Modeliranje kibernetičkih napada u programu Cyber Conflict Simulator korištenjem umjetne inteligencije

Gorup, Andrija

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 376

MODELIRANJE KIBERNETIČKIH NAPADA U PROGRAMU CYBER CONFLICT SIMULATOR KORIŠTENJEM UMJETNE INTELIGENCIJE

Andrija Gorup

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 376

MODELIRANJE KIBERNETIČKIH NAPADA U PROGRAMU CYBER CONFLICT SIMULATOR KORIŠTENJEM UMJETNE INTELIGENCIJE

Andrija Gorup

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 4. ožujka 2024.

DIPLOMSKI ZADATAK br. 376

Pristupnik:	**Andrija Gorup (0036522313)**
Studij:	Računarstvo
Profil:	Računarska znanost
Mentor:	izv. prof. dr. sc. Dario Bojanjac

Zadatak: **Modeliranje kibernetičkih napada u programu Cyber Conflict Simulator korištenjem umjetne inteligencije**

Opis zadatka:

U današnjem svijetu, povezanost na Internet je sveprisutna, a samim time i prijetnja od kibernetičkih napada. Stoga program Cyber Conflict Simulator (CCS), koji je razvila firma Utilis d.o.o, omogućava klijentima pripremu za razne kibernetičke prijetnje prije negoli se one ostvare. Međutim, jedna stavka koju CCS trenutno ne podržava je provođenje realističnih kibernetičkih napada korištenjem umjetne inteligencije. Stoga se u sklopu ovog diplomskog rada proučava mogućnost dodavanja navedene funkcionalnosti. U sklopu ovog diplomskog rada proučava se mogućnost dodavanja umjetne inteligencije za oponašanje kibernetičkih napada u program CCS. Kao referenca za akcije koje koriste hakerske grupe u stvarnim napadima koristit će se MITRE ATT&CK baza znanja. U sklopu rada provodit će se mapiranje radnji trenutno dostupnih unutar CCS-a na tehnike korištene u hakerskim napadima. Razmatrat će se i različiti pristupi umjetne inteligencije te formulacija problema za strojno učenje kako bi se ustvrdio optimalan pristup rješavanju problema.

Rok za predaju rada: 28. lipnja 2024.

Zahvaljujem se mojoj obitelji, mentoru i ekipi iz Utilisa na podršci

SADRŽAJ

1. Uvod **1**

1.1. Definicija kibernetičkog napada 2

2. Simulator kibernetičkih napada (CCS) **3**

3. Lanac kibernetičkog napada (CKC) **6**

4. Mapiranje CCS napadačkih akcija na MITRE ATT&CK tehnike **9**

4.1. Analiza mapiranja akcija 19

5. Primjer CKC-a u CCS-u **21**

6. Općenito o umjetnoj inteligenciji **24**

6.1. Strojno učenje 26

6.1.1. Nadzirano učenje 26

6.1.2. Nenadzirano učenje 28

6.1.3. Polunadzirano učenje 28

6.1.4. Podržano učenje 28

6.2. Duboko učenje 30

6.2.1. Neuronske mreže 31

7. Umjetna inteligencija za provođenje kibernetičkih napada u CCS-u **33**

7.1. Pregled mogućih pristupa 33

7.2. Podržano učenje u CCS-u 34

7.2.1. Q-učenje 36

7.2.2. Algoritam aktera i kritičara (A2C) 37

7.2.3. Proksimalna optimizacija strategije (PPO) 38

7.2.4. Implementacija 38

7.3. Heurističke metode 40

7.4. Ostali algoritmi 41

8. Rezultati **43**

9. Zaključak **45**

Literatura **46**

1. Uvod

U stoljeću obilježenom ubrzanim razvojem tehnologije i sveprisutnom digitalizacijom, kibernetička sigurnost igra sve veću ulogu u životima pojedinaca i organizacijskim poslovnim procesima. Međutim, unatoč povećanom broju kibernetičkih prijetnji, mnogi nisu spremni suočiti se s kibernetičkim napadom.

Istraživanja redovno ukazuju na konstantno povećanje broja kibernetičkih napada [1]. Dodatno, poznata je i činjenica da u kibernetičkim napadima, uz same napadače, sudjeluju i žrtve spomenutih napada. Nepromišljene akcije pojedinaca, ili manjak preventivnih akcija, često omogućavaju hakerima provođenje napada koji u suprotnom ne bi uvijek bili izvedivi [2]. Takva nespremnost na kibernetičke napade učestalo proizlazi iz manjka edukacije o računalnoj sigurnosti [3].

Ukoliko se na meti napada nađe pojedinac koji je dio šireg računalnog ekosustava, na primjer u ulozi zaposlenika firme, vršitelju napada se potencijalno otvaraju vrata čitavog tog sustava. Istraživanje firme Netwrix [4] pokazuje da su vodeći uzrok sigurnosnih incidenata phishing napadi koji sami po sebi ciljaju pojedince. Isto istraživanje pokazuje da IT stručnjaci stavljaju vlastite zaposlenike na prvo mjesto potencijalnih sigurnosnih rizika, čak iznad vanjskih aktera kao što su hakeri. Firmama je stoga jedan od glavnih sigurnosnih prioriteta edukacija vlastitih zaposlenika o potencijalnim kibernetičkim prijetnjama i dizanje kolektivne svijesti o sigurnosnim propustima.

Simulator kibernetičkih napada (engl. *Cyber Conflict Simulator*, **CCS**) je alat koji je razvila firma Utilis d.o.o u suradnji s Fakultetom elektrotehnike i računarstva (FER) koji pokušava riješiti upravo ovaj problem. CCS omogućuje provođenje organiziranih vježbi na razini firme koje pomažu odgovornim osobama razviti intuiciju o kibernetičkoj sigurnosti i upoznaju ih s najčešće korištenim vrstama napada kojima se hakeri koriste kako bi stekli željene podatke. CCS je opširnije opisan u zasebnom poglavlju.

Provođenje vježbe koristeći CCS trenutno je vremenski intenzivan proces. Glavni razlog tome je potreba za pobližom suradnjom između stranke zainteresirane za vježbu i tima stručnjaka kako bi se unutar simulacije stvorio kibernetički prostor koji što bliže odgovara stvarnoj mrežnoj topologiji zainteresirane stranke. Dodatno, potrebno je

osmisliti relevantan kibernetički napad. Određene hakerske grupe preferiraju provoditi napade na određene vrste žrtava. Na primjer, neke hakerske grupe ciljano napadaju organizacije s ciljem ostvarivanja financijske dobiti, neke grupe su u sklopu državne institucije specijalizirane za provođenje kibernetičkog ratovanja, dok neki aktivisti provode napade s ciljem promoviranja vlastitih političkih stavova [5]. Stoga, kako bi se postigla realnost simulacije, posebna pažnja se pridaje upravo osmišljavanju ispravnog napada. Dodatno, nakon što je napad osmišljen, potrebno je izvršavati akcije u ulozi napadača. Trenutno ovu ulogu izvršavaju stručnjaci za CCS. Samim time, vremensko opterećenje ovog koraka razvoja simulacije je veliko. Zato se u sklopu ovog rada proučava mogućnost i isplativost uvođenja umjetne inteligencije za određivanje i provođenje kibernetičkih napada u CCS-u.

1.1. Definicija kibernetičkog napada

Za početak, treba uspostaviti definiciju pojma *kibernetički napad*. Kao što je navedeno u [6], teško je uspostaviti sažetu definiciju kibernetičkog napada. Umjesto toga, bolja metoda je definirati određene atribute koji su zajednički većini kibernetičkih napada. U navedenom radu izdvojeno je pet konkretnih atributa koji definiraju kibernetički napad: sudionici napada, ciljana imovina, motivacija, učinak na ciljanu imovinu te trajanje napada. Od navedenih atributa, trajanje napada se rjeđe inkorporira u definicije kibernetičkog napada. Stoga će se u sklopu ovog rada koristiti sljedeća jednostavna definicija kibernetičkog napada:
Kibernetički napad je

- bilo koja radnja u kibernetičkom prostoru
- u koju su uključena barem dva sudionika (vršitelj i žrtva napada)
- sa definiranom motivacijom iza napada
- i štetnim posljedicama za žrtvu napada

Pritom je kibernetički prostor generalni pojam koji se odnosi na radnje provedene korištenjem računala ili računalnih mreža.

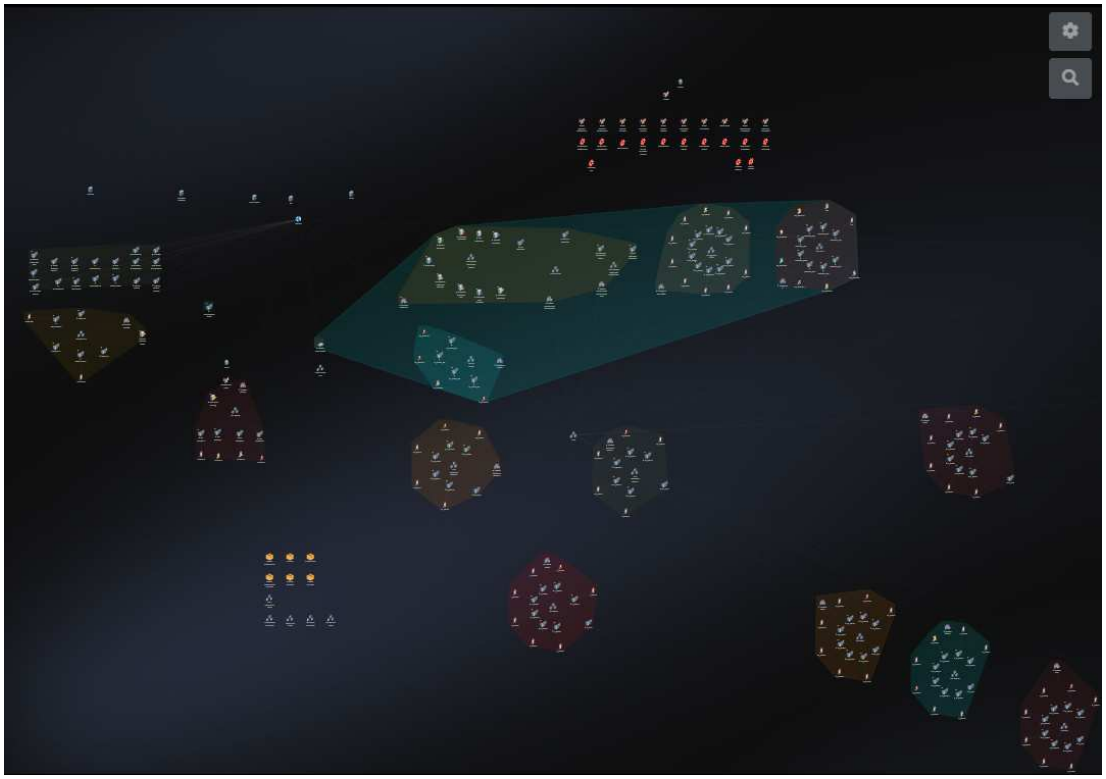
2. Simulator kibernetičkih napada (CCS)

Simulator kibernetičkih napada (engl. *Cyber Conflict Simulator*, **CCS**) je, kao što je spomenuto u uvodu, alat koji služi za provođenje realističnih vježbi odgovora na kibernetički incident [7]. CCS omogućuje sastavljanje detaljnih replika kibernetičkih prostora unutar simulacije kako bi se vježbe bolje mogle prilagoditi pojedinim organizacijama. Tako se na primjer mogu definirati fizički uređaji (računala, serveri), mrežne konfiguracije, postavljenje softvera na pojedine uređaje, kao i sami akteri koji sudjeluju u napadu. Razne kontrole, kao što su antivirusni sustavi i vatrozidi, stvaraju dodatnu komponentu realizma.

Unutar svake simulacije postoje branitelji i napadači. Braniteljsku ulogu igraju same organizacije, dok napadače (hakere) glume stručnjaci za sam alat. Nakon što je modeliran kibernetički prostor simulacije, simulacija može započeti. Na početku simulacije niti jedna stranka nije svjesna one druge. Ova svijest se postepeno gradi kroz niz napadačkih i obrambenih akcija koje su dostupne unutar simulatora. Ovisno o pojedinoj vježbi, konačni cilj napadačima, a samim time i braniteljima, neće biti uvijek isti. Ovo proizlazi iz činjenice da nisu sve organizacije izložene istim sigurnosnim rizicima. Dodatno, hakerske grupe se često specijaliziraju za određenu vrstu napada te ciljaju određenu vrstu organizacija. Stoga je potrebno omogućiti modeliranje različitih vrsta napada.

Simulacije su personalizirane za pojedine organizacije. Za početak, potrebno je izgraditi odgovarajući kibernetički prostor. Unutar CCS-a, izrada kibernetičkog prostora podrazumijeva postavljanje računala i organizaciju navedenih računala u pojedine grupe koje lokacijom i mrežnim postavkama odgovaraju stvarnoj mrežnoj topologiji navedene organizacije. Računala se definiraju u dubinu te se na njih postavljaju određeni operacijski sustavi, programi i podaci. Uz to, dodaju se i organizacijski akteri koji odgovaraju zaposlenicima organizacije. Akterima se mogu postaviti određene vještine, kao što su računalna forenzika te analiziranje logova. Nakon organizacije, potrebno je

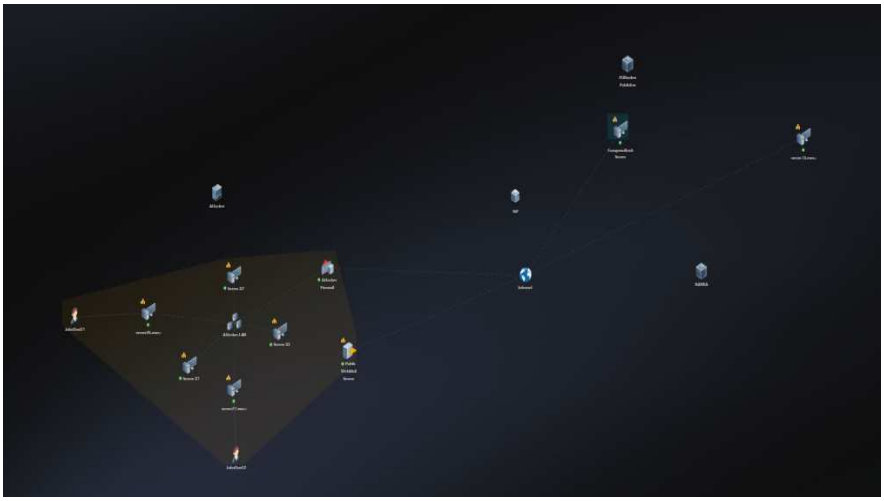
definirati sve komponente za napadača. Postavke kibernetičkog prostora sadrže puno više detalja nego što je ovdje opisano, no u sklopu ovog rada dodatni detalji se neće razmatrati. Na slici 2.1 je prikazan primjer kibernetičkog prostora za generičku banku.



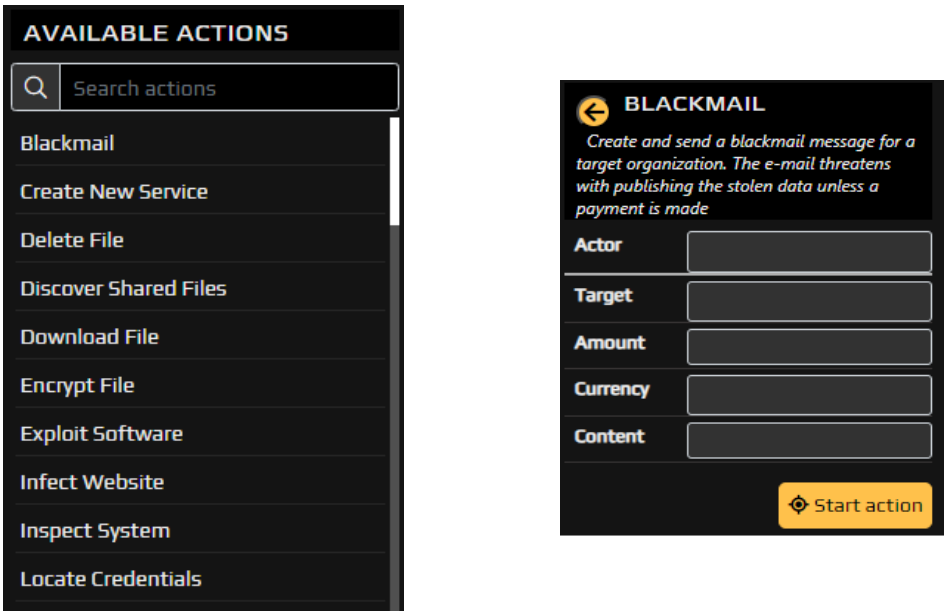
Slika 2.1: Kibernetički prostor za generičku banku

Na početku simulacije, organizacija i napadač nisu svjesni cijelog opsega druge stranke. Ovo je realna pretpostavka za stvarne hakerske napade. Stoga simulacija napada učestalo započinje s napadačem koji pokušava dobiti više informacija o organizaciji koju napada. Napadač raznim akcijama dobiva nove spoznaje o žrtvi koje zatim postaju vidljive u simulatoru. Slika 2.2 prikazuje sve što je napadaču vidljivo na početku jedne simulacije.

Glavni pokretač simulacije su razne akcije dostupne napadaču i žrtvi. Unutar korisničkog sučelja dostupan je izbornik sa svim mogućim akcijama. U određenom vremenskom trenutku nisu dostupne sve akcije. Na primjer, napadač ne može nekoj osobi poslati e-poštu dok nije svjestan njihove e-adrese. Na figuri 2.3 prikazan je izbornik akcija te primjer mogućih parametara za jednu od akcija. U daljnjim poglavljima su detaljnije opisane napadačke akcije u CCS-u te je dan primjer tijeka jednog kibernetičkog napada provedenog unutar CCS-a.



Slika 2.2: Opseg napadačevog znanja na početku simulacije u prostoru generičke banke



(a) Izbornik akcija za napadača **(b)** Primjer jedne napadačke akcije

Slika 2.3: Akcije u CCS-u

Budući da je cilj simulacije podučavanje o kibernetičkoj sigurnosti i prevenciji kibernetičkih prijetnji, simulacije generalno nemaju strogo definirano konačno stanje. Dok su u igrama konačna stanja jasna, u edukativnim simulacijama to nije uvijek slučaj. Ako je napadaču cilj prodati ukradene podatke, simulacija neće završiti nakon što napadač izvrši akciju *Prodaj podatke*, već se simulacija može vratiti nekoliko koraka unatrag kako bi se obrambenoj strani objasnilo koje su korake mogli poduzeti kako bi se spriječilo ostvarenje napadačevog cilja. Stoga su trajanje i tijek svake vježbe prilagođeni upravo za potrebe te vježbe.

3. Lanac kibernetičkog napada (CKC)

Hakerski napadi u stvarnosti često prate određeni redoslijed akcija koji započinje prikupljanjem podataka o žrtvi napada i ostvarivanjem inicijalnog pristupa ciljanom računalnom sustavu, a završava ostvarivanjem željenog cilja. Taj redoslijed akcija se zove lanac kibernetičkog napada (engl. *Cyber Kill Chain*, **CKC**). CKC pobliže opisuje tok standardnog kibernetičkog napada te ga raspoređuje u određene faze. Ove faze nisu nužno strogo definirane. Dok neki izvori dijele napade u 8 faza [8], u ovom radu će se koristiti definicija koju je uspostavila organizacija MITRE. MITRE dijeli napade u 14 faza koje su definirane u MITRE ATT&CK bazi podataka [9].

MITRE ATT&CK jedna je od najpoznatijih i najčešće korištenih baza podataka iz područja kibernetičke sigurnosti. Baza sadrži podatke o napadačkim i obrambenim taktikama i tehnikama, poznatim hakerskim grupama, zloćudnim programima te poznatim hakerskim napadima. Međutim, u sklopu ovog diplomskog rada, najviše će se koristiti podaci o napadačkim taktikama i tehnikama.

Kako bismo bolje razumjeli ATT&CK matricu, potrebno je pojasniti značenje pojedinih pojmova. Počnimo stoga od **taktika**. U sklopu ATT&CK matrice, taktike predstavljaju pojedine faze CKC-a. Prema MITRE-u, CKC se sastoji od 14 taktika (faza). Te taktike uključuju:

Izviđanje (engl. *Reconnaissance*) Napadač prikuplja informacije potrebne za buduće operacije. Ove se ne odnosi na tehničke informacije, kao što su postavke određenog računala ili mrežne postavke, već na osobne informacije o ciljanoj organizaciji, njezinim zaposlenicima, nadređenima te infrastrukturi. Ove informacije pomažu napadaču uspostaviti prvotni pristup u ciljanoj organizaciji.

Razvoj vlastitih resursa (engl. *Resource Development*) Napadač razvija resurse koji mu pomažu ostvariti buduće operacije. Ova faza uključuje kupovinu, krađu, ili vlastiti razvoj resursa kao što su infrastruktura ili zloćudni softveri.

Prvotni pristup (engl. *Initial Access*) Napadač prodire u žrtvinu računalnu mrežu.

Koriste se razni napadi i iskorištavanje ranjivosti kako bi se ostvario prvotni pristup mreži. Određene vrste pristupa, kao što su valjane vjerodajnice, mogu napadaču pružiti i dugotrajan pristup mreži.

Izvršavanje napada (engl. *Execution*) Napadač pronalazi način za pokretanje zloćudnog softvera na žrtvinom računalu. Rezultati izvršavanja zloćudnog koda mogu biti raznovrsni te se često povezuju i s ostalim fazama napada. Neki od rezultata zloćudnog koda uključuju krađu podataka i istraživanje mreže.

Zadržavanje (engl. *Persistence*) Napadač koristi različite metode kako bi zadržao pristup mreži.

Povećanje povlastica (engl. *Privilege Escalation*) Napadač pokušava povisiti razinu vlastitih povlastica na mreži ili sustavu. Napadač učestalo započne napad s niskim povlasticama koje pokušava povisiti kako bi si omogućio lakše provođenje daljnjih akcija.

Izbjegavanje obrane (engl. *Defense Evasion*) Napadač pokušava izbjeći detekciju. Ovo se može postići uklanjanjem ili deaktivacijom obrambenih softvera, obfuskacijom i enkripcijom podataka i programa. Napadač dodatno briše tragove koje neizbježno ostavlja prilikom kretanja po mreži.

Pristup vjerodajnicama (engl. *Credential Access*) Napadač pokušava ukrasti vjerodajnice za pristup korisničkim računima. Ispravne vjerodajnice mogu omogućiti napadaču izravan pristup sustavu te im generalno olakšavaju daljnju provedbu napada.

Otkrivanje (engl. *Discovery*) Napadač otkriva razne podatke o čitavom okruženju. Ovo uključuje podatke o sustavu i mreži. U ovu svrhu se često koriste alati ugrađeni u sam operacijski sustav.

Lateralno kretanje (engl. *Lateral Movement*) Napadač se pokušava kretati po mreži. Dok u fazi otkrivanja napadač otkriva podatke o pojedinom sustavu, u ovoj fazi napadač kretanjem po mreži istražuje različite sustave.

Sakupljanje (engl. *Collection*) Napadač skuplja relevantne podatke. U ovoj fazi napadač se koristi raznim tehnikama kako bi prikupio i pripremio podatke za naknadnu krađu podataka.

Upravljanje i kontrola (engl. *Command and Control*) Napadač komunicira s kompromitiranim sustavima kako bi uspostavio kontrolu. Često se oponaša uobičajeni mrežni promet kako bi se izbjegla detekcija.

Eksfiltracija (engl. *Exfiltration*) Napadač krade skupljene podatke. Učestalo uključuje različite metode, kao što su enkripcija i sažimanje podataka, kako bi se izbjegla detekcija tijekom uklanjanja podataka. U ovu svrhu se koriste kanali uspostavljeni u prijašnjoj fazi.

Učinak (engl. *Impact*) Napadač pokušava ostvariti traženi cilj. Nakon što su provedeni svi potrebni prethodni koraci, moguće je ostvariti učinak na različite načine. Ovaj korak ovisi o konkretnom cilju pojedinog napadača, a može uključivati ciljeve kao što su financijska dobit, remećenje rada ili uništavanje ugleda napadnute organizacije, uništavanje podataka i brojne druge.

Za ostvarivanje pojedine taktike, napadaču na raspolaganju stoje određene **tehnike**. Tehnike dakle predstavljaju konkretne akcije koje napadač može poduzeti kako bi ostvario pojedinu fazu napada. Tako, na primjer, za krađu vjerodajnica napadač može koristiti tehnike poput pogađanja lozinki, krađe nezaštićenih lozinki koje su spremljene na zaraženom računalu ili krađe web kolačića. Tehnike se dodatno dijele na podtehnike. Na primjer, phishing tehnika se dalje dijeli na spearphishing sa servisom, spearphishing s privitkom, spearphishing s poveznicom te spearphishing s glasovnom komunikacijom. Podtehnike neće biti dublje obrađene u sklopu ovog rada, no dobro je znati da postoje. U ATT&CK bazi su opisane sveukupno 202 tehnike koje se dalje dijele na 435 podtehnika. Neke tehnike će biti dodatno spomenute u poglavlju **??**.

Poznavanje taktika i tehnika je korisno kako bismo mogli bolje modelirati realistične kibernetičke napade. Ako želimo napraviti simulator koji može modelirati realistične napade, simulator bi trebao podržavati modeliranje pojedinih faza napada (taktika), što bi se trebalo omogućiti provođenjem određenih napadačkih akcija (tehnika). Stoga je u idućem poglavlju provedeno detaljno mapiranje napadačkih akcija dostupnih unutar CCS-a na napadačke tehnike opisane unutar ATT&CK baze.

4. Mapiranje CCS napadačkih akcija na MITRE ATT&CK tehnike

Za ispravno modeliranje realističnih kibernetičkih napada, potrebno je razumjeti koje napadačke akcije u CCS-u odgovaraju kojim MITRE tehnikama. Uz ovo znanje, lakše je slagati određene napadačke akcije u faze napada. CCS se u praksi upravo ovako i koristi. Napadač na početku simulacije nema dovoljno informacija kako bi odmah izvršio napad, već prvo mora prikupiti informacije o svojoj žrtvi. Ovo odgovara upravo prvoj taktici iz ATT&CK matrice. Napad se dalje odvija prateći generalni tijek CKC-a.

Postoje dva glavna razloga zašto bi ovo mapiranje moglo biti korisno:

1. Poznavanje direktnih MITRE ekvivalenta CCS akcija bi moglo olakšati implementaciju napada korištenjem umjetne inteligencije. Ukoliko se ispostavi da više akcija odgovara tehnikama iste faze napada, neke akcije bi se mogle izostaviti radi pojednostavljenja modela. Dodatno, omogućen je detaljan pregled akcija za lakše sastavljanje realnog napada.

2. Može se zaključiti koje tehnike nisu trenutno prisutne u CCS-u. Ovo saznanje može biti od koristi u budućim doradama CCS-a koje se potom mogu fokusirati na dodavanje nezastupljenih tehnika. Također je vidljiv raspored trenutnih akcija po fazama. Potom je moguće fokusirati se na dodavanje akcija za one faze napada koje trenutno imaju najmanje implementiranih tehnika.

Trenutno je u CCS-u implementirano 44 napadačkih akcija. Ove akcije su raznovrsne te nemaju sve direktni pandan u MITRE tehnikama. Međutim, u nastavku je provedeno generalno mapiranje CCS napadačkih akcija na MITRE tehnike. Akcije koje su previše generalne ili zbog nekog drugog razloga nemaju izravno mapiranje na MITRE tehniku, pod *MITRE tehnika* i *MITRE taktika* imaju napisano -. Dodatno, neke akcije pokrivaju više tehnika te su u tom slučaju sve pokrivene tehnike navedene.

Analiziraj snimke zaslona

- Opis: Analiziraju se snimke zaslona dobivene zloćudnim programom. Pokušavaju se otkriti podaci o mreži na koju je spojeno računalo.
- MITRE tehnika: Snimka zaslona (engl. *Screen Capture*)
- MITRE taktika: Sakupljanje

Započni snimati tipkanje

- Opis: Započinje praćenje korisničkog unosa na određenom računalu.
- MITRE tehnika: Praćenje unosa (engl. *Input Capture*)
- MITRE taktika: Pristup vjerodajnicama, Sakupljanje

Pošalji ucjenjivačku poruku

- Opis: Stvori se i pošalje ucjenjivačka poruka određenoj organizaciji. Porukom se prijeti publikacija ukradenih podataka ukoliko žrtva na napravi traženu uplatu.
- MITRE tehnika: Financijska krađa (engl. *Financial Theft*)
- MITRE taktika: Učinak

Pošalji ucjenjivačku poruku s primjerom ukradenih podataka

- Opis: Stvori se i pošalje ucjenjivačka poruka određenoj organizaciji koja sadrži primjer ukradenih podataka. Porukom se prijeti publikacija ukradenih podataka ukoliko žrtva na napravi traženu uplatu.
- MITRE tehnika: Financijska krađa (engl. *Financial Theft*)
- MITRE taktika: Učinak

Izmjena DNS zapisa

- Opis: Promijene se DNS zapisi kako bi se mrežni promet preusmjerio s legitimne web stranice na lažnu aplikaciju koja odgovara istoj domeni.
- MITRE tehnika: Kompromitiranje infrastrukture (engl. *Compromise Infrastructure*)
- MITRE taktika: Razvoj vlastitih resursa

Povezivanje na VPN

- Opis: Određeno računalo se poveže na VPN mrežu.
- MITRE tehnika: Vanjske udaljene usluge (engl. *External Remote Services*)
- MITRE taktika: Prvotni pristup

Probijanje lozinki

- Opis: Otkriva lozinke iz zaštićene datoteke s lozinkama.
- MITRE tehnika: Gruba sila (engl. *Brute Force*)
- MITRE taktika: Pristup vjerodajnicama

Postavi softver kao uslugu

- Opis: Pokreće određeni program kao servis. Ovime se ostvaruje zadržavanje napadača na mreži.
- MITRE tehnika: Stvori ili izmijeni sustavski proces (engl. *Create or Modify System Process*)
- MITRE taktika: Zadržavanje, Povećanje povlastica

Izradi phishing web stranicu

- Opis: Stvori se web stranica za krađu vjerodajnica koja imitira neku legitimnu stranicu.
- MITRE tehnika: Napad obmanom (engl. *Phishing*)
- MITRE taktika: Prvotni pristup

Kreiraj spearphishing mail s programom za iskorištavanje ranjivosti

- Opis: Stvara se ciljana e-pošta koja u privitku uključuje program za iskorištavanje ranjivosti.
- MITRE tehnika: Napad obmanom (engl. *Phishing*), Unutarnji ciljani napad obmanom (engl. *Internal Spearphishing*), Korisničko izvršavanje (engl. *User Execution*)
- MITRE taktika: Prvotni pristup, Lateralno kretanje, Izvršavanje napada

Kreiraj spearphishing e-poštu s poveznicom

- Opis: Stvara se ciljana e-pošta koja u privitku uključuje poveznicu na zloćudnu web stranicu kako bi se korisniku ukrale vjerodajnice.
- MITRE tehnika: Napad obmanom (engl. *Phishing*), Unutarnji ciljani napad obmanom (engl. *Internal Spearphishing*)
- MITRE taktika: Prvotni pristup, Lateralno kretanje

Kreiraj spearphishing mail sa zloćudnim programom

- Opis: Stvara se ciljana e-pošta koja u privitku uključuje zloćudni program.
- MITRE tehnika: Napad obmanom (engl. *Phishing*), Unutarnji ciljani napad obmanom (engl. *Internal Spearphishing*), Korisničko izvršavanje (engl. *User Execution*)
- MITRE taktika: Prvotni pristup, Lateralno kretanje, Izvršavanje napada

Dekriptiraj datoteku

- Opis: Dekriptira se kriptirana datoteka ukoliko je dostupan ključ za dekripciju.
- MITRE tehnika: -
- MITRE taktika: -

Izbriši datoteku

- Opis: Izbriše se određena datoteka.
- MITRE tehnika: Uništavanje podataka (engl. *Data Destruction*), Izmjena podataka (engl. *Data Manipulation*), Uklanjanje indikatora (engl. *Indicator Removal*)
- MITRE taktika: Učinak, Izbjegavanje obrane

Izbriši logove

- Opis: Izbrišu se lokalne kopije logova.
- MITRE tehnika: Uklanjanje indikatora (engl. *Indicator Removal*)
- MITRE taktika: Izbjegavanje obrane

Deaktiviraj kontrolu

- Opis: Onemogući se određena vrsta kontrole na ciljanom računalu.
- MITRE tehnika: -
- MITRE taktika: -

Isključi VPN

- Opis: Određeno računalo se odjavi s VPN mreže.
- MITRE tehnika: Vanjske udaljene usluge (engl. *External Remote Services*)
- MITRE taktika: Prvotni pristup

Otkrij dijeljene datoteke

- Opis: Pronađu se dijeljene datoteke koje su izložene na udaljenom računalu.
- MITRE tehnika: Otkrivanje datoteka i direktorija (engl. *File and Directory Discovery*), Otkrivanje izvora podataka dijeljenih preko mreže (engl. *Network Share Discovery*), Otkrivanje programa (engl. *Software Discovery*)
- MITRE taktika: Otkrivanje

Preuzmi datoteku

- Opis: Preuzme se određena datoteka s udaljenog računala.
- MITRE tehnika: Eksfiltracija preko alternativnog protokola (engl. *Exfiltration Over Alternative Protocol*)
- MITRE taktika: Eksfiltracija

Ispusti predmet

- Opis: Akter ispusti predmet koju trenutno nosi. Koristi se, na primjer, kod premještanja računala s jedne fizičke lokacije na drugu.
- MITRE tehnika: -
- MITRE taktika: -

Kriptiraj datoteku

- Opis: Stvara se kriptografski ključ te se iskoristi za kriptiranje određene datoteke.
- MITRE tehnika: Kriptiranje podataka radi učinka (engl. *Data Encrypted for Impact*), Obfuskacija podataka (engl. *Data Obfuscation*), Kriptiranje podataka (engl. *Data Encoding*), Obfuskacija datoteka ili informacija (engl. *Obfuscated Files or Information*)
- MITRE taktika: Izbjegavanje obrane, Upravljanje i kontrola, Učinak

Iskoristi ranjivost softvera

- Opis: Pokuša se iskoristiti ranjivost u određenom programu.
- MITRE tehnika: Iskorištavanje ranjivosti radi pokretanja koda (engl. *Exploitation for Client Execution*), Iskorištavanje ranjivosti radi pribavljanja vjerodajnica (engl. *Exploitation for Credential Access*), Iskorištavanje ranjivosti radi izbjegavanja obrane (engl. *Exploitation for Defense Evasion*), Iskorištavanje

ranjivosti radi povećanja povlastica (engl. *Exploitation for Privilege Escalation*)

- MITRE taktika: Izvršavanje napada, Povećanje povlastica, Izbjegavanje obrane, Pristup vjerodajnicama

Zarazi web stranicu

- Opis: Određena web stranica se zarazi zloćudnim programom ili ranjivošću kako bi se izvršio *watering hole* napad.
- MITRE tehnika: Pribavljanje infrastrukture (engl. *Acquire Infrastructure*), Usputna zaraza (engl. *Drive-by Compromise*)
- MITRE taktika: Razvoj vlastitih resursa, Prvotni pristup

Istraži datotečni sustav

- Opis: Izvrši se detaljna analiza sustava kako bi se otkrili instalirani programi, mreže na koje je računalo spojeno, kontrole i pokrenuti procesi.
- MITRE tehnika: Otkrivanje programa (engl. *Software Discovery*), Otkrivanje podataka o sustavu (engl. *System Information Discovery*), Otkrivanje lokacije (engl. *System Location Discovery*), Otkrivanje mrežnih postavki (engl. *System Network Configuration Discovery*), Otkrivanje mrežnih spojeva (engl. *System Network Connections Discovery*), Otkrivanje usluga (engl. *System Service Discovery*), Otkrivanje vremenskih postavki (engl. *System Time Discovery*)
- MITRE taktika: Otkrivanje

Pronađi vjerodajnice

- Opis: Pokušaju se pronaći vjerodajnice koje su spremljene na određenom računalu.
- MITRE tehnika: Dobavljanje vjerodajnica iz spremnika lozinki (engl. *Credentials From Password Stores*), Dobavljanje nezaštićenih vjerodajnica (engl. *Unsecured Credentials*)
- MITRE taktika: Pristup vjerodajnicama

Lokalna prijava

- Opis: Akter se prijavi na računalo koje se nalazi na istoj fizičkoj lokaciji kao i sam akter.

- MITRE tehnika: Iskorištavanje valjanih korisničkih računa (engl. *Valid Accounts*)
- MITRE taktika: Prvotni pristup, Zadržavanje, Povećanje povlastica, Izbjegavanje obrane

Udaljena prijava

- Opis: Akter se prijavi na udaljeno računalo.
- MITRE tehnika: Iskorištavanje valjanih korisničkih računa (engl. *Valid Accounts*)
- MITRE taktika: Prvotni pristup, Zadržavanje, Povećanje povlastica, Izbjegavanje obrane

Odjava

- Opis: Akter se odjavi iz određenog računala.
- MITRE tehnika: Iskorištavanje valjanih korisničkih računa (engl. *Valid Accounts*)
- MITRE taktika: Prvotni pristup, Zadržavanje, Povećanje povlastica, Izbjegavanje obrane

Modificiraj podatke

- Opis: Podaci se modificiraju kako bi se izvršio određeni cilj. Može se koristiti za stvaranje vlastite koristi ili nanošenje štete ciljanoj organizaciji.
- MITRE tehnika: Izmjena podataka (engl. *Data Manipulation*)
- MITRE taktika: Učinak

Skeniranje mreže

- Opis: Izvrši se detaljna analiza mreže kako bi se otkrila računala koja su spojena na mrežu te mrežni servisi.
- MITRE tehnika: Dobavljanje informacija o žrtvinoj mreži (engl. *Gather Victim Network Information*), Otkrivanje mrežnih usluga (engl. *Network Service Discovery*), Otkrivanje izvora podataka dijeljenih preko mreže (engl. *Network Share Discovery*)
- MITRE taktika: Izviđanje, Otkrivanje

Pokreni napad ucjenjivačkim zloćudnim programom na datoteku

- Opis: Izvrši se napad ucjenjivačkim zloćudnim programom na određenu datoteku/datoteke.
- MITRE tehnika: Financijska krađa (engl. *Financial Theft*)
- MITRE taktika: Učinak

Pokreni napad ucjenjivačkim zloćudnim programom na datotečni sustav

- Opis: Izvrši se napad ucjenjivačkim zloćudnim programom nad čitavim računalom.
- MITRE tehnika: Financijska krađa (engl. *Financial Theft*)
- MITRE taktika: Učinak

Objavi podatke

- Opis: Javno se objave ukradeni podaci kako bi se nanijela šteta organizaciji od koje su podaci ukradeni.
- MITRE tehnika: Financijska krađa (engl. *Financial Theft*)
- MITRE taktika: Učinak

Izviđanje

- Opis: Izvrši se detaljno izviđanje kako bi se otkrili podaci o ciljanoj organizaciji ili osobi.
- MITRE tehnika: Dobavljanje informacija o žrtvinom poslužitelju (engl. *Gather Victim Host Information*), Dobavljanje informacija o žrtvinom identitetu (engl. *Gather Victim Identity Information*), Dobavljanje informacija o žrtvinoj organizaciji (engl. *Gather Victim Org Information*)
- MITRE taktika: Izviđanje

Premjesti aktera

- Opis: Akter se premjesti s jedne fizičke lokacije na drugu.
- MITRE tehnika: -
- MITRE taktika: -

Prodaj podatke

- Opis: Prodaju se ukradeni podaci na crnom tržištu.
- MITRE tehnika: Financijska krađa (engl. *Financial Theft*)
- MITRE taktika: Učinak

Pošalji e-poštu

- Opis: Pošalje se unaprijed pripremljena e-pošta.
- MITRE tehnika: -
- MITRE taktika: -

Postavi SCADA vrijednost

- Opis: Postavi se nova vrijednost SCADA parametra. SCADA se odnosi na računalne sustave za nadzor, a promjena ovih parametara može dovesti do uzrokovanja štete te može biti konačni cilj napadača.
- MITRE tehnika: Izmjena podataka (engl. *Data Manipulation*)
- MITRE taktika: Učinak

Isključi računalo

- Opis: Isključi se određeno računalo.
- MITRE tehnika: Isključivanje ili resetiranje sustava (engl. *System Shutdown/Reboot*)
- MITRE taktika: Učinak

Pokreni program / uslugu

- Opis: Započne se određeni program ili usluga.
- MITRE tehnika: -
- MITRE taktika: -

Zaustavi uslugu

- Opis: Zaustavi se određena usluga.
- MITRE tehnika: Zaustavljanje usluge (engl. *Service Stop*)
- MITRE taktika: Učinak

Uzmi predmet

- Opis: Akter uzme određenu imovinu te ju privremeno drži kod sebe.
- MITRE tehnika: -
- MITRE taktika: -

Uključi računalo

- Opis: Upali se određeno računalo.
- MITRE tehnika: -
- MITRE taktika: -

Pošalji datoteku

- Opis: Određena datoteka se pošalje na udaljeno računalo.
- MITRE tehnika: Eksfiltracija preko alternativnog protokola (engl. *Exfiltration Over Alternative Protocol*)
- MITRE taktika: Eksfiltracija

4.1. Analiza mapiranja akcija

Nakon provedenog mapiranja akcija, moguće je bolje sagledati u kojoj mjeri su pojedine faze napada zastupljene u CCS-u. Tablica 4.1 prikazuje koliko različitih akcija je zastupljeno za svaku fazu napada. Dodatno, u tablici je prikazan ukupan broj tehnika za pojedinu taktiku opisanih u MITRE ATT&CK bazi te postotak zastupljenosti tehnika u CCS-u.

Taktika	Tehnika u CCS	Ukupno tehnika	Zastupljenost (%)
Izviđanje	4	10	40
Razvoj vlastitih resursa	2	8	25
Prvotni pristup	4	10	40
Izvršavanje napada	2	14	14.3
Zadržavanje	2	20	10
Povećanje povlastica	3	14	21.4
Izbjegavanje napada	4	43	9.3
Pristup vjerodajnicama	5	17	29.4
Otkrivanje	10	32	31.3
Periferno kretanje	1	9	11.1
Sakupljanje	2	17	11.8
Upravljanje i kontrola	2	18	11.1
Eksfiltracija	1	9	11.1
Učinak	6	14	42.9

Tablica 4.1: Rezultati mapiranja CCS akcija na MITRE tehnike

Iz prikazane tablice lagano je zaključiti koje faze napada su bolje, odnosno lošije zastupljene u CCS-u. Tako su na primjer faze *Zadržavanje* i *Izbjegavanje napada* proporcionalno najlošije zastupljene, dok su faze *Periferno kretanje* i *Eksfiltracija* najlošije ako se gleda ukupni broj zastupljenih tehnika. Ukoliko se ubuduće budu dodavale nove akcije u CCS, ova opažanja bi mogla pridonijeti odluci o tome koje akcije bi trebalo dodati najprije.

Međutim, jedno opažanje koje je za ovaj rad bitnije od kvantitativnog pregleda akcija po fazama je upravo to da su sve faze u nekoj mjeri zastupljene. Drugim riječima, CCS omogućava modeliranje napada koje prati čak i najstrožu i najdetaljnije razrađenu definiciju CKC-a, a to je upravo MITRE definicija. Međutim, bitno je napomenuti da

ne prate svi kibernetički napadi u praksi MITRE definiciju tijeka napada, nego ova definicija služi kao generalni pregled tijeka napada. Većina napada je donekle jednostavnija te se može opisati i CKC-om s manje od 14 faza.

Dodatna opaska koju je bitno napomenuti je da CCS akcije nemaju nužno izravno preslikavanje na MITRE tehnike, već ponekad obuhvaćaju više tehnika u jednu akciju, a ponekad se jedna MITRE tehnika može opisati kroz više CCS akcija. Samim time, u nekim je situacijama teško donijeti konačnu odluku postoji li izravno mapiranje ili ne. Također, neke CCS akcije, kao što su na primjer *Uključi računalo* ili *Pošalji e-poštu* su preopćenite da bi imale izravno mapiranje na neku MITRE tehniku već se koriste u kombinaciji s drugim akcijama kako bi se postigao željeni učinak. Stoga se gore opisani rezultati ne trebaju razmatrati kao konačna odluka o broju zastupljenih tehnika, već kao generalni pregled navedenog mapiranja.

5. Primjer CKC-a u CCS-u

Ukoliko je poznato koje korake sadržava jedan realan napad unutar simulacije, jednostavnije je u konačnici slične napade modelirati umjetnom inteligencijom. Stoga je u ovom poglavlju detaljnije je opisan tijek jednog kibernetičkog napada unutar CCS-a. Napad opisan u ovom poglavlju sastavili su stručnjaci iz firme Utilis d.o.o kao primjer napada koji bi se mogao koristiti u stvarnoj vježbi. Ova simulacija se provodi unutar kibernetičkog prostora koji opisuje generičku banku.

Napadač na početku simulacije, kao što je uobičajeno u napadima, ne zna za pojedinosti ciljane banke, već zna samo da ciljana banka postoji te koji mu je cilj napada. Stoga, kako bi spoznao određene informacije o žrtvi, napadač prvo mora izvršiti akciju istraživanja. Unutar CCS-a, ovo se provodi akcijom *Izviđanje*. Akcija *Izviđanje* napadaču će otkriti osnovne informacije o žrtvi. Neke od informacije koje su ovime otkrivene uključuju: poslovne jedinice unutar banke, usluge koje banka izvršava, zaposlenike banke te fizičke lokacije na kojima određeni zaposlenici rade. Kao što je opisano u poglavlju 4, CCS akcija *Izviđanje* odgovara raznima MITRE tehnikama koje se koriste u sklopu prve faze napada, odnosno faze izviđanja. Međutim, jedna akcija izviđanja nije nužno dovoljna za prijelaz na iduću fazu napada. Stoga s unutar ovog simuliranog napada nakon prve akcije izviđanja izvršavaju dodatne akcije izviđanja. Dok je prva akcija otkrila koji zaposlenici rade u banci, naknadne akcije izviđanja za metu uzimaju pojedine zaposlenike te pokušavaju saznati dodatne informacije o njima. Ovim akcijama saznaje se, na primjer, koje adrese e-pošte pojedini zaposlenici koriste kako bi se mogle te informacije iskoristiti u daljnjim akcijama.

Nakon što je napadač izvršio više uzastopnih akcija izviđanja te dobio sve informacije koje su mu potrebne, može prijeći na iduću fazu napada. Prema MITRE, iduća faza napada je razvoj vlastitih resursa. Ova faza inače podrazumijeva izradu ili nabavu zloćudnih programa, pribavljanje zaraženih web stranica i servera. U ovom primjeru napada, napadač zatim pokušava zaraziti javnu web stranicu. Izvršene akcije izviđanja su napadaču otkrile da ovu web stranicu koriste neki zaposlenici banke. Stoga, ako se uspješno zarazi web stranica, mogao bi se dobiti i pristup računalima zaposlenika

koji tu stranicu posjećuju. Kako bi zarazio stranicu, napadač prvo koristi akciju iskorištavanja ranjivosti programa. Konkretno, iskorištava se WordPress ranjivost navedene stranice. Ovime napadač dobiva pristup serveru na kojemu je web stranica pokrenuta. Nakon toga, napadač na server kojemu je dobio pristup sa svoga računala šalje zlćudni program. Konačno, poslani zlćudni program koristi se za zarazu web stranice. Također, nakon uspješne zaraze web stranice, napadač na poslužitelj web stranici postavlja i dodatne zlćudne programe koji će se kasnije koristiti za izvlačenje podataka od zaposlenika banke.

Sljedeća faza napada je ostvarivanje prvotnog pristupa. U ovom primjeru se ovo ostvaruje upravo zaražavanjem web stranice. Korisnici koji nakon zaraze pristupe web stranici automatski će napadaču omogućiti pristup njihovom računalu. Bitno je napomenuti da je ovaj proces deterministički. Naime, sudionici vježbe ne upravljaju svakog pojedinačnog zaposlenika već njima upravlja simulacija. Samim time, svi akteri koji inače posjećuju ovo web stranicu njoj će automatski pristupiti. Budući da se unutar CCS-a simulira i vrijeme, ako je web stranica zaražena tijekom vikenda u simulaciji, akteri će web stranici pristupiti tijekom prvog sljedećeg radnog dana (u ponedjeljak). Pristup računalima dodatnih korisnika ostvaren je slanjem ciljanih e-pošta obmane zaposlenicima banke. Same poruke unutar e-pošte nisu bitne u CCS-u, ali ove e-pošte dodatno sadrže i zlćudni program koji se pokreće na žrtvinom računalu povodom otvaranja e-pošte. Ovime je ostvaren prvotni pristup na računala zaposlenika banke.

Potom napadač na računala zaposlenika banke preuzima zlćudne programe koje je u prethodnom koraku postavio na zaraženi poslužitelj. Ovime napadač dobiva potpunu kontrolu nad računalima zaposlenika banke.

Nakon ove točke raspored akcija po fazama napada postaje nešto zamućeniji. Naime, kibernetički napadi rijetko savršeno prate MITRE strukturu napada. Ovaj napad nije iznimka pravilu. Konkretno, budući da se radi o jednostavnijem napadu, faze zadržavanja, povećanja povlastica, izbjegavanja obrane, pristupa vjerodajnicama, lateralnog kretanja te sakupljanja nisu izravno implementirane u ovom napadu. Na primjer, izbjegavanje obrane nije implementirano budući da se ovaj napad može pokrenuti bez da itko sudjeluje u obrani te je samim time izbjegavanje obrane bespotrebno.

Umjesto toga, nakon što je dobio pristup žrtvinim računalima, napadač prvo skenira mrežu banke te potom istražuje kojim dijeljenim datotekama može pristupiti sa zaraženih računala. Kada je otkrio potencijalno zanimljive datoteke, potrebno ih je na neki način prebaciti na napadačeva računala kako bi ih dalje mogao koristiti za ostvarivanje profita. Međutim, napadač ne može izravno preuzeti ukradene podatke s računala banke na vlastita računala zbog vatrozida bankovne mreže. Stoga, kako bi

zaobišao vatrozid, napadač prvo šalje ukradene podatke s bankovnih računala na zaraženi poslužitelj te u konačnici preuzima iste podatke na vlastita računala sa zaraženog servera.

U zadnjoj fazi napada, napadač koristi ukradene podatke kako bi ostvario vlastite ciljeve. Ovi ciljevi ovise od napada do napada. Na primjer, ukradeni podaci mogu se koristiti za ucjenjivanje banke kako bi se zatražila otkupnina ili se mogu objaviti na nekoj javno dostupnoj stranici u slučaju da je napadač neka vrsta aktivista. U ovom primjeru, jednostavnosti radi, pretpostavlja se da je iduća akcija prodaja ukradenih podataka na crnom tržištu. Ovime je napad priveden kraju.

Ovdje opisani napad kasnije može poslužiti kao inspiracija za traženo ponašanje implementirane umjetne inteligencije koja je opisana u daljnjim poglavljima.

6. Općenito o umjetnoj inteligenciji

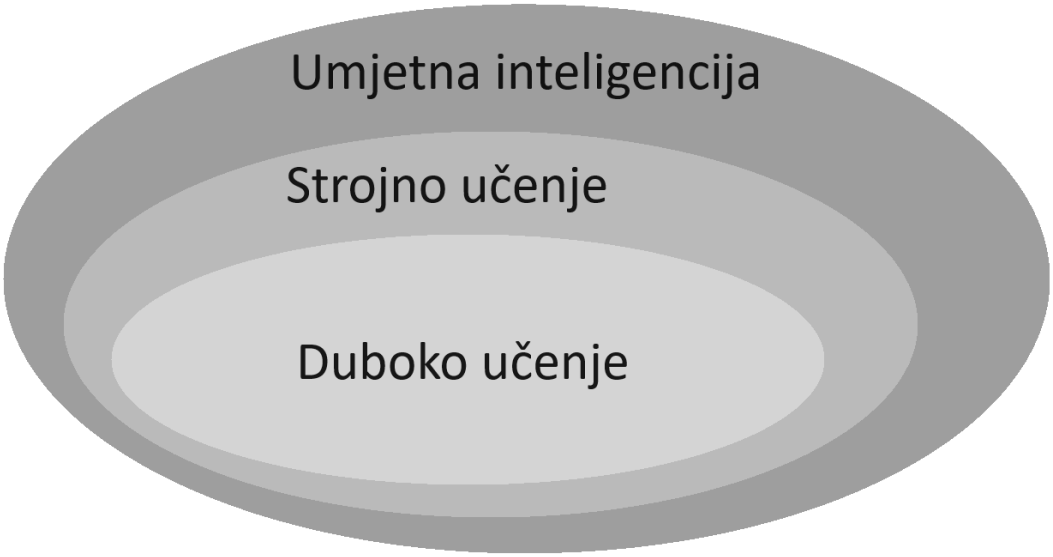
Pojam umjetna inteligencija počinje se koristiti od 1950-ih godina, a u svojem širem značenje označava bilo koji program koji pokušava imitirati ljudsko ponašanje [10] [11]. Jedan od prvih predloženih testova za provjeru umjetne inteligencije je Turingov test u kojemu umjetna inteligencija pokušava uvjeriti ispitivača da je ona zapravo ljudsko biće čime bi dokazala svoju "inteligenciju". Ipak, Turingov test se danas ne smatra sigurnom potvrdom umjetne inteligencije, budući da prevariti ljudskog ispitivača nije nužno ekvivalentno s posjedovanjem inteligencije.

Međutim, umjetna inteligencija te njezine razne manifestacije su se znatno razvile od svojih začetaka pa sve do današnjeg dana. Danas, umjetna inteligencija se pouzdano koristi za rješavanje širokog raspona zadataka. Programi koji bi spadali pod širi pojam umjetne inteligencije danas se koriste u medicini, autonomnim vozilima, računalnom vidu, računalnim igrama i brojnim drugim primjenama. Budući da uporaba umjetne inteligencije iznimno široko rasprostranjena, pod umjetnu inteligenciju spadaju jednostavni zadaci, kao što je igranje igre križić-kružić koje se može opisati skupom pravila, ali i oni složeniji, kao što je prepoznavanje određenih objekata unutar slike. Oba ova primjera pokušavaju imitirati ljudsko ponašanje, ali dok prvi primjer za tu svrhu koristi nekolicinu unaprijed definiranih pravila, drugi primjer bi trebao koristiti nešto kompliciranije strukture, kao što su neuronske mreže.

Dodatan pojam koji treba razjasniti je strojno učenje. Strojno učenje je u zadnje vrijeme sve popularniji pristup rješavanju problema te ga se često zabunom koristi kao sinonim za umjetnu inteligenciju, a ipak ono predstavlja tek jedan podskup šireg polja umjetne inteligencije. Dok umjetna inteligencija podrazumijeva sve programe koji oponašaju umjetnu inteligenciju, strojno učenje je podskup tih programa koji mogu "učiti" na temelju danih primjera [12]. Ovime se pokušava postići ponašanje sličnije ljudima koji uče iz životnih iskustava te na temelju učenja postaju bolji u izvršavanju određenih zadataka.

Unutar strojnog učenja dodatno postoji i podskup zvan duboko učenje. Duboko učenje označava sve programe koji uče na temelju primjera, što je slučaj za sve pro-

grame strojnog učenja, ali pritom za donošenje odluka koriste duboke neuronske mreže. Budući da je za duboko učenje učestalo potrebna velika količina podataka, ovo područje postaje atraktivnije i popularnije u moderno doba zbog dostupnosti podataka putem interneta. Slika 6.1 prikazuje odnos pojmova umjetne inteligencije, strojnog učenja i dubokog učenja.



Slika 6.1: Pregled umjetne inteligencije

Zbog relevantnosti u širem području umjetne inteligencije i u ovom radu, pojmovi strojno učenje i duboko učenje naknadno su razrađeni u sljedećim potpoglavljima.

6.1. Strojno učenje

Strojno učenje označava podskup umjetne inteligencije u koji se svrstavaju algoritmi koji uče na skupu podataka kako bi se riješio neki praktični problem. Intuicija iza strojnog učenja je vrlo očita. Naime, ljudi kroz vježbu i ponavljanje uče kako postati bolji pri rješavanju određenih zadataka. Stoga je nada iza strojnog učenja da, ako se algoritmu daju primjeri nekog fenomena, algoritam na temelju tih primjera može nešto "zaključiti" te samim time postati bolji u rješavanju određenog praktičnog problema. Glavni izvor korišten za strojno učenje u ovom radu je [12].

Strojno učenje dijeli se na četiri glavne vrste:

1. Nadzirano učenje
2. Nenadzirano učenje
3. Polunadzirano učenje
4. Podržano učenje

6.1.1. Nadzirano učenje

Nadzirano učenje koristi se kada je na raspolaganju skup označenih primjera. Označeni primjeri sastoje se od ulaznih značajki, inače označenih znakom x, te oznaka tih značajki, označenih znakom y. Nadzirano učenje vrlo je intuitivno.

Za primjer se uzima model koji predviđa visinu djeteta kada odraste. U ovom primjeru može se koristiti sljedeći skup značajki:

- Spol djeteta
- Trenutna dob djeteta
- Trenutna visina djeteta (cm)
- Visina majke (cm)
- Visina oca (cm)

Naravno, kako bi ovaj model bolje radio, treba biti upoznat sa samom domenom problema što bi omogućilo da u skup značajki uvedu i dodatne značajke koje imaju relevantan utjecan na konačnu visinu djeteta. Model će stoga na ulazu primati navedeni skup značajki, a na izlazu modela očekuje se predviđena visina djeteta kada odraste. Na primjer, ako se radi o muškom djetetu koje trenutno ima 4 godine, visoko je 100

centimetara, majka mu je visoka 165 centimetara, a otac 185 centimetara, ulazne značajke bi u tom slučaju bile $x = (0, 4, 100, 165, 185)$, pod pretpostavkom da je muški spol označen kao 0, a ženski kao 1. Na izlazu modela potom očekujemo odraslu visinu djeteta, na primjer $y = 187$.

Kako bi se ostvarilo ovakvo ponašanje modela, ideja nadziranog učenja je da se prikupi velika količina stvarnih podataka, odnosno parova (x, y), iz kojih će model potom naučiti predviđati ispravne oznake y za dane ulazne značajke x. Način na koji se ovakvo ponašanje postiže zavisan je o samoj vrsti modela, no u konačnici se svodi na minimizaciju pogreške između oznaka koje predviđa model te stvarnih oznaka dostupnih podataka. Jedna od češćih metoda minimizacije pogreške je gradijentni spust kod kojega se parametri modela pomiču u smjeru suprotnom od gradijenta pogreške. Budući da gradijent pokazuje u smjeru najbržeg rasta funkcije, smjer suprotan gradijentu pogreške se kreće u smjeru najbržeg pada pogreške. Samim time, pomicanje parametara modela u tom smjeru dovodi do smanjenja pogreške. Gradijentni spust se pogotovo često koristi kod neuronskih mreža koje su detaljnije objašnjene u poglavlju o dubokom učenju.

Bitno je također napomenuti da se u gore navedenom primjeru s predviđanjem visine djeteta radi o regresijskom problemu. Regresija označava skup problema kod kojih se na izlazu modela dobiva kontinuirana varijabla. U navedenom slučaju to je visina djeteta u centimetrima. Druga vrsta problema zove se klasifikacija. Kod klasifikacije na izlazu modela dobiva se diskretna varijabla koja poprima jednu od vrijednosti iz mogućeg skupa izlaznih vrijednosti. Primjer klasifikacijskog problema bi bilo svrstavanje ljudi na temelju određenih značajki u kategorije $\{zdrav, bolestan\}$.

Za nadzirano učenje koriste se brojni algoritmi, a neki od konkretnih uključuju:

- Linearna regresija (korištena prvenstveno za regresiju)
- Logistička regresija (korištena prvenstveno za klasifikaciju)
- Stabla odluke (korištena prvenstveno za klasifikaciju)
- Stroj potpornih vektora (engl. *Support Vector Machine*, korišten prvenstveno za klasifikaciju)

6.1.2. Nenadzirano učenje

U nenadziranom učenju, za razliku od nadziranog učenja, skup podataka za učenje modela ne sadrži ispravne oznake primjera. Drugim riječima, nije unaprijed poznato što bi model trebao dati na izlazu za primjere iz dostupnog skupa. Stoga se nenadzirano učenje koristi u svrhe kao što su grupiranje primjera i smanjenje dimenzionalnosti.

Primjer algoritma koji se koristi u nenadziranom učenju za provođenje grupiranja primjera je algoritam k-sredina. Algoritam k-sredina na početku odabire k (hiperparametar postavljen unaprijed) srednjih vrijednosti grupa te se primjeri svrstavaju u jednu od grupa na temelju udaljenosti do pojedinih sredina grupa. Kako bi se postiglo bolje grupiranje, treniranje modela uključuje pomicanje sredina grupa čime se dobivaju bolje definirane grupe.

6.1.3. Polunadzirano učenje

Polunadzirano učenje u suštini pokušava postići isto što i nadzirano učenje. Za razliku od nadziranog učenja, u skupu podataka nemaju svi primjeri definirane odgovarajuće oznake. U načelu je broj podataka bez oznaka u skupu podataka znatno veći od broja podataka s oznakama.

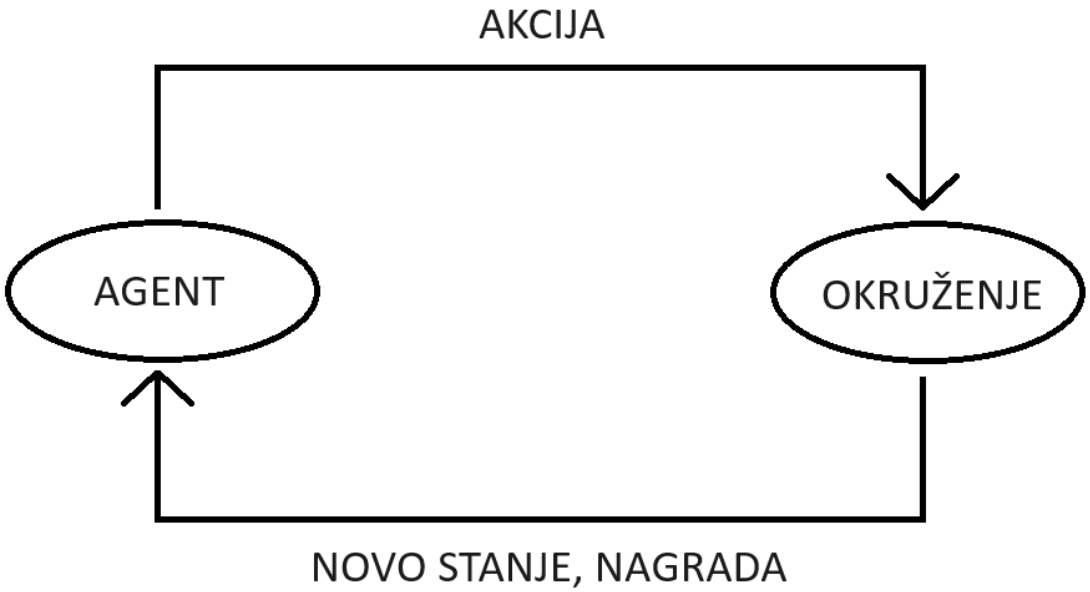
6.1.4. Podržano učenje

Podržano učenje je podosta drukčije od ostalih vrsta strojnog učenja. Kod podržanog učenja ne koristi se skup podataka za treniranje modela, već se modelom pokušava simulirati agent koji se nalazi u određenom okruženju te izvršava najbolje moguće akcije u ovisnosti o trenutnom stanju okruženja.

Dobar primjer za razumijevanje podržanog učenja je treniranje modela za igranje igara. Za primjer se može razmatrati igranje šaha. Ukoliko je cilj naučiti model za igranje šaha koristeći nadzirano učenje, trebalo bi se na raspolaganju imati ogromnu količinu podataka iz prethodnih igara. Naime, u šahu ima previše mogućih stanja ploče da bi se mogao dobiti skup podataka koji bi pokrivao čak i mali podskup mogućih stanja. S druge strane, podržano učenje omogućava alternativu u kojoj nije potrebna golema količina podataka. Kod podržanog učenja, model bi predstavljao agenta, trenutno stanje šahovske ploče bi bilo okruženje u kojemu se agent nalazi, a mogući potezi s obzirom na trenutno stanje ploče bili bi moguće akcije agenta.

Dakle, podržano učenje je u svojoj suštini jednostavno. Ono se sastoji od agenta koji donosi odluke i okruženja koje informira agenta kako bi on mogao donositi odluke.

Međutim, potrebno je definirati i način na koji agent uči donositi ispravne odluke. Za učenje agenta stoga se uvodi i pojam nagrade. Nagrada je zapravo signal koji okruženje šalje agentu nakon svake donesene odluke koji govori koliko je novo stanje okruženja povoljno te se koristi za ažuriranje parametara modela. Stoga je agentov cilj maksimizirati nagradu tijekom cijele sekvence akcija. Također, za razliku od ostalih vrsta strojnog učenja kod kojih se ažuriranje parametara modela može provoditi nakon evaluacije pojedinačnog podatka, kod podržanog učenja to nije slučaj. Naime, kod igranja šaha, igrači nisu nužno svjesni ispravnosti svakog pojedinog poteza u trenutku kada je taj potez izvršen. Umjesto toga, ispravnost svih poteza postaje jasna tek u trenutku kada se igra završi, odnosno kada je jasno je li došlo do pobjede ili poraza. Stoga se kod podržanog učenja ažuriranje parametara modela izvršava tek po završetku čitave sekvence akcija, kada se zna konačan ishod akcija. Slika 6.2 prikazuje osnovnu petlju podržanog učenja.



Slika 6.2: Osnovna petlja podržanog učenja

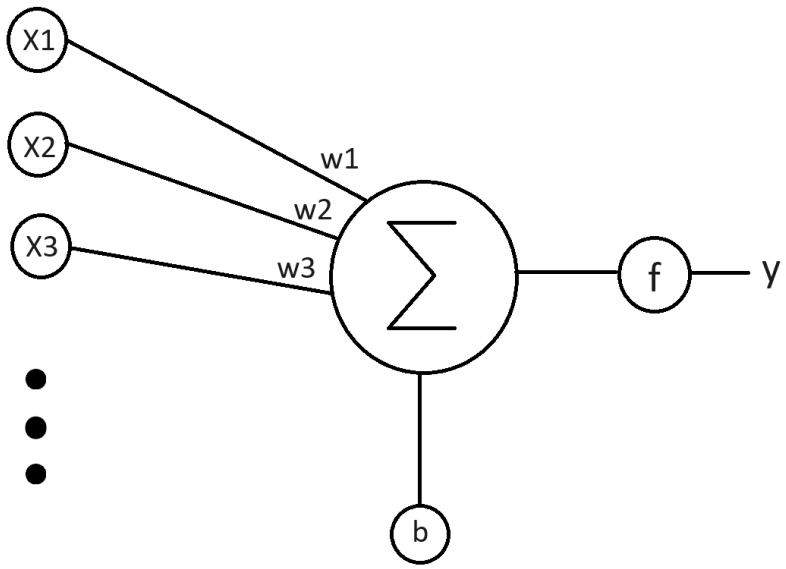
Konkretni algoritmi podržanog učenja se dodatno razrađuju u kasnijim poglavljima, a kao glavni izvor informacija o podržanom učenju u ovom radu koristila se knjiga [13].

6.2. Duboko učenje

Duboko učenje podskup je strojnog učenja u kojemu model zadužen za donošenje rezultata na osnovi ulaznih značajki predstavlja duboka neuronska mreža. Neuronske mreže jedna su od dominantnih tehnologija današnjice na području umjetne inteligencije. Intuicija iza neuronskih mreža potječe od ljudskih neurona. Neuroni su stanice koje sačinjavaju ljudski mozak i živčani sustav [14]. Neuroni primaju električne signale, transformiraju ih te potom propagiraju dalje ostalim neuronima. Stoga je ideja iza neuronskih mreža upravo modeliranje sitnih računalnih jedinica zvanih neurona koji oponašaju ljudske neurone tako što akumuliraju više brojčanih signala, množe ih određenim težinama, transformiraju ih određenom funkcijom te potom novi generirani brojčani signal propagiraju daljnjim neuronima. Također, često se uz ulazne signale dodaje i pomak (engl. *bias*). Ovakav matematički model neurona se zove perceptron. Stoga se izlaz iz perceptrona može opisati formulom:

$$y = f(\sum_{i=1}^{d} x_i \cdot w_i + b)$$

Pri tome d označava dimenzionalnost ulaza, f aktivacijsku funkciju, x ulaze u perceptron, w težine perceptrona, a b pomak. Perceptron je prikazan na slici 6.3.

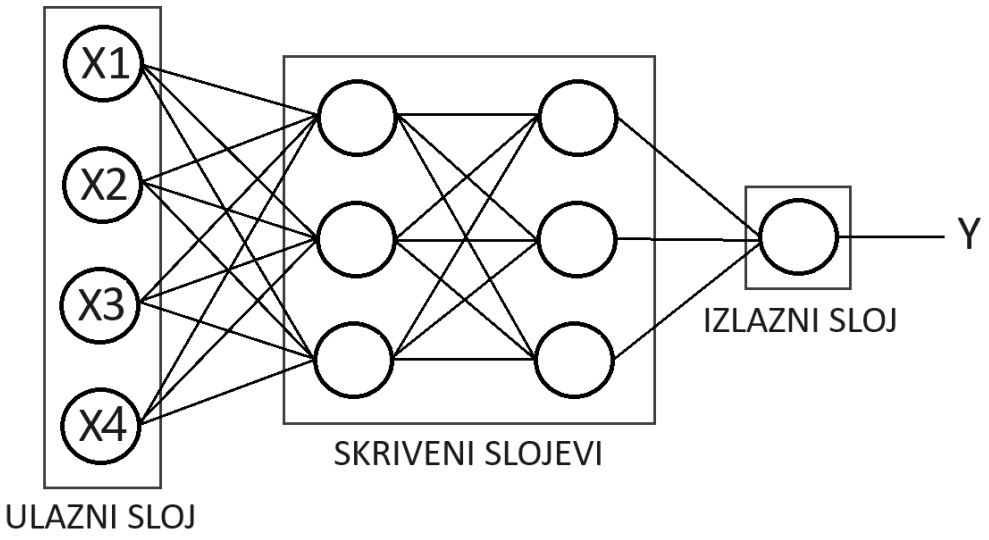


Slika 6.3: Prikaz perceptrona

Glavna referenca za podatke vezane uz duboko učenje u ovom radu je [15].

6.2.1. Neuronske mreže

Jedan perceptron sam po sebi nije sposoban rješavati kompleksne zadatke, no više perceptrona je moguće slagati u slojeve čime se dobivaju neuronske mreže. Neuronske mreže sastoje se od više slojeva neurona, pri čemu su ulazi u neurone jednog sloja upravo izlazi iz neurona prošlog sloja. Na ovaj način postižu se kompleksne transformacije ulaznih podataka koje omogućavaju rješavanje zahtjevnijih zadataka. Pri tome se neuronska mreža sastoji od ulaznog sloja na koji se dovode promatrani podaci, skrivenih slojeva koji služe za ostvarivanje kompleksnijih ovisnosti podataka, te izlaznog sloja na kojemu se generira konačni izlaz za dobivene podatke. Primjer jednostavne neuronske mreže prikazan je na slici 6.4.



Slika 6.4: Primjer neuronske mreže

Kod neuronskih mreža je također bitan i odabir aktivacijskih funkcija pojedinih neurona. Naime, kada se ne bi koristile aktivacijske funkcije na neuronima, oni bi zapravo provodili linearne transformacije. Međutim, ulančavanje više linearnih transformacija je i samo po sebi linearna transformacija. Samim time, kako bismo mogli modelirati kompleksnije nelinearne transformacije, potrebno je uvođenje nelinearnosti kroz različite aktivacijske funkcije. Neke od najčešće korištenih aktivacijskih funkcija su:

- Sigmoidalna funkcija: $f(x) = \frac{1}{1+e^{-x}}$
- Tangens hiperbolni: $f(x) = \frac{2}{1+e^{-2x}}$

- Zglobnica (engl. *Rectified Linear Unit*, ReLu): $f(x) = max(0, x)$
- Propusna zglobnica (engl. *Leaky Rectified Linear Unit*, LReLu): $f(x) = \begin{cases} x, & x \geq 0 \\ \alpha, & \text{u suprotnom} \end{cases}$

Kod neuronskih mreža, težine pojedinih neurona zapravo označavaju ono što je neuronska mreža trenutno naučila. Kako bi se te težine modificirale tijekom učenja koristi se algoritam unatražne propagacije. Postupak unatražne propagacije je zapravo gradijentni spust kod kojega se gradijent pogreške računa u odnosu na izlazni sloj neuronske mreže. Nakon toga se računa gradijent pogreške po parametrima zadnjeg skrivenog sloja mreže koristeći lančano pravilo derivacija te tako sve do ulaznog sloja mreže. Potom se težine pojedinih neurona mogu modificirati tako da ih se pomakne u smjeru suprotnom od izračunatog gradijenta čime se smanjuje pogreška neuronske mreže. Također se uvodi i stopa učenja. Stopa učenja je mala konstanta kojom se množe svi gradijenti prije ažuriranja težina kako bi se postigla stabilnost učenja. Ako se ne bi koristila stopa učenja, promjene parametara bi vjerojatno bile prevelike te bi zbog toga treniranje modela bilo nestabilno.

Kod dubokog učenja koriste se duboke neuronske mreže. Duboke neuronske mreže su one koje imaju više skrivenih slojeva. Korištenje dubokih mreža je korisno zbog dodatne kompleksnosti koja se dobije kroz više slojeva mreže. Međutim, dodavanje više slojeva u neuronsku mrežu također ima i negativne posljedice. Glavna negativna posljedica većeg broja slojeva je otežano treniranje mreže. Ova posljedica proizlazi iz činjenice da će kod dubokih neuronskih mreža funkcija gubitka biti kompleksnija te se unatražna propagacija može ponašati nepredviđeno.

7. Umjetna inteligencija za provođenje kibernetičkih napada u CCS-u

Za dodavanje umjetne inteligencije u CCS, najprije je potrebno detaljno proučiti sve okolnosti. Za početak, potrebno je naglasiti da su u firmi Utilis d.o.o već prije promatrane mogućnosti dodavanja umjetne inteligencije upravo u ovu svrhu. U tom prethodnom razvoju razmatrane su određene heurističke metode, koje su dobro radile na primjeru simulacije napravljene upravo za svrhu testiranja umjetne inteligencije, te algoritmi podržanog učenja koji nikada nisu razvijeni do kraja. Ti prethodno razvijeni algoritmi služila su u jednoj mjeri i kao osnova za ovaj rad. Dodatno, dio koda napisanog za te algoritme se koristio u ovom radu.

7.1. Pregled mogućih pristupa

Što se tiče mogućih pristupa rješavanju ovog problema, prvi pristup koji je razmatran u ovom radu je strojno učenje. Ovaj izbor se činio očitim zbog sve češće uporabe strojnog učenja u velikom broju različitih područja te zbog postojanja djelomično razvijenog prethodnog koda za podržano učenje. Međutim, kako se strojno učenje dijeli na nekoliko vrsta opisanih u poglavlju 6.1, potrebno je odrediti koju vrstu strojnog učenja je najbolje koristiti u ovu svrhu. Nadzirano, nenadzirano i polunadzirano učenje u ovu svrhu nisu primjenjivi. Ovo razmatranje proizlazi iz dvije činjenice:

1. CCS je simulacija u kojoj izvršene akcije imaju utjecaj na buduće akcije unutar iste simulacije. Dodatno, prostor mogućih stanja unutar CCS-a je praktički beskonačno velik. Dok je šahovska ploča ograničena na 64 polja te 16 figurica sa svake strane (čime se i dalje dobiva ogroman broj mogućih stanja), u CCS-u ne postoji ograničenje na maksimalan broj računala koji je moguće dodati u simulaciju, broj različitih mrežnih topografija i raznih drugih postavki. Vremenska

ovisnost akcija te neograničen broj mogućih stanja pokazuju na podržano učenje kao idealna vrsta strojnog učenja za ovaj projekt.

2. Ne postoji skup podataka s dobrim i lošim akcijama u određenim stanjima simulacije. Naime, za protuprimjer može se razmatrati šah koji se igra već tisućama godina, ima velik broj igrača te je na raspolaganju ogroman broj odigranih igara iz kojih se može sastaviti skup podataka sa željenim potezima u određenom stanju ploče. S druge strane, CCS je relativno novi proizvod zatvorene prirode koji se koristi isključivo u organiziranim vježbama. Ova činjenica automatski isključuje mogućnost korištenja nadziranog, nenadziranog ili polunadziranog učenja u ovu svrhu.

Iz ovih razloga je prvi pristup koji je razmatran za ovu svrhu upravo podržano učenje.

Uz podržano učenje, budući da su se heurističke metode pokazale obećavajućima u prethodnom razvoju, u sklopu ovog rada također su promatrani neki heuristički pristupi rješavanju ovog problema.

7.2. Podržano učenje u CCS-u

Samo podržano učenje detaljnije je opisano u poglavlju 6.1.4, dok se u ovom poglavlju razmatra kako se podržano učenje može primijeniti na problem provođenja kibernetičkih napada u CCS-u. Dodatna referenca za informacije u ovom poglavlju je Huggingface tečaj o podržanom učenju [16]. Potrebno je također naglasiti da je CCS web aplikacija te da je sav dodatan kod napisan u sklopu ovog rada odvojen od samog CCS-a s kojim komunicira isključivo preko HTTP poruka.

Za početak, potrebno je poistovjetiti pojmove iz podržanog učenja s elementima ovog problema. Agent je element podržanog učenja zadužen za donošenje odluka na temelju trenutnog stanja. Agenta mogu predstavljati različiti modeli, kao što su neuronske mreže, te agent glumi napadača u simulaciji. Okruženje je u ovom primjeru upravo CCS. Agent akcije šalje CCS-u te mu CCS nakon svake akcije vraća trenutno stanje kibernetičkog prostora kojega je napadač svjestan. Dodatno, agent od okruženja prima i nagradu. Ovo je osnovna struktura rješenja koja bi trebala biti implementirana kako bi se moglo započeti učenje agenta.

Dok su agent i okruženje u ovom problemu prilično dobro definirani, stanje koje okruženje šalje agentu te nagrada koju agent dobiva su znatno teži za definirati. Kako

bi se bolje razumio ovaj problem, najbolje je razmotriti stanje na nekom jednostavnijem primjeru. U ovu svrhu može se razmatrati što bi bilo stanje u igri šaha. Tada je prilično evidentno da bi stanje u igri šaha bilo upravo trenutno stanje šahovske ploče. To bi dakle uključivalo položaje svih trenutno aktivnih figurica na ploči. Ovo stanje daje agentu sve potrebne informacije na temelju kojih agent može donositi odluke. Međutim, stanje u CCS-u nije tako dobro definirano kao što je stanje u šahu. Ukoliko se za modeliranje agenta koriste neuronske mreže, stanje na ulazu bi uvijek trebalo biti fiksne dimenzionalnosti. Ako za stanje želimo koristiti sve informacije koje su dostupne napadaču u nekom trenutku, dimenzionalnost stanja bi se mijenjala budući da se novi dijelovi kibernetičkog prostora otkrivaju napadaču kroz simulaciju korištenjem određenih akcija. To bi značilo da bi na ulazu u neuronsku mrežu na početku simulacije bilo manje podataka nego pred kraj simulacije kada napadač ima više informacija pri ruci. Stoga je potrebno na neki način izvući bitne značajke za određivanje trenutne akcije, te je bitno da se dimenzionalnost tih značajki ne mijenja kroz simulaciju.

Drugi problematičan detalj koji je potrebno implementirati je nagrada za agenta. Naime, kako bi agent naučio odabirati ispravne akcije, on mora znati jesu li na kraju jedne simulacije akcije koje je odabrao bile dobre ili loše. Ako se za primjer opet uzme šah, glavnu pozitivnu nagradu bi agent trebao dobiti kada uspješno pobjedi protivnika. Međutim, nije dobra praksa postaviti isključivo jednu nagradu tijekom cijelog procesa učenja. U igri koja je kompleksna poput šaha, ukoliko počnemo od agenta koji odabire nasumične akcije, vjerojatnost da će taj agent u bilo kojoj igri doći do pobjede nad drugim agentom koji do neke mjere odabire ispravne akcije je jako mala. Na ovaj način, trenirani agent vjerojatno nikada neće dobiti pozitivnu nagradu te samim time nikada neće moći naučiti koje su akcije dobre. Stoga je potrebno dodati i druge nagrade za akcije kao što su uklanjanje neprijateljskih figura s ploče. Ove nagrade dodatno mogu biti otežane ovisno o tome koliko je uklonjena figura korisna. Dodatno se mogu postaviti i negativne nagrade za slučaj kada agent izabere lošu akciju. Međutim, kako je cilj agenta u podržanom učenju maksimizirati ukupnu nagradu dobivenu tijekom jedne epizode učenja, potrebno je pripaziti da se agent ne nauči skupljati nagrade iz nekih izvora u nadi maksimizacije nagrade, a da pritom zanemari glavnu nagradu, to jest onu koja se dobije prilikom pobjede protivnika. Agenti na ovaj način mogu pronaći načine za maksimizaciju nagrade koje programer nije nužno predvidio.

Još jedan detalj koji može biti nejasan je kako definirati prostor akcija. Prostor akcija označava sve moguće akcije koje agent može poduzeti u nekom trenutku te bi, isto kao i prostor stanja, trebao biti fiksne dimenzionalnosti ukoliko se agent modelira neuronskim mrežama. Na primjer, u šahu svaki igrač na početku ima 16 figura od kojih

svaka ima dobro definirane moguće akcije. Iako se broj mogućih akcija može smanjivati tijekom igre, na primjer ako igrač izgubi jednu figuru, gornja granica mogućeg broja akcija je dobro definirana te se može fiksirati. S druge strane, iako CCS u načelu ima svega 44 napadačke akcije, od kojih nisu sve dostupne u svakom trenutku, nije isto hoće li se akcija *Preuzmi datoteku* izvršiti na jednom računalu ili drugom računalu. S takvim pogledom na stvari, CCS u praksi ima neograničeno mnogo akcija budući da kibernetički prostor simulacije nije jednak za sve simulacije te nema definirana gornja granica broja elemenata unutar simulacije.

Moguća rješenja svih ovih problema opisana su u poglavlju 7.2.4 koje pokriva konačnu implementaciju dok je u nastavku dan kratki opis glavnih algoritama podržanog učenja.

7.2.1. Q-učenje

Generalno, algoritmi podržanog učenja dijele se u dvije skupine:

1. Algoritmi zasnovani na strategiji
2. Algoritmi zasnovani na vrijednosti

Strategija je zapravo dio agenta koji je zadužen za donošenje odluka. Algoritmi zasnovani na strategiji su oni koji izravno donose odluku o idućoj akciji na temelju trenutnog stanja. Na primjer, ako se algoritmu da trenutno stanje šahovske ploče, na izlazu će se dobiti iduća odabrana akcija. Drugim riječima, takvi algoritmi na temelju trenutnog stanja znaju samo koja je, prema njima, najbolja iduća akcija. S druge strane, algoritmi zasnovani na vrijednosti računaju procjene vrijednosti stanja te na temelju toga mogu odlučiti koja je iduća najbolja akcija. Takvi algoritmi koriste unaprijed definiranu strategiju koja bira akciju na temelju vrijednosti stanja. Primjer jedne takve strategije je pohlepna strategija koja će uvijek odabrati stanje s najvećom vrijednosti. Dakle, algoritmi zasnovani na vrijednosti prvo računaju vrijednosti sljedećih stanja, a potom unaprijed definiranom strategijom odabiru akciju koja ih vodi u optimalno iduće stanje. Osim vrijednosti pojedinih stanja, u algoritmima zasnovanima na vrijednosti moguće je računati vrijednosti parova stanje-akcija. U ovom pristupu za svaku kombinaciju akcije i stanja računa se vrijednost te se potom unaprijed zadanom strategijom može odabrati optimalna akcija za trenutno stanje.

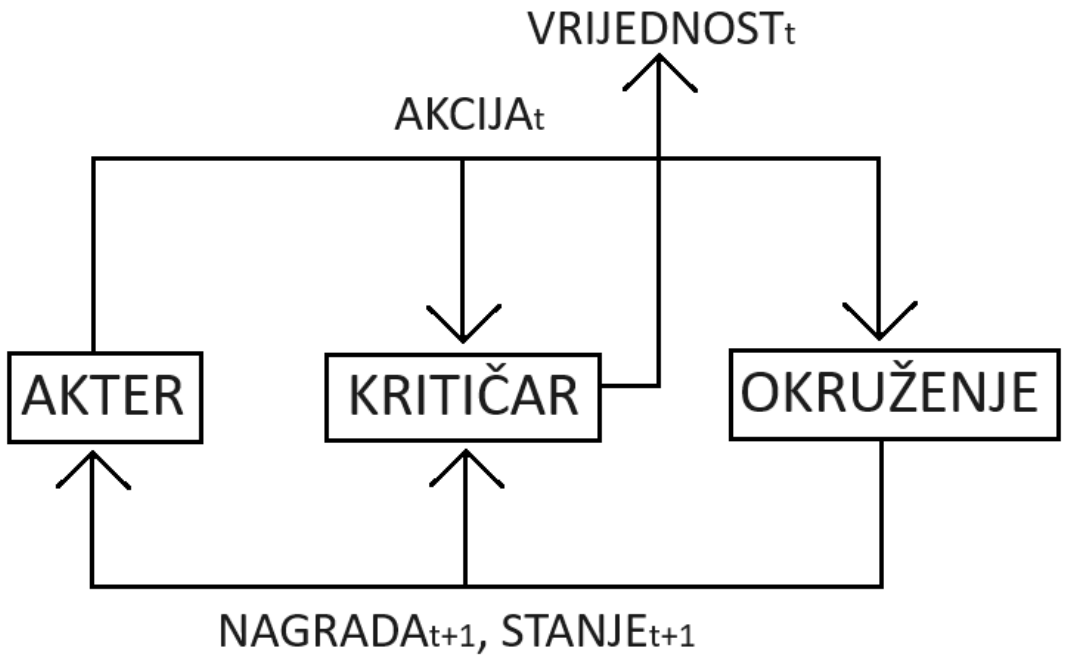
Q-učenje je jedan od pristupa podržanom učenju zasnovanom na vrijednosti. U Q-učenju uči se Q-tablica, tablica u kojoj se pamte sve vrijednosti parova stanje-akcija. Potom se donošenje odluka izvršava pregledavanjem Q-tablice te odabirom

akcije koja za trenutno stanje ima najveću vrijednost. Vrijednosti Q-tablice uče se različitim Monte Carlo metodama.

Kod problema u kojemu su prostori stanja kontinuirani, a ne diskretni, Q-tablicu može zamijeniti neuronska mreža. U tom slučaju, vrijednosti akcija za pojedino stanje dobivaju se izlazu neuronske mreže postavljanjem traženog stanja na ulaz neuronske mreže. Učenje se u ovom slučaju odvija standardnom propagacijom unatrag.

7.2.2. Algoritam aktera i kritičara (A2C)

Algoritam aktera i kritičara (engl. *Advantage Actor-Critic*, **A2C**) je algoritam koji kombinira pristup zasnovan na strategiji te pristup zasnovan na vrijednosti. Ideja iza ovog algoritma je vrlo intuitivna. Postoje dva modela: akter i kritičar. Akter je zadužen za donošenje odluka na temelju trenutnog stanja. Nakon što je akter donio odluku, ona se prosljeđuje kritičaru i okruženju. Kritičar izračuna procjenu vrijednosti donesene akcije s obzirom na trenutno stanje. Kod ažuriranja parametara, akter koristi vrijednost koju je izračunao kritičar. Akter potom prima novo stanje i nagradu od okruženja i donosi novu odluku. Kritičar sada može, na temelju razlike između vrijednosti dvije uzastopne akcije i nagrade dobivene od okruženja, ažurirati svoje parametre. Osnovna arhitektura A2C algoritma prikazana je na slici 7.1.



Slika 7.1: Arhitektura A2C algoritma

7.2.3. Proksimalna optimizacija strategije (PPO)

Proksimalna optimizacija strategije (engl. *Proximal Policy Optimization*, **PPO**) je u suštini isti algoritam kao i A2C algoritam. Jedina razlika je to što se u PPO algoritmu ograničava veličina moguće promjene parametara tijekom učenja kako bi se postigla bolja stabilnost treniranja. Pritom se uzima omjer između nove i stare vjerojatnosti za određenu akciju u nekom stanju te se taj omjer ograničava na interval $[1 - \epsilon, 1 + \epsilon]$. Ovime se osigurava da promjena u strategiji između dvije iteracije ne bude prevelika.

Budući da se PPO pokazao kao dosta pouzdan algoritam [17], upravo je ovaj algoritam odabran za implementaciju podržanog učenja u CCS okružju.

7.2.4. Implementacija

Generalni pseudokod implementacije podržanog učenja u CCS-u izgleda ovako:

```
inicijalizacija (spajanje sa CCS-om)
petlja(broj epizoda):

    petlja(broj koraka):
        dohvati trenutno stanje
        odaberi sljedecu akciju
        posalji CCS-u akciju na izvrsavanje
        pricekaj dok akcija ne zavrsi s izvrsavanjem
        azuriraj interne podatke
        pohrani akciju, stanje, nagradu

    azuriraj tezine modela
```

Što se tiče okruženja, ono je implementirano kao omotač koji enkapsulira svu komunikaciju prema CCS-u. To uključuje dohvaćanje trenutnog stanja, dohvaćanje trenutnih akcija koje su napadaču na raspolaganju, izvršavanje akcija i provjera statusa započete akcije. Dodatno, unutar okruženja su se pamtili dodatni interni podaci koji su se koristili prilikom donošenja odluka o akcijama. Primjer ovih podataka bi bilo na koja računala unutar simulacije se već postavio neki zlоćudni kod kako se ovakve akcije ne bi nepotrebno izvršavale više puta.

Što se tiče prostora akcija, on je definiran kao prostor fiksne dimenzionalnosti. Drugim riječima, model je uvijek na izlazu imao 44 vrijednosti budući da u CCS-u postoje 44 napadačke akcije. Međutim, prilikom dohvaćanja akcija koje su dostupne napadaču

u nekom trenutku, CCS vrati listu svih dostupnih akcija kao i moguće parametre za te akcije. Na primjer, akcija *Pošalji ucjenjivačku poruku* mora sadržavati aktera koji izvršava ucjenu, žrtvu ucjene, sadržaj ucjene te cijenu koju akter traži od žrtve. Kako bi se omogućio odabir ovih parametara pružaju se dvije mogućnosti. Prva, vrlo kompliciranа mogućnost, bi bila stvoriti jedan model strojnog učenja za svaku od akcija koji bi odabrao optimalne parametre. Druga, jednostavnija opcija je ručno osmisliti heuristike na temelju kojih se određuju parametri pojedine akcije. Naime, ukoliko se napadaču pruža opcija da ukrade podatke od 5 različitih korisnika bez da zna ikakve dodatne informacije iz kojih bi mogao unaprijed naslutiti vrijednost podataka pojedinog korisnika, najbolje što može učiniti je nasumično odabrati žrtvu. Jedina opcija gora od nasumičnog odabira bi bio odabir jednog od vlastitih aktera kao žrtve napada. Stoga bi se kod ovakve akcije provodila provjera da žrtva nije akter pod vlasništvom napadača. Slično vrijedi i za ostale napadačke akcije. Dodatno, ovakva heuristika je već bila razmatrana u prijašnjim razvijanjima umjetne inteligencije za CCS te je stoga korištena i u ovom slučaju.

Prostor stanja je jedan od kompliciranijih elemenata u ovom procesu. Napadač u CCS-u ima pristup jako velikoj količini podataka. S druge strane, destilirati taj veliki broj podataka u ključne podatke na temelju kojih se mogu donositi odluke se pokazalo kao jedan od ključnih problema. Stoga ideja koja je korištena u ovom radu vuče inspiraciju iz CKC-a. Ideja je sljedeća: ako su napadaču poznate akcije koje su mu trenutno dostupne, na temelju toga on može pretpostaviti u kojoj fazi napada se nalazi. Samim time, može odrediti koju akciju bi trebao odabrati u trenutnom stanju. Za neke faze napada ovo je trivijalno, ako je napadaču ponuđena akcija *Prodaj podatke*, on zna da se nalazi u završnoj fazi napada te sve što treba napraviti jest iskoristiti tu akciju i u određenom smislu je napad završio. Za ostale faze napada zaključivanje nije tako trivijalno, ali intuicija je i dalje ista. Dakle, model aktera korištenog u ovom radu je neuronska mreža kojoj su i ulaz i izlaz dimenzije 44. Model kritičara na ulazu ima dimenziju 44 dok na izlazu vraća jednu vrijednost (vrijednost akcije u trenutnom stanju).

Zadnji element koji treba razjasniti je nagrada koju okruženje vraća agentu. Ovaj element također se uspostavio kao jedan od kompliciranijih za ispravnu implementaciju. Konačni cilj modela postavljen je tako da simulacija završava kada napadač izvrši akciju *Prodaj podatke* (jednostavnosti radi). U tom trenutku napadač "pobjeđuje". Samim time, ako napadač ikad izvrši navedenu akciju, vraća mu se velika pozitivna nagrada. Međutim, kako bi napadač napredovao i bez da dođe do konačne akcije, potrebne su mu pozitivne nagrade i u nekim međukoracima. Ovdje je opet iskorištena

intuicija CKC-a, da se napadaču vrati manja pozitivna nagrada ukoliko se novo stanje razlikuje od prethodnog, odnosno ako su se napadaču omogućile neke nove akcije. Ovo bi moglo ukazivati na napredovanje po CKC-u te postizanje novih faza napada. U svakom drugom slučaju napadač je dobio malu negativnu nagradu.

Konkretna implementacija koda rađena je po uzoru na [18]. Bitno je također naglasiti i da je prilikom testiranja ovog pristupa napadaču bila onemogućena velika količina akcija. Ovo je napravljeno iz dva razloga:

1. Neke akcije su zastarjele u CCS-u te se ne koriste često u praksi.

2. Jednostavnosti radi, neke akcije su isključene. Na primjer, nakon što napadač ukrade žrtvine podatke, može ih prodati, javno objaviti, ili pokušati ucijeniti žrtvu za financijsku dobit. Koju god akciju napadač odabrao, smatra se da je simulacija završila te da je napadač ostvario svoj cilj. Stoga je od ovih akcija ostavljena jedino akcija prodaje podataka. Slično vrijedi i za druge skupine napadačkih akcija.

Budući da ovakva implementacija nije davala zadovoljavajuće rezultate, što je detaljnije objašnjeno u poglavlju 8, razmatrani su i neki drugi pristupi.

7.3. Heurističke metode

U prijašnjem razvoju uspostavilo se da ručno osmišljene heurističke vrijednosti akcija mogu dobro funkcionirati u kontroliranom kibernetičkom prostoru. Konkretno, metoda koje je bila razvijena prolazila je kroz sve akcije te im dodijelila heurističku vrijednost na temelju njihovih parametara. Nakon toga odabire se akcija s najboljom vrijednošću ili se nasumično odabire akcija iz distribucije u kojoj su težine zapravo vrijednosti akcija. Bitno je naglasiti da se u ovom kontekstu dvije iste napadačke akcije s različitim parametrima smatraju različitim akcijama. Iz tog razloga, u ovom pristupu se javlja problem neograničenog prostora akcija. Ako se razmatra samo jedna napadačka akcija, na primjer akcija *Pošalji ucjenjivačku poruku* kojoj je jedan od parametara žrtva kojoj se šalje poruka, budući da u CCS-u nema ograničenje na broj aktera u simulaciji, ova akcija može imati neograničeno mnogo kombinacija parametara. Zbog toga se u ovom pristupu nude dvije mogućnosti: prihvatiti sporost programa ili ograničiti broj mogućih akcija na neku gornju granicu. U drugoj će se opciji, kod kompleksnih kibernetičkih prostora, najvjerojatnije izgubiti velika količina mogućih akcija. U suprotnom, ukoliko se ne ograniči broj akcija, može doći do kombinatorne eksplozije

zbog načina na koji CCS računa kombinacije parametara te će se program izvršavati vrlo sporo.

Naime, CCS prilikom dohvaćanja mogućih napadačkih akcija vraća listu akcija te za svaku akciju vraća listu nezavisnih parametara i parametarskih kombinacija. Nezavisni parametri su oni koji se mogu pojaviti u bilo kojoj kombinaciji dok su parametarske kombinacije skup parametara koji su zavisni jedan o drugome te se mogu pojaviti isključivo u određenim kombinacijama. Ako se žele dobiti sve moguće kombinacije parametara, potrebno je raditi kartezijev produkt svih nezavisnih parametara te parametarskih kombinacija. Jedan način za rješavanje ovog problema je nasumično uzorkovanje manjeg broja kombinacija parametara za svaku napadačku akciju. Na ovaj način je moguće dobiti jednaku zastupljenost svih napadačkih akcija te istovremeno sačuvati računalnu jednostavnost algoritma. Na primjer, ako se za svaku napadačku akciju nasumično uzorkuje najviše 10 kombinacija parametara, maksimalan broj akcija koje je moguće proslijediti na heurističko ocjenjivanje bi bilo $44 \cdot 10 = 440$. Naravno, ovime bi se izbacila velika količina kombinacija iz razmatranja što nije optimalno.

7.4. Ostali algoritmi

Primjer jednog algoritma koji je isproban u sklopu ovog rada nastoji pobliže modelirati CKC. Modelirana je pojednostavljena verzija CKC-a s 5 faza. Za svaku fazu su izdvojene napadačke akcije koje pripadaju toj fazi. Dodatno, stvorena je lista u kojoj se pamte trenutne vrijednosti za pojedinu fazu. Ove vrijednosti mogu se shvatiti kao vjerojatnosti da je napadač trenutno u pojedinoj fazi. Faze uključuju:

1. Izviđanje
2. Pristup
3. Otkrivanje
4. Eksfiltracija
5. Učinak

Na početku je vjerojatnost faze *Izviđanje* postavljena na 1, dok su sve ostale vjerojatnosti postavljene na 0. To znači da će na početku algoritam sigurno odabrati jednu akciju iz liste akcija za fazu izviđanja. Nakon izvršavanja svake akcije ažuriraju se vjerojatnosti faza. Trenutnoj fazi se vjerojatnost smanji za 0.1. Ako je akcija iz jedne od rubnih faza (*Izviđanje* ili *Učinak*), vjerojatnosti faze koja graniči s rubnom fazom

dodaje se 0.1. Inače, ako se radi o jednoj od unutarnjih faza (*Pristup*, *Otkrivanje* ili *Eksfiltracija*), vjerojatnosti susjednih faza se povećavaju za 0.05. Na ovaj način je zbroj svih vjerojatnosti uvijek jednak 1. Ovaj pristup omogućava kretanje po CKC-u kroz faze. Ukoliko je napadač zapeo na nekoj fazi, povećanjem vjerojatnosti prijašnje faze omogućeno je vraćanje unatrag po CKC-u.

Ovo je osnovna inačica algoritma te je jedina koja je implementirana u sklopu rada. Algoritam je moguće dodatno poboljšati kompleksnijim provjerama i evaluacijom izvršenih akcija. U tom bi se slučaju vjerojatnost iduće faze povećavala jedino ako je akcija uspjela te su se otvorile nove mogućnosti za akcije iz sljedeće faze. U suprotnom bi se povećala isključivo vjerojatnost prijašnje faze.

Dodatni algoritmi u nastavku nisu implementirani u sklopu rada, ali su navedeni kao moguća rješenja problema.

Prijašnji algoritmi uključivali su značajne količine nedeterminizma. Međutim, kibernetički napadi često znaju pratiti prilično konzistentan tijek. Ovo je pogotovo očigledno u CCS-u. Na primjer, ako je napadač poslao e-poštu sa zloćudnim programom žrtvi koja je tu istu e-poštu otvorila i samim time omogućila napadaču pristup svom računalu, logično je da će napadač u sljedećem koraku iskoristiti novostečenu kontrolu nad žrtvinim računalom kako bi napravio inspekciju sustava te proveo skeniranje mreže. Ako nakon toga pronađe podatke koje je moguće prodati, logično je da će ih probati izvući iz žrtvine mreže te preuzeti na vlastito računalo. Stoga se postavlja mogućnost modeliranja umjetne inteligencije temeljene na pravilima. Ovakav pristup omogućio bi izravnu kontrolu stručnjaka prilikom definiranja pravila te bi se izbjegli brojni problemi koji u trenutnom izdanju ometaju ostale navedene algoritme.

Za kraj, korisno je spomenuti i neke algoritme koji se koriste u video igrama za donošenje odluka entiteta kojima ne upravlja igrač. Primjer jednog takvog algoritma je takozvani *Utility AI* koji koristi naprednija evaluacijska stabla za donošenje odluka. Ovakvi algoritmi nisu dalje razrađeni u ovom radu, no korisno ih je spomenuti. Detaljnije o ovakvim algoritmima može se pronaći u sljedećoj knjizi [19].

8. Rezultati

Algoritmi koji su razvijeni u sklopu ovog rada nisu pokazivali zadovoljavajuću sposobnost modeliranja kibernetičkih napada unutar CCS-a. Iako je razvijeni algoritam s podržanim učenjem uspješno ažurirao težine te je na nekim vrlo jednostavnim primjerima bilo očito da se prilagođava zadanom zadatku, postoje drugi problemi koji su spriječili ovaj algoritam da se do kraja nauči. Jedan od glavnih razloga zašto je do ovoga došlo je i sam CCS. Bitno je imati na umu da CCS kao proizvod nije namijenjen za širu javnost već se koristi isključivo u kontroliranim uvjetima pri čemu su na vježbama uvijek prisutni i stručnjaci. Ovime se većinski smanjuje mogućnost otkrivanja nasumičnih pogrešaka u kodu koje proizlaze iz vrlo neočekivanih korisničkih unosa. Ovaj problem se očitovao na način da je nakon velikog broja izvršenih napadačkih akcija unutar CCS-a izazvana iznimka od koje se program ne može oporaviti. Iako je nekolicina ovakvih grešaka ispravljena u sklopu ovog rada, riješiti se svih grešaka bilo bi vrlo zahtjevno. Stoga je velika količina pokušaja treniranja završavala upravo izazivanjem iznimke ili u nekim sličnim nepredviđenim okolnostima. Dodatno, CCS nije razvijen s umjetnom inteligencijom na umu. Primjer toga bi bili svi pozadinski procesi koji se događaju u CCS-u, a nisu korisni prilikom učenja umjetne inteligencije. Ovo uključuje procese kao što su spremanje zapisa o simulaciji, osvježavanje prikaza za sve igrače, povremeno spremanje kopija simulacije i brojni drugi procesi. Ovime se znatno usporava proces učenja umjetne inteligencije.

Jedan od većih problema je način na koji CCS računa sve kombinacije parametara za napadačke akcije. Prilikom razvoja CCS-a ukazao se problem predugog računanja svih mogućih kombinacija parametara. Kako bi se ovaj problem riješio, isključena je provjera ispravnosti nekih kombinacija parametara čime se izbjegava računanje kartezijevog produkta. Ovime se znatno ubrzava proces računanja parametarskih kombinacija te u praksi nije predstavljalo veliki problem. Budući da akcije u simulaciji izvršavaju ljudi (uz prisustvo stručnjaka), lako je ignorirati opcije koji su očigledno neispravne. Međutim, kada je za donošenje odluka odgovorno računalo, praćenje svih ovih kompleksnih odnosa između parametara kroz razna stanja simulacije bi se tre-

balo implementirati, kako računalo ne bi odabiralo neispravne akcije. Budući da ove provjere nisu implementirane u sklopu ovog rada, većina akcija koje je algoritam podržanog učenja odabirao tijekom učenja bile su neispravne ili besmislene.

Algoritam koji nastoji modelirati CKC također je patio od prijašnjeg problema. Iako su napadačke akcije koje je algoritam odabirao većim dijelom bile dobro odabrane, parametri tih akcija su bili loše postavljeni. Ovaj problem bi se također mogao potencijalno riješiti detaljnom provjerom parametara koja uključuje praćenje izmjena kibernetičkog prostora kroz različita stanja.

U sklopu ovog rada također je isprobano mijenjanje konfiguracijskih datoteka za određene napadačke akcije kako bi CCS algoritmu isporučivao isključivo valjane kombinacije parametara. Iako se u ovom slučaju značajno smanjio broj neispravnih akcija koje je model odabrao, izvršavanje programa se značajno usporilo. Usporenje je naizgled postajalo sve značajnije što je model izvršavao više akcija, čime se razvijala napadačka osviještenost o kibernetičkom prostoru. Stoga je ovo rješenje ocijenjeno kao neodrživo.

Za kraj je bitno napomenuti da su u sklopu ovog rada napravljene brojne druge promjene koda kako bi se omogućilo treniranje umjetne inteligencije. Jedna od većih promjena uključuje dohvaćanje mogućih napadačkih akcija od CCS-a. Zbog sporosti računanja svih kombinacija parametara, dohvaćanje akcija iz CCS-a je u nekom trenutku prebačeno na WebSocket protokol. Ovo je napravljeno kako bi se poboljšalo korisničko iskustvo unutar simulacija. Korištenjem WebSocket protokola korisniku se akcije mogu dostavljati u manjim segmentima, čime se smanjuje vrijeme vizualnog ažuriranja izbornika akcija. Međutim, za treniranje umjetne inteligencije WebSocket nije idealno rješenje. Stoga je unutar CCS-a implementirana i inačica dohvaćanja korisničkih akcija koja radi preko HTTP protokola.

9. Zaključak

CCS je moćan alat koji omogućuje korisnicima simuliranje realističnih kibernetičkih napada. Ovime se kod korisnika razvija intuicija o kibernetičkoj sigurnosti te se uvježbava spremnost za potencijalne kibernetičke napade. Mapiranje CCS akcija koje je provedeno u sklopu ovog rada pokazuje da se CCS-om mogu simulirati kibernetički napadi koji u potpunosti prate CKS opisan MITRE ATT&CK bazom podataka. Dodatno, provedeno mapiranje ukazuje na potencijalne faze napada koje bi se mogle doraditi u budućim nadogradnjama CCS-a.

Iako je u prijašnjem razvoju umjetne inteligencije za CCS uspostavljeno da je moguće modelirati jednostavnije kibernetičke napade u specifičnoj vrsti simulacije, svi algoritmi koji su isprobani u sklopu ovog rada ostvarili su nezadovoljavajuće rezultate. Međutim, određeni su neki ključni problemi koji bi se trebali popraviti kako bi se u budućim iteracijama CCS-a mogla ostvariti kompleksna umjetna inteligencija. Dodatno, uspostavljeno je da su neki algoritmi zasnovani na heurističkim procjenama i pravilima jednostavnija alternativa strojnom učenju, koja bi u konačnici mogla pružati i bolje rezultate pri simuliranju kibernetičkih napada korištenjem umjetne inteligencije.

LITERATURA

[1] M. H. U. Sharif and M. A. Mohammed, "A literature review of financial losses statistics for cyber security and future trend," *World Journal of Advanced Research and Reviews*, vol. 15, no. 1, pp. 138–156, 2022.

[2] A. Bendovschi, "Cyber-attacks–trends, patterns and security countermeasures," *Procedia Economics and Finance*, vol. 28, pp. 24–31, 2015.

[3] M. Zwilling, G. Klien, D. Lesjak, Ł. Wiechetek, F. Cetin, and H. N. Basim, "Cyber security awareness, knowledge and behavior: A comparative study," *Journal of Computer Information Systems*, vol. 62, no. 1, pp. 82–97, 2022.

[4] "Hybrid security trends report." `https://www.netwrix.com/2023_hybrid_security_trends_report.html`, 2023.

[5] S. Chng, H. Y. Lu, A. Kumar, and D. Yau, "Hacker types, motivations and strategies: A comprehensive framework," *Computers in Human Behavior Reports*, vol. 5, p. 100167, 2022.

[6] M. Kadivar, "Cyber-attack attributes," *Technology Innovation Management Review*, vol. 4, no. 11, 2014.

[7] "Cyber Conflict Simulator." `https://ccs.utilis.biz/`.

[8] M. Buckbee, "What is The Cyber Kill Chain and How to Use it Effectively." `https://www.varonis.com/blog/cyber-kill-chain`, 2023.

[9] "MITRE ATT&CK." `https://attack.mitre.org/`.

[10] J. M. Helm, A. M. Swiergosz, H. S. Haeberle, J. M. Karnuta, J. L. Schaffer, V. E. Krebs, A. I. Spitzer, and P. N. Ramkumar, "Machine learning and artificial intelligence: definitions, applications, and future directions," *Current reviews in musculoskeletal medicine*, vol. 13, pp. 69–76, 2020.

[11] J. N. Kok, E. J. Boers, W. A. Kosters, P. Van der Putten, and M. Poel, "Artificial intelligence: definition, trends, techniques, and cases," *Artificial intelligence*, vol. 1, no. 270-299, p. 51, 2009.

[12] A. Burkov, *The Hundred-Page Machine Learning Book*. 2019.

[13] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. 2018.

[14] A. Woodruff, "What is a neuron?." `https://qbi.uq.edu.au/brain/brain-anatomy/what-neuron`.

[15] C. C. Aggarrwal, *Neural Networks and Deep Learning 2nd edition*. 2023.

[16] "Huggingface Deep RL Course." `https://qbi.uq.edu.au/brain/brain-anatomy/what-neuron`.

[17] C. Nalty, "A Comparison of Policy Gradient Methods for Multitask Learning,"

[18] C. Huang, "CleanRL example." `https://huggingface.co/learn/deep-rl-course/unit8/hands-on-cleanrl`.

[19] "Game ai pro 3." `https://www.gameaipro.com/`, 2017.

Modeliranje kibernetičkih napada u programu Cyber Conflict Simulator korištenjem umjetne inteligencije

Sažetak

U današnjem svijetu, povezanost na Internet je sveprisutna, a samim time i prijetnja od kibernetičkih napada. Stoga program Cyber Conflict Simulator (CCS), koji je razvila firma Utilis d.o.o, omogućava klijentima pripremu za razne kibernetičke prijetnje prije negoli se one ostvare. Međutim, jedna stavka koju CCS trenutno ne podržava je provođenje realističnih kibernetičkih napada korištenjem umjetne inteligencije. Stoga se u sklopu ovog diplomskog rada proučava mogućnost dodavanja navedene funkcionalnosti.

Ključne riječi: Cyber Conflict Simulator, CCS, podržano učenje, algoritam, simulacija, kibernetički napad, Proximal Policy Optimization, PPO, MITRE

Modeling cyber attacks in the Cyber Conflict Simulator software using artificial intelligence

Abstract

In today's world, the omnipresent Internet connectivity brings with it an omnipresent threat of cyber attacks. The Cyber Conflict Simulator (CCS) software, developed by the company Utilis LLC, provides its clients with the ability to prepare for different cyber attacks before they actually occur. However, one feature that CCS does not currently possess is simulating realistic cyber attacks using artificial intelligence. Therefore, this thesis aims to test the viability of such a functionality.

Keywords: Cyber Conflict Simulator, CCS, reinforcement learning, algorithm, simulation, cyber attack, Proximal Policy Optimization, PPO, MITRE