Mentoru Stjepanu Grošu na velikom strpljenju i pomoći i Ivanu Kovačeviću na svim savjetima i angažmanu

SADRŽAJ

1. Uvod

Kibernetičke vježbe bitan su dio izgradnje sposobnosti obrane tvrtki. Svrha kibernetičkih vježbi je uvježbavanje timova koji brane neki informacijski sustav kako bi u slučaju stvarnih napada reagirali što bolje i na taj način umanjili štetu koja nastaje organizaciji. Za provođenje vježbi potrebno je imati odgovarajući informacijski sustav koji će "plavi" tim braniti, a "crveni" napadati. S obzirom da se produkcijski sustavi ne smiju napadati, koriste se sintetički informacijski sustavi koji se nazivaju kibernetički poligoni[16]. Kibernetički poligoni predstavljaju simulaciju IT infrastrukture mreže neke organizacije. Infrastruktura je sačinjena od računala koje koriste zaposlenici organizacije poput programera i direktora. Koriste ih da bi obavljali svakodnevne zadatke, kao što je čitanje elektroničke pošte (eng. *e-mail*), korištenje uređivača teksta kao što je *Libreoffice*, pretraživanje stranica na internetu itd. Budući da su ljudi jedan od ranjivijih dijelova sigurnosne infrastrukture, malicizoni napadač može razmotriti opciju napadanja mreže preko infekcije računala djelatnika organizacije. Također, na mreži organizacije se nalazi baza podataka kojoj neki od korisnika imaju pristup. Baza podataka može sadržavati povjerljive informacije kao što su korisničke vjerodajnice, povjerljive osobne podatke ili možda čak brojevi kartica. Cilj završnog rada bio je priprema okoline za kibernetičke vježbe i demonstracija rješavanja zadatka u toj okolini. Kako bi se omogućilo višestruko korištenje komponenti kibernetičkih poligona i modularno postavljanje zadataka za vježbavanje, korisno je u što većoj mjeri automatizirati postavljanje kibernetičkog poligona i simuliranog informacijskog sustava.

2. Temeljni pojmovi kibernetičkih poligona i vježbi

U ovom poglavlju će se dodatno objasniti kako su koncipirani kibernetički poligoni i kako oni služe za provođenje kibernetičkih vježbi. Također, objasniti će se što je kibernetički lanac prekidanja i dati jedan mogući scenarij kako napadač prolazi kroz korake kibernetičkog lanca prekidanja. U tom scenariju, napadač će poslati zlonamjerna poruku elektronička pošte koja je zapravo *phisihing* poruka.

2.1. Kibernetički poligoni

Nacionalni institut za standarde i tehnologiju (NIST) definira kibernetičke poligone kao "interaktivne i simulirane prikaze koji se odnose na lokalnu mrežu organizacije, sustav, alate i aplikacije". Kibernetički poligoni su stvarne platforme za obuku stručnjaka iz područja informacijske i komunikacijske tehnologije protiv širokog spektra kibernetičkih napada i scenarija kibernetičkog ratovanja. U njima je moguće razviti scenarije koji repliciraju različite vrste kibernetičkih napada, kao što su zaustavljanje širenja zloćudnog programa (eng. *malware*) mrežom, napad uskraćivanja usluge (eng. *Denila of service*) i sl. Kibernetički poligoni poboljšavaju obuku operatera i korisnika unutar prepoznatljivih okruženja u identifikaciji i strategijama ublažavanja posljedica kibernetičkog napada. Obuka u umjetnom okruženju ubrzava učinkovito učenje najbolje prakse, a dinamička interakcija u stvarnom vremenu promiče dublje razumijevanje posljedica svake radnje. Grupe korisnika također mogu trenirati na daljinski dostupnoj platformi kako bi definirali, optimizirali i procijenili učinak koordiniranog odgovora na kibernetičke napade. Grupna obuka koja uključuje više timova koji se sastoje od različitih skupova znanja poboljšava svijest organizacije o kibernetičkoj situaciji i poboljšava vrijeme odgovora za prepoznavanje i zaustavljanje kibernetičkog napada [16]. "Kibernetički poligoni su sačinjeni od tri glavna elementa":

- jezgrene tehnologije

- infrastrukturne tehnologije
- *front-end* tehnologije

Jezgrena tehnologija kibernetičkog poligona govori o tome kako su realizirani strojevi na kojima se odvijaju kibernetičke vježbe. Za slučaj ovog završnog rada, jezgrena tehnologija je Docker, koja je bazirana na kontejnerizaciji. Infrastrukturne tehnologije su tehnologije koje pokreću i upravljaju jezgrene tehnologije. Služe kao poveznica između jezgrenih tehnologija i front-end tehnologija. Infrastrukturna tehnologija korištena za potrebe ovog završnog rada je *Docker Compose*. *Front-end* tehnologije povezuju krajnjeg korisnika kibernetičkog poligona sa strojevima, koji su ovom slučaju Docker kontejneri. X11 klijent-poslužitelj *front-end* tehnologija korištena za potrebe ovog završnog rada.

2.2. Kibernetičke vježbe

Vježbe su važna sastavnica obrazovnog procesa. Oni su oblik aktivnog učenja, a dobro je poznato da je aktivno učenje učinkovitije od tradicionalnog, pasivnog obrazovanja. Kibernetička sigurnost nije izuzeće od toga, a mnoge vježbe kibernetičke sigurnosti provode u bilo koje vrijeme, ciljajući i na početnike i stručnjake iz područja kibernetičke sigurnosti. Tip vježbe se kreće od individualnog jednostavnog oblika *Capture the Flag* tj. CTF vrste vježbi do velikih koordiniranih multinacionalnih vježbi koje uključuju nekoliko nacija i stotine ili čak tisuće stručnjaka [1].

2.3. Kibernetički lanac prekidanja

Napadi prolaze korake u kojima ih se u teoriji može otkrivati i zaustavljati. Kibernetički lanac prekidanja (eng. *kill chain*) je jedan način od načina sistematiziranja tih koraka. Kibernetički lanac prekidanja je model kibernetičke sigurnosti koji je stvorio Lockheed Martin koji prati faze kibernetičkog napada, identificira ranjivosti i pomaže sigurnosnim timovima da zaustave napade u svakoj fazi lanca [10]. Izraz lanac prekidanja preuzet je iz vojne terminologije, koja koristi ovaj izraz za opis strukture napada. Kibernetički lanac prekidanja ima sedam ključnih faza koje prikazane na slici 2.1:

1. izviđanje (eng. reconnaissance)
2. naoružavanje (eng. weaponization)
3. dostavljanje (eng. delivery)

4. iskorištavanje (eng. exploitation)
5. instalacija (eng. installation)
6. zapovijedanje i kontrola (eng. command and control)
7. radnja prema cilju (eng. actions on objectives)

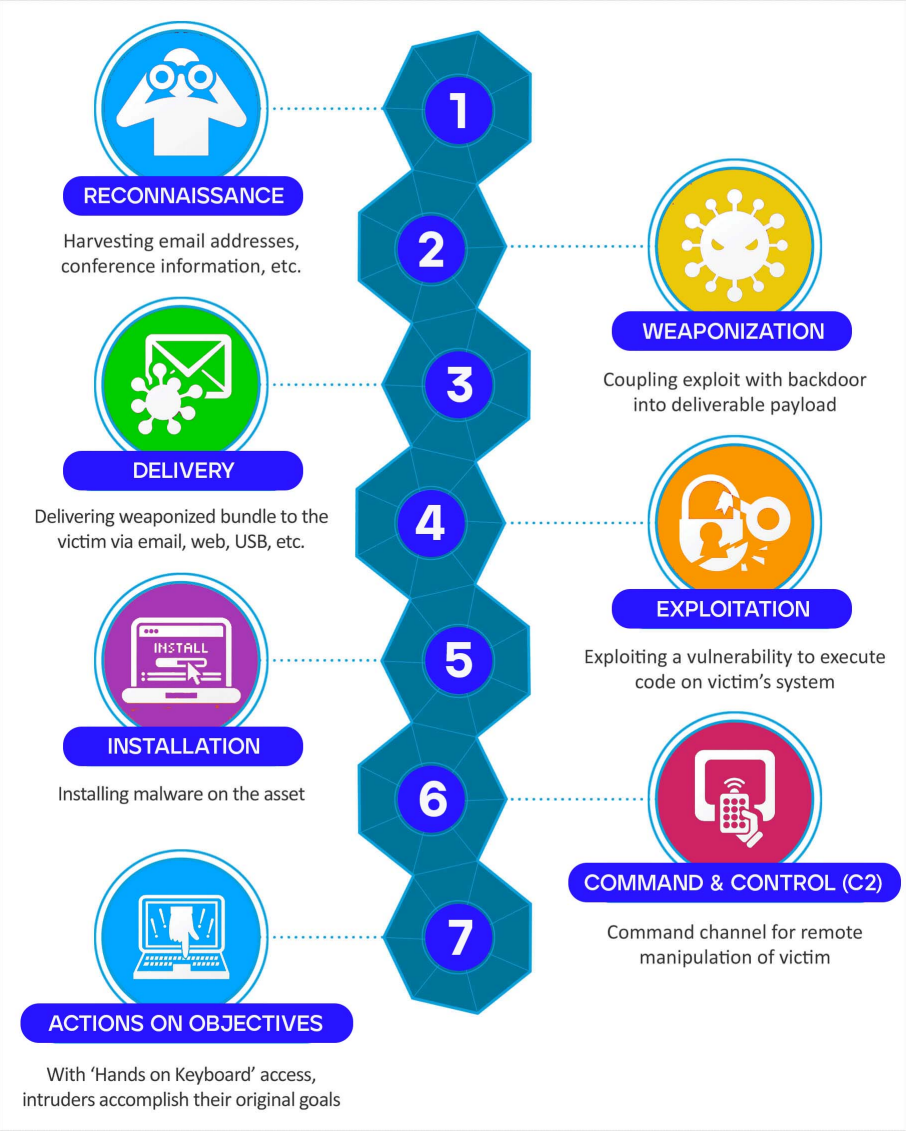
Izviđanje obuhvaća prikupljanje što više informacija o meti koji su javno dostupni. Naoružavanje je proces kreiranja tereta (eng. *payload*) koji će se dostaviti žrtvi. Dostavljanje obuhvaća metodu prenošenja *payloada* s napadačkog računala na računalo žrtve. Iskorištavanje je proces izvršavanja pošiljke na računalu žrtve koja će iskoristiti ranjivost koja je prisutna na računalu žrtve. Instalacija je korak u kojem *payload* koji je ranije poslan na računalo uspostavlja pristupnu točku na računalu žrtve kojoj će napadač kasnije pristupiti. Zapovijedanje i kontrola je dio kibernetičkog lanca prekidanja kada napadač uspostavi udaljenu sjednicu sa svog računala na računalo žrtve. Zadnji korak tj. radnja prema cilju je poduzimanje koraka na računalu žrtve gdje napadač pokušava ostvariti svoj glavni motiv napada, koji može biti preuzimanje osjetljiv podataka i sl.

2.4. Phishing mailovi

Phishing mailovi su zlonamjerne poruke elektroničke pošte čiji najčešći cilj je ukrasti osobne podatke, novac ili vjerodajnice žrtve. Phishing poruke se predstavljaju kao da dolaze od legitimnog izvora, npr. od kolege s posla ili od banke. U sebi mogu sadržavaju poveznicu na neku stranicu koja traži korisnika da upiše svoje podatke ili mogu sadržavati zlonamjernu pošiljku koji žrtva mora pokrenuti [11]. U svrhu ovog završnog rada, napadač će poslati žrtvi elektroničku poruku u kojoj se predstavlja da je zaposlenik iste organizacije kao žrtva i da uz poruku šalje datoteku u kojoj se nalaze podatci koji su žrtvi čine da su od važnosti.

2.5. Zlonamjerna pošiljka (eng. malicious payload)

U području kibernetičke sigurnosti, pošiljka se odnosi na zloćudni program kojeg napadač želi dostaviti na računalo žrtve s ciljem izvršavanja te pošiljke kako bi zarazio računalo žrtve. Zlonamjerna elektronička pošta obuhvaća fazu naoružavanja, dostavljanja i instalacije u kibernetičkom lancu prekidanja. Pošiljke mogu biti pohranjene u



Slika 2.1: Prikaz faza kibernetičkog lanca prekidanja[6]

obliku izvršne datoteke (.exe) ili u nekom neizvršnog obliku kao što je primjerice, .pdf, .jpeg i sl.

3. Korištene tehnologije

Računala u mreži su realizirana pomoću Docker kontejnera. Za simulaciju mreže je korišten *Docker-compose*. *Metasploit* i *Social Engineering Toolkit* su korišteni za kreiranje zlonamjerne elektroničke pošte, za slanje istog i za uspostavljanje udaljenog terminala da napadač može izvršiti arbitraran kod na računalu žrtve (eng. *remote code execution*) kako bi realizirao svoj motiv napada.

3.1. Docker

Docker je platforma otvorenog koda za razvijanje, postavljanje i upravljanje kontejneriziranim aplikacijama. Napravljen je da zamjeni virtualne strojeve. Dok virtualni strojevi virtualiziraju cijela računala, Docker je napravljen da virtualizira operacijski sustav. Docker pruža mogućnost upakiravanja i pokretanja aplikacije u izoliranom okruženju koje se naziva kontejner. Svaki Docker kontejner se izgrađuje iz Docker slika (eng. *image*). Postavke za izgradnju Docker kontejner su definirane unutar *Dockerfilea*. Izolacija i sigurnost omogućuju pokretanje više spremnika istovremeno na računalu korisnika. Kontejneri su kompaktni i sadrže sve što je potrebno za pokretanje aplikacije, tako da se korisnik ne mora oslanjati na programsku podršku koja je trenutačno instalirana na njegovom računalu. Docker koristi klijent-poslužitelj arhitekturu. Docker klijent komunicira s Docker poslužiteljskim procesom (eng. daemon) preko kojeg se izvršava izgradlja, pokretanje i distribucija Docker kontejnera. Docker klijent i poslužiteljski proces komuniciraju preko aplikacijskog programskog sučelja za prijenos reprezentacijskog stanja (eng. *Representational State Transfer Application Programming Interface*), skraćeno REST API, ili preko UNIX pristupnih točki ili preko mrežnog sučelja [4].

3.2. Docker-compose

Docker-compose je alat koji dolazi sa Dockerom koji nam omogućava da izgradimo i istovremeno pokrećemo više Docker kontejnera pomoću jedne naredbe. Također, pomoću njega možemo postaviti zajedničke spremnike ili konfigurirati da se kontejneri nalaze na istoj mreži. Prvotno se pokreće naredba `docker compose build` kako bi se svi potrebni Docker slike izgradile. Nakon toga, naredbom `docker compose up` se pokreće cijeli *stack* Docker kontejnera. *Docker-compose* u sebi ima ugrađenu funkcionalnost povezivanja Docker kontejnera u mrežu. Podrazumijevane mrežne postavke su da se svi Docker kontejneri nalaze na istoj mreži u mostnom (eng. `bridge`) načinu rada [3].

3.3. Metasploit

Metasploit je modularna platforma za penetracijska testiranja koja omogućava kreiranje, testiranje i izvršavanje metoda iskorištavanja (eng. exploit). Metode iskorištavanja korisnik može sam izraditi ili preuzeti iz baze podataka koja sadrži najnovije već otkrivene i modularne metode iskorištavanja. U sebi sadrži zbirku često korištenih alata koji pružaju cjelovito okruženje za penetracijsko testiranje i razvoja metoda iskorištavanja. Za pokretanje *Metasploita* potrebno je upisati naredbu `msfconsole` nakon koje će se pokrenuti interaktivni ljuska koja omogućava korištenje alata za testiranje sigurnosnih ranjivosti, skeniranje mreža, izvođenja napada i izbjegavanje detekcije da se događaju sumnjive aktivnosti. Prilikom provođenja penetracijskog testa često je izazov pratiti sve što je učinjeno na ciljnoj mreži ili prema njoj. *Metasploit* ima ugrađenu podršku za bazu podataka u koju se mogu spremati informacijama o skeniranju i daje nam mogućnost uvoza i izvoza rezultata skeniranja. S ovim alatima, *Metasploit* obuhvaća svih sedam faza kibernetičkog lanca prekidanja [15] . Nadalje u radu će se pokazati kako se *Metasploit* može koristiti za kreiranje maliciozne pošiljke i uspostavljanje udaljene sjednice na računalu žrtve.

3.4. Social Engineering Toolkit

Jedan od najčešće korištenih alata koji se koristi za "social engineering" napade je *Social Engineering Toolkit* ili skraćeno SET [8]. To je alat otvorenog koda koji nam omogućava razne napade, kao što su kloniranje web stranice kako bismo ukrali nečije

vjerodajnice, ili, što će biti korišteno u ovom slučaju, slanje *phisining-mail*-ova i slanje malicioznih pošiljki.

3.5. Mail server

Docker Mailserver[9] je prethodno podešena Docker slika koji na sebi ima postavljene alate Postfix, Dovecot, Amavis i SpamAssasin. Kako bi naša implementacija radila, potrebno će biti promjeniti prepostavljanu Postfix, Dovecot i Amavis konfiguraciju. Postfix je *Mail Transfer Agent* (MTA) koji služi kao relej između mail servera na Internetu. U ovom završnom radu, Postfix služi za primanje i slanje elektroničke pošte. Dovecot je *Mail Delivery Agent* (MDA) koji služi za dostavljanje e-mail poruka na završnu točku tj. na račun elektorničke pošte nekog korisnika. Također, Dovecot je korišten za konfiguraciju korisničkih računa. Amavis je alat koji se koristi za filtriranje elektroničke pošte prije nego se ona prikaže kranjem korisniku. Amavis može koristi za gledanje zaglavlja elektroničke pošte i ako poruka nema sva potrebna zaglavlja, onda se ne proslijeđuje do krajnjem korisnika.

3.6. Kali Linux

Kali Linux distribucija Linuxa koja je temeljena na Debianu. Kali Linux sadrži alate usmjerene na različite zadatke informacijske sigurnosti, kao što su penetracijsko testiranje, sigurnosno istraživanje, računalna forenzika i sl. Dolazi sa ugrađenom *Metasploit* konzolom, SET-om, alati koji potrebni će biti potrebni za demonstraciju napada nad postavljenom mrežom. Kali Linux u sebi sadržava te alate koji mogu služiti za svaki dio kibernetički lanac prekidanja. Primjerice, alat `nmap` se može koristiti za izviđanje. `nmap` služi za prolanaženje otvorenih pristupnih vrata na mreži. Alat *Metasploit* se može koristiti za naoružavanje, a za dostavljanje tereta (eng. *payload*) se može koristiti SET. Osim *Metasploita*, primjer alata za iskorištavanje je `SQLmap`. `SQLmap` je alat koji služi za automatiziranje napada ubacivanja SQL izraza (eng. *SQL injection*). Za instalaciju, zapovijedanje i kontrola i finalno napredovanje prema cilju, najpogodniji alat je *Metasploit*. Za zapovijedanje i kontrolu i finalno napredovanje omogućava uspostavljanje udaljene sjednice preko koje se mogu zadati naredbe koje se trebaju izvršiti na računalu žrtve.

4. Postavljanje okruženja

Unutar ovog poglavlja je opisano kako su postavljeni *Dockerfileovi* pojedinih računala. Također je opisano kako je postavljen poslužitelj elektroničke pošte, kako su pohranjeni podatci korisnika mreže i kako je postavljen X11 klijent kako bi se moglo pristupiti aplikacijama unutar Docker kontejnera koje zahtijevaju grafičko sučelje. Cilj postavljanja je rekreirati konkretan IT sustav koji će služiti za provedbu kibernetičkih vježbi.

4.1. Dockerfile korisničkih računala

Prvi korak je bio konfigurirati *Dockerfileove* koji predstavljaju konfiguraciju pojedinih računala. Pomoću njega možemo instalirati dodatne pakete, postaviti varijable okoline (eng. *environment variables*), dodati novog korisnika i grupu i postaviti početni direktorij u kojem ćemo se nalaziti prilikom ulaska u Docker kontejner. Primjer konfiguracije *Dockerfilea* za računalo razvojnog programera izgleda:

```
1  FROM ubuntu:16.04
2
3  WORKDIR /installation
4
5  COPY install.sh install.sh
6
7  RUN chmod +x install.sh
8
9  # instalacija libre offica, codea, firefoxa i thunderbirda
10 RUN ./install.sh
11
12 RUN useradd -ms /bin/bash junior
13 RUN groupadd developer
14
15 ENV DISPLAY :3
16
```

```
17 USER junior:developer
18 WORKDIR /home/junior
19
20 CMD ["tail", "-F", "anything"]
```

Ispis 4.1: Dockerfile junior programera

Na 1. liniji specifiramo da želimo pokrenuti Docker sliku od *Ubuntu* verzije 16.04. Ova verzija je specifično odbrana jer na toj verziji, tj. Kernela verzije 4.4 postoji ranjivost koja omogućuje povisivanja privilegija (eng. Local Privilage Escalation) [2]. Za tu ranjivost postoji *Metasploit module* što znači da ju možemo iskoristiti pomoću *Metasploit*-ove konzole. Na 3. liniji smo postavili koji će biti početni direktorij kada se spojimo na Docker kontejner. Na 5., 7. i 10. liniji kopiramo skriptu sa našeg računala na Docker kontejneru i izvršavamo ju na kontejneru. Skripta `install.sh` izgleda:

```
1  # install visual studio code
2  wget -qO- https://packages.microsoft.com/keys/microsoft.asc | gpg --
       dearmor > packages.microsoft.gpg
3  install -o root -g root -m 644 packages.microsoft.gpg /etc/apt/
       trusted.gpg.d/
4  sh -c 'echo "deb [arch=amd64,arm64,armhf signed-by=/etc/apt/trusted.
       gpg.d/packages.microsoft.gpg] https://packages.microsoft.com/
       repos/code stable main" > /etc/apt/sources.list.d/vscode.list'
5  rm -f packages.microsoft.gpg
6
7  apt install apt-transport-https
8  apt update
9  apt install -y code -v 1.57.0
10
11 # install libre office
12 add-apt-repository ppa:libreoffice/ppa
13 apt-get update
14 apt-get install -y libreoffice
15
16 apt install firefox -y
17 apt install thunderbird -y
18
19 apt install iputils-ping -y
20 apt install net-tools -y && apt install nano -y
```

Ispis 4.2: Install.sh skripta za instalaciju softverskih apketa

Kako se *Visual Studio Code* ne nalazi u službenim *Ubuntu* repozitorijima, moramo ga prvo dodati da bismo ga instalirali. To obuhvaćaju linije 2 do 5. Insaliramo ver-

ziju *1.57.0* jer za nju postoji RCE ranjivost (eng. remote code execution) i ima CVSS vrijednost od 7.8. CVSS vrijednost dodjeljuje organizacija NIST tj. *National Institute of Standards and Technology* Ujedinjenih Američkih Država. Ta je vrijednost broj iz intervala [1, 10] i daje procjenu koliko štete može nanjeti pojedina ranjivost. Za procjenu ovog broja se uzima u obzir koliko je lagano iskoristiti sljedeću ranjivost, koliko je rasprostranjen software, koje razine ovlasti su potrebne da se iskoristi ranjivost itd. NIST pruža kalkulator uz pomoć kojeg možemo izračunati koja je CVSS vrijednost neke ranjivosti [13]. Nažalost, za tu ranjivost ne postoji *Metasploit* modul. Skriptu također koristimo za instaliranje *Libre officea*. On se također ne nalazi u službenim *Ubuntu* repozitorijima. Preko `install.sh` se također instalira *Firefox*, sučelje za pristup elektroničkoj pošti *Thunderbird* tj. MUA (eng. mail user agent), alat `ping` kako bismo mogli provjeriti dostupnost ostalih računala na mreži i terminalni uređivač teksta `nano` za izmjenu konfiguracijskih datoteka direktno na Docker kontejneru. Na linijama 12, 13 i 17 dodajemo novog korisnika i novu grupu `developer` i dodajemo korisnika u tu grupu. Sa zastavicom "-m" specificiramo da želimo kreirati korisnikov direktorij u *home* mapi i sa zastavicom "-s" govorimo da se koristi `/bin/bash` ljuska kao zadana korisnikova ljuska. Na liniji 15 dodajemo varijablu okoline (eng. environment variable) da poslužitelj virtualne radne površine (eng. desktop) `Xephyr` zna na koji pristup (eng. *port*) se treba spojiti. Konačno, na liniji 18 postavljamo koji će biti početni direktorij kada uđemo u kontejner. Docker kontejneri su napravljeni tako da kada izvrše sve potrebne naredbe koje se nalaze u *Dockerfileu* se automatski ugase. Na liniji 20 govorimo koju naredbu ćemo izvršiti nakon što se izvrše sve ostale naredbe iz *Dockerfilea*, a ta naredba baš postiže to da se kontejner ne ugasi. *Dockerfileovi* su analogno ovome postavljeni. Jedina razlika između CEO-a i ostalih korisnika je da CEO nema instaliran *Visual Studio Code*.

4.2. Mail server konfiguracija

Na poslužitelju elektroničke pošte nalaze se servisi koji su zaduženi za različite stvari. Svi oni su pokrenuti u obliku poslužiteljskih procesa (eng. daemon). Poslužiteljski procesi rade u pozadini i nadziru sustav ili pružaju funkcionalnost drugim procesima. Kada se promijeni konfiguraciju nekog servisa, primjerice Amavisa, taj proces se treba ponovno pokrenuti pomoću naredbe `service [ime servisa] restart`.

4.2.1. Dodavanje korisničkih računa

Za dodavanje korisničkih računa korištena je skripta `setup.sh` koja dolazi u paketu s Mailserverom. Ona nam omogućava dodavanje korisničkog e-mail računa naredbom:

```
./setup.sh email add <user@domain> [<password>]
```

Ova naredba automatski konfigurira korisničke Dovecot račune kako bi oni mogli pristupiti poslužitelju elektroničke pošte. Podatke korisnika pohranjuje unutar `/etc/dovecot/userdb` datoteke. U toj datoteci su pohranjena korisnička imena u jasnom tekstu, šifre koje su enkriptirane sa SHA512 algoritmom i lokacija gdje se spremaju korisničke poruke. Poruke se spremaju u `/var/mail/mail.cyberpoligon.local/[ime korisnika]`.

4.3. Postfix konifiguracija

Unutar Postfixove konfiguracijske datoteke `main.cf` nalaze se konfiguracije za ime poslužitelja, SASL postavke, kako Postfix sprema poštu koju dobije i postavke koje sprečavaju nadolazeće neželjene poruke (eng. spam) [14]

4.3.1. Spremanje pošte i korisnički računi

Nadalje, kod koji se nalazi niže pokazuje kako izgledaju Postfix postavke za spremanje pošte i gdje su spremljeni korisnički računi elektroničke pošte. Na liniji 2 je navedeno da koristimo LMTP (local mail transfer protocol) od Dovecota. LMTP je alternativa SMTP-u za situacije kada prijemna strana nema implementiran red (eng. queue) za elektroničku poštu. Red služi kao među zona između pošiljatelja i primatelja poruke. Na liniji 3 navodimo da za svu poštu koja je namijenjena domenama koje se nalaze u direktoriju `/etc/postfix/vhost`, je dostavljena preko *virtual trasport* koji je naveden na prijašnoj liniji. Na liniji 4 je navedeno gdje su spremljene sve validne adrese primatelja elektroničke pošte i gdje se sprema pristigla pošta. Na linije 5 je navedeno da sva pošta koja je došla na domenu `cyberpoligon.local` će se preusmjeriti na lokalni klijent koji će dostaviti poštu do svog finalnog odredišta.

```
1 # Mail directory
2 virtual_transport = lmtp:unix:/var/run/dovecot/lmtp
3 virtual_mailbox_domains = /etc/postfix/vhost
4 virtual_mailbox_maps = texthash:/etc/postfix/vmailbox
```

```
5 virtual_alias_maps = texthash:/etc/postfix/virtual
```

Ispis 4.3: Postfix postavke kako se sprema pošta

4.3.2. Prevencija neželjene elektroničke pošte

U Postfix `main.cf` konfiguraciji se može postaviti što je dozvoljeno kada se na poslužitelj netko spoji npr. preko `telnet-a`. U kodu koji je priložen ispod ovog teksta su navedene postavke koje spriječavaju osobu koja se spoji na poslužitelj da ne izvršava ili da mora izvršiti pojedine naredbe. Na liniji 1 je navedno da, ako se neka osoba želi spojiti na mail klijent, mora izvršiti `HELO` naredbu, koja dohvaća podatke od poslužitelja na kojeg se ta osoba spaja. Na liniji 2 je navedno koje adrese su dozvoljene uz `HELO` naredbu. Uz tu naredbu smije samo stajati mreža `cyberpoligon.local`. Također ta osoba ne može poslati zapis koji nije FQDN i naredba permit eksplicitno navodi da vrijede zadana politika Postfixa. Na liniji 3 nalaze se ograničenja koja su definirana nad `RCPT TO` naredbom. Parametar `permit_sasl_authenticated` dozvoljava zahtjev kada je klijent autentificaran preko SASL protokola. Parametar `permit_mynetworks` dozvoljava da klijent pošalje zahtjev ako šalje s neke od adresa koje su napisane uz parametar `mynetworks`.
Konačno parametar `reject\_unauth\_destination` govori Postfixu da odobri mail kada domena napisana uz `RCPT TO` zahtjev odgovara parametru `mydestination` koji u ovom slučaju predstavlja da promet mora biti upućen na `cyberpoligon.local` domenu. Analogno paramteri vrijede za `smtpd\_client\_restrictions` što upravlja restrikcijama za klijenta koji se pokušava spojiti na poslužitelj. Na liniji 5 nalaze se ograničenja za `SEND FROM` naredbu. Paramteri koji se nalaze i na prijašnje dvije linije, označavaju istu stvar, samo u kontekstu `SEND FROM` naredbe. Jedini dodatak je parametar `reject_unknown_sender_domain` govori da obaci poruke u kojima završno odredište poruke nije ovaj Postfix poslužitelj. Konačno na liniji 6 se onemogućava naredba `VRFY`, koja omogućava korisniku da provjeri koje su važeće adrese ovog poslužitelja.

```
1 smtpd_helo_required = yes
2 smtpd_helo_restrictions = permit_mynetworks,
     reject_invalid_helo_hostname, permit
3 smtpd_recipient_restrictions = permit_sasl_authenticated,
     permit_mynetworks, reject_unauth_destination
4 smtpd_client_restrictions = permit_sasl_authenticated,
     permit_mynetworks, reject_unauth_destination
```

```
smtpd_sender_restrictions = permit_sasl_authenticated,
   permit_mynetworks, reject_unknown_sender_domain
disable_vrfy_command = yes
```

Ispis 4.4: Postfix postavke za prevenciju neželjene pošte

4.4. Dovecot konfiguracija

Dovecot je alat koji koristimo kako bi korisnici mogli pristupiti svojim porukama preko `IMAP` i `POP` protokola. `POP` protokola skladišti poruke na računalu s kojeg pristupamo, dok `IMAP` skladišti poruke na samom serveru, znači možemo pristupiti njima s bilokojeg računala. Kao što je navedeno u Postfix `main.cf` datoteci, `SMTP` autentifikacija je implementira pomoću Dovecot `SASL-a`. To znači da kad god korisnik pristupa poslužitelju elektroničke pošte mora upisati svoje korisničko ime i lozinku. Postavljeno je da authetifikacija za `MUA` (eng. mail user agent) ide preko `TCP` porta 587. `SASL` je akronim za *Authentication and Security Layer* i on dodaje autentifikaciju na konekcije protokole kao što je `TCP`. `SASL` također odvaja mehanizme autentifikacije od aplikacijskih protokola, što teoretski dopušta bilo kojem mehanizmu provjeru autentifikacije kojeg `SASL` podržava da se koristi u bilo kojem aplikacijskom protokolu koji koristi `SASL`. [5].

4.5. Amavis konfiguracija

Amavis je alat koji služi za filtriranje nadolazećih poruka. Može detektirati neželjenu poštu, viruse i maliciozne programe. Također, filtrira elektroničku poštu koja je sintaktički neispravna ili nemaju sve odgovarajuće zaglavlja. Za svrhu testiranja, omogućeno je unutar */etc/amavis/conf.d/49-docker-mailserver* da na poslužitelj stižu poruke koje nemaju sva potrebna zaglavlja. U protivnom, ako se poruka šalje preko *Social engineering toolkita*, poruka se ne dostavi do klijenta jer ju Amavis blokira. Potrebno je samo promjeniti paramter `final_bad_header_destiny` koji govori Amavisu da propušta promet bez da provjerava jesu li zaglavlja poruke validna.

```
# Bounce spam, the default option for buster is D_PASS to deliver
$final_spam_destiny       = D_BOUNCE;
$final_bad_header_destiny = D_PASS;
```

Ispis 4.5: Amavis filter neispravnih zaglavlja

4.6. X11 poslužitelj

Za prikazivanje grafičkog sučelja Docker kontejnera korišten je `X11` protokol. Korištena je arhitektura klijent-poslužitelj, gdje klijent predstavlja aplikaciju na udaljenom stroju, a poslužitelj je `X11` aplikacija koja se nalazi na kojem želimo prikazati grafičko sučelje. Neke od mogućih poslužiteljskih aplikacija su XEphyr, TigerVNC, XQuartz itd. Kako bih `X11`klijent-poslužitelj ispravno radio, klijent i poslužitelj trebaju sinkronizirati konfiguracijske datoteke za `X11` protokol kao što je objašnjeno u poglavlju 5.2 . Prvotno, potrebno je pokrenuti `X11` poslužitelj na računalu na kojem želimo prikazati grafičko sučelje. Primjerice, za XEphyr naredbom `Xephyr 800x600 :3` pokrećemo poslužitelj na pristupnim vratima `:3` s veličinom prozora 800x600. Nadalje, potrebno je postaviti `DISPLAY` okolinsku varijablu (eng. *environment variable*) na računalu na kojem se nalazi aplikacija od koje se želi prikazati grafičko sučelje, tako da odgovara pristupnim vratima poslužiteljske aplikacije. To je moguće napraviti s naredbom

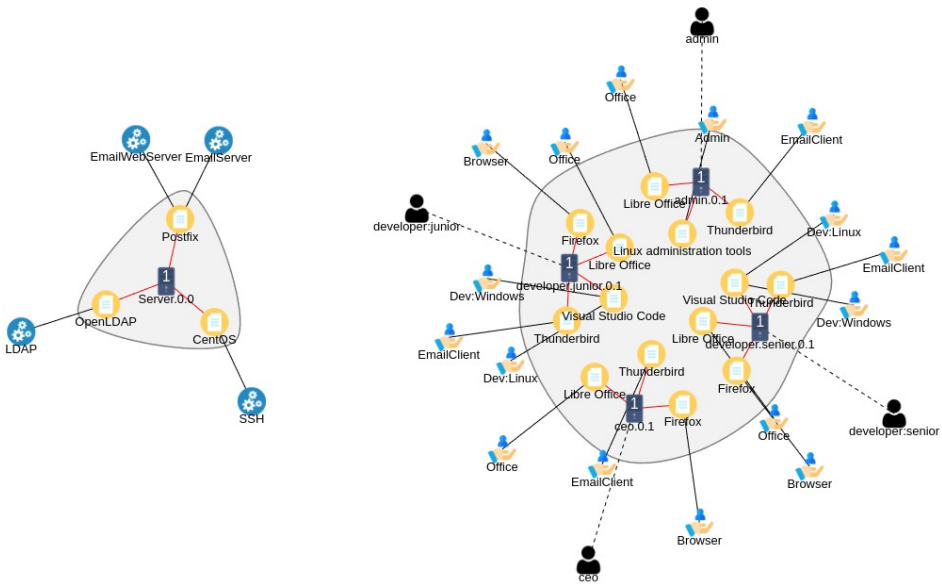
```
export DISPLAY=:3
```

Nakon ovoga, preko poslužitelja se grafičko sučelje udaljene aplikacije prikazuju na lokalnom računalu i preko njega se unos tipkovnice i miša šalje na udaljenu aplikaciju.

5. Opis implementacije

U ovom poglavlju opisano je kako su Docker kontejneri povezani u mrežu i kako je ostvaren dijeljeni direktorij prisutan na mreži opisan unutar `docker-compose.yml` konfiguracijske datoteke. Također je objašnjeno kako je ostvarena poveznica između mreže i virtualnog stroja.

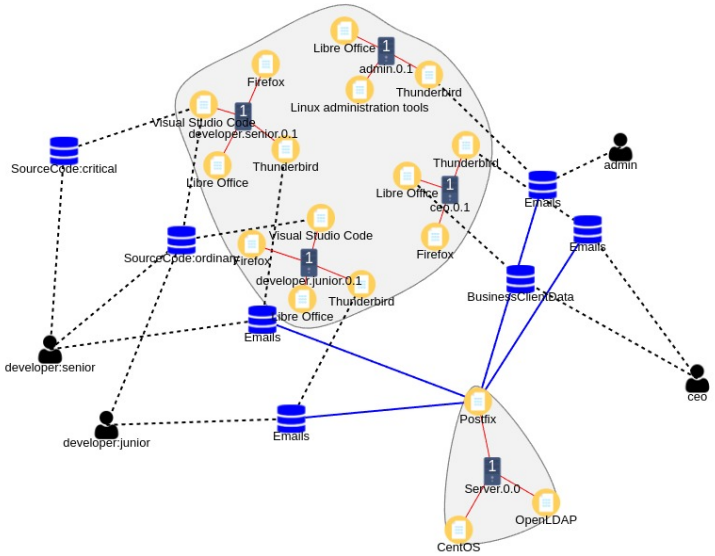
5.1. Izgled mreže



Slika 5.1: Izgled mreže računala

Mreža na slici je sačinjena od četiri korisničkih računala i jednog poslužitelja elektroničke pošte čije usluge koristi svako računalo na mreži. Na slici su plavom ikonicom kotačića prikazane usluge koje nudi poslužitelj elektroničke pošte. Softverski paketi koji su instalirani na računalima su prikazani žutom ikonicom, a ruka simbolizira usluge koje koriste korisnici na svojim računalima. Također, korisnici mogu pristupiti

dijeljenim direktorijima. Na slici 4.2 je prikazano kojim korisnicima je omogućen pristup pojedinim direktorijima. Na primjer, senior programeru je omogućen pristup i osjetljivom i normalnom izvornom kodu, a junior programeru je omogućen pristup samo normalnom izvornom kodu. Nadalje, CEO je jedina osoba koja može vidjeti podatke poslovnih korisnika.



Slika 5.2: Dijeljeni direktoriji

5.2. Docker compose

Docker compose je alat koji dolazi instaliran s Dockerom i omogućava definiranje sustava koji koriste više Docker kontejnera uz pomoć jedne konfiguracijske datoteke. U datoteci `docker-compose.yml` nalazi se konfiguracija koja definira i pokreće cijelu kibernetičku mrežu. Pod `services` su definirani koji Docker kontejneri čine mrežu. Na ispisu 5.1. je prikazano kako izgleda konfiguracija za zaposlenika organizacije.

```
developer_junior:
    build: ./docker/junior_workstation
    volumes:
      - source_code_ordinary:/home/junior/code
      - /tmp/.X11-unix:/tmp/.X11-unix
    restart: always
```

```
7     command: tail -F anything
8     networks:
9       - company_machines
10    extra_hosts:
11      - "cyberpoligon.local:172.19.1.20"
```

Ispis 5.1: Konfiguracija za zaposlenika organizacije unutar Docker Composea

Na liniji 2 je definirano iz koje Docker slike će se kontejner izgraditi. Pod `volumes` je opisano koji su direktoriji kojima kontejner ima pristup. Primjerice, na liniji 4 je opisano da dijeljeni direktorij`source\_code\_ordinary` se mapira na direktorij `/home/junior/code` unutar samog kontejner. Kako bih X11 mogao raditi, trebaju se kopirati konfiguracijski podatci s domaćinskog stroja na Docker kontejner, što je opisano na liniji 5 za stroj `developer\_junior`. Na liniji 7 je opisano koja će se naredba izvršiti prilikom pokretanja ovog kontejnera. Ova naredba je izabrana kako bi Docker kontejner ostao u pripravnom stanju dok ga se ne isključi. Naredba radi uz pretpostavku da datoteka `anything` ne postoji na računalu. Naredba `tail` je napravljena kako bi se ispisalo zadnjih 10 linija datoteke. Uz zastavicu `-F` `tail` će provjeravati je li se datoteka `anything` ažurirala. Ako pretpostavka vrijedi tj. datoteka `anything` ne postoji, nikakve se promjene na njoj neće dogoditi i `tail` će nastaviti provjeravati je li se ta datoteka promijenila. Docker contaneri su dizajnirani da kada oni izvrše naredbu ili slijed naredbi koji im je zadan, automatski će se izgasiti. Na linijama 8 i 9 je specificirano kojoj mreži pripada kontejner `developer\_junior`, analogno je napravljeno za ostale kontejnere. Na liniji 10 nalazi se direktiva koja će promijeniti `/etc/hosts` datoteku u kojoj će asocirati domenu `cyberpoligon.local` s IP adresom `172.19.1.20`. Za svaki pojedini kontejner mogu se specificirati iz koje Docker slike će se izgraditi. Za sve Docker kontejnere osim Mailserver, postavke su napisane u svojim zasebnim `Dockerfileovima` koji se nalaze u `/docker/` mapi. Na ispisu 5.2. je prikazana kako izgleda konfiguracija za Mailserver unutar `docker-compose.yml`.

```
mailserver:
    image: docker.io/mailserver/docker-mailserver:latest
    container_name: mailserver
    hostname: mail
    domainname: cyberpoligon.local
    env_file: mailserver.env
    # https://docker-mailserver.github.io/docker-mailserver/edge/
    config/security/understanding-the-ports/
    ports:
```

```
9       - "25:25" # SMTP
10      - "143:143" # IMAP4
11      - "465:465" # ESMTP
12      - "587:587" # ESMTP
13      - "993:993" # IMAP4
14    volumes:
15      - ./docker-data/dms/mail-data/:/var/mail/
16      - ./docker-data/dms/mail-state/:/var/mail-state/
17      - ./docker-data/dms/mail-logs/:/var/log/mail/
18      - ./docker-data/dms/config/:/tmp/docker-mailserver/
19      - /etc/localtime:/etc/localtime:ro
20    restart: always
21    stop_grace_period: 1m
22    networks:
23      company_machines:
24        ipv4_address: 172.19.1.20
```

Ispis 5.2: Konfiguracija za Mailserver unutar Docker Composea

Za njega moramo definirati pod `hostname` i `domainname` koji domenu će koristi Mailserver. Na linijama od 8 do 13 su definirani koja pristupna vrata su dostupna ostalim kontejnerima na mreži. Na liniji 24 je definirana statička IP adresa za Mailserver kako bi zapisi pohranjeni u `/etc/hosts` datoteci na ostalim računalima bili ispravni. U ispisu 5.3, pod `networks` su definirane mrežne postavke.

```
1 networks:
2   company_machines:
3     driver: bridge
4     ipam:
5       config:
6         - subnet: 172.19.1.0/24
```

Ispis 5.3: Mrežne postavke unutar Docker Composea

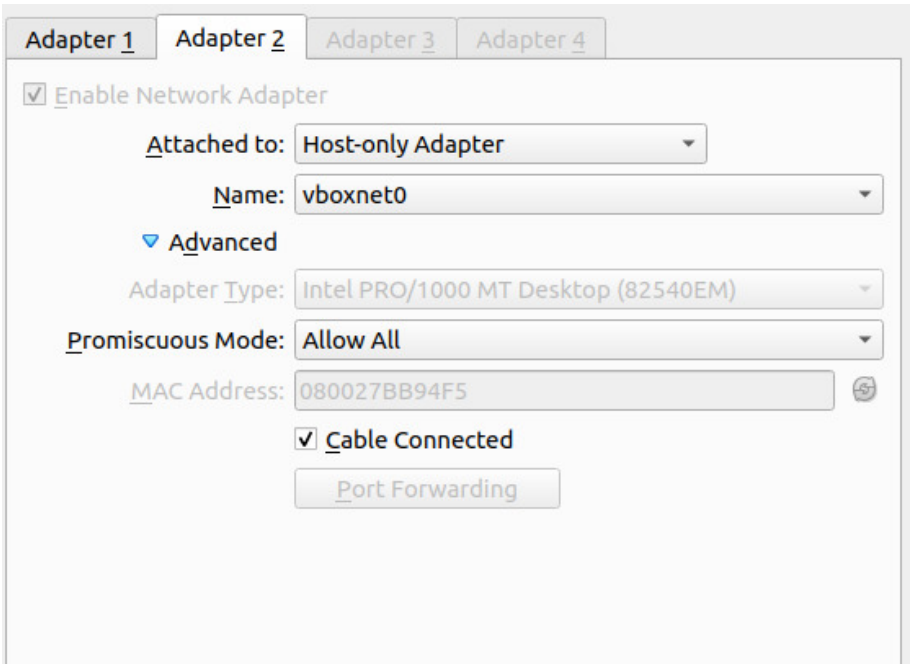
Dodijeljen je raspon `172.19.1.0/24`, na kojem stane 30 računala, što je i više nego dovoljno za potrebe ove kibernetičke mreže. Također, na liniji 3 dodijeljen je `bridge` tip mreže, koji je i podrazumijevani.`bridge` direktiva omogućava Docker kontejnerima da međusobno komuniciraju. Konačno, pod `volumes` su definirani dijeljeni direktoriji koji su prisutni na slici 4.2 i to je prikazano u ispisu 5.4.

```
1 volumes:
2   source_code_critical:
3   source_code_ordinary:
4   bussiness_client_data:
```

Ispis 5.4: Postavke za dijeljeni direktorij unutar Docker Composea

5.3. Postavljanje virtualne mreže uz pomoć VirtualBoxa

Mreža *Docker kontejnera* je odvojena od sustava s kojeg želimo pristupiti toj mreži. Taj sustav je Kali Linux virtualni stroj. Da bi virtualni stroj i Docker mreža mogli komunicirati, Docker mreža se mora postaviti na mostni (eng. *bridge*) tip, koji je ujedno i podrazumijevani tip mreže. Mostni tip mreže kreira sučelje `docker0` na domaćinskom računalu preko kojeg se može pristupiti pojedinim Docker kontejnerima. Za virtualni stroj se moraju postaviti dva sučelja unutar VirtualBoxa. Jedno sučelje se treba postaviti na *Network Address Translation* ili skraćeno NAT tip rada, a za drugo sučelje se koristi *Host-only* tip mreže i to sučelje je imenovano `vboxnet0` unutar domaćinskog operativnog sustava. NAT sučelje dodjeljuje sučelje na domaćinskom operativom sustavu preko kojeg možemo pristupiti unutar virtualnog stroja. Ovo omogućuje uspostavljane veze između Docker kontejnera i TCP slušača koji će biti podalje objašnjeno u poglavlju 7.4. *Host-only* tip mreže služi tome da unutar virtualnog stroja možemo pristupiti IP adresama koje se nalaze na domaćinskom operativnom sustavu. Ovo sučelje se koristi za slanje poruke elektroničke pošte koje je dodatno objašnjeno u poglavlju 7.3. Na slici 5.3 je prikazano kako izgledaju mrežne postavke za *Host-only sučelje* unutar VirtualBoxa.



Slika 5.3: VirtualBox konfiguracija Host-only mrežnog sučelja

6. Modularnost rješenja

Zbog modularne prirode Dockera, postavke koje su su definirane u konfiguracijskoj datoteci `docker-compose.yaml` se mogu lako promijeniti. `docker-compose.yaml` bi se mogao konstiruirati na način da sadrži bazne konfiguracije za postavljanje mreže i da se na to mogu nadograđivati kontejneri koji specificiraju postavke za pojedini zadatak. Ovakav sistem bi se mogao adaptirati po potrebi i služiti kao kibernetički poligon koji služi za testiranje koliko je neka mreža sigurna ili za treniranje kibernetičkih napada ili za potrebe treniranja koje zahtijevanju veću infrastrukturu od jednog stroja.

6.1. Mijenjanje verzija softverskih paketa

Docker ima implementiranu naredbu `docker compose exec` koja omogućava izvršavanje naredbe, kao što bi se ta naredba izvršila na terminalu tog kontejnera. Također Na ovaj način, može se primjerice, ažurirati softverski paket koji u sebi ima ugrađenu ranjivost. Naredbe se mogu staviti u specijalnu skriptu koja bi mijenjala zadatak po potrebi. Nadalje je prikazana skripta koji bi ažurirao softver *Libre office* na najnoviju dostupnu verziju na računalu od Junior programera i koja bi uklanjala instalirani internetski preglednik *Firefox*:

```
1 docker compose exec developer_junior apt-get update
2 docker compose exec developer_junior apt-get install -y libreoffice
3 docker compose exec developer_junior apt remove --purge firefox
```

Ispis 6.1: Primjer skripte uz pomoć koje se modificiraju postavke na Docker kontejneru

Također, postoji opcija kompletnog micanaja nekog softverskog paketa pomoću naredbe `apt-get remove --purge -y [ime paketa]`. Na ovaj način se mogu mijenjati postavke kontejnera dok je on upaljen. Nadalje, kada su izgašeni kontejneri, mogu se promjeniti `install.sh` skripte koje se nalaze u `/docker/[ime kontejnera]/` direktoriju za odgovarajući kontejner.

7. Demonstracija napada nad postavljenim poligonom

U ovom poglavlju će se demonstrirati jedan moguć način iskorištavanje ranjivosti koja je prisutna na kibernetičkom poligonu. Također će se opisati koji je ranjivost prisutna na kibernetičkom poligonu i koje alate je napadač koristio kako bi iskoristio postavljenu ranjivost.

7.1. Planiranje napada i postavljanje scenarija

Da bi pristupili mreži neke organizacije trebamo gledati koji resursi su javno dostupni za tu organizaciju. To može biti neki web servis koja je javno dostupna na internetu, adrese elektroničke pošte zaposlenika te organizacije i slično. Za ovaj testni slučaj je zamišljeno da se adresa elektroničke pošte senior programera našla na internetu npr. na *Linkedinu* i da napadač pošalje poruka s priloženim Libre office dokumentom koji prilikom pokretanja uspostavlja udaljenu sjednicu (eng. remote access) s računalom napadača. Također ako je poslužitelj elektroničke pošte koji se nalazi na mreži neispravno postavljen, napadač se može predstavi kao neki drugi zaposlenik te organizacije tj. lažirati pošiljateljsku adresu (eng. mail spoofing).

7.2. Kreiranje maliciozne pošiljke

Za kreiranje maliciozne pošiljke je korišten *Metasploit* modul `exploit/multi/fileformat/libreoffice_macro_exec` koji će generirati maliciozne `.odt` datoteku koja iskorištava ranjivost u *Libreoffice* verzijama 6.1.0-6.1.4.1 na napad kretanja direktorijima (eng. *directory traversal attack*) [7][12]. Prije kreiranje maliciozne pošiljke trebaju se promijeniti podatci na koji IP će se spojiti zaraženo računalo kada se otvorti maliciozna datoteka. Za prikazivanje koji se parametri

mogu promijeniti za neki Metasploit `exploit`, pokreće se naredba `show options`. Njezin ispis se može vidjeti na ispisu 7.1. `LHOST` opcija treba odgovarati IP adresi napadačkog računala ili sučelju na kojem će napadač slušati, a `LPORT` označava na kojim pristupnim vratima napadač sluša nadolazeće konekcije. Također, pod `target` treba specificirati koji operacijski sustav koristi žrtva, što je za ovaj primjer Linux. Pokretanjem naredbe `exploit` se kreira maliciozni Libre dokument.

```
msf6 exploit(multi/fileformat/libreoffice_macro_exec) > show options

Module options (exploit/multi/fileformat/libreoffice_macro_exec):
   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   FILENAME  librefile.odt    yes       Output file name
   SRVHOST   0.0.0.0          yes       The local host or network
    interface to listen on. This must be an address on the local
                                        machine or 0.0.0.0 to listen
     on all addresses.
   SRVPORT   8080             yes       The local port to listen on.
   SSL       false            no        Negotiate SSL for incoming
    connections
   SSLCert                    no        Path to a custom SSL
    certificate (default is randomly generated)
   URIPATH                    no        The URI to use for this
    exploit (default is random)

Payload options (windows/meterpreter/reverse_tcp):
   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  process          yes       Exit technique (Accepted:
    '', seh, thread, process, none)
   LHOST     eth0             yes       The listen address (an
    interface may be specified)
   LPORT     4444             yes       The listen port

   *DisablePayloadHandler: True   (no handler will be created!)*

Exploit target:
   Id  Name
   --  ----
   1   Linux
```

Ispis 7.1: Dostupne postavke za Libreoffice iskorištavanje ranjivosti

7.3. Korištenje SET za slanje maliciozne pošiljke

Nakon što je kreirana maliciozni dokument uz pomoć *Metasploita*, može se iskoristiti Social Engineering Toolkit za prijenos te datoteke na računalo žrtve. Za pokratenja SET-a je iskorišten `python 2.7.18`. SET podržava opciju masovnog slanja elektroničke pošte (eng. *Mass mailer attacka*), ali se može poslati poruka na samo jedan račun elektroničke pošte, što će i biti korišteno za ovaj primjer. Za slanje poruke trebaju se upisati koja je odredišna adresa, pošiljateljska adresa, koja je domena poslužitelja elektroničke pošte kojeg koristimo, što je u ovom slučaju `cyberpoligon.local`. Još se treba upisati apsolutna putanja do malicioznog dokumenta i tijelo poruke. Kada se šalje poruka koristeći SET, poruka nema sva zaglavlja, ali je na poslužitelju isključeno provjeravanje njih kao što je opisano u potpoglavlju 4.5.

7.4. Postavljanje TCP slušača preko Metasploit konzole

Nakon što je žrtvi dostavljen maliciozni dokument, žrtva ju samo treba pokrenuti. Napadač treba imati upaljen `handler` koji će slušati hoće li se netko spojiti na njega. Za ovu svrhu koristiti će se `multi/handler` s konfiguriranim istim payloadom s kojim je kreiran maliciozni dokument. TCP slušatelj pruža podršku za pristupnu točku TCP poslužitelja na određenim pristupnim vratima. Pristupna točka će prihvaćati veze i primati poruke od TCP klijentske aplikacije, koja je u ovom slučaju *Metasploit*. Prilikom odabira payload, dostupna je opcija `reverse_tcp` i `bind_tcp` TCP slušača. `reverse_tcp` se koristi kada se žrtva pokušava spojiti na računalo napadača, a `bind_tcp` kada se napadač pokušava spojiti na računalo žrtve. Za ovaj primjer se koristi `linux/x86/reverse_tcp` jer se konekcija treba uspostaviti tek kada žrtva otvori malicioznu datoteku.

8. Zaključak

Treniranje na pojedinačnim ranjivim sustavima je lako pronaći, ali treniranje nad simulacijom kibernetičke mreže nije još rasprostranjeno jer je simulirati složenu mrežu u praksi nezgrapno i kompliciranо. Kibernetički poligoni pružaju stvarnu platforme za obuku stručnjaka iz područja informacijske i komunikacijske tehnologije protiv širokog spektra kibernetičkih napada. Alat Docker uparen s *Docker Composeom* pojednostavljuje postavljanje kibernetičkih poligona za vježbe gdje se je potrebno rekreirati IT infrastrukturu neke organizacije. Docker je korišten zato što pruža modularnost što znači da je scenarij zadatka moguće promijeniti uvođenjem novih Docker kontejnera na mrežu. Docker također ima ugrađenu funkcionalnost, kao što je postavljanje dijeljenih direktorija na mrežu koji bi u sebi sadržavali osjetljive informacije, što omogućava provođenja vježbi sa više timova u stvarnom vremenu. Napadački tim bi pokušao saznati osjetljive informacije, a obrambeni bi trebao spriječiti napadački tim. Za buduće iteracije ovog projekta, može se postaviti *Ansible* koji bi služio za pokretanje posebnih skripti koje bi mijenjale zadatak kako je opisano u poglavlju 6.

LITERATURA

[1] Ivona Brajdić, Ivan Kovačević i Stjepan Groš. *Review of National and International Cybersecurity Exercises Conducted in 2019*. 2019.

[2] Exploit DB. *Linux Kernel 4.4 Local Privilege Escalation*. https://www.exploit-db.com/exploits/40759. 2020.

[3] Docker. *Docker Compose offical documentation*. https://docs.docker.com/compose/.

[4] Docker. *Docker overview*. https://docs.docker.com/get-started/overview/.

[5] Dovecot. *How to setup Dovecot*. https://doc.dovecot.org/configuration_manual/howto.

[6] IEEE. *Cyber Kill Chain Phases*. https://www.computer.org/publications/tech-news/trends/what-is-the-cyber-kill-chain-and-how-it-can-protect-against-attacks/.

[7] Alex Inführ i Shelby Pace. *LibreOffice Macro Code Execution - Metasploit*. https://www.infosecmatter.com/metasploit-module-library/?mm=exploit/multi/fileformat/libreoffice_macro_exec. 2020.

[8] David Kennedy. *Social engineering toolkit*. https://github.com/trustedsec/social-engineer-toolkit. 2020.

[9] Casper Klein i dr. *Docker Mailserver*. https://github.com/docker-mailserver/docker-mailserver. 2022.

[10] Lockheed Martin. *Cyber Kill Chain*. https://www.lockheedmartin.com/en-us/capabilities/cyber/cyber-kill-chain.html/.

[11] Microsoft. *Protect yourself from phishing*. https://support.microsoft.com/en-us/windows/protect-yourself-from-phishing-0c7ea947-ba98-3bd9-7184-430e1f860a44/.

[12] NIST. *CVE-2018-16858*. https://nvd.nist.gov/vuln/detail/CVE-2018-16858. 2018.

[13] NIST. *CVSS caluculator*. https://nvd.nist.gov/vuln-metrics/cvss/v3-calculator.

[14] Postfix. *Postfix offical documentation*. https://www.postfix.org/.

[15] Rapid7. *Metasploit offical documentation*. https://docs.rapid7.com/metasploit/.

[16] Ukwandu i Elochukwu. *A Review of Cyber-Ranges and Test-Beds: Current and Future Trends*. https://www.mdpi.com/1424-8220/20/24/7148/. 2020.

Automatizirano postavljanje kibernetičkog poligona za potrebe kibernetičkih vježbi

Sažetak

Tema ovog završnog rada je kreiranje kibernetičkog pogliona koji modelira mrežu neku organizacije u svrhu provođenja kibernetičkih vježbi. Poligon je sačinjena od Docker kontejnera i povezana je uz pomoć Docker composea. Kontejneri dolaze s ugrađenim ranjivostima koje napadač može probati iskoristiti. Na mreži je prisutan poslužitelj elektroničke pošte i dijeljeni direktoriji koji mogu sadržavati povjerljive informacije. Korištenje Metasploita za kreiranje zlonamjerne elektroničke pošte i izvršavanje arbitrarnog koda na udaljenoj sjednici. Nadač šalje phishing poruku elektroničke pošte i pokušava uspostaviti udaljenu sjednicu na računalu žrtve.

Ključne riječi: Kibernetičke mreže, kibernetičke vježbe, zlonamjerna elektroničke pošte, Docker, Docker compose, ranjivosti, kibernetičko treniranje

Automation of the process of building cyber networks for the purpose of training

Abstract

The theme of this final thesis is the creation of a cyber polygon that models the network of an organization for the purpose of conducting cyber exercises. The polygon is made up of Docker containers and is connected using Docker compose. Containers come with built-in vulnerabilities that an attacker can try to exploit. There is an e-mail server and shared directories on the network that may contain confidential information. Using Metasploit for creating a malicious payload and executing arbitrary code on a victim's computer. The attacker sends a phishing email and tries to establish a remote session on the victim's computer.

Keywords: cyber networks, cyber exercises, phishing mail, Docker, Docker compose, vulnerabilities, cyber training