VISOKO-INTERAKTIVNA REDIS LIMANICA Z ELK ANALITIKO

Marin Gazvoda de Reggi[1], Sara Mihalič[1], Samo Hribar[1], Ana Bračić[1], Matevž Pesek[1]
Univerza v Ljubljani, Fakulteta za računalništvo in informatiko, Večna pot 113, 1000 Ljubljana
{mg4234, sm0770, sh8397, ab1165}@student.uni-lj.si, matevz.pesek@fri.uni-lj.si

Izvleček
Redis je zaradi svoje široke uporabe in pogosto nepravilne konfiguracije postal priljubljena tarča kibernetskih napadov, kar ustvarja potrebo po boljšem razumevanju in analizi varnostnih groženj. V tem delu predstavljamo implementacijo visoko-interaktivne Redis limanice (angl. *honeypot*), ki omogoča transparentno prestrezanje in beleženje vseh povezav ter ukazov na Redis strežnik. Sistem temelji na posredniškem strežniku v programskem jeziku Go, ki prestrežene povezave posreduje interni Redis instanci, pri tem pa vse interakcije v realnem času beleži in analizira preko integracije z naborom orodij ELK (Elasticsearch, Logstash, Kibana). Celotna rešitev je implementirana kot vsebniška aplikacija z uporabo tehnologije Docker. Eksperimentalna evalvacija je pokazala, da sistem učinkovito zaznava različne vrste napadov, od preprostih poskusov skeniranja do sofisticiranih večstopenjskih napadov. Razviti sistem predstavlja pomemben prispevek k boljšemu razumevanju varnostnih izzivov Redis strežnikov in demonstrira uporabnost limanic pri raziskovanju kibernetskih groženj.

Ključne besede: ELK analitika, kibernetska varnost, limanica, Redis, varnostne grožnje

HIGH-INTERACTIVE REDIS HONEYPOT WITH ELK ANALYTICS

Abstract
Redis has become a popular target for cyberattacks due to its widespread use and frequent misconfigurations, creating a need for better understanding and analysis of security threats. This work presents the implementation of a high-interactive Redis honeypot that enables transparent interception and logging of all connections and commands to a Redis server. The system is based on a proxy server implemented in Go programming language, which forwards intercepted connections to an internal Redis instance while logging and analyzing all interactions in real-time through integration with the ELK stack (Elasticsearch, Logstash, Kibana). The entire solution is implemented as a containerized application using Docker technology. Experimental evaluation demonstrated that the system effectively detects various types of attacks, from simple scanning attempts to sophisticated multi-stage attacks. The developed system represents an important contribution to better understanding Redis server security challenges and demonstrates the utility of honeypots in cybersecurity threat research.

Keywords: cybersecurity, ELK analytics, honeypot, Redis, security threats

1 UVOD

V sodobnem digitalnem okolju predstavljajo podatkovne baze eno ključnih komponent informacijske infrastrukture, saj podpirajo delovanje praktično vseh spletnih storitev, od družbenih omrežij do finančnih sistemov. Med njimi je pomnilniška podatkovna shramba Redis (Remote Dictionary Server), ki je postala ena najpogosteje uporabljenih rešitev za predpomnjenje in upravljanje s podatki v realnem času [1]. Redis je še posebej priljubljen v visoko-zmogljivih aplikacijah, pri katerih je ključen hiter dostop do podatkov, saj lahko izvede več kot 100.000 operacij na sekundo.

Žal njegova široka uporaba prinaša tudi večje varnostno tveganje. Wright [2] je v svoji raziskavi odkril več kot 18.000 izpostavljenih Redis instanc na internetu, od katerih je bilo kar 72 % že tarča napada. Ta podatek jasno kaže na obseg problematike in potrebo po boljšem razumevanju varnostnih groženj. Kljub jasnim priporočilom razvijalcev, da Redis ni namenjen neposredni izpostavitvi na internet [3], se v praksi pogosto dogaja, da so instance nepravilno konfigurirane in nezaščitene. Posledice takšnih napak so lahko resne – od

kraje podatkov do popolnega prevzema nadzora nad sistemom, kar lahko vodi v velike finančne izgube in okrnjen ugled podjetij.

Za učinkovito zaščito je zato ključno razumevanje načinov, kako napadalci odkrivajo in izkoriščajo ranljive Redis strežnike. Pri tem so limanice (angl. *honeypots*) oz. namensko vzpostavljeni navidezno ranljivi sistemi – postale ključno orodje v sodobni varnostni analitiki. Delujejo kot vabe, ki privabljajo napadalce in beležijo njihove aktivnosti ter tako nudijo vpogled v njihove tehnike, orodja in motivacije. Z natančnim spremljanjem in analizo teh interakcij lahko varnostni strokovnjaki identificirajo nove vrste napadov in razvijajo učinkovitejše obrambne mehanizme za zaščito kritične internetne infrastrukture.

Prav na opisanem principu delovanja limanic temelji naš pristop, s katerim smo se lotili proaktivnega odkrivanja in analize napadov na strežnike Redis. V ta namen smo razvili visoko-interaktivno limanico, integrirano z analitično platformo ELK (Elasticsearch, Logstash, Kibana). Implementirana rešitev omogoča natančno spremljanje poskusov vdorov, beleženje uporabljenih tehnik in analizo vzorcev napadov v realnem času. S tem prispevamo k boljšemu razumevanju varnostnih groženj in razvoju učinkovitejših zaščitnih mehanizmov za kritično internetno infrastrukturo.

2 TEORETIČNI KONCEPTI IN TEHNOLOGIJA

2.1 Redis podatkovna baza

Redis (Remote Dictionary Server) je odprtokodna, v pomnilniku delujoča podatkovna shramba, ki se uporablja kot podatkovna baza, predpomnilnik in posrednik sporočil [1]. Redisova ključna prednost je izjemna hitrost delovanja, saj podatke hrani v glavnem pomnilniku (RAM), kar omogoča zelo nizke latence pri dostopu do podatkov. Podpira različne podatkovne strukture, kot so nizi, seznami, množice, razpršene tabele in urejene množice, zaradi česar je primeren za širok nabor uporabniških scenarijev [4].

Redisova arhitektura temelji na modelu strežnik–odjemalec, kjer strežnik upravlja s podatkovnimi strukturami v pomnilniku, odjemalci pa komunicirajo s strežnikom preko preprostega tekstovnega protokola RESP (REdis Serialization Protocol) [5]. Taka zasnova, čeprav učinkovita z vidika hitrosti, predstavlja določena varnostna tveganja, še posebej v primeru nepravilne konfiguracije ali izpostavljenosti na javno omrežje.

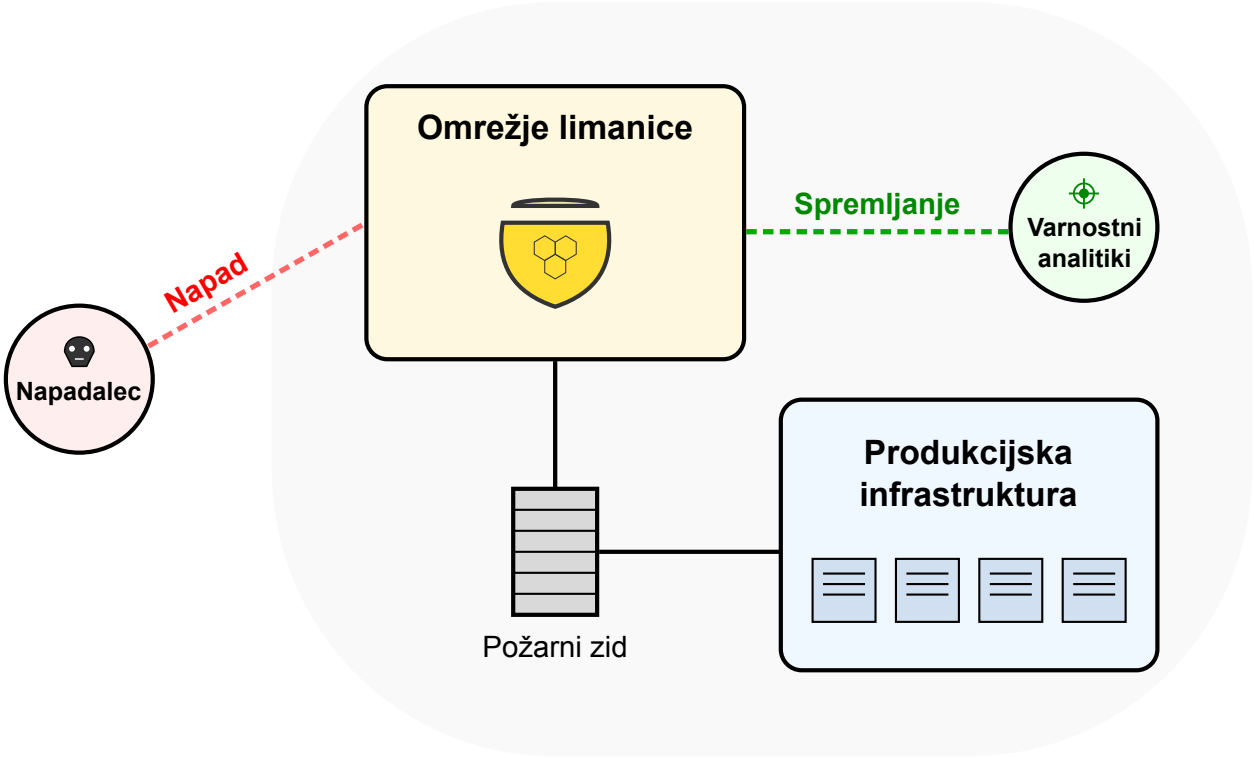
2.2 Limanice

Limanice (angl. *honeypots*) so varnostna orodja, zasnovana za simulacijo ranljivih sistemov z namenom privabljanja, odkrivanja in analize zlonamernih aktivnosti [6]. Po namenu uporabe jih delimo na raziskovalne in produkcijske limanice. Raziskovalne limanice so namenjene pridobivanju znanja o napadalčevih tehnikah, orodjih in motivacijah, medtem ko produkcijske limanice služijo zaščiti dejanske infrastrukture z odkrivanjem in upočasnitvijo napadov. Te delujejo kot vabe, ki preusmerijo napadalčevo pozornost s kritičnih sistemov na nadzorovan in izolirani sistem, kjer se njihova aktivnost beleži brez nevarnosti za ključno infrastrukturo.

Glede na stopnjo interakcije, ki jo dopuščajo napadalcem, jih delimo v tri kategorije:

- **Nizko-interaktivne limanice** simulirajo samo osnovne funkcionalnosti in so primerne predvsem za zbiranje statističnih podatkov o poskusih napadov.
- **Srednje-interaktivne limanice** ponujajo večji nabor funkcionalnosti in nudijo bolj poglobljeno analizo napadov, vendar še vedno v kontroliranem okolju.
- **Visoko-interaktivne limanice** predstavljajo popolnoma funkcionalne sisteme, ki zagotavljajo največjo stopnjo interakcije in s tem najnatančnejšo analizo napadalčevega vedenja [7].

Učinkovitost limanice je odvisna od njene prepričljivosti simulacije ciljnega sistema, pri čemer mora biti dovolj privlačna za napadalce, a hkrati varna za upravljanje in analizo [8]. Pri implementaciji je ključnega pomena ravnotežje med stopnjo realističnosti oz. pristnosti simulacije in varnostjo sistema.



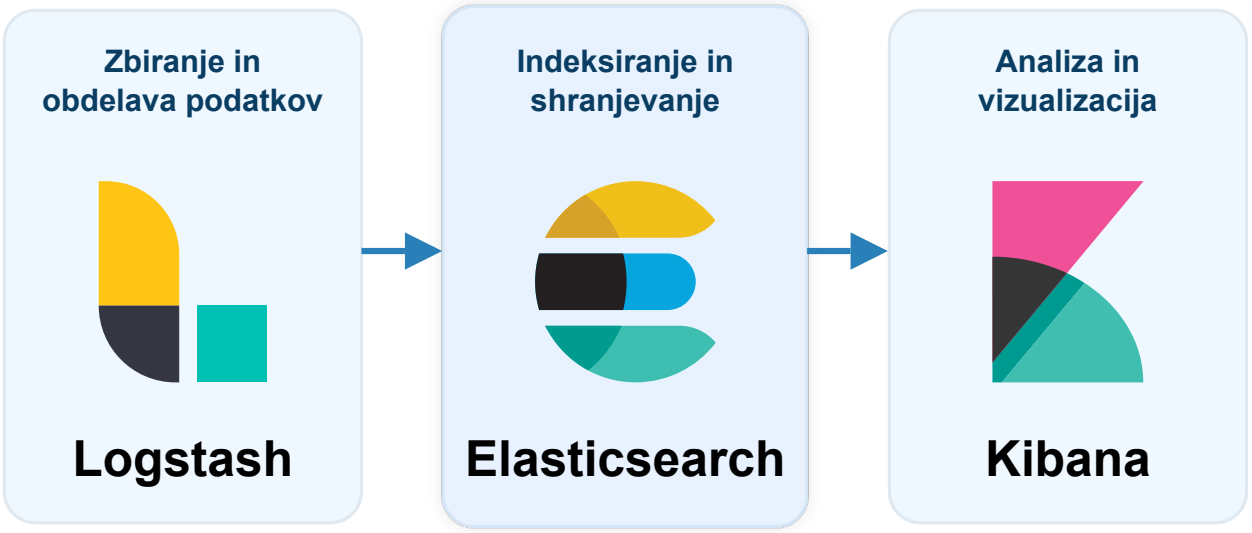
Slika 1: Napadalec skuša napasti produkcijsko aplikacijo, ki je zaščitena za požarnim zidom. Da preusmerimo napadalčevo pozornost in pridobimo informacije o napadalčevih tehnikah, namestimo limanico kot navidezno ranljivo tarčo. Limanica tako služi dvojnemu namenu: odvrne pozornost od kritične infrastrukture in hkrati omogoča varnostnim analitikom beleženje ter analizo napadalčevih aktivnosti za razvoj boljših obrambnih strategij.

2.3 Nabor orodij ELK

ELK predstavlja nabor odprtokodnih orodij za zbiranje, procesiranje, shranjevanje in vizualizacijo podatkov [9]. Sestavljajo ga tri ključne komponente:

- **Elasticsearch:** porazdeljeno iskalno in analitično orodje, optimizirano za delo z velikimi količinami strukturiranih in nestrukturiranih podatkov.
- **Logstash:** orodje za procesiranje podatkovnih tokov, ki omogoča zbiranje podatkov iz različnih virov, njihovo transformacijo in posredovanje v Elasticsearch.
- **Kibana:** spletni vmesnik za vizualizacijo in analizo podatkov, shranjenih v Elasticsearch.

V kontekstu varnostne analitike je nabor orodij ELK uporaben za učinkovito zbiranje in analizo varnostnih dogodkov v realnem času. Zaradi njegove fleksibilnosti pri obdelavi različnih formatov podatkov in zmogljive možnosti vizualizacije je posebej primeren za analizo podatkov iz limanic [10].



Slika 2: Nabor orodij ELK (Elasticsearch, Logstash, Kibana).

2.4 Integracija komponent

Redis limanica je v kombinaciji z naborom orodij ELK uporabna za celovito varnostno analizo, kjer limanica prestreza dostope do Redis strežnika, Logstash zbira in strukturira podatke o interakcijah, Elasticsearch jih shrani, Kibana pa ponuja njihovo vizualizacijo [11, 12].

3 Pregled področja

Redis je zaradi svoje arhitekture, ki prednostno obravnava hitrost in enostavnost uporabe pred varnostjo, posebej ranljiv za različne vrste napadov. Carlson [1] v svojem delu izpostavlja, da ta zasnova dodatno povečuje tveganje za napade, še posebej v primerih nepravilne konfiguracije. Antirez [3] poudarja, da je Redis zasnovan za uporabo v zaupanja vrednih okoljih in ni namenjen neposredni izpostavitvi na internet. Kljub temu pa raziskave kažejo, da so napačne konfiguracije in nenamerna izpostavljenost Redis strežnikov pogost pojav, kar odpira vrata različnim vrstam napadov [13].

Fan in sod. [14] v svojem preglednem delu identificirajo dva ključna elementa limanic – vabo (angl. *decoy*) in prestreznik (angl. *captor*), ki skupaj omogočata učinkovito simulacijo ranljivih sistemov in beleženje napadov. Holz in Raynal [7] opozarjata na izzive pri implementaciji prepričljivih limanic, saj napredni napadalci razvijajo tehnike odkrivanja in izogibanja le-tem.

Sodobni pristopi k analizi podatkov iz limanic se močno opirajo na napredna orodja za agregacijo in vizualizacijo podatkov. Yang in sod. [9] predstavljajo uporabo nabora orodij ELK za analizo kibernetskih napadov in učinkovito obdelavo velikih količin dnevniških zapisov v realnem času. V kontekstu oblačnih limanic so Izhikevich in sod. [10] identificirali ključni pomen geografske porazdelitve in selektivnosti napadalcev pri izbiri tarč.

Na področju Redis limanic obstaja več obstoječih implementacij. Onishi [15] je razvil osnovno Redis limanico, ki implementira najpogostejše Redis ukaze. Beelzebub [16] predstavlja naprednejši pristop z uporabo umetne inteligence za simulacijo vedenja sistemov. Oosterhof [17] je zasnoval Cowrie, srednje do visoko interaktivno SSH/Telnet limanico, ki omogoča tako emulacijo kot posredovanje povezav na dejanske sisteme. T-Pot [18] ponuja celovito platformo za postavitev različnih vrst limanic, vključno z vizualizacijskimi orodji. Anirudh in sod. [19] so pokazali, da so limanice posebej učinkovite pri zaznavanju in blaženju DoS napadov na IoT naprave, kar je relevantno tudi za Redis okolja. Vendar pa te implementacije bodisi ne ponujajo dovolj natančne simulacije Redis funkcionalnosti bodisi ne podpirajo učinkovitega prestrezanja in analize kompleksnejših napadov.

Raziskave kažejo, da so napadi na Redis strežnike pogosto avtomatizirani in sledijo predvidljivim vzorcem. Bythwood in sod. [20] v svoji analizi avtomatiziranih napadov ugotavljajo, da večina napadalcev cilja na znane ranljivosti in uporablja standardizirane pristope. To dejstvo dodatno poudarja potrebo po razvoju specializiranih limanic, ki lahko natančno simulirajo ranljive Redis instance in omogočajo podrobno analizo napadov.

Spitzner [6] izpostavlja pomembnost limanic pri odkrivanju notranjih groženj, medtem ko Wang in sod. [21] predstavljajo teoretični model za optimizacijo postavitve limanic z uporabo teorije iger. Učinkovitost limanic pri odkrivanju izsiljevalskih napadov je v svoji raziskavi predstavil Moore [22], Vishwakarma in Jain [23] pa predlagata uporabo strojnega učenja za izboljšanje zmogljivosti limanic pri odkrivanju botnet DDoS napadov.

Priya in Chakkaravarthy [11] sta raziskala uporabo vsebniških limanic v oblačnem okolju, ki zagotavlja boljšo skalabilnost in enostavnejše upravljanje. Rabzelj in sod. [24] so razvili fleksibilno in skalabilno HTTP platformo za limanice, ki lahko simulira različne spletne storitve. Izboljšan model limanice s poudarkom na večji interakciji z napadalci ob hkratnem ohranjanju integritete in privlačnosti sistema sta predstavila Abbas-Escribano in Debar [8], medtem ko so v preglednem članku o pomenu odprtokodnih limanic pri razvoju in raziskavah na področju kibernetske varnosti so svoje ugotovitve objavili Ilg in sod. [12].

Sistematičen pregled obstoječih raziskav kaže, da trenutne implementacije Redis limanic in varnostne analitike ne zagotavljajo celovitega pristopa k spremljanju in analizi napadov. Večina obstoječih rešitev se osredotoča le na osnovno prestrezanje Redis ukazov ali splošno analitiko varnostnih dogodkov, pri čemer pogrešamo integracijo visoko-interaktivne Redis limanice z zmogljivostmi analize v realnem času.

4 METODOLOGIJA

V raziskavi smo uporabili kombinacijo eksperimentalnega in analitičnega pristopa, ki temelji na implementaciji visoko-interaktivne Redis limanice ter integraciji z naprednimi analitičnimi orodji. Naš metodološki okvir je zasnovan tako, da podpira celovito spremljanje in analizo poskusov vdorov v Redis strežnike, od začetnega odkrivanja do podrobne forenzične analize. Raziskava je potekala v treh ključnih fazah: razvoj limanice, implementacija analitike in validacija sistema.

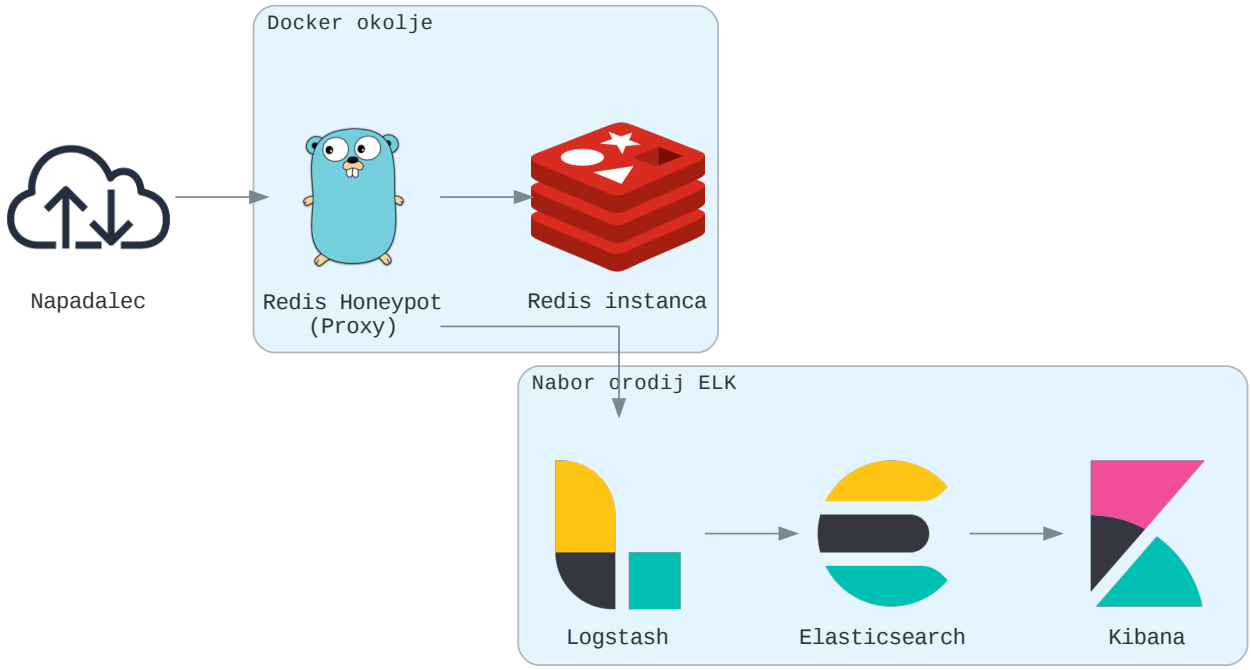
V prvi fazi smo razvili posredniški strežnik v programskem jeziku Go, ki deluje kot transparentna limanica med napadalcem in interno Redis instanco. Ta komponenta je uporabna za natančno prestrezanje in beleženje vseh povezav ter ukazov, ne da bi napadalec zaznal prisotnost nadzornega sistema.

Druga faza je obsegala vzpostavitev analitične infrastrukture z uporabo sistema orodij ELK, kjer smo implementirali napredno procesiranje in vizualizacijo podatkov v realnem času. Celoten sistem smo implementirali kot vsebniško aplikacijo z uporabo tehnologije Docker, ki zagotavlja konsistentno delovanje v različnih okoljih in enostavno postavitev dodatnih instanc za potrebe skaliranja.

V zaključni fazi validacije smo sistem javno izpostavili na internetu ter spremljali in analizirali realne poskuse vdorov. Analiza zbranih podatkov nam je ponudila vpogled v njihove pristope, orodja in vzorce obnašanja.

4.1 Implementacija

Implementacija Redis limanice temelji na več ključnih komponentah, ki skupaj tvorijo celovito rešitev za prestrezanje in analizo Redis povezav. Arhitekturni diagram sistema je prikazan na Sliki 3.



Slika 3: Arhitektura sistema: Redis limanični posredniški (proxy) strežnika v Docker okolju prestreza napadalčeve povezave, nabor orodij ELK (Elasticsearch, Logstash, Kibana) pa beleži in analizira vse interakcije.

4.1.1 Redis posredniški strežnik

Osrednji del sistema predstavlja posredniški strežnik, implementiran v programskem jeziku Go, ki nudi učinkovito upravljanje s sočasnimi povezavami in enostavno integracijo s protokolom RESP z uporabo knjižnice `redcon`. Strežnik deluje kot transparentni posrednik med odjemalcem in ciljnim Redis strežnikom. Posluša na standardnih Redis vratih (6379), kjer prestreza vse povezave in ukaze. Te zahteve nato posreduje na interno Redis instanco, ki teče na vratih 6380. Med delovanjem strežnik beleži vse interakcije med odjemalcem in ciljnim strežnikom.

Ključni del implementacije predstavlja struktura `HoneypotServer`, ki vzdržuje povezave s ciljnim Redis strežnikom, sistemom za beleženje in Logstash strežnikom:

- `client` – Redis odjemalec za komunikacijo z interno instanco
- `logger` – komponenta za lokalno beleženje dogodkov
- `logstashConn` – TCP povezava za pošiljanje strukturiranih zapisov

Naš pristop zagotavlja visoko stopnjo prepričljivosti za napadalce, kjer vaba temelji na avtentični Redis instanci namesto na simulaciji protokola. Tako zagotavljamo, da se sistem odziva z enako funkcionalnostjo, semantiko ukazov in lastnostmi kot produkcijski Redis strežniki. Posredniški strežnik deluje transparentno in ne spreminja Redis protokola RESP, zato napadalci ne zaznajo prisotnosti prestreznega sistema. To omogoča natančno analizo pristnih napadov brez tveganja za razkritje limanice.

4.1.2 Beleženje in analiza

Sistem implementira dvonivojsko beleženje dogodkov. Vsak dogodek se zabeleži lokalno v dnevniško datoteko in hkrati posreduje v Logstash za nadaljnjo analizo. Struktura zapisov je zasnovana tako, da podpira učinkovito analizo. Vsak zapis vsebuje časovni žig v ISO 8601 formatu ter IP naslov odjemalca, ki je vzpostavil povezavo. Poleg tega se beleži vrsta dogodka, ki je lahko vzpostavitev povezave, izvedba ukaza ali prekinitev povezave. Pri izvedbi ukazov se shranijo tako Redis ukaz kot njegovi argumenti. V primeru napak pri izvajanju se zabeležijo tudi podrobnosti o napaki.

Implementacija uporablja strukturo `LogEntry` za serializacijo podatkov v JSON format:

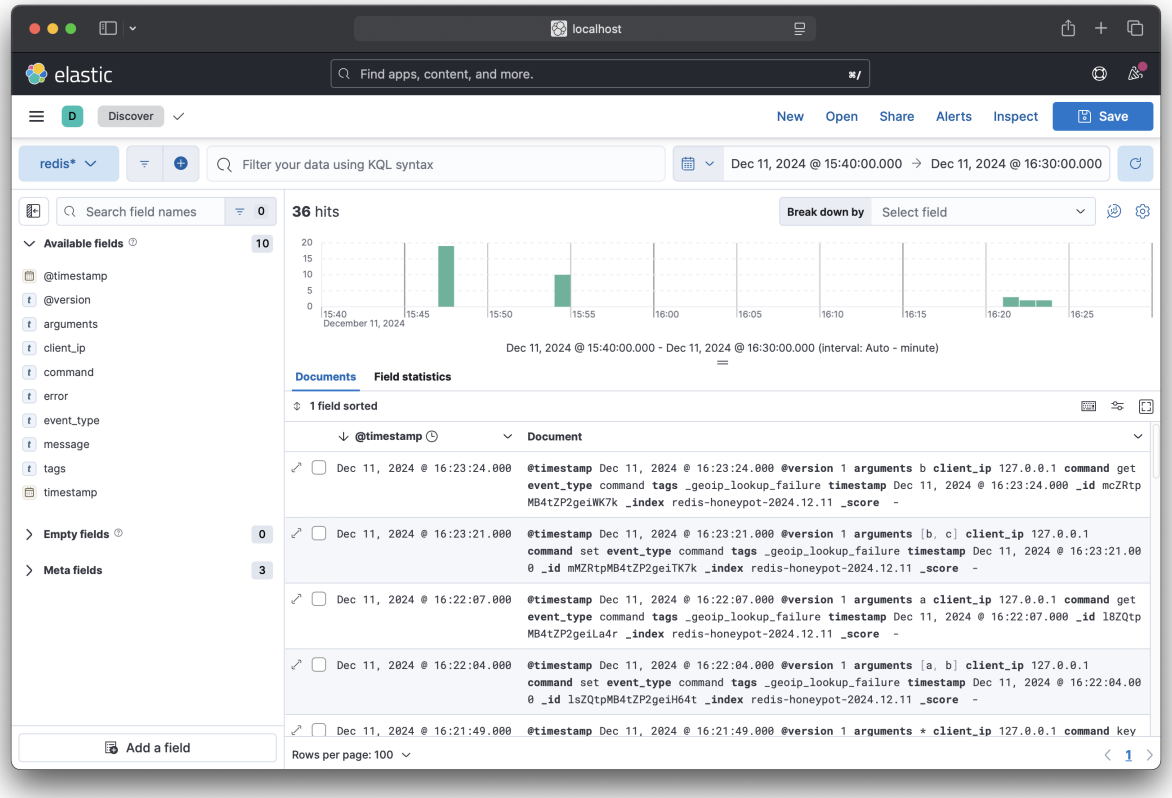
```
{
    "timestamp": "2024-12-24T15:04:05Z",
    "client_ip": "192.168.1.1",
    "command": "set",
    "arguments": ["key", "value"],
    "event_type": "command"
}
```

4.1.3 Vzpostavitev vsebniškega okolja

Sistem je zasnovan za izvajanje v vsebniških okoljih, kar zagotavlja enostavno vzpostavitev in skaliranje. Docker kompozicija vključuje štiri ključne storitve:

- `redis-honeypot` – glavni posredniški strežnik z interno Redis instanco
- `elasticsearch` – podatkovna baza za shranjevanje dogodkov
- `logstash` – procesiranje in razširjanje dnevniških zapisov
- `kibana` – vizualizacija in analiza podatkov

Postavitev sistema je avtomatizirana z uporabo Docker Compose, ki zagotavlja izolacijo posameznih komponent, trajnost podatkov med ponovnimi zagoni, omrežno povezljivost med vsemi storitvami ter centralizirano upravljanje z dnevniškimi zapisi. S tem pristopom je upravljanje celotnega sistema enostavno in zanesljivo.



Slika 4: Pregled povezav in izvedenih ukazov na nadzorni plošči Kibane.

4.1.4 Integracija z naborom orodij ELK

Logstash konfiguracija je optimizirana za obdelavo Redis dogodkov. Vključuje TCP vhod za sprejemanje JSON formatiranih zapisov ter časovno normalizacijo dogodkov. Sistem dodaja tudi geolokacijske podatke iz IP naslovov napadalcev. Vsi dogodki se indeksirajo v dnevno rotirajoče Elasticsearch indekse.

Elasticsearch shema je prilagojena učinkovitemu iskanju in združevanju podatkov o napadih. Kibana nudi vnaprej pripravljene nadzorne plošče za pregled geografske porazdelitve napadov in časovno analizo aktivnosti. Poleg tega so na voljo tudi statistični pregledi najpogostejših ukazov ter analiza poskusov izkoriščanja znanih ranljivosti v sistemu.

Za preverjanje delovanja smo sistem najprej testirali v nadzorovanem laboratorijskem okolju, kjer smo preizkusili različne scenarije Redis povezav in ukazov. Nadzorna plošča Kibane, prikazana na Sliki 4, nudi celovit pregled nad vsemi zabeleženimi dogodki v sistemu. Vizualizacija prikazuje časovno porazdelitev povezav oz. uporabljenih Redis ukazov. Sistem beleži in vizualizira vse ključne metrike: čas povezave, izvorni IP naslov, uporabljene ukaze in njihove argumente ter morebitne napake pri izvajanju. Za hitro identifikacijo vzorcev v obnašanju odjemalcev je posebej uporabna možnost filtriranja in agregacije podatkov po različnih dimenzijah.

4.2 Eksperimentalna evalvacija

Za evalvacijo učinkovitosti implementirane Redis limanice smo sistem izpostavili na javno dostopnem strežniku. Instanca je bila dostopna 24 ur na privzetih vratih 6379 brez dodatnih varnostnih mehanizmov. Namen takšne konfiguracije je bil privabiti čim več potencialnih napadalcev in analizirati njihove tehnike izkoriščanja Redis strežnikov.

Opazovalno obdobje je bilo razmeroma kratko za poglobljeno znanstveno analizo, vendar smo z eksperimentom predvsem želeli preveriti, ali sistem v praksi deluje in tehnična implementacija izpolnjuje zastavljena pričakovanja. V tem okviru smo uspešno potrdili zanesljivo delovanje sistema ter demonstrirali, da limanica zazna tako osnovne kot tudi naprednejše napade na Redis strežnik.

Sistem smo namestili na virtualiziranem strežniku v podatkovnem centru. Za zagotavljanje konsistentnega delovanja smo uporabili Docker kompozicijo z vsemi potrebnimi komponentami (Redis, ELK stack) in poskrbeli za persistenco dnevniških zapisov. Vsi poskusi povezav in izvedenih ukazov so se v realnem času beležili lokalno in v Elasticsearch bazi.

5 Rezultati in analiza

5.1 Enostavni napadi

Analiza dnevniških zapisov je razkrila povezave iz osmih različnih IP naslovov, pri čemer so se napadi razlikovali tako po kompleksnosti kot po namenu. Ti poskusi so bili večinoma kratkotrajni in so se zaključili po nekaj osnovnih ukazih, kar kaže na avtomatizirano odkrivanje potencialno ranljivih Redis strežnikov. To dodatno potrjujejo konstantni časovni intervali med ukazi, ki običajno trajajo med 100 in 200 ms. Pri večini poskusov je bilo opaziti tudi ponavljajoč vzorec prekinitev in ponovnih vzpostavitev povezave, kar je značilno za orodja, ki skenirajo velike bloke IP naslovov.

Podrobnejša analiza enostavnih napadov je razkrila več značilnih vzorcev. V prvi fazi so napadalci tipično uporabili ukaz `INFO`, s katerim so pridobili osnovne informacije o Redis strežniku, vključno z verzijo, operacijskim sistemom in konfiguracijo pomnilnika. Sledil je ukaz `CONFIG GET *`, s katerim so poskušali pridobiti podrobnosti o sistemski konfiguraciji. V več primerih so napadalci nato želeli spremeniti direktorij za shranjevanje podatkov z ukazom `CONFIG SET dir`, vendar brez nadaljnjih poskusov izkoriščanja.

Kljub svoji tehnični preprostosti ti osnovni napadi predstavljajo pomemben vpogled v začetne faze napadov na Redis strežnike in razkrivajo razširjeno uporabo avtomatiziranih orodij za iskanje ranljivih sistemov. Njihova pogostost in predvidljivi vzorci nakazujejo potrebo po razvoju naprednih varnostnih mehanizmov, ki bi omogočali zgodnje zaznavanje in blokiranje potencialnih vdorov.

5.2 Sofisticirani napadi

Posebej zanimiv je bil sofisticiran večfazni napad, ki je demonstriral napredne tehnike zlorabe Redis strežnikov. Napadalec je začel z osnovnim prepoznavanjem preko ukaza `INFO SERVER`, nato pa nadaljeval s poskusom namestitve zlonamerne kode preko `SET` ukaza z Base64 kodirano vsebino. Poskusil je tudi z izvajanjem Lua kode za vzpostavitev povratne povezave ter z namestitvijo SSH ključa za trajni dostop. Ko ti poskusi niso bili uspešni, je z ukazom `SLAVEOF` poskušal izkoristiti Redis replikacijski mehanizem ter naložiti zlonamerni modul.

Geografska analiza izvornih IP naslovov je pokazala globalno porazdelitev napadov in potrdila, da so Redis strežniki tarča avtomatiziranih napadov iz različnih delov sveta. Časovna porazdelitev napadov pa je pokazala enakomerno aktivnost s povečano intenziteto v večernih urah po UTC. Napadi so trajali tudi do 15 minut, z več premori med fazami.

Podrobneje si lahko pogledamo večfazni napad, ki je demonstriral tehnike zlorabe Redis strežnikov in je potekal v več jasno definiranih fazah:

1. **Pridobivanje informacij:** Napad se je začel z osnovnim prepoznavanjem preko ukaza `INFO SERVER`, s katerim so bile pridobljene informacije o različici in konfiguraciji Redis strežnika.
2. **Priprava okolja:** Sledil je ukaz `FLUSHDB` za čiščenje podatkovne baze, kar je tipična priprava za nadaljnje zlonamerne aktivnosti.
3. **Nameščanje zlonamerne kode:** Napadalec je poskusil namestiti zlonamerno kodo preko `SET` ukaza. Analiza Base64 kodirane vsebine je razkrila poskus vzpostavitve povratne povezave (reverse shell) na predefiniran IP naslov preko vrat `60148`.
4. **Spreminjanje sistemske konfiguracije:** Nato je sledil poskus spreminjanja Redis konfiguracije za dostop do različnih sistemskih direktorijev. Tarče so bili `/var/spool/cron/` za namestitev zlonamernih cronjob opravil, `/root/.ssh/` za vstavljanje SSH ključev za trajni dostop in `/tmp/` za pisanje v začasni direktorij.
5. **Izvajanje Lua kode:** Napadalec je nato poskusil z naprednejšo tehniko – izvedbo Lua skripte za povratno povezavo z ukazom `eval`.
6. **Zloraba replikacijskega mehanizma:** Po neuspehu je želel izkoristiti Redis replikacijski mehanizem z ukazom `SLAVEOF <IP> 60148`, kar bi omogočilo prenos zlonamernih podatkov preko Redis protokola.
7. **Nalaganje modulov:** Redis razširitveni sistem je bil tarča napada, ko je napadalec želel naložiti zlonamerni modul `exp.so` z uporabo ukaza `MODULE LOAD`.
8. **Čiščenje sledi:** Ob koncu napada je poskusil izbrisati sledi z ukazi `system.exec` za odstranitev datotek in `MODULE UNLOAD` za odstranitev modulov.

Dekodiranje sumljive Base64 vsebine je pokazalo poskus vzpostavitve povezave z oddaljenim strežnikom za potencialno vključitev v botnet. Napadalec je sistematično testiral različne tehnike vdora. SSH ključ, ki ga je poskušal namestiti, je bil generiran za uporabnika `root@localhost.localdomain`, kar razkriva uporabo standardnih napadalnih orodij.

Ta primer prikazuje sodobne napade na Redis strežnike, kjer napadalci spretno prilagajajo svoje pristope. Vse uporabljene tehnike izkoriščajo znane ranljivosti in pogoste napačne konfiguracije, kar kaže na nujnost temeljitega pregleda varnostnih nastavitev.

Rezultati potrjujejo uspešnost implementirane limanice pri beleženju napadov in identifikaciji različnih tehnik izkoriščanja, s čimer sistem služi kot učinkovito orodje za raziskovanje avtomatiziranih skeniranj Redis strežnikov.

6 DISKUSIJA

Predlagana rešitev v nasprotju z osnovnimi implementacijami, kot je Onishijeva limanica [15], ki simulira le izbrane Redis ukaze, uporablja dejansko Redis instanco. Ta pristop zagotavlja popolno podporo vsem funkcionalnostim in omogoča zaznavanje tudi sofisticiranih napadov, kot so Lua skripte, manipulacija datotečnih poti ali zloraba replikacijskih mehanizmov.

Ključna prednost sistema je integracija z naborom orodij ELK, ki v skladu z ugotovitvami Yanga in sod. [9] omogoča napredno analitiko in vizualizacijo podatkov v realnem času. Z uporabo vsebniške tehnologije Docker, podobno kot pri pristopu Priye in Chakkaravarthyja [11], zagotavljamo enostavno vzpostavitev in razširljivost sistema.

Kljub prednostim ima implementacija tudi določene omejitve. Izvajanje dejanske Redis instance zahteva več sistemskih virov v primerjavi z enostavnimi simulacijami, kar lahko pri velikem številu sočasnih povezav predstavlja performančno ozko grlo. Dodatno pomanjkljivost raziskave predstavlja relativno kratko obdobje opazovanja, saj bi daljša izpostavljenost omogočila bolj poglobljeno analizo različnih vzorcev napadov in časovnih trendov.

Opažanja iz naše študije se ujemajo z ugotovitvami raziskave Bythwooda in sod. [20], ki prav tako poroča o uporabi standardiziranih tehnik za izkoriščanje znanih ranljivosti. Tudi mi smo opazili, da so napadalci sledili predvidljivim vzorcem in uporabljali uveljavljene tehnike. Skladnost teh rezultatov kaže, da naša metodologija ustrezno odraža realne varnostne izzive v Redis okoljih.

7 ZAKLJUČEK

Predstavili smo implementacijo Redis limanice za učinkovito prestrezanje in analizo Redis povezav. Sistem temelji na transparentnem posredniškem strežniku, implementiranem v programskem jeziku Go, ki prestreže vse povezave na Redis vrata in jih posreduje interni Redis instanci. Ključni dosežek predstavlja uspešna integracija s sistemom ELK, ki omogoča celovito zbiranje, shranjevanje in vizualizacijo podatkov o povezavah in izvedenih ukazih.

Sistem je avtomatiziran z uporabo vsebniške tehnologije Docker, ki podpira enostavno vzpostavitev in upravljanje vseh komponent. Predstavlja celovito rešitev za spremljanje in analizo Redis povezav ter razumevanje varnostnih groženj.

Eksperimentalna postavitev je uspešno demonstrirala tehnično zmogljivost sistema pri zaznavanju in analizi različnih vrst napadov, od preprostih poskusov skeniranja do sofisticiranih večstopenjskih napadov z uporabo naprednih tehnik zlorabe Redis strežnikov. Ta validacija predstavlja temelj za prihodnje in obsežnejše raziskave.

Za nadaljnji razvoj sistema predlagamo več možnih izboljšav. Prva je implementacija samodejnega čiščenja oz. periodičnega resetiranja notranje Redis instance, s čimer bi preprečili prekomerno kopičenje podatkov in zagotovili tekoče delovanje sistema. Druga pomembna nadgradnja bi bila integracija strojnega učenja za analizo vzorcev napadov, ki bi prispevala k avtomatski identifikaciji in klasifikaciji zlonamernih aktivnosti.

Sistem bi lahko razširili v distribuirano arhitekturo z več instancami limanic in s tem omogočili zbiranje podatkov iz različnih geografskih lokacij in boljše razumevanje globalnih vzorcev napadov. Smiselna bi bila tudi implementacija avtomatiziranih odzivov na zaznane napade, kot so dinamično blokiranje IP naslovov ali prilagajanje simuliranega vedenja sistema. Dolgoročno bi lahko arhitekturo sistema razširili za podporo drugim protokolom in storitvam, kot so HTTP, SSH in Telnet, s čimer bi pridobili obsežnejši vpogled v različne vrste kibernetskih napadov.

Razviti sistem predstavlja pomemben korak k boljšemu razumevanju varnostnih izzivov Redis strežnikov in demonstrira uporabnost limanic pri raziskovanju kibernetskih groženj. Z implementacijo in eksperimentom smo dokazali, da je mogoče učinkovito združiti različne odprtokodne tehnologije v celovit sistem za varnostno analitiko, ki lahko pomembno prispeva k izboljšanju varnosti Redis postavitev v produkcijskih okoljih.

LITERATURA

[1] Josiah L. Carlson. *Redis in Action*. USA: Manning Publications Co., 2013.

[2] Jordan Wright. *Over 18,000 Redis Instances Targeted by Fake Ransomware*. Avg. 2016. URL: https://duo.com/decipher/over-18000-redis-instances-targeted-by-fake-ransomware (pridobljeno 23. 12. 2024).

[3] antirez. *A few things about Redis security*. Nov. 2015. URL: https://antirez.com/news/96 (pridobljeno 23. 12. 2024).

[4] Redis. *Redis Data Types*. 2024. URL: https://redis.io/docs/data-types/ (pridobljeno 24. 12. 2024).

[5] Redis. *Redis Protocol specification*. 2024. URL: https://redis.io/docs/reference/protocol-spec/ (pridobljeno 24. 12. 2024).

[6] L. Spitzner. "Honeypots: catching the insider threat". V: *19th Annual Computer Security Applications Conference, 2003. Proceedings.* Dec. 2003, str. 170–179. DOI: 10.1109/CSAC.2003.1254322.

[7] T. Holz in F. Raynal. "Detecting honeypots and other suspicious environments". V: *Proceedings from the Sixth Annual IEEE SMC Information Assurance Workshop*. Jun. 2005, str. 29–36. DOI: 10.1109/IAW.2005.1495930.

[8] Marwan Abbas-Escribano in Hervé Debar. "An Improved Honeypot Model for Attack Detection and Analysis". V: *Proceedings of the 18th International Conference on Availability, Reliability and Security*. ARES '23. Benevento, Italy: Association for Computing Machinery, 2023. DOI: 10.1145/3600160.3604993.

[9] Chao-Tung Yang in sod. "Cyberattacks detection and analysis in a network log system using XGBoost with ELK stack". V: *Soft Computing* 26.11 (jun. 2022), str. 5143–5157. DOI: 10.1007/s00500-022-06954-8.

[10] Liz Izhikevich in sod. "Cloud Watching: Understanding Attacks Against Cloud-Hosted Services". V: *Proceedings of the 2023 ACM on Internet Measurement Conference*. IMC '23. Montreal QC, Canada: Association for Computing Machinery, 2023, str. 313–327. DOI: 10.1145/3618257.3624818.

[11] V. S. Devi Priya in S. Sibi Chakkaravarthy. "Containerized cloud-based honeypot deception for tracking attackers". V: *Scientific Reports* 13.1 (jan. 2023), str. 1437. DOI: 10.1038/s41598-023-28613-0.

[12] Niclas Ilg in sod. "A survey of contemporary open-source honeypots, frameworks, and tools". V: *Journal of Network and Computer Applications* 220 (2023), str. 103737. DOI: 10.1016/j.jnca.2023.103737.

[13] Lacework Labs. *Anatomy of a Redis Exploit*. Okt. 2018. URL: https://www.lacework.com/blog/anatomy-of-a-redis-exploit (pridobljeno 23. 12. 2024).

[14] Wenjun Fan in sod. "Enabling an Anatomic View to Investigate Honeypot Systems: A Survey". V: *IEEE Systems Journal* 12.4 (dec. 2018), str. 3906–3919. DOI: 10.1109/JSYST.2017.2762161.

[15] Kazuki Onishi. *redis-honeypot*. URL: https://github.com/0n1shi/redis-honeypot (pridobljeno 23. 12. 2024).

[16] Mario Candela. *beelzebub*. URL: https://github.com/mariocandela/beelzebub (pridobljeno 23. 12. 2024).

[17] Michel Oosterhof. *cowrie*. URL: https://github.com/cowrie/cowrie (pridobljeno 23. 12. 2024).

[18] Deutsche Telekom Security GmbH. *tpotce*. URL: https://github.com/telekom-security/tpotce (pridobljeno 23. 12. 2024).

[19] M Anirudh, S Arul Thileeban in Daniel Jeswin Nallathambi. "Use of honeypots for mitigating DoS attacks targeted on IoT networks". V: *2017 International Conference on Computer, Communication and Signal Processing (ICCCSP)*. Jan. 2017, str. 1–4. DOI: 10.1109/ICCCSP.2017.7944057.

[20] Willie Bythwood, Justin Bentley in Iman Vakilinia. "Analyses of Automated Malicious Internet Traffic Using Open-Source Honeypots". V: *SoutheastCon 2023*. Apr. 2023, str. 68–75. DOI: 10.1109/SoutheastCon51012.2023.10115073.

[21] Kun Wang in sod. "Strategic Honeypot Game Model for Distributed Denial of Service Attacks in the Smart Grid". V: *IEEE Transactions on Smart Grid* 8.5 (sep. 2017), str. 2474–2482. DOI: 10.1109/TSG.2017.2670144.

[22] Chris Moore. "Detecting Ransomware with Honeypot Techniques". V: *2016 Cybersecurity and Cyberforensics Conference (CCC)*. Avg. 2016, str. 77–81. DOI: 10.1109/CCC.2016.14.

[23] Ruchi Vishwakarma in Ankit Kumar Jain. "A Honeypot with Machine Learning based Detection Framework for defending IoT based Botnet DDoS Attacks". V: *2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI)*. Apr. 2019, str. 1019–1024. DOI: 10.1109/ICOEI.2019.8862720.

[24] Matej Rabzelj in sod. "Designing and Evaluating a Flexible and Scalable HTTP Honeypot Platform: Architecture, Implementation, and Applications". V: *Electronics* 12.16 (2023). DOI: 10.3390/electronics12163480.

Marin Gazvoda de Reggi je študent na Fakulteti za računalništvo in informatiko Univerze v Ljubljani. Zanimajo ga področja razvoja programske opreme, kibernetske varnosti in umetne inteligence. Njegovi raziskovalni interesi zajemajo teorijo programskih jezikov in njihovo varnost.

Sara Mihalič je študentka na Fakulteti za računalništvo in informatiko Univerze v Ljubljani. Navdušujejo jo področja povezana z digitalno forenziko, algoritmi in umetno inteligenco, najbolj pa jo zanima delo na interdisciplinarnih področjih, ki povezujejo tehnologijo z reševanjem konkretnih družbenih izzivov.

Samo Hribar je študent na Fakulteti za računalništvo in informatiko Univerze v Ljubljano. Deluje na področju razvoja mobilnih aplikacij, zanima pa ga tudi hitro rastoče področje umetne inteligence. Na raziskovalnem področju ga zanima varnost aplikacij, od nizkonivojskih do spletnih.

Ana Bračić je študentka na Fakulteti za računalništvo in informatiko Univerze v Ljubljani. Zanimajo jo področja kibernetske varnosti, kriptografije in umetne inteligence. Njeni raziskovalni interesi se osredotočajo na varnostne izzive v digitalnem okolju in uporabo naprednih tehnologij za zagotavljanje varnosti informacijskih sistemov.

Matevž Pesek je izredni profesor in raziskovalec na Fakulteti za računalništvo in informatiko Univerze v Ljubljani, kjer je diplomiral (2012) in doktoriral (2018). Od leta 2009 je član Laboratorija za računalniško grafiko in multimedije. Od leta 2024 izvaja predmet Varnost programov.