VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

BAKALÁŘSKÁ PRÁCE

Brno, 2019 Samuel Sidor

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ
DEPARTMENT OF TELECOMMUNICATIONS

ANALÝZA A DETEKCE MALWARU TYPU RAT
ANALYSIS AND DETECTION OF RAT MALWARE

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **Samuel Sidor**
AUTHOR

VEDOUCÍ PRÁCE — **doc. Ing. Jan Hajný, Ph.D.**
SUPERVISOR

BRNO 2019

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní obor **Informační bezpečnost**
Ústav telekomunikací

Student: Samuel Sidor ***ID:*** 195167
Ročník: 3 ***Akademický rok:*** 2018/19

NÁZEV TÉMATU:

Analýza a detekce malwaru typu RAT

POKYNY PRO VYPRACOVÁNÍ:

Studujte problematiku škodlivého kódu (malware) a jeho jednotlivých typů. Zaměřte se především na tzv. typ RAT (z angl. Remote Access Trojan). Seznamte se s metodami statické i dynamické analýzy binárního spustitelného kódu jako jsou reverzní inženýrství, sandboxing, dekompilace atd. Po domluvě s vedoucím a konzultantem detailně analyzujte vybrané malware kmeny. Studujte techniky v nich obsažené a metody jak se proti nim bránit. Navrhněte a vytvořte přesné detekční vzory (alespoň 10) pro tyto kmeny a techniky v nich obsažené v některém ze standardních jazyků pro detekci malwaru (např. YARA, Snort atd.).

DOPORUČENÁ LITERATURA:

[1] Sikorski, Michael, and Andrew Honig. Practical Malware Analysis : a Hands-On Guide to Dissecting Malicious Software. San Francisco: No Starch Press, 2012. Print.

[2] Eilam, Eldad, and Elliot J. Chikofsky. Reversing : secrets of reverse engineering. Indianapolis, IN: Wiley, 2005. Print.

Termín zadání: 1.2.2019 ***Termín odevzdání:*** 27.5.2019

Vedoucí práce: doc. Ing. Jan Hajný, Ph.D.
Konzultant: Jakub Křoustek (AVAST Software s.r.o.)

prof. Ing. Jiří Mišurec, CSc.
předseda oborové rady

ABSTRAKT

Cieľom tejto bakalárskej práce je štúdium problematiky jednotlivých typov malvéru so špecifickým zameraním na kategóriu RAT (Remote Access Trojan). Práca taktiež oboznámi čitateľa o statickej a dynamickej analýze binárneho spustiteľného kódu a pojmami ako reverzné inžinierstvo, sandboxing, dekompilácia a pod. Ďalej sa v práci analyzujú vybrané malvérové rodiny na ktoré sú vytvorené detekčné pravidlá v jazyku YARA. Okrem toho bude čitateľ oboznámeny aj s ochranou proti RAT malvéru a nakoniec budú vyhodnotené dáta získané z podrobného skúmania jednotlivých malvérových vzoriek.

KĽÚČOVÉ SLOVÁ

Malvér, RAT, Remote Access Trojan, Reverzné inžinierstvo, YARA

ABSTRACT

Goal of this bachelor's thesis is studying problematics of various types of malware with specific focus on RAT (Remote Access Trojan) category. This thesis will also acquaint reader with static and dynamic binary analysis and terms like reverse engineering, sandboxing, decompilation, etc. Then chosen malware families will be analysed and for these families detection rules in YARA language will be created. Except this, reader will be acquainted also with protection against RAT malware and finally data acquired from detail analysis will be evaluated.

KEYWORDS

Malware, RAT, Remote Access Trojan, Reverse engineering, YARA

SIDOR, Samuel. *Analýza a detekce malwaru typu RAT*. Brno, 2019, 47 s. Bakalárska práca. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací. Vedúci práce: doc. Ing. Jan Hajný, Ph.D.

VYHLÁSENIE

Vyhlasujem, že som svoju bakalársku prácu na tému „Analýza a detekce malwaru typu RAT“ vypracoval samostatne pod vedením vedúceho bakalárskej práce, využitím odbornej literatúry a ďalších informačných zdrojov, ktoré sú všetky citované v práci a uvedené v zozname literatúry na konci práce.

Ako autor uvedenej bakalárskej práce ďalej vyhlasujem, že v súvislosti s vytvorením tejto bakalárskej práce som neporušil autorské práva tretích osôb, najmä som nezasiahol nedovoleným spôsobom do cudzích autorských práv osobnostných a/alebo majetkových a som si plne vedomý následkov porušenia ustanovenia § 11 a nasledujúcich autorského zákona Českej republiky č. 121/2000 Sb., o práve autorskom, o právach súvisiacich s právom autorským a o zmene niektorých zákonov (autorský zákon), v znení neskorších predpisov, vrátane možných trestnoprávnych dôsledkov vyplývajúcich z ustanovenia časti druhej, hlavy VI. diel 4 Trestného zákoníka Českej republiky č. 40/2009 Sb.

Brno ....

podpis autora

POĎAKOVANIE

Rád by som poďakoval vedúcemu bakalárskej práce pánovi doc. Ing. Janu Hajnému, Ph.D. za užitočné pripomienky, ochotu a usmernenie pri písaní práce. Taktiež ďakujem Ing. Jakubovi Křoustokovi, Ph.D. zo spoločnostni Avast za odborné konzultácie, trpezlivosť, podnetné návrhy a cenné rady.

Brno ....

podpis autora

Obsah

Zoznam obrázkov

Úvod

V súčastnosti sú informačné technológie neoddeliteľnou súčasťou našich životov. Avšak každý používateľ by mal byť pri dennodennej interakcii s online technológiami ostražitý, lebo na neho striehne veľa nebezpečenstiev a jedným z ních sú aj škodlivké kódy (malvér). Nielenže malvér môže využívať veľkú časť výkonu zariadenia, ale okrem toho môže napadnutú obeť do značnej miery poškodiť. Spomínaný škodlivý softvér sa rozdeľuje na viacero druhov. Nakoľko je v dnešnej dobe čím ďalej tým viac populárny typ Remote Access Trojan tzv. RAT, tak sme sa rozhodli zamerať práve naň. Motiváciou tejto práce je zanalyzovať existujúce malvérové rodiny typu RAT a navrhnúť konkrétne spôsoby ich presnej detekcie.

V tejto práci sa budeme zameriavať na RATy z viacerých pohľadov. Jedním z nich je samotná funkctionalita vďaka ktorej budeme schopní vytvoriť detekčné pravidlá pomocou nástroju YARA, ktorý sa v dnešnej dobe berie ako štandard pri detekcii malvérov na základne statických častí alebo behaviorálnych prvkov. Detekcia malvéru je nekonečný boj, ktorý má dlhú históriu. Bezpečnostný experti bojujú s týmto problémom dennodenne a vo svojej podstate je to len hra na mačku a myš, keďže útočníci neustále vytvárajú nové spôsoby, ako sa vyhnúť detekcii a s tým súvisí neustále prispôsobovanie ochranných mechanizmov ako napr. antivírusových programov na tieto nové antidetekčné praktiky. Mimo detekcie je naším cieľom dozvedieť sa o každom analyzovanom malvéri čo najviac zaujímavých informácií ako napríklad meno jeho tvorcu, krajinu pôvodu, programovací jazyk, špeciálne funkcie a iné, ktoré následne vyhodnotíme. Okrem toho nám detailné informácie z analýzy pomôžu aj pri tvorbe detekčných pravidiel, pretože len vďaka komplexným znalostiam môžeme navrhovať komplexné riešenia. Napríklad zistíme na ktoré programovacie jazyky má zmysel vytvárať alebo upravovať analytické nástroje, aké vlastnosti sledovať pri sandboxingu atď.

V prvej kapitole oboznámime čitateľa so základným delením škodlivého kódu ako aj podrobnejším popisom RAT.

Druhá kapitola oboznámi čitateľa s pojmami ako reverzné inžinierstvo, statická analýza, dynamická analýza, sandboxing ako aj používané analyzačné nástroje.

Kapitola tri načrtne historický vývoj problematiky RAT malvéru a bude riešiť najčastejšie sa vyskytujúce rodiny podľa ich časového výskytu.

Štvrtou kapitolou sa presunieme do samotnej analýzy malvérových vzorkov, akými spôsobmi sú analyzované, ako sú na základe analýzy vytvorené detekčné pravidlá a samozrejme ako sa proti tomuto typu malvéru brániť.

V poslednej kapitole budú zhrnuté výsledky analýzy najčastejších RAT, vyhodnotenie výskytu jednotlivých vlastností a funkcií ako aj účinnosti naších detekčných pravidiel.

1 Taxonómia škodlivého kódu

Škodlivý kód taktiež nazývaný ako malvér, je softvér zámerne navrhnutý tak, aby spôsobil škodu počítaču, serveru alebo počítačovej sieti. Tento malvér začne vykonávať svoj kód po tom, čo sa nejakým spôsobom dostane na jedno zo spomínaných zariadení. Môže mať formu spustiteľného kódu, skriptu, makra a podobne. Zvyknú sa tu zaraďovať aj programy, ktoré boli napísané rôznymi spoločnosťami, ale ich kód tajne koná proti záujmu používateľov. Z počiatku bol prvý malvér písaný len ako súčasť experimentovania s novými technológiami, alebo ako druh žartu [1].

1.1 Delenie škodlivého kódu

Najznámejšie typy malvéru, ako napríklad vírusy a červy sú známe hlavne kvôli spôsobu ich šírenia ako kvôli špecifickej funkcionalite.

Potreba definovať jednotnú terminológiu pri škodlivom softvéri je stará ako samotné počítače. Problém pri zaradzovaní malvéru sa komplikuje už aj v tom, že viaceré kategórie majú isté spoločné vlastnosti. Nižšie uvedené rozdelenie nevylučuje, že malvér môže patriť aj do viac ako jednej kategórie [1]. Z dôvodu prehľadnosti nebudeme uvádzať zdroj [1] pri každom type rozdelenia malvéru. Z toho vyplýva, že text v tejto podkapitole ku ktorému nie je priradená citácia je súčasťou tohto zdroja.

- Vírusy – Počítačový vírus je kód, ktorý sa rekurzivne replikuje za účelom šírenia s tým, že každá jeho replika môže byť istým spôsobom vyvinutejšia. Vírusy infikujú hostiteľský súbor, systémovú oblasť alebo jednoducho upravjú odkaz na takéto objekty, aby prevzali kontrolu a potom sa ďalej rozmnožili.
- Červy – Sú sieťové vírusy, ktoré sa replikujú v sieťach a vyskytujú sa zvyčajne ako samostatné programy bez hostiteľského programu. Obvykle sa červ spustí sám automaticky na vzdialenom počítači bez akejkoľvek interakcie používateľa. Existujú však aj také, napr. poštové červy, ktoré sa nevykonajú bez manuálneho spustenia.
- Logické bomby – Medzi tzv. logické bomby patria legitímne programy, ktoré obsahujú istý druh poruchy alebo špeciálneho správania, ktoré môžu napr. v prípade vypršania licencie vymazať celý program z disku, v prípade cracknutia sa môže vykonať kód, ktorý zásadne poruší systém daného používateľa a veľa ďalších.
- Exploity – Kód exploitov je špecifický k jednotlivým zraniteľnostiam alebo skupinám zraniteľností, na ktoré je použitý. Cieľom exploitov je spustiť malvér na zväčša vzdialenom zariadení automaticky bez nutnosti zásahu používateľa alebo poskytnúť útočníkovi vysoký druh privilegovaného prístupu.

- Downloadre – Sú ďalší z druhu malvérových programov, ktorý sťahuje ďalší škodlivý obsah z webových stránok alebo iných online umiestnení. Po stiahnutí tieto položky následne spustí. Zvyčajne býva zaslaný formou e–mailu.
- Droppery – Originálny názov tejto kategórie bol „inštalátor“. Najjednoduchšie sa dá dropper popísať ako program, ktorý pomáha inému malvérovému programu s nainštalovanim do systému. Obzvlášť, keď boli v minulosti písané malvéry v binárnej forme a na prvotné spustenie a nainštalovanie bol nevyhnutný dropper. Po nainštalovaní sa mohol už daný vírus šíriť sám.
- Injectory – Sú špeciálnym druhom dropperov, ktoré zvyčajne inštalujú kód vírusov priamo do pamäte. Injector môže byť použitý taktiež na vloženie vírusu v aktívnej forme do obsluhy prerušenia disku. Sieťový injector je špeciálny poddruh a slúži na vloženie alebo upravenie dát v sieťovej komunikácii.
- Kity – Tvorcovia vírusov vytvárajú tzv. Kity, ktoré slúžia na generovanie nových počítačových vírusov automaticky za použitia jednoduchého grafického menu. S týmito nástrojmi sú schopní aj ľudia neznalí danej problematiky vytvárať dosť nebezpečné programy.
- Spammery – Spammery alebo tzv. spamovacie programy sa používajú na odosielanie nevyžiadaných správ veľkému množstvu ľudí, na rôzne fóra, SMS a podobne. Hlavnou motiváciou tvorcov týchto programov je zarábať peniaze generovaním návštevnosti na webových stránkach alebo zúčastňovaním sa na rôznych affiliate programoch. Okrem toho sa nevyžiadaná pošta používa aj na šírenie phishingových stránok.
- Floodery – Hackeri používajú floodery na zaplavovanie sieťovej prevádzky s veľkým množstvom dát alebo požiadavkou a tým vykonávajú známy útok Denial of Service(DoS) tzv. odmietnutie služby.
- Keyloggery – Keylogger zachytáva stlačené klávesy na kompromitovanom systéme a tým pádom sa útočník dostane k užívateľským menám, heslám, číslam kreditných kariet, osobným údajom a podobne. Tieto dáta si útočník buď fyzicky skopíruje zo zariadenia alebo mu ich keylogger pošle na nejaký druh internetového umiestnenia (cloud, FTP). Často sa keyloggery používajú na krádež identity.
- Rootkity – Sú špeciálny súbor hackerských nástrojov, ktoré sú použité po tom, čo sa útočník dostal do systému na získanie administrátorských práv. V prípade, že sa nainštalujú ako modifikovaná verzia bežného nástroju, hovoríme o rootkite bežiacom v užívateľskom móde. Sofistikovanejšie rootkity bežia v móde jadra (kernel–móde) a sú podstatne nebezpečnejšie, pretože môžu zmeniť správanie jadra operačného systému.
- Žartovné programy – Zvyčajne nie sú škodlivé a veľakrát rozdiel medzi skutočným vírusom a žartovným programom záleží len na zmysle pre humor danej

obete. Tieto programy menia správanie napadnutého systému a vytvárajú istý druh rozptýlenia, alebo množstvo nepríjemných akcií, ako napríklad náhodné vypínanie obrazovky, otáčanie obrazovky a podobne.

- Adware a Spyware – V nedávnej dobe sa objavili nové druhy aplikácií ako priamy dopad na zvýšenia používania internetu na služby ako nakupovanie a zábava. Veľa firiem zaujíma, čo návštevníci internetu prehliadajú a aké produkty zvyknú kupovať. Preto sa rozšírili malé programy, ktoré sledujú používateľské správanie a na jeho základe sa na napadnutom systéme objavujú vyskakovacie okna s produktami, ktoré by ich mohli zaujímať.
- Coinminery – S veľkým zviditeľnením kryptomien za posledné 2 roky (2017 – 2018) sa objavilo veľa tzv. coinminerov, ktoré po napadnutí systému využívajú obrovské množstvo výpočtového výkonu na ťaženie kryptomien pre útočníkov. Tieto škodlivé programy priniesli hneď niekoľko problémov napadnutým obetiam. Systém na ktorom bežali začal byť spomalený a vďaka využívaniu veľkého výkonu sa drasticky opotrebovával hardvér čím sa zvýšil odber elektrickej energie.
- Ransomware – Ransomware je špecifický typ škodlivého softvéru, ktorý vydiera svoje obete. Vo väčšina prípadov tento malvér zašifruje všetky data na disku a pokiaľ do istej doby obeť nezaplatí za dešifrovanie, tak jej dáta budú nenávratne stratené. Napádané zariadenia môžu byť počítače, smartfóny, databázy a iné.
- Trojské kone – Pravdepodobne najjednoduchší druh škodlivého programu je práve Trojský kôň. Trojský kôň sa pokúša javiť ako dôveryhodná aplikácia, alebo sa snaží inak zaujať užívateľa so zaujímavými funkciami za účelom spustenia. V iných prípadoch sa snažia tvorcovia malvéru priamo do dôveryhodných aplikácií vložiť tieto trójske kone za účelom maskovania a zvýšenia vierohodnosti sťahovanej aplikácie a presvedčenie užívateľa, že sťahuje skutočnú aplikáciu, ktorá aj zároveň splní účel za ktorým bola sťahovaná.
- Password–Stealing Trojans – Táto podkategória trojanov je natoľko rozsiahla, že sa už zvyčajne zaraďuje zvlášť. Zameriava sa výhradne na kradnutie uložených hesiel aplikácií na kompromitovanom systéme a následne ich odosiela útočníkovi. Často bývajú tieto trojany kombinované aj s keyloggermi.
- Backdoor – Ďalšia veľká podkategória trojanov. Tiež je známy ako „zadné vrátka“ čo predstavuje aplikáciu povoľujúcu vzdialené pripojenie na počítač. Rozdiel medzi týmto typom malvéru a legitimnou aplikáciou s podobnou funkcionalitou je ten, že inštalácia backdooru je bez vedomia používateľa. Typická funkcionalita obsahuje možnosť posielať súbory na napadnutý počítač, spúšťanie príkazov a súborov ako aj spätné zasielanie dát útočníkovi [2].
- RAT (Remote Access Tool/Trojan) – Je typ backdooru, cez ktorý môže útočník

prevziať kontrolu nad napadnutým systémom. Tieto RAT malvéry sú inštalované na počítačoch obete ako aj na počítači útočníka. Bližšie si tento druh malvéru popíšeme v kapitole Analýza RAT [3]. U obete býva nainštalovaný RAT server a RAT klient naopak na zariadení útočníka. Funkcie RAT trojanov sa líšia, ale najčastejšie z nich sú:

- Blokovanie myši a klávesnice,
- sťahovanie, uploadovanie súborov a dát,
- manipulácia s procesmi,
- manipulácia so súbormi a diskami,
- manipulácia so systémovými prvkami,
- vzdialená plocha,
- keylogger,
- password–stealer,
- screenshoty a kamerové záznamy.

Následne vytvorený log súbor obsahujúci zachytené klávesy, heslá a vlastnosti systému je zaslaný útočníkovi [3].

Na správne fungovanie RAT sú nutné 2 programy. Klientsky program, ktorý beží na počítači útočníka a počúva komunikáciu vysielanú serverovým programom na špecifikovanom porte. Útočníková časť pozostáva z GUI (Grafického rozhrania) cez ktoré môže zasielať na RAT server rôzne príkazy na vykonanie útoku. Serverový program beží skrytý v pozadí na zariadení obete a pokúša sa nadviazať spojenie s útočníkom vždy, keď je online a na základe príkazov, ktoré obdrží, vykoná isté akcie. Spojenie je zvyčajne tvorené pomocou TCP komunikácie. RATy zvyknú využívať 2 spôsoby nadväzovania spojenia:

1. Priamé spojenie – V tomto prípade sa klient pripája k jednému alebo viacerým serverom priamo. Servery sú multivláknové a tým pádom umožňujú pripojenie viacerých klientov v jednom okamihu.
2. Reverzné spojenie – Má viacero výhod, ako napríklad to, že smerovače v tomto prípade neblokujú prichádzajúce dáta, pretože spojenie je nadviazané z vnútra von. Taktiež nám tento druh spojenia umožňuje hromadné aktualizovanie serverov broadcastovými príkazmi, práve kvôli tomu, že viacero serverov môže byť pripojených na 1 klienta.

Najčastejšími spôsobmi nákazy týmto škodlivým programom sú: nákaza prílohou v e–maile, vydávanie sa za legitímny a pre používateľa zaujímavý softvér alebo rôznymi spamovacími technikami [3].

2 Reverzné inžinierstvo

Reverzné inžinierstvo (RE) je proces porozumenia istému systému. Systém môže byť hardvérové zariadenie, softvérový program, fyzický alebo chemický proces a tak ďalej [4]. Pre účely tejto práce budeme považovať za systém softvérový program.

RE v tomto význame je teda technika používaná na analyzovanie softvéru za účelom zistenia z akých komponentov je zložený a aká je jeho funkcia. Zvyčajné dôvody na použitie RE sú znovuvytvorenie už existujúceho programu za účelom vylepšenia vlastného produktu, hľadanie zraniteľností programu alebo analýza malvéru.

2.1 Statická analýza

Základná statická analýza pozostáva z preskúmania spustiteľného súboru bez hlbšieho náhľadu na inštrukcie pre procesor. Už základnou statickou analýzou môžme veľakrát zistiť či sa jedná o škodlivý súbor alebo nám môže poskytnúť informácie o jeho funkcionalite. Tento typ analázy je jednoduchý, vcelku rýchly, ale je do značnej miery neúčinný proti sofistikovanému škodlivému softvéru lebo nám neodhalí všetko.

Pokročilá statická analýza pozostáva z RE internej štruktúry malvéru načítaním spustiteľného súboru do disassemblera vďaka ktorému získame náhľad na procesorové inštrukcie, vďaka ktorým môžme zistiť čo program robí. Avšak pokročilá statická analýza má podstatne strmšiu krivku učenia sa. Základná statická analýza vyžaduje špecializované vedomosti o disassemblovaní kódu, konštruktov a koncepcií operačného systému na ktorom sa vykonáva (v našom prípade OS Windows) [5].

Vo veľkom množstve prípadov je ale statická analýza nepostačujúca lebo väčšina malvéru (viac ako 90%) používa packery, obfuskátory alebo cryptery. Z toho dôvodu v takýchto prípadoch je nevyhnutné použiť dynamickú analýzu.

2.1.1 Disassembler

Disassemblery sú programy, ktorých vstup tvorý binárny tvar spustiteľného programu z ktorého následne generujú textový formát obsahujúci assemblerovské inštrukcie pre celý program, alebo pre jeho časti. Jedná sa relatívne o jednoduchý proces. Disassemblovaný kód je špecifický pre každý typ procesoru, ale niektoré disassemblery podporujú viacero procesorových architektúr. Kvalitný disassembler je kľúčový komponent v zozname nástrojov každého reverzného inžiniera, avšak niektorí z nich preferujú vstavané disassembery, ktoré sú súčasťou istých nízko úrovňových debuggerov [6].

2.1.2 Dekompilátor

Dekompilátor vezme spustiteľný binárny súbor a pokúša sa z neho vytvoriť čitateľný kód vo vysokoúrovňovom jazyku. Cieľom je pokúsiť sa obrátiť proces kompilácie, získať pôvodný zdrojový kód, alebo kód jemu podobný. Na veľkej väčšine platforiem proces obnovenia pôvodného zdrojového kódu nie je možný. Dôvod prečo je tomu tak je, že kompilátor jazyka vynecháva veľakrát rôzne dôležité prvky a je nemožné ich presne obnoviť. Dekompilátory sú nástroje, ktoré v niektorých situáciách a prostrediach môžu rekonštruovať z binárneho programu dostatočne čitateľný zdrojový kód [6].

2.2 Dynamická analýza

Základné techniky dynamickej analýzy zahŕňajú spustenie malvéru a pozorovanie jeho správania v systéme s cieľom jeho odstránenia, vytvárania efektívnych detekčných vzorov alebo oboch vecí zároveň. Pred tým, ako môžeme bezpečne spustiť škodlivý softvér, je potreba nastaviť testovacie prostredie, ktoré umožňuje študovať beh škodlivého softvéru bez rizika poškodenia vlastného systému alebo siete. Rovnako ako základné statické metódy analýzy, základné metódy dynamickej analýzy môžu používaj aj ľudia bez hlbších znalostí programovania. Avšak tieto spôsoby analýzy nemusia byť účinné pri všetkých malvéroch, lebo nie vždy zobrazujú všetky ich činnosti v systéme.

Pokročilá dynamická analýza používa ladiaci nástroj (tiež nazývaný debugger) na spustenie a preskúmanie vnútorného stavu škodlivého spustiteľného súboru. Pokročilé techniky dynamickej analýzy poskytujú ďalší spôsob, ako získať podrobné informácie zo spustiteľného súboru. Tieto techniky sú najužitočnejšie, pri snahe získať informácie, ktoré je komplikované ba až nemožné získať inými technikami. Najlepšou kombináciou v rámci analýzy je použitie pokročilej dynamickej analýzy spolu s pokročilou statickou analýzou s cieľom získať všetky potrebné informácie [5].

2.2.1 Debugger

Základnou myšlienkou Debuggeru (ladiaceho programu) je, že si programátori nedokážu predstaviť všetko čo ich program môže vykonať. Programy sú zvyčajne príliš zložité na to, aby človek skutočne predpovedal každý potenciálny výsledok a práve preto sú tu debuggery. Debugger je program, ktorý umožňuje vývojárom sledovať ich softvér počas behu. Dve najzákladnejšie funkcie v debuggery sú schopnosť nastaviť breakpointy (body prerušenia) a schopnosť sledovať kód. Pre reverzného analytika je debugger takmer rovnako dôležitý, ako pre softvérových vývojárov, ale pre

jemne odlišné dôvody. V prvom rade používajú reverzní analytici debuggery v režime disassembleru. V tomto režime používa debugger vstavaný disassembler na dissasemblovanie objektového kódu za behu. Analytici môžu prechádzať disassemblovaným kódom a v podstate sledovať procesor ako spracúvavá program inštrukciu po inštrukcii a daný program si zastavovať v prípade potreby [6].

2.3 Sandboxing

Sandboxing je bezpečnostný mechanizmus na spúšťanie nedôveryhodných alebo potencionálne škodlivých programov v bezpečnom prostredí bez obáv z poškodenia hlavného systému. Sandboxy obsahujú virtualizované prostredie, ktoré zvyčajne simuluje sieťové služby, aby sa zaistilo, že testovaný softvér bude fungovať normálne. Veľa malvérových sandboxov, ako napríklad: Norman SandBox, Cuckoo, GFI Sandbox, Anubis, Joe Sandbox, ThreatExpert, BitBlaze, a Comodo Instant Malware Analysis umožňujú analyzovať malvér zadarmo. V súčasntej dobe sú Norman SandBox, Cuckoo a GFI Sandbox najpopulárnejšie medzi profesionálmi v rámci informačnej bezpečnosti. Tieto sandboxy poskytujú ľahko zrozumiteľný výstup a sú skvelé na získanie prehľadu o funkcionalite v nich spusteného softvéru [5].

2.4 Nástroje

- Process Explorer – zobrazuje informácie o procesoch, ich referenciách na prostriedky (resources) a DLL súboroch, ktoré boli načítané alebo otvorené. Grafické prostredie pozostáva z 2 hlavných podokien. Horné okno vždy zobrazuje zoznam aktuálne bežiacich procesov. Tento zoznam taktiež môže obsahovať použité prostriedky, popis, vlákna a podobne. Spodné okno pozostáva z informácií podľa aktuálne zvoleného módu [7].
- Process Monitor – je pokročilý monitorovací nástroj pre Windows, ktorý zobrazuje systém súborov v reálnom čase, ako aj aktivitu procesov a registrov. Spája funkcie dvoch starších nástrojov Filemon a Regmon. Okrem toho pridáva rozsiahlý zoznam vylepšení vrátene komplexného filtrovania, vlastností udalostí ako sú ID relácií, informácie o procese, súčasné zaznamenávanie do súboru a oveľa viac [8].
- PEview – poskytuje rýchly a jednoduchý spôsob zobrazenia štruktúry a obsahu 32–bitových súborov Portable Executable (PE) a Component Object File Format (COFF). Dokáže zobraziť hlavičky, sekcie, adresáre, tabuľku importov, tabuľku exportov v rámci EXE, DLL, OBJ, LIB, DBG a iných typov súborov.

- Detect It Easy (DIE) – je program určujúci typ súborov. Je to aplikácia, ktorá bola vytvorená ako identifikátor packerov. Tento nástroj je ľahko ovládateľný, rýchly a poskytuje širokú škálu nástrojov ako zobrazovanie reťazcov v binárnom súbore, zobrazenie binárneho kódu a mnoho ďalších [9].
- IDA PRO – tento nástroj kombinuje interaktívny programovateľný multiprocesorový disassembler spojený s lokálnym a vzdialeným debuggerom rozšírený o množstvo dodatočný doplnkov [10].
- x64dbg – je debugger pre 32 a 64 bitové aplikácie. Aktívne sa stále na ňom pracuje. Debugger má v súčastnosti 3 časti: DBG, GUI, Bridge. DBG je ladiaca časť debuggeru. Zaoberá sa ladením (pomocou nástroja TitanEngine) a poskytuje údaje pre GUI. GUI je grafická časť. Je postavené na vrchole Qt SDK a poskytuje interakciu používateľa. Bridge je komunikačná knižnica pre časť DBG a GUI (a do budúcna aj pre ďalšie časti). Môže byť použitý na prácu s novými funkciami, bez nutnosti aktualizácie kódu ostatných častí [11].
- DnSpy – je debugger a editor .NET assembly. Môže byť použitý na editovanie a debuggovanie assembly aj keď nemáme prístupný pôvodný zdrojový kód [12].
- Exe2Aut – je navrhnutý tak, aby bol najjednoduchší a najvšestrannejší dekompilátor pre AutoIt3 skripty. Na rozdiel od dekompilátora, ktorý je dodávaný s AutoIt3, Exe2Aut je dokonca schopný dekompilovať spustiteľné súbory, ktoré boli packnuté a chránené pomocou AutoIt3Camo, Themida, Armadillo, Safengine a podobne [13].
- De4dot – vydaný ako open source (GPLv3) .NET deobfuskátor a unpacker napísaný v C#. Jeho účelom je čo najlepšie obnoviť packnutý a obfuskovaný zdrojový kód do jeho pôvodného stavu. Väčšinu obfuskovania možno úplne obnoviť (napr. šifrovanie reťazcov). Na druhú stranu napríklad premenovanie symbolov a metód je zvyčajne nevratný proces, pretože pôvodné mená nebývajú súčasťou obfuskovaného súboru [14].
- YARA – je nástroj zameraný najmä na pomoc výskumným pracovníkom v oblasti malvéru na identifikáciu a klasifikáciu vzoriek škodlivého softvéru. S programom YARA je možné vytvárať popisy rodín škodlivého softvéru (ako aj iného rozličného softvéru). Tieto popisy inak nazývané taktiež detekčné pravidlá sú založené na textových alebo binárnych vzorkách. Každé pravidlo pozostáva zo sady reťazcov a booleovského výrazu, ktoré určujú jeho logiku [15].
- Cuckoo sandbox – je modulárny, open–source automatizovaný systém na analýzu malvéru. Dokáže analyzovať veľa rôznych škodlivých súborov (spustiteľné súbory, Office dokumenty, pdf súbory, e–maily a pod.), ako aj škodlivé webové stránky. Cuckoo nemá problém ani s analýzou sieťovej prevádzky dokonca aj v prípade, že je šifrovaná pomocou SSL/TLS. Okrem iného taktiež vykonáva pokročilú analýzu pamäte infikovaného virtuálneho systému [16].

3 Historický vývoj RAT malvéru

RATy sú súčasťou informačných technológií už niekoľko desaťročí počnúc programom Netsupport Manager, ktorý bol vytvorený ešte v roku 1989 [17]. Od tej doby prešli značným vývojom v rámci funkcionality ako aj dizajnu. Čo sa popularity týka radia sa medzi jedny z najpoužívanejších malvérov kvôli ich multifunkčnosti v oblasti sledovania a kradnutia osobných údajov.

3.1 Najaktívnejšie RAT rodiny

Množstvo všetkých malvérových rodín presahuje desiatky kusov. Bavíme sa o stovkách rôznorodých rodín nehovoriac o tom, že každá z týchto rodín máva veľké množstvo ďalších upravených mutácií. Samozrejme, niektoré RAT rodiny sú aktívnejšie ako iné a preto budeme klásť väčší dôraz práve na tie najčastejšie sa vyskytujúce. Táto analýza bude robená chronologicky podľa časového obdobia, kedy sa daná vzorka objavila a budú popísané jej základné údaje a prípadné špecifické funkcie.

Gh0st

Pravdepodobne jeden z najstarších aktuálne používaných RATov je práve Gh0st, ktorého vývoj sa datuje až do roku 2001. Bol vytvorený čínskou skupinou programátorov C.Rufus Security Team v programovacom jazyku Visual Basic [18]. Okrem klasických vlastností má možnosť na zabránenie zadávania vstupu užívateľom do napadnutého počítača.

Apocalypse

Hneď v niekoľkých verziách sa v roku 2009 objavil RAT Apocalypse napísaný v jazyku Delphi vývojármi s Tureckým pôvodom. Bolo zrejmé, že sa vývojar o svoje dielo príkladne staral, lebo vydával pravidelne záplaty na chyby v jeho RATe. Medzi špeciálne metódy nachádzajúce sa v tomto malvéri považujeme manipuláciu s prehliadačom obete alebo taktiež posielanie textových správ na jej zariadenie.

B0–K

Ďalšia rodina písaná v jazyku Visual Basic .NET je práve B0–K, ktorá sa objavila v roku 2012. Jej autor figuroval pod menom Majdi Saad a pochádzal pravdepodobne z Palestiny. Tento RAT nemal veľké množstvo funkcí a zrejme najzaujímavejšou z nich bolo získanie dôkladných informácií o počítači obete.

NjRAT

Rodina s najväčším množstvom mutácií s ktorými sme sa pri analýze stretli je práve rodina NjRAT taktiež známa ako Bladabindi ktorá bola vytovrená koncom roka 2012 hackerskou organizáciou Sparclyheason z ktorej najhlavnešou postavou bol člen známy ako Njq8 [19]. Táto rodina obsahuje všetku podstatnú funkcionalitu a jej typickým znakom je šírenie sa na disky, flash disky alebo sieťové zariadenia. Dôvod prečo sa táto rodina tak moc šírila je zapríčinený tým, že v roku 2013 bol jej zdrojový kód zverejnený na stránke GitHub a taktiež preto, že bol tento RAT napísaný v jednoduchom jazyku Visual Basic .NET.

LuminosityLink

Rok 2012 zakončuje známy Visual Basic .NET RAT LuminosityLink, ktorý bol avšak predávaný až od mája 2015. Do roku 2017 bolo predaných viac ako 8600 licencií, čo pri cene 40$ za kus činilo sumu 344 000$. Jeho vývojár známy pod prezývkou KFC Watermelon, pravým menom Colton Ray Grubbs narodený v roku 1997 v Kentucky USA bol dolapený a odsúdený [20].

NanoCore

Táto RAT rodina sa objavila v roku 2013 a bola vytvorená americkým vývojárom Taylorom Huddlestonom. Vývojár bol v roku 2018 dolapený a odsúdený na 3 roky vo väzení avšak jeho výtvor sa šíri ďalej [21]. NanoCore je písaný v programovacom jazyku Visual Basic .NET a používa na zabezpečenie obfuskátor a packer, zvyčajne Smart Assembly. Mimo tradičnej funkcionality ponúka NanoCore aj možnosť vydierania obete doplnkovým ransomware modulom, ktorý nebýva bežný pri tomto druhu malvéru.

Orcus

Rok 2014 patril výtvoru známému pod názvom Orcus, ktorý bol napísaný v programovacom jazyku C#. Tváril sa ako nevinný komerčný softvér na vzdialenú správu avšak bol známy najmä po hackerských fórach pre jeho funkcie ako keylogger, kradnutie hesiel a iné, ktoré sa v legálnych softvéroch tohto typu nenachádzajú. V apríli 2019 bol jeho hlavný vývojár John Rezvesz z Toronta v Kanade obvinený z tvorby a komerčnej distribúcie tohto škodlivého malvéru [21].

Akid

V roku 2015 sa objavila malvérová rodina Akid napísaná v jazyku Visual Basic .NET. Tvorcovia boli s najväčšou pravdepodobnosťou arabi, keďže sa tento malvér ocitol na stránkach s arabským popisom. Okrem bežnej funkctionality obsahuje aj možnosť DOS (Denial of Serivce), to znamená, že útočník môže obeti odoprieť prístup k službe. Ďalšou špeciálnou funkciou bolo získanie CD kľúčov od niektorých známych hier.

Babylon

Babylon RAT nám ukazuje, že vývojari malvéru ovládajú viac ako jeden programovací jazyk keďže jeho klient (kontrolné centrum) je písané v jazyku C# zatiaľ čo server v C++. Babylon bol vytvorený v roku 2015 a aj v dnešnej dobe sa aktívne šíri v podobe rôznych mutácii a bol aktívne predávaný na rôznych hackerských fórach.

Warzone

Warzone taktiež zvaný aj Ave Maria kvôli reťazcu ktorý sa nachádza v jeho binánych súboroch je jeden z najnovších a bol vytvorený koncom roka 2018 v jazyku C++. Stále je aktívne predávaný a jeho cena sa pohybuje v rozmedzí 17-23$ [22].

4 Analýza a návrh detekčných vzorov

V prvej časti tejto kapitoly si povieme akým spôsobom sa analyzuje činnosť vykonaná RAT malvérom na zariadení obete a následne sa zameráme už na samotnú tvorbu detekčných vzorov slúžiacich na detekciu týchto nežiadúcich malvérov.

4.1 Spôsob analýzy

Analýza RAT malvéru prebieha rovnako ako každá iná analýza škodlivého kódu. V prvom rade sa pozrieme na všeobecné informácie o našom súbore pomocou nástroja PEview alebo open-source programu RetDec.[1] Už po tomto kroku si môžme veľakrát všimnúť viaceré podozrivé vlastnosti, ako napríklad časové známky, mena sekcií, veľkosti sekcií, importy a iné dôležité informácie. Následne zistíme v akom programovaciom jazyku bola naša vzorka napísaná a či je nejakým spôsobom zabezpečená packerom alebo obfuskátorom. Z našich štatistík vyplýva, že najčastejšie programovacie jazyky na tvorbu RAT sú: .NET, ktorý zastupuje jazyky Visual Basic a C#, ďalej je tu Delphi, C++ a Java. Dôvodom prečo sú práve .NET a Delphi najviac používané je zrejmý a je ním jednoduchosť samotného jazyka ako aj dostupnosť kompiléru - prekladača (zadarmo alebo v prípade komerčných aj nelegálne získané verzie). Tým pádom sú schopní aj nie moc skúsení programátori napísať RAT bez väčších ťažkostí. Veľa týchto RATov zdieľa značné časti kódu a z toho tiež vyplýva, že používajú veľakrát rovnaké moduly, ako napríklad keylogger modul alebo modul na získavanie uložených hesiel z prehliadačov. Na zistenie týchto informácií, ako sú použitý programovací jazyk a prípadný obfuskátor alebo packer sme najčastejšie používali program DIE (všetky použité programy sú popísané v kapitole 2). Podľa zisteného programovacieho jazyka a prítomných techník zabezpečenia sme vždy zvolili následný postup. Podľa použitých packerov a obfuskátorov sme vedeli, ktorý nástroj na ich odstránenie použiť.

Pri .NET jazykoch sme si zvyčajne vystačili s unpackerom De4dot, ktorý nám pomohol odstrániť obfuskáciu a známe packery, ako napríklad .NET Reactor. Po tom, čo sme mali očistenú našú vzorku od týchto protekčných techník bol použitý dekompilér DnSpy, ktorý nám odhalil vnútornú štruktúru a samotnú funkcionalitu na základe ktorej sme boli schopní začať písať detekčné pravidlá. Veľa iných jazykov na rozdiel od .NETu nie je možné dekompilovať do čitateľnej podoby a zisťovanie ich funkcie pomocou dissasemblovania by bolo veľmi komplikované, zdĺhavé a neefektívne. Preto sme na zistenie ich funkcionality použili práve nástroje ako Process Monitor a Process Explorer, vďaka ktorým sme boli schopní vidieť správanie

[1] https://github.com/avast/retdec

procesov, ich zápisy do registrov, sieťovú komunikáciu, vytváranie a zápisy do súborov, tvorbu mutexov, semafórov ako aj iných synchronizačných prvkov. Aj keď je tento spôsob podstatne rýchlejší ako čítanie disassemblovaného kódu, tak stále nie je najefektívnejší, lebo napríklad v Process Monitore sa nám zobrazuje počas behu programu aj niekoľko tisíc zápisov do registrov, súborov a to len preto, lebo veľa z ních vykonáva samotný systém. Kvôli neefektivite a značnému množstvu nepodstatných informácií v Process Monitore sme sa rozhodli použiť na analýzu sandbox Cuckoo. Analýza každej našej vzorky zvyčajne trvá aj niekoľko minút, ale rozhodne výsledné informácie stoja za to čakanie. Cuckoo poskytuje informácie o type súboru, jeho štruktúre, potencionálne nebezpečných alebo podozrivých činnostiach cez sieťovú kominukáciu, použité registre, mutexy, importy, súbory a mnoho ďalších. Tento sandbox nám dokáže vyfiltrovať veľa nepodstatných informácií, ale aj napriek tomu sa tam nájdu pre nás nepodstatné dáta. V porovnaní s nadbytočným počtom informácií v Process Monitore to je ale len zanedbateľná časť. Po naštudovaní výpisu zo sandboxu môže začať samotné písanie pravidiel na základe statických alebo behaviorálnych vlastností. Samozrejme v rámci tejto práce je nevyhnutné získať o danej malvérovej vzorke aj iné informácie okrem tých, ktoré sú nevyhnutné na ich detekovanie a sú nimi práve funkcie malvéru. Keďže RATy disponujú veľkým množstvom funkcií, ktoré sú uverejnené na propagačnej webovej stránke daného malvéru. V prípade, že malvér takouto stránkou nedisponuje je nutnosť nahliadnuť do samotného programu. Pri zistení funkcionality z malvéru je potreba si stiahnuť klientskú časť tzv. builder a následne na tom istom zariadení zapnúť serverovú časť, ktorá sa pripojí ku klientovi (časť RATu určená pre útočníka). Po úspešnom nadviazaní komunikácie sa v klientovi zobrazí pripojená obeť a povolia sa nám možnosti na manipuláciu s jej zariadením. Tieto možnosti bývajú rôzne, ale tie najčastejšie z nich sú: vzdialená správa, keylogger, odchytávanie uložených hesiel, používanie webkamery, manipuláciu so súbormi a ich sťahovanie, vzdialený príkazový riadok a samozrejme manipulácia procesov.

4.2 Tvorba detekčných pravidiel

Ako už bolo spomínané, detekčné pravidlá budú tvorené v jazyku YARA. Jednotivé vytvorené pravidlá môžu mať rôznu formu. Z toho dôvodu má viacero firiem zaužívané rôzne zvyklosti na základe ktorých tieto pravidlá píšu. Ich dodržiavanie je nevyhnutné v rámci firmy, nakoľko si tým pracovníci uľahčujú navzájom prácu (pri úprave cudzieho pravidla platí rovnaká štruktúra a tým pádom je zrejmé kde čo patrí).

Každé naše pravidlo preto pozostáva z troch hlavných častí, ktorými sú: známe pomenované objekty, známe správanie a známe sekvencie.

1. Do prvej skupiny tzv. pomenovaných objektov zaraďujeme synchronizačné objekty, ktorými sú napríklad: mutexy, akcie, semafory, atómy a sekcie. Zvyčajne na základe týchto objektov vie malvér či bol už na danom počítači spustený a v prípade, že áno, aby sa nespúšťal opäť.
2. V ďalšej skupine tzv. behaviorálnych prvkov radíme akcie na základe ktorých vieme detekovať malvér ako napr. manipulácia s registrami, súbormi, sieťová komunikácia, spustené služby, príkazy alebo použité Windows API.
3. Tretia skupina obsahujúca známe sekvencie pozostáva zo statických prvkov, ktorými sú: textové reťazce, typ súboru, verzia jazyka, použitý packer, počet sekcií, názvy sekcií, počet a názvy prostriedkov, hash importov, časové známky a mnoho ďalších.

Každá z týchto skupín obsahuje ešte aj vnútornú štruktúru, ktorá zahŕňa metadáta, v niektorých prípadoch textové reťazce a samozrejme najdôležitejšiu vec a tou je podmienka tvoriaca hlavnú funkcionalitu. Do metadát sa zvyčajne píšu údaje ako meno autora, popis daného pravidla, názov rodiny malvéru, typ pravidla a pre jednoduchšie dohľadanie aj hash našej vzorky. Zvačajne len statické pravidla zahŕňajú do svojej štruktúry aj textové reťazce, ale taktiež je možné aj pri inom druhu pravidla posilniť jeho behaviorálnu detekciu aj reťazcom ktorý sa bude nachádzať v danej vzorke. Posilňovanie pravidiel sa vykonáva, keď behaviorálne správanie nie je dostatočne jedinečné a mohlo by pri detekcii viesť ku množstvu falošne pozitívnych zhôd. Posledná časť vnútornej štruktúry, ktorá tvorí jadro celého pravidla, je podmienka, na základe ktorej sa rozhodne či vzorka na ktorú bolo pustené naše pravidlo nástrojom YARA bude detekovaná. V podmienke je použité logické vetvenie v podobe `if` podmienok a prípadne aj cyklov.

4.3 Ukážka pravidiel

Následuje reálna ukážka nami vytvorených pravidiel na detekciu niektorých rodín RAT malvéru. Popíšeme si pri tom, čo jednotlivé časti znamenajú a ako sme ich našli v našej vzorke.

B0–K

```
import "cuckoo"

rule b0k_known_behavior_high
{
  meta:
    author = "Samuel Sidor"
    description = "detection based on known behavior"
    reliability = "test"
    strain = "B0-K"
```

```
    type = "rat"
    severity = "malware"
    rule_type = "behavioral"
    hash = "67c6bffca8009861be00b20f5d59378b"
    version = "0.1.3"
  condition:
    cuckoo.filesystem.file_write(/\\AppData\\Local\\Temp\\melt\.txt$/i) and
    cuckoo.filesystem.file_write(/\\AppData\\Roaming\\Microsoft\\
    svchost\.exe$/i) and
    cuckoo.registry.key_read(/HKEY_CURRENT_USER\\Software\\Microsoft\\
    Windows\\CurrentVersion\\Run\\azoz$/i)
}

rule b0k_known_sequences
{
  meta:
    author = "Samuel Sidor"
    description = "detection based on known sequences"
    reliability = "test"
    strain = "B0-K"
    type = "rat"
    severity = "malware"
    rule_type = "static"
    hash = "67c6bffca8009861be00b20f5d59378b"
    version = "0.1.3"
  strings:
    $s01 = "melt.txt" wide
    $s02 = "AhnLab-V3" wide
    $s03 = "F-Prot" wide
    $s04 = "F-Secure" wide
    $s05 = "Jiangmin" wide
    $s06 = "sbamtray" wide
    $s07 = "Microsoft\\svchost.exe" wide
    $s08 = "software\\microsoft\\windows\\currentversion\\run" wide
    $s09 = "FileManager||Error" wide
    $s10 = "info||myID||" wide
    $s11 = "8c0a0be4-c2d9-43fd-8362-d331a08ed069" ascii
    $s12 = "C:\\Users\\XaY_Tn" ascii
  condition:
    MSIL and
    not b0k_known_behavior_high and
    11 of ($s*)
}
```

Detekčné pravidlo 4.1: Ukážka pre B0–K RAT

Detekcia malvéru B0–K je tvorená dvomi pravidlami: `b0k_known_behavior_high` a `b0k_known_sequences`. Najprv je dôležité spomenúť, že YARA moduly a funkcie, ktoré berú reťazec znakov na získanie zhody, používajú regulárne výrazy z dôvodu efektívneho vyhľadávania a možnosti prispôsobovania sa istým obmenám v týchto reťazcoch. Obe pravidlá pre túto rodinu obsahujú sekciu `meta` v ktorej sú metadáta ako: meno autora, popis pravidla, spoľahlivosť daného pravidla (či je overené natoľko, že nedáva falošne pozitívne zhody), názov rodiny malvéru, typ malvéru, závažnosť (či sa jedná o malvér, potencionálne nechcený program, alebo iný nástroj), typ pravidla (behaviorálne alebo statické), hash vzorky (v tomto prípade MD5) a nakoniec verzia danej vzorky, keďže aj tvorcovia malvéru neustále svoje projekty vyvíjajú. Na-

koľko je časť `meta` vo väčšine pravidiel takmer zhodná, nebudeme sa ňou už zaoberať a z dôvodu kompaktnosti nebude už v ďalších pravidlách uvádzaná. Podmienka v behaviorálnom pravidle `b0k_known_behavior_high` je tvorená prevažne 2 behaviorálnymi prvkami ktorými sú: zápisy do súboru a čítanie z registra. Konkrétne sa jedná o zápisy do súborov \AppData\Local\Temp\melt.txt a \AppData\Roaming \Microsoft\svchost.exe, nakoľko sú tieto zápisy dostatočne jedinečné. Ďalej tu máme čítanie z registru HKEY_CURRENT_USER\Software\Microsoft\Windows\Current Version\Run\azoz. Táto hodnota v registri je zodpovedná za spustenie programu pomenovaného „azoz“ po každom prihlásení používateľa do Windows konta. Medzi týmito časťami podmienky je logický operand `and`, ktorý zabezpečuje, že bude vzorka pozitívna na toto pravidlo len v prípade, že budú všetky tri časti vyhodnotené ako `true`. Pravidlo `b0k_known_sequences` obsahuje o jednu sekciu viac a tou je práve sekcia `strings`. Nachádzajú sa v nej reťazce obsiahnuté v našej vzorke. Máme dva hlavné typy reťazcov, ktorými sú: `wide` - predstavujúci unicode reťazec, ktorý na jeden znak používa dva bajty a ascii ktorý využíva jeden bajt na znak. V prípade, že je v malvérovej vzorke použitá obfuskácia alebo packer, je získanie týchto reťazcov problematické. Samotná podmienka obsahuje privátne pravidlo MSIL, ktoré slúži na detekovanie toho, že vzorka je spustiteľný súbor a je písaná v jendom z .NET jazykov. Ďalej tu je negácia pravidla `b0k_known_behavior_high`. Dôvod využitia tejto negácie je taký, že pokiaľ už bola vzorka detekovaná behaviorálnym pravidlom a bola vyhodnotená ako pozitívna, tak už nie je potreba ďalšej statickej detekcie vrámci celého pravidla 4.1. Posledná časť pravidla nám hovorí, že sa v binárnom súbore musí nachádzať aspoň jedenásť reťazcov zo sekcie strings začínajucích na písmeno „s“.

Akid

```
import "cuckoo"

rule akid_known_behavior_high
{
  condition:
    cuckoo.network.http_get(/http:\/\/whatismyip\.com\/automation\/
    n09230945\.asp$/) and
    cuckoo.process.resolved_api(/user32\.dll!SetWindowsHookExA$/) and
    cuckoo.sync.mutex(/(^|\\)([0-9]{1,76}|True)$/)
}

rule akid_known_sequences
{
  strings:
    $s01 = "$523e2cdb-4a0a-46e7-8ba1-e2037bb534de" ascii
    $s02 = "Advanced PDF Password Recovery" wide
    $s03 = "Password of Victim" wide
    $s04 = "Starting Log of Victim:" wide
    $s05 = "=== Passwords ===" wide
```

```
    $s06 = "windowsupdatekb17.exe" wide
    $s07 = "Set CDAudio Door Open" wide
    $s08 = "SWAPMOUSE" wide
    $s09 = "(?<host>\\S+) (?<port>\\d+) (?<threads>\\d+) (?<sockets>\\d+)"
    wide
    $s10 = "HKEY_LOCAL_MACHINE\\SOFTWARE\\Ubisoft\\Splinter Cell Pandora
    Tomorrow" wide
    $s11 = "HKEY_LOCAL_MACHINE\\SOFTWARE\\Activision\\
    Call of Duty" wide

  condition:
    MSIL and
    not akid_known_behavior_high and (
      9 of ($s*)
    )
}
```

Detekčné pravidlo 4.2: Ukážka pre Akid RAT

Detekcia malvérovej rodiny Akid nachádzajúca sa v detekčnom pravidle 4.2 je opäť tvorená len dvomi časťami, pretože nie vždy je možné nájsť jedinečné pomenované objekty, keďže si skúsení tvorcovia malvéru dávajú na tieto veci pozor. Prvé pravidlo popisujúce známe správanie obsahuje behaviorálnu metódu na `http_get` dotaz z modulu `cuckoo`. Z toho vyplýva, že sa vzorka snažila získať odpoveď zo stránky „whatismyip.com“, aby zistila IP adresu obete. Druhou podmienkou zhody je použitie Windows API SetWindowsHookExA, ktoré je používaná na keylogger modul a poslednou časťou je štruktúra mutexu. Tento typ mutexu nie je dostatočne jedinečný na to, aby bol samostatne v pravidle `named_objects` a preto sa používa na posílenie detekcie v behaviorálnom pravidle. V druhom pravidle popisujúcom známe sekvencie sa zameriavame najmä na reťazce obsiahnuté v našej malvérovej rodine. Hneď prvým reťazcom je tzv. projekt Guid začínajúci znakom dolára, ktorý je jedinečný a kvôli tomu slúži ako dobré detekčné vodítko. Ostatné použité reťazce sú taktiež dostatočne jedinečné a ich kombináciou vytvoríme relatívne presný detekčný vzor. Podmienka pozostáva v prvom kroku z detekovania či je naša vzorka napísaná v niektorom z .NET programovacích jazykov. V druhej časti nás zaujíma či nebolo splnené prvé pravidlo (v prípade, že by bolo splnené by sa táto detekcia ukončila z dôvodu šetrenia zdrojov a veľkého množstva podvojních detekcií). Poslednú časť tvorí skupina minimálne deväť z jedenástich reťazcov zo skupiny `strings`.

Apocalypse

```
import "cuckoo"
import "pe"

rule apocalypse_known_named_objects
{
  condition:
    cuckoo.sync.mutex(/(^|\\).*Thread$/) and
    cuckoo.sync.mutex(/(^|\\).*Stop$/) and
```

```
    (
      cuckoo.process.executed_command(/C:\\Program Files\\Internet Explorer
      \\iexplore\.exe -bs$/i) or
      cuckoo.process.executed_command(/C:\\Windows\\apocalyps32\.exe -bs$/i)
      or
      cuckoo.process.executed_command(/-bs$/i)
    )
}

rule apocalypse_known_behavior_high
{
  condition:
    not apocalypse_known_named_objects and
    (
      cuckoo.filesystem.file_access(/C:\\Program Files\\Internet Explorer\\
      iexplore\.exe$/i) and
      cuckoo.filesystem.file_write(/\\AppData\\Local\\Temp\\.*\.htm$/) and
      cuckoo.registry.key_read(/HKEY_CURRENT_USER\\Software\\Microsoft\\
      Windows\\CurrentVersion\\Explorer\\FileExts\\.htm\\UserChoice\\
      Progid$/i) and
      cuckoo.registry.key_read(/HKEY_LOCAL_MACHINE\\SOFTWARE\\Microsoft
      \\Windows NT\\CurrentVersion\\Drivers32\\msvideo$/i) and
      cuckoo.registry.key_write(/HKEY_CURRENT_USER\\Software\\.*\\Metin$/i)
      and
      cuckoo.registry.key_write(/HKEY_CURRENT_USER\\Software\\.*\\
      MsgGoster$/i) and
      cuckoo.registry.key_write(/HKEY_CURRENT_USER\\Software\\.*\\
      YuklenenDizin$/i)
    )
}

rule apocalypse_known_sequences
{
  strings:
    $s01 = "Portions Copyright (c) 1999,2003 Avenger by NhT" ascii
    $s02 = "htm.htm" ascii
    $s03 = "ap0calypse" ascii
    $s04 = "IsNTAdmin" ascii
    $s05 = "MsgGoster" ascii
    $s06 = "Metin" ascii
    $s07 = "YuklenenDizin" ascii
  condition:
    EXE and
    pe.is_32bit() and
    not apocalypse_known_named_objects and
    not apocalypse_known_behavior_high and
    (
      pe.timestamp == 866766137 and
      pe.number_of_resources > 0 and
      (
        pe.resources[0].name_string == "A\x00Y\x00A\x00R\x00" or
        pe.resources[pe.number_of_resources-1].name_string == "P\x00A\x00C
        \x00K\x00A\x00G\x00E\x00I\x00N\x00F\x00O\x00"
      ) and
      (
        UPX or
        5 of ($s*) //unpacked strings
      )
    )
}
```

Detekčné pravidlo 4.3: Ukážka pre Apocalypse RAT

Pri malvérovej rodine Apocalypse sme vytvorili všetky tri druhy pravidiel ako je možné vidieť v ukážke 4.3. V prvej skupine pomenovaných objektov používame dva formáty mutexov vyskytujúcich sa v našej vzorke. V tejto rodine sa pri týchto mutexoch pred slovami Thread a Stop vždy nachádza jedinečná kombinácia znakov a čístic preto, sme tam použili možnosť zhody pri akýchkoľvek znakoch pred týmito slovami tzv. divokú kartu (wild card). Toto pravidlo je posilnené detekciou jedneho z troch variant spúšťania príkazu „-bs“, ktorý je v tejto rodine vždy prítomný. Následne sa v behaviorálnom pravidle zameriavame na pístup k súboru iexplore.exe, ktorý nie je moc bežný. Ďalej si môžte všimnúť, že sa snažíme zistiť, či bol prečítaný súbor s príponou .htm pod súborovou hierarchiou priečinka Temp. Potom sa dostávame už len k čítaniu a zapisovaniu v registroch. Čítanie z registru msvideo nám hovorí, že bude použitá webkamera. Následné zápisy do registrových kľúčov Metin, MsgGoster a YuklenenDizin nám zabezpečujú dostatočnú jedinečnosť pri detekcií tejto aktuálnej malvérovej rodiny a vyvarovaniu sa detekcii iných falošne pozitívnych výsledkov. Pri pravidle pozostávajúcom zo známych sekvencií máme tentokrát menej jedinečných reťazcov a aj tieto reťazce sú známe až po deobfuskácii, preto v podmienke použijeme aj iné statické spôsoby detekcie. Podmieka nám začína zistením, či sa jedná o exe formát a následne, či je práve skenovaná vzorka 32 bitová. Ďalej vylúčime zhody z predchádzajúcich pravidiel. Následne porovnáme časovú známku našej vzorky s časovou známkou 866766137, ktorá je vždy prítomná v tejto rodine. Počet zdrojov (resources) je taktiež dôležitý ukazovateľ a veľakrát nám pomôžu aj ich názvy. Rodina Apocalypse používa v zdrojoch názvy ako AYAR alebo PACKAGEINFO. Poslednou časťou pravidla je zistenie či je daná vzorka chránená packerom UPX[2] a v prípade, že nie je, hľadáme zhodu aspoň päť reťazcov zo sekcie strings. UPX je privátne pravidlo kontrolujúce názvy sekcií packnutého EXE súboru. Po týchto všetkých podmienkach je detekcia dokončená a ak pravidlo bolo vyhodnotené ako true (pravda), tak skenovaná vzorka patrí do rodiny Apocalypse.

Babylon

```
import "cuckoo"

rule babylon_known_named_objects
{
  condition:
    cuckoo.sync.mutex(/(^|\\)\w{8}-\w{4}-\w{4}-\w{4}-\w{12}$/) and
    cuckoo.signature.name(/^infostealer_keylog$/) and
    cuckoo.signature.name(/^process_interest$/) and
    cuckoo.signature.name(/^stealth_file$/) and
    cuckoo.filesystem.file_access(/Roaming\\ConfigsEx$/i) and
    cuckoo.filesystem.file_access(/\\ConfigsEx\\\d{4} \d{2} \d{2} -
    \d{2} \d{2} [AP]M$/i)
```

[2]https://upx.github.io/

```
}

rule babylon_unpacked_known_sequences
{
  strings:
    $pdb = "C:\\Users\\Stefan\\documents\\visual studio 2013\\Projects\\
    sqliteProject\\Release\\sqliteProject.pdb"
    $s01 = "bss_server.usrRelay"
    $s02 = "SELECT origin_url, username_value, password_value FROM logins"
    $s03 = "SELECT encryptedUsername, encryptedPassword, formSubmitURL FROM
    moz_logins"
    $s04 = "ClipBoard.txt" wide
    $s05 = "[PRINT]" wide
    $s06 = "DoS Already Active..." wide
    $s07 = "Babylon RAT Client" wide
    $s08 = "A Babylon RAT client is currently running on this PC." wide
    $s09 = "Close this window to end the client." wide
    $s10 = "SetWindowsHookExW"
  condition:
    EXE and
    not babylon_known_named_objects and
    (
      (
        $pdb and
        $s10
      ) or
      7 of ($s0*)
    )
}
```

Detekčné pravidlo 4.4: Ukážka pre Babylon RAT

Pri rodine Babylon sa nachádzajú predovšetkým všeobecné behaviorálne prvky a z toho dôvodu použijeme najmä menné objekty, ktoré sú posilené o 2 behaviorálne prvky a taktiež jedno pravidlo so statickými sekvenciami. Jednotlivé časti pravidla `babylon_known_named_objects` sú podmienky spojené logickým prvkom `and`, to znamená, že musia byť všetky splnené pre vrátenie hodnoty `true`. V prvom pravidle na začiatku menných objektov kontrolujeme tvar mutexu, ktorý musí spĺňať presnú štruktúru: 8 znakov, pomlčka, 3krát po 4 znaky s pomlčkami a nakoniec ďalších 12 znakov. Ďalej musí byť splnená signatura hľadajúca prvky keyloggeru, keďže táto vzorka obsahuje túto vlastnosť. Signatura `process_interest` nám hovorí, že sa malvér zaujímal o niektorý proces, `stealth_file` zase, že sa malvér pokúsil skryť nejaký súbor. Nakoniec tu máme 2 behaviorálne prvky prístupu k súboru v ceste ConfigsEx. Pri druhom z nich je ešte upresnený tvar, ktorý predstavuje dátum a čas vo formáte rok, mesiac, deň, pomlčka, hodiny, minúty a nakoniec „AM“ alebo „PM“. Keďže je malvér Babylon vždy chránený packerom, zvyčajne UPX, ktorý zťažuje statickú analýzu, vzorka musela byť najprv zbavená tejto ochrany a preto je pomenované toto statické pravidlo `unpacked`. V podmienke najprv kontrolujeme či sa jedná o EXE súbor pomocou privátneho pravidla kontrolujúceho `MZ` sekvenciu v spustiteľnom súbore a taktiež či nebola splnená predchádzajúca podmienka v iných častiach v rámci pravidla 4.4. Ďalej rozdeľujeme podmienku pomocou logického

operátoru `or` na 2 podčasti z ktorých musí byť minimálne jedna splnená. V prvej časti kontrolujeme, či sa nachádza v danej vzorke debug pdb cesta a zároveň je prítomné Windows API „SetWindowsHookExW“, ktoré slúži na hookovanie predovšetkým stlačených kláves. Druhá časť pozostáva z vyhľadania minimálne 7 z 10 „s“ reťazcov v danej vzorke.

LuminosityLink

```
import "cuckoo"

rule luminositylink_known_behavior_high
{
  condition:
    cuckoo.signature.name(/^rat_luminosity$/) and
    cuckoo.signature.name(/^antisandbox_productid$/) and
    cuckoo.signature.name(/^antisandbox_sleep$/) and
    cuckoo.signature.name(/^copies_self$/) and
    cuckoo.signature.name(/^crypto_api_activity$/) and
    cuckoo.signature.name(/^infostealer_keylog$/)
}

rule luminositylink_known_sequences
{
  strings:
    $guid = "$3a574e5b-bd2e-4e23-a175-583f6c78f50f"
    $s01 = "SetWindowsHookEx"
    $s02 = "finished successfully. Attacks Sent: " wide
    $s03 = "Content-length: 5235" wide
    $s04 = "M1N3R" wide
    $s05 = "ST4T10N" wide
    $s06 = "Qu1Ck" wide
    $s07 = "DESK*" wide
    $s08 = "*Started*BYT3S*" wide
    $s09 = "LuminosityLink is Running" wide
    $s10 = "=P4CK3T=" wide
    $s11 = "PROXY*" wide
    $s12 = "Cannot Get Saved Keylogs:" wide
    $s13 = "LuminosityCryptoMiner" wide
    $s14 = "KEYLOGS*" wide
    $s15 = "FUCKUP" wide
    $s20 = "http://cachefly.cachefly.net/5mb.test" wide
    $s21 = "F4Z3Z1LLA" wide
    $s22 = "CHR0M3" wide
    $s23 = "client does not use Luminosity's startup!" wide
    $s24 = "REMOVEGUARD" wide
  condition:
    EXE and
    not luminositylink_known_behavior_high and (
      (
        $guid and
        $s01
      ) or
      (
        10 of ($s*)
      )
    )
}
```

Detekčné pravidlo 4.5: Ukážka pre LuminosityLink

Pravidlo `luminositylink_known_named_objects` tvorí kombinácia signatur spojených logickým operátorom `and`. Najpodstatnejšia z nich je práve signatura `rat_luminosity`, ktorú tak ako väčšinu ostatných napísala komunita cuckoo sandboxu. Táto signatura je sama o sebe schopná detekovať či sa jedná o LuminosityLink na základe istých behaviorálnych prvkov. Ostatné signatury sú tam ako doplnok na podporu detekcie a minimalizovaniu možných falošne-pozitívnych zhôd. Už z ich názvu vyplýva čo detekujú. Kontrola, či si vzorka získa productid sandboxu a či je nastavený sleep na oneskorenie alebo oklamanie sandbox detekcie. Ďalej signatura, ktorá určuje či sa vzorka niekde nakopírovala, či sa použilo Windows cryptoapi a nakoniec či má funkciu keyloggeru. Čo sa týka `luminositylink_known_sequences` tak opäť je prvá časť rovnaká ako obvykle, kontrola formátu EXE a kontrola, či nebola splnená predchádzajúca podmienka. Ďalej cez `or` rozdeľujeme podmienku do 2 logických celkov. V prvom hľadáme MSIL GUID, ktoré je jedinečé pre každý projekt v .NET jazyku a podporujeme ho vyhľadaním Windows API „SetWindowsHookEx“. V druhej časti hľadáme zhodu aspoň 10 z 24 reťazcov „s“. Stačí aby jedna z týchto dvoch častí bola splnená a pravidlo bude vyhodnotené ako `true`.

Warzone

```
import "cuckoo"
import "pe"

rule warzone_known_named_objects
{
  condition:
    cuckoo.sync.event(/(^|\\)sqlite3DEvent$/) and
    cuckoo.sync.event(/(^|\\)rdpWrap32$/) and
    cuckoo.sync.event(/(^|\\)rdpWrap64$/) and
    cuckoo.sync.event(/(^|\\)rdpWrapIni$/) and
    cuckoo.sync.event(/(^|\\)vncEvent$/) and
    (
      cuckoo.process.resolved_api(/avicap32\.dll!capGetDriverDescriptionW$/)
      and
      cuckoo.registry.key_write(/\\SYSTEM\\CurrentControlSet\\Control\\
      MediaResources\\msvideo$/i) and
      cuckoo.registry.key_write(/\\Software\\Microsoft\\Window\\
      CurrentVersion\\Explorer\\\w+$/i) and
      cuckoo.registry.key_access(/\\Software\\Microsoft\\Windows\\
      CurrentVersion\\App Paths\\\w+\.exe$/i) and
      cuckoo.registry.key_access(/\\Software\\Microsoft\\Windows\\
      CurrentVersion\\Explorer\\\w+$/i)
    )
}

rule warzone_known_sequences
{
  strings:
```

```
    $s01 = "PK11_CheckUserPassword" ascii
    $s02 = "127.0.0.2" ascii
    $s03 = "SMTP Password" wide
    $s10 = "AVE_MARIA" ascii
    $s11 = "svchost.exe -k" wide
    $s12 = "MortyCrypter\\MsgBox.exe" wide
    $s13 = "Hey I'm Admin" wide
    $s14 = "Bla2" ascii
  condition:
    EXE and
    not warzone_known_named_objects and
    (
      2 of ($s0*) and
      (
        2 of ($s1*) or
        (
          1 of ($s1*) and
          pe.resources[0].type_string == "W\x00M\x00_\x00D\x00S\x00P\x00"
        )
      )
    )
}
```

Detekčné pravidlo 4.6: Ukážka pre Warzone RAT

V časti `warzone_known_named_objects` pravidla 4.6 nie sú menné objekty dostatočne konkrétne na určenie toho, že sa jedná o Warzone RAT a z toho dôvodu sú podporené behaviorálnymi prvkami. Čo sa týka eventov tak ich použitie určuje, že daný malvér používa VNC (Virtual Network Computing). Ďalej kontrolujeme, či je použité Windows API, ktoré slúži na prácu s webkamerou. Behaviorálne prvky obsahujú kontrolu zapisovania a prístupu do registrov Windowsu na uvedené lokácie. Vonkajšia časť pravidla na kontrolu známych sekvencií ostáva nezmenená. Prvé vnorenie kontroluje či sa vo vzorke nachádzajú aspoň 2 z „s0“ reťazcov a zároveň musí byť splnená jedna z 2 nasledujúcich podmienok. Buď budú nájdené 2 z reťazcov „s1“ alebo len 1 reťazec z „s1“ a zárveň prvý resource bude pomenovaný „WM_DSP“, ktorý je typický práve pre tento malvér.

4.4 Návrh obrany proti RAT

V dnešnej dobe je problematické si dostatočne zabezpečiť svoje zariadenie. Aj keď to však nie je 100% možné, existujú spôsoby, ktorými sa dá možnosť nákazy malvérom radikálne znížiť.

Najlepšou obranou aj naďalej ostáva samotná prevencia. Je nevyhnutnosť dbať na aktualizácie operačného systému, ako aj iného softvéru inštalovaného na užívateľskom zariadení, kvôli možnostiam zneužitia zraniteľností útočníkmi v starších verziách. Odporúča sa nastaviť si automatické sťahovanie bezpečnostných aktualizácii. Ďalšou dôležitou vecou je mať zapnutý firewall, ktorý je dostatočne nastavený.

Jednou z najdôležitejších vecí je však mať na používanom zariadení zapnutý antivírus s automatickými aktualizáciami. Treba sa vyhýbať sťahovaniu súborov (nie len tých spustiteľných) z neoverených zdrojov. V prípade e-mailov, ktoré bývajú častným cieľom útočníkových aktivít, sa neodporúča klikať na podozrivé odkazy a prílohy od neznámych odosielateľov. A nakoniec si treba dávať pozor aj počas samotného prehliadania webu a neklikať na podozrivé odkazy alebo vyskakovacie okná. V prípade podozrenia, že je zariadenie napadnuté RATom alebo iným malvérom je žiadúce si to overiť. V prvom rade treba skontrolovať „Programy po spustení“ v ktorých sa nachádzajú všetky programy, ktoré sa spustia po tom, čo sa zapne počítač. Malvér sa zvyčajne snaží zapnúť hneď po štarte operačného systému, takže toto môže byť dobré prvé vodítko. Následne sa odporúča skontrolovať správcu úloh a v ňom spustené procesy, ako aj zoznam nainštalovaných programov v ovládacom paneli.

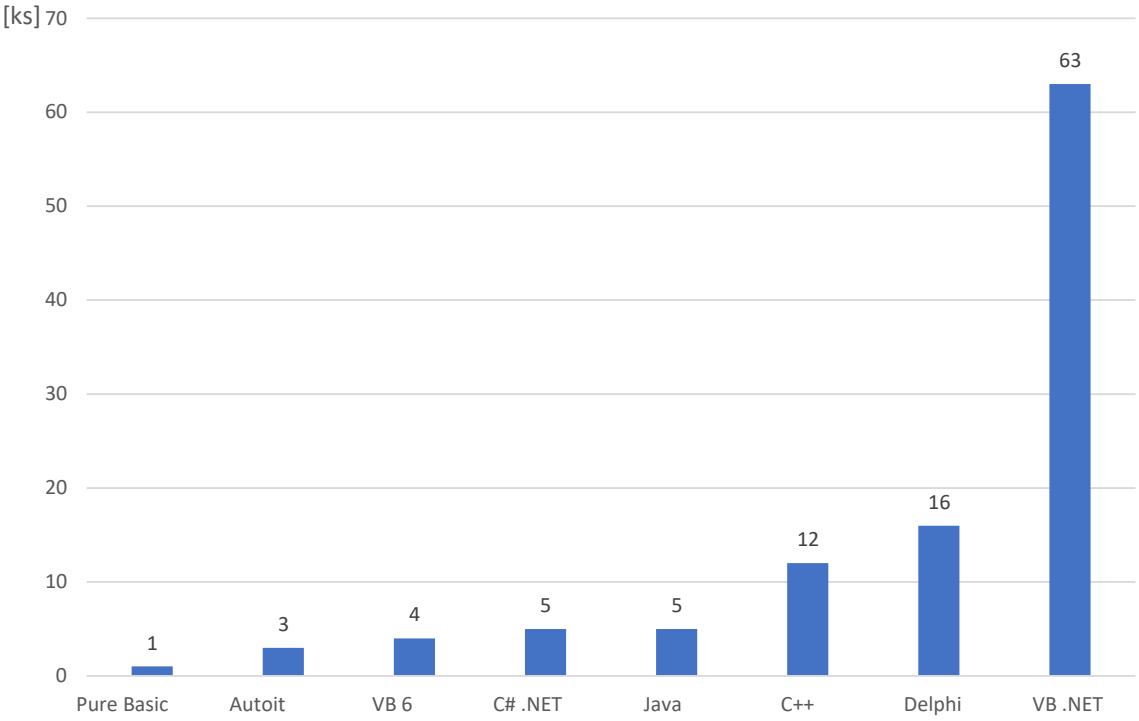
V prípade, že sa zariadenie nakazilo je nevyhnuté rýchlo konať. Dôvodom je, že RATy vo veľkom množstve prípadov zasielajú útočníkovi citlivé užívateľské dáta. Práve preto je najlepšou voľbou stiahnuť antivírus (AV) alebo niektorý zo známejších antimalvérov (AM). Pre zníženie množstva odoslaných súborov je možné počítač odpojiť od internetu akonáhle bola hrozba detekovaná, stiahnutie AV alebo AM cez iné zariadenie a prenesenie inštalačného súboru cez flash-disk do nakazeného zariadenia. Nakoľko nie každý AV a AM dokáže nájsť a odstrániť RAT, ktorý sa nachádza v napadnutom počítači, je dôležité vybrať jeden z populárnejších, alebo iný na základe recenzií. Po týchto krokoch nasleduje samotná inštalácia a zapnutie stiahnutého programu. Iným spôsobom odstránenia RAT je preinštalovanie alebo obnova systému. V provom rade je nutná záloha všetkých dôležitých dát na cloudové úložisko alebo externý disk. Nasleduje obnova výrobných nastavení systému, alebo vykonanie úplnej inštalácie systému. Po prečistení systému by mal byť ako prvý nainštalovaný AV a následne obnovené dáta zo záloh, ktoré AV preskenuje pre prípad, že aj tie boli nakazené. Po dokončení týchto krokov by sa v systéme už nemali nachádzať žiadne malvéry. Avšak je taktiež nutné zmeniť všetky užívateľské hesla osoby, ktorá dané zariadenie používala, lebo sa jej heslá mohli dostať k útočníkovi. Keďže sa mohlo jednať aj o heslá od online bankovníctva je nutnosť preveriť aj peňažné pohyby na účtoch a v prípade podozrivej aktivity kontaktovať banku a patričné orgány.

5 Výsledky výskumu a vyhodnotenie

Po dôkladnej analýze viac ako 58 rôzných RAT rodín, ktoré pozostávali z vyše 110 rôznych variant a verzií sme dospeli k viacerým záverom a štatistikám. Táto kapitola sa bude zaoberať štatistikami o najčastejšie používaných programovacích jazykoch, krajinách výskytu, využívaných obfuskátoroch a packeroch, časový horizont výskytu, ako aj funkcie analyzovaných malvérov.

5.1 Programovacie jazyky

V prvom rade sa zameráme na použité programovacie jazyky. V nami detekovaných rodinách s veľkým náskokom prevládal jazyk VB .NET, ktorý sa vyskytoval v 63 rôznych variantách analýzovaných rodín čo predstavuje 57% výskyt. Druhé a tretie miesto patrí jazykom Delphi a C++, ktoré oba mali zastúpenie viac ako 10%.

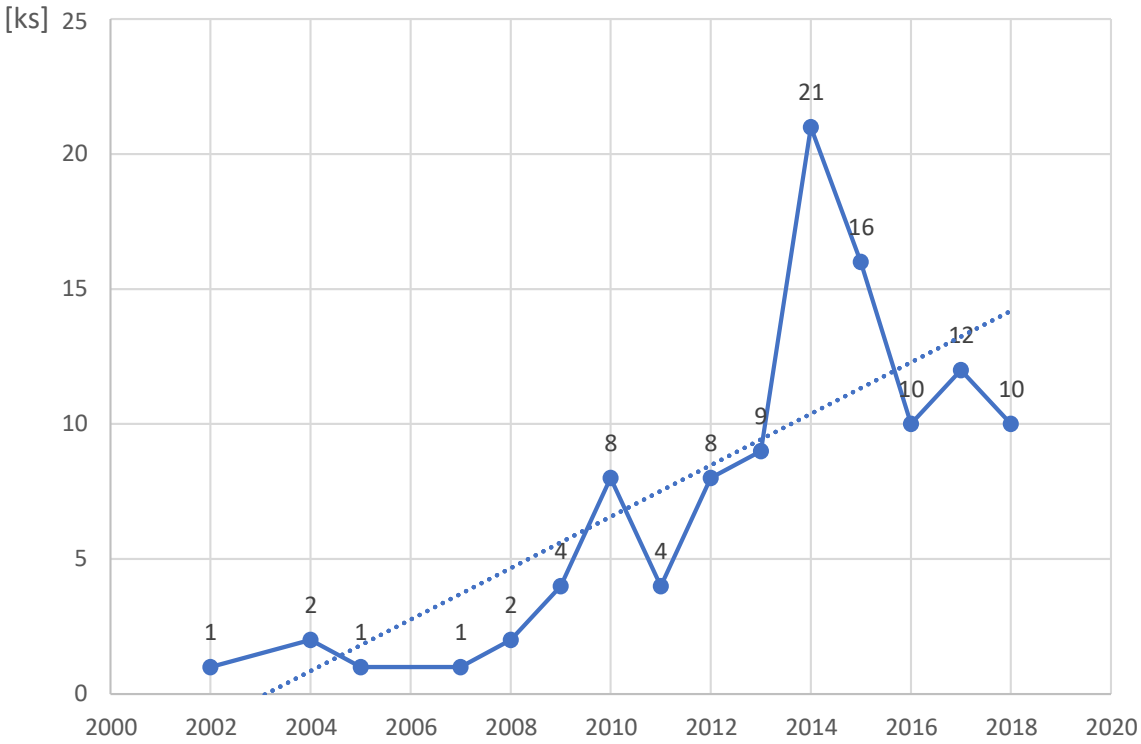


Obr. 5.1: Počet variant analyzovaných rodín zoskupených podľa jazyka

5.2 Časový vývoj počtu detekcií

Čo sa týka časového horizontu výskytu jednotlivých RAT malvérov, ktoré sme analyzovali tak práve rok 2014 bol prevratný a to z toho dôvodu, že sa dostal viac do povedomia zdrojový kód rodiny NjRAT, ktorý bol natoľko populárny, že vďaka

nemu vzniklo veľké množstvo mutácií. Taktiež si môžte všimnúť, že RAT malvér ma z historického hľadiska stúpajúci trend.

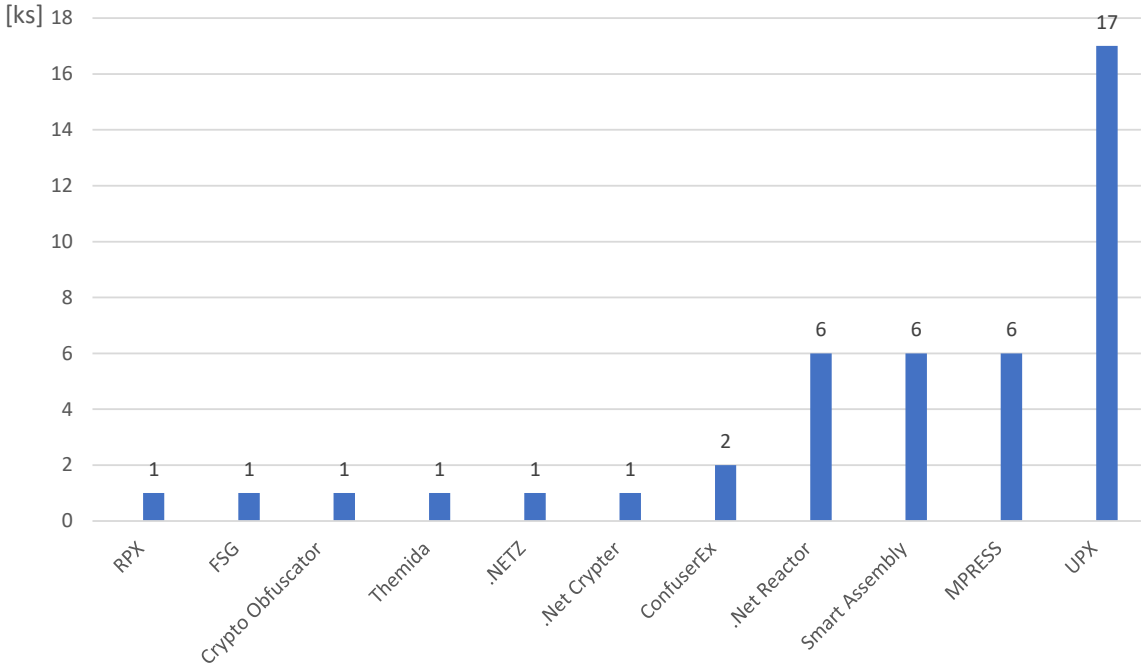


Obr. 5.2: Počet variant analyzovaných rodín zoskupených podľa roku vydania

5.3 Packery a obfuskátory

Z dôvodu ochrany zdrojového kódu, zníženiu detekcií a skomplikovaniu práce malvérovým analytikom používajú tvorcovia malvéru packery a obfuskátory. Väčšina .NET packerov priamo obsahuje aj obfuskáciu zdrojového kódu a nie je potreba využívať dodatočný softvér. Packery, ktoré sú uvedené v tejto kapitole sú všeobecne známe a sú na ne vytvorené detekčné vzory v rôznych analytických nástrojoch. Čo sa týka pokročilejšej protekcie tak si útočníci zakupujú packery na mieru, packery ktorých počet predajných kusov je obmedzený ako napr. Octopus Protector,[1] alebo používajú privátne packery. Zo všetkých analyzovaných RATov 41,9% rodín používalo packer a 38,1% rodín bolo obfuskovaných. Najčastejšie používaný obfuskátor bol práve známy UPX. Zastúpenie jednotlivých packerov môžte vidieť na grafe 5.3.

[1] https://breaking-security.net/octopus/



Obr. 5.3: Počet variant analyzovaných rodín zoskupených podľa využitého packeru

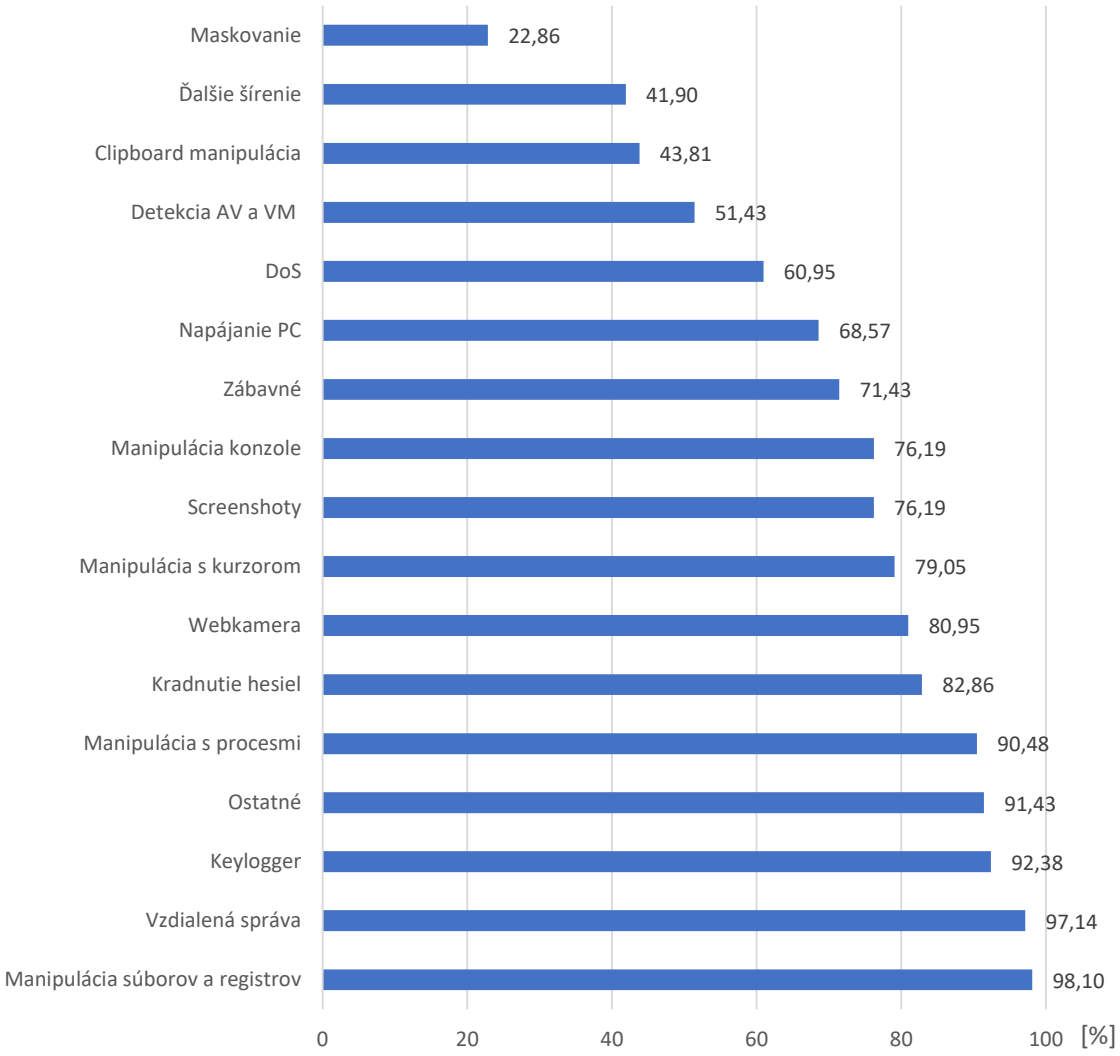
5.4 Funkcionalita

Škála výskytu jednotlivých funkcií RAT malvéru je veľmi široká a všetky funkcie budú popísané od najčastejších až po tie najzriedkavejšie. Keďže sa tento typ malvéru používa najmä pri špehovaní a kradnutí citlivých súborov je pochopiteľné, že práve manipulácia so súborovým systémom je práve tá najčastejšie sa vyskytujúca funkcia. Pri manipulácii so súbormi je myslené: čítanie, mazanie, prepisovanie a prípadne odosielanie súborov útočníkovi. Druhá najčastejšie obsahovaná funkcia je samotná vzdialená správa počítača s možnosťou ovládať kurzor myši a klávesnicu. Čo sa týka efektívneho špehovania táto funkcia nesmie chýbať. Z toho dôvodu jednotlivé RATy, ktoré neobsahovali túto funkciu neboli natoľko populárne. Ďalej sa jedná o keylogger funkcionalitu vďaka ktorej útočník zaznamenáva všetky stlačené klávesy. Keylogger sám o sebe môže byť aj samostatný program a práve tu sa nám potvrdzuje, že RAT obsahuje veľké množstvo funkcií iných typov malvéru. Kategória „Ostatné“ zahňa veľké množstvo špecifických fukcií, ktoré sa nevyskytovali tak často, aby boli obsiahnuté v hlavných stĺpcoch tabuľky,[2] ale taktiež nemohli byť ignorované. Jednalo sa napríklad o ransomware modul, coin-miner modul, chat s útočníkom, obnovu systému, povolenie alebo zakázanie zobrazovania istých častí OS (napr. regedit alebo Správca úloh), vypnutie svetla zapnutej kamery, torrent

[2]Tabuľka podrobnej detailnej analýzy sa nachádza v prílohe kvôli jej rozsiahlemu obsahu.

seeder alebo napríklad aj samotný antispyware, ktorý zabezpečoval, aby počítač nenapadol iný malvér len daný RAT. Ďalšou častou funkciou bola manipulácia s procesmi čo zahŕňalo ukončovanie, pozastavovanie alebo reštartovanie konkrétneho procesu. Kradnutie hesiel je taktiež značne populárne a keď užívateľské heslá útočník získa, vie nimi spôsobiť veľké nepríjemnosti. Najmä čo sa jedná hesiel od e-mailu, banky alebo sociálnych sietí. Z toho dôvodu sa odporúča dvojfaktorová autentizácia (2FA). V RAToch bolo najčastejšie použité kradnutie uložených hesiel zo všetkých známych prehliadačov, Outlooku, FTP klienta a taktiež herných klientov. K špehovaniu neodlúčiteľne patrí aj snímanie toho čo človek za svojim počítačom robí a z toho dôvodu mal každý lepší RAT aj funkciu na snímanie obrazu alebo zvuku z webkamery. Funkcie na manipulácia s kurzorom a robenie screenshotov sú úzko spojené so vzdialenou správou a rozširovali jej funkcie. Podstatná a dosť nebezpečná je manipulácia konzole. V prípade OS Windows CMD alebo Powershell. Nebezpečná kvôli tomu, že cez konzolu je útočník schopný urobiť čokoľvek. Zmeniť rôzne veci v OS, sťahovať súbory, ktoré následne spustí a podobne. Veľké množtvo vývojárov RAT malvérov má zaujímavý zmysel pre humor, lebo viac ako 70% všetkých analyzovaných vzoriek malo tzv. „zábavnú“ funkcionalitu. Do tejto kategórie sme zaradili veci ako: prehodenie tlačidiel myši, skrytie ikon plochy, zmena pozadia, prehrávanie rôznej hudby, prevod textu na reč, ktorý bol následne prehraný, strašidelné animácie s hlasnými zvukmi, zobrazovanie rôzneho pornografického obsahu alebo aj falošná chyba zastavenia systému (Modrá smrť). Čo sa týka vlastností napájania ku ktorým patrí: vypnutie, hybernácia alebo reštart systému, tak sme ich zaznamenávali do špecifickej kategórie „Napájanie PC“. Dôvod prečo sme vytvorili pre túto funkciu špeciálnu kategóriu je ten, že sa jedná o veľmi praktickú vec aj keď môže byť plne nahradená jednoduchým príkazom v konzole. Funkcia DoS alebo taktiež „odmietnutie služby“ bola dosť prekvapivá a zaraďovali sme do nej 2 hlavné veci. DoS obete, ktorý bol najčastejšie spôsobený modrou smrťou, ktorá bola zapríčinená ukončením kritického procesu, ktorý samotný RAT vytvoril. Za druhú vec považujeme DDoS (Distribuovaný DoS) a každá obeť v botnete daného RATu po zapnutí tejto funkcie vysielala TCP alebo UDP pakety na špecifikovanú webovú lokalitu, ktorú chcel útočník zneprístupniť. Jedná sa o jednoduchý útok, ale zato efektívny. V priemere každý druhý RAT kontroloval či sa na OS na ktorom bol spustený nenachádza nejaký antivírus (AV) alebo či nebol spustený vo virtualizovanom prostredí (VM). Následne sa buď snaží daný AV vypnúť, poškodiť alebo sa snaží pridať do výnimiek jeho detekcie. Čo sa týka VM, tak na toto prostredie tiež reagujú malvéry rôzne. Niektoré sa začnú správať úplne rozdielne a nespustia svojú hlavnú funkcionalitu. Môžu taktiež vyvolať výnimku vďaka ktorej ich hlavný proces prestane regovať alebo sa jednoducho vypnú. Jednoduchou a pritom elegantnou funkciou je manipulácia clipboardu (kopírovacej schránky). Túto funkciu využívali väčšinou útočníci buď

na vkladanie zábavných textov alebo mazanie toho čo si obeď skopíruje. V malom množstve prípadov sa našli aj programy, ktoré automaticky menili čísla účtov bank alebo bitcoinových adries na čísla účtov útočníkov alebo na ich krypto peňaženky. Z toho dôvodu je nevyhnutné si vždy skontrolovať aspoň začiatok a koniec skopírovaného textu. Aby útočníci získali väčšie množstvo obetí tak do svojich malvérov implementujú funkcionalitu vďaka ktorej je malvér schopný ďalej sa šíriť. Čo sa týka šírenia môže sa jednať o kopírovanie na iné disky, flash-disky alebo sieťové umiestnenia. 41,9% výskyt nie je vôbec zanedbateľný. Poslednou sledovanou funkciou bolo maskovanie samotného malvéru. Zaraďovali sme tu RATy, ktoré nastavovali svojim súborom atribút skrytého súboru alebo používali hooky na vyfiltrovanie svojich procesov zo Správcu úloh alebo iných manažérov procesov. Táto funkcia mala najmenší percentuálny výskyt a to z toho dôvodu, že konkrétne hooky sú už pokročilejšie techniky skrývania aké používajú napr. rootkity.



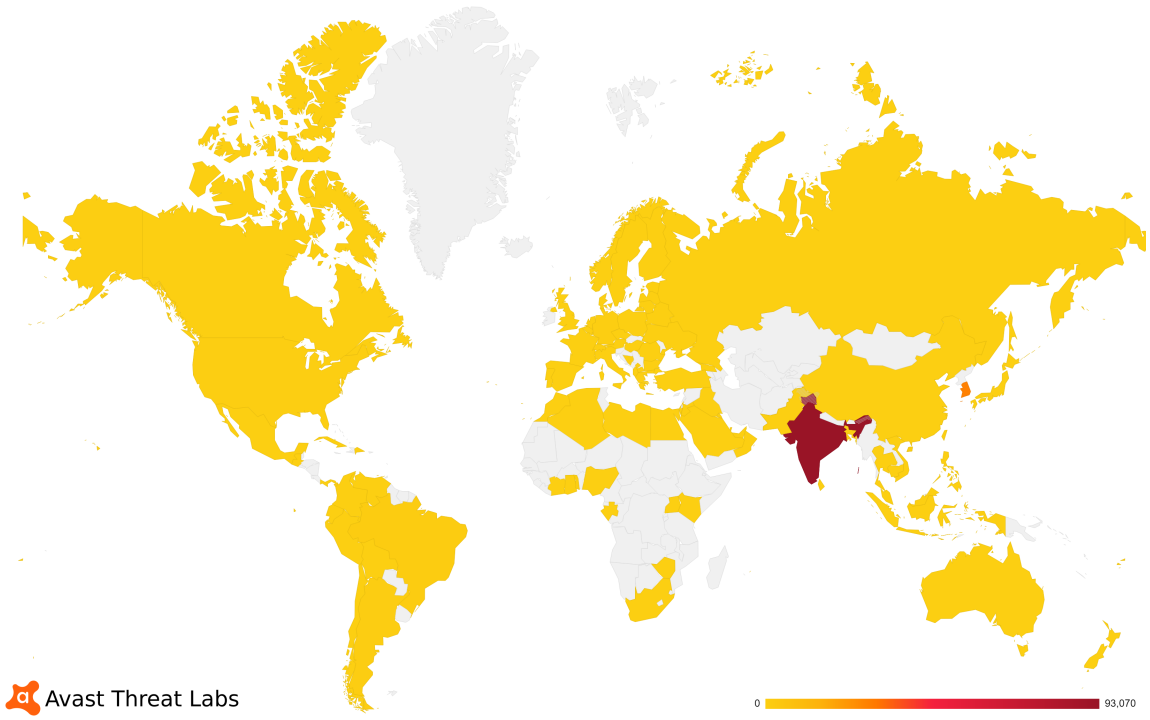
Obr. 5.4: Percentuálny výskyt jednotlivých funkcií

5.5 Krajina pôvodu

Čo sa týka krajín pôvodu je veľmi problematické zistiť z akej krajiny daný tvorca malvéru pochádza. Zvyčajne sme sa snažili vystopovať vývojára cez sociálne siete ako Twitter alebo Facebook, weby ako Youtube alebo sme sledovali aj na akých iných stránkach a fórach boli tieto malvéry zdieľané. Okrem toho sa dal vydedukovať pôvod aj podľa použitého jazyka, alebo prípadných metadát súboru, ktoré avšak mohli byť nepravdivé. V prípadoch kedy bolo možné dohľadať krajinu pôvodu sme sa dozvedeli, že to zvyčajne bývajú štáty ako USA, Rusko, Kanada, španielsky hovoriace štáty (Španielsko, Mexiko), Francúzsko, Turecko a množstvo arabsky hovoriacich krajín.

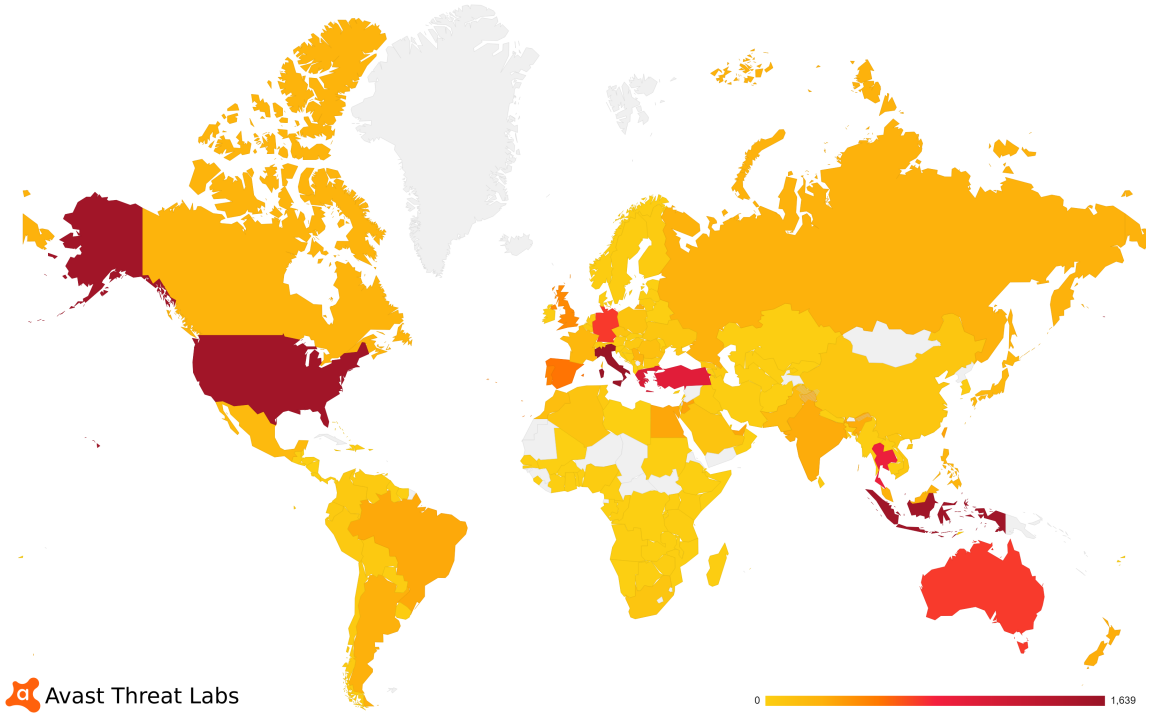
5.6 Efektivita napísaných pravidiel

V tejto sekcii vyhodnotíme efektivitu a funkčnosť napísaných pravidiel, ktoré boli použité na detekciu RAT rodín LuminosityLink, Warzone a Babylon. Štatistiky boli vytvorené z dát získaných od 01.01.2019 do 14.05.2019 a zahŕňajú počty používateľov antivírusového programu Avast u ktorých bol daný malvér zablokovaný. K týmto detekciám nepriamo prispeli aj nami napísané detekčné pravidlá nakoľko zobrazené štatistiky ukazujú počet detekcií vytvorených rôznymi detekčnými definíciami u používateľov.



Obr. 5.5: Mapa užívateľov u ktorých bol RAT LuminosityLink zablokovaný [23]

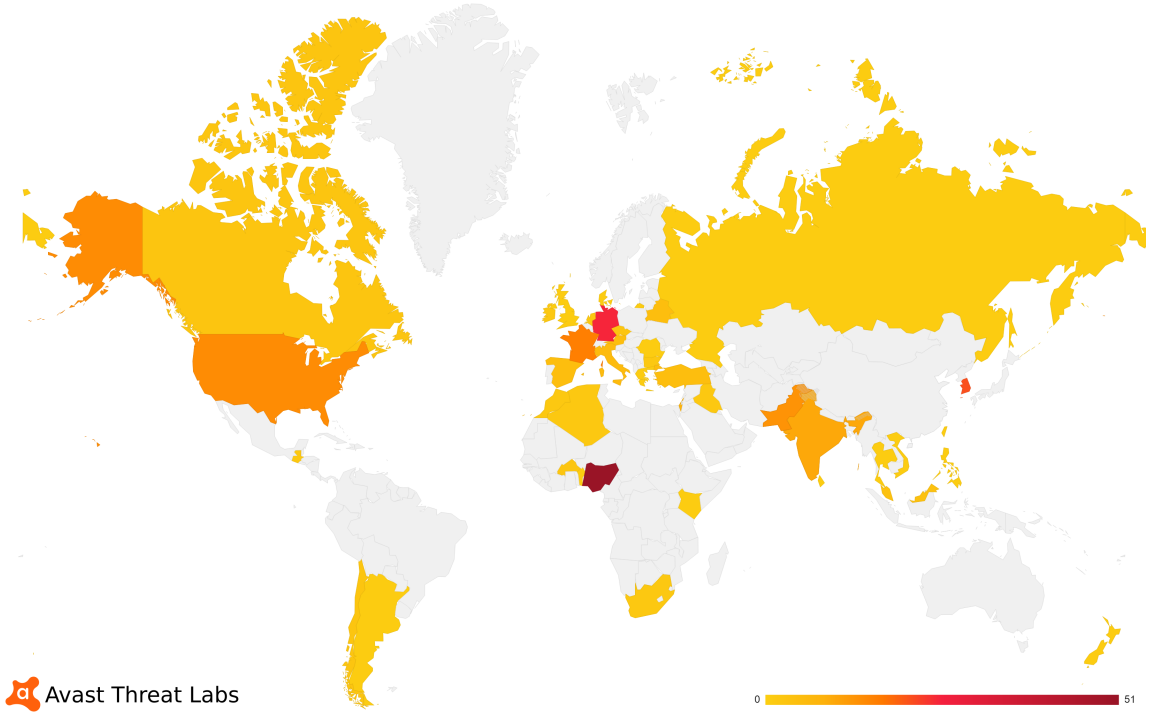
Čo sa týka RAT malvéru LuminosityLink tak celkový počet užívateľov u ktorých bol tento malvér detekovaný a zablokovaný od 01.01.2019 presahoval 127000. Z toho najčastejšie sa vyskytoval v Indii kde to bolo 93070 používateľov, 28667 v Južnej Kórei a 1646 v Brazílii. V rozsahu od 27.04. do 18.05. naše pravidlá samé detekovali 155 zástupcov tejto rodiny a z toho 17 na základe menných objektov a 138 na základe statických sekvencií, ktoré neobsahovali žiadne falošne-pozitívne zhody.



Obr. 5.6: Mapa užívateľov u ktorých bol RAT Warzone zablokovaný [24]

Malvér Warzone, ktorý bol vytvorený koncom roka 2018 a je jedným z najnovšie pokrytých RATov. Bol zablokovaný AV detekciou za dané obdobie u 19064 užívateľov. Z toho 1639 v Taliansku, 1576 v Indonézii, 1569 v USA a zvyšok v ostatných štátoch sveta. Pri prvom testovaní tohto pravidla bolo detekovaných viac ako 400000 vzorkov, ale jednalo sa o falošne-pozitívne zhody kvôli tomu, že statická časť pravidla obsahovala veľmi všeobecné reťazce, ktoré využívali nie len iné RATy ale aj legitimný softvér. Jednalo sa o prístup k heslám od Outlooku a FTP klientov. Po odladení statickej časti pravidla, ktorá tieto falošné detekcie spôsobovala dochádza k takýmto falošným detekciám len v minimálnej akceptovateľnej miere (aj to len v rámci malvéru) a z toho dôvodu je možné ho používať pre sledovanie aktivít tejto rodiny. Odladenie statickej časti prebehlo dňa 5.5. a počet detekovaných zástupcov rodiny Warzone sa k dátumu 18.05. pohyboval okolo 1563. Z toho 1415 vďaka pravidlu zameranému na statické sekvencie a 148 pravidlom obsahujúcim menné objekty a behaviorálne prvky. V detekciách statickou časťou sa už falošne-pozitívne

zhody nenachádzajú, ale behaviorálne pravidlo obsahuje zhruba 5% iných RATov nepatriacih do tejto rodiny.



Obr. 5.7: Mapa užívateľov u ktorých bol RAT Babylon zablokovaný [25]

Babylon RAT bol vytvorený až v roku 2015 a jeho detekčné pravidlo bolo vytvorené začiatkom apríla 2019. Počet užívateľov u ktorých bol tento malvér zablokovaný je 231. To najmä z krajín ako Nigéria (51), Nemecko (25) a Južná Kórea (21). Po preskúmaní samostatných detekcií bol statickou časťou nájdený len 1 predstaviteľ tejto rodiny. Dôvod je ten, že táto rodina používa u každej vzorky packer a pravidlo detekuje iba vzorky bez neho. Tento problém bude postupom času odstránený po implementácií unpackerov a deobfuskátorov priamo do automatizovanej analýzy a následnej detekcie. Čo sa týka časti menných objektov a behaviorálnych prvkov tak tá od 03.05.-19.05. našla 49 rôznych zastupiteľov tejto rodiny.

6 Záver

Po dôkladnom štúdiu malvérov Remote Access Trojan a taktiež spôsobmi akými sa analyzuje nie len tento druh, ale malvér obecne, sme dospeli k viacerým záverom. V prvom rade je ním neustály vývoj v tejto oblasti, ako aj nové techniky tvorcov malvéru na zakrytie správania ich výtvorov a taktiež skomplikovanie práce pre malvérových analytikov. Z toho dôvodu je dobrou praktikou sa neustále adaptovať na nové spôsoby detekcie. To bol hlavný dôvod prečo sme sa zamerali na modernú tvorbu detekčných pravidiel pomocou nástroju YARA, vďaka ktorému sme boli schopní dosiahnúť dostatočné detekčné výsledky pre reálne použitie v praxi. Taktiež sme sa dozvedeli, že množstvo aktívnych rodín neustále rastie so stúpajúcou popularitou tohto typu malvéru, obzvlášť v tejto dobe plnej sledovania, v ktorej sú privátne informácie cenné. Podarilo sa nám vytvoriť detekčné vzory obsiahnuté v kapitole 4 na jedny z najpoužívanejších RAT malvérov, ako aj získať väčšinu zaujímavých informácií týkajúcich sa práve týchto vzoriek.

Okrem iného sme zistili aké programovacie jazyky preferujú tvorcovia RAT malvéru, aké druhy protekcie používajú na skomplikovanie práce reverzným analytikom, taktiež sme zhodnotili aké funkcie by mal obsahovať každý plnohodnotý RAT a okrem toho aj funkcie ktoré nám prišli zaujímavé. Ďalej sme sa zamerali aj na oblasti šírenia vybraných malvérov ako aj krajiny pôvodu ich tvorcov.

Z tejto práce, ako aj zo samotnej problematiky malvéru je zrejmé, že sa naskytá možnosť ďalšieho pokračovania v rámci detekcie a analýzy RAT malvéru. Z toho dôvodu si myslíme, že by sa táto práca dala rozšíriť o ďalšie aktuálne a zaujímavé rodiny, ktoré používajú špecifické techniky pri ktorých by bola tvorba detekčných pravidiel skutočnou výzvou. Taktiež by si špeciálnu pozornosť zaslúžili aktuálne trendy posledných rokov, ktorými sú ransomware a coin-miner moduly priamo implementované do RATov. Mimo iné k problematike sledovania, dohľadávania autorov a distribútorov malvéru by bolo vhodné navrhnúť lepší systém predávania informácií pravným zložkám, ktoré by v podobných prípadoch mali jednať podstatne svižnejšie nakoľko súdny systém nie je pripravený na dnešnú rýchlu dobu.

Literatúra

[1] SZOR, Peter. *The art of computer virus research and defense.* Upper Saddle River, NJ: Addison-Wesley, 2005. ISBN 0321304543.

[2] ESET. *Backdoor, Remote Access Tool/Remote Access Trojan (RAT) [online].* [cit. 2018-12-05]. Dostupné z: www.virusradar.com/en/glossary/backdoor

[3] KONDALWAR, Manjeri N. a Prof. C.J. SHELKE. *Remote Administrative Trojan/Tool (RAT).* International Journal of Computer Science and Mobile Computing. 2014, 2014(3), 6. ISSN 2320–088X.

[4] DANG, Bruce, Alexandre GAZET, Elias BACHAALANY a Sebastien JOSSE. *Practical reverse engineering: x86, x64, ARM, Windows Kernel, reversing tools, and obfuscation.* Indianapolis, Indiana: Wiley, [2014].

[5] SIKORSKI, Michael a Andrew HONIG. *Practical malware analysis: the hands-on guide to dissecting malicious software.* Upper Saddle River, NJ: Addison-Wesley, 2005. ISBN 0321304543.

[6] EILAM, Eldad a Elliot J CHIKOFSKY. *Reversing: secrets of reverse engineering.* Indianapolis, IN: Wiley, c2005. ISBN 9780764574818.

[7] Microsoft. *Process Explorer [online].* 2017-05-16 [cit. 2018-12-05]. Dostupné z: www.docs.microsoft.com/en-us/sysinternals/downloads/process-explorer

[8] Microsoft. *Process Monitor [online].* 2017-09-12 [cit. 2018-12-05]. Dostupné z: www.docs.microsoft.com/en-us/sysinternals/downloads/procmon

[9] *Detect It Easy [online].* 2018-07-20 [cit. 2018-12-05]. Dostupné z: www.github.com/horsicq/Detect-It-Easy

[10] Hex-Rays SA. *Executive Summary: IDA Pro – at the cornerstone of IT security [online].* 2 [cit. 2018-12-05]. Dostupné z: www.hex-rays.com/products/ida/ida-executive.pdf

[11] x64dbg. *What is x64dbg [online].* 2016 [cit. 2018-12-05]. Dostupné z: www.x64dbg.readthedocs.io/en/latest/what.html

[12] DnSpy. *DnSpy [online].* 2007 [cit. 2018-12-05]. Dostupné z: www.github.com/0xd4d/dnSpy

[13] *Autoit3 decompiler – Exe2Aut [online].* 2014 [cit. 2018-12-05]. Dostupné z: www.domoticx.com/autoit3-decompiler-exe2aut/

[14] *De4dot [online].* 2011 [cit. 2018-12-05]. Dostupné z: www.github.com/0xd4d/de4dot

[15] VirusTotal. *YARA's documentation [online].* [cit. 2018-12-05]. Dostupné z: www.yara.readthedocs.io/en/v3.7.0/

[16] Stichting Cuckoo Foundation. *Cuckoo Sandbox - Automated Malware Analysis [online].* [cit. 2018-12-05]. Dostupné z: www.cuckoosandbox.org/

[17] *NetSupport Manager Version 8.0: Remote Control Software Launched [online].* 31.01.2003 [cit. 2019-05-18]. Dostupné z: https://www.itninja.com/blog/view/netsupport-manager-version-8-0-remote-control-software-launched

[18] *Cyberspies' code a click away [online].* 31.03.2009 [cit. 2019-05-18]. Dostupné z: https://www.thestar.com/news/gta/2009/03/31/cyberspies_code_a_click_away.html

[19] *NJRat [online].* 19.01.2017 [cit. 2019-05-18]. Dostupné z: https://www.cyber.nj.gov/threat-profiles/trojan-variants/njrat

[20] *LuminosityLink [online].* 18.07.2018 [cit. 2019-05-12]. Dostupné z: https://krebsonsecurity.com/2018/07/luminositylink-rat-author-pleads-guilty/

[21] *Orcus rat [online].* 19.04.2019 [cit. 2019-05-12]. Dostupné z: https://krebsonsecurity.com/2019/04/canadian-police-raid-orcus-rat-author/

[22] *Warzone 1.88 Plans [online].* 03.09.2018 [cit. 2019-05-18]. Dostupné z: https://warzone.io/pricing.html

[23] *Mapa výskytu LuminosityLink [online].* 18.05.2019 [cit. 2019-05-20]. Dostupné z: https://www.avast.com

[24] *Mapa výskytu Warzone RAT [online].* 18.05.2019 [cit. 2019-05-20]. Dostupné z: https://www.avast.com

[25] *Mapa výskytu Babylon RAT [online].* 18.05.2019 [cit. 2019-05-20]. Dostupné z: https://www.avast.com

Zoznam symbolov, veličín a skratiek

RAT	Remote Access Trojan
GUI	Graphical user interface
TCP	Transmission Control Protocol
RE	Reverse engineering
OS	Operačný systém
DLL	Dynamic-link library
PE	Portable Executable
COFF	Component Object File Format
DIE	Detect it easy
SDK	Software development kit
GPL	General public license
DoS	Denial of Service
DDoS	Distributed Denial of Service
API	Application Programming Interface
VNC	Virtual Network Computing
AV	Antivírus
VM	Virtual Machine

Zoznam príloh

A Obsah přiloženého CD

Príloha umiestnená na CD nosiči obsahuje tabuľku „Podrobná analýza RAT“, ktorá obsahuje doposiaľ analyzované malvérové rodiny vo formáte s príponou .xlsx určenú pre program Microsoft Excel 2016 a zložku nazvanú „Detekčné pravidlá“ v ktorej sa nachádza 10 detekčných YARA pravidiel.