Bilježenje aktivnosti osoba tijekom rješavanja CTF zadataka

Prpić, Vjekoslav

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 678

BILJEŽENJE AKTIVNOSTI OSOBA TIJEKOM RJEŠAVANJA CTF ZADATAKA

Vjekoslav Prpić

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 678

BILJEŽENJE AKTIVNOSTI OSOBA TIJEKOM RJEŠAVANJA CTF ZADATAKA

Vjekoslav Prpić

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 4. ožujka 2024.

DIPLOMSKI ZADATAK br. 678

Pristupnik:	**Vjekoslav Prpić (0036520088)**
Studij:	Računarstvo
Profil:	Znanost o mrežama
Mentor:	izv. prof. dr. sc. Stjepan Groš

Zadatak: **Bilježenje aktivnosti osoba tijekom rješavanja CTF zadataka**

Opis zadatka:

Edukcija i trening penetracijskih testera obavlja se najčešće korištenjem različitih CTF zadataka. Pri tome je vrlo bitno imati mogućnost mjerenja kako bi se moglo utvrditi trenutno stanje kompetencija i vještina osoba koje rješavaju CTF zadatke, a temeljem tog pristupa može se onda mjeriti i napredak koji se postiže vježbanjem. To je vrlo složen i ne-tehnički zadatak jer se mjerenjem karakteristika ljudi bavi psihologija. Međutim, tehnički dio mjerenja je prikupljanje odgovarajućih podataka koje će se koristiti da bi se ostvarilo mjerenje. Tu se prvenstveno misli na što detaljnije bilježenje aktivnosti tijekom rješavanja CTF zadataka. U diplomskom radu potrebno je složiti metodu bilježenja što više relevantnih aktivnosti osoba dok rješavaju CTF zadatke. Prije programske implementacije potrebno je definirati što se sve može i treba mjeriti, a temeljem tako definiranih zahtjeva treba razviti pripadajuću programsku podršku. Analizirati moguć utjecaj na privatnost te složiti prijedloge kako bi se mogli prevladati ti negativni utjecaji.

Rok za predaju rada: 28. lipnja 2024.

SADRŽAJ

1. Uvod

U današnjem dobu sigurnost informacijskih sustava vrlo je važna kako bi se organizacije zaštitile od konstantno rastućih i sofisticiranijih kibernetičkih prijetnji. Organizacije moraju osigurati da njihove mreže, podaci i sustavi budu zaštićeni od potencijalnih napada. Penetracijski testeri su bitan dio održavanja sigurnosti sustava. Njihov primarni zadatak je simulirati stvarne napade na sustave kako bi identificirali ranjivosti prije nego što ih zlonamjerni napadači iskoriste.

Edukacija i obuka penetracijskih testera obično se odvija kroz različite *Capture The Flag* (CTF) zadatke. CTF zadaci su zadaci specifično dizajnirani kako bi se penetracijskim testerima omogućilo usavršavanje svojih vještina u sigurnom i kontroliranom okruženju. Pojedini CTF zadaci se bave različitim aspektima sigurnosnog testiranja, uključujući eksploataciju sustava sa specifičnim ranjivostima, forenziku, web sigurnost i druga područja. Glavni cilj ovakvih zadataka nije samo poboljšati tehničke sposobnosti penetracijskih testera već i omogućiti kontinuirano praćenje njihovog napretka.

Obuka penetracijskih testera kroz CTF zadatke ima nekoliko prednosti. CTF zadaci penetracijskim testerima pružaju priliku za vježbanje simulirajući stvarne scenarije, omogućavajući im da steknu iskustvo koje je teško steći kroz teoretsko učenje. Redovito sudjelovanje u CTF natjecanjima omogućuje pentesterima da budu u toku s najnovijim tehnikama i alatima koje koriste napadači. Analizirajući i iskorištavajući ranjivosti u kontroliranom okruženju, pentesteri uče kako prepoznati ranjivosti u stvarnim sustavima. Na taj način CTF zadaci poboljšavaju tehničku sposobnost pentestera.

Mjerenje kompetencija i vještina penetracijskih testera za vrijeme rješavanja CTF zadataka je složeno, dijelom zbog tehničkih zahtjeva prikupljanja i analize podataka, a dijelom i zbog psiholoških aspekata procjene ljudskih karakteristika. Za procjenu trenutnog stanja kompetencija i vještina te praćenje napretka tijekom vremena potrebno je razviti programsku podršku koja omogućuje bilježenje što većeg broja relevantnih aktivnosti penetracijskih testera tijekom rješavanja CTF zadataka.

Tehnički dio aktivnosti zapisivanja ostvaren je korištenjem alata kao što su Filebeat za prikupljanje logova, Elasticsearch za pohranjivanje i pretraživanje prikupljenih podataka te Kibana za vizualizaciju i analizu. Ovi alati omogućuju automatizirano prikupljanje i analizu

velikih količina podataka te tako ostvaruju praćenje napretka u rješavanju CTF zadataka.

Međutim, prikupljanje tako detaljnih podataka o aktivnostima sudionika otvara pitanja privatnosti. Potrebno je analizirati moguće utjecaje na privatnost sudionika i izraditi prijedloge za minimiziranje negativnih učinaka. To uključuje usvajanje načela zaštite privatnosti i anonimnosti podataka gdje je to moguće.

U ovom diplomskom radu pokazat će se način bilježenja relevantnih aktivnosti penetracijskih testera pri rješavanju CTF zadataka, uključujući tehničke aspekte implementacije te analizu utjecaja na privatnost. Cilj rada je razviti programsku podršku koja će omogućiti detaljno praćenje aktivnosti penetracijskog testera.

2. Capture the flag (CTF)

CTF zadaci u računalnoj sigurnosti su vježbe u kojima sudionici pokušavaju pronaći tekstualne nizove, zvane "zastavice", koje su skrivene u namjerno ranjivim programima ili web aplikacijama. Ova vrsta natjecanja može se koristiti u natjecateljske ili obrazovne svrhe.

Postoje dvije glavne vrste CTF natjecanja. U prvoj vrsti sudionici rješavaju izazove koje su postavili organizatori natjecanja. Sudionici se natječu u pronalaženju zastavica koje su skrivene u unaprijed pripremljenim zadacima. Natjecanja mogu uključivati skrivanje zastavica u ranjivim programima, web aplikacijama, kriptografskim tekstovima i mnogobrojnim drugim lokacijama. U drugoj vrsti, sudionici pokušavaju ukrasti zastavice od drugih sudionika, što je poznato kao CTF natjecanja u stilu napada/obrane (*attack-defense* style). U ovoj vrsti natjecanja, timovi moraju braniti svoje sustave dok istovremeno pokušavaju provaliti u sustave drugih timova kako bi ukrali njihove zastavice. [1]

Također popularni format CTF natjecanja je *jeopardy style* CTF natjecanje koje se sastoji od niza izazova podijeljenih u različite kategorije kao što su kriptografija, forenzika i druge koje su navedene kasnije u radu. Svaki izazov ima pridodanu bodovnu vrijednost ovisno o težini zadatka, a sudionici ili timovi biraju koje izazove žele riješiti, pri čemu za svaki uspješno riješen izazov dobivaju odgovarajući broj bodova [2]. Cilj natjecanja je prikupiti što više bodova unutar zadanog vremenskog ograničenja. Ovi izazovi zahtijevaju kombinaciju tehničkih vještina, kreativnog razmišljanja i timskog rada te obuhvaćaju zadatke poput rješavanja šifriranih podataka, analize binarnih datoteka, pronalaženja ranjivosti na web stranicama, analize digitalnih tragova, eksploatacije ranjivosti u programima i rješavanja specifičnih problema programiranjem.

2.1. Vrste CTF zadataka

CTF zadaci pokrivaju širok spektar područja omogućujući sudionicima da se suoče s realnim sigurnosnim izazovima u simuliranom okruženju.

Cryptography zadaci zahtijevaju dešifriranje ili šifriranje poruka, koristeći razne kriptografske algoritme i njihove ranjivosti. Takvi zadaci često uključuju analizu i napad na algoritme poput Blowfish, RSA ili DES, kao i primjenu metoda poput frekventne analize.

Na primjer, zadana je šifrirana poruka korištenjem RSA algoritma. Sudionik mora pronaći privatni ključ kako bi dešifrirao poruku.

Exotic Data Storage uključuje dekodiranje podataka iz neobičnih formata pohrane. Na primjer, datoteka s podacima pohranjena je u vlastitom binarnom formatu. Sudionik mora napisati skriptu koja će dekodirati ovaj format i izvući korisne informacije.

Forensics zadaci bave se analizom digitalnih dokaza za otkrivanje detalja o datotekama i događajima na sustavu, uključujući analizu mrežnog prometa, pregledavanje memorije ili povrat izbrisanih podataka. Na primjer, sudionik dobiva sliku diska i mora pronaći tragove koji ukazuju na vrijeme i način kompromitiranja sustava.

Jail Escaping zahtijeva izlazak iz ograničenih okruženja ili ljuske, pri čemu penetracijski testeri moraju iskoristiti ranjivosti u sustavu kako bi dobili više privilegije ili napustili *sandbox* okruženje.

JavaScript zadaci se odnose na iskorištavanje ranjivosti u JavaScript kodu, uključujući zadatke poput *cross-site scripting* (XSS) napada ili manipulacije DOM-om kako bi se pristupilo skrivenim podacima.

Malware Analysis fokusira se na ispitivanje funkcija i ponašanja zlonamjernog programa.

Pwnage Linux zadaci uključuju eksploataciju Linux sustava, sudionik mora pronaći poznatu ranjivost u zadanoj verziji Linux servera i iskoristiti je kako bi dobio pristup sistemu. *Reverse Engineering* podrazumijeva analizu softvera bez pristupa izvornom kodu, gdje se koriste tehnike poput disasembliranja i dekompiliranja kako bi se otkrila funkcionalnost programa.

Software Defined Radio uključuje manipulaciju radio signalima putem softvera, a *SQL Injection* se fokusira na eksploataciju sigurnosnih propusta u SQL bazama podataka.

Steganography zadaci se bave skrivanjem poruka unutar drugih datoteka ili medija. Na primjer, slika sadrži skrivenu poruku koju sudionik mora pronaći koristeći steganografske tehnike.

Web zadaci uključuju pronalaženje i iskorištavanje slabosti u web aplikacijama, kao što su ranjivosti u autentikaciji, autorizaciji i upravljanju sesijama. [3]

Ovi zadaci omogućuju sudionicima da razvijaju i testiraju svoje vještine u simuliranim uvjetima, pripremajući ih za sigurnosne prijetnje u stvarnom okruženju. Svaka vrsta zadatka pruža specifične izazove koji zahtijevaju raznolike vještine, od tehničkog znanja do kreativnog razmišljanja i brzine rješavanja problema.

U ovom radu je primarno fokusirano na *Web Challenge*, *Eksploataciju*, *Kriptografiju* i *Reverse Engineering* s obzirom da su to najčešće vrste CTF zadataka i najsličnije realnom *pentesting* scenariju.

3. Definiranje zahtjeva za bilježenje aktivnosti

Prilikom rješavanja CTF zadatka penetracijskog testera potrebno je pratiti različite vrste aktivnosti kako bi se mogla evaluirati sposobnost i kompetencija. Ovo poglavlje obuhvaća sve aktivnosti koje se bilježe tijekom CTF natjecanja. Definiranje zahtjeva za bilježenje aktivnosti omogućava nam da identificiramo ključne aspekte rada penetracijskih testera i da osiguramo precizno praćenje njihovih postupaka.

3.1. Početak rješavanja CTF zadatka

Bilježenje vremena kada sudionik započinje rješavanje zadatka, zajedno s informacijama o penetracijskom testeru, poput IP adrese i MAC adrese, potrebno je za određivanje ukupnog vremena provedenog rješavajući CTF zadatak. Ovi podaci omogućuju procjenu učinkovitosti i brzine rješavanja zadataka.

3.2. Modificiranje datoteka

Prikupljanje podataka o promjenama datoteka skripti i programa koje penetracijski tester razvija tijekom CTF natjecanja. Na primjer, kod rješavanja kriptografskih zadataka, penetracijski tester često mora napisati skriptu koja će dekriptirati kriptirani sadržaj. U okviru ovog rada, predviđeno je da se sve dodatno razvijene skripte pohranjuju u unaprijed kreirani direktorij *Workspace* kako bi se optimiziralo prikupljanje podataka i izbjeglo nepotrebno bilježenje velikog broja promjena datoteka.

Detaljno praćenje svih promjena na datotekama uključuje:

- Kreiranje, brisanje i izmjene datoteka.
- Vrijeme izvršenja svake promjene.

Praćenje promjena datoteka omogućava detaljan uvid u proces razvoja i modifikacije programskih rješenja tijekom rješavanja CTF zadataka. Takvo bilježenje može se iskoristiti za analizu učinkovitosti penetracijskog testera u rješavanju složenih problema. Kroz ovakvu

analizu, moguće je identificirati trenutke u kojima tester pokazuje inovativnost i sposobnost prilagodbe, što je često presudno za uspjeh u CTF natjecanjima.

3.3. HTTP zahtjevi

Tijekom rada potrebno je zabilježiti sve HTTP/HTTPS zahtjeve i odgovore koji prolaze kroz Burp Suite, uključujući:

- URL-ove, metode (GET, POST, itd.), tijelo zahtjeva i odgovora.
- Vrijeme slanja zahtjeva i primanja odgovora.
- HTTP zahtjeve različitih modova Burp Suita (Proxy, Intruder, Repeater ...)

HTTP promet koji je generirao penetracijski tester je bitan parametar za praćenje njegovog toka razmišljanja i puta do eksploatacije web aplikacije. U velikoj većini CTF zadataka početna ranjivost, odnosno prvi korak do eksploatiranja ranjivog stroja, je upravo eksploatacija ranjivosti unutar dobivene web aplikacije. Prateći HTTP zahtjeve i odgovarajuće odgovore moguće je utvrditi način kretanja i razmišljanja pentracijskog testera. Također moguće je primjietiti je li propustio neku očitu ranjivost. Na primjer, ako je u HTTP odgovoru vidljivo da je neka komponenta ranjiva na određenu ranjivost, a penetracijski tester je to propustio, to će biti vidljivo u HTTP zapisima.

Zabilježavanje zapisa HTTP prometa ostvareno je zasebno razvijenim dodatkom za Burp Suite.

3.4. Izvršene naredbe

Praćenje svih izvršenih naredbi u terminalu koristan je podatak bilježenja aktivnosti penetracijskog testera tijekom CTF natjecanja. Zapisi izvršenih naredbi uključuju sve naredbe koje penetracijski tester izvršava, zajedno s vremenom izvršetka naredbe, što omogućava kronološku analizu. Također, bilježe se izlazne informacije (*output*) svake naredbe, pružajući uvid u rezultate koje naredbe generiraju. Ovi podaci su važni za mapiranje koraka koje penetracijski tester poduzima. Logovi naredbi također služe za identifikaciju potencijalnih pogrešaka testera, što se kasnije može iskoristiti u određivanju kompetencije.

3.5. Logovi korištenih aplikacija

Bilježenje aktivnosti unutar specifičnih aplikacija koje se koriste tijekom rješavanja zadatka kao što su zapisi alata koji koriste grafičko sučelje čiji ispis i tok nije moguće pratiti. Aktivnost penetracijskog testera unutar takvih aplikacija nije moguće direktno pratiti, ali zajedno

sa ostalim navedenim vrstama zapisa omogućava djelomičnu rekonstrukciju postupaka penetracijskog testera.

3.6. Mrežni promet

Analiza mrežnog prometa uključuje:

- Pakete poslanih i primljenih podataka, pružajući cjelovitu sliku o protoku informacija između različitih sustava.
- Vrijeme slanja svakog paketa
- Detalje o mrežnim protokolima (npr. TCP, UDP)
- IP adrese i portove izvora i odredišta, što omogućava identifikaciju krajnjih točaka mrežne komunikacije.

Potpuni mrežni promet se bilježi koristeći alat `tcpdump`. Kako bi se ostvarilo kompletno bilježenje mrežnog prometa zapisani su svi poslani i primljeni paketi. Ovi podatci nisu slani na ElasticSearch, već su naknadno prezentirani u *pcap* formatu. Kasnije se mogu pregledavati proizvoljnim alatom, na primjer WireShark.

4. Programska podrška

U ovom poglavlju bit će opisani alati i tehnologije korišteni za bilježenje i analizu aktivnosti penetracijskih testera tijekom rješavanja CTF zadataka koji omogućuju prikupljanje, pohranu i vizualizaciju podataka.

Kali Linux, operativni sustav napravljen za penetracijsko testiranje, korišten je kao platforma za izvođenje zadataka. Elasticsearch je korišten za pohranu i pretraživanje velikih količina podataka prikupljenih tijekom zadataka. Kibana, alat za vizualizaciju podataka, omogućuje analizu i prikazivanje podataka pohranjenih u Elasticsearchu kroz interaktivne nadzorne ploče. Filebeat prikuplja logove iz različitih izvora i šalje ih u Elasticsearch, dok Burp Suite, Tcpdump i prilagođene skripte omogućuju detaljno praćenje web prometa, mrežnog prometa, naredbi i promjena datoteka.

4.1. Kali Linux

Kali Linux je operativni sustav prilagođen za potrebe penetracijskog testiranja i računalne forenzike. Temeljen je na Debian distribuciji te dolazi sa unaprijed instaliranim velikim brojem alata potrebnim penetracijskim testerima.[4]

Kali Linux razvija i održava Offensive Security, organizacija poznata po svojim treninzima sigurnosti i certifikacijama. Pokrenut je kao nastavak na prethodni projekt BackTrack. Glavna prednost Kali Linuxa je zbirka alata za testiranje sigurnosti, koja uključuje alate za penetracijsko testiranje, sigurnosnu analizu, forenzičku analizu, reverzno inženjerstvo i brojne druge. Neki od najistaknutijih alata uključuju *Nmap* za skeniranje mreža, *Burp Suite Community* za presretanje HTTP prometa i *Hashcat* za razbijanje lozinki.[5]

Kali Linux u svrhu ovog diplomskog rada služi kao unaprijed konfigurirana virtualna mašina u *Open Virtual Appliance (.ova)* formatu. Virtualna mašina je postavljena kako bi bilježila relevantne zapise aktivnosti penetracijskog testera za vrijeme rješavanja CTF zadatka u skladu sa zahtjevima i implementacijom opisanom u ovom radu.

Slika 4.1: Operativni sustav Kali Linux sa prikazanim kategorijama alata

4.2. Elasticsearch

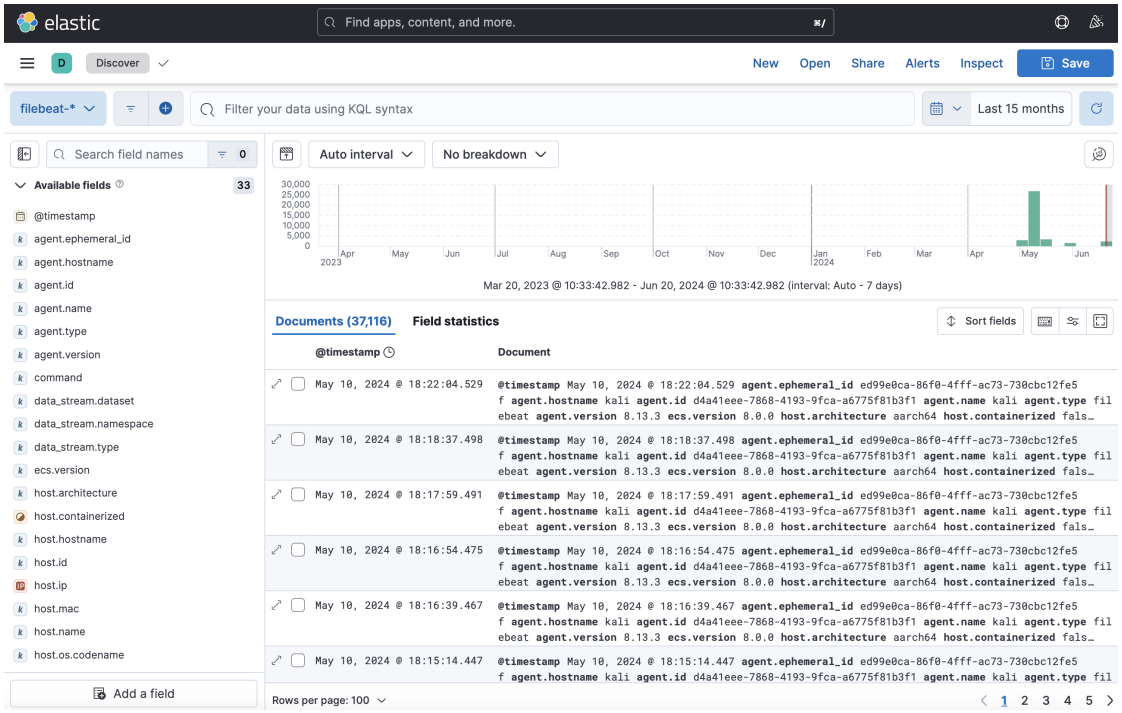
Elasticsearch je sustav otvorenog koda za pretraživanje i analizu podataka. Razvijen je za skalabilnost, fleksibilnost i pretraživanje u stvarnom vremenu[6]. Koristi distribuiranu arhitekturu te podršku za *full-text* pretraživanje koje uključuje ocjenjivanje relevantnosti i složene upite. Elasticseach koristi RESTful API, što omogućuje jednostavnu interakciju i upravljanje koristeći HTTP metode[7]. Integracija sa alatom Kibana omogućuje korištenje grafičkog sučelja za nadzornu ploču i vizualizaciju. Prilikom postavljanja Elasticsearch sustava uzeta je u obzir sigurnost informacija u tranzitu, postavljanjem HTTPS protokola.

Elasticsearch je ključan za prikupljanje i pohranu logova iz različitih izvora tijekom rješavanja CTF zadataka. Omogućava brzo pretraživanje i analizu velikih količina podataka kao što je praćenje aktivnosti penetracijskih testera. Koristeći Filebeat i posebno razvijene skripte za prikupljanje logova i njihovo slanje u Elasticsearch omogućena je obrada podataka u stvarnom vremenu.

4.3. Kibana

Kibana je alat za vizualizaciju podataka unutar Elastic Stacka, omogućujući analizu i vizualizaciju podataka pohranjenih u Elasticsearchu[8]. Namjenjen je za stvaranje interaktivnih nadzornih ploča koje mogu prikazivati različite vrste grafova, poput linijskih, stupčastih i tortnih grafikona što olakšava razumijevanje složenih skupova podataka. Uz podršku za na-

predne funkcije pretraživanja i filtriranja, Kibana omogućuje postavljanje složenih upita i analizu podataka u stvarnom vremenu. Jedan od elementa Kibane, *Discover*, služi za pregledavanje, pretraživanje, filtriranje i sortiranje prikupljenih zapisa prikazan je na slici 4.2



Slika 4.2: Kibana pregled logova

Dashboard modul omogućava kreiranje prilagođene nadzorne ploče koje mogu integrirati različite vizualizacije i metrike na jednom mjestu.

Visualize modul omogućava kreiranje vizualizacije pomoću različitih vrsta grafikona i dijagrama. Vizualizacije su prilagodljive kako bi se prikazali podatci na način koji najbolje odgovara njihovim potrebama. Vizualizacije se mogu temeljiti na različitim vrstama podataka, uključujući vremenske serije, distribucije i korelacije.

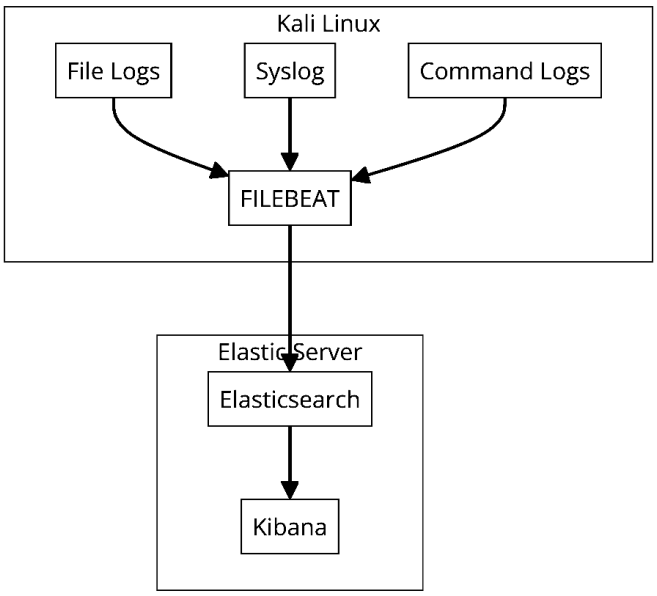
4.4. Filebeat

Filebeat je agent otvorenog koda za prijenos podataka koji je dio Elastic Stacka, dizajniran za slanje i prijenos zapisa i događaja na Elasticsearch[9]. Glavna svrha Filebeata je prikupljanje i prijenos logova iz različitih izvora u stvarnom vremenu čime se olakšava centralizacija i analiza podataka. Filebeat može prikupljati logove iz raznih formata, uključujući logove aplikacija, sistemske logove i prilagođene logove, te ih slati na obradu i pohranu u Elasticsearch.

Postavke korištene za konfiguraciju alata Filebeat u svrhu prikupljanja sistemskih logova vidljive su sljedećem isječku:

```
- type: log
  enabled: true
  paths:
    - /home/kali/.command_output_func.log
- type: filestream
  id: filestream
  enabled: true
  paths:
    - /var/log/*.log
    - /var/log/audit/audit.log
- type: journald
  id: everything
```

Konfiguracija omogućuje alatu Filebeat da bilježi sve sistemske logove, kao i informacije koje je programe penetracijski tester pokrenuo u formatu koji je kasnije jednostavno filtrirati u alatu Kibana. To se prvenstveno odnosi na programe sa grafičkim sučeljem čiji se ispis ne može direktno zabilježiti, te se oslanjamo na same zapise koje program zapisuje u logovima. Arhitektura Elasticsearch sustava prikazana je slici (Slika 4.3).



Slika 4.3: Arhitektura Filebeat sustava

4.5. Praćenje naredbi

Napisana je `bash` skripta za praćenje izvršavanja svih naredbi penetracijskog testera unutar zadane `zsh` ljuske te pohranjena u `.zshrc` datoteku kako bi se svaka izvršena naredba kao i njezin ispis zabilježili te pomoću alata Filebeat zatim poslali na Elasticsearch.

Skripta radi na način da svaku naredbu koju penetracijski tester izvrši u zsh ljusci bilježi zajedno s njenim izlazom i vremenskim oznakama te pohranjuje u log datoteku. Ove informacije se potom procesiraju i šalju u Elasticsearch u JSON formatu.

Ispis 4.1: Praćenje izvršavanja naredbi

```
log_command() {
local start_time=$(date +"%Y-%m-%d_%H-%M-%S")
local log_file="$HOME/.command_output_${start_time}.log"
local command="$*"
```

U prvom dijelu skripte, definira se putanja do privremene log datoteke gdje će se pohraniti ispis naredbe te se spremi sama naredba.

Ispis 4.2: Izvršavanje script naredbe

```
script -q "$log_file" -c "$command"
sleep 1
```

Naredba `script` koristi se za bilježenje cjelokupnog ispisa naredbe u privremenu log datoteku. `script` je standardna Unix naredba koja bilježi sve što se prikazuje na terminalu. U ovom slučaju, koristi se s opcijom *-q* za tihi način rada, što znači da neće prikazivati dodatne informacije na terminalu, te s opcijom *-c* koja definira koju naredbu treba izvršiti i bilježiti njen izlaz. Kratka pauza (*sleep 1*) omogućuje preciznije bilježenje završetka interaktivnih programa.

Ispis 4.3: Formiranje JSON objekta

```
local json_output
json_output=$(cat "$log_file" | sed 's/"/\"/g' | tr '\n' '\n' | sed
    -r "s/\x1B[([0-9]{1,2}(;[0-9]{1,2})?)?[m|K]//g")
```

Ispis naredbe se obrađuje kako bi se uklonili posebni znakovi i formatira se u JSON oblik prikladan za pregledavanje u alatu Kibana. Naredba `sed` uklanja sve dvostruke navodnike i specijalne znakove, dok naredba `tr` zamjenjuje znakove novog reda
`n`.

Ispis 4.4: Pohrana u log datoteku

```
printf '{"timestamp": "%s", "command": "%s", "output": "%s"}\n'
"$timestamp" "$(echo "$command" | sed 's/"/\"/g')" "$json_output"
   >> "$HOME/.command_output_func.log"
rm -f "$log_file"
}
```

Konačno, obrađeni podaci se zapisuju u log datoteku u JSON formatu te se privremena log datoteka uklanja.

Pomoću alata Filebeat, prikupljeni logovi se automatski šalju na Elasticsearch server. Na taj način, omogućeno je detaljno praćenje aktivnosti i naredbi koje penetracijski tester pokreće tijekom rješavanja CTF zadataka.

Primjer izvršavanja naredbe `nmap` i njen ispis u JSON formatu prikupljen iz alata Kibana koji je izvršio penetracijski tester u početnoj fazi otkrivanja CTF zadatka prikazan je u isječku (Ispis 4.5).

Ispis 4.5: JSON zapis izvršene naredbe

```
{
    "timestamp": "2024-06-17 12:43:42",
    "command": "nmap 192.168.0.3",
    "output": "Script started on 2024-06-17 12:43:40-04:00
    [COMMAND=\"nmap 192.168.0.3\"
    TERM=\"xterm-256color\"
    TTY=\"/dev/pts/0\"
    COLUMNS=\"236\"
    LINES=\"61\"]
    \Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-06-17
    12:43 EDT
    Nmap scan report for 192.168.0.3
    \Host is up (0.016s latency).
    \Not shown: 998 closed tcp ports (conn-refused)
    \PORT   STATE SERVICE
    \22/tcp open  ssh
    \80/tcp open  http
    \
    \Nmap done: 1 IP address (1 host up) scanned in 0.39 seconds
    \\Script done on 2024-06-17 12:43:41-04:00 }@*)
    [COMMAND_EXIT_CODE=\"0\"]\"}
}
```

Primjer završnog koraka eksploatacije mašine prijavljivanjem korisnika u SSH ljusku, te pronalazak tražene zastavice je prikazano u ispisu (Ispis 4.6).

Ispis 4.6: SSH pronalazak korisničke zastavice

```
{"timestamp": "2024-06-17 20:18:44", "command": "ssh losy@192
   .168.0.3", "output": "Script started on 2024-06-17
   20:18:28-04:00 [COMMAND=\"ssh losy@192.168.0.3\" TERM=\"xterm
   -256color\" TTY=\"/dev/pts/0\" COLUMNS=\"235\" LINES=\"61\"]\
losy@192.168.0.3's password:
\Welcome to Ubuntu 20.04.3 LTS (GNU/Linux 5.4.0-81-generic x86_64)
```

```
\
\ * Documentation:  https://help.ubuntu.com
\ * Management:     https://landscape.canonical.com
\ * Support:        https://ubuntu.com/advantage
\
\  System information as of Tue 18 Jun 2024 12:18:32 AM UTC
\
\  System load:  0.08               Processes:                198
\  Usage of /:   52.8% of 12.73GB   Users logged in:          1
\  Memory usage: 38%                IPv4 address for enp0s17:
   192.168.0.3
\  Swap usage:   0%
\Last login: Mon Jun 17 16:39:55 2024 from 192.168.0.106

\]0;losy@darkhole: ~losy@darkhole:~$ ls
\user.txt
\]0;losy@darkhole: ~losy@darkhole:~$ cat user.txt
\DarkHole{'This_is_the_life_man_better_than_a_cruise'}
\
\]0;losy@darkhole: ~losy@darkhole:~$ exit
\logout
\Connection to 192.168.0.3 closed.

\\Script done on 2024-06-17 20:18:43-04:00 [COMMAND_EXIT_CODE=\"0\"
   ]\"}
```

4.6. Praćenje promjena datoteka

Praćenje promjena datoteka dokumentira razvoj dodatnih alata i skripti koje penetracijski tester piše tijekom rješavanja CTF zadataka. Neki CTF zadaci zahtijevaju razvoj posebnih skripti za rješavanje. Najčešći oblik CTF zadatka koji zahtijeva takav pristup su kriptografski zadaci. U sklopu ovog diplomskog rada razvijena je `Python` skripta koja detektira i bilježi svaku promjenu određenih datoteka. Skripta pretpostavlja da penetracijski tester pohranjuje datoteke u prethodno postavljen direktorij `Workspace`. Ovaj pristup optimizira praćenje i pohranu podataka u Elasticsearch.

Skripta koristi biblioteku `watchdog` za praćenje promjena u zadanom direktoriju te biblioteku `elasticsearch` za pohranu podataka u Elasticsearch. Skripta definira klasu `FileChangeHandler` koja obrađuje različite događaje kao što su kreiranje, modifikacija,

brisanje i premještanje datoteka. U metodi `process`, događaj se zapisuje u log zapis s tipom događaja, putanjom do datoteke i trenutnim vremenom. Ako je datoteka modificirana, njen sadržaj se pokušava pročitati i dodati u zapis.

Ispis 4.7: Metoda FileChangeHandler klase

```
class FileChangeHandler(FileSystemEventHandler):
    def process(self, event):

        log_entry = {
            'event_type': event.event_type,
            'src_path': event.src_path,
            'timestamp': time.strftime('%Y-%m-%dT%H:%M:%SZ', time.
               gmtime())
        }

        if event.event_type == 'modified':
            try:
                with open(event.src_path, 'r') as file:
                    log_entry['content'] = file.read()
            except Exception as e:
                log_entry['error'] = str(e)

        es.index(index=index_name, document=log_entry)
```

Metoda process unutar klase `FileChangeHandler` obrađuje događaje promjene datoteka. Ako je datoteka modificirana, njen sadržaj se čita i dodaje u zapis. Zapis se zatim šalje u Elasticsearch indeks.

Ispis 4.8: Metode promjene

```
    def on_modified(self, event):
        if not event.is_directory:
            self.process(event)

    def on_created(self, event):
        if not event.is_directory:
            self.process(event)

    def on_deleted(self, event):
        if not event.is_directory:
            self.process(event)
```

```
    def on_moved(self, event):
        if not event.is_directory:
            self.process(event)
```

Metode unutar FileChangeHandler klase prikazane u ispisu (Ispis 4.8) definiraju kako se obrađuju različiti tipovi događaja: kreiranje, modifikacija, brisanje i premještanje datoteka. Svaka metoda poziva process metodu za daljnju obradu događaja.

Ispis 4.9: Metoda main

```
if __name__ == "__main__":
    directory_to_watch = '/home/kali/Desktop/Workspace/'
    event_handler = FileChangeHandler()
    observer = Observer()
    observer.schedule(event_handler, path=directory_to_watch,
       recursive=True)
    observer.start()

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

Glavni dio skripte prikazan u ispisu (Ispis 4.9) postavlja *Observer* koji prati zadani direktorij i koristi `FileChangeHandler` za obradu događaja. Skripta se vrti u beskonačnoj petlji. Skripta omogućuje prikupljanje detaljnih informacija o promjenama datoteka, uključujući tip događaja, putanju do datoteke i sadržaj datoteke. Ovo omogućuje detaljno praćenje aktivnosti penetracijskih testera tijekom rješavanja CTF zadataka. Primjer zapisa zabilježenog stvaranja datoteke prikazan je u ispisu (Ispis 4.10.)

Ispis 4.10: Primjer zapisa promjene datoteka


```
{
  "_index": "file_changes",
  "_id": "wA3fN5ABQOUE7lrDV6D2",
  "_version": 1,
  "_score": 0,
  "_source": {
    "event_type": "modified",
    "src_path": "/home/kali/Desktop/Workspace/track.txt",
    "timestamp": "2024-06-10T21:55:46Z",
```

```
    "content": "import base64\n\nXorflag = base64.b64decode('
       RU9CRC43aWdxNDsxaWtiNTFpYk9PMDs6NDFS')\nflag=''\n\nfor i in
       Xorflag:\n    flag = flag + chr(ord(i) ^ 3)\n\nprint(flag)\n
       "
  },
  "fields": {
    "content.keyword": [
      "import base64\n\nXorflag = base64.b64decode('
         RU9CRC43aWdxNDsxaWtiNTFpYk9PMDs6NDFS')\nflag=''\n\nfor i
         in Xorflag:\n    flag = flag + chr(ord(i) ^ 3)\n\nprint(
         flag)\n"
    ],
    "event_type": [
      "modified"
    ],
    "src_path.keyword": [
      "/home/kali/Desktop/Workspace/track.txt"
    ],
    "event_type.keyword": [
      "modified"
    ],
    "src_path": [
      "/home/kali/Desktop/Workspace/track.txt"
    ],
    "content": [
      "import base64\n\nXorflag = base64.b64decode('
         RU9CRC43aWdxNDsxaWtiNTFpYk9PMDs6NDFS')\nflag=''\n\nfor i
         in Xorflag:\n    flag = flag + chr(ord(i) ^ 3)\n\nprint(
         flag)\n"
    ],
    "timestamp": [
      "2024-06-10T21:55:46.000Z"
    ]
  }
}
```

4.7. Detektiranje uspostavljenih konekcija

Detektiranje uspostavljenih konekcija ostvareno je kreiranjem Bash skripte koja u zadanom vremenskom intervalu bilježi uspostavljene konekcije Kali Linux stroja penetracijskog testera. Ovakvi zapisi su relevantni jer nam omogućuju prikaz uspostave konekcija sa strojem kojega se pokušava eksploatirati. Ako penetracijski tester uspije otvoriti reverznu ljusku prema ranjivom stroju na primjer, to će biti vidljivo pomoću ovakvih zapisa.

Skripta koristi naredbu `lsof` za popis otvorenih konekcija. Njena opcija `-iTCP` filtrira prikaz samo na TCP veze, dok `-sTCP:ESTABLISHED` osigurava da se prikažu samo veze koje su trenutno uspostavljene. Opcija `-nP` sprječava pretvaranje IP adresa u host imena, čime se povećava brzina izvršavanja skripte. Za izdvajanje specifičnih informacija iz svakog zapisa koristi se naredba `awk`, a naredbom `curl -X POST` se šalje kreirani zapis u JSON formatu u Elasticsearch bazu.

Varijable *ES_HOST* i *ES_PORT* definiraju adresu Elasticsearch servera, dok *INDEX_NAME* označava naziv indeksa u kojem će podaci biti pohranjeni. *INTERVAL* određuje vremenski interval (u sekundama) između provjera. Skripta radi u beskonačnoj petlji, periodički prikupljajući trenutne TCP veze, uspoređujući ih s prethodnim stanjem (*LAST_CONNECTIONS*). Ako su se veze promijenile, izrađuje se JSON objekt s detaljima svake veze te se šalje na Elasticsearch server pomoću naredbe *curl*.

Skripta započinje inicijalizacijom varijabli koje definiraju Elasticsearch server i ostale varijable korištene kroz skriptu. Nakon toga, ulazi se u beskonačnu petlju koja periodički prikuplja informacije o trenutnim TCP vezama.

Ispis 4.11: Prikupljanje uspostavljenih konekcija

```
while true; do
    CURRENT_CONNECTIONS=$(sudo lsof -iTCP -sTCP:ESTABLISHED -nP)

    if [[ "$CURRENT_CONNECTIONS" != "$LAST_CONNECTIONS" ]]; then
        echo "$CURRENT_CONNECTIONS" | while read -r connection; do
            COMMAND=$(echo "$connection" | awk '{print $1}')
            PID=$(echo "$connection" | awk '{print $2}')
            USER=$(echo "$connection" | awk '{print $3}')
            FD=$(echo "$connection" | awk '{print $4}')
            TYPE=$(echo "$connection" | awk '{print $5}')
            DEVICE=$(echo "$connection" | awk '{print $6}')
            SIZE_OFF=$(echo "$connection" | awk '{print $7}')
            NODE=$(echo "$connection" | awk '{print $8}')
            NAME=$(echo "$connection" | awk '{print $9}')
```

```
            TIMESTAMP=$(date -u +"%Y-%m-%dT%H:%M:%SZ")
```

U svakoj iteraciji petlje prikupljaju se informacije o trenutno uspostavljenim TCP vezama koristeći `lsof` naredbu. Ako su trenutne veze različite od prethodnih, skripta započinje obradu svake veze pojedinačno. Korištenjem `awk` naredbe, skripta izdvaja specifične informacije iz svake veze.

Ispis 4.12: Formiranje JSON objekta

```
            JSON_DATA=$(cat <<EOF
{
    "timestamp": "$TIMESTAMP",
    "command": "$COMMAND",
    "pid": "$PID",
    "user": "$USER",
    "fd": "$FD",
    "type": "$TYPE",
    "device": "$DEVICE",
    "size_off": "$SIZE_OFF",
    "node": "$NODE",
    "name": "$NAME"
}
EOF
            )
```

Iz prikupljenih podataka kreira se JSON objekt.

Korištenjem naredbe `curl`, JSON objekt se šalje na Elasticsearch. Nakon toga, skripta postavlja varijablu LAST_CONNECTIONS i prekine se na prethodno postavljeni interval prije nego što ponovo provjeri uspostavljene konekcije.

Ispis 4.13: Slanje JSON objekta na Elasticsearch

```
            curl -X POST "http://$ES_HOST:$ES_PORT/$INDEX_NAME/_doc
               " \
                 -H 'Content-Type: application/json' \
                 -d "$JSON_DATA"
        done

        LAST_CONNECTIONS="$CURRENT_CONNECTIONS"
    fi

    sleep $INTERVAL
done
```

Ova skripta omogućuje prikupljanje detaljnih informacija o trenutnim TCP vezama, uključujući naziv naredbe koja je otvorila vezu, PID (*Process ID*) procesa, korisnika koji je pokrenuo proces, datoteka deskriptor (FD), tip veze, uređaj, veličinu/pomak, čvor i naziv veze čime se ostvaruje detaljno praćenje mrežne aktivnosti penetracijskih testera tijekom rješavanja CTF zadataka. Primjer zapisa uspostavljenih konekcija prikazan je u ispisu (Ispis 4.14).

Ispis 4.14: Primjer zapisa uspostavljenih konekcija


```
{"_index":"outbound_connections","_id":"QnaTJZABu1Br0I_h_wzV","
   _version":1,"_score":0,"_source":{"timestamp":"2024-06-17T09
   :42:53Z","command":"java","pid":"1805","user":"kali","fd":"41u",
   "type":"IPv6","device":"12329","size_off":"0t0","node":"TCP","
   name":"10.0.2.15:39293->10.8.0.2:6002"},"fields":{"device.
   keyword":["12329"],"node.keyword":["TCP"],"command.keyword":["
   java"],"name.keyword":["10.0.2.15:39293->10.8.0.2:6002"],"pid":[
   "1805"],"type":["IPv6"],"user.keyword":["kali"],"command":["java
   "],"fd.keyword":["41u"],"node":["TCP"],"size_off":["0t0"],"pid.
   keyword":["1805"],"type.keyword":["IPv6"],"name":["
   10.0.2.15:39293->10.8.0.2:6002"],"device":["12329"],"size_off.
   keyword":["0t0"],"user":["kali"],"fd":["41u"],"timestamp":["
   2024-06-17T09:42:53.000Z"]}}
```


4.8. Dodatak za alat Burp Suite

Burp Suite je program koji je razvio PortSwigger namijenjen penetracijskim testerima za testiranje sigurnosti web aplikacija koji omogućuje analizu, otkrivanje i eksploataciju ranjivosti unutar web aplikacija. Burp Suite pruža niz funkcionalnosti uključujući *proxy* za presretanje i modifikaciju HTTP/HTTPS prometa, skener ranjivosti, alat za napade (*Intruder*), *Repeater* za ponavljanje i modifikaciju HTTP zahtjeva te ostale korisne alate [10]. U kontekstu CTF zadataka, Burp Suite je koristan alat za eksploataciju CTF zadataka baziranih na eksploataciji web aplikacija.

S obzirom da uz već spomenuti Kali Linux operativni sustav dolazi instaliran i Burp Suite Community Edition, koji je besplatna varijanta Burp Suite programa, razvijen je dodatak za bilježenje HTTP/HTTPS prometa kao još jedan od korisnih zapisa za bilježenje aktivnosti.

Burp Suite podržava BurpExtender API koji omogućuje razvijanje i integriranje vlastitih ekstenzija pisanih u programskim jezicima Java, Python i Ruby. Ekstenzije napisane u Pythonu oslanjaju se na Jython, implementaciju Pythona za Java platformu, jer je Burp Suite pisan u programskom jeziku Java. Ova mogućnost značajno proširuje funkcionalnost alata, omogućujući korisnicima da prilagode Burp Suite specifičnim potrebama svojih zadataka ili

dodaju nove funkcionalnosti koje nisu dostupne u osnovnoj verziji alata[11].

Razvijena ekstenzija radi na način da presreće sve HTTP zahtjeve i odgovore koji prolaze kroz Burp Suite, prikuplja relevantne podatke, formatira ih u JSON oblik i šalje u Elasticsearch. Na taj način omogućuje detaljnu analizu mrežnog prometa što je korisno za razumijevanje koraka koje sudionici poduzimaju tijekom rješavanja CTF zadataka.

Kao preduvjet razvoja dodatka je definiranje klase ekstenzije koja implementira potrebna sučelja IBurpExtender i IHttpListener. Sučelje IBurpExtender omogućuje ekstenziji da se registrira s Burp Suiteom, dok sučelje IHttpListener omogućuje ekstenziji da presreće i obrađuje HTTP/HTTPS promet.

Nakon inicijalizacije, potrebno je definirati metodu za procesiranje HTTP poruka. Ova metoda presreće HTTP zahtjeve i odgovore, analizira ih i priprema podatke za slanje u Elasticsearch.

Ispis 4.15: Metoda za procesiranje HTTP poruka

```
def processHttpMessage(self, toolFlag, messageIsRequest,
   messageInfo):
        try:
            timestamp = datetime.datetime.utcnow().isoformat() + 'Z
               '
            if messageIsRequest:
                # Process request
                self._stdout.println("Processing request")
                request = messageInfo.getRequest()
                request_info = self._helpers.analyzeRequest(request
                   )
                headers = request_info.getHeaders()
                body = request[request_info.getBodyOffset():].
                   tostring()
```

U ovom dijelu *processHttpMessage* metoda prikuplja informacije o HTTP zahtjevu, uključujući zaglavlja i tijelo zahtjeva. Zaglavlja sadrže ključne informacije kao što su HTTP metoda (npr. GET, POST) i putanja URL-a.

Ispis 4.16: Metoda za procesiranje HTTP poruka

```
request_line = headers[0]
method, path, _ = request_line.split(' ', 2)
http_service = messageInfo.getHttpService()
scheme = 'https' if http_service.getPort() == 443 else 'http'
url = "{}://{}:{}{}".format(scheme,
                        http_service.getHost(),
```

```
                        http_service.getPort(), path)
data = {
    'tool': self._callbacks.getToolName(toolFlag),
    'type': 'request',
    'timestamp': timestamp,
    'url': url,
    'headers': list(headers),
    'body': body
}
self._stdout.println("Request data prepared: " + json.dumps(data))
```

U isječku (Ispis 4.16) se iz zaglavlja zahtjeva izvlače metoda i putanja URL-a, te se konstruira puni URL koristeći informacije o HTTP servisu. Prikupljeni podaci formatiraju se u JSON objekt prikladan za Elastcsearch.

Sličan postupak koristi se za procesiranje HTTP odgovora. Kod procesiranja HTTP odgovora, prikupljaju se informacije kao što su statusni kod, zaglavlja i tijelo odgovora te se ti podaci također formatiraju u JSON objekt.

Na kraju, potrebno je definirati metodu za slanje prikupljenih podataka u Elasticsearch. Ova metoda koristi *requests* biblioteku za slanje HTTP POST zahtjeva na Elasticsearch server, osiguravajući da su podaci pravilno pohranjeni.

Ispis 4.17: Metoda za slanje JSON objekta na Elasticsearch

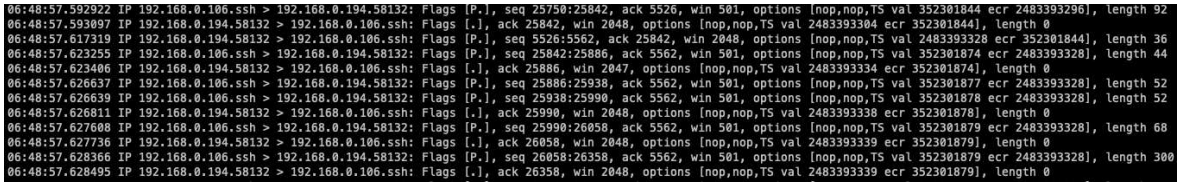
```
def send_to_elasticsearch(self, data):
    try:
        headers = {'Content-Type': 'application/json'}
        response = requests.post(self.es_url, data=json.dumps(data)
           , headers=headers)
        self._stdout.println("Elasticsearch response status: {}".
           format(response.status_code))
        if response.status_code not in [200, 201]:
            self._stderr.println("Failed to send data to
               Elasticsearch: " + response.text)
    except Exception as e:
        self._stderr.println("Error sending data to Elasticsearch:
           " + str(e))
```

Ova metoda formatira podatke kao JSON i šalje ih na definirani URL Elasticsearch servera.

4.9. Tcpdump

Tcpdump je alat otvorenog koda za presretanje mrežnih paketa koji omogućava korisnicima da pregledavaju i analiziraju mrežni promet u stvarnom vremenu kao i naknadno. Alat omogućuje filtriranje paketa prema različitim kriterijima kao što su IP adrese, portovi i protokoli. Tcpdump se najčešće koristi na Unix/Linux operativnim sustavima i može se pokrenuti iz komandne linije, pružajući detaljan uvid u mrežni promet.

U kontekstu ovog diplomskog rada, tcpdump je korišten kao dodatni alat za prikupljanje mrežnog prometa, osiguravajući pokrivenost i za slučaj da ostali mehanizmi prikupljanja podataka propuste neke aspekte mrežne aktivnosti. Korištenjem tcpdump-a omogućeno je potpuno bilježenje mrežnog prometa, uključujući sve poslane i primljene pakete, čime se stvara sveobuhvatan zapis za detaljnu retrospektivnu analizu. Snimljeni podaci nisu odmah poslani na ElasticSearch, već su pohranjeni u *pcap* formatu, omogućujući kasniju analizu pomoću alata kao što je Wireshark. Time je osigurano temeljito praćenje i analiza mrežnih aktivnosti penetracijskog testera tijekom CTF natjecanja, pružajući sveobuhvatan uvid u njegove metode i strategije. Na slici (Slika 4.4) je prikazan primjer mrežnog prometa snimljen alatom tcpdump.



Slika 4.4: Primjer mrežnog prometa

5. Utjecaj na privatnost

Privatnost je vrlo bitna stavka na koju je potrebno obratiti pozornost prilikom prikupljanja znatne količine podataka. Ovi podatci se koriste kako bi se analizirala i ocjenjivala sposobnost penetracijskih testera za vrijeme rješavanja CTF zadataka. Obzirom na prirodu prikupljanja podataka, anonimnost nije izvediva. Anonimni podatci ne bi omogućili relevantne zaključke. Stoga je važno implementirati stroga pravila o zaštiti podataka.

Pravila o zaštiti podataka trebala bi definirati tko ima pristup prikupljenim podacima. Također je potrebno odrediti koliko dugo se podatci čuvaju i pod kojim uvjetima se mogu koristiti. Pristup podacima treba biti ograničen samo na ovlaštene osobe koje su direktno uključene u analizu i procjenu vještina penetracijskih testera. Važna je transparentnost u načinu prikupljanja i korištenja podataka. Sudionici moraju biti jasno informirani o vrsti podataka koja se prikuplja. Također, moraju biti obaviješteni kako će se podatci koristiti.

Potrebno je osigurati da sudionici daju informirani pristanak prije sudjelovanja u bilo kakvim aktivnostima koje uključuju prikupljanje podataka. Enkripcija podataka u tranzitu i skladištenju osigurava zaštitu od neovlaštenog pristupa. Korištenje sigurnosnih protokola smanjuje rizik od kompromitiranja podataka.

Regulacije poput Opće uredbe o zaštiti podataka (GDPR) u Europskoj uniji postavljaju stroge zahtjeve za prikupljanje i obradu osobnih podataka [12]. Nepoštivanje ovih regulativa može rezultirati ozbiljnim pravnim posljedicama. Stoga je potrebno osigurati da svi aspekti prikupljanja i obrade podataka budu u skladu s važećim zakonima i etičkim smjernicama.

Primjenom navedenih mjera i pristupa moguće je znatno smanjiti negativne utjecaje na privatnost. Također, moguće je osigurati da prikupljanje podataka tijekom rješavanja CTF zadataka bude etički i pravno prihvatljivo. Za budući rad potrebno je izraditi pravni okvir te kreirati dokument koji bi penetracijski tester i izvršavatelj analize podataka trebali potpisati. Na taj način bi se obvezali na zaštitu privatnosti prikupljenih podataka.

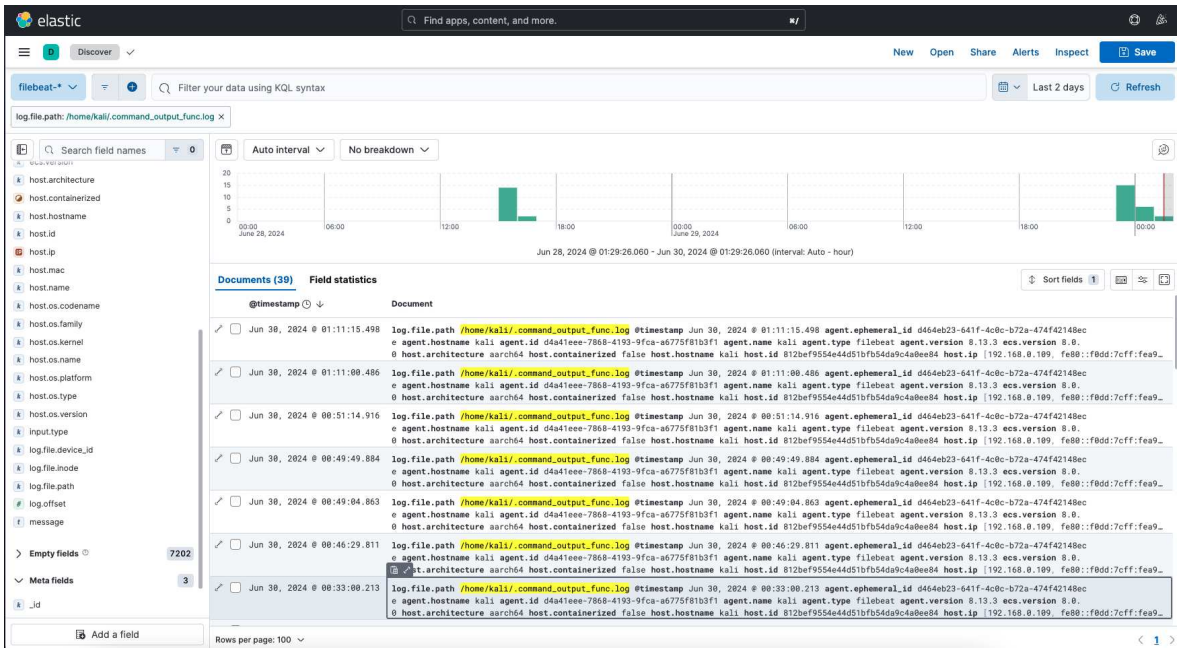
6. Praktični primjeri rješavanja CTF zadatka

U ovom poglavlju je prikazano praćenje aktivnosti rješavanja praktičnih primjera eksploatacijskog i kriptografskog CTF zadatka. Bit će demonstrirano koje je korake izvršio penetracijski tester kako bi došao do rješenja, odnosno zastavica.

6.1. Eksploatacija ranjivog stroja

Primjer eksploatacijskog CTF zadatka je ranjivi stroj pod nazivom *DarkHole: 2* preuzet sa stranice *Vulnhub*. *Vulnhub* je platforma koju sponzorira kompanija *OffSec* te služi kao kolekcija ranjivih strojeva namijenjenih za rješavanje CTF zadataka.

Ranjivi stroj *DarkHole: 2* unaprijed je konfiguriran sa ranjivostima koje je potrebno iskoristiti kako bi se došlo do zastavica. Stroj je podijeljen u dvije razine rješenja - korisnička (*user*) zastavica i administratorska (*root*) zastavica. Koraci za iskorištavanje ranjivosti će biti prikazani u nastavku zajedno sa prikupljenim podacima penetracijskog testera. Na slici (Slika 6.1) je prikazano sučelje Kibane filtrirano da prikazuje zabilježene naredbe sortirane u vremenu.



Slika 6.1: Prikaz naredbi u vremenu

Naredba s kojom penetracijski tester započinje rješavanje CTF zadatka je skeniranje otvorenih vrata *IP* adrese ranjivog stroja alatom *nmap* u čijem ispisu pronalazi `.git` direktorij na vratima 80 i *ssh* protokol na vratima 22. Naredba je prikazana u ispisu (Ispis 6.1).

Ispis 6.1: Naredba nmap

```
{"timestamp": "2024-06-29 17:34:34", "command": "nmap 192.168.0.3
   -A", "output":
\[...]
\PORT   STATE SERVICE VERSION
\22/tcp open ssh     OpenSSH 8.2p1 Ubuntu 4ubuntu0.11 (Ubuntu Linux
   ; protocol 2.0)
\| ssh-hostkey:
\|   3072 57:b1:f5:64:28:98:91:51:6d:70:76:6e:a5:52:43:5d (RSA)
\|   256 cc:64:fd:7c:d8:5e:48:8a:28:98:91:b9:e4:1e:6d:a8 (ECDSA)
\|_  256 9e:77:08:a4:52:9f:33:8d:96:19:ba:75:71:27:bd:60 (ED25519)
\80/tcp open  http    Apache httpd 2.4.41 ((Ubuntu))
\| http-git:
\|   192.168.0.3:80/.git/
\|     Git repository found!
\|     Repository description: Unnamed repository; edit this file '
   description' to name the...
\|_    Last commit message: i changed login.php file for more
   secure
```

```
\| http-cookie-flags:
\|   /:
\|     PHPSESSID:
\|_      httponly flag not set
\|_http-title: DarkHole V2
\|_http-server-header: Apache/2.4.41 (Ubuntu)
\[...]}
```

Zatim koristeći alat `git_dumper.py` preuzima repozitorij te pregledava promjene naredbama `git log` i `git diff` pomoću kojih dolazi do inicijalnih vjerodajnica.

Ispis 6.2: Naradbe pregledavanje git repozitorija

```
{"timestamp": "2024-06-29 17:37:48", "command":  "python3
   git_dumper.py http://192.168.0.3/.git/
   git_dump", "output": [...]}

{"timestamp": "2024-06-29 17:38:02", "command":  "git
   log", "output":
\[...]
\
\commit a4d900a8d85e8938d3601f3cef113ee293028e10
\Author: Jehad Alqurashi <anmar-v7@hotmail.com>
\Date:   Mon Aug 30 13:06:20 2021 +0300
\
\    I added login.php file with default credentials
\
\[...]}

{"timestamp": "2024-06-29 17:38:12", "command":  "git diff
   a4d900a8d85e8938d3601f3cef113ee293028e10", "output": "
diff --git a/login.php b/login.php
\index 8a0ff67..0904b19 100644
\--- a/login.php
\+++ b/login.php
\[...]
 if($_POST['email'] == "lush@admin.com"  $_POST['password'] ==
   "321")
\[...]
}
```

Prijavu na web aplikaciju sa vjerodajnicama iz ispisa (Ispis 6.2) moguće je vidjeti u

HTTP prometu prikazanom na slici (Slika 6.2) koji je zabilježen dodatkom za *Burp Suite*.

Slika 6.2: Prijava u web aplikaciju

Iskorištavanje *SQL injection* ranjivosti ranjivog *URL* parametra *id* koristeći alat `sqlmap` dobivene su vjerodajnice koje su iskorištene za prijavu putem `ssh` protokola.

Povezivanje na *ssh* poslužitelj vidljivo je i putem nadziranja odlaznih konekcija:

k _index	outbound_connections
# _score	4.054
t command	ssh
t device	46189
t fd	3u
t name	192.168.0.109:37432->192.168.0.3:22
t node	TCP
t pid	14568
t size_off	0t0
timestamp	Jun 29, 2024 @ 23:47:59.000
t type	IPv4
t user	kali

Slika 6.3: SSH konekcija

Pronalazi *cronjob* koji se pokreće kao korisnik *losy* u kojem se lokalno na vratima 9999 pokreće *php* aplikacija sa ranjivim parametrom `cmd`.

Ispis 6.3: Cronjob popis

```
jehad@darkhole:~$ cat /etc/crontab
[...]
\* * * * * losy  cd /opt/web && php -S localhost:9999
[...]
```

Uspješno iskorištava *RCE (Remote code execution)* ranjivost te pokreće *reverse shell* kao korisnik *losy* te dolazi do prve *user* zastavice.

Ispis 6.4: User zastavica

```
{"timestamp": "2024-06-29 17:49:22", "command": "nc -lvnp 1234", "
   output": "
[...]
\$ cat user.txt
\DarkHole{'This_is_the_life_man_better_than_a_cruise'}
```

Do konačne *root* zastavice dolazi iskorištavanjem prava korisnika *losy* iz grupe *sudo* kako bi pokrenuo *python* naredbu kao *root* korisnik naredbom `sudo python -c 'import os; os.system(¨/bin/sh¨)'`.

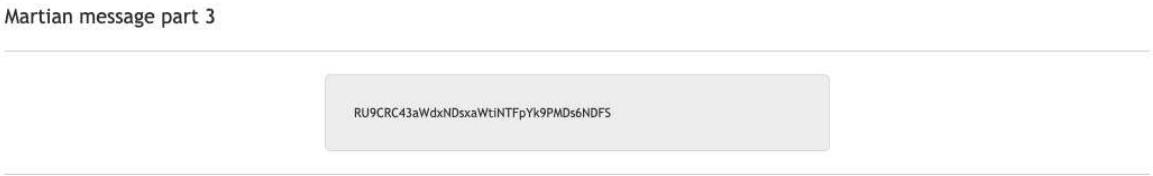
Ispis 6.5: Root zastavica

```
root@darkhole:~# cat root.txt
\DarkHole{'Legend'}
```

Praćenjem rada penetracijskog testera dobivamo uvid u poduzete korake za rješavanje CTF zadatka. Uz ispise naredbi te preostale prikupljene podatke moguće je zaključiti jesu li propuštene ranjivosti te vrijeme koje je potrebno penetracijskom testeru za uspješno detektiranje i iskorištavanje pojedinačnih ranjivosti.

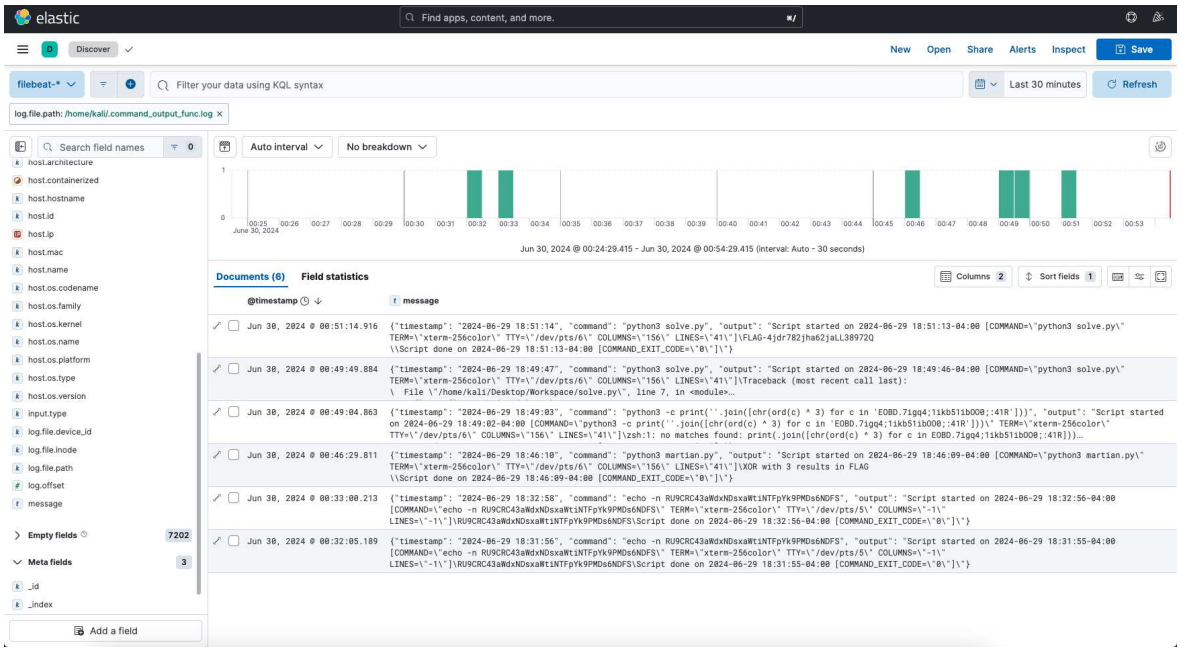
6.2. Kriptografski CTF zadatak

Primjer kriptografskog CTF zadatka naziva *Martian message part 3* na *RingZer0 Team Online CTF* [13] platformi koji penetracijskom testeru zadaje niz znakova bez dodatnog konteksta iz kojeg je potrebno dešifrirati zastavicu. Zadatak je prikazan na slici (Slika 6.4).



Slika 6.4: RingZero Kriptografski CTF zadatak

Rješenje zadatka uključuje dekodiranje zadanog niza znakova iz *base64* formata te izvođenje operacije *XOR* sa `0x03` nad dobivenim nizom. Relevantni podaci prikupljeni za CTF zadatak su naredbe i promjene datoteka koje je penetracijski tester izvršavao.



Slika 6.5: Izvršene naredbe

Iz prikupljenih podataka moguće je primijetiti kako je penetracijski tester kao prvu naredbu pokušao dekodirati niz znakova kao *base32* format, zatim kao *base64* format. Potom kreira skriptu koja iskorištava poznavanje formata zastavice (`FLAG*`) te izvršava operacije nad početnim nizom i uspoređuje dobiveni niz sa nizom znakova *FLAG*. Zabilježena skripta alatom Kibana je prikazana na slici (Slika 6.6).

Slika 6.6: Skripta za određivanje operacije

Uspješno pronalazi zastavicu dekodiranjem potpunog niza znakova XOR operacijom.

Zabilježenim naredbama i promjenama datoteka moguće je odrediti vrijeme potrebno za otkrivanje zastavice te poduzete korake.

7. Zaključak

Kroz ovaj diplomski rad je opisana i implementirana metoda za bilježenje aktivnosti osoba tijekom rješavanja CTF zadataka. Korištenjem različitih alata i tehnika, uključujući Elasticsearch, Filebeat, Kibanu, Tcpdump i posebno razvijene skripte, omogućeno je detaljno praćenje i analiza aktivnosti penetracijskih testera. Ovi alati omogućili su bilježenje i pregledavanje zapisa, čime je moguće rekreirati korake koje je osoba poduzimala za vrijeme rješavanja CTF zadataka.

Primjenom opisane metodologije postavljena je osnova za daljnje unapređenje bilježenja aktivnosti vezanih za CTF natjecanja. Preporučuje se daljnje istraživanje i primjena naprednih tehnika poput strojnog učenja za bolje razumijevanje ponašanja penetracijskih testera. Kao budući rad, moguće je koristiti strojno učenje za ostvarivanje sustava koji prikupljene zapise mapira na *MITRE ATT&CK* [14] matricu.

Razvijeni sustav omogućava ne samo praćenje napretka i kompetencija penetracijskih testera, već i identifikaciju ključnih trenutaka u kojima tester pokazuje inovativnost i sposobnost prilagodbe. Ovo je često presudno za uspjeh u CTF natjecanjima i ukupnu sigurnost informacijskih sustava organizacija koje ih zapošljavaju. Nadalje, posebna pažnja posvećena je zaštiti privatnosti sudionika, implementirajući stroge mjere zaštite podataka i osiguravajući da prikupljanje podataka bude etički i pravno prihvatljivo.

LITERATURA

[1] Defcon Capture the Flag: defending vulnerable code from intense attack

[2] Field Effect Blog: Capture the Flag Cybersecurity

`https://fieldeffect.com/blog/capture-the-flag-cybersecurity`

[3] RingZer0 CTF Challenges

`https://ringzer0ctf.com/challenges`

[4] Kali Linux Documentation

`https://www.kali.org/docs`

[5] Kali Linux Tools

`https://www.kali.org/tools/`

[6] Clinton Gormley, Zachary Tong, Elasticsearch: The Definitive Guide, O'Reilly Media, 2015.

[7] Alberto Paro, Elasticsearch: A Complete Guide, Packt Publishing, 2020.

[8] Yuvraj Gupta, Kibana Essentials, Packt Publishing, 2020.

[9] Elastic.co: Getting Started with Filebeat

`https://www.elastic.co/guide/en/beats/filebeat/current/filebeat-overview.html`

[10] PortSwigger: Burp Suite

`https://portswigger.net/burp`

[11] PortSwigger: Creating Extender API Examples

`https://portswigger.net/burp/documentation/desktop/extensions/creating/extender-api-examples-legacy`

[12] EUR-Lex: GDPR Regulation 2016/679

`https://eur-lex.europa.eu/eli/reg/2016/679/oj`

[13] RingZer0 CTF Challenges

`https://ringzer0ctf.com/challenges/26`

[14] MITRE ATT&CK

`https://attack.mitre.org/`

Bilježenje aktivnosti osoba tijekom rješavanja CTF zadataka

Sažetak

U današnjem dobu, sigurnost informacijskih sustava je ključna za zaštitu organizacija od sve sofisticiranijih kibernetičkih prijetnji, pri čemu su penetracijski testeri važni u održavanju sigurnosti sustava. Njihova obuka često uključuje rješavanje Capture The Flag (CTF) zadataka, koji simuliraju stvarne napade kako bi se testirale i poboljšale njihove vještine u kontroliranom okruženju. Ovaj diplomski rad fokusira se na bilježenje aktivnosti penetracijskih testera tijekom rješavanja CTF zadataka, definirajući zahtjeve za bilježenje različitih aktivnosti kao što su modificiranje datoteka, HTTP zahtjevi, izvršene naredbe, logovi korištenih aplikacija i mrežni promet. Korištenjem alata poput Kali Linuxa, Elasticsearcha, Kibane, Filebeata, Tcpdumpa i prilagođenih skripti, omogućeno je automatizirano prikupljanje i analiza podataka. Analizirani su mogući utjecaji na privatnost i izrađeni prijedlozi za minimiziranje negativnih učinaka kroz načela zaštite privatnosti podataka gdje je to moguće. Cilj rada je razvijanje programske podrške koja omogućuje detaljno praćenje aktivnosti penetracijskih testera.

Ključne riječi: CTF, penetracijsko testiranje, kibernetička sigurnost, bilježenje aktivnosti, Kali Linux, Elasticsearch, Kibana, Filebeat

Logging of participants' activities while solving CTF challenges

Abstract

In today's age, information systems security is critical to protecting organizations from increasingly sophisticated cyber attacks, with penetration testers being important in maintaining system security. Their training often involves solving Capture The Flag (CTF) tasks, which simulate real world attacks to test and improve their skills in a controlled environment. This thesis focuses on recording the activities of penetration testers while solving CTF tasks, defining the requirements for recording different activities such as file modifications, HTTP requests, executed commands, logs of applications used and network traffic. Using tools like Kali Linux, Elasticsearch, Kibana, Filebeat, Tcpdump and custom scripts, automated data collection and analysis is enabled. Possible impacts on privacy were analyzed and proposals were made to minimize negative effects through data privacy protection principles where possible. The goal of the work is to develop software support that enables detailed monitoring of the activities of penetration testers.

Keywords: CTF, penetration testing, cyber security, activity logging, Kali Linux, Elasticsearch, Kibana, Filebeat