ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ – ΤΜΗΜΑ ΨΗΦΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ
ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ
«ΑΣΦΑΛΕΙΑ ΨΗΦΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ»

Ανάπτυξη Μηχανισμού Εντοπισμού Κακόβουλου Λογισμικού σε Εφαρμογές Διαδικτύου με Χρήση Αλγορίθμου Μηχανικής Μάθησης

Development of a Mechanism that Detects Malicious Software on Web Applications using Machine Learning Algorithm

Τίτλος Διατριβής	Ανάπτυξη Μηχανισμού Εντοπισμού Κακόβουλου Λογισμικού σε Εφαρμογές Διαδικτύου με Χρήση Αλγορίθμου Μηχανικής Μάθησης
Ονοματεπώνυμο Φοιτητή	Τριανταφύλλου Γεώργιος
Πατρώνυμο	Δημήτριος
Αριθμός Μητρώου	MTE14026
Επιβλέπων	Δρ. Νταντογιάν Χριστόφορος
Ημερομηνία Παράδοσης	Απρίλιος 2017

Η σελίδα αφέθηκε σκοπίμως κενή

Τριμελής Εξεταστική Επιτροπή

Ο επιβλέπων	1ος Συνεξεταστής	2ος Συνεξεταστής
Χ. ΝΤΑΝΤΟΓΙΑΝ	Κ. ΛΑΜΠΡΙΝΟΥΔΑΚΗΣ	Χ. ΞΕΝΑΚΗΣ
(υπογραφή)	(υπογραφή)	(υπογραφή)

Η σελίδα αφέθηκε σκοπίμως κενή

Ευχαριστίες

Η παρούσα διπλωματική διατριβή πραγματοποιήθηκε στα πλαίσια του μεταπτυχιακού προγράμματος «Ασφάλεια Ψηφιακών Συστημάτων» του Πανεπιστημίου Πειραιώς του οποίου είχα την χαρά και τιμή να υπάρξω φοιτητής. Ολοκληρώνοντας αυτό το στάδιο της ζωής μου, νιώθω την ανάγκη να ευχαριστήσω όλους όσους μου συμπαραστάθηκαν σε αυτή την απαιτητική για έμενα περίοδο. Αρχικά θα ήθελα να ευχαριστήσω τους καθηγητές του πανεπιστημίου κ.κ. Κάτσικα, Λαμπρινουδάκη, Ξενάκη για τις γνώσεις που αποκόμισα όντας φοιτητής τους. Επίσης θα ήθελα να ευχαριστώ θερμά τον Δρ. Νταντογιάν ως καθηγητή και κυρίως ως υπεύθυνο της παρούσας διατριβής για την εμπιστοσύνη που έδειξε στο άτομο μου αναθέτοντας μου ένα τόσο σημαντικό έργο, την πολύτιμη καθοδήγηση του, την συνεργασία και τις γνώσεις που αποκόμισα με την βοήθεια του σε έναν τόσο ιδιαίτερο κλάδο όπως αυτόν της πληροφορικής .

Ιδιαιτέρες ευχαριστίες θα ήθελα να αποδώσω στους Μιχάλη Ηλιόπουλο, Ιωάννη Πετρόπουλο και Αθανασία Κούρμπαση, συμφοιτητές και φίλοι μου για την ομαδικότητα, την σοβαρότητα και αγαστή συνεργασία που επιδείξαν ως μέλη των ομαδικών εργασιών που αναλάβαμε καταφέρνοντάς έτσι να ολοκληρώσουμε επιτυχώς τις σπουδές μας.

Θα ήθελα επίσης να ευχαριστήσω τους γονείς μου Δημήτριο και Μαρία καθώς και τον αδελφό μου Γρηγόρη για την στήριξη που μου παρείχαν όλα τα χρόνια των σπουδών μου.

Τέλος, θα ήθελα να ευχαριστήσω την αγαπημένη σύζυγο μου Βίκυ, η οποία με την ηθική της συνεισφορά, την παρότρυνση, την εμπιστοσύνη και την συνεχή στήριξη της στις επιλογές μου, με βοηθά να εξελίσσομαι ακόμα περισσότερο τόσο ως επιστήμων όσο και ως άνθρωπο.

Η σελίδα αφέθηκε σκοπίμως κενή

Πίνακας Περιεχομένων

Η σελίδα αφέθηκε σκοπίμως κενή

Περίληψη

Η παρούσα διατριβή επιχειρεί να δώσει λύση σε μία από τις σημαντικότερες απειλές που καλείται να αντιμετωπίσει ένας διαχειριστής ιστοτόπου, αυτό του εντοπισμού αγνώστου κακόβουλου λογισμικού, όταν αυτό προσβάλει τα συστήματα τους, καθώς οι περισσότεροι μηχανισμοί anti-malware αδυνατούν να εντοπίσουν έγκαιρα τέτοιου είδους λογισμικά, με αποτέλεσμα την έκθεση σε κίνδυνο αυτών των συστημάτων να είναι σχεδόν βεβαία.

Σε αντίθεση με τα υφιστάμενα αντιμέτρα, ο παρόν μηχανισμός εντοπισμού κακόβουλου λογισμικού σχεδιάστηκε και υλοποιήθηκε με την εφαρμογή εργαλείων ανοιχτού κώδικα και είναι σε θέση να εντοπίζει κακόβουλο λογισμικό με χρήση μηχανισμών μηχανικής μάθησης και συγκεκριμένα με τον αλγόριθμο Naïve Bayes. Ο Naïve Bayes μετά από την απαραίτητη εκπαίδευση, έχει την δυνατότητα να είναι σε θέση να εντοπίζει με μεγάλο ποσοστό επιτυχίας άγνωστο κυρίως κακόβουλο λογισμικό.

Στην συνέχεια σχεδιάστηκαν δύο δοκιμασίες έτσι ώστε να γίνει όσο το δυνατόν ακριβέστερη η μέτρηση της απόδοσης του μηχανισμού, ενώ οι αξιολογήσεις έγιναν βασιζόμενες στις αρχές της αποτίμησης της αποτελεσματικότητας. Στις δοκιμασίες αυτές έγινε σύγκριση των αποτελεσμάτων με επιπλέον τρεις αντίστοιχους μηχανισμούς οι οποίοι είναι ανοιχτού κώδικα και χρησιμοποιούνται ευρέως για τον εντοπισμό κακόβουλου λογισμικού από τους διαχειριστές εξυπηρετητών διαδικτύου.

Τα αποτελέσματα τα οποία εμφανίζει ο μηχανισμός κρίνονται ιδιαιτέρως ικανοποιητικά καθώς αυτός καθίσταται ικανός σύμφωνα με το επίπεδο της εκπαίδευσης του να εντοπίσει το σύνολο του γνωστού αλλά και αγνώστου κακόβουλου λογισμικού και ταυτόχρονα να ελαχιστοποιήσει τα σφάλματα τύπου false positive.

Λέξεις Κλειδιά: Μηχανική Μάθηση, Naïve Bayes, Python, Μηχανισμός Ανίχνευσης, Κακόβουλο Λογισμικό

Η σελίδα αφέθηκε σκοπίμως κενή

Abstract

This postgraduate thesis aims to resolve one of the most serious threats web administrators face in their jobs: Quick identification of unknown malicious software, when this compromises their systems. Using the current solutions, it is not possible for this malicious software to be directly detected thus resulting in the jeopardization of information systems.

Unlike currently existing counter-measures, this project has been designed and developed to be able to detect malicious software using machine learning and the Naïve Bayes algorithm, which is used in spam filters and has the ability to detect, with high percentage, an unknown malicious file.

Two tests were designed to measure the results of the anti-malware project. These were based on the fundamentals of assessing the effectiveness. In these tests, the project was compared, in terms of effectiveness, with three other open source anti-malware applications, which are broadly used by web administrators in order to detect malicious software.

The final results of the mechanism showed that it can achieve high detection rates, because it detected all of the malicious software that had been injected to the server, whereas it also decreases false positive rate errors drastically.

Keywords: Machine Learning, Naïve Bayes, Python, anti-malware mechanism, malicious software

Η σελίδα αφέθηκε σκοπίμως κενή

Εισαγωγή

Η αλματώδης τεχνολογική εξέλιξη των τελευταίων δεκαετιών έχει συντελέσει καταλυτικά στην ανάπτυξη και διάδοση του ίντερνετ ώστε αυτό να γίνει αναπόσπαστο κομμάτι της καθημερινότητας των πολιτών, του δυτικού κυρίως κόσμου, όχι μόνο ως μέσω επικοινωνίας, αλλά και ως μέσο συναλλαγής με το δημόσιο και ιδιωτικό τομέα. Έτσι ένας πολίτης έχει στο διαδίκτυο κρίσιμες πληροφορίες που τον αφορούν, όπως στοιχεία της προσωπικής του ζωής, το κοινωνικό του δίκτυο, την οικονομική του κατάσταση, τα ήθη, τα πολιτικά και θρησκευτικά του πιστεύω, τις ιδιαίτερες προσωπικές του προτιμήσεις, καθώς και άλλα στοιχεία όπως , οικονομικά, προσωπικά κλπ. Η αύξηση αυτή των διαθέσιμων πληροφοριών των χρηστών, έχει ως αποτέλεσμα, οι ιστότοποι, να αποτελούν στόχο κυβερνοεπιθέσεων και πηγές άντλησης των δεδομένων που φιλοξενούν, μέσω της εξαπάτησης του μέσου χρήστη, με σκοπό το οικονομικό κέρδος ή ακόμα και την προώθηση πολιτικών, θρησκευτικών και άλλων, παράνομων πολλές φορές, πεποιθήσεων και ιδεών.

Έτσι, όταν ένας ο ιστότοπος παραβιαστεί, οι διαχειριστές του εκτελούν ελέγχους με προγράμματα anti-malware με σκοπό τον εντοπισμό της κερκόπορτας ή άλλου είδους λογισμικού έτσι ώστε να τον επαναφέρουν στην αρχική του κατάσταση. Το πρόβλημα με τις υπάρχουσες τεχνικές είναι ότι βασίζονται κυρίως στην τεχνική εντοπισμού «ύποπτων» συναρτήσεων ή υπογραφών, οι οποίες χρησιμοποιούνται κυρίως από κακόβουλες εφαρμογές, καθώς και την καταγραφή ενός κακόβουλου αρχείου συνήθως με την αποθήκευση της τιμής συνάρτησης κατακερματισμού (όπως η MD5 ή SHA1). Αυτές οι τεχνικές έχουν τρία βασικά ελαττώματα:

α) Στην περίπτωση εντοπισμού ύποπτων συναρτήσεων, η εφαρμογή παρουσιάζει έναν μεγάλο αριθμό σφαλμάτων true - negative καθώς η ύπαρξη μιας ή περισσοτέρων «ύποπτων» συναρτήσεων στα υπό εξέταση αρχεία, δεν αποτελεί αξιόπιστο στοιχείο, για την ύπαρξη ή όχι κακόβουλου κώδικα.

β) Στην περίπτωση καταγραφής της τιμής κατακερματισμού των αρχείων, απαιτείται το αρχείο να είναι ήδη γνωστό στους μηχανισμούς, αλλά λόγω της φύσης των εν λόγω μηχανισμών δημιουργίας σύνοψης, η παραμικρή αλλαγή στα περιεχόμενα του αρχείου θα οδηγήσει σε τελείως διαφορετική τιμή με αποτέλεσμα, η σύγκριση μεταξύ των τιμών να οδηγήσει σε λανθασμένο συμπέρασμα.

γ) Σε όλες τις περιπτώσεις το κακόβουλο λογισμικό πρέπει να είναι γνωστό στον μηχανισμό. Καθώς οι κακόβουλοι χρήστες γνωρίζουν αυτήν αδυναμία, είναι σε θέση να τροποποιούν ή να αναβαθμίζουν τους υπάρχοντες κακόβουλους κώδικες ή ακόμα και να τους επανασχεδιάζουν, με σκοπό να ξεγελούν τους μηχανισμούς αντιμετώπισης πετυχαίνοντας έτσι την όσο το δυνατόν μεγαλύτερη διάρκεια παραμονής τους στους ιστότοπους.

Η μηχανική μάθηση, αν και ως έννοια υφίσταται από το 1959 όταν ο Arthur Samuel ορίζει τη μηχανική μάθηση ως "*Πεδίο μελέτης που δίνει στους υπολογιστές την ικανότητα να μαθαίνουν, χωρίς να έχουν ρητά προγραμματιστεί*" [1] εν τούτοις

άρχισε να αξιοποιείται τα τελευταία χρόνια που τα υπολογιστικά συστήματα είναι σε θέση να εκτελούν πολύπλοκους μαθηματικούς υπολογισμούς σε γρήγορο χρονικό διάστημα. Έτσι, μέσω των αλγορίθμων μηχανικής μάθησης, ο υπολογιστής είναι σε θέση να διαχωρίσει ένα email αν είναι ανεπιθύμητο (spam) ή όχι, να «διαβάσει» οπτικούς χαρακτήρες από κάμερες (OCR) κ.α.

Στο 1^{ο} κεφάλαιο παρουσιάζονται τα είδη του κακόβουλου λογισμικού που εγκαθίσταται στους σύγχρονούς ιστότοπους, τους λόγους που εγκαθίσταται ένα τέτοιο λογισμικό, πως επιδρούν σε αυτόν, καθώς και τους υφιστάμενους μηχανισμούς αντιμετώπισης τους.

Στο 2^{ο} κεφάλαιο παρουσιάζεται ο τρόπος λειτουργίας του μηχανισμού που αναπτύχθηκε, γίνεται ανάλυση των εσωτερικών λειτουργιών ανίχνευσης και τέλος προβάλλονται στιγμιότυπα από την εκτέλεση του.

Στο 3^{ο} κεφάλαιο γίνεται μια αναφορά περί της μηχανικής μάθησης, δίδεται επεξήγηση γιατί επιλέχθηκε ο αλγόριθμος Naïve Bayes και γίνεται ανάλυση μέσω παραδειγμάτων του τρόπου αξιοποίησης του από τον υπό ανάπτυξη μηχανισμό.

Στο 4^{ο} κεφάλαιο δίδεται επεξήγηση του τρόπου εκπαίδευσης και γίνεται καταγραφή του δειγματικού χώρου στον οποίο εισήχθησαν οι πληροφορίες του μηχανισμού μηχανικής μάθησης.

Στο 5^{ο} κεφάλαιο αναλύεται η περίοδος κατά την οποία δοκιμάστηκε ο υπό ανάπτυξη μηχανισμός, περιγράφεται η μεθοδολογία καταγραφής και αξιολόγησης, γίνεται η σύγκριση μεταξύ του μηχανισμού και τριών (3) επιπλέον μηχανισμών ανοιχτού κώδικα που χρησιμοποιούνται ως αντίμετρα για την αντιμετώπιση κακόβουλου λογισμικού και παρουσιάζονται τέλος τα προβλήματα και οι αδυναμίες του κάθε ενός ξεχωριστά. Τέλος πραγματοποιείται εκτέλεση ελέγχου σε περιβάλλον με πραγματικές συνθήκες ήτοι σε ιστότοπο ο οποίος είχε παραβιαστεί και στον οποίο είχαν εγκατασταθεί περισσότερες από 32 άγνωστες στον μηχανισμό κερκόπορτες, από όπου και καταγράφηκαν τα αποτελέσματα του εν λόγω ελέγχου.

Στο 6^{ο} κεφάλαιο παρουσιάζονται τα συμπεράσματα και οι παρατηρήσεις για την ορθή λειτουργία του μηχανισμού ως προς τον εντοπισμό κακόβουλου λογισμικού και κατά πόσο είναι εφικτός ο περιορισμός τυχόν λαθών με σκοπό την βέλτιστη λειτουργία του.

Τέλος στο 7^{ο} κεφάλαιο και επίλογο της διατριβής συμπεραίνουμε κατά πόσο ο μηχανισμός είναι σε θέση να αξιοποιηθεί σε πραγματικές συνθήκες και να αποδώσει ικανοποιητικά αποτελέσματα.

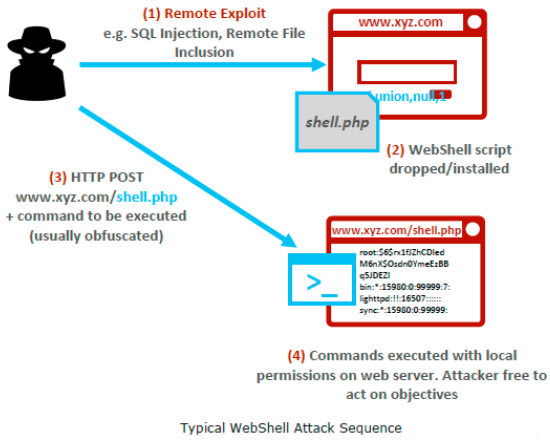
Κεφάλαιο 1^{ο} - Είδη κακόβουλου λογισμικού ειδικά σε εξυπηρετητές διαδικτύου.

Όταν ένας κακόβουλος χρήστης έχει καταφέρει να παραβιάσει τους μηχανισμούς ασφαλείας ενός διακομιστή διαδικτύου και έχει αποκτήσει πρόσβαση στα αρχεία και στις υπηρεσίες του, το πρώτο βήμα που θα πραγματοποιήσει είναι η εγκατάσταση κακόβουλου (malicious) λογισμικού είναι μία συνήθης και ευρέως γνωστή διεργασία, το αποτέλεσμα της οποίας αποκαλείται **Προηγμένη Μόνιμη Απειλή** (Advanced Persistent Threat). Αυτό αποσκοπεί στην εύκολη και κατά το δοκούν είσοδο του εισβολέα στο σύστημα, χωρίς να είναι αντιληπτό από τους διαχειριστές και φυσικά χωρίς να χρειάζεται να εκτελεστεί μια σαφώς δυσκολότερη διαδικασία παραβίασης, λόγω κάποιας ευπάθειας του λογισμικού η οποία να έχει εσφαλμένα θεωρηθεί ότι έχει επιλυθεί. Το λογισμικό αυτό βέβαια δεν εκτελείται με την ίδια επιτυχία σε όλους τους εξυπηρετητές καθώς απαιτούνται επιπρόσθετες γνώσεις όπως το λειτουργικό σύστημα, η γλώσσα σεναρίων (scripting language) για την οποία (php ή asp κλπ,) είναι παραμετροποιημένος ο web server κλπ. Γνώσεις τις οποίες όμως ένας μέσος χρήστης δεν κατέχει και έτσι δεν μπορεί να εντοπίσει στην φάση της συλλογής πληροφορίων και εφόσον δεν έχουν τηρηθεί τα στοιχειώδη μέτρα προφύλαξης από τον διαχειριστή. Αυτού του είδους το λογισμικό ανάλογα με το είδος της εργασίας που εκτελείται χωρίζεται στις ακόλουθες κατηγορίες:

1.1 - Απομακρυσμένη Πρόσβαση (Remote Access).

Σκοπός του κακόβουλου λογισμικού είναι η δυνατότητα επιτυχημένης απομακρυσμένης πρόσβασης στα περιεχόμενα του εξυπηρετητή, με τρόπο να παρακάμπτονται όλοι οι μηχανισμοί ταυτοποίησης του νόμιμου χρήστη.

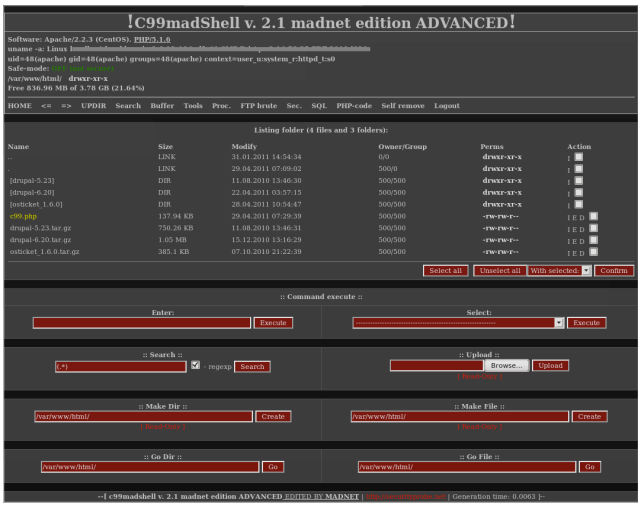
Αρχικά αυτά τα προγράμματα δημιουργήθηκαν για να εκτελούν βασικές εντολές όπως προβολή αρχείων, τροποποίηση, διαγραφή, μεταφορά αρχείων από και προς τον server, καθώς και μια υποτυπώδης γραμμή εντολών για την εκτέλεση εντολών του συστήματος.



Εικόνα 1 Σχηματική Απεικόνιση Απομακρυσμένης Πρόσβασης

Στην συνέχεια όμως εμπλουτίστηκαν με επιπλέον δυνατότητες όπως πλοήγηση και επεξεργασία βάσης δεδομένων, λειτουργίες επίθεσης ωμής βίας (brute force attack) σε υπηρεσίες όπως ftp ή ssh κλπ. Το βασικό πλεονέκτημα τους είναι ότι δεν απαιτείται συγκεκριμένο λογισμικό για την εκτέλεση τους, καθώς είναι σχεδιασμένα να εκτελούνται όπως οι σελίδες που σερβίρονται από τον εξυπηρετητή. Αυτό σημαίνει ότι αρκεί μια απλή τοποθέτηση ενός τέτοιου κακόβουλου λογισμικού, στον δημόσιο τομέα του server (εφόσον αυτό είναι εφικτό) και στην συνέχεια πληκτρολογώντας από οποιοδήποτε φυλλομετρητή την πλήρης διαδρομή του, τότε αυτό εκτελείται άμεσα.

Παράδειγμα ενός τέτοιου είδους λογισμικού μπορούμε να το διακρίνουμε στην Εικόνα 2. Πρόκειται για το κακόβουλο πρόγραμμα C99 το οποίο λόγω των δυνατοτήτων του είναι ιδιαίτερα γνωστό στην κοινότητα των hackers καθώς προσφέρει μια πλειάδα βοηθημάτων, τα οποία ο εισβολέας μπορεί να τα αξιοποιήσει έτσι ώστε να εκτελέσει επιτυχώς επιθέσεις.



Εικόνα 2 στιγμιότυπο εικόνας λειτουργίας του κακόβουλου λογισμικού C99

1.2 - Κερκόπορτες (backdoors).

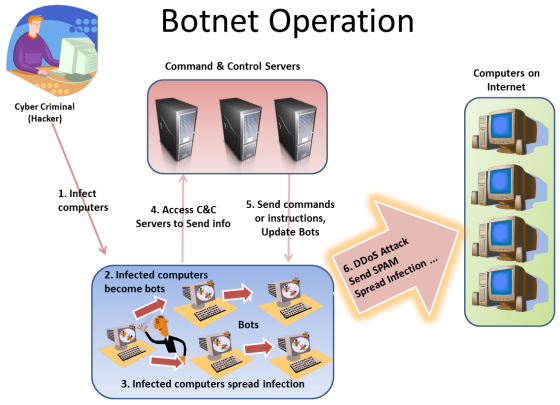
Πρόκειται για αρχεία τα οποία επιτρέπουν στον επιτιθέμενο να αποκτήσει πρόσβαση στην γραμμή εντολών του εξυπηρετητή εκτελώντας λειτουργίες εκμεταλλευόμενος πιθανές ευπάθειες τις οποίες είναι σε θέση να εντοπίσει οι οποίες, να μπορεί να τον οδηγήσουν στην ανέλιξη του σε χρήστη με δικαιώματα διαχειριστή και κατά συνέπεια στην απόκτηση του πλήρη ελέγχου του υπολογιστικού συστήματος. Για την λειτουργεία συνήθως χρησιμοποιείται ένα πρόγραμμα διασύνδεσης όπως το *netcat* , αν και πλέον ένας προχωρημένος εισβολέας είναι πιθανό να χρησιμοποιήσει ένα εργαλείο σαν το *meterpreter*. Παράδειγμα αυτού του είδους επίθεσης βλέπουμε στην Εικόνα 3 από όπου ο επιτιθέμενος έχει αποκτήσει τον έλεγχο του διακομιστή και μπορεί να εκτελέσει εντολές του προσβεβλημένου συστήματος.

```
root@ddos:/usr/share/weevely# weevely http://192.168.1.8/dvwa/hackable/uploads/dd
os.php ddos

[+] weevely 3.2.0

[+] Target:     www-data@ddos:/etc
[+] Session:    /root/.weevely/sessions/192.168.1.8/ddos_0.session
[+] Shell:      System shell

[+] Browse the filesystem or execute commands starts the connection
[+] to the target. Type :help for more information.

weevely>
```

Εικόνα 3 Η κερκόπορτα weevely

1.3 - Κέντρο Διαχείρισης δικτύου Botnet (Command and Control Center)

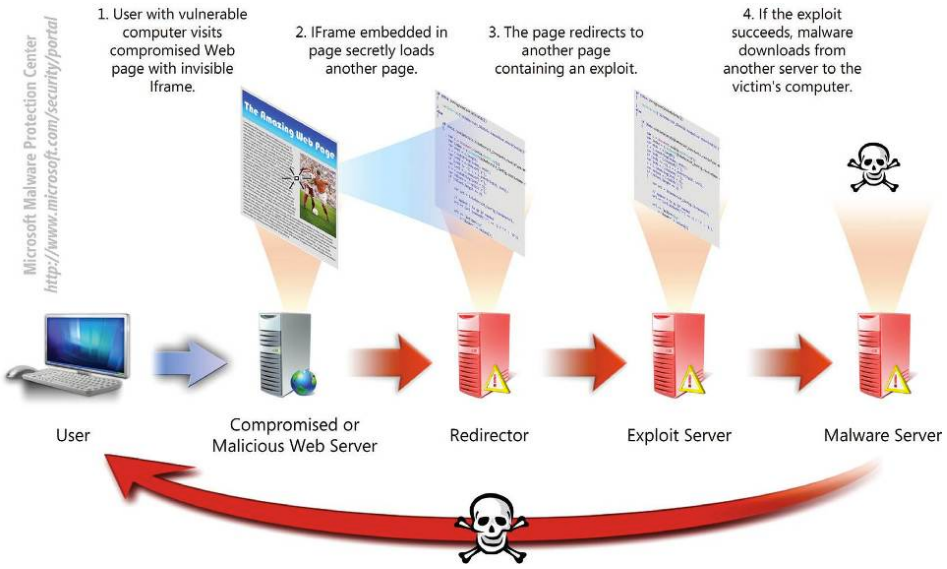
Πρόκειται για μηχανισμό ο οποίος δεν εμπλέκεται με τον τρόπο λειτουργίας του ιστοτόπου. Μάλιστα οι απλοί χρήστες ή πελάτες του εξυπηρετητή δεν θα αντιληφθούν την ύπαρξη του κακόβουλου λογισμικού καθώς ο στόχος του επιτιθέμενου δεν είναι οι χρήστες του, αλλά η διαχείριση ενός δικτύου τύπου botnet με υπολογιστές μολυσμένους με κακόβουλο λογισμικό, γνωστούς ως «ζόμπι» ή bots το δε δίκτυο καλείται botnet. Η διαχείριση αυτών των υπολογιστών απαιτεί την ύπαρξη ενός τουλάχιστον κέντρου ελέγχου (Command & Control Center), από το οποίο θα αποστέλλονται οι εντολές του κακόβουλου χρήστη στους υπολογιστές στόχους. Αυτό συμβαίνει διότι αυτά τα Control Centers απαιτούν αρκετή υπολογιστική ισχύ, διαρκή σύνδεση με τους μολυσμένους υπολογιστές και αρκετά μεγάλο εύρος ζώνης (bandwidth) καθώς συντονίζουν εκατοντάδες ή και χιλιάδες bots. Όλα τα παραπάνω απαιτούν υψηλό κόστος για την αγορά και την συντήρηση τους. Καθώς ο στόχος του διαχειριστή ενός τέτοιου δικτύου είναι να παραμένει ανώνυμος καθ' όλη την διάρκεια της επίθεσης, ευπαθείς ιστότοποι καθίστανται ιδανικοί στόχοι σε κυβερνοεπιθέσεις. Στην Εικόνα 4 βλέπουμε πως ένας κακόβουλος χρήστης μπορεί να εκμεταλλευτεί έναν εξυπηρετητή ως κέντρο ελέγχου ενός δικτύου botnet σε επιθέσεις τύπου «*κατανεμημένης επίθεσης άρνησης υπηρεσιών*"(DDOS). Τελευταία δε οι επιθέσεις αυτές παρέχουν την δυνατότητα εκμετάλλευσης των υπολογιστικών πόρων των μελών ενός τέτοιου δικτύου, για την δημιουργία κρυπτονομισμάτων τύπου bitcoins [2].



Εικόνα 4 σχηματική απεικόνιση επίθεσης DDOS με χρήση botnet

1.4 - Drive by Download attack

Αυτή η επίθεση στοχεύει αποκλειστικά τους πελάτες του μολυσμένου Server. Στόχος του επιτιθέμενου είναι η αλλοίωση ενός ιστότοπου με τέτοιο τρόπο ώστε αυτός να εγκαταστήσει ειδικό μηχανισμό, ο οποίος με συγκεκριμένες μεθοδολογίες εντοπίζει αν ο φυλλομετρητής (browser) του επισκέπτη, περιέχει γνωστές ή άγνωστες (0-day) ευπάθειες. Σε καταφατική περίπτωση ανάλογα και με το είδος της ευπάθειας που θα εντοπίσει, τον κατευθύνει «σιωπηρά» σε άλλον ή άλλους ιστότοπους. Οι ιστότοποι αυτοί είναι ήδη παραβιασμένοι και το κακόβουλο λογισμικό έχει εγκατασταθεί εν αγνοία των διαχειριστών ή οι server βρίσκονται σε χώρα που δεν εφαρμόζονται πολιτικές ταυτοποίησης των διαχειριστών. Όταν η ανωτέρω διαδικασία ολοκληρωθεί φτάνοντας στον τελικό ιστότοπο, εκμεταλλευόμενο την ευπάθεια που έχει εντοπίσει, εγκαθιστά στην υπολογιστική πλατφόρμα του πελάτη το κακόβουλο λογισμικό το οποίο είτε έχει την μορφή ransomware, δηλαδή τοποθετείται μηχανισμός ο οποίος κρυπτογραφεί τα προσωπικά αρχεία του χρήστη με σκοπό την καταβολή λύτρων για να γίνει η αποκρυπτογράφηση τους, είτε εγκαθιστά μηχανισμό ο οποίος εντάσσει τον υπολογιστή στόχο σε δίκτυο botnet, είτε με την μορφή ιομορφικού λογισμικού με σκοπό την συλλογή ατομικών / εταιρικών δεδομένων ή οποιαδήποτε άλλη κακόβουλη χρήση επιλέξει ο επιτιθέμενος. Σχηματική απεικόνιση της ανωτέρω επίθεσης εμφανίζεται στην Εικόνα 5.



Εικόνα 5 Επίθεση τύπου Drive by Download

Κεφάλαιο 2 - Μεθοδολογίες Εντοπισμού Κακόβουλου Λογισμικού και η Εφαρμογή τους στον υπό ανάπτυξη Μηχανισμό

Κατά την διάρκεια εκπόνησης της εργασίας μελετήθηκαν διάφορες μεθοδολογίες οι οποίες χρησιμοποιούνται συχνά από τους υπάρχοντες μηχανισμούς για τον εντοπισμό κακόβουλων αρχείων. Πολλές από τις υπάρχουσες μεθοδολογίες δεν βρίσκουν εφαρμογή σε γλώσσες σεναρίου, οπότε και δεν αναφέρονται στην παρούσα εργασία. Αναφέρεται επιγραμματικά ότι οι μεθοδολογίες που χρησιμοποιούνται για τον εντοπισμό κακόβουλου λογισμικού χωρίζονται σε δύο κυρίως κατηγορίες ανάλογα με τον τρόπο ανάλυσης τους. Την στατική και στην δυναμική ανάλυση[3].

α. Η στατική ανάλυση αναζητά συγκεκριμένα μοτίβα λέξεων, συγκρίνει γνωστές υπογραφές με σκοπό τον εντοπισμό λέξεων οι οποίες έχουν ήδη εντοπιστεί, στοχεύοντας κυρίως σε γνωστό κακόβουλο λογισμικό με αποτέλεσμα να έχει μεγάλα ποσοστά επιτυχίας στον εντοπισμό αυτών, αλλά παράλληλα όταν πρόκειται για άγνωστο ως προς την ανάλυση λογισμικό να εμφανίζονται συχνά True Negative σφάλματα .

β. Η δυναμική ανάλυση η οποία αφού εκτελέσει τον ύποπτο κώδικα συνήθως σε κάποιο απομονωμένο περιβάλλον, στην συνέχεια παρακολουθεί τις διεργασίες που επιτελεί και αν εντοπίσει κάποια ύποπτη δραστηριότητα τότε απομονώνει το αρχείο για περαιτέρω ανάλυση.

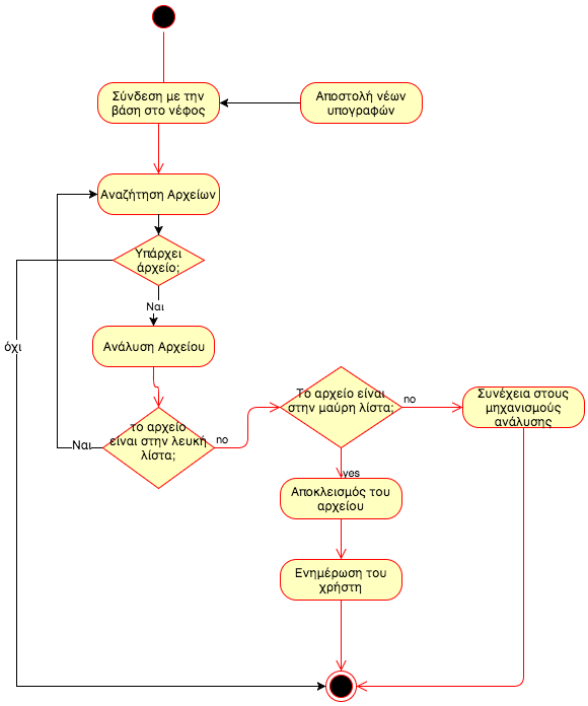
2.1 - Χαρτογράφηση και έλεγχος ακεραιότητας των αρχείων του ιστότοπου.

Αυτή η μέθοδος είναι από τις πρώτες που έχουν χρησιμοποιηθεί για την αντιμετώπιση κυρίως κακόβουλου ιομορφικού λογισμικού. Ο διαχειριστής έχει στην κατοχή του ένα πρόγραμμα το οποίο χαρτογραφεί τις θέσεις και το όνομα των αρχείων ενός φακέλου και των υποφακέλων του[4]. Από αυτά δημιουργείται για το κάθε αρχείο ένα στιγμιότυπο (hash value) το οποίο και αποθηκεύεται σε μια βάση δεδομένων. Όποτε γίνεται έλεγχος των αρχείων αυτών η τιμή τους πρέπει να παραμένει αμετάβλητη, εκτός των περιπτώσεων νόμιμης τροποποίησης των, την οποία όμως οφείλει να γνωρίζει ο διαχειριστής οπότε και θα ενημερώσει την ανωτέρω βάση. Σε διαφορετική περίπτωση ή σε κάθε εντοπισμό νέου αρχείου το πρόγραμμα ενημερώνει τον διαχειριστή ο οποίος στην συνέχεια ελέγχει αν τα νέα αρχεία έχουν προστεθεί ή τροποποιηθεί νόμιμα. Σε κάθε περίπτωση όμως αυτή η μέθοδος για να λειτουργήσει αποτελεσματικά, θα πρέπει να έχει γίνει η καταγραφή των αρχείων πριν ο webserver μολυνθεί και ο διαχειριστής να έχει συνεχή υποτύπωση όλων των νόμιμων μεταβολών των αρχείων, πράγμα το οποίο όμως δεν είναι εφικτό, ειδικά στους μεγάλης κλίμακας ιστότοπους.

Επίσης, μέσω του ελέγχου ακεραιότητας των αρχείων μπορεί όχι μόνο να εντοπιστεί αλλά και να ταυτοποιηθεί ένα κακόβουλο λογισμικό, αν η υπογραφή τού είναι ήδη περασμένη στην βάση δεδομένων του μηχανισμού ανίχνευσης. Αυτό

βέβαια προϋποθέτει τον εντοπισμό του κακόβουλου προγράμματος σε προγενέστερό χρόνο έτσι ώστε αυτό να έχει εκ των προτέρων αναλυθεί, και να έχει δημιουργηθεί η υπογραφή του και αυτή να έχει εισαχθεί στην βάση δεδομένων. Όμως αυτή η μέθοδος εύκολα εξαπατάται αν τροποποιηθεί έστω και ελάχιστα ο αρχικός κώδικας, καθώς στους μηχανισμούς ελέγχου ακεραιότητας που χρησιμοποιούνται, οι τιμές εισόδου μεταβάλλουν πλήρως την τιμή εξόδου με αποτέλεσμα και η παραμικρή μεταβολή της, να επιφέρει ένα εντελώς διαφορετικό αποτέλεσμα από το αρχικό.

Λαμβάνοντας υπόψη τα παραπάνω σχεδιάστηκε αλγόριθμος ο οποίος εκτελεί τις αυτές τις λειτουργίες, σχηματική απεικόνιση των οποίων βλέπουμε στην Εικόνα 6. Υπάρχουν δύο μηχανισμοί: Ο πρώτος ελέγχει αν το αρχείο είναι στην «μαύρη λίστα» και ένας δεύτερος ο οποίος ελέγχει αν το αρχείο είναι στην «λευκή λίστα». Στην μαύρη λίστα τοποθετούνται τα αρχεία τα οποία έχουν ήδη ταυτοποιηθεί ως κακόβουλα. Αυτή η λίστα δημιουργήθηκε κατά την διάρκεια της εκπαίδευσης που θα εξηγήσουμε παρακάτω και βρίσκεται εγκατεστημένη μαζί με το πρόγραμμα. Επιπλέον, σε κάθε επιβεβαιωμένη ύπαρξη κακόβουλου λογισμικού, η λίστα ενημερώνεται με τις νεότερες πληροφορίες, επιλέγοντας το αντίστοιχο μενού επιλογών. Επικουρικά, έχει δημιουργηθεί μια βάση δεδομένων σε υπολογιστικό νέφος, στην οποία συλλέγονται οι υπογραφές των επιβεβαιωμένων κακόβουλων αρχείων. Οι χρήστες του προγράμματος παροτρύνονται ώστε να συμβάλλουν στην αποστολή των υπογραφών και των αρχείων που έχουν εντοπίσει έτσι ώστε με κάθε ύπαρξη νέου κακόβουλου λογισμικού οι υπόλοιποι χρήστες να ενημερώνονται και να τις προλαμβάνουν πριν αυτές χτυπήσουν και τα δικά τους υπολογιστικά συστήματα.



Εικόνα 6 Διάγραμμα ροής μηχανισμού "λευκής" και "μαύρης" λίστας

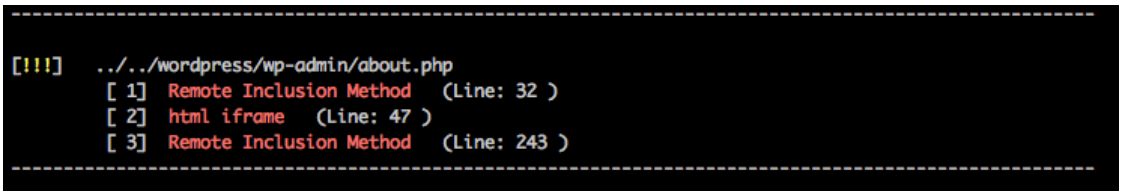
Στην «λευκή» λίστα εισάγονται όλα τα αρχεία του ιστότοπου, κατά προτίμηση αμέσως μετά την εγκατάσταση του. Στα αρχεία αυτά υπολογίζεται για το κάθε ένα χωριστά, μια τιμή hash της μορφής SSDeep [5] και αποθηκεύεται μαζί με το όνομα και την πλήρη διαδρομή του αρχείου σε μια βάση δεδομένων. Η συνάρτηση SSDeep είναι μια συνάρτηση κατακερματισμού στην οποία δοθείσας μιας τιμής εισόδου οποιασδήποτε μεγέθους επιστρέφει μια τιμή κατακερματισμού μοναδική για κάθε διαφορετική είσοδο που δέχεται. Ο λόγος που χρησιμοποιήθηκε αυτός ο αλγόριθμος είναι γιατί ενώ όλες τις γνωστές συναρτήσεις όπως η MD5 ή ο SHA, όπου όταν η τιμή εισόδου έχει δεχτεί μια μικρή μεταβολή, αυτές επιστρέφουν ένα εντελώς διαφορετικό αποτέλεσμα σε σχέση με το προηγούμενο, αντίθετα η τιμή που επιστρέφεται από την SSDeep έχει υποστεί αλλοίωση τόση, όση έχει υποστεί και το προς σύγκριση αρχείο. Αυτό κατά την διάρκεια των δοκιμών απεδείχθη ιδιαίτερα αποτελεσματικό ως μέτρο εντοπισμού καθώς η εισαγωγή ενός κακόβουλου κώδικα στα περιεχόμενα των αρχείων του ιστότοπου εντοπίζονταν με επιτυχία. επίσης, και τα «ασφαλή» αρχεία μπορούσαν να τροποποιηθούν , με μικρές βέβαια αλλαγές, και εξακολουθώντας να εντοπίζονται από την «λευκή λίστα» ως ασφαλή.

2.2 - Στατική Ανάλυση ή Στοχευμένη Αναζήτηση.

Η στατική ανάλυση ή στοχευμένη αναζήτηση, αποτελεί την βασική μορφή αναζήτησης κακόβουλου λογισμικού. Βασίζεται στην λογική ότι υπάρχουν κάποιες συναρτήσεις οι οποίες, υπό φυσιολογικές συνθήκες, δεν απαιτούνται για την εύρυθμη λειτουργία ενός ιστοτόπου. Για παράδειγμα αν εντοπιστεί μια συνάρτηση που δίνει την δυνατότητα στους απλούς χρήστες να εκτελούν εντολές του λειτουργικού συστήματος, είναι ένα δείγμα ότι στον ιστότοπο έχει εγκατασταθεί κακόβουλο λογισμικό [6][7]. Αυτές οι συναρτήσεις είναι διαφορετικές αναλόγως της γλώσσας και του λειτουργικού συστήματος που είναι εγκατεστημένες στον web server και γι’ αυτό ο διαχειριστής θα πρέπει να γνωρίζει ποιες συναρτήσεις θεωρούνται ύποπτες και να πραγματοποιεί συχνούς ελέγχους για τον εντοπισμό τους. Εκτός όμως από αυτές τις συναρτήσεις μπορεί να αναζητούνται και συγκεκριμένα μοτίβα ή κείμενα όπως πχ το μανιφέστο του εισβολέα, το ψευδώνυμο του κλπ τα οποία Αποτελούν την πρωταρχική λύση για τον εντοπισμό κακόβουλου λογισμικού. Κύρια μειονεκτήματα τους όμως, λόγω της αποκλειστικής τους στόχευσης είναι η αδυναμία εντοπισμού νέων ή παλαιοτέρων απειλών καθώς συχνά αυτές τροποποιούνται μέσω της διαδικασίας του πολυμορφισμού, οδηγώντας έτσι σε εσφαλμένα αποτελέσματα.

Με βάση αυτές τις αρχές σχεδιάστηκε ο μηχανισμός ο οποίος αφού ανιχνεύσει και εντοπίσει όλα τα αρχεία σε έναν ιστότοπο, στην συνέχεια εκτελεί ενδελεχή έλεγχο για τον εντοπισμό αυτών των λέξεων. Αν και υπάρχει πρόβλεψη για επέκταση του μηχανισμού και σε άλλες γλώσσες, στην συγκεκριμένη φάση, αφού έγινε η σχετική έρευνα, καταγράφηκαν όλες οι συναρτήσεις οι οποίες είναι δυνάμει κακόβουλες, καθώς και κάθε άλλο χαρακτηριστικό το οποίο αποτελεί ένδειξη ότι είναι ύποπτο για την γλώσσα PHP. Στην συνέχεια ο μηχανισμός είναι σε θέση, αφού εντοπίσει τα εν λόγω χαρακτηριστικά, να ενημερώσει τον χρήστη για το όνομα του αρχείου και την γραμμή στην οποία εντοπίστηκε για περαιτέρω ανάλυση όπως

βλέπουμε στην Εικόνα 7. Το συγκεκριμένο αρχείο αποτελούσε βασικό αρχείο του συστήματος wordpress οπότε ο μηχανισμός εμφάνισε σφάλμα False Positive, και για αυτό τον λόγο τέτοιου είδους μηχανισμοί πρέπει να αντιμετωπίζονται από έμπειρους χρήστες, καθώς είναι πολύ πιθανή η απομόνωση ζωτικής σημασίας αρχείων από τον Server όταν αυτά θα έχουν εσφαλμένα σημανθεί ως κακόβουλα.



Εικόνα 7 Στιγμιότυπο κατά την αναζήτηση του μηχανισμού στατικής ανάλυσης.

2.3 - Υπολογισμός εντροπίας αρχείων

Η διαδικασία εντοπισμού αρχείων που περιέχουν κακόβουλο λογισμικό, είναι μια διαδικασία συνεχώς μεταβαλλόμενη καθώς ο επιτιθέμενος έχει το πλεονέκτημα του αιφνιδιασμού και ανάπτυξης νέων τρόπων επίθεσης. Αυτό για τον αμυνόμενο σημαίνει ότι είναι δεν δυνατή η εναπόθεση των αντιμέτρων του μόνο στους γνωστούς τρόπους καθώς ο επιτιθέμενος έχει ήδη σκεφτεί τρόπους παράκαμψης τους. Ένα από τα νεότερα χαρακτηριστικά του κακόβουλου λογισμικού είναι αυτό του πολυμορφισμού, της ικανότητας δηλαδή να κρύβει τις πραγματικές του λειτουργίες μέσω της κρυπτογραφίας. Στην Εικόνα 8 βλέπουμε ένα χαρακτηριστικό παράδειγμα της νεότερης έκδοσης του λογισμικού C99 που προαναφερθεί, το οποίο είναι πλήρως «κρυπτογραφημένο» με κωδικοποίηση Base64 και συμπιεσμένο με συμπίεση gzip. Αν και η Base64 δεν είναι μέθοδος κρυπτογράφησης αλλά κωδικοποίησης για μετάδοση της πληροφορίας μεσώ του πρωτοκόλλου HTTP, εν τούτοις χρησιμοποιείται κατά κόρων για την απόκρυψη κακόβουλων συναρτήσεων καθώς η στατική ανάλυση και η ταυτοποίηση υπογραφών ως μηχανισμοί αναζήτησης όπως γίνεται εύκολα αντιληπτό, θα αποτύχουν.

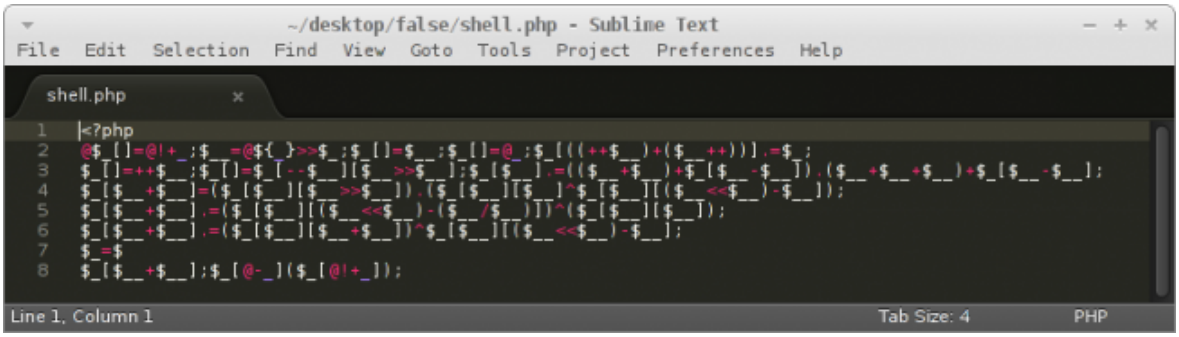
Εικόνα 8 Κρυπτογραφημένος πηγαίος κώδικας κακόβουλου λογισμικού C99

Από την θεωρία της πληροφορίας γνωρίζουμε ότι είναι δυνατή η ποσοτικοποίηση της, σύμφωνα με την κατά Shannon εντροπία. Εν συντομία θα λέγαμε ότι είναι ένα μέτρο της «τυχαιότητας» των δεδομένων σε ένα αρχείο, η κλίμακα του οποίου μετριέται από 1 έως 8 (8 bits σε ένα byte), όπου τα τυπικά αρχεία κειμένου θα έχουν μια χαμηλή τιμή περίπου από 1-4, και τα κρυπτογραφημένα ή συμπιεσμένα αρχεία θα έχουν μία τιμή > 5.

Βασιζόμενοι σε αυτή την αρχή μπορούμε να υπολογίζουμε στα υπό εξέταση αρχεία την εντροπία τους και εφόσον αυτή είναι μεγαλύτερη ενός φυσιολογικού ορίου, ο μηχανισμός αυτός θα ενημερώνει τον χρήστη. Το εύρος της εντροπίας είναι από 1-8 bits , όσο δηλαδή και το πλήθος σε ένα τυπικό αρχείο php με κωδικοποίηση ASCII. Εδώ πρέπει να τονιστεί ότι οι τιμές αφορούν μόνο τα αρχεία με την εν λόγω κωδικοποίηση, τα οποία αποτελούν και την πλειοψηφία των κακόβουλων αρχείων, για λόγους τυποποίησης. Σε αντίθετη περίπτωση, ο μηχανισμός θα εμφανίζει λανθασμένα αποτελέσματα οπότε και δεν θα προτείνεται η χρήση του συγκεκριμένου μηχανισμού. Άλλωστε αυτός ο μηχανισμός λειτουργεί επικουρικά καθώς πρέπει να πληρούνται αρκετές ακόμα προϋποθέσεις έτσι ώστε το αρχείο να είναι πράγματι κακόβουλο. Κατά την διάρκεια των δοκιμών παρατηρήθηκε ότι όταν η τιμή της κατά Shannon εντροπίας ήταν μεγαλύτερη του 5,5, τότε τα αρχεία περιείχαν μεγαλύτερο ποσοστό κρυπτογραφημένου περιεχομένου σε σχέση με τα υπόλοιπα, οπότε και τέθηκε αυτό το όριο για τον εντοπισμό τους.

2.4 - Στατιστική ανάλυση αρχείων.

Όπως έχει προαναφερθεί, ο κώδικας στις γλώσσες σεναρίων είναι ευδιάκριτος και κατανοητός από τον άνθρωπο. Πρόσφατα εντοπίστηκε κακόβουλο λογισμικό το οποίο αν και δεν είναι κρυπτογραφημένο εν τούτης περιέχει μη διακριτές εντολές. Όπως βλέπουμε στην Εικόνα 9, πρόκειται για κώδικα σε γλώσσα PHP ο οποίος εκμεταλλεύεται τις συμβάσεις τις γλώσσας οι οποίες αντικαθιστούν τις συναρτήσεις κάνοντας χρήση ειδικών χαρακτήρων όπως η `$_[]` ή `^` [8]. Με αυτό τον τρόπο καμία από τις προαναφερθείσες μεθόδους δεν πρόκειται να εντοπίσει το εν λόγω backdoor.



Εικόνα 9 Κώδικας Backdoor ο οποίος δεν χρησιμοποιεί συμβατικές συναρτήσεις (blog.sucuri.net)

Η μέθοδος που υλοποιήθηκε είναι η στατιστική ανάλυση των ASCII χαρακτήρων. Αφού γίνει ανάκτηση του κώδικα από το αρχείο, αρχικά υπολογίζεται το πλήθος των αλφαριθμητικών / ειδικών χαρακτήρων και στην συνέχεια η αναλογία μεταξύ τους. Αν η αναλογία των αλφαριθμητικών χαρακτήρων είναι μεγαλύτερη, τότε το αρχείο περνάει επιτυχώς τον έλεγχο, σε αντίθετη περίπτωση ενεργοποιείται ο μηχανισμός και ενημερώνεται ο χρήστης. Ο συλλογισμός βασίζεται στην θεωρία ότι ένα ασφαλές αρχείο περιέχει ευκρινή και επαρκώς σχολιασμένο κώδικα, καθώς και οι συναρτήσεις / μεταβλητές θα είναι ευδιάκριτες για λόγους απλότητας, με αποτέλεσμα το πλήθος των ειδικών χαρακτήρων να είναι μικρότερο από το πλήθος

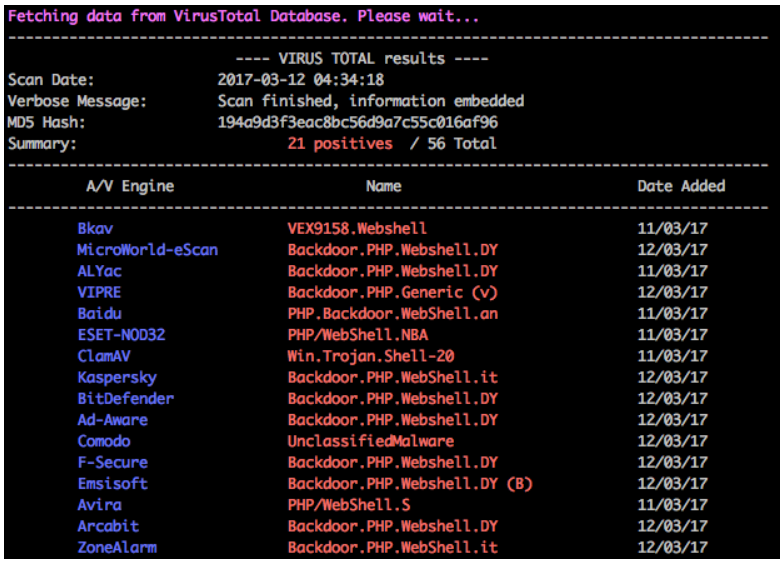
των απλών χαρακτήρων. Όπως είναι εύκολα αντιληπτό και στην Εικόνα 9, το πλήθος των ειδικών χαρακτήρων είναι μεγαλύτερο από το πλήθος των γραμμάτων (υπάρχουν μόνο 3 αλφαριθμητικοί χαρακτήρες), οπότε αυτόματα ο μηχανισμός θα ενεργοποιηθεί και θα ενημερώσει τον χρήστη για το ποσοστό των μη αλφαριθμητικών χαρακτήρων.

2.5 - Αλληλεπίδραση με τον Ιστότοπο VirusTotal.

Ο ιστότοπος VirusTotal αποτελεί μέρος του δικτύου της Google από το 2012 και ο κύριος σκοπός του είναι η αλληλεπίδραση με 56 μηχανισμούς εντοπισμού κακόβουλου λογισμικού, με τον τελικό χρήστη [9]. Ο χρήστης, μέσω ενός φιλικού περιβάλλοντος, αποστέλλει μέσω μιας ειδικής πλατφόρμας ένα ύποπτο αρχείο ή την τιμή hash του και ο ιστότοπος, αφού εκτελέσει τις απαιτούμενες ερωτήσεις στους μηχανισμούς του, παρουσιάζει τα αποτελέσματα και αν το ζητούμενο αρχείο είναι κακόβουλο ή όχι, υποδεικνύοντας παράλληλα ποιες μηχανές το ταυτοποίησαν. Η υπηρεσία αυτή παρέχεται δωρεάν και μέσω διαφόρων διεπαφών που διαθέτει, μπορεί να αξιοποιηθεί από πολλές γλώσσες προγραμματισμού. Σε αυτήν την περίπτωση όμως ο χρήστης υπόκειται στον περιορισμό των 4 ερωτήσεων/λεπτό.

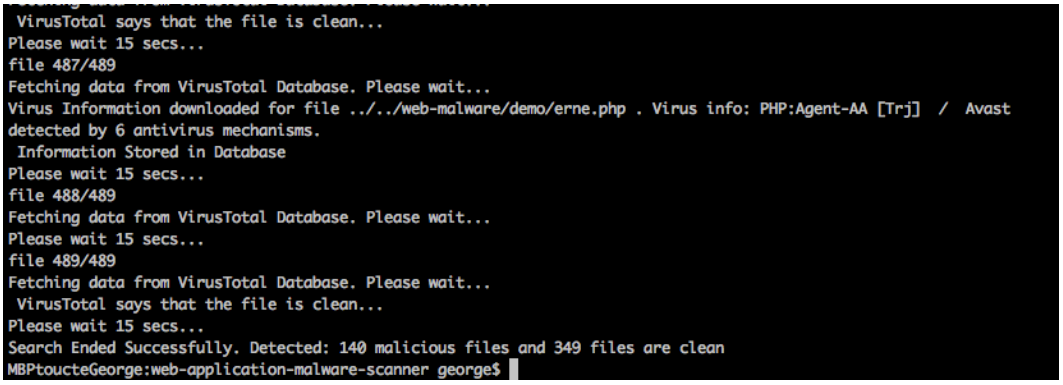
Εικόνα 10 Η διεπαφή του ιστοτόπου VirusTotal.com με τον χρήστη

Κατά την σχεδίαση του μηχανισμού σχεδιάστηκε επίσης διεπαφή μεταξύ του ιστότοπου και της εφαρμογής. Όπως βλέπουμε και στην εικόνα 10, ο μηχανισμός αφού εντόπισε ύποπτο αρχείο υπολόγισε την τιμή hash σε MD5 και απέστειλε το αποτέλεσμα στο VirusTotal. Μόλις παραληφθεί η απάντηση από τον ιστότοπο και μετά κατάλληλης επεξεργασίας, προβάλλεται στον χρήστη για να τον ενημερώσει για τα αποτελέσματα της αναζήτησης. Όπως βλέπουμε, το υπό διερεύνηση αρχείο αναγνωρίστηκε ως κακόβουλο και προβάλλεται η μηχανή anti-malware, η ημερομηνία εντοπισμού του καθώς και την κωδική ονομασία του. Αξίζει να σημειωθεί το συγκεκριμένο αρχείο είναι το **CSH,** ένα σχετικά παλιό κακόβουλο πρόγραμμα με πρώτη αναφορά τον Φεβρουάριο του 2013 [10] και όπως παρατηρούμε, σε λιγότερο από τους μισούς μηχανισμούς που υποστηρίζονται από τον εν λόγω ιστότοπο, σημάνθηκε ως κακόβουλο.



Εικόνα 11 Προβολή αποτελεσμάτων μέσω της διεπαφής με τον ιστότοπο VirusTotal

Κατά την διάρκεια της περιόδου αξιολόγησης διαπιστώθηκε, οι μηχανισμοί του VirusTotal δεν είναι σίγουρο ότι θα ταυτοποιήσουν με επιτυχία ένα ύποπτο αρχείο. Στην πραγματικότητα, αν και όπως έχει αναφερθεί, υπάρχουν 56 διαφορετικοί μηχανισμοί για τον εντοπισμό κακόβουλου λογισμικού, πιθανότατα δεν είναι σε θέση να ανιχνεύσουν κακόβουλο λογισμικό για ιστοσελίδες καθώς παρουσιάζουν πολύ μικρά ποσοστά επιτυχίας. Έτσι για να ελέγξουμε την αποδοτικότητα των μηχανισμών του VirusTotal, δημιουργήθηκε ένα μικροπρόγραμμα το οποίο έστελνε κάθε 1 λεπτό 4 hash στο Virustotal τα hash των κακόβουλων αρχείων που συλλέχθηκαν για τις ανάγκες της παρούσης διατριβής. Είναι χαρακτηριστικό ότι από το σύνολο των 489 επιβεβαιωμένων κακόβουλων αρχείων που συλλέχθηκαν, το VirusTotal ταυτοποίησε μόλις τα 140 ή το ~29% των υφιστάμενων αρχείων και μάλιστα αρκετά από αυτά εντοπίστηκαν μόνο από έναν έως έξι μηχανισμούς (Εικόνα 12). Για τον λόγο αυτό τα αποτελέσματα των ελέγχων, ειδικά δε τα αρνητικά αποτελέσματα, θα πρέπει να λαμβάνονται υπόψη επικουρικά και όχι να θεωρούνται δεδομένα καθώς όπως αποδείχτηκε υπάρχει μεγάλο ποσοστό εμφάνισης false positive ένδειξης.



Εικόνα 12 Αποτελέσματα του μικροπρογράμματος απόδοσης ιστοτόπου VirusTotal

Η σελίδα αφέθηκε σκοπίμως κενή

Κεφάλαιο 3 - Μηχανισμός Μηχανικής Μάθησης

3.1 – Εισαγωγή στην μηχανική μάθηση

Την τελευταία δεκαετία χρησιμοποιείται συχνά η τεχνητή νοημοσύνη και ειδικότερα η Μηχανική Μάθηση (Machine Learning) η οποία έχει αποκτήσει μεγάλο εύρος εφαρμογής. Η μηχανική μάθηση εκφράζεται ως η δυνατότητα ενός πληροφοριακού συστήματος, το οποίο βασιζόμενο σε ένα σύνολο δεδομένων, μπορεί να προβλέψει όλες τις πιθανές καταστάσεις ενός γεγονότος, όπως για παράδειγμα τον καιρό. Όπως είναι εύκολα αντιληπτό, η ακρίβεια των αποτελεσμάτων αυτών, βασίζεται στην ποιότητα και ποσότητα των δεδομένων, έτσι ώστε να έχει όσο το δυνατόν ακριβέστερα αποτελέσματα.

Η μηχανική μάθηση διακρίνεται στην ***μάθηση με επίβλεψη*** (supervised learning), στην ***μάθηση χωρίς επίβλεψη*** (unsupervised learning) και στην ***ενισχυτική μάθηση*** (reinforcement learning). Η μάθηση με επίβλεψη χρησιμοποιείται σε προβλήματα ταξινόμησης, πρόγνωσης και διερμηνείας όταν από τα δεδομένα που εισάγονται κατά την διάρκεια της εκπαίδευσης (ορίζονται ως σύνολο εκπαίδευσης) προκύπτουν γνωστές επιθυμητές έξοδοι με σκοπό την επέκταση τους για εισόδους με άγνωστη έξοδο. Η μάθηση χωρίς επίβλεψη χρησιμοποιείται σε προβλήματα ανάλυσης συσχετισμών και ομαδοποίησης, όταν το σύνολο εκπαίδευσης δεν μπορούμε να το κατατάξουμε σε κάποια ομάδα και δεν γνωρίζουμε τις επιθυμητές εξόδους. Τέλος, η ενισχυτική μάθηση χρησιμοποιείται κυρίως σε προβλήματα σχεδιασμού στρατηγικών ενεργειών μετά από αλληλεπίδραση με το περιβάλλον, κυρίως σε ρομποτικές εφαρμογές.

3.2 – Περιγραφή του ταξινομητή Naïve Bayes

Για την υλοποίηση του μηχανισμού εντοπισμού κακόβουλου λογισμικού, θα χρησιμοποιήσουμε την μάθηση με επίβλεψη. Στην μάθηση με επίβλεψη το εκπαιδευμένο μοντέλο που προκύπτει από την εφαρμογή ενός αλγορίθμου ταξινόμησης σε ένα σύνολο διανυσμάτων χαρακτηριστικών καλείται ***ταξινομητής*** (classifier). Η κατηγοριοποίηση βασίζεται στην εξέταση των χαρακτηριστικών ενός νέου αντικειμένου το οποίο κατατάσσεται (ταξινομείται) με βάση τα χαρακτηριστικά του. Ο ταξινομητής που θα υλοποιηθεί, βασίζεται στην μάθηση κατά Bayes και λέγεται ***Αφελής Ταξινομητής Bayes*** (Naïve Bayes Classifier ή NB) [11][12]. Ο ταξινομητής Naïve Bayes χρησιμοποιείται όταν έχουμε κάποιες καταστάσεις και θέλουμε να εξάγουμε την πιθανότητα να συμβεί ένα γεγονός κάνοντας χρήση των στατιστικών μεθόδων. Κλασικό παράδειγμα ταξινόμησης κατά Naïve Bayes αποτελεί το φίλτρο ανεπιθύμητης ηλεκτρονικής αλληλογραφίας (spam filter), από όπου το πρόγραμμα αλληλογραφίας διαχωρίζει τα email σε spam ή όχι ανάλογα με τις λέξεις που περιέχονται σε αυτό. Πρακτικά όταν ένα email εισέρχεται στην ηλεκτρονική μας θυρίδα τότε ο μηχανισμός αφού αντλήσει όλες τις λέξεις από τις οποίες απαρτίζεται, στην συνέχεια υπολογίζει την πιθανότητα η κάθε λέξη, να συμπεριλαμβάνεται σε ανεπιθύμητο email ή όχι. Στην συνέχεια υπολογίζεται το γινόμενο αυτών των αποτελεσμάτων. Αν το γινόμενο είναι > 1 τότε το email είναι ανεπιθύμητο, ενώ σε

διαφορετική περίπτωση είναι ασφαλές. Ο μαθηματικός τύπος που εκφράζει τον Naïve Bayes Classifier είναι [11]:

$$P(c_i|w) = \frac{P(W|c_i)P(c_i)}{P(w)}$$

όπου:

$c_i = \{spam, ham\}$

$W = \{w_1, w_2, w_3, \dots, w_n\} \rightarrow το\ σύνολο\ των\ λέξεων\ του\ email$

$P(c_i|w)$ η πιθανότητα το email να είναι c_i δοθέντος της λέξης w

$P(w|c_i)$ η πιθανότητα να παραχθεί η λέξη w δοθείσας της κατηγορίας c_i

$P(c_i)$ η πιθανότητα το email να είναι c_i

$P(w)$ η πιθανότητα να περιέχεται η λέξη w στο email

Με τον παραπάνω τύπο όμως ισχύει για την ταξινόμηση ενός email που περιέχει μία μόνο λέξη. Επειδή όμως ένα mail όπως είναι κατανοητό απαρτίζεται από περισσότερες λέξεις, ο ταξινομητής Naïve Bayes ορίζεται ως

$$P(c_i|W) = \prod_{k=1} P(w_k c_i) \times P(c_i)$$

Τέλος για κάθε κλάση έχουμε:

$$P(c_i) = \frac{P(c_i|W)}{P(c_i|W) + P(c_j|W)}$$

όπου $c_i = \{spam|ham\}$ και $c_j = \{ham|spam\}$

Για να κατανοήσουμε την λειτουργία του ταξινομητή θα παρουσιάσουμε ένα παράδειγμα με τον τρόπο λειτουργίας του φίλτρου spam. Έστω τα ακόλουθα email τα οποία έχουν ταξινομηθεί από τον χρήστη, αναλόγως του περιεχομένου τους:

A/A	Περιεχόμενο Email	Κατάταξη
E1	Send us your password	Spam
E2	Send us your review	Ham
E3	Review your password	Ham
E4	Review us	Spam
E5	Send your password	Spam
E6	Send us your account	Spam

Στην συνέχεια για κάθε λέξη που εμπεριέχεται στα μηνύματα αυτά κατατάσσονται σύμφωνα με τον χαρακτηρισμό του email. Για παράδειγμα η λέξη

password περιέχεται σε 2 από τα 4 emails ως spam και σε 1 από τα 2 emails ως ham (μη spam).

Με βάση τα παραπάνω υπολογίζουμε το $P(w|c_i)$ για το σύνολο των λέξεων ως ακολούθως:

Πίνακας συχνότητας λέξεων			
spam	Ham	Λέξη	P(w)
2/4	1/2	Password	3/6=0.5
1/4	2/2	Review	3/6=0.5
3/4	1/2	Send	4/6=0.66
3/4	1/2	Us	4/6=0.66
3/4	2/2	Your	5/6=0.83
1/4	0/2	account	1/6=0.166
$P(spam) = \frac{13}{20} = 0{,}65$	$P(ham) = \frac{7}{20} = 0{,}35$		

$$P(password|spam) = \frac{2}{4} = 0{,}5$$

$$P(password|ham) = \frac{1}{2} = 0{,}5$$

$$P(review|spam) = \frac{1}{4} = 0{,}25$$

$$P(review|ham) = \frac{2}{2} = 1$$

$$P(send|spam) = \frac{3}{4} = 0{,}75$$

$$P(send|ham) = \frac{1}{2} = 0{,}5$$

$$P(us|spam) = \frac{3}{4} = 0{,}75$$

$$P(us|ham) = \frac{1}{2} = 0{,}5$$

$$P(your|spam) = \frac{3}{4} = 0{,}75$$

$$P(your|ham) = \frac{2}{2} = 1$$

$$P(account|spam) = \frac{1}{4} = 0{,}25$$

$$P(account|ham) = \frac{0}{2} = 0$$

Αφού εκπαιδεύσαμε το μοντέλο μας με τα ανωτέρω δεδομένα, έστω ότι λαμβάνουμε το μήνυμα «send your review».

Με βάση τα παραπάνω σύμφωνα με τον τύπο του ταξινομητή Naïve Bayes έχουμε ότι:

$$P(c|W) = \prod_{k=1} P(w_k|c) \times P(c) \Longrightarrow$$

$$P(spam|send\ your\ review) = P(send|spam) \times P(your|spam) \times P(review|spam) \times P(spam) = 0{,}75 \times 0{,}59 \times 0{,}32 \times 0{,}65 = 0{,}092$$

$$P(ham|send\ your\ review) = P(send|ham) \times P(your|ham) \times P(review|ham) \times P(ham) = 0{,}5 \times 1 \times 1 \times 0{,}35 = 0{,}175$$

Τέλος για να βρούμε το ποσοστό spam και ham εκτελούμε ως ακολούθως:

$$P(spam) = \frac{0{,}092}{0{,}092+0{,}175} \approx 0{,}345 \text{ και}$$

$$P(ham) = \frac{0{,}175}{0{,}092 + 0{,}175} \approx 0{,}655$$

άρα το μήνυμα δεν είναι spam με πιθανότητα 65%.

3.3 - Εφαρμογή του ταξινομητή Naïve bayes ως εργαλείο εντοπισμού κακόβουλου κώδικα.

Αρχικά κρίνεται σκόπιμο να διευκρινιστεί ότι ο εν λόγω μηχανισμός έχει πρακτική εφαρμογή μόνο σε script languages που χρησιμοποιούνται από τους web servers. Καθώς λόγω του ότι τα προς διερεύνηση αρχεία δεν είναι υπό την μορφή bytecode αλλά σε γλώσσα που προσομοιάζεται στην ανθρώπινη γλώσσα, βασίστηκε η ιδέα της εφαρμογής του εν λόγω ταξινομητή.

Ο συλλογισμός που ακολουθήθηκε είναι ο εξής. Έστω υπάρχει ένα αρχείο σε γλώσσα PHP, στο αρχείο αυτό επίσης υπάρχουν συναρτήσεις f οι οποίες εκτελούν κάποια εργασία στον κώδικα. Το αρχείο αυτό περιέχει κώδικα ο οποίος καλεί συναρτήσεις f έτσι ώστε να εκτελεστεί το πρόγραμμα. Οι συναρτήσεις αυτές μπορεί είτε να εμπεριέχονται στο βασικό set συναρτήσεων της γλώσσας, είτε εμπεριέχονται σε πρόσθετο πακέτο που έχει εγκατασταθεί στον server, είτε τέλος να έχουν γραφτεί από τον μηχανικό του προγράμματος για να εκτελέσει μια συγκεκριμένη διεργασία.

Γνωρίζουμε δε, ότι κάποιες από αυτές τις συναρτήσεις, ιδίως αυτές που αλληλεπιδρούν με το λειτουργικό σύστημα, χρησιμοποιούνται κυρίως από κακόβουλους χρήστες, έτσι ώστε να καταφέρουν να συνδεθούν με το υπολογιστικό σύστημα στο οποίο και είναι εγκατεστημένο το πρόγραμμα. Όμως οι ίδιες συναρτήσεις μπορούν να χρησιμοποιηθούν και από τους νόμιμους χρήστες έτσι ώστε να εκτελέσουν την ίδια ακριβώς εργασία για διαφορετικούς λόγους. Σε αυτή την περίπτωση τα προγράμματα που χρησιμοποιούνται για τον εντοπισμό κακόβουλου κώδικα, αποτυγχάνουν να εντοπίσουν με ακρίβεια αν το υπό εξέταση αρχείο είναι επισφαλές ή όχι, καθώς θεωρούν ότι εφόσον περιέχει τουλάχιστον μία από τις προκαθορισμένες «ύποπτες» συναρτήσεις, το αρχείο αποτελεί δυνητικά απειλή για το σύστημα. Η λογική δεν είναι εσφαλμένη αλλά όμως όταν υπάρχουν πολλά αρχεία στον web server, καθώς οι προς αναζήτηση συναρτήσεις εκτελούν και νόμιμες εργασίες, τότε η ανωτέρω μορφή αναζήτησης θα εντοπίζει όλα τα αρχεία, με μεγάλη πιθανότητα παρουσίασης σφάλματος, καθώς δεν είναι δυνατή με αυτή την μεθοδολογία να εντοπιστεί η διάκριση μεταξύ ασφαλούς και κακόβουλου λογισμικού.

Η ανωτέρω διάκριση μπορεί να επιτευχθεί μέσω της χρήσης της τεχνικής spam filter της προηγούμενης ενότητας. Όπως γίνεται στο φίλτρο για spam emails, που εξάγουμε όλες τις λέξεις που περιέχονται σε ένα mail και ανάλογα με το περιεχόμενο τους ταξινομούμε το mail σε spam ή όχι, έτσι και εδώ εξάγουμε όλες τις συναρτήσεις που περιέχονται σε ένα αρχείο php και γίνεται ταξινόμηση αν το αρχείο είναι κακόβουλο ή όχι. Με βάση τα παραπάνω έχουμε:

$F = \{f_1, f_2, \dots f_n\}$ όπου f_i μια συνάρτηση που περιέχεται σε ένα αρχείο τύπου php,

$y = \{malicious, harmful\}$,

Αν αντικαταστήσουμε όπου $w \to f, c \to y$ που είδαμε στο παράδειγμα με το spam filter τότε ο ταξινομητής Naïve Bayes

$$P(malicious|F) = \prod_{k=1} P(f_k|malicious) \times P(malicious)$$

και

$$P(harmful|F) = \prod_{k=1} P(f_k|harmful) \times P(harmful)$$

οπότε αν $P(malicious) > P(harmful)$ τότε το αρχείο είναι κακόβουλο κατά $P(malicious)$ % διαφορετικά είναι ασφαλές κατά $P(harmful)$ %.

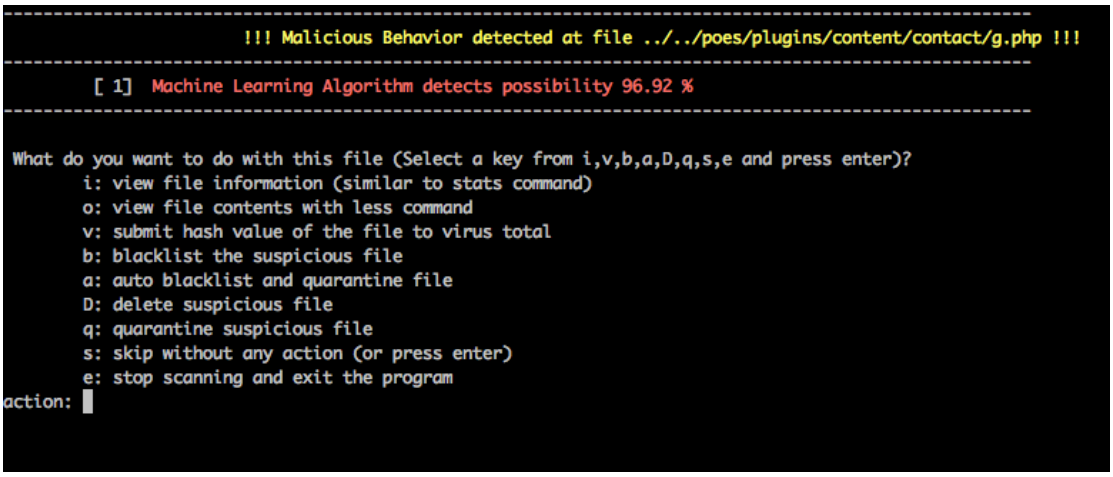
3.4 - Υλοποίηση Μηχανισμού Εντοπισμού Κακόβουλου Λογισμικού με Αλγόριθμο Μηχανικής Μάθησης.

Η γλώσσα που σχεδιάστηκε ο μηχανισμός επιλέχθηκε για έναν κύριο λόγο. Η Python διαθέτει πανίσχυρες βιβλιοθήκες και για αυτό τον λόγο χρησιμοποιείται σε διάφορα επιστημονικά και επαγγελματικά έργα, ιδιαιτέρως δε σε εφαρμογές μηχανικής μάθησης και τεχνητής νοημοσύνης. Οι βιβλιοθήκες που εντοπίστηκαν όμως δεν μπορούν να αξιοποιηθούν, καθώς το πρόγραμμα που τις χρησιμοποιεί πρέπει να εκπαιδεύεται σε κάθε χρήση. Για τον λόγο αυτό έγινε αναζήτηση περισσότερο ευέλικτων μηχανισμών, οι οποίοι θα μπορούσαν να διατηρούν τα αποτελέσματα των μετρήσεων τους με την δυνατότητα αυτά να ανακαλούνται όποτε απαιτούνταν.

Ο μηχανισμός που αναπτύχθηκε χρησιμοποιεί την βιβλιοθήκη του Naïve Bayes Classification με χρήση τοπικής βάσης δεδομένων sqlite [12]. Πρόκειται για μια βιβλιοθήκη η οποία είναι ανοιχτού κώδικα και σε αντίθεση με τις υπόλοιπες, αποθηκεύει τις τιμές ταξινόμησης σε μια βάση δεδομένων, με αποτέλεσμα αυτές να είναι άμεσα προσπελάσιμες κάθε φορά που το πρόγραμμα εκτελείται χωρίς να απαιτείται εκ νέου εκπαίδευση. Επίσης, σε κάθε νέα εκπαίδευση αυτή λαμβάνει υπόψη της τις προηγούμενες τιμές, με αποτέλεσμα ο αλγόριθμος να εκπαιδεύεται σε κάθε νέα εφαρμογή του.

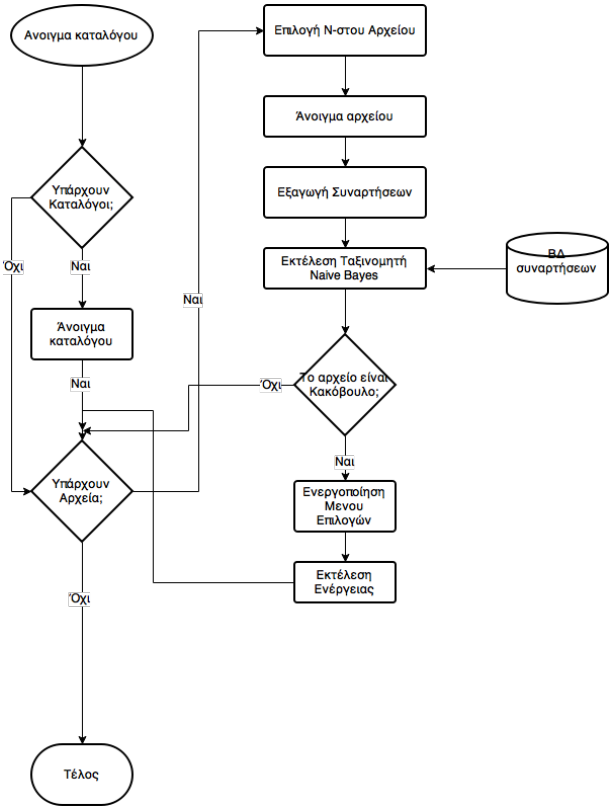
Ο τρόπος λειτουργίας του μηχανισμού περιγράφεται ως εξής: Σε κάθε αρχείο που προσπελάζεται, εντοπίζονται οι συναρτήσεις του, στην συνέχεια αυτές απομονώνονται και εισέρχονται στον μηχανισμό μηχανικής μάθησης. Εκεί για κάθε συνάρτηση υπολογίζεται μια τιμή η οποία υποδηλώνει το ποσοστό $P(malicious)$ η συνάρτηση να περιέχεται σε κακόβουλο καθώς και το ποσοστό να περιέχεται σε ασφαλή κώδικα $P(harmful)$. Στην συνέχεια οι τιμές αυτές πολλαπλασιάζονται σύμφωνα με τους τύπους της προηγούμενης ενότητας. Αν το αποτέλεσμα της πιθανότητας $P(malicious) > P(harmful)$ τότε το αρχείο είναι κακόβουλο κατά $P(malicious)$ % διαφορετικά είναι ασφαλές κατά $P(harmful)$ %.

Στην εικόνα 13 που ακολουθεί βλέπουμε οπτικοποιημένο το αποτέλεσμα του αλγορίθμου που αναλύσαμε στην προηγούμενη παράγραφο. Στο εν λόγω στιγμιότυπο ο μηχανισμός έχει εντοπίσει ένα ύποπτο αρχείο με ονομασία g.php για το οποίο μάλιστα παρουσιάζει μεγάλο ποσοστό πιθανότητας να είναι κακόβουλο. Στην συνέχεια εμφανίζεται ένα μενού επιλογών έτσι ώστε ο χρήστης να εκτελέσει ένα πλήθος ενεργειών για το συγκεκριμένο αρχείο. Για παράδειγμα αν ο χρήστης επιλέξει το πλήκτρο «a» έχει επιβεβαιώσει ότι το αρχείο είναι κακόβουλο οπότε ο μηχανισμός απομονώνει το αρχείο μετονομάζοντας και μεταφέροντας το σε ασφαλή τοποθεσία, στην συνέχεια δημιουργεί την μοναδική υπογραφή του και την αποθηκεύει στην «μαύρη λίστα» και τέλος εκπαιδεύεται ο μηχανισμός έτσι ώστε την επόμενη φορά που θα εντοπίσει ένα κακόβουλο αρχείο, να εμφανίσει περισσότερες πιθανότητες επιτυχίας.



Εικόνα 13 Στιγμιότυπο από την λειτουργία του μηχανισμού μηχανικής μάθησης.

Τέλος, στο ανωτέρω παράδειγμα, αξίζει να αναφερθεί ότι το αρχείο g.php ήταν πράγματι κακόβουλο και μάλιστα επρόκειτο για αρχείο στο οποίο ο μηχανισμός δεν είχε εκπαιδευτεί, με αποτέλεσμα να έχουμε επιτύχει να δημιουργήσουμε έναν μηχανισμό ο οποίος είναι σε θέση να εντοπίζει άγνωστο κακόβουλο λογισμικό. Στην εικόνα 12 που ακολουθεί, παρατηρούμε το διάγραμμα ροής του μηχανισμού μηχανικής μάθησης.



Εικόνα 14 Διάγραμμα ροής μηχανισμού μηχανικής μάθησης

Η σελίδα αφέθηκε σκοπίμως κενή

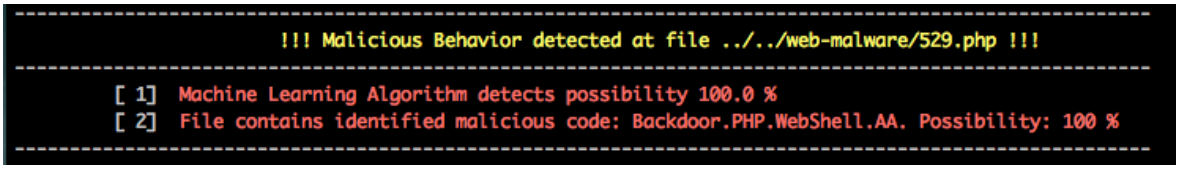
Κεφάλαιο 4 - Εκπαίδευση του Μηχανισμού

4.1 - Οριοθέτηση Δειγματικού Χώρου

Μετά την ολοκλήρωση του σχεδιασμού και της υλοποίησης περνάμε στην φάση της εκπαίδευσης του μηχανισμού για τον εντοπισμό κακόβουλου λογισμικού. Η φάση αυτή ολοκληρώθηκε σε τρία (3) στάδια. Στο πρώτο στάδιο έγινε αναζήτηση στο διαδίκτυο και συγκεντρώθηκαν συνολικά 483 αρχεία που περιέχουν κακόβουλο λογισμικό, τα οποία στοχεύουν αποκλειστικά σε εξυπηρετητές που εκτελούν την γλώσσα προγραμματισμού PHP. Τα αρχεία αυτά δεν φέρουν πνευματικά δικαιώματα και έχουν συγκεντρωθεί από διάφορες κοινότητες ανοιχτού λογισμικού[14][15]. Επίσης δημιουργήθηκαν 6 αρχεία από το εργαλείο Meterpreter για να επιτύχουμε σύνδεση μέσω της ανάστροφης διαδικασίας tcp (reverse tcp) σε servers με υποστήριξη PHP scripts. Έτσι ο δειγματικός χώρος που περιέχει κακόβουλο λογισμικό αποτελείται από 489 αρχεία. Ο χώρος αυτός διασπάστηκε σε δύο υποσύνολα, με το υποσύνολο Α να αποτελείται από 469 αρχεία και το υποσύνολο Β με τα υπόλοιπα 20. Αυτό έγινε διότι σε επόμενη φάση θέλουμε η μηχανή τεχνητής νοημοσύνης να εντοπίσει άγνωστα σε αυτήν αρχεία, οπότε τα αρχεία που βρίσκονται στο υποσύνολο Β θα χρησιμοποιηθούν για να ελεγχθεί η απόδοση του μηχανισμού.

4.2 – Δημιουργία Υπογραφών Μηχανισμού

Στο δεύτερο στάδιο έγινε σάρωση των αρχείων του υποσυνόλου Α στα οποία υπολογίστηκε για κάθε ένα η τιμή hash τους σε μορφή SSDeep. Αυτές οι τιμές αποτελούν τις υπογραφές των κακόβουλων αυτών αρχείων οι οποίες και αποθηκεύονται στην βάση δεδομένων του μηχανισμού για μελλοντική χρήση. Για να γίνει πιο ρεαλιστική η αναγνώριση αυτών των αρχείων, η τιμή hash αποστέλλεται στην δημόσια υπηρεσία VirusTotal, έτσι ώστε τα ήδη ταυτοποιημένα αρχεία να λάβουν την ονοματοδοσία που έχει χορηγηθεί σε αυτά από την υπηρεσία, ενώ αυτά που δεν έχουν ταυτοποιηθεί να αποθηκεύονται με μια γενικευμένη ονομασία. Με αυτόν τον τρόπο, όταν εντοπίζεται κακόβουλο λογισμικό, εμφανίζεται στον χρήστη το όνομα του κακόβουλου λογισμικού όπως βλέπουμε στην Εικόνα 15, έτσι ώστε αυτός γνωρίζοντας πλέον την απειλή να εκτελέσει τα αντίστοιχα αντίμετρα.



Εικόνα 15 Μήνυμα ενημέρωσης του χρήστη σχετικά με το εντοπισθέν κακόβουλο λογισμικό

4.3 – Εκπαίδευση Μηχανισμού Μηχανικής Μάθησης

Στο τελικό στάδιο γίνεται εκπαίδευση του μηχανισμού μηχανικής μάθησης. Η χρήση του αλγορίθμου που χρησιμοποιήσαμε απαιτεί να καθοριστούν οι κλάσεις ταξινόμησης, δηλαδή ο κακόβουλος και ο μη κακόβουλος κώδικας. Οπότε πρώτα θα εκπαιδεύσουμε την μηχανή με τον κακόβουλο κώδικα κατ' αντίστοιχο τρόπο όπως

στο προηγούμενο στάδιο. Η εφαρμογή τέθηκε σε κατάσταση εκπαίδευσης και ορίστηκαν τα αρχεία που βρίσκονταν στο υποσύνολο Α ως κακόβουλα. Για την εκπαίδευση της μηχανής ΤΝ με το μη κακόβουλο λογισμικό θα πρέπει να βεβαιωθούμε ότι ο προς εκπαίδευση κώδικας δεν είναι μολυσμένος. Οπότε έγινε λήψη των 2 πιο γνωστών προγραμμάτων διαχείρισης περιεχομένου ανοιχτού κώδικα, το Joomla (έκδοση 3.6.4) [16] και το Wordpress (έκδοση 4.7)[17] στην συνέχεια τέθηκε η μηχανή σε κατάσταση εκπαίδευσης οπότε και αναλύθηκε η συμπεριφορά των αρχείων τους ως ασφαλή.

	Κακόβουλο Λογισμικό		Ασφαλές Λογισμικό	
Υποσύνολα	A	B	Joomla	Wordpress
Πλήθος Αρχείων	469	20	2643	1480
Σύνολα	489		7816	

Πίνακας 1 Δειγματικός Χώρος Αξιολόγησης Εφαρμογής

Κεφάλαιο 5 – Αξιολόγηση

5.1 – Μεθοδολογία Αξιολόγησης Μηχανισμού Μηχανικής Μάθησης.

Αφού ολοκληρώθηκε η εκπαίδευση περνάμε στην φάση της αξιολόγησης του μηχανισμού για τον εντοπισμό κακόβουλου λογισμικού. Για την αξιολόγηση του μηχανισμού όπως προαναφέρθηκε, θα χρειαστούμε δύο ευρέως γνωστές εφαρμογές διαχείρισης περιεχομένου ανοιχτού λογισμικού, το WordPress και το Joomla. Θεωρούμε ότι τα εν λόγω λογισμικά επειδή ελήφθησαν μέσω των επίσημων ιστότοπων τους, δεν περιέχουν κακόβουλο λογισμικό. Εδώ θα πρέπει να τονιστεί ότι δεν έγινε εισαγωγή των αρχείων τους στην «λευκή» λίστα του μηχανισμού, καθώς σκοπός είναι να αναδειχτεί η ευαισθησία του σε όσο το δυνατόν πραγματικές καταστάσεις, να εντοπισθεί ο ρυθμός κατάδειξης του κακόβουλου λογισμικού αλλά και την ύπαρξη «λανθασμένων συναγερμών» (false alarms). Στην συνέχεια θα επιλέξουμε από το υποσύνολο Α αλλά και από το υποσύνολο Β που περιγράψαμε στο προηγούμενο κεφαλαίο τα αρχεία με κακόβουλο λογισμικό. Αυτά θα τοποθετηθούν σε διάφορες θέσεις των ανωτέρω ιστοτόπων, ενώ σε μερικά αρχεία θα γίνει τροποποίηση του υπάρχοντος κώδικα και θα εισαχθεί κακόβουλος έτσι ώστε να ξεγελάσουμε τον μηχανισμό ανίχνευσής. Παράλληλα για την αξιολόγηση της εφαρμογής δοκιμάστηκαν τα προγράμματα **Neopi** [18], **Shell Detector** [19] και **ClamAV** [20] έτσι ώστε να γίνει σύγκριση των αποτελεσμάτων που παρήγαγε ο μηχανισμός, με εργαλεία που ήδη χρησιμοποιούνται για τον εντοπισμό κακόβουλου λογισμικού. επίσης, ενεργοποιήθηκαν μόνο οι μηχανισμοί μηχανικής μάθησης και μαύρης λίστας καθώς οι υπόλοιποι χρησιμοποιούνται από τις NeoPi και Shell Detector, ενώ στον ClamAV η αναζήτηση έγινε στην αυτόματη κατάσταση. Εδώ κρίνεται σκόπιμο να επισημανθεί ότι σε όλους τους μηχανισμούς **έγινε αναζήτηση στο σύνολο των αρχείων** του υπό εξέταση ιστότοπου και όχι σε στοχευμένα αρχεία καθώς με διάφορες τεχνικές είναι δυνατό να εκτελεστεί κακόβουλος κώδικας και σε αρχεία φωτογραφιών (jpg), κειμένου (txt) κ.α [29].

Για την αξιολόγηση των ανωτέρω μηχανισμών ακολουθήθηκαν οι διαδικασίες που έχουν καθοριστεί σύμφωνα με τον Anti-Malware Testing Standards Organization [21] προκειμένου να επιτύχουμε όσο το δυνατόν ακριβέστερες μετρήσεις. Αρχικά έγινε αναζήτηση και με τους τέσσερις μηχανισμούς στους υπό δοκιμή ιστότοπους με σκοπό να εντοπιστεί ο ρυθμός False Positive που εμφανίζουν. Εδώ θα πρέπει να ληφθεί υπόψη ότι στους μηχανισμούς anti-malware υπάρχουν 4 καταστάσεις οι οποίες μεταφράζονται ως ακολούθως:

1. **True Positive(TP)**: το αρχείο που εντοπίστηκε ως κακόβουλο, είναι κακόβουλο.
2. **False Positive(FP)**: το αρχείο που εντοπίστηκε ως κακόβουλο, είναι ακίνδυνο. Συχνά αναφέρεται και ως «Λάθος Συναγερμός» (False Alarm).
3. **True Negative(TN)**: το αρχείο που εντοπίστηκε ως ακίνδυνο, είναι ακίνδυνο.
4. **False Negative(FN)**: το αρχείο που εντοπίστηκε ως ακίνδυνο, είναι κακόβουλο.

Αυτές οι καταστάσεις, γνωστές και ως Binary Classification, αναπαριστώνται σε μια δομή, γνωστή ως Confusion Matrix [22] και περιγράφεται ως εξής:

		Πραγματική Κατάσταση	
		Θετική (Positive)	Αρνητική (Negative)
Πρόβλεψη	Αληθές (True)	True Positive (TP)	False Positive (FP)
	Ψευδές (False)	False Negative (FN)	True Negative (TN)

Πίνακας 2 Confusion Matrix των καταστάσεων μηχανισμού anti-malware

Για να μπορέσουμε να υπολογίσουμε την απόδοση με βάση τις παραπάνω καταστάσεις θα χρειαστούμε όρους από την μετρική αποτίμηση της αποτελεσματικότητας . Θα αναφερθούμε σε όρους όπως η **ορθότητα** (Accuracy) η οποία υποδηλώνει το ποσοστό σωστής ταξινόμησης από τον μηχανισμό στο σύνολο των αρχείων, η **ακρίβεια** (Precision) ως το ποσοστό των επιλεγμένων αρχείων είναι πράγματι κακόβουλα, η **ανάκληση** (Recall) ή ποσοστό ρυθμού True Positive ή ευαισθησία (Sensitivity) που είναι το ποσοστό των κακόβουλων αρχείων που εντοπίστηκαν ως προς το σύνολο των κακόβουλων αρχείων που υφίστανται στον υπό εξέταση ιστότοπο και η **ειδικότητα** (Specificity) η οποία είναι το ποσοστό των ασφαλών αρχείων που εντοπίστηκαν ως προς το σύνολο των ασφαλών αρχείων. Τα παραπάνω εκφράζονται με τους ακόλουθους τύπους [22] [23]:

$$P = TP + FP$$

$$N = TN + FN$$

$$True\ Positive\ Rate\ (TPR)\ \text{ή}\ recall = \frac{TP}{(TP + FN)}$$

$$True\ Negative\ Rate\ (TNR)\ \text{ή}\ specificity = \frac{TN}{(TN + FP)}$$

$$False\ Positive\ Rate\ (FPR) = \frac{FP}{(FP + TN)} = 1 - TNR$$

$$False\ Negative\ Rate\ (FNR) = \frac{FN}{(FN + TP)} = 1 - TPR$$

$$precision\ \text{ή}\ Positive\ Predictive\ Value\ (PVV) = \frac{TP}{(TP + FP)}$$

$$accuracy = \frac{TP + TN}{(TP + TN + FP + FN)}$$

$$Negative\ Predicted\ Value\ (NPV) = \frac{TN}{(TN + FN)}$$

$$False\ Discovery\ Rate\ (FDR) = \frac{FP}{(FP + TP)} = 1 - PVV$$

Οι παραπάνω τύποι χρησιμοποιούνται ευρέως για την αξιολόγηση anti-malware λογισμικού, ιατρικών διαγνωστικών και στατιστικών δυαδικών καταστάσεων ως μέτρο σύγκρισης των αποτελεσμάτων τους. Κατά συνέπεια αποτελούν τα ιδανικά εργαλεία για τον έλεγχο της απόδοσης του υπό ανάπτυξη μηχανισμού.

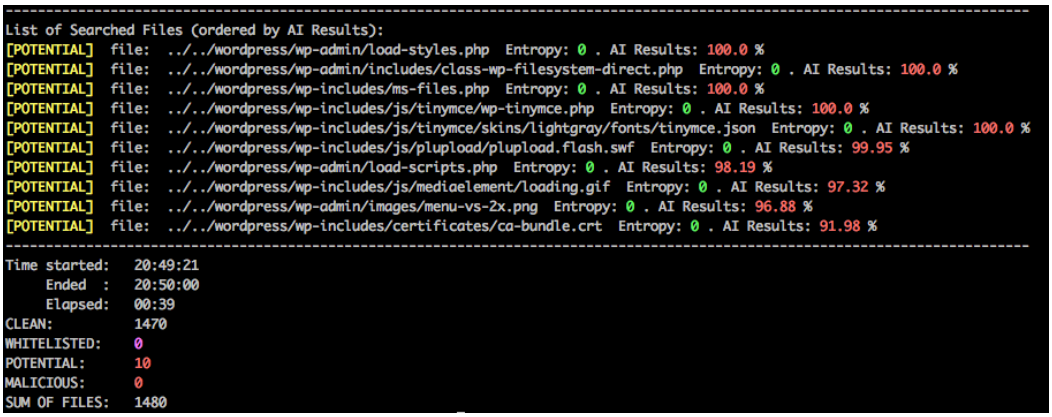
5.2 – 1η αξιολόγηση του μηχανισμού σε αρχεία ιστοτόπου τύπου Wordpress

5.2.1 - Έλεγχος ύπαρξης καταστάσεων τύπου False Positive

Αρχικά έγινε αναζήτηση στον ιστότοπο χωρίς να έχουμε εγκαταστήσει το κακόβουλο λογισμικό για να εντοπίσουμε όλες τις πιθανές καταστάσεις False Positive. Πρώτα έγινε δοκιμή μόνο με τον μηχανισμό μηχανικής μάθησης. Όπως έχει εξηγηθεί σε προηγούμενο κεφάλαιο, ο συγκεκριμένος μηχανισμός αφού αναλύσει τα αρχεία σύμφωνα με την συμπεριφορά τους, τα ταξινομεί αναλόγως του ποσοστού πιθανότητας η συμπεριφορά τους να προσιδιάζει αυτής του κακόβουλου προγράμματος. Όπως βλέπουμε και στην Εικόνα 16 ο μηχανισμός εντόπισε 10 πιθανά αρχεία ως κακόβουλα και μάλιστα με μεγάλο ποσοστό πιθανότητας. Αυτό συμβαίνει διότι τα δείγματα με τα οποία εκπαιδεύτηκε ο μηχανισμός μηχανικής μάθησης δεν είναι αρκετά έτσι ώστε να είναι δυνατή η ακριβέστερη διάκριση μεταξύ των αρχείων με αποτέλεσμα να παρουσιάζουν συμπεριφορά αντίστοιχη με αυτή ενός κακόβουλου κώδικα. Άλλωστε ας μην ξεχνάμε ότι ο εν λόγω μηχανισμός σχεδιάστηκε για να είναι σε θέση να εντοπίζει κυρίως άγνωστα κακόβουλα λογισμικά όπως επίσης και ότι στην παρούσα φάση δεν ενεργοποιήθηκε η «λευκή λίστα». Με βάση τα παραπάνω αποτελέσματα παρατηρούμε ότι ο μηχανισμός μηχανικής μάθησης παρουσιάζει έναν ρυθμό απόδοσης **False Positive** μόλις

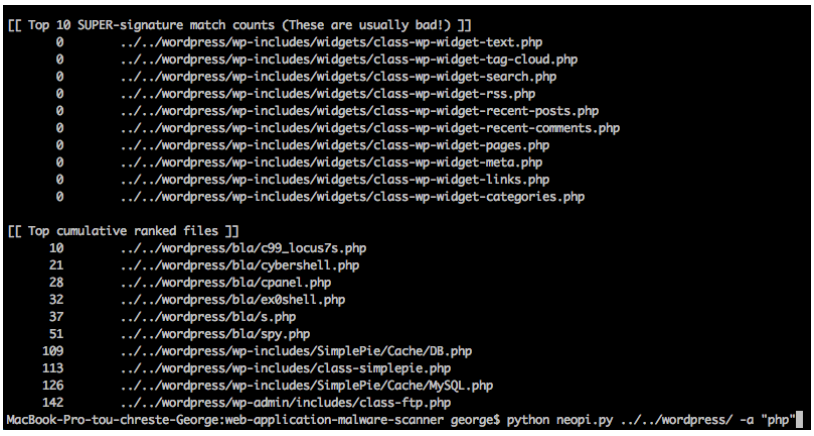
$$FPR = \frac{FP}{(FP + TN)} = \frac{10}{10 + 1471} = 0{,}68\ \%$$

όπως είναι κατανοητό η ιδανική τιμή του $FPR \to 0$.



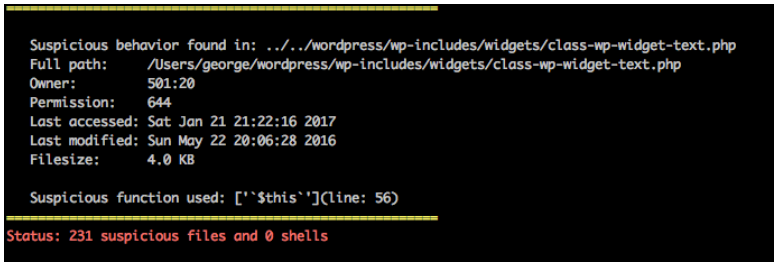
Εικόνα 16 Αποτελέσματα ελέγχου ιστότοπου wordpress με μηχανισμό μηχανικής μάθησης χωρίς την ύπαρξη κακόβουλου λογισμικού

Στην συνέχεια εκτελέστηκε αναζήτηση με το NeoPi τα αποτελέσματα του οποίου εμφανίζονται στην Εικόνα 17. Στην εν λόγω εφαρμογή, σε κάθε αρχείο που ερευνάται, εκτελούνται υπολογισμοί με 5 διαφορετικούς μηχανισμούς. Στο τέλος της έρευνας τα αποτελέσματα εμφανίζονται σε 5 πίνακες οι οποίοι περιλαμβάνουν τα 10 αρχεία με την υψηλότερη βαθμολογία, η οποία και καταδεικνύει τον βαθμό ύπαρξης κακόβουλου μηχανισμού. Για τον λόγο αυτό δεν μπορούμε να εξάγουμε ασφαλή συμπεράσματα για την αποδοτικότητα του καθώς τα 10 αυτά αρχεία που εντοπίζει έκαστος εκ των 5 μηχανισμών δεν σημαίνει απαραιτήτως ότι είναι κακόβουλα. Πέραν αυτών, αρχεία με μικρότερη βαθμολογία δεν εμφανίζονται με αποτέλεσμα να μην είναι δυνατός ο εντοπισμός εκείνων των οποίων αν και η κατάταξη είναι χαμηλότερη εν τούτοις είναι κακόβουλα. Επιπλέον ένα αρχείο είναι δυνατόν να εμφανιστεί σε περισσότερους από έναν μηχανισμούς, σε τέτοια περίπτωση, αν και υποδηλώνει περισσότερες πιθανότητες το αρχείο αυτό να είναι κακόβουλο εν τούτοις αυτό στερεί από τον χρήστη την δυνατότητα εμφάνισης κάποιου άλλου ίσως και νέου αρχείου, στο οποίο ο μηχανισμός θα εντοπίσει κάποια συνάφεια. Εκτελώντας την έρευνα με την εν λόγω εφαρμογή, οι εσωτερικοί μηχανισμοί του μας κατέδειξαν 50 αρχεία με την μέγιστη βαθμολογία ανά μηχανισμό, άρα έχουμε ένα ποσοστό **False Positive** της τάξης των **2,90%**.



Εικόνα 17 Αποτελέσματα αναζήτησης με την εφαρμογή NeoPi

Στην συνέχεια εκτελέστηκε έλεγχος με το πρόγραμμα Shell Detector. Στο Shell Detector γίνεται αναζήτηση για γνωστές συναρτήσεις που χρησιμοποιούνται όπως περιγράφτηκε στο κεφάλαιο 2 από κακόβουλα προγράμματα εμφανίζοντας όλα τα αρχεία που περιέχουν έστω και μία από αυτές τις συναρτήσεις. Παράλληλα ο μηχανισμός συνδέεται σε μια κεντρική βάση δεδομένων στο διαδίκτυο και αποστέλλει τα hash σε μορφή MD5 των υπό διερεύνηση αρχείων έτσι ώστε αν εντοπιστεί κακόβουλος κώδικας να ενημερωθεί ο χρήστης. Τα αποτελέσματα του ανωτέρω ελέγχου παρουσιάζονται στην Εικόνα 18 και αντιστοιχούν σε ποσοστό **False Positive 15,60%**. Αυτό είναι λογικό να συμβαίνει διότι ο μηχανισμός εμφανίζει τα αρχεία που περιέχουν «ύποπτες» συναρτήσεις καθώς αυτές μπορούν να χρησιμοποιηθούν και για κακόβουλή χρήση. Όπως φαίνεται αυτή η προσέγγιση είναι λανθασμένη ειδικά σε ιστότοπους με μεγάλο όγκο αρχείων όπως το wordpress, καθώς καθίσταται μη παραγωγική η ανάλυση όλων των αρχείων που εμφάνισε ο μηχανισμός (στο παράδειγμα μας 231) για τον εντοπισμό κακόβουλου λογισμικού.



Εικόνα 18 Αποτελέσματα αναζήτησης με την εφαρμογή Shell Detector

Τέλος, εκτελέστηκε αναζήτηση με το πρόγραμμα ClamAV. Το ClamAV είναι μια εφαρμογή αντι-ιϊκού λογισμικού ανοιχτού κώδικα, σχεδιασμένη να υποστηρίζει όλα τα λειτουργικά και όλους τους τύπους κακόβουλου λογισμικού. Το εν λόγω antivirus δεν έχει μηχανισμούς εντοπισμού αγνώστου κακόβουλου αρχείου αλλά βασίζεται κυρίως στην βάση δεδομένων του στην οποία οι χρήστες μπορούν να υποβάλλουν μέσω του επίσημου ιστότοπου ένα αρχείο ως κακόβουλο και η εν λόγω βάση να ενημερωθεί. Αν και στοχεύει κυρίως σε κακόβουλο λογισμικό που εκτελείται στον πυρήνα ή σε διεργασίες του συστήματος, εν τούτοις είναι σε θέση να εντοπίσει κώδικα που είναι εγκατεστημένος σε ιστοσελίδες και τοπικές βάσεις δεδομένων. Η βάση δεδομένων του (virus definitions) ενημερώθηκε την ημέρα των δοκιμών πριν την εκτέλεση των ελέγχων μέσω της εντολής freshclam.

```
sansforensics@siftworkstation:~/Desktop$ freshclam;clamscan -ir wordpress/
ClamAV update process started at Sat Jan 28 10:47:40 2017
main.cvd is up to date (version: 57, sigs: 4218790, f-level: 60, builder: amishh
ammer)
daily.cvd is up to date (version: 22958, sigs: 1477638, f-level: 63, builder: ne
o)
bytecode.cld is up to date (version: 290, sigs: 55, f-level: 63, builder: neo)

----------- SCAN SUMMARY -----------
Known viruses: 5690973
Engine version: 0.99.2
Scanned directories: 138
Scanned files: 1480
Infected files: 0
Data scanned: 42.32 MB
Data read: 19.76 MB (ratio 2.14:1)
Time: 20.924 sec (0 m 20 s)
sansforensics@siftworkstation:~/Desktop$
```

Εικόνα 19 Αποτελέσματα σάρωσης με τον μηχανισμό ClamAV σε ασφαλή ιστότοπο wordpress

Όπως βλέπουμε στην Εικόνα 19 ο μηχανισμός παρότι διαθέτει ~5.700.000 υπογραφές κακόβουλου λογισμικού εντούτοις δεν εντόπισε κάποιο ύποπτο αρχείο, αυτό σημαίνει ότι η τιμή της ορθότητας, όπως και ο ρυθμός True Negative είναι 100% ενώ ο ρυθμός False Positive είναι 0% που σημαίνει ότι σε αυτή την φάση παρουσιάζει το ιδεατό σενάριο της σωστής ταξινόμησης των αρχείων.

Στον πίνακα που ακολουθεί παρατίθενται όλα τα ανωτέρω αποτελέσματα

	Μηχανισμός ΜΜ		*Shell Detector*		*NeoPi*		*ClamAV*	
	πλήθος	%	πλήθος	%	πλήθος	%	Πλήθος	%
True – Positive	-	-	-	-	-	-	-	-
True - Negative	1471	99,32 %	1250	84,40%	1421	97,10%	1481	100%
False - Positive	10	0,68%	231	15,60%	50	2.90%	0	0%
False - Negative	-	-	-	-	-	-	-	-

Πίνακας 3 Αποτελέσματα Μηχανισμών σε μη Μολυσμένο Ιστότοπο WordPress

5.2.2 - Αξιολόγηση με εισαγωγή κακόβουλων αρχείων.

Στην συνέχεια πήραμε 10 κακόβουλα αρχεία από το υποσύνολο A (γνωστά στον μηχανισμό) και 11 κακόβουλα αρχεία από το υποσύνολο B (άγνωστα στον μηχανισμό), τα τοποθετήσαμε εντός του ιστότοπου, όπου και εκτελέσαμε εκ νέου την αναζήτηση. Αρχικά η αναζήτηση έγινε με τον προτεινόμενο μηχανισμό μηχανικής μάθησης συν τον μηχανισμό μαύρης λίστας, τα αποτελέσματα της οποίας απεικονίζονται στην Εικόνα 20.

Εικόνα 20 Αποτελέσματα ελέγχου ιστότοπου wordpress με μηχανισμό μηχανικής μάθησης με την ύπαρξη κακόβουλου λογισμικού

Από τα αποτελέσματα της αναζήτησης συμπεραίνουμε ότι ο μηχανισμός εντόπισε 29 αρχεία με κακόβουλο κώδικα. Γνωρίζοντας ότι ο μηχανισμός, στην προηγούμενη αναζήτηση, παρουσίασε 10 αρχεία με **False Positive** ένδειξη, συμπεραίνουμε ότι εντοπίστηκαν 29-10 = 19 αρχεία με συμπεριφορά κακόβουλου κώδικα. Από αυτά τα 12 ταυτοποιήθηκαν από τον μηχανισμό "μαύρης λίστας" και τα υπόλοιπα 7 από τον μηχανισμό μηχανικής μάθησης οπότε έχουμε ένδειξη **True Positive** σε 19 αρχεία. Όμως κατά την διάρκεια των δοκιμών εισήχθησαν 20 κακόβουλα αρχεία, άρα ο μηχανισμός απέτυχε να εντοπίσει 20-19 = 1 αρχείο οπότε έχουμε ένδειξη **False Negative** σε ποσοστό **5%**. Επίσης ενώ εισήχθησαν 10 αρχεία από το υποσύνολο Α, εν τούτης ταυτοποιήθηκαν από το μηχανισμό 12. Αυτό έγινε διότι διαπιστώθηκε ότι τα 2 αυτά αρχεία ήταν μεταγενέστερες εκδόσεις από κακόβουλο λογισμικό που η υπογραφή τους έχει εισαχθεί στην μαύρη λίστα. Συμπερασματικά λόγω του αλγορίθμου fuzzy hash που χρησιμοποιείται, η ταυτοποίηση των εν λόγω αρχείων ήταν άμεση.

	TRUE POSITIVE	**FALSE POSITIVE**	**TRUE NEGATIVE**	**FALSE NEGATIVE**
ΜΗ ΚΑΚΟΒΟΥΛΑ ΑΡΧΕΙΑ	-	10	1471	-
ΜΗΧΑΝΙΣΜΟΣ «ΜΑΥΡΗΣ ΛΙΣΤΑΣ»	12	-	0	-
ΜΗΧΑΝΙΣΜΟΣ ΜΗΧΑΝΙΚΗΣ ΜΑΘΗΣΗΣ	7	-	0	1
ΣΥΝΟΛΟ	19	10	1471	1

Πίνακας 4 Αποτελέσματα αναζήτησης κακόβουλου λογισμικού προτεινόμενου μηχανισμού

Με βάση τα παραπάνω αποτελέσματα υπολογίζουμε την απόδοση ως ακολούθως:

$$TPR \text{ ή } recall = \frac{TP}{(TP + TN)} = 95\ \%$$

$$FPR = \frac{FP}{(FP + TN)} = 0{,}68\ \%$$

$$precision = \frac{TP}{(TP + FP)} = 65{,}52\ \%$$

$$accuracy = \frac{TP + TN}{(TP + TN + FP + FN)} = 99{,}27\ \%$$

Από τα παραπάνω βλέπουμε ότι ο μηχανισμός ήταν σε θέση να εντοπίσει το 95% του εγκατεστημένου κακόβουλου λογισμικού από το οποίο η τιμή της **ορθότητας** (accuracy) τείνει στο 100% ενώ ταυτόχρονα το ποσοστό εμφάνισης σφάλματος περιορίζεται σε μόλις 0,68% που σημαίνει ότι **ήταν σε θέση να αναγνωρίσει αρχεία με κακόβουλο κώδικα, είτε αυτός είναι γνωστός, είτε άγνωστος σχεδόν σε όλες τις περιπτώσεις**.

Στην συνέχεια, έγινε αναζήτηση με χρήση του προγράμματος NeoPi, όπου μέρος των αποτελεσμάτων βλέπουμε στην Εικόνα 21 και αναλυτικά τα αποτελέσματα αποτυπώνονται στον πίνακα 5.

```
[[ Top 10 longest word files ]]
   34833        ../../wordpress/wp-includes/css/dashicons.min.css
   34833        ../../wordpress/wp-includes/css/dashicons.css
   31483        ../../wordpress/bla/c99_locus7s.php
   25791        ../../wordpress/bla/s.php
   19149        ../../wordpress/wp-content/themes/twentyfifteen/genericons/genericons.css
   18654        ../../wordpress/wp-content/themes/twentysixteen/genericons/genericons.css
    5853        ../../wordpress/wp-includes/js/wp-emoji-release.min.js
    5853        ../../wordpress/wp-includes/js/twemoji.min.js
    5851        ../../wordpress/wp-includes/js/twemoji.js
    4787        ../../wordpress/wp-includes/js/tinymce/skins/lightgray/skin.min.css

[[ Top 10 signature match counts ]]
     141        ../../wordpress/bla/c99_locus7s.php
      45        ../../wordpress/bla/ex0shell.php
      32        ../../wordpress/wp-includes/js/tinymce/tinymce.min.js
      32        ../../wordpress/bla/poison.php
      32        ../../wordpress/bla/Gamma Web Shell.php
      30        ../../wordpress/bla/spy.php
      26        ../../wordpress/wp-admin/includes/class-ftp.php
      22        ../../wordpress/wp-includes/js/jquery/jquery.js
      14        ../../wordpress/bla/KAdot Universal Shell v0.1.6.php
      13        ../../wordpress/wp-includes/Text/Diff/Engine/native.php
```

Εικόνα 21 Δείγμα αποτελεσμάτων μηχανισμού NeoPi με κακόβουλο λογισμικό

Όπως έχουμε ήδη αναφέρει και στην προηγούμενη ενότητα το NeoPi δεν είναι σε θέση να εμφανίσει περισσότερα των 50 αρχείων συνολικά. Αυτό σημαίνει ότι τα παρακάτω αποτελέσματα δεν είναι δυνατόν να έχουν την απαιτούμενη ακρίβεια που χρειαζόμαστε για να εξάγουμε ασφαλή συμπεράσματα. Επιπλέον, από τα εικονιζόμενα αποτελέσματα, εντοπίστηκαν 7 αρχεία με κακόβουλο κώδικα, ενώ σε κάποιους μηχανισμούς αυτά επαναλαμβάνονταν με αποτέλεσμα το πλήθος των μοναδικών αποτελεσμάτων να μειωθεί από 50 σε 31. Έτσι σύμφωνα με τα νεότερα δεδομένα έχουμε:

TRUE POSITIVE	FALSE POSITIVE	TRUE NEGATIVE	FALSE NEGATIVE
7	24	1470	13

Πίνακας 5 Αποτελέσματα αναζήτησης με την εφαρμογή NeoPi

Σύμφωνα με τα ανωτέρω αποτελέσματα έχουμε

TPR ή $recall = 35\%$

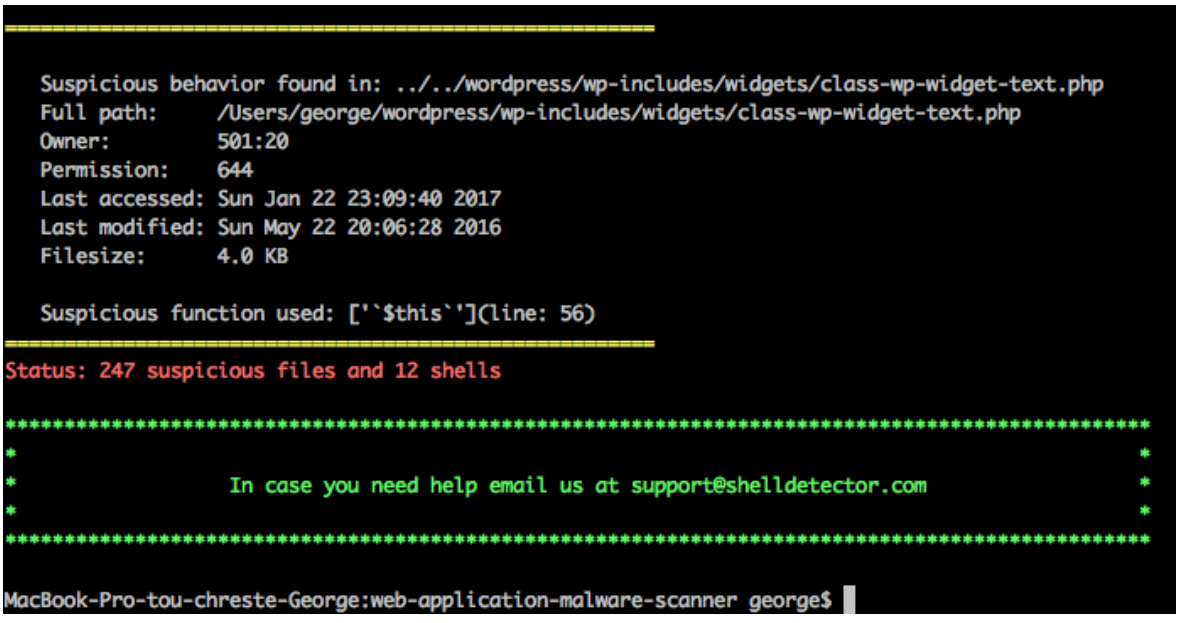
$FPR = 1{,}61\%$

$precision = 22{,}58\%$

$accuracy = 97{,}56\%$

Από τα παραπάνω προκύπτει ότι ο μηχανισμός έχει ρυθμό True Positive μόλις 35% το οποίο δεν είναι αποδεκτό για την λειτουργία ενός σύγχρονου anti-malware μηχανισμού. Ωστόσο το ποσοστό ορθότητας, αν και είναι 97,56%, υποδεικνύει ότι ο μηχανισμός είναι σε θέση να κάνει αρκετά σωστή διάκριση μεταξύ των αρχείων. Εν τούτοις αυτό δεν είναι αρκετό καθώς η ακρίβεια μεταξύ αυτών είναι μόλις στο 22,58% που σημαίνει ότι πολλά αρχεία τα οποία είναι ασφαλή εμφανίζονται ως κακόβουλα.

Ο επόμενος μηχανισμός που εξετάστηκε είναι ο Shell Detector τα αποτελέσματα του οποίου εμφανίζονται στην Εικόνα 22 και απεικονίζονται στον πίνακα 6. Όπως βλέπουμε ο μηχανισμός εντόπισε 247 αρχεία που θεωρεί ότι είναι ύποπτα για κακόβουλο λογισμικό και ταυτοποίησε 12 ως κακόβουλο λογισμικό. Γνωρίζουμε από την προηγούμενη ανάλυση ότι τα False Positive αρχεία είναι 231 άρα 247-231=16 αρχεία εκ των οποίων τα 12 ταυτοποιήθηκαν από την βάση δεδομένων της.



Εικόνα 22 Αποτελέσματα μηχανισμού Shell Detector με την ύπαρξη κακόβουλου λογισμικού

TRUE POSITIVE	FALSE POSITIVE	TRUE NEGATIVE	FALSE NEGATIVE
16	231	1249	5

Πίνακας 6 Αποτελέσματα shell detector

Με βάση τα παραπάνω έχουμε:

TPR ή $recall = 76{,}19\%$

$FPR = 15{,}61\%$

$precision = 6{,}48\%$

$accuracy = 84{,}28\%$

Παρατηρούμε ότι ο Shell Detector έχει πολύ μικρό ποσοστό ακρίβειας, μόλις 6,48%. Αυτό συμβαίνει διότι παρουσιάζει πολλά False Positive σφάλματα κατά την ανάλυση των αρχείων. Όμως είναι σε θέση να εντοπίσει με μεγάλο ποσοστό ορθότητας 84,28% όπως και τα True Positives που εμφανίζει είναι ορθά κατά 76,19%.

Τέλος, εξετάστηκε ο μηχανισμός ClamAV τα αποτελέσματα του οποίου απεικονίζονται στην εικόνα 23.

```
sansforensics@siftworkstation:~/Desktop$ clamscan -ir wordpress/
wordpress/bla/h4ntu shell [powered by tsoi].php: Win.Trojan.Remoteadmin-4 FOUND
wordpress/bla/ex0shell.php: Win.Trojan.Shell-15 FOUND
wordpress/bla/cpanel.php: Win.Trojan.Shell-58 FOUND
wordpress/bla/c99_locus7s.php: Php.Trojan.C99Shell-2 FOUND
wordpress/bla/cybershell.php: Win.Trojan.HackerTool-2 FOUND

----------- SCAN SUMMARY -----------
Known viruses: 5690973
Engine version: 0.99.2
Scanned directories: 139
Scanned files: 1501
Infected files: 5
Data scanned: 43.07 MB
Data read: 20.26 MB (ratio 2.13:1)
Time: 21.200 sec (0 m 21 s)
```

Εικόνα 23 Αποτελέσματα ανάλυσης με χρήση του μηχανισμού ClamAV

Γνωρίζουμε ότι ο μηχανισμός δεν εμφανίζει σφάλμα False Positive κατά την πρώτη φάση οπότε αναλύοντας τα στοιχεία που εμφανίζονται στην εικόνα 23 έχουμε:

TRUE POSITIVE	FALSE POSITIVE	TRUE NEGATIVE	FALSE NEGATIVE
5	0	1480	16

Με βάση τα παραπάνω έχουμε:

TPR ή $recall = 23{,}81\%$

$FPR = 0\%$

$precision = 100\%$

$accuracy = 98{,}93\%$

Παρατηρούμε ότι παρόλο που ο μηχανισμός απέτυχε να ταυτοποιήσει τα ¾ του κακόβουλου λογισμικού που βρίσκονταν εγκατεστημένο, καθώς το ποσοστό ανάκλησης είναι μόλις 23,81%, εντούτοις διαπιστώνουμε ότι έχει ποσοστό ορθότητας εξαιρετικά υψηλό 98,93 % ενώ το ποσοστό ακριβείας είναι 100%. Αυτό

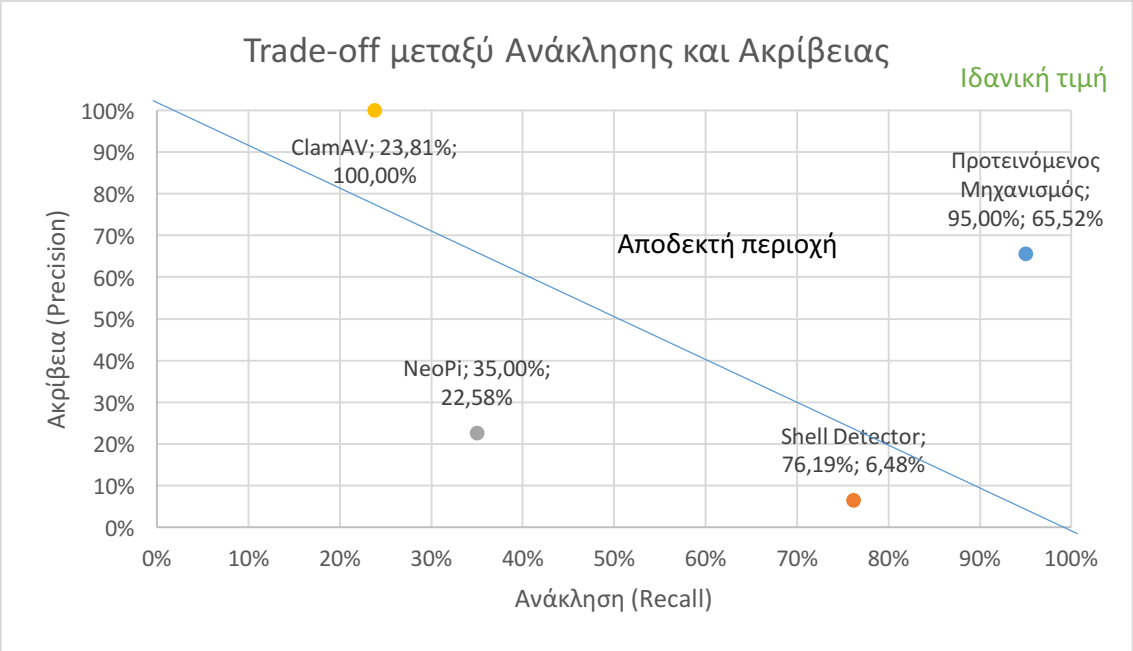
οφείλεται στο γεγονός ότι δεν ταυτοποίησε ασφαλή αρχεία ως κακόβουλα με αποτέλεσμα το ποσοστό εμφάνισης False Positive να είναι 0%.

Συμπεραίνουμε λοιπόν ότι απαιτούνται όλοι οι μετρητές ώστε να μας καταδείξουν μια γενική εικόνα κατάταξης των ανωτέρω μηχανισμών anti-malware. Στον πίνακα 7 εμφανίζονται συγκεντρωτικά όλα τα δεδομένα που συλλέξαμε ανά μηχανισμό.

	Πλήθος			
	Χρήση TN	Shell Detector	NeoPi	ClamAV
Σύνολο Αρχείων	1501	1501	1501	1501
Σύνολο Μολυσμένων Αρχείων Εντοπίστηκαν	20 29	247	50	5
True Positive	19	16	7	5
True Negative	1471	1249	1457	1480
False Positive	10	231	24	0
False Negative	1	5	13	16

Πίνακας 7 Συγκεντρωτικά δεδομένα μηχανισμών anti-malware κατά την αξιολόγηση στον ιστότοπο wordpress

Για να γίνει ξεκάθαρη η χρησιμότητα των τιμών ανάκλησης και ακρίβειας, στην εικόνα 22 έγινε οπτικοποίηση της αντιπαραβολής μεταξύ τους [23][24]. Όσο κοντά στην πάνω αριστερή γωνία είναι η τιμή, τόσο ο μηχανισμός είναι ακριβέστερος στον εντοπισμό κακόβουλου λογισμικού, έχει λιγότερα false positive σφάλματα, εμφανίζοντας όμως λιγότερα true positive. Στον αντίποδα των παραπάνω όσο πλησιέστερα στην κάτω δεξιά γωνία βρίσκεται ο μηχανισμός, τόσο είναι σε θέση να εμφανίζει περισσότερα αρχεία ως κακόβουλα. Επιλέγονται δηλαδή περισσότερα αρχεία ως κακόβουλα με αποτέλεσμα να εμφανίζει μεγάλο false negative σφάλμα. Ιδανική τιμή είναι όταν ο δείκτης βρίσκεται όσο το δυνατόν κοντινότερα στην πάνω δεξιά γωνία που σημαίνει ότι είναι σε θέση να εμφανίζει πολλά true positive αρχεία με όσο το δυνατόν λιγότερα σφάλματα false positive. Η μπλε γραμμή χωρίζει το γράφημα σε 2 περιοχές είναι η γραφική απεικόνιση του μέσου όρου μεταξύ ανάκλησης και ακρίβειας στην οποία οι τιμές που εμφανίζονται στην άνω περιοχή θεωρούνται αποδεκτές, ενώ αυτές που βρίσκονται στην κάτω περιοχή μη αποδεκτές [25].



Εικόνα 24 Αντιπαραβολή μεταξύ βαθμού ανάκλησης και βαθμού ακριβείας για τον ιστότοπο Wordpress

Όπως βλέπουμε στην εικόνα 23 ο μηχανισμός μηχανικής μάθησης είναι ο μοναδικός που βρίσκεται όσο το δυνατόν κοντινότερα στην ιδεατή τιμή και μάλιστα με μεγάλη διαφορά σε σχέση με τους υπολοίπους τρεις, γεγονός που καταδεικνύει ότι ο μηχανισμός λειτουργεί σε πολύ ικανοποιητικά επίπεδα. Τέλος ένας επιπλέον δείκτης μέτρησης είναι η τιμή F1 (F1 Score ή F-measure) η οποία εκφράζεται ως η **αρμονική τιμή της ανάκλησης και ακρίβειας**, η οποία έχει εύρος $F_1 = \{0,1\}$ και εκφράζεται με τον τύπο

$$F_1 = \frac{2TP}{2TP + FP + FN}$$

όπου 1 η ιδανική και 0 η χείριστη τιμή [26]. Στον πίνακα 8 που ακολουθεί βλέπουμε συγκεντρωτικά τα αποτελέσματα καθώς και τις τιμές όλων των μετρητών που χρησιμοποιήσαμε για να αξιολογήσουμε τους μηχανισμούς. Με πράσινο χρώμα εμφανίζονται οι βέλτιστες τιμές και με κόκκινο οι χείριστες. Παρατηρούμε ότι η αρμονική τιμή του μηχανισμού μηχανικής μάθησης είναι πάλι υψηλότερη σε σχέση με τους υπολοίπους μηχανισμούς.

Ένα επιπλέον εργαλείο που θα δανειστούμε από την στατιστική για την αξιολόγηση των μηχανισμών, είναι το γράφημα ROC. Το γράφημα ROC είναι ένα γράφημα δύο διαστάσεων στο οποίο στον άξονα x αποτυπώνεται η τιμή TPR ενώ στον y η τιμή FPR, με σκοπό την αντιπαραβολή των ορθών και λανθασμένων ταξινομήσεων. Οι τιμές που απεικονίζονται εκτείνονται από [0,0] έως [1,1] τα οποία εκφράζονται ως ακολούθως [27][28]:

α) [0,0] καμία ορθή πρόβλεψη

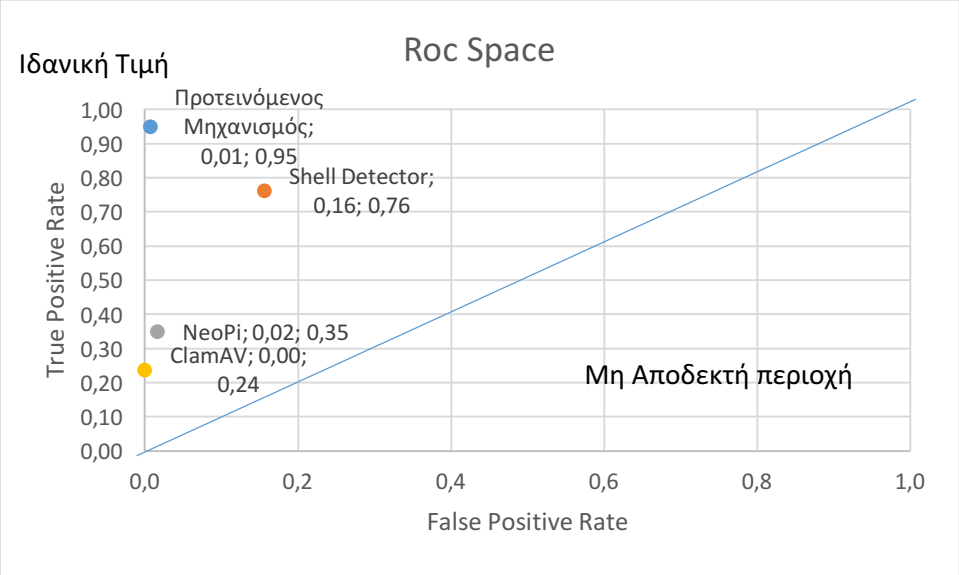
β) [0,1] όλες οι προβλέψεις TP είναι ορθές και δεν εμφανίστηκαν FP (ιδανική τιμή)

γ) [1,0] καμία ορθή πρόβλεψη TP αλλά όλες οι FP είναι ορθές.

δ) [1,1] όλες οι TP προβλέψεις είναι σωστές αλλά όλες οι FP προβλέψεις είναι λάθος.

Με βάση τα παραπάνω θα πρέπει οι τιμές των μετρήσεων να είναι όσο το δυνατόν πλησιέστερα στο σημείο [0,1] ενώ οι τιμές που βρίσκονται κάτω από την μπλε διαγώνια γραμμή που βλέπουμε στην εικόνα 21, είναι μη αποδεκτές και θα πρέπει να αποφεύγονται κατά τον σχεδιασμό παρομοίων μηχανισμών.

Στο γράφημα που ακολουθεί, μπορούμε να διακρίνουμε ότι ο μηχανισμός μηχανικής μάθησης, έχει πετύχει υψηλότερα ποσοστά επιτυχίας στον εντοπισμό κακόβουλων αρχείων και παρουσιάζει μεγαλύτερη αξιοπιστία, σε σχέση με τους υπολοίπους μηχανισμούς.



Εικόνα 25 Γράφημα ROC για την αξιολόγηση των μηχανισμών στον ιστότοπο Wordpress

Τέλος, σε αυτή την φάση της αξιολόγησης και με βάση τον πίνακα 8 παρατηρούμε ότι ο μηχανισμός μηχανικής μάθησης παρουσιάζει μεγάλο ποσοστό εντοπισμού και ταυτοποίησης κακόβουλων προγραμμάτων (ανεξαρτήτως αν είναι γνωστά ή όχι), μεγάλο ποσοστό ακρίβειας των αποτελεσμάτων και τέλος παρουσιάζει τα λιγότερα σφάλματα True Negative σε σχέση με τους υπάρχοντες μηχανισμούς anti-malware.

	Προτειν. Μηχανισμός	Shell Detector	NeoPi	ClamAV
True Positive Rate (TPR)	95,00%	76,19%	35,00%	23,81%
True Negative Rate (SPC)	99,32%	84,39%	98,38%	100,00%
Ακρίβεια (Precision) or Positive Predictive Value	65,52%	6,48%	22,58%	100,00%
Negative Predictive Value	99,93%	99,60%	99,12%	98,93%
False Positive Rate	0,68%	15,61%	1,62%	0,00%
False Negative Rate	5,00%	23,81%	65,00%	76,19%
False Discovery Rate	34,48%	93,52%	77,42%	0,00%
Ορθότητα (Accuracy)	99,27%	84,28%	97,53%	98,93%
F1 Score	0,77	0,11	0,27	0,38

Πίνακας 8 Σύγκριση Αποτελεσμάτων Μηχανισμών anti-malware στον ιστότοπο WordPress

5.3 – 2η αξιολόγηση του μηχανισμού σε αρχεία ιστοτόπου Joomla

Όπως και στον ιστότοπο wordpress έτσι και εδώ εκτελέσαμε αρχικά αναζήτηση για να εντοπίσουμε τα false positives που εμφανίζουν οι μηχανισμοί. Στην συνέχεια εγκαταστάθηκαν 21 αρχεία με κακόβουλο λογισμικό με την ίδια μεθοδολογία που ακολουθήθηκε στην προηγούμενη ενότητα. Τέλος, στον ιστότοπο εκτελέστηκε ανάλυση και καταγράφηκαν τα αποτελέσματα στον πίνακα που ακολουθεί.

	Χρήση TN	Shell Detector	NeoPi	ClamAV
Σύνολο Αρχ.	2643			
Μολυσμένων Αρχείων	21			
True Positive	21	15	13	12
True Negat.	2614	2305	2597	2622
False Positive	8	317	25	0
False Negat.	0	6	8	9

Πίνακας 9 Αποτελέσματα ανάλυσης αρχείων στον ιστότοπο Joomla

Με βάση αυτά τα αποτελέσματα, όπως και στην προηγούμενη ενότητα έγινε υπολογισμός των τύπων. Παρατηρούμε λοιπόν ότι ο μηχανισμός μηχανικής μάθησης **ταυτοποίησε το σύνολο του κακόβουλου λογισμικού γνωστού και αγνώστου**, η ακρίβεια του αυξήθηκε ενώ ταυτόχρονα ο ρυθμός απόδοσης σφαλμάτων έχει μειωθεί σε σχέση με τις προηγούμενες μετρήσεις. Αυτό οφείλεται στο γεγονός ότι ο μηχανισμός μηχανικής μάθησης σε **κάθε διαφορετικό έλεγχο** που εκτελεί, εκπαιδεύεται έτσι ώστε να είναι σε θέση να εντοπίζει με μεγαλύτερη ακρίβεια το νέο

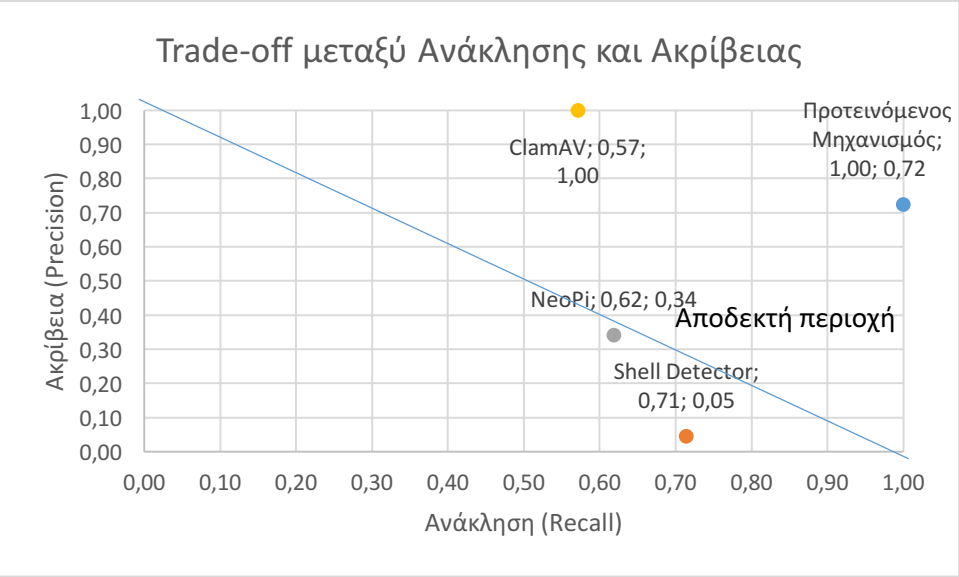
κακόβουλο λογισμικό, και η ταξινόμηση των αρχείων να βελτιώνεται όλο και περισσότερο. Στον πίνακα 10 που ακολουθεί παρουσιάζονται τα αποτελέσματα των υπολογισμών για τον ιστότοπο τύπου Joomla.

	Μηχανισμός Μηχ. Μάθησης	Shell Detector	NeoPI	ClamAV
True Positive Rate (TPR)	100,00%	71,43%	61,90%	57,14%
True Negative Rate (SPC)	99,69%	87,91%	99,05%	100,00%
Ακρίβεια (Precision) or Positive Predictive Value	72,41%	4,52%	34,21%	100,00%
Negative Predictive Value	100,00%	99,74%	99,69%	99,66%
False Positive Rate	0,31%	12,09%	0,95%	0,00%
False Negative Rate	0,00%	28,57%	38,10%	42,86%
False Discovery Rate	27,59%	95,48%	65,79%	0,00%
Ορθότητα (Accuracy)	99,70%	87,78%	98,75%	99,66%
F1 Score	84,00%	8,50%	44,07%	72,73%

Πίνακας 10 Σύγκριση Αποτελεσμάτων Μηχανισμών anti-malware στον ιστότοπο Joomla

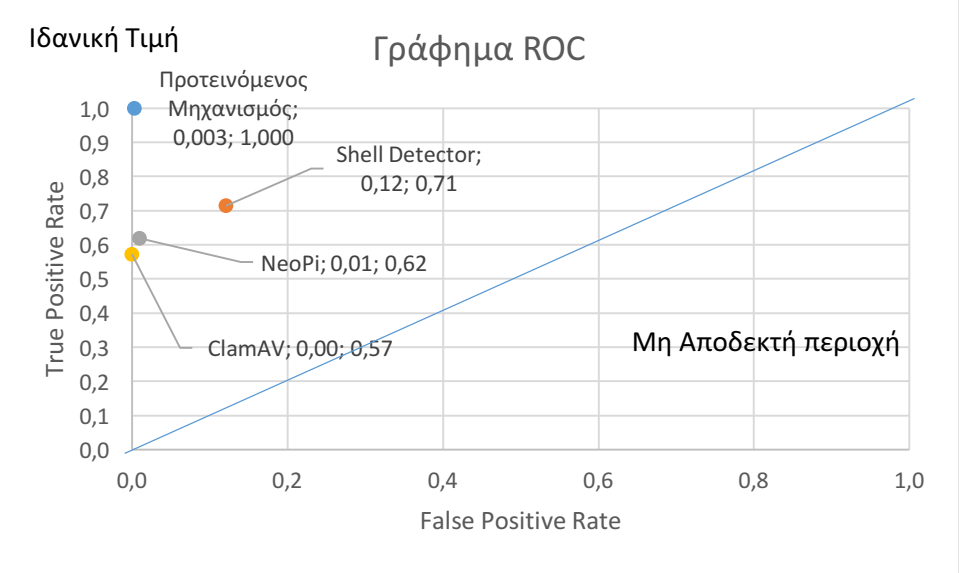
Σύμφωνα με τα αποτελέσματα που απεικονίζονται στον ανωτέρω πίνακα παρατηρούμε ότι ο μηχανισμός μηχανικής μάθησης έχει επιτύχει καλύτερες τιμές σε 5 από τους 9 δείκτες αποδοτικότητας ενώ στους υπολοίπους 4, καμία τιμή δεν αποτελεί την χειρότερη τιμή των μετρήσεων. Το ποσοστό True Positive αυξήθηκε από το 95% στο 100% το οποίο εκτός του γεγονότος ότι αποτελεί την καλύτερη τιμή μεταξύ των 4 μηχανισμών, σημαίνει ότι έχει επιτευχθεί το ιδανικό ποσοστό για τον εντοπισμό κακόβουλου λογισμικού. Παρόλα αυτά όμως ο μηχανισμός εξακολουθεί να παρουσιάζει σφάλματα False Positive, μειωμένα όμως σε σχέση με την προηγούμενη αξιολόγηση που σημαίνει ότι ο μηχανισμός επιδέχεται επιπλέον εκπαίδευση έτσι ώστε να είναι σε θέση να τα ελαττώσει. Τέλος παρατηρούμε ότι η απόδοση F1 Score εμφανίζει τιμή 84% ενώ η αμέσως επόμενη τιμή είναι ~73 %, το οποίο σημαίνει ότι ο προτεινόμενος μηχανισμός έχει μεγαλύτερα ποσοστά επιτυχίας, σε σχέση με τις τρέχουσες τεχνολογίες.

Στην εικόνα 24 που ακολουθεί παρατηρούμε την αντιπαραβολή μεταξύ της Ακρίβειας και της Ανάκλησης. Σε αυτή την αξιολόγηση όλοι οι μηχανισμοί τα πήγαν καλύτερα σε σχέση με τις προηγούμενες μετρήσεις που παρουσίασαν. Όπως παρατηρούμε την καλύτερη απόδοση την έχει επιτύχει ο μηχανισμός μηχανικής μάθησης ο οποίος βρίσκεται πολύ κοντά στην ιδανική τιμή [1,1]



Εικόνα 26 Αντιπαραβολή μεταξύ βαθμού ανάκλησης και βαθμού ακριβείας για τον ιστότοπο Joomla

Τέλος κατά την αποτύπωση του γραφήματος ROC μπορούμε να διακρίνουμε ότι ο υπό ανάπτυξη μηχανισμός έχει επιτύχει και σε αυτή την περίπτωση, την μέγιστη τιμή στα TP και την σχεδόν μηδενική τιμή στον ρυθμό FP, με τον εν λόγω μηχανισμό να επιτυγχάνει εξαιρετικές επιδόσεις και σε αυτή την αξιολόγηση.



Εικόνα 27 Γράφημα ROC της αξιολόγησης των μηχανισμών anti-mailware στον ιστότοπο Joomla

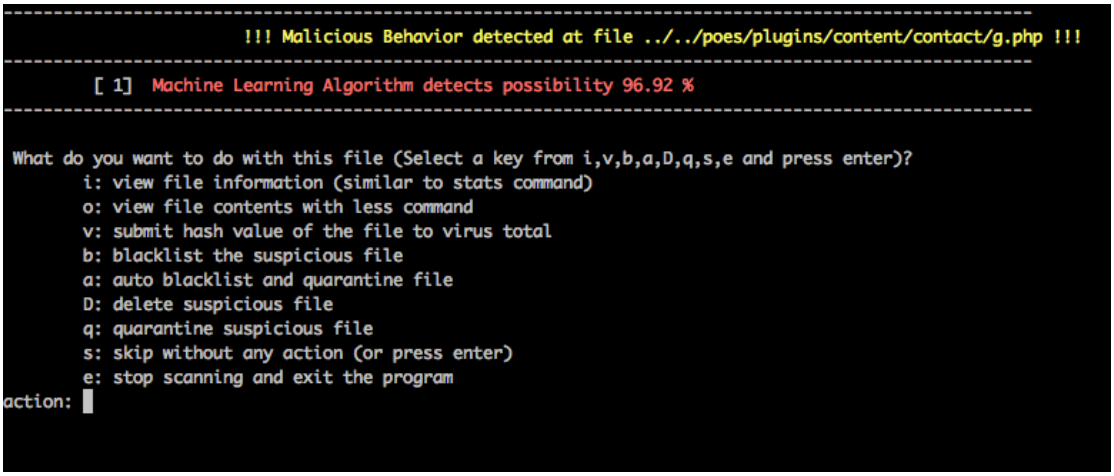
5.4 - Αξιολόγηση του μηχανισμού σε πραγματικές συνθήκες

Κατά την διάρκεια συγγραφής της παρούσας πτυχιακής εργασίας κλήθηκα να βοηθήσω σε περιστατικό όπου εισβολέας είχε εγκαταστήσει πολλαπλό κακόβουλο λογισμικό σε ιστότοπο τύπου WordPress. Συγκεκριμένα το λογισμικό αυτό, ενεργοποιούνταν αποκλειστικά όταν φορητή συσκευή επισκέπτονταν τον ιστότοπο και στην συνέχεια εγκαθιστούσε σε αυτή, ανάλογα με το λειτουργικό σύστημα που

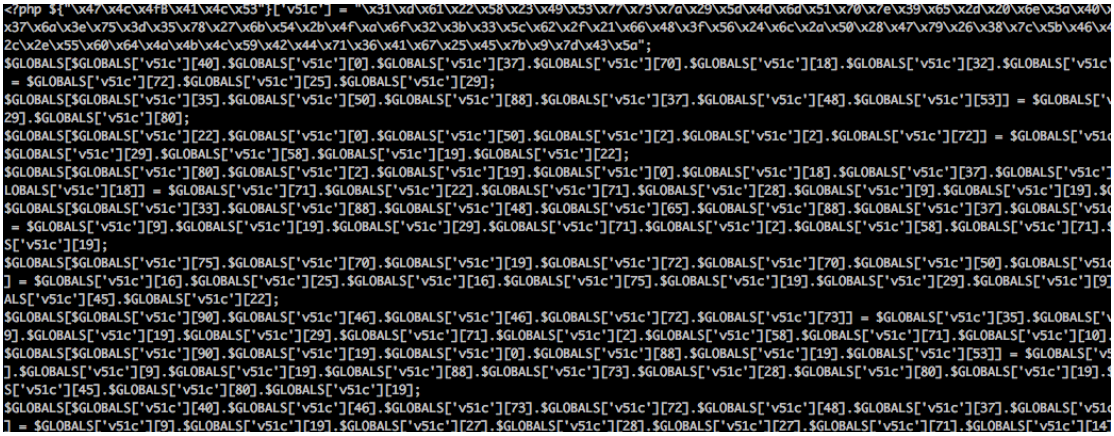
έφερε, κακόβουλο λογισμικό. Συνολικά εντοπίστηκαν 36 κακόβουλα αρχεία διασκορπισμένα σε διάφορες θέσεις του ιστοτόπου. Τα εν λόγω αρχεία ήταν άγνωστα ως προς την εκπαίδευση του μηχανισμού, μάλιστα σε αρκετά σημεία είχε τροποποιηθεί ο πρωτότυπος κώδικας του ιστοτόπου, και είχε εισαχθεί σε αυτό ως εμβόλιμα ο κακόβουλος κώδικας. Σε όλες τις περιπτώσεις ο επικίνδυνος κώδικας, εντοπίστηκε και απομονώθηκε ενώ εμφάνισε μόλις 4 σφάλματα τύπου false positive.

Αξίζει να τονιστεί ότι ο ιστότοπος αρχικά πέρασε από έλεγχο με εμπορικό anti-malware λογισμικό, το οποίο κατάφερε να ταυτοποιήσει μόλις 17 κακόβουλα αρχεία, ενώ δεν είχε καμιά επιτυχία στον κώδικα που εισήχθη στα πρωτότυπα αρχεία του ιστοτόπου.

Στην εικόνα 28 βλέπουμε ένα δείγμα κατά την διάρκεια ανάλυσης των αρχείων του ιστοτόπου. Παρατηρούμε ότι το όνομα του αρχείου είναι κοινό και δεν ξεχωρίσει έτσι ώστε να κινήσει τις υποψίες του διαχειριστή. Τέλος, στην εικόνα 29 παρατηρούμε ένα δείγμα από τον κώδικα που εμπεριείχε το συγκεκριμένο αρχείο.



Εικόνα 28 Στιγμιότυπο κατά την ανάλυση των αρχείων σε παραγωγικό ιστότοπο που έχει παραβιαστεί.



Εικόνα 29 Στιγμιότυπο των περιεχομένων του υπό ανάλυση αρχείου g.php

Η σελίδα αφέθηκε σκοπίμως κενή

Κεφάλαιο 6 – Αδυναμίες και Βελτιώσεις του Μηχανισμού Μηχανικής Μάθησης.

Ο μηχανισμός μηχανικής μάθησης πέτυχε αναμφίβολα εξαιρετικές επιδόσεις στον εντοπισμό κακόβουλων αρχείων, όμως εμφάνισε και αρκετές αδυναμίες οι οποίες αξίζουν να εντοπιστούν και να διορθωθούν ώστε αυτός να μπορεί να χρησιμοποιηθεί με μεγαλύτερη ακρίβεια και από απλούς χρήστες.

6.1 - Αδυναμίες

Όπως παρατηρήσαμε στο προηγούμενο κεφάλαιο, ο μηχανισμός είναι σε θέση να εντοπίσει κακόβουλο λογισμικό σε πολύ μεγάλο ποσοστό επιτυχίας, ακόμα και όταν αυτό είναι άγνωστο, με ρυθμό απόδοσης True Positive που πλησιάζει το απόλυτο 100%. Παρόλα αυτά όμως και στις δύο δοκιμασίες ο αλγόριθμος εμφάνισε, έστω και σε μικρό ποσοστό, σφάλμα False Positive. Αυτό σημαίνει ότι οι χειριστές του προγράμματος πρέπει να είναι αρκετά εξοικειωμένοι με τρόπο στον τον οποίο θα κληθούν να διαχειριστούν στα μηνύματα του μηχανισμού, καθώς σε περίπτωση λάθους, υπάρχει το ενδεχόμενο να διαγραφούν κρίσιμα αρχεία για την λειτουργία του ιστοτόπου. Για τον λόγο αυτό άλλωστε δεν παρέχεται η δυνατότητα αυτόματης διαγραφής των ύποπτων αρχείων.

Λόγω της φύσεως του αλγορίθμου μηχανικής μάθησης στον οποίο βασίζεται ο μηχανισμός, απαιτείται να γίνει αρχικά η απαιτούμενη εκπαίδευση για να είναι δυνατή η λειτουργία του με ικανοποιητικά αποτελέσματα. Όμως δεν είναι δυνατή η εύκολη συλλογή κακόβουλων προγραμμάτων με σκοπό την εκπαίδευση του μηχανισμού. Αυτή η αδυναμία έχει εν μέρει ικανοποιηθεί καθώς ο μηχανισμός παρέχει την δυνατότητα να αποθηκεύει τα δεδομένα του, σε μόνιμη βάση sqlite οπότε είναι δυνατή μεταφορά τους και σε άλλα συστήματα, όμως σε περίπτωση καταστροφής της τότε ο μηχανισμός πρακτικά θα τεθεί εκτός λειτουργείας μέχρι την επαρκή επανεκπαίδευση του. Επίσης λόγω της συνεχής βελτίωσης του αλγορίθμου, αν ο χειριστής δεν κάνει σωστή διάκριση μεταξύ του ασφαλούς και κακόβουλου αρχείου, τότε ο μηχανισμός θα εκπαιδευτεί με λανθασμένα κριτήρια οπότε τα επόμενα αποτελέσματα θα παράγονται με εσφαλμένη ταξινόμηση.

Μία επιπλέον αδυναμία του μηχανισμού μηχανικής μάθησης είναι ο μη δυνατός εντοπισμός αρχείων που περιέχουν obfuscated κώδικα με χρήση μη αλφαριθμητικών χαρακτήρων, όπως αυτός που εμφανίζεται στην Εικόνα 9. Αυτό συμβαίνει διότι στα υπό εξέταση αρχεία δεν υπάρχουν συναρτήσεις, οπότε δεν είναι δυνατή η εκτέλεση του αλγορίθμου. Άλλωστε ο λόγος για τον οποίο απέτυχε ο μηχανισμός στην ενότητα 5.2.2 να ταυτοποιήσει ένα αρχείο ως κακόβουλο ήταν αυτός.

Η ανάλυση των αρχείων από τον μηχανισμό μηχανικής μάθησης γίνεται σε πολύ μικρό χρονικό διάστημα. Για παράδειγμα στην Εικόνα 20 βλέπουμε ότι έγινε σάρωση 1501 αρχείων σε μόλις 49 δευτερόλεπτα. Όμως παρατηρήθηκε ότι όταν γίνονταν ανάλυση στα αρχεία που είχαν διαφορετική από την ASCII κωδικοποίηση (πχ αρχεία UDF-8), αυτή γίνονταν σε πολύ μεγάλο χρονικό διάστημα, ενώ τα

περιεχόμενα των αρχείων τους αγνοούνταν από τον μηχανισμό και εμφανίζονταν ως κενά. Αυτό συνέβαινε διότι η Python χρειάζονταν επιπλέον βιβλιοθήκες έτσι ώστε να είναι σε θέση να διαβάσει τέτοιου είδους κωδικοποίηση.

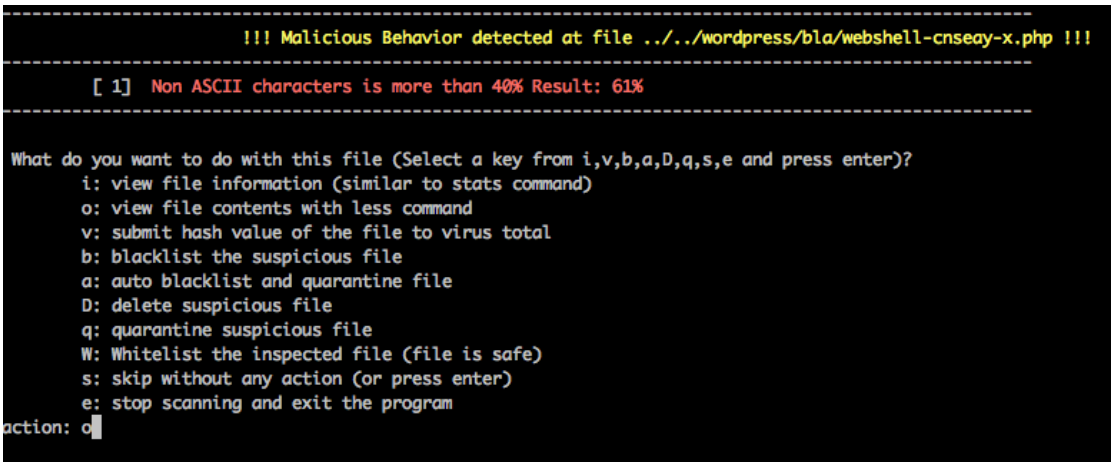
6.2 - Βελτιώσεις

Η υπό ανάπτυξη εφαρμογή εκτός από τον μηχανισμό μηχανικής μάθησης περιέχει επιπλέον άλλους τέσσερις που περιγράφονται αναλυτικά στο κεφάλαιο 2. Η ενεργοποίηση τους σίγουρα θα επιφέρει μεγαλύτερα ποσοστά εντοπισμού κακόβουλου λογισμικού, όμως ταυτόχρονα θα αυξηθεί το ποσοστό του ρυθμού False Positive. Εξάλλου η ταυτόχρονη ενεργοποίηση 5 μηχανισμών θα αυξήσει εκθετικά τον χρόνο ο οποίος απαιτείται για την ανάλυση όλων των αρχείων σε έναν ιστότοπο. Για τον λόγο αυτό δοκιμάστηκαν περισσότερο ευέλικτοι τρόποι έτσι ώστε να αυξήσουμε τα ποσοστά True Positive / False Negative και παράλληλα να μειώσουμε τα ποσοστά False Positive / True Negative.

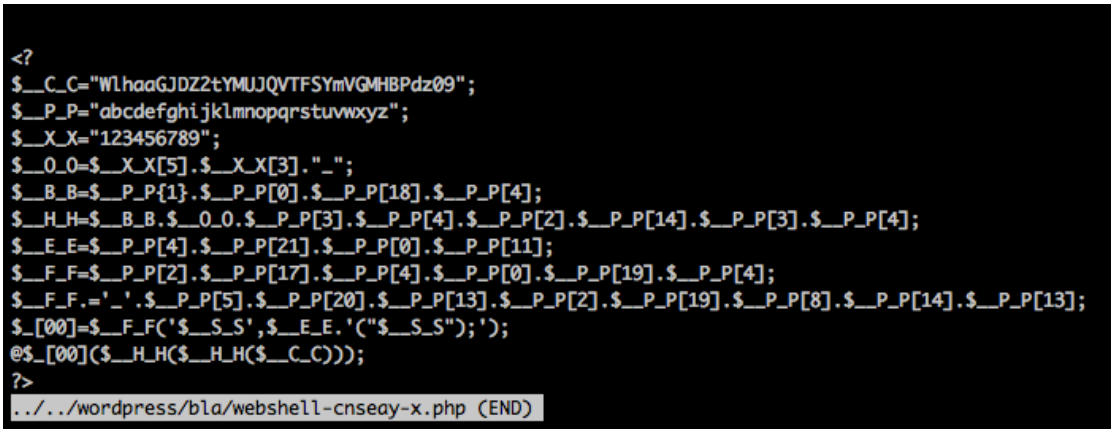
Ως πρώτο βήμα θα πρέπει να ενεργοποιήσουμε τον μηχανισμό «λευκής λίστας» (η μαύρη λίστα είναι εξ ορισμού ενεργή). Με αυτή την ενέργεια, έχουμε κρατήσει ένα στιγμιότυπο όλων των αρχείων του ιστοτόπου, με αποτέλεσμα ο μηχανισμός συγκρίνει την τιμή του αρχείου με αυτήν που βρίσκεται αποθηκευμένη στην βάση δεδομένων του και αν είναι η ίδια τότε σταματάει η ανάλυση του, το θεωρεί ασφαλές, και συνεχίζεται η ανάλυση σε επόμενο αρχείο. Με αυτό τον τρόπο πετυχαίνουμε να μηδενίσουμε τα true negative σφάλματα. Επίσης ο έλεγχος των αρχείων γίνεται σε λιγότερο χρονικό διάστημα καθώς δεν περνάνε από την φάση της ανάλυσης. Βέβαια για να ενεργοποιηθεί η λευκή λίστα θα πρέπει να ήμαστε σίγουροι, ότι ο μηχανισμός δεν περιέχει ήδη κακόβουλο λογισμικό. Άρα θα πρέπει να ενεργοποιηθεί πριν ο ιστότοπος έχει δημοσιευθεί στο διαδίκτυο.

Η αδυναμία του μηχανισμού μηχανικής μάθησης να εντοπίσει κακόβουλο λογισμικό με χρήση μη αλφαριθμητικών χαρακτήρων, χρησιμοποιήθηκε ώστε να βελτιωθεί. Όταν ο μηχανισμός εντοπίσει ένα τέτοιο αρχείο, τότε ενεργοποιείται ο μηχανισμός της στατιστικής ανάλυσης και διενεργείται ερευνάται ο ρυθμός εμφάνισης αλφαριθμητικών χαρακτήρων. Αν ο μηχανισμός αυτός εντοπίσει περισσότερους ειδικούς χαρακτήρες τότε ενημερώνει τον χρήστη για περισσότερες ενέργειες. Με αυτή την μικρή τροποποίηση ο μηχανισμός είναι πλέον σε θέση να εντοπίζει τέτοιου είδους κακόβουλα αρχεία.

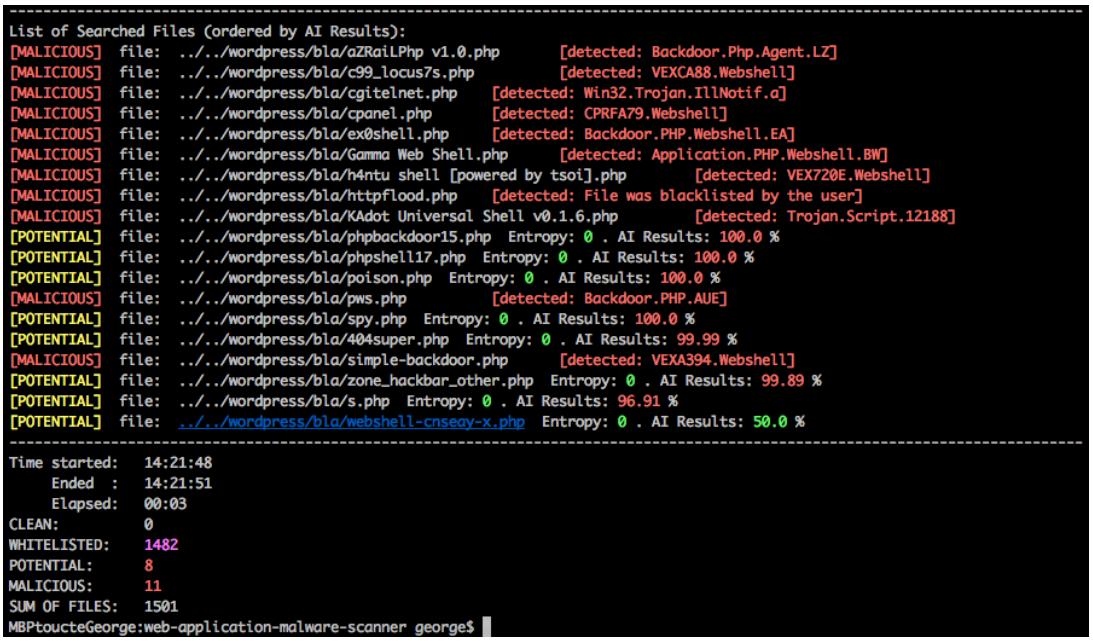
Τέλος στις εικόνες 30-32 εμφανίζονται τα αποτελέσματα του ελέγχου στον ιστότοπο wordpress της 1ης αξιολόγησης του κεφαλαίου 5.2. Παρατηρούμε ότι ο συλλογισμός ήταν σωστός καθώς ο μηχανισμός ενώ αρχικά απέτυχε να εντοπίσει το αρχείο webshell-cnseay-z.php πλέον μετά από την εφαρμογή των παραπάνω παρατηρήσεων **εντόπισε όλα τα κακόβουλα αρχεία χωρίς να παρουσιάσει σφάλμα False Positive, ενώ ο χρόνος εντοπισμού είναι μόλις 3 sec.**



Εικόνα 30 Εντοπισμός κακόβουλου αρχείου με ελάχιστο αλφαριθμητικό περιεχόμενο



Εικόνα 31 Περιεχόμενα του αρχείου webshell-cnseay-x.php



Εικόνα 32 Τελικά Αποτελέσματα του ελέγχου μετά τις αλλαγές που προτάθηκαν.

Η σελίδα αφέθηκε σκοπίμως κενή

Κεφάλαιο 7ο – Επίλογος

Ο στόχος της παρούσης διπλωματικής διατριβής ήταν η υλοποίηση ενός μηχανισμού ο οποίος θα είναι σε θέση να εντοπίζει άγνωστο κακόβουλο λογισμικό. Μετά από αρκετή αναζήτηση στο διαδίκτυο καταλήξαμε στο συμπέρασμα ότι η χρήση μηχανικής μάθησης ήταν επιβεβλημένη για να μπορέσουμε να επιτύχουμε αυτά τα αποτελέσματα. Ο μηχανισμός μηχανικής μάθησης ήταν σε θέση, στην αξιολόγηση που έγινε, να εντοπίζει αρχικά το 95% και στην συνέχεια το 100% του κακόβουλου λογισμικού που εγκαταστάθηκε σε δύο γνωστούς ιστότοπους.

Στην συνέχεια ο μηχανισμός δοκιμάστηκε σε πραγματικές συνθήκες καθώς ένας γνωστός ιστότοπος παραβιάστηκε από αγνώστους και κλήθηκα να βοηθήσω στην εξουδετέρωση της απειλής. Στον ιστότοπο αυτό είχαν εγκατασταθεί 32 διαφορετικά κακόβουλα λογισμικά, κυρίως κατηγορίας κερκόπορτας, αλλά και διάφορα άλλα όπως ανακατεύθυνση σε ιστότοπο με μολυσμένο περιεχόμενο κλπ. Ο μηχανισμός εντόπισε με επιτυχία όλα τα κακόβουλα αρχεία, ενώ χαρακτηριστικό είναι ότι κανένα από αυτά δεν είχε χρησιμοποιηθεί για την εκπαίδευση του υπό ανάπτυξη μηχανισμού ενώ κάποια από αυτά βρίσκονταν ως εμβόλιμα μέσα σε αρχεία του συστήματος.

Η τεχνολογία που χρησιμοποιήθηκε για την κατασκευή του εν λόγω μηχανισμού είναι η γλώσσα προγραμματισμού Python, λόγω της ύπαρξης βιβλιοθηκών μηχανικής μάθησης, του εύκολου προγραμματισμού της και της ευρύτατης χρήσης της σε εξυπηρετητές διαδικτύου. Ο μηχανισμός σχεδιάστηκε έτσι ώστε να είναι σε θέση να λειτουργεί στο σύνολο των γλωσσών σεναρίων, στην διάρκεια των δοκιμών όμως εκπαιδεύτηκε στις γλώσσες PHP και Javascript. Ο μηχανισμός όμως αν και “έξυπνος” δεν αναλαμβάνει πρωτοβουλίες. Αυτό σημαίνει ότι μόλις εντοπίσει ένα ύποπτο αρχείο αναμένει από τον χειριστή να εκτελέσει τις απαιτούμενες ενέργειες, καθώς όπως διαπιστώθηκε αν και έχει εξαιρετικά μικρά ποσοστά εμφάνισης false positive σε σχέση με τους υπολοίπους μηχανισμούς εντούτοις για όσο εμφανίζονται αυτά είναι εξαιρετικά ριψοκίνδυνη η αυτόματη ενέργεια, πχ η διαγραφή ενός αρχείου. Οπότε επαφίεται στην κρίση του έμπειρου χρήστη αν ένα αρχείο είναι τελικά κακόβουλο ή επιτελώντας μόνο την υπόδειξη του.

Η χρήση της μηχανικής μάθησης χρησιμοποιείται ολοένα και περισσότερο για την αντιμετώπιση δύσκολων προβλημάτων, τα οποία παλαιότερα λόγω της χαμηλής υπολογιστικής ισχύος δεν ήταν δυνατό να υλοποιηθούν. Ήδη η μηχανική μάθηση βρίσκει υψηλό ενδιαφέρον σε εφαρμογές όπως, τα ιατρικά διαγνωστικά τεστ, οι ευφυείς συσκευές, η εξόρυξη δεδομένων και προσφάτως οι εφαρμογές αυτόματου οδηγού οχημάτων, ο εντοπισμός απάτης σε τραπεζικά συστήματα κλπ. Η δημιουργία ενός συστήματος anti-malware είναι μόνο η κορυφή του παγόβουνου καθώς οι δυνατότητες της μηχανικής μάθησης μπορούν να εξελιχθούν με τέτοιο τρόπο, που το μόνο πλέον που μπορεί να τις περιορίσει είναι η ανθρώπινη φαντασία και η ηθική.

Η σελίδα αφέθηκε σκοπίμως κενή

Βιβλιογραφία – Παραπομπές

[1]. https://el.wikipedia.org/wiki/Μηχανική_μάθηση (Ημ. Επίσκεψης 2/10/2016)

[2]. http://www.mcafee.com/us/resources/reports/rp-quarterly-threat-q1-2014.pdf (Ημ. Επίσκεψης 4/10/2016)

[3]. Babak Bashari Rad, Maslin Masrom, Suhaimi Ibrahim3, *Evolution of Computer Virus Concealment and Anti-Virus Techniques: A Short Survey*,2011 (p.2-3)

[4]. Babak Bashari Rad, Maslin Masrom, Suhaimi Ibrahim3, *Evolution of Computer Virus Concealment and Anti-Virus Techniques: A Short Survey*,2011 (p.4)

[5]. Vasil Rousev , *"An Evaluation of forensic similarity hashes"* ,2011, (p.3-5) http://roussev.net/pubs/2011-DFRWS--sdhash-vs-ssdeep.pdf (Ημ. Επίσκεψης 1/11/2016)

[6]. https://blog.sucuri.net/2014/02/php-backdoors-hidden-with-clever-use-of-extract-function.html (Ημ. Επίσκεψης, 14/11/16)

[7]. Babak Bashari Rad, Maslin Masrom, Suhaimi Ibrahim3, *Evolution of Computer Virus Concealment and Anti-Virus Techniques: A Short Survey*,2011 (p.4-5)

[8]. https://blog.sucuri.net/2013/09/ask-sucuri-non-alphanumeric-backdoors.html (Ημ. Επίσκεψης 25/11/2016)

[9]. https://en.wikipedia.org/wiki/VirusTotal (Ημ. Επίσκεψης 30/11/2016)

[10]. http://fr.pastebin.ca/2311702 (Ημ. Επίσκεψης 12/12/2016)

[11]. https://en.wikipedia.org/wiki/Naive_Bayes_classifier (Ημ. Επίσκεψης 28/12/2016)

[12]. Tianhao Sun, *"Spam Filtering Based on Naïve Bayes Classification", 2009 (p.5-9) http://www.cs.ubbcluj.ro/~gabis/DocDiplome/Bayesian/000539771r.pdf (Ημ. Επίσκεψης 5/1/17)*

[13]. http://codebox.org.uk/pages/naive-bayesian-classifier-in-python (Ημ. Επίσκεψης 16/10/2016)

[14]. https://github.com/nikicat/web-malware-collection (Ημ επίσκεψης 20/1/2017)

[15]. https://tennc.github.io/webshell/ (Ημ επίσκεψης 20/1/2017)

[16]. https://www.joomla.org (Ημ. Επίσκεψης 20/1/2017)

[17]. https://www.wordpress.org (Ημ. Επίσκεψης 20/1/2017)

[18]. https://github.com/Neohapsis/NeoPI (Ημ. Επίσκεψης 22/1/2017)

[19]. http://www.shelldetector.com (Ημ. Επίσκεψης 22/1/2017)

[20]. https://www.clamav.net (Ημ. Επίσκεψης 22/1/2017)

[21]. AntiMalware Testing Standards Organization, Fundamental Principles of Testing, http://www.amtso.org/download/amtso-fundamental-principles-of-testing/ (Ημ. Επίσκεψης 25/1/2017)

[22]. Confusion Matrix, https://en.wikipedia.org/wiki/Confusion_matrix (Ημ. Επίσκεψης 27/1/2017)

[23]. Tom Fawcett, «*An introduction to ROC analysis*», 2005 (p.2)

[24]. Jesse Davis, Mark Goadrich, "*The Relationship Between Precition-Recall and ROC Curves*» , http://pages.cs.wisc.edu/~jdavis/davisgoadrichcamera2.pdf (Ημ. Επίσκεψης 1/2/2017)

[25]. Precision and Recall, http://mlwiki.org/index.php/Precision_and_Recall (Ημ. Προσπέλασης 3/2/2017)

[26]. F1 Score, https://en.wikipedia.org/wiki/F1_score (Ημ. Προσπέλασης 6/2/2017)

[27]. Roc, Analysis, http://mlwiki.org/index.php/ROC_Analysis (Ημ. Προσπέλασης 6/2/2017)

[28]. Tom Fawcett, «*An introduction to ROC analysis*», 2005 (p.3-4)

[29]. Malware Hidden inside JPG EXIF Headers, https://blog.sucuri.net/2013/07/malware-hidden-inside-jpg-exif-headers.html (Ημ. Προσπέλασης 14/2/2017)