Ανοικτό Πανεπιστήμιο Κύπρου

Σχολή Θετικών και Εφαρμοσμένων Επιστημών

Μεταπτυχιακό Πρόγραμμα Σπουδών

Πληροφοριακά και Επικοινωνιακά Συστήματα

Μεταπτυχιακή Διατριβή

Μελέτη για Αντιμετώπιση κακόβουλου λογισμικού (APT και Ransomware) με χρήση Τεχνητής Νοημοσύνης.

Δημήτριος Καλλιμάνης

Επιβλέπων Καθηγητής

Δρ. Σταύρος Σιαηλής

Μάιος 2018

Ανοικτό Πανεπιστήμιο Κύπρου

Σχολή Θετικών και Εφαρμοσμένων Επιστημών

Μεταπτυχιακό Πρόγραμμα Σπουδών

Πληροφοριακά και Επικοινωνιακά Συστήματα

Μεταπτυχιακή Διατριβή

Μελέτη για Αντιμετώπιση κακόβουλου λογισμικού (APT και Ransomware) με χρήση Τεχνητής Νοημοσύνης.

Δημήτριος Καλλιμάνης

Επιβλέπων Καθηγητής

Δρ. Σταύρος Σιαηλής

Η παρούσα μεταπτυχιακή διατριβή υποβλήθηκε προς μερική εκπλήρωση των απαιτήσεων για απόκτηση μεταπτυχιακού τίτλου σπουδών στα Πληροφοριακά Συστήματα από τη Σχολή Θετικών και Εφαρμοσμένων Επιστημών του Ανοικτού Πανεπιστημίου Κύπρου.

Μάιος 2018

ΛΕΥΚΗ ΣΕΛΙΔΑ

Περίληψη

Στη σύγχρονη εποχή με τη ραγδαία τεχνολογική ανάπτυξη τόσο στους τομείς της Πληροφορικής όσο και των υπολοίπων επιστημών αναγεννήθηκαν νέες προκλήσεις και με πολύ εξειδικευμένη γνώση που χρειάζονται επίλυση. Ειδικότερα με την εξάπλωση της οικονομικής κρίσης σε παγκόσμια εμβέλεια και με την διάχυτη και ανυπολόγιστη ποσοτικά γνώση παρεχόμενη από το διαδίκτυο, προέκυψαν και προκύπτουν σχεδόν καθημερινά θέματα ασφάλειας που αφορούν όχι μόνο επιχειρήσεις, οργανισμούς και κυβερνητικές οργανώσεις αλλά και ακόμη τον πιο απλό χρήστη. Έφτασε λοιπόν η στιγμή που θα πρέπει να αναφέρουμε και να μελετήσουμε την ύπαρξη κακόβουλων λογισμικών καθώς και το ρόλο της τεχνητής νοημοσύνης στην αντιμετώπιση αυτών των νέων απειλών σε διεθνές επίπεδο. Καθώς οι συγγραφείς κακόβουλου λογισμικού χρησιμοποιούν όλο και πιο εξειδικευμένες τεχνικές αποφυγής του εντοπισμού οι υπάρχουσες Signatures-based λύσεις αποδεικνύονται ελλιπείς. Λύση σε αυτό έρχεται να δώσει η ανάλυση του κακόβουλου χρησιμοποιώντας αλγόριθμους που βοηθούν την επίτευξη μοντέλων μηχανικής μάθησης. Ιδιαίτερα σε μια περίοδο με απεριόριστη ελευθερία από το βαθύ και σκοτεινό δίκτυο("deep web" και "dark web") η χρήση του τομέα μηχανικής μάθησης στην αντιμετώπιση και ανάλυση κακόβουλου λογισμικού αποτελεί αναγκαιότητα εξαιτίας του μεγάλου όγκου των δεδομένων από κακόβουλα λογισμικά και ιούς. Επομένως λοιπόν, ο αυξανόμενος όγκος του κακόβουλου λογισμικού επιβάλλει την εύρεση λύσεων για την ταξινόμησή του, και καλύτερα την συσταδοποίησή του ώστε να δημιουργούνται μοντέλα να μαθαίνουν από αυτά με σκοπό την αντιμετώπισή τους η τουλάχιστον τον περιορισμό των συνεπειών που προκαλούν. Στην παρούσα διπλωματική θα δούμε αναλυτικά τις μορφές του κακόβουλου λογισμικού διευκρινίζοντας τα επίπεδα του διαδικτύου και παρουσιάζοντας σχετικές δουλειές με μοντέλα μηχανικής μάθησης θα προτείνουμε μια πλατφόρμα στατικής και δυναμικής ανάλυσης κακόβουλου λογισμικού, η οποία χρησιμοποιεί τεχνικές μηχανικής μάθησης τόσο για την ταξινόμησή του όσο και για την συσταδοποίησή του. Αρχικά, εξηγούνται τα είδη κακόβουλων λογισμικών, οι έννοιες τις δυναμικής και στατικής ανάλυσης και της μηχανικής μάθησης . Στη συνέχεια, παρουσιάζονται οι προσεγγίσεις αντιμετώπισης και ανάλυσης κακόβουλων λογισμικών παρουσιάζοντας πλατφόρμες και τέλος, προτείνοντας μία από αυτές.

Summary

Nowadays, within a context of rapid technological development both in the fields of Information Technology and in Science, new challenges have emerged that require specific knowledge and know-how. At the same time, in times of global financial crisis, the diffuse and incomparable quantitative knowledge provided through the internet, security issues have emerged almost daily, involving not only businesses, governmental and other organizations but also the front-end user. It is, therefore, high time to report and study the existence of malicious software as well as the role of artificial intelligence in dealing with these new threats at international level. While malware writers use increasingly specialized anti-detection techniques, existing signature-based solutions prove to be incomplete. Solution to this comes from malicious analysis using algorithms that help develop engineering learning models. Especially in a period of unlimited freedom from the “deep and dark” web, the use of engineering the management and analysis of malicious software is a necessity that comes in view of a large amount of data from malicious software and viruses. Therefore, the growing volume of malware requires finding solutions for its classification, and its clustering so as to create models that will enable us to combat malicious software or, at least, limit the consequences it causes. In the current master’s thesis, we analyze the forms of malware by clarifying the levels of internet and, by presenting related work on models of machine learning, we will propose a platform for static and dynamic analysis of malware that uses engineering techniques for both classifying and aggregation. Initially, we explain the types of malware and the concepts of dynamic and static analysis, and machine learning. Next, we present the approaches to addressing and analyzing malicious software by presenting potential platforms. We finally propose one of these platforms as the best approach analysis and detecting malwares via unsupervised machine learning.

Ευχαριστίες

Σε σχέση με τη Διπλωματική Εργασία, θα ήθελα να ευχαριστήσω θερμά τον επιβλέποντα Καθηγητή μου Δρ. Σταύρο Σιαηλή, προς τον οποίο αισθάνομαι ευγνώμων, τόσο για την ουσιαστική βοήθεια και καθοδήγησή του στην υλοποίηση της Διπλωματικής Εργασίας, όσο και για το γεγονός ότι δε μού μετέδωσε απλώς τεχνικές και επιστημονικές γνώσεις, αλλά μού εμφύσησε έναν νέο τρόπο σκέψης και προσέγγισης των πραγμάτων.

Επιπλέον θα ήθελα να ευχαριστήσω την οικογένειά μου που με στήριξε κατά την διάρκεια της διπλωματικής όσο και κατά την διάρκεια της πρακτικής μου στο εξωτερικό.

Περιεχόμενα

Κεφάλαιο 1
Εισαγωγή

Στη σύγχρονη εποχή με τη ραγδαία τεχνολογική ανάπτυξη τόσο στους τομείς της Πληροφορικής όσο και των υπολοίπων επιστημών αναγεννήθηκαν νέες προκλήσεις και με πολύ εξειδικευμένη γνώση που χρειάζονται επίλυση. Ειδικότερα με την εξάπλωση της οικονομικής κρίσης σε παγκόσμια εμβέλεια και με την διάχυτη και ανυπολόγιστη ποσοτικά γνώση παρεχόμενη από το διαδίκτυο, προέκυψαν και προκύπτουν σχεδόν καθημερινά θέματα ασφάλειας που αφορούν όχι μόνο επιχειρήσεις, οργανισμούς και κυβερνητικές οργανώσεις αλλά και ακόμη τον πιο απλό χρήστη. Έφτασε λοιπόν η στιγμή που θα πρέπει να αναφέρουμε και να μελετήσουμε την ύπαρξη των κακόβουλων λογισμικών καθώς και το ρόλο της τεχνητής νοημοσύνης στην αντιμετώπιση αυτών των νέων απειλών σε διεθνές επίπεδο.

Σε αυτό το σημείο αναφέρουμε κάποια πραγματικά γεγονότα όπως η χώρα του Ιράν που μολύνθηκε από το Stuxnet το 2010, η Flame το 2012 με αποτέλεσμα την καταστροφή ιρανικών φυγοκέντρων που εμπλουτίζουν το ουράνιο(Iran Says Malicious Software Hit Its Petrochemical Complexes, 2016) . Και τα δύο κακόβουλα προγράμματα χρησιμοποιούσαν ελαττώματα ασφαλείας zero-day που δεν εντοπίστηκαν από την ημέρα που εντοπίστηκαν τα κακόβουλα προγράμματα. Επιπλέον, το γεγονός ότι ήταν απόλυτα μυστική για τον χρήστη και τα συστήματα προστασίας από ιούς εγείρει το ερώτημα πόσο καιρό αυτοί οι ιοί, όπου εγκαταστάθηκαν στα συστήματα πριν από την ανακάλυψη και ποιες σημαντικές πληροφορίες διαρρεύσαν κατά τη διάρκεια αυτής της περιόδου. Προχωρώντας μπροστά, ένα πιο πρόσφατο παράδειγμα για να τονιστεί η σπουδαιότητα αυτής της έρευνας είναι η ανακάλυψη του εξαιρετικά προχωρημένου προγράμματος hack στις 25 Αυγούστου 2016 (Trident Zero-Day IOS Vulnerabilities Lead to Real-World Espionage ,2016). Αυτό το κακόβουλο πρόγραμμα έλαβε το όνομα Πήγασος και στοχεύει σε κινητές συσκευές. Η ζημιά είναι και πάλι απόλυτα μυστική για τον χρήστη όπως ήταν για το Stuxnet και τη Flame και θα μπορούσε να μολύνει οποιοδήποτε iPhone / iPad που είχε όλη την ασφάλεια ενημερώθηκε εγκατεστημένη λόγω του γεγονότος ότι είναι εξοπλισμένο με αξεπέραστα ελαττώματα ασφαλείας. Με αυτό το τρόπο θα μπορούσε να κλέψει και να παρακολουθήσει τα πάντα στο μολυσμένο τηλέφωνο. Από την άλλη πλευρά, τα κακόβουλα λογισμικά καθίστανται ολοένα και περισσότερο το κακόβουλο λογισμικό εκμετάλλευσης που χρησιμοποιείται από τους χάκερ για να κάνουν γρήγορο buck. Μια έρευνα που διεξήχθη από την εταιρία cyber-security της NCC Group βρήκε ότι σχεδόν το ήμισυ των 60 (NHS

Trusts υπέστη το τελευταίο έτος Pegasus Spyware: What You Need to Know Now. N.p., n.d. Web. 02 Sept. 2016).

Αυτά τα εξελιγμένα προγράμματα hack / ιοί δεν είναι μόνο για επιτραπέζιους υπολογιστές, όπως μπορεί κάποιος να συμπεράνει, αλλά για κάθε συσκευή που χρησιμοποιεί ένα λειτουργικό σύστημα και είναι συνδεδεμένο στο διαδίκτυο. Τα συστήματα προστασίας που είναι διαθέσιμα σήμερα δεν επαρκούν για την εξάλειψη αυτών των επιθέσεων, επομένως υπάρχει ανάγκη για νέες ευφυείς μεθόδους προκειμένου να μετριαστούν ή και να εξαλειφθούν αυτές οι αυξανόμενες απειλές. Επιπλέον, η νέα τάση που ονομάζεται Ίντερνετ των πραγμάτων (IoT) που αναπτύσσεται ταχέως είναι ένας άλλος σημαντικός παράγοντας που πρέπει να λάβουμε υπόψη καθώς θα διευκολύνει τη διάδοση της μόλυνσης.

Σκοπός μας είναι να προτείνουμε λύση σε αυτού του είδους τις απειλές με μηχανική μάθηση (Machine Learning) αναλύοντας στατικά αλλά και δυναμικά τέτοιου είδους κακόβουλα λογισμικά συγκίνοντας τα εργαλεία Cuckoo και Laika Boss και προτείνοντας αλγόρθμους μηχανικής μάθησης με βάση το συμπέρασμα της σύγκρισης για την δημιουργία μιας μελλοντικής πλατφόρμας.

Στην αρχή της εργασίας παρουσιάζεται η κατάσταση που επικρατεί σήμερα όσον αφορά το κακόβουλο λογισμικό.

Στη συνέχεια, έγινε αναφορά στις βασικές έννοιες που πραγματεύεται η εργασία και δόθηκαν γενικές πληροφορίες για αυτές. Αρχικά, δόθηκε ο ορισμός του κακόβουλου λογισμικού και κάποια ιστορικά στοιχεία για αυτό. Ακολούθησαν οι κατηγορίες στις οποίες χωρίζεται ανάλογα με τον τρόπο που μεταδίδεται και τις λειτουργίες που εκτελεί στο μολυσμένο σύστημα. Αναφέρθηκαν επίσης οι επικρατέστεροι τρόποι διάδοσης του κακόβουλου λογισμικού. Προχωρώντας παρακάτω αναλύθηκαν οι δύο μέθοδοι ανάλυσης κακόβουλου λογισμικού. Δόθηκαν οι ορισμοί τους, οι μορφές που μπορούν να πάρουν, οι τεχνικές με τις οποίες επιτυγχάνονται, καθώς και τα πλεονεκτήματα και μειονεκτήματα της κάθε μίας. Μετά, έγινε μια εισαγωγή στη μηχανική μάθηση, δίνοντας τον ορισμό της και εξερευνώντας τις κατηγορίες της και τους αλγορίθμους με τους οποίους υλοποιείται. Τέλος, έγινε αναφορά στη σχετική, με το θέμα της εργασίας, βιβλιογραφία.

Στο τρίτο κεφάλαιο δόθηκαν οι αρχιτεκτονικές προδιαγραφές που θα πρέπει να πληρή η πλατφόρμα. Συζητήθηκαν οι απαιτήσεις που υπάρχουν από αυτή (κλιμακωσιμότητα) και ο τρόπος με τον οποίο θα πρέπει να λειτουργεί ώστε να επιτυγχάνονται τα καλύτερα αποτελέσματα.

Έπειτα έχουμε συγκρίνει τα δύο εργαλεία ανάλυσης κακόβουλου λογισμικού που θα χρησιμοποιηθούν στην υλοποίηση της πλατφόρμας και βοήθησαν ώστε να καταλήξουμε σε ένα συμπέρασμα για τους αλγορίθμους μηχανικής μάθησης που μπορούν να χρησιμοποιηθούν.

Τέλος παρουσιάστηκαν τα σημαντικότερα κομμάτια του κώδικα της πλατφόρμας, ο οποίος χρησίμευσε στην διασύνδεση των μερών της και στην δημιουργία της διοχέτευσης.

Κεφάλαιο 2
Θεωρητικό υπόβαθρο

Σε αυτό το κεφάλαιο θα παρουσιαστούν συνοπτικά κάποιες βασικές έννοιες απαραίτητες για την ομαλή εισαγωγή στο αντικείμενο της εργασίας και την καλύτερη κατανόηση του. Ειδικότερα, θα αναλυθεί η έννοια του κακόβουλου λογισμικού, οι μέθοδοι για την ανάλυσή του και ορολογία από το πεδίο της μηχανικής Τέλος, θα γίνει αναφορά σε σχετικές εργασίες πάνω στις οποίες βασίστηκε η εργασία και οδηγήθηκε σε μια τελική άποψη όσον αφορά τα αποτελέσματα και το βαθμό αντιμετώπισης της χρήσης της τεχνητής νοημοσύνης στα κακόβουλα λογισμικά.

2.1 Έννοια και σημασία Κακόβουλου Λογισμικού

Ο απλός ορισμός και σύμφωνα με τους (Nutan Kumar Panda; Sudhanshu Chauhan, 2015) που μπορεί να προκύψει από αυτό είναι ότι οποιοδήποτε λογισμικό που εκτελεί κακόβουλη δραστηριότητα μπορεί να θεωρηθεί κακόβουλο λογισμικό. Υπάρχουν διάφοροι τύποι malwares με βάση τη συμπεριφορά τους. Τα διαφορετικά κακόβουλα προγράμματα έχουν διαφορετικές λειτουργίες και διαφορετικούς τρόπους διάδοσης της "μόλυνσης". Εάν σκεφτόμαστε να μολύνουμε ένα στοχευόμενο κοινό ή άτομο τότε η συλλογή πληροφοριών που σχετίζονται με αυτόν μπορεί να βοηθήσει πολύ. Αν γνωρίζουμε προσωπικά το θύμα μας, τότε μπορούμε να του παράσχουμε το λογισμικό του / της που έχει μολυνθεί από κακόβουλο λογισμικό απευθείας σε οποιαδήποτε συσκευή αποθήκευσης ή αποστέλλοντάς του / της μια σύνδεση λήψης από απόσταση. Εάν χρειαστεί να εκτελέσουμε την πρώτη φορά, τότε η συλλογή πληροφοριών για το λειτουργικό σύστημα και την εφαρμογή

ασφαλείας σε αυτό θα βοηθήσει πολύ. Τα περισσότερα από τα κακόβουλα προγράμματα προέρχονται από το διαδικτυακό τόπο όπου προσπαθούμε να αποκτήσουμε πρόσβαση σε ορισμένους περιορισμένους ιστότοπους, όπως ιστότοπους για ενηλίκους, δωρεάν μουσική ή ιστότοπους φιλοξενίας λογισμικού κ.λπ. Υπάρχουν διάφορες ταξινομήσεις κακόβουλου λογισμικού, ορισμένες από τις οποίες ορίζονται κατωτέρω.

2.1.1 Μορφές και είδη κακόβουλου λογισμικού

Ιός (Virus).

Είναι ένας όρος που λαμβάνονται από τον κανονικό ιό που επηρεάζει το άτομο και μπορεί να είναι ο λόγος για διάφορες ασθένειες. Ομοίως, ο ιός του υπολογιστή είναι ένας κακόβουλος κώδικας που, όταν εκτελείται σε ένα σύστημα, μολύνει το σύστημα και εκτελεί κακόβουλη δραστηριότητα όπως διαγραφή δεδομένων, καταστροφή μνήμης, προσθήκη τυχαίων δεδομένων κλπ. Η μόνη αδυναμία του ιού είναι ότι χρειάζεται να ενεργοποιηθεί από το θύμα-χρήστη για να εκτελεστεί. Εάν το σύστημά μας περιέχει ένα κακόβουλο λογισμικό που επηρεάζεται από τον ιό αν δεν το εγκαταστήσουμε στο σύστημά μας, δεν υπάρχει τίποτα που να φοβόμαστε. Για την αποφυγή μιας μόλυνσης από ιούς, χρησιμοποιήούμε ένα γνήσιο ενημερωμένο πρόγραμμα καταπολέμησης ιών (antivirus)(Nutan Kumar Panda; Sudhanshu Chauhan, 2015).

Trojan.

Ένα πολύ ενδιαφέρον κακόβουλο λογισμικό, συνήθως εμφανίζεται με τη μορφή διαφημίσεων όπως παράδειγμα ότι κερδίσαμε ένα iPhone, "κάντε κλικ εδώ" για να εφαρμόσετε και όλα ή σε δημοφιλή παιχνίδια ως δωρεάν, προσελκύει σε αυτό τον χρήστη και δημειουργεί ένα backdoor και θα παρέχει όλες τις ενέργειες των χρηστών στον εισβολέα. Έτσι για να εξαπλωθεί ένας Trojan, αν ο επιτιθέμενος επιλέξει μια δημοφιλής απαιτητική πληρωμένη εφαρμογή, παιχνίδι, ταινία ή τραγούδι τότε οι πιθανότητες να μολύνει περισσότερους χρήστες είναι πολύ πιθανή. Σε αυτό το σημείο θα πρέπει να αναφέρουμε τί είναι το Backdoor)(Nutan Kumar Panda; Sudhanshu Chauhan, 2015).

Backdoor.

Πρόκειται για κώδικα, ο οποίος παρακάμπτει τους υπάρχοντες μηχανισμούς ασφαλείας (κωδικοί πρόσβασης, κρυπτογράφηση) σε ένα σύστημα, πρόγραμμα ή συσκευή με σκοπό την απόκτηση πρόσβασης σε αυτό. Παρουσιάζει ομοιότητες με το bot όσον αφορά την απομακρυσμένη πρόσβαση, η διαφορά τους όμως έγκειται στο ότι τα bots αποτελούν πάντα τμήμα ενός μεγαλύτερου botnet (Ed Skoudis & Lenny Zeltser,2003) .

Bot.

Ένα bot θα μολύνει χιλιάδες, μερικές φορές ακόμη και εκατομμύρια, υπολογιστές και στη συνέχεια θα παραμείνει περιμένοντας οδηγίες. Έπειτα, ο έλεγχος των μολυσμένων

υπολογιστών θα πωλείται στον σκοτεινό διαδίκτυο στον υπερθεματιστή, ο οποίος θα μπορεί στη συνέχεια να καταγράφει πληκτρολογήσεις (όπως τα ονόματα χρήστη και τους κωδικούς πρόσβασης) από τους υπολογιστές, να έχει πρόσβαση σε backdoor σε αυτούς ή να εκτελεί κατανεμημένη άρνηση υπηρεσίας (DDoS) επιθέσεις που θα πληγούν τις υπηρεσίες Διαδικτύου και τους διακομιστές ιστού συγκεκριμένων εταιρειών με τόσο μεγάλη επισκεψιμότητα και σε τόσο παρατεταμένη περίοδο ότι οι διακομιστές θα αποτύχουν (David Harley; Robert S. Vibert; Ken Bechtel; Michael Blanchard; Henk K. Diemer; Andrew Lee; Igor Muttik; Bojan Zdrnja, 2011).

Είναι η μορφή κακόβουλου λογισμικού είναι το λογισμικό που συνήθως κρύβεται πίσω από ένα άλλο πρόγραμμα.Συνήθως εμφανίζεται σε προγράμματα που προσφέρονται δωρεάν ενώ πωλούνται στο εμπόριο(David Harley; Robert S. Vibert; Ken Bechtel; Michael Blanchard; Henk K. Diemer; Andrew Lee; Igor Muttik; Bojan Zdrnja, April 2011).

Keylogger.

Το Keylogger είναι ένα κομμάτι malware που συλλέγει όλες τις πληκτρολογήσεις και τις στέλνει το ίδιο στον εισβολέα. Έτσι, όταν ο χρήστης εισάγει οποιαδήποτε πιστοποίηση για οποιονδήποτε ιστότοπο, τα διαπιστευτήρια μπορούν να καταγράφονται και να αποστέλλονται πίσω στον εισβολέα και μπορούν αργότερα να χρησιμοποιηθούν από τον εισβολέα για την εξαγορά του λογαριασμού. Η σύσταση για αυτό είναι εάν πληκτρολογείτε τα διαπιστευτήρια για οποιοδήποτε site που σχετίζεται με συναλλαγή ή σχετίζονται με την αξία σε οποιαδήποτε κρίσιμη πληροφορία, χρησιμοποιείτε πάντα πληκτρολόγιο οθόνης)(Nutan Kumar Panda; Sudhanshu Chauhan, 2015).

Worm.

Είναι περίπου ότι και ο ιός ενός υπολογιστή: είναι ένα κακόβουλο πρόγραμμα - malware το οποίο έχει της φιλοσοφία της μετάδοσης και εξάπλωσής του σε διάφορα άλλα συστήματα. Η διαφορά είναι ότι το **worm** δεν μεταδίδεται τοπικά σε αρχεία, αλλά "κυκλοφορεί" σε κάποιο δίκτυο και μολύνει άλλους υπολογιστές και συστήματα που είναι συνδεμένα σε αυτό το δίκτυο (όπως για παράδειγμα ένα τοπικό δίκτυο, ή το internet). Έτσι αν κάποιος υπολογιστής ενός δικτύου έχει κολλήσει κάποιο τέτοιο, τότε είναι πολύ πιθανό αν υπάρχει μειωμένη ασφάλεια, να κολλήσουν και άλλοι υπολογιστές του δικτύου αυτού)(Nutan Kumar Panda; Sudhanshu Chauhan, 2015).

Adware-Spyware.

Σύμφωνα με τον (John Aycock. 2006) , ως spyware ορίζεται ένα πρόγραμμα το οποίο συλλέγει πληροφορίες από έναν υπολογιστή και τις μεταφέρει σε κάποιον τρίτο. Ως εκ τούτου, στην παρούσα εργασία ορίζουμε ως spyware, το λογισμικό που συλλέγει και μεταδίδει πληροφορίες από έναν υπολογιστή σε έναν άλλον, χωρίς τη γνώση του χρήστη ή συγκατάθεση του ιδιοκτήτη του. Συνήθως ο στόχος του spyware, είναι να συλλέγει προσωπικά δεδομένα όπως όνομα, διεύθυνση ηλεκτρονικού ταχυδρομείου,

στοιχεία πιστωτικών καρτών και κωδικούς πρόσβασης. Αυτές οι πληροφορίες στη συνέχεια αξιοποιούνται από τον επιτιθέμενο για την αλίευση ευαίσθητων πληροφοριών και για τη διενέργεια κακόβουλων πράξεων. Τα adware, διαφοροποιούνται ως προς τα spyware σε σχέση με τον τύπο των πληροφοριών που υποκλέπτουν. Συνήθως στοχεύουν στην άντληση πληροφοριών οι οποίες σχετίζονται με τους χρήστες και τις συνήθειές τους (John Aycock 2006) (π.χ. παρακολούθηση της αγοραστικής συμπεριφοράς των χρηστών κατά την περιήγηση στο διαδίκτυο και στη συνέχεια αποστολή ή εμφάνιση διαφημιστικών μηνυμάτων).

Ransomware.

Ransomware είναι η μορφή ψηφιακού εκβιασμού αφου αφαιρεί το δικαίωμα πρόσβασης του χρήστη-θύμα στα αρχεία του ή ακόμα και στο πληροφοριακό του σύστημα ζητώντας του λύτρα. Μπορεί να χωριστεί σε δύο κύριους τύπους και στη συνέχεια να υποδιαιρεθεί με βάση τις οικογένειες που αντιπροσωπεύουν.

Οι δύο κύριες μορφές ransomware είναι εκείνες που κρυπτογραφούν, διαψεύδουν ή αποκλείουν την πρόσβαση σε αρχεία και εκείνες που περιορίζουν την πρόσβαση ή αποκλείουν τους χρήστες από τα ίδια τα συστήματα. Αυτές οι απειλές δεν περιορίζονται σε κάποια συγκεκριμένη γεωγραφία ή λειτουργικό σύστημα και μπορούν να δράσουν σε οποιοδήποτε αριθμό συσκευών. Τα πάντα από τις συσκευές Android, τα συστήματα iOS ή τα συστήματα των Windows κινδυνεύουν από αυτό το είδος εκμετάλλευσης μέσω ransomware. Ανάλογα με τον στόχο, η μέθοδος συμβιβασμού της συσκευής μπορεί να είναι διαφορετική και οι τελικές ενέργειες που θα ληφθούν περιορίζονται από την ίδια την ικανότητα της συσκευής (Allan Liska; Timothy Gallo, 2016).

Η μορφή κακόβουλου λογισμικού Ransomware θεωρήθηκε ως μια σημαντική απειλή στην κορυφή της λίστας προβλέψεις στην ασφάλεια στον κυβερνοχώρο για το 2016 από τον Bogdan Dumitru(Bitdefender). Πολλοί ερευνητές από διάφορες εταιρίες αντιμετώπισης κακόβουλων λογισμικών αποκάλυψαν ότι ransomware ιοί απειλούσαν χρήστες Mac για πρώτη φορά, όπου διαπιστώθηκε ότι το είδος ιού ransomware εισβάλει ακόμα και σε λειτουργικά συστήματα όπως MAC και Linux (Palo Alto Networks, 2016).

Με οικονομικές απώλειες εκατοντάδων εκατομμυρίων και ίσως δισσεκατομμύριων, πολλές εταιρίες κακόβουλου λογισμικού αναζητούν συνεχώς την ανάπτυξη και την κατάρτισή τους για τον εντοπισμό νέων και άγνωστων απειλών. Κατασκευαστές αντιμετώπισης κακόβουλου λογισμικού υστερούν σε ειδική ανίχνευση ransomware, αλλά ερευνητές λένε ότι υπάρχουν νέοι τρόποι προσέγγισης αντιμετώπισης και ότι το πρόβλημα θα λυθεί. Όπως θα αναλύσουμε παρακάτω η τεχνητή Νοημοσύνη και μηχανική μάθηση είναι απαραίτητη για την καταπολέμηση ιών όπως τα ransomware που αποτελεί ένα τοπίο των απειλών που είναι μεγαλύτερο και πιο εξελιγμένο από ποτέ.

Αλγόριθμοι μηχανικής μάθησης έχουν τη δυνατότητα να βελτιώσουν σημαντικά το χρόνο ανίχνευσης για ransomware απειλές, καθώς είναι σε θέση να αναλύσουν μεγάλες ποσότητες δεδομένων πολύ πιο γρήγορα από ό, τι οποιαδήποτε ανθρώπινη παρακολούθηση ή παρέμβαση. Οι αλγόριθμοι μηχανικής μάθησης μπορούν να έχουν ένα υψηλό ποσοστό ανίχνευσης ακόμα και σε νέα ή άγνωστα δείγματα.

Η συγχώνευση της ανθρώπινης εφευρετικότητας σε συνδυασμό με την ταχύτητα μηχανικής μάθησης και την αμείλικτη ανάλυση των δεδομένων, μειώνει σημαντικά το χρόνο αντίδρασης κατά των νέων δείγματα ransomware, προσφέροντας προστασία ακόμα και από προηγμένα άγνωστα δείγματα ransomware. Ωστόσο, δεν σημαίνει ότι έχουμε πάντα έναν μόνο αλγόριθμο μηχανικής μάθησης που κάνει την ανίχνευση.

Ανίχνευση ransomware απαιτεί τη χρήση πολλών αλγορίθμων, κάθε ειδικεύεται στην ανίχνευση συγκεκριμένων οικογενειών ransomware με ιδιαίτερες ξεχωριστές συμπεριφορές. Αυτό αυξάνει σημαντικά τις πιθανότητες ανίχνευσης και την εμφάνιση ransomware ενώ μειώνει την ποσότητα των ψευδώς θετικών αποτελεσμάτων.

Με την κατάρτιση των αλγορίθμων μηχανικής μάθησης για μεγάλα σύνολα δεδομένων ransomware, μπορούμε να βοηθήσουμε στην εύρεση της λύσης για την ασφάλεια και πρόληψη νέων ή άγνωστων δειγμάτων ransomware από την κρυπτογράφηση αρχείων.

Μεταμορφικό και Πολυμορφικό κακόβουλο λογισμικό.

Στο σημείο αυτό δε θα πρέπει να παραλείψουμε κάποια ιδιαίτερα είδη όπως το μεταμορφικό και το πολυμορφικό κακόβουλο λογισμικό, τα οποία μπορούν εύκολα να παρακάμψουν την απλή στατική ανάλυση, αφού αλλάζουν τη δυαδική αναπαράσταση πάνω στην οποία βασίζεται. Για παράδειγμα, υπάρχει κακόβουλο λογισμικό το binary (δυαδική υπογραφή-ταυτότητα) του οποίου είναι κρυπτογραφημένο, αποκρυπτογραφείται μόνο στην μνήμη κατά την εκτέλεσή του (Michael Sikorski και Honig Andrew 2012). Επομένως λοιπόν καταλαβαίνουμε όπως έχουμε διατυπώσει και αναλύσει παρακάτω ότι τα τελευταία χρόνια είναι αναγκαία η δυναμική ανάλυση κακόβουλου λογισμικού ώστε να έχουμε όσο το δυνατόν μια ολοκληρωμένη εικόνα της τεχνικής συμπεριφοράς κακόβουλων λογισμικών.

2.1.2 Τρόποι λειτουργίας και δράσης κακόβουλου λογισμικού.

Ανεξάρτητα από το τι και πώς μολύνει σε ένα σύστημα, ο ιός πρέπει να εξασφαλίσει ορισμένες βασικές συνθήκες, προκειμένου να δράσει. Συγκεκριμένα, πρέπει να μπορεί να εκτελέσει τον κώδικά του και να εξασφαλίσει πρόσβαση σε μέσα αποθήκευσης (κύρια στο σκληρό δίσκο, αλλά όχι μόνο). Γι' αυτό το λόγο, πολλοί ιοί προσκολλώνται σε εκτελέσιμα (executable) αρχεία είτε του λειτουργικού συστήματος (Windows) είτε του κανονικού

λογισμικού ενός συστήματος. Εξασφαλίζουν έτσι δύο πράγματα: Πρώτον, ότι θα μπορούν να αναπαραχθούν και δεύτερον ότι θα μπορέσουν να εκτελέσουν τον κώδικά τους.

2.2 Ανάλυση Κακόβουλου Λογισμικού

Υπάρχουν δύο κοινές μεθοδολογίες της διαδικασίας ανάλυσης κακόβουλου λογισμικού που χρησιμοποιούνται συνήθως από τους αναλυτές κακόβουλου λογισμικού: στατική ανάλυση (ή ανάλυση κώδικα) και δυναμική ανάλυση (ή ανάλυση συμπεριφοράς). Αυτές οι δύο τεχνικές επιτρέπουν στους αναλυτές να κατανοούν γρήγορα και λεπτομερώς τους κινδύνους και τις προθέσεις ενός συγκεκριμένου δείγματος κακόβουλου λογισμικού.

Στατική Ανάλυση

Κατά τη διάρκεια της διαδικασίας στατικής ανάλυσης, δεν ενεργοποιείται το κακόβουλο λογισμικό. Γενικά, ο πηγαίος κώδικας των δειγμάτων κακόβουλου λογισμικού δεν είναι άμεσα διαθέσιμος. Γίνεται πρώτα αποσυμπίεση, και μετά την επιτυχή εκτέλεση της αντίστροφης μηχανικής, όπου γίνεται ανάλυση στον κώδικα αποσυναρμολόγησης χαμηλού επιπέδου. Οι περισσότεροι αναλυτές κακόβουλου λογισμικού εκτελούν μια στατική ανάλυση σε ένα προγενέστερο στάδιο της διαδικασίας ανάλυσης κακόβουλου λογισμικού, επειδή είναι ασφαλέστερο από ότι η δυναμική ανάλυση. Η πρόκληση στη στατική ανάλυση είναι η πολυπλοκότητα του σύγχρονου κακόβουλου λογισμικού, όπου ορισμένα από τα κακόβουλα προγράμματα εφαρμόζουν συστήματα αντιμετώπισης σφαλμάτων για την αποτροπή της ανάλυσης των επιμέρους τμημάτων του κώδικα από τους αναλυτές κακόβουλου λογισμικού.

Ένα γενικό πρόβλημα που αντιμετωπίζει η συγκεκριμένη ανάλυση είναι ότι πολλές από τις ερωτήσεις που αφορούν μία εφαρμογή και τις ιδιότητες της, παραμένουν συνήθως αναπάντητες. Επειδή τέτοια λογισμικά έχουν κατασκευαστεί κατευθείαν από κυβερνοεγκληματίες μπορεί να είναι δημιουργημένα σκόπιμα έτσι, ώστε να είναι δύσκολο να αναλυθούν. Συγκεκριμένα, ο επιτιθέμενος μπορεί να κάνει χρήση τεχνικών δυαδικής σύγχυσης (binary obfuscation) για να εμποδίσει τόσο την αποσυναρμολόγηση του κώδικα, όσο και την ανάλυση του, μεθόδους που χρησιμοποιούν οι τεχνικές στατικής ανάλυσης.

Αξιοσημείωτο είναι ότι ο κώδικας του κακόβουλου λογισμικού που αναλύεται με ένα στατικό αναλυτή δεν είναι απαραίτητα ο κώδικας που λειτουργεί πραγματικά. Όπως αναφέραμε και παραπάνω αυτό είναι πραγματικότητα για τα αυτοτροποποιούμενα κακόβουλα λογισμικά που χρησιμοποιούν πολυμορφικές και μεταμορφικές τεχνικές (Szor, 2005) κι ενωμένα εκτελέσιμα που ξαναεμφανίζονται κατά τη διάρκεια της εκτέλεσης (Oberhumer,M., 2004).

Δυναμική Ανάλυση

Η δυναμική ανάλυση (ανάλυση συμπεριφοράς) είναι μια διαδικασία ανάλυσης κακόβουλου λογισμικού κατά την εκτέλεση του ίδιου του κακόβουλου προγράμματος και παρατηρεί τη δραστηριότητα κακόβουλου λογισμικού. Παρατηρεί επίσης τις αλλαγές που συμβαίνουν όταν εκτελείται το κακόβουλο λογισμικό. Η μόλυνση ενός συστήματος με κακόβουλο λογισμικό μπορεί να είναι πολύ επικίνδυνη. Η εφαρμογή από κακόβουλο λογισμικό στο σύστημα μπορεί να προκαλέσει βλάβη, όπως διαγραφή αρχείου, αλλαγή στο μητρώο, τροποποίηση αρχείου, κλοπή εμπιστευτικών δεδομένων / πληροφοριών κλπ. Όταν πραγματοποιούμε ανάλυση κακόβουλου λογισμικού, χρειαζόματε ένα ασφαλές περιβάλλον και το δίκτυο δεν πρέπει να είναι συνδεδεμένο με άλλα συστήματα. Με δυναμική ανάλυση, μπορούμε να παρακολουθήσουμε τις αλλαγές που έγιναν στο σύστημα αρχείων, το μητρώο, τις διαδικασίες και την επικοινωνία στο δίκτυο. Το πλεονέκτημα της εκτέλεσης δυναμικής ανάλυσης είναι ότι μπορούμε να κατανοήσουμε πλήρως πώς λειτουργεί το κακόβουλο λογισμικό (Digit Oktavianto, Iqbal Muhardianto 2013).

2.3 Μηχανική Μάθηση

Η μάθηση μέσω προσωπικής εμπειρίας και γνώσης, η οποία διαδίδεται από γενιά σε γενιά, βρίσκεται στην καρδιά της ανθρώπινης νοημοσύνης. Επίσης, στην καρδιά κάθε επιστημονικού πεδίου βρίσκεται η ανάπτυξη μοντέλων (συχνά ονομάζονται θεωρίες) προκειμένου να εξηγηθούν τα διαθέσιμα πειραματικά στοιχεία σε κάθε χρονική περίοδο. Με άλλα λόγια, μαθαίνουμε πάντα από τα δεδομένα. Τα διαφορετικά δεδομένα και οι διαφορετικές επικεντρώσεις στα δεδομένα δημιουργούν διαφορετικούς επιστημονικούς κλάδους.

Η μηχανική μάθηση είναι η ανάπτυξη αποτελεσματικών αλγορίθμων για το σχεδιασμό των μοντέλων αλλά και για την ανάλυση και την πρόβλεψη. Το δεύτερο μέρος κερδίζει σημασία στην αυγή αυτού που ονομάζουμε μεγάλη εποχή δεδομένων, όταν κάποιος πρέπει να αντιμετωπίσει τεράστιες ποσότητες δεδομένων, οι οποίες μπορεί να εκπροσωπούνται σε χώρους μεγάλης διαστάσεως. Η ανάλυση δεδομένων για τέτοιες εφαρμογές θέτει απαιτήσεις σε αλγόριθμους ώστε να είναι υπολογιστικά αποδοτικές και ταυτόχρονα ισχυρές στις επιδόσεις τους, επειδή ορισμένα από αυτά τα δεδομένα είναι μολυσμένα με μεγάλο θόρυβο και επίσης, σε ορισμένες περιπτώσεις, τα δεδομένα μπορεί να έχουν ελλείπουσες τιμές.

Τέτοιες μέθοδοι και τεχνικές βρίσκονται στο επίκεντρο της επιστημονικής έρευνας για αρκετές δεκαετίες σε διάφορους κλάδους, όπως Στατιστική και Στατιστική Μάθηση, Αναγνώριση Προτύπων, Επεξεργασία και ανάλυση σημάτων και εικόνων, Επιστήμη Υπολογιστών, Εξόρυξη Δεδομένων, Μηχανή Όραμα, Βιοπληροφορική, Βιομηχανική Αυτοματοποίηση και Ιατρική Διάγνωση με Υπολογιστή, για να αναφέρουμε μερικές. Παρά τα διαφορετικά ονόματα, υπάρχει ένα κοινό σύνολο τεχνικών που χρησιμοποιούνται σε όλα αυτά, και θα αναφερθούμε σε τέτοιες μεθόδους όπως η μηχανική μάθηση. Αυτό το όνομα έχει κερδίσει δημοτικότητα την τελευταία δεκαετία περίπου. Το όνομα υποδεικνύει τη

χρήση μιας μηχανής / υπολογιστή για να μάθει ανάλογα με τον τρόπο με τον οποίο ο εγκέφαλος μαθαίνει και προβλέπει(Stergios Theodoridis,2015).

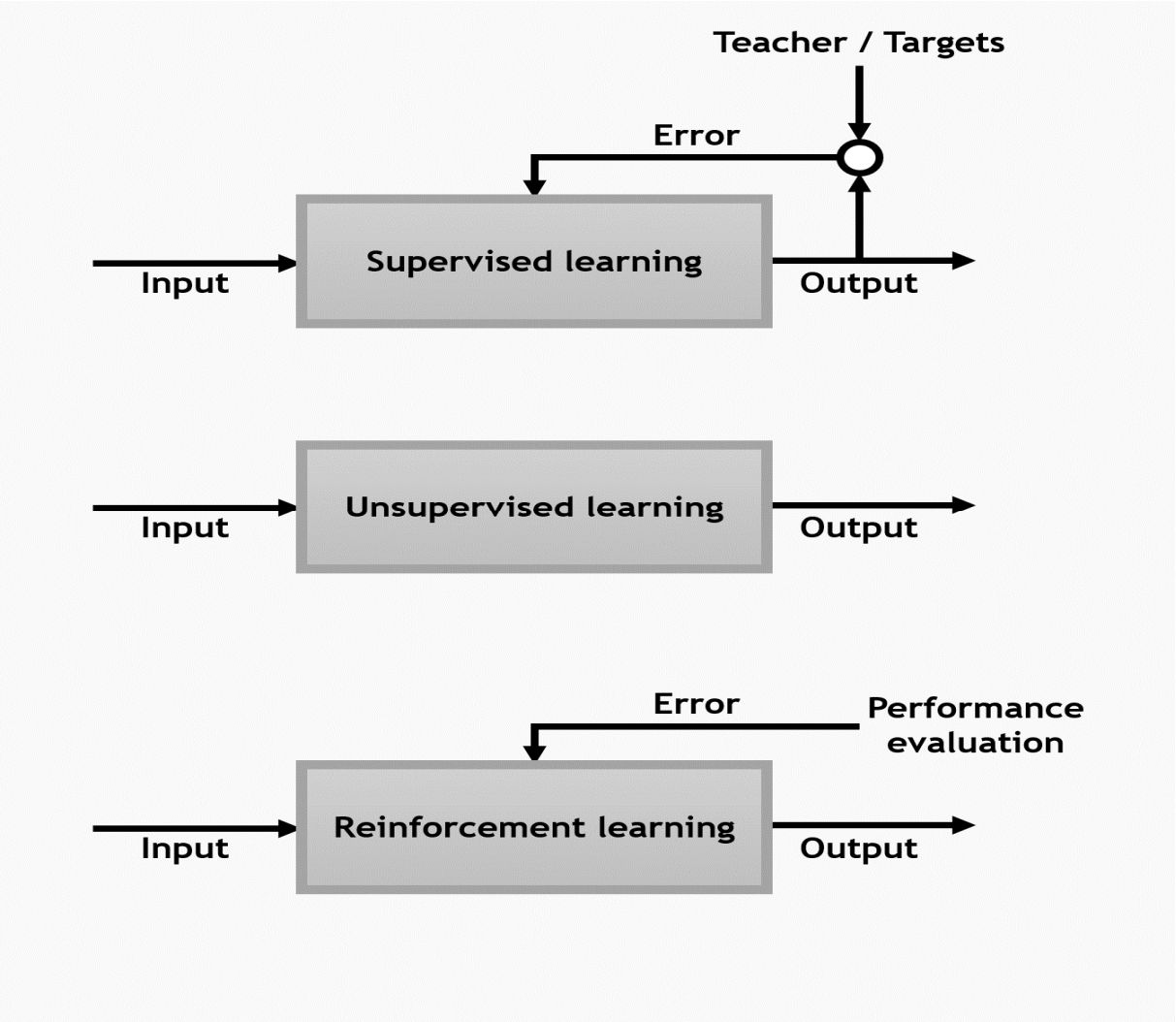
2.3.1 Κατηγορίες Μηχανικής Μάθησης.

Τα περισσότερα προβλήματα μηχανικής μάθησης ανήκουν σε μία από τις ακόλουθες τρεις κύριες κατηγορίες:

Στην εποπτευόμενη μάθηση, κάθε σημείο δεδομένων είναι συσχετισμένο με μια κατηγορία. Ένα παράδειγμα μιας ετικέτας κατηγοριοποίησης αποδίδει μια εικόνα ως γάτα ή σκύλο. Ένα παράδειγμα μιας ετικέτας αξίας είναι η τιμή πώλησης που σχετίζεται με ένα μεταχειρισμένο αυτοκίνητο. Ο στόχος της εποπτευόμενης μάθησης είναι να μελετηθούν πολλά παραδείγματα με ετικέτες όπως αυτά (τα αποκαλούμενα δεδομένα εκπαίδευσης) προκειμένου να γίνουν προβλέψεις σχετικά με τα μελλοντικά σημεία δεδομένων (αποκαλούμενα δεδομένα δοκιμών). Αυτές οι προβλέψεις έρχονται σε δύο γεύσεις, όπως ο προσδιορισμός νέων φωτογραφιών με το σωστό ζώο (που ονομάζεται πρόβλημα ταξινόμησης) ή η ανάθεση ακριβών τιμών πώλησης σε άλλα μεταχειρισμένα αυτοκίνητα (που ονομάζεται πρόβλημα παλινδρόμησης).

Στην μάθηση χωρίς επίβλεψη, τα σημεία δεδομένων δεν έχουν σχετικές ετικέτες. Αντ 'αυτού, ο στόχος ενός αλγόριθμου μάθησης χωρίς επίβλεψη είναι η οργάνωση των δεδομένων με κάποιο τρόπο ή η περιγραφή της δομής τους. Αυτό μπορεί να σημαίνει ομαδοποίηση τους σε ομάδες ή εύρεση διαφορετικών τρόπων αναζήτησης σύνθετων δεδομένων, ώστε να φαίνονται απλούστεροι.

Στη μάθηση ενίσχυσης, ο αλγόριθμος παίρνει να επιλέξει μια ενέργεια σε απόκριση σε κάθε σημείο δεδομένων. Πρόκειται για μια κοινή προσέγγιση στη ρομποτική, όπου το σύνολο των μετρήσεων αισθητήρων σε ένα χρονικό σημείο είναι ένα σημείο δεδομένων και ο αλγόριθμος πρέπει να επιλέξει την επόμενη ενέργεια του ρομπότ. Είναι επίσης φυσική εφαρμογή για εφαρμογές του Internet of Things, όπου ο αλγόριθμος μάθησης λαμβάνει ένα σήμα σε σύντομο χρονικό διάστημα στο μέλλον, υποδεικνύοντας πόσο καλή ήταν η απόφαση. Με βάση αυτό, ο αλγόριθμος τροποποιεί τη στρατηγική του για να επιτύχει την υψηλότερη αποτελεσματικότητα (Michael Beyeler,2017).



2.3.2 Αλγόριθμοι Μηχανικής Μάθησης

3.4.1Μηχανές Διανυσμάτων Υποστήριξης (Support Vector Machines(SVM))

Ένας από τους αλγόριθμους που σκεφτήκαμε να χρησιμοποιήσουμε για την πλατφόρμα είναι η μηχανή διανυσμάτων υποστήριξης(SVM) . Οι λόγοι για την επιλογή αυτή σχετίζονται κυρίως με την αποτελεσματικότητα και την ταχύτητα του αλγόριθμου. Ας τα πάρουμε όμως τα πράγματα από την αρχή.

Τι είναι ο αλγόριθμος SVM;

Είναι μια τεχνική που χρησιμοποιείται για κατηγοριοποίηση δεδομένων γενικότερα και εφαρμόζεται με πολύ μεγάλη επιτυχία στην κατηγοριοποίηση των αρχείων κειμένου και κατ' επέκταση των email την οποία αφορά το θέμα αυτής της εργασίας. Δημιουργήθηκε από τους Vapnik και Chervonenkis το 1992 και συγκαταλέγεται ανάμεσα στους πιο αποδοτικούς κατηγοριοποιητές καθώς έχει μια μοναδική ικανότητα να χειρίζεται ιδιαίτερα μεγάλα σύνολα χαρακτήρων όπως για παράδειγμα μεγάλα σε όγκο είδη κειμένου.

Οι μηχανές διανυσμάτων υποστήριξης (SVM) μπορούν να χειρίζονται δεδομένα ταξινομώντας τα αυτόματα σε κατηγορίες αποτελούμενες από αντικείμενα-πρότυπα(το κάθε πρότυπο αποτελείτε από ένα συγκεκριμένο αριθμό χαρακτηριστικών, απαραίτητη προϋπόθεση είναι να έχουν όλα τα πρότυπα σε κάθε πείραμα που πραγματοποιούμε ίδιο πλήθος χαρακτηριστικών αλλιώς θα προκύψει σφάλμα στο πείραμα), μέσω μιας διαδικασίας η οποία αποτελείτε από δύο (2) στάδια.

Αρχικά γίνεται η εκπαίδευση ανά δύο (2) κατηγορίες κάθε φορά (γίνεται εκπαίδευση δεδομένων τόσες φορές όσες είναι και οι δυνατοί συνδυασμοί που προκύπτουν μεταξύ των διαθέσιμων κατηγοριών που κατέχουμε) όπου ο αλγόριθμος μας "μαθαίνει" από τα δεδομένα(training set {80%}
) που του εισήχθησαν ούτως ώστε να μπορεί να γίνει εφικτή στο μετέπειτα στάδιο η αυτόματη κατηγοριοποίηση του εναπομείναντα μέρους των δεδομένων(testing set {20%}).
Να "μάθει" ο αλγόριθμος τι χαρακτηρίζει την κάθε κατηγορία, σε τι διαφέρει από τις άλλες κατηγορίες και πότε κάποιο αντικείμενο-πρότυπο θα ανήκει σε αυτή.

Ένα πρόβλημα που παρουσιάζεται είναι ότι ο αλγόριθμος κατατάσσει ένα δεδομένο σε μία(1) από δύο(2) διαφορετικές κατηγορίες, είτε σε αυτή που συμβολίζουμε με -1 ή σε εκείνη που συμβολίζουμε με 1, πράγμα που σημαίνει ότι σε κάθε ταξινόμηση που θα πραγματοποιείτε θα πρέπει να γίνεται πάντα μόνο ανά δύο(2) κατηγορίες. Αυτός ο τρόπος ταξινόμησης λέγεται δυαδική ταξινόμηση και με αυτό τον τρόπο δουλεύουμε εμείς στα πειράματα μας. Η διαδικασία αυτή γίνεται ανά δύο κατηγορίες κάθε φορά και θα γίνει τόσες φορές όσες είναι και οι μεταξύ των κατηγοριών δυνατοί συνδυασμοί (π.χ. αν έχουμε 4 κατηγορίες έχουμε τους συνδυασμούς: K1-K2, K1-K3, K1-K4, K2-K3, K2-K4, K3-K4). Ο διαχωρισμός των δεδομένων σε κατηγορίες γίνεται με βάση κάποιες ξεχωριστές ιδιαιτερότητες(χαρακτηριστικά) που έχει η κάθε κατηγορία από τις άλλες(π.χ.
έχουμε συνολικά εκατό (100) παραδείγματα όπου αυτά ανήκουν σε τέσσερις (4)
- 8 - διαφορετικές κατηγορίες και η κάθε κατηγορία αποτελείται από πέντε (5) κοινά χαρακτηριστικά, χρησιμοποιούμε διανύσματα επειδή το κάθε παράδειγμα αναπαριστάται ένα 5-
διάστατο διάνυσμα, δηλαδή αντιμετωπίζουμε κάθε παράδειγμα ως μια ακολουθία πέντε αριθμών. Άρα τα δεδομένα
μας μαζί με όλες τις πληροφορίες που μεταφέρουν δεν είναι για μας παρά ένας πίνακας 100x5(ένα διάνυσμα ανά γραμμή). Η επιτυχία της SVM τεχνικής αποδίδεται κυρίως στις δυνατές θεωρητικές βάσεις βασισμένες στην Vapnik-Chervonenkis (VC) θεωρία.

Υπάρχουν μερικοί περιορισμοί στον κυρίως (standard)
SVM κώδικα όπου μειώνουν την πρακτική χρησιμότητα του αλγορίθμου:

- Μη-γραμμικά μοντέλα μπορούν να μεγαλώσουν κατά πολύ σε μέγεθος, κάνοντας την διαδικασία ολοκλήρωσης αργή και μη-πρακτική.

•Ο
αλγόριθμος εκτελεί μια συγκεκριμένη λειτουργία, μόνο ανά δύο κατηγορίες μπορεί να χειριστεί.

Ως μια εναλλακτική λύση κάποιος μπορεί να λάβει υπόψη του αυτόνομες(stand-alone) ακαδημαϊκές εφαρμογές SVM (π.χ. SPIDER Matlab Toolbox, LibSVM, WEKA, TinySVM, SVMLight, SVMTorch, HeroSVM).

Η ταξινόμηση είναι ένα από
τα αντικείμενα που μας απασχολεί χρησιμοποιώντας τον αλγόριθμο γραμμικής ταξινόμησης SVM, αφού χρησιμοποιούμε αλγόριθμο ταξινόμησης σημαίνει ότι το σετ δεδομένων που θα εισάγεται πρέπει να έχει όλα τα δεδομένα ίδιο μέγεθος(ίσο πλήθος χαρακτηριστικών) και αφού ορίσουμε εμείς δικούς μας κανόνες που θα ταξινομηθούν τα δεδομένα θα πρέπει να έχουν ένα (1) τουλάχιστον χαρακτηριστικό το οποίο να είναι διαχωρίσιμο από το εκάστοτε ολόκληρο σετ δεδομένων.

Υπάρχουν δύο (2) τρόποι εκτέλεσης γραμμικής κατηγοριοποίησης: 1. Hard Margin
SVM όπου δεν επιτρέπεται καμία λάθος ταξινόμηση κατά την δημιουργία του μοντέλου πρόβλεψης(Εκπαίδευση (training)) 2. Soft Margin
SVM όπου επιτρέπονται μερικές λάθος ταξινομήσεις κατά την δημιουργία του μοντέλου πρόβλεψης(Εκπαίδευση (training)) και το μοντέλο πρόβλεψης θεωρείται ως σωστό. Στην παρούσα πτυχιακή εργασία χρησιμοποιούμε τον Hard Margin
SVM όπου δεν επιτρέπεται καμία λάθος ταξινόμηση κατά την δημιουργία του μοντέλου πρόβλεψης, το μοντέλο πρόβλεψης είναι το σημαντικότερο σημείο γιατί με βάση αυτό γίνεται η μετέπειτα αυτόματη ταξινόμηση των νέων «άγνωστων» παραδειγμάτων στο στάδιο της δοκιμής (test).

Ανάλογα με τον πυρήνα kernel που εκάστοτε χρησιμοποιείται δημιουργείτε και το ανάλογο υπερεπίπεδο, με ένα ίδιο σετ δεδομένων διαφορετικό υπερεπίπεδο θα κατασκευαστεί με τον linear kernel svm και διαφορετικό υπερεπίπεδο όταν χρησιμοποιηθεί π.χ. ο πολυωνυμικός kernel(Naiyang Deng; Yingjie Tian; Chunhua Zhang,2012).

Μέθοδοι πυρήνων (kernel methods):

Οι συναρτήσεις πυρήνα (kernel functions) είναι απεικονίσεις των διανυσμάτων εισόδου x στο σύνολο R, οι οποίες έχουν συγκεκριμένη μορφή και ιδιότητες και γενικεύουν σε μεγάλο βαθμό τις εφαρμογές των αλγορίθμων ταξινόμησης. Σύμφωνα με τη μέθοδο μετασχηματίζουμε κατάλληλα τα διανύσματα εισόδου ώστε να επιτύχουμε πιο εύκολη/γενικεύσιμη λύση του προβλήματος ταξινόμησης. Για να γίνει αυτό, πηγαίνουμε αρχικά από τον χώρο εισόδου (input space) σε έναν μετασχηματισμένο χώρο χαρακτηριστικών (feature space) με πιθανόν υψηλότερη διάσταση με μια διάσταση με μια

(μη γραμμική μη γραμμική) απεικόνιση απεικόνιση φ (x). Η συνάρτηση πυρήνα ορίζεται τότε ως: είναι με άλλα λόγια ένα εσωτερικό γινόμενο (inner product) μεταξύ των διανυσμάτων φ (x) και φ (x') στον καινούριο χώρο χαρακτηριστικών. Η πιο απλή συνάρτηση πυρήνα προκύπτει για τη μοναδιαία απεικόνιση φ (x)= x: είναι δηλ το εσωτερικό γινόμενο μεταξύ των 2 διανυσμάτων. Οι συναρτήσεις πυρήνα μπορούν να ερμηνευθούν ως ένα μέτρο της ομοιότητας (similarity) μεταξύ δύο διανυσμάτων στο χώρο εισόδο υ και ήδη χρησιμοποιήσαμε κάποιες από αυτές για τη μη παραμετρική εκτίμηση κατανομών πιθανότητας.

Κάποιες συναρτήσεις πυρήνα που χρησιμοποιούνται συχνά είναι:

• Πολυωνυμικές

• Ανομοιογενείς πολυωνυμικές

• Γκαουσιανές / radial basis function kernels

• Υπερβολική εφαπτομένη: σημείωση – η συνάρτηση αυτή δεν είναι θετικά ορισμένη αλλά έχει δώσει καλά αποτελέσματα στην πράξη.

Επιλογή Πυρήνων(Kernels).

a. Γραμμικός

b. Πολυώνυμος

c. Gaussian (RBF)

d. Sigmoid

Όσον αφορά την επιλογή των πυρήνων εξαρτάται από την εκάστοτε χρήση και τα αποτελέσματα που δίνει ο καθένας.

3.4.2 Αναγνώριση Προτύπων - Νευρωνικά Δίκτυα (Neural Networks)

Ο άνθρωπος ξεχωρίζει από τα διάφορα άλλα όντα(ζώα) στην νοημοσύνη χάρις στον εγκέφαλο που διαθέτει ο οποίος αποτελείτε από διάφορα τμήματα, ο εγκέφαλος έχει πάρα πολλούς νευρώνες όπου αναπτύσσονται - δημιουργούνται κατά πολύ στην παιδική ηλικία και λιγότερο στο υπόλοιπο της ζωής του. Ο εγκέφαλος ενός ανθρώπου είναι ένα σύνθετο σε πολυπλοκότητα, μη γραμμικό και με παράλληλη δομή σύστημα επεξεργασίας πληροφοριών. Αρκετοί επιστήμονες προσπαθούσαν να αντιγράψουν τον ανθρώπινο εγκέφαλο δημιουργώντας τεχνητά νευρωνικά δίκτυα (ΤΝΔ), πράγμα το οποίο είναι αδύνατο να πραγματοποιηθεί. Συγκρίνοντας την δομή και

την πολυπλοκότητα του ανθρώπινου εγκεφάλου οι νευρώνες που χρησιμοποιούμε για κατασκευή νευρωνικού δικτύου είναι “πρωτόγονοι”. Ο ανθρώπινος εγκέφαλος έχει περίπου 100 δισεκατομμύρια νευρώνες και ο κάθε νευρώνας περίπου έχει 1000 συνάψεις πράγμα που μας δίνει να καταλάβουμε το πόσο δύσκολο είναι να αντιγραφεί ο εγκέφαλος σε μορφή ΤΝΔ. Γίνεται ταυτόχρονος παραλληλισμός της επεξεργασίας από τους νευρώνες και ακολούθως γίνεται η κατανομή της πληροφορίας στα διάφορα μέρη του εγκεφάλου.

Νοημοσύνη:

• Η αναγνώριση εικόνων.

• Η αναγνώριση φωνής.

• Η λήψη αποφάσεων.

• Η κατάστρωση στρατηγικής.

• Η λογική, η ανάπτυξη επιχειρημάτων.

• Η μάθηση και η αυτοπροσαρμογή.

• Η μνήμη.

• Η αυτόματη πλοήγηση στο χώρο.

Η αναγνώριση προτύπων από μηχανές μπορούν να αντικαταστήσουν ποικίλες επίπονες ανθρώπινες εργασίες παρέχοντας ταχύτατα αποτελέσματα (π.χ. αναγνωρίζοντας πρόσωπα από φωτογραφίες οι οποίες είναι αποθηκευμένες σε ένα Η/Υ για ταυτοποίηση προσώπων). Ένα άλλο παράδειγμα αναγνώρισης προτύπων από μηχανές είναι αναγνωρίζοντας φωνές ατόμων από τηλεφωνικές συνδιαλέξεις για ταυτοποίηση φωνής. Επίσης σε μία πανεπιστημιακή βιβλιοθήκη μπορούν να ψηφιοποιηθούν όλα τα βιβλία της σε ηλεκτρονική μορφή με κωδικοποίηση κατά ASCII από μία μηχανή που σαρώνει(scanning) τα βιβλία αυτά υπο μορφή εικονοστοιχείων (pixels) αποθηκεύοντας τα βιβλία στον σκληρό δίσκο ενός Η/Υ(FTP Server) χρησιμοποιώντας και ένα πρόγραμμα OCR(Optical Character Recognition). Τα βιβλία με αυτόν τον τρόπο μετατρέπονται σε ηλεκτρονική μορφή και παραμένουν αναλλοίωτα από την πάροδο του χρόνου από τυχόν φθορές που μπορούν να συμβούν στα βιβλία(σκισίματα, μουντζούρες). Τέλος όταν τα βιβλία ψηφιοποιηθούν σε μορφή .pdf θα μπορεί να γίνεται ταχύτατη αναζήτηση σε αυτά μέσω του εύχρηστου κουμπιού αναζήτησης(search) ενός pdf προγράμματος πετυχαίνοντας έτσι γρήγορη πρόσβαση σε

συγκεκριμένα σημεία ενός βιβλίου που ενδιαφέρουν τον φοιτητή με επαγγελματισμό και χωρίς να χρονοτριβή.

Οι δενδρίτες είναι είσοδοι, δέχονται ηλεκτρικά σήματα από άλλους νευρώνες. Οι συνάψει ς είναι σημεία ένωσης μεταξύ διακλαδώσεων του άξονα ενός νευρώνα και των δενδριτών από άλλους νευρώνες. Ο άξονας είναι έξοδος και έχει μήκος από 1mm έως > 1m, στέ λνει ηλεκτρικούς παλμούς σταθερού πλάτους αλλά μεταβλητής συχνότητας. Σε δύο βασι κές κατηγορίες χωρίζονται τα νευρωνικά δίκτυα:

1. Σε δίκτυα εμπρόσθιας διάδοσης(feedforward networks) και

2. Ανατροφοδοτούμενα δίκτυα(recurrent networks)

Στην αναγνώριση προτύπων γίνεται κατηγοριοποίηση προτύπων στις σωστές κατηγορίε ς αναπαριστώντας τα πρότυπα σε μορφή διανυσμάτων.

Λειτουργία Νευρωνικών Δικτύων

Ένα νευρωνικό δίκτυο, έχει δύο βασικές λειτουργίες:

• Εκπαίδευση.

• Πρόβλεψη.

Το πρώτο στάδιο είναι η εκπαίδευση του νευρωνικού δικτύου. Στο στάδιο αυτό δημιουργείται ένα σύνολο μάθησης (training set), δηλαδή ένα σύνολο από διανύσματα εισόδων και επιθυμητών εξόδων – αποτελεσμάτων. Τα διανύσματα αυτά ονομάζονται πρότυπα εκπαίδευσης. Χρησιμοποιώντας το σύνολο μάθησης και κατάλληλο αλγόριθμο, το νευρωνικό δίκτυο εκπαιδεύεται, δηλαδή υπολογίζει τα βάρη του (και τις πολώσεις του, εφόσον υπάρχουν). Τελικός σκοπός της εκπαίδευσης του νευρωνικού δικτύου είναι η ελαχιστοποίηση του σφάλματος πρόβλεψης στο σύνολο μάθησης. $e = |y - y|$

Ένας απλοϊκός τρόπος υπολογισμού του σφάλματος φαίνεται στην παραπάνω εξίσωση. Όπου, e το σφάλμα εξόδου του νευρωνικού. Με y συμβολίζεται η προβλεπόμενη τιμή – έξοδος, από το νευρωνικό δίκτυο και y είναι η πραγματική – επιθυμητή τιμή, σύμφωνα με το σύνολο μάθησης. Μετά την εκπαίδευση του νευρωνικού δικτύου ακολουθεί το στάδιο της πρόβλεψης. Στο στάδιο αυτό δημιουργείται ένα σύνολο ελέγχου (test set), δηλαδή ένα σύνολο από πρότυπα ελέγχου. Στο στάδιο αυτό δίνονται μόνο τα διανύσματα εισόδου στο νευρωνικό δίκτυο και αυτό υπολογίζει τα προβλεπόμενα διανύσματα εξόδου. Ο υπολογισμός αυτός γίνεται, χρησιμοποιώντας τις τιμές των βαρών (και των πολώσεων, εφόσον υπάρχουν) που υπολογίστηκαν κατά το στάδιο της εκπαίδευσης. Το σφάλμα πρόβλεψης στο σύνολο ελέγχου προκύπτει από το σφάλμα των προβλεπόμενων εξόδων

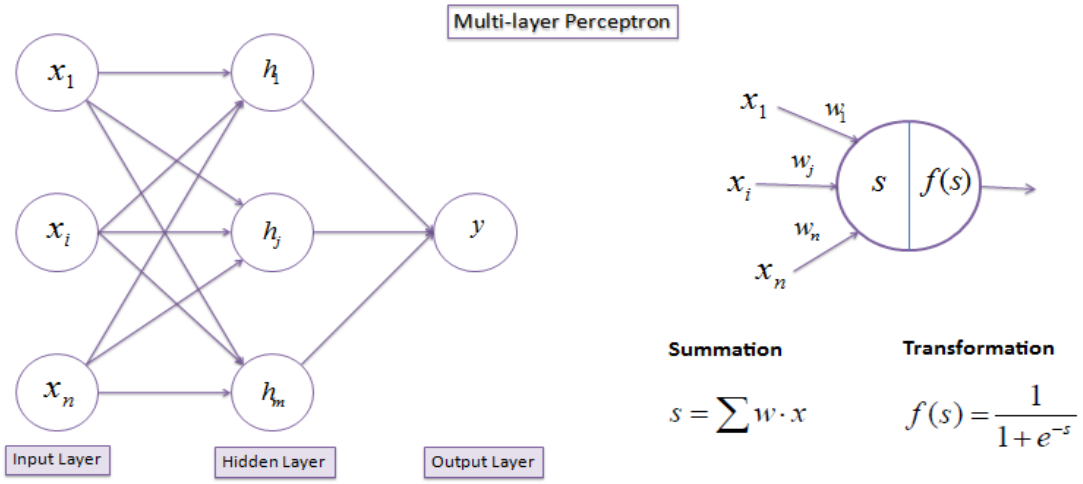
του νευρωνικού δικτύου ως προς τις επιθυμητές εξόδους για κάθε ένα από τα πρότυπα ελέγχου. Για τη βέλτιστη λειτουργία του νευρωνικού δικτύου, θα πρέπει κατά τα δύο στάδια της λειτουργίας του να ληφθούν υπόψη ορισμένοι βασικοί παράγοντες που το χαρακτηρίζουν.

• Η δομή και ο καθορισμός της αρχιτεκτονικής του νευρωνικού δικτύου. Εξετάζεται ο αριθμός των κρυφών στρωμάτων, ο αριθμός των νευρώνων ανά στρώμα, η συνάρτηση ενεργοποίησης, ο αλγόριθμος εκπαίδευσης του νευρωνικού δικτύου, ο μέγιστος αριθμός επαναλήψεων του αλγορίθμου εκπαίδευσης, κτλ.

• Ο καθορισμός της κατάλληλης δομής των συνόλων εκπαίδευσης και ελέγχου του νευρωνικού δικτύου, για παράδειγμα ο βέλτιστος αριθμός των προτύπων εκπαίδευσης.

• Η ικανότητα γενίκευσης του νευρωνικού δικτύου, η οποία διασφαλίζεται όταν το νευρωνικό δίκτυο παρουσιάζει μικρό σφάλμα πρόβλεψης τόσο στο σύνολο εκπαίδευσης όσο και στο σύνολο ελέγχου. Η εκπαίδευση του νευρωνικού δικτύου είναι μια σχετικά χρονοβόρα επαναληπτική διαδικασία, ιδίως όταν τα πρότυπα εκπαίδευσης είναι πολλά και όταν επίσης είναι πολλοί οι νευρώνες των στρωμάτων εισόδου και εξόδου. Ο αριθμός των νευρώνων του στρώματος εισόδου είναι ίσος με τον αριθμό των μεταβλητών εισόδου του προβλήματος πρόβλεψης. Παρόμοια, ο αριθμός των νευρώνων του στρώματος εξόδου είναι ίσος με τον αριθμό των μεταβλητών εξόδου του προβλήματος πρόβλεψης. Ο αριθμός των νευρώνων του κρυφού στρώματος, πρέπει να προσδιοριστεί με επαναληπτικές δοκιμές, καθώς δεν υπάρχει γενική μέθοδος προσδιορισμού του. Αν οι κρυφοί νευρώνες είναι υπερβολικά λίγοι, το νευρωνικό δίκτυο δε μπορεί να μάθει τις πολύπλοκες σχέσεις μεταξύ εισόδων και εξόδων και ίσως αντιμετωπίσει πρόβλημα σύγκλισης κατά τη διάρκεια της εκπαίδευσης του. Αν ο αριθμός των κρυφών νευρώνων είναι υπερβολικά μεγάλος, η διαδικασία εκπαίδευσης θα διαρκέσει περισσότερο και ίσως επηρεάσει αρνητικά την ικανότητα γενίκευσης του νευρωνικού δικτύου. Ο αριθμός των νευρώνων του κρυφού στρώματος μεταβάλλεται για διαφορετικές εφαρμογές και συνήθως εξαρτάται από το μέγεθος του συνόλου εκπαίδευσης και τον αριθμό των νευρώνων του στρώματος εισόδου.

Το Perceptron είναι η απλούστερη μορφή Νευρωνικού δικτύου όπως φαίνεται και στο σχήμα , το οποίο χρησιμοποιείται για την ταξινόμηση ενός ειδικού τύπου προτύπων, που είναι γραμμικά διαχωριζόμενα (δηλαδή πρότυπα που βρίσκονται στις αντίθετες πλευρές ενός υπερεπιπέδου, το οποίο ορίζει τις περιοχές απόφασης) **(**Nazmul Siddique; Hojjat Adeli.2013).



2.5 Σχετικές Εργασίες

Η χρήση της μηχανικής μάθησης και ειδικότερα των αλγορίθμων ταξινόμησης των δειγμάτων κακόβουλου λογισμικού σε συνδυασμό εργαλεία ανάλυσης έχει απασχολήσει πολλούς ερευνητές και υπάρχουν αρκετές εργασίες και βιβλιογραφία, που πραγματεύονται το συγκεκριμένο θέμα.

Στις περισσότερες περιπτώσεις η ταξινόμηση είναι βασισμένη στην υπογραφή και στις τεχνικές που ταξινομείται σε δύο κατηγορίες: βασισμένη στην υπογραφή και στις τεχνικές που βασίζονται στη συμπεριφορά. Οι συγγραφείς (K. Rieck, P. Trinius, C.Willems, and T. Holz.2011) ακολουθούν μια παρόμοια γραμμή σκέψεων για την εξαγωγή χαρακτηριστικών, και χρησιμοποιούν μηχανή διανυσμάτων υποστήριξης SVM, για την ταξινόμηση των δειγμάτων. Σχετικά,οι συγγραφείς(Y. Park, D. Reeves, V. Mulukutla, and B. Sundaravel, 2010) χρησιμοποιούν γραφήματα συμπεριφοράς που ταιριάζουν για να ταυτοποιήσουν και να ταξινομούν οικογένειες δειγμάτων κακόβουλου λογισμικού, με υψηλό κόστος των λειτουργιών γραφημάτων. Οι συγγραφείς χρησιμοποιούν δυναμική ανάλυση για να μοντελοποιήσουν την ακολουθία των συναρτήσεων που κλήθηκαν ως γράφους και ως n-grams αντίστοιχα (Younghee Park ,2010 &P.V. Shijo και A. Salim,2015).

Αξιόλογο έγγραφο είναι η δημιουργία AMAL, ενός αυτοματοποιημένου εργαλείου ανάλυσης κακόβουλου λογισμικού που εκπονήθηκε από τους Aziz Mohaisen1 and Omar Alrawi (AMAL: High-Fidelity, Behavior-based Automated Malware Analysis and Classification), το οποίο αποτελείται από δύο υποσυστήματα, το AutoMal και MaLabel. Το AutoMal με μέθοδο μη επιβλεπόμενης μάθησης, εφαρμόζει με πολλαπλούς αλγόριθμους συσταδοποίσης την ομαδοποίηση των δειγμάτων. Μια αξιολόγηση τόσο των AutoMal όσο και του MaLabel σε μέτρια κλίμακα (4.000 δείγματα) και μεγάλης κλίμακας σύνολα δεδομένων (πάνω από 115.000 δείγματα) που συλλέχθηκαν και αναλύθηκαν από το AutoMal σε διάστημα 13 μηνών παρουσιάζουν τα αποτελέσματα της AMAL

αποτελεσματικότητα στον ακριβή χαρακτηρισμό, ταξινόμηση και ομαδοποίηση δειγμάτων κακόβουλου λογισμικού. Το MaLabel επιτυγχάνει ακρίβεια 99,5% και 99,6% για ορισμένες οικογένειες κακόβουλου λογισμικού, ταξινόμηση και περισσότερο από 98% ακρίβεια.

Μια ακόμη παρόμοια προσέγγιση με την δική μας, και με ιδιαίτερο ενδιαφέρον παρουσιάζει η δημιουργία αυτοματης ανάλυσης της συμπεριφοράς κακόβουλων λογισμικών του Konrad Rieck1, Philipp Trinius2, Carsten Willems2, and Thorsten Holz2 (2011) με την πλατφόρμα malheur η οποία είναι μια ενδιαφέρουσα προσέγγιση είναι η ανακάλυψη νέων κατηγοριών κακόβουλου λογισμικού με συσταδοποίηση (clustering) και με διάκριση μεταξύ γνωστών κατηγοριών (ταξινόμηση) να αλληλοσυμπληρώνονται διότι και οι δύο είναι απαραίτητες για αποδοτική και αποτελεσματική ανάλυση κακόβουλου λογισμικού. Έχει επιτευχθεί η αυξητική ανάλυση η οποία μειώνει τις απαιτήσεις μνήμης κατά 94% και παράγει έναν συντελεστή επιτάχυνσης επιτρέποντας τελικά την επεξεργασία 33.000 αναφορών συμπεριφοράς κακόβουλου λογισμικού σε λιγότερο από 25 λεπτά.

Εν κατακλείδη μια πολύ ενδιαφέρουσα προσέγγιση όσον αφορά την καλύτερη επιλογή ταξινόμησης με βάση τους αλγόριθμους μηχανικής μάθησης είναι στη δημοσίευση της Kateryna Chumachenko. Τα προβλήματα ταξινόμησης που χρησιμοποιούμε στην ανάλυση κακόβουλων λογισμικών,δώσανε σε διαφορετικά μοντέλα διαφορετικά αποτελέσματα. Το χαμηλότερον ακρίβεια επιτεύχθηκε με τον αλγόριθμο Naive Bayes (72,34% και 55%), ακολουθούμενο από τους "κοντινότερους γείτονες" και μηχανισμοί διάνυσμα υποστήριξης (87%, 94,6% και 87,6%,94,6% αντιστοίχως). Η υψηλότερη ακρίβεια επιτεύχθηκε με τα J48 και Random Forest μοντέλα, και ήταν ίσο με 96,8% όσον αφορά τη δυαδική ταξινόμηση.

Κεφάλαιο 3

Αρχιτεκτονική της Εφαρμογής

Πριν την υλοποίηση οποιουδήποτε συστήματος είναι απαραίτητο να καθοριστούν οι προ-διαγραφές του. Με αυτόν τον τρόπο η διαδικασία της υλοποίησης έχει έναν ξεκάθαρο στόχο και μπορεί να προχωρήσει πιο εύκολα σε σχεδιαστικές αποφάσεις. Στις επόμενες ενότητες θα αναλυθούν οι προδιαγραφές της πλατφόρμας καθώς και αυτές του κάθε δομικού της μέρους ξεχωριστά.

3.1 Η Πλατφόρμα

Ζητούμενο της εργασίας είναι να κατασκευαστεί μια πλατφόρμα, η οποία θα εκτελεί αυτοματοποιημένα στατική και δυναμική ανάλυση κακόβουλου λογισμικού. Επίσης, θα παρέχει τη δυνατότητα για ταξινόμησή του με βάση τα ευρήματα των δύο αναλύσεων. Η ταξινόμηση θα πρέπει να γίνεται με την βοήθεια της μηχανικής μάθησης. Προκείμενου να πετύχει τα παραπάνω θα πρέπει να έχει κάποια βασικά χαρακτηριστικά και μια συγκεκριμένη μέθοδο λειτουργίας.

3.1.1 Προδιαγραφές

Πρώτα απ' όλα, η πλατφόρμα θα πρέπει να είναι βασισμένη σε εργαλεία ανοικτού κώδικα (open source). Τα εργαλεία ανοικτού κώδικα είναι προγράμματα ή βιβλιοθήκες των οποίων ο πηγαίος κώδικας είναι δημόσιος και ελεύθερα προσβάσιμος σε όλους. Επίσης, τις περισσότερες φορές τα εργαλεία ανοικτού κώδικα συνοδεύονται από άδειες, οι οποίες επιτρέπουν την ελεύθερη επεξεργασία και αναδιανομή τους. Τα προγράμματα ανοικτού κώδικα παρουσιάζουν κάποια σημαντικά πλεονεκτήματα έναντι των πιο συνηθισμένων προγραμμάτων κλειστού κώδικα. Συνήθως είναι πιο ασφαλή και αξιόπιστα, καθώς ο αριθμός των προγραμματιστών που τα αναπτύσσουν είναι πολύ μεγαλύτερος και τα διάφορα σφάλματα στον κώδικα εντοπίζονται πολύ πιο γρήγορα. Ταυτόχρονα, ο συνδυασμός των διαφορετικών προσεγγίσεων των προγραμματιστών επιταχύνει την καινοτομία και παράγει νέες ιδέες. Επίσης, η διαδικασία ανάπτυξής τους επικεντρώνεται μόνο στα τεχνικά μέρη του εργαλείου χωρίς να υπάρχει ο σκοπός της κερδοφορίας, όπου με αυτό το τρόπο αλλάζει ο σκοπός και η πραγματική ποιότητα της δουλειάς. Το γεγονός ότι προσφέρονται δωρεάν τα κάνει ακόμα πιο ελκυστικά για χρήση σε μία διπλωματική εργασία. Τέλος, η ανοικτή πρόσβαση στον κώδικα επιτρέπει ευκολότερη παραμετροποίηση της πλατφόρμας και τυχόν μελλοντική της επέκταση.

Η πλατφόρμα θα πρέπει επιπλέον να είναι ικανή να ανταποκρίνεται σε αυξανόμενα φορτία με μόνο περιορισμό τις δυνατότητες του υποκείμενου υλικού (scalable). Αυτό είναι απαραίτητο καθώς σύμφωνα με μελέτες που έχουν γίνει από τις εταιρίες antivirus ο αριθμός του κακόβουλου λογισμικού συνεχώς αυξάνεται (McAfee Labs. Threats Report.). Ακόμα, το υπολογιστικό κόστος της δυναμικής ανάλυσης είναι μεγάλο, καθώς απαιτεί την (παράλληλη) εκτέλεση εικονικών μηχανών (virtual/machines/sandboxes). Επομένως, το υλικό της πλατφόρμας απαιτεί συνεχή αναβάθμιση και η πλατφόρμα θα πρέπει να είναι έτσι δομημένη, ώστε να μπορεί να την ακολουθεί.

Τέλος, είναι απαραίτητο η πλατφόρμα να μπορεί να πραγματοποιεί τόσο δυαδική κατηγοριοποίηση (binary classification), δηλαδή να διαχωρίζει τα δείγματα σε κακόβουλα και μη όσο και πολλαπλή (multiclass classification), όπου τα δείγματα κατηγοριοποιούνται σε οικογένειες κακόβουλου λογισμικού. Η πρώτη κατηγοριοποίηση είναι απαραίτητη σε περιπτώσεις όπου η πλατφόρμα αποτελεί μέρος ενός μεγαλύτερου συστήματος εντοπισμού

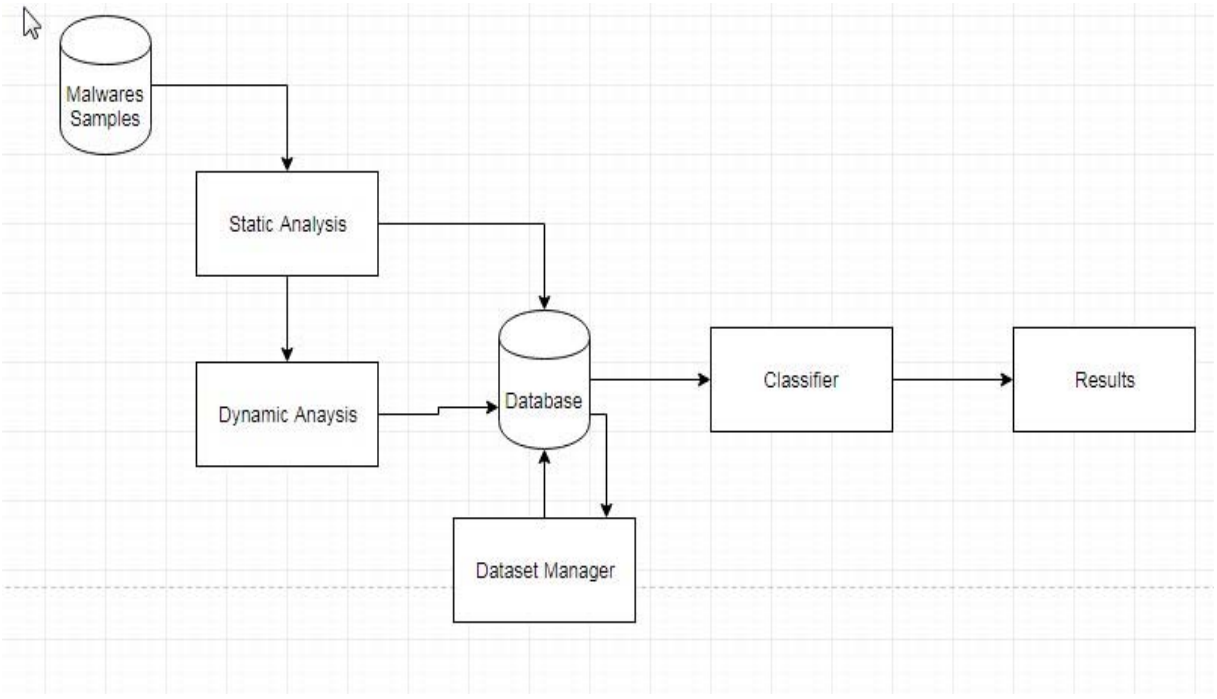
εισβολών (intrusion detection system). Σε αυτήν την περίπτωση, το σύστημα αναλύει την εισερχόμενη κίνηση στο δίκτυο ενός οργανισμού και τα ύποπτα αρχεία (συνημμένα σε email) προωθούνται στην πλατφόρμα, η οποία αρκεί να αποφανθεί αν είναι κακόβουλα ή όχι. Η δεύτερη είναι χρήσιμη όταν η πλατφόρμα χρησιμοποιείται σαν εργαλείο από τους αναλυτές κακόβουλου λογισμικού καθώς τους βοηθά να επικεντρωθούν μόνο στις οικογένειες κακόβουλου λογισμικού που τους ενδιαφέρουν. Αν, για παράδειγμα, μια εταιρία ενδιαφέρεται για κακόβουλο λογισμικό που στοχεύει τράπεζες, τότε μπορεί να εκπαιδεύσει την πλατφόρμα να αναγνωρίζει τέτοιου είδους κακόβουλο λογισμικό.

3.2 Καταστάσεις Λειτουργίας

Οι αλγόριθμοι επιβλεπόμενης μάθησης, όπως αναφέρθηκε και στο προηγούμενο κεφάλαιο έχουν δύο φάσεις, αυτή της εκπαίδευσης και αυτή της ταξινόμησης. Συνεπώς, αφού η πλατφόρμα χρησιμοποιεί αυτούς τους αλγορίθμους θα πρέπει να έχει και δύο καταστάσεις λειτουργίας. Η πρώτη είναι η κατάσταση εκπαίδευσης κατά την οποία η πλατφόρμα συλλέγει χαρακτηριστικά από τα δείγματα κακόβουλου λογισμικού προκειμένου να εκπαιδεύσει τους ταξινομητές. Για να γίνει αυτό, τα γνωστά δείγματα περνούν πρώτα από τη στατική ανάλυση και στη συνέχεια από τη δυναμική. Τα αποτελέσματα και των δυο αναλύσεων αποθηκεύονται μαζί με την κατηγορία κάθε δείγματος στη βάση δεδομένων, ώστε να χρησιμοποιηθούν στη συνέχεια για τη δημιουργία των datasets. Αυτή η κατάσταση λειτουργίας είναι offline και μπορεί να εκτελεστεί όταν η πλατφόρμα δεν δουλεύει.

Η δεύτερη είναι η κατάσταση ταξινόμησης σχήμα 1, η οποία είναι online και αποτελεί τη βασική λειτουργία της πλατφόρμας. Σε αυτή τα άγνωστα δείγματα αρχικά αναλύονται και στη συνέχεια γίνεται κατηγοριοποίησή τους με βάση τα ευρήματα των αναλύσεων. Ανάλογα με το είδος της ταξινόμησης υπάρχουν δύο εναλλακτικές. Αν η ταξινόμηση είναι δυαδική τότε στην αρχή γίνεται μόνο στατική ανάλυση και συνήθως αυτό είναι αρκετό για να εντοπιστούν συνηθισμένα δείγματα κακόβουλου λογισμικού. Σε περίπτωση όμως που το αποτέλεσμα της ταξινόμησης καταδεικνύει το δείγμα ως μη κακόβουλο, τότε αναλύεται και δυναμικά και η ταξινόμησή του επαναλαμβάνεται.

Σχήμα 1: Καταστάσεις Λειτουργίας



Η διαχείριση των datasets, όπου δημιουργήθηκε ένα πρόγραμμα γραμμής εντολών για την εύκολη δημιουργία datasets, το οποίο μπορεί να χρησιμοποιηθεί και σαν βιβλιοθήκη για ενσωμάτωση σε άλλο κώδικα που προτείνεται όπως θα δούμε παρακάτω για τον συνδυασμό TensorFlow και Keras. Ουσιαστικά αποτελείται από την κλάση DatasetManager που χρησιμοποιείται και από τις προαναφερθείσες κλάσεις. Το πραγματικό dataset δεν αποθηκεύεται στη βάση, αλλά δημιουργείται από την μέθοδο load() τη στιγμή που θα χρειαστεί. Στη βάση αποθηκεύεται μόνο μια λίστα με τα δείγματα που το αποτελούν. Αυτό γίνεται για εξοικονόμηση χώρου και αυξημένη ευελιξία.

Επίσης αν η ταξινόμηση είναι πολλαπλή, τότε τα δείγματα αναλύονται στατικά και δυναμικά. Η προσέγγιση αυτή χρησιμοποιείται για βελτίωση του χρόνου απόφασης για κάθε δείγμα κακόβουλου λογισμικού, καθώς και για εξοικονόμηση πόρων.

3.3 Δομικά Μέρη

Η δομή της πλατφόρμας υιοθετεί ένα συνδυαστικό μοντέλο και αποτελείται από επιμέρους ανεξάρτητα δομικά στοιχεία τα οποία διασυνδέονται μεταξύ τους σχηματίζοντας μια

διοχέτευση(pipeline) .Το μοντέλο αυτό επιτρέπει μεγαλύτερη ευελιξία κατά τη διαδικασία της ανάπτυξης και ανεξαρτησία μεταξύ των δομικών μερών. Τα μέρη της πλατφόρμας είναι ο στατικός και ο δυναμικός αναλυτής, η βάση δεδομένων, οι ταξινομητές και ένα ξεχωριστό εργαλείο γραμμής εντολών για την διαχείριση των datasets.

Ο στατικός αναλυτής (static analyzer) έχει σαν στόχο την ανάλυση του δείγματος χωρίς να προβεί σε εκτέλεσή του. Η ανάλυση επιτυγχάνεται μέσω της σάρωσης της δυαδικής αναπαράστασης (binary) του δείγματος με ειδικούς parsers για τον εκάστοτε τύπο αρχείου, ώστε να εξάγουμε αντιπροσωπευτικά χαρακτηριστικά. Τα αποτελέσματα της σάρωσης αφορούν κυρίως τη δομή του δείγματος και χωρίζονται σε δύο κατηγορίες. Η μια περιέχει χαρακτηριστικά κοινά για όλους τους τύπους αρχείων, όπως το μέγεθος και ο τύπος, ενώ η άλλη χαρακτηριστικά που διαφέρουν για κάθε τύπο, όπως τις επιμέρους ενότητες στις οποίες χωρίζεται ένα Windows PE ή τον αριθμό των σελίδων ενός PDF. Μέσα από την στατική ανάλυση εξάγονται επίσης και διάφορες στατικές υπογραφές, παρόμοιες με αυτές που χρησιμοποιούν τα antivirus.

Σημαντική απαίτηση από τον στατικό αναλυτή είναι η γρήγορη ταχύτητα εκτέλεσης των αναλύσεων, καθώς έτσι αυξάνεται η διεκπεραιωτή ικανότητα (throughput) της πλατφόρμας, ιδίως σε περιπτώσεις συνηθισμένου κακόβουλου λογισμικού όπως περιγράφηκε παραπάνω.

Επίσης, επιθυμητή είναι η δυνατότητα ανάλυσης μιας μεγάλης ποικιλίας τύπων αρχείων (Windows\PE, PDF, MS Office) και η δυνατότητα κλιμάκωσης (scalability).

Ο σκοπός του δυναμικού αναλυτή (dynamic analyzer) είναι η ανάλυση του δείγματος ενώ αυτό εκτελείται. Όσο η εκτέλεσή του δείγματος πραγματοποιείται σε κάποιο ασφαλές και απομονωμένο περιβάλλον (sandbox), ο αναλυτής συλλέγει διάφορα χαρακτηριστικά που αφορούν τη λειτουργία του δείγματος, όπως τον αριθμό των αρχείων και των εγγραφών του μητρώου (registry entries) που προσπελάστηκαν ή τον αριθμό των συνδέσεων δικτύου που εγκαθιδρύθηκαν.

Η κύρια προϋπόθεση που θα πρέπει να ικανοποιεί ο δυναμικός αναλυτής είναι να μην αποκαλύπτει στο αναλυόμενο πρόγραμμα ότι εκτελείται σε εικονικό περιβάλλον. Αυτό είναι απαραίτητο, καθώς πολλά κακόβουλα προγράμματα χρησιμοποιούν τεχνικές ανίχνευσης εικονικών μηχανών και αλλάζουν την συμπεριφορά τους όταν βρεθούν μέσα σε μία, ώστε να αποφύγουν το εντοπισμό. Επίσης θα πρέπει να υποστηρίζει την ανάλυση διαφόρων τύπων αρχείων και είναι κλιμακώσιμος (scalable).

Η βάση δεδομένων (Database) είναι το κεντρικό σημείο συλλογής της πληροφορίας της πλατφόρμας. Σε αυτήν αποθηκεύονται τα δείγματα προς ανάλυση καθώς και τα αποτελέσματα της στατικής και δυναμικής ανάλυσης. Επίσης, στη βάση δεδομένων βρίσκονται και τα datasets που χρησιμοποιούνται για την ταξινόμηση των δειγμάτων (Aziz

Mohaisen, Omar Alrawi και Manar Mohaisen,2015). Η βάση θα πρέπει να υποστηρίζει μεγάλο συνολικό όγκο δεδομένων καθώς και μεγάλες μεμονωμένες εγγραφές, αφού οι αναφορές αποτελεσμάτων της δυναμικής ανάλυσης μπορούν να γίνουν πολύ μεγάλες. Επίσης, θα πρέπει να μπορεί να εκτελεί γρήγορα τις διάφορες ενέργειες πάνω στις εγγραφές.

Η βιβλιοθήκη των ταξινομητών που θα επιλεγεί θα πρέπει πρώτα από όλα να παρέχει μια μεγάλη ποικιλία αλγορίθμων ικανών να εκτελούν και τα δύο είδη της ταξινόμησης (δυαδική, πολλαπλή). Σε δεύτερο επίπεδο οι αλγόριθμοι θα πρέπει να είναι υλοποιημένοι με τέτοιο τρόπο, ώστε να μπορούν να διαχειρίζονται μεγάλα datasets με καλές επιδόσεις και αξιόπιστα (stability).

Κεφάλαιο 4

Σύγκριση των επιμέρους εργαλείων ανάλυσης Cuckoo και Laika Boss

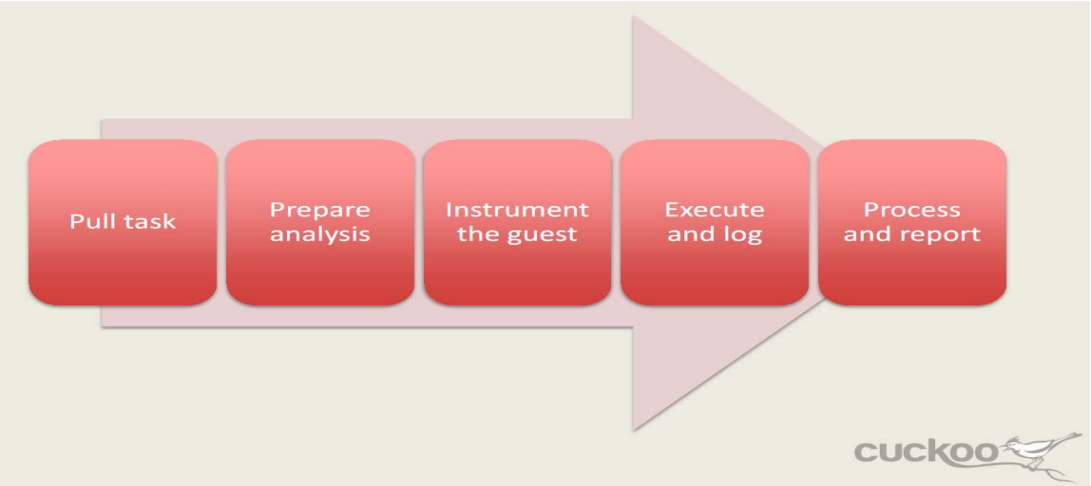
Στο κεφάλαιο αυτό θα παρουσιάσουμε τα εργαλεία που πρόκειται να χρησιμοποιηθούν στην πλατφόρμα. Έχουμε επιλέξει να συγκρίνουμε τα δύο εργαλεία το Cuckoo και το Laika Boss τα οποία όπως έχουμε αναφέρει και παραπάνω είναι ανοιχτού λογισμικού που μπορούν να εξελίσσονται συνεχώς και να προσφέρουν καλύτερα αποτελέσματα όσον αφορά την ανάλυση κακόβουλων λογισμικών σε βάθος και την αντιμετώπισή τους με τους καλύτερους δυνατούς αλγόριθμους μηχανικής μάθησης.

Σε αυτό το σημείο θα πρέπει να συγκρίνουμε τα δύο εργαλεία ανοιχτού κώδικα ανάλυσης του κακόβουλου λογισμικού του Cuckoo Sandbox και του Laika BOSS σε δύο βασικούς τομείς που είναι η στατική και η δυναμική ανάλυση.

4.1 Παρουσίαση των Cuckoo Sandbox και Laika BOSS.

Το Cuckoo Sandbox είναι ένα από τα πιο δημοφιλή ανοικτού κώδικα συστήματα ανάλυσης κακόβουλου λογισμικού. Έχει πολλές ενσωματώσεις για την εκτέλεση της ανάλυσης κακόβουλων προγραμμάτων ύποπτων αρχείων. Οι απαιτήσεις εγκατάστασης περιλαμβάνουν εξαρτήσεις και άλλα λογισμικά όπως VirtualBox, yara, ssdeep και μεταβλητότητα. Επίσης, η ανάλυση VM είναι Windows και απαιτούνται ορισμένες προϋποθέσεις για την εκτέλεση της ανάλυσης.

Στο παρακάτω σχήμα βλέπουμε τα βασικά βήματα από την διεργασία του Cuckoo Sandbox(Bremer, 2014).



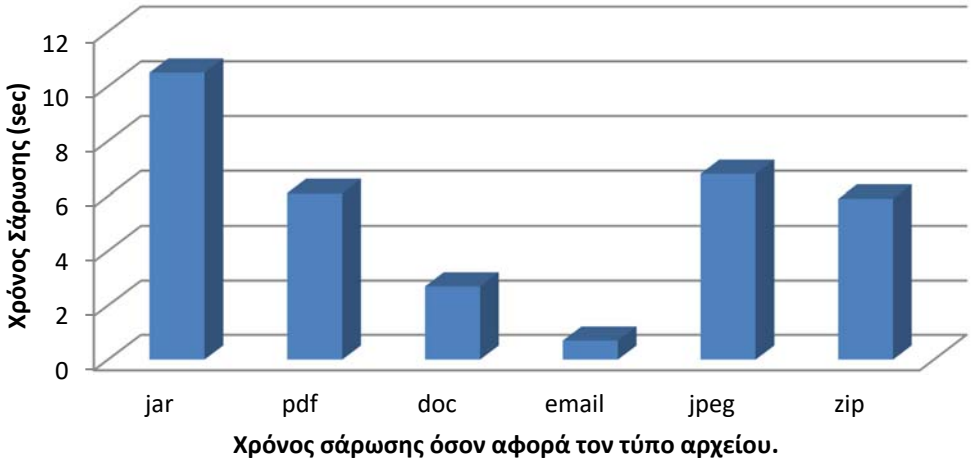
Το Laika είναι κυριώς ένα εργαλείο στατικής ανάλυσης αρχείων γραμμένο σε Python και YARA.

4.2 Οφέλη από τη χρήση του Laika BOSS.

Οι αναλυτές εκτελούν συνήθως διάφορες εργασίες στο πλαίσιο της ανάλυσης αρχείων ή αντικειμένων. Συγκεκριμένα μια ολοκληρωμένη ανάλυση του κακόβουλου λογισμικού, προϋποθέτει, όπως έχουμε αναφέρει και παραπάνω την εκπόνηση στατικής και δυναμικής ανάλυσης. Κάθε μία από αυτές τις φάσεις μπορεί να αναλυθεί περαιτέρω ώστε να συμπεριληφθεί τόσο η βασική όσο και προηγμένη ανάλυση που απαιτεί ποικίλα σύνολα ικανοτήτων για να ολοκληρωθεί (Sikorski, 2012). Το Laika BOSS έχει δομηθεί με τέτοιο τρόπο ώστε να είναι κλιμακώσιμο (scalable), μπορεί να δουλεύει σε πολλά συστήματα, να είναι ευέλικτο, υποστηρίζοντας οποιοδήποτε τύπο αρχείου. Επίσης επεξεργάζεται τα αρχεία και εκτελεί συγκεκριμένη λειτουργία, όπως εξαγωγή υποαρχείων, συλλογή μεταδεδομένων (metadata) ή εντοπισμό κακόβουλου λογισμικού μέσω σάρωσης (scanning) και άλλων τεχνικών.

Κατάτη διάρκεια της πειραματικής διαδικασίας και ανάλυσης κακόβουλου λογισμικού καταλήξαμε στο συμπέρασμα ότι έχουμε πολύ καλό χρόνο σάρωσης αρχείων ανάλογα με τον τύπο τους και την μορφή τους όσον αφορά το στάδιο της στατικής ανάλυσης. Επίσης επειδή έχουμε δοκιμάσει διάφορους τύπους αρχείων ακόμα και με συμπιεσμένη μορφή ο χρόνος σάρωσης όπως θα δούμε και παρακάτω στο σχήμα 2 είναι αρκετά ικανοποιητικός σε σχέση με το χρόνο σάρωσης της ανάλυσης του Cuckoo.

Σχήμα 2. Γραφική αναπαράσταση του χρόνου σάρωσης σε σχέση με τον τύπο αρχείου στο λογισμικό ανάλυσης Laika BOSS



Σε αυτό το σημείο θα πρέπει να επισημάνουμε ότι έχουμε δυνατότητα να σαρώνουμε αρχεία με πολύ καλό χρόνο σάρωσης και με μεγάλο όγκο δεδομένων όσον αφορά την στατική ανάλυση και σε αυτό το σημείο εντοπίζεται η διαφορά σε σχέση με τη δυναμική ανάλυση του Cuckoo Sandbox. Άλλωστε τα αποτελέσματα είναι όσον αφορά την στατική ανάλυση του Laika BOSS και με την δυναμική ανάλυση του Cuckoo Sandbox είναι πολύ ικανοποιητικά και με περαιτέρω πρόοδο στην ακρίβεια. Αυτό συμβαίνει διότι και τα δύο λογισμικά ανοιχτου κώδικα έχουν την ίδια βάση και εργαλεία αφού έχουν γραφτεί με Python και χρησιμοποιούν το Yara ανάλυσης όσον αφορά τόσο το αρχικό της στάδιο της στατικής ανάλυση όσο και της δυναμικής ανάλυσης.

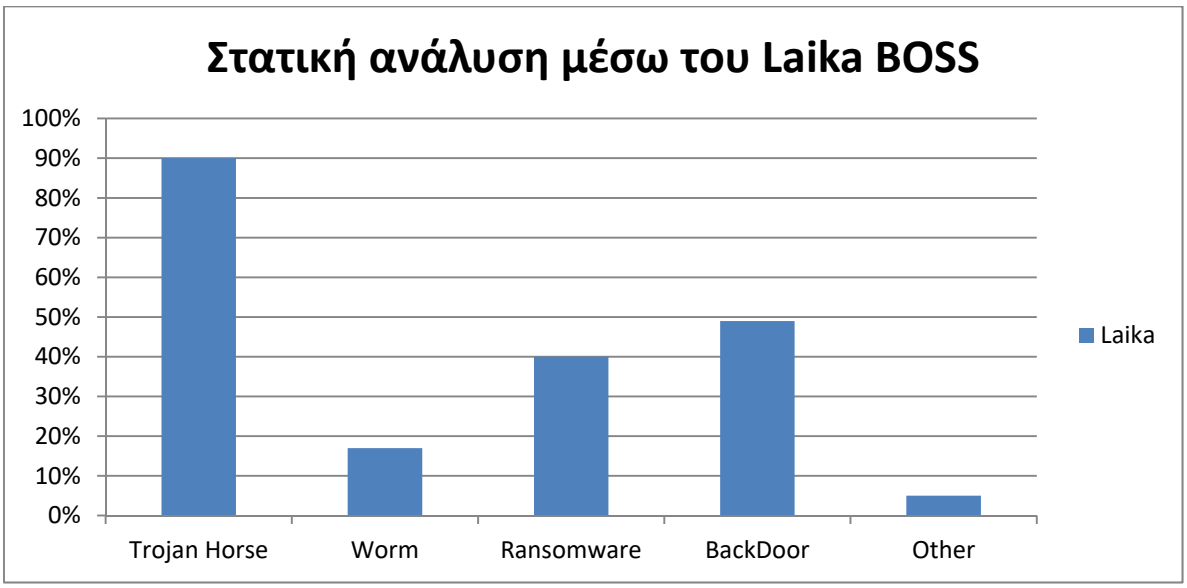
Βέβαια υπάρχουν διαφορές μεταξύ των δύο λογισμικών είναι αρκετές. Μια από τις σημαντικότερες είναι ότι το Cuckoo Sandbox είναι περισσότερο εδραιωμένο και γνωστό στην έρευνα και ανάλυση του κακόβουλου λογισμικού έχει αναπτυχθεί περισσότερο και αναβαθμίζεται συνεχώς από τους προγραμματιστές, αλλά είχε δημιουργηθεί και νωρίτερα από το Laika BOSS.

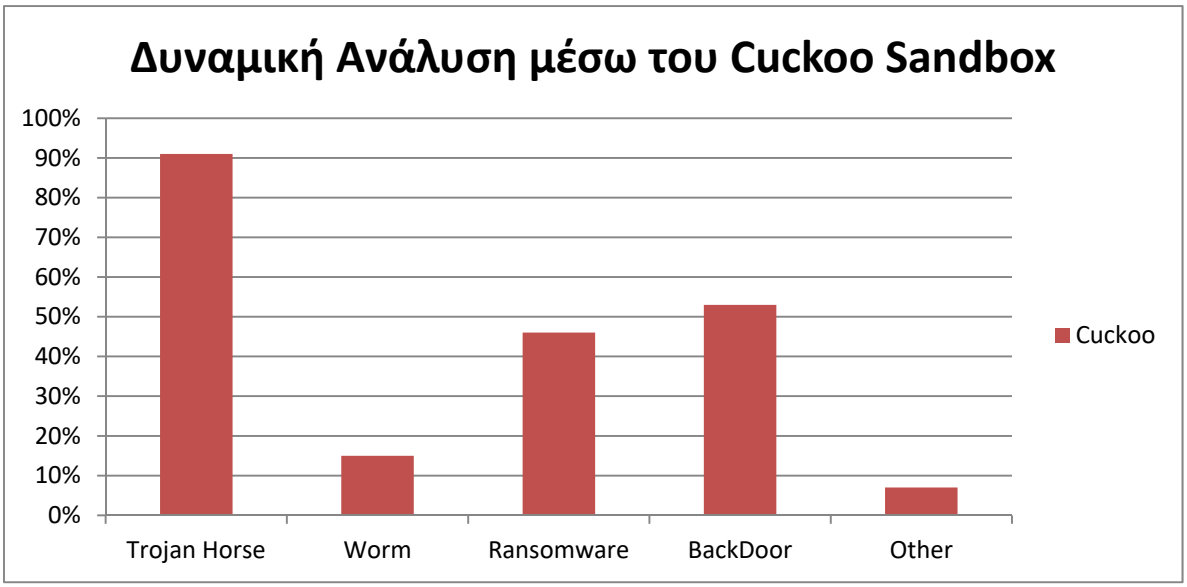
Κατά τη διάρκεια της πειραματικής διαδικασίας διαπιστώθηκε ότι το Cuckoo Sandbox είναι ένα αυτοματοποιημένο περιβάλλον δοκιμών κακόβουλου λογισμικού που χρησιμοποιείται για την εκτέλεση δυναμικής ανάλυσης δειγμάτων με πολύ ικανοποιητικά αποτελέσματα. Ένα απομονωμένο εικονικό μηχάνημα Windows χρησιμοποιείται για την εκτέλεση του δείγματος και την ανάλυση των αποτελεσμάτων. Ως εκ τούτου εκτελέσιμα αρχεία, DLL, διευθύνσεις URL, PDF και αρχεία Microsoft Office, μεταξύ άλλων, μπορούν να αναλυθούν σε αυτό το περιβάλλον.

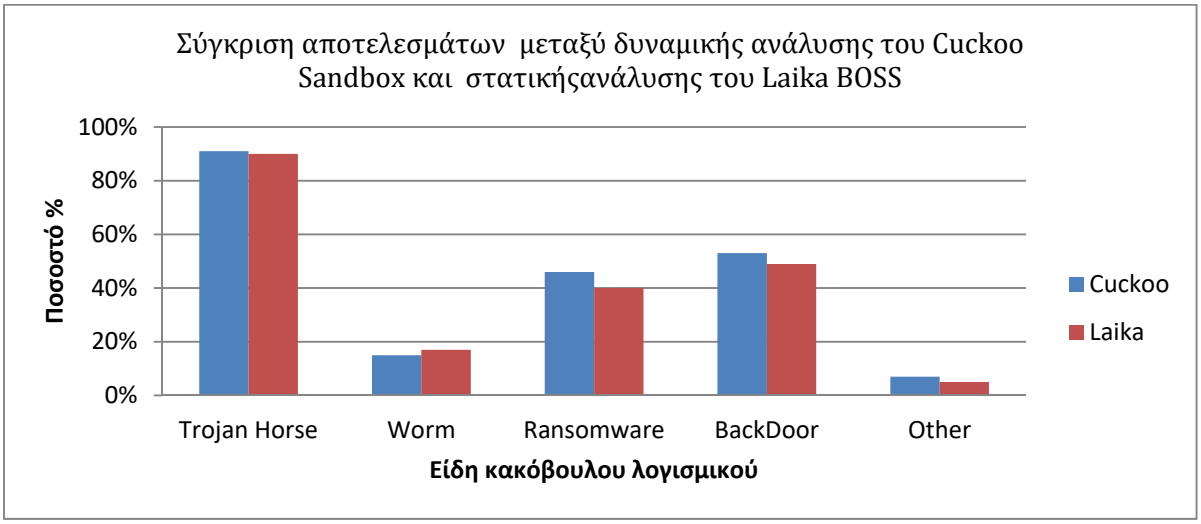
Εγκαταστάθηκαν τα προαπαιτούμενα που χρειάζεται το Cuckoo για την ανάλυση του ιομορφικού λογισμικού και συγκεκριμένα εγκαταστάθηκαν Python 2.7. Η βασική βιβλιοθήκη της Python, με την οποία είναι γραμμένος ο κώδικας του Cuckoo. Το SQLAlchemy μία συλλογή με εργαλεία για βάσεις δεδομένων που χρησιμοποιεί η Python. ToPython BSON για τον χειρισμό αρχείων Json στην Python.Το Dpkt για την εξαγωγή πληροφοριών από αρχεία PCAP. Το Jinja2 για τον χειρισμό των αναφορών σε μορφή HTML και του web interface. Το Magic που χρησιμοποιείται για την αυτόματη αναγνώριση του τύπου των αρχείων που είναι υπό ανάλυση. Το Pydeep για τον υπολογισμό του ssdeep fuzzy hash των αρχείων. Είναι απαραίτητη η λήψη του από εξωτερική πηγή (Ssdeep, 2016) κι όχι από το αποθετήριο του Ubuntu. Το Pymongo για την αποθήκευση των αποτελεσμάτων στην βάση δεδομένων MongoDB. Το YARA και Yara Python για το ταίριασμα των υπογραφών των αρχείων, με το αντίστοιχο της Yara (Yara, 2016).Το Bottlepy για την χρήση του api.py ή του web.py του Cuckoo. Το Django για την εμφάνιση των αποτελεσμάτων σε web interface. Το Volatility για την ενσωμάτωση του Cuckoo με το Volatilit είναι προαιρετική. Ωστόσο, η μείωση του χρόνου που απαιτείται για την ανάλυση ενός δείγματος και την αύξηση των πιθανών δεικτών μόλυνσης, αξίζουν τα πρόσθετα βήματα που απαιτούνται για την ενσωμάτωση του. Το Chardet για την ανίχνευση κωδικοποίησης συμβολοσειρών.

Συμπεραίνουμε εύκολα ότι με την χρήση του Cuckoo Sandbox έχουμε δυναμική ανάλυση σε μεγαλύτερη έκταση και βάθος ώστε να μπορέσουμε να ανιχνεύουμε και να αναλύουμε κακόβουλα λογισμικά υψηλού προγραμματιστικού επιπέδου.

Σε αυτό το σημείο θα πρέπει να αναφέρουμε ότι κατά τη διενέργεια στατικής ανάλυσης σε 20 διαφορετικά κακόβουλα λογισμικά διαφόρετικών κατηγοριών στο λειτουργικό σύστημα Windows 7 Professional και με βάση τα παρακάτω αποτελέσματα θα διαπιστώσουμε ότι η στατική ανάλυση με το Laika BOSS και με μια διαφορά χρόνου σάρωσης σε σχέση με το Cuckoo όπως αναφέραμε παραπάνω αποτελεί ενα πολύ καλό εργαλείο για την στατική ανάλυση. Παρακάτω στα διαγράμματα παρουσιάζουμε τα ακόλουθα αποτελέσματα από ένα δείγμα 20 διαφορετικών ειδών κακόβουλου λογισμικού τόσο στην στατική ανάλυση μέσω του Laika BOSS όσο και στη δυναμική ανάλυση μέσω του Cuckoo Sandbox.


Στατική ανάλυση μέσω του Laika BOSS
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
Trojan Horse
Worm
Ransomware
BackDoor
Other
Laika


Δυναμική Ανάλυση μέσω του Cuckoo Sandbox
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
Trojan Horse
Worm
Ransomware
BackDoor
Other
Cuckoo


Σύγκριση αποτελεσμάτων μεταξύ δυναμικής ανάλυσης του Cuckoo Sandbox και στατικήςανάλυσης του Laika BOSS
Ποσοστό %
100%
80%
60%
40%
20%
0%
Trojan Horse
Worm
Ransomware
BackDoor
Other
Είδη κακόβουλου λογισμικού
Cuckoo
Laika

Συμπερασματικά λοιπόν θα χρησιμοποιηθεί για την δημιουργία της μελλοντικής πλατφόρμας το Laika BOSS όσον αφορά τη στατική ανάλυση και όσον αφορά την δυναμική ανάλυση το Cuckoo Sandbox.

Κεφάλαιο 5

Επίλογος

Στο κεφάλαιο αυτό θα παρουσιαστούν περιληπτικά τα βασικά σημεία της εργασίας, καθώς και προτάσεις για περαιτέρω βελτιώσεις στην πλατφόρμα και μελλοντική έρευνα.

5.1 Σύνοψη

Στην αρχή της εργασίας παρουσιάστηκε η κατάσταση που επικρατεί σήμερα όσον αφορά το κακόβουλο λογισμικό. Το κακόβουλο λογισμικό αυξάνει διαρκώς και οι υπάρχουσες μέθοδοι των υπογραφών δεν μπορούν να αντεπεξέλθουν πλήρως. Για το λόγο αυτό, χρειάζεται να δημιουργηθούν νέες τεχνικές που θα βοηθούν στην ανάλυση του κακόβουλου λογισμικού και θα είναι αυτοματοποιημένες. Προς αυτή την κατεύθυνση κινείται και η παρούσα εργασία, στην οποία παρουσιάζεται η δημιουργία μιας πλατφόρμας για την αυτοματοποιημένη δυναμική και στατική ανάλυση κακόβουλου λογισμικού, συγκρίνοντας δύο εργαλεία ανοιχτού κώδικα όπου σε συνδυασμό με τους προτεινόμενους αλγόριθμους μηχανικής μάθησης θα μπορέσουν να αποτελέσουν την ολοκλήρωση της πλατφόρμας.

Στη συνέχεια, έγινε αναφορά στις βασικές έννοιες που πραγματεύεται η εργασία και δόθηκαν γενικές πληροφορίες για αυτές. Αρχικά, δόθηκε ο ορισμός του κακόβουλου λογισμικού και σημαντικές πληροφορίες για τις ανάγκες της παρούσας διατριβής. Ακολούθησαν οι κατηγορίες στις οποίες χωρίζεται ανάλογα με τον τρόπο που μεταδίδεται και τις λειτουργίες που εκτελεί στο μολυσμένο σύστημα. Αναφέρθηκαν επίσης οι επικρατέστεροι τρόποι διάδοσης του κακόβουλου λογισμικού. Δόθηκαν οι ορισμοί τους, οι μορφές που μπορούν να πάρουν, οι τεχνικές με τις οποίες επιτυγχάνονται, καθώς και τα πλεονεκτήματα και μειονεκτήματα της κάθε μίας Μετά, έγινε μια εισαγωγή στη μηχανική μάθηση, δίνοντας τον ορισμό της και εξερευνώντας τις κατηγορίες της και τους αλγορίθμους με τους οποίους υλοποιείται. Έπειτα, έγινε αναφορά σχετικών εργασιών, με το θέμα της εργασίας, και την βιβλιογραφία.

Στο τρίτο κεφάλαιο δόθηκαν οι αρχιτεκτονικές προδιαγραφές που θα πρέπει να πληροί η πλατφόρμα. Συζητήθηκαν οι απαιτήσεις που υπάρχουν από αυτή (κλιμακωσιμότητα) και ο

τρόπος με τον οποίο θα πρέπει να λειτουργεί ώστε να επιτυγχάνονται τα καλύτερα αποτελέσματα.

Επίσης, συγκρίναμε τα δύο λογισμικά ανοιχτού κώδικα του Cuckoo Sandbox και του LaikaBOSS με αποτελέσματα που μας βοηθήσανε να τα χρησιμοποιήσουμε όσο το δυνατόν αποδοτικότερα στην δημιουργία της μελλοντικής πλατφόρμας αλλά και να καταλήξουμε σε πιθανούς αλγόριθμους μηχανικής μάθησης που για την καλύτερη ταξινόμηση και αποτελεσματικότητα με συσταδοποίηση όσον αφορά την ανάλυση και αντιμετώπιση του κακόβουλου λογισμικού.

Στην αρχή, έγινε μια παρουσίαση της λειτουργίας και της δομής όλων των προγραμμάτων που επελέγησαν να χρησιμοποιηθούν και αναφέρθηκαν οι λόγοι για τους οποίους καλύπτουν τις απαιτήσεις του τρίτου κεφαλαίου, με κυριότερους την δυνατότητα κλιμακωσιμότητας και την επεκτασιμότητα. Στη συνέχεια, έγινε μια σύντομη αναφορά στο υλικό της πλατφόρμας πριν αναλυθεί η Python, η οποία και επιλέχθηκε ως γλώσσα υλοποίησης της πλατφόρμας. Στην επόμενη ενότητα παρουσιάστηκε η διαδικασία της εγκατάστασης των προγραμμάτων και κυριότερα αυτή της ρύθμισής τους ώστε να δουλεύουν με τον επιθυμητό τρόπο. Τέλος παρουσιάστηκαν τα σημαντικότερα κομμάτια του κώδικα της πλατφόρμας, ο οποίος χρησίμευσε στην διασύνδεση των μερών της και στην δημιουργία της διοχέτευσης.

5.2 Μελλοντικές Επεκτάσεις

Το αντικείμενο αυτής της εργασίας, δηλαδή η μελλοντική κατασκευή της πλατφόρμας, προσφέρει πολλές προοπτικές για βελτίωση και επιπλέον έρευνα.

Επομένως καταλαβαίνουμε την αναγκαιότητα και χρήση της τεχνητής νουμοσύνης όσον αφορά την αποδοτικότητα και αποτελεσματικότητα στην αντιμετώπιση και ανάλυση του κακόβουλου λογισμικού.

Σε αυτό το σημείο θα πρέπει να αναφέρουμε ότι η αξία στο Laika BOSS παρέχει τη δυνατότητα ανάπτυξης και βελτίωσης, καθώς οι αναλυτές απαιτούν διαφορετικούς τύπους ανάλυσης λόγω της συνεχής εξέλιξης και βελτίωσης των κακόβουλων λογισμικών. Ως αποτέλεσμα, ο συγγραφέας έχει διερευνήσει και σκέφτηκε διαφορετικούς τομείς για μελλοντική έρευνα που σχετίζεται με την ανάπτυξη ενότητας για τη Laika BOSS.

Συμπερασματικά το Laika BOSS όπως έχει μελετηθεί στην παρούσα προσέγγιση μπορεί να εξελιχτεί ώς ένα εργαλείο με απεριόριστη λειτουργικότητα ανάλογα με την φαντασία και την ανάγκη του αναλυτή.

Σύμφωνα με την βιβλιογραφία, τα χαρακτηριστικά που χρησιμοποιούμε στην ανάλυση κακόβουλων λογισμικών σε προβλήματα ταξινόμησης, διαφορετικά μοντέλα έδωσαν

διαφορετικά αποτελέσματα. Το χαμηλότερο ακρίβεια επιτεύχθηκε με τον αλγόριθμο Naive Bayes (72,34% και 55%), ακολουθούμενο από τους "κοντινότερους γείτονες" και τους μηχανισμούς διανύσματος υποστήριξης (87%, 94,6% και 87,6%,94,6% αντιστοίχως). Η υψηλότερη ακρίβεια επιτεύχθηκε με τα J48 και Random Forest μοντέλα, και ήταν ίσο με 94,6% και 96,8% για τη δυαδική ταξινόμηση αντίστοιχα.

Τέλος, ερευνητικό ενδιαφέρον παρουσιάζει η μελέτη ενός μηχανισμού, ο οποίος θα αποφασίζει για ποια δείγματα αρκεί μόνο η στατική ανάλυση και ποια δείγματα πρέπει να αναλυθούν και δυναμικά. Εξίσου ενδιαφέρουσα είναι και η δοκιμή διαφόρων συνδυασμών ταξινομητών (bayes, decision trees,) με διάφορα σύνολα χαρακτηριστικών (feature sets) για την εύρεση του συνδυασμού που επιτυγχάνει τα καλύτερα αποτελέσματα ταξινόμησης. Το πιο αξιόλογο μελλοντικά εγχείρημα θα είναι η χρήση αλγορίθμου συσταδοποίησης και συγκεκριμένα του αλγόριθμου SOINN πολύ χρήσιμος για εφαρμογή σε μη επιβλεπόμενη μάθηση.

Παράρτημα Α

Κώδικας Εφαρμογής

Κώδικας της Πλατφόρμας

Α.1 start.py

```
#! /usr/bin/env python
import subprocess
import sys
import os
import core
import multiprocessing
import time
import classifiers
import config as cfg
from Queue import Empty, Full
from dtrx import ExtractorApplication


def check_running(procs):
  for process in procs:
    if process.poll() != None:
      print "\x1b[31mA process terminated abnormally.
      Exiting...\x1b[0m"
```

```
            for p in procs:
                p.terminate()
            return False
    return True

class Platform():

    def __init__(self):
        if cfg.CLASSIFICATION_TYPE == "multiclass":
            self.sclf = classifiers.Classifier()
        else:
            self.sclf                    =
            {"stage1":classifiers.Cl
            assifier(),
            "stage2":classifiers.Cl
            assifier()}
        self.monitor = core.Monitor(cfg.POOL_FOLDER,
        cfg.POOL_FOLDER + " monitor", self.sclf)

            def
            _prepare_classifiers(s
            elf,        dset_names):
                for stage, dset in
            dset_names.iteritems(
            ):
                try:
                    self.sclf[stage].train(dset)
                except TypeError:
                    print "\x1b[31mThere is not such
                    dataset:",dset,"\x1b[0m" sys.exit(1)

        def main(self):
            try:
                if cfg.MODE == "classification":
                    print "\x1b[33mPreparing the classifiers...\x1b[0m"
                    self._prepare_classifiers(cfg.DATASETS_TO_LOAD)
                    print "\x1b[32mClassifiers trained
                    sucsessfully!\x1b[0m"
                print "\x1b[33mStarting laika...\x1b[0m"
                try:
                    laika = subprocess.Popen(["laikad.py"])
                except OSError as e:
                    print "\x1b[31mFailed starting laika:" + e.strerror +
                    "\x1b[0m"
                    sys.exit(1)
                print "\x1b[32mSuccess!\n\x1b[0m"


                self.monitor.start()
                while check_running([laika]):
                    time.sleep(5)
                sys.exit(1)
            except KeyboardInterrupt:
                laika.terminate()
                self.monitor.stop()
                print "\x1b[36m\nStop\x1b[0m"
```

```
if __name__ == "__main__":
    platform = Platform()
    platform.main()
```

A.2 core.py

```
"""
This module provides the Monitor and the Scanner classes. These classes provide the main functionality of the platform. The first collects the samples and dispatches them to the second, which performs all the
necessary actions.
"""

import threading
import subprocess
import json
import os
import sys
import requests
import time
import tempfile
import zipfile
import gridfs
import re
import magic

from dtrx import ExtractorApplication
from pymongo import MongoClient
import config as cfg
from hashlib import md5
from collections import Counter
from jq import jq
from sklearn.feature_extraction import DictVectorizer
#from Queue import Empty, Full
#Possibly unneccesary

DNS_RECORDS_TO_MONITOR = ["A", "CNAME", "PTR", "MX"]
HTTP_METHODS_TO_MONITOR = ["GET", "HEAD", "POST"]
HTTP_RESPONSES_TO_MONITOR = ["200", "300", "400", "500"]

class Monitor(threading.Thread):
    """
    The Monitor class checks for new files in a directory. Once a new file is
    detected it is uncompressed (if applicable) and then dispatched to a new
```

```
        Scanner object.

        :param path: the path to the directory to monitor.
        :type path: str
        :param name: the name
        of the Monitor thread.
        :type name: str
        :param sclf: the static classifier to be passed to the scanner.
        """

        def __init__(self, path, name, sclf):
            """
            The constructor of the class. It sets the parameters of the
class.
            It also creates a threading.Event to stop the thread when
necessary.
            """
            threading.Thread.__init__(self, name = name)
            self.path_to_monitor = path
            self.sclf = sclf
            self._stop = threading.Event()

        def _unpack(self, obj):
            """
            Unpacks the new file in the case it's compressed. It uses
the drtx
            module. The extracted file is situated in the same directory
as the
                archive.

                :param obj: The
                compressed archive to
                extract.
                """
                os.chdir(self.path_to_monitor)
                ex = ExtractorApplication(["-n", "-o", "-f", obj])
                ex.run()
                os.chdir(cfg.FRAMEWORK_PATH)
                os.remove(self.path_to_monitor+obj)

        def stop(self):
            """
            Sets the internal Event flag,
            so the thead can exit.
            """
            self._stop.set()

        def run(self):
            """
            The main method of the class. It is called by the start
method
            of the class. While the thread must not be stopped this
functions
            inspects the designated folder for new files.
```

```
                Once it finds one
                it checks if it is compressed and
                uncompresses it. Next it forks
                a new Scanner thread passing it the file.
                The Scanner thread is
                deamonized so it will terminate immediately
                when the parent python
 85             process exits.
                """
                regex = r"(" +
                "|".join(cfg.COMPRESSE
                D) + ")" while not
                self._stop.isSet():
                  if os.listdir(self.path_to_monitor) == []:
 90                 print "\x1b[36mWaiting for submissions...\x1b[0m",
#Formatting
                                                  #via ANSI escape codes
                    sys.stdout.flush() #necessary for
                    the formatting to work
                    time.sleep(cfg.CHECK_INTERVAL)
                    print "\x1b[2K\r", #Back space ANSI code
 95               else:
                    for obj in os.listdir(self.path_to_monitor):
                      if obj[0] != '.' and
                      os.path.isfile(self.path_to_monitor
                                     + obj):
                        file_type =
                        magic.from_file(self.path_to_monitor
100                                    + obj)
                        if re.match(regex, file_type, re.I):
                          self._unpack(obj)
                          break #the extraction is done,
                               #the archive is deleted, restart the
                               loop to find
105                     #the new file
                        os.rename(self.path_to_monitor+obj,
                                cfg.CHECKED_FOLDER+obj)
                        if cfg.MODE == "training":
                          scanner = Scanner(cfg.CHECKED_FOLDER+obj,


110                                       "scanner--"+obj, self.sclf,
                                          cfg.CLASS)
                    else:
                      scanner = Scanner(cfg.CHECKED_FOLDER+obj,
                                        "scanner--"+obj, self.sclf)
115                 scanner.daemon = True
                    scanner.start()


              class Scanner(threading.Thread):
                """
120             Each instance of this class is responsible for the handling of
an
```

```
        individual file. It passes it to the scanners, adds it to the
        database and classifies it.


125     :param path: the path of the sample to process.
        :type path: str
        :param name: the name of the thread to be created.
        :type name: str
        :param sclf: the static classifier to use.
130     :param target_class: when in training mode, the class to be
assigned
        at the sample.
        :type target_class: str or list
        """
        def __init__(self, path, name, sclf, target_class = None):
          """
135       Creates the necessary pathnames and initializes the sample
          representation to be
          written in the database.
          """
          threading.Thread.__init__(self,name=name)
          self.path = path
140       self.sclf = sclf
          self.fname = os.path.basename(self.path)
          self.rpath =
          cfg.RESULTS_FOLDER+"{0[0]:02d}{0[1]:02d}{0[2]:02d}\
          {0[3]:02d}{0[4]:02d}{0[5]:02d}".format(time.localtime())+"_"+s
          elf.fname
          self.lpath = self.rpath + "/static"
145       self._connect_to_db()
          if target_class:
            self.mongo_obj = {"class":target_class,
                      "name":
                      md5(open(self.path,"rb").read()).hexdigest()}
          else:
150         self.mongo_obj = {"name":
            md5(open(self.path,"rb").read()).hexdigest()}


        def _send_to_laika(self):
          """
155       Sends the sample for static analysis and filters out the
          results to
          get the flags generated by the Yara rules.


                :return lresult: the filtered results.
                """
160             print "\x1b[36mStarting static analysis for " +
                self.fname + "\x1b[0m"
                proc = subprocess.Popen(["cloudscan.py", self.path],
                           stdout=subprocess.PIPE,
                           stderr=subprocess.PIPE)
                output = proc.communicate() #get the process output
```

```
        if not proc.returncode: #if the process exited successfully
            result = json.loads(output[0])
            self.mongo_obj["laika_analysis"] = result jqFilter = ".scan_result[0] |\
            {action: .moduleMetadata.DISPOSITIONER.Disposition.Result,\
            flags: .flags, file: .filename, scanned_with: .scanModules}"
            result = jq(jqFilter).transform(result)
            self.mongo_obj["static_featres"] = result["flags"] result = json.JSONEncoder().encode(result)
            #print lresult
            print "\x1b[32mStatic analysis for " + self.fname + \
              " completed succesfully\x1b[0m"
            return result
          else:
            print "\x1b[32mStatic analysis for " + self.fname +\
            " failed with: " + output[0] +"x1b[0m"
            return None

        def _send_to_cuckoo(self):
          """
          Handles the procedure of the dynamic analysis.
          It submits the sample, wait for the analysis
          completion and then invokes the collection of the
          features.

          :return task_id: the id that cuckoo gives to the sample.
          :rtype: int
          """
          print "Submitting " + self.fname + " for cuckoo scan... " task_id = self._submit_sample()
          while self._get_cuckoo_status(task_id) != "reported":
            time.sleep(30)
          self._collect_dyn_feat(task_id)
          print "\x1b[32mDynamic analysis for " + self.fname +\
                                  " terminated.\x1b[0m"
          return task_id


        def _collect_dyn_feat(self, task_id):
          """
          Collects the features from the dymanic analysis results
```

```
205         and adds them to the sample representation.

            :param task_id: the id of the task/sample to collect the
        features for


          """
          creport = open(cfg.CUCKOO_PATH+"storage/analyses/"
210               + str(task_id) +"/reports/report.json","r")
          cresults = json.load(creport)
          df = Counter()
          #---FILE SYSTEM FEATURES---#
          for proc in cresults["behavior"]["generic"]:
215         for action in proc["summary"]:
              if "file" in action:
                df[action]          +=
          len(proc["summary"][action
          ])                #---REGISTRY
          FEATURES---#
              if "regkey" in action:
220             df[action] += len(proc["summary"][action])
          #---NETWORK FEATURES---#
          df["tcp_count"]             =
          len(cresults["network"]["tc
          p"])  df["udp_count"]     =
          len(cresults["network"]["ud
          p"]) unique_ips = []
225       for item in cresults["network"]:
            for connection in cresults["network"][item]:
              if type(connection) == dict:
                try:
                  if connection["src"] not in unique_ips:
230                 unique_ips.append(connection["src"])
                  if connection["dst"] not in unique_ips:
                    unique_ips.ap
                pend(connection["
                dst"])        except
                KeyError:
                    pass
235       df["unique_ips"] = len(set(unique_ips +
        cresults["network"]["hosts"]))
          #get
          rid
              # of possible duplicates in hosts
          df["unique_ips"] -= 1 #remove one for machine's own
          ip     (eg.     192.168.56.1xx)     for     connection     in
          cresults["network"]["http_ex"]:
            if connection["method"] in HTTP_METHODS_TO_MONITOR:
240             df["HTTP_" + connection["method"]] += 1
          for connection in cresults["network"]["http_ex"]:
            if len(connection["response"]) > 10:
              response      =      connection["response"][9]+"00"
              #regular expression possibly if response in
              HTTP_RESPONSES_TO_MONITOR:
```

```
            df["HTTP_" + response] += 1
        for connection in cresults["network"]["dns"]:
          if connection["type"] in
DNS_RECORDS_TO_MONITOR:
            df["DNS_" +
connection["type"]] += 1

        self.mongo_obj["dynamic_feat
        ures"] = dict(df)
        #---DYNAMIC SIGNATURES---#
        self.mongo_obj["dynamic_features"].update(
          {"dynamic_signatures":[sig["name"] for sig in
cresults["signatures"]]}) creport.seek(0)
        self.mongo_obj["cuckoo_analysis"] = self.fs.put(creport)
        creport.close()

    def _submit_sample(self):
      """
      Actually submits the sample for dynamic anlaysis with
      cuckoo via a Rest API call and retrieves the id cuckoo
      has assigned to it.
      """
      with open(self.path,"rb") as sample:
        multipart_file = {"file": (self.fname, sample)}
        request = requests.post(cfg.REST_URL+"create/file",
                                files=multipart_file,
                                data={"platform":"windows"})
      json_dec = json.JSONDecoder()

      if request.status_code == 200:
        print "Submission successful. Now we wait for the
results..."
        return json_dec.decode(request.text)["task_id"]
      else:
        print "Submission failed."
        return None


    def
    _get_cuckoo_st
    atus(self,
    task_id):
      """
      Executes a Rest API call to retrieve the status
      of a task (pending, reported...).

      :param
      task_id: the
      id of the
      task :type
      task_id: int
      :return status: the status of the task
      :rtype: str
      """
```

```
            request                                    =
            requests.get(cfg.REST_URL+"view/"
            +str(task_id))        json_dec             =
            json.JSONDecoder()
            status  =  json_dec.decode(request.text)["task"]["status"]
290         return  status


        def  _connect_to_db(self):
            self.mongo_cl  =  MongoClient()
295         self.db  =  self.mongo_cl["framework"]
            self.fs  =  gridfs.GridFS(self.db)  #necessary  for  storing
            documents
            #> 2MB  to  the  db  (cuckoo  reports,  samples)



300     def  _add_to_db(self):
                with  zipfile.ZipFile(self.path+".zip","w")  as  zipf:
                                    zipf.write
                            (self.path,self.fn
                            ame)
                            os.remove(self.pa
                            th)
                            self.mongo_obj["binary"]  =
                            self.fs.put(open(self.path+".zip","rb"))
305                         self.db.malwr.insert_one(self.mongo_obj)


        for.
        def  _static_classify(self):
            """
            This  method  handles  the  static  classification  procedure.
310         """
            sample  =  {flag:1  for  flag  in
            self.mongo_obj["static_features"]}
            vec  =  DictVectorizer()
            data                    =
            vec.fit_transform(sa
            mple).toarray()
            features                =
            vec.get_feature_nam
            es()
315         classes,  tn  =  self.sclf["stage1"].classify(data,  features)
            print  tn
            if  tn  !=  "benign":
                print  "Classified  as  malware"
                return  True
320         else:
                print  "Classified  as  benign,  proceeding  to  dynamic  analysis"
                return  False


        def  _classify(self):
            """
325
```

```
        This method handles the combined classification procedure.
        """
        sample = {flag:1 for flag in
        self.mongo_obj["static_features"]}
        d_sigs =
        self.mongo_obj["dynamic_features"].pop("dynamic_signatures")
330     sample.update(self.mongo_obj["dynamic_features"])
        #sample.update({flag:1 for flag in d_sigs})
        vec = DictVectorizer()
        data =
        vec.fit_transform(sa
        mple).toarray()
        features =
        vec.get_feature_nam
        es()
335     try:
          classes, tn = self.sclf["stage2"].classify(data, features)
        except AttributeError:
          classes, tn =
        self.sclf.classify(data,
        features) print "Classified
        as:", classes, tn
340


    def _save_results(self,
    lresult, cuckoo_task_id):
      """
345   Saves the results in the directory specified at config.py. The
    results are
      also stored in the database.
      """
      if lresult:
        res = open(self.lpath,"w")
350     res.writelines(lresult)
        res.close()
      if cuckoo_task_id:
        os.symlink(cfg.CUCKOO_P
        ATH+"storage/analyses/"+
            str(cuckoo_task_id),
        self.rpath+"/cuckoo")
355
       def run(self):
         print "\nProcessing file:" + self.fname
         res =
         self.db.malwr.find({"name":self.mon
         go_obj["name"]}) if res.count() != 0:
360          answer = raw_input("Seems that the file " +
    self.fname +
             " already exists in our database classified as:"
             + ", ".join(res[0]["class"]) + ".\n Do you want
             to resubmit? (y/n)") while answer not in
             ["y","n"]:
               answer = raw_input("Seems that the file " +
```

```
                self.fname +
365             " already exists in our database classified as:"
                + ", ".join(res[0]["class"]) + ".\n Do you want t
              o resubmit? (y/n)")
              if answer == "n":
                return
          os.mkdir(self.rpath)
370       if cfg.CLASSIFICATION_TYPE == "binary":
            result = self._send_to_laika()
            if self._static_classify(): #if
            malware from static, exit
              return
            else: #if benign, run the dynamic
375           cuckoo_task_id = self._send_to_cuckoo()
          else: #if multiclass run them both
            result = self._send_to_laika()
            cuckoo_task_id = self._send_to_cuckoo()


380       if "class" in self.mongo_obj: #if in training mode
            self._save_results(result, cuckoo_task_id)
            self._add_to_db()
          else:
            self._classify()
```

A.3 classifiers.py

```
import utils
import numpy
from sklearn import svm
from sklearn import preprocessing
5
class Classifier():
  def __init__(self):
    self.clf = svm.LinearSVC()
    self.scaler = preprocessing.MinMaxScaler()
10
  def train(self, dset_name):
    dsmngr = utils.DatasetManager()
    self.dataset = dsmngr.load(dset_name)
    self.scaler = self.scaler.fit(self.dataset.data)
15  self.dataset.data =
self.scaler.transform(self.dataset.data)
    self.clf.fit(self.dataset.data[0:-1],
    self.dataset.target[0:-1])

def save_model(self):
#for future expansion
  pass
20

def classify(self, sample, features):
  temp = sample.tolist()

  for index, feature in
  enumerate(self.dataset.feature_names):
    if feature not in features:
```

```
        features.insert(index, feature)
        temp[0].insert(index,0)
    sample =
    numpy.array(temp,
    dtype=numpy.float64)
    sample =
    self.scaler.transform(samp
    le)
    classes =
    self.clf.predict(sample)
    return classes, self.dataset.target_names[classes[0]]
```

A.4 utils.py

```
#! /usr/bin/env python
import argparse
import datetime
import magic
import os
import requests
import time
from jq import jq
from sklearn.feature_extraction import DictVectorizer
from sklearn.datasets.base import Bunch
from pymongo import MongoClient
from collections import Counter


class DatasetManager():
  def __init__(self, operation = None, args = None):
    self.operation = operation
    self.args = args
    self._connect_to_db()


  def create(self, name, feats_type, classes, descr = None):
    for c in classes:
      c = c.lower() #We only allow lowercase in the database
    results = self.samples.find({"class": {"$in": classes}},
                                                        ["_i
    d"])
    samples = [result["_id"] for result in results]
    dataset = {"name": name,
            "created_on": datetime.datetime.utcnow(),
            "description": descr,
            "class_names": classes,
            "feats_type": feats_type,
            "size": results.count(),
            "samples": samples}
    self.datasets.insert_one(dataset)

    def load(self, name):
      dataset = self.datasets.find_one({"name": name})
      try:
```

```
            target_names =
        dataset["class_na
        mes"]       except
        TypeError:
            raise
        results = self.samples.find({"_id": {"$in":
dataset["samples"]}})

        if dataset["feats_type"] == "static":
            samples,                target                =
        self._extract_static_feats(results, target_names)
        elif dataset["feats_type"] == "dynamic":
            samples, target = self._extract_dynamic_feats(results,
target_names)
        else:
            samples, target = self._merge_feats(results,
            target_names)


        vec = DictVectorizer()
        data = vec.fit_transform(samples).toarray()
        feature_names = vec.get_feature_names()

        dataset = Bunch(data=data, target=target,
        target_names=target_names,
                        DESCR=dataset["description"],
                        feature_names=feature_names)
        return dataset

    def     _extract_static_feats(self,
    results,            target_names):
        samples = []
        target = []

        for item in results:
            samples.append({flag:1   for   flag   in
            item["static_features"]})  for    c    in
            item["class"]:
                if c not in target_names:
                    item["class"].remove(c) #We keep only the class
we 're
interested     in
            target.append(target_names.index(item["class"][0]))


        return samples, target


    def _extract_dynamic_feats(self, results, target_names):
        samples = []
        target = []


        for item in results:
            d_sigs =
item["dynamic_features"].pop("dynamic_signatures")
            samples.append(item["dynamic_features"])
```

```
                #samples.append({flag:1 for
                flag in d_sigs}) works for c
                in item["class"]:
                  if c not in target_names:
                    item["class"].remove(c) #We keep only the class
we 're
interested in
                target.append(target_names.index(item["class"][0]))
        return samples, target

    def _merge_feats(self, results, target_names):
        samples = []
        target = []

        for item in results:
            features = {flag:1 for flag in item["static_features"]}
            d_sigs = item["dynamic_features"].pop("dynamic_signatures")
            features.update(item["dynamic_features"])
            #features.update({flag:1 for flag in d_sigs}) works
            samples.append(features)
            for c in item["class"]:
                if c not in target_names:
                    item["class"].remove(c) #We keep only the class we are
                    interested

                    in
            target.append(target_names.index(item["class"][0]))

        return samples, target

    def _connect_to_db(self):
        self.mongo_cl = MongoClient()
        self.datasets = self.mongo_cl["framework"]["datasets"]
        self.samples = self.mongo_cl["framework"]["malwr"]

    def main(self):
        if self.operation == "create":
            self.create(self.args["name"], self.args["features_type"],
            self.args["classes"],
        self.args["description"])
        else:
            self.load(self.args["name"])

if __name__ == "__main__":
    parser = argparse.ArgumentParser()
    subparsers = parser.add_subparsers(title="commands"
                    description="operations supported by the tool",
```

```
                    help="for the usage of each command see the\
                    corresponding help message")

    subpar_create = subparsers.add_parser("create")
    subpar_create.add_argument("name", type=str,
                               help="the name of the dataset to be
                               created")
    subpar_create.add_argument("features_type", type=str,
                    choices=["static", "dynamic", "both"],
    help="Wether the dataset contains static, dynamic fetures or
    both")
    subpar_create.add_argument("classes", type=str, nargs="+")
        subpar_create.add_argument("-d", "--description",
        type=str, help="A short description for the dataset")

        subpar_load = subparsers.add_parser("load")
        subpar_load.add_argument("name", type=str,
        help="the name of the dataset to be loaded")


        args = vars(parser.parse_args())
        if "classes" in args:
            operation = "create"
        else:
            operation = "load"
        manager =
        DatasetManager(ope
        ration, args)
        manager.main()

    def clearDuplicates(db, dataset):
        mongo_cl = MongoClient()
        dataset =
        mongo_cl[
        db][datase
        t]
        res =
        dataset.fin
        d({},
        ["name"])
        res = [result["name"] for result in res]
        duplicates = [item for item, occurences
        in Counter(res).items() if occurences > 1]
        for item in duplicates:
            dataset.delete_one({"name":item})

    ok = ["PE32 executa", "Rich Text Fo",
    "PDF document",
    "Microsoft Wo", "RAR
    archive ", "RAR self-ext",
    "Zip archive ", "PE
    Unknown P", "DOS
    executab", "Java Jar fil"]
    def cleanvs(path):
        """Keeps only exe,
        dll, office, pdf files"""
        log =
```

```
        open("log.txt","w+")
        os.chdir(path)
        for f in os.listdir(path):
165         ftype = magic.from_file(f)
            if ftype[:12] not in ok:
                os.remove(f)
                log.write("removi
ng                  "+f+"
"+ftype+"\n")
        log.close()
```

A.5 config.py

```
        FRAMEWORK_P
        ATH             =
        "/home/framework
        /"
        CUCKOO_PATH
        =
        "/home/cuckoo/c
        uckoo/"
        MONGO_PATH =
        "/home/mongo/"
        #POOL_FOLDER = "/var/www/html/uploads/" for production
5       POOL_FOLDER = MONGO_PATH + "pool/"
        CHECKED_FOLDER  =
        MONGO_PATH      +
        "checked/"
        RESULTS_FOLDER  =
        MONGO_PATH      +
        "results/"
    REST_URL        =
"http://localhost:808
7/tasks/"
CHECK_INTERVAL = 10
10  MODE = "classification"
CLASS = ["benign"]
CLASSIFICATION_TYPE = "binary"
DATASETS_TO_LOAD = {"stage1":"bclass_stat1",
"stage2":"bclass_both1"}
COMPRESSED =
["zip","rar","7z","tar","gz","bz","bzip","bz2","lzma","xz"]
```

Βιβλιογραφία

"Iran Says Malicious Software Hit Its Petrochemical Complexes." - Middle East. N.p., n.d. Web. 02 Sept. 2016. Available online:

http://www.israelnationalnews.com/News/News.aspx/217008 [Accessed: 30th August 2016]

"Trident Zero-Day IOS Vulnerabilities Lead to Real-World Espionage." Pegasus Spyware: What You Need to Know Now. N.p., n.d. Web. 02 Sept. 2016. Available online: https://www.lookout.com/trident-pegasus-enterprise-discovery [Accessed: 30th August 2016]

Pegasus Spyware: What You Need to Know Now. N.p., n.d. Web. 02 Sept. 2016. Available online: https://www.lookout.com/trident-pegasus-enterprise-discovery [Accessed: 30th August 2016]

Digit Oktavianto; Iqbal Muhardianto."Cuckoo Malware Analysis".Malware Analysis Methodologies.October,2015. ISBN: 978-1-78216-923-9

Nutan Kumar Panda; Sudhanshu Chauhan." Hacking Web Intelligence".May 1, 2015. ISBN-13: 978-0-12-801912-2

David Harley; Robert S. Vibert; Ken Bechtel; Michael Blanchard; Henk K. Diemer; Andrew Lee; Igor Muttik; Bojan Zdrnja. "AVIEN Malware Defense Guide for the Enterprise".April 18, 2011. ISBN: 978-1-59749-164-8

Allan Liska; Timothy Gallo."Ransomware". December 7, 2016. Print ISBN-13: 978-1-4919-6788-1

Stergios Theodoridis."Machine Learning". Academic Press. April 2, 2015. ISBN-13: 978-0-12-801722-7

John Aycock." Computer Viruses and Malware". 2006.ISBN 978-0-387-34188-0

Michael Beyeler."Machine Learning for OpenCV". July 14, 2017. Web ISBN-13: 978-1-78398-029-1. Print ISBN-13: 978-1-78398-028-4.

Michael Sikorski και Honig Andrew. *"Practical Malware Analysis. The Hands-On Guide to Dissecting Malicious Software*". English. Πρόλογος υπό Richard Bejtlich. No Starch Press, 2012, σσ. xxviii, 2, 10, 167. ISBN: 978-1-59327-290-6.

Szor, P. (2005) 'The Art of Computer Virus Research and Defense'. Addison-Wesley Professional. Available at: http://dl.acm.org/citation.cfm?id=1050957 (Accessed: 25 March 2016).

Oberhumer,M., M. (2004) *"The Ultimate Packer for eXecutables - Homepage*". Available at: http://upx.sourceforge.net/ (Accessed: 25 March 2016).

Konrad Rieck1, Philipp Trinius2, Carsten Willems2, and Thorsten Holz2,3 "Automatic Analysis of Malware Behavior using Machine Learning" (2011) Journal of Computer Security, IOS Press, http://www.iospress.nl σελ.30

Kateryna Chumachenko." MACHINE LEARNING METHODS FOR MALWARE DETECTION AND CLASSIFICATION",2017.

Aziz Mohaisen1 and Omar Alrawi." AMAL: High-Fidelity, Behavior-based Automated Malware Analysis and Classification."

K. Rieck, T. Holz, C. Willems, P. D¨ussel, and P. Laskov." Learning and classification of malware behavior". In Detection of Intrusions and Malware, and Vulnerability Assessment, pages 108–125, 2008.

Shahid Alam κ.ά. "A framework for metamorphic malware analysis and real-time detection".Στο: Computers & Security 48 (2015), σελ. 212–233. ISSN: 0167-4048. DOI: 10 . 1016 /j.cose.2014.10.011.URL:http://www.sciencedirect.com/science/article/pii/S0167404814001576.

Mariette Awad; Rahul Khanna." Efficient Learning Machines: Theories, Concepts, and Applications for Engineers and System Designers".(May 23, 2015)ISBN: 978-1-4302-5989-3. DOI: 10.1007%2F978-1-4302-5990-9

Manuel Egele κ.ά. "A Survey on Automated Dynamic Malware-analysis Techniques and Tools". Στο: ACM Comput. Surv. 48 (Μαρ. 2008), σελ.212-233.ISSN: 0360-0300. DOI 10 . 1145/2089125.2089126. URL: http://doi.acm.org/10.1145/2089125.2089126.

Aziz Mohaisen, Omar Alrawi και Manar Mohaisen. "AMAL: High-fidelity, behavior-based automated malware analysis and classification". Στο: *Computers & Security* 52 (2015),σελ. 251–266. ISSN: 0167-4048. DOI: 10 . 1016 / j . cose . 2015 . 04 . 001. URL: http ://www.sciencedirect.com/science/article/pii/S0167404815000425.

Shen, F. & Hasegawa, O., 2006. "An Incremental Network for On-line Unsupervised Classification and Topology Learning. *Neural Networks,"* 19(1), σελ. 90-106.

Andrej Karpathy blog.(25/10/2015)" The Unreasonable Effectiveness of Recurrent Neural Networks".

Python Software Foundation. Python Documentation. URL: https : / /docs .python . org/2/tutorial/index.html (επίσκεψη 01/07/2017).

A. Moser, C. Kruegel και E. Kirda. “Limits of Static Analysis for Malware Detection”. Στο: Twenty-Third Annual Computer Security Applications Conference (ACSAC 2007). Δεκ.2007, σελ. 421–430. DOI: 10.1109/ACSAC.2007.21.

Younghee Park κ.ά. “Fast Malware Classification by Automated Behavioral Graph Matching”.Στο: Proceedings of the Sixth Annual Workshop on Cyber Security and Information Intelligence Research. CSIIRW ’10. Oak Ridge, Tennessee, USA: ACM, 2010, 45:1–45:4. ISBN: 978-1-4503-0017-9. DOI: 10 . 1145 / 1852666 . 1852716. URL: http :// doi. acm. org /10 .1145/1852666.1852716.

Malin, C., Casey, E. & Aquilina, J., 2012. Chapter 6: Analysis of a Malware Specimen. In:Malware Forensics Field Guide For Windows Systems: Digital Forensics Guides.Massachuseetts, USA: Elsevier, pp. 444-446.

Nazmul Siddique & Hojjat Adeli "Computational Intelligence: Synergies of Fuzzy Logic, Neural Networks and Evolutionary Computing"2013 ISBN**:** 978-1-118-33784-4.

F. Pedregosa κ.ά. “Scikit-learn: Machine Learning in Python”. Στο: Journal of MachineLearning Research 12 (2011), σελ. 2825-2830.

Ping Wang και Yu-Shih Wang. “Malware behavioral detection and vaccine development by using a support vector model classifier”. Στο: *Journal of Computer and System Sciences*"81.6 (2015). Special Issue on Optimisation, Security, Privacy and Trust in E-business

Ed Skoudis και Lenny Zeltser. Malware: Fighting Malicious Code. Prentice Hall PTR, 2003, σ. 23. ISSN: 0-13-101405-6.

Ιωάννης Βλαχαβάς κ.ά. Τεχνητή Νοημοσύνη. 3η έκδοση. Εκδόσεις Πανεπιστημίου Μακεδονίας, Ιούν. 2011, σ. 336. ISSN: 978-960-8396-64-7. URL: http://aibook.csd.auth.gr.

Ιωάννης Μάριος Κοζυράκης. “Τεχνικές ανίχνευσης rootkit και ανάπτυξη εφαρμογής για την αφαίρεσή του”. Διπλωματική Εργασία. Πανεπιστήμιο Πατρών, 2009, σ. 3.