Πανεπιστήμιο Πειραιώς

ΤΜΗΜΑ ΨΗΦΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

[Π.Μ.Σ. ΤΔΑΨΣ] - ΚΑΤ. ΑΣΦΑΛΕΙΑΣ ΨΗΦΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

Σχεδιασμός & Υλοποίηση Επιβλαβούς Λογισμικού με Ενσωμάτωση Anti-Analysis Μηχανισμών

Σαρρής Αντώνης
ΜΤΕ1534

Επιβλέπων Καθηγητής:
Δρ. Χρ. Ντάντογιάν

Αθήνα, Μάρτιος 2017

Στην Μαρία

Περίληψη

Η παρούσα εργασία παρουσιάζει την ανάπτυξη ενός επιβλαβούς λογισμικού σε περιβάλλον του Λειτουργικού Συστήματος των Windows, με ενσωμάτωση anti-Analysis Μηχανισμών. Συγκεκριμένα, σχεδιάστηκε και υλοποιήθηκε ένα επιβλαβές λογισμικό τύπου Ransomware, το οποίο μετά την επιτυχή εκτέλεσή του κρυπτογραφεί τα αρχεία του χρήστη που βρίσκονται στον φάκελο-στόχο, με τον Αλγόριθμό κρυπτογράφησης AES (Advanced Encryption Standard) συμμετρικού κλειδιού και την αποστολή του κλειδιού σε έναν απομακρυσμένο εξυπηρετητή. Στην συνέχεια ερευνήθηκαν διάφοροι anti-Analysis μηχανισμοί, όπως η προστασία από Debuggers και Disassemblers, αποφυγή Virtual Machines, Sandboxes & μηχανές AVs. Επίσης, αναπτύχθηκαν διάφοροι επιπρόσθετοι μηχανισμοί προστασίας σε όλο το εύρος του anti-Analysis. Οι παραπάνω μηχανισμοί ενσωματώθηκαν στο επιβλαβές λογισμικό καθιστώντας το ανθεκτικότερο σε παρόμοιους ελέγχους. Με τον τρόπο αυτό, το επιβλαβές λογισμικό αποκτά μια ασπίδα προστασίας, τόσο σε ενδεχόμενη στατική όσο σε και δυναμική ανάλυση. Η διενέργεια μιας σειράς δοκιμών / πειραμάτων -στο πλαίσιο της ανάλυσης επιβλαβούς λογισμικού- επικυρώνουν την αποτελεσματικότητα της τελικής υλοποίησης.

Λέξεις Κλειδιά: Ασφάλεια, Επιβλαβές Λογισμικό, Μηχανισμοί Προστασίας Ανάλυσης, Αντικά

Abstract

This master thesis focuses on the problem of malware software development through the utilization of anti-Analysis mechanisms in the context of the Windows operating system. In particular, this study emphasizes on the designing and implementation processes of Ransomware-type malware software. This type of malware software, after being successfully executed, encrypts all user files in a target-folder by employing the AES (Advanced Encryption Standard) symmetric key encryption algorithm and subsequently sends the encryption key to a remote server. Moreover, extended research has been conducted on a wide range of anti-Analysis mechanisms such as protection against Debuggers and Disassemblers, or the avoiding of Virtual Machines, Sandboxes & AV machines. Furthermore, several additional protection mechanisms have been developed in the broad spectrum of anti-Analysis. The aforementioned protection mechanisms have been incorporated within the implemented malware software rendering it resilient to similar static and dynamic analysis tests. The overall efficiency of the proposed implementation has been justified through a series of experimental tests that have been conducted within the context of malware software analysis.

Keywords: Security, Malware, Anti-Analysis, Antivirus, Packing

«... Μη μετανιώσεις για τίποτα. Ποτέ.
Μόνο για την αδράνεια.»

~Κωνσταντίνος Καβάφης

Πίνακας περιεχομένων

Γραφήματα

Σχήματα

Εικόνες

Πίνακες

1 *Εισαγωγή*

1.1 Αντικείμενο Εργασίας

Από την αρχή της δεκαετίας του ’70 που εμφανίστηκε το πρώτο επιβλαβές λογισμικό μέχρι και σήμερα, υπήρχε πάντα μια ανάγκη για την ανάπτυξη λογισμικών που να είναι σε θέση να εμποδίζουν την δράση του. Ως παρατηρητής διαπιστώνει κανείς πως αυτή η «μάχη» δεν μπορεί να τελειώσει ποτέ. Αυτό συμβαίνει γιατί όσο θα βελτιώνεται ένας μηχανισμός αναπτύσσοντας καινούργια εργαλεία αντιμετώπισης απέναντι σε ένας επιβλαβές λογισμικό, τόσο με την σειρά του το επιβλαβές λογισμικό θα γίνεται καλύτερο για να τον υπερνικήσει κ.ο.κ.

Στόχος και αντικείμενο της παρούσας μεταπτυχιακής διατριβής είναι η κατασκευή ενός επιβλαβούς λογισμικού σε περιβάλλον του Λειτουργικού Συστήματος των Windows, με χρήση anti-Analysis Μηχανισμών.

Αναλυτικότερα, σχεδιάστηκε και υλοποιήθηκε ένα επιβλαβές λογισμικό το οποίο ανήκει σε μια σχετικά νέα κατηγορία με όνομα Ransomware. Η κατηγορία αυτή μοιάζει με Scareware αλλά αντί να προσπαθεί να ξεγελάσει το θύμα για να αγοράσει πλαστά λογισμικά προστασίας από ιούς, κρυπτογραφεί τα αρχεία του, ζητώντας λύτρα για την αποκρυπτογράφηση. Το επιβλαβές λογισμικό κρυπτογραφεί τα αρχεία του χρήστη που βρίσκονται στον φάκελο-στόχο, με τον Αλγόριθμό κρυπτογράφησης AES (Advanced Encryption Standard) συμμετρικού κλειδιού μήκους 256 bits με λειτουργία CBC (Cipher Block Chaining). Στην συνέχεια αποστέλλει τον κωδικό κρυπτογραφημένο σε έναν απομακρυσμένο εξυπηρετητή.

Στην συνέχεια ερευνήθηκαν και ενσωματώθηκαν μια σειρά από μηχανισμούς anti-Analysis. Μεταξύ των μηχανισμών ξεχωρίζουν αυτοί του anti-Debugging και anti-Disassembly, αλλά και μηχανισμοί αποφυγής Virtual Machines, Sandboxes &

μηχανών AVs. Η υλοποίηση έγινε με την γλώσσα προγραμματισμού C++/CLI που αποδεικνύεται πιο ανθεκτική στο τομέα της anti-Analysis συνδυάζοντας managed και unmanaged κώδικα C++.

Η ενσωμάτωση των παραπάνω μηχανισμών καθιστούν το επιβλαβές λογισμικό ανθεκτικότερο, αυξάνοντας τις πιθανότητες μιας «αθόρυβης» εκτέλεσης του. Στο πλαίσιο της στατικής όσο και δυναμικής ανάλυσης επιβλαβούς λογισμικού, η διενέργεια μιας σειράς δοκιμών και πειραμάτων που πραγματοποιήθηκαν στην παρούσα εργασία επικυρώνουν την αποτελεσματικότητα της τελικής υλοποίησης.

Το αντικείμενο της εργασίας ενώ αρχικά δίνει την εντύπωση ότι συμβάλλει στην πλευρά εκείνων «με τα μαύρα καπέλα», η διαφανής διαδικασία σχεδιασμού και υλοποίησης του επιβλαβούς λογισμικού αλλά και των πρόσθετων μηχανισμών προστασίας αυτού, αποδεδειγμένα συμβάλλει με αποτελεσματικό τρόπο στην σωστή αντιμετώπιση κακόβουλων ενεργειών, αφού βοηθά την Ανάλυσή τους.

1.2 Οργάνωση Μεταπτυχιακής Διατριβής

Η παρούσα εργασία αποτελείται από δέκα (10) κεφάλαια. Συνοπτικά αναφέρονται τα περιεχόμενα σε κάθε ένα από αυτά:

1ο κεφάλαιο, αποτελεί την εισαγωγή της εργασίας. Αναφέρεται ο σκοπός και οι στόχοι της παρούσας εργασίας.

2ο κεφάλαιο, παρουσιάζεται μια οργανωμένη καταγραφή της ερευνητικής περιοχής που εξετάζεται στην παρούσα εργασία καθώς και κρίσιμων τεχνικών ζητημάτων που εμπλέκονται σε αυτή,

3ο κεφάλαιο, αναφέρεται η έννοια του επιβλαβούς λογισμικού, οι τυπικές κατηγορίες που μπορεί να χωριστούν τα είδη του, καθώς και μια ιστορική αναδρομή για την πορεία αυτού του «ιδιαίτερου» λογισμικού μέχρι σήμερα,

4ο κεφάλαιο, παρουσιάζονται τα κυριότερα συστήματα αντιμετώπισης επιβλαβών λογισμικών καθώς και οι σημαντικότεροι τρόποι λειτουργίας τους,

5ο κεφάλαιο, γίνεται αναφορά στις τεχνικές ανάλυσης του επιβλαβούς λογισμικού και τα επιμέρους είδη του,

6ο κεφάλαιο, αναλύονται οι τεχνικές anti-Analysis, όπου μέρος τους θα προστεθούν στην υλοποίηση του επιβλαβούς λογισμικού της παρούσας εργασίας,

7ο κεφάλαιο, περιγράφεται ο Σχεδιασμός και Υλοποίηση του επιβλαβούς λογισμικού τύπου Ransomware που αναπτύχθηκε στην παρούσα εργασία, αναλύοντας το σύνολο του κώδικα με επιπλέον τα στάδια που ακολουθήθηκαν ώστε να αποφεύγει προγράμματα Ανάλυσης,

8ο κεφάλαιο, εκτελούνται δοκιμές για την ανάλυση του επιβλαβούς λογισμικού που υλοποιήθηκε στην παρούσα εργασία,

9ο κεφάλαιο, αποτελεί τον επίλογο της εργασίας, ενώ παράλληλα αναφέρονται τα συμπεράσματα καθώς και οι μελλοντικές επεκτάσεις,

10ο κεφάλαιο, παρατίθεται η βιβλιογραφία καθώς και οι αναφορές της παρούσας εργασίας.

2 *Βιβλιογραφική Επισκόπηση*

Με την ανίχνευσή του κακόβουλου λογισμικού ransomware έχει ασχοληθεί ο Moore C. [1] όπου διερεύνησε μεθόδους για την υλοποίηση ενός honeypot για την ανίχνευση ransomware δραστηριότητας. Οι Scaife N., Carter H., Traynor P., Butle K.R.B., [2] ανέπτυξαν ένα σύστημα ανίχνευσης έγκαιρης προειδοποίησης CryptoDrop που ειδοποιεί τον χρήστη κατά την εκτέλεση ύποπτης δραστηριότητας στα αρχεία του. Χρησιμοποιώντας ένα σύνολο δεικτών συμπεριφοράς, το CryptoDrop μπορεί να σταματήσει μια διαδικασία που φαίνεται να παραβιάζει ένα μεγάλο αριθμό δεδομένων του χρήστη. Επιπλέον, συνδυάζοντας ένα σύνολο κοινών παραμέτρων των ransomware, γίνεται παραμετροποίηση για την ταχεία ανίχνευση και με μειωμένα ψευδώς θετικά συμπεράσματα.

Όπως μπορεί εύκολα να διαπιστωθεί όλες οι σχετικές με τα ransomware εργασίες προσπαθούν να εντοπίσουν και να αποτρέψουν την κρυπτογράφηση των αρχείων την ώρα της εκτέλεσής και δεν ασχολούνται με την ανάλυσή του. Μια προσπάθεια διερεύνησης τέτοιου είδους απειλών βασιζόμενη στο Reverse-Engineering πραγματοποιήθηκε από τον Gazet A. [3]

Όπως και στα άλλα είδη κακόβουλου λογισμικού μπορούμε να χρησιμοποιήσουμε τις κλασσικές μεθόδους ανάλυσης. Υπάρχουν πολλές εργασίες και βιβλία που προτείνουν μεθόδους ανάλυσης κακόβουλου λογισμικού. Η μεθοδολογία που προτείνει ο Skoudis [4] είναι ένα γραμμικό μοντέλο όπου όταν ολοκληρώνεται το ένα βήμα περνάμε στο άλλο, όμως σε αυτή την τεχνική υπάρχει ο κίνδυνος το κακόβουλο λογισμικό να αποκρύψει τις λειτουργίες του, όταν γίνει αντιληπτή η ανάλυσή του στα αρχικά βήματα.

Ο Zeltser [5] δημιούργησε έναν οδηγό για την ανάλυση κακόβουλου λογισμικού αρχίζοντας με την δημιουργία ενός ασφαλούς εργαστηριακού περιβάλλοντος και χρησιμοποιώντας ελεύθερα διαθέσιμα εργαλεία λογισμικού. Η γενική μεθοδολογία που παρουσιάζεται είναι η εξής:

- Εκτέλεση του κακόβουλο λογισμικό σε απομονωμένο εργαστήριο
- Παρακολούθηση των αλληλεπιδράσεων μεταξύ του συστήματος και του δικτύου

- Κατανόηση του πηγαίου κώδικα του προγράμματος
- Επανάληψη της διαδικασίας μέχρι να συγκεντρωθούν οι απαραίτητες πληροφορίες.

Μια πιο γενικευμένη προσέγγιση στην ανάλυση κακόβουλου λογισμικού μαζί με τα απαραίτητα εργαλεία ανάλυσης παρουσιάζονται από τους Sikorski M. και Honig A. [6]. Στο βιβλίο τους γίνεται παρουσίαση της σειράς που ακολουθείται για την ανάλυση η οποία χωρίζεται σε βασική στατική, βασική δυναμική, προχωρημένη στατική και προχωρημένη δυναμική. Κάθε βήμα είναι ανεξάρτητα από το προηγούμενο αλλά η συνολική εικόνα είναι αυτή που θα βοηθήσει τον αναλυτή στην κατανόηση του ιού.

Μια άλλη σημαντική παράμετρος που πρέπει να ληφθεί υπόψη κατά την ανάλυση ενός κακόβουλου λογισμικού είναι ότι ο προγραμματιστής του θα έχει προβλέψει και μεθόδους για την αποφυγή της ανάλυσης. Στον τομέα της anti-Analysis υπάρχουν πολλές μελέτες και αξιόλογη βιβλιογραφία. Στις μελέτες των τεχνικών αυτών παρέχονται κομμάτια κώδικα με τις επεξηγήσεις τους όπου μπορούν να ενσωματωθούν σε ήδη υπάρχοντα κώδικα. Ακόμα εντοπίστηκαν και πολλές μελέτες όπου συγκρίνονται έτοιμα εργαλεία. Για τις μεθόδους anti-Analysis υπάρχει πολύ μεγάλη βιβλιογραφία όπου κάθε μια εστιάζει σε συγκεκριμένη περιοχή. Οι κυριότερες τεχνικές anti-Analysis είναι:

- **Anti-Virtual Machine**

  Οι μελέτες των Garfinkel et al. [7] και Raffetseder et al. [8] παρέχουν πλούσια έρευνα σχετικά με τις υφιστάμενες τεχνικές για την ανίχνευση των εικονικών μηχανών και των εξομοιωτών. Σε αυτές τις μελέτες, οι συγγραφείς έχουν κάνει ποιοτική σύγκριση των τεχνικών. Τα ποσοτικά αποτελέσματα των τεχνικών αυτών μελετήθηκαν και παρουσιάστηκαν από τον O Yoshihiro Oyama. [9]

- **Anti-Debugging**

  O Ferrie P [10] παρουσιάζει ένα μεγάλο αριθμό από τεχνικές anti-debugging σε διαφορετικά επίπεδα (hardware, process-level, system-level, user-interface). Οι Gao S. et al. [11] παρέχουν μια επισκόπηση του μηχανισμού εντοπισμού σφαλμάτων στα Windows και μια ταξινόμηση των πιο διαδεδομένων εργαλείων.

- **Anti-Virtualization**

  Οι Chen X. et al, [12]ανέπτυξαν μια ταξινόμηση των τεχνικών Anti-Analysis εστιάζοντας σε Anti -virtualization και συμπεριφορά Anti-Debugging.

- **Anti-Disassembly**

  Οι Linn C. και Debray S.K. [13] παρουσίασαν έναν μηχανισμό για να εμποδίσουν τη διαδικασία disassembly. Οι δοκιμές σε δυο γνωστούς αλγόριθμους disassembly έδειξαν ότι απέτυχαν να αποσυναρμολογήσετε σωστά το 65% των εντολών και το 85% των συναρτήσεων.

- **Packers**

  Οι Li A, Zhang, Y., Zhang, J., Zhu, G. [14] προτείνουν έναν packer κρυπτογράφησης που ενισχύει την εμπιστευτικότητα στο προστατευμένο αρχείο PE με κωδικό ασφαλείας. Ο συσκευαστής επίσης ενσωματώνει τεχνικές anti-debugging, anti-dumping, and anti-tracking για να αποτρέψει τα αρχεία PE από το reverse engineering. Οι Liță et al, [15] παρουσίασαν τα αποτελέσματα από την εξέταση γνωστών Packers.

3 *Επιβλαβές Λογισμικό*

Σε αυτό το κεφάλαιο αναφέρεται η έννοια του επιβλαβούς λογισμικού, οι τυπικές κατηγορίες που μπορεί να χωριστεί, τα είδη του καθώς και μια ιστορική αναδρομή για την πορεία αυτού του ιδιαίτερου λογισμικού μέχρι σήμερα.

3.1 Έννοια Επιβλαβούς Λογισμικού

Το επιβλαβές λογισμικό ή malicious software (συντομογραφία malware) είναι ένα από τα μεγαλύτερα προβλήματα που αντιμετωπίζει η επιστήμη της Πληροφορικής και ειδικότερα ο τομέας της ασφάλειας των Πληροφοριακών Συστημάτων. [16] Η ονομασία του προέρχεται από την πρόθεση του προγραμματιστή που το δημιούργησε, που σε αυτή την περίπτωση δεν είναι η προσθήκη μιας επιπλέον λειτουργικότητας σε ένα Πληροφοριακό Σύστημα αλλά αντίθετα η βλάβη αυτού. Το επιβλαβές λογισμικό έχει σχεδιαστεί προκειμένου να προκαλεί διακοπή ή άρνηση κάποιας υπηρεσίας, να συγκεντρώνει (παράνομα) πληροφορίες και ιδιωτικά/προσωπικά στοιχεία χρηστών, να επιτρέπει την απομακρυσμένη διαχείριση των μολυσμένων μηχανημάτων κ.λπ.

Το εν λόγω πρόγραμμα μπορεί να ξεκινήσει είτε αυτοβούλως και αυτόνομα - χωρίς τις ενέργειες κάποιου - είτε να ενεργοποιηθεί από έναν τρίτο κάτω από οποιεσδήποτε προϋποθέσεις που του έχουν τεθεί. Επίσης μπορεί να αναπαραχθεί αυτόματα ή με ανθρώπινη παρέμβαση καθιστώντας το ιομορφικό ή μη, λογισμικό. [17]

3.2 Ιστορική Αναδρομή

Το 1971, εμφανίστηκε το πρόγραμμα «Creeper» που για πολλούς θεωρείται το πρώτο επιβλαβές λογισμικό. Το Creeper μεταδιδόταν μέσω του δικτύου ARPANET, το γνωστό δίκτυο του αμερικανικού στρατού που υπήρξε άλλωστε και ο προπομπός του internet. Το Creeper είχε γραφτεί για το τότε δημοφιλές λογισμικό σύστημα "Tenex" και μπορούσε να αποκτήσει πρόσβαση μέσω ενός modem και να αντιγράψει τον εαυτό του στο απομακρυσμένο σύστημα. Το αποτέλεσμα ή τελική ενέργεια του Creeper ήταν να εμφανίζει στην οθόνη του χρήστη η φράση: «I'm the Creeper, catch

me, if you can». Δηλαδή «Είμαι ο Creeper, πιάσε με, αν μπορείς». Στην ουσία ο χρήστης φαίνεται ότι δεν είχε κάποια σοβαρή επίπτωση από ένα απλό μήνυμα στην οθόνη αλλά αντίθετα για την ασφάλεια των πληροφοριακών συστημάτων αποδείχθηκε ότι ένας ικανός προγραμματιστής μπορεί να φτιάξει ένα λογισμικό που μπορεί να «μεταφερθεί» στο δίκτυο και να πετύχει μία μη προδιαγεγραμμένη ενέργεια. Λίγο αργότερα, εμφανίστηκε το πρόγραμμα Reaper και δημιουργήθηκε για να διαγράψει το Creeper. Το Reaper ήταν ένας ιός που εξαπλωνόταν σε δικτυωμένα μηχανήματα και εάν εντόπιζε τον ιό Creeper τον διέγραφε.

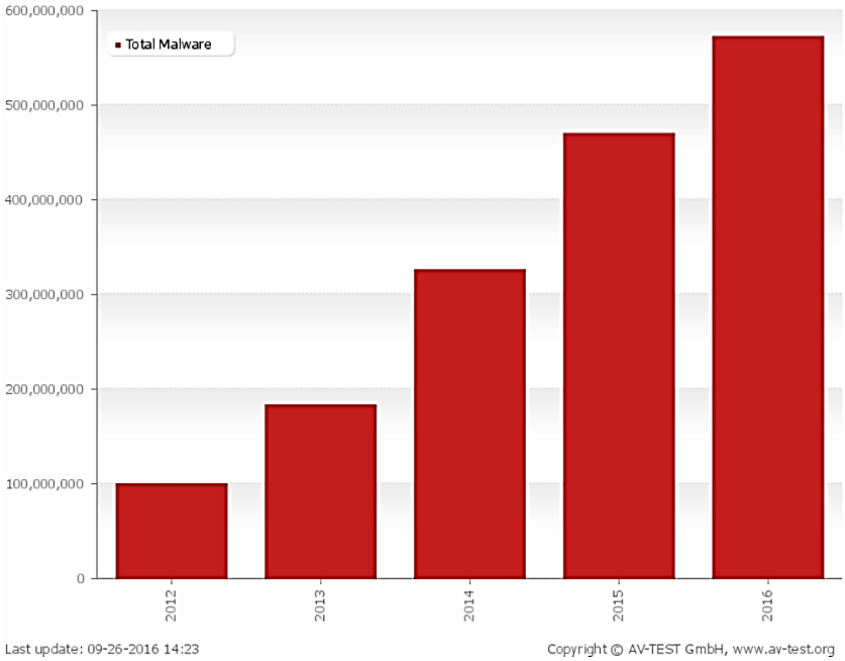
Το πρώτο Trojan εμφανίστηκε το 1975 μέσα από το παιχνίδι "Pervading Animal" που είχε γραφτεί για το λειτουργικό σύστημα Univac 1108. Το παιχνίδι προσπαθούσε κάνοντας ερωτήσεις στον χρήστη να μαντέψει πιο ζώο είχε σκεφτεί ο χρήστης. Το παιχνίδι ήταν εξοπλισμένο με μια συνάρτηση αυτό-διόρθωσης όπου εάν το πρόγραμμα δεν μάντευε σωστά το ζώο θα ανανεωνόταν και θα εισέρχονταν νέες ερωτήσεις. Η νέα εκσυγχρονισμένη έκδοση αντικατέστησε την παλαιά έκδοση, αλλά εκτός από αυτό, η ίδια αντιγράφονταν και σε άλλους καταλόγους του υπολογιστή. Μετά από λίγο καιρό όλοι οι φάκελοι του υπολογιστή περιείχαν αντίγραφο του παιχνιδιού και αυτό καταλάμβανε σημαντικό χώρο στον δίσκο. Έγιναν προσπάθειες να δημιουργηθεί πρόγραμμα που να διαγράφει το παιχνίδι (όπως και με τον πρώτο ιό) αλλά η οριστική λύση δόθηκε με την παρουσίαση του νέου λειτουργικού Exec 8.

Καθώς οι υπολογιστές γίνονταν πιο δημοφιλείς, όλο και περισσότερα άτομα άρχισαν να γράφουν τα δικά τους προγράμματα. Με την πρόοδο στον τομέα των τηλεπικοινωνιών μπορούσαν να παρέχονται κανάλια για την ανταλλαγή προγραμμάτων μέσω διακομιστών ανοικτής πρόσβασης, όπως το BBS (Bulletin Board System). Τελικά οι BBS servers εξελίχθηκαν σε μια παγκόσμια τράπεζα δεδομένων και ήταν διαθέσιμοι σε όλες τις ανεπτυγμένες χώρες. Τότε αρχίσαν να εμφανίζονται μαζικά επιβλαβή λογισμικά που δεν μπορούσαν μεν να αναπαραχθούν ή να διαδοθούν, αλλά έκαναν ζημιά στα συστήματα κάθε φορά που κάποιος τα κατέβαζε και τα εγκαθιστούσε. Το Δεκέμβριο του 1987, η πρώτη μεγάλη επιδημία σε τοπικό δίκτυο συνέβη με το σκουλήκι "Christmas Tree'. Το σκουλήκι ξέσπασε στο δίκτυο Bitnet στις 9 Δεκεμβρίου μέσω της πύλης ενός Ευρωπαϊκού Ακαδημαϊκού Ερευνητικού Δικτύου (EARN) και στη συνέχεια μέσω του Vnet της IBM. Μέσα σε τέσσερις ημέρες ο ιός είχε πλημμυρίσει το δίκτυο. Μετά τη φόρτωση, ο ιός εμφάνιζε ένα χριστουγεννιάτικο δέντρο στην οθόνη και έστελνε αντίγραφα του εαυτού του σε

όλους τους χρήστες του δικτύου των οποίων οι διευθύνσεις υπήρχαν στα NAMES και NETLOG αρχεία του συστήματος.

Από την δεκαετία του ’80 και μετά καταγράφεται ένας μεγάλος αριθμός από επιβλαβή λογισμικά και αυτό είχε ως συνέπεια οι χρήστες να αρχίσουν να παίρνουν πιο σοβαρά την ασφάλεια και να μάθουν πως να προστατεύουν τους εαυτούς τους από ιούς. Αρχικά οι πιο προσεκτικοί χρήστες έμαθαν να παρακολουθούν το μέγεθος του αρχείου command.com και γνώριζαν ότι μια αύξηση του μεγέθους του αρχείου ήταν το πρώτο σημάδι μιας πιθανής μόλυνσης. Κοντά στο 1988 άρχισαν να κυκλοφορούν στη αγορά και τα πρώτα Antivirus.

Κάποια από τα πιο σημαντικά επιβλαβή λογισμικά που προκάλεσαν ζημιές πολλών εκατομμυρίων είναι τα “ILOVEYOU”, “Stuxnet”, “Code Red”, “Melissa”, “Zbot/Zeus”, “Flame”, “CryptoLocker” κ.α.. Όπως προκύπτει και από την σύντομη ιστορική αναδρομή τα αρχικά επιβλαβή λογισμικά είχαν ως στόχο να κάνουν “πλάκα” ή να κάνουν με την προφανέστατη εκτέλεσή τους τον δημιουργό τους να γίνει “γνωστός”. Σήμερα όμως η δημιουργία ενός επιβλαβούς λογισμικού μπορεί να χρησιμοποιηθεί για την κλοπή χρημάτων, δεδομένων ακόμα και ως κρυφό επιθετικό όπλο με γεωπολιτικά κίνητρα.

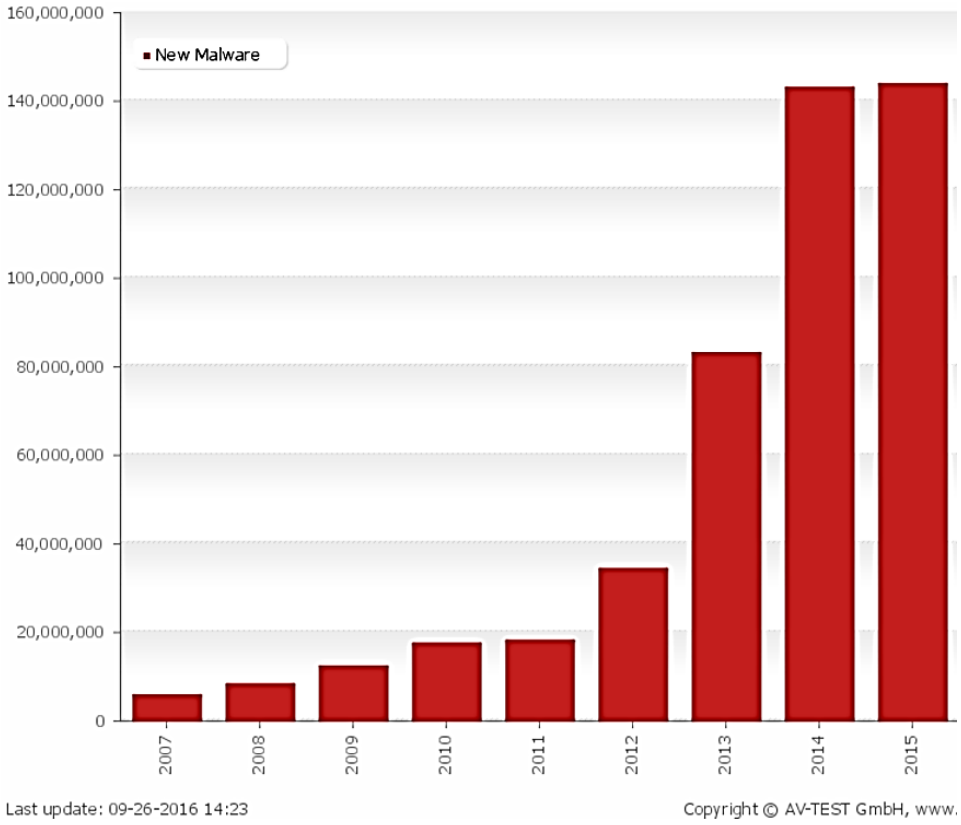


Γράφημα 1: Total Malwares

Τα συμπτώματα που μπορεί να εμφανίσει ένας μολυσμένος υπολογιστής είναι καθυστέρηση στη λειτουργία, εμφάνιση και εκτέλεση άγνωστων αρχείων, αλλαγές

στις ρυθμίσεις του συστήματος ή σε ήδη εγκατεστημένα προγράμματα, ανεπιθύμητη διαδικτυακή δραστηριότητα και εμφάνιση παραθύρων ή διαφημίσεων.

Συνεπώς, για να αναλογιστούμε το μέγεθος του προβλήματος που λέγεται «επιβλαβές λογισμικό» αρκεί να υπολογίσουμε την διαφορά τους ενός (1) malware που εμφανίστηκε το 1971 με τα 600 εκατομμύρια που υπολογίζεται ότι πλησιάζουμε σήμερα. Το Ινστιτούτο AV-TEST καταγράφει χαρακτηριστικά τα γραφήματα 1 & 2. [18]

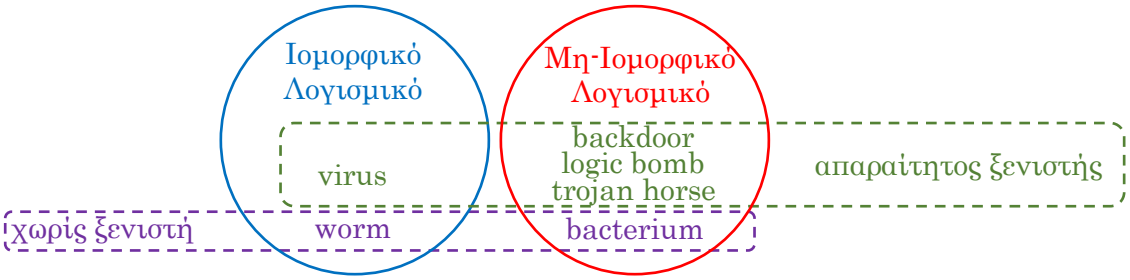


Γράφημα 2: New Malwares

3.3 Κατηγοριοποίηση Επιβλαβούς Λογισμικού

Υπάρχουν πολλοί διαφορετικοί τρόποι να κατηγοριοποιηθεί ένα επιβλαβές λογισμικό. Η αρχική κατηγοριοποίηση είχε γίνει βάση της «ανάγκης» εκτέλεσής του από ένα δεύτερο πρόγραμμα ή μη. Σε αυτή την περίπτωση έχουμε δύο βασικές κατηγορίες επιβλαβούς λογισμικού, σε εκείνα που χρειάζονται ξενιστή πρόγραμμα και σε εκείνα που δεν χρειάζονται και λειτουργούν αυτόματα. Στην πρώτη περίπτωση –απαιτείται ξενιστής- ανήκει ο ηλεκτρονικός ιός «virus», η λογική βόμβα «logic bomb», η κερκόπορτα «backdoor», ο δούρειος ίππος «Trojan horse» και άλλα. Στην περίπτωση μη απαίτησης ξενιστή ο αναπαραγωγός «worm» και το βακτήριο «bacterium» είναι αντιπροσωπευτικά.

Μια επιπλέον κατηγορία είναι η έννοια του ιομορφικού λογισμικού και του μη ιομορφικού λογισμικού. Στο ιομορφικό λογισμικό ανήκει το επιβλαβές πρόγραμμα που μπορεί και αναπαράγεται από μόνο του όπως τα virus & worm. Αντίθετα στο μη ιομορφικό λογισμικό ανήκει το επιβλαβές πρόγραμμα που δεν αναπαράγεται μόνο του, παρά μόνο με ανθρώπινη παρέμβαση, όπως τα backdoor, logic bomb, trojan horse. Σχηματικά φαίνονται, ταυτόχρονα και οι δυο αυτές κατηγοριοποιήσεις (Σχήμα 1)



Σχήμα 1: Κατηγοριοποίηση Malware

Ακολουθούν αναλυτικά οι βασικοί τύποι των επιβλαβών λογισμικών, όπου για λόγους απλότητας θα αναφέρονται με την ονομασία τους στα Αγγλικά.

3.3.1 Virus

Το virus είναι η πρώτη έννοια που αποδόθηκε για το επιβλαβές λογισμικό και περιγράφει στην γενική περίπτωση εκείνο που μπορεί να αντιγράψει τον εαυτό του και στην συνέχεια να μολύνει επιπλέον πληροφοριακά συστήματα. Το άνοιγμα ενός εκτελέσιμου προγράμματος από τον χρήστη μπορεί εν δυνάμει να τον ενεργοποιήσει. Η μετάδοση του μπορεί να επιτευχθεί μετά από χρήση ενός usb stick ή ενός εξωτερικού σκληρού δίσκου που χρησιμοποιούνται για την μεταφορά αρχείων. Συνήθως, χρησιμοποιείται για την διαγραφή κάποιων δεδομένων, που ανάλογα πόσο σημαντικά είναι για το πληροφοριακό σύστημα μπορεί να οδηγήσουν και στην κατάρρευση αυτού.

3.3.2 Worm

Το worm έχει αρκετά κοινά χαρακτηριστικά με το virus, με επιπλέον την δυνατότητα να μεταδίδεται μέσω δικτύων ή email. Επιπλέον μία βασική διαφορά τους είναι ότι δεν είναι απαραίτητο κάποιο πρόγραμμα ξενιστή και αποτελεί ένας πλήρες πρόγραμμα. Συνήθως, χρησιμοποιείται για την αποστολή προσωπικών

δεδομένων -όπως passwords- ή την υπερφόρτωση του δικτύου με άχρηστη κίνηση πληροφορίας, κοινώς θόρυβος.

3.3.3 Trojan Horse

Το Trojan horse είναι ένα μη-ιομορφικό επιβλαβές λογισμικό και αρχικός στόχος του είναι να μην γίνει αντιληπτή η παρουσία του. Για αυτό τον σκοπό η σύνηθες συμπεριφορά του είναι να αντικαθιστά υπάρχοντα αρχεία του πληροφοριακού συστήματος, ξεγελώντας τον χρήστη. Για παράδειγμα θα μπορούσε να υποκαταστήσει το πρόγραμμα του music player που έχουμε όλοι στον υπολογιστή μας, συνεχίζοντας να λειτουργεί φαινομενικά ως player αλλά παράλληλα να κάνει επιβλαβείς ενέργειες σε βάρος του χρήστη κάθε φορά που εκτελείται από αυτόν.

3.3.4 Backdoor / Trapdoor

Το backdoor, μπορεί εν δυνάμει να θεωρηθεί υποκατηγορία του Trojan horse και είναι ένα επιβλαβές λογισμικό που δίνει στον κακόβουλο χρήστη την δυνατότητα της απομακρυσμένης διαχείρισης του υπολογιστή του θύματος. Παρόλο που έχει την ικανότητα είτε να είναι αυτόνομο, είτε να αποτελεί κομμάτι κάποιου άλλου προγράμματος, δεν μπορεί να διαδοθεί.

3.3.5 Bot / Botnet / Zombie

Το bot αποτελεί συντομογραφία της γνωστής λέξη robot και είναι ένα πρόγραμμα που ενεργοποιείται σε ένα πληροφοριακό σύστημα εκτελώντας μια αυτοματοποιημένη διεργασία. Συνεπώς, όταν αναφερόμαστε σε bot δεν σημαίνει ότι εννοούμε απαραίτητα ένα επιβλαβές λογισμικό, αφού μπορεί η διεργασία που εκτελεί να προσφέρει μόνο θετικά για το πληροφοριακό σύστημα. Ένα επιβλαβές bot επιτρέπει σε έναν εισβολέα να πάρει τον έλεγχο του υπολογιστή ενός ανυποψίαστου χρήστη. [19]

Ένα botnet δεν είναι τίποτα άλλο από μία σειρά από συνδεδεμένους υπολογιστές, οι οποίοι συνεργάζονται για να εκτελέσουν μαζί μια συγκεκριμένη εργασία επιβλαβή ή μη. Το botnet ελέγχεται εξ αποστάσεως από τον λεγόμενο botherders ή botmaster χωρίς τη γνώση ή την έγκριση των κατόχων των μεμονωμένων υπολογιστών και μπορεί να εκτελέσει μια μαζική-συντονισμένη δράση. Οι υπολογιστές που είναι μέλη του δικτύου αυτού ονομάζονται zombies. [20] Όταν ο botmaster πάρει τον έλεγχο, συνήθως χρησιμοποιεί τα μηχανήματα των χρηστών για να εκτελέσει

επιβλαβείς ενέργειες. Η πρόσβαση στον κάθε zombie-υπολογιστή είναι αντίστοιχη με το να βρισκόταν ο ίδιος ο εισβολέας μπροστά σε αυτόν και φυσικά είναι δυνατή τόσο η πρόσβαση στα αρχεία του συστήματος όσο και η χρήση της σύνδεσης δικτύου. Οι ενέργειες που εκτελούνται από αυτά τα επιβλαβή botnets συνήθως περιλαμβάνουν αποστολή spam emails σε εκατομμύρια χρήστες, επιθέσεις DDoS (Distributed Denial of Services) δημιουργία πλασματικής κίνησης και αντικατάσταση διαφημίσεων σε banners σε συγκεκριμένους ιστότοπους για οικονομικές απολαβές.

3.3.6 Downloader

Ως downloader αναφέρεται ο κακόβουλος κώδικας που συνήθως τοποθετείται από τους επιτιθέμενους μόλις πάρουν πρόσβαση στο σύστημα του θύματος για να κατεβάζουν και να εγκαθιστούν επιπλέον επιβλαβές λογισμικό. [6]

3.3.7 Rootkit

Το rootkit είναι ένα επιβλαβές λογισμικό που χρησιμοποιεί τεχνικές απόκρυψης, για να εισχωρήσει όσο το δυνατόν σε περισσότερο «βάθος» στο λειτουργικό σύστημα ενός πληροφοριακού συστήματος χωρίς να γίνει αντιληπτό. Προέρχεται από την λέξη root που δεν είναι άλλος από τονAdministrator ενός Unix συστήματος και την λέξη kit που σημαίνει εργαλειοθήκη. [21] Για να παραμείνει «αθόρυβο», το rootkit συνήθως αντικαθιστά κρίσιμα συστεμικά αρχεία με αντίστοιχα παραποιημένα. Αξίζει να αναφερθεί ότι συχνά συνεργάζονται με το επιβλαβές λογισμικό backdoor. Το πιο γνωστό rootkit ήταν εκείνο της εταιρίας Sony Music για την προστασία των μουσικών CD από την πειρατεία.

3.3.8 Scareware

Το scareware έχει δημιουργηθεί για να φοβίσει τον χρήστη, ώστε εκείνος να προβεί σε μια αγορά. Συνήθως το περιβάλλον εργασίας μοιάζει με πρόγραμμα antivirus ή άλλο πρόγραμμα προστασίας / ασφάλειας υπολογιστών. Ειδοποιεί ψευδώς τον χρήστη ότι υπάρχει κακόβουλος κώδικας στο σύστημά του και ο μόνος τρόπος για να το απομακρύνει είναι η αγορά ενός ειδικού λογισμικού, ενώ στην πραγματικότητα το λογισμικό που αγοράζει δεν κάνει τίποτα άλλο από το να αφαιρεί το scareware. [22]

3.3.9 Ransomware

Το ransomware είναι ένα σχετικά νέο είδος επιβλαβούς λογισμικού που καταφέρνει να «κλειδώσει» τα αρχεία του χρήστη-θύματος μέχρι εκείνος να πληρώσει κάποια χρήματα ως λύτρα. Η ονομασία ransomware προέρχεται από τη λέξη ransom που σημαίνει λύτρα και βέβαια την κατάληξη «ware» από το software - λογισμικό. [23]

Το συγκεκριμένο είδος επιβλαβούς λογισμικού συνήθως αρχίζει την δράση του, παρόμοια με ένα scareware. Όμως αντί να προσπαθεί να ξεγελάσει το θύμα για να αγοράσει πλαστά λογισμικά προστασίας από ιούς κρατάει όμηρο τον υπολογιστή ζητώντας λύτρα. Παλαιότερα μπλόκαρε την πρόσβαση σε ολόκληρο τον υπολογιστή εμφανίζοντας κάποιο μήνυμα στην οθόνη, όμως αργότερα εξελίχθηκε κρυπτογραφώντας συγκεκριμένα αρχεία του χρήστη με ένα ή περισσότερα κλειδιά κρυπτογράφησης που τα αποθήκευε είτε τοπικά στον υπολογιστή είτε σε κάποιον server απομακρυσμένα.

3.3.10 Application Launcher

Ο application launcher χρησιμοποιείται για την εκκίνηση άλλων προγραμμάτων επιβλαβών ή μη. Συνήθως χρησιμοποιεί μη παραδοσιακές τεχνικές για να εκκινήσει άλλα προγράμματα, προκειμένου να διασφαλίσουν την διακριτικότητά τους όταν αποκτούν πρόσβαση με αυξημένα δικαιώματα. [24]

3.3.11 Information-stealing

Το συγκεκριμένο είδος συλλέγει πληροφορίες από τον υπολογιστή του θύματος και συνήθως τα στέλνει στον επιτιθέμενο. Η συνηθέστερη χρήση του είναι η πρόσβαση σε λογαριασμούς e-banking ή απλώς λογαριασμούς ηλεκτρονικού ταχυδρομείου. Στην κατηγορία αυτή θα μπορούσε να τοποθετηθεί και το είδος του keystroke logging.

3.3.12 Keystroke Logging

Τα συγκεκριμένο είδος είναι ένα εργαλείο κατασκευασμένο για να καταγράφει τα κουμπιά που έχει πληκτρολογήσει ο χρήστης, από την στιγμή που αυτό έχει ενεργοποιηθεί, ώστε να ανακτηθούν αργότερα. [25] Στην κακόβουλη εκδοχή του επιτρέπει στον εισβολέα να αποκτήσει πρόσβαση σε εμπιστευτικές πληροφορίες που πληκτρολογήθηκαν, όπως οι κωδικοί χρήστη, ο αριθμός πιστωτικής κάρτας, το περιεχόμενο ενός εμπιστευτικού μηνύματος ή άλλα προσωπικά δεδομένα.

3.3.13 Logic Bomb

Μια «λογική βόμβα» αποτελεί ένα λογισμικό που σχεδιάστηκε για να εκτελεστεί κάτω από συγκεκριμένες περιστάσεις, όπως για παράδειγμα μετά το πέρας ενός χρονικού διαστήματος ή μετά την εκτέλεση μια ενέργειας του χρήστη. Όταν ενεργοποιηθεί μπορεί να εμφανίζει πλαστά μηνύματα, να διαγράφει ή να καταστρέφει δεδομένα ή να έχει άλλες ανεπιθύμητες ενέργειες. [26]

3.3.14 Adware

Το συγκεκριμένο είδος συγκεντρώνει και στέλνει σε διαφημιστικές εταιρείες πληροφορίες σχετικές με τη διαδικτυακή δραστηριότητα του χρήστη. Για παράδειγμα τα sites που επισκέπτεται συχνότερα κατά τη διάρκεια της ημέρας ή τα αιτήματα αναζήτησής του. Σκοπός του είναι να στείλει στοχευμένες διαφημίσεις στον χρήστη, για αυτό δημιουργεί ένα προφίλ / βάση με όλες αυτές τις πληροφορίες. Όταν το εν λόγω λογισμικό δεν ειδοποιεί τον χρήστη ή δεν παίρνει την συγκατάθεσή του θεωρείται επιβλαβές. [27]

3.3.15 Spyware

Ως spyware αναφέρεται κάθε είδους επιβλαβές λογισμικό που φορτώνεται στον υπολογιστή του χρήστη χωρίς την άδεια ή/και συγκατάθεσή του. Μεταξύ άλλων, αποτελεί υπερσύνολο των κατηγοριών Adware & Trojan horse. [28] Εκτελεί διάφορες ενέργειες στο παρασκήνιο όπως συλλογή προσωπικών δεδομένων αλλά ταυτόχρονα έχει την δυνατότητα να αλλάζει τις ρυθμίσεις του υπολογιστή ώστε να εκτελεί δραστηριότητες που μπορεί να τον ενοχλήσουν.

3.3.16 Spam-sending

Επιβλαβές λογισμικό που μολύνει το μηχάνημα του χρήστη και στη συνέχεια χρησιμοποιεί αυτό το μηχάνημα για την αποστολή spam. Αυτό το επιβλαβές λογισμικό δημιουργεί έσοδα για τους επιτιθέμενους, επιτρέποντάς τους να πωλούν τις υπηρεσίες αποστολής spam σε τρίτους.

3.3.17 Rogue Security Software

Το rogue security software είναι μια μορφή επιβλαβούς λογισμικού που παραπλανά τους χρήστες ώστε να πιστέψουν ότι υπάρχει ένας ιός στον υπολογιστή τους. [29] Στην συνέχεια ο χρήστης στην πραγματικότητα αντί να εγκαταστήσει ένα εργαλείο αφαίρεσης, εισάγει ένα επιβλαβές λογισμικό στον υπολογιστή του. Αποτελεί μια

συγγενική μορφή των scarewares που χειραγωγούν τους χρήστες μέσω του φόβου αλλά στην πραγματικότητα δεν απεγκαθίστανται με τόση ευκολία όσο αυτά.

3.3.18 Browser Hijacker

Το είδος αυτό είναι μια μορφή ανεπιθύμητου λογισμικού που τροποποιεί, χωρίς την άδεια του χρήστη, τις ρυθμίσεις ενός web browser για να εισάγει ανεπιθύμητα διαφημιστικά στο πρόγραμμα περιήγησης του. [30] Ο κακόβουλος χρήστης αντικαθιστά την υπάρχουσα αρχική σελίδα ή τη σελίδα αναζήτησης με μία της επιλογής του. Αυτός ο τρόπος αυξάνει την κίνηση στην συγκεκριμένη ιστοσελίδα, αυξάνοντας τα διαφημιστικά έσοδα της.

3.3.19 Flooder

Το επιβλαβές λογισμικό flooder επιτρέπει σε έναν εισβολέα να στείλει ένα τεράστιο ποσό δεδομένων σε ένα συγκεκριμένο στόχο. Συνήθη χρήση των flooders χρησιμοποιούνται στα κανάλια IRC όπου στέλνοντας τεράστιοι όγκοι δεδομένων σε ένα κανάλι ή σε ένα επιλεγμένο χρήστη διαταράσσοντας την επικοινωνία. [31]

Γενικεύοντας την παραπάνω ανάλυση των επιβλαβών λογισμικών μπορούμε να αναφέρουμε ότι κάποιες από τις λειτουργίες που εκτελεί ένα τέτοιο πρόγραμμα είναι η αλλαγή ρυθμίσεων του υπολογιστή, των εγκατεστημένων προγραμμάτων και παραμέτρων της registry, η αναπαραγωγή του ή/και αποστολή του εαυτού του, το άνοιγμα θυρών στο σύστημα και η παροχή απομακρυσμένης πρόσβασης, η τροποποίηση των ρυθμίσεων ασφαλείας του συστήματος, η συγκέντρωση προσωπικών στοιχείων ή ευαίσθητων πληροφοριών και η καταγραφή των πληκτρολογήσεων, η σύνδεση σε απομακρυσμένους υπολογιστές, η αυτόματη λήψη και εκτέλεση αρχείων από το Διαδίκτυο και η εισαγωγή κώδικα σε άλλα προγράμματα και διεργασίες.

Η ραγδαία αύξηση των επιβλαβών λογισμικών και των επιθέσεων σε ηλεκτρονικούς υπολογιστές έχει δημιουργήσει πολλά προβλήματα σε μεμονωμένους χρήστες αλλά και ζημιές εκατομμύριών σε μεγάλες εταιρείες. Αναμενόμενο ήταν ταυτόχρονα με την ανάπτυξη των επιβλαβών λογισμικών να αναπτυχθούν και συστήματα αντιμετώπισης των λογισμικών αυτών.

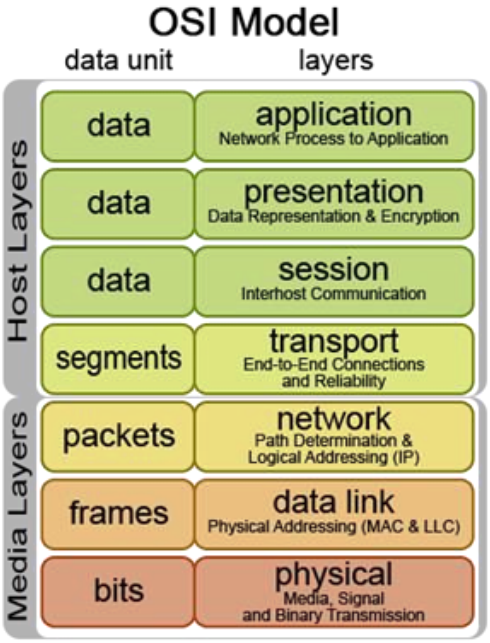
4 *Συστήματα Αντιμετώπισης Επιβλαβών Λογισμικών*

Το 1984 ο Fred Cohen, στην επιστημονική εργασία του «Computer viruses – Theory and Experiments» αναφέρει πως δεν υπάρχει αλγόριθμος που να μπορεί να εντοπίζει με πλήρη επιτυχία κάθε επιβλαβές λογισμικό είτε αυτό υπάρχει ήδη, είτε θα δημιουργηθεί στο μέλλον. [32] Συνεπώς αφού δεν υπάρχει πανάκεια μέθοδος, οι χρήστες κάθε πληροφοριακού συστήματος είναι υποχρεωμένοι να χρησιμοποιούν ένα σύνολο εργαλείων / συστημάτων προστασίας, χωρίς αυτό να τους εφησυχάζει αφού θα πρέπει να παραμένουν πολύ προσεκτικοί σε πιθανές απειλές. Τα βασικά συστήματα αντιμετώπισης είναι τα εξής:

4.1 Firewall (Τείχος Προστασίας)

Firewall ή αλλιώς το Τείχος Προστασίας είναι ένα σύστημα που ελέγχει τη ροή της κυκλοφορίας των πακέτων δεδομένων ενός δικτύου. [33] Για παράδειγμα, πολλές επιχειρήσεις χρησιμοποιούν firewall για τον περιορισμό της συνδεσιμότητας -προς και από- τα εσωτερικά δίκτυα που χρησιμοποιούνται για υπηρεσίες με πιο ευαίσθητες λειτουργίες, όπως οικονομικά στοιχεία ή στοιχεία προσωπικού. Με αυτόν τον τρόπο μπορεί να αποτρέψει τη μη εξουσιοδοτημένη πρόσβαση, σε συγκεκριμένα συστήματα καθώς και σε πόρους αυτών.

Τα συστήματα firewalls παρουσιάζουν μια ποικιλία δυνατοτήτων, αλλά πριν προχωρήσουμε σε αυτές είναι σημαντικό να αναφερθούμε στα επίπεδα (layers) που εξετάζεται στο γνωστό μοντέλο αναφοράς ανοιχτής διασύνδεσης, OSI (Εικόνα 1).



Εικόνα 1: Μοντέλο OSI

Όταν ένας χρήστης θέλει να μεταφέρει δεδομένων στα δίκτυα, τα δεδομένα περνούν από το υψηλότερο -μέσω των ενδιάμεσων- στο χαμηλότερο επίπεδο, με κάθε επίπεδο να προσθέτει περισσότερες πληροφορίες. Οι διευθύνσεις στο επίπεδο data link, αναφέρονται ως Media Access Control (MAC) και αποτελούν την φυσική διεύθυνση της κάρτας δικτύου. Οι πολιτικές προστασίας (Policies) ενός firewall σπάνια ασχολούνται με αυτό το επίπεδο. Αντίθετα υπάρχει αυξημένος έλεγχος στις IP διευθύνσεις στο επίπεδο του network (δικτύου). Στο επίπεδο transport (μεταφοράς) προσδιορίζονται συγκεκριμένες εφαρμογές δικτύου καθώς και συνεδρίες (sessions) επικοινωνίας. Στο ίδιο επίπεδο έχει νόημα η έννοια της port, που αποτελεί τον αριθμό της θύρας που επικοινωνεί μια υπηρεσία τόσο στον τομέα του προορισμού όσο και στο τομέα της πηγής. Τα γνωστά πρωτόκολλα TCP & UDP στο transport επίπεδο λειτουργούν αποκλειστικά με ports. Ο συνδυασμός της διεύθυνσης IP μαζί με την port -της πηγής- και της διεύθυνσης IP μαζί με την port –του προορισμού-, χαρακτηρίζουν την συνεδρία (session). Το υψηλότερο επίπεδο, application (εφαρμογής), αναφέρεται στον χρήστη και μπορεί να ελεγχθεί από συστήματα firewalls ως βάση για την εφαρμογή των πολιτικών (Policies) ασφαλείας. Στην πλειονότητά τους, τα firewalls λειτουργούν σε ένα ή μερικά επίπεδα αλλά υπάρχουν και πιο εξελιγμένα που ελέγχουν σχεδόν το σύνολο των επιπέδων.

Τα συστήματα firewall συχνά συνδυάζονται με άλλες τεχνολογίες δρομολόγησης όπως το Network Address Translation (NAT), που μερικές φορές θεωρείται -χωρίς να ισχύει- ως μια πιο εξελιγμένη τεχνολογία firewall. Επίσης, πολλοί firewalls

έχουν δυνατότητες φιλτραρίσματος για την επιβολή πολιτικών (policies), που δεν σχετίζονται άμεσα με την ασφάλεια. Η αλήθεια είναι ότι ένα τείχος προστασίας τοποθετείται στην περίμετρο ενός δικτύου. Συνεπώς διαθέτουν δύο βασικές διασυνδέσεις, μια εξωτερική (μη προστατευμένη) και μία εσωτερική διασύνδεση (προστατευμένη). Ωστόσο, οι πολιτικές προστασίας μπορεί να λειτουργήσουν και στις δύο κατευθύνσεις. Για παράδειγμα, ίσως να υπάρξει μια πολιτική για την πρόληψη για την αποστολή εκτελέσιμου κώδικα (exe) από το εσωτερικό της περιμέτρου σε περιοχές έξω από την περίμετρο.

Οι βασικές δυνατότητες των συστημάτων firewalls αναλύονται ως εξής:

4.1.1 Packet Filtering

Το packet filtering (φίλτρο πακέτων) θεωρείται μια από τις βασικότερες δυνατότητες ενός συστήματος firewall. Ακόμη και σήμερα υπάρχουν συστήματα firewalls που παρέχουν αποκλειστικά και μόνο αυτή την δυνατότητα, χωρίς να τους απασχολεί το περιεχόμενο των πακέτων. Η λειτουργικότητα ελέγχου πρόσβασης τους, διέπεται από ένα σύνολο οδηγιών που αναφέρονται ως σύνολο κανόνων (ruleset). Ένα παράδειγμα ενός απλού packet filtering firewall είναι ένας δρομολογητής δικτύου (router) που χρησιμοποιεί λίστες ελέγχου πρόσβασης. Τα εν λόγω firewalls λειτουργούν συνήθως στο επίπεδο network συνεπώς συμπεριλαμβάνουν ελέγχους:

- στο πρωτόκολλο δικτύου ή μεταφοράς που χρησιμοποιούνται για την επικοινωνία μεταξύ της πηγής και του προορισμού, όπως TCP & UDP.
- την διεπαφή της διαδρομής που διανύει το πακέτο.
- τα χαρακτηριστικά των συνόδων επικοινωνίας στο επίπεδο transport, όπως πηγή και προορισμός session καθώς και τα ports τους.
- τα πακέτα της πηγής και του προορισμού

4.1.2 Stateful Inspection

Η ικανότητα του stateful inspection (της αυστηρής επιθεώρησης) βελτιώνει τις λειτουργίες των φίλτρων πακέτων από την πλευρά της παρακολούθησης της κατάστασης των συνδέσεων και το κλείδωμα των πακέτων που αποκλίνουν από την αναμενόμενη συμπεριφορά. Αυτό επιτυγχάνεται με την ενσωμάτωση μεγαλύτερης ευαισθητοποίησης στο επίπεδο transport. Όπως και στο packet filtering αναχαιτίζονται τα πακέτα στο επίπεδο network και γίνεται έλεγχος για να διαπιστωθεί αν επιτρέπονται βάση των υπαρχόντων κανόνων, αλλά στην λειτουργία stateful inspection παρακολουθείται κάθε σύνδεση σε έναν πίνακα καταστάσεων.

Γενικότερα υπάρχουν τρεις βασικές καταστάσεις για την κυκλοφορία/κίνηση της TCP, η εγκαθίδρυση της σύνδεσης, ή χρήση και ο τερματισμός. Η κατάσταση του κάθε πακέτου ελέγχεται για να διαπιστωθεί αν έρχεται σε αντίθεση με την αναμενόμενη κατάσταση. Για παράδειγμα, ένας εισβολέας θα μπορούσε να δημιουργήσει ένα πακέτο με header τέτοιο ώστε να δείχνει ότι είναι μέρος μιας υπάρχουσας σύνδεσης, με την ελπίδα ότι θα περάσει μέσα από το firewall. Σε μια τέτοια περίπτωση θα μπορούσε να απορριφθεί ένα τέτοιο πακέτο με έναν απλό έλεγχο στον πίνακα καταστάσεων.

4.1.3 Application Firewalls

Στην περίπτωση των application firewalls μπορεί να είναι δυνατή η ταυτοποίηση απροσδόκητων ακολουθιών εντολών, όπως για παράδειγμα η έκδοση της ίδια εντολή επανειλημμένα ή η έκδοση μιας εντολής που δεν προηγήθηκε από κάποια άλλη εντολή για την οποία αυτή εξαρτάται. Αυτές οι ύποπτες εντολές συχνά προέρχονται από επιθέσεις υπερχείλιση μνήμης (overflow attacks), DoS και άλλες μορφές προσβολής που πραγματοποιούνται στο πλαίσιο των πρωτοκόλλων εφαρμογής, όπως το HTTP. Ένα άλλο χαρακτηριστικό που μπορεί να ελεγχθεί από application firewalls είναι η συμμόρφωση της κυκλοφορίας / κίνησης ως προς το εκάστοτε πρωτόκολλο.

4.1.4 Web Application Firewalls

Το πρωτόκολλο HTTP που χρησιμοποιείται σε διακομιστές web έχει αξιοποιηθεί από επιτιθέμενους με πολλούς τρόπους. Για παράδειγμα, η τοποθέτηση ενός επιβλαβούς λογισμικού στον υπολογιστή μέσω του διαδικτύου. Για αυτές τις περιπτώσεις χρησιμοποιούνται εξειδικευμένοι firewalls applications με την ονομασία web Application firewallsμε συνηθισμένη τοπολογία να λειτουργούν ακριβώς μπροστά από τον web server.

4.1.5 Application-proxy Gateways

Το συγκεκριμένο χαρακτηριστικό των προηγμένων firewalls συνδυάζει τον έλεγχο της πρόσβασης του χαμηλότερου επιπέδου με την λειτουργικότητα του ανώτερου επιπέδου. Πιο συγκεκριμένα αυτά τα firewalls περιέχουν έναν agent-proxy που ενεργεί ως ενδιάμεσος μεταξύ αυτών που επιθυμούν να επικοινωνούν, απαγορεύοντας μια άμεση σύνδεση μεταξύ τους. Κάθε επιτυχημένη προσπάθεια σύνδεσης στην πραγματικότητα έχει ως αποτέλεσμα την δημιουργία δύο ξεχωριστών συνδέσεων-μία μεταξύ του πελάτη και του διακομιστή μεσολάβησης, και μία άλλη

μεταξύ του διακομιστή μεσολάβησης και του πραγματικού προορισμού. Συνεπώς, οι εσωτερικές διευθύνσεις IP δεν είναι ορατές στον εξωτερικό κόσμο.

4.1.6 Virtual Private Networking

Όπως διαπιστώνεται, λόγω της τοπολογίας τους, πολλές φορές ζητούνται από τα firewall συστήματα να κάνουν περισσότερα από ένα συνηθισμένο μπλοκάρισμα της ανεπιθύμητης κίνησης ενός πληροφοριακού συστήματος. Στην περίπτωση του virtual private networking (VPN), η επιπλέον λειτουργικότητα είναι η κρυπτογράφηση και η αποκρυπτογράφηση συγκεκριμένων «ρευμάτων» κυκλοφορίας δικτύου, μεταξύ του προστατευόμενου δικτύου και του εξωτερικού δικτύου. Τα VPN χρησιμοποιούνται πιο συχνά για να παρέχουν ασφαλείς επικοινωνίες του δικτύου πάνω από μη αξιόπιστα δίκτυα.

4.2 Σύστημα Ανίχνευσης Εισβολής (IDS)

Το Σύστημα Ανίχνευσης Εισβολής (ΣΑΕ) (αγγλ. Intrusion Detection System, IDS) αποτελεί σύστημα παρακολούθησης και ανάλυσης των συμβάντων, τα οποία λαμβάνουν χώρα τόσο στους ίδιους τους ηλεκτρονικούς υπολογιστές όσο και στα δίκτυα υπολογιστών. [34] Στόχος είναι ο εντοπισμός ενδείξεων για πιθανές προσπάθειες εισβολής, κατά τις οποίες συχνά εντοπίζονται ίχνη παραβίασης της ακεραιότητας, της εμπιστευτικότητας και της διαθεσιμότητας των πληροφοριακών πόρων.

Στα IDS υπάρχουν διαφορετικές προσεγγίσεις στην κατηγοριοποίησής τους. Πιο συγκεκριμένα μπορούν να χαρακτηριστούν από (3) τρεις βασικές λειτουργίες ως εξής:

4.2.1 Πηγές Πληροφορίας (Information Sources, IS)

Το IDS χρησιμοποιεί πηγές ώστε να συλλέξει την κατάλληλη πληροφορία. Οι πηγές πληροφορίας μπορεί να είναι σε επίπεδο παρακολούθησης συστήματος (Host) ή δικτύου (Network).

4.2.1.1 Host Intrusion Detection Systems (HIDS)

Στα Host Intrusion Detection Systems (HIDS) η πληροφορία συλλέγεται μόνο από ένα σύστημα. Αυτό δίνει την δυνατότητα στα HIDS να δίνουν πληροφορίες ακόμη και για τις διαδικασίες (processes), του συστήματος που προστατεύουν, προσφέροντας λεπτομερή πληροφόρηση. Επιπλέον, μπορούν να ελέγχουν τα αρχεία

καταγραφής του συστήματος. Τέλος, σε μερικές περιπτώσεις έχουν την δυνατότητα χρήσης μίας κοινής κονσόλας διαχείρισης και ελέγχου πολλών συστημάτων, ώστε να υπάρχει μια ολοκληρωμένη εικόνα.

4.2.1.2 Network Intrusion Detection Systems (NIDS)

Τα Network Intrusion Detection Systems (NIDS) είναι πιο διαδεδομένα στην σημερινή εποχή. Είναι ειδικότερα επιφορτισμένα να παρακολουθούν και να αναλύουν κάθε πακέτο του δικτύου (traffic). Επεξεργάζονται κάθε πακέτο που περνάει από το σημείο που έχουν εγκατασταθεί, το αναλύουν τοπικά σε πραγματικό χρόνο και καταγράφουν τα αποτελέσματά. Τα NIDS μπορούν να αποτελούνται και από μικρότερα συστήματα, ως Sensors. Οι Sensor τοποθετούνται σε διάφορα σημεία ενός δικτύου και εκτελούν ακριβώς τις ίδιες λειτουργίες με την διαφορά ότι στέλνουν αυτές τις πληροφορίες στο κεντρικό σύστημα. Η βασική δυναμική των Sensors είναι ότι μπορούν να κάνουν κρυφή την παρουσία τους, έτσι ώστε να μην γίνεται αντιληπτή η ύπαρξη ή η θέση τους.

4.2.2 Ανάλυση (Analysis)

Το IDS έχει διάφορες μεθόδους ανάλυσης για να οργανώνει τα δεδομένα από τις Πηγές Πληροφορίας (IS) και να αποφασίζει ποια από αυτά μπορεί να αποτελούν μία επίθεση. Πιο συγκεκριμένα έχουμε τις μεθόδους Misuse Detection, Anomaly Detection και Protocol Anomaly Detection.

Με την μέθοδο Misuse Detection ελέγχεται ένα δίκτυο για να εντοπιστούν γεγονότα που ταιριάζουν με κάποια προκαθορισμένα πρότυπα γνωστών επιθέσεων. Τα πρότυπα αυτά ονομάζονται Υπογραφές (Signatures). Μια Υπογραφή περιγράφει κάποια χαρακτηριστικά ενός πακέτου και ενός συγκεκριμένου λεκτικού που χρησιμοποιείται για μία επίθεση.

Με την μέθοδο Anomaly Detection ελέγχεται η ασυνήθιστη –μη φυσιολογική- συμπεριφορά ενός συστήματος ή δικτύου. Δηλαδή βασίζεται, κατά κύριο λόγο, στις διαφορές από την φυσιολογική δραστηριότητα. Στην αρχή δημιουργούνται κάποια Πρότυπα (Patterns) σε μια χρονική περίοδο που θεωρείται «φυσιολογική» σε επίπεδο συστημάτων ή/και του traffic ενός δικτύου και αποθηκεύονται ως δείγματα. Στη συνέχεια δεν γίνεται τίποτα άλλο από έλεγχος της διαφοράς των Προτύπων με τα νέα δεδομένα, παρακολουθώντας μόνο την διαφορά τους.

Η μέθοδος του Protocol Anomaly Detection είναι αρκετά νέα και αποτελεί ,όπως προδίδει και το όνομά της, παραλλαγή της Anomaly Detection. Η Protocol Anomaly

Detection παρατηρεί επίσης τις διαφορές, επικεντρώνοντας όμως αποκλειστικά στην σωστή χρήση των πολύ ευαίσθητων πρωτοκόλλων επικοινωνίας και ειδικότερα στο TCP/IP. Αυτή η μέθοδος έλυσε αρκετά προβλήματα αφού βρέθηκαν πολλές επιθέσεις να στοχεύουν το ενδιαφέρον τους στα πρωτόκολλα αυτά. Τα RFCs (Request For Comments) που περιγράφουν τα Standards, ορίζουν την θεωρητική χρήση των πρωτοκόλλων. Οι επιθέσεις αυτές στοχεύουν σε ενέργειες, που είτε έχουν γίνει λόγω κακής χρήσης των RFCs, είτε έχουν παραληφθεί γενικότερα.

4.2.3 Απόκριση (Response)

Το IDS, όταν ανιχνεύσει μία επίθεση, θα εκτελέσει μια ακολουθία ενεργειών ή αποκρίσεων. Τα είδη αποκρίσεων μπορούν να χαρακτηριστούν Παθητικές (Passive) ή Ενεργητικές (Active).

Οι Παθητικές Αποκρίσεις συνήθως καταγράφουν το γεγονός της επίθεσης και ενημερώνουν με κάποιο τρόπο τους υπεύθυνους, ώστε αυτοί να πάρουν τα κατάλληλα μέτρα.

Μια τέτοια ενημέρωση μπορεί να πραγματοποιηθεί με 2 (δύο) βασικούς τρόπους:

4.2.3.1 Ανακοίνωση των Alerts

Alert στην περίπτωσή μας ονομάζεται μια επισήμανση για την ανίχνευση μιας ή περισσότερων επιθέσεων. Ο τρόπος, ο χρόνος και τα άτομα που θα παρουσιάζονται τα εν λόγω alerts ρυθμίζονται κατά την εγκατάσταση ενός ολοκληρωμένου συστήματος IDS ή/και αργότερα. Ο χρόνος μπορεί να διαφέρει και να οριστεί ώστε η ενημέρωση να γίνεται ανά τακτά χρονικά διαστήματα είτε ακόμη και σε πραγματικό χρόνο. Ο τρόπος –η μορφή- των alerts μπορεί να είναι είτε λεπτομερείς και αναλυτική αναφέροντας το πακέτο και το εργαλείο που δημιούργησε το πρόβλημα, είτε απλή αναφορά του θύματος και του επιτιθέμενου. Τέλος, υπάρχει να ειδοποιούνται με email, με sms ή ακόμη και με τηλεφωνική κλήση συγκεκριμένοι χρήστες που, είτε γιατί μπορούν να ενεργήσουν κατάλληλα για να αποκρούσουν την επίθεση, είτε γιατί είναι τα θύματα αυτής.

4.2.3.2 SNMP Traps

Με την χρήση SNMP Traps υπάρχει η δυνατότητα να αναφερθούν τα alerts σε ένα κεντρικό σύστημα διαχείρισης του δικτύου. Με αυτόν τον τρόπο συγκεντρώνονται όλα τα απαραίτητα στοιχεία που έχουν σταλεί και από άλλα συστήματα και πηγές όπως Firewall, Routers, Switches κ.α., συσχετίζοντας με σαφήνεια και πληρότητα

τα αποτελέσματα για μεγαλύτερη ευκολία στην διαδικασία των κρίσιμων αποφάσεων αντιμετώπισης των επιθέσεων..

Οι Ενεργητικές Αποκρίσεις επικεντρώνουν ειδικά στην αυτοματοποιημένη αντιμετώπιση της επίθεσης από το ίδιο το IDS.

Γενικότερα υπάρχουν 3 (τρία) είδη Ενεργητικών Αποκρίσεων:

4.2.3.2.1 Συλλογή Επιπρόσθετων Πληροφοριών

Λειτουργούν συλλέγοντας επιπλέον πληροφορίες για μια πιθανή επίθεση ώστε να διαλευκάνουν περισσότερο την κατάσταση παίρνοντας επιπλέον μέτρα προστασίας. Στην ουσία μπορεί να επιλέγει συγκεκριμένες επιθέσεις που θεωρείται ότι χρειάζονται μεγαλύτερη διερεύνηση και να επιστρατεύει και επιπλέον Sensors μαζεύοντας περισσότερες πληροφορίες με στόχο στην συνέχεια να χρησιμοποιηθούν για τον εντοπισμό του hacker ή/και να αποτελούν πειστήριο για πιθανή δίωξή του.

4.2.3.2.2 Παρεμπόδιση του επιτιθέμενου

Το IDS εντοπίζει τα πακέτα που έχουν μια συγκεκριμένη IP διεύθυνση, από την οποία προέρχεται ο επιτιθέμενος ή τουλάχιστον «φαίνεται» ότι προέρχεται ο επιτιθέμενος. Συνεπώς, αν κάποιος hacker αφήνει να φαίνεται μια ψεύτικη IP, που είναι και το πιθανότερο για έναν «έμπειρο», πολλές φορές δεν είναι αξιόπιστη λύση αφού δεν φαίνεται ο επιτιθέμενος προσωπικά. Σε αυτές τις περιπτώσεις μπορούν στέλνονται πακέτα (με active RST flag on TCP header) για να τερματίσουν την επικοινωνία του συστήματος στόχου με τον hacker. Επιπλέον, να γίνουν σε Firewall & Router ρυθμίσεις, ώστε εκτός από την εμπόδιση των εν λόγω πακέτων, να κλείσουν οι πόρτες και τα πρωτόκολλα που χρησιμοποιήθηκαν.

4.2.3.2.3 Δράση εναντίον του επιτιθέμενου

Στην πιο ακραία μορφή Ενεργητικής Απόκρισης θα μπορούσε ο hacker να αποτελεί στόχο δικιά μας επίθεσης μετά τον εντοπισμό του. Θα μπορούσε αρχικά κάποιος να ισχυριστεί ότι μια τέτοια ενέργεια είναι αρκετά δίκαιη, όμως μια πιο ψύχραιμη ματιά στο θέμα θα έκανε εμφανείς πολλούς κινδύνους. Πρώτον σίγουρα μια τέτοια ενέργεια θα μπορούσε να χαρακτηριστεί ως παράνομη αλλά το χειρότερο είναι ότι μπορεί και να χαρακτηριστεί και εντελώς άσκοπη ή/και επικίνδυνη. Δεν υπάρχει κανένας που να εξασφαλίζει ότι η IP που θα οριστεί ως η IP του επιτιθέμενου είναι σωστά εντοπισμένη. Συνεπώς μια τέτοια ενέργεια εκτός των άλλων μπορεί να βλάψει κάποιον που δεν έχει ουδεμία σχέση με την επίθεση. Τέλος, ακόμη και να είμαστε

βέβαιοι για τα δικτυακά στοιχεία του επιτιθέμενου θα πρέπει να αναλογιστούμε τι μπορεί να σημαίνει ένα άνοιγμα τύπου «βεντέτας» απέναντί του και τι επιπτώσεις μπορεί να έχει για τον οργανισμό ή την εταιρεία κάτι τέτοιο.

4.3 Σύστημα Πρόληψης Εισβολών (IPS)

Το Σύστημα Πρόληψης Εισβολής (Intrusion prevention system, IPS) δεν μένει στην ανίχνευση μιας επίθεσης αλλά προχωράει στην αυτόματη αντιμετώπιση της. Αρκετά συχνά πολλές φορές ένα IPS αποτελεί «αναπόσπαστο» κομμάτι ενός IDS και λειτουργούν ως μια ενιαία οντότητα. Ένα κλασσικό IPS, αποτελείται από ένα firewall και ένα IDS.

Μπορούν να χαρακτηριστούν από (4) τέσσερις βασικές λειτουργίες ως εξής:

4.3.1 Host-Based (HIPS)

Τα HIPS παρακολουθούν, διαχειρίζονται και αποτρέπουν την κίνηση σε ένα συγκεκριμένο μηχάνημα.

4.3.2 Network-based (NIPS)

Τα NIPS παρακολουθούν, διαχειρίζονται και αποτρέπουν την κίνηση ενός δικτύου.

4.3.3 Content-based (CBIPS)

Τα CBIPS παρακολουθούν, διαχειρίζονται και αποτρέπουν πακέτα με συγκεκριμένες Υπογραφές.

4.3.4 Rate-based (RBIPS)

Τα RBIPS παρακολουθούν και εκπαιδεύονται σε «φυσιολογικές» συμπεριφορές της κίνησης. Στην συνέχεια αναγνωρίζουν την διαφορά και εφαρμόζουν διάφορους μηχανισμούς αποτροπής.

4.4 Antivirus

Το πιο γνωστό εργαλείο αντιμετώπισης ενός επιβλαβούς λογισμικού είναι τα προγράμματα antivirus όπου διαθέτουν αυτοματοποιημένους τρόπους πρόληψης και προστασίας σε ένα πληροφοριακό σύστημα. Ένα antivirus είναι επιφορτισμένο να ανακαλύπτει γνωστά patterns επιβλαβών λογισμικού σε όλα τα είδη αρχείων, σε ιστοσελίδες καθώς και σε πακέτα δικτύου. Επιπλέον, τα σύγχρονα antivirus προσπαθούν να ανακαλύψουν νέες κακές συμπεριφορές και θεωρητικά προστατεύουν τους χρήστες από άγνωστα επιβλαβή λογισμικά. Για τον εντοπισμό

του επιβλαβούς λογισμικού, χρησιμοποιούν διάφορες τεχνικές ώστε να μπορέσει να άρει τις όποιες αυτό-προστασίες του.

Στα βασικά στοιχεία ενός antivirus συγκαταλέγεται ο πυρήνας, ο σαρωτής, οι υπηρεσίες συστήματος (system services) ή οι δαίμονες (daemons), προγράμματα οδήγησης φίλτρων συστήματος αρχείων & δικτύου, καθώς και κάποια άλλα βοηθητικά πρόγραμμα υποστήριξης.

Ο πυρήνας, που αποτελεί την καρδιά ενός antivirus, περιέχει βιβλιοθήκες με ρουτίνες για την αποσυσκευασία εκτελέσιμων προγραμμάτων, συμπιεστές, protectors και cryptors.

Αναλύοντας τα προγράμματα antivirus βάση των κοινών τους στοιχείων έχουμε:

4.4.1 Σαρωτές (Scanners)

Τα περισσότερα antivirus χρησιμοποιούν πλέον γραφική διεπαφή χρήστη (GUI) με έναν σαρωτή που ελέγχει κάθε αρχείο που δημιουργείται ή αλλάζει, είτε από το λειτουργικό σύστημα, είτε από τα προγράμματα του χρήστη. Αυτό συνήθως γίνεται σε πραγματικό χρόνο, αλλά υπάρχει και η δυνατότητα της χειροκίνητης σάρωσης. Για παράδειγμα, όταν ο χρήστης δοκιμάσει να ανοίξει ένα έγγραφο, ο σαρωτής ελέγχει αν ταυτόχρονα υπάρχει κάποια παράξενη συμπεριφορά ώστε να σταματήσει το επιβλαβές λογισμικό πριν προλάβει να βλάψει το σύστημά μας. Αν ο παραπάνω έλεγχος δεν πραγματοποιηθεί σε πραγματικό χρόνο μπορεί να οδηγήσει για παράδειγμα σε «αφοπλισμό» του antivirus προσωρινά ή μόνιμα έως ότου ο χρήστης προβεί στην επανεκκίνηση, με όλα τα συνεπαγόμενα κόστη αυτής της ενέργειας. Η χειροκίνητη σάρωση είναι χρήσιμη για περιπτώσεις που υποψιαζόμαστε ότι κάποιο μέσο, για παράδειγμα εάν usb stick, περιέχει κάποιο επιβλαβές λογισμικό και επιθυμούμε να κάνουμε έναν έλεγχο πριν την οποιαδήποτε χρήση.

4.4.2 Υπογραφές (Signatures)

Ο σαρωτής για να καθοριστεί εάν τα αρχεία ή τα πακέτα είναι επιβλαβή χρησιμοποιεί ένα σύνολο υπογραφών. Οι υπογραφές περιγραφικά είναι τα γνωστά patterns των επιβλαβών στοιχείων. Οι υπογραφές τυπικά είναι hashes ή byte-streams που χρησιμοποιούνται για να καθοριστεί αν ένα αρχείο περιέχει επιβλαβή στοιχεία. Κάθε antivirus έχει ένα δικό του signature-scheme.

Οι αλγόριθμοι hashes που χρησιμοποιούνται για τις υπογραφές είναι συνήθως CRC ή MD5, διότι είναι γρήγοροι και δεν φορτώνουν επιπλέον το σύστημα. Κάθε είδος

υπογραφής έχει πλεονεκτήματα και μειονεκτήματα. Για παράδειγμα υπάρχουν υπογραφές που μπορούν να έχουν υψηλό ποσοστό false positives, αλλά δεν καθυστερούν καθόλου. Όπως υπάρχουν και υπογραφές που καθυστερούν στο matching αλλά έχουν χαμηλό ποσοστό false positives.

4.4.3 Συμπιεστές (Compressors)

Είναι σημαντικό ένα antivirus να είναι σε θέση να ελέγχει όλους τους τύπους αρχείων. Συνεπώς, ένα άλλο βασικό στοιχείο των antivirus που βρίσκεται μέσα στον πυρήνα (kernel) είναι η υποστήριξη των πιο γνωστών συμπιεσμένων μορφών αρχείων, όπως zip, rar, 7z, κλπ. Αυτός ο τομέας έχει αρκετά μεγάλο ενδιαφέρον αφού προκύπτουν συχνά ευπάθειες που επηρεάζουν την ορθή λειτουργία του συστήματος.

4.4.4 Αποσυσκευαστές (Unpackers)

Οι αποσυσκευαστές είναι ρουτίνες που αναπτύχθηκαν για την αποσυσκευασία συμπιεσμένων ή προστατευμένων εκτελέσιμων αρχείων. Ένα επιβλαβές λογισμικό συνήθως συσκευάζεται χρησιμοποιώντας διάφορούς απλούς συσκευαστές, είτε αυτοί είναι ελεύθεροι προς χρήση, είτε πληρώνοντας ένα κόστος για αυτούς. Υπάρχουν όμως και πολύ πολύπλοκοι συσκευαστές που μετατρέπουν τον κωδικό σε bytecode καθιστώντας την αποκάλυψη της λογικής του επιβλαβούς λογισμικού πολύ δύσκολη και χρονοβόρα υπόθεση.

4.4.5 Λοιπές Μορφές Αρχείων

Εκτός από τα συμπιεσμένα και πακεταρισμένα αρχεία, ο πυρήνας ενός antivirus θα πρέπει να υποστηρίζει και έναν πολύ μακρύ κατάλογο λοιπών μορφών αρχείων, προκειμένου να προστατεύσει το σύστημά. Μερικά μόνο από αυτά τα αρχεία είναι “pdf”, “doc”, “docx”, “xls”, “xlsx”, “ppt”, “pptx”, “jpg”, “png”, “gif”, “tiff”, “psd”, “raw”, “wav”, “mp3», “mp4“, “mov”, “ico” κλπ. Έτσι γίνεται αντιληπτό πως εκτός του ότι είναι πάρα πολλά τα είδη, αυξάνονται διαρκώς αναγκάζοντας όσους αναπτύσσουν συστήματα antivirus να συμπεριλαμβάνουν ότι καινούργιο υπάρχει σε παγκόσμιο επίπεδο.

4.4.6 Εξομοιωτές (Emulators)

Η υποστήριξη μιας σειράς εξομοιωτών (emulators), από τους πυρήνες των συστημάτων antivirus είναι πλέον απαραίτητη. Όπως ήταν αναμενόμενο ο πιο διαδεδομένος εξομοιωτής είναι ο Intel x86, ενώ ακολουθούν οι ARM & AMD.

Επιπλέον υπάρχουν εξομοιωτές για ορισμένες εικονικές μηχανές όπως η ActionScript της Adobe, η VBScript κ.α.

4.4.7 Πολλαπλές Μηχανές Antivirus (Online)

4.4.7.1 Virus Total

Η εταιρεία Hispasec το 2004 δημιούργησε ένα website με domain name virustotal.com. Στόχος της ήταν ο online έλεγχος ύποπτων ή μη αρχείων από πολλαπλές μηχανές antivirus. Ο συγκεκριμένος έλεγχος αποτέλεσε πόλος έλξης πολλών χρηστών, αφού μπορούσε ο καθένας δωρεάν να ελέγξει αρχεία σε πολλά antivirus ταυτόχρονα, κάτι που φάνταζε αδύνατο στο προσωπικό του σύστημα μέχρι τότε. Το 2012 το virustotal εξαγοράστηκε από την Google, με αποτέλεσμα την χρησιμοποίηση μια σειράς από υπηρεσίες της. Επίσης, εκτός από τον έλεγχο των αρχείων, το virustotal επιτρέπει στους χρήστες να εισάγουν μια διεύθυνση URL για να ελέγξουν τις ιστοσελίδες τους για πιθανές απειλές επιβλαβούς λογισμικού.

Επίσης αξίζει να σημειωθεί ότι πρόσφατα προσέφερε addons του virustotal στους γνωστούς browsers περιήγησης όπως Chrome, Firefox, Internet Explorer και Safari. Το μεγέθους για τα αρχεία που μπορεί κανείς να ανεβάσει είναι 128MB, όριο που για ένα εκτελέσιμο αρχείο είναι υπέρ αρκετό. Τέλος, για την αναγνώριση των αρχείων που έχουν ανέβει κατ' επανάληψη, χρησιμοποιεί μηχανισμό checksum SHA-256.

Σήμερα, για τις ανάγκες των ελέγχων -το virustotal- έχει σε συνεχή χρήση 56 antivirus, 62 μηχανές σάρωσης ιστοσελίδων/συνόλων δεδομένων και 18 εργαλεία χαρακτηρισμού αρχείων & συνόλων δεδομένων.

4.4.7.2 Metascan Online

Η συγκεκριμένη online μηχανή σάρωσης αρχείων πραγματοποιείται με το Metadefender της εταιρείας OPSWAT. Το Metadefender επιτρέπει τη σάρωση όλων των λήψεων από το internet ή συγκεκριμένων αρχείων, για επιβλαβή προγράμματα χρησιμοποιώντας 40+ εμπορικές μηχανές anti-malware. Η χρήση πολλών μηχανών anti-malware διασφαλίζει ότι περισσότεροι ιοί, spyware, ttrojans και άλλες απειλές μπορεί να ανιχνευθούν. Το μέγιστο μέγεθος των αρχείων που μπορεί να σαρώσει είναι 140MB.

4.4.7.3 VirSCAN

Το VirSCAN είναι μια δωρεάν online υπηρεσία σάρωσης, η οποία ελέγχει τα απεσταλμένα αρχεία για επιβλαβές λογισμικό, χρησιμοποιώντας τις μηχανές antivirus, που υπάρχουν στην VirSCAN λίστα. Το VirSCAN σαρώνει και ελέγχει τα αρχεία για τις εξής συγκεκριμένες κατηγορίες ιούς, trojans, backdoors, spyware, dialers, key-logger. Το μέγιστο μέγεθος των αρχείων που μπορεί να σαρώσει είναι 20MB.

4.4.7.4 VirusCheckMate

Το VirusCheckMate είναι ένα online εργαλείο ελέγχου του περιεχόμενου ενός αρχείου για ανίχνευση των ιών, worms, trojans, και κάθε είδους επιβλαβές λογισμικό. Επίσης μπορεί να πραγματοποιήσει και σάρωση ιστοσελίδων. Χρησιμοποιεί 43 μηχανές αναζήτησης. Το μέγιστο μέγεθος των αρχείων που μπορεί να σαρώσει είναι 20MB.

5 *Τεχνικές Ανάλυσης Επιβλαβούς Λογισμικού*

Η ανάλυση malware ορίζεται ως «η τέχνη του κατακερματισμού του κακόβολου λογισμικού ώστε να κατανοηθεί πώς λειτουργεί, πώς μπορεί να αναγνωριστεί, και πώς να αντιμετωπιοθεί ή να εξαλειφθεί». Για να είναι αποτελεσματική μια τέτοια ανάλυση απαιτούνται μηχανισμοί ανίχνευσης μεγάλης ακρίβειας. Αυτοί περιλαμβάνουν κλασικές προσεγγίσεις βασιζόμενες στις δυαδικές υπογραφές και σε τεχνικές ανίχνευση συμπεριφοράς κ.α.

Η ανάλυση επιβλαβούς λογισμικού είναι μια σημαντική διαδικασία καθώς μπορεί να βοηθήσει στην εύρεση της κατάλληλης τεχνικής για την αφαίρεση του επιβλαβούς λογισμικού σε περίπτωση παραβίασης της ασφάλειας ή να αποκαλύψει το στόχο του επιβλαβούς λογισμικού. Επί του παρόντος, τα επιβλαβή λογισμικά χρησιμοποιούν μια σειρά από μεθόδους για να ματαιώσουν την ανάλυση, όπως επεκτείνοντας τον χρόνο ενεργοποίησης τους και την απόκρυψη τους στόχους τους. Αυτές οι τεχνικές αντίστασης στην ανάλυση περιλαμβάνουν κωδικό συσκότισης (code obfuscation), κώδικές αυτοελέγχου (self-checking) και αυτό-τροποποίηση (self-modifying), πολυμορφισμό, μεταμόρφωση και άλλα

Σκοπός της ανάλυσης ενός επιβλαβούς λογισμικού είναι η απάντηση στα εξής ερωτήματα:

Ποιος είναι ο σκοπός του επιβλαβούς λογισμικού;

Πώς εισέρχεται σε ένα σύστημα;

Τι ενέργειες πραγματοποιεί μετά την εκτέλεση του;

Πώς γίνεται η διάδοσή του από σύστημα σε σύστημα;

Πότε και πώς δημιουργήθηκε το επιβλαβές λογισμικό και αν βασίζεται σε γνωστές βιβλιοθήκες;

Χρησιμοποιεί μεθόδους για την ματαίωση της ανάλυσής του και ποιες.

Για την ανάλυση επιβλαβούς λογισμικού συνήθως χρησιμοποιούνται δύο μέθοδοι η στατική και δυναμική ανάλυση. Η διάκριση μεταξύ των δύο τεχνικών είναι ότι η δυναμική ανάλυση παρατηρεί την κακόβουλη συμπεριφορά, εκτελώντας ένα δείγμα κώδικα, ενώ στη στατική ανάλυση δεν εκτελείται ο κώδικας. Παρόλο που οι τεχνικές που χρησιμοποιούνται στις δύο αναλύσεις είναι διαφορετικές, υπάρχει ένας αριθμός

μεθόδων και εργαλείων που εξυπηρετούν τους ίδιους στόχους της ανάλυσης επιβλαβούς λογισμικού.

5.1 Στατική ανάλυση

Η στατική ανάλυση περιγράφει την διαδικασία ανάλυσης του κώδικα ή της δομής ενός προγράμματος ώστε να καθορίσει τις λειτουργίες του. Κατά τη διάρκεια αυτής της ανάλυσης δεν εκτελείται ο κώδικας. Γίνεται δηλαδή προσπάθεια να κατανοηθεί πώς ενεργεί το επιβλαβές λογισμικό, μόνο με στοιχεία που λαμβάνονται χωρίς την αλληλεπίδραση του με το σύστημα. Η στατική ανάλυση έχει δύο στάδια: το απλό και το προχωρημένο. Η προχωρημένη στατική ανάλυση αφορά στην εισαγωγή του εκτελέσιμου αρχείου σε έναν disassembler, όπου παρατηρούμε κι εξετάζουμε των κώδικά του. Η στατική ανάλυση μπορεί να βοηθήσει στην αξιολόγηση των σφαλμάτων μνήμης και μπορεί να βελτιώσει την ορθότητα της εκτέλεσης του προγράμματος.

Η λήψη χρήσιμων πληροφοριών από το εκτελέσιμο αρχείο μπορεί να γίνει με διάφορους τρόπους και εργαλεία. Στο κεφάλαιο αυτό θα αναφερθούν επιγραμματικά οι τεχνικές που υπάρχουν και οι πληροφορίες που μπορούμε να πάρουμε από κάθε τεχνική.

Οι τεχνικές που θα αναφερθούν είναι:

Η χρήση μηχανών antivirus για την επιβεβαίωση επιβλαβούς λογισμικού.

Η χρήση της τεχνικής του κατακερματισμού (hashing) για την αναγνώριση επιβλαβούς λογισμικού

Η λήψη πληροφοριών από τα αρχεία αλφαριθμητικών, τεχνικών packing ή obfuscation, επικεφαλίδων, συνδεδεμένων συναρτήσεων και βιβλιοθηκών.

Κάθε τεχνική παρέχει διαφορετικές πληροφορίες και η χρήση της κάθε μίας εξαρτάται από τους στόχους που θέλουμε να πετύχουμε και την στατική ανάλυση. Συνήθως εφαρμόζονται όλες οι παραπάνω μέθοδοι καθώς θέλουμε να συλλέξουμε όσο περισσότερες πληροφορίες μπορούμε για το επιβλαβές λογισμικό.

5.1.1 Χρήση antivirus

Ένα χρήσιμο πρώτο βήμα είναι να ελέγξουμε το επιβλαβές αρχείο σε διάφορες μηχανές antivirus για να δούμε αν έχει αναγνωριστεί και καταχωρηθεί νωρίτερα. Τα προγράμματα antivirus βασίζονται κυρίως στη βάση δεδομένων που έχουν ώστε να αναγνωρίζουν συγκεκριμένα κομμάτια κώδικα που θεωρούνται ύποπτα. Επίσης

με την παρατήρηση της συμπεριφοράς του προγράμματος και συγκρίνοντάς τη με ορισμένα γνωστά μοτίβα μπορούν να εντοπίσουν το επιβλαβές λογισμικό. Ο τρόπος λειτουργίας των antivirus έχει παρουσιασθεί αναλυτικά στο κεφάλαιο 4.

5.1.2 Κατακερματισμός (Hashing)

Το hashing είναι μέθοδο που χρησιμοποιείται ώστε να αναγνωρίσει μοναδικά ένα επιβλαβές λογισμικό. Η συνάρτηση κατακερματισμού, είναι μια μαθηματική συνάρτηση που δέχεται ως είσοδο κάποιο δεδομένο τυχαίου μεγέθους και αυτός βάσει των δεδομένων εισόδου, επιστρέφει ένα αλφαριθμητικό συγκεκριμένου πάντοτε μήκους. Οι τιμές που επιστρέφει η συνάρτηση κατακερματισμού ονομάζονται τιμές κατακερματισμού (hash values), κώδικες κατακερματισμού (hash codes), αθροίσματα κατακερματισμού (hash sums) ή απλά τιμές κατακερματισμού (hashes). Επομένως όταν ένα εκτελέσιμο αρχείο επιβλαβούς λογισμικού εισαχθεί σε ένα πρόγραμμα κατακερματισμού η τιμή κατακερματισμού που επιστρέφεται χαρακτηρίζει μοναδικά το επιβλαβές λογισμικό (σαν δακτυλικό αποτύπωμα) ακόμα και αν το όνομα του επιβλαβούς λογισμικού είναι διαφορετικό.

Αλγόριθμοι κατακερματισμού υπάρχουν διάφοροι. Ο γνωστότερος αλγόριθμός που χρησιμοποιείται για την ανάλυση επιβλαβούς λογισμικού είναι ο MD5 (Message-Digest Algorithm) αλλά και ο SHA-1 (Secure Hash Algorithm) είναι επίσης γνωστός. Αφού παραχθεί η μοναδική τιμή κατακερματισμού με τη χρήση του αλγορίθμου μπορούμε κατόπιν να ενεργήσουμε ως εξής:

Να τη χρησιμοποιήσουμε ως "ετικέτα" την αναγνώρισή του

Να αναζητήσουμε πληροφορίες από το διαδίκτυο προκειμένου να μάθουμε αν έχει ήδη αναγνωριστεί από άλλους αναλυτές

Να την διαμοιράσουμε σε άλλους αναλυτές, ώστε να τους βοηθήσουμε στην αναγνώριση και ταυτοποίηση του υπό εξέταση επιβλαβούς προγράμματος.

5.1.3 Αναζήτηση αλφαριθμητικών

Ένα αλφαριθμητικό (string) είναι μια ακολουθία χαρακτήρων. Στα προγράμματα συνήθως εμπεριέχονται διάφορα αλφαριθμητικά για την εκτύπωση μηνυμάτων στο χρήστη, τη σύνδεση σε ένα συγκεκριμένο URL ή σε μια διεύθυνση IP, την εκτέλεση ενεργειών σε αρχεία με συγκεκριμένη διαδρομή στο δίσκο κ.α. Εξετάζοντας τα αλφαριθμητικά στοιχεία, μεταξύ άλλων, μπορούμε να πάρουμε χρήσιμες πληροφορίες και να αποκτήσουμε μια εικόνα για το τι μπορεί να κάνει ένα επιβλαβές λογισμικό. Παρόλο που όταν ένα πρόγραμμα έχει μεταγλωττιστεί ο αρχικός, πηγαίος

κώδικας, δεν είναι διαθέσιμος, τα strings παραμένουν εντός του εκτελέσιμου και μπορούν ν' ανακτηθούν. Υπάρχουν προγράμματα Strings που αναζητούν στο εκτελέσιμο αρχείο αλφαριθμητικά που συνήθως αποθηκεύονται σε μορφή ASCII ή Unicode. Όμως στα αποτελέσματα εμφανίζονται και αλφαριθμητικά που δεν είναι μηνύματα προς τον χρήστη αλλά αναπαριστάνουν στοιχεία του υπολογιστή όπως διευθύνσεις μνήμης, εντολές προς τη CPU κ.ά., που δεν είναι στην ουσία "πραγματικά" strings. Η διαλογή των αλφαριθμητικών που μας ενδιαφέρουν σε μια τέτοια ανάλυση γίνεται κρατώντας τα αλφαριθμητικά που σχηματίζουν γνωστές λέξεις καθώς αφορούν μηνύματα για τον χρήση. Άλλες ακολουθίες αλφαριθμητικών που δεν είναι αναγνωρίσιμες μπορούμε να τις προσπερνάμε γρήγορα.

5.1.4 Packed και Obfuscated επιβλαβών λογισμικών

Οι προγραμματιστές επιβλαβούς λογισμικού χρησιμοποιούν συχνά τεχνικές όπως το packing ή obfuscation ώστε να κάνουν τα αρχεία τους πιο δύσκολα να αναγνωριστούν κατά την ανάλυση. Τα obfuscated προγράμματα είναι εκείνα που ο δημιουργός τους έχει προσπαθήσει να κρύψει τη λειτουργικότητα τους, ενώ τα packed προγράμματα είναι ειδική περίπτωση των obfuscated όπου το επιβλαβές πρόγραμμά έχει συμπιεστή και δεν μπορεί να αναλυθεί. Και οι δύο τεχνικές θα αποδυναμώσουν σε μεγάλο βαθμό τις προσπάθειες στατικής ανάλυσης. Για παράδειγμα ενώ τα προγράμματα συνήθως περιλαμβάνουν πολλά αλφαριθμητικά, τα επιβλαβή λογισμικά τα οποία είναι packing ή obfuscation περιέχουν πολύ λίγα αλφαριθμητικά και δεν μπορεί υπάρχουν αποτελέσματα κατά την αναζήτηση τους.

Όταν ένα εκτελέσιμο γίνεται packed, ενσωματώνεται στην αρχή του ένα δεύτερο πρόγραμμα. Δουλειά του είναι η αποσυμπίεση του κώδικα, όταν το πακεταρισμένο αρχείο εκτελείται. Αποτέλεσμα της διαδικασίας του packing είναι να αποκρύπτονται τα εμφανή στοιχεία του επιβλαβούς αρχείου, όπως, π.χ., τα strings ή κλήσεις συναρτήσεων. Υπάρχουν εργαλεία όπως το PEiD και το PE Detective όπου μπορούν να ανιχνεύσουν αν έχει γίνει packed και τον τύπο του packed που έχει χρησιμοποιηθεί κατά την συμπίεση.

5.1.5 Ανάλυση επικεφαλίδων

Η μορφή του αρχείου μπορεί να αποκαλύψει πολλά για το πρόγραμμα και τη λειτουργικότητα του. Η μορφή εκτελέσιμού αρχείου χωρίς εγκατάσταση (Portable Executable File Format - PE) χρησιμοποιείται από τα Windows και τα DLLs. Η μορφή PE είναι μια δομή δεδομένων που περιέχει τις απαραίτητες πληροφορίες για

το λειτουργικό σύστημα των Windows για τη διαχείριση και εκτέλεση του κώδικα του προγράμματος.

Τα PE αρχεία ξεκινούν με μια επικεφαλίδα που περιλαμβάνει πληροφορίες σχετικά με τον κώδικα, ο τύπος της εφαρμογής, τις απαιτούμενες λειτουργίες της βιβλιοθήκης, και τις απαιτήσεις χώρου. Οι πληροφορίες στην κεφαλίδα PE έχουν μεγάλη αξία στην ανάλυση επιβλαβούς λογισμικού.

5.1.6 Συνδεδεμένες βιβλιοθήκες και συναρτήσεις

Μια από τις πιο χρήσιμες πληροφορίες που μπορούμε να ανακτήσουμε από ένα εκτελέσιμο αρχείο είναι η λίστα των συναρτήσεων που εισάγει. Οι εισαγόμενες συναρτήσεις είναι συναρτήσεις που χρησιμοποιεί το πρόγραμμα που στην πραγματικότητα είναι αποθηκευμένες σε διαφορετικά προγράμματα όπως οι βιβλιοθήκες. Κατά κανόνα, την ώρα της εκτέλεσής του, το λογισμικό, θα χρησιμοποιεί κάποιες συναρτήσεις από το σύστημα, οι οποίες προσφέρουν ειδικές λειτουργίες όπως, π.χ., δημιουργία αρχείου. Οι βιβλιοθήκες μπορούν να συνδεθούν με το κυρίως εκτελέσιμο πρόγραμμα με linking. Οι χρήση συνδεδεμένων συναρτήσεων γίνεται για να μην χρειάζεται ο προγραμματιστής να δημιουργήσει την συνάρτηση από την αρχή. Οι βιβλιοθήκες μπορούν να συνδεθούν με το λογισμικό στατικά, κατά την διάρκεια εκτέλεσης ή δυναμικά.

Η στατική διασύνδεση χρησιμοποιείται κυρίως στα προγράμματα UNIX και LINUX και ολόκληρος ο κώδικας της συνάντησης αντιγράφεται από το τη βιβλιοθήκη στο εκτελέσιμο. Αποτέλεσμα αυτής της αντιγραφής είναι η δυσκολία αναγνώρισης της συνδεδεμένης συνάρτησης από τον υπόλοιπο κώδικα και η δημιουργία εκτελέσιμων μεγάλου μεγέθους.

Η σύνδεση κατά την διάρκεια της εκτέλεσης χρησιμοποιείται σε επιβλαβή λογισμικά κυρίως όταν χρησιμοποιούν τεχνικές packing ή obfuscation και η σύνδεση με την βιβλιοθήκη γίνεται μόνο όταν απαιτείται η συνάρτηση να εκτελεστεί.

Τέλος κατά την δυναμική διασύνδεση το λειτουργικό σύστημα ψάχνει να βρει τις κατάλληλες συναρτήσεις από τις βιβλιοθήκες κατά τη φόρτωση ενός προγράμματος που χρησιμοποιεί συναρτήσεις οι οποίες εμπεριέχονται σε αυτά. Όταν ακολούθως το πρόγραμμα κάνει χρήση της εκάστοτε συνάρτησης, αυτή εκτελείται μέσω της αντίστοιχης βιβλιοθήκης. Από όλες τις μεθόδους σύνδεσης, η δυναμική σύνδεση είναι η πιο κοινή και η πιο ενδιαφέρουσα για ανάλυση. Το αρχείο PE αποθηκεύει στην επικεφαλίδα του πληροφορίες για κάθε βιβλιοθήκη που θα φορτωθεί και για

κάθε λειτουργία που θα χρησιμοποιηθεί από το πρόγραμμα, επομένως ο προσδιορισμός τους είναι ιδιαίτερα σημαντικός, διότι μας επιτρέπει να συμπεράνουμε τι κάνει το πρόγραμμα.

Υπάρχουν προγράμματα όπως το Dependency Walker που εμφανίζουν τις βιβλιοθήκες και τις συναρτήσεις που καλεί το πρόγραμμά μας κατά την εκτέλεσή τους.

5.2 Προχωρημένο στάδιο στατικής ανάλυσης

Η ανάλυση αυτή στηρίζεται στη λογική της αντίστροφης-μηχανικής (reverse-engineering) του επιβλαβούς κώδικα και είναι ιδιαίτερα πολύπλοκή καθώς στο στάδιο αυτό της στατικής ανάλυσης γίνεται εξέταση του κώδικα μηχανής του επιβλαβούς λογισμικού. Για την επιτυχημένη ανάλυση είναι απαραίτητη η γνώση των διαδικασιών συναρμολόγησης (assembly) και αποσυναρμολόγησης (disassembly) ενός κώδικα. Η διαδικασία που ακολουθείτε στο στάδιο αυτό είναι η χρησιμοποίηση ενός αποσυναρμολογητή (disassembler) για να δημιουργήσει τον κώδικα μηχανής όπου θα μπορέσουμε να τον διαβάσουμε και να τον αναλύσουμε ώστε να καταλάβουμε πως λειτουργεί το επιβλαβές λογισμικό. Κατά το disassembly θα μπορέσουμε να αναγνωρίσουμε τις συναντήσεις που εκτελούνται και τις τοπικές μεταβλητές που χρησιμοποιούνται ενώ θα είναι δυνατή και η ανάλυση στοίβας. Τις περισσότερες φορές τα επιβλαβή λογισμικά δημιουργούνται με γλώσσες προγραμματισμού υψηλού επιπέδου και χρησιμοποιούν μεταγλωττιστή για τη δημιουργία κώδικα μηχανής για να εκτελεστεί το πρόγραμμα από την CPU.

Όταν το επιβλαβές λογισμικό είναι αποθηκευμένο σε ένα δίσκο, είναι συνήθως σε δυαδική μορφή σε επίπεδο κώδικα μηχανής. Ο κώδικας μηχανής είναι η μορφή του κώδικα που ο υπολογιστής μπορεί να τρέξει γρήγορα και αποτελεσματικά. Στην αποσυναρμολόγηση βάζουμε ως είσοδο το επιβλαβές λογισμικό σε δυαδική μορφή και παράγεται από τον disassembler κώδικα γλώσσας assembly ως έξοδο. Η γλώσσα Assembly είναι στη ουσία μια γλώσσα προγραμματισμού που κάθε διάλεκτό της χρησιμοποιείται για να προγραμματίζει οικογένειες μικροεπεξεργαστών. Ο πιο γνωστός disassembler είναι: IDA Pro (Interactive Disassembler Professional).

Ένας σημαντικός δείκτης στην στατική ανάλυση είναι η μέτρηση της αβεβαιότητας σε μια σειρά bytes. Το μέγεθος αυτό ονομάζεται μέση ποσότητα πληροφορίας ή αλλιώς Εντροπία. Αν X είναι μια διακριτή τυχαία μεταβλητή με δειγματοχώρο X =

{x_1, x_2, ..., x_n} και συνάρτηση πιθανότητας μάζας $p(x_i)$, τότε η εντροπία της X, -H(X)- δίνεται από τη σχέση:

$$H(X) = -\sum_{i=1}^{n} p(x_i) \log p(x_i)$$

Η τιμή της εντροπίας φράζεται στο διάστημα [0, 8]. [35]

5.3 Δυναμική ανάλυση

Στην δυναμική ανάλυση όπως αναφέρθηκε και παραπάνω το επιβλαβές λογισμικό εκτελείται και η ανάλυση περιλαμβάνει κυρίως την παρατήρηση της συμπεριφοράς του εντός του περιβάλλοντος εκτέλεσης. Η παρατήρηση της συμπεριφοράς μπορεί να είναι οι αλλαγές στη registry των Windows, οι απόπειρες σύνδεσης στο Διαδίκτυο, τα αρχεία που δημιουργεί ή τροποποιεί, τα δεδομένα που στέλνει κ.α. Για να μην καταστρέψουμε τον υπολογιστή μας ή διαδώσουμε, κατά λάθος, το επιβλαβές λογισμικό σε άλλους υπολογιστές του δικτύου μας πρέπει πρώτα να δημιουργήσουμε ένα ασφαλές περιβάλλον.

Υπάρχουν δύο τρόποι για την δημιουργία ασφαλούς περιβάλλοντος, ο ένας είναι να έχουμε έναν απομονωμένο υπολογιστή μόνο για αυτή την εργασία. Όμως η έλλειψη σύνδεσης στο διαδίκτυο μπορεί να μην αφήσει το επιβλαβές λογισμικό να αναπτυχθεί κανονικά καθώς η σύνδεση με το διαδίκτυο για ενημερώσεις, διοίκηση και έλεγχο μπορεί να είναι απαραίτητη. Ο άλλος τρόπος είναι η χρήση μιας εικονικής μηχανής (Virtual Machine, VM), όπου μπορούμε εύκολα να κατασκευάσουμε χρησιμοποιώντας το VMware Workstation ή το VirtualBox. Ένα σημαντικό πλεονέκτημα στη χρήση VM είναι η δυνατότητα για λήψη στιγμιότυπων (snapshots), όπου αποθηκεύει, στην ουσία, την τρέχουσα κατάσταση της εκάστοτε μηχανής στον δίσκο του αληθινού υπολογιστή, με όλες τις ρυθμίσεις, τα αρχεία, τις ιδιότητές και μπορούμε οποιαδήποτε στιγμή να κάνουμε επαναφορά στην κατάσταση της επιλογής μας. Η χρήση VM για την δυναμική ανάλυση είναι η πιο συνηθισμένη αλλά έχει το μειονέκτημα ότι το επιβλαβές λογισμικό μπορεί να ανιχνεύσει ότι εκτελείται μέσα σε μια εικονική μηχανή με συνέπεια τη διαφορετική του εκτέλεση απ' ότι σε ένα κανονικό μηχάνημα.

Η δυναμική ανάλυση πρέπει να πραγματοποιείται μετά τη στατική ανάλυση που περιεγράφηκε παραπάνω. Η δυναμική ανάλυση, εφόσον περιλαμβάνει την εκτέλεση του malware, θέτει το σύστημα αλλά και το τοπικό μας δίκτυο σε κίνδυνο. Επομένως, πριν το εκτελέσουμε, θα ήταν καλό να έχουμε τουλάχιστον μια ιδέα για

το τι πρόκειται να συμβεί. Επίσης, ένα επιβλαβές πρόγραμμα θα μπορούσε να δέχεται παραμέτρους όταν εκτελείται σε γραμμή εντολών, οι οποίες ενεργοποιούν διαφορετικές λειτουργίες. Χωρίς τη στατική ανάλυση θα ήταν απίθανο να μαντέψουμε ποιες είναι αυτές οι παράμετροι ώστε να το αναγκάσουμε να εκτελέσει όλες τις λειτουργίες του.

Κατά την δυναμική ανάλυση χρησιμοποιούμε εργαλεία που μας βοηθούν να εντοπίσουμε τις λειτουργίες του επιβλαβές λογισμικού. Τα εργαλεία αυτά ελέγχουν τις μεταβολές που δημιουργεί η εκτέλεση του λογισμικού στο σύστημα. Τα στοιχεία του συστήματος που ελέγχουμε είναι:

Η Registry, με την εξέταση της registry διαπιστώνουμε τις αλλαγές στις ρυθμίσεις του συστήματος που γίνονται από το malware και να καταλάβουμε τον σκοπό που εξυπηρετούν. Για παράδειγμα το επιβλαβές λογισμικό μπορεί να προσθέσει τον εαυτό του στα προγράμματα που κάνουν autorun κατά την εκκίνηση του λειτουργικού.

Τα αρχεία, εξερευνώντας την αλληλεπίδραση του συστήματος αρχείων μπορεί να εντοπιστούν όλα τα αρχεία που το επιβλαβές λογισμικό δημιουργεί ή αρχεία ρυθμίσεων που χρησιμοποιεί. Ένα malware μπορεί να δημιουργεί, να τροποποιεί και να διαγράφει αρχεία του συστήματος. Προφανώς, μας ενδιαφέρει να παρατηρήσουμε τις αλλαγές που πραγματοποιούνται κατά την εκτέλεση του malware. Διατηρώντας μία λίστα των αρχείων του συστήματος μπορούμε να δούμε ποια αρχεία προστέθηκαν, ποια τροποποιήθηκαν και ποια σβήστηκαν.

Οι διεργασίες, εξετάζουμε αν ξεκίνησαν νέες διεργασίες στο σύστημα και αν άλλαξε η κατάσταση σε υπηρεσίες που εκτελούνταν ήδη.

Η προσωρινή μνήμη (RAM). Η ανάλυση της RAM καθώς το επιβλαβές λογισμικό μπορεί να χειρίζεται τη μνήμη με ανορθόδοξους τρόπος.

Εκτός από τα στοιχεία του συστήματος κατά την δυναμική ανάλυση ελέγχουμε και την δραστηριότητα του δικτύου κατά την εκτελέσει του επιβλαβούς λογισμικού. Κατά κανόνα, ένα malware θα προσπαθήσει να συνδεθεί στο Διαδίκτυο σε διάφορες τοποθεσίες για να κάνει ενημερώσεις ή για να στείλει τα στοιχεία που έκλεψε από τον υπολογιστή όπως κωδικούς πιστωτικών καρτών, λογαριασμούς email κ.α. Τα στοιχεία του δικτύου που ελέγχουμε είναι η δικτυακή κίνηση (network traffic). Πρέπει να παρακολουθήσουμε τη δικτυακή κίνηση που σχετίζεται με το malware. Αυτή τη διαδικασία την πραγματοποιούμε με ειδικά εργαλεία που βλέπουν όλα τα πακέτα δεδομένων που έρχονται και φεύγουν από το σύστημα.

Εφόσον έχουμε στην κατοχή μας ένα αντίγραφο της δικτυακής κίνησης που προκλήθηκε από την εκτέλεση του malware, ξεκινάμε τη διαδικασία αναζήτησης ύποπτων στοιχείων. Ως τέτοια θα μπορούσαν να θεωρηθούν οι διευθύνσεις IP από γνωστά επιβλαβή site. Στην αναζήτησή μας αυτή μπορούμε να χρησιμοποιήσουμε υπάρχουσες βάσεις δεδομένων (π.χ. IPVoid). Τέλος μελετάμε τα δεδομένα που προσπαθεί να στείλει και να λάβει το επιβλαβές λογισμικό στις συγκεκριμένες διευθύνσεις. Μ' αυτόν τον τρόπο μπορούμε να εντοπίσουμε τα στοιχεία που συλλέγει το malware από τον υπολογιστή μας.

Μετά την δυναμική ανάλυση εκτός του ότι μπορούμε να περιγράψουμε με ακρίβεια διάφορα χαρακτηριστικά γνωρίσματα του malware μπορούμε επίσης να ανακαλύψουμε και τον τρόπο εξάπλωσης του.

5.4 Προχωρημένο στάδιο δυναμικής ανάλυσης

Στο προχωρημένο στάδιο της δυναμικής ανάλυσης με κατάλληλα εργαλεία κάνουμε debugging στο επιβλαβές λογισμικό. Το debugging είναι ένα κρίσιμο εργαλείο για τη συλλογή πληροφοριών σχετικά με ένα επιβλαβές πρόγραμμα το οποίο μας δίνει πληροφορίες για το λειτουργικό οι οποίες θα ήταν δύσκολο να συλλεχτούν μόνο από την disassembly της προχωρημένης στατικής ανάλυσης. Κάνοντας debugging σε ένα πρόγραμμα μπορούμε να δούμε τη τιμή της κάθε μνήμης και του κάθε καταχωρητή σε όλες τις συναρτήσεις. Θέλει μεγάλη εμπειρία για να μπορέσει κανείς να είναι σε θέση να αναλύσει το επιβλαβές λογισμικό αποτελεσματικά, μέσα από debugging.

Ένα από τα πιο δημοφιλή προγράμματα εντοπισμού σφαλμάτων λειτουργίας για την δυναμική ανάλυση επιβλαβούς λογισμικού είναι το OllyDbg. Με τη βοήθεια του debugger μπορούμε να παρατηρήσουμε τον χάρτη μνήμης και να ελέγξουμε πώς το πρόγραμμα απλώνεται στη μνήμη και να παρατηρήσουμε όλα τα τμήματα της μνήμης του. Επίσης μπορούμε να παρατηρήσουμε τα τρέχοντα νήματα (threads) του προγράμματος και την κατάστασή του (Ενεργή, σε παύση ή αναστολή). Το εργαλείο OllyDbg δεν μπορεί να κάνει debug σε επιβλαβή λογισμικά λειτουργίας πυρήνα, όπως το rootkits και σε προγράμματα οδήγησης συσκευών όπου για την ανάλυση αυτή θα χρειαστούν άλλα προγράμματα όπως το WinDbg.

6 *Αποφυγή Antivirus & Τεχνικών Ανάλυσης*

6.1 Έλεγχος Επιβλαβούς Λογισμικού από AV

Στο 4ο κεφάλαιο αναπτύχθηκαν γενικότερα οι τεχνικές ανάλυσης επιβλαβούς λογισμικού. Ειδικότερα στον τομέα των antivirus, είναι αρκετά γνωστό ότι υπάρχουν δύο βασικοί τρόποι για την ανίχνευση επιβλαβών αρχείων. Ο πρώτος είναι με signatures, όπου όταν ένα αρχείο είναι γνωστό ότι είναι επιβλαβές, προσδιορίζεται με ένα μοναδικό «pattern» και αυτό μπορεί να χρησιμοποιηθεί για τον εντοπισμό του. Ο δεύτερος τρόπος είναι ο έλεγχος της συμπεριφοράς του αρχείο όταν εκτελείται, είτε σε ένα εικονικό είτε σε πραγματικό περιβάλλον. Αν ένας από τους παραπάνω τρόπους κρίνουν ένα αρχείο ως επιβλαβές, τότε το αρχείο μπορεί να επισημανθεί και είτε να μπει σε καραντίνα ή να αφαιρεθεί. Συνεπώς, ένα σύστημα antivirus αξιολογείται κατά πόσο αξιόπιστα μπορεί να διακρίνει πότε ένα λογισμικό είναι επιβλαβές ή όχι.

6.1.1 Τεχνικές Αποφυγής Antivirus

Τον Απρίλιο του 2015 στο Συνέδριο του RSA ο Christopher Kruegel εκπροσωπώντας την Lastline, μια εταιρεία ασφαλείας που εστιάζει στην ανάλυση των προηγμένων επιβλαβών λογισμικών σε πραγματικό χρόνο, εξέδωσε νέα έκθεση σχετικά με το εξελισσόμενο τοπίο των τεχνικών αποφυγής antivirus και τίτλο «Evasive Malware Exposed and Deconstructed». [36]

Η έκθεση αναφέρει ότι, ενώ το 2014 μόνο ένα μικρό ποσοστό του επιβλαβούς λογισμικού χρησιμοποιούσε τεχνικές αποφυγής antivirus, σήμερα μια αρκετά μεγάλη μερίδα χρησιμοποιεί ένα συνδυασμό περισσότερων των 500 τεχνικών με σκοπό να αποφύγουν τον εντοπισμό και την ανάλυση.

Τα antivirus λειτουργούν με την αναγνώριση της υπογραφής έτσι εάν η υπογραφή του ιού παραχθεί, το antivirus εύκολα θα τον εντοπίσει και θα τον εξαλείψει. Αλλά αν ο ιός δεν έχει ανιχνεύεται ακόμη επομένως δεν υπάρχει καμία υπογραφή, τότε το antivirus θα πρέπει να εκτιμήσει αν το αρχείο είναι ένας ιός ή όχι. Η μέθοδος που χρησιμοποιούν τα antivirus είναι η Heuristic (Ευρετική). Επομένως οι τεχνικές αποφυγής των antivirus βασίζονται κυρίως σε μεθόδους απόκρυψης ή αλλαγής της

υπογραφής. Γενικότερα, οι βασικότερες μέθοδοι που χρησιμοποιούνται από ένα επιβλαβές λογισμικό για την αποφυγή antivirus είναι οι εξής:

6.1.1.1 Τεχνικές Συσκότισης Κώδικα (Code Obfuscation Technics)

Για την αποφυγή των antivirus συνήθως στα επιβλαβή λογισμικά χρησιμοποιούν τεχνικές συσκευασίας και συσκότισης ώστε να γίνει πιο δύσκολο το αρχείο να εντοπιστεί και να αναλυθεί. Η τεχνική της συσκότισης χρησιμοποιείται κυρίως από τα επιβλαβή λογισμικά για να κρυφθούν ενώ η συσκευασία είναι υποσύνολο της συσκότιση και χρησιμοποιείται για να συμπιεστεί το πρόγραμμα και να είναι δύσκολο να αναλυθεί. Οι τεχνικές συσκότισης παρέχουν συνήθως συσκότιση της ροής ελέγχου, κωδικοποίηση των String και παραποίηση ονομάτων μεθόδων,

Control Flow not Obfuscated

```
private TreeNode<T> Pair(TreeNode<T> p, TreeNode<T> q)
{
    TreeNode<T> result;
    if (this._comparer.Compare(p.Value, q.Value) < 0)
    {
        p.Right = q.Left;
        q.Left = p;
        p.Parent = q;
        result = q;
    }
    else
    {
        q.Right = p.Left;
        p.Left = q;
        q.Parent = p;
        result = p;
    }
    return result;
}
```

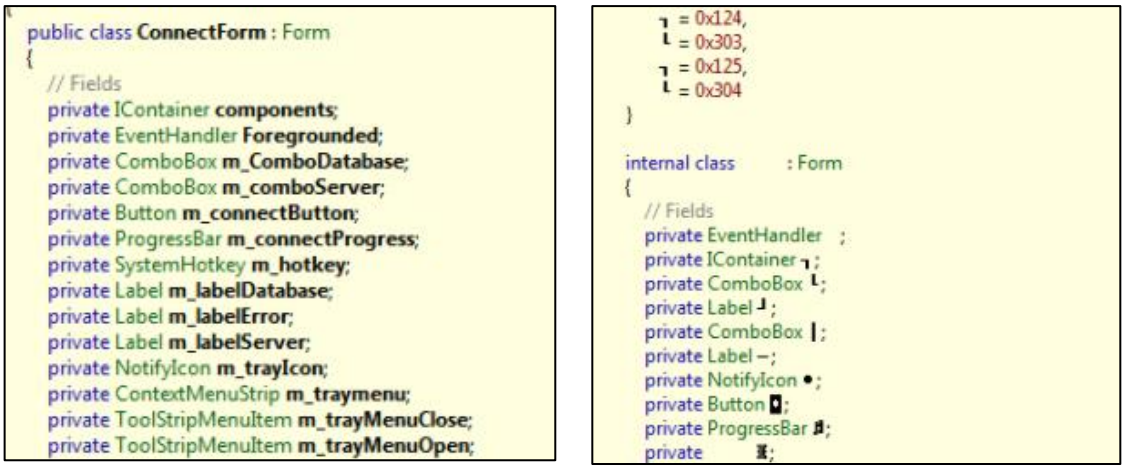
Control Flow Obfuscated

```
private ⍰<⍰> ⍰(⍰<⍰> ⍰, ⍰<⍰> ⍰)
{
    ⍰<⍰> result;
    if (((this.⍰.Compare(⍰.⍰, ⍰.⍰) < ⍰.⍰(1)) ? 1 : 0) != ⍰.⍰(1))
    {
        while (true)
        {
            IL_67:
            int num = ⍰.⍰(26);
            int num2 = -2;
            while (true)
            {
                num2 ^= 72;
                switch (num2 + 76)
                {
                case 0:
                    goto IL_67;
                case 1:
                    switch (num + 73)
                    {
                    case 0:
                        ⍰.⍰ = ⍰;
                        num = -6;
                        goto IL_41;
                    case 1:
                        ⍰.⍰ = ⍰.⍰;
                        num = ⍰.⍰(18);
                        goto IL_41;
```

Εικόνα 2: Παράδειγμα συσκότιση της ροής ελέγχου

συναρτήσεων και πεδίων. Επίσης με τις τεχνικές συσκότισης μπορούν όλα τα DLL αρχεία να συγχωνευτούν έτσι ώστε να μην χρειάζεται να φορτωθούν πολλά αρχεία DLL και επομένως να μπορούν να είναι ασαφή. Ακόμα υπάρχει η δυνατότητα με την συσκότιση να δημιουργεί ένας proxy για κλήσεις προς τις μεθόδους εκτός του προγράμματος ώστε να είναι δύσκολο να βρεθούν οι εξωτερικές κλήσεις.

Με τη χρήση τεχνικών συμπίεσης τα malware τροποποιούν τον κώδικά τους προκειμένου να παρακάμψουν την υπογραφή στην οποία βασίζονται τα antivirus. Η γλώσσα που έχει αναπτυχθεί το επιβλαβές λογισμικό είναι σημαντικός παράγοντας στην δημιουργία συσκοτισμένου κώδικα. Οι C, C++ και Perl είναι οι γλώσσες προγραμματισμού που ευκολά μπορείς να συσκοτίσεις. Στις μέρες μας όμως κυκλοφορούν πολλές εφαρμογές για συσκότιση του κώδικά μερικές από αυτές είναι το Babel Obfuscator, ConfuserEx (ανοιχτού κώδικα και διανέμεται δωρεάν),



Εικόνα 3: Παράδειγμα κώδικα πριν και μετά τη χρήση τεχνικών συσκότισης

SmartAssembly κ.α.

6.1.1.2 Πολυμορφικός Κώδικας (Polymorphic code)

Τα περισσότερα antivirus, όπως αναφέρθηκε και σε προηγούμενο κεφάλαιο προσπαθούν να εντοπίσουν το επιβλαβές λογισμικό εντοπίζοντας μοτίβα που αντιστοιχούν σε γνωστούς ιούς και λαμβάνουν τα κατάλληλα μέτρα. Όταν ένα επιβλαβές λογισμικό κάνει χρήση πολυμορφικού αλγόριθμου τότε γίνεται δύσκολος ο εντοπισμός του επειδή συνεχώς μεταλλάσσεται. Η μετάλλαξη του ιού εμφανίζεται με διαφορετικούς τρόπους όπως η αλλαγή του ονόματος ή η κρυπτογράφηση.

6.2 Αποφυγή Τεχνικών Ανάλυσης

Η αποφυγή της ανάλυσης ενός επιβλαβούς λογισμικού είναι μια πολύ πιο δύσκολή διαδικασία από την αποφυγή των antivirus καθώς την ανάλυση την πραγματοποιούν άνθρωποι και όχι μηχανές. Όπως θα διαπιστωθεί και από τα παρακάτω κεφάλαια η αποφυγή των τεχνικών ανάλυσης σχεδόν ταυτίζεται με το anti-Reverse-Engineering επομένως χρησιμοποιούνται κοινά εργαλεία. Κάθε εταιρία ανάπτυξης λογισμικού

φροντίζει ώστε να δυσκολέψει το reverse-engineering για τον κώδικα της, ώστε να αποτρέψει το ξεκλείδωμα, την παραποίηση ή/και την αντιγραφή αυτού. Είναι αυτονόητο πως και οι μηχανικοί ανάπτυξης ενός malware έχουν την ίδια ακριβώς γνώμη. Σε πρώτη ανάγνωση θα μπορούσαμε να ισχυριστούμε πως οι τεχνικές στατικής και δυναμικής ανάλυσης -που αναφέρονται στο 5ο κεφάλαιο- καθιστούν αδύνατο το anti-reverse-engineering, όμως γρήγορα μπορεί να διαπιστωθεί ότι υπάρχουν τεχνικές που μπορούν να ακολουθηθούν για να κάνει αρκετά δύσκολο το έργο των αναλυτών. Οι τεχνικές αυτές είτε δυσκολεύουν σημαντικά την διαδικασία, είτε την καθυστερούν τόσο, ώστε να μην αξίζει τον κόπο να προχωρήσουν.

6.2.1 Anti-Dumping

Anti-dumping είναι η διαδικασία κατά την όποια γίνεται προσπάθεια ώστε να αποτραπεί η ανάκτηση της καταγραφής της κατάστασης μνήμης ώστε να μη χρησιμοποιηθεί από τους αναλυτές για τον εντοπισμό του τρόπου λειτουργίας του επιβλαβούς λογισμικού.

6.2.1.1 Erase PE header from memory

Αυτή είναι μια απλή τεχνική anti-dumping που αφαιρεί στοιχεία από τη μνήμη της επικεφαλίδας των εκτελέσιμων PE κατά το χρόνο εκτέλεσης. Με τον τρόπο αυτό, στο image θα λείπουν σημαντικές πληροφορίες όπως η RVA σημαντικών πινάκων, το σημείο εισόδου, καθώς και άλλες πληροφορίες που τα Windows χρησιμοποιούν όταν πρόκειται να το φορτώσουν. Αυτή η τεχνική μπορεί να δημιουργήσει προβλήματα καθώς υπάρχουν αλλά προγράμματα και Windows APIs που για να λειτουργήσουν χρειάζονται πρόσβαση σε αυτές τις πληροφορίες που έχουν αφαιρεθεί.

6.2.1.2 SizeOfImage

Η τιμή SizeOfImage στο PEB μπορεί να αλλάξει, έτσι ώστε η διαδικασία πρόσβασης να παρεμποδίζεται, και να διακόπτεται η σύνδεση του debugger με την διεργασία.

6.2.2 Anti-Sandbox

Το Sandbox είναι ένα απομονωμένο υπολογιστικό περιβάλλον ώστε τα προγράμματα ή αρχεία να εκτελούνται σε αυτό χωρίς να υπάρχει καμία επίπτωση στο υπόλοιπο σύστημα. Ένα sandbox είναι στην ουσία ένας μηχανισμός ασφαλείας για την εκτέλεση μη έμπιστων προγραμμάτων σαν θωρακισμένο κουτί. Σε ένα sandbox μπορούν να προστεθούν πολλές λειτουργίες, όπως για παράδειγμα η προσομοίωση

δικτυακών υπηρεσιών, ώστε το εκάστοτε malware να ξεγελαστεί και να πιστέψει ότι εκτελείται σε έναν κανονικό υπολογιστή. Επομένως για το anti-Sanbox το επιβλαβές λογισμικό θα προσπαθήσει να ανιχνεύσει αν πάει να εκτελεστεί σε τέτοιο περιβάλλον και σε περίπτωση που το εντοπίσει ή δεν θα εκτελεστεί ή θα δράσει με εντελώς παραπλανητικό τρόπο, για να μπερδέψει τους αναλυτές. Για την ανίχνευση του sandbox οι δημιουργοί επιβλαβών λογισμικών έχουν βασιστεί στις κύριες αδυναμίες του, που παρουσιάζονται παρακάτω.

6.2.2.1 Ανθρώπινη Αλληλεπίδραση

Ένα sandbox εκτελεί το επιβλαβές πρόγραμμα όπως του το δίνουμε, χωρίς τα command-line options που είναι πιθανό να δέχεται. Επομένως, αν αυτές οι παράμετροι ενεργοποιούν βασικές λειτουργίες του malware δεν θα εκτελεστούν ποτέ. Επίσης η μη ύπαρξη ανθρώπινης δραστηριότητας όπως το πάτημα ενός κουμπιού η χρήσης του ποντικιού είναι ένας σημαντικός παράγοντας ανίχνευσης του sandbox.

6.2.2.2 Χρονισμός

Ένα άλλο κλασικό μειονέκτημα είναι το χρονικό όριο που έχει το εκάστοτε sandbox για την εκτέλεση και την ανάλυση ενός malware. Η χρονική καθυστέρηση που χρησιμοποιείται από τα επιβλαβή λογισμικά για την αποφυγή της ανάλυσης από το sandbox ώστε να μην καταγραφούν όλα τα γεγονότα είναι ίδια με αυτή που χρησιμοποιείται για το anti-debugging που αναλύεται παρακάτω.

6.2.2.3 Injection hooks

Το sandbox δημιουργεί άγκιστρα (hooks) σε ένα πρόγραμμα για να πάρει ειδοποιήσεις για κλήσεις ρουτίνας. Τα άγκιστρα είναι μια αδυναμία που μπορεί να ανιχνευθεί από το επιβλαβές λογισμικό.

6.2.3 Anti-Virtualization

Η επίγνωση του περιβάλλοντος του συστήματος επιτρέπει στο επιβλαβές λογισμικό την ανίχνευση του βασικού περιβάλλοντος εκτέλεσης του συστήματος που προσπαθεί να μολύνει. Το επιβλαβές λογισμικό μπορεί να χρησιμοποιήσει τεχνικές για να καθοριστεί εάν εκτελείται σε εικονική μηχανή (VM) ή πραγματικό συστημικό περιβάλλον. Τα πλέον διαδεδομένα VM είναι το VMware, το VirtualBox, το Parallels και το Windows Virtual PC.

Το VMware αναπτύχθηκε από την εταιρεία VMware, Inc. που είναι εταιρεία που παρέχει λογισμικό εικονικοποίησης. Τα προϊόντα της VMware είναι σχεδιασμένα για λειτουργικά συστήματα Microsoft Windows, Linux και macOS. Τα προϊόντα παρέχουν ένα πλήρως εικονικοποιημένο hardware στο φιλοξενούμενο λειτουργικό σύστημα, με εικονική κάρτα γραφικών, εικονικό σκληρό δίσκο, εικονικούς οδηγούς για τις θύρες USB, τις παράλληλες και τις σειριακές θύρες. Έτσι τα εικονικά μηχανήματα μπορούν να μεταφερθούν από υπολογιστή σε υπολογιστή και να μην έχουν προβλήματα συμβατότητας. Τα προϊόντα της VMware είναι κλειστού-κώδικα με μειωμένα δικαιώματα του φιλοξενούμενου συστήματος.

Το VirtualBox είναι το VM της ORACLE και είναι ανοιχτού κώδικα και διανέμεται δωρεάν. Επί του παρόντος, το VirtualBox τρέχει σε Windows, Linux, Macintosh, και φιλοξενεί Solaris και υποστηρίζει ένα μεγάλο αριθμό λειτουργικών συστημάτων όπως Windows (NT 4.0, 2000, XP, Server 2003, Vista, Windows 7, Windows 8, τα Windows 10), DOS / Windows 3.x, Linux (2.4, 2.6, 3.x και 4.x), Solaris και OpenSolaris, OS / 2, και το OpenBSD.

Το Parallels Desktop για Mac, από την Parallels, είναι λογισμικό που παρέχει virtualization για υπολογιστές Macintosh με επεξεργαστές Intel. Το Parallels Desktop για Mac χρησιμοποιώντας την τεχνολογία hypervisor, λειτουργεί με τη χαρτογράφηση των πόρων του υλικού του κεντρικού υπολογιστή άμεσα με τους πόρους της εικονικής μηχανής. Έτσι, κάθε εικονική μηχανή λειτουργεί με τον ίδιο τρόπο σε έναν αυτόνομο υπολογιστή, με σχεδόν όλους τους πόρους του φυσικού μηχανήματος.

Το Windows Virtual PC είναι ένα πρόγραμμα virtualization για τα Microsoft Windows που έχει αναπτυχθεί από την Microsoft. Η πιο πρόσφατη έκδοση, το Windows Virtual PC, δεν λειτουργεί σε εκδόσεις των Windows παλαιότερες από τα Windows 7, και δεν υποστηρίζει συστήματα MS-DOS ή λειτουργικά νωρίτερα από τα Windows XP Professional SP3. Το Windows Virtual PC μπορεί να επιτρέψει στο φιλοξενούμενο λειτουργικό να αλληλεπιδρά με το λειτουργικό τους σύστημα υποδοχής και να γίνεται κοινή χρήση φυσικών συστατικών του υλικού ή ανταλλαγή δεδομένων.

Η ανίχνευση από το κακόβολο λογισμικό αν πάει να εκτελεστεί σε εικονική μηχανή γίνεται με τον ίδιο τρόπο για όλα τα VM. Όλα τα VM χρησιμοποιούν ένα κλειδί μητρώου για τη θέση εγκατάστασης τους (πχ HKLM\Software\VMware, Inc.\VMware Tools\InstallPath), το Malware μπορεί να ελέγξει για την παρουσία

του κλειδιού αυτού ώστε να συμπεριφερθεί ανάλογα. Μια άλλη τεχνική είναι να χρησιμοποιηθεί το Windows Management Instrumentation (WMI) ώστε να γίνει έλεγχος αν κάποια από τις διευθύνσεις MAC που χρησιμοποιούνται ανήκουν σε VM. Ο έλεγχος αυτός γίνεται με ένα απλό ερώτημα WMI για έλεγχο στις εγκατεστημένες συσκευές. Τέλος ένα VM μπορεί να γίνει αντιληπτό στην προεπιλεγμένη του διαμόρφωση από το επιβλαβές λογισμικό μέσω του καταλόγου υλικού.

6.2.4 Τεχνικές Injection

Η διεργασία injection χρησιμοποιείται για την έγχυση του κώδικα σε μια άλλη ενεργή διεργασία, όπου τελικά το επιβλαβές λογισμικό εκτελείται από εκεί. Με τον τρόπο αυτό μπορούν να παρακαμφθούν ειδική μηχανισμοί ασφάλειας που βασίζονται σε firewalls.

6.2.4.1 SetWindowsHooksEx

Ο μηχανισμός Hooks των Windows μπορεί να χρησιμοποιηθεί, και με την κλήση της SetWindowsHookEx που επιτρέπει στη διεργασία «στόχο» να φορτώσει ένα συγκεκριμένο αρχείο dll στο χώρο μνήμης του εκτελέσιμου και να επιλεγεί μια συνάρτηση ως Hook για να χειριστεί ένα συγκεκριμένο γεγονός. Όταν γίνει το γεγονός αυτό, η διεργασία «στόχος» εκτελεί τον επιβλαβές κώδικα.

6.2.4.2 CreateRemoteThread

Μια άλλη μέθοδος είναι η χρήση της injection σε μία βιβλιοθήκη. Ένα καινούριο thread δημιουργείται με την CreateRemoteThread στην διεργασία που χρησιμοποιείται για να φορτώσει την κακόβουλη βιβλιοθήκη.

6.2.5 Anti-Disassembly και Anti-Debugging

Στην προχωρημένη στατική ανάλυση γίνεται εισαγωγή του εκτελέσιμου αρχείου σε έναν Disassembler όπου λαμβάνοντας τον κώδικα μηχανής τον επιστρέφει σε γλώσσα αναγνώσιμή από τον άνθρωπο ώστε να μπορεί να εξεταστεί ο κώδικας. Με το debugging γίνεται παρατήρηση της συμπεριφοράς του προγράμματος που εκτελείται κατά τον χρόνο εκτέλεσής του. Με το debugging μπορεί να εξεταστεί το περιεχόμενο των μεταβλητών στο πρόγραμμά χωρίς να χρειάζεται να εισάγετε πρόσθετες κλήσεις για την έξοδο των τιμών. Σκοπός ενός επιβλαβούς λογισμικού πριν εκτελεστεί είναι να μπορέσει να ανιχνεύσει αν εκτελείται μέσα σε κάποιον disassembler ή αν γίνεται debugging.

6.2.5.1 Προγράμματα Disassembly και Debugging

6.2.5.1.1 WinDbg

Πρόκειται για ένα πρόγραμμα εντοπισμού σφαλμάτων της Microsoft το οποίο μπορεί να χρησιμοποιηθεί για τον εντοπισμό σφαλμάτων πυρήνα και λειτουργίας ενός προγράμματος, για την ανάλυση crash dumps και για την εξέταση των μητρώων της CPU. Το WinDbg δεν είναι πολύ εύκολο στη χρήση και χρειάζεται χρόνος για να προσαρμοστεί κανείς στο περιβάλλον εργασίας και στις κύριες εντολές. Παρόλα αυτά παραμένει ένα από τους δημοφιλέστερους debuggers.

6.2.5.1.2 OllyDbg

Το πρόγραμμα OllyDbg είναι ένας x86 debugger που αναπτύχθηκε από τον Oleh Yuschuk και χρησιμοποιείται για περισσότερο από μια δεκαετία. Παρέχει τη δυνατότητα για ανάλυση malware εν ώρα λειτουργίας του, χρησιμοποιείται συνήθως για ανάλυση malware και για Reverse-Engineering, καθώς είναι εύκολο στη χρήση, έχει πολλά plugins που επεκτείνουν τις δυνατότητες του και διατίθεται δωρεάν. Το OllyDbg, πριν γίνει δημοφιλές για την ανάλυση malware, το χρησιμοποιούσαν για να ξεκλειδώνουν προγράμματα οι λεγόμενοι «Crackers". Η κύρια αυτή επιλογή των αναλυτών όταν έφτασε στην έκδοση 1.1 αγοράστηκε από την Immunity security και μετονομάστηκε σε Immunity Debugger (ImmDbg). Η Immunity security επιδιόρθωσε σφάλματα του OllyDbg, τροποποίησε το GUI και προσέθεσε ένα πλήρως λειτουργικό διερμηνέα Python με APIs. Οι αλλαγές αυτές οδήγησαν ορισμένους χρήστες να αρχίσουν να χρησιμοποιούν το ImmDbg αντί OllyDbg.

6.2.5.1.3 IDA Pro

Το Interactive Disassembler Professional (IDA Pro) είναι ένας εξαιρετικά ισχυρός disassembler διανέμεται από την Hex-Rays. Δύο εκδόσεις IDA Pro είναι εμπορικά διαθέσιμες και οι δύο εκδόσεις υποστηρίζουν x64, η προηγμένη έκδοση υποστηρίζει πολλούς περισσότερους επεξεργαστές κυρίως x86. Το πρόγραμμα IDA Pro υποστηρίζει επίσης διάφορες μορφές αρχείων, όπως Portable Executable (PE), Common Object File Format (COFF), Executable and Linking Format (ELF), και a.out. Η έκδοση IDA Pro διατίθεται και δωρεάν αλλά έχει περιορισμένες λειτουργίες. Τον εν λόγω πρόγραμμα έχει επικρατήσει αφού μπορεί να γίνει disassemble σε ένα ολόκληρο πρόγραμμα εκτελώντας μεταξύ άλλων σημαντικές

εργασίες, όπως εύρεση συναρτήσεων, ανάλυση στοίβας, αναγνώριση τοπικών μεταβλητών κ.α.

6.2.5.2 Τεχνικές Anti-Debugging

6.2.5.2.1 Τεχνικές Δομής PEB

Η Microsoft παρέχει μια σειρά από συναρτήσεις που μας επιτρέπουν να αντιληφθούμε την διαδικασία του debugging. Μια από πιο γνωστές συναρτήσεις είναι η «IsDebuggerPresent» η όποια ελέγχει αν η καλούμενη διεργασία γίνεται debugged από κάποιο debugging πρόγραμμα του χρήστη. Στην ομάδα των "debugging functions" της Microsoft ανήκει και η «CheckRemoteDebuggerPresent», με την οποία γίνεται έλεγχος αν η συγκεκριμένη ή/και επιλεγμένη διεργασία γίνεται debugged. Ουσιαστικά ο έλεγχος γίνεται από την σημαία (flag) «BeingDebugged» στο Process Environment Block (PEB). Το εν λόγω περιβάλλον έχει δεσμευτεί από το λειτουργικό σύστημα για να παρέχει στον χρήστη κάθε πληροφορία που αφορά τις διεργασίες σε πραγματικό χρόνο. Η Microsoft παρέχει μια επίσημη δομή (PEB Structure) με τις πληροφορίες που μπορούμε να πάρουμε από το PEB, σχετικά με διάφορα flags που αφορούν την μνήμη, πληροφορίες διεργασιών της heap, του kernel κλπ. Στο PEB εκτός από το σημείο που αναφέρθηκε, μπορούμε να διαπιστώσουμε αν υπάρχει debugger και από τα heap flags και forceflags. Ανάλογα με το λειτουργικό σύστημα οι τιμές αυτές εμφανίζονται σε διαφορετική θέση. Συγκεκριμένα, στα Windows XP είναι στην διεύθυνση 0x0C για τα 32-bit συστήματα και 0x14 στα 64-bit συστήματα, ενώ -από τα Windows Vista μέχρι και σήμερα- είναι στην διεύθυνση 0x40 για τα 32-bit συστήματα και 0x70 στα 64-bit συστήματα.

6.2.5.2.2 Τεχνικές βιβλιοθήκης Ntdll.dll

Στην βιβλιοθήκη Ntdll.dll βρίσκουμε την «NtQueryInformationProcess» που ανακτά πληροφορίες που είναι σχετικές με μια δεδομένη διεργασία. Η συνάρτηση παίρνει ως όρισμα το handle της διεργασίας και το είδος της πληροφορίας που ζητείται. Ανάμεσα στα είδη της πληροφορίας είναι τα ProcessDebugPort, ProcessDebugFlags και το ProcessDebugObject. Η ProcessDebugPort επιστρέφει 0 εάν στην διεργασία δεν γίνεται debugging, διαφορετικά θα επιστρέψει μια τιμή port. Η ProcessDebugFlags επιστρέφει 0 εάν στην διεργασία δεν γίνεται debugging, διαφορετικά μια τιμή flag. Τέλος, η ProcessDebugObject επιστρέφει 0 εάν στην

διεργασία δεν γίνεται debugging, διαφορετικά επιστρέφει 1 που επιβεβαιώνει την παρουσία ενός handle.

Μια άλλη τεχνική βασίζεται στην ρουτίνα «ZwSetInformationThread» της Microsoft που καθορίζει τις προτεραιότητες των threads (νημάτων) των Windows. Συγκεκριμένα θέτοντας την παράμετρο της "ThreadInformationClass" στην «ThreadHideFromDebugger» την τιμή 0x11 πετυχαίνουμε να κρύψουμε το συγκεκριμένο thread από τους debuggers.

Η NtQueryObject είναι ένα άλλο παράδειγμα anti-debugging. Χρησιμοποιώντας τις κλάσεις ObjectAllTypesInformation και ObjectTypesInformation, μας επιστρέφουν μια λίστα με όλους τους τύπους αντικειμένων μεταξύ των οποίων και εκείνης του debug handle.

Η ntdll συνάρτηση NtYieldExecution (ισοδύναμη συνάρτηση SwitchToThread του kernel32) επιτρέπει στο ισχύον νήμα να εγκαταλείψει το χρονικό του παράθυρο επομένως και τον υπόλοιπό του χρόνου και να επιτρέψει στο επόμενο προγραμματισμένο νήμα να εκτελεστεί. Αν δεν υπάρχει προγραμματισμένο νήμα για εκτέλεση ή όταν το σύστημα είναι απασχολημένο και δεν επιτραπεί να γίνει η αλλαγή, τότε η NtYieldExecution () συνάρτηση επιστρέφει το STATUS_NO_YIELD _PERFORMED (0x40000024) κατάσταση, η οποία προκαλεί τη λειτουργία kernel32 SwitchToThread() να επιστρέψει μηδέν.

6.2.5.2.3 Τεχνικές Handle

Η χρήση της CloseHandle μπορεί να χρησιμοποιηθεί επίσης για την ανίχνευση debugger. Όταν μια διεργασία είναι σε διαδικασία debugging, καλώντας την, με ένα μη έγκυρο handle δημιουργείται ένα exception με την status_invalid_handle να έχει την τιμή 0xC0000008. Εναλλακτικά, μπορούμε να ελέγξουμε και την τιμή της exception_handle_not_closable που πρέπει να είναι 0xC0000235. Τα συγκεκριμένα exceptions μπορούμε να τα αντιληφθούμε, δημιουργώντας μια συνάρτηση χειρισμού εξαιρέσεων ώστε να αντιληφθούμε την παρουσία debugger.

Η UnhandledExceptionFilter() είναι μια συνάρτηση η οποία μας επιτρέπει να αντικαταστήσουμε το top-level exception handler ενός νήματος διεργασίας. Πιο συγκεκριμένα, καλώντας αυτή την συνάρτηση, εάν ένα exception συνέβη σε μια διεργασία που δεν γίνεται debugging σε ένα unhandled exception φίλτρο, το φίλτρο αυτό θα καλέσει την συνάρτηση exception φίλτρου που καθορίζεται από την παράμετρο lpTopLevelExceptionFilter.

Η τεχνική της συνάρτησης OutputDebugString() λειτουργεί βάση της αποτίμησης εάν η OutputDebugString δημιουργεί σφάλματα. Ένα τέτοιο σφάλμα θα συμβεί μόνο στην περίπτωση που δεν υπάρχει ενεργός debugger για την διεργασία. Συνεπώς, αν δεν βρεθεί το σφάλμα θα είμαστε σε θέση να αναγνωρίσουμε την ύπαρξη προγράμματος debugger.

6.2.5.2.4 Τεχνικές με χρήση των Breakpoints

Πολύ γνωστές τεχνικές είναι εκείνες που ελέγχουν για hardware, software και memory breakpoints, μια συνηθισμένη τακτική των επεξεργαστών αρχιτεκτονικής της Intel. Στην περιοχή του hardware ο έλεγχός τους γίνεται με την χρήση των registers DR0-DR7. Στην περιοχή του λογισμικού υπάρχει ο INT3 που δημιουργεί ένα opcode byte (CC) προσδιορίζοντας την κλήση του debug exception handler. Τέλος στα memory breakpoints, συγκαταλέγεται η ανίχνευση των guard pages που βασίζεται στην μίμηση της συμπεριφοράς ενός debugger. Στην εν λόγω περίπτωση αν έχουμε τιμή στην status_guard_page_violation σημαίνει ότι δεν έχουμε ενεργό debugger.

6.2.5.2.5 Software Interrupts

Εκτελώντας interrupt 0x2d, αν δεν γίνεται debugging, θέτει ένα breakpoint exception. Αντίθετα αν υπάρχει debugger, δεν δημιουργείται exception και η εντολή εκτελείτε κανονικά, αφού δεν εκτελείται με trace flag. Αντίστοιχες τεχνικές υπάρχουν με τα interrupts 0x2C, 0xCC, 0xCD & 0x03.

6.2.5.2.6 SeDebugPrivilege

Αν γίνεται debugging και η διεργασία έχει προνόμια SeDebugPrivileges τότε η κλήση της OpenProcess θα είναι επιτυχής. Τα προνόμια SeDebugPrivilege είναι απενεργοποιημένα σε μια διεργασία από προεπιλογή. Τα εργαλεία που κάνουν debugging δίνουν τα προνόμια SeDebugPrivilege στα token πρόσβασής τους, επομένως η διεργασία που γίνεται debugging θα κληρονομήσει τα token πρόσβασης συμπεριλαμβανομένου και τα SeDebugPrivilege. Ο έλεγχος για το αν είναι ενεργοποιημένα τα προνόμια SeDebugPrivilege γίνεται προσπαθώντας να ανοίξουμε το Csrss.exe.

6.2.5.2.7 Parent Process

Με τον έλεγχο της διεργασία Parent της τρέχουσας διεργασίας μπορούμε να καθορίσουμε αν η διεργασία αυτή γίνεται debugging ή όχι καθώς περιμένουμε η

διεργασία Parent να είναι η συνήθης διεργασία explorer.exe που ξεκινάει από τον χρήστη.

6.2.5.3 Timing-Based Αποφυγή

Η timing-based αποφυγή επιτρέπει στο επιβλαβές λογισμικό να τρέχει σε συγκεκριμένες ώρες ή/και μετά από ορισμένες ενέργειες από τον χρήστη/θύμα. Οι επιθέσεις χρονισμού ή ύπνωσης του malware επωφελούνται του γεγονότος ότι ο κώδικας που εκτελείται μέσα από τον debugger πρόκειται να πάρει περισσότερο χρόνο για να εκτελεστεί από ότι όταν δεν τρέχει σε debugger. Υπάρχουν ρουτίνες που μετρούν το χρόνο που έχει παρέλθει και συγκρίνουν τον χρόνο αυτόν με τον τυπικό χρόνο εκτέλεσης. Αν χρειάστηκε περισσότερο χρόνο για να τρέξει από το αναμενόμενο, τότε είναι πιθανόν να τρέχει σε έναν debugger.

Η εντολή RDTSC (Read Time Stamp Counter) μπορεί να χρησιμοποιηθεί πριν και μετά από μια ρουτίνα για να καθορίσει τον χρόνο που έχει παρέλθει. Στην βιβλιοθήκη kernel32.dll υπάρχει μια λειτουργία ονομαζόμενη GetTickCount που επιστρέφει τον αριθμό των msec που έχουν παρέλθει από τότε που ξεκίνησε το σύστημα.

Μια συνάρτηση που χρησιμοποιείται είναι η Sleep ή η NtDelayExecution, όπου το malware ρυθμίζεται να κοιμηθεί για κάποιο χρονικό διάστημα πριν από την έναρξη της κακόβουλης δραστηριότητας.

7 Σχεδιασμός & Υλοποίηση Επιβλαβούς Λογισμικού με Προστασία Ανάλυσης

Στο πλαίσιο της παρούσας εργασίας δημιουργήθηκε ένα malware τύπου Ransomware, αποκλειστικά για εκπαιδευτικούς λόγους. Όπως έχει αναφερθεί και στο 2ο κεφάλαιο, ο τύπος επιβλαβούς λογισμικού Ransomware αφού ενεργοποιηθεί ακούσια από το θύμα/χρήστη, κρυπτογραφεί συγκεκριμένο τύπο αρχείων και στην συνέχεια ζητάει λύτρα για την αποκρυπτογράφησή τους. Το Ransomware που σχεδιάστηκε και υλοποιήθηκε ονομάζεται love4lock και στο εξής θα αναφέρεται στην εργασία με αυτή την ονομασία.

7.1 Σχεδιασμός

7.1.1 Μηχανισμός Μόλυνσης & Ενεργοποίησης

Το love4lock ακολουθώντας την λογική κάθε ransomware, σχεδιάστηκε ώστε να κρυπτογραφεί τα αρχεία του θύματος ζητώντας στην συνέχεια λύτρα για την αποκρυπτογράφηση αυτών. Ο μηχανισμός μόλυνσης μπορεί να πραγματοποιηθεί είτε μέσω ενός fishing email που λαμβάνει το θύμα, είτε γενικότερα μέσω διαδικτύου, αλλά και μέσω κάποιας μνήμης usb. Για την καλύτερη αποτελεσματικότητα θα μπορούσε να χρησιμοποιηθεί ένας Binder που έχει την δυνατότητα να δεσμεύσει ή να συνδυάσει περισσότερα αρχεία του ενός, σε ένα όνομα με συγκεκριμένη επέκταση. Ειδικότερα, τα προγράμματα Binder δημιουργούν μια βιβλιοθήκη κλάσεων που βασίζεται σε XML (DTD, XDR ή XSD). Η εν λόγω βιβλιοθήκη είναι εύκολο να διαβάσει και να γράψει προγραμματιστικά, έγγραφα XML. Με αυτό τον τρόπο δίνει την εντύπωση ότι πρόκειται για ένα αρχείο. Συνεπώς, μπορούμε να δεσμεύσουμε ένα καθαρό αρχείο (π.χ. xls) μαζί με το love4lock και το αποτέλεσμα να είναι τα προγράμματα Antivirus να μην μπορούν να τα ξεχωρίσουν εύκολα, αφήνοντας εκτεθειμένο το σύστημα στο όποιον κίνδυνο.

7.1.2 Αλγόριθμος Επιβλαβούς Λογισμικού

Το love4lock μόλις ενεργοποιηθεί κρυπτογραφεί τα αρχεία του χρήστη που βρίσκονται στο φάκελο My Documents. Η κρυπτογράφηση των αρχείων γίνεται με τον αλγόριθμό κρυπτογράφησης AES (Advanced Encryption Standard) και πιο συγκεκριμένα υλοποιήθηκε με την πιο αντιπροσωπευτικό αλγόριθμο, Rijndael. Ο αλγόριθμος που περιγράφεται από τον AES είναι μια κρυπτογράφηση συμμετρικού κλειδιού, που σημαίνει το ίδιο κλειδί χρησιμοποιείται τόσο για την κρυπτογράφηση και την αποκρυπτογράφηση των δεδομένων.

Για να πραγματοποιηθεί η διαδικασία της κρυπτογράφησης των αρχείων ακολουθήθηκαν τα παρακάτω βήματα. Με την ενεργοποίηση του love4lock το πρόγραμμα εντοπίζει το όνομα του υπολογιστή (Computer Name) που εκτελείται και το όνομα του χρήστη (Username). Οι παραπάνω πληροφορίες είναι απαραίτητες για να εντοπιστεί ο φάκελος-στόχος My Documents. Στη συνέχεια δημιουργείται ένας τυχαίος κωδικός κρυπτογράφησης / αποκρυπτογράφησης που αποτελείται από αλφαριθμητικούς χαρακτήρες και ορισμένα ειδικά σύμβολα. Ο κωδικός- κλειδί χρησιμεύει τόσο στην κρυπτογράφηση των αρχείων όσο και στην αποκρυπτογράφησή τους, επομένως είναι απαραίτητη η αποθήκευσή του ώστε να μπορεί να ανακτηθεί και για να αποκρυπτογραφηθούν τα αρχεία όταν το "θύμα πληρώσει τα λύτρα". Ο παραπάνω κωδικός αποστέλλεται μέσω μιας ειδικά διαμορφωμένης ιστοσελίδας σε μια βάση δεδομένων που τηρεί τον κωδικό κρυπτογράφησης, το όνομα του υπολογιστή και το όνομα του χρήστη, δηλαδή όλα τα απαραίτητα στοιχεία για να σταλεί ο κωδικός στον κατάλληλο χρήστη που θα κάνει την αποκρυπτογράφηση. Το πρόγραμμα μόλις έχει όλα τα παραπάνω αρχεία συνδέεται αυτόματα στην ιστοσελίδα και στέλνει τα στοιχεία χωρίς ο χρήστης να το αντιληφθεί.

Στον υπολογιστή του "θύματος" και συγκεκριμένα στο φάκελο-στόχο My Documents το πρόγραμμα αναζητά όλα τα αρχεία και τους υποφακέλους που υπάρχουν κρυπτογραφώντας τα αρχεία κειμένου (doc, docx, odt), τα υπολογιστικά φύλλα (xls, xlsx, csv), τις φωτογραφίες (jpg, png, psd), τις παρουσιάσεις (ppt, pptx), τις βάσεις δεδομένων (sql, mdb) και τις ιστοσελίδες (php, html, asp, aspx, xml). Όλα τα κρυπτογραφημένα αρχεία έχουν την κατάληξη .l4l. Όταν ολοκληρωθεί η κρυπτογράφηση όλων των αρχείων παράγεται ένα αρχείο κειμένου txt για ενημέρωση του χρήστη για τον τρόπο πληρωμής των λύτρων και την επαναφορά του συστήματος του στην αρχική κατάσταση.

Παράλληλα με το love4lock αναπτύχθηκε και το πρόγραμμα που κάνει την αποκρυπτογράφηση. Το εν λόγω πρόγραμμα και το κλειδί είναι απαραίτητα για να επανέλθουν τα αρχεία στην αρχική του μορφή.

7.1.3 Ενημέρωση Θύματος / Χρήστη

Ο χρήστης/θύμα όπως αναφέρθηκε και παραπάνω ενημερώνεται για την ενεργοποίηση του ιού και την κρυπτογράφηση των αρχείων του από το αρχείο txt με όνομα Read_For_Unlock!.txt. Το Read_For_Unlock! ενημερώνει τον χρήστη για τις ενέργειες που πρέπει να ακολουθήσει με το παρακάτω κείμενο "Your files are encrypted! If you want your files to be decrypted and successfully return your computer to its original state, you have to pay $ 1,000 ransom. Contact us at love4lock@ased.tk. ... But do not worry!!! This ransomware applied for education purposes, you can find the encryption key in KEY.txt"

Όπως αναφέρεται και στο μήνυμα, επειδή το ransomware έχει αναπτυχθεί για εκπαιδευτικούς λόγους ο κωδικός / κλειδί καταγράφεται επιπλέον και σε ένα αρχείο με όνομα KEY.txt που δημιουργείται μέσα στον φάκελο-στόχο My Documents.

7.1.4 Αποφυγή Τεχνικών Ανάλυσης

Όπως αναφέρθηκε και στο κεφάλαιο 6 υπάρχουν πολλές και διαφορετικές τεχνικές που μπορούν να εφαρμοστούν ώστε κατά την ανάλυση το πρόγραμμα να μπερδέψει τον αναλυτή για να μην καταλάβει ότι πρόκειται για ένα επιβλαβές λογισμικό. Το ίδιο ισχύει και για τα antivirus. Οι τεχνικές που επιλέχθηκαν έχουν σχέση με τη γλώσσα προγραμματισμού που είναι γραμμένο το πρόγραμμα και με τις ενέργειες που εκτελεί αυτό.

Επιλέχθηκε το love4lock πριν εκτελεστεί και κρυπτογραφήσει τα αρχεία να κάνει διάφορους ελέγχους στο σύστημα. Αν από τους ελέγχους γίνει αντιληπτό ότι γίνεται ανάλυση του προγράμματος τότε το επιβλαβές λογισμικό δεν εκτελείται και το πρόγραμμα τερματίζει την λειτουργία του επιτυχώς. Οι έλεγχοι που πραγματοποιούνται είναι:

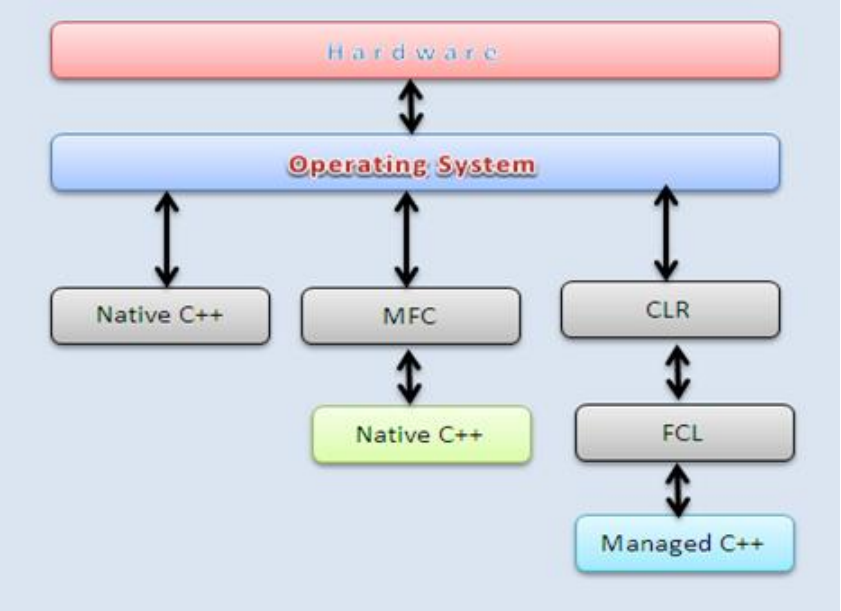
- Έλεγχος αν το πρόγραμμα εκτελείται σε εικονική μηχανή (Virtual Machine).
- Έλεγχος αν το πρόγραμμα εκτελείται μέσα από Sandbox.
- Έλεγχος αν γίνεται προσπάθεια Ανάλυσης ή/και Disassembly
- Έλεγχος αν στο πρόγραμμα προσπαθεί να γίνει Dumping
- Έλεγχος αν στο πρόγραμμα προσπαθεί να γίνει Debugging.
- Έλεγχος για την ύπαρξη Antivirus

Για την αποφυγή διαφόρων μορφών Reverse Engineering έχει γίνει συσκότιση (obfuscation) και πακετάρισμα (packing) του κώδικά με την χρήση κατάλληλων προγραμμάτων. Τέλος, το επιβλαβές λογισμικό γίνεται binding με ένα γνωστό αρχείο ώστε να μπερδέψει τον υποψήφιο χρήστη/θύμα.

7.2 Υλοποίηση

7.2.1 Εργαλεία Προγραμματισμού

Για την υλοποίηση του εν λόγω προγράμματος χρησιμοποιήθηκε η γλώσσα προγραμματισμού C++/CLI [Common Language Infrastructure]. [37] Υπάρχουν δύο θεμελιωδώς διαφορετικά είδη εφαρμογών C++ που μπορούν να αναπτυχθούν. Στη πρώτη κατηγορία ανήκουν εφαρμογές που εκτελούνται εγγενώς στον υπολογιστή, ορίζονται από τα πρότυπα γλωσσών προγραμματισμού ISO / ANSI και αναφέρονται ως προγράμματα γραμμένα σε native C++. Στην δεύτερη κατηγορία δημιουργούνται εφαρμογές να τρέχουν κάτω από τον έλεγχο της CLR [Common Language Runtime] (μια εκτεταμένη έκδοση της C ++), ορίζονται από το πρότυπο γλωσσών προγραμματισμού ESMA και ονομάζονται προγράμματα C ++ / CLI.

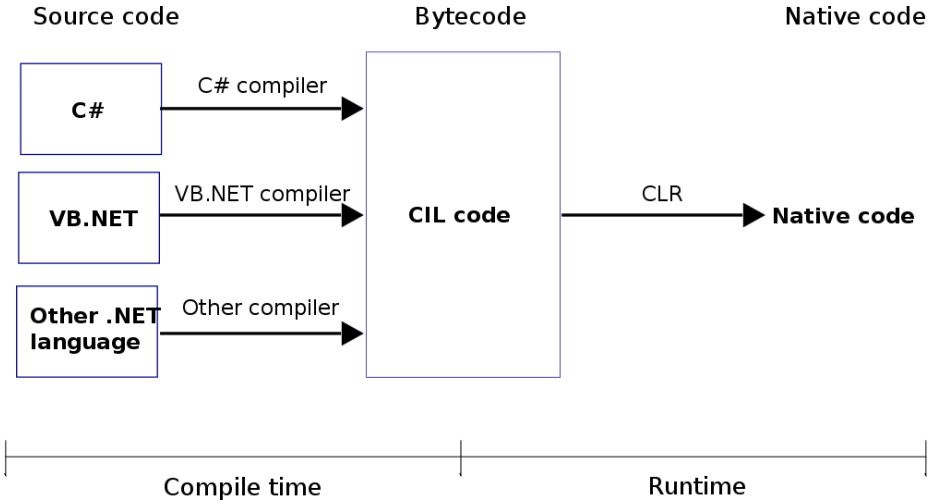


Σχήμα 2: Διάγραμμα C++ / CLI

Η CLR είναι ένα τυποποιημένο περιβάλλον για την εκτέλεση των προγραμμάτων γραμμένο σε ένα ευρύ φάσμα των γλωσσών υψηλού επιπέδου, συμπεριλαμβανομένων των C#, VB και C++.

Η CLI είναι ουσιαστικά μια προδιαγραφή για ένα εικονικό περιβάλλον μηχανή που επιτρέπει σε εφαρμογές γραμμένες σε άλλες γλώσσες υψηλού επιπέδου να

εκτελούνται σε διαφορετικά περιβάλλοντα συστήματος χωρίς την μεταγλώττιση του πηγαίου κώδικα.



Σχήμα 3: Λογική CLR

Ο κώδικας C++ που εκτελείται με την βοήθεια της CLR περιγράφεται ως managed C++, διότι τα δεδομένα και ο κώδικας διαχειρίζονται από την CLR. Έτσι με αυτόν τον τρόπο μπορεί εύκολα να απελευθερωθεί αυτόματα μνήμη που έχει δυναμικά δεσμευτεί για την αποθήκευση δεδομένων, εξαλείφοντας τα κυριότερα προβλήματα που εμφανίζονται στην native C++. Το μεγάλο πλεονέκτημα της C++/CLI είναι η ικανότητα της ανάμιξης native & managed C++. Συνεπώς, χρησιμοποιήθηκαν νεότερες βιβλιοθήκες και μηχανισμοί από την γλώσσα προγραμματισμού C# για την κρυπτογράφηση, ενώ παράλληλα χρησιμοποιήθηκαν παλαιότερες βιβλιοθήκες C/C++ για τον εντοπισμό ειδικών σημάνσεων του PEB.

Για την συγγραφή του κώδικα χρησιμοποιήθηκε το επίσημο IDE [Integrated development environment] Visual Studio της εταιρίας Microsoft.

7.2.2 Ανάλυση Πηγαίου Κώδικα

Ο πηγαίος κώδικας αποτελείται από δυο βασικά μέρη. Στο πρώτο μέρος αναπτύσσεται η αλγόριθμος του επιβλαβούς λογισμικού και στο δεύτερο μέρος οι προστασίες της ανάλυσης του.

7.2.2.1 Ransomware

Για την ανάπτυξη του Ransomware έχουν δημιουργηθεί οι εξής κλάσεις:

7.2.2.1.1 PassCreator

Στην συγκεκριμένη κλάση γίνεται η δημιουργία ενός τυχαίου κωδικού μήκους 20 αλφαριθμητικών που αποτελείται από κεφαλαία & μικρά γράμματα: ABCDEFGHIJKLMNOPQRSTUVWXYZ | abcdefghijklmnopqrstuvwxyz, αριθμούς: 1234567890 και ειδικούς χαρακτήρες: *!=?/. Για την τυχαιότητα του χρησιμοποιήθηκε μια γεννήτρια που είχε ως «σπόρο» seed την τιμή του χρόνου.

7.2.2.1.2 GetInfo

Αποτελεί την κλάση που εντοπίζει τις πληροφορίες του ονόματος υπολογιστή, το όνομα χρήστη καθώς και τα «μονοπάτια» paths των φακέλων-στόχων του επιβλαβούς λογισμικού. Για τις παραπάνω ανάγκες χρησιμοποιήθηκαν υποκλάσεις της επίσημης βιβλιοθήκης Environment της Microsoft, θεωρώντας ότι δεν δημιουργούν δυσλειτουργίες.

7.2.2.1.3 Sender

Η κλάση είναι επιφορτισμένη με την συγκέντρωση των στοιχείων του χρήστη, του υπολογιστή που εκτελείται το πρόγραμμα, του τυχαίου αλφαριθμητικού κωδικού καθώς και την αποστολή αυτών σε μία απομακρυσμένη βάση δεδομένων. Η βάση δεδομένων είναι ειδικά διαμορφωμένη ώστε να λαμβάνει τα παραπάνω στοιχεία με την διαμεσολάβηση μιας σελίδας php και την επιπλέον καταχώρηση ενός timestamp. Για μεγαλύτερη ασφάλεια η διεύθυνση αποστολής URL της παραπάνω σελίδας είναι κρυπτογραφημένη στον πηγαίο κώδικα και αποκρυπτογραφείται λίγο πριν την αποστολή. Η μέθοδος αποστολής για λόγους ενίσχυσης της ασφάλειας επιλέχθηκε να είναι η POST. Ο κωδικός που αποστέλλεται αποτελεί ένα από τα ευαίσθητα στοιχεία, αφού η πιθανή διαρροή του μπορεί υπό συνθήκες να οδηγήσει στην αποκρυπτογράφηση των αρχείων του χρήστη. Συνεπώς, πριν την αποστολή του με την μέθοδο της POST κρυπτογραφείται με την βοήθεια της συνάρτησης XoRDecoder(). Η συγκεκριμένη συνάρτηση με την βοήθεια ενός επιπλέον κλειδιού κρυπτογραφεί τον κωδικό πριν την αποστολή του. Η ιστοσελίδα που λαμβάνει την κρυπτογραφημένη πληροφορία χρησιμοποιεί μία αντίστοιχη συνάρτηση γραμμένη σε PHP και αφού αποκρυπτογραφήσει την πληροφορία την αποθηκεύει στην βάση δεδομένων. Η ίδια συνάρτηση χρησιμοποιείται για την αποκρυπτογράφηση της διεύθυνσης της ιστοσελίδας που θα γίνει η αποστολή, αφού το συγκεκριμένο URL βρίσκεται ήδη κρυπτογραφημένο στον πηγαίο κώδικα για μεγαλύτερη ασφάλεια. Στις ειδικές περιπτώσεις που δεν μπορεί να προσπελαστεί η σελίδα, είτε διότι δεν

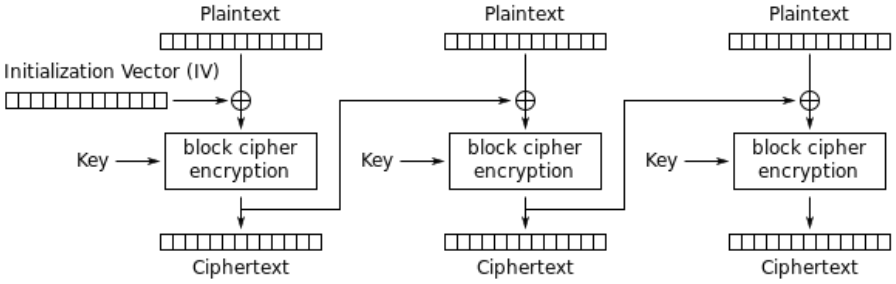
υπάρχει σύνδεση internet στον client, είτε δεν είναι προσπελάσιμος ο Apache Server το πρόγραμμα τερματίζει την λειτουργία του χωρίς να πραγματοποιήσει καμία ενέργεια.

7.2.2.1.4 Chooser

Η κλάση αυτή επιλέγει τα αρχεία που πληρούν τις προδιαγραφές για να κρυπτογραφηθούν. Πιο συγκεκριμένα τα αρχεία με τις καταλήξεις ".doc", ".docx", ".xls", ".xlsx", ".ppt", ".pptx", ".odt", ".jpg", ".png", ".csv", ".sql", ".mdb", ".php", ".asp", ".aspx", ".html", ".xml", ".psd" κρυπτογραφούνται. Ο παραπάνω μηχανισμός φροντίζει να εκτελείται στον φάκελο-στόχο αλλά και στους υποφακέλους αυτού, μέχρι να φτάσει στο μέγιστο δυνατό βάθος του δέντρου των διαδρομών.

7.2.2.1.5 Encryptor

Η Encryptor αποτελεί την καρδιά του προγράμματος. Είναι υπεύθυνη για την κρυπτογράφηση των αρχείων που επιλέγει η Chooser αλλά και για την μετονομασία της διαφορετικής κατάληξης των κρυπτογραφημένων αρχείων. Όπως αναφέρθηκε στην φάση του σχεδιασμού, ο αλγόριθμος κρυπτογράφησης που χρησιμοποιήθηκε είναι ο AES. Πιο συγκεκριμένα, ο AES επιβάλει η κρυπτογράφηση να γίνεται σε δέσμες (blocks) μεγέθους 128 bit (16 bytes), ενώ για ένα υψηλό επίπεδο ασφαλείας, επιλέχθηκε το κλειδί να έχει μήκος 256 bits (32 bytes). Η επιλογή των 256 bits αυξάνει τους γύρους επεξεργασίας που απαιτούνται για την ολοκλήρωση της κρυπτογράφησης σε 14. Σε κάθε γύρο επεξεργασίας γίνεται μια σειρά μετασχηματισμών σε επίπεδο bytes. Αρχικά γίνεται ένας μετασχηματισμός αντικατάστασης bytes χρησιμοποιώντας έναν πίνακα αντικατάστασης, στην συνέχεια ένας μηχανισμός ολίσθησης bytes κατά διαφορετικά offsets για να ακολουθήσει μια ανάμειξη των bytes ανά στήλη και μια τελική προσθήκη ενός κλειδιού. Τέλος, η λειτουργία του AES που επιλέχθηκε είναι η CBC [Cipher Block Chaining] όπου ορίζει προτού γίνει η κρυπτογράφηση ενός νέου block plaintext, να γίνεται XOR, μεταξύ του block αυτού και του ciphertext μπλοκ που είχε παραχθεί μόλις πριν. Με αυτή την μέθοδο 2 παρόμοια plaintext ποτέ δεν κρυπτογραφούνται στο ίδιο ciphertext.



Σχήμα 4: Λειτουργία Κρυπτογράφησης Cipher Block Chaining (CBC)

Είναι σημαντικό να αναφερθεί πως πριν την διαδικασία της κρυπτογράφησης με τον αλγόριθμος AES, ως κλειδί δεν χρησιμοποιείται το τυχαίο password που δημιούργησε η κλάση PassCreator αλλά το αποτέλεσμα μετά το πέρασμά του μέσα από μία συνάρτηση κατακερματισμού. Η συνάρτηση κατακερματισμού που χρησιμοποιήθηκε ονομάζεται SHA-512 [Secure Hash Algorithm] και ανήκει στην κατηγορία SHA-2. [38] Ο SHA-512 χρησιμοποιεί, όπως και κάθε συνάρτηση κατακερματισμού, δύο βασικά στάδια. Αρχικά της προ επεξεργασίας (preprocessing) και στην συνέχεια του υπολογισμού (hash computation). Στο πρώτο στάδιο της προ επεξεργασίας γίνεται η διαδικασία του padding και στην συνέχεια χωρίζεται σε blocks. Στο τελευταίο στάδιο δημιουργείται το «message-schedule» μετά από τους κατάλληλους μετασχηματισμούς καταλήγει στο «message-digest».

Αλγόριθμος	SHA-512
Μέγεθος Εξόδου (bits)	512
Μέγεθος Εσωτερικής Κατάστασης (bits)	512 (8 × 64)
Μέγεθος Block (bits)	1024
Μέγιστο Μέγεθος Μηνύματος (bits)	$2^{128} - 1$
Γύροι Επεξεργασίας	80
Μετασχηματισμοί	And, Xor, Rot, Add (mod 2^{64}), Or, Shr
Security (bits)	256

Πίνακας 1: Χαρακτηριστικά Συνάρτησης Κατακερματισμού SHA-512

Μετά την επιτυχή κρυπτογράφηση γίνεται η προσθήκη της επέκτασης .l4l που αποτελεί ένα ακρωνύμιο της ονομασίας του προγράμματος love4lock. Η νέα επέκταση .l4l δεν επηρεάζει την αληθινή επέκταση του αρχείου αφού τοποθετείται μετά από αυτή.

7.2.2.1.6 TXTCreator

Με την συμβολή της TXTCreator δημιουργούνται τα txt αρχεία της ενημέρωσης του χρήστη για την αποστολή των λύτρων. Επειδή το πρόγραμμα πραγματοποιείται για εκπαιδευτικούς σκοπούς παράγεται και ένα επιπλέον txt αρχείο που περιέχει το τυχαίο κωδικό που παράχθηκε.

7.2.2.2 Προστασία Ανάλυσης

Για την προστασία ανάλυσης έχουν δημιουργηθεί οι εξής κλάσεις:

7.2.2.2.1 Προστασία Debugging

Για την προστασία από το Debugging, κοινώς το anti-Debugging χρησιμοποιήθηκαν αρκετοί μηχανισμοί που ομαδοποιήθηκαν σε 7 κλάσεις.

Στην κλάση aDEB_BeingDebugged γίνεται ένας έλεγχος της σημαίας (flag) «BeingDebugged» στο Process Environment Block (PEB) που επιτρέπει να αντιληφθούμε την διαδικασία του Debugging, ανάλογα αν το σύστημα είναι 32/64bit.

Η κλάση aDEB_CheckBreakpoints παρακολουθεί γενικότερα τα breakpoints. Εμπεριέχει μηχανισμούς που αναγνωρίζουν τα hardware & software breakpoints. Επίσης περιέχει μια μέθοδο που εντοπίζει τα memory breakpoints, επιστρέφοντας μια συνολική αποτίμηση.

Η aDEB_CheckHeapFlags είναι επιφορτισμένη να εξετάζει τρεις βασικούς μηχανισμούς anti-debugging. Ελέγχει την τιμή DWORD της NtGlobalFlag μέσα στο PEB, όπου φαίνονται σημαίες (flags) που αλλάζουν αναλόγως αν το πρόγραμμα βρίσκεται σε διαδικασία debugging. Επίσης ελέγχει τα λεγόμενα heap flags και force heap flags που φανερώνουν σε διάφορα σημεία τους συστήματος αν γίνεται debugging ανάλογα την έκδοση του λειτουργικού συστήματος και την αρχιτεκτονική του.

Η aDEB_IsDP αναλαμβάνει τον έλεγχο δύο συναρτήσεων που ανήκουν στο Win32 API. Συγκεκριμένα, της συνάρτησης IsDebuggerPresent() και της CheckRemoteDebuggerPresent() όπου επιστρέφουν μη μηδενικές τιμές στην περίπτωση που γίνεται debugging.

Στην aDEB_CheckProcessDebug ελέγχεται η συνάρτηση NtQueryInformation Process που έμμεσα χρησιμοποιούσε και η CheckRemoteDebuggerPresent. Επίσης χρησιμοποιείται η NtQuerySystemInformation που κάνει προσπάθεια να

ανακτήσει ένα handle της τρέχουσας διαδικασίας σφαλμάτων. Αν αυτό γίνει με επιτυχία σε αυτή την περίπτωση αντιλαμβανόμαστε ότι γίνεται debugging. Τέλος, όταν καλείται η NtQueryProcessInformation μέσω της κλάσης ProcessDebugFlags, η συνάρτηση επιστρέφει το αντίστροφο της EPROCESS-> NoDebugInherit, γεγονός που προδίδει το debugging.

Η aDEB_GetWindowThread αποτελεί μια κλάση που ελέγχει συνοπτικά αν το πρόγραμμα μας εκτελείται απευθείας μέσω του επίσημου shell process της Microsoft, δηλαδή του explorer.exe για να συμπεράνει αν πραγματοποιείται κάποιο είδος debugging.

Με την κλάση aDEB_NtQueryObject ανιχνεύεται η δημιουργία ενός debug «αντικειμένου» με την βοήθεια ενός handle που είναι συνδεδεμένο με αυτό, έτσι είναι δυνατό να αντιληφθούμε ότι υπάρχει debugging session.

7.2.2.2.2 Προστασία απο Εργαλεία Ανάλυσης

Η κλάση aANA_Check4AnalysisTools εμπεριέχει 2 βασικούς ελέγχους που αφορούν εργαλεία ανάλυσης. Η συνάρτηση MATRun() | Monitor Analysis Tools Run, εξετάζει αν είναι ενεργά στο περιβάλλον του υπολογιστή τα εργαλεία ανάλυσης που εμπεριέχει η λίστα της. Σε περίπτωση που εντοπιστεί ένα από αυτά τα εργαλεία, θα αποτρέψει την εκτέλεση του επιβλαβούς λογισμικού. Η εν λόγω λίστα έχε τα εξής εργαλεία:

- Debugger Immunity
- Debugger OllyDebug
- Disassembler IDA Pro
- Hook Explorer
- Import Reconstructor
- Lord PE
- Network Traffic Tool
- PE Tool
- Process Hacker
- Sandbox Joe
- Sandboxie
- SysAnalyzer iDefense
- SysInspector
- Sysinternals Suite
- WinDbg
- Wireshark

Η επόμενη συνάρτηση MATExists() | Monitor Analysis Tools Exists, ανιχνεύει τα εργαλεία ανάλυσης ή/και προγράμματα Antivirus από την προτεινόμενη θέση εγκατάστασης τους στο γνωστό φάκελο C:\Program Files\ χωρίς απαραίτητα να εκτελούνται εκείνη την στιγμή.

7.2.2.2.3 Προστασία απο Sandboxes

Η κλάση aSBX_Checks εξετάζει αν το πρόγραμμα εκτελείται μέσα από κάποιο Sandbox. Συγκεκριμένα η κλάση εντοπίζει modules, από ορισμένα dlls όπως τα dir_watch.dll, cmdvrt32.dll, SxIn.dll, SbieDll.dll και snxhk.dll αναγνωρίζοντας ότι εκτελείται μέσα από κάποιο πρόγραμμα sandbox, δίνοντας εντολή μη εκτέλεσης του προγράμματος. Τα προγράμματα Sandboxes που μπορούν να εντοπιστούν είναι τα Sunbelt, Comodo, Qihoo 360, Sandboxie και Avast.

7.2.2.2.4 Προστασία από Εικονικές Μηχανές (Virtual Machines)

Μέσα από μια ομάδα τεσσάρων κλάσεων εντοπίζεται αν το πρόγραμμα εκτελείται μέσα από μία εικονική μηχανή (Virtual Machine).

Η κλάση aVM_Checks4Parallels ελέγχει αν εκτελούνται οι διεργασίες prl_cc.exe & prl_tools.exe ώστε να εντοπίσει το γνωστό Virtual Machine Parallels.

Στην κλάση aVM_Checks4VB υπάρχουν αρκετοί μηχανισμοί που εξετάζουν την ύπαρξη του VirtualBOX. Συγκεκριμένα, οι συναρτήσεις VBRegKEYv() & VBRegKEY() ελέγχουν συγκεκριμένες τιμές στην Registry. Η συνάρτηση system32VB() εξετάζει την ύπαρξη συγκεκριμένων αρχείων μέσα στο γνωστό φάκελο των windows, system32. Η συνάρτηση directoryVB() ελέγχει για την ύπαρξη συγκεκριμένων φακέλων όπως ο «oracle\virtualbox guest additions\». Η macVB() εξετάζει αν η προεπιλεγμένη MAC Address ξεκινάει από 08:00:27, που είναι by default προεπιλεγμένη στο VirtualBOX της Oracle. Η PseudoDevicesVB() ελέγχει την ύπαρξη των ψευδο-Devices στο VirtualBOX. Η WindowClassVB() αναγνωρίζει το Class: “VBoxTrayToolWndClass” και το Window: “VBoxTrayToolWnd”. Η SharedFolderVB() εντοπίζει την ονομασία "VirtualBox Shared Folders" που δίνεται από την συγκεκριμένη εικονική μηχανή. Ο έλεγχος των συγκεκριμένων processes "vboxservice.exe", "vboxtray.exe" γίνεται από την κλάση VBProcesses(). Τέλος με τις συναρτήσεις WMIeventsVB(), WMImacVB(), WMIDevicesVB() εντοπίζονται παρόμοιες πληροφορίες με αυτές που έχουν αναφερθεί ήδη μέσω του WMI [Windows Management Instrumentation].

Η κλάση aVM_Checks4VMware είναι υπεύθυνη για την αναγνώριση της εικονικής μηχανής VMware. Ο πρώτη μέθοδος ονομάζεται loadedVMdlls() και ελέγχει την ύπαρξη ενός συγκεκριμένου αρχείου (dbghelp.dll) που προδίδει την ύπαρξη του εν λόγω virtual machine. Οι συναρτήσεις VMwareRegKEYv() & VMwareRegKEY() ελέγχουν την ύπαρξη συγκεκριμένων κλειδιών / τιμών στην Registry. Ο έλεγχος της

ύπαρξης των αρχείων "system32\drivers\vmmouse.sys" και "system32\drivers \vmhgfs.sys" πραγματοποιείται με την μέθοδο system32VMware. Η directoryVMware() εξετάζει την ύπαρξη του φακέλου VMware στη προεπιλεγμένη θέση του Program Files. Στην μέθοδο macVMware() γίνεται ο έλεγχος των συγκεκριμένων MAC Address που συνήθως ξεκινάνε οι εικονικές μηχανές του VMware. Συγκεκριμένα οι διευθύνσεις ξεκινούν με 00:05:69, 00:0C:29, 00:1C:14 και 00:50:56. Η PseudoDevicesVMware() καταφέρνει να εντοπίζει τα ψευδο-devices του VMware με τα χαρακτηριστικά «vmci» & «HGFS». Τέλος, η μέθοδος VMwareProcesses() εντοπίζει τις βασικές διεργασίες του VMware, όπως «vmtoolsd.exe», «vmwaretray.exe» και «vmwareuser.exe».

Η κλάση aVM_Checks4VP εξετάζει την ύπαρξη της εικονικής μηχανής Windows Virtual PC, μέσω 2 μηχανισμών. Η μέθοδος VPProcesses() διαπιστώνει την παρουσία των διεργασιών "VMSrvc.exe" και "VMUSrvc.exe" που γίνεται κατά την χρήση του Virtual PC. Στην loadedVPdlls() εντοπίζει την βιβλιοθήκη vmcheck.dll που «προδίδει» την εν λόγω εικονική μηχανή.

7.2.2.2.5 Συγκέντρωση Αποτιμήσεων | CHECKoConditions

Η κλάση CHECKoConditions είναι επιφορτισμένη να συγκεντρώνει τα αποτελέσματα όλων των ομάδων προστασίας από Debuggers, Sandboxes, Virtual Machines και γενικότερα Εργαλεία Ανάλυσης ώστε να δίνει την τελική αποτίμηση για την εκκίνηση ή μη των ενεργειών του επιβλαβούς λογισμικού.

7.2.2.3 SetTimer

Η κλάση SetTimer αναλαμβάνει να δημιουργήσει μια εικονική καθυστέρηση ώστε να εκτελεστεί το επιβλαβές λογισμικό σε δεύτερο χρόνο. Η εικονική καθυστέρηση η οποία έχει οριστεί είναι της τάξεως των τριών (3) λεπτών, ώστε να αποφεύγει τους γρήγορους ελέγχους. Παράλληλα εξετάζεται το χρονικό διάστημα που έχει παρέλθει, ώστε να διαπιστωθεί ότι δεν υπάρχουν χειροκίνητες ενέργειες που μπορούν να τον παρακάμψουν. Επίσης, στην συγκεκριμένη μέθοδο γίνεται χρήση του Windows API GetTickCount, που επιστρέφει τον αριθμό των χιλιοστών του δευτερολέπτου (milliseconds) που το σύστημα είναι σε λειτουργία. Έχει οριστεί έναν παράθυρο χρόνου 20 λεπτών όπου είναι απαραίτητο να λειτουργεί ήδη το σύστημα στο οποίο θα εκτελεστεί το πρόγραμμά μας. Με αυτό τον τρόπο αποτυγχάνουν πιθανοί έλεγχοι από Sandboxes, Virtual Machines και γενικότερα συστήματα που έχουν εκκινηθεί ευκαιριακά για τον έλεγχο του συγκεκριμένου προγράμματος.

7.2.2.4 hidePfromTaskbar()

Στην hidePfromTaskbar() ενεργοποιείται ένας μηχανισμός που αναλαμβάνει να «εξαφανίσει» το πρόγραμμα από το την κεντρική μπάρα του λειτουργικού συστήματος Windows. Συνεπώς, όταν εκτελείται το πρόγραμμα δεν γίνεται αντιληπτό από τον χρήστη.

7.2.3 PHP & MySQL

Για την υποδοχή των στοιχείων του χρήστη πραγματοποιείται η συντήρηση μιας βάσης δεδομένων MySQL που περιέχει τον πίνακα rescue [id, pn, un, pss, time]. Στον συγκεκριμένο πίνακα γίνεται η αποθήκευση του ονόματος υπολογιστή, του ονόματος χρήστη και του αλφαριθμητικού κωδικού που χρησιμοποιήθηκε για την κρυπτογράφηση. Παράλληλα γίνεται και αποθήκευση ενός timestamp για την αποτύπωση της ακριβής χρονικής στιγμής που έγινε η κρυπτογράφηση. Για την διαμεσολάβηση μεταξύ της λήψης των παραπάνω στοιχείων και την αποθήκευση στην βάση υλοποιήθηκε κατάλληλη PHP σελίδα. Η ιστοσελίδα είναι επιφορτισμένη να δέχεται τα δεδομένα μέσω τις μεθόδου POST και αφού τα αναγνωρίσει να κάνει την κατάλληλη εισαγωγή τους στην βάση δεδομένων. Όπως έχει αναφερθεί και παραπάνω ο κωδικός πριν αποθηκευτεί στην βάση αποκρυπτογραφείται με κατάλληλη συνάρτηση με όνομα Keydecoder() που έχει υλοποιηθεί σε php. Τέλος, η έκδοση PHP που χρησιμοποιήθηκε είναι v5.6, που σε συνδυασμό με την MySQL 5.7 γίνει δυνατή η χρησιμοποίηση της επέκτασης MySQLi που περιέχει μηχανισμούς για την εξάλειψη των SQL injections.

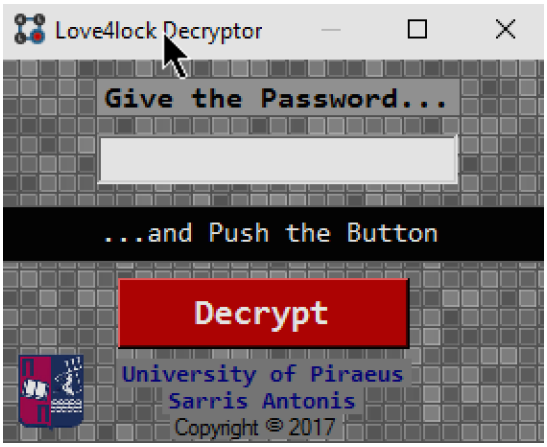
7.2.4 Love4lock Decryptor

Σε κάθε υλοποίηση κρυπτογράφησης θεωρείται απαραίτητη η δημιουργία ενός ασφαλούς αντίστοιχου προγράμματος αποκρυπτογράφησης, ώστε να επανέλθει το όποιο αντικείμενο στην αρχική του μορφή. Το πρόγραμμα «Love4lock Decryptor» υλοποιήθηκε συμπληρωματικά με το ransomware Love4lock, ώστε μετά την εισαγωγή του κωδικού να προβαίνει στην αποκρυπτογράφηση των αρχείων που είχαν κρυπτογραφηθεί.

Εικόνα 4: Love4lock

Για την υλοποίηση του χρησιμοποιήθηκε η γλώσσα προγραμματισμού C++/CLI. Ο «Love4lock Decryptor» απαρτίζεται από μια απλή φόρμα (form) που περιμένει την εισαγωγή του κωδικού. Μετά την εισαγωγή του κωδικού και πατώντας ο χρήστης το κουμπί Decrypt, το πρόγραμμα αναγνωρίζει το όνομα χρήστη, θα εντοπίσει τα

αρχεία, θα ξεκινάει την αποκρυπτογράφηση των αρχείων αλλά και το σβήσιμο των μηνυμάτων ειδοποίησης του χρήστη. Συγκεκριμένα η κλάση GetInfo αναλαμβάνει να εντοπίσει το όνομα χρήστη και την θέση-στόχο – My Documents. Στην συνέχεια στην κλάση Chooser αφού εισαχθεί το μονοπάτι, εντοπίζονται τα κρυπτογραφημένα αρχεία. Η κλάση Decryptor καλείται να αποκρυπτογραφήσει τα αρχεία του φακέλου και των υποφακέλων αφού πρώτα, ο κωδικός εισαχθεί μέσα από την συνάρτηση κατακερματισμού SHA-512. Μετά το τέλος της επιτυχούς κρυπτογράφησης ο χρήστης ενημερώνεται με σχετικό μήνυμα στην φόρμα. Επίσης, ενημερώνεται και για την έλλειψη αρχείων για κρυπτογράφηση. Τέλος, η εφαρμογή σβήνει τις ενημερώσεις προς τον χρήστη, αφού τα αρχεία πλέον έχουν αποκρυπτογραφηθεί.



Εικόνα 5: Love4lock Decryptor Main Form

7.2.5 Obfuscation | Packing Εκτελέσιμου

Όπως έχει ήδη αναφερθεί το packing είναι μια διαδικασία που συμπιέζει και κωδικοποιεί το εκτελέσιμο σε ένα νέο «πακεταρισμένο» εκτελέσιμο. [39]. Η χρήση τους έχουν το πλεονεκτήματα ότι προστατεύουν το εκτελέσιμο από Reverse-Engineering, ενώ ταυτόχρονα το συμπιέζουν για εξοικονόμηση χώρου. Από την άλλη έχουν το μειονέκτημα ότι αυτή η συμπίεση μπορεί να προκαλέσει καθυστερήσεις στην αποσυμπίεση και επιπλέον ότι τα Antivirus βλέπουν με μεγαλύτερη καχυποψία το εκτελέσιμο. Για το packing χρησιμοποιήθηκαν δύο (2) διαφορετικά προγράμματα, συνεπώς προέκυψαν δύο (2) επιπλέον εκδοχές του τελικού εκτελέσιμου.

- **ConfuserEX**

  Το πρόγραμμα ConfuserEx είναι ανοιχτού κώδικα λογισμικό –δωρεάν- και

θεωρείται ευρέως ως ένα από τα ισχυρότερα obfuscators διαθέσιμα σε .NET. Δεδομένου ότι είναι ανοικτού κώδικα, μπορεί να τροποποιηθεί ώστε να ταιριάζει στις ανάγκες του love4lock. Διατίθεται σε 2 εκδόσεις GUI & CLI και διαθέτει αρκετά plugins που ενισχύουν σημαντικά την ασφάλεια.

Συγκεκριμένα στο love4lock επιλέχθηκαν τα εξής plugins:

- Anti-Dump Protection
- Constants Protection
- Control Flow Protection
- Name Protection

- **Themida**

  Η Themida αποτελεί έναν ισχυρότατο εμπορικό πρόγραμμα *Obfuscator* / Packer, παρέχοντας αρκετές επιλογές που βοηθούν στην προστασία ενός εκτελέσιμου από το Reverse-Engineering. [40] Η επιλογή του συγκεκριμένου προγράμματος –εκτός ότι αποτελεί μια από της καλύτερες λύσεις- έγινε επειδή είναι δυνατή η χρήση του ως trial για ορισμένες ημέρες, καλύπτοντας όλες τις απαιτήσεις για την παρούσα εργασία. Οι δυνατότητες που χρησιμοποιήθηκαν είναι οι εξής:

  - Anti-memory dumpers techniques for any Ring3 and Ring0 dumpers
  - Advanced Entry point protection
  - Dynamic encryption in target application
  - Compression of target application, resources and protection code

8 *Ανάλυση Love4lock & Αποτελέσματα*

Υστέρα από την επιτυχή υλοποίηση του συνολικού προγράμματος, σε αυτό το κεφάλαιο καταγράφονται οι δοκιμές που πραγματοποιήθηκαν, καθώς και τα αποτελέσματα αυτών. Οι δοκιμές έγιναν με την χρήση χρήσιμων εργαλείων ανάλυσης που διατίθενται είτε open source, είτε δωρεάν, είτε σε trial εκδόσεις με ορισμένους περιορισμούς από μικρές, μεγάλες εταιρίες, αλλά και μεμονωμένους Προγραμματιστές. Σκοπός της συγκεκριμένης ενότητας δεν είναι εξαντλητική ανάλυση, που άλλωστε δεν ήταν και σκοπός στην παρούσα εργασία, αλλά η ανάδειξη των κύριων διαφορών μεταξύ των εκδοχών που θα μελετήσουμε.

8.1 Διάρθρωση

Οι δοκιμές που πραγματοποιήθηκαν έγιναν σε δύο βασικές καταστάσεις του προγράμματος. Η πρώτη κατάσταση βρίσκει το πρόγραμμα υλοποιημένο αλλά χωρίς τα εργαλεία anti-Analysis. Η δεύτερη κατάσταση, αποτελεί της δοκιμές που πραγματοποιήθηκαν με όλους του μηχανισμούς ενεργοποιημένους με επιπλέον τους packers που επιλέχθηκαν.

Η διάρθρωση που επιλέχθηκε ακολουθεί τα βήματα μιας φυσιολογικής ανάλυσης λογισμικού ώστε τα αποτελέσματά και κατά προέκταση τα συμπεράσματα που θα προκύψουν να ανταποκρίνονται πιο κοντά στην πραγματικότητα. Οι δοκιμές θα πραγματοποιηθούν μαζί με τους μηχανισμούς anti-Analysis που υλοποιήθηκαν στην παρούσα εργασία, αλλά και δίχως αυτούς. Επίσης, θα δοκιμαστούν και 2 διαφορετικοί packers πάντα σε συνδυασμό με τους μηχανισμούς anti-Analysis.

Συνεπώς οι 2 καταστάσεις θα δίνουν 4 εκδοχές του επιβλαβούς λογισμικού ως εξής:

[i] Love4lock: **δίχως** χρήση των μηχανισμών anti-Analysis

[ii] Love4lock: **με** χρήση των μηχανισμών anti-Analysis

[iii] Love4lock: με χρήση των μηχανισμών anti-Analysis και Packing ConfuserEX

[iv] Love4lock: με χρήση των μηχανισμών anti-Analysis και Packing Themida

Συνεπώς για λόγους ευκολίας της αναφοράς θα αναφέρεται ανάλογα με την εκδοχή επιπλέον και ο εκάστοτε λατινικός αριθμός.

8.2 Στατική Ανάλυση Love4lock

8.2.1 Στατική Ανάλυση με Antivirus

Όλες οι παραπάνω εκδοχές του love4lock δοκιμάστηκαν στα εξής Antivirus:

- Kaspersky Anti-Virus
- NOD32 Antivirus
- Norton Security
- Bitdefender Antivirus

Όλες οι εκδόσεις ήταν δοκιμαστικές για περιορισμένο διάστημα, αλλά ήταν πλήρως λειτουργικές. Η εγκατάσταση έγινε σε πραγματικό περιβάλλον σε ηλεκτρονικό υπολογιστή με τα εξής χαρακτηριστικά:

- ✓ OS: Microsoft Windows 10 Edu / 64-bit
- ✓ CPU: Intel Core i7 4790K 4GHz
- ✓ RAM: 4GB

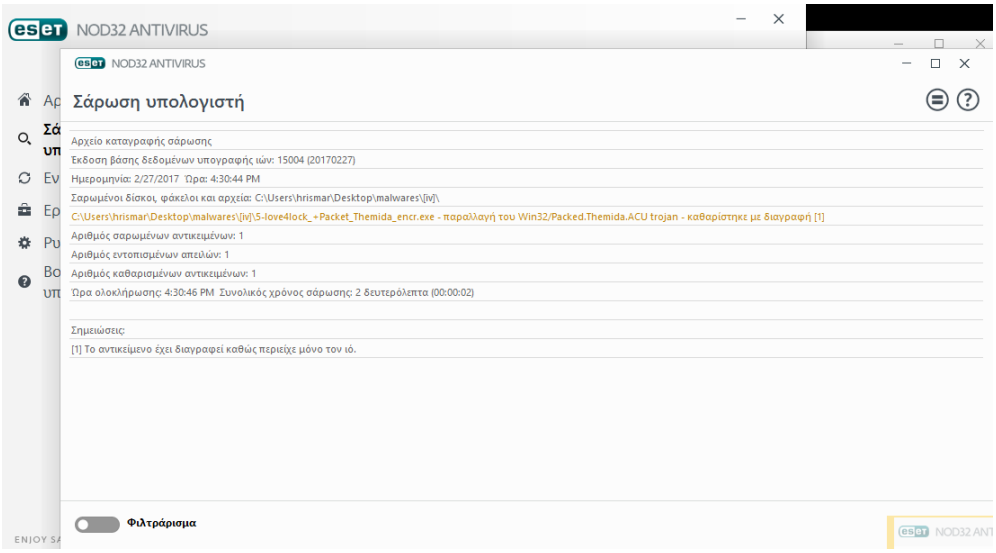
Στο πλαίσιο της στατικής ανάλυσης η μόνη ενέργεια που πραγματοποιήθηκε ήταν η σάρωση των αρχείων χωρίς την εκτέλεση τους.

Αν και τα αποτελέσματα ποικίλουν (πίνακας 2) ανάλογα με το πρόγραμμα AV, οι εκδοχές [i] & [ii] έδειξαν να μην ανιχνεύονται από κανένα AV. Οι εκδοχές [iii] & [iv] διαγράφηκαν άμεσα από το NOD32 Antivirus (εικόνα 6) προφανώς λόγω των packer, αφού δεν υπήρχε άλλη διαφορά από την εκδοχή [ii]. Το Norton Security διέγραψε άμεσα μόνο την [iv] εκδοχή ενώ δεν ενέργησε την εκδοχή [iii]. Ο Bitdefender & Kaspersky δεν εντόπισαν καμία απειλή σε όλες τις εκδοχές.

Πίνακας 2: AVs (Στατική Ανάλυση)

	NOD32	Kaspersky Antivirus	Norton Security	Bitdefender
[i]	μη ανίχνευση	μη ανίχνευση	μη ανίχνευση	μη ανίχνευση
[ii]	μη ανίχνευση	μη ανίχνευση	μη ανίχνευση	μη ανίχνευση
[iii]	άμεση διαγραφή	μη ανίχνευση	μη ανίχνευση	μη ανίχνευση

[iv]	άμεση διαγραφή	μη ανίχνευση	άμεση διαγραφή	μη ανίχνευση



Εικόνα 6: NOD32 Antivirus

Κατά την δυναμική ανάλυση θα εξεταστούν και οι επιπλέον δοκιμές εκτέλεσης του επιβλαβούς λογισμικού, ώστε να εξεταστεί η ανίχνευση της συμπεριφοράς τους.

8.2.2 Ανάλυση των Strings

Όπως έχει αναφερθεί και στην θεωρία, η ανάλυση των strings ενός εκτελέσιμου προγράμματος μπορεί να «προδώσει» τους σκοπούς μιας εφαρμογής. Για την χρήση των δοκιμών χρησιμοποιήθηκε το open source πρόγραμμα BinText, το οποίο τρέχει σε γραφικό περιβάλλον.

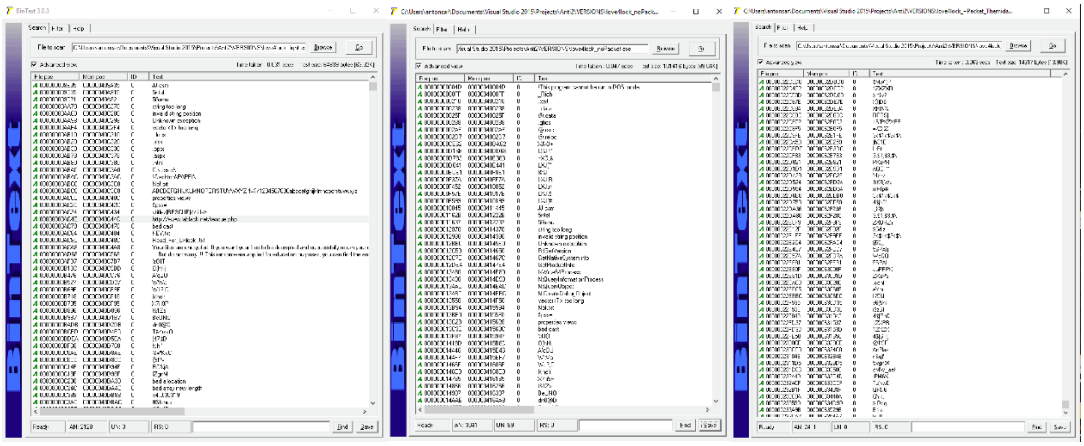
Στην [i] εκδοχή του love4lock (εικόνα 7α) διαπιστώθηκαν να φαίνονται κρίσιμα strings που δείχνουν την διάθεση του εκτελέσιμου. Φάνηκαν κάποια μονοπάτια (paths), η διεύθυνση μιας ιστοσελίδας php, μηνύματα που φαίνεται να απευθύνονται στον χρήστη και μια σειρά από σχεδόν όλους τους αλφαριθμητικούς εκτυπώσιμους χαρακτήρες. Μπορεί κάποιος καταγράφοντας μεμονωμένα τα παραπάνω στοιχεία, να μην έχουν να του δώσουν κάποια πληροφορία, αλλά αν τα σκεφτεί συνδυαστικά -όπως είναι υποχρεωμένος να κάνει ένας αναλυτής- να μπορέσει να βγάλει αρκετά συμπεράσματα αναζητώντας περισσότερες πληροφορίες για τα παραπάνω.

Επίσης φαίνονται και οι ονομασίες όλων των κλάσεων που χρησιμοποιήθηκαν κατά την ανάπτυξη του προγράμματος. Το συγκεκριμένο στοιχείο είναι πολύ σοβαρό αφού από την ονομασία τους μπορεί κανείς να συμπεράνει την λειτουργία τους.

Στην κοινότητα των αναλυτών είναι γνωστές κάποιες συναρτήσεις των Windows ως «ύποπτες». Αυτό βέβαια δεν σημαίνει ότι όποιο πρόγραμμα τις καλεί είναι κακόβουλο, αλλά σε κάθε περίπτωση χρειάζεται η ανάλογη προσοχή. Τέτοιες συναρτήσεις βρέθηκαν ανάμεσα στα strings και συνεπώς διαπιστώνει κανείς ότι καλούνται. Ενδεικτικά αναφέρουμε τις εξής:

	[i]	[ii]	[ii]	[iv]
IsDebuggerPresent	x	x	x	-
CoCreateInstance	x	x	x	-
GetModuleHandle	x	x	x	-
GetStartupInfo	x	x	x	-
QueryPerformanceCounter	x	x	x	-
system	x	x	x	-
WideCharToMultiByte	x	x	x	-
CheckRemoteDebuggerPresent		x	x	-
CreateFile		x	x	-
CreateToolhelp32Snapshot		x	x	-
FindWindow		x	x	-
GetAdaptersInfo		x	x	-
GetProcAddress		x	x	-
GetThreadContext		x	x	-
GetTickCount		x	x	-
GetWindowsDirectory		x	x	-
IsWoW64Process		x	x	-
LoadLibrary		x	x	-
NtQueryInformationProcess		x	x	-
OpenProcess		x	x	-
Process32First/Process32Next		x	x	-
ReadProcessMemory		x	x	-

Πίνακας 3: "Υποπτες" Συναρτήσεις Windows



Εικόνα 7: BintText

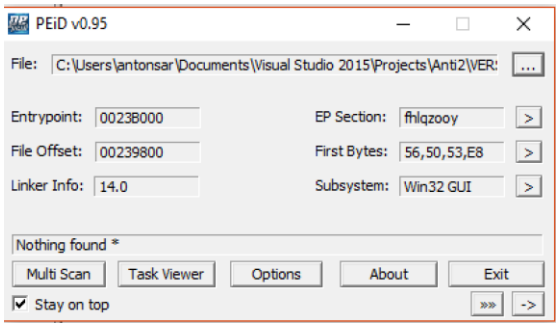
Στην [ii] εκδοχή του love4lock (εικόνα 7β) δεν διαπιστώθηκαν χρήσιμα λεκτικά strings, αλλά όπως φαίνεται στον πίνακα 3 φαίνονται πολύ περισσότερες από τις οι γνωστές «ύποπτες» συναρτήσεις των Windows. Αξίζει να αναφερθεί ότι οι ονομασίες των κλάσεων που αναπτύχθηκαν κατά την διάρκεια της υλοποίησης εξακολουθούν να φαίνονται.

Στην εκδοχή [iii] εξακολουθούν και εμφανίζονται οι «ύποπτες» συναρτήσεις των Windows αλλά όχι και οι ονομασίες των κλάσεων που δημιουργήθηκαν.

Στην εκδοχή [iv] δεν εμφανίζεται σχεδόν κανένα λεκτικό που να μπορεί να διαβαστεί. Επιπλέον είναι αδύνατο να διαβαστεί το όνομα οποιαδήποτε συνάρτησης. Αυτό φυσικά κάνει εμφανές το γεγονός ότι τα strings έχουν υποστεί κρυπτογράφηση, συνεπώς το μόνο που μπορεί να κάνει κάποιος σε αυτή την περίπτωση είναι να σκεφτεί ότι πρόκειται για κάτι «ύποπτο».

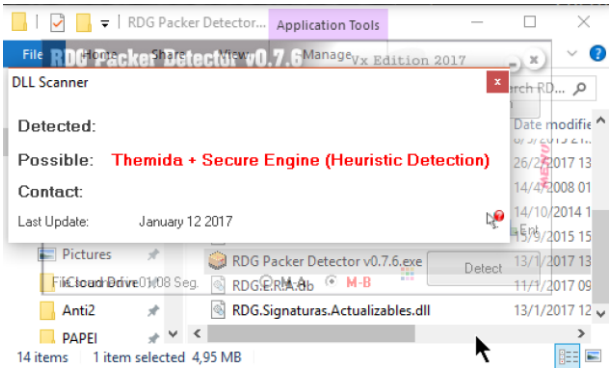
8.2.3 Ανίχνευση Στοιχείων PE & Έλεγχος Packing

Για τον έλεγχο των στοιχείων PE χρησιμοποιήθηκε το προγράμματα PEiD. Σε αυτό το στάδιο το μόνο που μας απασχολεί είναι να εντοπίσουμε τον packer και να ελέγξουμε μήπως φαίνεται κάτι ύποπτο στο EP Section. Το PEiD δεν κατάφερε να εντοπίσει τους packer στις εκδοχές που υπήρχαν, δηλαδή [iii] & [iv]. Παρόλα αυτά εντοπίστηκε στο EP Section τις [iv] ένα περίεργο λεκτικό «fhlqzooy» που μαρτυρά την ύπαρξη packer. (εικόνα 8)



Εικόνα 8: PEiD

Στην συνέχεια χρησιμοποιήθηκε το πρόγραμμα RDG Packer Detector που κατάφερε να διαπιστώσει τον packer στις περιπτώσεις [iii] & [iv]. Αυτό σημαίνει ότι μπορεί κανείς να εντοπίσει τους αντίστοιχους unpackers για να προχωρήσει σε μεγαλύτερη ανάλυση. Ενδεικτικά στην εικόνα 9 φαίνεται ο εντοπισμός του packer Themida.

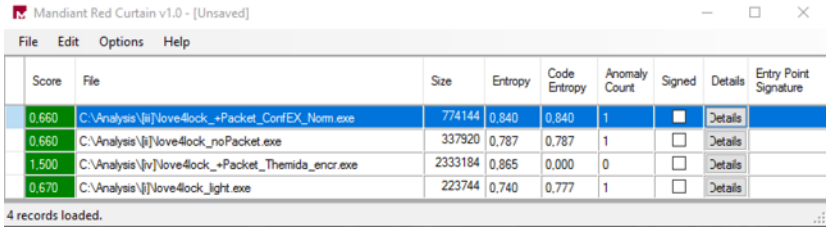


Εικόνα 9: RDG Packer Detector

8.2.4 Εξέταση Εντροπίας

Ένα χρήσιμο εργαλείο της στατικής ανάλυσης είναι το Mandiant Red Curtain όπου μπορούμε να εξετάσουμε την εντροπία του επιβλαβούς λογισμικού. Στην εικόνα 10 στην στήλη Entropy φαίνεται η εντροπία για την κάθε μια από τις τέσσερις εκδοχές. Το score προκύπτει από όλους τους δείκτες που βρίσκει το εργαλείο ανάλογα την βαρύτητα που δίνει στον καθένα από αυτούς. Το score μπορεί να είναι από 0 έως 10. Οτιδήποτε είναι μεγαλύτερο του 1 έχει ενδιαφέρον η περεταίρω ανάλυση του.

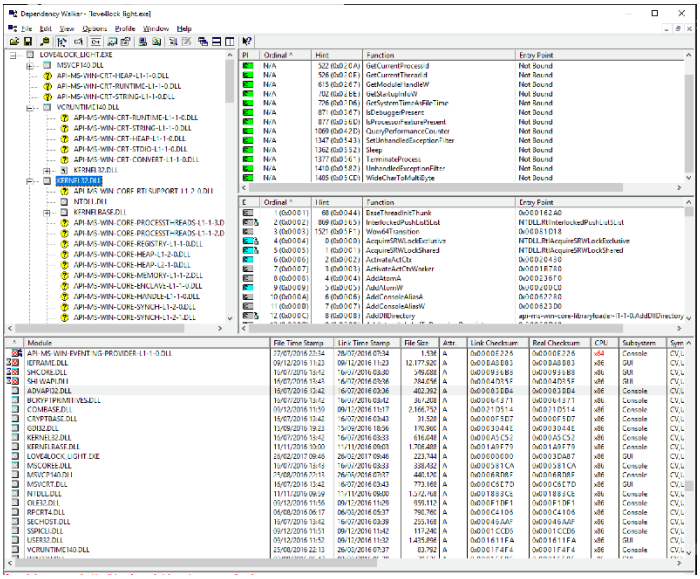
Παρατηρείται ότι στις εκδοχές [i] & [ii] το μέγεθος της εντροπίας παραμένει σε πολύ χαμηλά επίπεδα αφού είναι unpacked. Στην εκδοχή [iii] υπάρχει μια μικρή αύξηση χωρίς όμως να φτάνει σε πολύ μεγάλο μέγεθος αφού δεν έχει επιλεγεί κάποιου είδους συμπίεση. Επιπλέον βλέπουμε ότι μέγεθος του αρχείου έχει διπλασιαστεί. Στη εκδοχή [iv] έχουμε επιπλέον αύξηση της εντροπίας, αλλά βλέπουμε πολύ μεγάλη διαφορά στο μέγεθος του αρχείου αλλά και στον δείκτη του score που υπερδιπλασιάστηκε μετακινώντας το αρχείο στην επικίνδυνη ζώνη. Αντίθετα στην συγκεκριμένη εκδοχή δεν φαίνεται να υπάρχει ούτε το «checksum_is_zero» στην στήλη Anomaly Count που παρατηρήθηκε στις άλλες εκδοχές αφού είναι αδύνατο να εντοπιστεί από την κρυπτογράφηση.



Εικόνα 10: Ένδειξη Εντροπίας

8.2.5 Έλεγχος DLLs

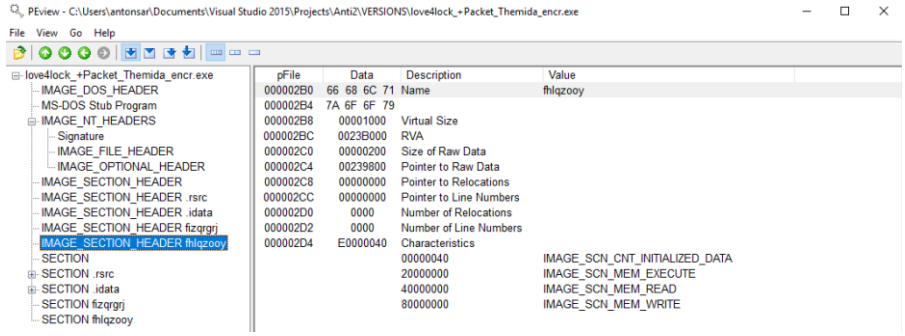
Με το εργαλείο Dependency Walker εξετάστηκαν τα DLLs που καλούν όλες οι εκδοχές. Καμία εκδοχή δεν κατάφερε να κρύψει τα DLLs που καλούνται. Στις εκδοχές [iii] & [iv] διαπιστώθηκε μια προσπάθεια μεγαλύτερης «σύγχυσης» προφανώς για να υπάρχει μεγαλύτερη πολυπλοκότητα στην ανάλυση, αλλά το αποτέλεσμα είναι ακριβώς το ίδιο. Μια απεικόνιση της εκδοχής [i] εμφανίζεται στην εικόνα 11.



Εικόνα 11: Dependency Walker

8.2.6 Ανάλυση Στοιχείων PE

Για την ανάλυση των στοιχείων PE χρησιμοποιήθηκε το PEview & το FileAlyzer. Διαπιστώθηκε ότι σε όλες τις εκδοχές φαίνεται η ακριβείς ημερομηνία και ώρα που έχει γίνει compile το πρόγραμμα, ανεξάρτητα αν υπάρχει packing. Επίσης με το συγκεκριμένο εργαλείο διαπιστώθηκαν ύποπτες αποκλίσεις ή/και ταυτίσεις μεταξύ του «Virtual Size» και του «Size of Raw Data». Ενδεικτικά στην εικόνα 12 φαίνεται μια διαφορά υπερβολική σε ποσοστό.



Εικόνα 12: PEview

Για μια πιο ολοκληρωμένη αποτύπωση οι διαφορές που διαπιστώθηκαν συγκεντρώθηκαν στον πίνακα 4. Με πορτοκαλί χρώμα διαπιστώνονται στην εκδοχή [iii] να υπάρχουν διπλά τα .text και .rsrc και με κόκκινο στην εκδοχή [iv] φαίνονται οι επεμβάσεις του packer Themida.

Εκδοχή [i]

Section	Virtusl size	Size of raw data	diff
.text	A39A	A400	102
.data	F34	A00	-1332
.rdata	20652	20800	430
.rsrc	A598	A600	104
.gfids	38	200	456
.reloc	77C	800	132

Εκδοχή [ii]

Section	Virtusl size	Size of raw	diff
.text	1215A	12200	166
.data	1734	E00	-2356
.rdata	33E6C	34000	404
.rsrc	A598	A600	104
.gfids	38	200	456
.reloc	A54	C00	428

Εκδοχή [iii]

Section	Virtusl size	Size of raw data	diff
.text	1215A	12200	166
.data	1734	E00	-2356
.rdata	33E6C	34000	404
.rsrc	A598	A600	104
.gfids	38	200	456
.reloc	A54	C00	428
.text	60034	60200	460
.rsrc	A590	A600	112

Εκδοχή [iv]

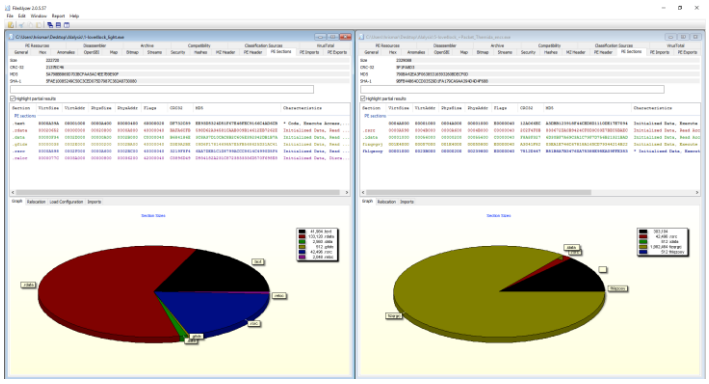
Section	Virtusl size	Size of raw	diff
.text			0
.data			0
.rdata			0
.rsrc	A598	A600	104
.gfids			0
.reloc			0
"empty"	4A000	4A000	0
.idata	1000	200	-3584
fizqrgrj	1E4000	1E4000	0
fhlqzooy	1000	200	-3584

Πίνακας 4: Virtual size vs Size of raw data

Το FileAlyzer είναι ένα δωρεάν εργαλείο ανάλυσης των πληροφοριών που είναι αποθηκευμένες στα έγγραφα PE, στις κεφαλίδες των αρχείων και σε άλλα τμήματα. Το πρόγραμμα ανοίγει ένα παράθυρο με πολλές καρτέλες επιλογών.

Η αρχική καρτέλα "Γενικά" παρουσιάζει μερικά βασικά στοιχεία: την τοποθεσία, το μέγεθος, την έκδοση, τη δημιουργία / τελευταία πρόσβαση / τελευταία εγγραφή, χαρακτηριστικά, και μερικά hashes (CRC-32, MD5, SHA-1). Η καρτέλα "PE Κεφαλίδα" εμφανίζει διάφορες λεπτομέρειες από την επικεφαλίδα του προγράμματος όπως αν το εκτελέσιμο είναι 32 ή 64-bit.

Στη καρτέλα «PE SECTION» μπορούμε να δούμε γραφικά την κατανομή του προγράμματος και μπορούμε να συγκρίνουμε δυο προγράμματα με τα παράλληλα παράθυρα.

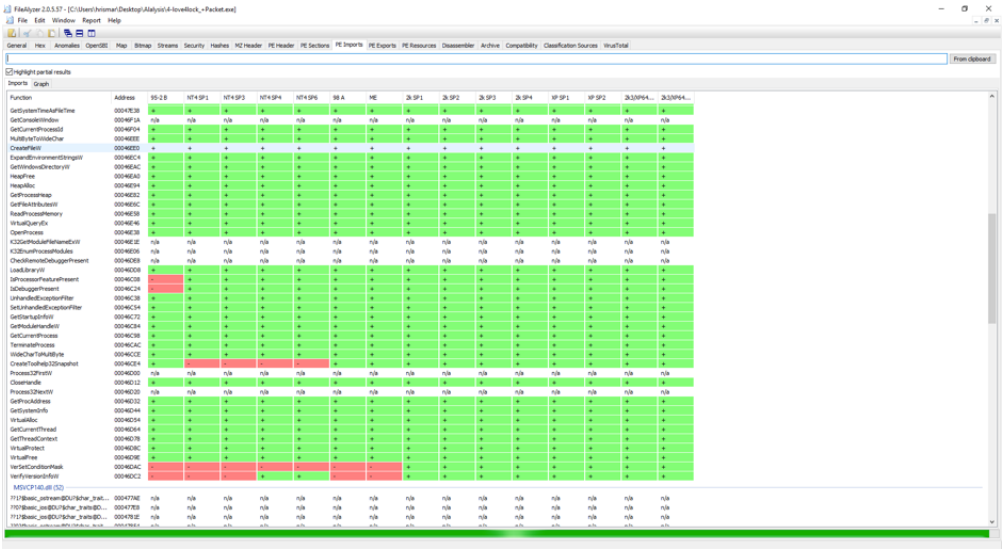


Εικόνα 13: FileAlyzer

Η καρτέλα «Anomalies» καταγράφει τα προβλήματα και τις ανωμαλίες που διαπιστώνονται κατά την ανάλυση του εκτελέσιμου. Στη εκδοχή [iv] ο FileAlyzer αναγνωρίζει ότι μάλλον το εκτελέσιμο είναι packed, καθώς βγάζει το μήνυμα “Section fhlqzooy has execution access enabled to a section that is also marked as

writable. This is typical for packed files that are extracted at runtime". Στις άλλες εκδοχές του προγράμματος δεν καταγράφονται προβλήματα.

Στην καρτέλα «PE imports» παρουσιάζονται τα dlls που καλούνται. Στην καρτέλα αυτή παρατηρούμε ότι ενώ σε όλες τις εκδοχές παρουσιάζονται μερικά καλούμενα dlls, στη εκδοχή [iv] ο FileAlyzer δεν μπορεί να εντοπίσει κανένα dll. Στην καρτέλα αυτή μπορεί να γίνει και γραφική απεικόνιση των καλούμενων dlls.



Εικόνα 14: PE Imports

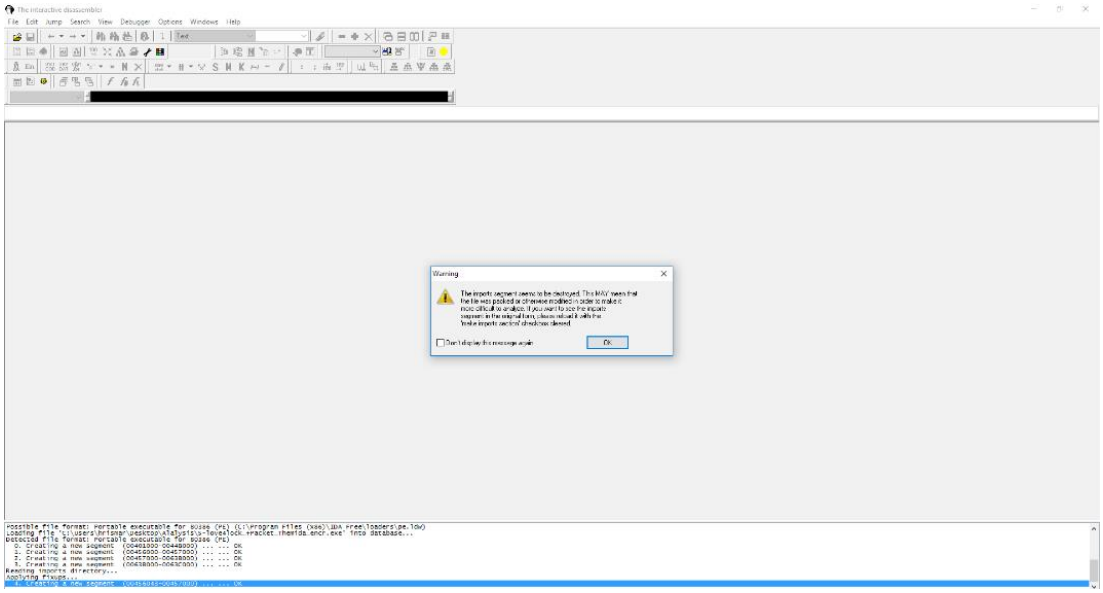
8.2.7 Ανάλυση με χρήση Disassembly

Το IDA PRO είναι ίσως το καλύτερο εργαλείο για disassembly ανάλυση. Η χρήση του θέλει πολύ καλή γνώση της γλώσσας assembly. Στα πλαίσια αυτής της εργασίας έγινε με την χρήση του IDA PRO μια γρήγορή ανάλυση ώστε να εντοπιστούν οι διαφορές στις διαφορετικές εκδοχές του προγράμματος.

Με τη φόρτωση του εκτελέσιμου το IDA PRO ένα από τα στοιχεία που εξετάζονται έχει την ονομασία «Name Window» όπου είναι καταγεγραμμένη σε μια λίστα, κάθε διεύθυνση με όνομα συμπεριλαμβανομένου των συναρτήσεων, την ονομασία κώδικα, δεδομένων και των αλφαριθμητικών. Η λίστα αυτή για τις εκδοχές [i] & [ii] όταν δεν υπάρχουν οι έλεγχοι ή/και με τους ελέγχους είναι σχετικά μικρή 154 και 236 αντίστοιχα. Με είσοδο των εκδοχών [iii] & [iv] η λίστα αυτή γίνεται 2918 γραμμές κάνοντας πολύ πιο δύσκολή και επίπονη την ανάλυσή τους.

Τέλος, η εκδοχή [iv] με την χρήση του packer Themida εμφάνισε αμέσως μήνυμα ότι το πρόγραμμα μάλλον έχει συμπιεστεί για να γίνει πιο δύσκολη η ανάλυση και θέλει ξεχωριστή μεταχείριση (εικόνα 15). Άλλωστε όπως είχε διαπιστωθεί νωρίτερα,

έχουν εξαφανιστεί τόσο τα strings όπως είδαμε στην βασική στατική ανάλυση όσο και οι ονομασίες, με αποτέλεσμα η λίστα μας τώρα να έχει μόνο 5 γραμμές.



Εικόνα 15: IDA Pro

Σε συνέχεια της ανάλυσης με το IDA PRO έγινε έλεγχος της εύρεσης του αλγορίθμου κρυπτογράφησης που χρησιμοποιεί το malware. Τα plugins FindCrypt1 & 2 και Signsrch χρησιμοποιούνται για την εύρεση παρόμοιων αλγορίθμων εντοπίζοντας τα «magic constants». Δυστυχώς όμως σε καμία από τις εκδοχές δεν λειτούργησαν και ο λόγος είναι ότι η γλώσσα προγραμματισμού με την οποία είναι γραμμένο το επιβλαβές λογισμικό είναι C++/CLI καθώς και ο αλγόριθμος κρυπτογράφησης που έχει χρησιμοποιηθεί έχει εισαχθεί από βιβλιοθήκη της .NET.

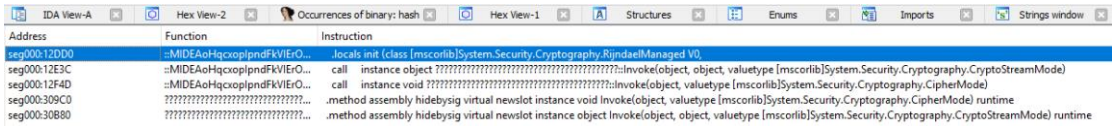
Σε κάθε περίπτωση ο αναλυτής λαμβάνει υπόψη του την λίστα των γνωστών αλγορίθμων συμμετρικού και ασύμμετρου κλειδιού που δίνονται από τις βιβλιοθήκες της Microsoft όπως φαίνονται στον πίνακα 5.

System.Security.Cryptography	
AsymmetricAlgorithm	Aes
DSA	DES
ECDiffieHellman	RC2
ECDsa	Rijndael
RSA	TripleDES

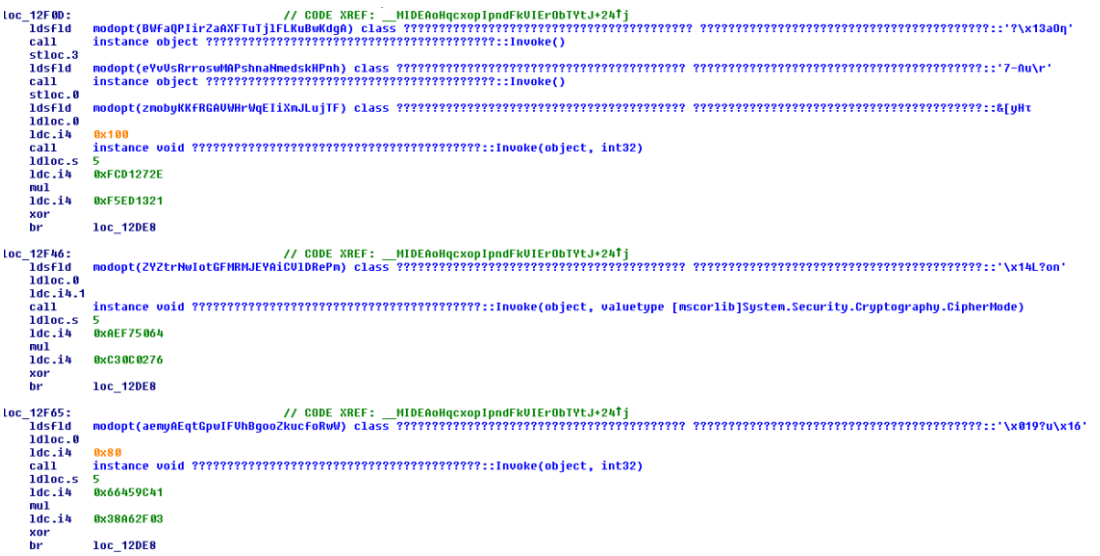
Πίνακας 5: System Security Cryptography

Στις εκδοχές [i] & [ii] υπήρχε μεγάλη ευκολία -από την πλευρά του αναλυτή- στην εύρεση του αλγορίθμου κρυπτογράφησης αφού εμφανίζονται οι αυθεντικές ονομασίες των κλάσεων/συναρτήσεων συνεπώς θα αρκούσε κανείς να αναζητήσει με το όνομα τους. Στην εκδοχή [iii] η κατάσταση δεν είναι τόσο εύκολη, αφού ο κώδικας έχει υποστεί «συσκότιση». Σε αυτή την περίπτωση δοκιμάστηκε με επιτυχία η αναζήτηση του namespace: System.Security.Cryptography, όπου όπως φαίνεται στην εικόνα 16 εντοπίστηκε στις instructions η κλάση System.Security. Cryptography.RijndaelManaged που αποκαλύπτει τον αλγόριθμο κρυπτογράφησης αφού πρόκειται για την πιο αντιπροσωπευτικό του AES. Στην συνέχεια γνωρίζοντας ότι δηλώνονται τα key size και το block size εντοπίστηκαν σε 16-δική μορφή 0x100 & 0x80 όπως φαίνεται στην εικόνα 17.

Στην εκδοχή [iv] δεν κατέστη δυνατό να γίνει καμία αναγνώριση αφού δεν είναι δυνατή η ανάγνωση. Σε αυτή την περίπτωση ο μόνος τρόπος θα ήταν να προηγηθεί σε ένα προηγούμενο στάδιο της ανάλυσης κάποια διαδικασία unpacking.



Εικόνα 16: Cryptography Rijndael Managed

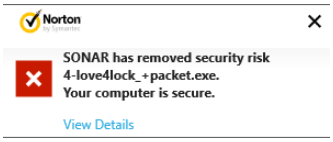


Εικόνα 17: Μέγεθος Key 0x100 [256] - Μέγεθος Block 0x80 [128]

8.3 Δυναμική Ανάλυση Love4lock

8.3.1 Δυναμική Ανάλυση με Antivirus

Στο συγκεκριμένο στάδιο της δυναμικής ανάλυσης και οι τέσσερις εκδοχές εκτελέστηκαν ενώ ήταν ενεργοποιημένο το real time protection του εκάστοτε προγράμματος Antivirus.



Εικόνα 18: Norton detect Themida

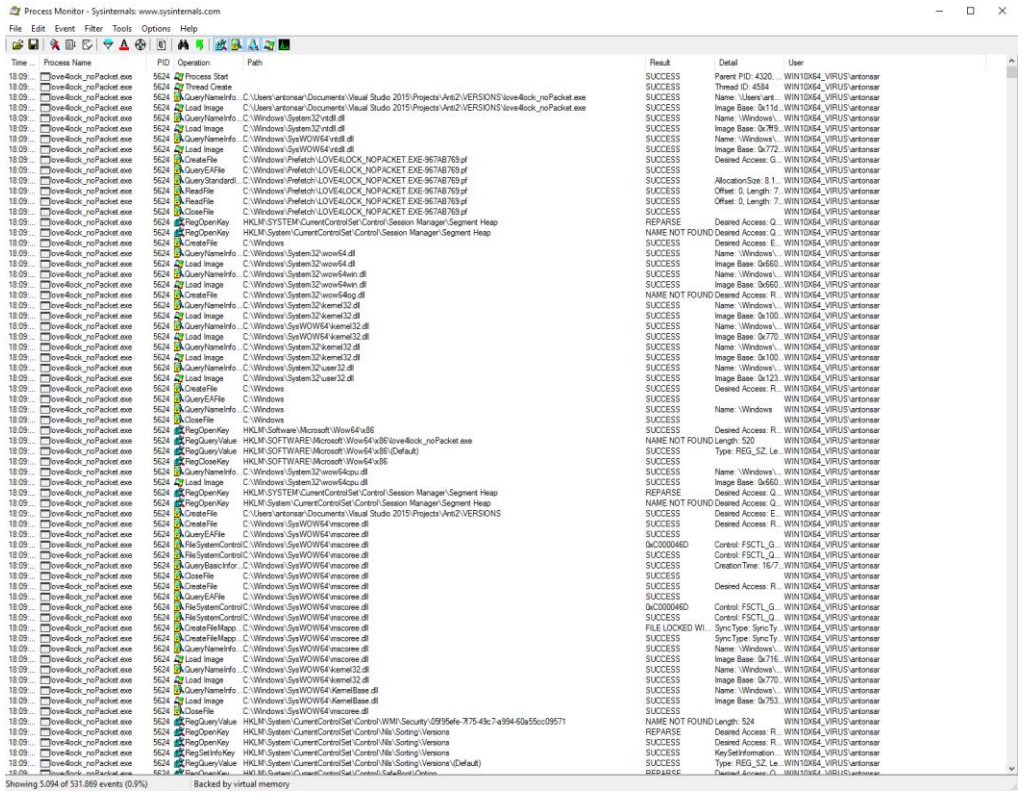
Τα Antivirus μη έχοντας την υπογραφή του συγκεκριμένου ransomware θα μπορούσαν ενδεχομένως να αντιληφθούν τον κίνδυνο από την συμπεριφορά του εκτελέσιμου. Οι δοκιμές έγιναν διπλή φορά με ενεργοποιημένο τον μηχανισμό anti-Analysis Tools και χωρίς αυτόν. Τα αποτελέσματα φαίνονται στο πίνακα 6 και φαίνεται ότι δεν υπάρχει κοινή προσέγγιση. Επίσης φαίνεται να παίζει πολύ σημαντικό ρόλο ο packer αφού στις περισσότερες των περιπτώσεων υπήρχε ακαριαία διαγραφή.

	Kaspersky Anti-Virus		NOD32 Antivirus		Norton Security		Bitdefender Antivirus	
anti-Analysis Tools	ON	OFF	ON	OFF	ON	OFF	ON	OFF
[i]	RUN		RUN		RUN (alert)		RUN	
[ii]	Don't RUN	RUN	Don't RUN	RUN	Don't RUN	RUN (alert)	Don't RUN	RUN
[iii]	Don't RUN	RUN	Deleted	Deleted	Deleted	Deleted	Deleted	Deleted
[iv]	Don't RUN	RUN	Deleted	Deleted	Deleted	Deleted	Deleted	Deleted

Πίνακας 6: Αποτελέσματα Antivirus

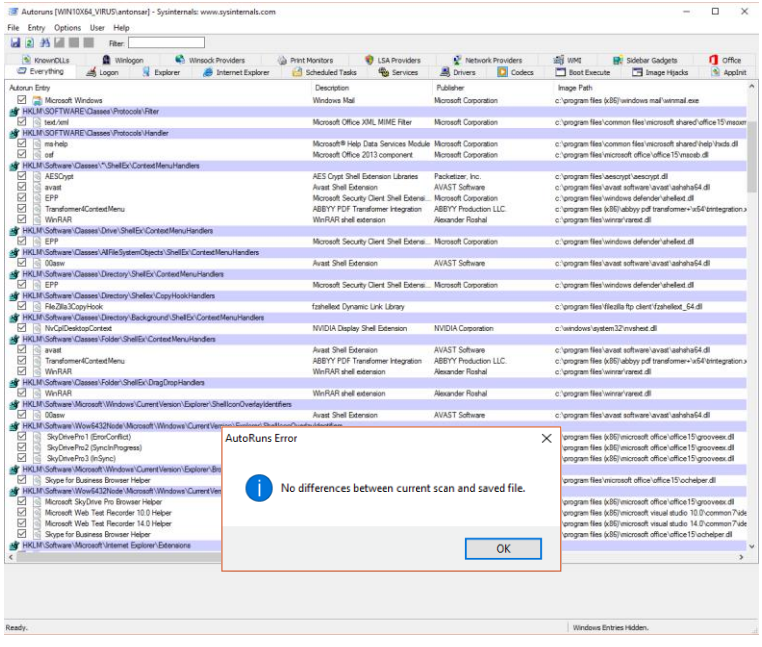
8.3.2 Συστεμική Ανάλυση

Για την γενικότερη συστεμική ανάλυση χρησιμοποιήθηκε το εργαλείο Process Monitor (εικόνα 19) το οποίο είναι σε θέση να αντιληφθεί κάθε ενέργεια που γίνεται στο σύστημά μας. Αυτή η ενέργεια μπορεί να είναι από μια απλή ανάγνωση ενός τύπου αρχείου μέχρι και την τροποποίηση, κλήση ή/και διαγραφή του. Η καταγραφή αυτή είναι τόσο μεγάλη που αν δεν περιοριστεί με κατάλληλα φίλτρα δεν είναι δυνατό να αναγνωστεί. Συνεπώς ένας περιορισμός (φίλτρο) για την εκδοχή [i] έδειξε ότι οι «αλληλεπίδραση» με το σύστημα υπήρχε σε πολύ μικρότερο βαθμό από τις εκδοχές [ii],[iii] & [iv].



Εικόνα 19: Process Monitor

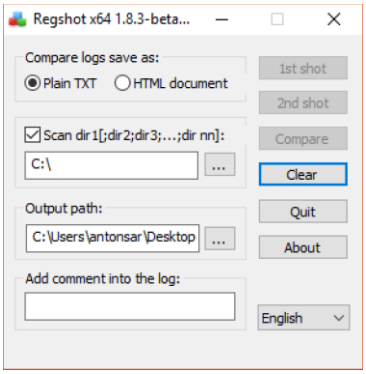
Επίσης στο πλαίσιο της συστεμικής ανάλυσης χρησιμοποιήθηκε το πολύ γνωστό εργαλείο Autoruns. Το συγκεκριμένο κρατάει μια εικόνα του συστήματος πριν την οποιαδήποτε ενέργεια του επιβλαβούς λογισμικού και την συγκρίνει με την εικόνα που θα συσχετίσει μετά την εκτέλεσή του. Το εν λόγω πρόγραμμα έδειξε και για τις τέσσερις εκδοχές ότι δεν υπήρχαν μεταβολές (εικόνα 20).



Εικόνα 20: Autoruns

8.3.3 Ανάλυση Registry

Το σύστημα εξετάστηκε με το εργαλείο Regshot που είναι σε θέση να συγκρίνει δύο καταστάσεις της Registry, πριν και μετά την εκτέλεση κάθε εκδοχής του επιβλαβούς λογισμικού (εικόνα 21). Η δοκιμή έδειξε πως δεν υπήρχαν ουσιώδεις διαφορές αφού ένα Ransomware δεν έχει κάποια επίδραση στην Registry. Για όλες τις εκδοχές αυτό το οποίο φάνηκε ήταν η διαγραφή (εικόνα 22) των αρχείων και η δημιουργία του txt της ενημέρωσης του χρήστη για τα λύτρα.



Εικόνα 21: Regshot Main Form

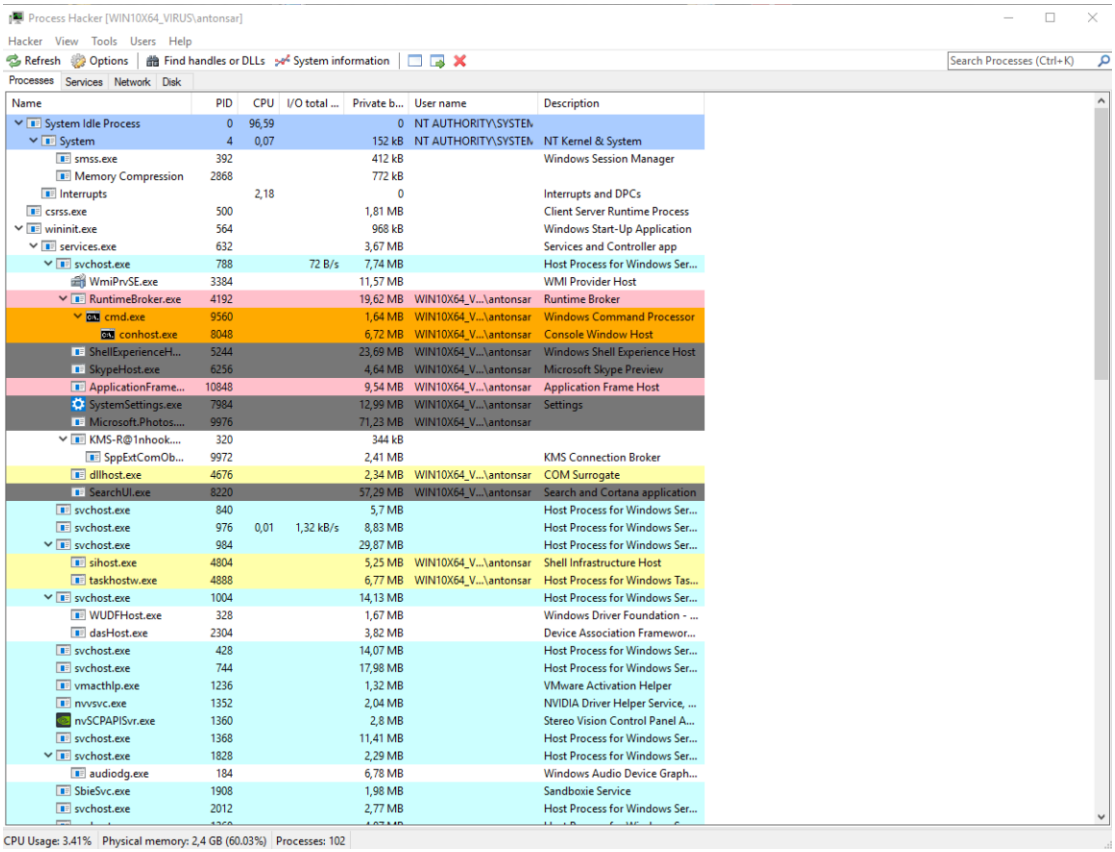
```
----------------------------------
Files added:2
----------------------------------
C:\Users\antonsar\Desktop\PAPEI\daizy.docx.l4l
C:\Users\antonsar\Desktop\PAPEI\Read_For_Unlock!.txt

----------------------------------
Files deleted:1
----------------------------------
C:\Users\antonsar\Desktop\PAPEI\daizy.docx
```

Εικόνα 22: Regshot – Output

8.3.4 Ανάλυση Διεργασιών (Processes)

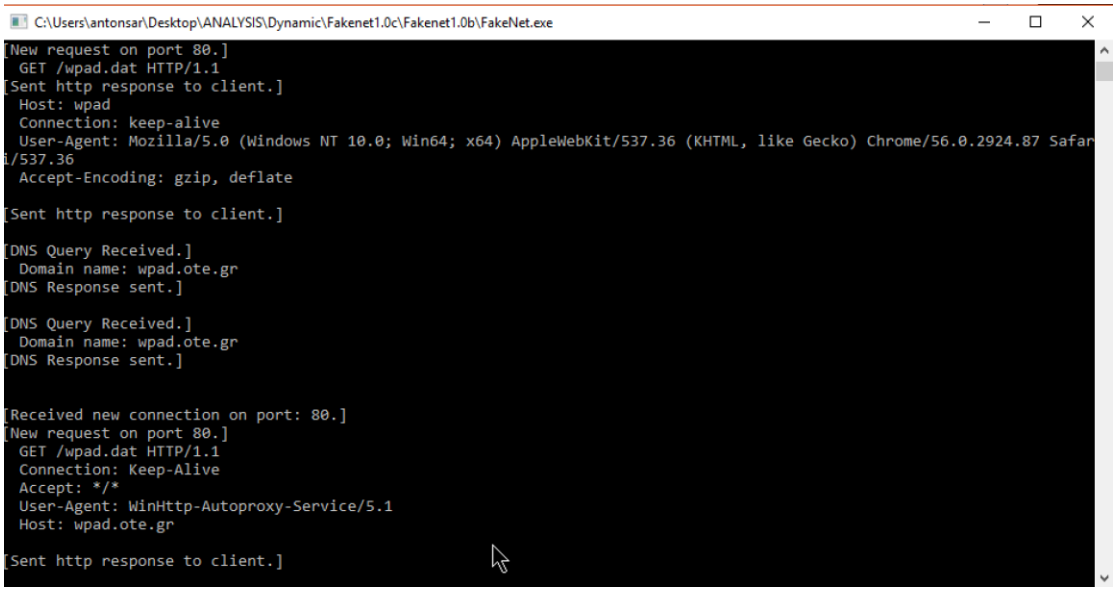
Στο κομμάτι της ανάλυσης των διεργασιών εξετάστηκε το πρόγραμμα Process Hacker (εικόνα 23) το οποίο μπορεί να αναγνωρίσει όλο το δέντρο των διεργασιών και των υποεργασιών. Στη εκδοχή [i] πραγματοποιήθηκε η αναγνώριση του lovelock και έγινε η εν λόγω εξέταση. Στις περιπτώσεις [ii],[iii] & [iv] που είναι ενεργοποιημένοι οι μηχανισμοί anti-Analysis Tools το επιβλαβές λογισμικό αντιλήφθηκε την παρουσία του εργαλείου και δεν εκτελέστηκε ποτέ.



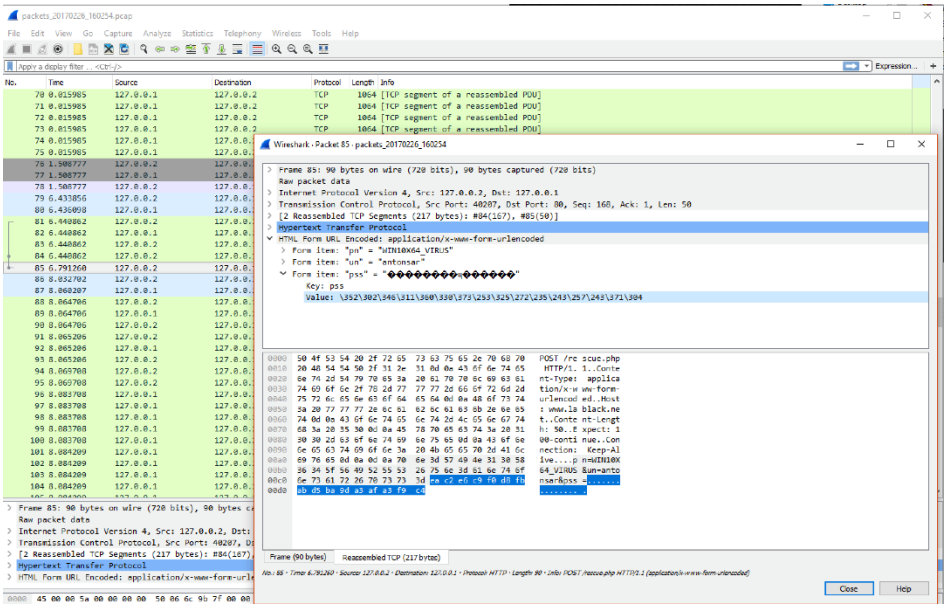
Εικόνα 23: Process Hacker

8.3.5 Έλεγχος Δικτύου

Στην περίπτωση που δεν υπήρχε δίκτυο δεν λειτουργούσε καμία εκδοχή. Στην εξέταση του δικτύου χρησιμοποιήθηκε το FakeNET (εικόνα 24) που δημιουργεί ένα περιβάλλον εικονικού δικτύου ώστε να πάρει την όποια πληροφορία προσπαθεί να επικοινωνήσει ένα επιβλαβές λογισμικό. Στην περίπτωση μας λειτούργησε μόνο στην εκδοχή [i], αφού στις εκδοχές [ii],[iii] & [iv] το επιβλαβές λογισμικό δεν λειτούργησε λόγω των μηχανισμών ασφαλείας που επιβάλουν πρώτα μια συγκεκριμένη απάντηση από τον Server, γεγονός που δεν ήταν σε θέση να το γνωρίζει το FakeNet. Η πληροφορία που λαμβάνεται από το FakeNet και την εκδοχή [i] φαίνεται με την βοήθεια του πολύ χρήσιμου εργαλείου Wireshark να είναι μέσω της μεθόδου POST. Παρόλα αυτά όπως φαίνεται στην εικόνα 25 είναι κρυπτογραφημένο το περιεχόμενο της πληροφορίας και δεν μπορεί να αναγνωστεί. Υπογραμμίζεται πως την αποκρυπτογράφηση αναλαμβάνει η php. Εννοείται πως ακόμη και στην δοκιμή που έγινε με πραγματική χρήση του διαδικτύου το αποτέλεσμα ήταν να εκτελεστεί το love4lock αλλά να στείλει κρυπτογραφημένα τα δεδομένα μέσω της μεθόδου POST.



Εικόνα 24: Fakenet

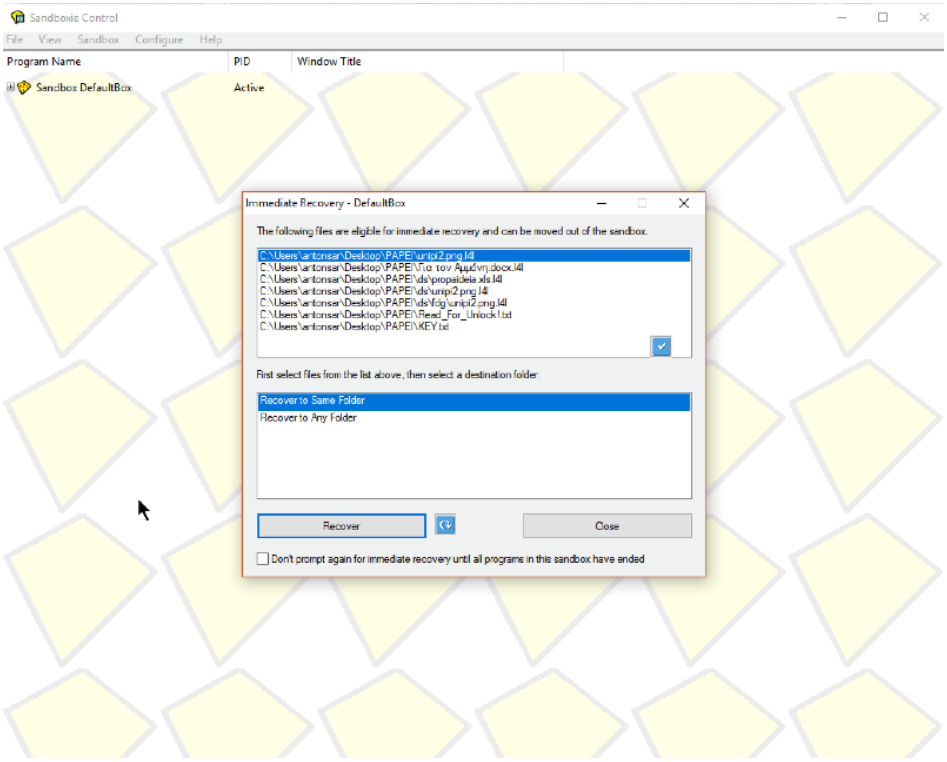


Εικόνα 25: Wireshark - Μέθοδος Post

8.3.6 Εξέταση Συμπεριφοράς σε Sandbox

Το sandbox που χρησιμοποιήθηκε είναι το Sandboxie. Αρχικά στην εκδοχή [i] το Sandboxie κατάλαβε ότι το πρόγραμμα θέλει να κάνει κάποια ενέργεια σε αρχεία που βρίσκονται εκτός του sandbox και έκανε την ερώτηση αν θέλω να συνεχίζω κρατώντας τα αρχεία σε backup (εικόνα 26). Συνεπώς ακόμη και που εκτελέστηκε το επιβλαβές λογισμικό τα αρχεία του χρήστη παρέμειναν προστατευμένα. Στις εκδοχές [ii],[iii] όπως ήταν αναμενόμενο το πρόγραμμα τερματίστηκε λόγω των μηχανισμών Anti-Sandboxes με βασικότερο εκείνο της βασικής καθυστέρησης της εκκίνησης

εκτέλεσης. Στην εκδοχή [iv] το πρόγραμμα δεν εκτελέστηκε ποτέ αφού το «έκοβε» το Sandboxie για λόγους ασφαλείας τους οποίους δεν εξηγούσε.



Εικόνα 26: Sandboxie

8.3.7 Virtual Machines

Στην διάρκεια των δοκιμών χρησιμοποιήθηκαν τα εξής Virtual Machines:

- VMware
- VirtualBOX
- Parallels

Όπως ήταν αναμενόμενο η εκδοχή [i] εκτελέστηκε σε όλες τις περιπτώσεις σε αντίθεση με τις εκδοχές [ii],[iii] & [iv] που το πρόγραμμα τερμάτιζε την λειτουργία του. Αυτό βέβαια συνέβαινε λόγων των μηχανισμών anti-Virtual Machine που έχουν ενσωματωθεί.

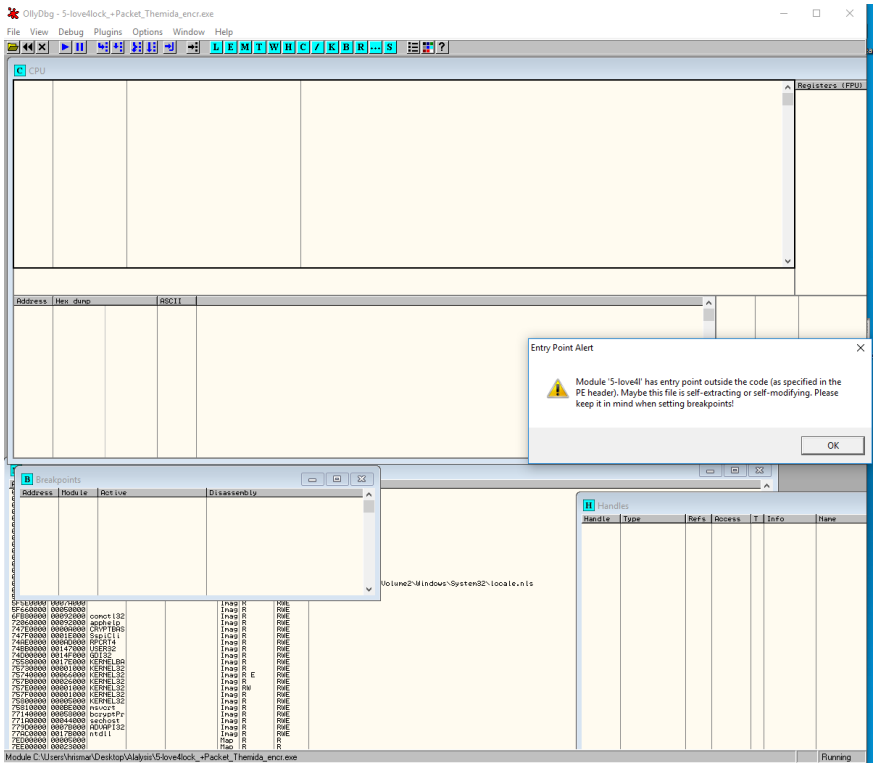
Εικόνα 27: Parallels - VMware - VirtualBox

8.3.8 Έλεγχος Debuggers

Για την ανάλυση ενός επιβλαβούς λογισμικού, όπως έχει αναφερθεί και στα παραπάνω κεφάλαια, χρησιμοποιούνται και οι τεχνικές του debugging. Για τον έλεγχο της συμπεριφοράς όλων των εκδοχών κατά την διάρκεια του debugging χρησιμοποιήθηκε το πρόγραμμα OllyDbg.

Η εκδοχή [i] του love4lock ανοίχθηκε από το OllyDbg και έγινε κανονικά το debugging. Μετά την ολοκλήρωση του Debugging το πρόγραμμα εκτελέστηκε και τα αρχεία κρυπτογραφήθηκαν χωρίς πρόβλημα. Στις εκδοχές [ii] & [iii], το επιβλαβές λογισμικό ανοίχθηκε κανονικά από το OllyDbg αλλά σταμάτησε την λειτουργία του, λόγω του μηχανισμού Anti-Debugging και τα αρχεία του υπολογιστή δεν κρυπτογραφήθηκαν ποτέ.

Τέλος, στην εκδοχή [iv] το OllyDbg δίνει προειδοποιητικό μήνυμα ότι το σημείο εισόδου βρίσκεται εκτός κώδικα (εικόνα 28). Αν επιλέξουμε να συνεχίσουμε την ανάλυσή μας εμφανίζεται ένα νέο μήνυμα που προειδοποιεί ότι το πρόγραμμα μπορεί να είναι συμπιεσμένο, κρυπτογραφημένο ή να περιέχει μεγάλο αριθμό ενσωματωμένων δεδομένων και επομένως η ανάλυσή του μπορεί να δώσει λάθος αποτελέσματα. Συνεπώς, η συνέχεια λόγω του anti-Debugging μηχανισμού δεν ήταν δυνατό να προχωρήσουμε στην ανάλυση του προγράμματος.



Εικόνα 28: OllyDbg

9 Συμπεράσματα & Μελλοντικές Επεκτάσεις

9.1 Συμπεράσματα

Στην παρούσα μεταπτυχιακή διατριβή, παρουσιάστηκε σε πρώτο στάδιο ο σχεδιασμός και η υλοποίηση ενός ransomware, ονόματι love4lock. Σε δεύτερο στάδιο υλοποιήθηκαν και ενσωματώθηκαν μια σειρά από μηχανισμοί anti-Analysis για να ακολουθήσει το τρίτο στάδιο των δοκιμών / ανάλυσης της συμπεριφοράς των τεσσάρων εκδοχών που επιλέχθηκαν για την καλύτερη αποτύπωση της σύγκρισης των αποτελεσμάτων.

Το πρώτο στάδιο περιείχε την ανάπτυξη του επιβλαβούς λογισμικού με την χρήση μόνο βασικών μηχανισμών αποφυγής υποκλοπής του κλειδιού κρυπτογράφησης κατά την αποστολή του στην απομακρυσμένη βάση δεδομένων. Από το πρώτο στάδιο προκύπτει η εκδοχή [i], όπου παρατηρήθηκε πως ήταν ευάλωτη κατά την στατική ανάλυση, στον έλεγχο των αλφαριθμητικών (strings), στην εύρεση του αλγορίθμου κρυπτογράφησης και τις εύρεσης της γενικότερης λογικής του πηγαίου κώδικα. Στο αντίποδα η εν λόγω εκδοχή ήταν ανθεκτική στην στατική ανάλυση των AVs, στην εξέταση της εντροπίας και την ανάλυση των στοιχείων PE. Κατά την δυναμική ανάλυση η εκδοχή [i] παρουσιάστηκε ευάλωτη σε όλους τους ελέγχους, πλην του ελέγχου των AVs που λόγω της έλλειψης εύρεσης γνωστού signature ή/και ταύτιση παρόμοιας συμπεριφοράς του, δεν ανιχνεύθηκε.

Στο δεύτερο στάδιο η γλώσσα προγραμματισμού C++/CLI που επιλέχθηκε αποδείχθηκε ιδιαίτερα σημαντική καθώς μπορούσε να συνδυάσει τους μηχανισμούς anti-Analysis σε συνδυασμό με τους σύγχρονους μεθόδους κρυπτογράφησης. Έτσι ενσωματώθηκαν οι μηχανισμοί αποφυγής ανάλυσης που υλοποιήθηκαν και εφαρμόστηκε επιπλέον κωδικοποίηση των αλφαριθμητικών. Μετά και τις προσθήκες στο στάδιο αυτό, προέκυψε η εκδοχή [ii]. Στη στατική ανάλυση, η εκδοχή [ii] ενισχύθηκε στο κομμάτι των αλφαριθμητικών (strings), χωρίς την σπουδαία αύξηση της εντροπίας αλλά δεν υπήρξε βελτίωση στην εξέταση με Disassemblers. Στην δυναμική ανάλυση υπήρξε βελτίωση σχεδόν σε όλους τους τομείς αφού στο σύνολο των περιπτώσεων δεν γινόταν η εκτέλεση του κακόβουλου μέρους του love4lock, με αποτέλεσμα να μην μπορεί να πραγματοποιηθούν οι κατάλληλοι έλεγχοι.

Γενικότερα, αξίζει να σημειωθεί ότι οι μηχανισμοί Anti-Analysis που ενσωματώθηκαν στο λογισμικό «υποψίασε» περισσότερα Antivirus όμως έγινε δυσκολότερη η περαιτέρω ανάλυση.

Στο τρίτο στάδιο χρησιμοποιήθηκε το πρόγραμμά obfuscator / packer ανοιχτού κώδικα ConfuserEX. Στο πλαίσιο αυτού του σταδίου, μεταξύ άλλων πραγματοποιήθηκε «συσκότιση» του εκτελέσιμου για να προκύψει η εκδοχή [iii]. Στην στατική ανάλυση η εκδοχή [iii] έφερε αύξηση της εντροπίας του προγράμματος επομένως αυτό είχε σαν συνέπεια ο love4lock να αναγνωρίζεται ως ιός από ορισμένα AVs. Στον αντίποδα όμως δεν μπορούν να αναγνωριστούν οι συναρτήσεις και οι μέθοδοι που έχουν χρησιμοποιηθεί ή έχουν αναπτυχθεί και το «spaghetti code» που δημιουργείται δυσκολεύει την ανάλυση του προγράμματος αισθητά στους Disassemblers.

Στο τέταρτο και τελευταίο στάδιο έγινε χρήση ενός ισχυρότερου obfuscator / packer, ονόματι Themida. Προστέθηκαν πολύ περισσότεροι μέθοδοι «συσκότισης» και προέκυψε η εκδοχή [iv]. Αποτέλεσμα του τέταρτου σταδίου ήταν η εντροπία του προγράμματος να αυξηθεί ακόμα περισσότερο και πλέον το μεγαλύτερο ποσοστό των antivirus να αναγνωρίζουν το love4lock ως ιό. Η ανάλυσή του είναι σχεδόν αδύνατη, εκτός αν προηγηθεί κάποιο στάδιο unpacking. Η εκδοχή [iv] θα μπορούσε κανείς να αναφέρει ότι έχει όλες τις ενδείξεις ότι πρόκειται για ιό, δεν μπορεί όμως να αναλυθεί ώστε να αναγνωριστεί το είδος του και με ποιόν τρόπο λειτουργεί.

Κατά την διάρκεια τόσο της στατικής, όσο και της δυναμικής ανάλυσης όλων των εκδοχών παρατηρήθηκαν και επιπλέον διαπιστώσεις. Η πρώτη είναι ότι ένας από τους σημαντικότερος λόγος που γίνεται αντιληπτός ο love4lock από τα AVs δεν ο τρόπος υλοποίησής ή η συμπεριφορά του, αλλά η χρήση του obfuscator / packer. Συνεπώς, η επιλογή του packer αποτελεί σημαντική απόφαση, αφού μπορεί να δείξει «false alarms» ακόμη και για προγράμματα που δεν αναπτύσσονται για κακόβουλο σκοπό. Επίσης, η χρήση μεμονωμένων εργαλείων δεν προσφέρει ικανοποιητικές πληροφορίες για την αναγνώριση των επιβλαβών λογισμικών αφού μπορούν να αποκλειστούν από μηχανισμούς Anti-Analysis. Ο συνδυασμός χρήσης πολλών και διαφορετικών εργαλείων καθώς και η εμπειρία του Αναλυτή είναι κρίσιμα στοιχεία για την επιτυχή αναγνώριση και ανάλυση του επιβλαβούς λογισμικού.

Εξετάζοντας όλες τις εκδοχές καταλήγουμε στο συμπέρασμα ότι τα λειτουργικά χαρακτηριστικά που θα προσδοθούν σε ένα επιβλαβές λογισμικό εξαρτώνται από τις

βλάβες που προορίζονται να προκαλέσουν και από το βάθος της ανάλυσης που στοχεύουν να αποφύγουν.

Όπως επικυρώνουν και τα αποτελέσματα των δοκιμών και πειραμάτων η ενσωμάτωση των μηχανισμών Anti-Analysis που παρουσιάσθηκαν παραπάνω κατέστησαν το ransomware love4lock ανθεκτικό στη ανάλυση και «έξυπνο» στη συμπεριφορά αφού το κακόβουλο μέρος της εφαρμογής εκτελείται μόνο σε ασφαλή περιβάλλοντα.

9.2 Μελλοντικές Επεκτάσεις

Στην μελλοντικές επεκτάσεις της παρούσας μεταπτυχιακής διατριβής προτείνεται η ανάπτυξη μιας εφαρμογής packer / obfuscator που να επιτυγχάνει την «συσκότιση» και την συμπίεση του εκτελέσιμου σε διαβαθμίσεις με επιπλέον την ενσωμάτωση επιλογής μηχανισμών anti-Analysis. Επιπρόσθετα είναι απαραίτητη η ενσωμάτωση ειδικού μηχανισμού μετατροπής των ονομασιών –μιας έξυπνης αυτόματης μετονομασίας- των strings με τέτοιο τρόπο ώστε να μην πραγματοποιείται η ανάγνωση από τον Αναλυτή αλλά και ταυτόχρονα να μην υποψιάζεται την όποια κρυπτογράφηση τους.

Είναι σημαντικό, το σύνολο της παραπάνω ανάπτυξης να γίνει με γνώμονα την μικρότερη απόκλιση και επηρεασμού του μεγέθους της εντροπίας του αρχικού εκτελέσιμου.

10 Βιβλιογραφία

[1] *Moore C., Detecting ransomware with honeypot techniques», Proceedings - 2016 Cybersecurity and Cyberforensics Conference, CCC 2016, 7600214, pp. 77-8.*

[2] *Scaife N., Carter H., Traynor P., Butle K.R.B., CryptoLock (and Drop It): Stopping Ransomware Attacks on User Data», Proceedings - International Conference on Distributed Computing Systems Volume 2016-August, Article number 7536529, Pages 303-312.*

[3] *Gazet A., Comparative analysis of various ransomware virii, Journal in Computer Virology 6 (1), pp. 77-90, 2010.*

[4] *Skoudis, E., & Zeltser, L. (2004). Malware Fighting Malicious Code. New Jersey: Prentice Hall..*

[5] *Zeltser, L. (2007). Reverse Engineering Malware: Tools and Techniques Hands-On. Bethesda: SANS Institute..*

[6] *Michael Sikorski, Andrew Honig-Practical Malware Analysis_ The Hands-On Guide to Dissecting Malicious Software-No Starch Press (2012).*

[7] *Garfinkel, T., Adams, K., Warfield, A., Franklin, J.: Compatibility is not transparency: VMM detection myths and realities. In: Proceedings of the 11thWorkshop on Hot Topics in Operating Systems (2007)].*

[8] *Raffetseder, T., Kruegel,C., Kirda, E.:Detecting system emulators.In: Proceedings of the 10th Information Security Conference, pp.1–18 (2007).*

[9] *Oyama Y., Trends of anti-analysis operations of malwares observed in API call logs, Journal of Computer Virology and Hacking Techniques pp. 1-17, 2017.*

[10] *Ferrie, P.: The ultimate anti-debugging reference, p 14. Tech. rep. (2011).*

[11] *Gao S., Lin Q.,Xia M., Yu M., Qi Z., Guan,H.: Debugging classification and anti-debugging strategies. In: Fourth International Conference on Machine*

Vision (ICMV 11), pp. 83503C–83503C. International Society for Optics and Photonics (2011).

[12] *Chen, X., Andersen, J., Mao, Z. M., Bailey, M., Nazario, J.: Towards an understanding of anti-virtualization and antidebugging behavior in modern malware. In: The 38th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, DSN2008, June 24–27, 2008, Anchorage, Alaska,USA, pp. 177–186 (2008).*

[13] *Linn, C., Debray, S.K.: Obfuscation of executable code to improve resistance to static disassembly. In: Proceedings of the 10th ACM Conference on Computer and Communications Security, CCS 2003, ACM, Washington, DC, October 27–30, 2003, pp. 290–299.*

[14] *Li A, Zhang, Y., Zhang, J. , Zhu, G. , A token strengthened encryption packer to prevent reverse engineering PE files, Proceedings of 2015 International Conference on Estimation, Detection and Information Fusion, ICEDIF 2015 7280213, pp. 307-312.*

[15] *Liță, C.V., Cosovan, D., Gavriluț, D., Anti-emulation trends in modern packers: a survey on the evolution of anti-emulation techniques in UPA packers, Journal of Computer Virology and Hacking Techniques pp. 1-20,2017.*

[16] "Malware Definition: https://techterms.com/definition/malware," [Online]. [Accessed 03 2017].

[17] *Christopher Elisan (5 September 2012): Malware, Rootkits & Botnets A Beginner's Guide. McGraw Hill Professional. pp. 10–. ISBN 978-0-07-179205-9.,* 2012.

[18] "AV-TEST Institute: https://www.av-test.org/en/statistics/malware/," [Online]. [Accessed 03 2017].

[19] "Botnet: https://www.techopedia.com/definition/384/botnet," 03 2017. [Online].

[20] *The Art of Memory Forensics: Detecting Malware and Threats in Windows, Linux, and Mac Memory 1st Edition by Michael Hale Ligh (Author), Andre.*

[21] *Rootkits: Subverting the Windows Kernel Paperback – July 22, 2005 by Greg Hoglund (Author), Jamie Butler (Author).*

[22] *"Symantec Security Response: Misleading Applications". Symantec. 2007-08-31. Retrieved 2017-03.*

[23] *Mehmood, Shafqat (3 May 2016). "Enterprise Survival Guide for Ransomware Attacks". SANS Information Security Training | Cyber Certifications | Research. www.sans.org.*

[24] "Comparison of Desktop Application Launchers: https://en.wikipedia.org/wiki/Comparison_of_desktop_application_launchers," [Online]. [Accessed 03 2017].

[25] *"Keylogger". Oxford dictionaries..*

[26] *Mackeown, Patrick (10 August 2006). "Bookscape: Short Story - Famous Computer Hoaxes". Bookscape. Archived on 13 November 2010..*

[27] *Tulloch, Mitch (2003). Koch, Jeff; Haynes, Sandra, eds. Microsoft Encyclopedia of Security. Redmond, Washington: Microsoft Press. p. 16. ISBN 0-7356-1877-1. Any software that installs itself on your system without your knowledge and displays advertisements when the user browses the Internet..*

[28] *Spyware Workshop: Monitoring Software On Your Personal Computer: Spyware, Adware, and Other Software: Report of the Federal Trade Commission Staff March 2005.*

[29] *"Symantec Report on Rogue Security Software" (PDF). Symantec. 2009-10-28. Retrieved 2017-03.*

[30] *Browser Hijacking Fix & Browser Hijacking Removal". Microsoft. Retrieved 2017/03..*

[31] "Flooder: https://www.f-secure.com/v-descs/flooder.shtml," [Online].

[32] *"Computer Viruses – Theory and Experiments", Fred Cohen, 1984.*

[33] *Hackers Proof: The Ultimate Guide to Network Security, Lars Klander (1997).*

[34] "Wikipedia, Σύστημα Ανίχνευσης Εισβολής: https://el.wikipedia.org/wiki/Σύστημα_Ανίχνευσης_Εισβολής," [Online].

[35] *Ζορκάδης Βασίλειος, Θεωρία Πληροφορίας και Κωδικοποίησης, 2002.*

[36] *Evasive Malware Exposed and Deconstructed, Christopher Kruegel, San Francisco, RSA Conference 20-24/04/2015.*

[37] "Common Language Runtime: https://en.wikipedia.org/wiki/Common_Language_Runtime," [Online]. [Accessed 03 2017].

[38] "SHA-2: https://en.wikipedia.org/wiki/SHA-2," [Online]. [Accessed 03 2017].

[39] "Virus Bulletin, Glossary Entry on a Packer: http://www.virusbtn.com/resources/glossary/packer.xml," [Online]. [Accessed 03 2017].

[40] "Ultimate Packer for Executables, Overview: http://upx.sourceforge.net/," 03 2017. [Online].