Detekcija i analiza višestupanjskog malvera

Žagar, Matija

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Sveučilišni studij računarstva

Detekcija i analiza višestupanjskog malvera

Završni rad

Matija Žagar

Osijek, 2023.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

Obrazac Z1P - Obrazac za ocjenu završnog rada na preddiplomskom sveučilišnom studiju

Osijek, 05.07.2023.

Odboru za završne i diplomske ispite

Prijedlog ocjene završnog rada na preddiplomskom sveučilišnom studiju

Ime i prezime Pristupnika:	Matija Žagar
Studij, smjer:	Računalno inženjerstvo
Mat. br. Pristupnika, godina upisa:	R4595, 27.07.2020.
OIB Pristupnika:	90526919898
Mentor:	izv. prof. dr. sc. Krešimir Grgić
Sumentor:	,
Sumentor iz tvrtke:	
Naslov završnog rada:	Detekcija i analiza višestupanjskog malvera
Znanstvena grana rada:	**Telekomunikacije i informatika (zn. polje elektrotehnika)**
Zadatak završnog rad:	Višestupanjski maliciozni softver (malver) prenosi se na odredišni uređaj kroz više stupnjeva, kako bi se time otežala ili onemogućila njegova pravodobna detekcija, te kao takav predstavlja globalno sve veću i ozbiljniju prijetnju. U završnom radu potrebno je istražiti i opisati različite vrste višestupanjskog malvera, provesti njihovu analizu, ispitati mogućnosti detekcije, te dati smjernice i preporuke za odgovarajuće preventivne mjere. Tema rezervirana za: Matija Žagar
Prijedlog ocjene završnog rada:	Izvrstan (5)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 3 bod/boda Postignuti rezultati u odnosu na složenost zadatka: 3 bod/boda Jasnoća pismenog izražavanja: 3 bod/boda Razina samostalnosti: 3 razina
Datum prijedloga ocjene od strane mentora:	05.07.2023.
Datum potvrde ocjene od strane Odbora:	12.07.2023.

Potvrda mentora o predaji konačne verzije rada:	*Mentor elektronički potpisao predaju konačne verzije.*
	Datum:

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

IZJAVA O ORIGINALNOSTI RADA

Osijek, 14.07.2023.

Ime i prezime studenta:	Matija Žagar
Studij:	Računalno inženjerstvo
Mat. br. studenta, godina upisa:	R4595, 27.07.2020.
Turnitin podudaranje [%]:	5

Ovom izjavom izjavljujem da je rad pod nazivom**: Detekcija i analiza višestupanjskog malvera**

izrađen pod vodstvom mentora izv. prof. dr. sc. Krešimir Grgić

i sumentora ,

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis studenta:

SADRŽAJ

1. UVOD

Tema ovog rada je višestupanjski ili višerazinski *malware* te pristupi njegovoj analizi i detekciji. Zloćudni programi svakim danom postaju sve napredniji i teži za otkriti. Višerazinski *malware* postao je svakidašnja pojava te je skoro pa u potpunosti zamijenio tradicionalni jednorazinski *malware*. Ova vrsta zloćudnog programa je teža za otkriti zbog svoje modularne prirode koja značajno povećava potrebno vrijeme za analizu nekog uzorka. S obzirom na to da pojedini dijelovi ovakvog napada najčešće nisu sami po sebi zloćudni, potrebno je promatrati širu sliku događaja kako bi se suzbili. Jedan od takvih programa koji je učinio veliku štetu krađom korisničkih podataka je trojanski konj Emotet. Sama definicija višerazinskog *malware*-a te opis njegovog ponašanja će biti razrađen u poglavlju 2. Zloćudni program Emotet će također biti predstavljen u tom poglavlju.

Rad je podijeljen na dva pristupa analizi i detekciji višerazinskog *malware*-a. Statički pristup opisan je u poglavlju 3. U tom poglavlju predstavljene su neke od najbitnijih metoda analize i detekcije u potpoglavljima 3.1 i 3.2. U poglavlju 3.3 razvijeno je rješenje za statičku detekciju u obliku YARA pravila.

Dinamički pristup obrađen je u poglavlju 4. U poglavljima 4.1 i 4.2 opisane su različite tehnike i metode koje stručnjaci koriste za razumijevanje zloćudnog koda prema njegovom ponašanju. Predstavljen je pregled i korištenje sigurnosnih alata poput virtualnih izoliranih okruženja (engl. *sandbox*), process monitora i drugih. Prikupljene informacije od ovih alata su korištene za razvoj Sigma pravila za detekciju Emotet *malware*-a

1.1. Zadatak završnog rada

Višestupanjski maliciozni softver (malver) prenosi se na odredišni uređaj kroz više stupnjeva, kako bi se time otežala ili onemogućila njegova pravodobna detekcija, te kao takav predstavlja globalno sve veću i ozbiljniju prijetnju. U završnom radu potrebno je istražiti i opisati različite vrste višestupanjskog malvera, provesti njihovu analizu, ispitati mogućnosti detekcije, te dati smjernice i preporuke za odgovarajuće preventivne mjere.

2. VIŠESTUPANJSKI MALWARE

U današnjem dobu, napadači kontinuirano razvijaju napredne tehnike za izbjegavanje sigurnosti, krađu financijskih podataka te intelektualnog vlasništva. Ovi napadi često uključuju više različitih smjerova napada te su analitičari konstantno suočeni s izazovom identifikacije smjerova napada, ciljeva napada, tehnika izbjegavanja i širenja te dosad neviđenim tehnikama. Ove napade često nazivamo višerazinski te oni predstavljaju veliku prijetnju tvrtkama, ali i svim osobama koje dođu u kontakt s njima [1].

Višestupanjski ili višerazinski *malware* (engl. *multi-stage malware*) predstavlja složene i sofisticirane oblike zlonamjernog softvera koji koriste različite slojeve i tehnike kako bi izbjegli otkrivanje i ostvarili svoje ciljeve. Ovaj tip malware-a je sposoban prilagoditi se okruženju, promijeniti svoje ponašanje i izvršavati niz aktivnosti koje omogućavaju prikrivanje, širenje i izvođenje napada.

2.1. Ponašanje

Višerazinski malware kombinira različite tehnike i najčešće sadrži nekoliko dijelova. Ti dijelovi mogu uključivati razne vrste malicioznih programa poput trojanskih konja, *ransomware*-a ili *spyware*-a, koji rade zajedno kako bi izbjegli detekciju i ostvarili napade. Trojanski konj je program koji se pretvara da je legitiman dok u pozadini izvršava zlćudni kod. *Ransomware* je vrsta *malware-a* koji kriptira podatke sustava te napadači najčešće traže novčanu nagradu u zamjenu za ključ za dekripciju. Spyware je svaka vrsta *malware*-a koji se trajno pričvrsti na sustav te prikuplja informacije o korisniku bez njegovog znanja te ih šalje napadaču [1, 2].

Neke od prepoznatljivih metoda koji višerazinski *malware* koristi su:

- Postupno izvršavanje - Višerazinski malware često ima modularnu strukturu koja omogućuje postupni napad. Ta metoda pomaže u izbjegavanju detekcije jer se pojedini moduli ne mogu klasificirati kao maliciozni, već samo kao cjelina. Primjerice, malware može prvo provjeriti prisutnost virtualnog okružena prije nego što nastavi s napadom, ili prikupljati izvršni kod iz više različitih mjesta.
- Lateralno kretanje - Višerazinski *malware* ima sposobnost širenja po mreži kako bi zarazio što više računala. To može uključivati iskorištavanje sigurnosnih slabosti, krađu korisničkih podataka za prijavu ili presretanje prometa.

- Skupljanje obavještajnih podataka - Višerazinski *malware* često prikuplja različite podatke o sustavima, korisnicima ili mrežnim aktivnostima. To im omogućuje da prilagode napad.

2.2. Emotet obitelj *malware*-a

Emotet je vrsta trojanskog konja koji se primarno širi kroz *phishing* elektroničku poštu. Phishing je vrsta prevare u kojoj se napadači predstavljaju kao legitiman izvor i pokušavaju navesti korisnika da preuzme maliciozni dokument ili klikne maliciozni link. Infekcija se događa pomoću maliciozne skripte ili Microsoft Word ili Excel dokumenta koji koristi makro naredbe. Ovi mailovi često sadrže poruke koje izgledaju kao legitimni email te pokušavaj nagovoriti korisnika na otvaranje datoteke sa riječima poput „Račun“, „podaci za plaćanje“ ili nešto vezano za nadolazeću pošiljku. Kada korisnik omogući makro naredbe na ovim datotekama, izvršava se skripta koja preuzima daljnje datoteke sa servera za kontrolu koji održavaju napadači.

Emotet koristi mnoštvo tehnika kako bi izbjegao detekciju i analizu. Jedna od najbitnijih je ta da prepoznaje nalazi li se unutar virtualnog stroja ili *sandbox*-a, što analitičari često koriste za stvaranje sigurnog i kontroliranog prostora za analizu.. Ukoliko zaključi da nije na stvarnom računalu, primiriti će svoje radnje kako bi izbjegao stvarnu analizu.

Ovaj *malware* također koristi i kontrolne servere (engl. *Command and control*) kako bi primio ažuriranja ili instalirao dodatni malware poput drugih trojanskih konja. Ovi serveri se također koriste za slanje prikupljenih financijskih informacija, korisničkih imena i zaporki te email adresa.

Upravo zbog njegovih višestrukih naprednih tehnika, Emotet je odabran kao glavni malware na kojemu će se u ovome radu demonstrirati funkcionalnosti sigurnosnih alata za detekciju i analizu.

3. STATIČKI PRISTUP ANALIZI I DETEKCIJI

Statički pristup analizi i detekciji malicioznih programa baziran je na principu analize izvršnog koda ili binarnih datoteka bez pokretanja. Ovaj način se pokazao kao pouzdan za detekciju malicioznog softvera u trenutku kada se izvršna datoteka prenese na sustav. Analizom se postižu znanja o specifičnostima programa u obliku nizova bajtova koji se pojavljuju u izvršnom kodu. Na temelju dobivenih znanja se mogu razviti pravila koja detektiraju taj zloćudni program metodom podudaranja tih nizova bajtova sa samim izvršnim kodom programa.

Ovaj pristup je izvrstan jer omogućava da se napad na sustav suzbije prije nego što se maliciozni kod izvede. S obzirom na to da su tradicionalni jednorazinski zlćudni programi (engl. *malicious software, malware)* postao previše jednostavan za detekciju, napadači sve češće koriste višerazinske tehnike kako bi izbjegli detekciju. Višerazinski *malware* koristi niz složenih koraka ili komponenti koje su međusobno povezane i često koristi tehnike kamuflaže i enkripcije kako bi se sakrio od tradicionalnih sigurnosnih mehanizama [3, 4].

3.1. Statička analiza

Statička analiza *malware*-a je proces proučavanja zlonamjernog softvera bez stvarnog pokretanja na sustavu. Ova tehnika pruža sigurnosnim stručnjacima dublje razumijevanje funkcionalnosti *malware*-a. Također pomaže u identificiranju ranjivosti i razvijanju odgovarajućih mjera zaštite. Statička analiza *malware*-a obuhvaća razne tehnike i alate za dobivanje podataka za razvoj sigurnosnih mjera. Neke od metoda statičke analize su dekompilacija i deasembliranje, ekstrakcija nizova znakova (engl. *string)* te analiza zaglavlja. Ove tehnike su opisane u daljnjim poglavljima.

3.1.1. Analiza zaglavlja

S obzirom na to da je dominantni operacijski sustav u današnje doba MS Windows, trend razvoja *malware-a* to također potvrđuje. Čak preko 95% novog *malware-a* u 2022. godini je namijenjeno za Windows operacijski sustav [5]. Sukladno tome, analiza zaglavlja (engl. *header)* formata izvršnih datoteka u Windows operacijskom sustavu je vrlo korisna te može sadržavati bitne indikatore zlćudnog *software-a.* Statičkom analizom zaglavlja prijenosnog izvršnog (engl. *Portable Executable, PE*) formata datoteke mogu se dobiti podaci kao što je popis svih dijeljenih biblioteka (engl. *Dynamic-link library, DLL*) i funkcija koje datoteka uvozi. Binarne izvršne datoteke (obično s ekstenzijama poput *exe, dll, sys, acm, mui* i drugih) koje se koriste u svim verzijama Windows operacijskog sustava danas su većinom u PE formatu datoteke koji je definiran točnom strukturom. Zaglavlje PE formata datoteke sadrži informacije potrebne *loader-u* Windows OS-a za upravljanje izvršnim kodom. Kao što samo ime implicira, PE format datoteke je prenosiv između svih verzija Windows OS-a (operacijskog sustava), bez obzira na način na koji procesor izvršava upute računalnog programa. Stoga se PE datoteka može izvršiti na 32-bitnim i 64-bitnim sustavima [3].

PE format datoteka sadrži mnoga polja sa raznim metapodacima, a neka od najkorisnijih za prepoznavanje *malware-a* su [6]:

- Ime sekcije (engl. *Section Name*) - Informacije o svakoj sekciji, uključujući njezino ime, nalaze se u odgovarajućem retku u tablici sekcija. Bezopasni programi gotovo uvijek imaju standardno ime za svaku sekciju, dok većina zloćudnih programa ima nasumično ime za neke sekcije ili čak sadrže neimenovanu sekciju. Većina zlonamjernih programa koristi alate za pakiranje ili nestandardne alate koji rezultiraju nestandardnim imenima sekcija. Ovi alati često koriste unaprijed definirana imena sekcija i ponekad koriste prazna imena sekcija. Na primjer, program UPX packer uvijek preimenuje sekcije .txt i .data u UPX0 i UPX1.
- Podatkovni direktoriji (engl. *data directories)* - PE format definira 16 mogućih podatkovnih direktorija, ali njihov broj u pojedinoj datoteci nije isti. Jedan od njih je direktorij za otklanjanje grešaka (engl. *debug directory*) koji tvorci *malware*-a najčešće izbrišu kako bi izbjegli detekciju i reverzno inženjerstvo. Nadalje, postoje i direktoriji koji sadrže resurse, uvoze, izvoze, konfiguracije i drugi čije postojanje ili odsustvo može doprinijeti mišljenju o zloćudnosti datoteke.
- Kontrolna suma (engl. *checksum*) - Svaka datoteka ima polje kontrolne sume u opcionalnom zaglavlju koje se koristi za validaciju izvršne datoteke prilikom učitavanja i može se koristiti kao provjera integriteta. Vrijednost ovog polja nula za mnoge zlonamjerne programe, dok bezopasni programi imaju vrijednosti različite od nule u ovom polju.
- Podaci o datoteci (engl. *FileInfo*) - Svaka PE datoteka može sadržavati skup polja kao što su ime proizvoda (engl. *ProductName*), verzija proizvoda (engl. *ProductVersion*), verzija datoteke (engl. *FileVersion*), opis datoteke (engl. *FileDescription*) i ime tvrtke (engl. *CompanyName*) koji se pohranjuju u *FileInfo*. Svaka datoteka ima različiti broj ovih polja. Bezopasni programi od pouzdanih pojedinaca ili tvrtki, obično sadrže ova polja, dok mnogi zlonamjerni programi nemaju ovaj skup ili imaju popunjen samo mali broj ovih polja.

3.1.2. Pretvaranje strojnog jezika u asemblerski jezik

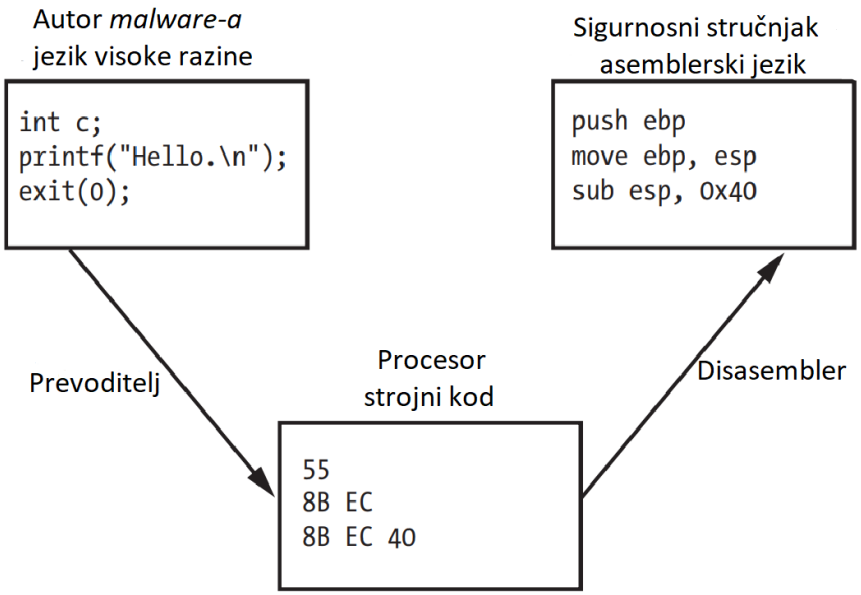
Osnovne tehnike statičke analize mogu pružiti neke preliminarne zaključke, no za cjelovitu sliku potrebna je dublja analiza. Na primjer, možemo otkriti da je određena funkcija uvezena, ali ne i kako se koristi ili je li uopće korištena. Kako bi se razvijeni programi mogli koristiti na više platformi, uglavnom se za njihov razvoj koriste programski jezici više razine (engl. *high level programming languages*). Tada se kod napisan u tim jezicima može kompilirati u strojni jezik (engl. *machine code*) koji se sastoji od operacijskih kodova (engl. *opcodes*), koji su u stvari

heksadekadski znakovi koji govore procesoru koju operaciju izvesti. S obzirom da je ovaj kod teško analizirati, za pretvorbu strojnog jezika u asemblerski jezik koriste se programi za pretvaranje strojnog jezika u asemblerski jezik (engl. *disassembler*). Asemblerski jezici su skupina jezika gdje je svaki vezan za specifičnu arhitekturu procesora. Tako postoje asemblerski jezici za x86, x64, ARM i druge arhitekture mikroprocesora. Ovi jezici se sastoje od kratkih riječi poput `mov` (engl. *move,* premjesti) ili `jmp` (engl. *jump*, skoči) koje predstavljaju simboličku reprezentaciju strojnog jezika [4]. Prema slici 3.1. vidimo da autori malicioznih programa uglavnom pišu kod u jezicima visoke razine te se taj kod onda kompilira u strojni jezik razumljiv procesoru. Iz tog strojnog koda primjenom *disassembler* programa se dobije asemblerski kod čitljiv sigurnosnim stručnjacima.

Disasembliranje omogućava identifikaciju ključnih funkcionalnosti programa, kao što su algoritmi za šifriranje ili komunikacija s vanjskim resursima. Koristeći alate za disasembliranje poput IDA i GHidra programa, analitičari zlonamjernog koda mogu detaljno istražiti unutarnje djelovanje programa, otkriti potencijalne sigurnosne prijetnje i razviti strategije za detekciju i obranu od napada. Disasembliranje je stoga neprocjenjivo sredstvo za razumijevanje i suzbijanje zlonamjernog koda te za unapređenje sigurnosti računalnih sustava.

3.1.3. Ekstrakcija nizova znakova

Izvlačenje sljedova Unicode ili ASCII znakova (engl. *string*) iz sumnjivog programa je još jedna metoda koju analitičari koriste. Cilj je pronaći tekstualne nizove koji imaju neko značenje u



Sl. 3.1. Primjer procesa kompilacije i deasembliranja *malware-a*

binarnim datotekama koji stvaraju slijed bajtova s vrijednostima u rasponu ispisivih znakova i koji završavaju bajtom nulte vrijednosti. Ovo je jednostavna tehnika ekstrakcije podataka iz izvršnih datoteka koja je često vrlo učinkovita. Ovi nizovi mogu sadržavati važne informacije kao što su IP adrese, URL-ovi, registarski ključevi, naredbe za internetsku komunikaciju i putanje datoteka s kojima zlonamjerni program radi. Međutim, ova metoda ne funkcionira s kriptiranim nizovima te tada rezultat može sadržavati velik broj beskorisnih informacija. Autori zlonamjernog softvera često koriste alate i tehnike za otežavanje analize i otkrivanja. Programi bez zlonamjernog koda obično sadrže veliki broj *string*-ova, dok zlonamjerni programi imaju samo nekoliko. Programi s malim brojem nizova su vjerojatno komprimirani i mogu sadržavati zlonamjerni kod. Nakon dešifriranja skrivenog dijela koda, može se ponovno koristiti izvlačenje nizova. Za pretragu nizova pohranjenih u programu mogu se koristiti specijalizirani programi poput Strings, HexDive ili BinText [3].

3.2. Statička detekcija

Statičkom analizom stručnjaci mogu donijeti mnoge zaključke o namjerama, opasnostima i validnosti programa, ali je taj proces najčešće dug i skup. Stoga se nastoji kao rezultat statičke analize odrediti značajke zloćudnog programa te na temelju tih značajki razviti sustave za automatsku detekciju zlonamjernih i opasnih programa.

Statička detekcija zlonamjernog programa obuhvaća proučavanje izvršnih datoteka kako bi se identificirali opasni dijelovi koda i njegove mogućnosti. Ova metoda omogućava analitičarima da otkriju sumnjive obrasce djelovanja na temelju specifičnih nizova bajtova ili strukture datoteka koji su često povezani sa zloćudnim aktivnostima.

Bitna prednost statičke detekcije je brzina i jednostavnost primjene. Analiza zlonamjernih datoteka može se provesti bez pokretanja programa, što štedi vrijeme i smanjuje rizik od izvršenja zlonamjernog koda. Osim toga, glavna prednost statičke detekcije je to što omogućava otkrivanje zlonamjernog softvera prije nego što dođe do pokretanja ili širenja. To je važno jer višerazinski *malware* često koristi sofisticirane tehnike za izbjegavanje otkrivanja tijekom izvršenja, što dinamičku analizu čini manje pouzdanom.

U daljnjim poglavljima ćemo objasniti i istražiti neke osnovne principe i tehnike statičke detekcije zlonamjernog softvera, kao što su detekcija prema potpisu (engl. *Signature-based detection*), heuristička detekcija (engl. *Heuristic-based detection*) i detekcija prema nizovima znakova (engl. *String-based detection*).

3.2.1. Detekcija prema potpisu

Jedan od najčešće korištenih pristupa za detekciju *malware*-a je analiza potpisa. Analiza potpisa se temelji na uspoređivanju potpisa datoteke s popisom poznatih potpisa zlonamjernih programa. Ovaj pristup nam omogućuje brzo identificiranje dobro poznatih prijetnji i zato se najčešće koristi kao glavna metoda detekcije u komercijalnim antivirusnim programima.

Analiza potpisa nije moguća bez postojeće baze podataka koja sadrži potpise poznatih zlonamjernih programa. Potpis je specifičan skup bajtova ili uzoraka koji se javljaju unutar zlonamjernih datoteka i koji ih razlikuju od legitimnih programa. Potpisi se mogu sastojati od niza bajtova, heksadekadskih vrijednosti ili drugih oblika koji predstavljaju karakteristične obrasce.

Prilikom detekcije, binarna datoteka koja se analizira uspoređuje se s potpisima iz baze podataka. Ako se pronađe podudaranje, to ukazuje da je datoteka potencijalno zlonamjerna. Ova metoda je vrlo učinkovita u otkrivanju već poznatih prijetnji jer se oslanja na prethodno prikupljene informacije o zlonamjernom softveru.

Prednosti analize potpisa su brzina i jednostavnost implementacije. Također, analiza potpisa može biti vrlo precizna i rijetko daje lažno pozitivne rezultate. Ipak, postoji nekoliko izazova i ograničenja povezanih s ovom metodom. Analiza potpisa ne može otkriti nove ili nepoznate prijetnje koje nemaju prethodno definirane potpise. To znači da zlonamjerni programi koji su nedavno razvijeni ili su promijenjeni mogu proći nezapaženo. Analiza potpisa može biti ograničena i u detekciji naprednih prijetnji koje koriste tehnike kamuflaže, enkripcije ili pakiranja kako bi se izbjeglo prepoznavanje [4].

3.2.2. Heuristička detekcija

Heuristička detekcija je metoda statičke detekcije *malware*-a koja se bavi prepoznavanjem sumnjivih ili opasnih uzoraka u izvršnim datotekama. Umjesto korištenja specifičnih potpisa, koristi se heuristički pristup razvijanju algoritama za otkrivanje potencijalno malicioznih radnji.

Heuristika je skup pravila ili algoritama koji se temelje na iskustvu i znanju stručnjaka za sigurnost. Oni analiziraju strukturu i sadržaj binarne datoteke kako bi identificirali sumnjive oblike ponašanja ili karakteristike koje su često povezane sa zlonamjernim softverom. Neki od ovih oblika su [2]:

- Nepravilnosti u strukturi datoteke - Analizira se struktura datoteke kako bi se otkrile nepravilnosti, poput nevaljanih referenci ili abnormalnih veličina i rasporeda sekcija.

- Sumnjive radnje s memorijom - Prati se upotreba memorije i provjeravaju se sumnjive operacije kao što su pisanje u zaštićena područja memorije ili sumnjivi skokovi.
- Pokušaji preuzimanja ili izvršavanja vanjskih resursa - Analiziraju se pokušaji komunikacije s vanjskim resursima, kao što su mrežni poslužitelji ili datoteke na disku.
- Sumnjive ili štetne operacije s datotekama: Neovlaštene operacije s datotekama, kao što su brisanje, preimenovanje ili promjena dozvola mogu ukazivati na zlonamjernu namjeru.
- Upotreba poznatih ranjivosti ili iskorištavanje sigurnosnih propusta: Provjerava se prisutnost koda ili korištenje tehnika koje su poznate kao sigurnosne ranjivosti ili metode eksploatacije kako bi se dobila neovlaštena kontrola nad sustavom.

Glavna prednost heurističke analize je mogućnost detekcije novih ili nepoznatih prijetnji koje nemaju prethodno definirane potpise. Prepoznavanje na temelju pojedinih obrazaca, a ne cjelokupnog potpisa omogućuje detekciju novih varijanti i vrsta *malware*-a. Može otkriti prijetnje koje su razvijene nakon ažuriranja baze podataka potpisa i može pružiti zaštitu od naprednih prijetnji koje koriste razne tehnike skrivanja i prevara.

Heuristička detekcija kao nedostatak nosi dosta visoku mogućnost davanja lažno pozitivnih rezultata, tj. označavanja bezazlenih datoteka kao zloćudnih. To se može dogoditi kada se neobični obrasci interpretiraju pogrešno ili kada legitimni programi imaju slična ponašanja onima koja su opisana heurističkom metodom. Kako bi se smanjilo postotak netočnih detekcija, sigurnosni timovi implementiraju tehnike analize koje koriste strojno učenje i umjetnu inteligenciju za bolju evaluaciju obrazaca [2]. Kombinacija provjere potpisa programa i heurističke analize pruža puno efikasniju detekciju *malware*-a pa zato antivirusni programi najčešće kombiniraju ove dvije metode.

3.2.3. Detekcija prema nizovima znakova

U prijašnjim poglavljima je opisano da je ekstrakcija nizova proces identifikacije i izdvajanja sekvenca Unicode ili ASCII znakova iz izvornog koda ili binarne datoteke. Kada se nizovi izdvoje, mogu se dalje analizirati i usporediti s bazom podataka poznatih zlonamjernih nizova ili prepoznatih uzoraka. Ovo omogućuje identifikaciju karakteristika *malware-a* ili potencijalno sumnjivih aktivnosti. Na primjer, određeni nizovi mogu ukazivati na pokušaj komunikacije s kontrolnim serverima, izvršavanje skrivenih funkcionalnosti ili manipulaciju datoteka i Windows registra.

Ova metoda pokazala se kao jedna od najuspješnijih metoda statičke detekcije za specifičan *malware*. Razvijeni su modeli strojnog učenja na temelju saznanja ekstrakcije *string*-ova kako bi

sigurnosni analitičari imali manje posla sa pojedinačnim datotekama [7]. Unatoč tome, modeli umjetne inteligencije i strojnog učenja zahtijevaju mnoštvo primjera nekog ponašanja da bi naučili i promijenili svoju klasifikaciju istog. Stoga takvi modeli nerijetko ne mogu otkriti inovativni *malware* koji koristi neke nove, još neviđene tehnike. S obzirom na to da je u sigurnosti računalnih sustava brza reakcija na prijetnju ključna, ovakve nadolazeće prijetnje sigurnosni analitičari analiziraju i razvijaju pravila koja se mogu odmah primijeniti za detekciju novopronađenog *malware-a*. Kao standard za razvoj pravila za statičku detekciju koristi se YARA (*Yet Another Recursive Acronym*). Stručnjaci nakon analize iz dobivenih saznanja razvijaju YARA pravila koja se onda primjenjuju u određenom stupnju sustava zaštite. Ova pravila se često javno objavljuju kako bi se i ostali korisnici mogli zaštititi od nove prijetnje.

3.3. Razvoj detekcijskih pravila

3.3.1. YARA pravila

YARA pravila način su identificiranja zlonamjernog softvera (ili drugih datoteka) stvaranjem pravila koja traže određene karakteristike. YARA je izvorno razvio Victor Alvarez iz tvrtke

Linija ***Kod***

```
1:      rule ime_pravila{
2:      meta:
3:           description="Ovo je opis pravila."
4:           author="Matija Zagar"
5:      strings:
6:           $s1="Tekstualni string."
7:           $s2={6A 40 68 00 30 00 00 6A 14 8D 91}
8:      condition:
9:           $s1 or $s2
10:     }
```

Sl. 3.2. Primjer jednostavnog YARA pravila

Virustotal i uglavnom se koristi u istraživanju i otkrivanju zloćudnih programa. Sadržava mogućnost opisivanja obrazaca koji identificiraju određene vrste ili cijele obitelji zloćudnih programa. Korištenjem YARA pravila mogu se definirati tekstualni ili binarni obrasci koji se pojavljuju u izvršnom ili izvornom kodu. Svako obrazac, tj. pravilo, sastoji se od skupa nizova znakova i izraza Bool-ove logike koji određuje njegovu logiku klasifikacije. Svako pravilo započinje ključnom riječi `rule` i sadrži tri dijela: meta, strings i condition [8].

Prva sekcija predstavlja metapodatke o pravilu. Neka od polja koja se najčešće navode u ovoj sekciji su `author` (ime autora), `source` (izvor pravila), `category` (kategorija), `description` (opis pravila), `malware` (ime *malware*-a) i `status` (status objavljenosti pravila).

U drugom dijelu se definiraju nizovi znakova koji će se tražiti u datoteci. Svaki niz ima identifikator koji se sastoji od znaka $ iza kojeg slijedi niz alfanumeričkih znakova i podvlaka. Ovi se identifikatori mogu koristiti u dijelu uvjeta za referenciranje na odgovarajući niz. Nizovi se mogu definirati u tekstualnom (linija 6 na slici 3.2) ili heksadekadskom (linija 7 na slici 3.2) obliku. Pri definiciji stringa smiju se koristiti *wild card*-ovi koji zamjenjuju neki raspon i broj znakova. Oni pomažu pri primjeni pravila na čitave obitelji zloćudnih programa i u suzbijanju metamorfnog *malware*-a.

Zadnji dio pravila je condition (uvjet). Ovaj odjeljak mora sadržavati izraz Boolove logike koji govori pod kojim okolnostima datoteka ili proces zadovoljava pravilo ili ne. Općenito, uvjet će se odnositi na prethodno definirane nizove pomoću njihovih identifikatora. Koriste se logički izrazi poput `or` (ili), `and` (i) te neki drugi poput `all of` (sva od navedenih). U ovom kontekstu identifikator niza se ponaša kao varijabla bool tipa (može imati samo vrijednost istina ili neistina) koja procjenjuje da je istinit ako je niz pronađen u datoteci ili procesnoj memoriji, ili lažan ako nije [8, 9].

3.3.2. Automatizirano stvaranje detekcijskih pravila

Automatizacija je ključni faktor učinkovite i skalabilne detekcije zlonamjernog softvera. Jedan od alata koji omogućuje automatizirano stvaranje detekcijskih pravila je YarGen. Ovaj program pruža analitičarima mogućnost generiranja YARA pravila na temelju uzoraka zlonamjernog softvera.

Automatizirano stvaranje detekcijskih pravila pomoću YarGena pruža analitičarima temeljne okvire za detekciju zlonamjernog softvera, ali daljnje prilagodbe i optimizacija pravila su potrebne za postizanje najboljih rezultata. Ovaj proces također zahtijeva redovito ažuriranje skupa uzoraka i pravila kako bi se održala relevantnost i učinkovitost detekcije. Ovaj alat smanjuje vrijeme i napore potrebne za ručno pisanje svakog pravila te omogućuje bržu identifikaciju i detekciju novih prijetnji, poboljšavajući ukupnu sigurnost i odgovor na zlonamjerni softver [10].

YarGen pravilo može biti ili jednostavno (engl. *simple*) pravilo ili super pravilo. Ako se koristi više datoteka uzoraka, YarGen će pokušati identificirati sličnosti između uzoraka i

kombinirati identificirane nizove u "super pravilo". Super pravila mogu se identificirati tako što će u meta odjeljku pravila vrijednost varijable `super_rule` biti postavljena na 1. Proces kombiniranja više pravila u jedno super pravilo ne uklanja jednostavna pravila generirana za svaku datoteku. To znači da će doći do preklapanja nizova pravila između jednostavnih pravila i super pravila te treba ukloniti jedne ili druge. To možemo postići korištenjem zastavica `nosuper` (ne kreiraj super pravila) i `nosimple` (ne kreiraj jednostavna pravila) tijekom pokretanja programa YarGen [10, 9].

YarGen generira pravila prema svim standardnim pravilima Yara pravilo. Međutim, YarGen kategorizira odjeljak nizova na temelju vjerojatnosti da su pokazatelj zlonamjernog softvera. Postoje tri kategorije ovih nizova, označene sa $s, $x i $z. Nazivi nizova koji počinju s $s su "vrlo specifični nizovi" koji se neće pojaviti u legitimnom softveru. Ovi nizovi mogu uključivati zlonamjerne adrese poslužitelja, nazive alata za hakiranje i zlonamjernog softvera i pogreške pri upisu u uobičajenim nizovima. Nizovi koji počinju s $x su "specifični nizovi" koji su vjerojatno pokazatelji zlonamjernih datoteka, ali se mogu pojaviti i u legitimnim datotekama. Nizovi koji počinju s $z vjerojatno su bezazleni, ali trenutno nisu uključeni u bazu podataka nizova dobrih nizova (engl. *goodware*). YarGen koristi kombinaciju čarobnog zaglavlja (nizovi bajtova koji određuju tip datoteke), veličine datoteke i nizova za sekciju uvjeta [9].

3.3.3. Razvoj detekcijskih pravila za Emotet

Emotet je jedan od najopasnijih i najraširenijih malvera koji je uzrokovao velike probleme u svijetu sigurnosti informacija. Najčešće je kategoriziran kao trojanski konj koji je namijenjen za krađu financijskih podataka i širenje drugih zlonamjernih programa. Emotet se često ažurira i prilagođava novim taktikama i tehnikama kako bi izbjegao sigurnosne mjere.

Razvoj detekcijskih pravila za Emotet malware postao je ključan u borbi protiv ove prijetnje. Program yarGen pruža analitičarima mogućnost efikasnog generiranja pravila koja mogu identificirati prisutnost Emotet malvera u sustavu. YarGen omogućuje analitičarima da izdvoje ključne obrasce ponašanja, kodne segmente, stringove i druge karakteristike koje su prisutne u Emotet *malware*-u. Na temelju tih karakteristika, yarGen generira YARA pravila koja mogu detektirati prisutnost Emotet malwarea u datotekama ili mrežnom prometu. Iako su vrlo efikasna, YARA pravila se ne mogu smatrati dovoljnom zaštitom od ove vrste prijetnje zbog njegove prirode različitih načina zaraze te mnoštva različitih oblika dostave zloćudnog koda [10, 9].

U svrhu prikupljanja uzoraka malware-a tipa Emotet korištena je web stranica MalwareBazaar [11] koja besplatno nudi bazu uzoraka zloćudnih programa. Preuzete su tri

inačice izvršnih datoteka za koje je potvrđeno da su željenog tipa. Izvršavanjem yarGen programa na ove tri datoteke dobivena su 3 jednostavna pravila te jedno super pravilo.

3.3.3.1. Pravilo za prvi uzorak

Generirano pravilo za detekciju prema prvoj datoteci (slika 3.3.) koristi više stringova koji predstavljaju određene nizove bajtova u datoteci, kao što su nazivi datoteka, dijelovi XML koda, IP adrese i druge karakteristike koje se mogu pojaviti u Emotet malwareu. Neki od najbitnijih stringova iz pravila su slijedeći:

- String $s1 (linija br. 9) - Ovaj string traži prisutnost niza "testa.EXE" kao punu riječ u datoteci. "Fullword" označava da se traži točno podudaranje cijelog niza, a "wide" ukazuje da se uzorak tretira kao Unicode. Datoteka na koju se referencira je potencijalno maliciozna te je zato potrebno nju odvojeno analizirati.
- String $s2 (linija br. 10) -: Ovaj string traži određeni XML kod koji opisuje identitet skupa datoteka. Pretražuje se prisutnost točno podudarajućeg niza znakova kao punu riječ, a "ascii" označava da se uzorak tretira kao ASCII tekst. Benigni programi rijetko koriste asemblerski kod neposredno te je stoga ovo ponašanje vrlo sumnjivo.
- String $s10 (linija br. 18): Ovaj string traži prisutnost IP adrese "38.47.220.53" kao punu riječ u datoteci. Ponovno, traži se točno podudaranje cijelog niza, a "ascii" označava da se uzorak tretira kao ASCII tekst. Ova informacija je izrazito bitna jer je za Emotet specifično ponašanje komunikacija sa kontrolnim serverom (engl. *Command and control, C&C*) koji mu pruža ažuriranja i naredbe.
- String $s14 (linija br. 22): Ovaj string predstavlja heksadecimalni zapis nekog niza bajtova. Analizirajući duljinu stringa (128 znakova), primjećujemo da vjerojatno predstavlja rezultat neke hash-funkcije. U kontekstu detekcije Emotet malwarea, ovaj string može predstavljati prepoznatu hash vrijednost zaražene datoteke koja je karakteristična za Emotet.

Možemo primjetiti da stringovi s8, $s12, $s15, $s16, $s17, $s18 sadrže nasumične nizove znakova koje je program yarGen izdvojio na temelju toga što su unikatni, ali u stvarnosti ne donose nikakvu dodanu vrijednost u detekciji ovog *malware*-a jer nisu esencijalni za njegovo izvođenje te se u novim inačicama vrlo lako mogu izostaviti. Ove stringove bi u ručnoj analizi trebalo ukloniti iz detekcijskih pravila.

3.3.3.2. Super pravilo

Analizom dostupnih datoteke, program yarGen je prepoznao da su druga dva uzorka dovoljno slična da se mogu detektirati jednim pravilom bez prevelikog broja lažno pozitivnih rezultata. Kao rezultat generirano je pravilo na slici 3.4. Ovo Yara pravilo koristi se za detekciju Emotet malwarea iz datoteka "strain2.exe" i "strain3.exe". Analizom ovog pravila možemo uočiti nekoliko bitnih stringova koji se koriste za prepoznavanje prisutnosti Emotet malwarea:

- $s2, $s3, $s6, $s8, $s9, $s14, $s15, $s16, $s17, $s18, $s19: Ovi stringovi sadrže poruke o neuspjelim operacijama koje se odnose na ključne funkcije ili biblioteke koje Emotet malware koristi tijekom izvršavanja.
- $s4: Ovaj string predstavlja definiciju "assemblyIdentity" koja se povezuje s "Microsoft.Windows.Common-Controls", a može se koristiti za detekciju Emotet-a koji pokušava iskoristiti ovu komponentu.
- $s1, $s5, $s7, $s10, $s11, $s12, $s13, $s20: Ovi stringovi predstavljaju nazive DLL datoteka koje su također povezane s Emotet malwareom. Može se primjetiti kako njihovi nazivi nalikuju na stvarne DLL-ove, ali su u stvarnosti ovo potpuno druge biblioteke. String $s12 prikazuje biblioteku „`bpython310.dll`“, dok legitimna biblioteka ima naziv „`python310.dll`“. String $s1 koristi biblioteku „`pywintypes310.dll`“, koja je pravog naziva te inače bezazlena, ali se specificira i njen roditeljski direktorij koji glasi „`bpywin32_system32`“, dok u stvarnosti bi se trebao zvati „`pywin32_system32`“. Kako bi izbjegli to da neka nova inačica promjeni početno slovo naziva biblioteka, možemo koristiti funkcionalnost *wild card.* Tada bi u kodu umjesto početnog slova koje je pronađeno stavili znak „?“. Prema tome bi za $s12 vrijednost bila „`?python310.dll`“.

Windows, poput mnogih operacijskih sustava, dopušta učitavanje DLL-ova tijekom izvođenja. Programi mogu odrediti lokaciju DLL-ova za učitavanje navodeći punu putanju do nje na računalu, korištenjem preusmjeravanja DLL-a ili pomoću manifesta programa koji sadržava metapodatke o programu. Ako nijedna od ovih metoda nije iskorištena, Windows pokušava locirati željeni DLL pretraživanjem unaprijed definiranog skupa direktorija određenim redoslijedom. Napadači zlouporabljuju ovaj način pretrage postavljanjem zlonamjernog DLL-a u jedan od direktorija koji se pretražuju. Tada Windows pronalazi i učitava zlonamjerni DLL prije pronalaska legitimne biblioteke. Stoga su korišteni DLL-ovi jako bitni pokazatelj namjere programa te se mogu koristiti za identifikaciju *malware*-a [12].

Uzorci iz kojih su ova pravila generirana drukčijih su veličina te drukčijih hash vrijednosti, ali su pronađeni stringovi isti. Ovo upućuje na to da Emotet primjenjuje tehnike metamorfizma kako bi promijenio svoju hash vrijednost te tako izbjegao detekciju prema potpisu.

Linija ***Kod***

```
1.      rule Emotet_1 {
2.         meta:
3.            description = "Emotet_malware - file strain1.exe"
4.            author = "Matija Zagar"
5.            reference = "https://github.com/Neo23x0/yarGen"
6.            date = "2023-06-29"
              hash1 = "0e70671889e441e9926c6a731ce722797526a7ebf785d4c8cd5fa6a
7.      e671c8924"
8.         strings:
9.            $s1 = "testa.EXE" fullword wide
              $s2 = "  <assemblyIdentity version=\"1.0.0.0\" processorArchitec
10.     ture=\"*\" name=\"YourAppName\" type=\"win32\"/>" fullword ascii
11.           $s3 = "-- loop recv thread --" fullword ascii
12.           $s4 = "---- running ----" fullword ascii
13.           $s5 = "---- CDwonKey ----" fullword ascii
14.           $s6 = "---- exit ----" fullword ascii
              $s7 = "        <requestedExecutionLevel level=\"requireAdministr
15.     ator\" uiAccess=\"false\"/>" fullword ascii
16.           $s8 = "c:\\ProgramData" fullword ascii
              $s9 = "  <trustInfo xmlns=\"urn:schemas-microsoft-com:asm.v3\">"
17.      fullword ascii
18.           $s10 = "38.47.220.53" fullword ascii
              $s11 = "  <description>Your application description here.
19.                 </description>" fullword ascii
20.           $s12 = "VWuBh,AC" fullword ascii
21.           $s13 = "  </trustInfo>" fullword ascii
              $s14 = "5f5962580c585979017935195aa97d75fa9cc89e2914fd4e192e6fcc
22.     c8328cdbb2c8df54a6261e6746c361ebde1037f01ebf" ascii
23.           $s15 = "Qbwvhf Pguddrhg Ekofkiop Urwk" fullword ascii
24.           $s16 = "Kphfyvsd Xabpalsls Wpiegkm Vapqedcg Pts" fullword ascii
25.           $s17 = "1a2aabc31c5d" ascii
26.           $s18 = "Intgva Wxkofiuv Hbw" fullword ascii
              $s19 = "6a555e88896d604d6060636f0c8e75f5a4153c6e032ea6f28825a44d
27.     42692f77ad588b2288e39b08251977001a2aabc31c5d" ascii
              $s20 = "4f5e596d885968545e596d580c122c0de022f5e9f1aefe6421538cb12
28.     c243bcbcd6f788d364f63a33c0db890" ascii
29.        condition:
30.           uint16(0) == 0x5a4d and filesize < 800KB and 8 of them
31.     }
```

Sl. 3.3. Generirano YARA pravilo za prvi Emotet uzorak

Linija Kod

```
1.      rule Emotet_super_rule {
2.         meta:
              description = "Emotet_malware - from files strain2.exe, strain3.
3.      exe"
4.            author = "Matija Zagar"
5.            reference = "https://github.com/Neo23x0/yarGen"
6.            date = "2023-06-29"
              hash1 = "bab5e34e2c6d0ee621ae10a59eadc5b76c6cafe19d60b48b99a15b2
7.      c0e127bec"
              hash2 = "c3c1aa22ce237c9c533f077f4d4f08a81a3f0d93d5deb45b18d837c
8.      d6b916320"
9.         strings:
10.           $s1 = "bpywin32_system32\\pywintypes310.dll" fullword ascii
              $s2 = "Failed to get address for PyImport_ExecCodeModule" fullwo
11.     rd ascii
              $s3 = "Failed to get address for Tcl_FindExecutable" fullword as
12.     cii
              $s4 = "      <assemblyIdentity type=\"win32\" name=\"Microsoft.W
        indows.Common-Controls\" language=\"*\" processorArchitecture=\"*\" ve
13.     r" ascii
14.           $s5 = "bVCRUNTIME140.dll" fullword ascii
15.           $s6 = "Failed to get address for Tcl_MutexUnlock" fullword ascii
16.           $s7 = "bpython3.dll" fullword ascii
              $s8 = "Failed to get address for Py_NoUserSiteDirectory" fullwor
17.     d ascii
18.           $s9 = "Failed to get address for Tcl_MutexLock" fullword ascii
19.           $s10 = "bVCRUNTIME140_1.dll" fullword ascii
20.           $s11 = "bsqlite3.dll" fullword ascii
21.           $s12 = "bpython310.dll" fullword ascii
22.           $s13 = "6python310.dll" fullword ascii
              $s14 = "Failed to extract %s: failed to open target file!" fullw
23.     ord ascii
              $s15 = "LOADER: Failed to expand environment variables in the ru
24.     ntime-tmpdir." fullword ascii
              $s16 = "LOADER: Failed to convert runtime-tmpdir to a wide strin
25.     g." fullword ascii
              $s17 = "LOADER: Failed to obtain the absolute path of the runtim
26.     e-tmpdir." fullword ascii
27.           $s18 = "multiprocessing.spawn)" fullword ascii
28.           $s19 = "anyio.abc._subprocesses)" fullword ascii
29.           $s20 = "blibcrypto-1_1.dll" fullword ascii
30.        condition:
31.           ( uint16(0) == 0x5a4d and filesize < 30000KB and ( 8 of them ) )
```

Sl. 3.4. Generirano YARA super pravilo za druga dva Emotet uzorka

3.3.4. Testiranje generiranih pravila

U ovom poglavlju ćemo analizirati uspjeh generiranih Yara pravila za detekciju malicioznih programa. Kako bismo dobili objektivnu procjenu uspješnosti, koristit ćemo rezultate skeniranja s web stranice RiskMitigation [13]. Ova platforma omogućuje skeniranje naših Yara pravila kroz bazu malicioznih programa na MalwareBazaar platformi [11].

Skeniranje putem ove platforme pruža uvid u učinkovitost naših Yara pravila u detekciji stvarnih prijetnji. Rezultati skeniranja pružaju informacije o broju detekcija koje su naša pravila postigla u odnosu na dostupnu bazu malwarea. Ovi rezultati pomažu u procjeni pouzdanosti i efikasnosti naših pravila te njihovoj sposobnosti otkrivanja višerazinskog *malware*-a.

Rezultati skeniranja za prvo pravilo bez ikakvih izmjena pokazali su samo jednu detekciju, što je upravo ona koju smo analizirali. Ovo može ukazivati na ograničenu učinkovitost pravila u detekciji višerazinskog *malware*-a, s obzirom na broj detekcija koji su postignuti.

S druge strane, "Super pravilo" je generiralo značajan broj detekcija. Detektirane su 399 instanci malicioznog softvera tipa Heodo (drugog naziva za Emotet), što ukazuje na dobru sposobnost pravila u identifikaciji ovog posebnog tipa *malware*-a. Također, zabilježene su 33 detekcije tipa RAT, 8 detekcija tipa CoinMiner i veliki broj detekcija nepoznatog tipa - njih 287.

Ovi rezultati ukazuju na veću učinkovitost generiranog super pravila u odnosu na prvo pravilo. Generirano super pravilo je pokazalo bolje rezultate u identifikaciji i detekciji različitih vrsta višerazinskog malwarea, ali je također neuspješno u detekciji prvog uzorka. Iz tog razloga je nužno održavati veliku bazu pravila za detekciju te sve nove uzorke analizirati te korigirati detekcijska pravila.

4. DINAMIČKI PRISTUP ANALIZI I DETEKCIJI

Dinamički pristup analizi i detekciji višerazinskog *malware*-a je esencijalan u borbi protiv kibernetičkih prijetnji današnjeg doba. Višerazinski *malware* često koristi tehnike izbjegavanja detekcije kako bi izbjegao otkrivanje statičkom analizom. Dinamička analiza se bavi proučavanjem ponašanja *malware*-a dok se on izvršava te otkrivanjem njegovih skrivenih radnji i povezanih dokumenata i programa. Iz svih tih podataka se generiraju izvještaji koji služe za dublje razumijevanje prijetnji te bolju zaštitu u budućim napadima.

U ovom poglavlju pruža se pregled dinamičkog pristupa analizi i detekciji višerazinskog *malware*-a. Opisane su bitne tehnike i alate koje analitičari najčešće koriste u analizi *malware*-a

u dinamičkom okruženju. Metode poput sandbox sustava, debugiranja (engl. *debugging*) i snimanja prometa omogućavaju donošenje odluka. Također ćemo istražiti metode za analizu ponašanja *malware*-a, kao što su praćenje mrežne aktivnosti, sustavskih poziva i promjena na disku.

4.1. Metode dinamičke analize višerazinskog *malware*-a

Dinamička analiza odnosi se na postupak analiziranja koda ili skripte izvršavanjem i promatranjem njegovih radnji. Te radnje mogu se promatrati na višim ili nižim razinama, od najnižih razina same izvedbe strojnog koda do visoke razine operacijskog sustava gdje se prate promjene npr. promjene u Windows registru ili datotekama. Cilj dinamičke analize je otkriti zlonamjerne aktivnosti izvršnog datotečnog programa dok se izvodi, ne ugrožavajući sigurnost analitičke platforme. S obrambene perspektive, postoji rizik od zaraze zlonamjernim softverom prilikom dinamičke analize, jer zahtijeva da se zlonamjerni program učita u radnu memoriju i izvrši na procesoru (engl. *Central Processing Unit, CPU*).

Za razliku od statičke analize, dinamička analiza se ne oslanja na analizu strojnog koda i obrazaca ili potpis. Takav pristup ranjiv je na mnoge tehnike izbjegavanja kao što su pakiranje i obfuskacija te mnoge druge. Osim toga, dinamička analiza ne prevodi binarni kod u kod na razini asemblerskog jezika (disasembliranje). Iako se postupak disasembliranja čini najučinkovitiji, postoje načini koje se koriste kako bi zavarali disasembler i proizveli drugačiji asemblerski kod od onog koji se zapravo izvršava. Izbjegavanjem postupka disasembliranja, dinamička izbjegava u potpunosti takve tehnike maskiranja malicioznog koda. nadalje, statička analiza ne može zamijetiti izmjene koda koje se provedu u tijeku izvršavanja, dok dinamička analiza može popratiti sve dinamičke promjene [14].

4.1.1. Sandbox

Sandbox sustavi su jedni od najkorisnijih alata korištenih za dinamičku analizu višerazinskog *malware*-a. Ovi sustavi simuliraju izolirano okruženje u kojem se izvršava sumnjiva datoteka kako bi se pratilo njezino ponašanje i identificirali potencijalni zlonamjerni postupci. Sandboxovi pružaju siguran prostor za izvođenje *malware*-a bez utjecaja na stvarni sustav.

Sandbox sustavi koriste se za prikupljanje informacija o aktivnostima *malware*-a tijekom njegove izvedbe. Oni prate i bilježe sve sistemske pozive, promjene registra, mrežnu komunikaciju i ostale relevantne događaje. Također, mogu pratiti interakciju *malware*-a s datotekama i mrežnim resursima te snimiti snimke zaslona ili zapise pritisaka tipkovnice.

Prednosti korištenja sandbox sustava u analizi višerazinskog *malware*-a su višestruke. Oni omogućavaju proučavanje ponašanja *malware*-a u kontroliranom okruženju u svrhu identifikacije njegovih karakteristika i funkcionalnosti. Također, sandboxovi pružaju sigurnost jer izoliraju zlonamjerne aktivnosti od stvarnog sustava, sprečavajući potencijalne negativne posljedice.

Postoji nekoliko popularnih sandbox sustava koji se koriste u analizi višerazinskog malwarea, uključujući:

- Cuckoo Sandbox: Cuckoo Sandbox je sandbox sustav otvorenog koda koji omogućava analitičarima da izvršavaju i analiziraju zlonamjerne datoteke. Ovaj sandbox nudi informacije o mrežnoj komunikaciji, registru, datotekama i drugim aktivnostima.
- VMRay Analyzer: VMRay Analyzer je sandbox platforma koja koristi virtualizaciju za analizu i detekciju malwarea. Pruža detaljne informacije o ponašanju malwarea, uključujući rezultate analize i vizualizaciju aktivnosti.
- FireEye Sandbox: FireEye Sandbox je integrirani sandbox sustav koji kombinira virtualizaciju, detekciju i analizu u stvarnom vremenu. Omogućava analitičarima da istraže ponašanje malware-a, identificiraju prijetnje i generiraju izvještaje o zlonamjernim aktivnostima.
- Triage Sandbox: Triage Sandbox je online sandbox koji omogućava prenošenje datoteke i analizu unutar web preglednika. Automatski generira statičko i dinamičko izvješće o željenoj datoteci te pruža mogućnost pristupa virtualnom stroju koji je pokrenuo datoteku. Također je moguće preuzeti podatke o razmijenjenim mrežnim paketima i preuzeti uzorak.

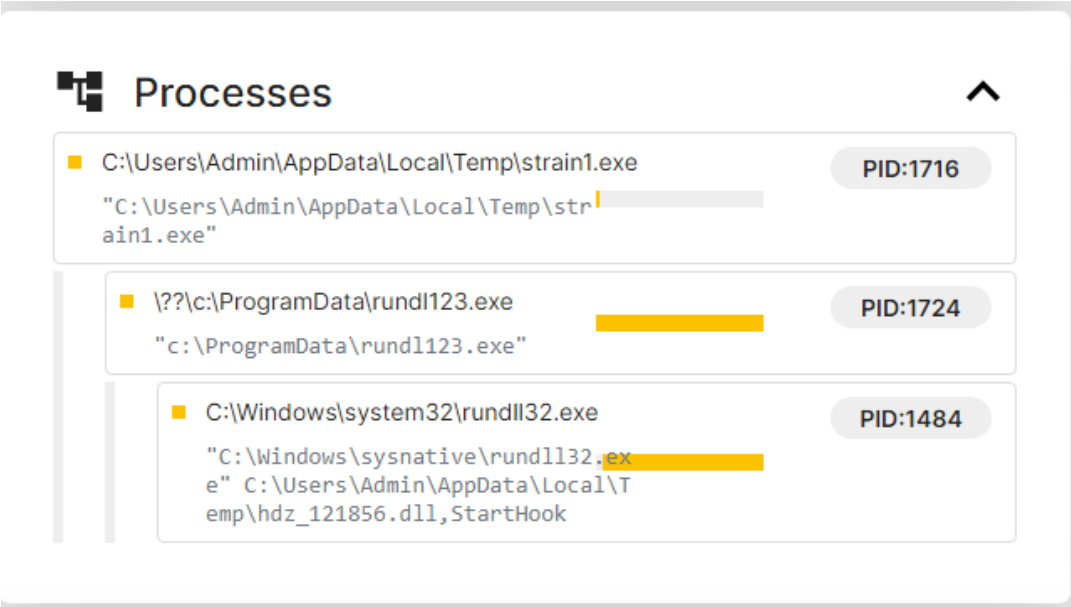
Međutim, sandbox sustavi također imaju neka ograničenja. Napredni napadači mogu prepoznati sandbox okruženja i prilagoditi svoje aktivnosti kako bi izbjegli otkrivanje. *Malware* često primjenjuje tehnike kao što je provjera broja jezgri i količine memorije. Zbog ograničenja resursa, sandboxovi pridjeljuju minimalno potreban broj jezgri procesoru virtualnog stroja, obično samo jednu, kako bi se mogli izvoditi paralelno na što više VM-ova na poslužitelju. Međutim, većina modernih računala ima više CPU jezgri. *Malware* će izvršiti upit za dohvat broja jezgri, i ako je vrijednost jedan, zaključuje da se izvodi unutar sandboxa. Kao i u slučaju CPU jezgri, virtualni strojevi obično imaju ograničenu količinu prostora na disku i fizičke memorije. Da bi otkrio radi li se na virtualnom stroju, *malware* provjerava je li ukupan prostor na disku nizak, kao što

je recimo manje od 80 GB. Slično tome, provjerava ima li malu količinu radne memorije, poput manje od 1 GB RAM-a (engl. *Random Access Memory*). Ove konfiguracije obično nisu prisutne na računalima stvarnih korisnika. Još jedna od metoda detekcije sandbox okruženja je čekanje. Sandboxevi često odustanu nakon nekog vremena pa zloćudni programi iskoriste tu činjenicu kako bi se pokrenuli u nekom trenutku u budućnosti te izbjegli detekciju [15, 4].

Unatoč tim problemima, sandbox sustavi i dalje su važan alat u analizi višerazinskog malwarea. Njihova sposobnost pružanja detaljnih informacija o ponašanju malwarea omogućuje analitičarima da razumiju njegove napredne tehnike i strategije te stoga igraju ključnu ulogu u otkrivanju i suzbijanju ovih sofisticiranih prijetnji.

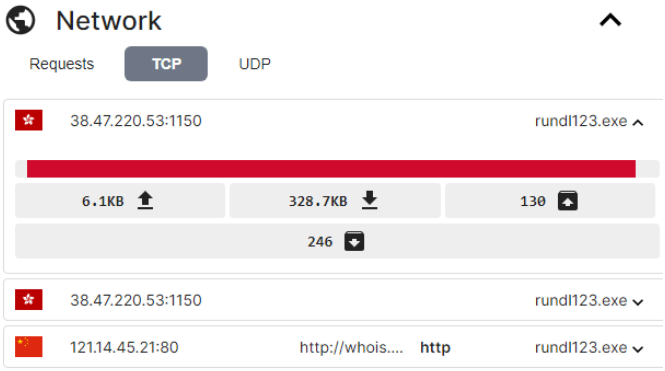
4.1.1.1. Analiza Emoteta u Triage sandboxu

Na online sandbox Triage [16] prenesen je prvi uzorak *malware*-a tipa Emotet koji je korišten i u 3. poglavlju. Ovaj uzorak pokrenut je u Windows 10 virtualnom stroju te je generirano izvješće o njegovoj aktivnosti. Na slici 4.1 možemo primjetiti da je inicijalna izvršna datoteka stvorila novi proces imena „rundl123.exe“. Ovo ime je smišljeno s namjerom da izgleda nalik na legitimnu biblioteku „rundll32.exe“, koja se pokreće kao proces dijete (engl. *child process*). Ova tehnika se zove prikrivanje (engl. *Masquerading*) te se često koristi kako bi analitičarima proces izgledao legitimno.



Sl. 4.1. Stablo procesa u Triage sandboxu

Sl. 4.2. Sadržaj mrežnih paketa komunikacije s nepoznatim serverom



Sl. 4.3. Popis mrežnih konekcija TCP protokolom

Stvoreni proces je poslao upit legitimnom DNS serveru za kako bi dobio IP adresu domene „whois.pconline.com.cn“. Ova domena izgleda legitimno, ali je u stvarnosti samo istog imena kao legitimni servis te je vršna domena u državi Kini. S tom adresom razmijenjeni su HTTP paketi (slika 4.2.). Proces je također razmijenio veliki broj paketa s IP adresom 38.47.220.53 na portu 1150 preko TCP (engl. *Transmission Control Protocol*) protokola (slika 4.3.).

Prikupljene informacije od Triage sandboxa su od velike vrijednosti te se uz pomoć njih može dublje analizirati ponašanje programa. Daljnja analiza bi mogla uključivati analizu paketa ili izvršnih funkcija datoteke „rundl123.exe“.

4.1.2. Snimanje prometa

Snimanje prometa (engl. *traffic capture*) je metoda koja omogućuje praćenje i snimanje mrežne komunikacije koju generira *malware* tijekom izvršavanja. Snimanje prometa omogućuje

sigurnosnim analitičarima da dobiju dublji uvid u komunikaciju između zaraženog sustava i udaljenih poslužitelja, čime se otkrivaju povezani C&C serveri, preuzimanje zlonamjernih datoteka i druge aktivnosti koje se odvijaju unutar mreže.

Postoje različite metode snimanja prometa koje se koriste u dinamičkoj analizi *malware*-a:

- Korištenje mrežnih alata: Mrežni alati poput Wiresharka, tcpdump-a ili TSharka mogu se koristiti za pasivno snimanje prometa na mrežnom sučelju zaraženog sustava. Ovi alati omogućuju analitičarima da snime sve mrežne pakete koji prolaze kroz sustav te ih detaljno analiziraju.
- Virtualna mrežna infrastruktura: Korištenje virtualnih mrežnih infrastruktura, poput virtualnih switcheva ili virtualnih mrežnih adaptera, omogućuje snimanje prometa između virtualnih strojeva ili između virtualnog stroja i vanjske mreže. Ova metoda omogućuje izoliranje prometa *malware*-a i detaljnu analizu njegovih mrežnih aktivnosti.
- Mrežni sandbox: Mrežni sandboxevi su posebna vrsta okruženja za izvršavanje i analizu *malware*-a. Oni pružaju funkcionalnosti snimanja prometa između *malware*-a i udaljenih poslužitelja te stvaraju detaljne izvještaje o mrežnoj aktivnosti zlonamjernog koda.

Snimanje prometa je široko primijenjena metoda analize jer može donijeti brojna saznanja o zloćudnom programu. Jedna takva informacija je identifikacija povezanih C&C poslužitelja. Snimanje prometa omogućuje otkrivanje C&C poslužitelja koji su povezani s *malware*-om. Analiza mrežne komunikacije otkriva adrese poslužitelja te pakete koji se razmjenjuju. Ove informacije kombinirane s popratnim ponašanjem tog procesa daju širu sliku o namjeri programa. Nadalje, snimanje prometa omogućuje otkrivanje novih varijanti *malware*-a koje se mogu razlikovati po komunikacijskom protokolu, enkripciji ili drugim karakteristikama. Ove informacije mogu biti ključne za učinkovitu detekciju. U slučaju višerazinskog *malware*-a je ovom metodom moguće zabilježiti preuzimanje dodatnih zlonamjernih datoteka s udaljenog poslužitelja. Ove datoteke je potrebno individualno analizirati te u budućim slučajima mogu služiti za prepoznavanje napada.

4.1.3. Debugiranje

Debuger (engl. *debugger*, program za otklanjanje pogrešaka) je alat koji se koristi za testiranje i analizu izvršavanja programa. Koristi se u razvoju jer programi često sadrže pogreške prilikom pisanja. Pri testiranju korisnik unosi ulazne podatke i promatra izlaz, ali ne može znati unutarnje mehanizme koji generiraju taj izlaz. Debugeri omogućuju praćenje rada programa tijekom

izvršavanja. Namijenjeni su programerima kako bi mogli pratiti i kontrolirati unutarnje stanje i izvođenje programa.

Debugeri pružaju informacije o programu koje je teško ili nemoguće dobiti obrnutim asembliranjem. Prevođenje strojnog koda u asemblerski daje uvid u izgled programa prije izvršavanja, dok debugeri omogućuju dinamički prikaz programa tijekom izvođenja. Na primjer, debugeri mogu prikazivati promjene vrijednosti memorijskih adresa tijekom izvršavanja programa.

Mogućnost mjerenja i kontroliranja izvođenja programa donosi bitne informacije prilikom analize zlonamjernog programa. Debugeri omogućuju uvid u vrijednosti memorijskih lokacija, registara i argumenata svake funkcije. Također je moguće promijeniti neke dijelove programa tijekom njegovog izvođenja kao što je na primjer vrijednost varijable u određenom trenutku. Ove funkcionalnosti možemo postići ako saznamo potrebne informacije o toj varijabli kao što je njezina lokacija u memoriji [4].

4.2. Tehnike dinamičke analize višerazinskog *malware*-a

Otkrivanje promjena koje je maliciozni program učinio na sustavu je esencijalni dio dinamičke analize. S obzirom na to da se u svakom trenutku na sustavu događa jako veliki broj promjena kao što su promjene u datotekama ili Windows registru, analiza promjena može biti izazovna. Sigurnosni analitičari moraju biti upoznati s bezazlenim aktivnostima u sustavu kako bi mogli otkriti potencijalne anomalije.

Postoji nekoliko tehnika na kojima se baziraju alati za dinamičku analizu, a to su [17]:

- Alati bazirani na hvatanju (engl. *hooking*) – Ovi alati hvataju sustavske API (engl. *Application Programming Interface*) pozive kako bi prikazali koje su promjene učinjene na sustavu. API pozivi su jedan od najboljih pokazatelja namjera programa jer je bez njih nemoguće izvesti radnje koje donose značajne promjene na sustavu. Primjer ovih alata su programi Process Monitor i pymon.py.
- Alati bazirani na razlici – Stvaranjem slike datotečnog sustava i Windows prije i nakon izvedbe programa, ovi alati mogu usporediti stanja sustava te prikazati promjene. Primjeri ovih alata su Regshot i inCtrl5.
- Obavještajni alati – Oni stvaraju rutine za obavještavanje koje sustav poziva kada se dogode određeni događaji. Takav događaj može biti stvaranje direktorija, gašenje procesa

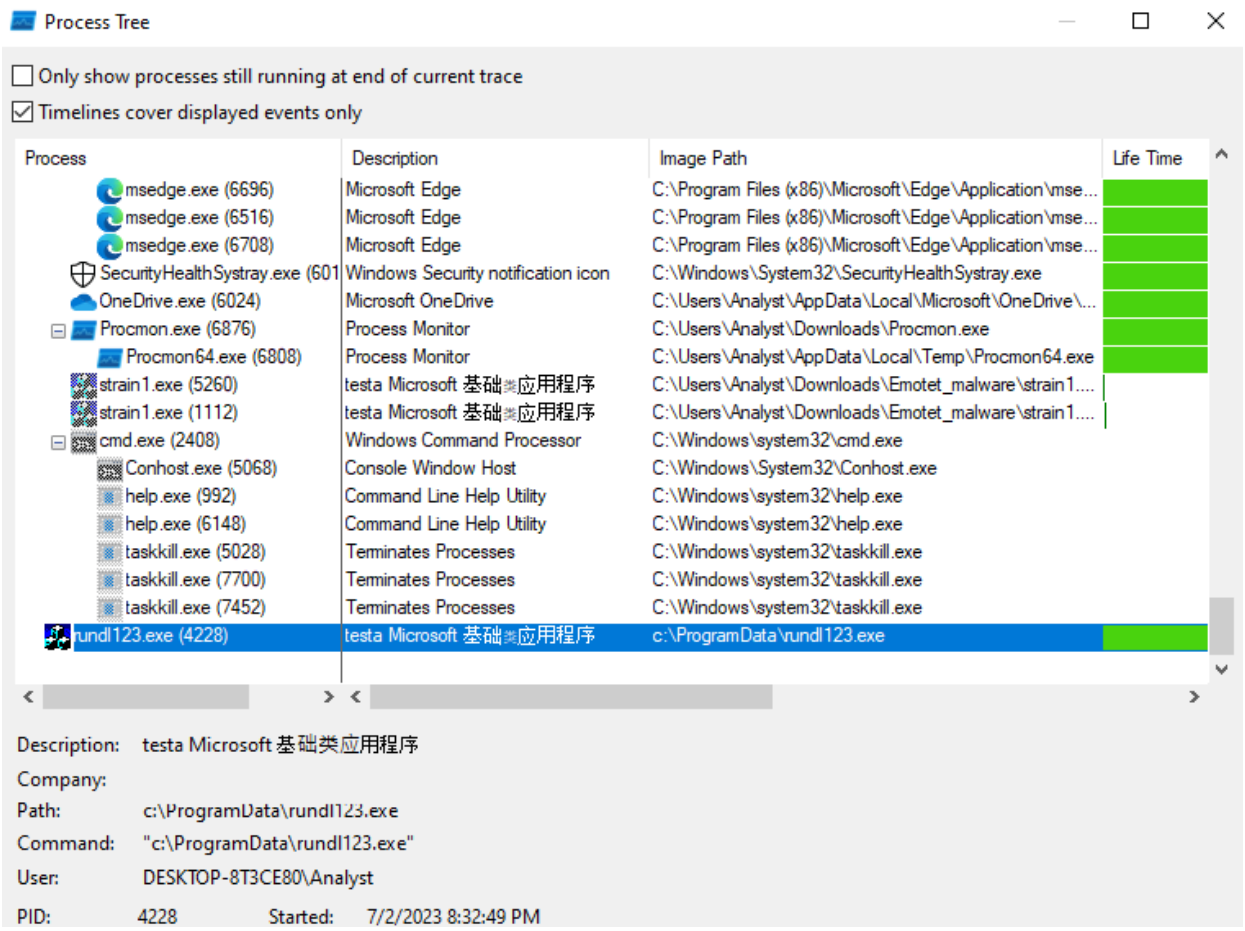
ili bilo koja druga promjena koju se može zabilježiti. Program Process Monitor može obavljati i ovu funkcionalnost, kao i program Preservation.

4.2.1. API pozivi

Kao što je već spomenuto, API pozivi su način za uspostavljanje komunikacije između aplikacija i biblioteka. Oni predstavljaju proceduru za izvršenje nekog zadatka. Ponekad su direktno vezani za neki sustavski poziv koji zahtjeva radnju od jezgre operacijskog sustava, a nekad mogu obaviti više sustavskih poziva. Najbitniji događaji u životnom ciklusu *malware*-a kao što su stvaranje procesa ili zauzeće memorije se moraju obaviti kroz API poziv operacijskom sustavu. Ove pozive možemo zabilježiti uz pomoć određenih programa te na temelju njih dublje razumjeti ponašanje promatranog procesa [17].

Process Monitor, skraćeno Procmon, je napredni alat za nadzor za Windows operacijski sustav koji se nalazi u skupini programa SysInternals Microsoft korporacije. Procmon prikazuje aktivnosti procesa i dretvi te promjene u Windows registru u stvarnom vremenu. Nastao je kao kombinacije dva starija alata iz SysInternals skupa alata: Filemon i Regmon koji su nadzirali samo datoteke ili samo registar pojedinačno. Procmon dodaje nove funkcionalnosti ovim programima kao što je filtriranje, dodatne informacije o događajima kao što su korisnička imena i identifikatori sesije te spremanje rezultata u datoteku. Koristi se kao esencijalan alat u analizi *malware*-a [17].

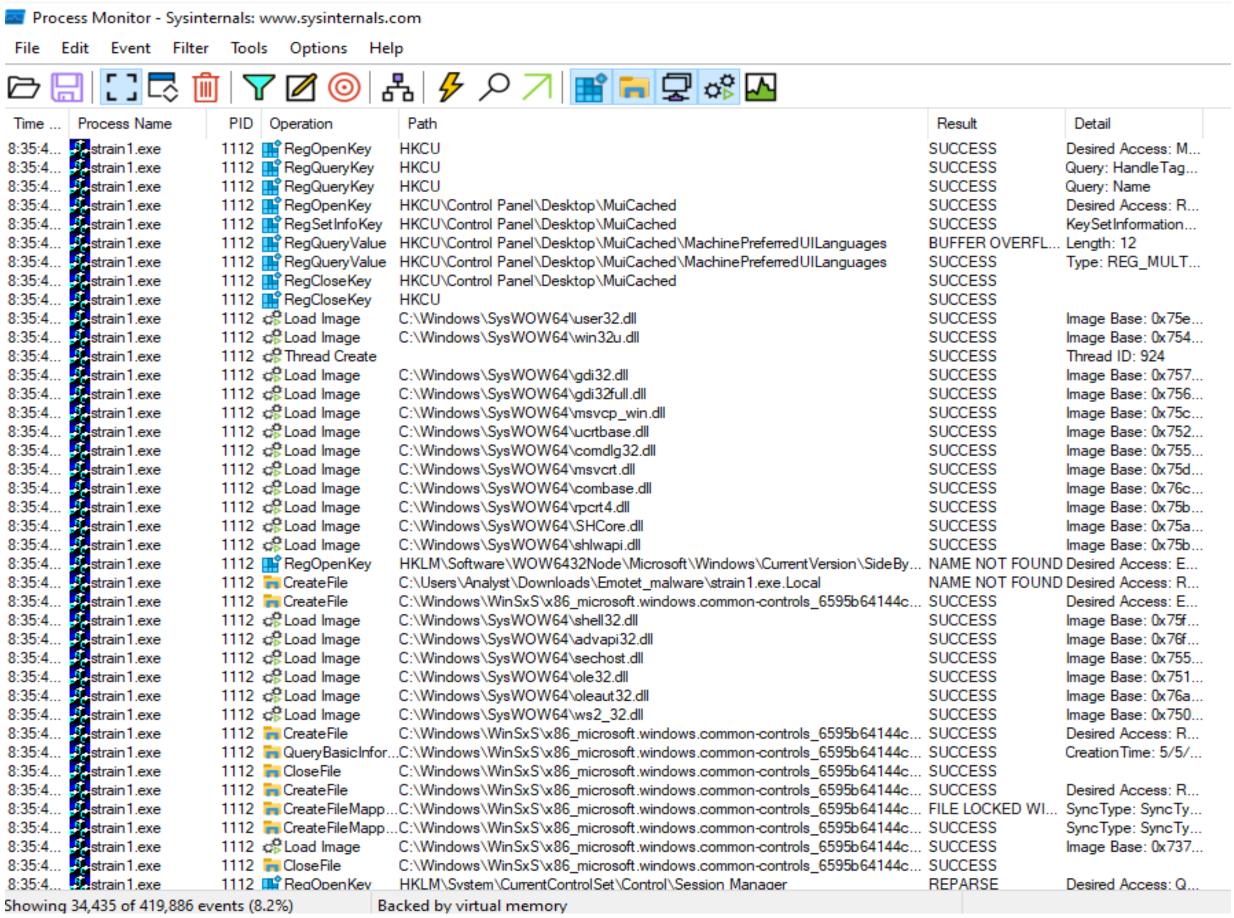
Kako bismo dublje razumjeli neku izvršnu datoteku, možemo ju pokrenuti dok Procmon nadzire računalo te pratiti njene radnje. U virtualnom stroju na kojemu je instaliran Windows 10 operacijski sustav pokrenut je prvi uzorak *malware*-a tipa Emotet koji je već korišten za generiranje YARA pravila u trećem poglavlju. U svrhu zaštite stvarnog računala i mreže, virtualnom stroju je uskraćen pristup Internetskoj vezi.



Sl. 4.4. Ispis procesnog stabla programa Procmon

Prema slici 4.4 možemo primijetiti da se pokrenuti uzorak kratko zadržao na sustavu kao inicijalni proces identifikacijskog broja (skraćeno ID) 5260, ali je pokrenuo novi proces ID-a 1112 koji je obavljao dodatne operacije. Sam naziv procesa je zavaravajući jer spominje riječ Microsoft kako bi dobio kredibilitet, ali i diže sumnju zbog korištenja nestandardnih znakova.

Na slici 4.5 možemo vidjeti izlistanje samo manjeg dijela radnji koje je napravio proces sa ID vrijednosti 1112. Stvorene su brojne datoteke i direktoriji te učitani mnogi DLL-ovi. Provedeno je puno provjera vrijednosti određenih unosa i Windows registru te su mnoge vrijednosti izmijenjene.



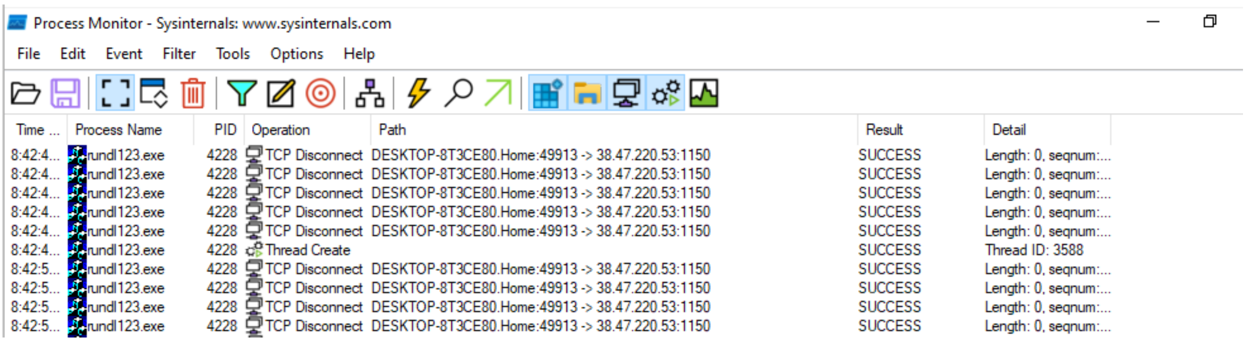
Process Monitor - Sysinternals: www.sysinternals.com

File Edit Event Filter Tools Options Help

Time ...	Process Name	PID	Operation	Path	Result	Detail
8:35:4...	strain1.exe	1112	RegOpenKey	HKCU	SUCCESS	Desired Access: M...
8:35:4...	strain1.exe	1112	RegQueryKey	HKCU	SUCCESS	Query: HandleTag...
8:35:4...	strain1.exe	1112	RegQueryKey	HKCU	SUCCESS	Query: Name
8:35:4...	strain1.exe	1112	RegOpenKey	HKCU\Control Panel\Desktop\MuiCached	SUCCESS	Desired Access: R...
8:35:4...	strain1.exe	1112	RegSetInfoKey	HKCU\Control Panel\Desktop\MuiCached	SUCCESS	KeySetInformation...
8:35:4...	strain1.exe	1112	RegQueryValue	HKCU\Control Panel\Desktop\MuiCached\MachinePreferredUILanguages	BUFFER OVERFL...	Length: 12
8:35:4...	strain1.exe	1112	RegQueryValue	HKCU\Control Panel\Desktop\MuiCached\MachinePreferredUILanguages	SUCCESS	Type: REG_MULT...
8:35:4...	strain1.exe	1112	RegCloseKey	HKCU\Control Panel\Desktop\MuiCached	SUCCESS	
8:35:4...	strain1.exe	1112	RegCloseKey	HKCU	SUCCESS	
8:35:4...	strain1.exe	1112	Load Image	C:\Windows\SysWOW64\user32.dll	SUCCESS	Image Base: 0x75e...
8:35:4...	strain1.exe	1112	Load Image	C:\Windows\SysWOW64\win32u.dll	SUCCESS	Image Base: 0x754...
8:35:4...	strain1.exe	1112	Thread Create		SUCCESS	Thread ID: 924
8:35:4...	strain1.exe	1112	Load Image	C:\Windows\SysWOW64\gdi32.dll	SUCCESS	Image Base: 0x757...
8:35:4...	strain1.exe	1112	Load Image	C:\Windows\SysWOW64\gdi32full.dll	SUCCESS	Image Base: 0x756...
8:35:4...	strain1.exe	1112	Load Image	C:\Windows\SysWOW64\msvcp_win.dll	SUCCESS	Image Base: 0x75c...
8:35:4...	strain1.exe	1112	Load Image	C:\Windows\SysWOW64\ucrtbase.dll	SUCCESS	Image Base: 0x752...
8:35:4...	strain1.exe	1112	Load Image	C:\Windows\SysWOW64\comdlg32.dll	SUCCESS	Image Base: 0x755...
8:35:4...	strain1.exe	1112	Load Image	C:\Windows\SysWOW64\msvcrt.dll	SUCCESS	Image Base: 0x75d...
8:35:4...	strain1.exe	1112	Load Image	C:\Windows\SysWOW64\combase.dll	SUCCESS	Image Base: 0x76c...
8:35:4...	strain1.exe	1112	Load Image	C:\Windows\SysWOW64\rpcrt4.dll	SUCCESS	Image Base: 0x75b...
8:35:4...	strain1.exe	1112	Load Image	C:\Windows\SysWOW64\SHCore.dll	SUCCESS	Image Base: 0x75a...
8:35:4...	strain1.exe	1112	Load Image	C:\Windows\SysWOW64\shlwapi.dll	SUCCESS	Image Base: 0x75b...
8:35:4...	strain1.exe	1112	RegOpenKey	HKLM\Software\WOW6432Node\Microsoft\Windows\CurrentVersion\SideBy...	NAME NOT FOUND	Desired Access: E...
8:35:4...	strain1.exe	1112	CreateFile	C:\Users\Analyst\Downloads\Emotet_malware\strain1.exe.Local	NAME NOT FOUND	Desired Access: R...
8:35:4...	strain1.exe	1112	CreateFile	C:\Windows\WinSxS\x86_microsoft.windows.common-controls_6595b64144c...	SUCCESS	Desired Access: E...
8:35:4...	strain1.exe	1112	Load Image	C:\Windows\SysWOW64\shell32.dll	SUCCESS	Image Base: 0x75f...
8:35:4...	strain1.exe	1112	Load Image	C:\Windows\SysWOW64\advapi32.dll	SUCCESS	Image Base: 0x76f...
8:35:4...	strain1.exe	1112	Load Image	C:\Windows\SysWOW64\sechost.dll	SUCCESS	Image Base: 0x755...
8:35:4...	strain1.exe	1112	Load Image	C:\Windows\SysWOW64\ole32.dll	SUCCESS	Image Base: 0x751...
8:35:4...	strain1.exe	1112	Load Image	C:\Windows\SysWOW64\oleaut32.dll	SUCCESS	Image Base: 0x76a...
8:35:4...	strain1.exe	1112	Load Image	C:\Windows\SysWOW64\ws2_32.dll	SUCCESS	Image Base: 0x750...
8:35:4...	strain1.exe	1112	CreateFile	C:\Windows\WinSxS\x86_microsoft.windows.common-controls_6595b64144c...	SUCCESS	Desired Access: R...
8:35:4...	strain1.exe	1112	QueryBasicInfor...	C:\Windows\WinSxS\x86_microsoft.windows.common-controls_6595b64144c...	SUCCESS	CreationTime: 5/5/...
8:35:4...	strain1.exe	1112	CloseFile	C:\Windows\WinSxS\x86_microsoft.windows.common-controls_6595b64144c...	SUCCESS	
8:35:4...	strain1.exe	1112	CreateFile	C:\Windows\WinSxS\x86_microsoft.windows.common-controls_6595b64144c...	SUCCESS	Desired Access: R...
8:35:4...	strain1.exe	1112	CreateFileMapp...	C:\Windows\WinSxS\x86_microsoft.windows.common-controls_6595b64144c...	FILE LOCKED WI...	SyncType: SyncTy...
8:35:4...	strain1.exe	1112	CreateFileMapp...	C:\Windows\WinSxS\x86_microsoft.windows.common-controls_6595b64144c...	SUCCESS	SyncType: SyncTy...
8:35:4...	strain1.exe	1112	Load Image	C:\Windows\WinSxS\x86_microsoft.windows.common-controls_6595b64144c...	SUCCESS	Image Base: 0x737...
8:35:4...	strain1.exe	1112	CloseFile	C:\Windows\WinSxS\x86_microsoft.windows.common-controls_6595b64144c...	SUCCESS	
8:35:4...	strain1.exe	1112	RegOpenKey	HKLM\System\CurrentControlSet\Control\Session Manager	REPARSE	Desired Access: Q...

Showing 34,435 of 419,886 events (8.2%) Backed by virtual memory

Sl. 4.5. Prikaz programa Procmon za pokrenuti Emotet uzorak



Process Monitor - Sysinternals: www.sysinternals.com

File Edit Event Filter Tools Options Help

Time ...	Process Name	PID	Operation	Path	Result	Detail
8:42:4...	rundl123.exe	4228	TCP Disconnect	DESKTOP-8T3CE80.Home:49913 -> 38.47.220.53:1150	SUCCESS	Length: 0, seqnum:...
8:42:4...	rundl123.exe	4228	TCP Disconnect	DESKTOP-8T3CE80.Home:49913 -> 38.47.220.53:1150	SUCCESS	Length: 0, seqnum:...
8:42:4...	rundl123.exe	4228	TCP Disconnect	DESKTOP-8T3CE80.Home:49913 -> 38.47.220.53:1150	SUCCESS	Length: 0, seqnum:...
8:42:4...	rundl123.exe	4228	TCP Disconnect	DESKTOP-8T3CE80.Home:49913 -> 38.47.220.53:1150	SUCCESS	Length: 0, seqnum:...
8:42:4...	rundl123.exe	4228	TCP Disconnect	DESKTOP-8T3CE80.Home:49913 -> 38.47.220.53:1150	SUCCESS	Length: 0, seqnum:...
8:42:4...	rundl123.exe	4228	Thread Create		SUCCESS	Thread ID: 3588
8:42:5...	rundl123.exe	4228	TCP Disconnect	DESKTOP-8T3CE80.Home:49913 -> 38.47.220.53:1150	SUCCESS	Length: 0, seqnum:...
8:42:5...	rundl123.exe	4228	TCP Disconnect	DESKTOP-8T3CE80.Home:49913 -> 38.47.220.53:1150	SUCCESS	Length: 0, seqnum:...
8:42:5...	rundl123.exe	4228	TCP Disconnect	DESKTOP-8T3CE80.Home:49913 -> 38.47.220.53:1150	SUCCESS	Length: 0, seqnum:...
8:42:5...	rundl123.exe	4228	TCP Disconnect	DESKTOP-8T3CE80.Home:49913 -> 38.47.220.53:1150	SUCCESS	Length: 0, seqnum:...

Sl. 4.6. Prikaz programa Procmon za pokrenuti Emotet uzorak

Na slici 4.4 možemo primijetiti da je stvoren još jedan proces od strane ovog *malware*-a, a to je proces imena „rundl123.exe“. Ova izvršna datoteka je bila zadužena za ostvarivanje veze sa C&C serverom od kojega je program vjerojatno trebao primiti daljnje instrukcije ili potrebne datoteke. Prema slici 4.6. možemo vidjeti da se proces pokušao spojiti na adresu 38.47.220.53:1150 koristeći TCP protokol, ali nije uspio zbog izoliranosti sustava. Ovi pokušaji su se nastavili svakih nekoliko sekundi.

4.2.2. Detekcija promjene prema razlici

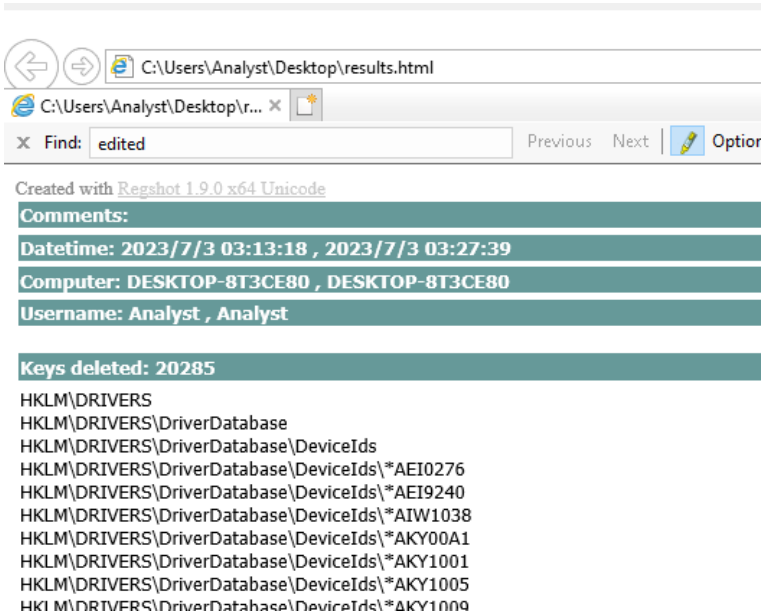
Alati bazirani na razlici između dvije snimke u vremenu mogu prikazati koje su se promjene dogodile u sustavu. Jednostavniji su od alata koji rade u stvarnom vremenu poput Procmon-a te su najčešće puno jednostavniji za korištenje. Njihov generirani sadržaj je prikazan u obliku izvješća te navodi sve promjene koje su se dogodile u datotečnom sustavu ili u registru.

Ovi programi imaju nekoliko nedostataka u usporedbi sa programima koji hvataju API pozive, a neki od njih su [17]:

- Ne mogu detektirati privremene datoteke. Pošto se baziraju samo na razlici, bilo koja datoteka stvorena te kasnije izbrisana prije uzimanja druge snimke sustava, neće biti zabilježena. Ovo predstavlja značajan problem kod analize *malware*-a jer on često preuzme datoteke te ih izbriše nakon korištenja.
- Ne prate vremenske oznake promjena u sustavu. Kod ovih alata nije moguće znati redoslijed izvršavanja radnji koje su zabilježene, što nekada može otežati razumijevanje ponašanja određenog programa.

Jedan od programa koji radi na ovom principu je Regshot. Regshot je besplatan program otvorenog koda. Sam po sebi je jednostavan te omogućava snimanje potpunog stanja registara u dva različita vremenska trenutka te pruža izvješće o njihovoj razlici. Regshot nudi jednostavno sučelje te možemo birati koje direktorije ćemo pratiti. Također se nudi opcija spremanja rezultata u običnom tekstualnom ili u formatu za web preglednik (napisan u označnom jeziku HTML). Prate se registarske vrijednosti koje su obrisane, one koje su izmijenjene te datoteke koje su stvorene ili izmijenjene [18].

Pomoću ovog programa napravljena je snimka sustava prije i nakon pokretanja uzorka Emotet *malware*-a te izgled rezultata u HTML obliku možemo vidjeti na slici 4.7. Možemo primijetiti da u tom kratkom razdoblju pokretanja programa se dogodilo preko 20000 brisanja vrijednosti nekog registra. Broj datoteka koje su izmijenjena je također bio jako visok. S obzirom na to da alati bazirani na razlici ne prate tko je učinio ove promjene, teško je znati koje radnje je učinio *malware*, a koje su legitimne i od strane sustava. Stoga je ovaj alat lošiji za koristiti ukoliko ne implementiramo vlastiti sustav koji će pratiti samo određene vrijednosti u svrhu nekakve preliminarne procjene stanja sustava.



Sl. 4.7. Regshot datoteka s rezultatima

4.3. Dinamička detekcija

S razvojem sve sofisticiranijeg *malware*-a koji svoje fizičke karakteristike vješto skriva, u proces detekcije bilo je nužno uključiti faktor koji je teško zamaskirati, a to je samo ponašanje. Dinamička detekcija je način klasifikacije opasnosti na temelju radnji koje program obavlja. Bihevioralnom analizom dolazimo do podataka o aktivnostima zloćudnih programa te na temelju njih možemo razvijati sustave koji će, po primitku podataka u istom formatu, donijeti odluku o tome je li neki proces maliciozan. S obzirom na to da se ovi podaci ne mogu prikupiti odjednom, već se prikupljaju kroz životni programa, potrebno je postaviti određeni nadzor nad sustavom. Ovi sustavi sakupljaju zapise događaja (engl. *log*) sa svih krajnjih točaka u mreži koju nadziru te ih često integriraju sa SIEM (engl. *Security Information and Event* Management) sustavom.

Sigurnosno upravljanje informacijama i događajima (SIEM) je tehnologija kibernetičke sigurnosti koja pruža pojednostavljen prikaz podataka, uvid u sigurnosne aktivnosti i operativne mogućnosti kako bi se mogli učinkovito otkrivati, istraživati i reagirati na sigurnosne prijetnje. SIEM rješenje može ojačati kibernetičku sigurnost pružajući potpunu vidljivost distribuiranog okruženja u stvarnom vremenu. Može se primijeniti na lokalnim, hibridnim ili okuženjima u oblaku. Kako bi se otkrilo prijetnje i druge nepravilnosti, SIEM rješenje prikuplja i pregledava velike količine podataka u vrlo kratkom vremenu kako bi upozorilo na neobično ponašanje. Analiza tolike količine podataka ne bi bila moguća bez automatizacije koju pruža SIEM [19].

4.3.1. Razvoj Sigma detekcijskih pravila

Sigma pravila su tektualni zapisi u YAML (engl. *Yet Another Markup Language*) označnom jeziku koji omogućuju detekciju anomalija u sustavu promatranjem zapisa o događajima u sustavu. Razvili su ih analitičari Florian Roth i Thomas Patzke te se koriste u SIEM sustavima za opis prijetnji. Glavna prednosti korištenja standardiziranog formata pravila je da su neovisna o platformi te se mogu primijeniti na svim SIEM proizvodima. Kao takav pruža sigurnosnim stručnjacima zajednički jezik za dijeljenje pravila neovisno o specifičnom programu koji se koristi za detekciju. Ova pravila se mogu konvertirati jezike specifične za određeni proizvod, ali zadržavajući istu logiku detekcije kao originalno Sigma pravilo. Dok se YARA pravila češće koriste za identifikaciju i klasifikaciju zlćudnih programa koristeći njihove karakteristike, Sigma pravila su usredotočena na pronalazak zapisa o događajima koji se podudaraju s pravilom.

Pravila sadržavaju različite oznake kojima se pridodaju vrijednosti. Najvažnije su oznake vezane za podatke o pravilu kao što su naslov (`title`) i opis (`description`), detekcijske uvjete (`detection`) te uvjete za okidanje pravila (`condition`). Prema podacima prikupljenim iz Triage sandboxa, na temelju specifičnog redoslijeda stvaranja procesa, napisano je Sigma pravilo za prepoznavanje ovog slijeda događaja iz zapisa (slika 4.8). Stvorena su 2 uvjeta za selekciju u pravilu. Prvo uvjet provjerava postoji li roditeljski proces u Temp direktoriju Windows sustava i je li stvorio proces koji je imena „rundl*“, što znači da mora počinjati s „rundl“, ali može imati bilo kakve znakove nakon toga. Ovo je implementirano u slučaju da se naziv promjeni u drugim instancama. Drugi detekcijski uvjet je to da je stvoren proces biblioteke „rundll32.exe“ koja je legitiman, ali ju je stvorio proces koji nosi slično ime što upućuje na maskiranje radnji te izvršavanje naredbi preko posrednika.

Izvršavanje naredbi preko posrednika je bitan aspekt kamuflaže ovog *malware*-a te je stoga dobar pokazatelj za detekciju. Ova tehnika se često primjenjuje sa bibliotekom „rundll32.exe“. Napadači koriste ovaj DLL jer tako mogu zaobići sigurnosne mjere koje možda ne nadziru taj proces zbog lažno pozitivnih detekcija [20].

Linija ***Kod***

```
1:      title: Emotet malware detection based on process creation
        description: Detects emotet malware based on the behaviour of
2:      starting malware from Windows Temp folder which then executes
3:      another executable that starts rundll32.exe.
4:      status: experimental
5:      date: 2023/07/03
6:      author: zagar-matija
7:      references:
8:          - https://tria.ge/230703-plfzjagd37/behavioral1#report
9:      logsource:
10:         category: process_creation
11:         product: windows
12:     detection:
13:         selection1:
14:             ParentImage:
15:               - 'C:\Users\*\AppData\Local\Temp\*.exe'
16:            Image:
17:          - '*rundl1*.exe'
18:         Selection2:
19:            ParentImage:
20:               - '*rundl1*.exe
21:            Image:
22:               - '*rundll32.exe'
23:         condition: selection1 or selection2
24:     falsepositives:
25:         - Unlikely
26:     level: mid
27:     tags:
28:         - attack.T1036.005
29:         - attack.T1218.011
30:
```

Sl. 4.8. Sigma pravilo za detekciju prvog uzorka Emotet *malware*-a

5. ZAKLJUČAK

Višestupanjski ili višerazinski *malware* predstavlja veliku prijetnju te je potrebno uložiti puno truda u istraživanje i razvoj sigurnosnih rješenja za suzbijanje njegovog širenja. Kao pristupi analizi ovih programa nude se statički i dinamički.

Statički analiza nudi sigurnost pri promatranju jer ne zahtjeva pokretanje te je vrlo učinkovita za stvaranje karakterističnih popisa za malware kako bi se isti uzorci mogli detektirati u budućnosti. Uz potpise, mogu se provoditi analize strukture datoteke koja se pokazala kao efikasan način za klasifikaciju te izvlačiti nizovi znakova koji nose značenje te su bitni za maliciozne funkcionalnosti programa. Koristeći alat za automatsko generiranje detekcijskih pravila yarGen razvijena su 2 rješenja za detekciju Emotet *malware*-a u obliku YARA detekcijskih pravila. Prvo pravilo je bilo uspješno u detekciji samo za svoj uzorak dok je drugo pravilo postiglo velik broj detekcija iz baze *malware*-a na stranici MalwareBazaar. Prvo pravilo je neuspješno jer je koristilo previše specifične značajke uzorka koje se lako promjene kao što su IP adrese. Iz toga je vidljivo da pravila mogu biti više ili manje uspješna, ali svakodnevnu uporabu ne određuje samo broj istinitih pozitivnih detekcija nego i broj neistinito pozitivnih detekcija koje nisu mjerene u ovom radu. Ovaj pristup je najčešće neuspješan u detekciji višerazinskog *malware*-a jer se maliciozni kod naknadno dostavi programu te nije dostupan bez pokretanja samog *malware*-a.

Dinamička analiza nudi metode za analizu ponašanja sumnjivih programa. Koristeći virtualna izolirana okruženja, analitičari mogu kontrolirati sustav u kojemu se *malware*-a pokreće te motriti sve njegove radnje. Podaci koji se prikupljaju su najčešće popis poziva funkcija koje izvršna datoteka koristi te njena mrežna komunikacija. Za promatranje ovih radnji se koristio program Procmon i integrirano okruženje za promatranje unutar Triage *sandbox*-a. Detekcija prema ponašanju programa može se opisati Sigma pravilima za detekciju. Ova pravila predstavljaju univerzalni jezik koji se može primijeniti na specifične SIEM alate. Razvijeno je Sigma pravilo za detekciju Emotet *malware*-a se temelji na procesnom stablu malicioznog procesa. Pri tome se za detekciju oslanjamo na detekciju tehnike maskiranja imena programa kao legitimnog DLL-a na Windows operacijskom sustavu te izvršavanju koda preko posrednika čime se izbjegava detekcija.

Oba pristupa su uspješna u detekciji, ali za veću preciznost i točnost potrebno je koristiti oba. Korištenje jednog ili drugog ovisi o znanju analitičara kao i o dostupnosti alata u nekom okruženju. Uočeno je da se dinamičkom analizom može lakše doći do određenih značajki kao i

do sumnjivih ponašanja malicioznog koda. Nasuprot tome, YARA kao standard nam je omogućila automatizirano generiranje pravila i može pokazati svoje prednosti u slučaju da se *malware* ne želi izvršiti unutar izolirane okoline ili ako se napravi vrlo detaljna statička analiza malicioznog koda

LITERATURA

[1] M. Ramilli i M. Bishop, »Multi-Stage Delivery of Malware,« 2010.

[2] P. Szor, The Art of Computer Virus Research and Defense, Upper Saddle River: Addison-Wesley, 2005..

[3] A. Fedák i J. Štulrajter, »Fundamentals Of Static Malware Analysis: Principles, Methods And Tools,« *Science & Military,* pp. 49-51, Siječanj 2020.

[4] M. Sikorski i A. Honig, Practical malware analysis, San Francisco: No Starch Press, 2012..

[5] R. C., »atlasVPN,« [Mrežno]. Available: https://atlasvpn.com/blog/over-95-of-all-new-malware-threats-discovered-in-2022-are-aimed-at-windows. [Pokušaj pristupa 21. 6. 2023].

[6] T. Rezaei i A. Hamze, »An Efficient Approach For Malware Detection Using PE Header Specifications,« Shiraz University, Shiraz, rad s konferencije, 2020.

[7] J. Sahs i L. Khan, »A Machine Learning Approach to Android Malware,« Dallas TX, rad s konferencije "2012 European Intelligence and Security Informatics Conference", 2012..

[8] »YARA rules documentation,« [Mrežno]. Available: https://yara.readthedocs.io/en/stable/index.html. [Pokušaj pristupa 24 lipanj 2023].

[9] V. Li, »Malware detection using YARA and YarGen,« [Mrežno]. Available: https://sec.okta.com/articles/2021/08/malware-detection-using-yara-and-yargen. [Pokušaj pristupa 27 6 2023].

[10] F. Roth, »YarGen dokumentacija,« [Mrežno]. Available: https://github.com/Neo23x0/yarGen. [Pokušaj pristupa 27 6 2023].

[11] »MalwareBazaar,« Abuse.ch, [Mrežno]. Available: https://bazaar.abuse.ch/. [Pokušaj pristupa 29 Lipanj 2023].

[12] A. Stewart, »DLL Side-loading: A Thorn in the Side of the Anti-Virus Industry,« FireEye, Inc, Milpitas, 2014.

[13] C. Camichel, »Yara scan service,« Risk Mitigation, [Mrežno]. Available: https://riskmitigation.ch/yara-scan/. [Pokušaj pristupa 30 Lipanj 2023].

[14] O. Or-Meir, N. Nissim, Y. Elovici i L. Rokach, »Dynamic Malware Analysis in the Modern Era - A State of the art survey,« *ACM Computing Surveys,* Rujan 2019.

[15] C. Kolbitsch, »Evasive Malware Tricks: How Malware Evades Detection by Sandboxes,« 1 11 2017. [Mrežno]. Available: https://www.isaca.org/resources/isaca-journal/issues/2017/volume-6/evasive-malware-tricks-how-malware-evades-detection-by-sandboxes. [Pokušaj pristupa 1 7 2023].

[16] »Triage,« Recorded future, [Mrežno]. Available: https://tria.ge/. [Pokušaj pristupa 3 Srpanj 2023].

[17] M. H. Ligh, S. Adair, B. Hartstein i M. Richard, Malware Analyst's Cookbook, Indianapolis: Wiley Publishing, Inc, 2012.

[18] M. Hendrikx, »How to Use Regshot To Monitor Your Registry,« How-To Geek, [Mrežno]. Available: https://www.howtogeek.com/198679/how-to-use-regshot-to-monitor-your-registry/. [Pokušaj pristupa 1 srpanj 2023].

[19] »What is SIEM,« Splunk, 1 Kolovoz 2022. [Mrežno]. Available: https://www.splunk.com/en_us/data-insider/what-is-siem.html. [Pokušaj pristupa 2 Svibanj 2023].

[20] »System Binary Proxy Execution: Rundll32,« Mitre , [Mrežno]. Available: https://attack.mitre.org/techniques/T1218/011/. [Pokušaj pristupa 3 Srpanj 2023].

[21] S. Kim, »PE Header Analysis for Malware Detection,« San Jose State University , San Jose, 2018.

[22] M. Carpenter i C. Luo, »Behavioural Reports of Multi-Stage Malware«.*IEEE Transactions on Sustainable Computing.*

Sažetak

Višerazinski *malware* predstavlja veliku prijetnju te zahtjeva veliki napor za razvoj sigurnosnih rješenja za suzbijanje njegovog širenja. U ovom radu opisan je višerazinski malware te Emotet *malware* koji je ovog tipa. Dan je pregled metoda i alata dva pristupa analizi i detekciji. Opisane su metode statičkog pristupa analizi koja obuhvaća analizu karakteristika izvršne datoteke bez izvršavanja. Razvijena su YARA detekcijska pravila za statičku detekciju Emotet *malware*-a koja se temelje na pronalasku karakterističnih slijedova znakova u binarnom zapisu datoteke. Dan je pregled tehnika dinamičke analize *malware*-a kao što su analiza API poziva i mrežne komunikacije. Opisani su alati korišteni za ponašajnu analizu i njihovo korištenje. Razvijeno je Sigma pravilo za otkrivanje Emotet-a na temelju procesnog stabla. Opisana su područja primjene te prednosti i nedostaci oba pristupa.

Ključne riječi: statička analiza, dinamička analiza, kibernetička sigurnost, *malware*, Emotet

Detection and analysis of multi-stage malware

Abstract

Multi-stage malware represents a large threat and requires substantial effort in developing security solutions for combating its spread. This paper describes multi-stage malware and Emotet malware, which falls into this category. An overview of methods and tools for two different approaches to analysis and detection is given. A static approach to malware analysis is described, which involves analysis of a binary file without execution. YARA detection rules based on string matching are developed for static detection of Emotet. Techniques for dynamic analysis of malware are discussed, such as the analysis of API calls and network communication. Tools used for behavioral analysis and their utilization is described. A Sigma rule for detecting Emotet based on the process creation tree is developed. The advantages and disadvantages of both approaches are discussed, along with their respective target applications.

Keywords: static analysis, dynamic analysis, cybersecurity, malware, Emotet

ŽIVOTOPIS

Matija Žagar je rođen 5.8.2001. u Vinkovcima. Osnovnu školu pohađao je u osnovnoj školi Bartola Kašića u Vinkovcima. Nakon osnovne škole, upisao je smjer opće gimnazije u gimnaziji Matije Antuna Reljkovića u Vinkovcima. Akademsku izobrazbu nastavlja na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija Osijek gdje redovno studira na smjeru Računarstvo – izborni blok računalno inženjerstvo. Tijekom studija aktivno sudjeluje u Nastavi kao demonstrator te sudjeluje u radu studentskog zbora fakulteta. Na temelju ocjena primio je priznanje za uspješnost u studiranju 2022. godine.

_____

Potpis autora