SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 2659

Emulacija automatiziranih napadača u kibernetičkoj sigurnosti

Hrvoje Fabris

Zagreb, lipanj 2021.

SADRŽAJ

1. Uvod

S razvojem tehnologije razvija se Internetska infrastruktura, sustavi postaju sve veći i složeniji i postaje sve teže otkloniti ranjivosti u sustavu i obraniti se od napada [1]. Da bi se pripremile za suočavanje s novim prijetnjama i napadima koji se mogu dogoditi i uskladile s normama i regulatornim zahtjevima kao što su NIS [13] i PCI DSS [14], organizacije često dogovaraju penetracijska testiranja ili testiranja crvenim timovima [7]. Takva testiranja iziskuju velike resurse i angažman stručnjaka zbog čega organizacije ne provode takva testiranja često [4].

Emulacija automatiziranih napada je jedan od načina na koji se može testirati rad obrambenih timova i pripremiti ih za napad na organizaciju [6]. Za izvršavanje emulacije potrebno je automatizirati napadača koji će tijekom emulacije napasti sustav koristeći tehnike i taktike koje ima na raspolaganju. Napadač treba moći koristiti različite tehnike tijekom napada, a za realističnu emulaciju napadača odabir tehnika treba ovisiti o napadačevom znanju i informacijama koje je prikupio o sustavu kojeg napada.

U drugom poglavlju je detaljnije objašnjeno kako izgraditi emulaciju i napadačkog agenta i spomenuti su neki od postojećih programa za emulaciju napada. Da bi se testiralo napadačkog agenta i emulirao kibernetički napad potrebno je izgraditi model sustava. U trećem poglavlju je opisan model kibernetičkog prostora koji je korišten za izradu emulacije. U četvrtom poglavlju se stavlja naglasak na napadača, kako je definiran napadač i kako je to implementirano. U petom poglavlju je objašnjeno kako se testiralo napadačkog agenta i koji su rezultati testiranja i opisana je komponenta za vizualizaciju modela. U šestom poglavlju su predložene promjene kako bi se ovaj sustav mogao izmijeniti da bude što sličniji stvarnom sustavu i napadima.

2. Automatizacija kibernetičkih napada

Problemi s kojima se organizacije suočavaju prilikom pripreme za kibernetički napad i planiranje obrambenih radnji su brojni [3]. Jedan od problema je nepredvidljivost i teško otkrivanje napada. Napadači koriste ranjivosti nultog dana (engl. *zero-day vulnerability*) za iskorištavanje radnjivosti unutar sustava kojeg napadaju [3]. Ranjivost nultog dana je ranjivost koja nije poznata i za koju ne postoji sigurnosno ažuriranje koje bi spriječilo iskorištavanje te ranjivosti [15]. Kibernetičke napade je teško predvidjeti i zbog velikog broja vektora napada koje napadač može iskoristiti da dobije pristup sustavu. Vektor napada je put ili sredstvo kojim napadač dobiva pristup sustavu kojeg napada [16]. Neki od primjera vektora napada su ubacivanje SQL izraza (engl. *SQL Injection*), loše sigurnosne postavke (engl. *Security Misconfiguration*), korištenje programa s poznatim ranjivostima (engl. *Using Apps With Known Vulnerabilities*) i slanje elektroničke pošte s malicioznim privitkom (engl. *Phishing*).

Još jedan od problema s kojim se organizacije mogu susresti je nemogućnost otkrivanja napadača nakon što on dobije pristup sustavu. U stvarnom svijetu napadači mogu imati pristup sustavu mjesecima prije nego krenu u izvršavanje napada, a tijekom tog vremena uspješno prikupljaju informacije o sustavu bez da organizacije znaju za njihovu prisutnost.

Sljedeći problem je složenost sustava organizacije. Računalni sustavi postaju sve složeniji sa sve više sklopovskih i programskih alata i aplikacija. Napadačima je dovoljno da pronađu samo jednu grešku u sustavu da dobiju pristup, a organizacije moraju redovito ažurirati svoj sustav da bi pokrpale sve ranjivosti.

Ovisnost o dobavnom lancu je još jedan problem o kojem bi organizacije trebale voditi računa. Organizacije mogu ovisiti o drugim proizvođačima opreme ili nekim drugim organizacijama koje za njih obavljaju određene usluge što može predstavljati još jedan način na koji napadač može pokrenuti napad na organizaciju. Primjer jednog takvog napada je napad zloćudnog programa *NotPetya* iz 2017 [17]. Zločudni program koristi

dvije različite metode iskorištavanja (engl. *exploits*), alat za penetracijska testiranja *EternalBlue* i alat *Mimikatz*. Jednom kada dobije pristup sustavu zločudni program kriptira *Master Boot Record* unutar kojeg se nalazi informacija o lokaciji operacijskog sustava koji je instaliran na računalu i iz radne memorije računala izvlači lozinke zaposlenika koje koristi za prelazak na druga računala unutar mreže. Napad je pokrenut koristeći poslužitelje Ukrajinske organizacije *Linkos Group* koji se koriste za ažuriranje programa *M.E.Doc*. Nakon što je napad pokrenut sva računala koja su imala instaliran program *M.E.Doc* su bila zaražena. Napadom na organizaciju *Linkos Group* napadači su izvršili napad i na druge organizacije kao što su *Maersk*, *Merck*, *FedEx*, *Saint-Gobain*, *Mondelēz* i *Reckitt Benckiser* jer je unutar sustava tih organizacija bio instaliran program *M.E.Doc*. Šteta nastala tijekom napada zloćudnog programa *NotPetya* se procjenjuje na 10 milijardi dolara [17].

Organizacije u suradnji sa stručnjacima provode sigurnosna testiranja svog sustava. Prilikom takvih testiranja stručnjaci imitiraju napadača i pokušavaju pronaći i iskoristiti ranjivost unutar sustava [4]. Testiranja se mogu provoditi na razne načine, a neki od njih su penetracijska testiranja i testiranja crvenim timovima [7]. Problem kod penetracijskih testiranja je da se baziraju na ranjivosti sustava i ne pripremaju organizacije na otkrivanje napadača koji se nalazi unutar sustava [2]. Drugi način na koji se organizacije mogu pripremiti na suočavanje s novim prijetnjama je testiranje sustava korištenjem crvenih timova. Testiranje crvenim timom se radi tako da grupa sigurnosnih stručnjaka nastoji kompromitirati sustav organizacije koristeći kompleksne napade [4]. Za razliku od penetracijskih testiranja, testiranje crvenim timom stavlja naglasak na događaje i radnje koje se dogode nakon što napadač dobije pristup sustavu [6; 5]. Tako se mogu testirati radnje koje organizacija provodi da bi se obranila i otkrila napade na svoju infrastrukturu. Nakon testiranja crvenim timom, uspoređuju se radnje koje je obrambeni tim izvršio da otkrije djelovanje crvenog tima s radnjama koje je crveni tim poduzeo te se tako može unaprijediti obrana organizacije.

Iako je testiranje crvenim timom za organizacije bolje jer ukazuje na potencijalne metode napada koje napadač može koristiti prilikom napada, organizacije ipak ne koriste često ovakva testiranja u praksi zbog niza nedostataka [4]. Testiranje crvenim timovima iziskuje velike resurse i angažman stručnjaka, a problem za provođenje ovakvih testova može biti i način na koji je sustav implementiran koji ne dozvoljava provođenje ovakvih testova. Primjer sustava za kojeg nije moguće provesti testiranje na stvarnom sustavu je SCADA (engl. *Supervisory Control And Data Acquisition*) sustav. SCADA sustav se koristi za nadzor, mjerenje i upravljanje industrijskih sustava. Takav sustav je osjetljiv na prekide i svaka pogreška koja bi se mogla dogoditi tijekom

testiranja takvih sustava može utecati na sam rad sustava [10].

Neke od problema kod testiranja crvenim timom se može zaobići automatizacijom crvenih timova i emulacijom stvarnih prijetnji. Automatizacijom crvenih timova može se napraviti i emulirati prijetnje na kojima će organizacije moći uvježbavati svoje obrambene timove češće i za manju cijenu [5]. Iako takve aplikacije ne mogu u potpunosti emulirati napade kakvi se pojavljuju u stvarnom svijetu, one mogu pomoći obrambenim timovima uvježbavanje radnji koje se poduzimaju u slučaju stvarnih kibernetičkih napada.

Neki od uvjeta koje je potrebno ostvariti da bi emulacija automatiziranih kibernetičkih napada bila uspješna su inteligentnost, jednostavnost, realističnost i modularnost [7]. Emulacija treba biti inteligentna odnosno tijekom emulacije sustav treba odabirati i povezivati radnje onako kako bi ih napadač odabirao tijekom stvarnog napada. Jednostavnost emulacije znači da je obrambeni tim može koristiti bez znanja konfiguracijskih detalja emulacije. Obrambenom timu nije potrebno znati detalje kako je emulacija napravljena, bitnije je da se emuliraju stvarne prijetnje da bi se mogli pripremiti za neki kibernetički napad. Emulacija treba biti realistična, odnosno izvršavati tehnike koje bi napadači izvršavali tijekom njihovih napada. Modularnost emulacije znači da prilikom pokretanja emulacije korisnici biraju tehnike i radnje koje se mogu izvršiti tijekom emulacije, i da se može dodavati nove tehnike i radnje unutar emulacije.

Emulacija napada se sastoji od informacijskog sustava i sudionika emulacije [10]. Informacijski sustav se sastoji od mrežnih čvorova, usmjernika, poslužitelja, bazi podataka i ostalih komponenata koji se mogu nalaziti unutar mreže organizacije. Sudionici emulacije uključuju napadača, korisnike i osobe zadužene za sigurnost sustava. Napadač predstavlja individualnog napadača, grupu ili organizaciju koju netko financira. Cilj napadača je dobiti pristup sustavu i koristeći tehnike na raspolaganju kompromitirati sustav. Korisnici predstavljaju zaposlenike organizacije koji obično imaju nisku razinu ovlasti. Oni su sudionici emulacije jer preko njih napadač dobiva pristup sustavu. Osoba zadužena za sigurnost sustava nadzire rad informacijskog sustava i pokušava ga zaštiti od napada. Za razliku od korisnika, ima veću razinu ovlasti i njihov rad je presudan u zaštiti organizacije od vanjskih i unutarnjih prijetnji. Još jedan sudionik koji se može pojaviti unutar emulacije je zaposlenik koji izvršava napad na organizaciju u kojoj je zaposlen. On ima karakteristike korisnika jer je zaposlen unutar organizacije i ima pristup informacijskom sustavu i ima karakteristike napadača jer pristupa sustavu s malicioznim namjerama. Tijekom emulacije takvog zaposlenika je moguće emulirati kao običnog zaposlenika koji u određenom trenutku emulacije kreće s malicioznim djelovanjem.

Automatizirani alati za iskorištavanje ranjivosti unutar sustava već postoje, primjer jednog takvog alata je Metasploit [11]. Takvi alati koriste poznate metode iskorištavanja s kojima pokušavaju iskoristiti ranjivosti unutar sustava. Iako se takvi alati mogu koristiti tijekom emulacije ti alati ne mogu realistično emulirati ponašanje napadača jer oni koriste samo jednu metodu iskorištavanja sustava kojeg napadaju [4].

Automatizirani napadač treba pratiti svoje stanje unutar sustava kojeg napada. Napadačevo stanje predstavlja cjelokupno znanje koje je on prikupio o sustavu prije napada i znanje koje je on naučio o sustavu tijekom napada. Uzimajući u obzir to znanje napadač treba planirati i izvoditi radnje kako bi ostvario svoj cilj [6; 1]. Planiranje radnji za napadača nije jednostavno za napraviti jer napadač nema cjelokupno znanje o sustavu kojeg napada. Zbog toga je prikupljanje informacija o sustavu važan dio napada [12].

2.1. Postojeći sustavi za emulaciju

Neki od već postojećih sustava za emulaciju prijetnji su CALDERA [21], Infection Monkey [23], Atomic Red Team [22] i APT Simulator [24].

CALDERA je okvir za računalnu sigurnost (engl. *cybersecurity framework*) kojeg je razvila organizacija MITRE. Okvir je razvijen da bi se sigurnosnim stručnjacima smanjilo potrebno vrijeme i resursi potrebni za provođenje rutinskih sigurnosnih testiranja. CALDERA omogućuje automatizaciju emulirane prijetnje, automatizaciju odgovora na kibernetički napad i pomoć u procjeni crvenog tima.
Automatizacija emulitane prijetnje podrazumijeva automatizaciju prijetnje koju je razvio crveni tim i provjera je li sustav ranjiv na tu prijetnju.
Automatizacija odgovora na kibernetički napad testira automatizirane radnje obrambenog tima na određenom poslužitelju.
CALDERA također može pomoći crvenom timu tijekom procjene sustava mijenjajući postojeći skup alata koji crveni tim ima na raspolaganju.

Infection Monkey je alat otvorenog koda koji simulira proboj i napad na podatkovni centar. Alat se sastoji od dva dijela, alata koji provodi napad i kompromitira sustav i udaljenog servera koji kontrolira alat koji provodi napad i vizualizira njegovo napredovanje unutar podatkovnog centra.

Atomic Red Team je knjižnica koja se sastoji od testova koji se mogu koristiti za testiranje sustava. Svaki test predstavlja različitu tehniku koju napadači koriste tijekom napada, a tehnike su preslikane iz MITRE ATT&CK okvira koji je objašnjen u potpoglavlju 4.2.1. Za testove je važno da se može testirati svaki dio sustava, da

izvođenje testa ne zahtijeva više od pet minuta i da se stalno prate nove tehnike koje napadači koriste u svojim napadima i da se na temelju novih tehnika može napraviti novi test.

APT Simulator je *Windows Batch* skripta koja koristi alate da bi prevarila sustav i natjerala ga da pomisli da je bio žrtva kibernetičkog napada. Za razliku od drugih alata za simulaciju napada, APT Simulator je napravljen jednostavnim pristupom, bez korištenja baze podataka, udaljenog servera ili virtualnih računala. Kod ovog programa naglasak nije na zločudnom programu koji se koristi tijekom napada nego na simulaciji aktivnosti napadača tijekom napada.

3. Modeliranje kibernetičkog prostora

Model kibernetičkog prostora unutar kojeg se emulira automatiziranog napadača predaje se programu prilikom pokretanja preko ulaznih datoteka. Kibernetički prostor se sastoji od modela sustava i agenata koji izvršavaju radnje unutar tog sustava.

3.1. Model sustava

Model IT sustava korišten za potrebe ovog rada sastoji se od pojednostavljenog prikaza kibernetičkog prostora banke. Kibernetički prostor banke sastoji se od međusobno povezanih komponenata koje tvore računalnu mrežu banke i zaposlenika banke.

3.1.1. Komponente sustava

Računalna mreža banke sastoji se od međusobno povezanih komponenata koje predstavljaju mrežne uređaje, radne stanice zaposlenika banke i mrežne poslužitelje. Svaka komponentu određuje jedinstvena oznaka, naziv komponente i lista drugih komponenata koje su povezane s tom komponentom. Radne stanice i mrežni poslužitelji imaju dodatna polja s informacijama o IP adresi komponente, instaliranom softveru, administratorima, broju radnika na toj komponenti i njihovom nazivu, razini privilegija, naziv domene kojoj komponenta pripada, popis komponenata kojima zaposlenici te komponente mogu udaljeno pristupiti i informacijama o kritičnim podacima koji se nalaze na toj komponenti.

Radne stanice i mrežni poslužitelji unutar sustava imaju neka stalna stvojstva i neka svojstva koja se mogu mijenjati tijekom izvođenja simulacije. Stalna svojstva uključuju listu svih komponenata s kojima je neka radna stanica ili mrežni poslužitelj povezan, listu administratorskih računa koji mogu udaljeno pristupiti toj komponenti, listu udaljenih komponenata na koje zaposlenici koji radu na toj komponenti mogu pristupiti, domena unutar koje se radna stanica ili mrežni poslužitelj nalazi, kritični podaci koji se nalaze na toj komponenti. Svojstva koja se mogu mijenjati uključuju

listu aktivnih spojeva s drugim radnim stanicama i listu aktivnih korisnika koji su pristupili toj radnoj stanici ili mrežnom poslužitelju. Lista aktivnih korisnika uključuje sve korisnike čija su korisnička imena i lozinke zapisane u memoriji te komponente, a lista aktivnih spojeva uključuje sve aktivne spojeve koji su otvoreni od te komponente i prema toj komponenti. Aktivni spojevi simulira mrežni promet i komunikaciju između dva zaposlenika unutar sustava i ta komunikacija je pojednostavljena tako da se mrežni promet ne simulira nego se samo zabilježi da spoj postoji.

Kao ulazni parametar programu se predaje naziv datoteke unutar koje se nalaze opisi svih komponenata sustava u JSON formatu. Komponente unutar datoteke se dijele na korisničke komponente pod koje spadaju radne stanice i mrežni poslužitelji i mrežne komponente pod koje spadaju mrežni uređaji. Ispis 3.1 prikazuje opis jedne radne stanice koja ima jedinstvenu oznaku *component*, a naziv radne stanice je *accountant_workstation*. Radna stanica ima IP adresu *192.168.53.11* i povezana je s komponentom *access_switch_accounting*. Na toj radnoj stanici je instaliran *Operacijski sustav Windows 10 v1703*, a od programa *Microsoft office 2016* i *Microsoft Edge v40*. Administratori koji mogu udaljeno pristupiti toj radnoj stanici su *System administrator* i *Security operator*. Unutar modela IT sustava može biti najviše *10* radnika koji rade na toj radnoj stanici, a naziv radnika je *Accountant*. Zaposlenici imaju razinu privilegija *1* koja odgovara zaposlenicima s niskom razinom privilegija. Radna stanica pripada *Central* domeni. Zaposlenici koji rade na toj komponenti mogu udaljeno pristupiti komponenti *mail_server*. Na komponenti u ispisu nisu zapisani nikakvi kritični podaci.

```
"component" : {
      "name" : "accountant_workstation",
      "connected_components" : [ "access_switch_accounting" ],
      "ip_address" : "192.168.53.11",
      "software" : [
          "Operacijski sustav Windows 10 v1703",
          "Microsoft office 2016",
          "Microsoft Edge v40"
          ],
      "administrators" : [
          "System administrator",
          "Security operator"
        ],
      "max_account_number" : "10",
      "worker_name" : "Accountant",
      "priviledge_level" : "1",
      "domain" : [ "Central" ],
      "remote" : ["mail_server"],
      "sensitive" : []
    }
```

Ispis 3.1: Primjer opisa radne stanice

3.1.2. Zaposlenici sustava

Svaka radna stanica unutar mreže ima korisnički račun zaposlenika koji radi na toj radnoj stanici. Osim toj radnoj stanici, zaposlenik može udaljeno pristupiti drugim radnim stanicama ili poslužiteljima koji se nalaze u opisu radne stanice na kojoj radi taj zaposlenik. Zaposlenici unutar sustava su modelirani razredom `employee` i sve informacije vezane za zaposlenika kao što su naziv zaposlenika, radna stanica na kojoj taj zaposlenik radi, razina privilegija, domena i udaljeni pristup se spremaju kao lokalne varijable tog zaposlenika i one se ne mijenjaju. Tijekom izvođenja zaposlenik može primati elektroničku poštu, pristupati i odspojiti se s pojedinih komponenata sustava na koje ima pristup i otvarati i zatvarati spojeve s drugim zaposlenicima s kojima mu je dopušteno komunicirati. Te informacije se također spremaju kao lokalne varijable u memoriju zaposlenika i one se mogu mijenjati.

Razred `employee` ima metode za dohvat nepročitane elektroničke pošte, dodavanje nove nepročitane elektroničke pošte, dodavanje i micanje spojeva s drugim zaposlenicima i aktivnih pristupanja komponentama sustava.

3.2. Agenti

Agenti unutar sustava predstavljaju automatizirane sudionike emulacije koji izvode radnje napadača, obrambenog tima i zaposlenika. Svi agenti su modelirani razredom `agent` kojeg nasljeđuju specifični agenti. Agenti imaju svoj naziv, listu radnji koje mogu izvesti, strategiju prema kojoj se radnje izvode, znanje o sustavu i listu alata koje ima na raspolaganju. Te informacije se spremaju kao varijable unutar razreda `agent`, a razred još ima metode za odabir radnje koja će se izvršiti i izvršavanje radnje.
Za potrebe simulacije su definirane tri vrste agenata. Sivi agent koji emulira rad zaposlenika banke, obrambeni agent koji emulira obrambeni tim, i napadački agent koji emulira napadača.

3.2.1. Sivi agent

Sivi agent je agent koji emulira radnje zaposlenika banke koji imaju različite razine privilegija. Modeliran je razredom `gray_agent` koji nasljeđuje razred `agent`, i njegov naziv je `gray_agent`. Sivi agent nema nikakve dodatne varijable ili metode, a njegov skup radnji uključuje radnje koje mogu izvršiti zaposlenici sustava.
Radnje koje zaposlenici mogu izvršiti su:

reboot_host : Ova radnja emulira ponovno pokretanje komponente. Preduvjeti za ovu radnju ne postoje i ona se uvijek može izvesti. Nakon izvođenja ove radnje svi zaposlenici koji su bili prijavljeni na tu komponentu će biti odjavljeni i svi aktivni spojevi komponente s drugim komponentama će biti prekinuti.

user_login_to_host : Radnja koja predstavlja korisničko prijavljivanje na komponentu. Ova radnja nema preduvjeta i može se izvesti ako postoji komponenta na kojoj zaposlenik nije prijavljen a dozvoljeno mu je prijavljivanje na tu komponentu. Ako korisnik nije prijavljen na svoju komponentu onda će biti prijavljen na nju, a ako je već prijavljen na svoju komponentu onda će biti prijavljen na neku drugu komponentu kojoj smije udaljeno pristupiti.

open_connection_between_hosts : Radnja koja prikazuje pojednostavljeno otvaranje spoja između dvije radne stanice. Otvaranje spoja je pojednostavljeno jer se za radne stanice između kojih je otvoren spoj samo zabilježi da je spoj otvoren, a mrežni promet koji bi nastao otvaranjem spoja je zanemaren. Radnja modelira komunikaciju između dva zaposlenika unutar sustava. Za izvršiti ovu radnju

zaposlenik treba biti prijavljen na svoju komponentu, zaposleniku treba biti dopušteno uspostavljanje spoja do drugog zaposlenika i spoj između ta dva zaposlenika ne smije postojati. Ako su preduvjeti ispunjeni i radnja se izvrši stvara se spoj između dva zaposlenika i njihovim komponentama se u memoriju dodaje znanje o postojećem spoju.

close_connection_between_hosts : Radnja koja zatvara spoj između dva zaposlenika. Preduvjet za izvršavanje ove radnje je da zaposlenik koji izvršava radnju mora biti prijavljen na svojoj komponenti i spoj treba postojati. Ako su preduvjeti ispunjeni i radnja se izvrši briše se spoj između zaposlenika tog spoja i iz njihovih komponenti se briše znanje o postojanju tog spoja.

open_email : Radnja koja predstavlja otvaranje elektroničke poruke zaposlenika. Preduvjeti radnje su da je zaposlenik prijavljen na svojoj komponenti, da može udaljeno pristupiti serveru elektroničke pošte i da ima barem jednu nepročitanu elektroničku poruku. Ako su preduvjeti ispunjeni i radnja se izvrši zaposlenik ima jednu nepročitanu elektroničku poruku manje, a ako je zaposlenik otvorio elektroničku poštu koju je poslao napadač onda napadač dobiva pristup komponenti tog zaposlenika.

send_email : Radnja koja predstavlja slanje elektroničke pošte. Za izvršiti ovu radnju zaposlenik treba biti prijavljen na svojoj komponenti i zaposlenik treba moći udaljeno pristupiti serveru elektroničke pošte. Ako su preduvjeti ispunjeni zaposlenik šalje elektroničku poštu nekom drugom zaposleniku kojem se onda povećava broj nepročitanih elektroničkih poruka.

browser_internet : Ova radnja predstavlja surfanje internetom. Za izvršiti ovu radnju zaposlenik treba biti prijavljen na svojoj komponenti. Tijekom surfanja internetom zaposlenik može otvoriti zaraženu web stranicu. Ako zaposlenik koji ima nisku razinu privilegija otvori zaraženu web stranicu, a napadač nema pristup sustavu onda napadač uvijek dobiva pristup komponenti tog zaposlenika.

do_nothing : Ova radnja znači da zaposlenik nije izvršio nikakvu radnju. Za nju ne postoje nikakvi preduvjeti niti se nakon njezinog izvođenja išta mijenja u sustavu.

Osim ovih radnji mogu se dodati i nove radnje koje onda zaposlenici banke mogu izvršiti. Svaka od ovih radnji ima određenu vjerojatnost izvođenja koja se može mijenjati,

a zadaje se prilikom opisa sivog agenta. Svaki krug se za svakog zaposlenika slučajno odabire jedna radnja koja će se izvršiti ako su preduvjeti za tu radnju ispunjeni.

3.2.2. Obrambeni agent

Obrambeni agent je agent koji emulira rad osobe zadužene za sigurnost sustava. Modeliran je razredom `defender` koji nasljeđuje razred `agent`, i njegov naziv je `defender`. Razlikuje se od sivog agenta jer ima administratorske ovlasti unutar cijelog sustava, a skup radnji uključuje skup radnji koje može izvršiti osoba zadužena za sigurnost sustava.
Za potrebe ovog rada pojednostavljen je prikaz te osobe i radnje koje agent može izvršiti su:

check_component_connections : Ova radnja predstavlja provjeru otvorenih spojeva neke komponente. Preduvjeti za ovu radnju su da radnju izvodi obrambeni agent. Ako su preduvjeti ispunjeni svaki spoj odabrane komponente se provjerava i ako je spoj otvorio napadač onda se napadaču uklanja pristup komponentama tog spoja. Ako je napadač imao administratorsku razinu privilegija na nekoj od tih komponenata, ta razina se također briše.

do_nothing : Ova radnja znači da obrambeni tim nije izvršio nikakvu radnju. Za nju ne postoje nikakvi preduvjeti niti se nakon njezinog izvođenja išta mijenja u sustavu.

Svaka od ovih radnji ima vjerojatnost izvođenja i prema toj vjerojatnosti se slučajnim odabirom svaki krug odabire jedna radnja koja će se izvršiti.

4. Napadački agent

Napadački agent je agent koji emulira rad napadača. Modeliran je razredom `attacker` koji nasljeđuje razred `agent`. Napadač je definiran svojim znanjem o sustavu opisanom u potpoglavlju 4.1, radnjama koje može izvesti koje su opisane u potpoglavlju 4.2.2 i jednoj od strategija prema kojoj odabire radnju koju će izvesti koje su opisane u potpoglavlju 4.3.

4.1. Znanje o sustavu

Znanje o sustavu je podijeljeno na znanje o okolini sustava i znanje o kompromitaciji sustava. Znanje o okolini sustava se odnosi na znanje koje je napadač saznao o sustavu tijekom kompromitacije, a znanje o kompromitaciji se odnosi na razinu kompromitacije sustava koju je napadač ostvario unutar sustava. Znanje o okolini i kompromitaciji je opisano u sljedećim potpoglavljima.

4.1.1. Znanje o okolini

Jednom kada napadač sazna neku informaciju o sustavu ta informacija se sprema unutar njegovog znanja i ona nikad ne zastarijeva. Ako u nekom trenutku dođe do promjene unutar sustava i napadačevo znanje postane zastarjelo odnosno to znanje više nije točno, napadač to neće odmah znati, ali to može saznati daljnjom provjerom sustava. U emulaciji se to odnosi na aktivne spojeve između komponenata koje se mogu zatvarati i otvarati bez napadačevog znanja.
Znanje o okolini uključuje znanje o vjerodajnicama zaposlenika sustava, povezanim komponentama unutar sustava, lokalnim administratorskim računima na pojedinim komponentava, aktivnim spojevima unutar sustava i komponentama na koje pojedini računi mogu udaljeno pristupiti.

credentials : Svi korisničkih računa za koje napadač zna korisničko ime i lozinku i koje onda može koristiti za kompromitaciju sustava.

connected : Komponente za koje napadač zna da su međusobno povezane i da je omogućena komunikacija između tih komponenata.

local_admins : Znanje o lokalnim administratorskim računima za svaku komponentu unutar sustava koju je napadač kompromitirao.

active_connections : Znanje o svim aktivnim spojeva između komponenata koje je napadač otkrio prilikom kompromitacije sustava.

remote : Znanje o svim komponentama sustava na koje udaljeno mogu pristupiti pojedini korisnički računi za koje napadač zna.

4.1.2. Znanje o kompromitaciji

Tijekom napada napadač dobiva pristup pojedinim komponentama, dobiva administratorske ovlasti na komponentama, izvlači podatke s njih, pretražuje komponente i okriva povezane komponente i lokalne administratorske račune. Znanje o komponentama koje je napadač kompromitirao se sprema unutar napadačevog znanja o kompromitaciji. To znanje se koristi za provjeru preduvjeta pojedinih radnji koje napadač izvodi, a znanje uvijek odgovara razini kompromitacije napadača unutar sustava. Napadač može gubiti pristup komponentama i izgubiti administratorske ovlasti ako ga obrambeni agent otkrije i u tom slučaju se iz njegovog znanja briše znanje o pristupu komponenti i administratoskih ovlastima.
Znanje o kompromitaciji uključuje:

footholds : Znanje o svim komponentama unutar sustava kojima napadač ima pristup

probed_accounts : Znanje o svim komponentama unutar sustava koje je napadač ispitao u potrazi za lokalnim administratorskim računima na toj komponenti

exfiltrated : Znanje o svim komponentama unutar sustava s kojih je napadač preuzeo osjetljive podatke koji se nalaze na toj komponenti

enumerated : Znanje o svim komponentama unutar sustava koje je napadač pretražio i otkrio povezane komponente i aktivne spojeve na toj komponenti.

escalated : Znanje o svim komponentama unutar sustava na kojima napadač ima administratorske ovlasti.

exploited : Znanje o svim komponentama unutar sustava na kojima je napadač iskoristio ranjivost da bi došao do pristupa toj komponenti.

4.2. Napadačke radnje

Napadački agent na svom raspolaganju ima niz radnji koje može izvršiti. Jedan od načina na koji se radnje mogu prikazati je korištenjem napadačkih tehnika i taktika opisanih u bazi znanja organizacije MITRE [8]. MITRE baza znanja se još naziva i ATT&CK (*Adversarial Tactics, Techniques & Common Knowledge*), a njena glavna karakteristika je da se zasniva na promatranju ponašanja napadača u kibernetičkim napadima.

4.2.1. ATT&CK baza znanja

Baza znanja ATT&CK sastoji se od tri modula: ATT&CK za poduzeća, ATT&CK za mobilne platforme i ATT&CK za industrijske upravljačke sustave. Za potrebe modeliranja napadača u sklopu ovog rada korišten je modul ATT&CK za poduzeća [9] koji se sastoji od taktika i tehnika koji se koriste za kibernetičke napade protiv poduzeća. Taktike prikazuju taktičke ciljeve koje napadač pokušava ostvariti tijekom napada, a tehnike radnje koje napadač izvršava da bi ostvario svoj taktički cilj. Svaka taktika se sastoji od jedne ili više različitih tehnika koje napadač može izvršiti da bi ostvario cilj. Tehnike se sastoje od opisa radnje koju napadač izvodi, primjera iz stvarnog napada u kojem je korištena navedena tehnika i radnjama koje obrambeni tim može poduzeti da bi spriječio napadača da izvrši tu tehniku ili da otkrije napadača koji je već izvršio tu tehniku. Svaka tehnika može imati jednu ili više pod tehnika koje prikazuju različite načine kako se određena tehnika može izvršiti.

Modul ATT&CK se sastoji od 14 taktika:

Izviđanje (engl. *Reconnaissance*) se sastoji tehnika koje napadač izvršava prije pokretanja napada da bi saznao što više informacija o mreži i korisnicima unutar sustava kojeg napada. Napadač potom koriste te informacije za planiranje napada i tijekom napada.

Razvijanje resursa (engl. *Resource Development*) uključuje razvijanje vlastitih ili prikupljanje i mijenjanje postojećih resursa koje napadač potom koristi za izvršavanje tehnika tijekom napada da ostvari taktički cilj.

Inicijalni pristup (engl. *Initial Access*) se sastoji od tehnika koje koriste različite vektore napada da napadač dobije početni pristup sustavu koji napada.

Izvođenje (engl. *Execution*) je skup tehnika koje za cilj imaju izvršavanje malicioznog koda kojeg napadač kontrolira unutar sustava kojeg se napada ili na nekom drugom udaljenom sustavu.

Ustrajnost (engl. *Persistence*) je taktika koja za cilj ima osigurati napadaču pristup sustavu kojeg napada bez obzira na događaje unutar sustava koji mogu utjecati na pristup sustavu.

Eskalacija privilegija (engl. *Privilege Escalation*) je taktika koju napadač izvršava da bi došao do veće razine privilegija unutar mreže.

Zaobilaženje zaštitnih mjera (engl. *Defense Evasion*) podrazumijeva izvođenje tehnika koje smanjuju vjerojatnost da će napadač biti otkriven tijekom napada.

Dohvat vjerodajnica (engl. *Credential Access*) je taktika koja se sastoji od tehnika za krađu korisničkih imena i lozinki koje onda napadač može koristiti tijekom napada.

Otkrivanje (engl. *Discovery*) uključuje izvršavanje tehnika s ciljem povećanja znanja o internoj konfiguraciji mreže i detaljima sustava unutar kojeg se nalazi.

Lateralna kretnja (engl. *Lateral Movement*) je tehnika koju napadač izvršava da dobije pristup nad udaljenim komponentama sustava.

Prikupljanje (engl. *Collection*) je taktika koju napadač izvršava da sakupi informacije sa sustava na kojem se nalazi i koje mu služe da bi ostvario određeni cilj.

Naredbe i kontrola (engl. *Command and Control*) su skup tehnika koje objašnjavaju kako napadač, koji je izvan sustava, komunicira sa sustavom kojeg on kontrolira unutar mreže.

Eksfiltracija (engl. *Exfiltration*) je skup tehnika koje opisuju kako napadač preuzima podatke koje je prikupio s mreže koju napada i premješta ih na lokaciju po njegovom izboru.

Utjecaj (engl. *Impact*) se sastoji od tehnika koje napadač izvršava s ciljem da poremeti dostupnost ili kompromitira integritet sustava i procesa koji se izvršavaju u sustavu kojeg napada.

4.2.2. Implementirane napadačke radnje

Za potrebe ovog rada napadaču je omogućeno izvršavanje niza radnji koje predstavljaju pojednostavljene radnje ili taktike iz ATT&CK baze znanja. Radnje koje napadač može napraviti su:

initial_access : Radnja je temeljena na tehnici *slanje elektroničke pošte s malicioznim privitkom* koja je dio taktike *inicijalni pristup* . U ovom radu ova radnja predstavlja slanje elektroničke pošte koja sadrži maliciozni privitak zaposlenicima sustava. Ako zaposlenik otvori elektroničku poštu koju je poslao napadač, napadač dobiva pristup komponenti na kojoj taj zaposlenik radi. Radnja se može izvršiti bez preduvjeta, a izvršava se dokle god napadač nema pristup barem jednoj komponenti unutar sustava.

escalate_priviledges : Ova radnja je temeljena na tehnici *zloupotreba mehanizma za kontroliranje razine privilegija* koja je dio taktike *eskalacija privilegija*. U ovom radu predstavlja radnje koje napadač koristi da bi ostvario administratorsku razinu ovlasti na nekoj komponenti unutar sustava. Preduvjet za izvršavanje ove radnje je da napadač ima pristup komponenti sustava i da na toj komponenti nema administratorsku razinu ovlasti. Nakon uspješnog izvršavanja ove radnje napadač dobiva administratorske ovlasti na komponenti.

dump_credentials : Temeljena na tehnici *odbacivanje vjerodajnica operacijskog sustava* taktike *dohvat vjerodajnica* radnja prikazuje tehnike koje napadač koristi da bi saznao korisnička imena i lozinke svih korisnika koji svoje podatke imaju zapisane u memoriji računala. Preduvjeti za ovu radnju su da napadač ima pristup komponenti i da ima administratorske ovlasti na toj komponenti. Nakon uspješnog izvršavanja ove radnje napadač saznaje lozinke svih korisnika koji su prijavljeni na komponentu koji napadač kompromitira.

enumerate_host : Radnja predstavlja različite tehnike koje su dio taktike *otkrivanje*. Za uspješno izvršiti ovu radnju napadač treba imati pristup komponenti i imati administratorsku razinu ovlasti na toj komponenti. Nakon izvršavanja ove radnje napadač zna sve aktivne spojeve kompromitirane komponente (tehnika *otkrivanje perifernih uređaja*) i sve komponente sustava s kojima je kompromitirana komponenta sustava spojena (tehnika *otkrivanje mrežnih veza sustava*).

exfiltrate_data : Radnja je temeljena na taktici *eksfiltracija*. Preduvjeti za ovu radnju su da napadač ima administratorske ovlasti na komponenti i da je napravio radnju

enumerate_host za tu komponentu. Ako su preduvjeti zadovoljeni i napadač izvrši ovu radnju, on saznaje sve osjetljive podatke koji se nalaze pohranjeni na toj komponenti.

account_discovery : Ova radnja je temeljena na pod tehnici *domenski računi* tehnike *otkrivanje računa* taktike *otkrivanje* i prikazuje radnje koje napadač koristi za otkriti administratorske račune na nekoj komponenti unutar sustava. Za izvršiti ovu radnju napadač ne smije imati pristup komponenti, ali treba imati administratorske ovlasti na nekoj drugoj komponenti unutar sustava koja je povezana s komponentom za koju želi saznati administratorske račune i treba znati da su te komponente povezane. Nakon što se uspješno izvrši ova radnja, napadač saznaje sve administratorske račune na kompromitiranoj komponenti.

run_exploit : Napadač može iskoristiti ranjivosti unutar sustava za ostvariti cilj u više različitih taktika. U sklopu ovog rada ova radnja označava da je napadač iskoristio ranjivost koja mu omogućava da dobije pristup nekoj novoj komponenti unutar sustava što je temeljeno na tehnici *iskorištavanje udaljenih usluga* taktike *lateralna kretnja*. Preduvjeti ove radnje su da napadač ima pristup i administratorske ovlasti na jednoj komponenti sustava, znati da je ta komponenta povezana s nekom drugom komponentom sustava kojoj napadač nema pristupa i da napadač nije već pokušao iskoristiti ranjivost te komponente. Ako napadač ima metodu iskorištavanja za neki program koji je instaliran na komponenti koju pokušava napasti onda uspješno iskorištava ranjivost i dobiva pristup toj komponenti. Ako između tih povezanih komponenata ne postoji aktivan spoj onda se otvara novi spoj između te dvije komponente.

lateral_movement : Tehnika *udaljene usluge* taktike *lateralna kretnja* omogućuje napadaču korištenje ukradenih vjerodajnica za udaljeno pristupanje drugim komponentama unutar sustava. Za izvršiti ovu radnju napadač treba imati pristup i administratorske ovlasti na jednoj komponenti sustava, znati da je ta komponenta povezana s nekom drugom komponentom sustava kojoj napadač nema pristupa i imati vjerodajnice računa kojemu je omogućeno udaljeno pristupanje komponenti na kojoj napadač nema pristupa. Ako napadač uspješno izvrši ovu radnju dobije pristup komponenti za koju su ispunjeni preduvjeti. Ako između te komponente i komponente s koje je napadač izvršio lateralni pomak ne postoji aktivan spoj onda se otvara novi spoj između te dvije komponente. Također, ako je napadač koristio vjerodajnice računa koji ima administratorske ovlasti na

komponenti koju napada, onda napadač dobiva administratorske ovlasti na toj komponenti. To je primjer korištenja tehnike *valjani računi* taktike *eskalacija privilegija*.

4.3. Napadačke strategije

Svaka strategija koja se koristi je definirana svojim razredom. Razred osim konstruktora ima metodu za odabir radnje i komponente za koju će radnja biti izvršena i metodu za osvježavanje zadnje izvršene radnje.
Napadač na svom raspolaganju ima jednu od sljedećih strategija za odabir radnje koja će biti izvršena:

Random : strategija koja slučajno odabire radnju koja će se izvršiti. Strategija uzima sve radnje koje napadač može izvršiti i raspoređuje ih slučajnim redoslijedom. Napadač pokušava izvršavati radnje po redu sve dok uspješno ne izvrši jednu radnju.

Greedy : strategija koja pohlepno odabire radnju koja će se izvršiti. Ova strategija uvijek prvo pokušava izvršiti radnje `exfiltrate_data`, `enumerate_host`, `lateral_movement` ili `account_discovery` tim redoslijedom. Ako se nijedna od tih radnji ne uspije izvršiti jer preduvjeti nisu zadovoljeni, strategija slučajno bira između ostalih radnji dok ne izabere radnju koja se uspješno izvrši.

Finite_State_Machine : strategija koja predstavlja stroj s konačnim brojem stanja. Ova strategija pokušava slijedno izvršavati radnje uvijek istim redoslijedom. Prilikom inicijalizacije radnje se raspoređuju u listu tako da se slijedno izvršavaju `escalate_priviledges`, `dump_credentials`, `enumerate_host`, `exfiltrate_data`, `account_discovery`, `lateral_movement`, `run_exploit`. Ako neka postoje još neke radnje koje se mogu izvršiti, one će biti dodane slučajnim redoslijedom nakon navedenih radnji.
Nakon što se neka radnja uspješno izvrši, strategija pamti zadnju izvršenu radnju i sljedeća radnja koja će se pokušati izvesti će biti ona koja slijedi izvršenu radnju. Iznimka je radnja `run_exploit` koja se neće izvršiti ako je prije nje uspješno izvršena radnja `lateral_movement`

Pretpostavka je da će tijekom testiranja napadač koji ima `Random` strategiju imati najslabije rezultate, a da će agent sa strategijom `Finite_State_Machine` imati nešto bolje rezultate od agenta s `Greedy` strategijom.

5. Testiranje napadača i vizualizacija sustava

Model kibernetičkog prostora i agenata unutar sustava korišten tijekom testiranja opisan je u prethodnim poglavljima. Tijekom testiranja mijenjali su se parametri sustava i zabilježeni su rezultati koje je napadač ostvario nakon svakog testiranja koristeći pojedine strategije opisane u poglavlju 4.3.

Svako testiranje je provedeno na uzorku od 50 emulacija sustava. Na početku svake emulacije program učita ulazne datoteke, modelira kibernetički sustav i inicijalizira zaposlenike i agente unutar sustava. Nakon toga sivi agent izvodi radnje zaposlenika sustava sve dok napadač ne dobije pristup sustavu. Napadač može dobiti pristup sustavu ako jedan zaposlenik otvori elektroničku poštu koju je poslao napadač ili ako surfajući internetom zaposlenik otvori zaraženu web stranicu. Slanje elektroničke pošte je dio emulacije i napadač šalje elektroničku poštu izvršavanjem radnje `initial_access`, a kompromitacija web stranice spada pod taktiku `Razvijanje resursa` i nije implementirana unutar emulacije nego je podrazumijevano da je napadač to napravio prije pokretanja napada i da postoji zaražena web stranica koja daje napadaču pristup sustavu ako je posjeti jedan zaposlenik sustava. Kada napadač dobije pristup sustavu emulacija se jednom pokrene da se zabilježe radnje zaposlenika i obrambenog agenta. Broj poteza unutar emulacije je ograničen i jednak je dvostrukom broju komponenata unutar emulacije. Radnje koje su izvršili zaposlenici i obrambeni agent je potrebno zabilježiti tako da se svaka strategija može testirati koristeći iste radnje. Tako se može usporediti rezultate svake strategije i vidjeti za koju strategiju automatizirani napadač daje najbolje rezultate.

Za potrebe testiranja mijenjani su parametri vjerojatnosti izvođenja radnje zaposlenika i obrambenog agenta. Tablica 5.1 prikazuje parametre zaposlenika koji su mijenjani tijekom testiranja, a tablica 5.2 parametre obrambenog agenta. Parametri u stupcu `Vrijednost` pokraj svake radnje predstavljaju težinu te radnje, a vjerojatnost da će neka radnja biti odabrana je jednaka težini radnje podijeljenoj sa zbrojem težina

svih radnji koje agent može izvršiti. Kombinacijom ovih dviju tablica dobiju se četiri različita testa za koje je provedena emulacija. Za svaki test je provedena emulacija i na kraju emulacije je zabilježeno koliko je napadač imao pristupa različitim komponentama u sustavu, s koliko različitih komponenti u sustavu je izvukao podatke i koliko je različitih korisničkih imena i lozinki napadač saznao.

Naziv testa	Opis	Vrijednost
Jednako	Sve radnje imaju jednaku vjerojatnost	`reboot_host` 1 `user_login_to_host` 1 `open_connection_between_hosts` 1 `close_connection_between_hosts` 1 `open_email` 1 `send_email` 1 `browser_internet` 1 `do_nothing` 1
Miješano	Radnje imaju miješanu vjerojatnost izvođenja	`reboot_host` 5 `user_login_to_host` 10 `open_connection_between_hosts` 25 `close_connection_between_hosts` 25 `open_email` 10 `send_email` 10 `browser_internet` 10 `do_nothing` 5

Tablica 5.1: Testiranje zaposlenika sustava

Naziv testa	Opis	Vrijednost
Pasivno	Napadač rijetko provjerava komponente	`check_component_connections` 25 `do_nothing` 75
Aktivno	Napadač često provjerava komponente	`check_component_connections` 75 `do_nothing` 25

Tablica 5.2: Testiranje obrambenog tima

5.1. Slučajna strategija

Rezultati testiranja za slučajnu strategiju prikazani su u tablici 5.3. Iz tablice je vidljivo da napadač ostvaruje bolje rezultate u slučaju kada je obrambeni agent pasivan, odnosno kada se rjeđe provodi provjeravanje komponenti unutar sustava. Vidljivo je da razlika u vjerojatnosti izvođenja radnji zaposlenika nije bitnije utjecala na broj komponenata kojima je napadač dobio pristup niti broju komponenata s kojih je napadač izvukao podatke.

Zaposlenici obrambeni	Uporišta	Eksfiltracija	Vjerodajnice
Jednako pasivno	8,49 %	7,98 %	12,52 %
Jednako aktivno	5,03 %	5,75 %	10,59 %
Miješano pasivno	9,92 %	8,50 %	14,79 %
Miješano aktivno	6,40 %	6,65 %	11,68 %

Tablica 5.3: Rezultati testiranja za slučaju strategiju

5.2. Pohlepna strategija

Rezultati testiranja za pohlepnu strategiju prikazani su u tablici 5.4. Iz tablice se vidi da pohlepna strategija ostvaruje minimalno lošije rezultate u slučaju kada obrambeni agent češće provjerava komponente sustava i pokušava otkriti prisutnost napadača unutar sustava ako zaposlenici izvode svoje radnje s različitim vjerojatnostima, dok u slučaju kada zaposlenici izvršavaju svoje radnje s jednakim zaposlenicima napadač ima 2,3 % manje uporišta i preuzima podatke s 1,7 % komponenata manje kada obrambeni agent provodi aktivnu potragu za napadačem.

Zaposlenici obrambeni	Uporišta	Eksfiltracija	Vjerodajnice
Jednako pasivno	11,21 %	9,87 %	8.68 %
Jednako aktivno	8,89 %	8,17 %	8,21 %
Miješano pasivno	10,49 %	8,89 %	9,34 %
Miješano aktivno	10,35 %	8,88 %	8,93 %

Tablica 5.4: Rezultati testiranja za pohlepnu strategiju

5.3. Strategija s konačnim brojem stanja

Rezultati testiranja za strategiju s konačnim brojem stanja prikazani su u tablici 5.5. Iz tablice se vidi da strategija s konačnim brojem stanja ostvaruje oko 1 % bolji rezultat kada zaposlenici obavljaju svoje radnje s jednakom vjerojatnošću nego kada obavljaju radnje s različitom vjerojatnošću. Također se vidi pad rezultata u slučaju kada obrambeni agent češće provodi provjeru komponenata, u slučaju kada zaposlenici izvode radnje s jednakom vjerojatnošću pad je oko 1,5 %, a u slučaju kada zaposlenici izvode radnje s različitom vjerojatnošću par je 0,6 %.

Zaposlenici obrambeni	Uporišta	Eksfiltracija	Vjerodajnice
Jednako pasivno	11,75 %	11,35 %	14,83 %
Jednako aktivno	10,26 %	9,91 %	13,37 %
Miješano pasivno	10,17 %	9,83 %	13,83 %
Miješano aktivno	9,55 %	9,21 %	14,16 %

Tablica 5.5: Rezultati testiranja za strategiju s konačnim brojem stanja

5.4. Usporedba strategija

U potpoglavlju 4.3 je iznesena pretpostavka da će najslabiji rezultat imati slučajna strategija dok će strategija s konačnim brojem stanja imati nešto bolje rezultate od pohlepne strategije. Prosječni rezultat za svaki parametar koji je zabilježen nakon emulacije je prikazan u tablici 5.6. Iz tablice je vidljivo da je iznesena pretpostavka potvrđena. Najbolja strategija prema rezultatima je `strategija s konačnim brojem stanja`, nešto lošija je `pohlepna strategija` dok je `slučajna strategija` najlošija. Napadači koji koriste `pohlepnu strategiju` ili `strategiju s konačnim brojem stanja` ostvaruju približno jednak broj uporišta, ali napadač sa `strategijom s konačnim brojem stanja` uspijeva izvući podatke s više komponenata unutar sustava. Iako napadač sa `slučajnom strategijom` uspijeva saznati veći broj vjerodajnica od napadača s `pohlepnom strategijom`, on te vjerodajnice ne koristi da bi ostvario veći broj uporišta unutar sustava i preuzeo podatke s više komponenata sustava zbog čega je `slučajna strategija` lošija od `pohlepne strategije`.

Prosječan rezultat	slučajna strategija	pohlepna strategija	strategija s konačnim brojem stanja
Uporišta	7,46 %	10,24 %	10,43 %
Eksfiltracija	7,22 %	8,95 %	10.08 %
Vjerodajnice	13.40 %	8,79 %	14.05 %

Tablica 5.6: Prosječni rezultat za svaku strategiju

Tablica 5.7 prikazuje koliko često su neke radnje izvršene tijekom emulacije različitim strategijama. Iz tablice se vidi da je postotak izvršavanja radnji podjednak za svaku strategiju, a najčešća radnje koja se izvodi je *dump_credentials*. To je jedina radnja koja se na nekoj komponenti može izvršiti više puta što utječe na to da se ona najčešće uspješno izvodi. Od ostalih radnji, učestalost izvođenja radnje *run_exploit* se može povećati tako da se poveća broj ranjivosti za koje napadači imaju metodu isko-

rištavanja. Tako će napadač moći dobiti uporište na više komponenata unutar sustava što će povećati i učestalost izvođenja ostalih radnji koje se mogu izvršiti samo jednom na pojedinoj komponenti. Iako se radnje izvode jednako često rezultati za pojedine strategije su bolji što se vidi u tablici 5.5. Iz toga se može zaključiti da za uspješno izvršavanje napada nije dovoljno samo odabrati radnje koje će biti izvršene, nego je bitan i redoslijed kojim se radnje izvršavaju.

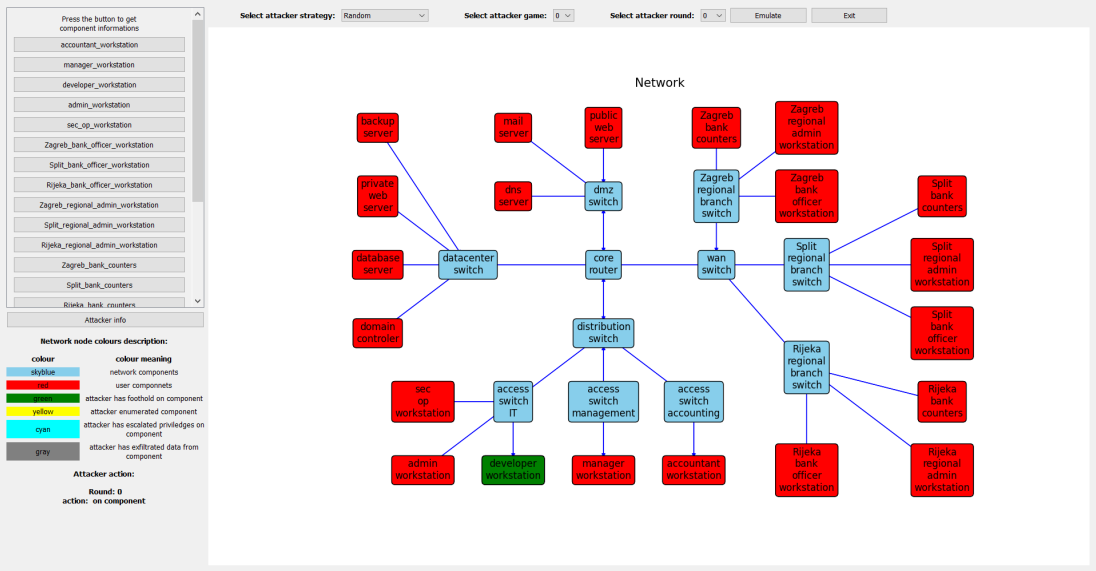
Prosječan rezultat	**slučajna strategija**	**pohlepna strategija**	**strategija s konačnim brojem stanja**
`escalate_priviledges`	7,01 %	7,14 %	6,98 %
`dump_credentials`	23,91 %	24,32 %	24,15 %
`enumerate_host`	10,26 %	10,29 %	10,09 %
`exfiltrate_data`	10,38 %	10,40 %	10,21 %
`account_discovery`	15,77 %	15,55 %	15,57 %
`run_exploit`	5,89 %	5,91 %	5,86 %
`lateral_movement`	3,40 %	3,35 %	3,69 %

Tablica 5.7: Prosječno izvršavanje radnji napadača

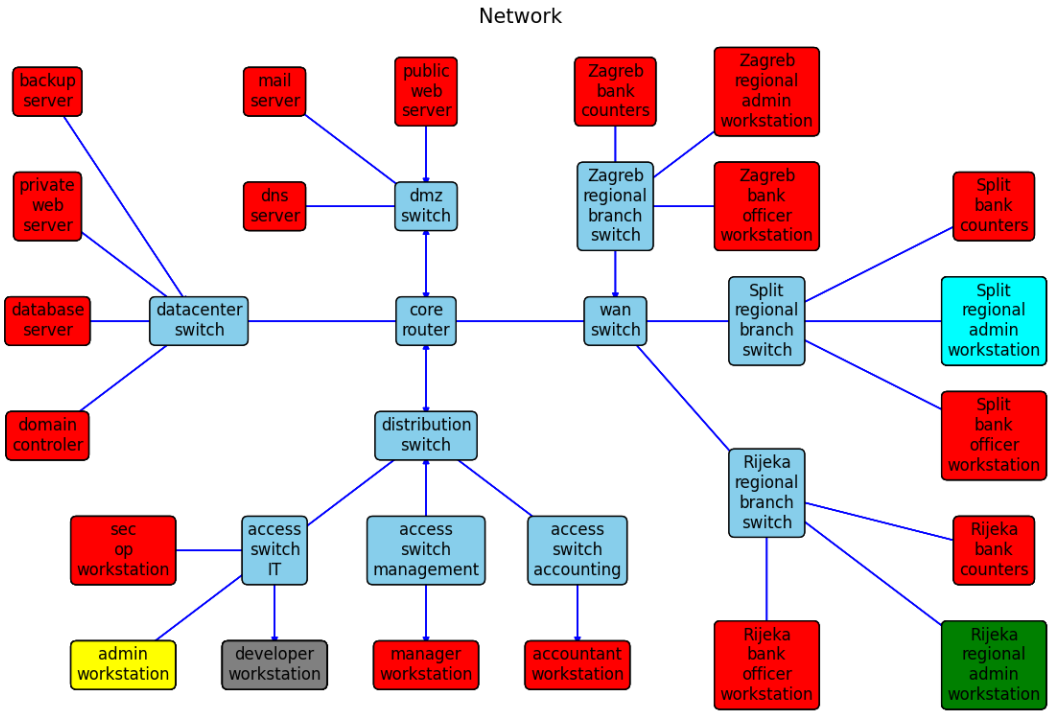
5.5. Vizualizacija kibernetičkog prostora

Program koji je napravljen u sklopu ovog rada vizualizira model informacijskog sustava nakon što emulira kibernetički napad. Za vizualizaciju modela informacijskog sustava korištene su knjižnice PyQt5 [18] i matplotlib [19]. Knjižnica PyQt5 je korištena za izradu korisničkog sučelja i dodavanje komponenata tom sučelju, a knjižnica matplotlib je korištena za izradu platna na kojem se vizualizirala mreža. Za vizualizaciju mreže korištena je knjižnica networkx [20] koja omogućava stvaranje i mijenjanje mreža koje su prikazane grafom u kojem čvorovi grafa predstavljaju komponente sustava, a bridovi povezanost između komponenata.

Slika 5.1 prikazuje vizualizaciju modela informacijskog sustava koji je predan programu prilikom pokretanja. Središnji dio slike je prikaz modela sustava. Kao ulazni parametar programu se predaje datoteka u kojoj se nalaze pozicije na kojima će komponente sustava biti smještene, a ako za neku komponentu ne postoji pozicija ona će biti slučajno smještena negdje na slici. Na slici 5.2 se može bolje vidjeti vizualizacija mreže koju je program generirao. Svaki čvor u mreži predstavlja jednu komponentu unutar kibernetičkog prostora organizacije. Sve komponente koje predstavljaju radne stanice zaposlenika banke koji rade na istim pozicijama nisu prikazane posebno nego je prikaz takvih komponenata pojednostavljen i prikazan kao samo jedna komponenta sustava.



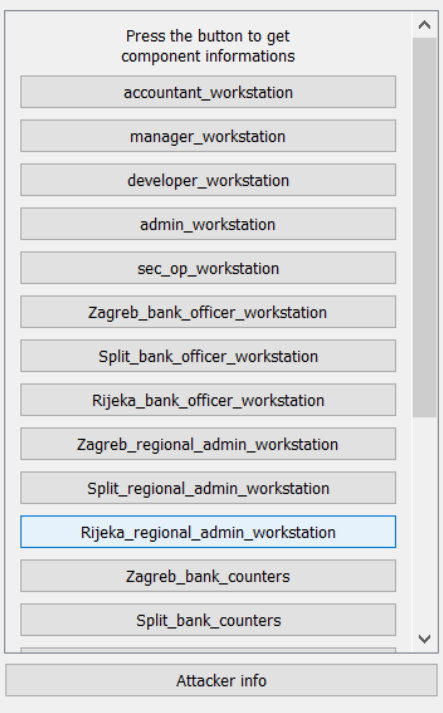
Slika 5.1: Prikaz modela informacijskog sustava



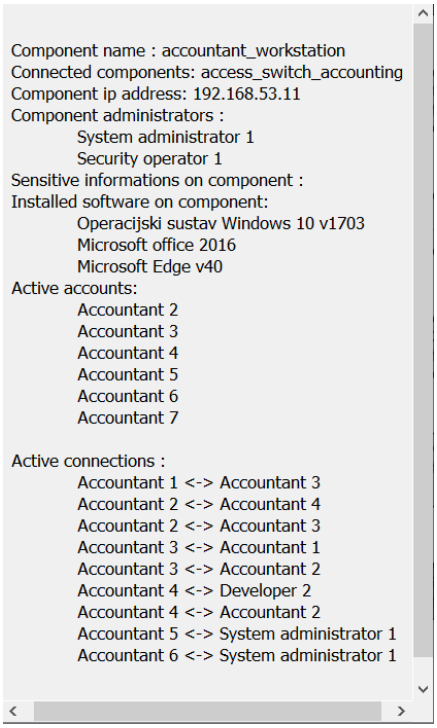
Slika 5.2: Vizualizacija mreže

Detalje o pojedinoj komponenti, kao što su naziv komponente, računi koji su pristupili pojedinoj komponenti, aktivni spojevi i druge informacije o komponenti je moguće dohvatiti pritiskom na gumb za pojedinu komponentu na lijevoj strani prikaza. Slika 5.3 prikazuje panel s gumbima za dobivanje informacija o pojedinoj komponenti, a

slika 5.4 prikazuje prozor koji se otvori pritiskom na gumb i koji sadrži detalje vezane za odabranu komponentu.



Slika 5.3: Panel s gumbima za komponente



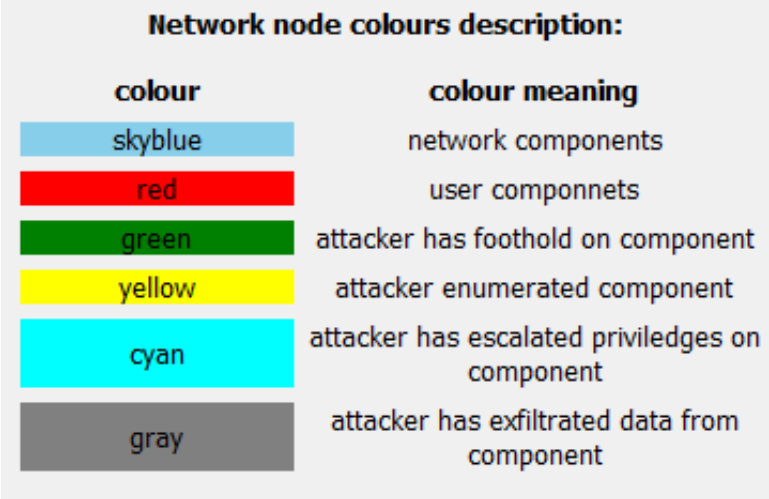
Slika 5.4: Prikaz detalja komponente sustava

Vizualizacija informacijskog sustava omogućuje analizu emuliranih kibernetičkih napada. Za odabir pojedine strategije i emulirane igre koristi se traka koja se nalazi iznad prikaza mreže, a prikazana je na slici 5.5. Za svaku strategiju i emuliranu simulaciju moguće je pogledati kako izgleda mreža u određenoj rundi. Strategija za koju se želi pogledati simulacija se odabire u padajućoj listi pokraj teksta *Select attacker strategy:*, a odabir simulacije u padajućoj listi pokraj teksta *Select attacker game:*. Nakon što je odabrana strategija i simulacija može se odabrati runda odabirom u padajućoj listi pokraj teksta *Select attacker round:*. Kada se odabere runda za koju se želi pogledati stanje mreže program vizualizira model sustava i različitim bojama označava kompromitirane komponente. Slika 5.6 prikazuje legendu u kojoj je opisano što koja boja označava. Koristeći legendu i mrežu na slici 5.2 može se zaključiti da napadač ima uporište na jednoj *Rijeka_regional_admin_workstation* komponenti, da je enumerirao komponentu *admin_workstation*, a izvukao podatke s komponente *developer_workstation* i da ima administratorske ovlasti na komponenti *Split_regional_admin_workstation*. Slika 5.7 prikazuje koju je radnju napadač izvršio tijekom runde koja se prikazuje.

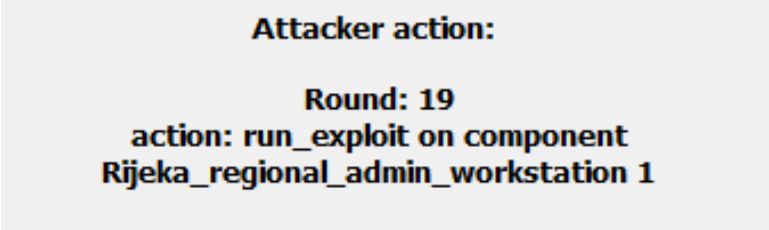
Pritiskom na gumb *Emulate* koji se nalazi unutar trake na slici 5.5 program pokreče

vizualizaciju igre i prikazuje kako se stanje u sustavu mijenjalo tijekom izvođenja igre. Tako je moguće vidjeti koje je radnje napadač izvršavao i kojim redoslijedom su komponente bile kompromitirane.
Pritiskom na gumb *Exit* program završava svoje izvođenje.

Slika 5.5: Odabir emulacije



Slika 5.6: Legenda



Slika 5.7: Prikaz radnje napadača

6. Poboljšanje emulacije

Emulacija izrađena za potrebe ovog rada je pojednostavljen prikaz emulacije kibernetičkog napada. Za svaku komponentu emulacije je moguće napraviti poboljšanja koja će doprinijeti realističnijem prikazu sustava i napada koji se emulira.

6.1. Poboljšanje sustava

Za emuliranje informacijskog sustava korišten je pojednostavljen prikaz kibernetičkog terena banke iz kojeg su prikazane samo komponente sustava i programi koji su instalirani na tim komponentama. Neke od stvari koje su zanemarene ili pojednostavljene za potrebe emulacije su mrežni promet, udaljeni pristup komponentama, vatrozid, sustavi za otkrivanje i sprječavanje napada i resursi komponenata.

U emulaciji je mrežni promet između komponenata i mrežni promet koji izlazi iz mreže organizacije zanemaren. Za aktivne spojeve između komponenata je u emulaciji samo zabilježeno da postoje dok će se u stvarnom svijetu između tih komponenata odvijati prijenos podataka putem mreže i u tom prijenosu će sudjelovati mrežni uređaji koji nisu korišteni tijekom emulacije. IP adrese komponenata u emulaciji postoje a one se također koriste u mrežnom prometu.

Udaljeni pristup komponentama je pojednostavljen. Zaposlenici mogu udaljeno pristupiti komponentama na kojima imaju pristup, ali se nigdje ne provjeravaju vjerodajnice zaposlenika nego se samo zabilježi da su udaljeno pristupili komponenti. Također se ne generira mrežni promet koji bi se inače generirao kada se netko pokušava udaljeno spojiti na neku komponentu.

Vatrozid je za potrebe emulacije u potpunosti zanemaren. Vatrozid obavlja filtriranje prometa i onemogućava komunikaciju između čvorova ako pravila nisu zadovoljena. Vatrozid može biti konfiguriran na više mrežnih uređaja i imati različita pravila. Emulacija koja sadrži vatrozid i pravila bi realističnije prikazala informacijski sustav neke organizacije.

Sustav za sprječavanje napada i sustav za otkrivanje napada su bitni dijelovi svake

organizacije u sprječavanju i otkrivanju napada. Sustav za otkrivanje napada prati ponašanje sustava i prometa na mreži s ciljem otkrivanja zloćudne aktivnosti unutar mreže. Sustav za sprječavanje napada uz otkrivanje radi i sprječavanje zloćudne aktivnosti. U emulaciji za potrebe ovog rada nisu korišteni ovi sustavi iako bi u realnom prikazu sustava organizacije oni bili prisutni.

Resursi komponenata kao što su potrošnja memorije i diska, rad procesora i specifikacije tih komponenata također se trebaju uzeti u obzir prilikom emulacije informacijskog sustava. Resursi komponenata bi se trebali uzeti u obzir prilikom računanja koliko je vremena potrebno da se izvrši neka radnja.

6.2. Poboljšanje agenata

Agenti u emulaciji glume osobe koji su sudionici simulacije. Radnje koje su korištene u emulaciji su pojednostavljen prikaz radnji koje se izvršavaju u stvarnom svijetu, a neke radnje su zanemarene što je objašnjeno u sljedećim potpoglavljima.

Tijekom emulacije pretpostavljeno je da je vrijeme izvođenja radnji jednako za svaku radnju svakog agenta. U stvarnom svijetu vrijeme izvođenja radnji ovisi o radnji koja se izvodi i mrežnom prometu kojeg stvara ta radnja. Također je i zanemareno vrijeme između izvođenja radnji. Tako u emulaciji agenti izvode svaku rundu jednu radnju dok se u stvarnom svijetu neke radnje mogu izvoditi paralelno i moguće je da protekne više vremena između dvije radnje nekog agenta.

Svim agentima je zajedničko da se njihove radnje uvijek uspješno izvode ako su zadovoljeni preduvjeti iako to u stvarnom svijetu neće uvijek biti tako. Unutar mreže je moguće da dođe do pogreške na nekoj komponenti ili da se zbog nekog vanjskog razloga neka radnja ne izvrši uspješno iako su preduvjeti ispunjeni. To je još jedna stvar o kojoj treba razmišljati prilikom izrade emulacije.

Još jedna stvar koja je zajednička svim agentima je da su ljudske osobine zanemarene. Nemaju svi sudionici emulacije jednake tehničke, socijalne i kognitivne mogućnosti i to je također nešto o čemu bi trebalo razmišljati prilikom izrade emulacije [10].

6.2.1. Sivi agent

Sivi agent emulira zaposlenike organizacije. U emulaciji zaposlenici imaju samo jednostavne radnje koje izvršavaju dok je u stvarnom svijetu broj radnji koje oni izvršavaju puno veći i radnje ovise o poslu kojeg zaposlenik unutar organizacije obavlja. Tako

na primjer zaposlenici banke mogu upravljati bankovnim računima korisnika banke, uplaćivati i isplaćivati im novac, otvarati i zatvarati račune u banci što nije podržano u simulaciji.

6.2.2. Obrambeni agent

Obrambeni agent koji glumi osobu zaduženu za sigurnost sustava je najmanje realistično prikazan u emulaciji. Neke od radnji koje bi trebao moći izvršiti su otkrivanje napada analizom zapisa u dnevniku, dodavanje pravila vatrozida, dodavanje novih i uklanjanje postojećih korisničkih računa, mijenjanje razine privilegija drugih korisnika na mreži, instalacija programa na komponentama. Ako obrambeni agent otkrije prisutnost napadača unutar sustava trebao bi moći poduzeti dodatne radnje koje uključuju vračanje komponenata na stanje iz sigurnosne kopije, iskapčanje računala s mreže ili slanje računala na forenzičku analizu.

6.2.3. Napadački agent

Naglasak ovog rada je bio na implementaciji napadačkog agenta. Radnje koje napadač izvodi su pojednostavljeni prikaz radnji koje se izvode u stvarnom svijetu. Iako u stvarnom svijetu napadač može imati više različitih tehnika s kojima ostvaruje cilj neke taktike u emulaciji je prikazana samo jedna tehnika za svaku taktiku.

Tijekom kompromitacije sustava napadač saznaje informacije o sustavu i zaposlenicima. Te informacije mogu koristiti napadaču prilikom planiranja ili izvršavanja različitih radnji. Informacije koje napadač saznaje mogu biti raznolike i potrebno je pronaći način kako zabilježiti te informacije unutar napadačevog znanja tako da se te informacije mogu koristiti prilikom izvršavanja radnji.

Daljnji napredak napadača je moguće ostvariti razvijanjem novih strategija koje će bolje iskoristiti znanje koje je napadač saznao o sustavu i na temelju tog znanja planirati radnje koje će se sljedeće izvršiti.

7. Zaključak

Provođenje sigurnosnim testiranja je važno za svaku organizaciju koja se želi zaštititi od kibernetičkih napada. Iako je nemoguće biti potpuno zaštićen od napada, provođenjem redovitih testiranja sustava moguće je pronaći ranjivosti i zakrpati ih prije nego ih napadači iskoriste. Osim pronalaska ranjivosti, provođenjem redovitih testiranja organizacije se mogu bolje pripremiti za napad na njihov sustav izradom plana djelovanja u slučaju napada za čiju izradu koriste iskustvo i znanje dobiveno tijekom testiranja sustava [1]. Emuliranje sigurnosnih testiranja omogućuje češće provođenje sigurnosnih testiranja zbog manje količine resursa potrebnih za izvođenje takvog testiranja. Takva testiranja moraju realistično prikazati tehnike koje napadač koristi prilikom napada da se rezultati iz takvih testiranja mogu koristiti za unapređenje sigurnosti sustava organizacije.

Automatizacija napadača predstavlja izazov jer je nemoguće znati koji su njegovi ciljevi, koje tehnike ima na raspolaganju kao ni njegove tehničke, socijalne i kognitivne osobine. Planiranje i redoslijed izvršavanja radnji također predstavljaju izazov prilikom automatizacije napadača jer tijekom napada napadač nema potpuno znanje o sustavu nego izvodi radnje za koje on smatra da će mu pomoći da ostvari svoj cilj. Napadač treba na temelju svog znanja o sustavu odabrati radnju koju će izvršiti, i nakon svake izvršene radnje treba ponovno prilagoditi plan napada s novim znanje kojeg je dobio izvršavanjem prošle radnje.

U sklopu ovog rada razvijen je program koji emulira kibernetički napad na organizaciju. Program prilikom pokretanja iz ulaznih parametara generira kibernetički prostor i agente koji sudjeluju u emulaciji. Nakon toga program izvodi emulaciju kibernetičkog napada u sklopu kojeg se testiranju strategije pomoću koji napadač odlučuje koju radnju će izvršiti. Za svaku emulaciju svi sudionici osim napadača uvijek izvršavaju iste radnje tako da se mogu usporediti rezultati koje je napadač ostvario i vidjeti za koju strategiju napadač ostvaruje cilj. Nakon što program izvrti sve emulacije, pokreće se komponenta za vizualizaciju sustava koja omogućuje praćenje napadačkih akcija i redoslijed kojim napadač kompromitira sustav.

Za testiranje sustava su korištene strategije koje ne koriste planiranje za odabir radnje koja će se sljedeća izvršiti. Svaka radnja se izvrši podjednak broj puta, neovisno o strategiji prema kojoj je odabrana, ali rezultati na kraju testiranja su različiti što pokazuje da je bitan i redoslijed izvršavanja radnji. Inicijalna pretpostavka da će strategija sa slučajnim odabirom radnji biti najlošija se pokazala točnom, a potvrđena je i pretpostavka da će od korištenih strategija najbolja biti ona s konačnim brojem stanja.
Za daljnje unapređenje sustava potrebno je napraviti strategije koje koriste planiranje za odabir radnje koja će se izvršiti i testirati takve strategije na navedenom sustavu. Za obrambenog agenta treba definirati veći skup radnji koje on može izvršiti, a model sustava se može unaprijediti dodavanjem mrežnog prometa unutar sustava.

LITERATURA

[1] Yoo, J. D., Park, E., Lee, G., Ahn, M. K., Kim, D., Seo, S., & Kim, H. K *Cyber Attack and Defense Emulation Agents* Ožujak 2020. [pristupljeno 16.6.2021]

[2] Ollmann, G. , "Penetration Testing With Honest-To-Goodness Malware" `https://www.darkreading.com/attacks-breaches/penetration-testing-with-honest-to-goodness-malware/d/d-id/1140588?` [pristupljeno 16.6.2021]

[3] Kello, L *The meaning of the cyber revolution: Perils to theory and statecraft.* International Security, 2013, 38.2: 7-40 [pristupljeno 16.6.2021]

[4] Applebaum, A., Miller, D., Strom, B., Korban, C., & Wolf, R *Intelligent, automated red team emulation.* Prosinac 2016. [pristupljeno 16.6.2021]

[5] Mudge, R. "Adversary Simulation Becomes a Thing" `https://blog.cobaltstrike.com/2014/11/12/adversary-simulation-becomes-a-thing/` Studeni 2014 [pristupljeno 16.6.2021]

[6] Applebaum, A., Miller, D., Strom, B., Foster, H., & Thomas, C. *Analysis of automated adversary emulation techniques.* Proceedings of the Summer Simulation Multi-Conference 2017. p. 1-12.

[7] Miller, D., Alford, R., Applebaum, A., Foster, H., Little, C., & Strom, B. *Automated adversary emulation: A case for planning and acting with unknowns* MITRE CORP MCLEAN VA MCLEAN 2018

[8] Adversarial Tactics, Techniques & Common Knowledge (ATT&CK), MITRE, `https://attack.mitre.org/` [pristupljeno 17.6.2021]

[9] Enterprise Matrix, MITRE, `https://attack.mitre.org/matrices/enterprise/` [pristupljeno 17.6.2021]

[10] Kavak, H., Padilla, J. J., Vernon-Bido, D., Gore, R., & Diallo, S. *A Characterization of Cybersecurity Simulation Scenarios* Travanj 2016 [pristupljeno 19.6.2021]

[11] Metasploit framework, `https://www.metasploit.com/` [pristupljeno 10.6.2021]

[12] Obes, J. L., Sarraute, C., & Richarte, G. *Attack Planning in the RealWorld* arXiv preprint arXiv:1306.4044 2013 [pristupljeno 19.6.2021]

[13] NIS Directive `https://www.enisa.europa.eu/topics/nis-directive?tab=details` [pristupljeno 25.6.2021]

[14] PCI DSS Directive `https://www.pcidss.com/` [pristupljeno 25.6.2021]

[15] Zero-day vulnerability `https://en.wikipedia.org/wiki/Zero-day_(computing)` [pristupljeno 25.6.2021]

[16] Some Common Attack Vectors `http://gauss.ececs.uc.edu/Courses/c6055/pdf/attackvectors.pdf` [pristupljeno 25.6.2021]

[17] Greenberg, A. *The Untold Story of NotPetya, the Most Devastating Cyberattack in History* Wired, August, 22. (2018) [pristupljeno 25.6.2021]

[18] PyQt5 `https://pypi.org/project/PyQt5/` [pristupljeno 27.6.2021]

[19] matplotlib `https://pypi.org/project/matplotlib/` [pristupljeno 27.6.2021]

[20] networkx `https://pypi.org/project/networkx/` [pristupljeno 27.6.2021]

[21] MITRE CALDERA `https://github.com/mitre/caldera` [pristupljeno 27.6.2021]

[22] Atomic Red Team `https://atomicredteam.io/` [pristupljeno 27.6.2021]

[23] Infection Monkey `https://github.com/guardicore/monkey` [pristupljeno 27.6.2021]

[24] APT Simulator `https://github.com/NextronSystems/APTSimulator` [pristupljeno 27.6.2021]

Emulacija automatiziranih napadača u kibernetičkoj sigurnosti

Sažetak

Provođenje sigurnosnih testiranja je važno za održavanje sigurnosti sustava unutar organizacije. Sigurnosna testiranja zahtijevaju resurse i angažman stručnjaka zbog čega se testiranja pokušavaju automatizirati. Prilikom izrade automatiziranog napadača potrebno je voditi računa o njegovom znanju, mogućnostima koje ima na raspolaganju i strategiji kojom želi ostvariti svoj cilj. U sklopu ovog rada napravljen je program koji modelira IT sustav i simulira kibernetički napad na taj sustav. Napravljene su različite napadačke strategije koje utječu na odabir radnje koju će napadač izvršiti tijekom napada. Provodeći niz testiranja došlo se do rezultata da napadači koji slučajno odabiru radnje ostvaruju najslabije rezultate, a najbolje rezultate napadači ostvaruju kada zaposlenici unutar sustava provode radnje s različitim vjerojatnostima. Na kraju je dan prijedlog mogućeg poboljšanja sustava.

Ključne riječi: Automatizacija napadača, emulacija napada

Emulation of Automated Attackers in Cybersecurity

Abstract

Security testing is an important part of assessing system security inside the organization. Factors such as cost and personnel are the reason why the security testing is being automated. Automated cyber attacker should be aware of its knowledge, capabilities it can execute and strategy it uses to reach his goal. A program which simulates IT system and attack on that system is also part of this paper. Different strategies which choose attackers action to execute were also made. Based on a series of tests, attacker which randomly chooses his action performed the worst, and the attacker had better results when the employees in the system performed their actions with different probability. At the end possible improvements directions were given.

Keywords: Cyber attack emulation, automated attackers