Usporedba metoda obfuskacije koda u svrhu izbjegavanja EDR i XDR zaštite

Bilušić, Josipa

Sveučilište Jurja Dobrile u Puli

Tehnički fakultet u Puli

Preddiplomski sveučilišni studij Računarstvo

JOSIPA BILUŠIĆ

USPOREDBA METODA OBFUSKACIJE KODA U SVRHU IZBJEGAVANJA EDR I XDR ZAŠTITE

Završni rad

Pula, kolovoz, 2023. godine

Sveučilište Jurja Dobrile u Puli

Tehnički fakultet u Puli

Preddiplomski sveučilišni studij Računarstvo

JOSIPA BILUŠIĆ

USPOREDBA METODA OBFUSKACIJE KODA U SVRHU IZBJEGAVANJA EDR I XDR ZAŠTITE

Završni rad

JMBAG: 0303098817, redoviti student

Studijski smjer: Sveučilišni preddiplomski studij Računarstva

Kolegij: Sigurnost računalnih sustava

Znanstveno područje: Tehničke znanosti

Znanstvena grana: Programsko inženjerstvo

Mentor: prof. dr. sc. Tihana Galinac Grbac

Pula, kolovoz, 2023. godine

IZJAVA O AKADEMSKOJ ČESTITOSTI

Ja, dolje potpisani Josipa Bilušić, kandidat za prvostupnika računarstva ovime izjavljujem da je ovaj Završni rad rezultat isključivo mojega vlastitog rada, da se temelji na mojim istraživanjima te da se oslanja na objavljenu literaturu kao što to pokazuju korištene bilješke i bibliografija. Izjavljujem da niti jedan dio Završnog rada nije napisan na nedozvoljen način, odnosno da je prepisan iz kojega necitiranog rada, te da ikoji dio rada krši bilo čija autorska prava. Izjavljujem, također, da nijedan dio rada nije iskorišten za koji drugi rad pri bilo kojoj drugoj visokoškolskoj, znanstvenoj ili radnoj ustanovi.

Student
Josipa Bilušić

U Puli, kolovoz, 2023. godine

IZJAVA
o korištenju autorskog djela

Ja, Josipa Bilušić dajem odobrenje Sveučilištu Jurja Dobrile
u Puli, kao nositelju prava iskorištavanja, da moj završni rad pod nazivom ”Usporedba metoda obfuskacije koda u svrhu izbjegavanja EDR i XDR zaštite” koristi na način da gore navedeno autorsko djelo, kao cjeloviti tekst trajno objavi u javnoj internetskoj bazi Sveučilišne knjižnice Sveučilišta Jurja Dobrile u Puli te kopira u javnu internetsku bazu završnih radova Nacionalne i sveučilišne knjižnice (stavljanje na raspolaganje javnosti), sve u skladu s Zakonom o autorskom pravu i drugim srodnim pravima i dobrom akademskom praksom, a radi promicanja otvorenoga, slobodnoga pristupa znanstvenim informacijama.
Za korištenje autorskog djela na gore navedeni način ne potražujem naknadu.

U Puli, 17. kolovoza, 2023. godine

Potpis
Josipa Bilušić

Sadržaj

1. Uvod

Kroz povijest kibernetičke sigurnosti, kako su se razvijali načini djelovanja zloćudnog koda, tako su se razvile i tehnike izbjegavanja otkrivanja te neprimjetnim kompromitiranja sustava. Među tim tehnikama, obfuskacija koda ističe se po svojoj strategiji otežavanja razumijevanja te analiziranja izvršnog koda, bez mijenjanja funkcije istog [1]. Ovakva strategija se može koristiti u cilju povećavanja informacijske sigurnosti, no jednako se tako može koristiti u zlonamjerne svrhe i narušavanju sigurnosti. Tako, namjernom manipulacijom strukture i logike koda, nastoji se skriti zlonamjerne aktivnosti te izbjeći EDR (eng. *Endpoint Detection and Response*) te XDR (eng. *eXtended Detection and Response*) zaštitu.

U ovom završnom radu će se razmotriti tema obfuskacije koda, počevši od elaboriranja pojma i povijesti zloćudnog koda te ukazivanja potencijalnih prijetnji koje on predstavlja. Isto tako, pružit će se sažeti pregled nekih od najznačajnijih tipova zloćudnog koda, ističući njihove distinktivne karakteristike. Povrh toga, istražit će se i značaj detekcije ovakvih prijetnji kroz programe namijenjene za detekciju malicioznih programa, kao i različite tehnike, alate i analize. Tako omogućujući dublje razumijevanje učinka prijetnji i zaštite informacijskih sustava, posebice ističući EDR te XDR kao zaštitu. U konačnici, razložit će se obfuskacija programa, kroz obfuskaciju izvorišnog/izvršnog koda i izvršnih datoteka. Tu se kroz različite tehnike i alate opisuje pojam i važnost obfuskacije programa, ali i pojasniti kako svaka od tehnika i alata funkcionira.

Cilj ovog rada je istražiti različite metode obfuskacije koda te procijeniti njihovu učinkovitost u izbjegavanju EDR te XDR zaštite. Ovo će se provesti korištenjem različitih *Command&Control* (C2) radnih okvira, kao što su *Metasploit*, *PowerShell Empire*, *Havoc C2* i *Hoaxshell*. Koristeći ove radne okvire generirat će se maliciozne datoteke koje se često koriste u stvarnim napadima kako bismo paralelno usporedili neke od tehnika obfuskacije, ali i detekcije malicioznih programa.

2. Zloćudni kod

Zloćudni kod (eng. *malicious software*, *malware*) naziv je koji predstavlja vrstu programa (eng. *software*) koji su zlonamjerni, odnosno namijenjeni da nanose štetu elektroničkim uređajima i njihovim korisnicima iskorištavajući njihove ranjivosti [2]. Ovaj zlonamjerni kod služi kao sredstvo kompromitiranja, oštećivanja, iskorištavanja ili manipuliranja širokog raspona meta, uključujući razne uređaje, pojedine korisnike, velike organizacije, računalne mreže, poslužitelje te druga tehnološka i informacijska okruženja, ovisno o namijeni zloćudnog koda [3]. Zloćudni kod može se širiti koristeći različite metode i tehnike koje su uvjetovane ciljem programa, kao i znanjem njegovog autora – *malware* se može širiti društvenim inženjeringom (eng. *social engineering*)[1], iskorištavanjem ranjivosti u drugim programima ili mrežnim uređajima, fizičkim prijenosom (primjerice USB memorija) i slično.

Razvojem kibernetičke sigurnosti mijenjao se i način djelovanja zloćudnog koda pa su se u isto vrijeme razvile i tehnike kamuflaže (eng. *obfuscation, evasion*) zloćudnog koda. Ove tehnike omogućavaju zloćudnom kodu da se neprimjetno sjedini s informacijskim okruženjem uređaja, skrivajući se od antivirusnih alata i povećavajući svoju sposobnost da nanesu veću štetu. Evolucija tehnika kamuflaže zloćudnog koda ključna je za temeljitije razumijevanje različitih metoda prikrivanja u kibernetičkoj sigurnosti. Iako postoje brojni načini obfuskacije koda i zaobilaženja detekcija [4], sve tehnike mogu se smjestiti u nekoliko kategorija - obfuskacija izvorišnog odnosno izvršnog koda, pasivno te aktivno izbjegavanje antivirusnih proizvoda. Prve dvije kategorije odnose se na modificiranje samih programa u izvorišnom ili u izvršnom obliku kako bi se otežala ručna analiza te kako bi se zaobišli osnovni podsustavi antivirusnih programa, poput sustava za usporedbu potpisa odnosno sažetaka datoteka. Pasivno izbjegavanje odnosi se na sposobnost zloćudnog programa da prepozna nedostatke antivirusnog programa (primjerice smanjenu vidljivost u dijelovima sustava) te to iskoristi, dok aktivno

[1] Definirano u pogl. 2.3. Napredne ustrajne prijetnje

izbjegavanje podrazumijeva da zloćudni kod sadržava funkcionalnost koja aktivno pokušava sabotirati antivirusno rješenje.

2.1. Povijest zloćudnog koda

Od početnih instanci zloćudnih kodova do izuzetno složenih kibernetičkih prijetnji u suvremenom digitalnom dobu, kontinuiran je razvoj tehnologije utjecao na napredak zloćudnih kodova, ali i na razvoj računalne sigurnosti. Prvobitni koncept zloćudnog koda stvorio je John von Neumann tijekom 1940-ih [5]. U samom početku, koncept je bio samo teoretski eksperiment kinetičkog modela stroja koji se sam umnožavao. Nakon toga, John von Neumann napisao je rad o stroju koji se sam umnožava pod nazivom "*Teorija samoreproducirajućih automata*" (eng. "*Theory of Self-Reproducing Automata*") koji je objavljen 1966. godine.

Prvi program nalik zloćudnom kodu napisao je Robert Thomas pod nazivom *Creeper*, 1971. godine [6]. Iako je program nalikovao računalnom crvu [7], zapravo nije napravljen da uzrokuje štetu zahvaćenim računalima ni mrežama zbog čega mu nedostaje karakteristika zlonamjernosti obično povezana s modernim zloćudnim kodom.

Prvi zloćudni kod koji se sam umnožava napisan je 1974. godine pod nazivom *Rabbit*, isto tako poznat i pod nazivom *Wabbit* [8]. Ovaj zlonamjerni program spada pod računalne crve jer je mogao napasti samo računala na koje je stavljen te radi na način da rapidno umnožava sam sebe, troši resurse i time ruši sustav.

2.2. Tipovi zloćudnog koda

Danas postoji više različitih tipova zloćudnih kodova koji se mogu podijeliti u određene kategorije, ovisno o karakteristikama koje posjeduju. Međutim, zbog određenih specifičnosti zloćudnih kodova, važno je napomenuti da neki od njih mogu pripadati u više kategorija istovremeno. Neke od poznatijih kategorija zloćudnih kodova su računalni crvi (eng. *computer worm*), trojanski konji (eng. *trojan horse*), ucjenjivački kodovi (eng.

ransomware), botovi odnosno mreže botova (eng. *bot*, *botnet*), *rootkit* i mnogi drugi. Svaka od ovih kategorija predstavlja različite strategije, taktike i tehnike koje omogućavaju ostvarenje ciljeva i izbjegavanje detekcije. Svaka od ovih kategorija predstavlja različite strategije, taktike i tehnike koje omogućavaju ostvarenje ciljeva i izbjegavanje detekcije.

2.2.1. Trojanski konj

Trojanski konj (eng. *Trojan horse*, *Trojan*) je tip zloćudnog koda koji se prikriva kao legitimni program ili datoteka iako sadrži jedan ili više zloćudnih kodova. Ime je potaknuto Trojanskim konjem iz grčke mitologije [9]. Ondje je Trojanski konj predstavljao drvenu strukturu koju je Grčka vojska izradila tijekom Trojanskog rata, dok su se unutar konja skrivali vojnici kako bi ih napali u ključnom trenutku. Na sličan način funkcionira i zloćudni kod Trojanskog konja – korisniku se naizgled čini kao autentičan program, no sadrži korisniku nevidljive maliciozne funkcionalnosti. Napadi pomoću trojanskog konja često se oslanjaju na različite oblike prijevara korisnika. Primjerice, ove prijevare često budu putem privitka u elektroničkoj pošti, oglasa, poveznica na internetskoj stranici, datoteke, igre, aplikacije, alata i čak programskih zakrpa (eng. *software patches*), unutar kojih se nalazi skriveni zloćudni kod [10].

2.2.2. Alat za udaljeni pristup

Alat za udaljeni pristup (eng. *Remote Access Tool/Trojan*, *RAT*), često se naziva i trojanskim konjem za udaljeni pristup te predstavlja tip zloćudnog koda koji omogućava udaljeni pristup ciljanom uređaju [11] [12]. Slično Trojancu, često se prikriva kao legitiman program ili datoteka, no nakon pokretanja na ciljanom uređaju omogućuje daljinsku kontrolu nad sustavom, izvođenje naredbi te pristup povjerljivim informacijama bez uzbunjivanja korisnika. Ova taktika podsjeća na prethodno spomenuti Trojanski konj, gdje se zloćudni kod prikriva unutar prividno korisnih elemenata kako bi se prevario korisnik i ostvarila kontrola nad sustavom. Glavni cilj ovog zloćudnog koda je omogućiti neovlašteni pristup na daljinu te time i kontrolu nad uređajem. Štoviše, ovakav zloćudni kod može služiti za nadzor i praćenje aktivnosti korisnika uređaja, pristup i krađu podataka,

špijunažu, izvođenje napada distribuirane uskraćenosti usluge te širenje drugih vrsta zloćudnih kodova, a da ostanu ne primijećeni i izbjegnu otkrivanje.

2.2.3. Špijunski program

Špijunski program (eng. *spyware*) je tip zloćudnog koda namijenjen neprimjetnom praćenju ponašanja korisnika te skupljanju povjerljivih informacija i uzoraka ponašanja s uređaja bez znanja ili potrebe za pristankom korisnika uređaja [13]. Osim praćenja ponašanja, špijunski program često ima sposobnost upravljanja ulazno/izlaznim uređajima pa tako primjerice napadači mogu uključivati mikrofon i kameru uređaja. Isto tako, napadači mogu iskoristiti i sposobnost snimanja zaslona i praćenja unosa s tipkovnice. Zbog ovakvih sposobnosti zloćudnog koda, napadači stječu opsežan nadzor nad korisnikom i dobivaju detaljan uvid u korisničko ponašanje i njihove aktivnosti, komunikacije i povjerljive informacije, kao što su PIN-ovi i lozinke.

2.2.4. Ucjenjivački kod

Ucjenjivački kod (eng. *ransomware*) je tip zloćudnog koda kojemu je cilj onemogućiti pristup podacima mete šifriranjem žrtvinih računala, odnosno podataka na njima [14]. Nakon što su podaci zaključani, napadači obavještavaju metu o napadu i zastrašuju te zahtijevaju otkupninu u zamjenu za ključ za dešifriranje podataka ili vraćanja pristupa podacima. Važno je istaknuti da povratak podataka nije osiguran plaćanjem otkupnine, čak ni nakon plaćanja otkupnine, a napadači nastavljaju izvršavati ovakve napade na druge mete. Ovo može uzrokovati ozbiljnu financijsku štetu i gubitak povjerljivih informacija što predstavlja ozbiljnu prijetnju računalnoj sigurnosti.

2.3. Napredne ustrajne prijetnje

Složeni i dugotrajni kibernetički napadi, odnosno grupe napadača, često s određenim ciljevima i namjenama, poznati su kao napredne ustrajne prijetnje (eng. *Advanced Persistent Threats*, APT) [15]. Označava ih prikriveni pristup, detaljno planiranje,

prilagođene strategije i taktike te dugotrajno prisustvo unutar ciljane mete. Njihov glavni cilj je ostvariti dugotrajno prisustvo kako bi imali pristup osjetljivim podacima i kontrolu nad ciljanom metom. Kako bi izbjegli sigurnosne mehanizme i ostali neotkriveni kroz dulje vremensko razdoblje, ove prijetnje koriste kombinaciju različitih društvenih i tehničkih strategija, taktika i napada, uključujući društveni inženjering, distribuciju zloćudnog koda i napade na identitete korisnika te autentifikacijske podatke. Društveni inženjering vrsta je tehnike koja predstavlja način prijevare i manipulacije pojedinca u svrhu dobivanja povjerljivih informacija korisnika ili na drugi način davanja napadaču pristup preko čovjeka [16]. Zbog njihove sposobnosti, APT se smatraju ozbiljnom prijetnjom da nanesu veliku štetu, imaju kontrolu nad vrijednim informacijama, kompromitiraju ključnu infrastrukturu i minimaliziraju sigurnost organizacija, pojedinaca ili drugih meta.

APT grupe često rade pod okriljem državnih službi [17] (eng. *state-sponsored*, *nation-state*) te njihovi motivi najčešće nisu financijske prirode, već ovise o motivima naručitelja, odnosno države. Motivi, odnosno ciljevi su često predsjednički izbori, kritična državna infrastruktura, velike korporacije i slično. Neki od najpoznatijih APT-ova su *Lazarus*, *Conti*, *HAFNIUM* i mnogi drugi.

3. Programi za detekciju malicioznih programa

U području informacijske sigurnosti, ključni je zadatak očuvati sigurnost informacijskih okruženja od prisutnih opasnosti i prijetnji zloćudnih kodova. U tu svrhu, postoji vrsta programa posebno stvorena i oblikovana za otkrivanje, identifikaciju, analizu i suzbijanje izvršavanja te širenje zloćudnih kodova unutar ciljanih uređaja, čime se sprječava njihova šteta i neželjeni utjecaj na informacijski sustav. Ovakvi se programi često (pogrešno) nazivaju antivirusnim programima te predstavljaju klijentsku aplikaciju koja u stvarnom vremenu provjerava i analizira sustav kako bi otkrila i eliminirala prijetnje.

Obzirom na napredak zloćudnih kodova i prisutne prijetnje, važno je imati raznolikost tehnika i pristupa kako bi se osiguralo otkrivanje zloćudnih aktivnosti, omogućilo pravovremeno prepoznavanje i reagiranje na nove i nepoznate prijetnje, te time poboljšala sigurnost informacijskih sustava. Tehnike otkrivanja mogu se podijeliti u dvije osnovne kategorije, a to su statičko otkrivanje s jedne strane i dinamičko, odnosno heurističko/ponašajno otkrivanje s druge strane. Kako bi se osigurala sveobuhvatna zaštita od različitih prijetnji, danas se programi za otkrivanje zlonamjernih programa oslanjaju na kombinaciju ovih tehnika otkrivanja.

Statičke metode temelje se na uspoređivanju raznih statičkih indikatora nekog programa s bazom indikatora dobro poznatih malicioznih programa. To primjerice mogu biti rasute tablice (eng. *hash tables*) datoteka, prisutni znakovni nizovi, certifikati kojim je program potpisan i slično [18]. S druge strane, dinamička analiza temelji se na prepoznavanju uzoraka ponašanja pokrenutog programa koji se često dodatno analiziraju temeljem opisa poznatih malicioznih programa (npr. SIGMA pravila) ili uz pomoć raznih algoritama strojnog učenja [19]. Ovi uzorci ponašanja mogu uključivati neovlašteno stvaranje, brisanje i modificiranje datoteka i direktorija sustava, mijenjanje sigurnosnih postavki registra te postavki i konfiguraciju mrežnih komponenti ili veza, kao što su uspostavljanje novih mrežnih veza s udaljenim serverima ili uređajima i mnogi drugi [20].

3.1. Klasični antivirusni programi temeljeni na potpisima

Glavna uloga antivirusnih programa temeljenih na potpisima (eng. *signature-based antivirus program*) je identifikacija i otkrivanje zloćudnih kodova [21]. To se radi usporedbom datoteka i uzoraka potencijalno zloćudnog koda s bazom podataka poznatih i unaprijed definiranih potpisa[2] zloćudnih kodova. Zbog ovoga su ovi programi još poznati kao antivirusna rješenja temeljena na definicijama ili uzorcima virusa i predstavljaju bitan dio strategija održavanja informacijske sigurnosti. Antivirusni programi temeljeni na potpisima sadrže različite prednosti u prepoznavanju i suzbijanju poznatih prijetnji, iako se suočavaju s određenim ograničenjima. Ovi su programi vrlo efektivni kod otkrivanja učestalih napada, kao što su mrežne krađe identiteta, zloćudnih kodova te napada uskraćivanjem resursa. Također, sadrže nizak postotak lažnih pozitivnih rezultata pa rijetko pogrešno prepoznaju legitimne datoteke kao prijetnju. Potpisi se temeljito testiraju prije dodavanja u bazu podataka te se baza podataka potpisa redovno ažurira. Uz to, implementacija i korištenje ovog programa je brza, jednostavna i ne zahtjeva puno tehničkog znanja. Unatoč brojnim prednostima, važno je naglasiti da su vrlo ograničeni u otkrivanju novih prijetnji i varijanti postojećih napada s nepoznatim potpisima, odnosno potpisima koji nisu u bazi podataka. Osim toga, potrebno ih je redovito ažurirati, kako bi zadržali svoju efikasnost i smanjili rizik od napada.

3.2. Otkrivanje i odgovor na krajnjim točkama (EDR)

Otkrivanje i odgovor na krajnjim točkama[3] (eng. *Endpoint Detection and Response*, EDR) predstavlja tehniku pristupa sigurnosti informacijskih sustava koji je koncentriran na neprekidnom praćenju, prepoznavanju i reagiranju na potencijalne sigurnosne prijetnje na pojedinačnim uređajima, odnosno krajnjim točkama unutar računalne mreže [22] [23]. EDR sustav funkcionira tako da analizira sve komponente aktivnosti na uređajima, uključujući mrežne prijenose, tijek izvođenja procesa, I/O operacije nad datotekama te

[2] Definirano u pogl. 3.5.1. Potpisi datoteka

[3] U nastavku EDR ili EDR sustav

ponašanje korisnika. Kako bi prepoznao nova ili zloćudna ponašanja, koristi tehnike poput heuristike i strojnog učenja. Nakon što prepozna i otkrije potencijalne prijetnje, EDR sustav uključuje brzu reakciju s ciljem smanjivanja moguće štete i sprječavanja širenja prijetnje na ostale uređaje ili dijelove mreže. Zbog toga, EDR sustav ima mogućnost automatskog izoliranja zahvaćenih uređaja ili korisničkih računa te obavještavanja o otkrivenim zloćudnim ponašanjima. Kako bi omogućio dublju analizu događaja, taktiku napada te strategije obrane u budućnosti, ovaj sustav, uz kombinaciju statičke i dinamičke analize, aktivno prikuplja i pohranjuje podatke o zloćudnim ponašanjima i potencijalnim prijetnjama. Iako je EDR sustav nužan u modernim okruženjima, nema mogućnost korelacije aktivnosti među više radnih stanica odnosno korisnika – takve sposobnosti pruža XDR.

3.3. Prošireno otkrivanje i odgovor (XDR)

Prošireno otkrivanje i odgovor[4] (eng. *eXtended Detection and Response*, *XDR*) predstavlja naprednu sigurnosno rješenje koje ima mogućnost zaustavljanja i obrane od naprednih prijetnji na informacijske sustave [24]. Slično EDR-u, XDR sustav ima sposobnost prepoznavanja, brzog reagiranja i uklanjanja složenih prijetnji. To postiže koristeći napredne metode analize podataka i neprestanog praćenja ponašanja kako bi prepoznao i uklonio zloćudne programe. Osnovna uloga XDR sustava je pružanje napredne sigurnosne zaštite te povećanje sposobnosti otkrivanja i odgovora na napredne prijetnje u realnom vremenu. Ovo se realizira na način da prikuplja i pohranjuje podatke iz uređaja te podatke o potencijalnim prijetnjama, prikuplja promet i analizira ponašanje unutar mreže te automatski odgovara na prijetnje. Upravo je dodavanje mrežne komponente, odnosno kontekstualne analize mrežnog prometa, korelacija ponašanja više radnih stanica i korisničkih računa ključna razlika između EDR i XDR sustava – pojednostavljeno, za razliku od EDR-a, koji sve radnje i funkcionalnosti usmjerava na pojedini uređaj, XDR uzima u obzir sve dostupne kontekstualne indikatore kako bi detektirao radi li se o napadu koji bi potencijalno zahvatio više od jednog uređaja odnosno

[4] U nastavku XDR ili XDR sustav

cijelu mrežu, primjerice kretanje napadača kroz mrežu i pokretanje ucjenjivačkog koda na svim radnim stanicama kao konačna faza napada.

3.4. Portable Executable format datoteke

Format datoteka koji se često koristi unutar *Windows* operacijskog sustava format je izvršnih datoteka u prijenosnom obliku (eng. *Portable Executable (PE) File Format*) [25]. On sadrži informacije o programskom kodu, podacima, resursima te metapodacima koji su potrebni *Windows* sustavu za učitavanje i izvođenje programa. PE format se na *Windows* uređajima koristi za izvršne programe (*.exe*), biblioteka dinamičkih veza (*.dll*), upravljačke programe uređaja (*.sys*) i druge sistemske datoteke. Struktura PE formata sastoji se od nekoliko važnih sekcija. Primarno, na početku svake PE datoteke prisutno je MZ odnosno PE zaglavlje, koje sadrži osnovne informacije o datoteci, primjerice kada je kreirana, za koju arhitekturu je izgrađena i slično. Isto tako, u zaglavlju se nalaze i podaci o veličinama pojedinih sekcija, koje su bitne kod detektiranja pakiranih datoteka. Također bitne sekcije su tablice s podacima uvezenih odnosno izvezenih programskih metoda (eng. *Import Address Table – IAT*, *Export Address Table – EAT*). Ove sekcije sadrže informacije o pojedinim funkcijama koje se u aplikaciju učitavaju iz raznih DLL datoteka te tako konačnoj aplikaciji omogućuju pozivanje funkcija iz učitanih DLL-ova. Ostale sekcije sadrže relokacijske informacije (informacije o raspodjeli datoteke na disku ukoliko se ne može cijela učitati u radnu memoriju), podatke o resursima (primjerice ikonica programa), debug informacije i slično [26].

3.5. Detekcija statičkom analizom

Detekcija statičkom analizom metoda je analiziranja koda, odnosno datoteka koje sadrže kod kako bi se otkrila potencijalna prisutnost zlođudnog koda ili zlonamjernih aktivnosti. Radi se bez izvođenja ili pokretanja koda i usmjerena je na samu strukturu i sadržaj datoteka kako bi se identificirali određeni znakovi ili uzorci koji ukazuju na prisutnost zlođudnog koda. To uključuje identifikatore, kao što su imena i tipovi datoteka, *stringovi*

poput IP adrese i domene, te potpisi[5] datoteka zloćudnog koda, uz mnoge druge karakteristike i logike zloćudnog koda. Ova analiza se provodi kako bi se otkrile potencijalne prijetnje i opasnosti unutar datoteke, bez da pritom ista nanese štetu ili drukčije naruši sigurnost sustava. Za razliku od dinamičke analize u kojoj se izvršava potencijalno zloćudni kod i prate njegove aktivnosti, uloga statičke analize je da ovo provodi kroz analizu samog sadržaja datoteka, bez njihovog izvođenja.

Nakon pripreme datoteke ili koda te okruženja za analizu, provodi se analiza koristeći različite tehnike za pregled koda. Neke od često korištenih tehnika uključuju korištenje alata poput heksadecimalnog editora, disassemblera, dekompajlera te alata za pronalaženje grešaka. Ukoliko se pronađu potencijalne opasnosti i prijetnje, na temelju rezultata rade se izvješća koja sadrže informacije o njima. U slučaju da se smatra da predstavljaju ozbiljnu prijetnju, donose se odluke o postupanju s datotekama. Ovi postupci mogu uključivati izolaciju, brisanje ili ispravke radi uklanjanja ranjivosti.

3.5.1. Potpisi datoteka

Jedna od osnovnih metoda za identifikaciju prijetnji i zloćudnog koda je pregled sadržaja datoteke kako bi se uočili jedinstveni identifikatori ili uzorci unutar datoteke, poznati kao potpis datoteka (eng. *file signature*) zloćudnog koda [27]. Potpis datoteka je specifični uzorak, niz koda ili karakteristika unutar datoteke, uključujući izvršne programe i dokumente, koji su povezani sa zloćudnim kodom te su jedinstveni ovisni o vrsti ovakvog koda. Zbog toga ima važnu ulogu u otkrivanju, identifikaciji i uklanjanju prijetnji koje programi za otkrivanje zloćudnih kodova pokušavaju detektirati. Statička se analiza oslanja na prepoznavanju ovih potpisa kako bi se prepoznale i otkrile poznate prijetnje prije nego što se datoteke izvrše ili otvore. Ova metoda omogućava brže i preciznije reagiranje te provođenje temeljite analize i postupanje s potencijalno opasnim datotekama.

[5] Definirano u pogl. 3.5.1.Potpisi datoteka

3.5.2. Analiza dijelova izvršnih datoteka u prijenosnom obliku

Postupak ispitivanja i procjene datoteka koje se mogu izvršiti na uređaju te koje su pohranjene u formatu koji se može lako prenositi naziva se analizom dijelova izvršnih datoteka u prijenosnom obliku[6] (eng. *analysis of PE file format*) [28]. Ova analiza omogućava raniju detekciju i upozoravanje protiv zlćudnih kodova i drugih prijetnji provjerom entropije datoteke odnosno njenih pojedinih sekcija, funkcionalnosti koja se poziva iz drugih DLL datoteka i slično. Podstrukture PE datoteka koje je moguće analizirati uključuju DOS i PE zaglavlje, konfiguracijsko zaglavlje (*IMAGE_OPTIONAL_HEADER*), tablica sekcija te pojedine sekcije. Neke od najkorisnijih sekcija za analizu su resursna sekcija, koja često sadržava resurse koje datoteka može koristiti, poput dodatnih učitanih datoteka ili ikonica samog programa, te .*text* sekcija koja sadrži izvršni kod, odnosno asemblerske instrukcije u heksadecimalnom zapisu.

Analiza PE datoteke uključuje ispitivanje različitih dijelova strukture datoteke, a jedan od važnijih dijelova koji se provjeravaju su imena sekcija odnosno zaglavlja unutar datoteke. One mogu otkriti puno informacija o pakirajućem programu ili kompajleru koji su se, i ukoliko su se, koristili. PE datoteka može sadržavati različite odjeljke, uključujući .*text*, .*code* ili .*data*, te se mogu koristiti razni pakirajući programi, uključujući *UPX* ili *MPress* pakere. Pomoću ovih naziva mogu se saznati bitne informacije o specifičnim pakirajućim programima.

Prilikom analize DOS zaglavlja, provjerava se prisutnost neobičnog ili zlonamjernog sadržaja. Provjerava se kako bi se osiguralo da je ispravno oblikovan i ne sadrži sumnjiv sadržaj, iako je rijetko da ovaj dio sadrži prijetnje. Kod analize DOS potprograma utvrđuje se postoje li neobična ponašanja (primjerice znakovni niz „*That program cannot be run in DOS mode*“ umjesto „*This program...*“) te prisutnost poruka koje sadrže zlćudne namjene, premda su prijetnje rijetke u ovom dijelu. Zatim, jedan od važnijih dijelova za analizu je zaglavlje PE datoteke. Ukoliko se tu uoče neobične ili sumnjive aktivnosti, poput čudnog vremena kompilacije ili nestandardne arhitekture, provodi se dublja analiza i provjera. Kod konfiguracijskog zaglavlja provjeravaju se informacije o datoteci te se osigurava da sadrži određena svojstva koja bi mogla ukazivati na neobična ponašanja ili

[6] U nastavku PE datoteka

potencijalnu prijetnju. Zloćudni kod se može skrivati i unutar tablice odjeljaka zbog čega se analizira postoji li što sumnjivo ili zloćudno unutar imena odjeljka, veličine ili dopuštenja (primjerice, odjeljci koji su izvršivi, *executable/X,* su iznimno sumnjivi ukoliko u pitanju nije *.text* sekcija). Manipulacije funkcionalnosti datoteke rade se unutar rječnika podataka, zbog čega se provjerava jesu li pokazivači na važne podatke ispravni i ne sadrže potencijalne zloćudne karakteristike. Konačno, valja spomenuti i analizu *Import* odnosno *Export Address Table*[7] struktura, koje sadržavaju informacije o funkcionalnostima koje aplikacija iskorištava (primjerice uvoženje funkcija iz DLL-ova) ili koje aplikacija izvozi (ako se radi o .DLL PE datoteci koju analiziramo). Prilikom analize EAT odnosno IAT strukture valja obratiti pozornost na metode koje se uvoze te njihovu količinu. Naime, nije neuobičajeno da zloćudni kod inicijalno uvozi tek nekoliko funkcija iz *Windows* API sustava (primjerice *LoadLibraryA*) kako bi sakrio prave sadržaje IAT-a, a zatim dinamički učitava funkcionalnost iz memorije, često koristeći navedenu funkciju ili neke slične, druge funkcije.

3.5.3. YARA i SIGMA pravila

Alat dizajniran za prepoznavanje uzoraka i detekciju temeljenu na pravilima poznat je kao YARA (eng. *Yet Another Ridiculous Acronym*) radni okvir. To je moćan alat otvorenog koda koji ima važnu ulogu u borbi protiv zlonamjernih kodova, koji radi na temelju određenih pravila, pod nazivom YARA pravila (eng. *YARA rules*) [29]. Ova pravila predstavljaju skup unaprijed prilagođenih i definiranih uzoraka i uvjeta koji opisuju specifične karakteristike i uzorke ponašanja prisutne u datotekama ili memoriji. Često su statična i temelje se na definiranim uzorcima podataka unutar datoteka. Ona se koriste u okviru YARA alata za otkrivanje određenih datoteka, ponašanja ili svojstava vezanih uz zloćudni kod, prepoznavanje poznatih ranjivosti, identifikaciju specifičnih aplikacija ili potencijalno opasnih operacija, te druge prijetnje. Primjerice, YARA pravila se mogu koristiti za detektiranje okteta asemblerskih instrukcija za posebnu vrstu enkripcije specifične za određeni ucjenjivački kod, za detekciju neispravnih potpisa u zaglavljima datoteka koji mogu ukazivati na manipulaciju i slično. Na sljedećoj slici (Slika 1.)

[7] U nastavku EAT, odnosno IAT

prikazano je YARA pravilo koje kao uvjet za karakterizaciju neke datoteke kao „*Linux.Virus.Vit*“ zahtijeva da se u datoteci nalazi znakovni niz „*vi324.tmp*“, da datoteka započinje s heksadecimalnim oktetima „464C457F“ te da datoteka na lokaciji početne instrukcije (eng. *entry point*) sadržava niz okteta definiranih u varijabli „*vit_entry_point*“. Iako YARA uvjeti gotovo nikad nisu u potpunosti točni, odnosno ne garantiraju da je datoteka koja zadovoljava YARA pravilo uistinu taj određeni tip datoteke za koje je pravilo napisano, vrlo su dobar indikator koji pomaže u smanjenju datoteka koje bi sigurnosni stručnjaci trebali razmatrati u slučaju analize incidenata ili datoteka.

reversinglabs-yara-rules / yara / virus / **Linux.Virus.Vit.yara**

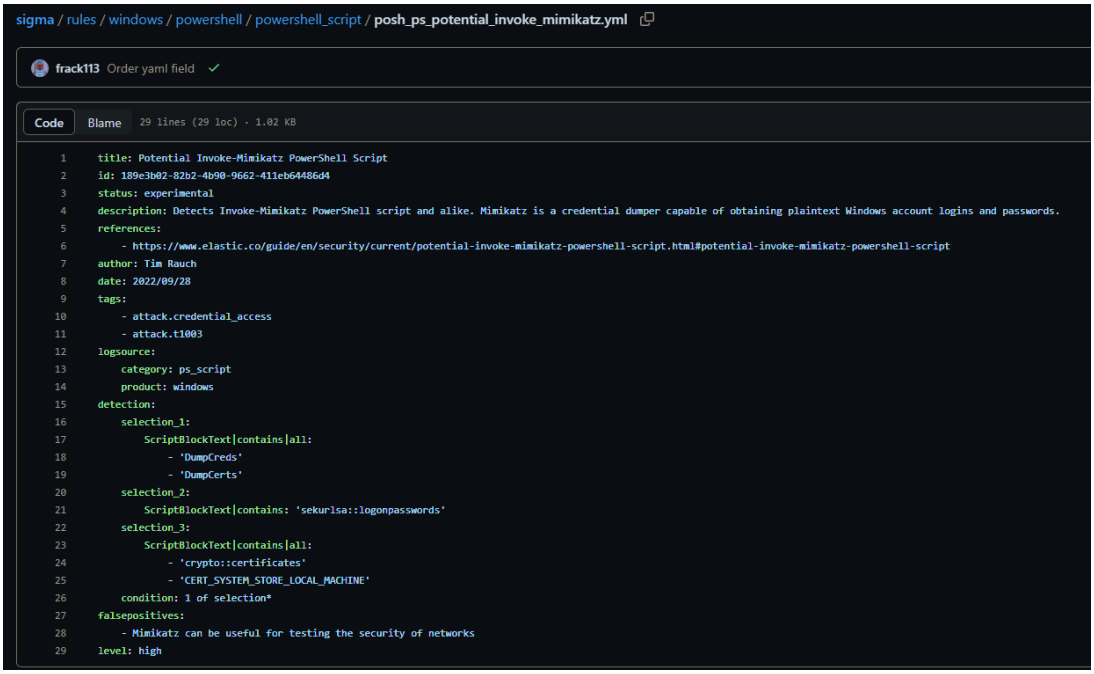
Threat Analyst Added new YARA rules. ✓

Code | Blame | 36 lines (28 loc) · 1.28 KB

```
import "elf"

rule Linux_Virus_Vit : tc_detection malicious
{
    meta:

        author              = "ReversingLabs"

        source              = "ReversingLabs"
        status              = "RELEASED"
        sharing             = "TLP:WHITE"
        category            = "MALWARE"
        malware             = "VIT"
        description         = "Yara rule that detects Vit virus."

        tc_detection_type   = "Virus"
        tc_detection_name   = "Vit"
        tc_detection_factor = 5

    strings:

      $vit_entry_point = {
        55 89 E5 81 EC 40 31 00 00 57 56 50 53 51 52 C7 85 D8 CE FF FF 00 00 00 00 C7 85 D4
        CE FF FF 00 00 00 00 C7 85 FC CF FF FF CA 08 00 00 C7 85 F8 CF FF FF B8 06 00 00 C7
        85 F4 CF FF FF AD 08 00 00 C7 85 F0 CF FF FF 50 06 00 00 6A 00 6A 00 8B 45 08 50 E8
        18 FA FF FF 89 C6 83 C4 0C 85 F6 0F 8C E6 01 00 00 6A 00 68 ?? ?? ?? ?? 56 E8 2E FA
        FF FF 83 C4 0C 85 C0 0F 8C C4 01 00 00 8B 85 FC CF FF FF 50 8D 85 00 D0 FF FF 50 56
        E8 2A FA FF FF 89 C2 8B 85 FC CF FF FF 83 C4 0C 39 C2 0F 85 9D 01 00 00 56 E8 E1 F9
        FF FF BE FF FF FF FF 6A 00 6A 00 E9
      }

      $vit_str = "vi324.tmp"

    condition:
        uint32(0) == 0x464C457F and $vit_entry_point at elf.entry_point and $vit_str
}
```

Slika 1. YARA pravila preuzeta s GitHub repozitorija (T. Pericin, F. Roth/GitHub repozitorij: reversinglabs-yara-rules, 7.6.2023.) https://github.com/reversinglabs/reversinglabs-yara-rules/blob/develop/yara/virus/Linux.Virus.Vit.yara (13.9.2023.)

SIGMA radni okvir je alat otvorenog koda namijenjen za prikupljanje, upravljanje i analizu prikupljenih podataka kojeg su kreirali Florian Roth i Thomas Patzke te predstavlja važnu ulogu u detekciji, analizi i reakciji na potencijalne prijetnje i incidente [30]. Pravila ili skripte koje se koriste u ovom okviru nazivaju se Sigma pravila (eng. *Sigma rules*). Ona predstavljaju definirana pravila ili uzorke ponašanja koja se primjenjuju na prikupljene podatke te služe za identifikaciju određenih ponašanja ili potencijalnih prijetnji. Ova pravila su više dinamička i mogu se prilagoditi okruženjima i potrebama. Primjerice, na sljedećoj slici (Slika 2.) prikazano je Sigma pravilo koje prepoznaje da se na sustavu pokreće maliciozni program za izvlačenje korisničkih pristupnih podataka (*mimikatz*) tako što u dnevničkim zapisima *Powershell*-a traži zapise o pokretanju naredbi „*DumpCreds*“, „*DumpCerts*“, „*sekurlsa::logonpasswords*“ ili drugih.



Slika 2. Sigma pravila preuzeta s GitHub repozitorija (T. Rauch/GitHub repozitorij:sigma, 26.10.2022.) https://github.com/SigmaHQ/sigma/blob/master/rules/windows/powershell/powershell_script/posh_ps_potential_invoke_mimikatz.yml *(13.9.2023.)*

YARA pravila se razlikuju od Sigma pravila po tome što su usmjerena na prepoznavanju i identifikaciji datoteka, objekata ili sadržaja na temelju njihovog sadržaja, kao što su uzorci podataka, znakovni nizovi, binarni kod i drugi karakteristični elementi. Isto tako, koriste se za detekciju određenih uzoraka vezanih uz zlćudni kod, zlonamjerne datoteke ili drugim potencijalnim prijetnjama na temelju njihovog statičkog sadržaja. Sigma pravila su, s druge strane, usmjerena na prepoznavanju neobičnih ili sumnjivih ponašanja u sustavima ili mrežama, uključujući uzorke ponašanja koji se mogu primijetiti kroz evidenciju događaja i aktivnosti na sustavima. Za razliku od YARA pravila, ova se pravila koriste za detekciju određenih ponašanja koja bi mogla ukazivati na potencijalne prijetnje, napade ili neovlaštene aktivnosti, pomoću analize dnevničkih zapisa i događaja. Uz to, YARA se često koristi kao prvi sloj detekcije te se tako brzo prepoznaju zlonamjerni uzorci, dok se Sigma koristi za praćenje i analizu većeg raspona događaja i otkrivanje neobičnih uzoraka ponašanja koji bi mogli biti potencijalne prijetnje.

3.6. Otkrivanje dinamičkom analizom

Za razliku od statičke analize koja se temelji na pregledu sadržaja određenih datoteka i programa, dinamička analiza uključuje izvršavanje potencijalno zlćudnog koda te praćenje njegove aktivnosti. Glavna uloga ove analize je prepoznavanje i detaljno proučavanje kako se određena datoteka ili program ponaša u stvarnom vremenu i potencijalno otkrivanje neželjenog ili zlonamjernog ponašanja, uključujući njegove interakcije s ostalim komponentama sustava [31].

Potencijalno zlonamjerna datoteka ili program pokreće se u kontroliranom okruženju, kao što je *sandbox* ili virtualno okruženje. Time se postiže izolacija procesa od ostatka sustava ili računala i sprječava širenje te oštećivanje stvarnog sustava. Tijekom izvršavanja, prate se aktivnosti programa koje uključuju mrežne aktivnosti, sistemske pozive, ponašanje datoteke i procesorskih registara, utjecaj na memoriju, interakcije programa s korisnicima, pokušaji eskalacije prava i druge operacije. Svi prikupljeni podaci tijekom izvršavanja koriste se za analiziranje te utvrđivanje potencijalnih znakova zlonamjernog ponašanja. Ukoliko se utvrde zlonamjerne aktivnosti nakon analiziranja

ovih podataka, može se izolirati i proučavati širenje štete, kako bi se bolje razumjele potencijalne posljedice i razumjele taktike napadača.

Dinamička se analiza često koristi kod iznimno kompleksnih primjeraka zloćudnog koda ili u slučaju nedostatka vremena za temeljitu statičku analizu, kako bi se u što kraćem vremenu prikupili indikatori bitni za analizu sigurnosnog incidenta.

3.6.1. Izolacija procesa

Pojam procesa podrazumijeva računalni program ili instancu programa u izvođenju. Ovaj proces EDR sustavi mogu izolirati prilikom dinamičke analize, u svrhu kontroliranog nadziranja tijekom izvršavanja [32]. Izolacija procesa predstavlja ograničavanje procesu da pristupa i širi se na razne dijelove sustava, kao što su računalni resursi, mrežni segment te dijelovi operacijskog sustava. Primjerice, u *Windows* okruženju postoji praktična implementacija izolacije procesa pod nazivom *AppContainer*. On sprječava korištenje korisničkih pristupnih podataka za pristup resursima ili prijavu u druga okruženja. Isto tako, stvara i identifikator koji koristi kombinirane identitete korisnika i aplikacije, tako da su pristupni podaci jedinstveni za svaki par korisnika i aplikacije te aplikacija ne može oponašati korisnika. Uz to, mogu izolirati aplikaciju od resursa na uređaju, kao što su pasivni senzori (kamera, mikrofon, GPS). Moguće je i kontrolirati pristup datotekama i registru gdje se određuje pristup za čitanje i pisanje korisnicima. Pristup za čitanje je većinom manje ograničen, a aplikacija uvijek ima pristup datotekama u memoriji stvorenim posebno za određeni kontejner.

3.6.2. Minifilteri

Filtrirajući upravljački programi, poznati i kao minifilteri, predstavljaju ključnu ulogu u arhitekturi *Windows* operacijskog sustava, posebice *Windows* I/O podsustava [33]. Ova komponenta omogućuje dodavanje dodatne funkcionalnosti te kontrolu nad uređajima i datotekama bez potrebe za izmjenama u upravljačkom programu tog uređaja. Na primjer, ovaj program može se koristiti za nadzor promjena u datotekama, praćenje i sprečavanje

prisutnosti zlonamjernih programa te šifriranje i dešifriranje podataka prilikom pristupa datotekama.

Filtar datotečnog sustava ključna je komponenta za poboljšanje sigurnosti i funkcionalnosti računalnih sustava koji koriste *Windows*. Zato se obično koristi u velikom rasponu svrha, uključujući antivirusno skeniranje i skeniranje zlоćudnog koda, praćenje i upravljanje programskim licencama, revizije i praćenje promjena na datotekama i drugih. Obzirom da vide koje su datoteke stvorene i zapisane, minifilteri često igraju ključne uloge u EDR sustavima kao podsustavi za nadzor radnji nad datotečnim sustavom. Dodatno, budući da se pričvršćuju na postojeće I/O uređaje, minifilteri također mogu „presresti" maliciozne programe pri pokretanju te poslati na analizu EDR sustavu prije nego se maliciozni kod počne izvršavati.

3.6.3. Praćenje događaja za Windows

Operacijski sustav *Windows* kao izvor za dnevničke zapise (eng. *event logs*) koristi sustav poznat kao *Event Tracing for Windows*[8] [34]. ETW je sustav koji operacijskom sustavu dozvoljava da centralizirano prikuplja podatke iz velike količine izvora i iz niskih razina OS-a (*ring0*, odnosno jezgra/*kernel*). Programi koji prikupljaju i šalju podatke na ETW sustav nazivaju se pružatelji (eng. *providers*) – ovo su jednostavne aplikacije napisane uz pomoć *Windows* API sustava koje zapise prikupljaju te prosljeđuju na ETW podsustave koji su na te zapise pretplaćeni – tzv. potrošači (eng. *consumers*).

Zbog mogućnosti pristupanja iznimno detaljnim i dobro skrivenim informacijama o ponašanju operacijskog sustava, ETW sustav poznat kao ETW *Threat Intelligence*[9], predstavlja važnu komponentu modernih EDR, odnosno XDR rješenja [35]. ETW-TI je ETW pružatelj kojeg je moguće instalirati jedino programom potpisanim specifičnim *Microsoft* digitalnim certifikatom (*Early Launch EKU* certifikat) kako bi se ograničila njegova uporaba te EDR rješenjima može pružiti drastično bolju vidljivost u unutrašnjost OS-a.

[8] U nastavku ETW ili ETW sustav

[9] U nastavku ETW-TI

3.6.4. DLL hooking

DLL hooking je tehnika koja uključuje presretanje i mijenjanje ponašanja funkcija koje programi ili procesi pozivaju iz DLL-ova [36] [37]. Cilj ove tehnike je postavljanje „kuke“ koja se nalazi između programa i DLL-ova koje on poziva. Kuka u ovom slučaju predstavlja dio koda kojeg autori EDR sustava mogu ubaciti u DLL-ove kako bi nadzirali pozive osjetljivih funkcija te na temelju toga mogli zaključiti koji je proces pozvao koju funkciju, što im tada dozvoljava i zaustavljanje izvođenja takvog procesa.

3.6.5. Ponašajna analiza strojnim učenjem

Analiza koja uključuje analiziranje uzoraka, ponašanja te trendova programa i korisnika poznata je kao ponašajna analiza (eng. *behavioral analysis*) [38]. Fokusirana je na kontinuirani nadzor i analizu ponašanja krajnjih točaka u stvarnom vremenu na temelju čega stvara profil koji sadrži informacije o navikama korisnika. Ukoliko dođe do promjene ili odskakanja od uobičajenih uzoraka i navika korisnika, ponašajna analiza to prepoznaje te šalje upozorenja o potencijalnoj prijetnji. Uloga ove analize je proučavanje i razumijevanje uzoraka korištenjem algoritama strojnog učenja za dubinsku analizu uzoraka, s ciljem prepoznavanja i brzog odgovora na potencijalne prijetnje. Oslanja se na nekoliko ključnih komponenti, uključujući prepoznavanje neobičnih ponašanja, modeliranja predviđanjem i prepoznavanje uzoraka. Važno je istaknuti da zbog svojih karakteristika ponašajna analiza strojnim učenjem omogućuje jednostavno prepoznavanje kako poznatih, tako i novih prijetnji informacijskom sustavu.

4. Obfuskacija programa

4.1. Obfuskacija izvorišnog/izvršnog koda

4.1.1. Slojevito učitavanje (Loaders)

Tehnika namijenjena implementaciji i izvršenju drugog zlućudnog koda na ciljnom sustavu poznata je kao slojevito učitavanje (eng. *loaders*) [39]. To je program koji dohvaća izvršne datoteke ili zlonamjerni teret (eng. *malicious payload*) s udaljenih izvora, kao što su napadačev poslužitelj ili kompromitirana internetska stranica putem *Command&Control*(C2) komunikacije ili *peer-to-peer* mreže. Uz to, rade na temelju trojanskog konja za udaljeni pristup, uključujući tehnike prijevare te način funkcioniranja i širenja zlućudnog koda. Uloga slojevitog učitavanja predstavlja stjecanje početne infekcije sustava, u svrhu omogućavanja implementiranja zlonamjernih aktivnosti i izvršavanja zlućudnog koda. Konačni cilj *loader* programa je „zbuniti“ EDR sustave kako bi se tijekom njihovog izvođenja izgubilo dovoljno konteksta odnosno unijelo dovoljno pomutnje u lanac izvršavanja što bi trebalo uzrokovati promašenu detekciju sustava. Primjer *loader* lanca su često maliciozni *Office* makroi pisani u *Visual Basic* jeziku koji preuzimaju sljedeću fazu izvršavanja u *Jscript* jeziku koji zatim preuzima sljedeću fazu pisanu u *PowerShell*-u i slično.

4.1.2. Manipulacija tekstom izvršnog koda

Tehnika namjernog izmjenjivanja teksta izvršnog koda predstavlja manipulaciju tekstom izvršnog koda, poznata je kao i obfuskacija izvršnog koda [1]. Uloga ove tehnike je otežavanje razumijevanja, analiziranja te mogućnosti reverznog inženjeringa izvršnog koda, pritom održavajući njegovu funkcionalnost. Primjenjuju se različite izmjene, uključujući promjena naziva varijabli i funkcija kako bi postali nejasni, šifriranje *stringova* i konstanti kako bi se otežalo izdvajanje važnih informacija iz koda te promjena redoslijeda koda kako bi se poremetio logički redoslijed koda. Iako ova tehnika uvelike doprinosi informacijskoj sigurnosti, važno je istaknuti da se može zloupotrijebiti i predstavljati potencijalnu prijetnju sigurnosti. Primjerice, zlućudni kod i zlonamjerni tereti

mogu se sakriti pomoću obfuskacije izvršnog koda (slično prethodnoj metodi) i tako izbjeći otkrivanje te prevariti korisnika da ih izvrše. Ova metoda se često kombinira s *loader* programima kako bi se povećala vjerojatnost za izbjegavanje antivirusnog rješenja.

4.1.3. Ubacivanje mrtvog koda

Jedna od jednostavnijih tehnika obfuskacije koda je ubacivanje mrtvog koda (eng. *dead code insertion*). Ova tehnika sastoji se od umetanja nepotrebnog ili nefunkcionalnog koda u izvorni kod, uključujući nepotrebne varijable, petlje ili uvjete. Cilj ove tehnike je otežati analizu i razumijevanje koda, bez utjecaja na njegovu osnovnu funkcionalnost. Uz to, ukoliko se koristi u legitimne svrhe, pruža mjeru zaštite intelektualnog vlasništva, tako što otežava kopiranje ili krađu koda, pronalaženje sigurnosnih ranjivosti te reverznog inženjeringa.

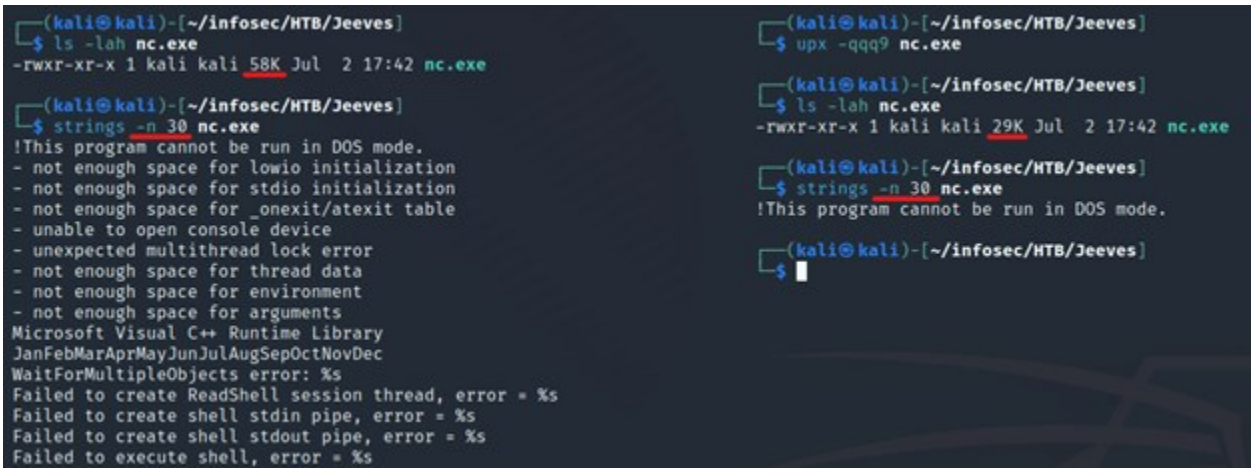
4.2. Obfuskacija izvršnih datoteka

4.2.1. Packer

Packer je alat koji pakira kod izvršne datoteke, podatke i ponekad resurse te također sadrži kod za raspakiranje programa [40]. *Packer*-i mogu pomoći autorima zlonamjernog softvera da sakriju zlonamjerni sadržaj ili kod iza sloja kompresije. Kod se raspakira i izvršava tek nakon što se zlonamjerni softver pokrene, što pomaže autorima zlonamjernog softvera da zaobiđu detekcije temeljene na statičkim indikatorima. *Packer*-i su programi koji se uglavnom koriste za komprimiranje binarne izvršne datoteke čime se smanjuje njihova ukupna veličina te nisu primarno stvoreni za skrivanje svojstava i sami po sebi nisu zlonamjerni. Poznatiji *Packer*-i su *UPX*, *ASPack*, *the enigma protector*, *MPRESS* i *Exe packer*.

U svrhu približeg objašnjavanja principa rada *Packer*-a, na Kali Linux operacijskom sustavu uspoređena je veličina i čitljivi znakovni nizovi duži od 30 ili više znakova prije i

nakon pakiranja *UPX Packer*-om. Na sljedećoj slici (Slika 3.) lako se može vidjeti da se datoteka nc.exe smanjila za 50% te su iz iste datoteke „uklonjeni“ gotovo svi znakovni nizovi, osim onoga (nužno) prisutnog u PE zaglavlju.



Slika 3. usporedbe veličine i čitljivosti PE datoteke prije i nakon pakiranja UPX-om

4.2.2. Crypter

Crypter predstavlja alat ili program koji se koristi za šifriranje, obfuskaciju i skrivanje zlućudnog koda [41]. Njegova glavna uloga je skrivanje zlućudnog koda i izbjegavanje njegovog otkrivanja, čime se otežava njihova identifikacija i detekcija (od strane sigurnosnih programa).

Crypter može sadržavati izrezak koda (eng. *stub*), što predstavlja mali dio koda namijenjen dešifriranju, izvršavanju ili pokretanju skrivenog zlućudnog tereta. Ovo omogućava zlućudnom kodu da ostane neotkriven do njegovog pokretanja te je otežano otkrivanje statičkom analizom. Isto tako, napadačima omogućava izvršavanje različitih tereta jednim *stub*-om te prilagođavanje određenih karakteristika, kao što su kako i kada će se teret izvršiti. Ovisno o vrsti *stub*-a, *Crypter* se može klasificirati kao statički i polimorfni. Kod statičkih *stub*-ova, određen je *stub* koji ostaje nepromijenjen te prilikom svakog izvršavanja *Crypter*-a, isti *stub* pokreće proces dešifriranja i izvršavanja zlućudnog koda. Zbog toga mogu biti manje efektivni u izbjegavanju otkrivanja, pogotovo statičkom analizom otkrivanja. Za razliku od toga, polimorfni *stub* ima sposobnost mijenjanja. Prilikom svakog izvršavanja *Crypter*-a, stvara se jedinstveni *stub*, čime se

uvelike olakšava izbjegavanje otkrivanja i analiziranja različitim metodama. Uz to, ovakvi *Crypter*-i sadrže i tehnike šifriranja pa se mijenja i šifrirani korisni teret pri svakom izvršavanju. Često se *Crypter* koristi u zlonamjerne svrhe u cilju izbjegavanja različitih metoda sigurnosnih mjera i izvršavanja zlonamjernih aktivnosti, važno je istaknuti da se koriste i u legitimne svrhe. To znači da se kroz tehnike šifriranja i obfuskacije *Crypter* može doprinijeti sigurnosti sustava, integritetu i zaštiti podataka.

Ovaj alat sličan je *Packer*-u, no *Crypter* prvenstveno radi obfuskaciju te šifriranje i skrivanje tereta unutar izvršne datoteke, dok *Packer* mijenja strukturu cijele izvršne datoteke te radi pakiranje i modifikaciju izvršne datoteke. Za razliku od *Packer*-a, *Crypter*-i se prvenstveno koriste u zlonamjerne svrhe.

4.2.3. Polimorfični enkoderi

Polimorfični enkoderi su algoritmi koji dinamički generiraju šifrirane izvršne datoteke, slično *Crypter*-ima [42]. Razlika između polimorfičnog algoritma i *Crypter*-a je u činjenici da se razne varijable i opcije koje polimorfični algoritam koristi prilikom šifriranja razlikuju prilikom svakog generiranja nove izvršne datoteke. Navedene varijable odnosno opcije mogu se odnositi na namjerno ubacivanje mrtvih dijelova koda, korištenje različitih ključeva za šifriranje datoteke, dinamičku promjenu strojnih naredbi i slično. Jedan od najpoznatijih polimorfičnih enkodera je *Shikata-Ga-Nai*[10].

SGN predstavlja kompleksni koder koji se temelji na konceptima polimorfizma, što znači da će se svaka datoteka razlikovati od sljedeće i to se postiže različitim tehnikama. Te tehnike uključuju dinamičku zamjenu instrukcija, dinamičko sređivanje blokova, nasumičnu izmjenu registara, nasumično sređivanje instrukcija, umetanje nefunkcionalnog koda, korištenje nasumičnog ključa i nasumična odredba razmaka između instrukcija. Isto tako, temelji se i na XOR aditivnim povratnim informacijama. Ključna komponenta XOR aditivne povratne informacije je da algoritam izvodi XOR buduće instrukcije putem nasumičnog ključa, a zatim tu instrukciju dodaje ključu kako bi se ponovno koristila za kodiranje sljedeće instrukcije. Dešifriranje datoteke je proces koji

[10] U nastavku SGN

slijedi korake unatrag kako bi se došlo do izvornog sadržaja. Detektiranje SGN kodiranih datoteka je izazovno, pogotovo ako se uvelike oslanja na statičke indikatore (izuzev YARA pravila). Zbog toga se većina platformi za otkrivanje oslanja na otkrivanje putem indikatora ponašanja i *sandbox*-a.

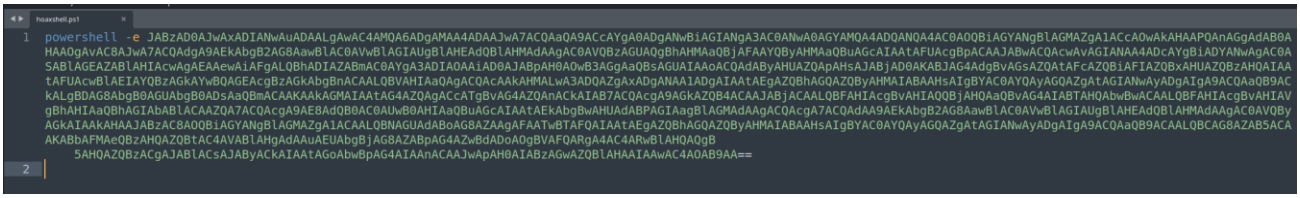
5. Studijski slučaj

Kao što je spomenuto na početku rada, kako bismo procijenili kvalitetu pojedinih metoda obfuskacije, iskorišteno je nekoliko *Command&Control* (C2) radnih okvira. Ukratko, C2 radni okvir podrazumijeva okruženje u kojem postoje 3 glavna elementa: napadački poslužitelj (eng. *Teamserver*), napadački klijent (eng. *client*) te agent (eng. *beacon*, *agent*). Poslužitelj je zaslužan za koordinaciju napadačkih klijenata i agenata koji su pokrenuti na računalima žrtve. Klijent je program koji napadačima omogućuje da generiraju agente, upravljaju zaraženim računalima, modificiraju ponašanje živućih odnosno aktivnih agenata, izvršavaju naredbe u žrtvinom okruženju i slično. Kako bi se kroz klijent moglo upravljati žrtvinim računalom, potrebno je generirati agenta koji će se pokrenuti na žrtvinom računalu i uspostaviti povratnu vezu (eng. *callback*) prema napadačkoj infrastrukturi. Svi radni okviri koji će se koristiti u ovoj studiji su programi otvorenog koda te su to redom *Hoaxshell* [43], *Metasploit* [44], *Powershell Empire* [45] i *Havoc C2* [46]. Uspješnost metoda obfuskacije testirat će se korištenjem javnog servisa *VirusTotal* kako bi se agentske datoteke jednostavno predale većem broju EDR proizvoda na provjeru.

5.1. Hoaxshell

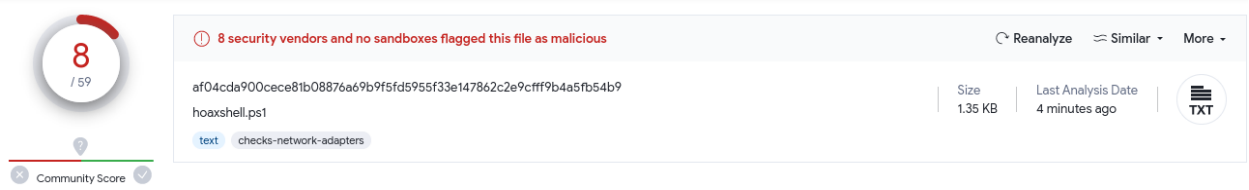
Počevši s najjednostavnijim, *Hoaxshell* je radni okvir koji za komunikaciju prema C2 poslužitelju koristi HTTP, naredbe prima i vraća sadržaj u tijelu HTTP zahtjeva, a agenta generira u obliku jednostavne, donekle obfuscirane *Powershell* skripte. Ovaj C2 radni okvir pruža iznimno ograničene dodatne funkcionalnosti te nema sposobnosti aktivnog zaobilaženja EDR sustava.

Na sljedećoj je slici (Slika 4.) vidljiv agentski kod kojeg je generirao Hoaxshell radni okvir – kako bi se žrtva povezala s C2 poslužiteljem, potrebno je pokrenuti prikazanu *Powershell* skriptu na njenom računalu.



Slika 4. Agentski kod generiran Hoaxshell radnim okvirom

Unatoč svojoj jednostavnosti, samo je 8 EDR sustava detektiralo *Hoaxshell*-ov agentski kod kao maliciozan, što se može vidjeti i na sljedećoj slici (Slika 5.). Pretpostavka je da upravo zbog svoje jednostavnosti i donekle normalnog ponašanja unutar sustava (slanje i primanje HTTP zahtjeva) EDR sustavi ne smatraju ovaj C2 agent malicioznim.



Slika 5. VirusTotal detekcije za Hoaxshell agent

5.2. Metasploit

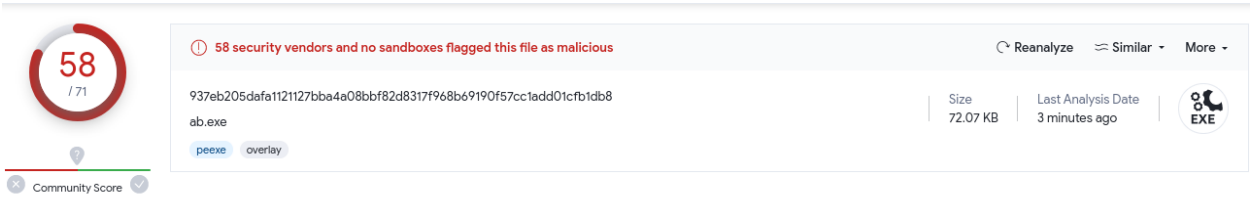
Metasploit je moćan radni okvir često korišten za penetracijska testiranja i općenito ispitivanje ranjivosti računalnih sustava. Sadrži skener ranjivosti, module za iskorištavanje raznih ranjivosti te omogućuje generiranje prilagođenih agenata pomoću alata *MSFVenom*. Funkcionalno najmoćniji agent unutar *Metasploit* okvira je *Meterpreter*, koji pruža potpunu kontrolu nad ciljanim sustavima, uključujući pristup datotekama, naredbeni redak, snimanje zaslona, automatiziranu eskalaciju privilegija, izvlačenje korisničkih vjerodajnica i druge. MSF je od posebnog interesa za ovaj rad obzirom da je jedna od mogućnosti *MSFVenom*-a šifriranje odnosno enkodiranje agenata raznim algoritmima, poput običnog binarnog XOR-a ili *Shikata-Ga-Nai* polimorfičnog enkodera. Za ovo testiranje generirana su 4 agenta (naredbe za njihovo kreiranje prikazane su na Slici 6.): obični *Meterpreter*, XOR-šifrirani *Meterpreter*, te 2 *Meterpreter* agenta dobivena

enkodiranjem pomoću *Shikata-Ga-Nai* enkodera – prvi samo jednim prolazom, a drugi koristeći 100 prolaza SGN-a nad izvršnom datotekom.

```
┌──(kali㉿kali)-[~/Desktop]
└─$ sha256sum *.exe
205a8e668637d7340af4ee96d2fa9ee8292b06d9776d413db78bfac91bcaf0f2  havoc_demon_fully_featured.exe
937eb205dafa1121127bba4a08bbf82d8317f968b69190f57cc1add01cfb1db8  meterpreter_normal.exe
34eec61c0a834aee81b4971c95db3e9e931be109e33dd4adda68e5b47c8dbcb4  meterpreter_shikata_100_passes.exe
258e823eaa35a63fd78d0d66ea9e0c63385acfecf7765f88cff9558e351b39f6  meterpreter_shikata_one_pass.exe
5a90e8a5834edfc10417c9f5b134248808b6984668fc90b7301f761a2e0a08f2  meterpreter_xor.exe

┌──(kali㉿kali)-[~/Desktop]
└─$ sha256sum *.dll
0deb32c602192a8a661ce008072564daad4800cc91dc4b0069203c9ecdadc9e0  empire.dll

┌──(kali㉿kali)-[~/Desktop]
└─$ sha256sum *.ps1
af04cda900cece81b08876a69b9f5fd5955f33e147862c2e9cfff9b4a5fb54b9  hoaxshell.ps1

┌──(kali㉿kali)-[~/Desktop]
└─$ ll | egrep -i "dll|ps1|exe"
-rw-r--r--  1 kali kali 665088 Sep 19 22:29 empire.dll
-rw-r--r--  1 kali kali  93184 Sep 19 22:46 havoc_demon_fully_featured.exe
-rw-r--r--  1 kali kali   1383 Sep 19 22:18 hoaxshell.ps1
-rw-r--r--  1 kali kali  73802 Sep 19 22:13 meterpreter_normal.exe
-rw-r--r--  1 kali kali  73802 Sep 19 22:14 meterpreter_shikata_100_passes.exe
-rw-r--r--  1 kali kali  73802 Sep 19 22:13 meterpreter_shikata_one_pass.exe
-rw-r--r--  1 kali kali   7168 Sep 19 22:16 meterpreter_xor.exe

┌──(kali㉿kali)-[~/Desktop]
└─$
```

Slika 6. Naredbe za generiranje Meterpreter agenata

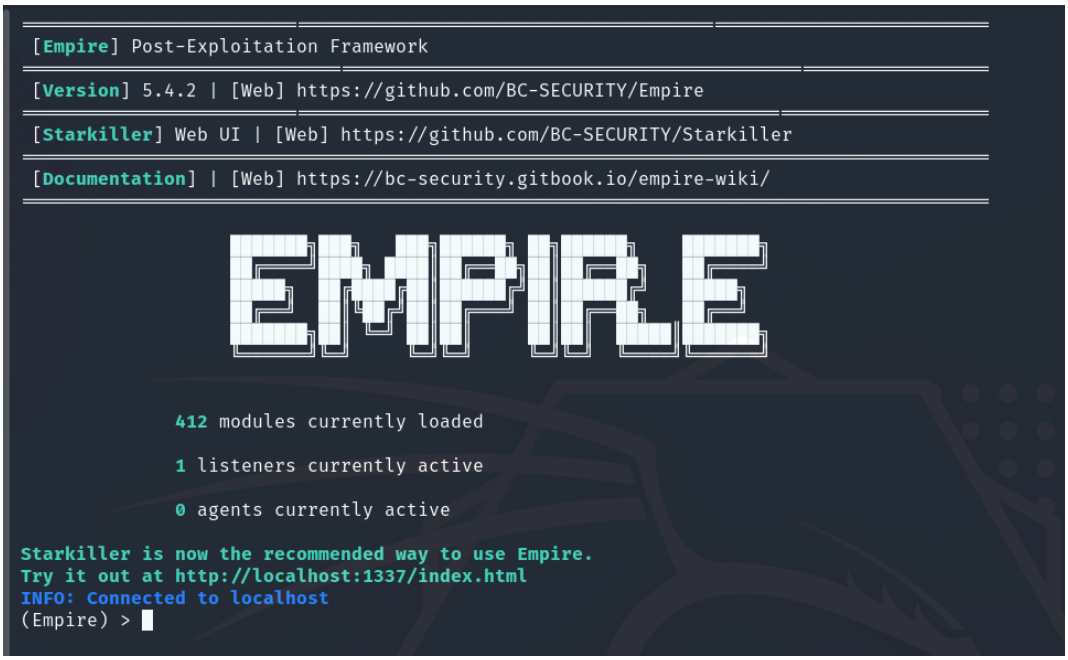
Iako se možda čini kontraintuitivno, najmanje detekcija generirao je XOR-ani *Meterpreter* (54), dok je ostale 3 verzije *Meterpreter*-a detektiralo čak 58 EDR sustava (prikazano na Slici 7.). Unatoč SGN enkodiranju, EDR sustavima je (kao što je ranije spomenuto u poglavlju 4.2.3) jednostavno prepoznati datoteke enkodirane SGN-om jer ovaj algoritam ostavlja specifičan potpis u obliku asemblerskih instrukcija koje je jednostavno prepoznati temeljem YARA pravila.



Slika 7. SGN-enkodirani i obični Meterpreter agenti na VirusTotal-u

5.3. Powershell Empire

Powershell Empire C2 radni okvir ime je dobio po činjenici da naredbe na *Windows* sustavima izvršava u kontekstu *Powershell*-a te omogućuje napadačima izvođenje naprednih napada na *Windows* sustavima. *Empire* nudi široku paletu naprednih funkcionalnosti, uključujući vlastite, prikrivene tehnike za izbjegavanje EDR alata koje među ostalim uključuju dinamičko učitavanje (*loaders*) i obfuskaciju znakovnih nizova. Također podržava ekstenzibilnost putem raznih modula koji se mogu prilagoditi za specifične potrebe napadača.



Slika 8. Powershell Empire radni okvir

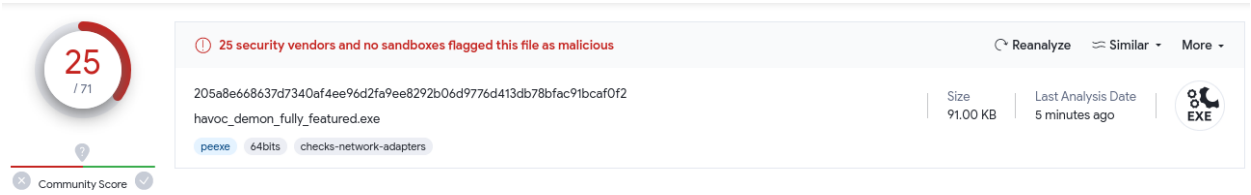
Unatoč relativno velikom broju tehnika za izbjegavanje detekcije, 32 EDR sustava su na *VirusTotal*-u uspjela detektirati Empire agenta u DLL formatu. Pretpostavka je da je ovo kombinacija činjenice da se radi o dobro poznatom radnom okviru te da generirani DLL nije drastično modificiran.

5.4. Havoc C2

Havoc C2 radni okvir je razvijen s naglaskom na naprednim funkcionalnostima i izbjegavanju detekcije od strane EDR sustava. *Havoc* koristi niz tehnika za izbjegavanje detekcije, uključujući korištenje vlastitih *.NET* modula za obfuskaciju, različite tehnike za kodiranje izvršnog koda, detekciju virtualnih okruženja i *sandbox*-a kako bi izbjegao dinamičku analizu, te brojne druge metode koje izlaze van okvira ovog rada. Ovaj radni okvir ima prilagodljivu arhitekturu (eng. *malleable C2*) koja omogućuje korisnicima stvaranje vlastitih modula za dodatne funkcionalnosti i integraciju različitih tehnika za napredno izbjegavanje detekcije. To je prikazano na sljedećoj slici (Slika 9.) unutar *Havoc C2* potprograma za generiranje agenta.

Slika 9. Havoc C2 potprogram za generiranje agenta

Unatoč brojnim i iznimno naprednim metodama za izbjegavanje EDR sustava, statičkom analizom *Havoc*-ov agent je detektiralo 25 EDR sustava (prikazano na Slici 10.). Naime, većina *Havoc*-ovih naprednijih metoda za obfusciranje i izbjegavanje detekcije odnosi se na agent u izvođenju. Kod C2 agenata koji se koriste u stvarnim napadima naglasak je upravo na dinamičkim tehnikama obzirom da se ovakvi agenti često ne pokreću ručno, primjerice dvoklikom korisnika na izvršnu datoteku agenta, već se dinamički učitavaju u memoriju. Tako pazeći na velik broj zamki koje EDR sustavi postavljaju, a koje daleko premašuju doseg i tehničke detalje prikazane u ovom radu.



Slika 10. Havoc C2 agent na provjeri VirusTotal-a

6. Zaključak

Ovaj rad istražio je i testirao velik broj jednostavnijih i naprednijih tehnika obfuskacije i metoda za izbjegavanje EDR, odnosno XDR sustava. Isto tako, prikazao je i nekoliko načina na koji razvojni programeri takvih sustava mogu detektirati maliciozne programe prije i tijekom njihova izvršavanja. Iako je konačnim testiranjem na samom kraju rada primijećeno da nijedan od korištenih C2 radnih okvira ne kreira agentsku datoteku koju nijedan EDR sustav neće statički detektirati, ovakvo ponašanje je donekle i očekivano. Naime, prilikom kreiranja testnih agenata nisu korištene nikakve dodatne niti vlastite tehnike za obfuskaciju izvršnih datoteka, već su agenti provjeravani u obliku kojeg je generirao C2 okvir. Kako su ovi radni okviri otvorenog koda, moguće je pretpostaviti da ih velik broj EDR autora koristi kao inspiraciju za pisanje YARA pravila ili modifikaciju vlastitih postojećih sustava kako bi prepoznavali ovakve datoteke na temelju njihovog potpisa. U *red team* testiranjima u stvarnom svijetu, od operatera se očekuje da C2 okvire koriste tek kao pomoćni mehanizam, a da samostalno „pripreme teren“ za izvršavanje ovakvih agenata. Priprema podrazumijeva inicijalni ulazak u sustav, uspostavljanje jednostavne inicijalne komunikacije s vlastitom infrastrukturom, dubinsko istraživanje i prepoznavanje sustava kojeg napadaju, pisanje vlastitog malicioznog koda i slično. Postojeći C2 agenti se najčešće u priču ubacuju tek pri samom kraju, kada su operateri koristeći vlastite metode i tehnike koje nisu nužno dobro poznate EDR autorima uspostavili određenu razinu kontrole nad sustavom. Konačno, iako smo spomenuli da statička obfuskacija nije nužno primarni cilj napadačima, odnosno pentesterima, svakako je korisno postaviti pitanje može li umjetna inteligencija pomoći pri generiranju raznovrsnih algoritama, enkodera, ili cjelokupnih izvršnih datoteka dinamički, s različitim parametrima, kako bi se jednostavna YARA pravila ili slični potpisi u budućnosti efektivnije zaobilazili.

7. Sažetak

Usporedba metoda obfuskacije koda u svrhu izbjegavanja EDR i XDR zaštite

Sažetak

Ovaj rad iznio je sažeti pregled tehnika obfuskacije i metoda izbjegavanja EDR, odnosno XDR zaštite. Isto tako, objašnjeni su pojmovi statičke i dinamičke analize datoteka, kao i osnovni mehanizmi za detekciju zlućudnih procesa unutar *Windows* okruženja. Na kraju rada proveden je studijski slučaj koji uspoređuje četiri *Command&Control* radna okvira (*Hoaxshell*, *Metasploit*, *Powershell Empire*, *Havoc C2*) uz pomoć *VirusTotal* alata kako bi se na jednostavan način uvidjela efikasnost statičkog i dinamičkog izbjegavanja EDR rješenja. Zaključak rada je da, iako su EDR sustavi relativno dobri u detektiranju zlućudnog koda na temelju statičke analize, sposobnost zaobilaženja takvih sustava većinom ovisi o sposobnosti operatera koji kreira vlastite programe te se ne oslanja na gotova rješenja kako bi izvršavao napade, bilo to u kontekstu kibernetičkog kriminala ili etičkog hakiranja. Konačno, rad iznosi hipotezu da problem detekcije zlućudnog koda odnosno zaobilaženja obrambenih sustava nikada neće biti do kraja „riješen“ problem te postavlja pitanje korištenja umjetne inteligencije za potpomognuto generiranje obfusciranog koda.

Ključne riječi: obfuskacija, EDR, XDR, analiza zlućudnog koda, detekcija zlućudnog koda, YARA, Sigma, Comand and Control, Portable Executable

8. Summary

Comparison of code obfuscation methods for the purpose of avoiding EDR and XDR protection

Summary

This thesis provides a concise overview of obfuscation techniques and methods for evading EDR (*Endpoint Detection and Response*) or XDR (*Extended Detection and Response*) protection. It also explains the concepts of static and dynamic file analysis, as well as basic mechanisms for detecting malicious processes within the Windows environment. At the end of the thesis, a case study was conducted comparing four *Command & Control* frameworks (*Hoaxshell*, *Metasploit*, *Powershell Empire*, *Havoc C2*) using the *VirusTotal* tool to easily assess the effectiveness of static and dynamic evasion of EDR solutions. The conclusion of the paper is that, although EDR systems are relatively good at detecting malicious code based on static analysis, the ability to bypass such systems largely depends on the operator's capability to create custom programs and does not rely on ready-made solutions for conducting attacks, whether in the context of cybercrime or ethical hacking. Finally, the paper poses the hypothesis that the problem of detecting malicious code or bypassing defensive systems will never be a fully "solved" issue and raises the question of using artificial intelligence for assisted code obfuscation generation.

Keywords: obfuscation, EDR, XDR, malware analysis, malware detection, YARA, Sigma, Command and Control, Portable Executable

9. Literatura

[1] ReversingLabs, "Code obfuscation," -. [Online]. Available: https://www.reversinglabs.com/glossary/code-obfuscation. [Accessed 5 9 2023.].

[2] A. T. Tunggal, "22 Types of Malware and How to Recognize Them in 2023," 6 4 2023.. [Online]. Available: https://www.upguard.com/blog/types-of-malware#toc-1. [Accessed 17 8 2023.].

[3] proofpoint, "What Is Malware?," -. [Online]. Available: https://www.proofpoint.com/us/threat-reference/malware. [Accessed 17 8 2023.].

[4] Cyfirma, "Malware detection: evasion tecniques," 13 9 2023.. [Online]. Available: https://www.cyfirma.com/outofband/malware-detection-evasion-techniques/. [Accessed 17 8 2023.].

[5] J. Von Neumman, Theory of Self-Reproducing Automata, 1966..

[6] J. S. Cunningham, "Interview with Ray Tomlinson on Creeper/Reaper," 6 4 2016.. [Online]. Available: https://www.osnews.com/story/29157/interview-with-ray-tomlinson-on-creeperreaper/. [Accessed 17 8 2023.].

[7] t. V. Encyclopedia, "Creeper," -. [Online]. Available: http://virus.wikidot.com/creeper. [Accessed 17 8 2023.].

[8] D. Snyder, "The very first viruses: Creeper, Wabbit and Brain.," 30 5 2010.. [Online]. Available: https://infocarnivore.com/the-very-first-viruses-creeper-wabbit-and-brain/. [Accessed 17 8 2023.].

[9] Fortinet, "Trojan Horse Virus," -. [Online]. Available: https://www.fortinet.com/resources/cyberglossary/trojan-horse-virus. [Accessed 21 8 2023.].

[10] Malwarebytes, "Trojan horse – Virus or malware?," -. [Online]. Available: https://www.malwarebytes.com/trojan. [Accessed 21 8 2023.].

[11] Helen, "[Tutorial] What's Remote Access Trojan & How to Detect/Remove It?," 9 2 2023.. [Online]. Available: https://www.minitool.com/backup-tips/remote-access-trojan.html. [Accessed 21 8 2023.].

[12] K. Yasar, "RAT (remote access Trojan)," 2022.. [Online]. Available: https://www.techtarget.com/searchsecurity/definition/RAT-remote-access-Trojan. [Accessed 21 8 2023.].

[13] kaspersky, "What is Spyware?," 2018.. [Online]. Available: https://www.kaspersky.com/resource-center/threats/spyware. [Accessed 21 8 2023.].

[14] D. Manky, "How to Prevent Evolving Ransomware Attacks," 2 5 2021.. [Online]. Available: https://www.fortinet.com/blog/industry-trends/protecting-against-evolving-ransomware-attack-trends?utm_source=blog&utm_campaign=ransomware. [Accessed 21 8 2023.].

[15] P. D. L. H. C. Chen, "A Study on Advanced Persistent Threats," 2014.. [Online]. Available: https://doi.org/10.1007/978-3-662-44885-4_5. [Accessed 22 8 2023.].

[16] TryHackMe, "Common Attacks," 24 2 2022.. [Online]. Available: https://tryhackme.com/room/commonattacks. [Accessed 10 9 2023.].

[17] SecurinInc, "Securin's AI-Based Insights into APT Groups and Their Arsenal," 14 4 2022.. [Online]. Available: https://www.securin.io/prevent-falling-victim-to-apt-groups-using-securins-ai-based-vulnerability-and-threat-intelligence/. [Accessed 22 8 2023.].

[18] K. Baker, "Malware Analysis," 17 4 2023.. [Online]. Available: https://www.crowdstrike.com/cybersecurity-101/malware/malware-analysis/. [Accessed 22 8 2023.].

[19] K. Baker, "Malware Analysis," 17 4 2023.. [Online]. Available: https://www.crowdstrike.com/cybersecurity-101/malware/malware-analysis/.

[20] S. Jones, "What Is Dynamic Malware Analysis?," 21 3 2023.. [Online]. Available: https://www.bitdefender.com/blog/businessinsights/what-is-dynamic-malware-analysis/. [Accessed 10 9 2023.].

[21] D. Richards, "What Is A Signature-Based Antivirus?," 22 9 2022.. [Online]. Available: https://www.infoexchangeja.com/blog/data-security/what-is-signature-based-antivirus/. [Accessed 22 8 2023.].

[22] Trellix, "What Is Endpoint Detection and Response (EDR)?," -. [Online]. Available: https://www.trellix.com/en-us/security-awareness/endpoint/what-is-endpoint-detection-and-response.html#definition. [Accessed 23. 8 2023.].

[23] A. Aarness, "What is Endpoint Detection and Response (EDR)?," 6 2 2023.. [Online]. Available: https://www.crowdstrike.com/cybersecurity-101/endpoint-security/endpoint-detection-and-response-edr/. [Accessed 23 8 2023.].

[24] vmware, "What is Extended Detection and Response (XDR)?," -. [Online]. Available: https://www.vmware.com/topics/glossary/content/xdr-extended-detection-and-response.html. [Accessed 23. 8 2023.].

[25] S. Daulaguphu, "A Comprehensive Guide To PE Structure, The Layman's Way," 15 8 2022.. [Online]. Available: https://tech-zealots.com/malware-analysis/pe-portable-executable-structure-malware-analysis-part-2/. [Accessed 24 8 2023.].

[26] Microsoft, "PE Format," 24 3 2023.. [Online]. Available: https://learn.microsoft.com/en-us/windows/win32/debug/pe-format. [Accessed 10 9 2023.].

[27] SentinelOne, "What Is A Malware File Signature (And How Does It Work)?," 12 8 2021.. [Online]. Available: https://www.sentinelone.com/blog/what-is-a-malware-file-signature-and-how-does-it-work/. [Accessed 24 8 2023.].

[28] R. Poonia, "Unpacking the Malwares," 14 3 2022.. [Online]. Available: https://hackerhood.redhotcyber.com/tutorial-di-malware-analysis-2/. [Accessed 24 8 2023.].

[29] Packetlabs, "What Are YARA Rules?," 20 4 2023.. [Online]. Available: https://www.packetlabs.net/posts/what-are-yara-rules/. [Accessed 28 8 2023.].

[30] T. G. Team, "The Ultimate Guide to Sigma Rules," 28 3 2023.. [Online]. Available: https://graylog.org/post/the-ultimate-guide-to-sigma-rules/. [Accessed 24 8 2023.].

[31] VMRAY, "Dynamic Analysis," -. [Online]. Available: https://www.vmray.com/glossary/dynamic-analysis/. [Accessed 23 8 2023.].

[32] M. S. K. W. S. A. A. Satran, "AppContainer isolation," 20 7 2023.. [Online]. Available: https://learn.microsoft.com/en-us/windows/win32/secauthz/appcontainer-isolation. [Accessed 27 8 2023.].

[33] O. Open Systems Resources, "An Introduction to Standard and Isolation Minifilters," 10 8 2020.. [Online]. Available: https://www.osr.com/nt-insider/2017-issue2/introduction-standard-isolation-minifilters/. [Accessed 28 8 2023.].

[34] R. T. Notes, "ETW: Event Tracing for Windows 101," 2020.. [Online]. Available: https://www.ired.team/miscellaneous-reversing-forensics/windows-kernel-internals/etw-event-tracing-for-windows-101. [Accessed 29 8 2023.].

[35] M. Cahyadi, "Introduction into Microsoft Threat Intelligence Drivers (ETW-TI)," 19 9 2022.. [Online]. Available: https://blog.maikxchd.com/introduction-into-microsoft-threat-intelligence-drivers-etw-ti. [Accessed 29 8 2023.].

[36] D. Das, "What Is DLL Hooking and How Does It Work?," 19 3 2023.. [Online]. Available: https://www.makeuseof.com/what-is-dll-hooking-and-how-does-it-work/. [Accessed 31 8 2023.].

[37] Unprotect, "DLL Unhooking," 3 7 2023.. [Online]. Available: https://unprotect.it/technique/dll-unhooking/. [Accessed 31 8 2023.].

[38] Xcitium, "Behavioral EDR," -. [Online]. Available: https://www.xcitium.com/edr-security/behavioral-edr/. [Accessed 2 9 2023.].

[39] C. Warner, "Malware Loaders & Droppers," 17 1 2023.. [Online]. Available: https://warnerchad.medium.com/malware-loaders-droppers-bca550d958aa. [Accessed 4 9 2023.].

[40] 0x00sec, "Packers - Executable Compression and Data Obfuscation," 2 7 2016.. [Online]. Available: https://0x00sec.org/t/packers-executable-compression-and-data-obfuscation/847. [Accessed 6 9 2023.].

[41] T. Micro, "Crypter," -. [Online]. Available: https://www.trendmicro.com/vinfo/us/security/definition/crypter. [Accessed 6 9 2023.].

[42] S. R. E. C. N. Miller, "Shikata Ga Nai Encoder Still Going Strong," 25 11 2022.. [Online]. Available: https://www.mandiant.com/resources/blog/shikata-ga-nai-encoder-still-going-strong. [Accessed 8 9 2023.].

[43] t3l3machus, "hoaxshell," 2023.. [Online]. Available: https://github.com/t3l3machus/hoaxshell. [Accessed 13 9 2023.].

[44] rapid7, "metasploit-framework," 2023.. [Online]. Available: https://github.com/rapid7/metasploit-framework. [Accessed 13 9 2023.].

[45] EmpireProject, "Empire," 1 8 2019.. [Online]. Available: https://github.com/EmpireProject/Empire. [Accessed 13 9 2023.].

[46] HavocFramework, "Havoc," 2023.. [Online]. Available: https://github.com/HavocFramework/Havoc. [Accessed 13 9 2023.].

[47] "VirusTotal," [Online]. Available: https://www.virustotal.com/gui/home/upload. [Accessed 13 9 2023.].

10. Popis slika