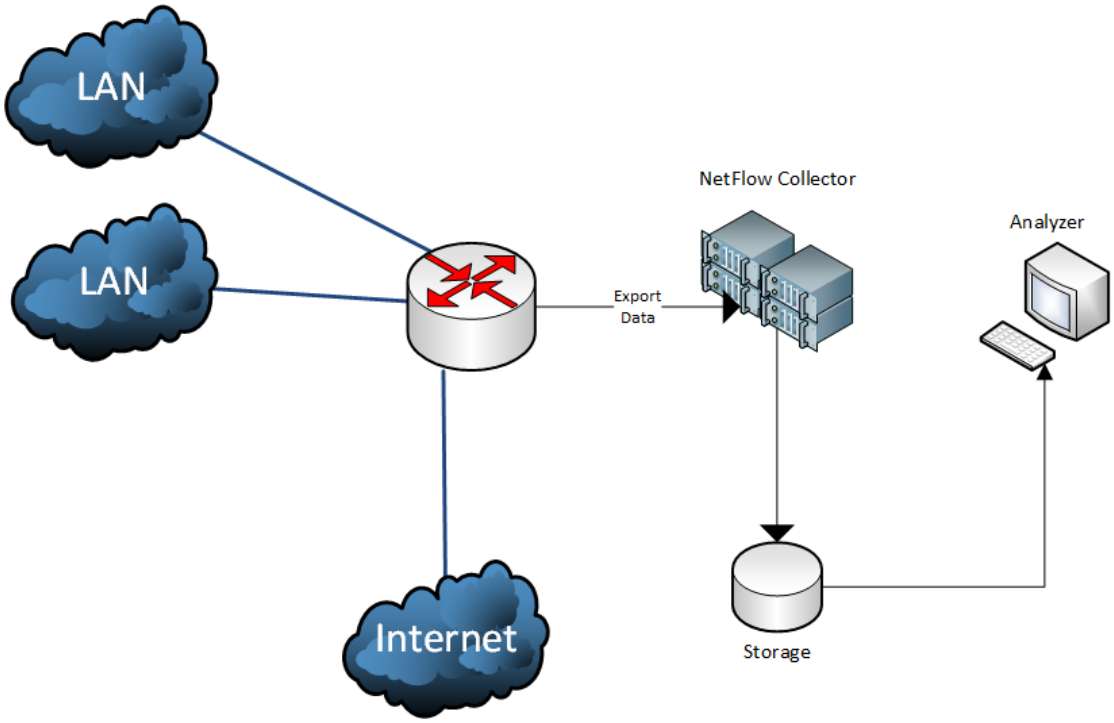
Το NetFlow αποτελεί ένα από τα ευρέως διαδεδομένα πρωτόκολλα για μοντέρνα δικτυακή παρακολούθηση και αποτίμηση της ασφάλειας ενός δικτύου. Το πρωτόκολλο αυτό πρωτοεμφανίστηκε στο Cisco iOS, αλλά πλέον έχει εδραιωθεί στο χώρο του δικτυακού κόσμου και έτσι και άλλοι οργανισμοί ανέπτυξαν τις δικές του εκδοχές του NetFlow. Ο τρόπος με τον οποίο χρησιμοποιείται το NetFlow είναι η εγκατάσταση του σε διάφορους δρομολογητές, οι οποίοι συλλέγουν δεδομένα για την εξερχόμενη ή/και εισερχόμενη κίνηση κάποιας διεπαφής (interface). Σε τακτά χρονικά διαστήματα οι δρομολογητές προωθούν τα δεδομένα αυτά σε ένα κεντρικό σημείο, το οποίο λειτουργεί ως αποθηκευτικός χώρος και το σημείο από το οποίο η συγκεντρωμένη πληροφορία μπορεί να επεξεργαστεί και να οπτικοποιηθεί.

Για τη χρήση του πρωτοκόλλου απαραίτητος είναι ο προσδιορισμός της διεύθυνσης και πόρτας όπου θα προωθούν οι δρομολογητές τα δεδομένα, καθώς επίσης και της διεπαφής της οποίας την κίνηση θέλουμε να παρακολουθούμε. Τα συλλεγόμενα δεδομένα των δρομολογητών προωθούνται στο κεντρικό σημείο, όταν μια ροή (flow) είναι ανενεργή για κάποιο διάστημα, όταν δηλαδή δεν έχει παρατηρηθεί κάποιο πακέτο που να ταιριάζει με την εν λόγω ροή ή στην περίπτωση που η ροή είναι ενεργή για ένα προκαθορισμένο χρονικό διάστημα. Οι πληροφορίες οι οποίες περιέχονται σε μια ροή φαίνονται παρακάτω:

- Διεπαφή εισόδου.
- Διεπαφή εξόδου (ή μηδέν, σε περίπτωση που το πακέτο απορρίφθηκε).
- Χρονικές στιγμές έναρξης και λήξης της ροής.
- Θύρες προορισμού και προέλευσης για TCP, UDP, SCTP.
- Αριθμός πακέτων και bytes που παρατηρήθηκαν.
- Διεύθυνση προέλευσης και προορισμού.
- Τύπος και κωδικός του ICMP.
- Το πρωτόκολλο επιπέδου 3 και το TOS (Type Of Service).

- Αν πρόκειται για TCP πακέτα, οι σημαίες οι οποίες χρησιμοποιήθηκαν.

Ακολουθεί σχηματική παρουσίαση πιθανής αρχιτεκτονικής παρακολούθησης ενός δικτύου με τη χρήση NetFlow.



Εικόνα 4 – Αρχιτεκτονική Netflow

2.3.2 Βαθύτερη επιθεώρηση πακέτου - Συστήματα Ανίχνευσης Εισβολής

Τα συστήματα ανίχνευσης εισβολής (Intrusion Detection Systems) συναντώνται σε μορφή λογισμικού, καθώς και ως αυτόνομα φυσικά μηχανήματα, και οι δύο όμως έχουν σκοπό την ανίχνευση επιθέσεων στο δίκτυο μας. Τα συστήματα αυτά χωρίζονται λοιπόν σε δύο κατηγορίες, τα δικτυακά και τα ανά-μηχάνημα. Τα πρώτα τοποθετούνται στα άκρα των δικτύων και συνήθως λειτουργούν στα επίπεδα μεταφοράς και δικτύου. Αντίθετα, τα δεύτερα τοποθετούνται σε όλα τα μηχανήματα του δικτύου και αφορούν ακόμα και το επίπεδο εφαρμογής καθώς επίσης ενδέχεται να παρακολουθούν και την τοπική λειτουργία του μηχανήματος (άνοιγμα αρχείων, αιτήσεις χρηστών για αναβάθμιση δικαιωμάτων κτλ.). Και οι δύο κατηγορίες, στην περίπτωση ανίχνευσης οποιασδήποτε κακόβουλης δραστηριότητας, ενημερώνουν τους κατάλληλους ανθρώπους, συστήματα ή υπηρεσίες. Εδώ θα πρέπει να σημειωθεί ότι η βιομηχανία χρησιμοποιεί αυτοματισμούς για την άμεση χρήση αυτής της πληροφορίας χωρίς ανθρώπινη παρέμβαση, αλλά σε κάθε περίπτωση αποτελεί μια λύση η οποία απαιτεί επιτήρηση από ανθρώπους για την επιβεβαίωση της ορθής λειτουργίας.

Όσον αφορά την υλοποίηση αυτών, έχουμε επίσης δύο βασικές κατηγορίες. Η πρώτη κατηγορία αφορά την ανίχνευση με βάση πρότυπα (signature based), τα οποία μπορεί να περιλαμβάνουν αλληλουχίες ενεργειών από χρήστες, όπως αιτήματα προς δικτυακές εφαρμογές. Η κατηγορία αυτή βασίζεται σε συνεχώς αναπτυσσόμενες βάσεις δεδομένων οι οποίες τροφοδοτούνται από τους δημιουργούς του συστήματος με σκοπό την γρήγορη εύρεση κακόβουλων προτύπων. Το μειονέκτημα αυτής της κατηγορίας είναι ότι όπως όλα τα στατικά συστήματα, η παραμικρή αλλαγή στο κακόβουλο πρότυπο που χρησιμοποιεί ο επιτιθέμενος οδηγεί στην παράκαμψη του ανανεωμένου προτύπου από το σύστημα, με αποτέλεσμα την αδυναμία αναγνώρισης του ως κακόβουλο.

Η δεύτερη κατηγορία αφορά μια πιο δυναμική και αυτό-συντηρούμενη μέθοδο, η οποία κάνει χρήση μηχανικής μάθησης και στατιστικών μεθόδων με σκοπό την καταπολέμηση των προαναφερόμενων αδυναμιών της πρώτης κατηγορίας(anomaly based). Σε αυτή την περίπτωση, όταν εμφανίζεται κακόβουλο πρότυπο το οποίο αλλάζει ελάχιστα με σκοπό την αποφυγή ανίχνευσης, το σύστημα μας θα προσαρμοστεί στην αλλαγή και θα αναγνωρίσει και την νέα έκδοση της επίθεσης. Το μειονέκτημα της κατηγορίας είναι το αυξημένο ποσοστό ψευδοθετικών συμβάντων που παράγουν τα συστήματα αυτά.

Σε αυτό το σημείο θα πρέπει να σημειωθεί η διαφορά των συστημάτων αυτών με τα τείχη προστασίας. Τα τείχη προστασίας τοποθετούνται συνήθως στα άκρα υποδικτύων, έτσι αποτελούν ένα πολύ καλό πρώτο μέτρο φύλαξης των δικτύων μας. Σκοπός αυτών είναι ο περιορισμός της κίνησης η οποία μπορεί να εισέθλει ή να εξέλθει από το δίκτυο μας. Από τα προαναφερθέντα, διαπιστώνουμε ότι πρόκειται για λύση η οποία επεμβαίνει άμεσα και επηρεάζει την δικτυακή κίνηση. Αντίθετα τα συστήματα ανίχνευσης εισβολής αποτελούν μια παθητική λύση, η οποία απλώς παράγει ειδοποιήσεις στην περίπτωση επιτυχούς ανίχνευσης. Ο λόγος ύπαρξης και των δύο είναι ότι τα τείχη μπορούν σε πολλές περιπτώσεις να παρακαμφθούν από επιτιθέμενους. Για παράδειγμα στην περίπτωση απόκτησης πλήρους ελέγχου ενός μηχανήματος στο εσωτερικό του δικτύου, τα τείχη μεταχειρίζονται την κίνηση που ενδεχομένως θα στείλει ο επιτιθέμενος, μέσω του μηχανήματος που έχει υπό την κατοχή του, ως «έμπιστη».

2.3.2.1 Snort

Το Snort αποτελεί το πρώτο εργαλείο-IDS, το οποίο με τη βοήθεια της κοινότητας του εξελίχθηκε και σε σύστημα πρόληψης εισβολής, δηλαδή ένα IPS (Intrusion Prevention System). Το πλεονέκτημα του εργαλείου αυτού σε σχέση με όλα τα υπόλοιπα είναι ότι λόγω της μεγάλης κοινότητας που διαθέτει, έχει μεγάλη υποστήριξη και ανάπτυξη, και αυτό είναι κάτι που δίνει τη σιγουριά στον χρήστη ότι το εργαλείο δεν θα πάψει να υφίσταται για ένα μεγάλο χρονικό διάστημα. Το μεγάλο του μειονέκτημα ωστόσο είναι ότι η υλοποίηση του

είναι ενός μόνο νήματος και έτσι δεν εκμεταλλεύεται τον διαρκώς αυξανόμενο αριθμό πυρήνων στα σύγχρονα συστήματα.

2.3.2.2 Suricata

Το συγκεκριμένο εργαλείο αποτελεί ολοκληρωμένη λύση διότι λειτουργεί ως IDS αλλά και IPS, αλλά και ως σύστημα παρακολούθησης ασφάλειας δικτύου ή αλλιώς NSM (Network Security Monitoring). Όπως πολλά παρόμοια IDS δικτύου, τα NIDS (Network-based IDS), βασίζονται σε μια γλώσσα με την οποία ο διαχειριστής μπορεί να ορίσει πρότυπα κακόβουλης κίνησης. Το μεγαλύτερο πλεονέκτημα και διαφορά σε σχέση με το Snort, είναι ότι ακολουθεί το μοτίβο του ανοιχτού κώδικα και επιπλέον η υλοποίηση του είναι πολυνηματική και έτσι κάνει χρήση όλων των πυρήνων του συστήματος. Τέλος, αξίζει να τονιστεί ότι το Suricata διαθέτει πολλά ακόμη χαρακτηριστικά τα οποία δεν θα καλύψουμε στο πλαίσιο αυτού του κεφαλαίου. Παρόλα αυτά θα πρέπει να ειπωθεί ότι οι κανόνες που έχουμε ορίσει στη γλώσσα του Snort λειτουργούν χωρίς προβλήματα και στο παρόν εργαλείο.

2.3.2.3 Bro

Το παρόν εργαλείο καλύπτει όλα τα εδραιωμένα χαρακτηριστικά ενός τυπικού IDS όπως τα παραπάνω αλλά διαφέρει καθότι δημιουργήθηκε ως ένα εργαλείο έρευνας και έτσι μέχρι πρότινος, γραφικό περιβάλλον, εύκολη εγκατάσταση κ.α. Σε αντίθεση όμως με όλα τα υπόλοιπα εργαλεία διαθέτει μια αρκετά περίπλοκη και ισχυρή μηχανή ορισμού πολιτικών, η οποία το διαφοροποιεί αλλά πρέπει να αναφερθεί ότι έχει διχάσει την κοινότητα των συστημάτων παρακολούθησης δικτύων. Έχει αποδειχθεί ότι το παρόν εργαλείο είναι σαφώς ταχύτερο στην επεξεργασία από όλα τα παραπάνω [1].

2.3.3 Γνωστά μειονεκτήματα

Οι δικτυακές μέθοδοι παρακολούθησης που προαναφέρθηκαν (sFlow, NetFlow), έχουν τη δυνατότητα να παρακολουθούν την διερχόμενη κίνηση και να καταγράφουν δεδομένα που αφορούν τα επίπεδα μεταφοράς (transport) και δικτύου (internet). Ως εκ τούτου, το σύνολο των επιθέσεων οι οποίες μπορούν να ανιχνευθούν με αυτές τις μεθόδους είναι περιορισμένο σε αυτά τα επίπεδα. Για παράδειγμα, μπορούμε να εντοπίσουμε επιθέσεις όπως:

a) DDoS, ειδικότερα τα ογκομετρικά είδη.
b) Port Scanning
c) SYN Flood

Όπως όμως προαναφέρθηκε, οι μέθοδοι αυτές δεν μπορούν να επεξεργαστούν πληροφορία η οποία βρίσκεται στο στρώμα του επιπέδου εφαρμογής.

Παράλληλα, λύσεις όπως τα IDS μπορούν να ανιχνεύσουν ένα μεγαλύτερο φάσμα επιθέσεων, λόγω του ότι λειτουργούν μέχρι και το στρώμα εφαρμογής. Έτσι λοιπόν, μπορούμε να ορίσουμε περίπλοκους κανόνες για την ανίχνευση δειγμάτων κακόβουλης κίνησης, και σε δεύτερο χρόνο να αναλάβουμε δράση. Και αυτή η μέθοδος όμως, όπως και οι δικτυακές μέθοδοι, έχει τους δικούς της περιορισμούς.

Ο πρώτος περιορισμός είναι το γεγονός ότι συνήθως τα IDS λειτουργούν παρακολουθώντας την κίνηση σε πραγματικό χρόνο. Έτσι λοιπόν εάν ο αριθμός των κανόνων, οι οποίοι ελέγχονται πριν προωθηθεί ένα πακέτο στον αντίστοιχο κόμβο, υπερβαίνει ένα λογικό κατώφλι, θα διαπιστωθεί μια αισθητή καθυστέρηση σε όλο το δίκτυο. Φυσικά κάποια IDS υποστηρίζουν μια «εκ των υστέρων» υπηρεσία, μέσω της οποίας το ίδιο το IDS κρατάει αρχεία καταγραφής που σε δεύτερο χρόνο εξετάζει για τυχόν επιθέσεις, αλλά αυτή η μέθοδος δεν είναι ευρέως χρησιμοποιούμενη.

Ο δεύτερος περιορισμός που παρουσιάζει αυτή η μέθοδος είναι ότι δεν μπορεί να εξετάσει την κρυπτογραφημένη κίνηση η οποία εισέρχεται ή εξέρχεται από τη διεπαφή που παρακολουθούμε. Η εξεταζόμενη κίνηση λοιπόν αφορά οποιοδήποτε πρωτόκολλο το οποίο δεν χρησιμοποιεί κρυπτογράφηση, όπως το HTTP, Telnet κ.α.

Στο σημείο αυτό θα πρέπει να αναφερθεί ότι η κοινότητα της ασφάλειας πληροφορίας έχει ήδη προειδοποιήσει επανειλημμένα ότι η χρήση του πρωτοκόλλου HTTPS θα πρέπει να χρησιμοποιείται καθολικά για όλες τις διαδικτυακές εφαρμογές, ανεξαιρέτως. Προς την επίτευξη αυτού του σκοπού η Google, τα τελευταία τρία χρόνια επιβραβεύει τη χρήση HTTPS με την απόδοση επιπλέον πόντων ως προς την κατάταξη των αποτελεσμάτων, σχετικών με την ίδια την ιστοσελίδα [10]. Επιπλέον, έχει δημιουργηθεί μια κίνηση από εταιρίες όπως Twitter, DuckDuckGo, Dropbox , Reddit, EFF και άλλες, η οποία ονομάζεται «Encrypt all the things» [11] («κρυπτογραφήστε όλα τα πράγματα»). Η κίνηση αυτή προωθεί ενεργά την καθολική χρήση του HTTPS καθώς και την ενημέρωση της κοινότητας του διαδικτύου για τα πλεονεκτήματα της κρυπτογράφησης.

Παρακάτω φαίνεται το ποσοστό των ιστοσελίδων οι οποίες χρησιμοποιούν HTTPS. Τα δεδομένα αυτά παρέχονται από το Firefox Telemetry.

Εικόνα 5 – Χρήση HTTPS

Ως αποτέλεσμα των παραπάνω, κρίνουμε ότι η χρήση των δικτυακών IDS (NIDS) θα εκλείψει καθώς η μόνη λύση για την αποκρυπτογράφηση της διερχόμενης κίνησης είναι η χρήση μεσολαβητών (proxies) και η εγκατάσταση ενδιάμεσων πιστοποιητικών, γεγονός που δεν διευκολύνει τους διαχειριστές δικτύων.

2.4 Διαχείριση και Ανάλυση Αρχείων Καταγραφής σε Σύγχρονα Δίκτυα και Εφαρμογές

Τα αρχεία καταγραφής (log files ή απλώς logs) υφίστανται έτσι ώστε να λαμβάνουμε πληροφορία για τη εσωτερική λειτουργία ενός λογισμικού, για την διερχόμενη κίνηση μιας διεπαφής μιας δικτυακής συσκευής είτε οποιαδήποτε άλλης ηλεκτρονικής συσκευής. Τα logs μπορεί να βρίσκονται διάσπαρτα σε διάφορες συσκευές. Έτσι λοιπόν, για να μπορέσουμε να παρακολουθήσουμε, να δεικτοδοτήσουμε και τέλος να αναζητήσουμε και να παράγουμε νέα πληροφορία με τη χρήση αυτών τον αρχείων, θα πρέπει να τα συγκεντρώσουμε σε ένα κεντρικό σημείο. Στη συνέχεια η διαχείριση αρχείων καταγραφής προσπαθεί να παράγει μετρικές από αυτά τα δεδομένα τα οποία έχουμε συγκεντρώσει στο κεντρικό σημείο. Έπειτα, θα πρέπει να προταθεί μια λύση για την αποθήκευση των δεδομένων με αποδοτικό τρόπο, έτσι ώστε να μπορούμε σε ένα λογικό διάστημα να αναζητήσουμε ή να επεξεργαστούμε τα δεδομένα αυτά. Επιπλέον, οι διαχειριστές των δικτύων χρησιμοποιούν κάποιου είδους λογισμικό που τους ειδοποιεί αυτόματα σε περίπτωση που διαπιστωθεί κάποια ανωμαλία στο δίκτυο ή την εφαρμογή που παρακολουθούν με τη χρήση αρχείων καταγραφής. Αυτό το

λογισμικό εντάσσεται στην κατηγορία της διαχείρισης αρχείων καταγραφής. Επιπροσθέτως, στο πλαίσιο της διαχείρισης των logs απαραίτητη είναι η διαγραφή παλαιότερων δεδομένων, καθότι η φύση της πληροφορίας είναι συνήθως παροδικής σημασίας, δηλαδή, προσφέρει αξία στον διαχειριστή για ένα μικρό χρονικό διάστημα. Φυσικά, αυτό εξαρτάται από τη φύση των δεδομένων, ή το περιβάλλον μέσα στο οποίο αυτά συλλέγονται, για παράδειγμα, μια εταιρία στο τέλος του χρόνου μπορεί να απαιτεί από τους διαχειριστές των δικτύων της να παρουσιάσουν με μετρικές την πρόοδο της διερχόμενης κίνησης, των αριθμό των επιθέσεων DDoS που δέχτηκαν στη διάρκεια του χρόνου.

Για την επίτευξη των παραπάνω στόχων, οι σουΐτες λογισμικών οι οποίες ασχολούνται με τα συγκεκριμένα ζητήματα αντιμετωπίζουν διάφορα προβλήματα στα διάφορα στάδια πορείας της πληροφορίας. Αρχικά, ανάλογα με το περιβάλλον της συλλογής της πληροφορίας, ο όγκος των δεδομένων μπορεί να αποτελέσει τεράστιο πρόβλημα, και έτσι λοιπόν οι περισσότερες εμπορικές και μη λύσεις στοχεύουν στην καλύτερη κλιμακωσιμότητα με σκοπό να προσελκύσουν μεγάλες εταιρίες και οργανισμούς.

Έπειτα, μεγάλο πρόβλημα αποτελεί ο τεράστιος αριθμός πηγών πληροφορίας, και ειδικότερα, η ανομοιογένεια της μορφής της (format). Για παράδειγμα, πολλές παλαιότερες συσκευές μπορεί να μην υποστηρίζονται από κάποιο λογισμικό, και έτσι να μην επιδέχεται αναβαθμίσεων, με αποτελέσματα τα αρχεία καταγραφής να έχουν διαφορετική μορφή. Αυτό οδηγεί στην ιδιαίτερη διαχείριση που απαιτείται για την ομογενοποίηση των δεδομένων.

Επιπλέον, λόγω του ότι διάφορες εφαρμογές απαιτούν παρακολούθηση πραγματικού χρόνου, υπάρχει ανάγκη για μεγάλη ταχύτητα του αγωγού (pipeline) μέσα από τον οποίο διέρχεται η πληροφορία, μέχρι αυτή να φτάσει στην τελική της μορφή. Επίσης αυτή την ταχύτητα την επηρεάζει και η ταχύτητα με την οποία παράγεται η πληροφορία από την πηγή. Η τελευταία μπορεί να είναι πολύ μεγάλη αν αναφερόμαστε σε εφαρμογές με εκατομμύρια χρήστες, ενώ όταν μιλάμε για παρακολούθηση μίας ΚΜΕ (CPU), μπορούμε να προσαρμόσουμε τη συχνότητα δειγματοληψίας έτσι ώστε η ταχύτητα παραγωγής να μην είναι τόσο υψηλή.

Για τους σκοπούς της παρούσας διπλωματικής, έχουμε επιλέξει να χρησιμοποιήσουμε τη σουΐτα λογισμικού Elastic. Η επιλογή αυτή έγινε μετά από αναζήτηση της κατάλληλης λύσης, και η συγκεκριμένη παρουσιάζει τα χαρακτηριστικά που αναζητούσαμε. Αρχικά, μας δίνει τη δυνατότητα αναζήτησης πλήρους κειμένου, κάτι που μας βοηθά στην ταχύτατη αναζήτηση με τη χρήση περίπλοκων ερωτημάτων. Θα πρέπει όμως να σημειωθεί ότι δεν είναι στο πλαίσιο της παρούσας διπλωματικής η σύγκριση και η εύρεση της βέλτιστης λύσης, καθώς κάτι τέτοιο θα απαιτούσε την εξερεύνηση πολλών λύσεων με γνώμονα πολλούς παράγοντες όπως η κλιμακωσιμότητα κ.α.

2.5 Σουΐτα Λογισμικού Elastic

Η σουΐτα λογισμικού Elastic (Elastic Stack) αποτελείται από τέσσερα αυτόνομα εργαλεία, τα οποία παρουσιάζονται αναλυτικά στις επόμενες υποενότητες. Η σουΐτα αυτή αποτελεί έναν ολοκληρωμένο αγωγό, μέσω του οποίου μπορούμε να διανείμουμε, να επεξεργαστούμε, να αποθηκεύσουμε, να αναζητήσουμε, και τέλος να οπτικοποιήσουμε οποιαδήποτε δομημένη πληροφορία (βλέπε ενότητα 4.2).

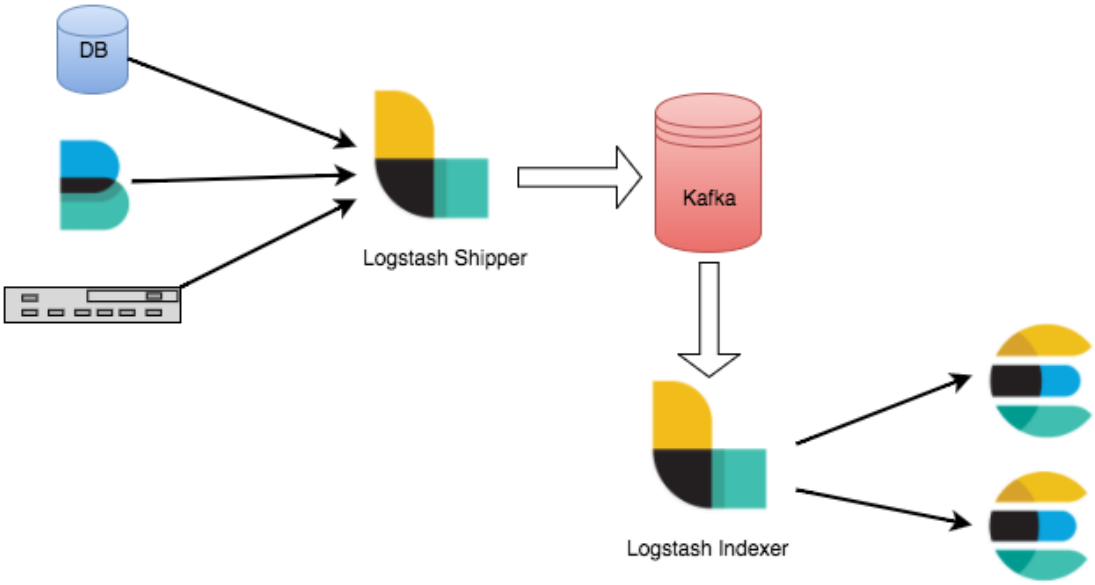
Όπως προαναφέρθηκε στην εισαγωγή, η ροή πληροφορίας ολοένα και μεγαλώνει στα σύγχρονα δίκτυα και μαζί της και η ανάγκη για αποτελεσματική και έγκαιρη επεξεργασία αυτής με σκοπό την βαθύτερη κατανόηση των δικτύων και των εφαρμογών μας. Για το σκοπό αυτό, η σουΐτα Elastic, καθώς και άλλα παρόμοια εργαλεία (MongoDB, Grafana, Statsd, Collectd, κτλ.) προσπαθούν να λύσουν τα εξής προβλήματα:

1. Απουσία συνέπειας στη διανομή της πληροφορίας, καθώς κάθε σύστημα, εφαρμογή ή συσκευή απαιτεί διαφορετική διαχείριση.
2. Απουσία κεντρικού σημείου συλλογής της πληροφορίας.
3. Δυσκολία πρόσβασης στην χρήσιμη πληροφορία, συχνά λόγω του όγκου πληροφορίας.

Οι χρήσεις της παραπάνω σουΐτας ποικίλουν ανάλογα με το είδος των δεδομένων, για παράδειγμα συναντάται στις εξής περιπτώσεις:

1. Παραγωγή επιχειρηματικής πληροφορίας.
2. Ασφάλεια δικτύων και πληροφορίας.
3. Ανάλυση δικτυακών εφαρμογών (Web Analytics).
4. Συμμόρφωση κανόνων λειτουργίας (Compliance).

Σε αυτό το σημείο θα πρέπει να σημειωθεί ότι καθένα από τα παρακάτω εργαλεία είναι αυτόνομα και μπορούν να χρησιμοποιηθούν χωρίς την ανάγκη χρήσης των υπόλοιπων.



Εικόνα 6 - Παράδειγμα χρήσης της σουΐτας Elastic

2.5.1 Ανάγνωση και προώθηση – Filebeat

Το Filebeat αποτελεί το πρώτο στάδιο του αγωγού, τον οποίο ακολουθεί η πληροφορία μέχρι την οπτικοποίηση της. Το εργαλείο αυτό είναι υπεύθυνο για την εποπτεία των αρχείων καταγραφής, την ανάγνωση τους και την προώθηση νέας πληροφορίας στο επόμενο στάδιο. Στις τελευταίες εκδόσεις του, έχει τη δυνατότητα να επικοινωνεί με το άκρο του επόμενου σταδίου και να προσαρμόζει το ρυθμό μεταγωγής της πληροφορίας, προς αποφυγή τυχόν βλάβης του αγωγού.

2.5.2 Επεξεργασία – Logstash

Το Logstash αποτελεί το δεύτερο στάδιο του αγωγού. Το παρόν εργαλείο είναι υπεύθυνο για την παραλαβή της πληροφορίας, την επεξεργασία της και τέλος την προώθηση της. Σε αυτό το στάδιο λύνουμε ένα πρόβλημα το οποίο ταλανίζει πολλά συστήματα, ειδικότερα εκείνα που τροφοδοτούνται από παγκόσμια δίκτυα. Το πρόβλημα αυτό είναι οι διαφορετικές χρονικές ζώνες από της οποίες συλλέγονται τα αρχεία καταγραφής. Το Logstash μετατρέπει την ώρα καταγραφής στη Συντονισμένη Παγκόσμια Ώρα (UTC).

Έπειτα, μας βοηθά να αναλύσουμε και να εμπλουτίσουμε την πληροφορία που μας παρέχεται. Για παράδειγμα, συχνά συναντάμε την ανάγκη να αποκρύψουμε κάποιες πληροφορίες πριν την προώθηση της σε κεντρικότερο σημείο. Αυτό μπορεί να συμβαίνει για λόγους ανωνυμίας, είτε για απόκρυψη ευαίσθητης πληροφορίας. Επιπλέον, μπορούμε να κατηγοριοποιήσουμε την πληροφορία και να την επεξεργαστούμε με τον αρμόζοντα τρόπο.

Τέλος, ανεξαρτήτως πηγής της πληροφορίας, το εργαλείο αυτό μας παρέχει τη δυνατότητα να ομογενοποιήσουμε το εξαγόμενο συμβάν (event) σε μορφή που είναι κατανοητή από τα στάδια που ακολουθούν στον αγωγό.

2.5.3 Δεικτοδότηση και Αναζήτηση - Elasticsearch

Στη συνέχεια η πληροφορία προωθείται στο Elasticsearch για αποθήκευση και περαιτέρω αναζήτηση. Το Elasticsearch αποτελεί μια μηχανή αναζήτησης η οποία υποστηρίζεται εσωτερικά από το Lucene, μια βιβλιοθήκη πάνω στην οποία έχουν δημιουργηθεί πολλές γνωστές μηχανές αναζήτησης, όπως το Apache Solr και το Elasticsearch. Επιπλέον, διαθέτει διαδικτυακή διεπαφή (web interface) μέσω HTTP για ευκολότερη επικοινωνία από οποιαδήποτε συσκευή/πελάτη, αποθηκεύει αρχεία JSON και δεν προϋποθέτει προκαθορισμένο σχήμα όπως μια SQL βάση δεδομένων, κάτι που μας δίνει ευελιξία όσον αφορά την δυναμική προσθήκη νέων πεδίων. Επιπρόσθετα, μας παρέχει τη δυνατότητα αναζήτησης πλήρους-κειμένου (full-text search) και έτσι μπορούμε να ψάχνουμε για λέξεις ή προτάσεις μέσα στα αρχεία που αποθηκεύουμε. Τέλος, ίσως ένα από τα σημαντικότερα χαρακτηριστικά του Elasticsearch είναι η κατανεμημένη αρχιτεκτονική της αποθήκευσης της πληροφορίας αλλά παράλληλα και η εξαιρετικά μεγάλη δυνατότητα για κλιμάκωση του συστήματος.

2.5.4 Οπτικοποίηση – Kibana

Τέλος, επειδή το Elasticsearch διαθέτει μόνο μια διαδικτυακή διεπαφή μέσω HTTP, για ευκολότερη χρήση και παρακολούθηση των δεδομένων που αποθηκεύουμε, σε πραγματικό χρόνο, η σουΐτα Elastic διαθέτει ένα εργαλείο που ονομάζεται Kibana. Το εργαλείο αυτό μας παρέχει ένα γραφικό περιβάλλον μέσω του οποίου μπορεί ο χρήστης να αποθηκεύσει σχήματα και μετρικές τις οποίες θέλει να παρατηρεί. Επιπλέον, διαθέτει μια κονσόλα για την εύκολη ανάπτυξη περίπλοκων αιτημάτων προς το Elasticsearch.

2.6 Ανάγκη για συνεργατική αντιμετώπιση

Όπως έχει αναφερθεί και στην εισαγωγή καθώς και στην πρώτη ενότητα του παρόντος κεφαλαίου οι μεγαλύτερες ομάδες εγκληματιών στο σύγχρονο κόσμο έχουν μεταφερθεί στο Internet, έτσι λοιπόν οι ομάδες αυτές αναζητούν εύκολους στόχους, συνήθως σκανάροντας μεγάλες περιοχές IP, είτε εξαπολύοντας επιθέσεις «ηλεκτρονικού ψαρέματος» σε μεγάλους οργανισμούς, με σκοπό να πάρουν υπό τον έλεγχο τους όσο το δυνατόν μεγαλύτερο αριθμό μηχανημάτων. Αυτά τα μηχανήματα χρησιμοποιούνται για διάφορους κακόβουλους σκοπούς, όπως η εξόρυξη κρυπτονομισμάτων, πώληση υπηρεσιών όπως DDoS μεγάλου εύρους,

κακόβουλου λογισμικού (ransomware, malware κ.α.), καθώς επίσης και στοχευμένης επίθεσης (spear phishing) μέσω ενδεχόμενης προηγούμενης παρείσφρησης στο εσωτερικό διαφόρων δικτύων.

Τα παραπάνω λοιπόν μας ωθούν στην ανταλλαγή πληροφορίας μεταξύ οντοτήτων, οι οποίες μπορεί να είναι εργαστήρια, οργανισμοί, κοινότητες είτε ακόμη και δημόσια ιδρύματα ή εταιρίες. Όπως φαίνεται και στο [4], όπου μελετήθηκε η αποτελεσματικότητα της συνεργασίας για την αντιμετώπιση επιθέσεων σε γειτονικά δίκτυα, διαπιστώθηκε ότι το ποσοστό αποτροπής επιθέσεων συνολικά σε ένα δείγμα από δίκτυα αυξάνεται αναλογικά με τον αριθμό των συμμετεχόντων στο προτεινόμενο σχήμα. Αυτό συμβαίνει διότι όπως προαναφέρθηκε οι επιτιθέμενοι εξαπολύουν επιθέσεις εναντίων πολλαπλών δικτύων και συνήθως αυτό συμβαίνει σειριακά, για σύντομο χρονικό διάστημα και με μεγάλο εύρος ζώνης. Έτσι λοιπόν, εφόσον το πρώτο δίκτυο-θύμα ενημερώσει του υπόλοιπους συμμετέχοντες μπορούν να αποτραπούν μελλοντικές επιθέσεις εναντίων τους.

Παράλληλα, η συνεργασία μεταξύ οντοτήτων και η αντίστοιχη αξία της έχει ήδη εδραιωθεί με τη χρήση κοινών λιστών αποκλεισμού IP (IP blacklists) οι οποίες είτε συντηρούνται από πολλούς συμμετέχοντες είτε πολλαπλές λίστες δημοσιεύονται σε κεντρικά σημεία και μπορούν να χρησιμοποιηθούν από όλους τους ενδιαφερόμενους [5].

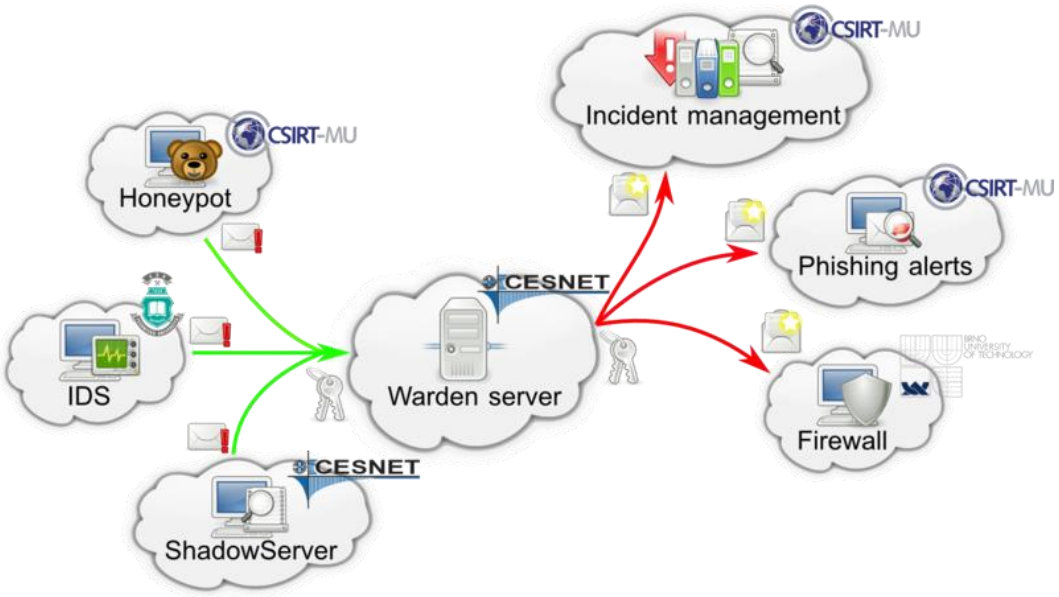
Τέλος άξιο αναφοράς είναι το γεγονός ότι στο χώρο της έρευνας γύρω από κακόβουλο λογισμικό (malware analysis) η ανταλλαγή πληροφορίας αποτελεί ακρογωνιαίο λίθο καθότι όλο και περισσότεροι επιτιθέμενοι αναπτύσσουν και διανέμουν καινούργιο κακόβουλο λογισμικό και η ερευνητική κοινότητα προσπαθεί μαζικά να διαπιστώσει εάν και άλλοι φορείς έχουν πληχθεί από την ίδια ομάδα επιτιθέμενων. Επίσης, η ανταλλαγή πληροφορίας βοηθά την κοινότητα να αναπτύσσει εργαλεία αυτοματισμού με σκοπό την κατηγοριοποίηση πιθανού κακόβουλου λογισμικού [6,7].

2.7 Σχετικές Υλοποιήσεις

2.7.1 Warden

Το Warden [8], αποτελεί την λύση την οποία προτείνει και έχει υιοθετήσει το CESNET, ο οργανισμός έρευνας και τεχνολογίας της Τσεχίας. Η παρούσα λύση προσπαθεί να προσεγγίσει το πρόβλημα της ανταλλαγής πληροφορίας μεταξύ οντοτήτων, καθώς επίσης και της οπτικοποίησης αυτών. Το Warden λοιπόν λειτουργεί ως η έμπιστη οντότητα για όλους τους συνεργαζόμενους φορείς, δεν αφορά ωστόσο τη συσχέτιση και περαιτέρω επεξεργασία της πληροφορίας που συλλέγεται.

Παρακάτω φαίνεται η αρχιτεκτονική του Warden:



Εικόνα 7 – Αρχιτεκτονική Warden

Όπως βλέπουμε λοιπόν, με πράσινο εμφανίζονται οι φορείς/υπηρεσίες που λειτουργούν ως αποστολείς (senders), και με κόκκινο αυτοί που λειτουργούν ως δέκτες (receivers). Εδώ θα πρέπει να σημειωθεί ότι ένα φορέας μπορεί να έχει διττό ρόλο στο σχήμα που παρουσιάζει το εργαλείο. Έτσι λοιπόν, έχουμε μια αρχιτεκτονική πελάτη-εξυπηρετητή, στην οποία κάθε φορέας μπορεί να λειτουργεί και με τους δύο αυτούς ρόλους.

Τα δεδομένα τα οποία μπορεί να δεχθεί το εργαλείο μπορεί να είναι πρωτογενή, όπως τα αρχεία καταγραφής για παράδειγμα ένα IDS, ένας εξυπηρετητής-honeypot κ.α., καθώς επίσης και δευτερογενή, όπως συσχετισμένα δεδομένα και μετρικές πάνω σε αυτά, είτε από έτοιμες λύσεις όπως άλλα παρόμοια εργαλεία.

Για την διανομή της πληροφορίας, την ομογενοποίηση και κατ'επέκταση την πιο εύκολη επεξεργασία αυτής, το Warden ακολουθεί το πρότυπο IDEA [9], το οποίο έχει ορίσει ο ίδιος οργανισμός. Το πλεονέκτημα αυτού είναι η εύκολη αναζήτηση της πληροφορίας, αυτό όμως προϋποθέτει ότι για κάθε νέα πηγή πληροφορίας, είναι απαραίτητο να αναπτυχθεί βιβλιοθήκη μετατροπής στο προτεινόμενο πρότυπο.

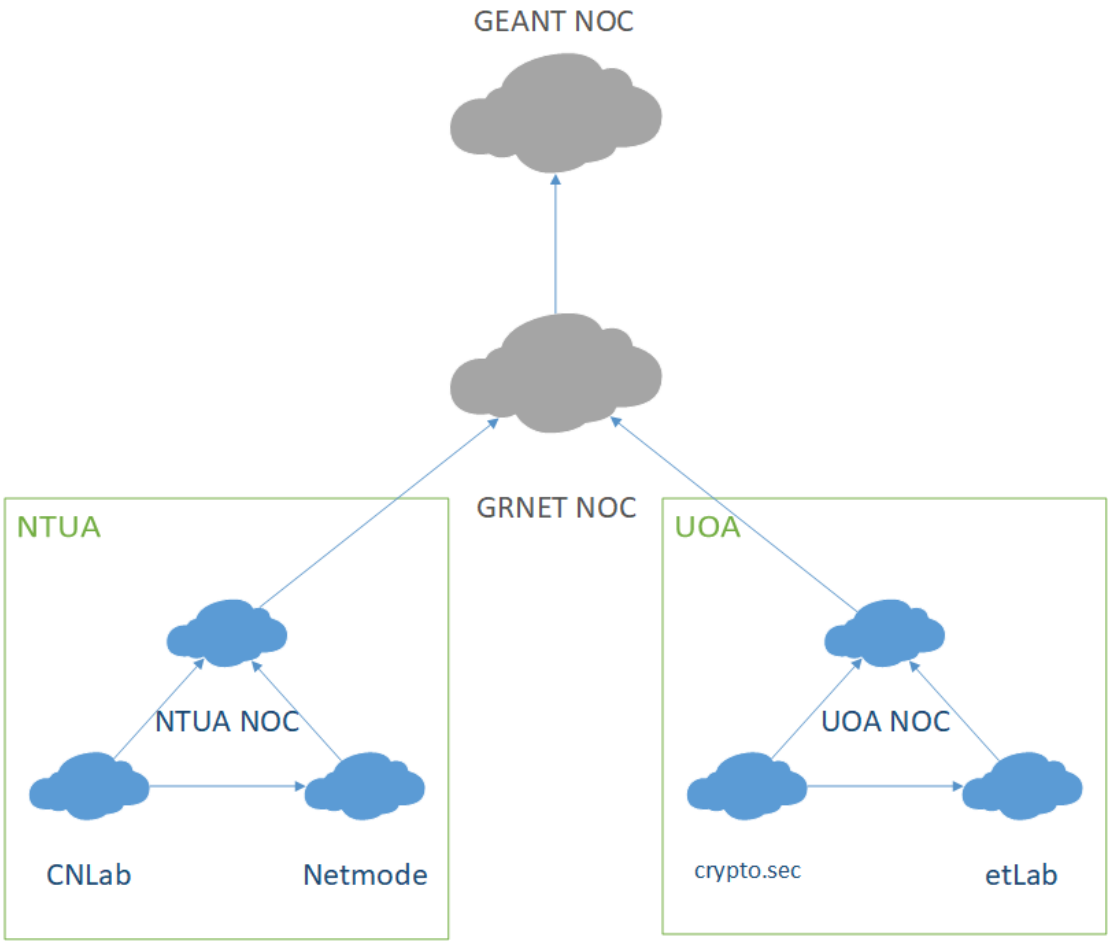
3

Σχεδιαστικές Αρχές

Σε όλα τα σύγχρονα δίκτυα παράγεται πληροφορία, η οποία αποθηκεύεται σε αρχεία καταγραφής, και ενίοτε παραμένει ανεκμετάλλευτη ή δεν αξιοποιείται στο μέγιστο δυνατό. Τα δίκτυα αυτά περιλαμβάνουν δικτυακές συσκευές όπως δρομολογητές, switches, firewalls, είτε εξυπηρετητές οι οποίοι παρέχουν υπηρεσίες όπως, SSH, δικτυακές εφαρμογές, DNS, FTP και άλλα. Όπως λοιπόν προαναφέρθηκε, τα σύγχρονα δίκτυα πλήττονται καθημερινά από επιθέσεις διαφόρων ειδών, όπως DDoS, Port Scanning, SQLi, XSS. Πολλές από αυτές λοιπόν, δεν μπορούν να ανιχνευθούν από τις περισσότερες μεθόδους παρακολούθησης δικτύων, όπως τα IDS και τα δικτυακά πρωτόκολλα παρακολούθησης, όπως sFlow/NetFlow/SNMP. Το σύνολο αυτών των επιθέσεων αφορούν συνήθως το επίπεδο εφαρμογής. Στόχος λοιπόν της παρούσας εργασίας είναι η δημιουργία ενός μηχανισμού ο οποίος θα προσπαθήσει να προσφέρει μεγαλύτερη εποπτεία των διαφόρων δικτύων/συμμετεχόντων και έναν τρόπο συλλογικής αντιμετώπισης/πρόληψης επιθέσεων.

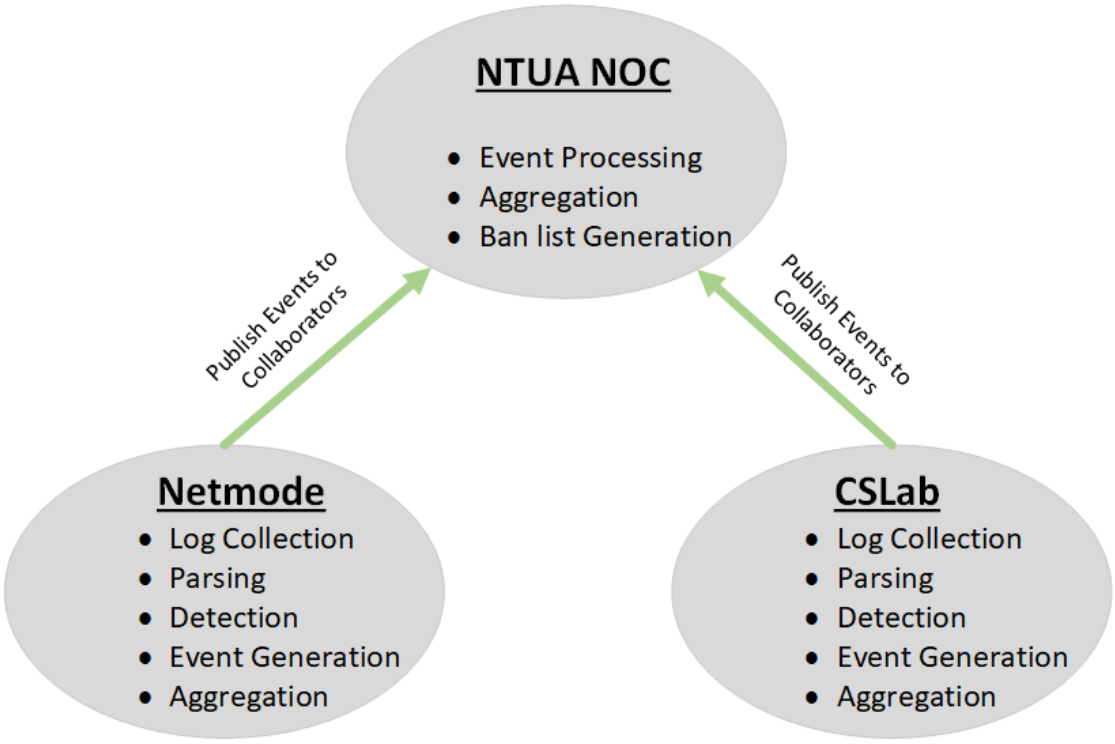
Η προτεινόμενη αρχιτεκτονική βασίζεται στην συμμετοχή διαφόρων οντοτήτων, οι οποίες ανταλλάζουν πληροφορία την οποία έχουν παράγει. Η πληροφορία αυτή μπορεί να είναι γεγονότα (events) που τα έχει παράξει είτε ένα IDS, είτε οι ανιχνευτές επιθέσεων οι οποίοι θα παρουσιαστούν στη συνέχεια της εργασίας. Με βάση την παρεχόμενη πληροφορία η κάθε οντότητα μπορεί να κρίνει τις υπόλοιπες καθώς και να κριθεί από αυτές, όσον αφορά την εγκυρότητα της πληροφορίας που διανέμουν.

Στο σημείο αυτό θα παρουσιάσουμε την υψηλού επιπέδου αρχιτεκτονική των ομότιμων οντοτήτων που μπορεί να διαρθρώνουν ένα παράδειγμα χρήσης του προτεινόμενου μηχανισμού. Για το παράδειγμα αυτό θα χρησιμοποιήσουμε τα πανεπιστήμια ως ομότιμους φορείς.



Εικόνα 8 - Παράδειγμα Περιβάλλοντος Χρήσης

Στο παρακάτω σχήμα φαίνεται η λειτουργικότητα κάθε συνδετικού κρίκου της αρχιτεκτονικής.



Εικόνα 9 – Λειτουργικότητα οντοτήτων

Όπως βλέπουμε, η κάθε συμμετέχουσα οντότητα είναι υπεύθυνη για την συλλογή και επεξεργασία των αρχείων καταγραφής, την παραγωγή δικτυακών γεγονότων και τέλος την ομαδοποίηση αυτών. Επίσης, κάθε οντότητα, κατά βούληση προωθεί τα δικτυακά γεγονότα στην τρίτη οντότητα (TTP), η οποία στο παραπάνω παράδειγμα συντηρείται από το NOC του ΕΜΠ. Επιπλέον, το TTP είναι υπεύθυνο για την συγκέντρωση των δικτυακών γεγονότων της κοινότητας, να τα ομαδοποιήσει και τέλος να παράξει λίστες αποκλεισμού για τις κακόβουλες IP διευθύνσεις που ανιχνεύθηκαν.

3.1 Συστήματα βασιζόμενα στη φήμη

Στο σημείο αυτό θα πρέπει να αναφερθούμε σε ένα σημαντικό κομμάτι της αρχιτεκτονικής, πριν προχωρήσουμε σε περισσότερες λεπτομέρειες για την εσωτερική λειτουργία του μηχανισμού. Σε ένα σχήμα μέσα στο οποίο διαμοιράζεται πληροφορία μεταξύ των διαφόρων οντοτήτων, θα πρέπει να υπάρχει κάποιος τρόπος να σιγουρευτούμε για την εγκυρότητα των γεγονότων τα οποία λαμβάνουμε. Στην παρούσα εργασία λοιπόν, η πρόταση μας είναι η χρήση της έννοια της φήμης (reputation), η οποία πιστεύουμε ότι σε βάθος χρόνου θα σταθεροποιείται σε κάθε σύστημα και έτσι θα υπάρχει μια ομαλή λειτουργία. Ο λόγος της ύπαρξης της φήμης, είναι η προσέλκυση περισσότερων οντοτήτων στο σχήμα, με την

απόδοση κινήτρων, εφόσον υπάρχει συμμετοχή και προφανώς η πληροφορία που μεταδίδεται θεωρείται από το ευρύ κοινό ως χρήσιμη.

Στο σημείο αυτό θα πρέπει να αναφερθεί ότι σε σχήματα όπως το προτεινόμενο, δεν υφίσταται απτός και αξιόπιστος τρόπος για την εξακρίβωση της εγκυρότητας της διαμοιραζόμενης πληροφορίας. Για παράδειγμα, σε κάποιο σημείο ενός σχήματος ανταλλαγής αρχείων ή αντικειμένων, μπορεί να διαπιστωθεί η ακεραιότητα και εγκυρότητα του διαφημιζόμενου προϊόντος, και έτσι να δράσουμε αναλόγως για την απόδοση αξιολόγησης του αποστολέα. Κάτι τέτοιο δε συμβαίνει στην περίπτωσή μας. Έτσι λοιπόν, στο υπόλοιπο της παρούσας ενότητας θα προτείνουμε μια λύση για την διαχείριση της φήμης των συμμετεχόντων

Η φήμη μιας οντότητας θα επηρεάζεται άμεσα από το πόσοι συμμετέχοντες θεωρούν ότι η πληροφορία που διανέμει είναι αξιόπιστη και χρήσιμη. Έτσι λοιπόν, εφόσον διαπιστωθεί για παράδειγμα, ότι περισσότεροι από έναν διαπιστώνουν ότι δέχονται επίθεση από συγκεκριμένες ομάδες (ίδια IP ή/και CIDRs), η πληροφορία αποκτά αξία για το σύνολο των συμμετεχόντων και έτσι οι αντίστοιχοι φορείς που τη διαδίδουν, θα αποκτήσουν καλύτερη φήμη.

Τα κίνητρα που μπορούν να δοθούν είναι πολλά και εξαρτώνται από το περιβάλλον χρήσης του μηχανισμού. Για παράδειγμα, έστω ότι βρισκόμαστε σε ένα περιβάλλον στο οποίο έχουμε διάφορα εργαστήρια και ένα κέντρο δικτύων από το οποίο διέρχεται η κίνηση από και προς όλα τα δίκτυα. Ένα κίνητρο μπορεί να είναι, η πιθανή βοήθεια από το κέντρο δικτύων σε περίπτωση επίθεσης εύρους ζώνης την οποία ένα εργαστήριο δεν μπορεί να αντέξει αλλά παράλληλα είναι αρκετά μικρού βελινεκούς για να το αντιληφθεί το ίδιο το κέντρο δικτύων και πιθανώς να μην επιληφθεί του θέματος.

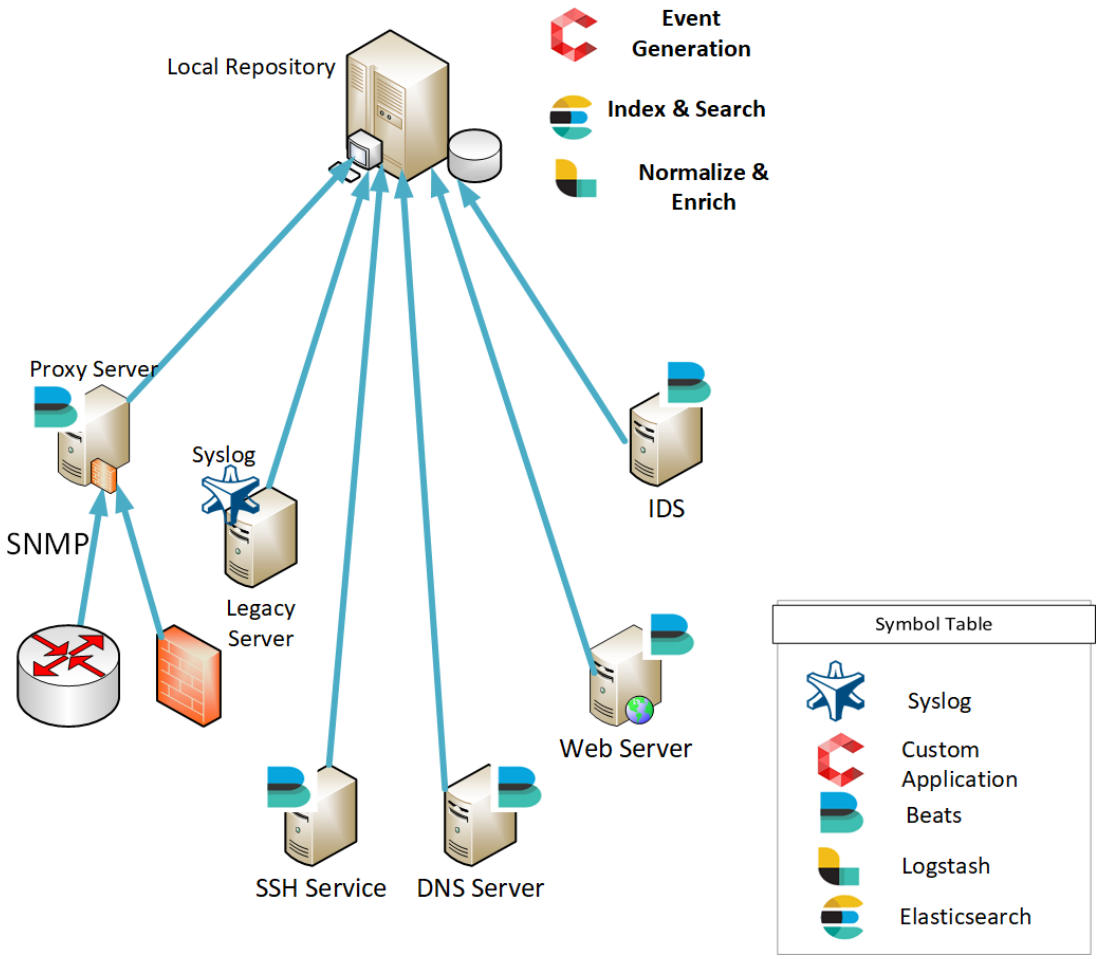
Τέλος, προτείνουμε για την ομαλή λειτουργία του μηχανισμού, μια περιοδική μείωση της φήμης κάθε συμμετέχοντα, έτσι ώστε να δίνεται ένα κίνητρο για τη συνεχή, ενεργή και καλόβουλη συμμετοχή στο μηχανισμό. Επιπλέον, για το διάστημα κατά το οποίο δεν έχει επέλθει η ομαλότητα στο μηχανισμό, όσων αφορά τη φήμη, προτείνεται να υπάρχει ανθρώπινη παρέμβαση για το χειρισμό του μηχανισμού. Εφόσον έχουμε διαπιστώσει ότι συμφωνούμε με την πληροφορία που ανακοινώνει κάποια οντότητα, μπορούμε να δημιουργήσουμε διαφόρων ειδών αυτοματισμούς για την άμεση χρήση αυτής της πληροφορίας, χωρίς την ανθρώπινη παρέμβαση.

3.2 *Ανάλυση Αρχιτεκτονικής*

Η αρχιτεκτονική του μηχανισμού αποτελείται από τρείς οντότητες και δύο νοητά επίπεδα.

Η πρώτη οντότητα ονομάζεται «τοπικός κόμβος» (local node), και περιλαμβάνει όλα τα μηχανήματα τα οποία προωθούν δεδομένα από τα αρχεία καταγραφής των υπηρεσιών που παρέχουν. Οι τοπικοί κόμβοι μπορεί να είναι δρομολογητές, ένα IDS, μεταγωγείς, εξυπηρετητές υπηρεσιών όπως μια διαδικτυακή εφαρμογή, εξυπηρετητές DNS.

Η δεύτερη οντότητα ονομάζεται «τοπικό αποθετήριο» (Local Repository), το οποίο συλλέγει την πληροφορία που στέλνουν οι διάφοροι τοπικοί κόμβοι. Οι αρμοδιότητες του τοπικού αποθετηρίου είναι η λήψη της πληροφορίας, η ομογενοποίηση αυτής, καθώς επίσης και η δεικτοδότηση/αναζήτηση. Παρατίθεται ένα παράδειγμα αρχιτεκτονικής ενός τοπικού αποθετηρίου με τους αντίστοιχους τοπικούς κόμβους. Η οντότητα αυτή μπορεί να είναι ένα πανεπιστήμιο, ένας δημόσιος οργανισμός, ένας πάροχος Internet κ.α.



Εικόνα 10 – Παράδειγμα Τοπικού Αποθετηρίου

Η τρίτη οντότητα ονομάζεται «απομακρυσμένο αποθετήριο» (Remote Repository), αποτελεί μια οντότητα εμπιστοσύνης (Trusted Third Party ή TTP) για το σχήμα που προτείνουμε. Στο σημείο αυτό συλλέγεται η πληροφορία που έχει παραχθεί από όλα τα τοπικά αποθετήρια, με σκοπό να γίνει διαθέσιμη σε όλα τα υπόλοιπα τοπικά αποθετήρια. Ο ρόλος του TTP περιλαμβάνει τη διατήρηση αποτελεί το συνδετικό κρίκο μεταξύ των συμμετεχόντων του σχήματος.

3.3 Περιγραφή Λειτουργιών

Σε αυτή την ενότητα θα παρουσιάσουμε με περισσότερη λεπτομέρεια με ποίο τρόπο η πληροφορία ξεκινά από τις διάφορες πηγές, φτάνει στα τοπικά αποθετήρια, επεξεργάζεται, προωθείται στο απομακρυσμένο αποθετήριο, όπου επεξεργάζεται εκ νέου για την παραγωγή νέας πληροφορίας και γίνεται διαθέσιμη στα διάφορα τοπικά αποθετήρια.

Σε αυτό το σημείο να σημειωθεί ότι στην ενότητα «Παράρτημα Β», παρέχουμε μια λίστα από προτεινόμενες ρυθμίσεις παραμέτρων για τα διάφορα εργαλεία που χρησιμοποιούμε.

3.3.1 Άντληση πληροφορίας στο τοπικό αποθετήριο

Για να συλλέξουμε όλα τα δεδομένα σε ένα κεντρικό σημείο, χρησιμοποιούμε το πρώτο εργαλείο της σουίτας Elastic, το Filebeat. Φυσικά, θα πρέπει να αναφερθεί ότι υπάρχουν συσκευές στις οποίες δεν υπάρχει δυνατότητα να χρησιμοποιήσουμε το συγκεκριμένο εργαλείο. Αυτό μπορεί να οφείλεται στο ότι μια συσκευή έχει περιορισμένη χωρητικότητα και έτσι τα προαπαιτούμενα για τη χρήση οποιουδήποτε εργαλείου είναι αδύνατη, για παράδειγμα, συσκευές του Internet of Things. Ένας άλλος λόγος μπορεί να είναι η περίπτωση ενός μηχανήματος με παλαιά εξαρτήματα, τα οποία δεν επιδέχονται αναβαθμίσεις και έτσι δεν υποστηρίζουν το Filebeat. Σε αυτή τη περίπτωση, θα πρέπει να χρησιμοποιήσουμε εναλλακτικές μεθόδους, όπως το πρωτόκολλο syslog, για να μεταδώσουμε την πληροφορία στο στάδιο της επεξεργασίας.

3.3.2 Επεξεργασία Πληροφορίας

Όπως προαναφέρθηκε, σε πρώτη φάση η πληροφορία συλλέγεται σε ένα κεντρικό σημείο, σε κάποιο τοπικό αποθετήριο. Εκεί χρησιμοποιούμε το δεύτερο εργαλείο της σουίτας λογισμικού, το Logstash. Το εργαλείο αυτό μας δίνει τη δυνατότητα να ομογενοποιήσουμε και να εμπλουτίσουμε τη διερχόμενη πληροφορία. Η ομοιογένεια μας εξυπηρετεί καθώς σε επόμενα στάδια θα χρειαστεί να επεξεργαστούμε τα δεδομένα αυτά, και έτσι θα έπρεπε να διαχειριστούμε κάθε ιδιαιτερότητα των διάφορων πηγών. Για παράδειγμα, όπως

διαπιστώθηκε, τα αρχεία καταγραφής της υπηρεσίας SSH για το λειτουργικό FreeBSD εμφανίζουν διαφορές με αυτά των Linux-βασιζόμενων λειτουργικών συστημάτων. Επίσης, το παρόν εργαλείο μας δίνει τη δυνατότητα να χωρίσουμε σε πεδία την πληροφορία και έτσι να βρεθεί σε μορφή συμβατή με το Elasticsearch, το τρίτο εργαλείο της σουίτας Elastic.

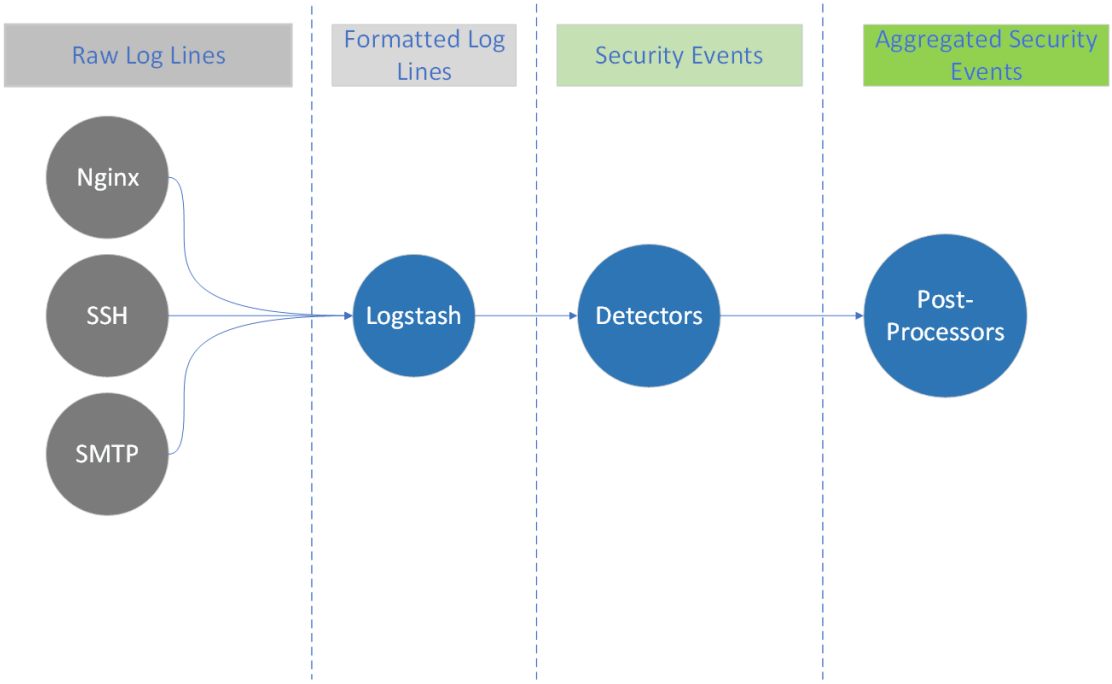
Σε δεύτερο στάδιο, η πληροφορία προωθείται στο Elasticsearch, όπου δεικτοδοτείται, τοποθετείται δηλαδή στο κατάλληλο σύνολο από γεγονότα και έτσι δίνεται ένας δείκτης για την μελλοντική του αναζήτηση από το χρήστη. Για την περαιτέρω ομογενοποίηση των δεδομένων, έχουμε δημιουργήσει ένα σύνολο από «επεξεργαστές» (processors) διαφόρων πηγών, έτσι ώστε να έχουμε τελικά μια συγκεκριμένη μορφή για όλα τα γεγονότα.

Όταν λοιπόν, ομογενοποιήσουμε όλα τα γεγονότα, με σκοπό να διαπιστώσουμε εάν υπάρχουν επιθέσεις μέσα στα διάφορα γεγονότα, χρησιμοποιούμε τους «ανιχνευτές» (detectors) που έχουμε αναπτύξει, οι οποίοι ειδικεύονται στην ανίχνευση ενός συνόλου επιθέσεων, όπως Brute force, Vulnerability Scanning, Banner Grabbing κ.α. Σε αυτό το σημείο έχουμε αναγνωρίσει όλες τις κακόβουλες IPs οι οποίες μας επιτίθενται, έτσι λοιπόν μπορούμε να περάσουμε στο επόμενο στάδιο της επεξεργασίας.

Σε αυτό το στάδιο έχουμε παράξει ένα σύνολο από κακόβουλες IP διευθύνσεις, οπότε μπορούμε να τις ομαδοποιήσουμε σε CIDRs, να διασταυρώσουμε εάν εμφανίζονται σε δημόσιες «μαύρες λίστες», οι οποίες συντηρούνται από τρίτες οντότητες, είτε να τις ομαδοποιήσουμε ανά χώρα προέλευσης.

Εφόσον ένα τοπικό αποθετήριο έχει παράξει τις λίστες με τις κακόβουλες IP διευθύνσεις, μαζί με το μέγεθος, και το είδος των επιθέσεων που δέχτηκε, μπορεί να επιλέξει να τις δημοσιεύσει, προωθώντας τες στο απομακρυσμένος αποθετήριο. Από εκεί κάθε άλλη οντότητα μπορεί να συγκρίνει τα αποτελέσματα της με αυτά που έχουν δημοσιεύσει οι υπόλοιπες οντότητες.

Στο σημείο αυτό παραθέτουμε σχηματικά πώς αλληλεπιδρούν οι διάφορες συνιστώσες του συστήματος για την επεξεργασία των γραμμών αρχείων καταγραφής σε γεγονότα και στη συνέχεια την προώθηση αυτών στο απομακρυσμένο αποθετήριο.



Εικόνα 11 – Ροή πληροφορίας

3.3.3 Προώθηση και επεξεργασία πληροφορίας από το απομακρυσμένο αποθετήριο

Όσον αφορά την επεξεργασία της παραγόμενης πληροφορίας, το απομακρυσμένο αποθετήριο, κάνει χρήση του εργαλείου Elasticsearch, καθώς χρειαζόμαστε τις ίδιες ιδιότητες, όπως και στα τοπικά αποθετήρια και έτσι δεν μπορούμε να καταφύγουμε σε μια απλούστερη λύση για την αποθήκευση, όπως μια βάση δεδομένων χωρίς σχήμα (NoSQL), για παράδειγμα MongoDB. Ζητάμε λοιπόν, από το εργαλείο που θα χρησιμοποιήσουμε τη δυνατότητα ταχείας ανάκτησης και αναζήτησης των δεδομένων.

Το απομακρυσμένο αποθετήριο αποθηκεύει λοιπόν όλα τα παραγόμενα δεδομένα που προωθεί κάθε τοπικό αποθετήριο, και έτσι ανά πάσα στιγμή ο κάθε συμμετέχων μπορεί να τα αναζητήσει. Επιπλέον, κάθε οντότητα προωθεί περιοδικά μια λίστα με τις τρέχουσες αξιολογήσεις όσων αφορά τη φήμη (reputation) κάθε άλλης οντότητας. Έτσι λοιπόν, ανά πάσα στιγμή, το απομακρυσμένο αποθετήριο διατηρεί μια εικόνα για τη «μέση φήμη» που κατέχει κάθε οντότητα ανάμεσα στην κοινότητα.

3.3.4 Μορφή εξαγόμενων events

Όπως προαναφέρθηκε στο δεύτερο κεφάλαιο, το Warden [8], εξάγει τα γεγονότα (events) στο ορισμένο πρότυπο IDEA [9]. Στην παρούσα εργασία, θα χρησιμοποιήσουμε μια πιο απλή

προσέγγιση για την μορφοποίηση των events. Παραθέτουμε ένα παράδειγμα ενός SSH fingerprinting event:

```
{
	‘event_processor’: ssh_processor,
	‘@timestamp’: ‘2017-01-08T11:45:05.754Z’,
	‘source_ip: ‘147.102.10.10’,
	‘target_host’: ‘ares.netmode.ntua.gr’,
	‘contributor’: ‘Netmode’,
	‘service’: ‘ssh’,
	‘attack_type’: ‘fingerprinting’,
	‘description’: ‘Version Fingerprinting’
}
```

Να επισημανθεί ότι υπάρχουν events τα οποία περιέχουν και επιπλέον πεδία όπως πληροφορία γύρω από τα HTTP αιτήματα, τις μεθόδους, το μέγεθος των αιτημάτων κ.α.

4

Ανάλυση Υλοποίησης

4.1 Πρότυπο JSON

Το JSON (Javascript Object Notation) αποτελεί ένα πρότυπο δεδομένων το οποίο είναι εύκολα αναγνώσιμο από τον άνθρωπο, καθώς επίσης και από τα συστήματα που επεξεργάζονται και παράγουν δεδομένα. Η κύρια χρήση του προτύπου είναι η μεταφορά δεδομένων με τη χρήση κειμένου. Το JSON είναι ένα πρότυπο ανεξάρτητο από τη γλώσσα στην οποία χρησιμοποιείται, όπως όμως προδίδει και το όνομα του, προέρχεται από τη γλώσσα Javascript. Με το πέρασμα των χρόνων όμως αναδείχθηκε η αξία του προτύπου σε σύγκριση με τα υπάρχοντα πρότυπα όπως XML, YAML κ.α. Επιπλέον, το JSON μας προσφέρει τη δυνατότητα να κωδικοποιούμε όποιο αντικείμενο θέλουμε σε μορφή συμβατή με το πρωτόκολλο και έτσι να το μεταδώσουμε. Ένα αρχείο JSON, αποτελείται από αντικείμενα τύπου key-value (κλειδί-τιμή) και πίνακες. Μέσω αυτών των δύο τύπων μπορούν να κωδικοποιηθούν και να αναπαρασταθούν αντικείμενα διαφόρων γλωσσών.

4.2 Σκελετός Ανάπτυξης Δικτυακών Εφαρμογών – Flask

Το Flask αποτελεί έναν μικρό-σκελετό (micro-framework), περιέχει δηλαδή μόνο την απαραίτητη λειτουργικότητα για την ανάπτυξη δικτυακών εφαρμογών. Ειδικότερα, δεν περιλαμβάνει πολλά χαρακτηριστικά που θα μας παρείχε ένα ολοκληρωμένο framework,

όπως επικύρωση φόρμας (form validation), αφηρημένο επίπεδο βάσης δεδομένων (database abstraction layer), μια διεπαφή δηλαδή για την επικοινωνία με τις διάφορες βάσεις δεδομένων.

Περιέχει παρόλα αυτά, όλα τα υπόλοιπα δομικά υλικά για την ανάπτυξη εφαρμογών, όπως cookies συνεδρίας (session cookies), templating system, debugging/development server κ.α. Συνήθως προτιμάται για ανάπτυξη μικροσκελών εφαρμογών, καθώς επίσης και για διεπαφές προγραμματικών εφαρμογών (API ή αλλιώς Application Programming Interface).

4.3 Ανάλυση Τοπικού Αποθετηρίου

Όπως έχει προαναφερθεί το τοπικό αποθετήριο (Local Repository) αποτελεί το άκρο ενός δικτύου, το οποίο συλλέγει και επεξεργάζεται την πληροφορία που παράγουν οι τοπικοί κόμβοι (local nodes). Επίσης, προωθεί την παραγόμενη πληροφορία, όπως banlists, ή γεγονότα (events) στο απομακρυσμένο αποθετήριο (Remote Repository). Τέλος, διαθέτει μια δικτυακή εφαρμογή μέσω της οποίας μπορεί ο διαχειριστής του δικτύου να τροποποιήσει τη φήμη (reputation) κάποιας γειτονικής οντότητας, όπως επίσης και να συγκρίνει τα αποτελέσματα που έχει παραγάγει, με αυτά ενός άλλου συμμετέχοντα.

4.3.1 Ανάλυση αρχείου run.py

Το παρόν αρχείο αποτελεί τη διεπαφή με την παρεχόμενη λειτουργικότητα μέσω γραμμής εντολών (δηλαδή, ένα Command Line Interface). Παρακάτω παρατίθενται οι παράμετροι τις οποίες μπορούμε να χρησιμοποιήσουμε και η αντίστοιχη λειτουργικότητα που παρέχουν:

a) -i (ή --indices): Ο χρήστης μπορεί να επιλέξει μια λίστα από δείκτες από τους οποίους θα παράγει νέα πληροφορία. Η παράμετρος αυτή δέχεται και πολλαπλούς δείκτες με τη χρήση του ειδικού συμβόλου *. Για παράδειγμα: netmode-filebeat-2017.06.*, δείκτης ο οποίος αντιστοιχεί σε όλους τους δείκτες του Ιουνίου 2017.

b) -r (ή --debug_off): Εφόσον ο χρήστης επιλέξει αυτή τη παράμετρο, κάθε ενέργεια προς εκτελέση θα έχει επιρροή στην κατάσταση του συστήματος, δηλαδή θα αποθηκευτούν δεδομένα στο τοπικό αποθετήριο και ίσως προωθηθούν και στο απομακρυσμένο αποθετήριο.

c) -s (ή service): Ο χρήστης μπορεί να επιλέξει την υπηρεσία την οποία θέλει να επεξεργαστεί (λ.χ. ssh).

d) -a (ή --agg): Ο χρήστης επιλέγει εάν θέλει να επεξεργαστεί την πληροφορία των αρχείων καταγραφής ή τα ήδη παραχθέντα γεγονότα.

   Οι παρακάτω παράμετροι αφορούν την επιλογή επεξεργασίας δικτυακών γεγονότων:

I. -lt/-gt: Ο χρήστης μπορεί να ορίσει το χρονικό διάστημα μέσα στο οποίο βρίσκονται τα γεγονότα ενδιαφέροντος.

II. --extra: Ο χρήστης μπορεί να επιλέξει να εμπλουτιστεί την παραγόμενη πληροφορία με επιπλέον υπηρεσίες (π.χ. ομαδοποίηση κακόβουλων διευθύνσεων σε CIDRs)

III. --mask: Ο χρήστης επιλέγει την μάσκα υποδικτύου για την ομαδοποίηση σε CIDRs (π.χ. 16, για την ομαδοποίηση των διευθύνσεων σε /16 δίκτυα.)

4.3.2 Ανάλυση αρχείου config.py

Στο αρχείο αυτό ορίζουμε τις απαραίτητες παραμέτρους που θα χρειαστούμε σε όλη την εφαρμογή και θέλουμε να κρατήσουμε σε ένα κεντρικό σημείο με σκοπό τον εύκολο επαναπροσδιορισμό τους.

Η λίστα με τις παραμέτρους φαίνεται παρακάτω:

- DISTRICT: Το όνομα της παρούσας οντότητας. Ορίζεται μέσω της αντίστοιχης παραμέτρου περιβάλλοντος.
- service_list: Λίστα με τις επιτρεπτές υπηρεσίες, για τις οποίες διαθέτουμε υποστήριξη.
- ssh_list: Λίστα με τους τοπικούς κόμβους οι οποίοι επιτρέπουν την πιστοποίηση ταυτότητας μέσω κωδικών πρόσβασης.
- extra: Λίστα με τις επιπλέον επιτρεπτές επεκτάσεις.
- config: Περιέχει τρείς ενότητες:
    - general: Γενικές πληροφορίες
        - ttp_ip/ttp_port: Διεύθυνση IP και πόρτα στην οποία ακούει η υπηρεσία Elasticsearch του TTP.
        - batch_size: το μέγεθος της παρτίδας που θα χρησιμοποιήσουμε για την επεξεργασία των γραμμών.
        - todays_index: Ο ημερήσιος δείκτης που χρησιμοποιούμε για να ανακτήσουμε τις γραμμές αρχείων καταγραφής.
    - ssh: Πληροφορίες σχετικά με την υπηρεσία SSH
        - blacklist_url: Ορίζουμε το URL στο οποίο βρίσκεται η IP blacklist με την οποία θα διασταυρώσουμε τα αποτελέσματά μας.
    - services_to_doctypes: Αντιστοιχία υπηρεσιών και ονομάτων τύπου αρχείων στο Elasticsearch.

4.3.3 Ανάλυση αρχείου detectors.py

Το παρόν αρχείο περιέχει όλους τους ανιχνευτές, οι οποίοι είναι υπεύθυνοι να αιτηθούν όλα τα καινούργια events και να τα προωθήσουν στους «επεξεργαστές» (processors). Οι επεξεργαστές με τη σειρά τους, θα ερευνήσουν εάν υπάρχει πιθανή επίθεση με βάση είτε το συγκεκριμένο event είτε ένα σύνολο από events. Διατηρούμε τη σειρά των events, έτσι ώστε εάν για παράδειγμα αναζητούμε ένα port scan, το οποίο χαρακτηρίζεται από συγκεκριμένες ακολουθίες από events, θα πρέπει να ξεφύγουμε από την ανάλυση ενός και μόνο event.

Όλες οι κλάσεις σε αυτό το αρχείο κληρονομούν από τη βασική κλάση BaseDetector. Κάθε κλάση λοιπόν πρέπει να υλοποιεί τρείς βασικές μεθόδους:

- build_query(self, *args): αφηρημένη μέθοδος η οποία χρησιμοποιείται με όποιες εισόδους χρειάζεται ο αντίστοιχος ανιχνευτής για την αναζήτηση των γραμμών αρχείου καταγραφής. Η παρούσα μέθοδος επιστρέφει ένα αντικείμενο τύπου elasticsearch_dsl.Search, το οποίο θα χρησιμοποιήσουμε αμέσως μετά για να εκτελέσουμε το ερώτημα που αντιστοιχεί στο αντικείμενο.
- execute_query(self, gt): αφηρημένη μέθοδος η οποία χρησιμοποιείται με μόνη είσοδο το κάτω άκρο του χρονικού διαστήματος, μέσα στο οποίο ανιχνεύουμε επιθέσεις. Για παράδειγμα, εάν ορίσουμε το gt ίσο με 1M τότε το χρονικό διάστημα μέσα στο οποίο θα ενεργήσουμε, θα είναι το [now-1M, now], δηλαδή τον τελευταίο μήνα. Περισσότερες πληροφορίες όσον αφορά τις επιτρεπτές τιμές για τα χρονικά διαστήματα, βρίσκονται στο [12]. Η μέθοδος επιστρέφει ένα generator μέσω του οποίου μπορούμε να ανακτήσουμε όλα τα γεγονότα που αντιστοιχούν στο ορισμένο χρονικό διάστημα.
- process_response(self, response): Μέθοδος η οποία δέχεται τον generator της προηγούμενης μεθόδου και προωθεί σε κάθε εγγεγραμμένο processor τις αντίστοιχες γραμμές αρχείων καταγραφής προς επεξεργασία. Η παρούσα μέθοδος είναι όμοια για όλες τις κλάσεις αλλά προφανώς μπορεί να υλοποιηθεί από την υποκλάση της BaseDetector. Αρχικά, ελέγχει εάν υπάρχουν εγγεγραμμένοι processors για τη συγκεκριμένη κλάση, έπειτα για κάθε παρτίδα γραμμών, προωθούνται οι αντίστοιχες γραμμές σε όλους τους processors. Στο τέλος λοιπόν, έχουμε έναν generator με όλες τις ανιχνευμένες επιθέσεις, τον οποίο χρησιμοποιούμε για να τις εμφανίσουμε ή για να τις προωθήσουμε πίσω στο Elasticsearch, ανάλογα με το αν βρισκόμαστε σε κατάσταση αποσφαλμάτωσης.

Στο αρχείο αυτό περιέχονται οι εξής κλάσεις οι οποίες αντιστοιχούν στις διάφορες υπηρεσίες:

- SSHDetector
- DovecotDetector

- WebDetector

4.3.4 Ανάλυση αρχείου utils.py

Στο συγκεκριμένο αρχείο περιέχονται κάποιες βασικές λειτουργίες, τις οποίες χρειαζόμαστε στην εφαρμογή:

- register_proc(*classes): Ένας decorator, ο οποίος μας δίνει τη δυνατότητα να αναρτήσουμε δυναμικά σε μια κλάση το όνομα της συνάρτησης η οποία τον καλεί. Χρησιμοποιείται για την προσθήκη των ενδιαφερόμενων processors στους διάφορους detectors.
- whoami(): Επιστρέφει το όνομα της καλούσας συνάρτησης

4.3.5 Ανάλυση αρχείου processors.py

Το παρόν αρχείο περιέχει όλους τους επεξεργαστές (processors) γραμμών, οι οποίες προέρχονται από τα αρχεία καταγραφής που έχουμε συλλέξει. Με τη βοήθεια του decorator, register_proc μπορούμε να εγγράψουμε έναν processor σε πάνω από έναν detector. Με αυτόν τον τρόπο, ένας processor μπορεί να αφορά πολλαπλές υπηρεσίες. Για παράδειγμα, μπορούμε να δημιουργήσουμε έναν processor ο οποίος ασχολείται με τις επιθέσεις brute force σε διάφορες υπηρεσίες. Ο σκοπός των processors είναι λοιπόν η μετατροπή ανεπεξέργαστων γραμμών αρχείων καταγραφής σε γεγονότα (events).

Στο αρχείο λοιπόν περιέχονται οι εξής processors:

- ssh_processor(batch): Δέχεται μια λίστα με την τρέχουσα παρτίδα την οποία θα εξετάσουμε. Το συγκεκριμένο processor τον χρησιμοποιούμε για την ανίχνευση brute force επιθέσεων στην υπηρεσία SSH. Αρχικά, ο processor διατρέχει την παρτίδα, και κοιτά να κατατάξει την κάθε γραμμή σε μια κατηγορία. Οι κατηγορίες είναι:
    - SSHBANNERGRAB: Απόπειρα αναγνώρισης έκδοσης του SSH agent.
    - SSHFAILEDLOGIN/SSHFAILEDLOGINF: Απόπειρα εύρεσης έγκυρων διαπιστευτηρίων.
    - SSHINVALIDUSER/SSHINVALIDUSERF/SSHINVALIDUSERPKI: Απόπειρα εύρεσης έγκυρου χρήστη.
    - SSHCONNCLOSED/SSHCONNCLOSEDPKI: Αποτυχία εισόδου με όλες τις αποδεκτές μεθόδους.

  Επιστρέφει έναν generator με όλα τα events στη μορφή που περιγράφεται στην ενότητα 3.3.4.

- mail_processor: Δέχεται επίσης μια παρτίδα από γραμμές. Θα πρέπει να τονιστεί ότι για να αποφύγουμε τα πολλά false positives από καλόβουλες κινήσεις, επικεντρωθήκαμε στην κίνηση της υπηρεσίας ηλεκτρονικού ταχυδρομείου Dovecot, εξαιρώντας τα αιτήματα προς υπαρκτούς χρήστες. Έτσι λοιπόν ο processor, παράγει ένα event για κάθε γραμμή παρέχοντας το domain και το όνομα χρήστη στο οποίο έγινε η απόπειρα για είσοδο με λανθασμένα διαπιστευτήρια. Όπως και ο ssh_processor επιστρέφει ένα generator ο οποίος παράγει όλα τα events για να επαναδεικτοδοτηθούν από το Elasticsearch.
- non_wp_website: Ομοίως με όλους τους υπόλοιπους processors δέχεται μια παρτίδα από γραμμές μεγέθους ίσης με την ορισμένη παράμετρο batch_size του αρχείου config.py. Ο παρών processor διατηρεί μια λίστα από endpoints (άκρα) τα οποία αναζητούν οι επιτιθέμενοι όταν ανιχνεύουν περιοχές IP για ευάλωτες υπηρεσίες σχετικές με το framework Wordpress. Η λίστα αυτή μετά από μια μικρή έρευνα περιέχει τα εξής endpoints:
    - wp-login
    - wp-content
    - wordpress
    - xmlrpc.php

  Έτσι λοιπόν, διατρέχοντας τα πεδία των διάφορων αιτημάτων (requests) αναζητούμε τα στοιχεία της παραπάνω λίστας.
- malformed_requests: Με αυτόν τον processor προσπαθούμε να ανιχνεύσουμε requests προς εξυπηρετητές, τα οποία στοχεύουν και πάλι στην αναζήτηση γνωστών ευάλωτων σημείων σε μια γκάμα από υπηρεσίες. Αυτές μπορεί να είναι, ένας ιδιαίτερος χειρισμός μιας επικεφαλίδας (header),ή μιας μεθόδου (method) από κάποιο framework ή από δικό μας κομμάτι κώδικα. Επίσης, μπορεί να στοχεύει στην παράκαμψη ενός IDS μέσω της τροποποίησης παραμέτρων κ.α. Αυτές οι επιθέσεις μπορεί να οδηγήσουν σε άρνηση παροχής υπηρεσίας (DoS) μέσω σφάλματος στη διαχείριση τέτοιων περιπτώσεων είτε ακόμη και στον πλήρη απομακρυσμένο έλεγχο του εξυπηρετητή. Για αυτό το σκοπό χρησιμοποιούμε ένα tag MALFORMED_REQUEST στο στάδιο της πρώτης επεξεργασίας του Logstash.
- non_php_website: Ομοίως με το παραπάνω, αναζητούμε requests τα οποία δεν περιλαμβάνονται στη λίστα του non_wp_website ούτε στα malformed_requests.

4.3.6 Ανάλυση αρχείου post_processors.py

Στο παρόν αρχείο περιλαμβάνονται όλες οι ενέργειες οι οποίες μπορούν να εκτελεστούν αφότου έχουμε παράξει και δεικτοδοτήσει τα events των διαφόρων processors.

Οι συναρτήσεις που περιέχονται είναι:

- send_to_ttp(district, today, ttp_ip, ttp_port, source, doc_type): Δέχεται τις εξής παραμέτρους:
    - district: Η οντότητα στην οποία βρισκόμαστε (π.χ. netmode)
    - today: Ο ημερήσιος δείκτης (π.χ. netmode-events-2017.05.05)
    - ttp_ip/ttp_port: Διεύθυνση IP και πόρτα
    - source: Ένα JSON αρχείο το οποίο θα προωθήσουμε
    - doc_type: Ο τύπος του αρχείου για το Elasticsearch (π.χ. sshd)

  Προωθεί το αρχείο της παραμέτρου «source» στο TTP.
- get_attackers(response, blacklist, extra, mask): Δέχεται τις εξής παραμέτρους:
    - response: Η απάντηση του αιτήματος προς το Elasticsearch, το οποίο περιέχει όλα τα events για το επιλεγμένο χρονικό διάστημα.
    - blacklist: Μαύρη λίστα με την οποία θα διασταυρώσουμε τα αποτελέσματά μας.
    - extra: Μια λίστα από τα επιπλέον χαρακτηριστικά τα οποία ζήτησε ο χρήστης
    - mask: Ένας ακέραιος που περιέχει την μάσκα υποδικτύου για την σύμπτυξη των αποτελεσμάτων σε υποδίκτυα.

  Οι δομές που χρησιμοποιούνται είναι οι εξής:

    - attacker_dict: περιέχει όλες τις επιτιθέμενες διευθύνσεις IP μαζί με το συνολικό μέγεθος της επίθεσης που έχει εξαπολύσει.
    - cidrs: περιέχει όλες τις IP διευθύνσεις ομαδοποιημένες σε υποδίκτυα με μάσκα υποδικτύου ίση με την παράμετρο «mask».

  Η συνάρτηση αυτή επιστρέφει ομαδοποιημένα τα αποτελέσματα, ανά κακόβουλη IP και εάν αυτό ζητηθεί, και σε CIDR με ορισμένη μάσκα υποδικτύου. Έτσι λοιπόν επιστρέφει μία ή ένα tuple από δύο λίστες ανάλογα με την επιλογή του χρήστη.
- get_victims(response, blacklist, extra, mask): Δέχεται τις ίδιες εισόδους όπως παραπάνω. Ομοίως επιστρέφει μια λίστα ή ένα tuple από δύο λίστες με τα ίδια περιεχόμενα, με τη διαφορά ότι η ομαδοποίηση γίνεται ανά IP διεύθυνση-θύμα.

- construct_blacklist(): Χρησιμοποιεί την παράμετρο «blacklist_url» του config.py για να ανακτήσει τη blacklist που έχουμε επιλέξει, και έπειτα την μορφοποιεί σε ένα Python Set το οποίο και επιστρέφει.
- ips_to_cidrs(attackers, mask): Δέχεται τις εξής παραμέτρους:
    - attackers: Μια λίστα με τις κακόβουλες IP διευθύνσεις
    - Τη μάσκα υποδικτύου την οποία θα χρησιμοποιήσουμε για την μετατροπή σε CIDRs.

  Μετατρέπει τη λίστα attackers σε μια λίστα από dictionaries, η οποία περιέχει τα CIDRs μαζί με μια λίστα με τους επιτιθέμενους, το μέγεθος της επίθεσης καθώς και το ποσοστό του μεγέθους της επίθεσης επί του μεγέθους του συνόλου των επιθέσεων.
- aggregator(service, indices, aggr_type, lt, gt, extra, mask): Οι παράμετροι εισόδου έχουν ως εξής:
    - service: Η υπηρεσία της οποίας θα ομαδοποιήσουμε τα γεγονότα.
    - indices: Οι δείκτες τους οποίους θα χρησιμοποιήσουμε.
    - aggr_type: Ο τύπος της ομαδοποίησης.
    - lt/gt: Ορίζουν το άνω και κάτω φράγμα του χρονικού διαστήματος, μέσα στο οποίο θα ομαδοποιήσουμε τα γεγονότα
    - extra: Επιπλέον χαρακτηριστικά τα οποία θα προσθέσουμε στο τελικό document που θα παράξουμε.
    - mask: Η μάσκα υποδικτύου, στην περίπτωση που επιλεχθεί η ομαδοποίηση σε CIDRs.

  Οι δομές δεδομένων που χρησιμοποιούνται είναι οι εξής:
    - source: Ένα dictionary το οποίο περιέχει το τελικό document που θα παράξουμε.
    - general_cfg: Dictionary το οποίο περιέχει τις απαραίτητες παραμέτρους από το αρχείο config.py.

  Η συνάρτηση ομαδοποιεί τα γεγονότα ανά κακόβουλη IP διεύθυνση, ή ανά IP-θύμα, έπειτα εμπλουτίζει το τελικό αρχείο JSON με τα κατάλληλα χαρακτηριστικά και τέλος τα προωθεί στο TTP, εάν αυτό επιλεχθεί.

4.4 Ανάλυση Δικτυακής Εφαρμογής Διαχείρισης Πληροφορίας

Η δικτυακή αυτή εφαρμογή παριστάνει το διαχειριστικό μέσο για κάθε τοπικό αποθετήριο στο οποίο μπορούμε να παρατηρήσουμε την τρέχουσα φήμη για κάθε οντότητα, να την επεξεργαστούμε, να περιεργαστούμε τα τρέχοντα ομαδοποιημένα αποτελέσματα της παρούσας οντότητας, να τα συγκρίνουμε με αυτά μιας άλλης συμμετέχουσας οντότητας και τέλος να δούμε το ενιαίο document με τα συνολικά αποτελέσματα τα οποία έχει παράξει το TTP.

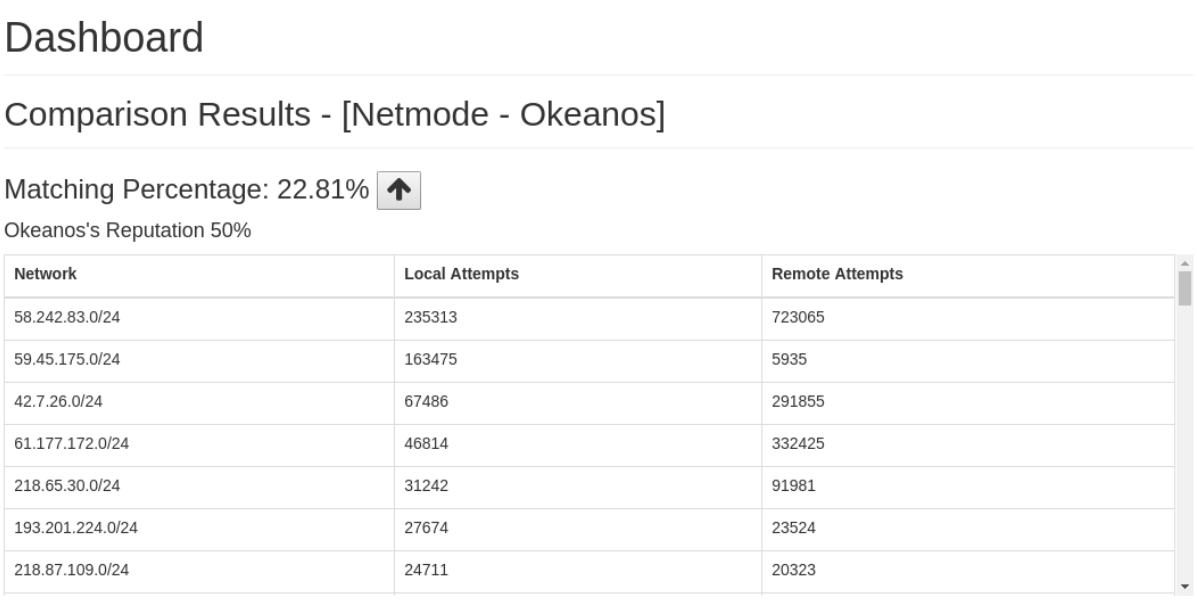
4.4.1 Ανάλυση αρχείου views.py

Στο παρόν αρχείο περιγράφονται όλα τα endpoints (άκρα) τα οποία παρέχει η δικτυακή εφαρμογή. Πριν παρουσιαστούν όμως τα άκρα, θα αναλύσουμε τις δύο βοηθητικές συναρτήσεις:

- push_to_ttp(config, district): Δέχεται δύο παραμέτρους:
    - config: Ένα σύνολο από παραμέτρους τις οποίες χρειαζόμαστε.
    - district: Το όνομα της οντότητας της οποίας θα τροποποιήσουμε τη φήμη.

  Τροποποιεί τη φήμη μίας οντότητας και την προωθεί στο TTP.
- ldict_to_ddict(aggregations): Μετατρέπει μια λίστα από dictionaries σε ένα dictionary από dictionaries.
- index(): Μας δίνει τη δυνατότητα να δούμε και να επεξεργαστούμε τη φήμη όλων των συμμετεχόντων. Δέχεται δύο μεθόδους:
    - GET: δίνει μια λίστα με τη φήμη όλων των οντοτήτων
    - POST: τροποποιεί τη φήμη μίας οντότητας και μας παρουσιάζει και πάλι ο,τι και η μέθοδος GET.
- decay_reputation(): Μειώνει τη φήμη όλων των οντοτήτων. Χρησιμοποιούμε αυτό το άκρο σε συνδυασμό με ένα cronjob και το καλούμε περιοδικά για να δίνεται κίνητρο στους συμμετέχοντες για συνεχή καλόβουλη και ενεργή χρήση του μηχανισμού.
- ttp_all_list(): Ανακτά τη λίστα με τα ομαδοποιημένα αποτελέσματα όλων των οντοτήτων σε ένα ενιαίο document, και μας το παρουσιάζει.
- ttp_feed(): Παρουσιάζει τα τελευταία ομαδοποιημένα αποτελέσματα της παρούσας οντότητας ανά κακόβουλη IP. Εάν επιλεχθεί μια άλλη συμμετέχουσα οντότητα, τότε συγκρίνονται τα ομαδοποιημένα αποτελέσματα των δύο οντοτήτων και

παρουσιάζονται τα αποτελέσματα. Στην παρακάτω εικόνα παρατίθεται ένα παράδειγμα σύγκρισης αποτελεσμάτων.



Εικόνα 12 – Σύγκριση αποτελεσμάτων δύο οντοτήτων.

4.4.2 Ανάλυση αρχείου models.py

Το συγκεκριμένο αρχείο περιέχει τα μοντέλα τα οποία αντιπροσωπεύουν τους πίνακες της βάσης δεδομένων.

Η μόνη κλάση-μοντέλο που βρίσκεται στο αρχείο είναι η κλάση "*Reputation"* και περιέχει τα εξής πεδία:

- name: Ένα string το οποίο περιέχει το όνομα μιας συμμετέχουσας οντότητας.
- rep: Ένας ακέραιος ο οποίος περιγράφει την τρέχουσα «φήμη» που διατηρεί κάθε οντότητα για όλες τις άλλες οντότητες
- date_modified/date_created: Ημερομηνίες τελευταίας αλλαγής και δημιουργίας της καταγραφής.

4.5 Ανάλυση Απομακρυσμένου Αποθετηρίου

Όπως έχει προαναφερθεί, το Απομακρυσμένο αποθετήριο (Trusted Third Party ή TTP) έχει τις εξής αρμοδιότητες:

- αποθηκεύει τα τελευταία αποτελέσματα που έχει διαλέξει να προωθήσει κάθε οντότητα μέσω του τοπικού αποθετηρίου.
- διατηρεί την τρέχουσα φήμη όλων των οντοτήτων προς όλες τις οντότητες.

- προσφέρει ομαδοποιημένα δεδομένα για την παραγωγή των οποίων χρησιμοποιεί τα συλλεγόμενα δεδομένα από όλες τις συμμετέχουσες οντότητες.

4.5.1 Ανάλυση αρχείου aggregate.py

Το συγκεκριμένο αρχείο μαζί με την υπηρεσία Elasticsearch, επιτυγχάνει τους παραπάνω ορισμένους στόχους.

Οι συναρτήσεις που παρουσιάζονται είναι οι εξής:

- ips_to_cidrs(attackers, mask): Η συγκεκριμένη συνάρτηση είναι παρόμοια με την ομώνυμη συνάρτηση του αρχείου post_processors.py του τοπικού αποθετηρίου.
- calculate_avg_reps(es_client, rep_index, district_list): Οι εξής παράμετροι δίνονται στην παρούσα συνάρτηση:
    - es_client: Ένα αντικείμενο για την επικοινωνία με το Elasticsearch.
    - rep_index: Το όνομα του δείκτη στον οποίο αποθηκεύουμε τη φήμη για όλες τις οντότητες.
    - district_list: Μια λίστα με τις συμμετέχουσες οντότητες.

  Η συγκεκριμένη συνάρτηση υπολογίζει τη μέση φήμη της κάθε οντότητας έτσι ώστε να είναι διαθέσιμη για κάθε συμμετέχουσα οντότητα, καθώς επίσης και για να την χρησιμοποιήσουμε για τη δημιουργία μιας ενιαίας λίστας με τα ομαδοποιημένα δεδομένα.
- all_aggregation(mask): Δέχεται σαν είσοδο μόνο τη μάσκα υποδικτύου για την ομαδοποίηση σε CIDRs.

  Η συγκεκριμένη συνάρτηση βρίσκει τη λίστα με τους συμμετέχοντες του μηχανισμού και στη συνέχεια υπολογίζει τη μέση «φήμη» κάθε οντότητας. Έπειτα, αναζητά τα τρέχοντα αποτελέσματα για κάθε οντότητα και τέλος τα ομαδοποιεί σε ένα ενιαίο document το οποίο προωθεί πίσω στο Elasticsearch.

5

Αξιολόγηση Υλοποίησης

Σε αυτό το κεφάλαιο θα αναλύσουμε την πειραματική διάταξη που χρησιμοποιήσαμε για να αξιολογήσουμε την υλοποίηση της προτεινόμενης αρχιτεκτονικής. Παράλληλα, θα παρουσιάσουμε τα εξαγόμενα δεδομένα που λάβαμε κατά τη διαδικασία της αξιολόγησης.

5.1 Πειραματική Διάταξη

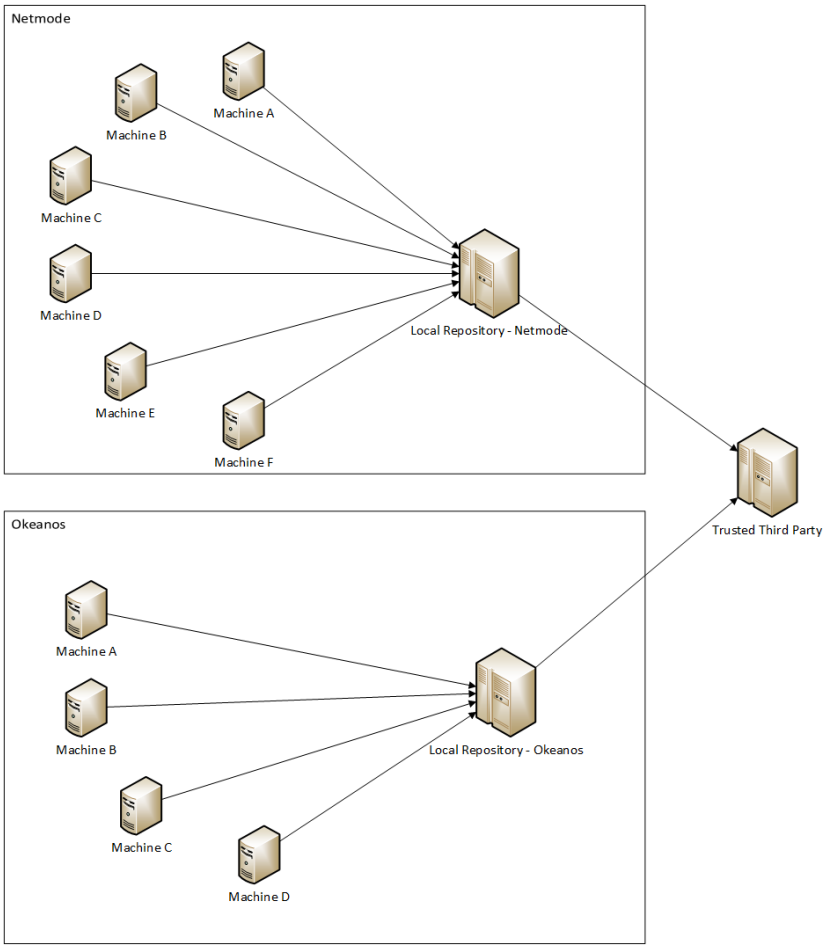
Όπως έχει ήδη προαναφερθεί στο 3ο και 4ο κεφάλαιο, για τη συλλογή των αρχείων καταγραφής (log files) χρησιμοποιούμε τη σουίτα λογισμικού Elastic. Πιο συγκεκριμένα έχουμε εγκαταστήσει το πρώτο εργαλείο της Elastic, το Filebeat, έτσι ώστε να συλλέγουμε δεδομένα από τα διάφορα μηχανήματα σε ένα κεντρικό σημείο. Στη συνέχεια, χρησιμοποιούμε το δεύτερο εργαλείο, το Logstash, για μια πρώτη επεξεργασία, η οποία θα μας βοηθήσει να κάνουμε οποιαδήποτε περαιτέρω αποθήκευση και επεξεργασία. Τέλος, προωθούμε τα δεδομένα στο Elasticsearch. Για περισσότερες πληροφορίες σχετικά με τη ροή της πληροφορίας βρίσκονται στα κεφάλαια 3 και 4. Επίσης, για τις πλήρεις ρυθμίσεις των μηχανημάτων και των αντίστοιχων εργαλείων, πληροφορίες παρέχονται στο παράρτημα Β.

Για την παρούσα διάταξη χρησιμοποιήσαμε δύο συμμετέχουσες οντότητες. Η πρώτη αποτελείται από 4 μηχανήματα και η δεύτερη από 6 μηχανήματα. Τα μηχανήματα χρησιμοποιούν μια πληθώρα από λειτουργικά συστήματα, καθώς σκοπός μας ήταν να εξετάσουμε τη λειτουργία του μηχανισμού σε πραγματικά δίκτυα, τα οποία είναι συνήθως ετερογενή. Η πρώτη οντότητα βρίσκεται (φυσικά) στην υπηρεσία «Ωκεανός», η οποία

παρέχεται από τον δημόσιο οργανισμό ΕΔΕΤ [13] . Ο «Ωκεανός» είναι μια «υποδομή ως υπηρεσία» (Infrastructure as a Service ή IaaS), η οποία μας παρέχει μηχανήματα για τη διεκπεραίωση πειραμάτων. Τα μηχανήματα της 2^{ης} οντότητας παρέχονται από το εργαστήριο με το οποίο συνεργαζόμαστε (Netmode, ECE NTUA). Στο σημείο αυτό θα πρέπει να τονιστεί η σημασία της λογικής και φυσικής απόστασης των δύο οντοτήτων, για την ανάδειξη των αποτελεσμάτων τα οποία λάβαμε.

Για τους σκοπούς της αξιολόγησης χρησιμοποιήσαμε το πρωτόκολλο SSH, καθώς αποτελεί τον πιο εύκολο τρόπο απόκτησης ελέγχου ενός μηχανήματος, εφόσον τα διαπιστευτήρια που χρησιμοποιούνται είναι αρκετά αδύναμα. Έτσι λοιπόν, εγκαταστήσαμε SSH agents σε όλα τα μηχανήματα και συλλέξαμε αποτελέσματα περίπου τριών μηνών. Κάποια από αυτά τα μηχανήματα χρησιμοποίησαν την προκαθορισμένη πόρτα 22, μερικά δέχονταν είσοδο αποκλειστικά με ζευγάρια κρυπτογραφικών κλειδιών (PKI), ενώ κάποια άλλα αποδέχονταν και κλασσικά διαπιστευτήρια (όνομα χρήστη – κωδικός πρόσβασης).

Παρατίθεται σχηματικά η διάταξη η οποία χρησιμοποιήθηκε.



Εικόνα 13 – Πειραματική Διάταξη

Όπως παρατηρούμε τα μηχανήματα κάθε οντότητας προωθούν την κίνηση τους στα αντίστοιχα τοπικά αποθετήρια (local repository) και έπειτα, αφού γίνει η απαραίτητη επεξεργασία, τα εξαγόμενα γεγονότα προωθούνται στο TTP.

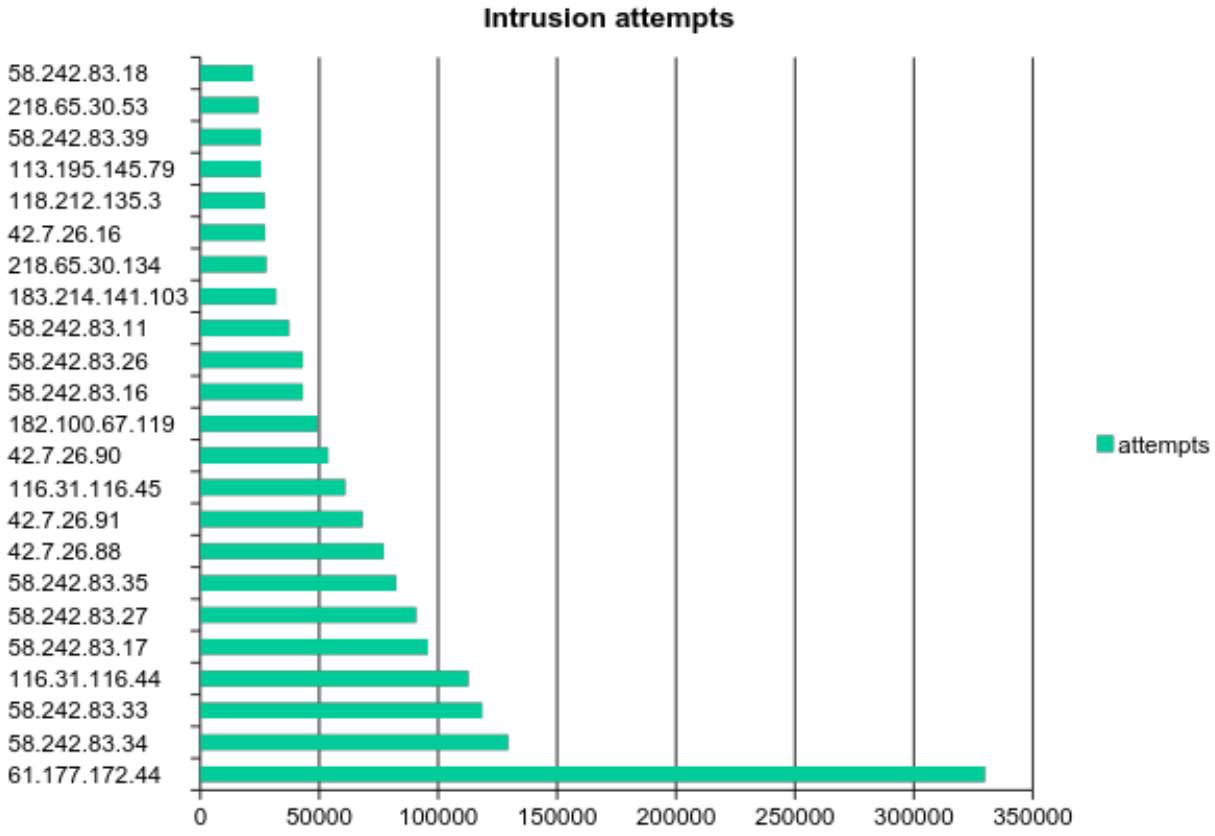
5.2 Αναλυτική παρουσίαση Αποτελεσμάτων

Σε αυτή την ενότητα θα παρουσιάσουμε τα αποτελέσματα που συλλέξαμε κατά τη διαδικασία της αξιολόγησης. Παράλληλα, θα τεκμηριώσουμε τα συμπεράσματα τα οποία εξάγονται από τα συγκεκριμένα αποτελέσματα.

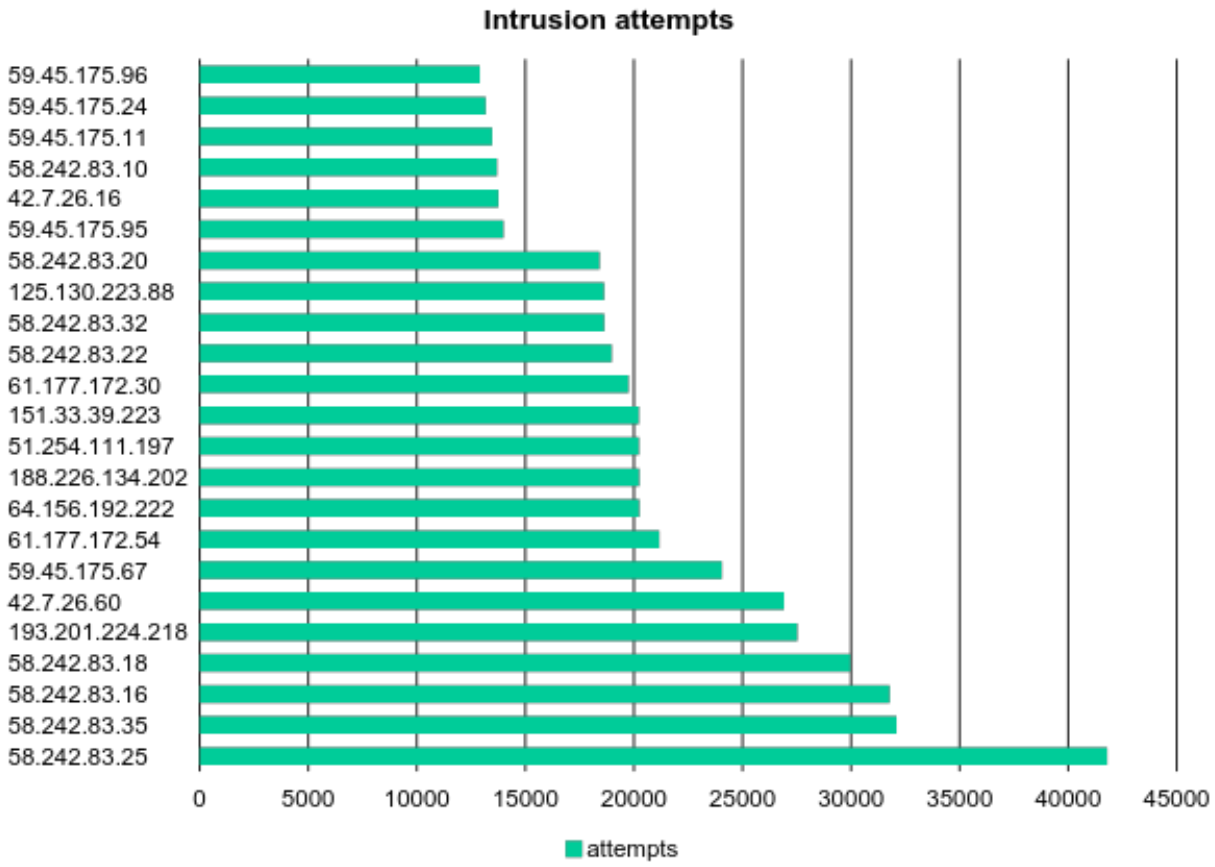
5.2.1 Ανάλυση ανά συμμετέχουσα οντότητα

Στους δύο πίνακες που ακολουθούν βλέπουμε τις κακόβουλες IP διευθύνσεις καθώς επίσης και τις προσπάθειες εισόδου στα διάφορα μηχανήματα των δύο οντοτήτων. Όπως προαναφέρθηκε, τα τεχνικά χαρακτηριστικά των μηχανημάτων είναι ετερογενή. Η ετερογένεια έγκειται στην επιλογή λειτουργικού συστήματος, μεθόδου αυθεντικοποίησης κ.α.

Επίσης, να σημειωθεί ότι για την καλύτερη κατανόηση της χρηστικότητας του μηχανισμού, δεν περιορίσαμε το ρυθμό αποστολής αιτημάτων (rate limiting) προς τα μηχανήματα.



Εικόνα 14 – Απόπειρες εισβολής για Okeanos.



Εικόνα 15 – Απόπειρες εισβολής για Netmode.

Τα αποτελέσματα συλλέχθηκαν από τις δύο οντότητες σε βάθος τριών μηνών. Αυτό που παρατηρούμε στα παραπάνω είναι ότι στα αρχεία καταγραφής που παρακολουθήσαμε περιείχαν αξιόλογη πληροφορία όσον αφορά τη διερχόμενη κίνηση. Η πληροφορία που λάβαμε δεν είχε κανένα κόστος στη λειτουργία του δικτύου καθώς συλλέγουμε τα δεδομένα παθητικά, αφότου η κίνηση περάσει στον αντίστοιχο κόμβο. Φυσικά, δεν πρέπει να παραλείψουμε την αναφορά στο γεγονός ότι η επεξεργασία της πληροφορίας δεν γίνεται σε πραγματικό χρόνο, καθώς η κίνηση διέρχεται από τον κεντρικό δρομολογητή και έτσι δεν μπορούμε να έχουμε άμεση αποτροπή ή αντιμετώπιση επιθέσεων. Ο μηχανισμός μας παρόλα αυτά αποσκοπεί στη συνεργασία των διαφόρων οντοτήτων για την αντιμετώπιση επιθέσεων στο επίπεδο των ίδιων των δικτύων που εξαπολύουν τις επιθέσεις αυτές.

5.2.2 Σύγκριση με δημόσιες λίστες αποκλεισμού

Στην παρούσα υποενότητα θα συγκρίνουμε τα εξαγόμενα δεδομένα με δημόσιες λίστες αποκλεισμού (blacklists), οι οποίες συντηρούνται από κοινότητες ανοιχτού λογισμικού, ομάδες μεγάλων εταιριών και οργανισμών [5]. Ο σκοπός μας είναι να διαπιστώσουμε εάν οι κακόβουλες IP διευθύνσεις οι οποίες επιτίθενται στην κάθε συμμετέχουσα οντότητα,

εμφανίζονται και στις διάφορες δημόσιες blacklistsκακόβουλες IP διευθύνσεις οι οποίες επιτίθενται στην κάθε συμμετέχουσα οντότητα, εμφανίζονται και στις διάφορες δημόσιες blacklists.

Adversarial IP	# of Attempts	Blacklisted	Adversarial IP	# of Attempts	Blacklisted
58.242.83.25	41795	FALSE	61.177.172.44	123408	FALSE
58.242.83.35	32095	FALSE	58.242.83.33	66230	FALSE
58.242.83.16	31780	TRUE	58.242.83.17	52548	FALSE
58.242.83.18	29969	TRUE	182.100.67.119	49161	TRUE
193.201.224.218	27529	TRUE	42.7.26.91	43178	TRUE
42.7.26.60	26913	FALSE	58.242.83.26	43147	FALSE
59.45.175.67	24051	TRUE	58.242.83.35	39211	FALSE
61.177.172.54	21168	FALSE	218.65.30.134	27779	TRUE
64.156.192.222	20252	FALSE	58.242.83.39	25556	FALSE
188.226.134.202	20250	FALSE	218.65.30.53	21672	TRUE
51.254.111.197	20238	FALSE	42.7.26.88	19893	FALSE
151.33.39.223	20232	FALSE	42.7.26.61	19526	FALSE
61.177.172.30	19756	FALSE	42.7.26.15	17634	FALSE
58.242.83.22	18979	FALSE	50.7.132.170	17216	FALSE
58.242.83.32	18634	FALSE	123.244.9.46	15657	TRUE
125.130.223.88	18628	FALSE	218.65.30.210	15391	FALSE
58.242.83.20	18420	TRUE	58.242.83.14	14917	TRUE
59.45.175.95	13995	TRUE	193.201.224.236	14416	TRUE
42.7.26.16	13756	FALSE	118.212.135.3	14065	FALSE
58.242.83.10	13696	TRUE	42.7.26.85	10595	FALSE
59.45.175.11	13470	TRUE	123.183.209.140	9598	TRUE
59.45.175.24	13173	TRUE	218.65.30.190	8768	TRUE
59.45.175.96	12904	TRUE	124.160.227.165	8536	FALSE
81.130.146.18	12870	FALSE	113.195.145.79	7298	TRUE

59.45.175.98	12823	TRUE	218.65.30.30	6686	TRUE
195.133.201.192	12797	FALSE	58.242.83.21	6684	FALSE
218.87.109.156	11874	TRUE	116.31.116.45	6547	FALSE
59.45.175.94	11653	TRUE	42.7.26.49	6271	FALSE
59.45.175.97	10547	TRUE	116.31.116.43	5984	FALSE
218.65.30.123	10539	TRUE	82.102.216.128	5415	FALSE
58.242.83.21	10498	FALSE	218.65.30.124	4731	TRUE

Πίνακας 1 – Αντιπαραβολή Αποτελεσμάτων με Δημόσια Blacklist. Netmode – Δεξιά / Okeanos – Αριστερά.

Στην τρίτη στήλη κάθε υποπίνακα βλέπουμε εάν η συγκεκριμένη IP διεύθυνση βρίσκεται στη λίστα αποκλεισμού. Όπως παρατηρούμε και στους παραπάνω πίνακες, οι περισσότερες κακόβουλες IP διευθύνσεις εμφανίζονται στη λίστα αποκλεισμού που χρησιμοποιήσαμε. Στο σημείο αυτό πρέπει να σημειωθεί ότι πολύ καλύτερα αποτελέσματα μπορούν να εξαχθούν εάν χρησιμοποιηθούν τα ημερήσια αποτελέσματα του μηχανισμού, διότι η λίστα που χρησιμοποιούμε λειτουργεί με παράθυρο 48 ωρών [14]. Παρόλα αυτά βλέπουμε ότι ακόμη και σε διάστημα τριών μηνών, διαπιστώνουμε ότι τα αποτελέσματα συναληθεύουν με τις δημόσιες λίστες αποκλεισμού.

5.2.3 Ομαδοποίηση κακόβουλων IP διευθύνσεων

Στο σημείο αυτό, θα ανακαλύψουμε εάν μπορούμε να ομαδοποιήσουμε αποδοτικά τις κακόβουλες IP διευθύνσεις σε CIDR, έτσι ώστε να διαπιστώσουμε εάν υπάρχει γειτονικότητα μεταξύ αυτών. Εφόσον υπάρχει δυνατότητα για ομαδοποίηση, τότε θα είμαστε σε θέση να οργανώνουμε κανόνες για τείχη προστασίας, και άλλες δικτυακές λύσεις, για τον πλήρη αποκλεισμό των υποδικτύων.

CIDR	# of participants	Percentage of Attack Volume	Total Attempts
58.242.83.0/24	10	31.93	255077
218.65.30.0/24	10	10.94	87421
42.7.26.0/24	7	14.68	117289
218.87.109.0/24	3	0.25	1977
116.31.116.0/24	3	1.91	15288
123.244.9.0/24	2	2.31	18425
121.194.2.0/24	2	0.07	539

137.74.167.0/24	2	0.02	131
61.177.172.0/24	2	15.66	125116
182.100.67.0/24	2	6.36	50829
5.188.10.0/24	2	0.08	649
78.155.121.0/24	2	0.08	653
193.201.224.0/24	2	1.92	15320
123.183.209.0/24	2	1.57	12578

Πίνακας 1- Αποτελέσματα σε /24 για το Okeanos

CIDR	# of participants	Percentage of Attack Volume	Total Attempts
119.193.140.0/24	32	0.02	192
58.242.83.0/24	16	24.69	235313
59.45.175.0/24	15	17.16	163475
185.165.29.0/24	15	0.12	1130
218.65.30.0/24	9	3.28	31242
191.96.249.0/24	9	0.02	203
42.7.26.0/24	8	7.08	67486
61.177.172.0/24	7	4.91	46814
202.109.143.0/24	6	0.47	4494
192.160.102.0/24	6	0.02	164
218.87.109.0/24	5	2.59	24711
121.18.238.0/24	5	0.75	7187
93.174.93.0/24	5	0.02	150
195.22.126.0/24	5	0.01	109
221.194.47.0/24	5	0.57	5440
59.63.166.0/24	4	1.32	12541
123.244.9.0/24	4	0.35	3294
103.207.37.0/24	3	0.07	654
31.207.47.0/24	3	0.02	186
121.194.2.0/24	3	0.06	541
176.126.252.0/24	2	0.01	102
39.155.134.0/24	2	0.02	237
193.201.224.0/24	2	2.9	27674

Πίνακας 2 – Αποτελέσματα σε /24 για το Netmode

Όπως παρατηρούμε στους παραπάνω πίνακες, υπάρχει αρκετή γειτονικότητα στις κακόβουλες IP διευθύνσεις. Διαπιστώσαμε μετά από δοκιμές, ότι ο ιδανικός διαχωρισμός βρίσκεται στη μάσκα υποδικτύου με 28 δυαδικά ψηφία (δηλαδή σε /28 υποδίκτυα). Παραθέτουμε τα αποτελέσματα από την ομαδοποίηση σε /28 υποδίκτυα.

CIDR	# of participants	Percentage of Attack Volume	Total Attempts
58.242.83.16/28	10	16.75	159552
192.160.102.160/28	6	0.02	164
218.87.109.144/28	5	2.59	24711
42.7.26.48/28	4	4.02	38324
123.244.9.32/28	4	0.35	3294
59.45.175.80/28	4	4.38	41739
58.242.83.0/28	4	2.63	25032
59.63.166.80/28	3	0.61	5820
59.45.175.96/28	3	3.81	36274
221.194.47.224/28	3	0.39	3721
185.165.29.112/28	3	0.02	152
59.45.175.64/28	3	4.39	41845
121.18.238.112/28	3	0.48	4565
121.194.2.240/28	3	0.06	541
176.126.252.0/28	2	0.01	102
91.197.232.96/28	2	0.78	7459
61.177.172.16/28	2	2.36	22504

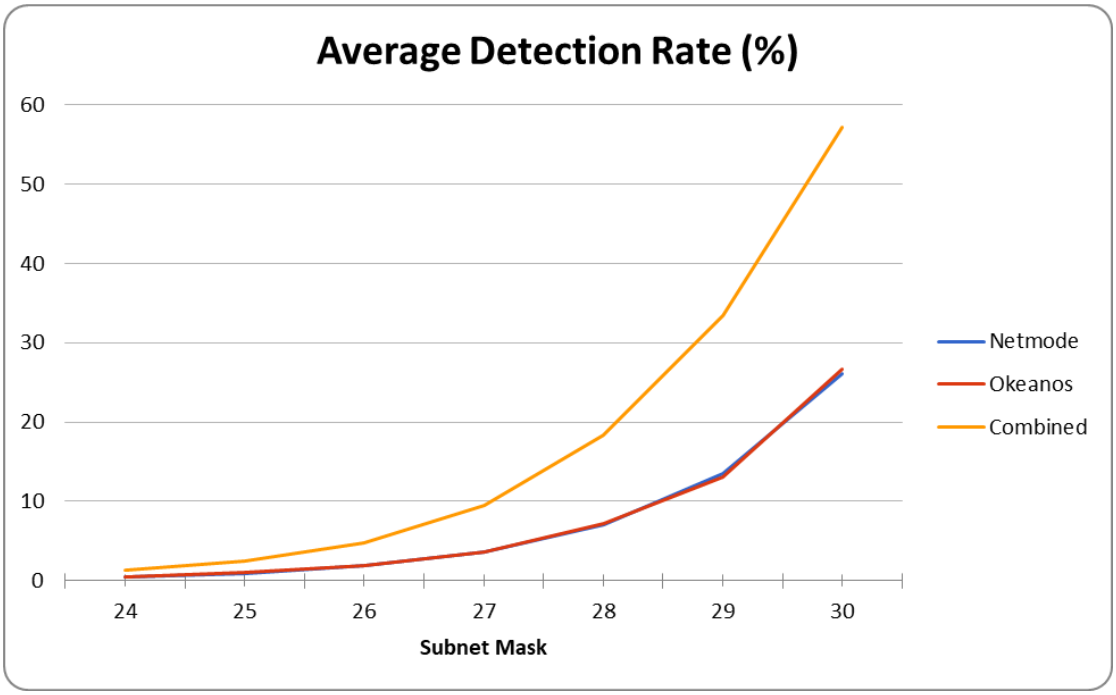
Πίνακας 3 – Αποτελέσματα σε /28 CIDRs για το Netmode.

CIDR	# of participants	Percentage of Attack Volume	Total Attempts
58.242.83.16/28	6	13.66	109163
42.7.26.80/28	3	9.22	73666
58.242.83.32/28	3	16.4	130997
218.87.109.144/28	3	0.25	1977
116.31.116.32/28	3	1.91	15288
218.65.30.112/28	3	0.89	7108
42.7.26.48/28	3	3.25	25989
123.183.209.128/28	2	1.57	12578
121.194.2.240/28	2	0.07	539
123.244.9.32/28	2	2.31	18425
218.65.30.16/28	2	0.84	6698

Πίνακας 4 – Αποτελέσματα σε /28 CIDRs για το Okeanos.

Όπως προαναφέρθηκε λοιπόν, μπορούμε να αυτοματοποιήσουμε μέρος της διαδικασίας και έτσι να εξάγουμε κανόνες αποκλεισμού των κακόβουλων υποδικτύων, χωρίς να υπάρχει κίνδυνος για εξάντληση του περιορισμένου αριθμού κανόνων που μπορούν να δεχτούν οι διάφορες δικτυακές συσκευές, καθώς όπως γνωρίζουμε όλες οι δικτυακές λύσεις αρχίζουν να μην λειτουργούν αποδοτικά μετά από ένα ορισμένο κατώφλι.

Παρακάτω βλέπουμε συγκεντρωτικά την ανάλυση του μέσου ποσοστού ανίχνευσης για διάφορες τιμές της μάσκας υποδικτύου, στην περίπτωση που ομαδοποιήσουμε με τα δεδομένα: α) του Okeanos, β) του Netmode, γ) το συνδυασμό των (α) και (β).



Εικόνα 16 – Μέσο ποσοστό ανίχνευσης για διάφορες μάσκες υποδικτύου

Αυτό που παρατηρούμε λοιπόν είναι ότι με τη χρήση του μηχανισμού μπορούμε να συνδυάσουμε την πληροφορία όλης της κοινότητας και έτσι να έχουμε πολύ πιο αποδοτική ομαδοποίηση των κακόβουλων IP διευθύνσεων.

5.2.4 Σύγκριση αποτελεσμάτων μεταξύ οντοτήτων

Παραθέτουμε στην παρακάτω εικόνα τη σύγκριση των αποτελεσμάτων των δύο οντοτήτων, με σκοπό να διαπιστώσουμε εάν υπάρχει γειτονικότητα μεταξύ τους.

Comparison Results - [Netmode - Okeanos]

Matching Percentage: 17.37%

Okeanos's Reputation 50%

Network	Local Attempts	Remote Attempts
58.242.83.0/24	235313	255077
59.45.175.0/24	163475	62
42.7.26.0/24	67486	117289
61.177.172.0/24	46814	125116
218.65.30.0/24	31242	87421
193.201.224.0/24	27674	15320
218.87.109.0/24	24711	1977

Εικόνα 13 - Συγκρίση αποτελεσμάτων για Netmode και Okeanos

Η παραπάνω εικόνα αποτελεί μέρος της web εφαρμογής που έχουμε δημιουργήσει, για τη διαχείριση και τη σύγκριση των αποτελεσμάτων της κάθε οντότητας με αυτά των υπόλοιπων οντοτήτων της κοινότητας. Όπως βλέπουμε λοιπόν, παρατηρούμε 17.37% ομοιότητα μεταξύ των οντοτήτων, κάτι που αποδεικνύει ότι εάν οι δύο αυτές οντότητες αντάλλασσαν τα αποτελέσματα αυτά, θα ήταν σε θέση να επιβεβαιώσουν, ακόμη και αυτοματοποιημένα, ότι οι συγκεκριμένες IP διευθύνσεις είναι όντως κακόβουλες. Εφόσον η μια οντότητα επιβεβαιώνει την άλλη, και οι δύο μπορούν να έχουν αυξημένη εμπιστοσύνη στα παραγόμενα δεδομένα, ότι δηλαδή δεν πρόκειται για «ψευδοθετικά» (false positives) γεγονότα.

Ως απόρροια των παραπάνω, επιβεβαιώνουμε την αξία του προτεινόμενου μηχανισμού, εφόσον όπως έχει προαναφερθεί στο 2^{ο} κεφάλαιο, οι ομάδες επιτιθέμενων στοχεύουν στην εύρεση «εύκολων στόχων», και έτσι ανιχνεύουν όλες τις περιοχές IP. Προφανώς υπάρχουν και στοχευμένες επιθέσεις εναντίων οργανισμών, εταιριών ή ακόμη και φυσικών προσώπων. Το μεγαλύτερο μέρος των διαδικτυακών επιθέσεων όμως αποτελείται από επιθέσεις της πρώτης κατηγορίας, εξαπόλυση «τυφλών» επιθέσεων προς περιοχές IP. Έτσι λοιπόν, εφόσον μιλάμε για κοινότητες από συνεργαζόμενους φορείς, ο διαμοιρασμός πληροφορίας αποτελεί ακρογωνιαίο λίθο για την αποτελεσματικότερη και πιο έγκαιρη αντιμετώπιση των επιθέσεων.

Στο σημείο αυτό πρέπει να τονιστεί ότι εφόσον εμπιστευόμαστε κάποια συνεργαζόμενη οντότητα, μπορούμε να αυτοματοποιήσουμε τη διαδικασία αποκλεισμού διευθύνσεων IP, χωρίς να περιμένουμε να δεχτούμε και εμείς την ίδια επίθεση. Με αυτόν τον τρόπο μπορούμε να πετύχουμε την πρόληψη μελλοντικών επιθέσεων. Με τη βοήθεια του γεγονότος ότι οι IP διευθύνσεις μπορούν να ομαδοποιηθούν αποτελεσματικά σε CIDR περιοχές, μπορούμε να δημιουργήσουμε κανόνες αποκλεισμού για τις συγκεκριμένες περιοχές.

6

Επίλογος

Στη συγκεκριμένη ενότητα θα παρουσιάσουμε συνοπτικά ό,τι έχει ήδη ειπωθεί στις προηγούμενες ενότητες καθώς επίσης και τα εξαγόμενα συμπεράσματα στα οποία καταλήξαμε. Τέλος, θα επισημάνουμε ποιες είναι οι πιθανές μελλοντικές επεκτάσεις του μηχανισμού.

6.1 Σύνοψη

Στην παρούσα διπλωματική εργασία παρουσιάστηκε ένας μηχανισμός για τη συλλογή πληροφορίας από αρχεία καταγραφής, τα οποία μπορεί να είναι υπηρεσίες όπως SSH, εξυπηρετητές Web εφαρμογών, IDSes, εξυπηρετητές ηλεκτρονικού ταχυδρομείου κ.α. Στη συνέχεια, παρέχεται η δυνατότητα επεξεργασίας της πληροφορίας καθώς επίσης και εύκολης επέκτασης του μηχανισμού, για την ανίχνευση νέων επιθέσεων. Επιπλέον, παρέχεται η παραγόμενη πληροφορία σε όλα τα μέλη της κοινότητας μέσω ενός κεντρικού σημείου, το TTP. Επιπροσθέτως, υπάρχει η υποδομή για την διαμόρφωση της «φήμης» όλων των οντοτήτων που συμμετέχουν στην κοινότητα μέσω της αξιολόγησης της πληροφορίας την οποία διαδίδουν στην κοινότητα. Επίσης, το TTP επεξεργάζεται όλη την παρεχόμενη πληροφορία με σκοπό να παραγάγει πληροφορία η οποία βασίζεται στα δεδομένα όλων των συμμετεχόντων.

Στο σημείο αυτό πρέπει να επισημάνουμε ότι κάθε οντότητα μπορεί να διαχειριστεί την παραγόμενη πληροφορία, δηλαδή ποιες IP διευθύνσεις ή περιοχές IP είναι κακόβουλες, με

όποιον τρόπο θέλει. Για παράδειγμα, εφόσον υπάρχει εμπιστοσύνη στα δεδομένα που παράγει ένα υποσύνολο των συμμετεχόντων, τότε μπορούμε να δημιουργήσουμε αυτοματισμούς με τους οποίους θα διαμορφώνουμε λίστες αποκλεισμού για τις IP διευθύνσεις που εμφανίζονται στα δεδομένα όλου του υποσυνόλου. Ένα άλλο παράδειγμα θα μπορούσε να είναι η δημιουργία εξυπηρετητών-παγίδων (honeypot servers) με τη βοήθεια των οποίων, ενώ δεν θα αποκλείσουμε τις συγκεκριμένες διευθύνσεις IP, θα παρακολουθήσουμε τη συμπεριφορά τους με σκοπό να την αναλύσουμε περαιτέρω σε δεύτερο χρόνο.

Τα συμπεράσματα και οι λόγοι για την εκπόνηση της συγκεκριμένης εργασίας είναι η πλούσια πληροφορία η οποία υπάρχει στα αρχεία καταγραφής και με τη βοήθεια της οποίας μπορούμε να αντιμετωπίσουμε συλλογικά πιθανές επιθέσεις. Όπως αποδείχθηκε στο 5^{ο} κεφάλαιο, οι ομάδες επιτιθέμενων στοχεύουν συνήθως περιοχές IP οι οποίες καλύπτουν διάφορους οργανισμούς και υπηρεσίες. Έτσι λοιπόν, η συνεργατική αντιμετώπιση των συγκεκριμένων ομάδων είναι πολύ πιο αποτελεσματική, και παράλληλα παρέχει εμπιστοσύνη στα δεδομένα που παράγει μια οντότητα/οργανισμός.

6.2 Βελτιώσεις και μελλοντικές επεκτάσεις

Αρχικά οι βελτιώσεις που μπορούν να αναπτυχθούν στον προτεινόμενο μηχανισμό είναι κυρίως επεκτάσεις αυτού. Για αυτό το λόγο, η αρχιτεκτονική του μηχανισμού έχει επικεντρωθεί στην εύκολη επέκταση του. Έτσι λοιπόν, θα μπορούσαμε να επεκτείνουμε του ανιχνευτές/επεξεργαστές να αναζητούν περισσότερα είδη επιθέσεων, καθώς επίσης να προσθέσουμε επιπλέον υπηρεσίες προς επεξεργασία. Στη συνέχεια, υπάρχει η δυνατότητα χρήσης άλλων μορφών για τα παραγόμενα events, όπως παρουσιάζεται στο [15], με σκοπό την πληρέστερη κάλυψη όλης της απαραίτητης πληροφορίας και της αποδοχής αυτών των events και από άλλους παρόμοιους μηχανισμούς.

Επιπροσθέτως, στο κομμάτι της ανίχνευσης επιθέσεων μπορούμε όπως και τα IDSes, να χρησιμοποιήσουμε μεθόδους μηχανικής εκμάθησης (machine learning), έτσι ώστε να κατηγοριοποιήσουμε την κακόβουλη συμπεριφορά, και έτσι να ξεφύγουμε από τους περιορισμούς του σαφούς ορισμού κανόνων για την ανίχνευση των επιθέσεων. Με αυτόν τον τρόπο λοιπόν, μπορούμε να ανιχνεύσουμε επιθέσεις ή εκδοχές μιας επίθεσης που δεν έχουμε συναντήσει προηγουμένως. Βέβαια, όπως και στην περίπτωση των IDSes, αυτή η λύση έχει κάποια μειονεκτήματα. Ένα από αυτά είναι τα ψευδοθετικά (false positive) events, κάτι που σε οργανισμούς οι οποίοι προχωρούν σε ανάλυση από φυσικά πρόσωπα, μπορεί να οδηγήσει σε σπατάλη εργατοωρών. Αυτό οφείλεται στη δυσκολία σαφούς ορισμού της «φυσιολογικής» κίνησης στα διάφορα δίκτυα, καθώς υπάρχουν πολλές παράμετροι οι οποίες αλλάζουν ανά περίσταση.

Επίσης, στο κομμάτι της διαμόρφωσης της φήμης κάθε οντότητας, αξίζει να αναζητηθεί ένα σύστημα εμπιστοσύνης το οποίο θα δίνει περισσότερα κίνητρα για την καλόβουλη και συνεχή συμμετοχή στην κοινότητα.

Στη συνέχεια, όπως έχει προαναφερθεί στο 4ο και 5ο κεφάλαιο, εφόσον η παραγόμενη πληροφορία του μηχανισμού αφορά κακόβουλες IP διευθύνσεις, θα μπορούσαν να δημιουργηθούν μηχανισμοί αυτοματισμού για τη δημιουργία κανόνων αποκλεισμού των περιοχών αυτών, είτε χωρίς καμία ανθρώπινη παρέμβαση, είτε για την επιβεβαίωση της προώθησης των κανόνων. Επιπλέον, αξία έχει η αναζήτηση μεθόδων για την ομαδοποίηση των διαφόρων περιοχών, οι οποίες θα λαμβάνουν υπόψη τρεις παραμέτρους:

1. Μέγιστος αριθμός κανόνων για τις διάφορες δικτυακές συσκευές.
2. Επιθυμία για αποκλεισμό όλων των κακόβουλων υποδικτύων.
3. Επιθυμία για το μικρότερο δυνατό αριθμό αποκλεισμού (ενδεχομένως) καλόβουλων διευθύνσεων μέσα στα συγκεκριμένα υποδίκτυα.

Ένας τέτοιος αλγόριθμος προτείνεται στο [16, §4.3.2].

7

Βιβλιογραφία

[1] Jonas Taftø Rødfoss.
Comparison of Open Source Network Intrusion Detection Systems, 2011.

[2] https://krebsonsecurity.com/2013/04/brute-force-attacks-build-wordpress-botnet/

[3] https://www.csid.com/wp-content/uploads/2012/09/CS_PasswordSurvey_FullReport_FINAL.pdf

[4] Haakon Andreas Ringberg: Privacy-Preserving Collaborative Anomaly Detection, 2009.

[5] https://zeltser.com/malicious-ip-blocklists/

[6] http://www.misp-project.org/

[7] https://www.virustotal.com

[8] https://warden.cesnet.cz/en/index

[9] PAVEL KÁCHA, IDEA: Classification of security events, their participants and detection probes, 2015

[10] https://webmasters.googleblog.com/2014/08/https-as-ranking-signal.html

[11] https://encryptallthethings.net/

[12] https://www.elastic.co/guide/en/elasticsearch/reference/current/common-options.html#date-math

[13] https://okeanos.grnet.gr/about/what/

[14] https://github.com/firehol/blocklist-ipsets/blob/master/firehol_level2.netset

[15] Jessica Steinberge, Anna Sperotto, Mario Golling and Harald Baier: «How to Exchange Security Events? Overview and Evaluation of Formats and Protocols», 2015.

[16] Kostas Giotis, George Androulidakis and Vasilis Maglaris: «A scalable anomaly detection and mitigation architecture for legacy networks via an OpenFlow middlebox», 2015.

8

Παράρτημα Α

8.1 Τοπικό Αποθετήριο (Local Repository)

8.1.1 models.py

```
from . import db
from app import app
import datetime

class Reputation(db.Model):
    id              = db.Column(db.Integer, primary_key=True)
    date_modified   = db.Column(db.DateTime,
default=datetime.datetime.utcnow,

onupdate=datetime.datetime.utcnow)
    date_created    = db.Column(db.DateTime,
default=datetime.datetime.utcnow)
    name            = db.Column(db.String(40), unique=True)
    rep             = db.Column(db.SmallInteger,
default=app.config['DEFAULT_REP'])
```

```
    def __init__(self, name):
        self.name = name

    def __repr__(self):
        return '%s' % self.name
```

8.1.2 views.py

```
from flask import render_template, request, jsonify, abort, url_for,
redirect
from . import app, db
from .models import Reputation
from elasticsearch import Elasticsearch
from elasticsearch_dsl import Search
from datetime import datetime, timedelta
from pprint import pprint
import json

#############################################################
######################## Views ##############################
#############################################################
@app.route('/', methods=['GET','POST'])
def index():
    if request.form:
        # Request district from database
        district_name = request.form.get('topup')
        district =
db.session.query(Reputation).filter_by(district=district_name).first
()

        # TODO: Define reputation formula
        rep_formula = district.rep + 10
        district.rep = min(rep_formula, 100)
        db.session.commit()

        # Push new reputation to TTP
        push_to_ttp(app.config, district)
```

```
        # Redirect to index to avoid resubmitting the reputation
increase form
        return redirect(url_for('index'))
    districts = Reputation.query.all()
    return render_template('app/index.html',districts=districts )

@app.route('/ttp',)
def ttp_feed():
    data= {}
    collaborator = request.args.get('coll')

    # Load district's latest per attacker aggregation
    with open(app.config['LOCAL_LATEST_AGGR']) as f:
        local_aggr_event = json.load(f)

    if collaborator:
        results = []
        index = collaborator + '-aggrevents-*'
        es_client = Elasticsearch(hosts=app.config['TTP_HOST'],
port=app.config['TTP_PORT'])

        # TODO: let user define the doc_type he wants to see
        # Or operate on all doc_types at once on post-processing!
        # Also, iterate over all districts' indices
        s = Search(using=es_client, index=index, doc_type='auth') \
            .query('match', aggregation_type='attacker') \
            .sort('-@timestamp') \
            .extra(size=1)          # Query remote district's latest
aggregation from TTP
        remote_aggr = s.execute()[0]

        indexable_aggr = ldict_to_ddict(remote_aggr) # Improves
performance greatly, read definition
        num_matching_events = 0
        num_total_events = 0
        for cidr in local_aggr_event['cidrs']:
            num_total_events += 1
            if cidr['network'] in indexable_aggr:
```

```
                results.append((
                    cidr['network'],
                    cidr['total_attempts'],
                    indexable_aggr[cidr['network']]
                ))
                num_matching_events += 1

        # Enrich document
        remote_district =
Reputation.query.filter_by(name=collaborator).first()
        data['remote_district'] = remote_district
        data['aggr_event'] = remote_aggr
        data['percentage'] = float(format((num_matching_events /
num_total_events) * 100, '.2f'))
        data['results'] = sorted(results, key=lambda tup: tup[1],
reverse=True)
    else:  # If we are not comparing against any other district,
show our local latest aggregation
        data['aggr_event'] = local_aggr_event

    data['local_distr'] = app.config['LOCAL_DISTRICT']
    data['districts'] = Reputation.query.all()
    return render_template('app/ttp.html', **data)

@app.route('/decay_rep')
def decay_reputation():
    """Decreases every district's reputation at regular intervals"""

    # Decay all reputations
    update_body = {
            'doc' :{}
        }
    try:
        for district in Reputation.query.all():
            district.rep = max(district.rep -
app.config['AGING_DECAY'], app.config['MIN_REP_VALUE'])
            db.session.add(district)
            db.session.commit()
```

```
            update_body['doc'][district.name] = district.rep
    except Exception as e:
        print(e)
        abort(500)

    # Push new reputations to TTP
    es = Elasticsearch(hosts=app.config['TTP_HOST'],
port=app.config['TTP_PORT'])
    rep_index = app.config['TTP_REP_INDEX']
    rep_doc_type = app.config['LOCAL_DISTRICT']
    es.update(index=rep_index, doc_type=rep_doc_type, id=1,
body=update_body)
    return 'Update Complete'

@app.route('/ttp_all')
def ttp_all_list():
    """Gets the latest aggregation for all districts from TTP and
serves it"""

    es = Elasticsearch(hosts=app.config['TTP_HOST'],
port=app.config['TTP_PORT'])
    aggr_index = app.config['TTP_AGGR_INDEX_PREFIX'] + '*'
    doc_type = 'all_raw'
    s = Search(using=es, index=aggr_index, doc_type=doc_type) \
        .sort('-@timestamp') \
        .extra(size=1)
    all_raw_event = s.execute()[0]
    attackers = sorted(all_raw_event['attackers'], key=lambda k:
k['attempts'], reverse=True)
    cidrs = sorted(all_raw_event['cidrs'], key=lambda k:
k['total_attempts'], reverse=True)
    return render_template('app/ttp_all.html',
event=all_raw_event,cidrs=cidrs, attackers=attackers)

##############################################################
################## Private Functions ##################
##############################################################
```

```
def push_to_ttp(config, district):
    es = Elasticsearch(hosts=config['TTP_HOST'],
port=config['TTP_PORT'])
    rep_index = config['TTP_REP_INDEX']
    rep_doc_type = config['LOCAL_DISTRICT']
    insert_body = {
        district.name: district.rep
    }
    update_body = {
        'doc' :{
                district.name: district.rep,
        }
    }
    if not es.indices.exists(index=rep_index):
        es.indices.create(rep_index)
    if not es.exists(index=rep_index, doc_type=rep_doc_type, id=1):
        es.index(index=rep_index, doc_type=rep_doc_type, id=1,
body=insert_body)
    else:
        es.update(index=rep_index, doc_type=rep_doc_type, id=1,
body=update_body)


def ldict_to_ddict(aggregations):
    """
    Converts a list of dictionaries to a dictionary of dictionaries.

    Note: We use this as an optimization since we want to cross
check all the subnets found
    in our local aggregations against the collaraborator's
aggregations. Therefore,
    if we use a listwe need O(n) time to find an item in the list,
    instead we convert it to a dictionary that requires O(1) in the
general case.

    Also, we cannot have a dictionary in the first place because of
Elasticsearch limitations.
```

```
    For more information and a deeper understanding check out
ESclient's Wiki
    on "Implementation Concerns"
    """
    d = {}
    for cidr in aggregations['cidrs']:
        d[cidr['network']] = cidr['total_attempts']
    return d
```

8.2 ESclient

8.2.1 detectors.py

```
from abc import ABC,abstractmethod
from elasticsearch import helpers
from elasticsearch_dsl import Search, Q
from pprint import pprint
import itertools

from .config import config as cfg
from . import userlog

class BaseDetector(ABC):
    """
    Base detector class. Contains the unified process_response logic
    """

    def __init__(self, es_client, service, indices="*",
doc_type=None):
        self.indices = indices
        self.doc_type = doc_type
        self.es_client = es_client
        self.service = service
        userlog.info('Running event generation on: -doc_type:\t
{0}'.format(doc_type))
        userlog.info('\t\t\t\t -service:\t {0}'.format(service))
        userlog.info('\t\t\t\t -indices:\t {0}'.format(indices))
```

```
    @abstractmethod
    def build_query(self, *args):
        pass

    def execute_query(self, gt):
        """ This function is used in order to perform periodic
processing of events """
        query = self.build_query()
        if gt:
            search = query.filter('range', ** { '@timestamp':
{'gte': 'now-'+gt, 'lt': 'now'}})
            return search.scan()
        return query.scan()
    def process_response(self, response):
        """
        Parses loglines, produces events, and reindexes them back to
today's index
        Log lines' index pattern is: "netmode-logstash-YYYY.MM.DD",
        Whereas, events follow this: "netmode-events-YYYY.MM.DD".
        """
        userlog.info('Received a generator for the requested
queries')
        batch_size = cfg['general']['batch_size']
        index = cfg['general']['todays_index']
        es = self.es_client

        # If there are any registered processors for the calling
object's class
        if hasattr(self, 'processors'):
            if not es.indices.exists(index=index):
                userlog.info('Index {0} does not
exist'.format(index))
                es.indices.create(index)
                userlog.info('Created index')
            else:
                userlog.info('Index {0} exists'.format(index))
```

```
            userlog.info('Batch processing started...')
            gen = iter(())
            while True:
                # Evaluate the response's generator in batches
                batch = list(itertools.islice(response, batch_size))
                for proc in self.processors:
                    batch_gen = ({
                            '_type': self.doc_type,
                            '_index': index,
                            '_source': event_source,
                        } for event_source in proc(batch))
                    gen = itertools.chain(batch_gen, gen)

                if len(batch) != batch_size:  # we reached the last
batch
                    break
            userlog.info('Batch processing ended...')
            if cfg['general']['DEBUG']:
                for x in gen:
                    pprint(x['_source']['@timestamp'])
                    #print('\n' + '----------------------------
','\n')
            else:
                userlog.info('Indexing started...')
                doc_count = helpers.bulk(es, gen)
                userlog.info('{0} documents were inserted into
{1}...'.format(doc_count, index))

class SSHDetector(BaseDetector):

    def build_query(self):
        batch_size = cfg['general']['batch_size']
        s = Search(using=self.es_client, index=self.indices,
doc_type=self.doc_type) \
            .query('match', service=self.service) \
            .sort('@timestamp') \
            .params(preserve_order=True, size=batch_size)
        return s
```

```
class DovecotDetector(BaseDetector):

    def build_query(self):
        s = Search(using=self.es_client, index=self.indices,
doc_type=self.doc_type) \
            .query(~Q('match', user='<VALID_USER>')) \
            .query('match', service=self.service) \
            .sort('@timestamp') \
            .params(preserve_order=True)
        return s

class WebDetector(BaseDetector):

    def build_query(self):
        s = Search(using=self.es_client, index=self.indices,
doc_type=self.doc_type) \
            .query('match', service=self.service) \
            .sort('@timestamp') \
            .params(preserve_order=True)
        return s

import app.processors
```

8.2.2 post_processors.py

```
from elasticsearch import Elasticsearch as ES
from elasticsearch_dsl import Search, Q
from app.config import config as cfg
from pprint import pprint
import itertools,ipaddress, requests, sys
from app import userlog
import datetime, json

def aggregator(service, indices, aggr_type, lt, gt, extra, mask):
    """
```

```
    Aggregates events out of the chosen indices and service
depending on aggr_type.
    If enabled, produces the /n subnets in order to group the
attackers.
    Lastly, lt and gt compose the time frame we are interested in.
    """
    internal_es = ES()

    doc_type = cfg['services_to_doctypes'][service]
    search = Search(using=internal_es, index=indices,
doc_type=doc_type) \
        .query('match', service=service) \
        .extra(size=0)

    userlog.info('Running event aggregation on:-doc_type:\t
{0}'.format(doc_type))
    userlog.info('\t\t\t\t -service:\t {0}'.format(service))
    userlog.info('\t\t\t\t -indices:\t {0}'.format(indices))
    userlog.info('Searching in the following timeframe:
[{0},{1}]'.format(gt,lt))
    userlog.info('In a per {0} manner'.format(aggr_type))

    search = search.filter('range', ** { '@timestamp': {'gte': gt,
'lt': lt}})

    # Get current timestamp in utc
    now = datetime.datetime.utcnow()
    ltime = now.strftime("%Y-%m-%dT%H:%M:%S") + ".%03d" %
(now.microsecond / 1000) + "Z"

    # Build document's base body
    source = {
        'district'          : cfg['general']['district'],
        'timeframe'         : '[{0},{1}]'.format(gt,lt),
        'aggregation_type'  : aggr_type,
        'source_indices'    : indices,
        '@timestamp'        : ltime,
        'source_doc_type'   : doc_type,
```

```
        'source_service'    : service,
    }
    # Enrich it with --extra flags
    if 'xcheck' in extra:
        source['blacklist_url'] = cfg['ssh']['blacklist_url']
    for token in extra:
        source[token] = True

    blacklist = None
    if 'xcheck' in extra:
        blacklist = construct_blacklist()

    if aggr_type == 'victim':
        search.aggs.bucket('per_victim','terms',
field='target_host.keyword', size=1000) \
                   .bucket('per_attacker', 'terms',
field='source_ip.keyword', size=100000)
        response = search.execute()

        source['victims'] = get_victims(response, blacklist, extra,
mask)
    elif aggr_type == 'attacker':
        search.aggs.bucket('per_attacker', 'terms',
field='source_ip.keyword', size=10000)
        response = search.execute()
        if 'cidrs' in extra:
            source['attackers'], source['cidrs'] =
get_attackers(response, blacklist, extra, mask)
        else:
            source['attackers'] = get_attackers(response, blacklist,
extra, mask)
    # Index to TTP only if debug is off and we aggregate per
attacker
    if not cfg['general']['DEBUG'] and aggr_type == 'attacker':
        general_cfg = cfg['general']
        index = send_to_ttp(
            general_cfg['district'],
            general_cfg['today'],
```

```
            general_cfg['ttp_ip'],
            general_cfg['ttp_port'],
            source,
            doc_type  # Aggrevents are stored in the same type as
the source events
        )

        # In order to compare our latest results to another
district's
        # we store our aggregations to
        source['index'] = index
        with open('latest_aggr.json','w+') as f:
            json.dump(source, f)
    elif aggr_type == 'victim':
        pprint_events(source, aggr_type)
        userlog.error('Victim-based aggregations cannot be pushed to
TTP')
    else:
        pprint_events(source, aggr_type)
        userlog.info('Debug is turned on, no events will be pushed
to TTP')

def send_to_ttp(district, today, ttp_ip, ttp_port, source,
doc_type):
    """
    Sends the districts newly produced aggregation to the TTP.
    Returns the index used to store the documents
    """
    index = '{0}-aggrevents-{1}'.format(district, today)
    userlog.info('Sending the aggregated events to TTP\'s ES
instance')
    userlog.info('TTP ip:port : {0}:{1}'.format(ttp_ip, ttp_port))
    userlog.info('Index: {0} \t doc_type: {1} \t '.format(index,
doc_type))

    ttp_es = ES(hosts=ttp_ip, port=ttp_port)
    ttp_es.index(index=index, doc_type=doc_type, body=source)
```

```
    return index

def get_attackers(response, blacklist, extra, mask):
    """
    Returns the attackers and cidrs ( if cidrs are enabled ).
    Attackers' pattern is shown below:
    [
        {attacker_ip: 187.22.11.3, attempts: 1234},
        ....
    ]
    If cidrs aggregation is enabled a 'cidrs' field is added
    and its format goes as follows:
    [
        {
            attackers:[
                {attacker_ip: 187.22.11.3, attempts: 1234},
                ....
            ]
        },
        network: 187.22.11.0/24,
        participants: 12
        percentage: 51.2
        total_attempts: 25500
    ]
    """
    attackers = []
    for attacker in response.aggregations.per_attacker.buckets:
        attacker_dict = {
            'attacker_ip'    : attacker.key,
            'attempts'       : attacker.doc_count,
        }
        if blacklist:
            blacklisted = (True if attacker.key in blacklist else
False)
            attacker_dict['blacklisted'] =  blacklisted
        attackers.append(attacker_dict)
    if 'cidrs' in extra:
        cidrs = ips_to_cidrs(attackers, mask)
```

```
        return (attackers, cidrs)
    return attackers

def get_victims(response, blacklist, extra, mask):
    """
    Return the a list of victims.
    Every victim follow the format shown below:
    {
        victim_host: onosvm,
        total_attempts: 1234,
        attackers: [
            {attacker_ip: 79.126.7.7, attempts: 1234},
            ...
        ]
    }

    If cidrs aggregation is enabled the 'attackers' dictionary
format goes as follows:
    {
        attackers: {
            [
                {attacker_ip: 79.126.7.7, attempts: 1234},
                {attacker_ip: 79.126.7.8, attempts: 1234},
            ]
                network: 79.126.7.0/24,
                participants: 2,
                percentage: 0.22,
                total_attempts: 2478
            }
        }
    }
    """
    victims = []
    for victim in response.aggregations.per_victim.buckets:
        victim_dict = {
            'victim_host'   : victim.key,
            'total_attempts': victim.doc_count,
            'attackers'     : [],
```

```
        }
        for attacker in victim.per_attacker.buckets:
            attacker_dict = {
                'attacker_ip'    : attacker.key,
                'attempts'       : attacker.doc_count,
            }
            if blacklist:
                blacklisted = (True if attacker.key in blacklist
else False)
                attacker_dict['blacklisted'] = blacklisted
            victim_dict['attackers'].append(attacker_dict)
        if 'cidrs' in extra:
            cidrs = ips_to_cidrs(victim_dict['attackers'], mask)
            victim_dict['cidrs'] = cidrs
        victims.append(victim_dict)
    return victims

def construct_blacklist():
    """
    Returns a set containing blacklisted IPs.
    """
    url = cfg['ssh']['blacklist_url']
    try:
        req = requests.get(url, timeout=6)
    except requests.exceptions.Timeout:
        userlog.error('The requested blacklist at {0} timed
out'.format(url))
        sys.exit()

    # Expands all /n networks into their respective IP ranges
    # There should be some ipaddress module magic that can handle
that.
    IP_blacklist = set()
    for line in req.text.split("\n"):
        if not line.startswith('#'):
            if '/' in line:
                for ip in ipaddress.ip_network(line):
                    IP_blacklist.add(ip.exploded)
```

```
            else:
                IP_blacklist.add(line)
    return IP_blacklist

def ips_to_cidrs(attackers, mask):
    """
    Produces a list of dictionaries containing the cidrs alongside
    the corresponding attackers for each cidr

    Note: Read the "Implementation Concerns" wiki about the
    conversion to list of dictionaries
    """
    cidrs={}
    total_attempts = 0
    for attacker in attackers:
        cidr = {}
        ip = ipaddress.ip_address(attacker['attacker_ip'])
        if type(ip) == ipaddress.IPv4Address:
            network =
ipaddress.IPv4Network(ip.exploded+'/'+str(mask),
strict=False).exploded
            if network in cidrs:
                cidrs[network]['participants'] += 1
                cidrs[network]['attackers'].append({
                    'attacker_ip'   : attacker['attacker_ip'],
                    'attempts'      : attacker['attempts'],
                    })
            else:
                cidrs[network] = {
                'participants': 1,
                'total_attempts': 0,
                'attackers':[{
                    'attacker_ip'   : attacker['attacker_ip'],
                    'attempts'      : attacker['attempts']
                    }]
                }
            cidrs[network]['total_attempts'] += attacker['attempts']
            total_attempts += attacker['attempts']
```

```
    output_cidrs = []
    for cidr,values in cidrs.items():
        cidrs[cidr]['percentage'] = \
            float(format((cidrs[cidr]['total_attempts'] /
total_attempts) * 100, '.2f'))
        values['network'] = cidr
        output_cidrs.append(values)
    return output_cidrs

def pprint_events(events, type):
    """ Prints events in a human readable manner """
    if type == 'victim':
        if 'cidrs' in events:
            print('Victim: {0:20} | {1:20}'.format('CIDR',
'Attempts'))
            for victim in events['victims']:
                for cidr in victim['cidrs']:
                    print('\t {0:20} | {1:20} |
{2}'.format(cidr['network'], cidr['total_attempts'],
cidr['participants']))
            return
        for victim in events['victims']:
            print('Victim: {0}'.format(victim['victim_host']))
            print('\t {0:20} | {1:10}'.format('Attacker',
'Attempts'))
            for attacker in victim['attackers']:
                print('\t {0:20} | {1:10} |
{2:20}'.format(attacker['attacker_ip'], attacker['attempts'],
attacker['blacklisted']))
    else:
        if 'cidrs' in events:
            print('{0:20} | {1:20} | {2:20}'.format('CIDR',
'Attempts', 'Number of Participants'))
            cidrs = sorted(events['cidrs'], key=lambda k:
k['participants'], reverse=True)
            for cidr in cidrs:
                cidr.pop('attackers')
                if cidr['total_attempts'] > 100:
```

```
                    print('{0:20} | {1:20} | {2:20}'.format(
                            cidr['network'], cidr['total_attempts'],
cidr['participants']))
            cidrs = [d for d in cidrs if d['total_attempts'] > 100]
            return
        print('{0:20} | {1:10} | {2}'.format('Attacker', 'Attempts',
'blacklisted'))
        for attacker in events['attackers']:
            print('{0:20} | {1:10} |
{2}'.format(attacker['attacker_ip'], attacker['attempts'],
attacker['blacklisted']))
```

8.2.3 processors.py

```
from datetime import datetime
from pprint import pprint

from app.utils import register_proc, whoami
from app.detectors import *
from app.config import config as cfg
from app.config import ssh_list

@register_proc(SSHDetector)
def ssh_processor(batch):
    processor = whoami()
    for hit in batch:
        # ssh accepts a whitespace as username,
        # but it is not caught by logstash's regex
        username = ' ' if not hasattr(hit, 'username') else
hit.username
        event= {
            'event_processor':  processor,
            '@timestamp' :      hit['@timestamp'],
            'source_ip':        hit.attacker_ip,
            'target_host':      hit.host,
            'contributor':      hit.district,
            'service':          hit.service,
            'attack_type':      'brute_force',
```

```
        }
        if hit.logline_type=='SSHBANNERGRAB':
            event.update({
                    'attack_type': 'fingerprinting',
                    'description': 'Version fingerprinting',
                })
        elif hit.logline_type=='SSHFAILEDLOGIN' or
hit.logline_type=='SSHFAILEDLOGINF':

            event.update({
                'description': 'Failed attempt on a valid user',
                'specifics': {
                    'username': username,
                    'valid_user': True,
                    'method': 'PW',
                },
            })
        elif hit.logline_type=='SSHINVALIDUSER' or
hit.logline_type=='SSHINVALIDUSERF':
            event.update({
                'description': 'Failed attempt to login on an
invalid user',
                'specifics': {
                    'username': username,
                    'valid_user': False,
                    'method': 'PW',
                },
            })
        elif hit.host not in ssh_list:
            if hit.logline_type=='SSHINVALIDUSERPKI':
                event.update({
                    'description': 'Failed attempt to login on an
invalid user',
                    'specifics': {
                        'username': username,
                        'valid_user': False,
                        'method': 'PKI',
```

```
                    },
                })
            elif hit.logline_type=='SSHCONNCLOSED' or
hit.logline_type=='SSHCONNCLOSEDPKI':
                event.update({
                    'specifics': {
                        'method': 'PKI',
                    },
                    'description': 'Failed all auth methods on a
password-disabled ssh daemon',
                })
        else:
            #print('Non-type ', hit.message, hit.logline_type)
            continue
        yield event

@register_proc(DovecotDetector)
def mail_processor(batch):
    who = whoami()
    for hit in batch:
        if hasattr(hit, 'user'):
            domain = '-' if not hasattr(hit, 'domain') else
hit.domain
            yield {
                '@timestamp' :      hit['@timestamp'],
                'event_processor':  who,
                'source_ip':        hit.attacker_ip,
                'target_host':      hit.host,
                'contributor':      hit.district,
                'service':          hit.service,
                'attack_type':      'brute_force',
                'specifics': {
                    'domain':   domain,
                    'user':     hit.user,
                }
            }

@register_proc(WebDetector)
```

```
def non_wp_website(batch):
    processor = whoami()
    wp_list = ['wp-login', 'wp-content', 'wordpress', 'xmlrpc.php']
    for hit in batch:
        if any(token in hit.message for token in wp_list):
            yield {
                '@timestamp' :      hit['@timestamp'],
                'event_processor':  processor,
                'source_ip':        hit.attacker_ip,
                'target_host':      hit.host,
                'contributor':      hit.district,
                'service':          hit.service,
                'attack_type':      'wpscan',
                'specifics': {
                    'request':      hit.request,
                    'response':     hit.response,
                    'bytes':        hit.bytes,
                    'http_method':  hit.verb,
                },
            }

@register_proc(WebDetector)
def non_php_website(batch):
    processor = whoami()
    wp_list = ['wp-login', 'wp-content', 'wordpress', 'xmlrpc.php']
    for hit in batch:
        if 'MALFORMED_REQUEST' not in hit.tags:
            if 'php' in hit.request:
                if not any(token in hit.message for token in
wp_list):
                    yield {
                        'event_processor':  processor,
                        '@timestamp' :      hit['@timestamp'],
                        'source_ip':        hit.attacker_ip,
                        'target_host':      hit.host,
                        'contributor':      hit.district,
                        'service':          hit.service,
                        'attack_type':      'fingerprinting',
```

```
                        'specifics': {
                            'request':      hit.request,
                            'response':     hit.response,
                            'bytes':        hit.bytes,
                            'http_method':  hit.verb,
                        },
                    }

@register_proc(WebDetector)
def malformed_requests(batch):
    processor = whoami()
    for hit in batch:
        if 'MALFORMED_REQUEST' in hit.tags:
            yield {
                '@timestamp' : hit['@timestamp'],
                'event_processor':  processor,
                'source_ip':    hit.attacker_ip,
                'target_host':  hit.host,
                'contributor':  hit.district,
                'service':      hit.service,
                'attack_type':  'Known Exploit Iteration',
                'specifics': {
                    'malformed_request': hit.malformed_req,
                },
            }
```

8.2.4 utils.py

```
# Decorator that registers processors to Detectors
def register_proc(*classes):
   """
   A simple decorator that attaches the registered functions
   to the appropriate classes
   """
   def decorated(f):
      for cls in classes:
         if hasattr(cls, 'processors'):
            cls.processors.append(f)
         else:
```

```
            setattr(cls, 'processors', [f])
        return f
    return decorated

class colors:
    PURPLE = '\033[95m'
    CYAN = '\033[96m'
    DARKCYAN = '\033[36m'
    BLUE = '\033[94m'
    GREEN = '\033[92m'
    YELLOW = '\033[93m'
    RED = '\033[91m'
    BOLD = '\033[1m'
    UNDERLINE = '\033[4m'
    END = '\033[0m'

import inspect

def whoami():
    """ Returns the name of the calling function """
    return inspect.stack()[1][3]

class UserLog:
    def info(self, format_string):
        print("{0}[*]
{1}{2}".format(colors.GREEN,format_string,colors.END))
    def warn(self, format_string, bold=True):
        if bold:
            print("{0}{1}[*]
{2}{3}".format(colors.YELLOW,colors.BOLD,format_string,colors.END))
        else:
            print("{0}[*]
{1}{2}".format(colors.YELLOW,format_string,colors.END))
    def error(self, format_string):
        print("{0}[*]
{1}{2}".format(colors.RED,format_string,colors.END))
```

9

Παράρτημα Β

9.1 Filebeat

9.1.1 Apache

```
###################### Filebeat Apache Configuration ##########################

#========================== Filebeat prospectors =============================

filebeat.prospectors:

- input_type: log
  paths:
    # Cover All Linux distributions
    - "/var/log/httpd/access.log"
    - "/var/log/httpd/error.log"
    - "/var/log/apache2/access.log"
    - "/var/log/apache2/error.log"
    - "/var/log/httpd-access.log"
    - "/var/log/httpd-error.log"
  document_type: webserver
  fields:
    service: apache
  fields_under_root: true

#================================ General =====================================

#================================ Outputs =====================================


#-------------------------- Logstash output -------------------------------
output.logstash:
```

```
  hosts : ["10.10.10.10:5044"]

  # List of root certificates for HTTPS server verifications
  #ssl.certificate_authorities: ["/etc/pki/root/ca.pem"]

  # Certificate for SSL client authentication
  #ssl.certificate: "/etc/pki/client/cert.pem"

  # Client Certificate Key
  #ssl.key: "/etc/pki/client/cert.key"
```

9.1.2 Auth (SSH)

```
######################### Filebeat sshd configuration ############################

#============================ Filebeat prospectors ==============================

filebeat.prospectors:

- input_type: log
  # Two filenames are specified to cover CentOS and Ubuntu flavored OSes
  paths:
    - "/var/log/auth.log*"
        # Avoid parsing "/var/log/securecustomservice.log" for example
    - "/var/log/secure"
    - "/var/log/secure.*"
  document_type: auth
  fields:
    service: sshd
  fields_under_root: true
  include_lines: ["Failed", "Did not receive", "Invalid user", "Connection closed
by", "PAM"]
  exclude_lines: ["keyboard"]
  exclude_files: ['\.gz$']

#================================ General =====================================

#================================ Outputs =====================================


#----------------------------- Logstash output --------------------------------
output.logstash:
  hosts : ["10.10.10.10:5044"]

  # List of root certificates for HTTPS server verifications
  #ssl.certificate_authorities: ["/etc/pki/root/ca.pem"]

  # Certificate for SSL client authentication
  #ssl.certificate: "/etc/pki/client/cert.pem"

  # Client Certificate Key
  #ssl.key: "/etc/pki/client/cert.key"
```

9.1.3 Nginx

```
######################## Filebeat NGINX Configuration ##########################

#========================== Filebeat prospectors =============================

filebeat.prospectors:

- input_type: log
  paths:
    - "/var/log/nginx/error.log"
    - "/var/log/nginx/access.log"
  document_type: webserver
  fields:
    service: nginx
  fields_under_root: true

#================================ General =====================================

# The name of the shipper that publishes the network data. It can be used to group
# all the transactions sent by a single shipper in the web interface.
name: Nginx

#================================ Outputs =====================================


#----------------------------- Logstash output --------------------------------
output.logstash:
  hosts : ["10.10.10.10:5044"]

  # List of root certificates for HTTPS server verifications
  #ssl.certificate_authorities: ["/etc/pki/root/ca.pem"]

  # Certificate for SSL client authentication
  #ssl.certificate: "/etc/pki/client/cert.pem"

  # Client Certificate Key
  #ssl.key: "/etc/pki/client/cert.key"
```

9.1.4 Dovecot

```
######################## Filebeat dovecot configuration ##########################

#========================== Filebeat prospectors =============================

filebeat.prospectors:

- input_type: log
  # Two filenames are specified to cover CentOS and Ubuntu flavored OSes
  paths:
    - "/var/log/dovecot.log"
  document_type: mail
  fields:
    service: dovecot
  fields_under_root: true
  include-lines: ["auth failed"]
#================================ General =====================================

#================================ Outputs =====================================


#----------------------------- Logstash output --------------------------------
```

```
output.logstash:
  hosts : ["10.10.10.10:5044"]

  # List of root certificates for HTTPS server verifications
  #ssl.certificate_authorities: ["/etc/pki/root/ca.pem"]

  # Certificate for SSL client authentication
  #ssl.certificate: "/etc/pki/client/cert.pem"

  # Client Certificate Key
  #ssl.key: "/etc/pki/client/cert.key"
```

9.2 Elasticsearch

9.2.1 Curator

```
# Change all "netmode" references to your district's name
actions:
  1:
    action: create_index
    description: "Creating monthly index"
    options:
      name: '<netmode-logstash-monthly-{now/M-1M{YYYY.MM}}>'
  2:
    description: "Reindex this month's indices into a monthly index"
    action: reindex
    options:
      wait_interval: 9
      max_wait: -1
      request_body:
        source:
          index: '<netmode-logstash-{now/M-1M{YYYY.MM}}.*>'
        dest:
          index: '<netmode-logstash-monthly-{now/M-1M{YYYY.MM}}>'
    filters:
      - filtertype: none
  3:
    description: "Deleting all daily indices for last month"
    action: delete_indices
    filters:
    - filtertype: pattern
      kind: prefix
      value: 'netmode-logstash-'
    - filtertype: age
      source: name
      direction: older
      timestring: '%Y.%m.%d'
      unit: days
      unit_count: 30
```

9.3 Logstash

9.3.1 Main Config

```
input {
          beats {
                    port => 5044
          }
          syslog {
                    type => "syslog"
                    port => 57889
          }
}
filter {
          mutate {
                    # TODO: Remove hardcoded district and add an environment variable.
                    add_field => [ "district", "netmode" ]
          }
          if [type] == "syslog" {
                    if "_grokparsefailure_sysloginput" in [tags] {
                    mutate { remove_tag => "_grokparsefailure_sysloginput" }
                    }
                    # Preprocess syslog events in order to follow the normal pipeline
                    mutate { add_field => [ "service" , "sshd" ] }
                    mutate { replace => { "type" => "auth" } }
                    mutate {
                              remove_field => ["priority", "facility","facility_label",
"severity", "severity_label", "timestamp"]
          }
          mutate { gsub => [ "message", "<38>", ""] }

          }
          if [service] == "nginx"{
                    grok {
                              patterns_dir => ["/opt/logstash/patterns"]
                              match => {"message" => ["%{NGINXACCESS}"]}
                              add_tag => "NGINXACCESS"
                    }
                    if "_grokparsefailure" in [tags] {
                              mutate { add_tag => "MALFORMED_REQUEST" }
                              mutate { remove_tag => "_grokparsefailure" }
                              grok {
                                        patterns_dir => ["/opt/logstash/patterns"]
                                        match => {"message" => "%{NGINXMALFORMED}"}
                              }
                    }
                    mutate { add_field => { "received_at" => "%{@timestamp}" } }
                    date{
                              match => ["timestamp", "dd/MMM/YYYY:HH:mm:ss Z"]
                              locale => "en"
                    }
          }else if [service] == "sshd"{
                    if "message repeated" in [message] {
                              # Hold the number of repeats in a variable
                              # and discard the "message repeated" part
                              grok {
                                        match => {"message" => "message repeated
%{NUMBER:repeat} times: \[ "}
                                        add_tag => "SYSLOGREPEAT"
                              }
                              mutate {
                                        gsub => [
                                                  "message", "message repeated (?<![0-9.+-
])(?>[+-]?(?:(?:[0-9]+(?:\.[0-9]+)?)|(?:\.[0-9]+))) times: \[ ", ""
                                        ]
                              }
                    }
                    # Concerns machines that mainly operate on password auth
                    if "invalid user" in [message] {
                              grok{
```

```
                                        patterns_dir => ["/opt/logstash/patterns"]
                                        match => {"message" => "%{SSHINVALIDUSER}"}
                                        add_field => { "logline_type" => "SSHINVALIDUSER" }
                              }
                    }else if "Failed" in [message]{
                              grok {
                                        patterns_dir => ["/opt/logstash/patterns"]
                                        match => {"message" => "%{SSHFAILEDLOGIN}"}
                                        add_field => { "logline_type" => "SSHFAILEDLOGIN" }
                              }
                    }
                    # Concerns machines that only accept PKI auth methods
                    else if "Invalid" in [message] {
                              grok {
                                        patterns_dir => ["/opt/logstash/patterns"]
                                        match => {"message" => "%{SSHINVALIDUSERPKI}"}
                                        add_field => { "logline_type" => "SSHINVALIDUSERPKI"
}
                              }
                    }else if "Connection closed by" in [message] {
                              grok {
                                        patterns_dir => ["/opt/logstash/patterns"]
                                        match => {"message" => "%{SSHCONNCLOSEDPKI}"}
                                        add_field => { "logline_type" => "SSHCONNCLOSEDPKI"
}
                              }
                    # Concerns FreeBSD machines
                    }else if "illegal user" in [message] {
                              grok {
                                        patterns_dir => ["/opt/logstash/patterns"]
                                        match => {"message" => "%{SSHINVALIDUSERF}" }
                                        add_field => { "logline_type" => "SSHINVALIDUSERF" }
                              }
                    }else if "PAM" in [message] {
                              grok {
                                        patterns_dir => ["/opt/logstash/patterns"]
                                        match => {"message" => "%{SSHFAILEDLOGINF}" }
                                        add_field => { "logline_type" => "SSHFAILEDLOGINF" }
                              }
                    # Concerns banner grabbing techinques
                    }else if "Did not receive" in [message] {
                              grok {
                                        patterns_dir => ["/opt/logstash/patterns"]
                                        match => {"message" => "%{SSHBANNERGRAB}"}
                                        add_field => { "logline_type" => "SSHBANNERGRAB" }
                              }

                    }else { mutate{ add_tag => "_grokparsefailure"} }

                    mutate { add_field => { "received_at" => "%{@timestamp}" } }

                    date{ match => ["syslog_time", "MMM  d HH:mm:ss", "MMM dd HH:mm:ss"] }

          }else if [service] == "dovecot"{
                    grok {
                              patterns_dir => ["/opt/logstash/patterns"]
                              match => {"message" => ["%{DOVECOTFAILED}"]}
                              add_tag => "DOVECOTFAILED"
                    }
                    mutate { add_field => { "received_at" => "%{@timestamp}" } }

                    date{ match => ["syslog_time", "MMM  d HH:mm:ss", "MMM dd HH:mm:ss"] }

          # If an event does not match any known service, drop it.
          }else { drop {} }

          mutate { remove_field => ["syslog_time","timestamp", "input_type", "beat"] }
}

output {
          if "_grokparsefailure" in [tags] or  "_dateparsefailure" in [tags]{
                    file {
                              path => "/var/log/logstash/logstash.std.out"
                              codec => rubydebug
```

```
                }
        }else{
                elasticsearch{
                        template_overwrite => true
                        template => "/etc/logstash/elasticsearch-template.json"
                        index => "%{district}-logstash-%{+YYYY.MM.dd}"
                        hosts => ["localhost:9200"]
                }
        }
}
```

9.3.2 Default Elasticsearch Mappings

```
{
        "template" : "*-logstash-*",
        "settings" : {
          "index.refresh_interval" : "5s"
        },
        "mappings" : {
          "_default_" : {
            "_all" : {"enabled" : true, "omit_norms" : true},
            "dynamic_templates" : [ {
              "message_field" : {
                "match" : "message",
                "match_mapping_type" : "string",
                "mapping" : {
                  "type" : "string", "index" : "analyzed", "omit_norms" : true,
                  "fielddata" : { "format" : "disabled" }
                }
              }
            }, {
              "string_fields" : {
                "match" : "*",
                "match_mapping_type" : "string",
                "mapping" : {
                  "type" : "string", "index" : "analyzed", "omit_norms" : true,
                  "fielddata" : { "format" : "disabled" },
                  "fields" : {
                    "raw" : {"type": "string", "index" : "not_analyzed",
      "doc_values" : true, "ignore_above" : 256}
                  }
                }
              }
            }, {
              "float_fields" : {
                "match" : "*",
                "match_mapping_type" : "float",
                "mapping" : { "type" : "float", "doc_values" : true }
              }
```

```
      }, {
        "double_fields" : {
          "match" : "*",
          "match_mapping_type" : "double",
          "mapping" : { "type" : "double", "doc_values" : true }
        }
      }, {
        "byte_fields" : {
          "match" : "*",
          "match_mapping_type" : "byte",
          "mapping" : { "type" : "byte", "doc_values" : true }
        }
      }, {
        "short_fields" : {
          "match" : "*",
          "match_mapping_type" : "short",
          "mapping" : { "type" : "short", "doc_values" : true }
        }
      }, {
        "integer_fields" : {
          "match" : "*",
          "match_mapping_type" : "integer",
          "mapping" : { "type" : "integer", "doc_values" : true }
        }
      }, {
        "long_fields" : {
          "match" : "*",
          "match_mapping_type" : "long",
          "mapping" : { "type" : "long", "doc_values" : true }
        }
      }, {
        "date_fields" : {
          "match" : "*",
          "match_mapping_type" : "date",
          "mapping" : { "type" : "date", "doc_values" : true }
        }
      }, {
        "geo_point_fields" : {
          "match" : "*",
          "match_mapping_type" : "geo_point",
          "mapping" : { "type" : "geo_point", "doc_values" : true }
        }
      } ],
      "properties" : {
        "@timestamp": { "type": "date", "doc_values" : true },
        "@version": { "type": "string", "index": "not_analyzed",
```

```
        "doc_values" : true },
                "geoip"  : {
                  "type" : "object",
                  "dynamic": true,
                  "properties" : {
                    "ip": { "type": "ip", "doc_values" : true },
                    "location" : { "type" : "geo_point", "doc_values" : true },
                    "latitude" : { "type" : "float", "doc_values" : true },
                    "longitude" : { "type" : "float", "doc_values" : true }
                  }
                }
              }
            }
          }
        }
```

9.3.3 Logstash Custom Patterns

9.3.3.1 Dovecot

```
WORDASH (\w|-)+
MULT_WORDS ((\w+|:)(\s)?)+
USEROREMAIL %{USERNAME:user}(@%{HOSTNAME:domain})?
DOVECOTFAILED %{CISCOTIMESTAMP:syslog_time} %{WORD:protocol}-login: Info:
%{MULT_WORDS:status} \(auth failed, \d+ attempts\): (user=<%{USEROREMAIL}>,
)?method=%{WORDASH:method}, rip=%{IP:attacker_ip}, lip=%{IP:host_ip}
```

9.3.3.2 SSH

```
SSHPREFIX %{SYSLOGTIMESTAMP:syslog_time} %{HOSTNAME:host_target}
sshd\[%{BASE10NUM}\]:
SSHINVALIDUSER %{SSHPREFIX} Failed (none|password) for invalid user
(%{USERNAME:username}?) from %{IP:attacker_ip} port %{BASE10NUM:port} ssh2
SSHFAILEDLOGIN %{SSHPREFIX} Failed (none|password) for (%{USERNAME:username}?) from
%{IP:attacker_ip} port %{BASE10NUM:port} ssh2
SSHBANNERGRAB %{SSHPREFIX} Did not receive identification string from
%{IP:attacker_ip}
SSHINVALIDUSERPKI %{SSHPREFIX} Invalid user (%{USERNAME:username}?) from
%{IP:attacker_ip}
SSHCONNCLOSEDPKI %{SSHPREFIX} Connection closed by %{IP:attacker_ip} (port
%{BASE10NUM:port} )?\[preauth\]
SSHFAILEDLOGINF %{SSHPREFIX} error: PAM: authentication error for
(%{USERNAME:username}?) from %{IP:attacker_ip}
SSHINVALIDUSERF %{SSHPREFIX} error: PAM: authentication error for illegal user
(%{USERNAME:username}?) from %{IP:attacker_ip}
```

9.3.3.3 Nginx

```
NGUSERNAME [a-zA-Z\.\@\-\+_%]+
NGUSER %{NGUSERNAME}
NGINXACCESS %{IPORHOST:attacker_ip} %{NGUSER:ident} %{NGUSER:auth}
\[%{HTTPDATE:timestamp}\] "%{WORD:verb} %{URIPATHPARAM:request}
HTTP/%{NUMBER:httpversion}" %{NUMBER:response} (?:%{NUMBER:bytes}|-)
(?:"(?:%{URI:referrer}|-)"|%{QS:referrer}) %{QS:agent}
NGINXMALFORMED %{IPORHOST:attacker_ip} %{NGUSER:ident} %{NGUSER:auth}
\[%{HTTPDATE:timestamp}\] %{GREEDYDATA:malformed_req}
```