

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 1207

**DETEKCIJA KIBERNETIČKIH NAPADA I ZAŠTITA VANJSKIH
SUSTAVA U KONTEKSTU MAMACA ZA OPERATORA
PRIJENOSNOG SUSTAVA**

Filip Šimičević

Zagreb, lipanj 2023.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 1207

**DETEKCIJA KIBERNETIČKIH NAPADA I ZAŠTITA VANJSKIH
SUSTAVA U KONTEKSTU MAMACA ZA OPERATORA
PRIJENOSNOG SUSTAVA**

Filip Šimičević

Zagreb, lipanj 2023.

Zagreb, 10. ožujka 2023.

ZAVRŠNI ZADATAK br. 1207

Pristupnik: **Filip Šimičević (0036532540)**

Studij: Elektrotehnika i informacijska tehnologija i Računarstvo

Modul: Računarstvo

Mentor: izv. prof. dr. sc. Stjepan Groš

Zadatak: **Detekcija kibernetičkih napada i zaštita vanjskih sustava u kontekstu mamaca za operatora prijenosnog sustava**

Opis zadatka:

Mamci imitiraju stvarne uređaje i sustave u računalnoj mreži kako bi privukli napadače. Kada napadači kompromitiraju mamac, moguće je proučavati njihovo djelovanje i prikupljati spoznaje o alatima i tehnikama koje koriste. Tako dobivene spoznaje mogu pomoći u povećanju razine sigurnosti stvarnih sustava. U sklopu ranijeg projekta razvijena je jezgra prototipa mamca koja imitira dio mreže operatora prijenosnog sustava. Navedeni prototip potrebno je dovršiti kako bi ga se moglo koristiti u daljnjim eksperimentima. U sklopu završnoga rada potrebno je istražiti javno dostupne alate za detekciju napada i metode logiranja koje se mogu koristiti u sklopu mamca. Na temelju pronađenih informacija potrebno je proširiti postojeći prototip mamca kako bi se pomoću njega moglo detektirati napade i bilježiti aktivnosti napadača. Nadalje, potrebno je postaviti konfiguraciju vatrozida koja će napadaču koji ostvari pristup mamcu onemogućiti napadanje vanjskih sustava, i postaviti VPN kojim će biti moguće pristupiti mamcu s vanjske mreže. Radu priložiti izvorni kôd. Citirati korištenu literaturu i navesti dobivenu pomoć.

Rok za predaju rada: 9. lipnja 2023.

SADRŽAJ

1. Uvod	1
2. Operatori prijenosnih sustava	2
2.1. Industrijski upravljači sustavi	2
2.2. PLC sustavi	2
2.2.1. Programski jezici	3
2.3. SCADA sustavi	3
2.3.1. Komponente	5
2.3.2. Ranjivosti SCADA sustava	6
2.4. Najčešće prijetnje	6
2.5. Zaštite koje se primjenjuju	7
2.6. OSINT nad operatorom prijenosnih sustava HEP	8
2.6.1. OSINT alati	8
3. Docker	12
3.1. Docker slika	12
3.1.1. Dockerfile	13
3.2. Docker kontejner	13
4. IMUNES	14
4.1. Topologija mreže	14
5. Mamac	16
5.0.1. Vrste mamaca	17
5.0.2. Mamci u kontekstu operatora prijenosnih sustava	18
5.1. TSO mamac izveden u IMUNES-u	18
5.1.1. VPN	21
5.1.2. Vatrozid	26
5.1.3. Skupljanje prometa i IDS (engl. <i>Intrusion Detection System</i>)	27

5.1.4. Log poslužitelj	28
6. Zaključak	32
Literatura	33

1. Uvod

Zbog napretka tehnologije pa tako i željom za pronalaženje boljih rješenja po pitanju održavanja električne mreže, operatori prijenosnih sustava se svakim danom razvijaju i koriste razne nove tehnologije i automatizacijske procese kako bi nam olakšali svakodnevni život. Time su uveli nove ranjivosti i rizike, posebno u području kibernetičke sigurnosti. Kibernetički napadi postali su značajna briga, prijeteci integritetu, dostupnosti i povjerljivosti osjetljivih informacija i kritične infrastrukture. Kako bi se osigurali od materijalne ili fizičke štete operatori prijenosnih sustava trebaju primjenjivati veće mjere zaštite i koristiti tehnologije koje pružaju optimalna rješenja za takve probleme. U ovom radu detaljno je obrađena zaštita koristeći koncept mamaca koji su samo jedan od temelja sigurnosti koji bi trebali biti implementirani u ovim sustavima. Implementacijom ovog sustava, napadači mogu naići na naizgled ranjivu mašinu koja se lako podvija njihovim zahtjevima no u stvarnosti u najmanju ruku troše vrijeme koje bi koristili za napad na pravi sustav. Također se mogu prikupljati dragocjeni podaci o samoj osobi koja stoji iza malicioznih radnji.

2. Operatori prijenosnih sustava

Uloge Operatora prijenosnih sustava (OPS) na tržištu električne energije uključuju upravljanje sigurnošću elektroenergetskog sustava u stvarnom vremenu i koordinaciju ponude i potražnje za električnom energijom čime se izbjegavaju fluktuacije u frekvenciji ili prekidi u opskrbi. Svi OPS-ovi dužni su održavati stalnu ravnotežu između opskrbe električnom energijom iz elektrana i potražnje potrošača, te osigurati osiguranje rezervi koje će omogućiti iznenadne nepredviđene situacije. Većinom je država vlasnik takvih institucija. [6]

2.1. Industrijski upravljači sustavi

Industrijski upravljački sustav (engl. *Industrial Control System - ICS*) zajednički je pojam za različite vrste upravljačkih sustava i pridružene instrumente koji uključuju uređaje, sustave, mreže i kontrole koji se koriste za rad ili automatizaciju industrijskih procesa. Ovisno o industriji, svaki ICS funkcionira drugačije i dizajniran je za učinkovito elektroničko upravljanje zadacima. Danas se uređaji i protokoli koji se koriste u ICS-u koriste u gotovo svim industrijama i kritičnim infrastrukturama, kao što su proizvodnja, transport, energija i obrada vode. Najpopularniji tipovi ICS-a su SCADA sustavi i distribuirani sustavi upravljanja. [3]

2.2. PLC sustavi

PLC je programabilni logički kontroler, odnosno industrijsko računalo sastavljeno od memorije, procesora, industrijskog ulaza i izlaza, pri čemu ulaz nije tipkovnica, već tipke i sklopke, odnosno razne vrste pretvarača ili senzora. PLC se uglavnom koristi u industriji kao osnovna komponenta sustava automatizacije upravljanja, a njegovi programi ili algoritmi mogu se lako mijenjati za brza rješenja i aplikacije. PLC je digitalno računalo u kojem se program izvršava u petlji koja se sastoji od tri faze:

- Čitanje ulazne varijable
- Izvođenje programskog koda
- Ispis rezultata logičke operacije na izlazu

Programi se pohranjuju u unutarnju memoriju uređaja čak i kada je napajanje isključeno. Dizajniran je za teške uvjete rada i otporan je na vibracije, promjene temperature i električne smetnje.[14]

2.2.1. Programski jezici

Za kodiranje PLC-ova koristi se 5 programskih jezika:

- Ljestvičasta logika
- Dijagram funkcijskih blokova (FBD)
- Strukturirani tekst (ST)
- Popis uputa (IL)
- Sekvencijalna funkcionalna shema (SFC)

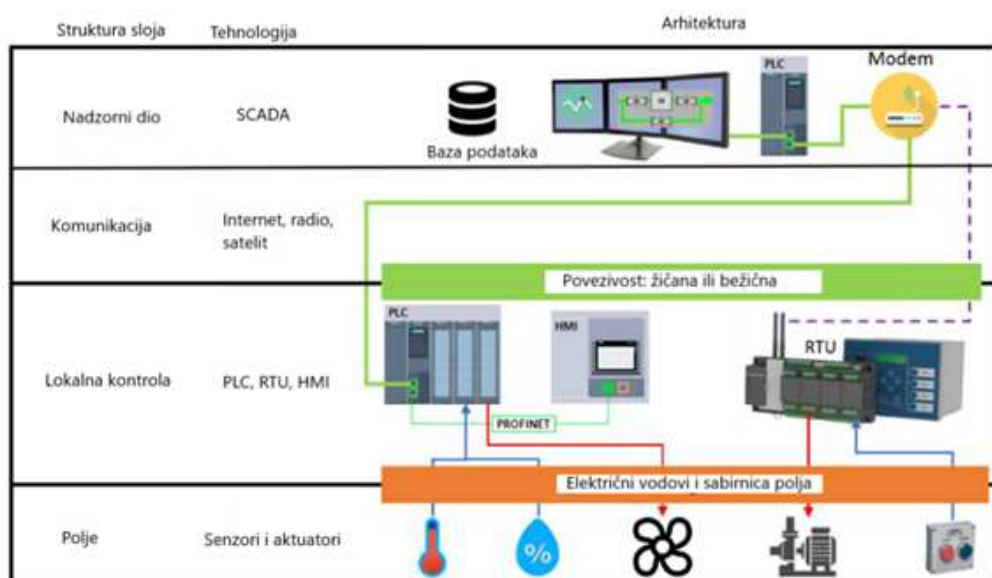
[8]

2.3. SCADA sustavi

Tipični SCADA sustav sastoji se od hijerarhije sljedećih komponenti:

- pretvornika i aktuatora
- RTU (eng. Remote Terminal Unit)
- komunikacijske mreže
- centralne stanice

Ove komponente čine kontrolnu petlju nadzorne povratne sprege u SCADA sustavu.[5]



Slika 2.1: Zaštita OPS sustava [13]

2.3.1. Komponente

Pretvornici i aktuatori

Pretvornik i aktuator predstavljaju početak lanca. Imaju električnu ili mehaničku vezu s procesom koji promatramo. Zadatak pretvornika je praćenje vrijednosti tlaka, protoka, temperature, brzine itd. te prijenos podataka o izmjerenom trenutnom stanju u RTU u analognom ili digitalnom obliku. Aktuator prima informacije od RTU-a, kao što je zatvaranje ili otvaranje ventila. [12]

RTU (engl. *Remote Terminal Unit*)

RTU-ovi su povezani s pretvornicima i aktuatorima i pohranjuju kontrolne parametre koje im senzori pošalju i izvršavaju programe koji izravno kontroliraju parametre električne energije. Stoga postoji stalna razmjena podataka i kontrola između RTU uređaja, pretvornika i aktuatora koje čine lokalnu povratnu kontrolnu petlju. RTU čuvaju prikupljene informacije u svojoj memoriji i čekaju zahtjev od centralne stanice za prijenos podataka. [12]

Komunikacijska mreža

Komunikacijska mreža povezuje sve komponente ovog sustava i omogućuje im komunikaciju. Također omogućuje praćenje podataka u stvarnom vremenu. [12]

Centralna stanica (engl. *MTU*)

Centralna stanica je glavna upravljačka jedinica koja sadrži stvarni SCADA softver, obično je povezana s mnogim RTU-ovim putem komunikacijskih kanala. Inicira sve komunikacije s RTU. Također, zadatak je centralne stanice da komunicira s drugim perifernim uređajima poput monitora, pisača, korporativne mreže i drugih informacijskih sustava.

Ona provjerava RTU-ove u redovitim vremenskim intervalima kako bi pročitala podatke koje je prikupio RTU. Informacije o centralnoj stanici prikazuju se na korisničkom sučelju kako bi se operaterima omogućilo praćenje i upravljanje procesima pametne mreže.

Operateri na centralnoj stanici imaju mogućnost poništiti, promijeniti, nadjačati kritične radne parametre u bilo kojem dijelu SCADA mreže kada je to potrebno. [12]

2.3.2. Ranjivosti SCADA sustava

SCADA sustavi sami po sebi nisu građeni da zaštite podatke s kojima rukuju. Njihova zadaća je očitavanje senzora te određene akcije povratnom spregom ovisno o grani primjene. Potrebne su implementacije dodatnih zaštita kako bi se osigurali svi sigurnosni zahtjevi.

Većina ovih sustava koriste Linux ili Windows operativni sustav koji imaju već globalno poznate ranjivosti koje napadači mogu iskoristiti. Još je veći rizik ako se koriste zastarjele verzije operativnog sustava.

Isto tako se u ovakvim sustavim treba obratiti pozornost na protokole koji se koriste u mreži. Ako ti modeli ne sadržavaju adekvatnu kontrolu pristupa to sa sobom donosi skup problema koji potencijalno nisu ni rješivi sa zakrpama (engl. *patch*). Najčešće korišteni protokoli su MODBUS i DNP3. Većina komponenata može komunicirati MODBUS protokolom i zato se često izabire dok DNP3 nudi veći standard zaštite.

Ovi izolirani sustavi se spajaju na globalnu mrežu kako bi operatorima omogućili spajanje i time dali uvid u komponente, njihova očitavanja i upravljanje, no to stvara priliku za potencijalne napadače da dobiju pristup sistemu. [12]

2.4. Najčešće prijetnje

Operatori prijenosnih sustava uz SCADA sustave koriste i razne druge koji omogućuju primjerice povezivanje zaposlenika unutar kompanije, isplaćivanje plaće, bilježenje radnog vremena. Oni također mogu biti kompromitirani i pružati vektor napada u sustav. Ako napadač dobije pristup na računalo nekog zaposlenika koji ima potrebne ovlasti za pristupanje nekom kritičnom dijelu sustava, cijeli taj blok postaje potencijalno kompromitiran. ENISA (engl. *The European Union Agency for Cybersecurity*) je predstavila glavne kategorije prijetnji za 2020. kako slijedi:

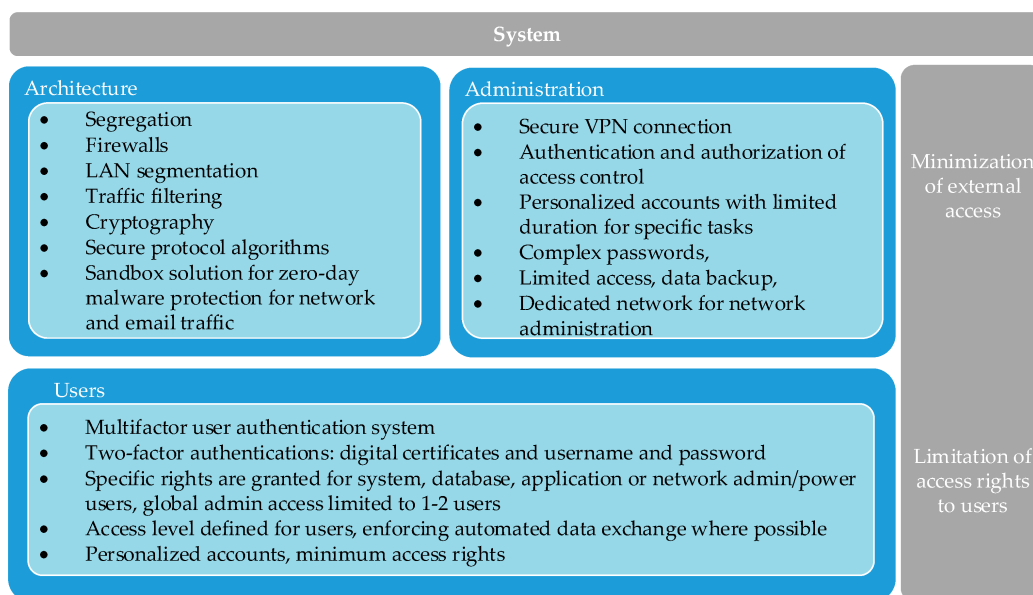
- zlonamjerni softver
- napadi temeljeni na webu/web aplikacijama
- društveni inženjering
- distribuirano uskraćivanje usluge
- krađa identiteta
- povreda podataka
- unutarnja prijetnja

- botnet
- fizička manipulacija i šteta
- curenje informacija
- ransomware
- kibernetičko ratovanje/špijunaža i kriptovaluta

[10]

2.5. Zaštite koje se primjenjuju

Implementacija segregacije i vatrozida su među najčešćim mjerama zaštite uz filtriranje prometa i antivirusnih programa. Administrativni poslovi na važnim infrastrukturnim objektima obavljaju se putem računa koji nameću određena ograničenja kao zaštita lozinkom, ograničenje trajanja vremena ili ograničenja pristupa. Uobičajeni pristup je implementacija višefaktorske autentifikacije korisnika, ali također se koriste i specifična prava pristupa za korisnike koji obavljaju zadatke u kritičnim okruženjima (definicija različitih razina pristupa za različite vrste korisnika, personalizirani računi i minimalna prava za pristup). Korisnici trebaju biti motivirani i obučeni za održavanje potrebne razine opreza. [10]



Slika 2.2: Zaštita OPS sustava [10]

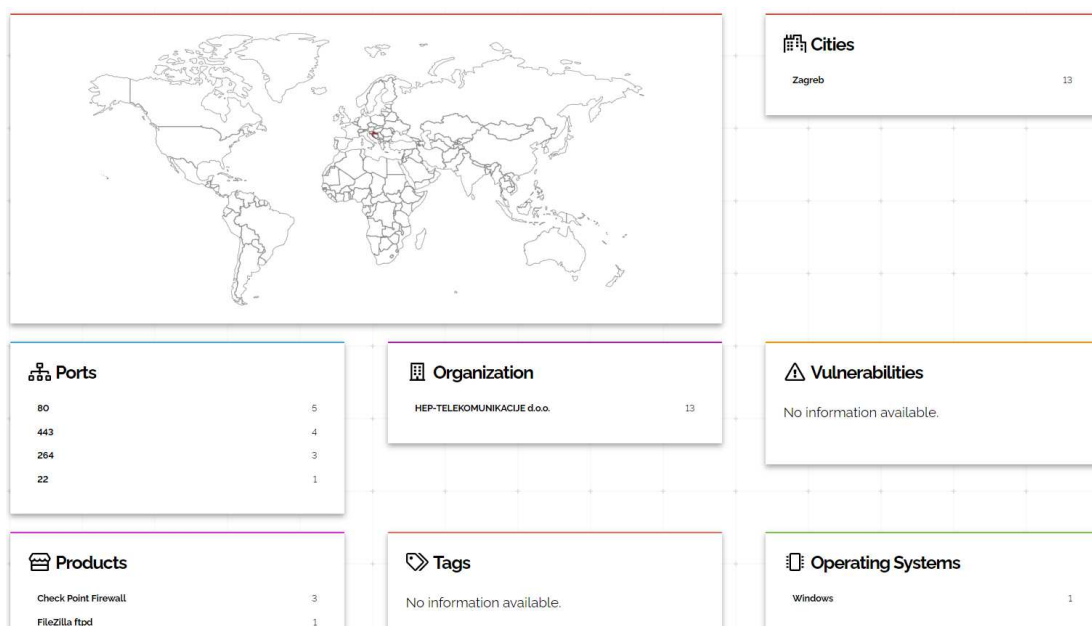
2.6. OSINT nad operatorom prijenosnih sustava HEP

OSINT je skraćenica za obavještajne informacije iz otvorenih izvora (engl. *Open-Source Intelligence Tools*), što se odnosi na informacije koje se mogu prikupiti iz javnih izvora o pojedincu/ki ili organizaciji. U praksi to obično znači informacije na internetu, ali tehnički sve informacije spadaju u ovu kategoriju. Od knjiga ili izvještajima u javnoj biblioteci, člancima u medijima do izjava danih u obraćanju javnosti. [4]

2.6.1. OSINT alati

Shodan

Shodan je tražilica koja pomoću raznih filtera skenira sve sustave otvorene na internet i dobiva podatke o njima. To mogu biti poslužitelji, usmjeritelji ili IoT uređaji. Izvodi skeniranje portova sustava koje detektira, otkriva servise koji se izvode na otvorenim portovima i otkriva verzije servisa. Mogu se pretraživati i korisnici servisa. Ako postoji bilo kakva ranjivost povezana s otkrivenim verzijama usluge, daje kratko objašnjenje o ranjivosti s CVE (engl. *Common Vulnerabilities and Exposures*) kodom ranjivosti.

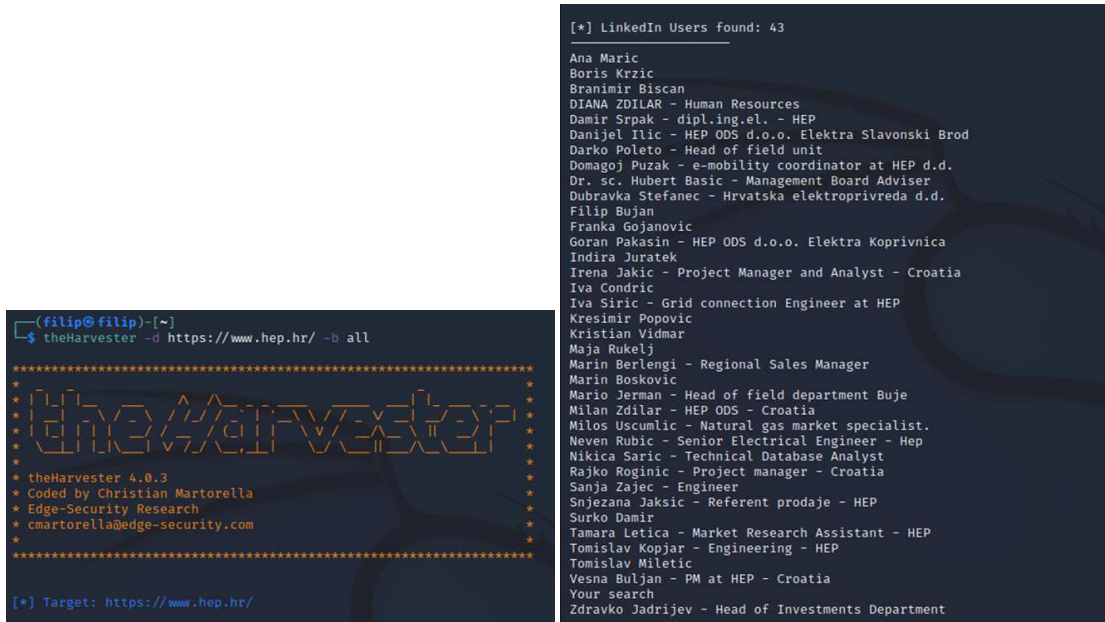


Slika 2.3: podatci dobiveni alatom shodan

Korištenjem Shodana prikupljene su neke informacije o poslužiteljima koji se nalaze unutar HEP-ove organizacije. Ti se poslužitelji koriste operacijskim sustavom Windows i nemaju poznatih ranjivosti. Također su navedeni otvoreni portovi te vrsta sadržaja koji se nalazi na njihovoj mreži.

The Harvester

Ovo je alat pisan u Pythonu koji korisnicima omogućuje pronalaženje informacija kao što su poddomene, mailovi, LinkedIn profili, poslužitelji i njihovi otvoreni portovi. Može se povezati i sa Shodanom tako da za sve pronađene poslužitelje prođe kroz njegovu bazu podataka i vrati nalaze. Dolazi odmah instaliran na Kali linux operativnim sustavima. [7]



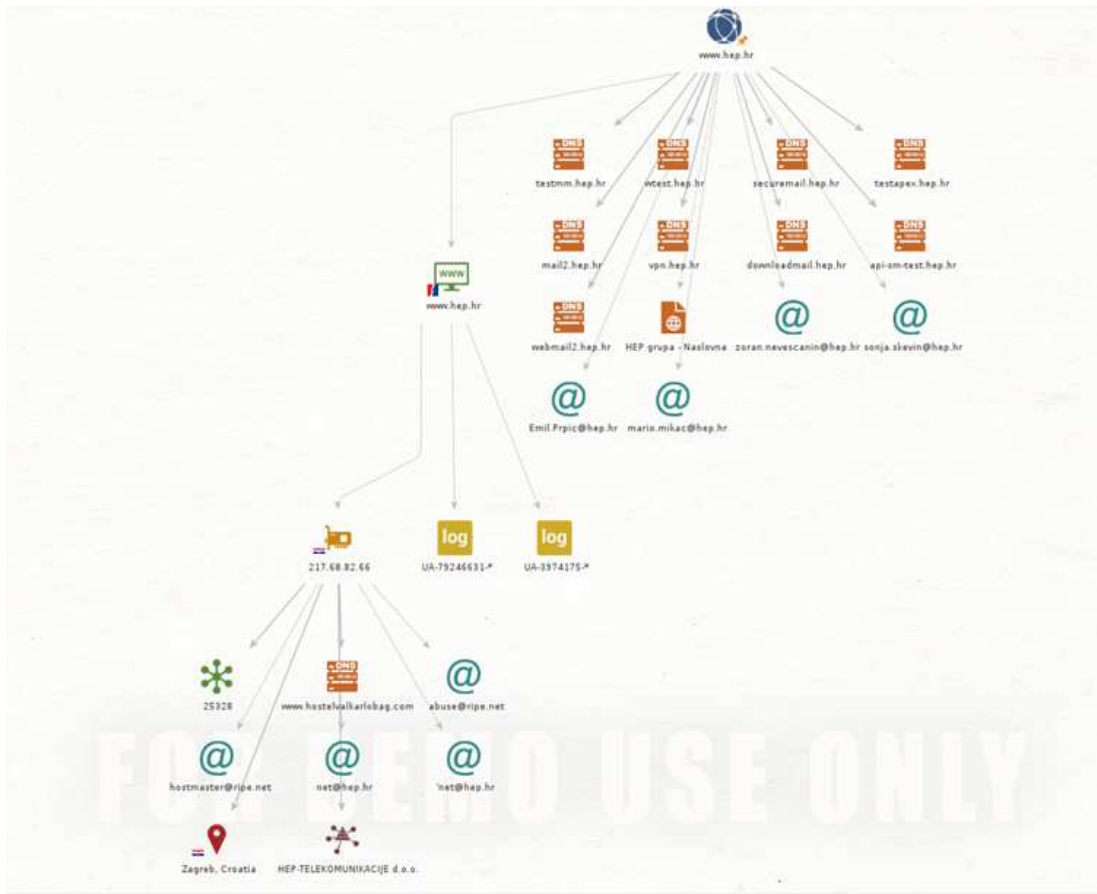
(a) Upit u alatu theHarvester

(b) Podatki dobiveni alatom theHarvester

U kontekstu HEP-ove mreže, skeniranjem pomoću ovog alata dobiveni su LinkedIn profili nekih od zaposlenika. Međutim nisu pronađene informacije o drugim vrstama podataka.

Maltego

Maltego je alat za pronalazak podataka koji su javno dostupni. Koristi prikupljanje podataka u stvarnom vremenu i vizualno je privlačan jer dobivene podatke pretvara u prikaz grafa. Sadrži integraciju sa Shodanom što dodatno pojačava njegovu efektivnost. Neizostavan je alat u penetracijskom testiranju.

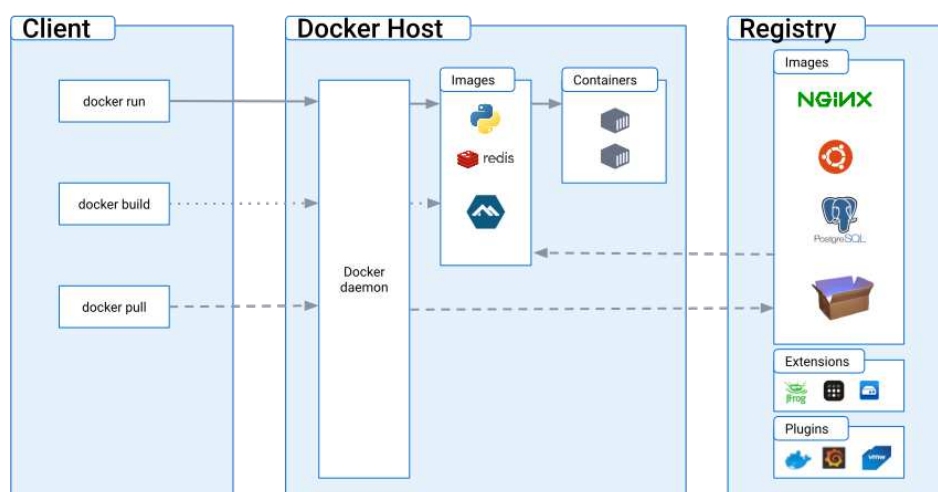


Slika 2.5: Podatci dobiveni alatom Maltego

Korištenjem tog alata postignut je najveći uspjeh u pretraživanju HEP-a. Različitim upitima je graf povećan za nekoliko čvorova, pružajući informacije o email adresama, logovima, DNS poslužiteljima, lokacijama i IP adresama. Napadač bi potencijalno mogao iskoristiti te IP adrese za slanje phishing emailova.

3. Docker

Docker je platforma pomoću koje se mogu razvijati i pokretati aplikacije. Aplikacije se vrte u izoliranom okruženju koje se zove kontejner i time se uklanja potreba za posebnim konfiguracijama iste aplikacije ovisno na kojem sustavu se pokreće. Koristi klijent - server arhitekturu. Pozivom s klijenta docker pozadinski proces koji se nalazi na poslužiteljskoj strani čeka docker API naredbe. On odrađuje većinu posla, primjerice gradi slike i pokreće kontejnere. Dockerov pozadinski proces se može pokretati na istom računalu kao i klijent. Također, jedan klijent može komunicirati s više pozadinskih procesa.[1]



Slika 3.1: Arhitektura Dockera [1]

3.1. Docker slika

U kontekstu Dockera slika označava skup naredbi koje se trebaju odvijati pri izgradnji kontejnerskog okruženja. Nisu promjenjive, samo se mogu čitati. Mogu se koristiti

za izgradnju proizvoljnog broja kontejnera i daju slobodu uređivanja kontejnera do sitnih detalja. Lako su prenosive i postoje mnoge javno dostupne baze podataka koje su namijenjene njihovom dijeljenju kao što je Docker hub.

3.1.1. Dockerfile

Ponekad se osnovna slika sama po sebi smatra dobrim temeljem za Docker kontejner, ali često ne sadrži sve što je potrebno ili ne koristi određene naredbe koje su bitne za pokretanje željenih procesa. Rješenje za takve probleme pruža se putem Dockerfile-a. On omogućuje izgradnju personaliziranih slika na temelju već postojećih.

Započinje naredbom FROM u kojoj je određeno koju sliku koristiti kao bazu i MAINTAINER koji označava ime i email osobe koja je stvorila docker file. Uz početak, neke od čestih naredbi koje se koriste su:

- RUN izvršava naredbu navedenu u njenim argumentima
- CMD označava što će se sve pokrenuti u konzoli pri stvaranju kontejnera
- ENV označava varijablu okruženja
- VOLUME omogućuje pristupanje kontejnera navedenom folderu s host računala
- EXPOSE Određuje na kojem će portu kontejner slušati za nadolazeći promet pri pokretanju

Docker slika se iz Dockerfile-a kreira naredbom:

```
$ docker build -t Docker_Image_Name -f Dockerfile .
```

3.2. Docker kontejner

Docker kontejner je instanca slike koju vrti u pozadini.

Pokreće se naredbom :

```
$ docker run Docker_image_name
```

Njihova manipulacija se izvodi korištenjem Docker API-a. Mogu se pokrenuti, zaustaviti i obrisati. Pri brisanju sve promjene koje nisu spremljene se poništavaju. Okruženje u kojem se vrti je izolirano, ali ta se izolacija može narušiti davanjem prevelikih prava kontejneru, dopuštanjem pristupa direktorijima host računala ili otvaranju nekog porta.

4. IMUNES

IMUNES (engl. *Integrated Multiprotocol Network Emulator / Simulator*) je mrežni simulator koji nudi visoku skalabilnost i performanse. S razvojem distribuiranog simulatora temeljenog na IMUNES-u, povećana je granica skalabilnosti jer koristi koncept lagane virtualnog stroja koja ne troši resurse glavnog stroja. Za razmjenu informacija između pokrenutih virtualnih strojeva koristi pokazivače na pakete što dodatno poboljšava performanse. To je aplikacija za specifikaciju topologije i upravljanje mrežom. [11]

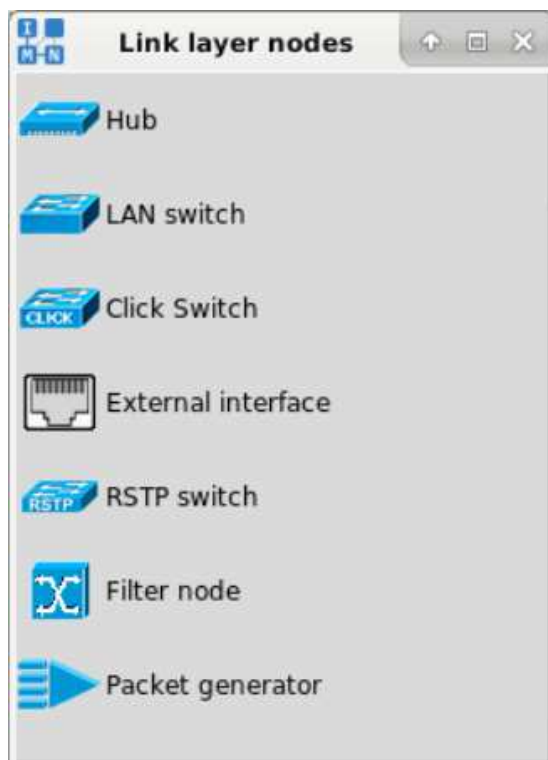
Pri pokretanju simulatora, potrebno mu je dodijeliti administratorska prava kako bi ispravno funkcionirao. Važno je pravilno završiti pokus tako da se aplikacija prvo terminira. U suprotnom, na operativnom sustavu mogu ostati aktivni Docker kontejneri koji su bili pokrenuti u svrhu simulacije, te će biti potrebno ručno ih zaustaviti putem konzole.

4.1. Topologija mreže

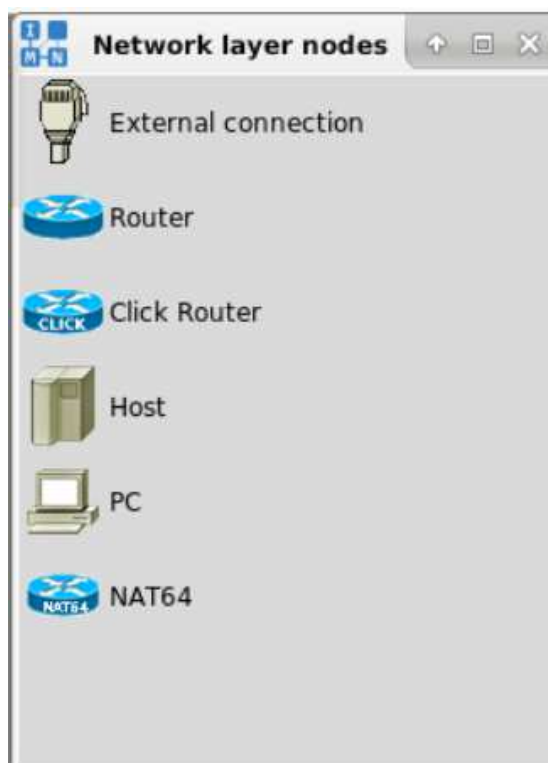
IMUNES koristi grafičko sučelje za crtanje i prikaz željene topologije mreže. Sastoji se od dvije osnovne jedinice: Čvorova i Veza. Čvorovi mogu postojati neovisno, dok veze uvijek povezuju dva različita čvora. Čvorovi se dalje dijele na čvorove sloja podatkovne poveznice i čvorove mrežnog sloja.

Osnovni čvorovi sloja podatkovne poveznice su LAN komutator, most i fizičko sučelje. Čvorovi ovog sloja nisu implementirani kao virtualni strojevi nego su samo čvorovi mreže.

Čvorovi mrežnog sloja su sposobni za obradu mrežnog sloja, a to su čvorovi implementirani u jezgri operacijskog sustava kao virtualni strojevi. Osnovni IMUNES čvorovi mrežnog sloja su poslužitelj, računalo i usmjerivač.



(a) Čvorovi sloja podatkovne poveznice



(b) Čvorovi mrežnog sloja

5. Mamac

Mamci (engl. *Honeypots*) su lažni poslužitelji ili sustavi postavljeni pored stvarnih sustava koji se koriste za proizvodnju. Dizajnirani su da izgledaju kao atraktivne mete i raspoređeni su kako bi omogućili praćenje sigurnosnih odgovora sustava i odvratili napadače od pristupa njihovim stvarnim metama.

Postoje različite vrste mamaca koji odgovaraju specifičnim potrebama. Budući da izgledaju kao legitimne prijetnje, djeluju kao zamke koje omogućuju rano otkrivanje napada i odgovarajuće odgovore. Ova im značajka omogućuje da se koriste na brojne načine kako bi se napadači držali podalje od kritičnih sustava. Kada napadač pristupi mamcu, mogu se prikupiti važne informacije o vrsti napada i metodama koje napadač koristi.

Mamci najbolje funkcioniraju na sustavima koji izgledaju legitimni. Drugim riječima, trebali bi simulirati isti proces kao stvarni proizvodni sustav. Osim toga, trebali bi sadržavati datoteke koje bi napadač smatrao prikladnima za ciljani proces. U mnogim slučajevima, najbolje je postaviti mamac iza vatrozida koji štiti mrežu. To omogućuje pregled prijetnji koje prolaze kroz vatrozid i sprječava napade usmjerene na pokretanje iz zaraženih mamaca. Kada dođe do napada, vatrozid koji se nalazi između honeypota i interneta može presresti i izbrisati podatke.

Mamci igraju ključnu ulogu u sigurnosti operatora prijenosnog sustava koji često rade sa izravno osjetljivim podacima i očitanjima. Oni se obično implementiraju u svim dijelovima organizacije, ali s posebnim fokusom na mamce SCADA sustava kako bi se otežalo kompromitiranje ispravnih sustava. Osim toga, više različitih verzija mamaca može se koristiti za povećanje sigurnosti i otežati napadačima otkrivanje stvarnog sustava. [2]

5.0.1. Vrste mamaca

Čisti mamac

Čisti Mamac je kompletan sustav koji radi na različitim poslužiteljima. U potpunosti oponaša proizvodni sustav. Sadrži podatke koji odaju dojam tajnovitosti, kao i "osjetljive" korisničke podatke opremljene nizom senzora za praćenje aktivnosti napadača. Ovaj tip mamaca omogućuje temeljito praćenje napadačevih postupaka i pomaže u razumijevanju njihovih ciljeva i metoda. [2]

Mamac visoke interakcije

Mamac visoke interakcije dizajniran je kako bi omogućio napadačima da ulože što više vremena u njega. To daje sigurnosnim timovima više mogućnosti za promatranje ciljeva i namjera napadača te više mogućnosti za pronalaženje ranjivosti u sustavu. Ovaj tip mamaca može simulirati kompleksne mreže, baze podataka i procese u koje napadači pokušavaju prodrijeti. Istraživači mogu detaljno analizirati kako napadači traže informacije, koje im informacije privlače pažnju i kako pokušavaju proširiti svoj pristup. [2]

Mamac srednje interakcije

Mamac srednje interakcije oponaša elemente aplikacijskog sloja, ali bez operativnog sustava. Njegova je svrha zbuniti ili odvratiti napadače, dajući organizacijama više vremena da shvate kako odgovoriti na takve napade. Ovaj tip mamaca može simulirati određene dijelove sustava koji izgledaju privlačno napadačima, ali nemaju stvarne operativne funkcionalnosti. [2]

Mamac niske interakcije

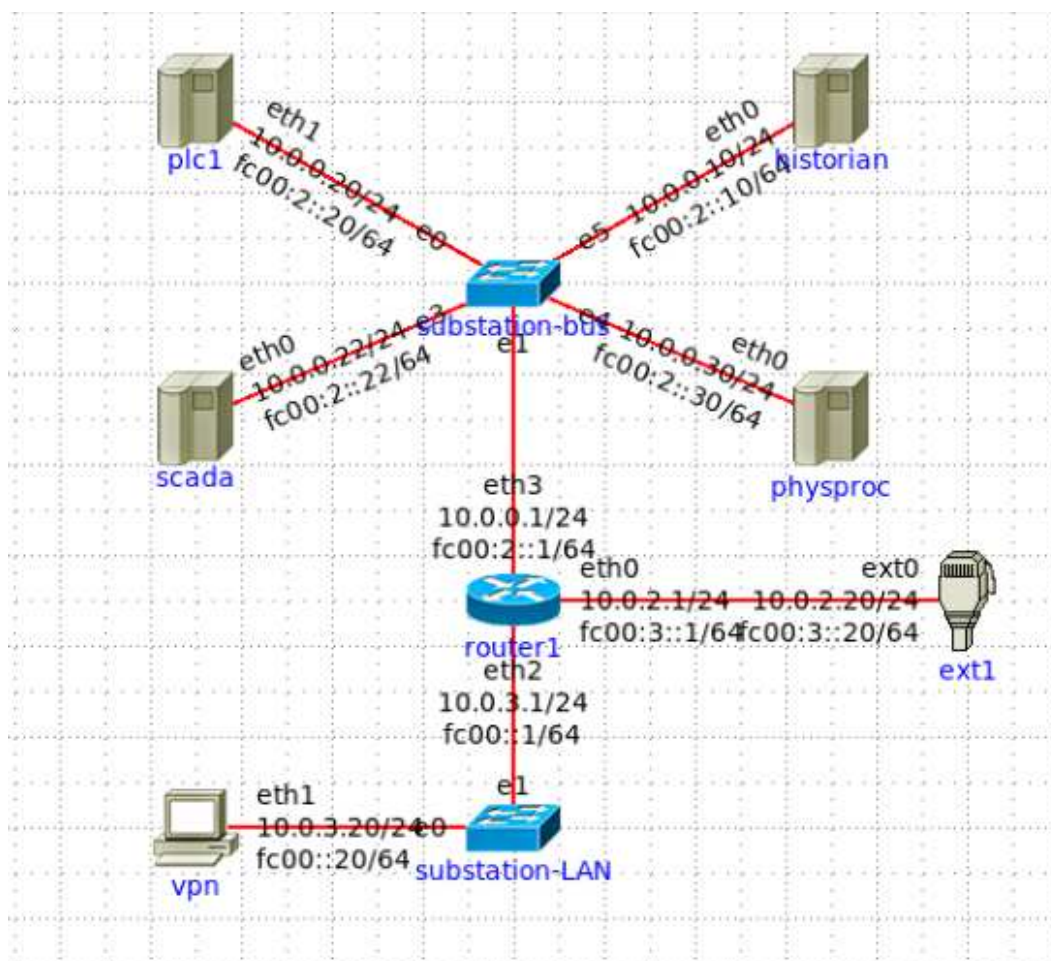
Mamac niske interakcije zahtijeva manje resursa i prikuplja osnovne informacije o vrsti prijetnje i njenom izvoru. Relativno ga je jednostavno postaviti i koristiti. Međutim, ne postoji ništa unutar njega što bi moglo dugo zadržati pažnju napadača. Ovaj tip mamaca često se koristi kao dodatna sigurnosna mjera koja pomaže u detekciji i praćenju prijetnji, ali nema kompleksne funkcionalnosti složenih verzija mamaca.[2]

5.0.2. Mamci u kontekstu operatora prijenosnih sustava

Mamci se često implementiraju u operatorima prijenosnih sustava kako bi se otežalo kompromitiranje pravih sustava. U tom kontekstu, verzije mamaca koje se najčešće koriste su čisti mamci i mamci visoke interakcije. Čisti mamci pružaju sveobuhvatno praćenje napadačevih aktivnosti i omogućuju detaljnu analizu napada. S druge strane, mamci visoke interakcije pružaju više mogućnosti za promatranje napadačevih namjera i pronalaženje ranjivosti u sustavu. Također, mamci niske interakcije mogu se koristiti za dodatnu sigurnost i detekciju prijetnji na PLC (engl. *Programmable Logic Controller*) komponentama. Kombinacija različitih verzija mamaca pruža sveobuhvatnu sigurnost i pomaže organizacijama u zaštiti svojih prijenosnih sustava. [2]

5.1. TSO mamac izveden u IMUNES-u

Za simulaciju TSO (engl. *Transmission system operator*) mamca korišten je Končarov SCADA sustav Hat Open. Pokretanje simulacije provedeno je u IMUNES-u, gdje svaki čvor unutar mreže pokreće vlastiti Docker kontejner koji izvršava odgovarajući sustav, ovisno o ulozi čvora. Ovaj mamac simulira jednostavnu mrežu trafostanice, sličnu onoj koja se može vidjeti u stvarnim sustavima.



Slika 5.1: Topologija mreže

Komponente mreže

Ova mreža sadrži simulirani PLC sklop (čvor plc1), cijeli SCADA sustav (čvor scada), historian poslužitelj (čvor historian) i simulaciju fizičkog procesa (čvor physproc).

Kada se pokrenu i inicijaliziraju komponente, sučelja PLC honeypota i SCADA sustava nalaze se na sljedećim adresama:

- 10.0.0.20:8800 - PLC mamac (Conpot)
- 10.0.0.22:23021 - Hat Orchestrator
- 10.0.0.22:23022 - Hat Monitor
- 10.0.0.22:23023 - Hat GUI
- 10.0.0.22:23024 - Hat Manager

Komponenta Hat Orchestrator je odgovorna za upravljanje pokretanjem i zaustavljanjem različitih Hat usluga, uključujući Hat GUI, Hat Monitor i Hat Manager. Ova komponenta koordinira njihovu međusobnu interakciju i osigurava da sve usluge rade usklađeno.

Hat GUI je HMI (engl. *Human Machine interface*) komponenta koja pruža korisnicima mogućnost pregleda statusa trafostanice. Izvorni prototip implementiran je kao tablica koja prikazuje važne podatke o stanju trafostanice. Međutim, planira se nadogradnja s ciljem proširenja funkcionalnosti i poboljšanja korisničkog iskustva.

Hat Monitor je komponenta koja prikazuje zapise SCADA sustava pohranjene u bazi podataka.

Hat Manager je komponenta koja omogućuje korisnicima pregled poruka koje generira simulator procesa. Hat Manager pruža mogućnost pregleda generiranih poruka kako bi korisnici mogli pratiti rad simulacije i donositi informirane odluke na temelju tih podataka.

Čvor historian se ne koristi jer nije napravljena migracija podataka sa SCADA čvora na njega. Zbog ovoga je pri pokretanju eksperimenta, IMUNES javljao grešku. Također je javio grešku i za čvor plc1 koji se također izbacio, ali to nije imalo utjecaja na praktični zadatak koji se radio na ovoj mreži i jedina stvar koja bi se morala dodati na te čvorove je Python skripta koja je napravljena za druge čvorove i objašnjena je u nastavku. [9]

Inicijalizacijska bash skripta

Nakon pokretanja eksperimenta mora se pokrenuti i inicijalizacijska skripta kako bi se mreža predviđeno ponašala. U njoj se određuje kako će se prosljeđivati paketi i određuju se pravila koja će se primjenjivati na određene čvorove. Skriptu treba pokrenuti s administratorskim ovlastima.

5.1.1. VPN

VPN poslužitelj

Prva akcija bila je postavljanje VPN poslužitelja na čvor VPN unutar virtualne mreže u IMUNES-u. Istraživane su mogućnosti koje bi najbolje odgovarale za potrebe TSO mamaca, te je odlučeno koristiti SoftEther VPN kao temeljnu Docker sliku. S drugim VPN rješenjima se javljao problem da zahtijevaju dodatne parametre prilikom pokretanja Docker slike, no pri stvaranju kontejnera putem IMUNES eksperimenta, parametri su definirani u naredbi komandne linije pri pokretanju. Ti parametri su:

```
$ exec docker run --detach --init --tty \
--privileged --cap-add=ALL --net=$network \
--name $node_id --hostname=[getNodeName $node] \
--volume /tmp/.X11-unix:/tmp/.X11-unix \
--sysctl net.ipv6.conf.all.disable_ipv6=0 \
--ulimit nofile=$ULIMIT_FILE --ulimit nproc=$ULIMIT_PROC \
$vrout
```

Korištenjem SoftEther VPN-a, većina potrebnih parametara postavljena je unutar prethodnog bloka koda. Ostaju samo podaci za autentifikaciju i konfiguracijska datoteka. Ove dvije stavke mogu se dodati stvaranjem vlastitog Dockerfile-a na temelju početne slike (siomiz/softethervpn). Korisničko ime i lozinku dodaje se kao varijable okruženja, dok se konfiguracijska datoteka stvara prema automatski generiranoj verziji s promjenom koja pozicionira klijenta u mrežu 10.0.3.0/24 prilikom spajanja. Prilikom stvaranja Docker slike, kopira se konfiguracijska datoteka na odgovarajuće mjesto gdje bi se inače automatski generirala, i u tom slučaju VPN prepoznaje da treba koristiti već postojeću verziju konfiguracije.

VPN client

Prva isprobana opcija za VPN klijenta je OpenVPN. Pri spajanju nije uspijevaao pri autentifikaciji. Kao moguće rješenje pokušano je zadati predefinirane certifikate kroz varijable okruženju u Dockerfile-u koji se koristi za izradu slike VPN poslužitelja, ali

ta metoda nije uspjela. Nakon dodatnog istraživanja postalo je očito da OpenVPN klijent ne radi sa SoftEther VPN poslužiteljem i da je najbolja opcija korištenje SoftEther VPN klijenta. Ta je opcija korištena u završnoj implementaciji. Potrebne su administratorske ovlasti pri pokretanju kako bi ispravno radio. Nakon instalacije na poslužiteljskom sustavu i pozicioniranja u taj direktorij klijent se pokreće koristeći naredbu:

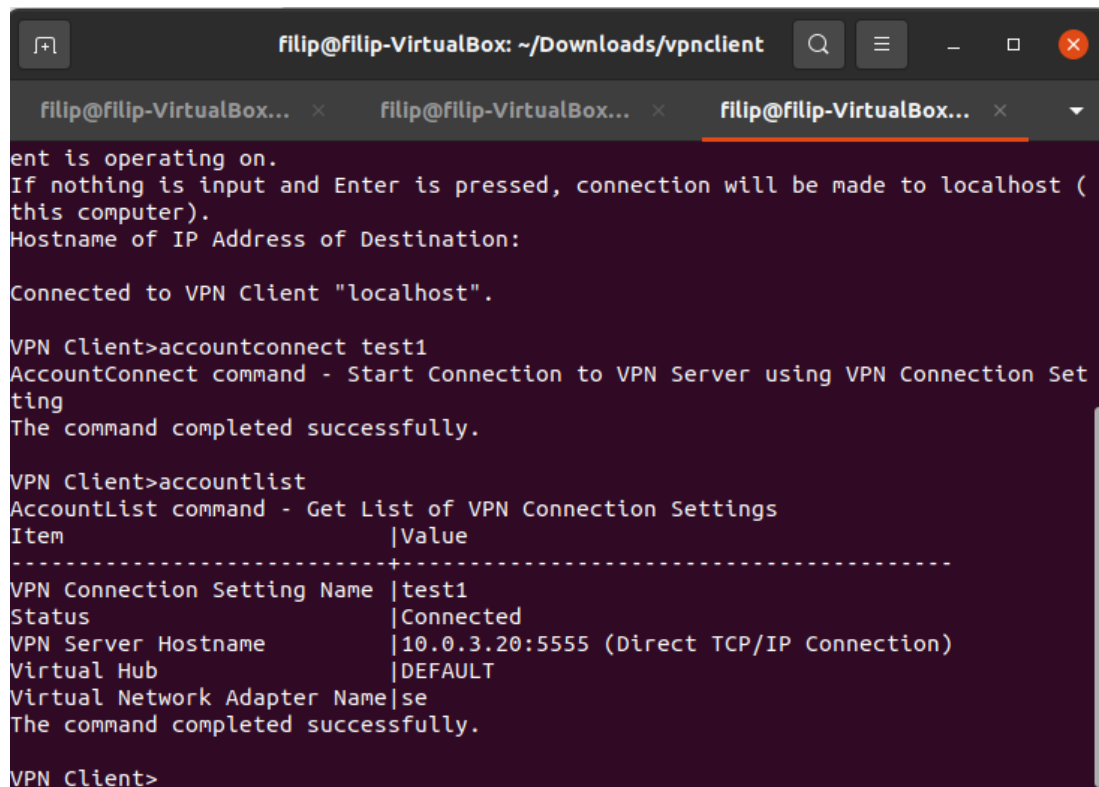
```
$ sudo ./vpncclient start
```

Kada je pokrenut, koristeći naredbu

```
$ sudo ./vpncmd
```

može se pristupiti alatu naredbenog retka koji dolazi uz klijent. On se koristi za uređivanje postavki VPN poslužitelja, upravljanje korisničkim računima, mijenjanje sigurnosnih postavki i upravljanje konekcijama. U kontekstu zadatka, trebalo je izraditi jednostavan VPN koji bi zahtijevao autentifikaciju prilikom spajanja koji bi se koristio kao ulazna točka u mrežu mamca. Za tu svrhu, konfiguriran je korisnički račun za spajanje na VPN poslužitelj s IP adresom 10.0.30.20 i portom 5555.

Zatraživanjem spajanja na klijentskoj strani vidljivo je da je konekcija uspješno uspostavljena.



```
ent is operating on.
If nothing is input and Enter is pressed, connection will be made to localhost (
this computer).
Hostname of IP Address of Destination:

Connected to VPN Client "localhost".

VPN Client>accountconnect test1
AccountConnect command - Start Connection to VPN Server using VPN Connection Set
ting
The command completed successfully.

VPN Client>accountlist
AccountList command - Get List of VPN Connection Settings
Item |Value
-----+-----
VPN Connection Setting Name |test1
Status |Connected
VPN Server Hostname |10.0.3.20:5555 (Direct TCP/IP Connection)
Virtual Hub |DEFAULT
Virtual Network Adapter Name|se
The command completed successfully.

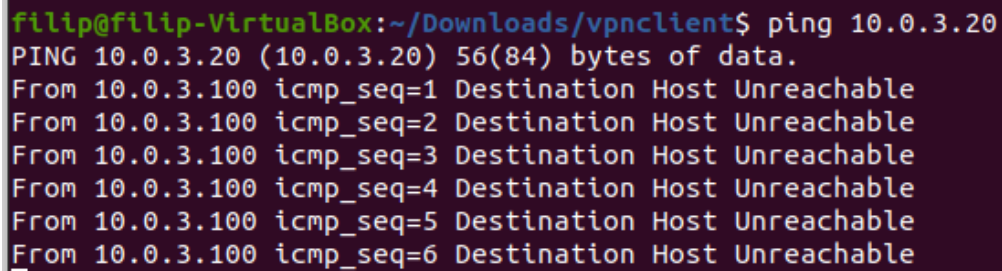
VPN Client>
```

Slika 5.2: Stanje korisničkog računa nakon zatraženog spajanja na poslužitelj

Pregledavanjem IP adresa po mrežnim sučeljima uočeno je da sučelje vpn_se kojim se spaja na poslužitelj nema zadanu IPv4 adresu. To je riješeno koristeći naredbu:

```
$ sudo dhclient vpn_se
```

kojom se pokreće proces DHCP klijenta na određenom sučelju, omogućavajući sustavu automatsko dobivanje IP adrese i konfiguracije mreže s DHCP poslužitelja. Nakon ove naredbe vpn_se sučelje dobiva IP adresu 10.0.3.100 s kojom bi trebalo biti u mreži VPN poslužitelja, ali korištenjem naredbe ping nad svim sučeljima u mreži dobiva se poruka o nedostupnosti.

A screenshot of a terminal window with a dark background. The prompt is 'filip@filip-VirtualBox:~/Downloads/vpnclient\$'. The command entered is 'ping 10.0.3.20'. The output shows the ping command details: 'PING 10.0.3.20 (10.0.3.20) 56(84) bytes of data.' followed by six lines of 'From 10.0.3.100 icmp_seq=1' through 'icmp_seq=6', all reporting 'Destination Host Unreachable'.

```
filip@filip-VirtualBox:~/Downloads/vpnclient$ ping 10.0.3.20
PING 10.0.3.20 (10.0.3.20) 56(84) bytes of data.
From 10.0.3.100 icmp_seq=1 Destination Host Unreachable
From 10.0.3.100 icmp_seq=2 Destination Host Unreachable
From 10.0.3.100 icmp_seq=3 Destination Host Unreachable
From 10.0.3.100 icmp_seq=4 Destination Host Unreachable
From 10.0.3.100 icmp_seq=5 Destination Host Unreachable
From 10.0.3.100 icmp_seq=6 Destination Host Unreachable
```

Slika 5.3: ping naredba upućena VPN poslužitelju

Pregledavanjem logova na poslužiteljskoj strani uočeno je da se klijent uspješno autentificira i poslužitelj za njega stvori konekciju, ali je odmah nakon toga i terminira.

```
2023-05-27 21:53:07.740 On the TCP Listener (Port 5555), a Client (IP address 10.0.2.20, Host name "10.0.2.20", Port number 52158) has connected.
2023-05-27 21:53:07.740 For the client (IP address: 10.0.2.20, host name: "10.0.2.20", port number: 52158), connection "CID-2" has been created.
2023-05-27 21:53:07.791 SSL communication for connection "CID-2" has been started. The encryption algorithm name is "TLS_AES_256_GCM_SHA384".
2023-05-27 21:53:07.791 Connection "CID-2" has been terminated.
2023-05-27 21:56:12.170 [HUB "DEFAULT"] Session "SID-SECURENAT-1": The DHCP server of host "5E-7F-8A-57-80-DE" (10.0.3.21) on this session allocated, for host "SID-FILIP-3" on another session "5E-1A-F8-1E-DC-2E", the new IP address 10.0.3.100.
2023-05-27 21:56:47.817 [HUB "DEFAULT"] Session "SID-FILIP-3": The session has been terminated. The statistical information is as follows: Total outgoing data size: 27456 bytes, Total incoming data size: 29437 bytes.
2023-05-27 21:56:47.849 Connection "CID-1" terminated by the cause "The VPN session has been deleted. It is possible that either the administrator disconnected the session or the connection from the client to the VPN Server has been disconnected." (code 11).
2023-05-27 21:56:47.849 Connection "CID-1" has been terminated.
2023-05-27 21:56:47.849 The connection with the client (IP address 10.0.2.20, Port number 52154) has been disconnected.
bash-5.1# █
```

Slika 5.4: VPN poslužitelj log
Uzrok ovog problema nije pronađen.

Klijent se gasi naredbom:

```
$ sudo ./vpncclient stop
```

Korištenje alternative

U ovoj alternativnoj metodi koristi se drugo virtualno računalo koje pokreće Tiny Core Linux kao svog posrednika. Spajanjem na sustav s ovog zasebnog virtualnog računala osigurava se sigurna i šifrirana veza za prijenos podataka.

Kako bi se održao robustan sigurnosni okvir, specifične konfiguracije su implementirane unutar postavki vatrozida. Vatrozid je konfiguriran tako da dopušta dolazne veze isključivo s ovog virtualnog stroja. Time se ograničava pristup isključivo virtualnom računalu koje se koristi kao pristupnik.

Ovom alternativnom metodom ne samo da se nudi sigurna veza, već se također pruža dodatni sloj zaštite od potencijalnih ranjivosti povezanih s tradicionalnim VPN softverom. Korištenjem zasebnog virtualnog stroja i ograničenja vatrozida, uspostavljen je pouzdan i prilagođen sustav koji odgovara specifičnim sigurnosnim potrebama.

Na glavnoj virtualnoj mašini je dodano *host only* sučelje i podešena je skripta da ovo sučelje i IP adrese 10.0.0.0/16 uspješno prosljeđuju pakete.

Na Tiny Core Linux virtualnoj mašini je također postavljeno *host only* sučelje. Omogućeno je prosljeđivanje paketa i kao posrednik je postavljeno *host only* sučelje sa glavne virtualne mašine.

Nakon dodavanja ovih pravila može se simulirati VPN konekcija.

5.1.2. Vatrozid

Nakon implementacije VPN-a bitna stavka je vatrozid kojim se može ograničiti kuda promet smije ići, a kuda ne. To je korisno kada se primjerice na nekim mjestima želi napraviti promet jednosmjernim osim u slučaju odgovaranja na zahtjeve ili u potpunosti zabrana prometa. Vatrozid je napravljen uz pomoć iptables alata koji jednostavnom sintaksom omogućuje stvaranje ovakvih pravila. Rađen je na usmjeritelju koji se nalazi unutar simulirane mreže. Za pozicioniranje na usmjeritelj kroz bash skriptu koristi se himage naredba uz navođenje imena čvora na kojem se pravila implementiraju. Naredbe koje su korištene za pravila:

```
# firewall on router
himage router1 iptables -P FORWARD DROP
himage router1 iptables -A FORWARD -i eth0 -o eth2 -j ACCEPT
himage router1 iptables -A FORWARD -i eth2 -o eth1 -p tcp -m multiport --dports 8080,8081 -j ACCEPT
himage router1 iptables -A FORWARD -i eth3 -o eth1 -p tcp -m multiport --dports 8080,8081 -j ACCEPT
himage router1 iptables -A FORWARD -i eth2 -o eth3 -j ACCEPT
himage router1 iptables -A FORWARD -i 192.168.124.5 -o eth3 -j ACCEPT
himage router1 iptables -I FORWARD -m state --state RELATED,ESTABLISHED -j ACCEPT
```

Ovaj blok je dodan u inicijalizacijsku bash skriptu. Iako je za VPN konfiguraciju korištena druga virtualna mašina, vatrozid je napravljen za taj slučaj i za slučaj kada se docker slika sa VPN-om pokreće sa VPN čvora. prvom linijom osigurano je da sav promet koji usmjeritelj prosljeđuje uvijek odbaci ako nije drukčije definirano. Zatim je omogućeno da se uspješno prosljeđuje vanjski promet u mrežu u kojoj je VPN čvor. Omogućen je promet iz mreže VPN-a u mrežu u kojoj se nalazi Log poslužitelj i mrežu u kojoj mi je mamac. Također je omogućena i direktna veza između mreže mamaca i Log poslužitelja. Sve veze na mrežu u kojoj je Log poslužitelj dopuštene su samo na portu 8080 i 8081 na kojima python skripte slušaju na promet. Zadnja stvar bilo je dopuštanje povratne veze tj. odgovora na zahtjeve koji se šalju iz bilo kojeg čvora. Drugim riječima ovim je omogućeno da npr. SCADA sustav može odgovoriti na upite, ali je i osigurano da ne može sam od sebe slati podatke u vanjsku mrežu što bi napadač potencijalno probao napraviti. Također je osigurano da napadač može raditi upite na mamac samo ako je u 10.0.3.0/24 mreži ili ako je spojen na virtualnu mašinu koja oponaša VPN na adresi 192.168.124.5 i na kojoj se vrti Tiny Core Linux, čime sprječavamo spajanje s bilo kojih drugih sustava. Log poslužitelj također može samo primati podatke i odgovarati na zahtjeve jer je nepoželjno da šalje promet van mreže sam od sebe.

5.1.3. Skupljanje prometa i IDS (engl. *Intrusion Detection System*)

Sakupljanje prometa je bitna stavka u mamcima jer nam omogućuje praćenje onoga što napadač radi i na temelju toga može se vidjeti koje ranjivosti želi iskoristiti i koji mu je tijek djelovanja. U topologiji naše mreže dovoljno je sakupljati promet s usmjeritelja jer se želi zabilježiti promet koji dolazi iz vana u virtualnu mrežu. To je lako ostvarivo alatom Wireshark.

Alati za otkrivanje prijetnji su također korišteni i pružaju još jedan stupanj sigurnosti tj. alarm za određene maliciozne radnje koje se mogu pratiti. Može se lako konfigurirati na što će se alarmi oglašavati i zabilježiti bilo kakav zločudan kod koji je poslan u sustav s idejom širenja zaraze. Postoji više popularnih alata ovog tipa. Najpoznatiji su Suricata i Snort i oba su projekti otvorenog koda.

Suricata

Suricata je najpopularniji alat za detekciju napada. Koristi višedretveni način rada koji omogućuje pregledavanje velike količine prometa i protokola koji se koriste. Mana ovog alata je što troši dosta resursa i zna paliti alarm na promet koji nije maliciozan. Može se birati što će se raditi s prometom za koji je upaljen alarm. Opcije su samo zabilježavanje ili i odbacivanje paketa. U kontekstu TSO mamaca isprobana je implementacija Suricate na usmjeritelju. Potražena je docker slika koja sadrži ovaj alat i lako se konfigurira kako ograničenja IMUNES-a ne bi pravila probleme pri pokretanju. Pri svakoj od ponuđenih opcija konfiguraciji, IMUNES zbog iznimki nije mogao pokrenuti eksperiment.

Snort

Sljedeća opcija bio je Snort koji radi u jednodretvenom načinu rada i lakši je za korištenje. Nije pronađena gotova docker slika koja bi bila kompatibilna za potrebe ove mreže pa je napravljena nova. Instaliran je Snort alat i potrebna konfiguracija je dodana u kontejner. U konfiguraciji se bilježi svaka vrsta prometa. Uspješno se bilježe logovi i zapisuju se u log folder na usmjeritelju. Nije uspjelo konfiguriranje snorta da automatski šalje logove na log poslužitelj, ni preko UDP portova ni preko TCP portova. Python skripta koja je slušala promet na portu 514 nije primila nikakve podatke. Razmatrala se opcija izrade skripte koja bi slala log datoteku svaki put kada se njen sadržaj promijeni preko TCP porta, no na kraju je odabrana druga opcija koja je objašnjena u sljedećem dijelu rada.

Wireshark

Wireshark je alat koji služi za analizu mrežnog prometa koji se može koristiti u stvarnom vremenu ili u offline načinu rada gdje se sve zapisuje u datoteke koje se kasnije mogu koristiti za analiziranje. Odlučeno je koristiti TShark, koji je sastavni dio paketa Wireshark i služi za automatizaciju hvatanja paketa što se u ovom slučaju koristi za povezivanje s drugim servisima kao što je Netcat koji taj promet šalje na log poslužitelj za daljnju analizu i skladištenje. Ima mnoštvo materijala o korištenju i zajednicu koja aktivno raspravlja o mogućnostima ovog alata. Instalacija je jednostavna koristeći Dockerfile i nije izazovan za korištenje. Nakon instalacije alata postavljena je bash skripta koja se pokreće uz docker kontejner i čeka dok se ne uspije uspostaviti konekcija na log poslužitelj na portu 8081 te se zatim pali TShark i počinje slati promet na poslužitelj. Taj promet se po potrebi može testirati kroz VirusTotal koristeći njihov API.

5.1.4. Log poslužitelj

Log poslužitelj je objedinjeni okvir za prikupljanje, pohranu i nadzor podataka dnevnika koje generiraju različiti servisi i aplikacije. Ovi dnevnici mogu sadržavati informacije o prilikama, pogreškama i upozorenjima koje su se dogodile unutar okvira. Razlog za Log poslužitelj je osiguravanje skrivenog mjesta gdje se može pristupiti podacima logova iz različitih izvora i analizirati ih. U kontekstu ovog mamca, Implementiran je jednostavni Python poslužitelj koji je smješten unutar Docker slike koja se pokreće na čvoru LogServer u trenutku stvaranja eksperimenta. Poslužitelj sluša na portu 8080 za sav promet koji mu dolazi, a zatim u direktorij "logs" nadopisuje sve informacije koje je primio u datoteku s nazivom čvora od koje ih je primio.

Glavni dio klijentskog koda:

```
...
class LogFileHandler(FileSystemEventHandler):
    # sto raditi pri zabiljezenoj modifikaciji datoteke
    def on_modified(self, event):
        if not event.is_directory:
            file_path = event.src_path
            send_new_logs(file_path)

    # sto raditi pri kreiranju datoteke
    def on_created(self, event):
        return
```

```

def monitor_logs():
    # postavljanje nadziranja logova
    event_handler = LogFileHandler()
    observer = Observer()
    observer.schedule(event_handler, DIRECTORY, recursive=False)
    observer.start()

    try:
        while True:
            time.sleep(1)
    except KeyboardInterrupt:
        observer.stop()

    observer.join()
...

```

Također je napravljena skripta koja sluša promet na portu 8081 i zapisuje logove u datoteku "in.log", a ona se sprema u direktorij "logs". Razlika između ove skripte i one na portu 8080 je u tome što poslužitelj na portu 8081 zapisuje podatke u bajtovima, dok poslužitelj na portu 8080 zapisuje podatke u UTF-8 formatu. Razlog za to je što prva skripta prati promjene u log datotekama unutar čvorova koji šalju podatke tom poslužitelju, dok druga skripta direktno prosljeđuje promet na poslužitelj. Ovo je postignuto pomoću python skripti koje se pokreću na svakom čvoru i koriste paket watchdog kako bi pratili promjene u folderu s logovima. Kada se promjena dogodi, promet te datoteke se šalje na Log poslužitelj. Na usmjeritelju je postavljen docker kontejner s instaliranim alatom TShark, koji sa sučelja eth0 bilježi sav promet i šalje ga na log poslužitelj radi daljnje obrade.

Glavni dio koda za slanje prometa koji dođe na usmjeritelj:

```
...
# stvaranje novog procesa koji ce pokretati TShark na sucelju eth0
tshark_command = ["tshark", "-i", "eth0", "-w", "-"]
process = subprocess.Popen(tshark_command, stdout=subprocess.PIPE)

try:
# spajanje na server
client_socket .connect(( server_ip , server_port ))
print ("Connected to the remote server .")

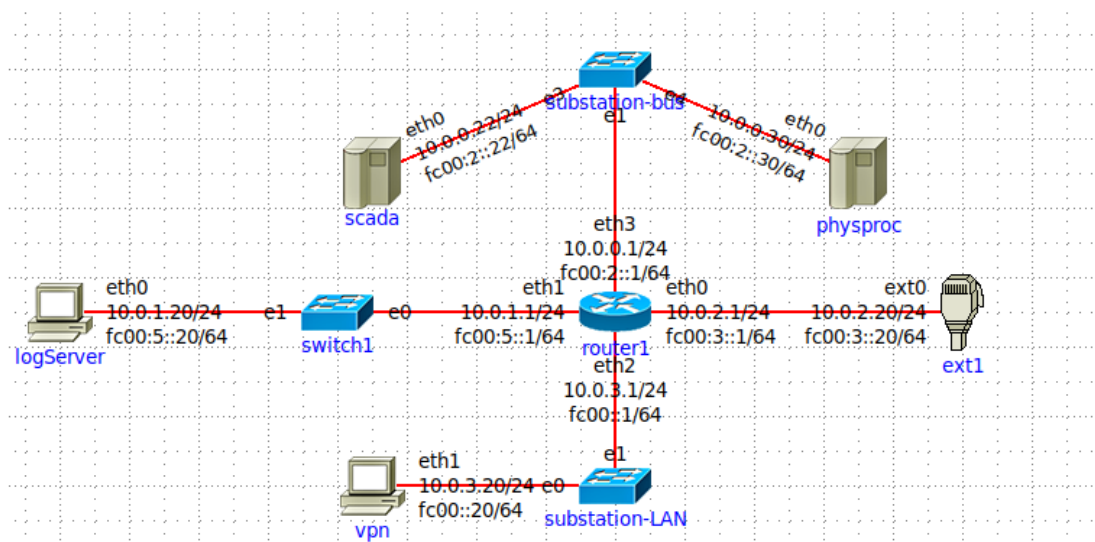
# postavljanje u ne blokirajuci nacin rada
client_socket .setblocking (0)

# koristeci select gledamo postoji li zabiljezen promet na tsharku koji mozemo prosljediti
while True:

    ready_sockets , _, _ = select .select ([ process . stdout ], [], [], 0)
    if ready_sockets :
        # procitaj pakete
        packet = ready_sockets [0]. readline ()
        packet = "ROUTER:" + packet.decode('utf-8')

        # posalji pakete na log server
        client_socket .sendall (packet)

...
```



Slika 5.5: Završna topologija TSO mamca

6. Zaključak

Ova istraživanja naglašavaju važnost razvijanja i primjene sigurnosnih mehanizama u kontekstu prijenosnih sustava. Kombinacija tehnologija poput Docker-a i mamaca može pružiti snažan okvir za detekciju, zaštitu i analizu sigurnosnih prijetnji. Mamci postaju sve napredniji i teže ih je detektirati, ali ne predstavljaju dovoljnu zaštitu sustava sami po sebi. Treba razmišljati i o ljudskom faktoru koji se većinom zloupotrebljava za dobivanje povjerljivih informacija unutar neke organizacije. To se može postići uz dobru politiku organizacije kao npr. korištenje principa najmanjih privilegija i osvještavanje njihovih zaposlenika. Pri konfiguriranju mamca trebamo ga napraviti dovoljno zanimljivim napadačima kako bi osigurali da se u njemu zadrže. To se može postići postavljanjem naizgled ranjivih podataka u sustav, ali ne smije ga se graditi da bude očito da nije stvarni sustav. Ulagajući u navedene tehnike i rješenja može se spriječiti financijska šteta te se osigurava sigurniji tok informacija kroz organizaciju. Ovaj rad pruža temelj za daljnje istraživanje i razvoj sigurnosnih rješenja u ovom području.

LITERATURA

- [1] Docker overview. URL <https://docs.docker.com/get-started/overview/>.
- [2] What is a honeypot. URL <https://www.fortinet.com/resources/cyberglossary/what-is-honeypot/>.
- [3] Industrial control system. URL <https://www.trendmicro.com/vinfo/us/security/definition/industrial-control-system>.
- [4] What is open source intelligence (osint)? URL <https://www.sentinelone.com/cybersecurity-101/open-source-intelligence-osint/>.
- [5] Općenito o scada sustavu. URL <https://elektrokem.hr/ek-sustavi/cijena/opcenito-o-scada-sustavu>.
- [6] transmission_system_operator. URL https://www.entsoe-event.eu/transmission_system_operator.html.
- [7] abhijith. What is theharvester? URL <https://www.cybervie.com/blog/what-is-the-harvester/>.
- [8] TechTarget Contributor. Plc. URL [https://www.techtarget.com/whatis/definition/programmed-logic-controller-PLC#:~:text=A%20programmable%20logic%20controller%20\(PLC,drum%20sequencers%20and%20cam%20timers](https://www.techtarget.com/whatis/definition/programmed-logic-controller-PLC#:~:text=A%20programmable%20logic%20controller%20(PLC,drum%20sequencers%20and%20cam%20timers).
- [9] Ivan Kovačević. Honeypot - tehnička dokumentacija početnog prototipa.
- [10] Aleksandra Krkoleva Mateska, Petar Krstevski, i Stefan Borožan. Overview and improvement of procedures and practices of electricity transmission sys-

- tem operators in south east europe to mitigate cybersecurity threats. URL <https://www.mdpi.com/2079-8954/9/2/39>.
- [11] Zrinka Puljiz i Miljenko Mikuc. Imunes based distributed network emulator.
- [12] B.E. Sudeeptha Rudrapattana. Cyber-security analysis in smart grid scada systems: A game theoretic approach. Magistarski rad. URL <https://ttu-ir.tdl.org/bitstream/handle/2346/58205/RUDRAPATTANA-THESIS-2013.pdf?sequence=1>.
- [13] Ivana Šabić. Scada u elektroenergetskom sustavu. URL <https://repozitorij.etfos.hr/islandora/object/etfos%3A3520/datastream/PDF/view>.
- [14] Marin Šepac. Programirljivi logički kontroleri (plc). Magistarski rad. URL <https://repository.ffri.uniri.hr/islandora/object/ffri%3A816/datastream/PDF/view>.

Detekcija kibernetičkih napada i zaštita vanjskih sustava u kontekstu mamaca za operatora prijenosnog sustava

Sažetak

Inicijalno, istraživanje pokriva pojam operatora prijenosnog sustava, objašnjavajući od kojih se komponenti sastoji i koja im je namjena. Pruža detaljan uvod u temu SCADA sustava, bitnog dijela sustava za prijenos električne energije koji podržavaju upravljanje i nadzor infrastrukture. Njihovo razumijevanje i prepoznavanje ranjivosti ključno je za razvoj učinkovitih sigurnosnih mjera za zaštitu kritične energetske infrastrukture. Naglašava koncept OSINT-a, koji pruža vrijedne informacije o potencijalnim prijetnjama i napadačima. Korištenje OSINT-a u analizi sigurnosti transportnog sustava omogućuje kreiranje strategija zaštite na temelju stvarnih podataka. Također je opisano stvarno prikupljanje podataka na jednom operatoru prijenosnog sustava. Rad zatim ispituje Docker, popularan alat za izgradnju i upravljanje kontejnerskim okruženjima. Docker omogućuje brzu implementaciju i testiranje različitih softverskih rješenja, što je ključno za razvoj sigurnosnih mehanizama u okruženjima kibernetičkih napada. Sljedeći odjeljci objašnjavaju osnovne postavke i korištenje web emulatora IMUNES. Pomoću ovog alata razvijen je mamac za operatora prijenosnog sustava koji je detaljno opisan radu. Istraživanje se fokusira na koncept mamaca, tehniku za privlačenje i zadržavanje kibernetičkih napada, kako bi se analizirale i razumjele tehnike napada. Mamci su korisni alati za otkrivanje i sprječavanje prijetnji i pružaju vrijedne informacije o napadačima i njihovim metodama. U zadnjem dijelu govori se o dobrim praksama u njihovoj izgradnji te su detaljno opisani i sami postupci koji su poduzeti u konfiguriranju jednog takvog mamca.

Ključne riječi: SCADA, Docker, OSINT, IMUNES, Kibernetički napad

Detection of cyber attacks and protection of external systems in context of a honeypot system for a transmission system operator

Abstract

Initially, the research covers the concept of the transmission system operator, explaining what components they consist of and what their purpose is. Provides an in-depth introduction to the topic of SCADA systems, an essential part of power transmission systems that support infrastructure management and monitoring. Their understanding and recognition of vulnerabilities are essential for the development of effective security measures to protect critical energy infrastructure. It emphasizes the concept of OSINT, which provides valuable information about potential threats and attackers. The use of OSINT in the analysis of the security of the transport system allows the creation of protection strategies based on real data. The actual data collection of one transmission system operator is also described. The article then examines Docker, a popular tool for building and managing containerized environments. Docker enables rapid deployment and testing of various software solutions, which is crucial for developing security mechanisms in cyberattack environments. The following sections explain the basic settings and usage of the IMUNES network emulator. Using this tool, we developed a honeypot for a transmission system operator. Additionally, research focuses on the concept of honeypots, a technique to attract and contain cyberattacks, to analyze and understand attack techniques. Honeypots are useful tools for detecting and preventing threats and provide valuable information about attackers and their methods. In the last part, I talk about good practices in their construction and describe in detail the procedures I took in configuring such a honeypot.

Keywords: SCADA, Docker, OSINT, IMUNES, cyberattack