UNIVERZITET U NIŠU

ELEKTROSKI FAKULTET
KATEDRA ZA RAČUNARSTVO

Digitalna forenzika

Sistem za analizu izmena na Linux fajl sistemu korišćenjem audit-a i ELK Stek-a.

Mentor:
Prof. dr Bratislav Predić

Student:
Đorđe Cvetković,
Br. indeksa 1366

Niš, Mart 2022. God.

Sadržaj

1. Uvod

Razvoj tehnologije i računarskih sistema doveo je do velike složenosti u njihovom načinu rada, načina funkcionisanja i čuvanja informacija. Usled velike složenosti, rad sistema postaje sve više samostalaniji i automatizovaniji, pa do narušavanja ili prestanka rada često dolazi usled nepredviđenih okolnosti, grešaka u obradi podataka ili zlonamernim napadima izvan sistema. Kako je sistem složen i kompleksan veoma je teško pronaći uzrok izazivanja problema, iz tog razloga kroz istoriju razvila se Digitalna forenzika kao nauka. Dogitalna forenzika predstavlja proces prikupljanja, analize, obradjivanja podataka i kreiranja izveštaja o nekom događaju na računarskom sistemu. Razvitkom tehnologije i sve većeg broja uređaja razvile su se i grane digitalne forenzike. Tako danas postoje grane koje se bave forenzikom mobilnih uređaja, mrežnog saobraćaja, cloud sistema, dešifrirovanjem podataka, oporavka i vračanja izgubljenih podataka... U digitalnoj forenzičkoj istrazi, veoma je važno da digitalni forenzičari imaju sva potrebna znanja o tome kako radi računarski sistem, mrežni uređaji i internet servisi, kao i potrebne veštine da efektivno i efikasno sakupe sve neophodne digitalne dokaze. Da bi sve to mogli da urade kreiranu su alati koji vrše nadgledanje, izdvajanje, formatiranje korisnih podataka tokom istrage.

U ovom radu će biti obrađen sistem za nadgledanje Linux fajl sistema koji prati sve promene nad određenim fajlovima u sistemu ili prati celokupni fajl sistem. Sistem će biti u mogućnosti da prikupi sve potrebne informacije, obradi i kroz sistem logova i monitoringa prikaže u realnom vremenu. Tehnologije koje će biti korišćene su aditd, filebeat i ELK stek.

Drugo poglavlje će biti posvećeno nadgledanju fajl sistema, praćenju promena, kreiranja ili brisanja datoteka sa fajl sistema. Takođe biće reči o alatima za praćenje datoteka i Auditd alatu.

Treće poglavlje se bavi logovima i njihovom obradom, biće opisano šta predstavlja log i koje informacije daje o događaju za koji je kreiran. Biće dati način obrade logova i analaza informacija iz loga. Takođe će biti reči o Filebeat servisu i ELK steku. Šta oni predstavljaju i koju ulogu imaju u procesu obrade logova.

Četvrto poglavlje se bavi implementacijom sistema za analizu izmena na Linux fajl sistemu, u ovom poglavlju će biti prvo data arhitektura sistema a u drugom delu ovog poglavlja biće reči o svakoj komponenti sistema o načinu implementacije i rezultatu koji daje određena komponenta.

2. Praćenje promena na fajl sistemu

Praćenje promena fajlova i datoteka na operativnom sistemu predstavlja jednu od sigurnosnih komponenti koja može osigurati stabilniji rad sitema i pružiti neophodne podatke prilikom istrage dešavanja na sistemu. Glavni cilj praćenja fajl sistema uz proveru integriteta fajlova je pružanje informacija o promenama, izmeni ili brisanju datoteka na sistemu. U okruženju operativnog sistema, svaka komponenta kao što su drajveri uređaja, korisnicki programi i ostali servisi čuvaju konfiguracione podatke u datotekama. Postoji ogroman broj datoteka koje se nalaze u okruženjima modernog operativnog sistema. Zbog toga sve češće datoteke postaju glavna meta napadača pri kompromitovanju operativnih sistema. Napad se može izvesti modifikovanjem ili izmenom postojećih datoteka, brisanjem, dodavanjem i sakrivanjem povezanih datoteka. Napadači mogu primeniti mnoge tehnike kako bi napali datoteke u okruženju operativnog sistema i učinili da zaštita datoteka postane vitalni zadatak. U tu svrhu je potrebna implementacija praćenja fajlova i drugih srodnih alata za bezbednost sistema.

2.1 Auditd

Auditd sistem za reviziju Linux operativnog sistema omogućava detaljno evidentiranje događaja vezanih za bezbednost sistema. Administrator konfiguriše pravila po kojima će se odrediti koji će događaji biti zabeleženi od strane audit-a. Neka od pravila koja mogu biti konfigurisana: pristup specijalnom fajlu ili direktorijumu, specifični sistemski poziv, sistemske komande pokrenute od strane korisničkog naloga, autentifikacija događaja, mrežni pristupi... [1]

Audit arhitektura uključuje [1]:
- glavnu (kernel) komponentu koja generiše događaje.
- auditd jedinicu koja prikuplja evente i upisuje detalje događaja u log file.
- audisp jedinicu koja prenosi događaje drugim aplikacijama radi dodatne obrade.
- auditctl kontrolna jedinica koja omogućava konfiguraciju.

Audit servis generiše događaje u skladu sa pravilima. Ova pravila se mogu dinamički postaviti pomoću auditctl-a ili se statički mogu upisati u /etc/audit/rules.d direktorijuim u posebne fajlove. Pravila u direktorijumu /etc/audit/rules.d automatski se kompajliraju u /etc/audit/audit.rules konfiguracionom fajlu kada se audit pokrene [1].

Postoje tri vrste pravila: kontrolno pravilo menja ponašanje audit-a, pravila za nadgledanje fajl sistema nadgleda datoteke ili direktorijume na fajl sistemu, pravilo sistemskog poziva generiše događaj za određeni sistemski poziv [1].

Kontrolna pravila:

Kontrolna pravila generalno uključuju konfigurisanje audit servisa umesto da mu govore šta treba da nadgleda. Ove komande obično uključuju brisanje svih pravila, postavljanje veličine reda zaostalih zadataka u jezgru, podešavanje režima greške, postavljanje ograničenja brzine događaja ili da se kaže auditctl da zanemari sintaksne greške u pravilima i nastavi učitavanje. Generalno, ova pravila su na vrhu datoteke koja sadrži pravila [2].

Neka od kontrolnih pravila [3]:

* -b size postavlja maksimalnu veličinu audit bafera, kod opterećenijih sistema potreban je veći bafer dok kod manje opterećenih može biti manji
* -D brisanje svih pravila, postavlja se na početku fajla za definisanje pravila
* -e [0...2] koristi se za dozvolu pokretanja konfiguracije audit servisa, 0 znaci da audit ne uzima u obzir konfiguraciju, 1 uzima u obzir konfiguraciju i 2 zakljucava konfiguraciju do sledećeg restarta sistema

Pravila za nadgledanje fajl sistema:

Ova pravila se koriste za reviziju pristupa datotekama ili direktorijumima celog ili određenog dela fajl sistema. Ako je putanja navedena u pravilu direktorijum, tada se pravilo koristi rekurzivno do dna stabla direktorijuma uključujući sve direktorijume koji su ispod navedenog direktorijuma. Sintaksa ovih pravila uglavnom prati ovaj format [2]:

-w path-to-file -p permissions -k keyname

-w definiše putanju do direktorijuma ili fajla koji se nadgleda, kao argument uzima apsolutnu putanju na fajl sistemu
-p definiše tip nadgledanja direktorijuma ili fajla, okida previlo kada se desi:
r - čitanje fajla
w - upis u fajl
x - izvršenje fajla
a - promena fajla
-k definise komentar koji će pratiti svaki događaj koji se kreira ovim pravilom

Pravila nadgledanja sistemskih poziva:
Pravila sistemskog poziva učitavaju se u odgovarajući mehanizam koji presreće svaki sistemski poziv koji naprave svi programi na sistemu. Zbog toga je veoma važno koristiti pravila sistemskih poziva samo kada je neophodno, jer ona utiču na performanse. Što je više pravila, veći je učinak. Međutim, kombinovanjem sistemskih poziva u jedno pravilo kad god je to moguće može se smanjiti uticaj na performanse [2].

Format pravila sistemskih poziva:

-a action,list -S syscall -F field=value -k keyname

action - ovo polje označava koju akciju treba preduzeti kada se detektuje pravilo. Može biti: always (označava da uvek treba kreirati događaj) ili never (nikada ne kreirati događaj)

list - Jezgro Linuksa ima 4 liste. To su: task, exit, user i exclude.
task lista zadataka se proverava samo tokom fork ili kloniranja sistemskih poziva. Retko se koristi u praksi.
exit lista je mesto na kome se procenjuju svi zahtevi sistemskih poziva i sistema datoteka.
user lista se koristi za filtriranje (uklanjanje) nekih događaja koji potiču iz korisničkog prostora. Podrazumevano je dozvoljen svaki događaj koji potiče iz korisničkog prostora.

Dakle, ako postoje neki događaji koje ne želite da vidite, onda je ovo mesto gde se neki mogu ukloniti.
excluded lista se koristi za isključivanje emitovanja određenih događaja. Polje msgtipe se koristi da kaže jezgru koje vrste poruka ne želite da snimite. Ovaj filter može ukloniti događaj u celini i nije selektivan u vezi sa bilo kojim drugim atributom. Korisnički i izlazni filteri su pogodniji za selektivnu reviziju događaja.

-S syscall - Ovo polje može biti ime ili broj sistemskog poziva. Radi čitljivosti, naziv se skoro uvek koristi. Može se zadati više od jednog sistemskog poziva u pravilu tako što se navede druga opcija -S. Kada se pošalju u jezgro, sva polja sistemskog poziva se stavljaju u istu masku tako da jedno poređenje može da utvrdi da li je sistemski poziv interesantan [9].

-F field=value - Opcija koja dozvoljava fino podešavanje onoga sa čim se podudara. Na primer: -F auid>=500 -F auid!=4294967295 deo pravila znači da sistemski id korisnika mora da bude veći od 500 i da njegova vrednost mora da se razlikuje od 4294967295 (što predstavlja vrednost -1)

-k keyname - definise komentar koji će pratiti svaki događaj koji se kreira ovim pravilom

Primer nekoliko audit pravila:

-a always,exit -F arch=b64 -S mount -S umount2 -F auid!=-1 -k mount

Ovo pravilo definiše da li je došlo do zakačivanja ili otkačivanja novog diska na sistemu pomoću sistemskih poziva mount i unmount

-w /etc/passwd -p wa -k etcpasswd

Ovo pravilo nadgleda da li je bilo upisa ili izmena u fajl na putanji /etc/passwd. Na linuksu ovaj faj se koristi za čuvanje informacija o korisničkim nalozima.

-w /usr/bin/wget -p x -k susp_activity

Proverava da li je došlo do izvršenja wget komande

-a always,exit -F arch=b32 -S all -k 32bit_api

Ovo pravilo proverava da li postoje 32-bitni sistemski pozivi na sistemu

3. Obrada logova

Log je zapis o događajima koji se dešavaju u sistemima i na mrežama sistema. Svaki unos loga sadrži informacije koje se odnose na određeni događaj koji se dogodio unutar sistema ili na mreži. Prvobitno su se logovi koristili za rešavanje problema, ali sada služe mnogim funkcijama, kao što su optimizacija performansi sistema i mreže, beleženje radnji korisnika i pružanje podataka korisnih za istraživanje zlonamernih aktivnosti [4].

Evidencija logova je evoluirala tako da sadrži informacije vezane za mnoge različite vrste događaja koji se dešavaju unutar mreža i sistema. Često unutar sistema, zbog rasprostranjene primene umreženih servera, radnih stanica i drugih računarskih uređaja i sve većeg broja pretnji mrežama i sistemima, logovi sadrže zapise koji se odnose na računarsku bezbednost; uobičajeni primeri ovih računarskih bezbednosnih logova su evidencije revizije koje prate pokušaje potvrđivanja identiteta korisnika i evidencije bezbednosnih uređaja koje beleže moguće napade. Ovo je stvorilo potrebu za upravljanjem evidencijom logova računarske bezbednosti, što je proces generisanja, prenosa, skladištenja, analize i odlaganja podataka logova sistema [4].

Ukoliko posmatramo logove sa strane bezbednosti i zaštite sistema možemo ih svrstati u tri kategorije[4]:

Bezbednosni logovi - Prvenstveno sadrže logove programa i sistema koji služe za bezbednost i zaštitu servera i mrežnih uređaja kao što su: programi za otrivanje virusa, sistemi za otkrivanje i sprečavanje napada, programi za udaljeni pristup, mrežni zaštitni uređaji...

Logovi operativnih sistema - Operativni sistemi servera, radnih računara i mrežnih uređaja u svoje logove upisuju razne informacije vezane za njihov rad: pokretanje, gašenje i restartovanje sistema, događaja koji su naišli na grešku u izvršavanju, informacije o uspešnim i neuspešnim logovanjima korisnika na sistem, korisničkim privilegijama...

Logovi aplikacija - Ovi logovi sadrže informacije o izvršenim zadacima koji su se dogodili u softverskoj aplikaciji. Aplikacija obavlja zadatke i stanje o izvršenju zapisuje u logu. Oni mogu uključivati greške i upozorenja, kao i informacije o izvršenim zadacima. Vrste informacija i format poruka koje se nalaze u logovima aplikacije razlikuju se u zavisnosti od aplikacije. To je zato što ove promenljive nisu određene spoljnim propisima ili operativnim sistemom već programeri softverske aplikacije kontrolišu ono što ulazi u logove.

Podaci zabeleženi za svaki od ovih logova uveliko se razlikuju, ali svaki log događaja ima vremensku oznaku, a druge dodatne informacije mogu uključivati status i kodove grešaka, ime usluge i korisnički ili sistemski nalog povezan sa događajem, poruku loga, kritičnost loga,

Upravljanje logovima

Upravljanje logovima može koristiti sistemu na mnogo načina. Pomaže da se obezbedi da se sigurnosni zapisi o računaru skladište dovoljno detaljno tokom odgovarajućeg vremenskog

perioda. Rutinski pregledi i analize logova korisni su za identifikovanje bezbednosnih incidenata, kršenja smernica, prevara i operativnih problema ubrzo nakon što su se dogodili, i za pružanje informacija korisnih za rešavanje takvih problema. Evidencije mogu biti korisne i za obavljanje revizije i forenzičke analize, za podršku unutrašnjim istragama organizacije, utvrđivanje osnove i identifikovanje operativnih trendova i dugoročnih problema.

Upravljanje logovima podeljeno je u tri nivoa i sastoji se od konstantnog prikupljanja logova sa uređaja, smeštanje i obradu logova i analiziranje logova.

Prvi nivo sadrži uređaje koji generišu podatke logova. Ovi uređaji obično pokreću klijentske aplikacije ili usluge za evidentiranje koje njihove logove čine dostupnim preko mreža za evidentiranje na servera u drugom sloju. Drugi uređaji čine svoje zapise dostupnim na druge načine, na primer dozvoljavajući serverima da im potvrde autentičnost i preuzmu kopije logova. Drugi nivo se sastoji od jednog ili više servera koji primaju podatke logova ili kopije podataka logova od uređaja u prvom sloju. Podaci se prenose na servere u realnom ili skoro realnom vremenu ili u povremenim grupama na osnovu rasporeda ili količine podataka logova koji čekaju na prenos. Serveri koji primaju podatke od više generatora logova ponekad se nazivaju sakupljači ili agregatori. Podaci dnevnika mogu se čuvati na samim serverima evidencije ili na zasebnim serverima baza podataka. Treći nivo sadrži konzole koje se mogu koristiti za pregled podataka logova i rezultata automatizovane analize. Konzole za praćenje logova se takođe mogu koristiti za generisanje izveštaja. U nekim infrastrukturama za upravljanje evidencijama logova, konzole se takođe mogu koristiti za obezbeđivanje upravljanja serverima logova na drugom nivou i uređajima na prvom nivou. Takođe, korisničke privilegije konzole ponekad mogu biti ograničene samo na potrebne funkcije i izvore podataka za svakog korisnika [4].

3.1 Filebeat

Filebeat je lagani alat za prosleđivanje i centralizaciju podataka logova. Instaliran kao agent na serverima, Filebeat nadgleda datoteke logova koje su navedene u konfiguraciji, prikuplja događaje logova i prosleđuje ih radi indeksiranja, obrade ili skladištenja [5].

Filebeat se sastoji od dve glavne komponente: inputs i harvesters. Ove komponente rade zajedno i šalju podatke o događaju na izlaze [5].

Harvester je odgovoran za čitanje sadržaja jedne datoteke logova. Harvester čita svaku datoteku, red po red, i šalje sadržaj na izlaz. Za svaku datoteku se pokreće jedan Harvester. Harvester je odgovoran za otvaranje i zatvaranje datoteke, što znači da deskriptor datoteke ostaje otvoren dok je harvester pokrenut. Ako se datoteka ukloni ili preimenuje dok se pribavljaju podaci iz nje, Filebeat nastavlja da čita datoteku. To ima nuspojavu da je prostor na disku rezervisan dok se harvester ne zatvori [5].

Inputs komponenta je odgovorna za upravljanje harvester-ima i pronalaženje svih izvora za čitanje. Ako je tip unosa log, inputs pronalazi sve datoteke na disku koje se podudaraju sa definisanim glob putanjama i pokreće harvester za svaku datoteku. Svaki ulaz radi u sopstvenoj rutini. Inputs omoućava i dodavanje novog polja prilikom učitavanja

podataka iz fajla pomoću dodatka fields. Polja mogu biti skalarne vrednosti, nizovi, rečnici ili bilo koja ugnežđena kombinacija. Podrazumevano će polja koja navedete biti grupisana unutar fields polja unutar pročitanih podataka [5].

Sledeći primer konfiguriše Filebeat za prikupljanje linija iz svih datoteka logova koje se podudaraju sa navedenim obrascima ispod paths sekcije i dodaje fields polje koje će pratiti podatke. Obe sekcije se nalaze ispod type, koji definiše tip podataka koji se prikupljaju:

```
filebeat.inputs:
 - type:  log
    paths:
          - /var/log/*.log
          - /var/path2/*.log
   fields:
          name: field_1
```

Filebeat čuva i ažurira stanje svake datoteke u memoriji u datoteci registra. Stanje se koristi za pamćenje poslednjeg zapisa loga sa kog je čitao i za osiguranje slanja svih linija loga. Ako izlaz, poput Elasticsearch ili Logstash, nije dostupan, Filebeat prati poslednje poslate redove i nastaviće da čita datoteke čim izlaz ponovo postane dostupan. Dok je Filebeat ugašen, informacije o stanju se takođe čuvaju u na disku za svaki ulaz. Kada se Filebeat ponovo pokrene, podaci iz datoteke registra se koriste za obnovu stanja, a Filebeat nastavlja svaki harvester na poslednjoj poznatoj poziciji [5].

Sve ulazne podatke Filebeat može slati samo preko jednog strima koji može biti konfigurisan tako da podatke ispisuje u konzolu, u fajlu specifiiciranom u konfiguraciji, da podatke preko mreže salje dalje softverima za obradu ili da ih upisuje u bazu podataka. Primer konfiguracije jednog izlaza dat je u nastavku.

```
output.elasticsearch:
        hosts: ["https://myESHost:9200"]
        username: "elasticsearch_writer"
        password: "PASSWORD"
```

Izlaz kao i sva ostala podešavanja Filebeat-a se obavljaju u konfiguracionom fajlu filebeat.yml.

3.2 ELK Stek

ELK je akronim za tri projekta otvorenog koda: Elasticsearch, Logstash i Kibana. Zajedno, ove različite komponente se najčešće koriste za nadgledanje, rešavanje problema i zaštitu IT okruženja. Komponente u ELK Steku su dizajnirane da međusobno komuniciraju bez previše komplikovane konfiguracije. Elasticsearch je sistem baza podataka. Logstash služi za obradu podataka na strani servera koji unosi podatke iz više izvora istovremeno, transformiše ih, a

zatim ih šalje u skladište poput Elasticsearch-a. Kibana omogućava korisnicima da vizualizuju podatke pomoću tabela i grafikona u grafičkom korisničkom interfejsu [6]. Šema arhitekture ELK steka je data na slici 1.

Slika 1. Šema arhitekture ELK stek-a.

3.2.1 Elasticsearch

Elasticsearch je distribuirani pretraživač i analitički mehanizam ELK Stack-a. Elasticsearch je mesto gde se izvršava logika indeksiranja, pretraživanja i analize. Elasticsearch pruža pretragu i analitiku za sve vrste podataka skoro u realnom vremenu. Bilo da ima struktuiran ili nestruktuiran tekst, numeričke podatke ili geoprostorne podatke. Elasticsearch može efikasno da ih skladišti i indeksira na način koji podržava brzu pretragu. Kako podaci i obim upita rastu, distribuirana priroda Elasticsearch-a omogućava da primena neprimetno raste zajedno sa njom [6].

Za obradu podataka i drugih zahteva Elasticsearch koristi REST API-je. Ovo omogućava interakciju sa Elasticsearch-om pomoću bilo kog klijenta koji šalje HTTP zahteve, kao što je curl. Takođe se može koristiti Kibana konzola za slanje zahteva Elasticsearch-u [6].

Primer slanja zahteva pomoću curl komande:

curl -X GET -u elastic:<password> http://localhost:9200/_cat/indeices?v

elastic - korisnički nalog elasticsearch-a
< password > - šifra korisničkog naloga
localhost:9200 - IP adresa ili hostname gde se nalazi Elasticsearch i port na kome radi
_cat/indeices?v - Označava upit koji vraća sve kreirane indekse unutar Elasticsearch-a

Dodavanje podataka se vrši pomoću JSON objekata koji se nazivaju dokumenti u Elasticsearch-u. Elasticsearch čuva ove dokumente u indeksima za pretraživanje. Za podatke o vremenskim serijama, kao što su logovi i metrike, obično dodaje dokumente u tok podataka koji se sastoji od više automatski generisanih indeksa podataka. Za tok podataka potreban je indeks koji odgovara njegovom imenu. Dokumenti poslati u tok podataka moraju imati polje @timestamp [6].

Primer dodavanje apache-acces loga unutar apache_log indeksa:

```
POST apache_log/_doc
{
        "@timestamp": "2099-05-06T16:21:15.000Z",
        "event": {
                "original": "192.0.2.42  [06/May/2099:16:21:15 +0000] \"GET /images/bg.jpg
HTTP/1.0\" 200"
        }
}
```

Kako bi imali što bolje performanse Elasticsearch je moguće kreirati kao klaster, grupu od jedne ili više instanci Elasticsearch-a koji su povezani zajedno. Snaga Elasticsearch klastera leži u distribuciji zadataka, pretraživanju i indeksiranju, po svim čvorovima u klasteru. Čvor je jedan server koji je deo klastera, skladišti podatke i učestvuje u mogućnostima indeksiranja i pretraživanja klastera i može se konfigurisati na različite načine [6]:

Glavni čvor - kontroliše klaster Elasticsearch i odgovoran je za sve operacije na čitavom klasteru, poput stvaranja/brisanja indeksa i dodavanja/uklanjanja čvorova.

Čvor podataka - skladišti podatke i izvršava operacije povezane sa podacima, poput pretraživanja i objedinjavanja.

Klijentski čvor - Prosleđuje zahteve klastera glavnom čvoru i zahteve vezane za podatke čvorovima podataka.

Unutar klastera Elasticsearch pruža mogućnost podele indeksa na više delova. Svaki deo je sam po sebi potpuno funkcionalan i nezavisan indeks koji može biti smešten na bilo kom čvoru unutar klastera. Distribucijom dokumenata u indeksu po više fragmenata i distribucijom tih fragmenata po više čvorova, Elasticsearch može osigurati redundantnost, koja štiti i od hardverskih kvarova i povećava kapacitet upita kako se čvorovi dodaju u klaster. Elasticsearch omogućava jednu ili više kopija fragmenata indeksa koji se nazivaju fragmenti replika ili samo replike. U osnovi, replika fragment je kopija primarnog fragmenta. Svaki dokument u indeksu pripada jednom primarnom fragmentu. Replike pružaju više kopija podataka radi zaštite od kvara hardvera i povećanja kapaciteta za opsluživanje zahteva za čitanje, poput pretraživanja ili preuzimanja dokumenta [6].

Dobijanje rezultata se vrši pomoću upita. Upit za pretragu ili upit je zahtev za informacijama o podacima u Elasticsearch tokovima podataka ili indeksima. Upit može biti kao pitanje, napisano na način koji Elasticsearch razume. Pretraživanje podataka se sastoji od jednog ili više upita koji se kombinuju i šalju Elasticsearch-u. Dokumenti koji odgovaraju upitima pretraživanja vraćaju se kao rezultat pretraživanja. Pretraživanje takođe može sadržati dodatne informacije koje se koriste za bolju obradu upita. Na primer, pretraga može biti ograničena na određeni indeks ili samo dati određeni broj rezultata. API za pretragu može se koristiti za pretraživanje i objedinjavanje podataka uskladištenih u Elasticsearch tokovima podataka ili indeksima. Parametar tela zahteva API-ja za upite prihvata upite napisane u Query DSL -u [6].

Primer zahteva koji pretražuje my-indeks koristeći upit match. Ovaj upit se podudara sa dokumentima sa user.id vrednošću my-user.

```
GET /my-indeks/_search
{
  "query": {
    "match": {
      "user.id": "my-user"
      }
  }
}
```

API odgovor vraća prvih 10 dokumenata koji odgovaraju upitu.

3.2.2 Logstash

Logstash je mehanizam za prikupljanje podataka otvorenog koda sa mogućnostima protoka u realnom vremenu. Logstash može dinamički objediniti podatke iz različitih izvora, normalizovati podatke i poslati na odredišta po izboru. Logstash je prvobitno služio za prikupljanje logova, a danas njegove mogućnosti se protežu daleko izvan tog slučaja upotrebe. Bilo koji tip događaja može se obogatiti i transformisati širokim spektrom dodataka za ulaz, filter i izlaz, mnogim izvornim funkcijama koje dodatno pojednostavljuju proces unosa. Logstash ubrzava uvide prikupljanjem veće količine i raznolikosti podataka [7].

Logstash agent za obradu koristi 3 faze: ulaz → filter → izlaz. Ulazi generišu događaje, filteri ih menjaju, izlazi ih šalju na drugo mesto. Svaka faza je poprilično fleksibilna jer Logstash sadrži dosta dodataka koji se mogu iskoristiti za bolje i lakše upravljanje podacima [7].

Dodaci za unos su važne komponente protokola Logstash koji rade kao posrednički softver između izvora unosa logova i funkcije filtriranja Logstasha. Generalno, svaki ulazni dodatak omogućava povezivanje sa određenim izvorom logova i unošenje tih logova pomoću njegovog API-ja. U Logstash-u, ulazne dodatke može instalirati i upravljati menadžerom dodataka koji se nalazi na file sistemu u direktorijummu bin/logstash-plugin. Međutim, neki od najpopularnijih dodataka za unos instalirani su van ovog direktorijuma [7].

Najkorišćeniji ulazni dodaci Logstash-a [7]:

beats - Ovaj dodatak za unos omogućava Logstash-u da prima događaje preko mrežnih konekcija, konfiguriše se tako da logstash postavi određeni port u mod slušanja i stalno osluškuje i kreira konekcije preko kojih prima podatke koje dalje šalje na obradu. Najčešce se Filebeat koristi kao pošiljalac podataka koje logstash prima na ovaj način.
Sledeći primer pokazuje kako konfigurisati Logstash da sluša na portu 5044 dolazne Beats veze:

```
input {
   beats {
      port => 5044
   }
 }
```

file - Omogućava čitanje podataka iz specificiranog fajla koji se nalazi na istom fajl sistemu kao i logstash. Dodatak prati trenutnu poziciju u svakoj datoteci snimajući je u zasebnu datoteku pod imenom sincedb. Ovo omogućava da se zaustavi i ponovo pokrene Logstash i nastavlja tamo gde je stao, a da ne propusti redove koji su dodati u datoteku dok je Logstash bio zaustavljen.

Pored navedenih postoje i: azure_event_hubs, cloudwatch, couchdb_changes, dead_letter_queue, elastic_agent, elasticsearch, exec, ganglia gelf, generator, github, google_cloud_storage, google_pubsub, graphite, heartbeat, http, http_poller, imap, irc, java_generator, java_stdin, jdbc, jms, kafka, kinesis, log4j, lumberjack, meetup, pipe, puppet_facter, rabbitmq, redis, relp, rss, s3, s3-sns-sqs, salesforce, snmp, snmptrap, sqs, stdin, stomp, syslog, tcp, twitter, udp, unix, varnishlog, websocket, wmi, xmpp

Najkorišćeniji filter dodaci Logstash-a [7]:

grok - Omogućava raščlanjivanje i struktuiranje proizvoljnog teksta. Grok je odličan način za raščlanjivanje nestrukturiranih podataka logova u lako čitljivi oblik. Ovaj alat je savršen za evidencije sislog logova, apache i druge zapise veb servera ili bilo koji format teksta koji je napisan da bude čitljiv ljudima. Grok funkcioniše tako što kombinuje tekstualne obrasce, napisane u obliku: **%{SYNTAX:SEMANTIC}**

- SINTAX je naziv uzorka koji će odgovarati tekstu. Na primer, 3.44 će se podudarati sa uzorkom NUMBER, a 55.3.244.1 sa IP obrascem. Najčešće korišćeni SINTAX parametri: COMBINEDAPACHELOG, DATESTAMP, TIME, PATH, IP, HOSTNAME, USERNAME, UUID, WORD, NUMBER, DATA
- SEMANTIC je identifikator koji daje naziv delu teksta koji se podudara. Na primer, 3,44 može biti trajanje događaja, pa se može nazvati jednostavno "vreme trajanja".

Sa obrascima sintakse i semantike, možemo izvući korisna polja iz uzorka logova poput sledećeg primera http zahteva:

55.3.244.1 GET /index.html 15824 0.043

Patern za ovakav log može biti:

%{IP:client} %{WORD:method} %{URIPATH:request} %{NUMBER:bytes} %NUMBER:duration}

%{IP:client} - Izvlači Ip adresu i stavlja je u client polje
%{WORD:method} - Izvlači reč i stavlja je u polje method
%{URIPATHPARAM:request} - Izvlači putanju i stavljaje u polje request
%{NUMBER:bytes} - Izvlači podatak tipa broj i stavlja ga u polje bytes

%{NUMBER:duration} - Izvlači podatak tipa broj i stavlja ga u polje duration

Rezultat primene paterna iz primera:

```
{
        client: "55.3.244.1"
        method: "GET"
        request: "/index.html"
        bytes: "15824"
        duration: "0.043"
}
```

Unutar grok dodatka moguće je koristiti i paterna oblika: **(?<field_name>the pattern here)**

Na primer, indeksiranje heksadecimalne vrednost od 10 ili 11 znakova. Izgleda ovako:

```
(?<heksa_broj>[0-9A-F]{10,11})
```

?<heksa_broj> - označava vrednost polja u koje se upisuje indeksirana vrednost
[0-9A-F] - označava da se indeksira vrednost koja se sastoji od cifara između 0 i 9 i velikih slova od A do F
{10,11} - označava da je dužina niza slova i brojeva 11 ili 12 karaktera

mutate - Filter mutacije omogućava da Logstash izvrši opšte mutacije nad poljima. Može da preimenuje, ukloni, zameni i izmeni polja unutar obrađenog loga.

geoip - Dodaje informacije o geografskoj lokaciji IP adresa, na osnovu podataka iz baza podataka MaxMind GeoLite2. Za prosleđenu IP adresu dodaje informacije u o državi, gradu, tačne koordinate, vremenskoj zoni, kontinentu...

Pored navedenih postoje i: aggregate, alter, bytes, cidr, cipher, clone, csv, date, de_dot, dissect, dns, drop, elapsed, elasticsearch, environment, extractnumbers, fingerprint, http, i18n, java_uuid, jdbc_static, jdbc_streaming, json, json_encode, kv, memcached, metricize, metrics, prune, range, ruby, sleep, split, syslog_pri, threats_classifier, throttle, tld, translate, truncate, urldecode, useragent, uuid, wurfl_device_detection, xml

Najkorišćeniji izlazni dodaci Logstash-a [7]:

elasticsearch - Logstah koristi ovaj dodatak kako bi direktno poslao obrađene podatke Elasticsearch-u koji ih skladišti, može se konfigurisati više izlaza pri čemu svakom može da se doda indeks kako bi Elasticsearch posebno skladištio ove podatke, takođe može da se navede samo jedna mašina ili ceo klaster Elasticsearch-a. Primer konfiguracije elasticsearch izlaznog dodatka:

```
output {
        elasticsearch{
                hosts => ["IPEC:PORT"]
```

}
}

IPEC - IP adresa ili hostname elasticsearch servera
PORT - Broj porta na kom je podešen da sluša Elasticsearch.

stdout - Jednostavan izlaz koji štampa na standardni izlaz komandne linije koja pokreće Logstash. Ovaj izlaz može biti prilično zgodan pri otklanjanju grešaka u konfiguracijama dodataka, omogućavajući trenutni pristup podacima o događajima nakon što prođu kroz ulaze i filtere. Ne postoje posebne opcije konfiguracije za ovaj dodatak, ali podržava dva tipa ispisa podataka u JSON formatu ili rubydebug formatu.

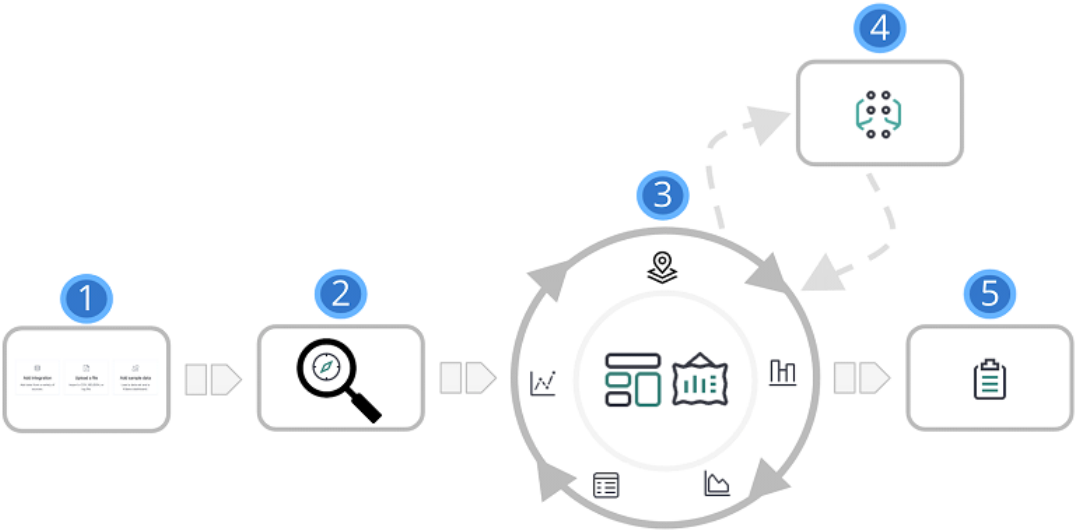
file - Obezbeđuje da se izlaz zapisuje u fajlovima na disku.

Pored navedenih postoje i: boundary, circonus, cloudwatch, csv, datadog, datadog_metrics, dynatrace, elastic_app_search, elastic_workplace_search, email, exec, ganglia, gelf, google_bigquery, google_cloud_storage, google_pubsub, graphite, graphtastic, http, influxdb, irc, java_stdout, juggernaut, kafka, librato, loggly, lumberjack, metriccatcher, mongodb, nagios, nagios_nsca, opentsdb, pagerduty, pipe, rabbitmq, redis, redmine, riak, riemann, s3, sink, sns, solr_http, sqs, statsd, stomp, syslog, tcp, timber, udp, webhdfs, websocket, xmpp, zabbix

3.2.3 Kibana

Kibana omogućava korisnicima da vizualizuju podatke pomoću tabela i grafikona na korisničkom interfejsu. Kibana može da pretražuje i analizira podatke, omogućava kreiranje upita, vizualizaciju rezultata upita na grafikonima, meračima, mapama sa mogućnošću da ih kombinuje na kontrolnoj tabli. Kibana radi sa svim vrstama podataka. Podaci mogu biti strukturirani ili nestrukturirani tekst, numerički podaci, podaci o vremenskim serijama, geoprostorni podaci, evidencije, metrike, bezbednosni događaji i drugo. Bez obzira na podatke, Kibana može da otkrije obrasce, odnose i vizualizuje rezultate [8].
Analiza podataka je osnovna funkcija Kibane. Brzo pretražuje velike količine podataka, istražuje polja i vrednosti, a zatim ih prikazuje interfejs prevlačenja i ispuštanja za brzu izradu grafikona, tabela, pokazatelja i još mnogo toga. [8].

Struktura komponenti Kibane je data na slici 2.



Slika 2. Komponente u strukturi Kibane

1) Add data - Ulazni tok podataka
2) Explore - Ovom komponenta može pretraživati podatke radi skrivenih uvida i odnosa. Omogućava kreiranje upita, a zatim filtriranje rezultata samo prema zahtevanim podacima. Može ograničiti svoje rezultate na najnovije dokumente dodate na ulazni tok podataka.
3) Visualize - Kibana nudi mnogo opcija za kreiranje vizualizacije podataka, od podataka o vremenskim serijama do geo podataka. Kontrolna tabla je polazna tačka za kreiranje vizualizacija, a zatim mogućnost njihovih spajanja da bi prikazali podatke iz više perspektiva.
4) Model data behavior - Omogućava korišćenje Mašinskog učenja za modeliranje ponašanja podataka, predvidi neobično ponašanje, regresiju i analizu podataka.
5) Share - Kibana nudi mnogo opcija podele podataka - pristup kontrolnoj tabli deljenjem linka, eksportovanje podataka u PDF formatu i još mnogo toga.

4. Implementacija sistema za analizu izmena Linux fajl sistema

U okviru ovog rada izvršena je implementacija sistema za prikupljanje i obradu poataka za potrebe nadgledanja i forenzike Linux fajl sistema. Sistem se sastoji od audit servisa inplementiranog na operativni sistem aplikacionog servera i ELK steka sa Filebeat-om za obradu i prikazivanje podataka evidentiranih prilikom promena na fajl sistemu.
Tokom implementacije sistema za izvršenje naredbi instalacije komponenti korišćen je *bash komandni terminal*, koji je ugrađen na Linux operativnim sistemima. Za kreiranje i konfigurisanje konfiguracionih fajlova komponenti korišćen je *nano* editor unutar bash komandnog terminala. Takođe korišćen je *netstat* paket za praćenje stanja mrežnih konekcija i otvorenih portova. Za praćenje mrežnog saobraćaja korišćen je paket *tcpdump*.
Najpre će biti predstavljena arhitektura i komponente celog sistema a potom implementacija i konfiguracije svake komponente posebno.

4.1 Arhitektura sistema

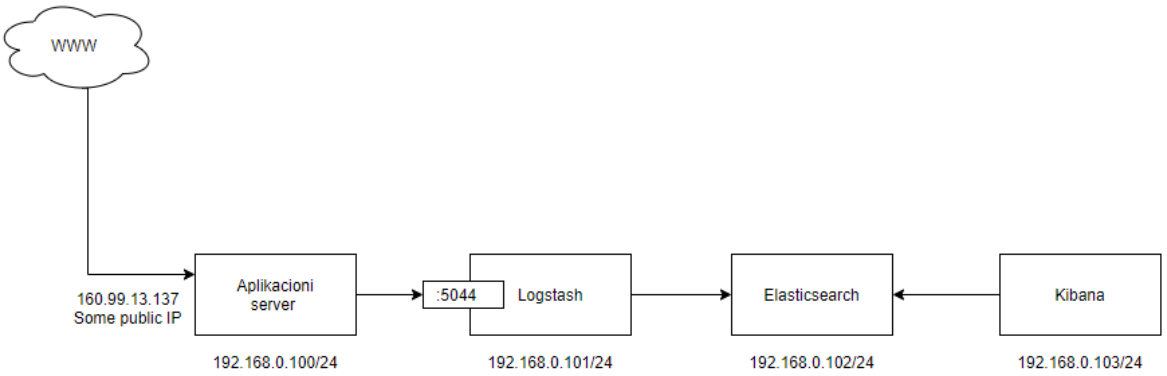
Sistem je implementiran na demo arhitekturi u svrhu predstavljanja nacina funkcionisanja celog sistema. Kompletan sistem čine ukupno 4 servera: aplikacioni server, server na kome je implementiran Logstash, server na kome je implementiran Elasticsearch i server na kome je Kibana.

Mreža na kojoj je implementiran sistem je 192.168.0.0/24 i predstavlja virtuelnu privatnu mrežu.

IP adrese i serveri:
192.168.0.100 - Aplikacioni server na kome se prate promene na fajl sistemu
192.168.0.101 - Logstash server
192.168.0.102 - Elasticsearch server
192.168.0.103 - Kibana server

Operativni sistemi na svim serverima: Ubuntu 20.1, Debian distribucija

Arhitektura sistema je implementirana tako da se nadgleda i prate promene na fajl sistem aplikacionog servera. Na ovom serveru je podignut i konfigurisan audit servis koji prati promene nad direktorijumima u kojima su smešteni podaci aplikacija. Zatim je podignut Filebeat koji skuplja logove sa ovog sistema o događajima koji se evidentiraju i šalje ih logstash serveru na obradu. Logstash je podešen tako da prima podatke sa aplikacionog servera na portu 5044, filtrira, indeksira i šalje Elasticsearch serveru koji ih smešta u svoje datoteke i čini dostupnim preko sistema upita. Kibana pribavlja podatke sa Elasticsearch klastera i prikazuje ih. Ceo sistem radi u realnom vremenu sa minimalnim kašnjenjem. Arhitektura ovog sistema je prikazana na slici 3.



Slika 3. Arhitektura sistema za nadgledanje fajl sistema

4.2 Implementacija komponemti sistema

4.2.1 Implementacija Audit komponente

Implementacija audit servisa obavljena je kako bi na serverima pratili dešavanja sa fajlovima u direktorijumu aplikacija. Na ovaj način nam audit daje potrebne informacije o brisanju, promeni ili ubacivanu novih nepoznatih fajlova koji mogu da budu neposredna pretnja sistemu.

Instalacije audit sisstema se pokreće komandom

sudo apt-get install auditd

Nakon istalacije kreiran je direktorijum na putanji /etc/audit/ u kome su smešteni konfiguracioni fajlovi i pravila audit servisa.
Podešavalje pravila za praćenje direktorijuma u kome se nalazi veb aplikacijama

-w /var/www/ -p rwxa -k aplication_directorium

-w označava da se radi o pravilu za nadgledanje fajl sistema
/vae/www/ označava direktorijum koji pratimo gde je smeštena veb aplikacija
-p rwxa definiše tip nadgledanja direktorijuma, okida pravilo kada se desi :
r - čitanje fajla
w - upis u fajl
x - izvršaenje fajla
a - promena fajla
-k aplication_directory označava komentar koji se dodaje na izlazne podatke pri evidenciji događaja

Nakon podešavanje pravila pokrenuta je komanda koja audit dodaje na listu za pokretanje servisa pri stratovanju sistema

sudo systemctl enable auditd

Nakon dodavanja u listu za pokretanje servisa pri stratovanju sistem, audit je pokrenut kao servis da radi na sistemu komandom

sudo systemctl start auditd

Pri radu audit servis sve događaje koje prepozna preko definisanih pravila evidentira u svom log fajlu koji se nalazi na putanji /var/log/audit/aduit.log. Primer jednog loga dat je u nastavku:

type=SYSCALL msg=audit(1632402971.406:21725): arch=c000003e syscall=42 success=no exit=-115 a0=3 a1=c00019d7cc a2=10 a3=0 items=0 ppid=1 pid=907 auid=4294967295 uid=0 gid=0 fsgid=0 tty=(none) ses=4294967295 comm="filebeat" exe="/usr/share/filebeat/bin/filebeat" key="tcp_connections"

Na osnovu analize loga možemo utvrditi da je pokrenut sistemski poziv sa indeksom tipa "42" što označava izvršenje exe fajla. Pokrenut je "filebeat.exe" servis koji koristi socket konekcjiju a čiji se izvršni fajl nalazi na putanji "/usr/share/filebeat/bin/filebeat" oznečenoj u polju exe unutar loga.
Objašnjenje ostalih polja unutar loga:
ppid =1 - označava id rodiljskog procesa koji je pokrenuo sistemski poziv
pid = 907 - id procesa koji je pokrenuo sistemski poziv
auid = 4294967295 - id audit korisničkog naloga
uid = 0 id korisničkog naloga koji je pokrenuo sistemski poziv
gid = 0 id grupe u kojoj je korisnički nalog koji je pokrenuo sistemski poziv
ses = 4294967295 id audit sesije koja je izvršila pravilo
comm = filebeat - komanda koja je pokrenula sistemski poziv

Ovakva instalacija i konfiguracija audit servisa je urađena na serveru:

- 192.168.0.100 - Aplikacioni server

4.2.2 Implementacija Filebeat komponente

Implementacija Filebeat servisa urađena je kako bi prikupljali zapise logova koje generiše audit servis i prosledili do Logstash servera. Konfiguracija je urađena tako da Filebeat nagdleda log fajl koji generiše audit servis gde evidentira sve dogadjaje definisane pravilima. Prikupljenim podacima se dodaje polje koje označavaja tip loga, zatim se podaci šalju Logstash serveru na obradu i indeksiranje.

Za instalacjiu Filebeat-a potrebno je pokrenuti dve komande. Prvo je potrebno komandom curl pribaviti filebeat paket a potom instalirati paket na sistem komandom dpkg. Izgled ovih komandi dat je u nastavku .

curl -L -O https://artifacts.elastic.co/downloads/beats/filebeat/filebeat-oss-7.8.1-amd64.deb

sudo dpkg -i filebeat-oss-7.8.1-amd64.deb

Nakon instalacije kreiran je direktorijum Filebeat servisa na putanji /etc/filebeat u kome se nalaze konfiguracioni fajlovi. Najznačajniji faj je filebeat.yml u kome se nalazi konfiguracija ulaza i izlaza Filebeat servisa.

Konfiguracija ulaznih fajlova Filebeat-a na serveru:
192.168.0.100 - Aplikacioni server na kome se prate promene na fajl sistemu

```
filebeat.inputs
        enabled: true
        - type: log
          paths:
              - /var/log/audit/audit.log
          fields:
              datatype: audit
```

U ovoj konfiguraciji podešen je ulazni fajl iz kojig filebeat uzima informacije i prosleđuje na izlaz. Ulazni fajl se nalazi na putanji:
- /var/log/audit/audit.log Audit log fajl gde su logovi promena fajl sistema i mrežnih konekcija
Pored podešavanja putanje odakle preuzimamo podatke, imamo konfiguraciju koja dodaje pročitanim podacima još jedno polje koje označavaja tip podatka. Kako bi kasnije kada se svi logovi nađu u jednoj bazi imali polja na osnovu kojih možemo lakše da radimo analizu.

Konfiguracija izlaza podataka Filebeat-a ka Logstash serveru:

```
output.logstash:
        hosts: ["192.168.0.101:5044"]
```

Filebeat je pokrenut na aplikacionom serveru da radi kao servis i omogućeno je startovanje prilikom restarta odnosno dodat je u listu za startovanje servisa prilikom startovanja operativnog sistema.

4.2.3 Implementacija Logstash komponente

Logstash je implementiran kao komponenta koja indeksira i obrađuje podatke logova koje dobije od aplikacionih servera. Podešen je da preko mreže prima podatke od aplikacionih servera, indeksira ih, filtrira i podatke u JSON obliku šalje Elasticsearch-u.

Za potrebe implementiranja Logstash komponente sistema neophodno je bilo instalirati i pokrenuti novi server. Na mreži je dodata nova mašina na kojoj je pokrenut Ubuntu server 20.1 operativni sistem. Podešena mrežna konfiguracija i pridodata IP adresa: 192.168.0.101/24.

Da bi Logstash radio neophondo je da na sistem bude instalirana Java 8 ili Java 11. Izvršena je instalacija Java 11 paketa komandom:

sudo apt install openjdk-11-jdk -y

Instalacija Logstash sistema je izvršena preko paketa repozitorijuma, pa je na početku potrebno pribaviti javni ključ za potpisivanje svih paketa. Prbavljanje je izvršeno komandom:

wget -qO - https://artifacts.elastic.co/GPG-KEY-elasticsearch | sudo apt-key add -

Nakon pribavljanja ključa neophodno je pribaviti i pakete Logstash-a:

echo "deb https://artifacts.elastic.co/packages/7.x/apt stable main" | sudo tee -a /etc/apt/sources.list.d/elastic-7.x.list

Po pribavljanju paketa izvršen je update sistema paketa na operativnom sistemu komandom

sudo apt-get update

Na kraju, instaliran je Logstash komandom

sudo apt-get install logstash

Nakon istalacije kreiran je folder na putanji /etc/logstash/ u kome se nalaze konfiguracioni fajlovi Logstash sistema. Na putanji /etc/logstash/conf.d/ kreiran je konfiguracioni fajl u kome su definisani ulazi, filteri za indeksiranje podataka i izlaz logstasha.

Ulaz podataka u Logstash konfigurisan je tako da podatke prima preko mrežnog interfejsa na portu 5044 . Za ovu konfiguraciju iskorišćen je beats dodatak ulaza logstash-a. Deo konfiguracionog fajla gde je konfigurisan ulaz dat je u nastavku:

```
input {
    beats {
        port => 5044
        include_codec_tag => false
      }
}
```

Za indeksiranje i obradu audit logova urađena je konfiguracija iz dva dela, prvi deo izvlači tip audit loga i celokupnu poruku koju je generisao, u drugom delu se vrši indeksiranje poruke pomoću grok paterna koji je izmešten u fajl na putanji /etc/logstash/patterns. U ovom fajlu su napisani grok paterni koji mogu da indeksiraju svaki tip poruka audit loga. Takođe kod audit loga imamo mutate dodatak koji vrši izmenu tipa podacima i izbacivanje liste polja koju nisu od značaja za dalju analizu događaja.

Izgled dela kofiguracije za indeksiranje audit loga:

```
  if [fields][datatype] == "audit" {
              grok {1
                                                match => [ "message", "type=%{WORD:audit_type}
msg=audit\(%{NUMBER:audit_epoch}:%{NUMBER:audit_counter}\):%{GREEDYDATA:msg}" ]
              }
```

```
        grok {2
            patterns_dir => "/etc/logstash/patterns"
            match => [3
                "msg", "%{AUDITD_1}",
                "msg", "%{AUDITD_2}",
                "msg", "%{AUDITD_3}",
                "msg", "%{AUDITD_4}",
                "msg", "%{AUDITD_5}",
                "msg", "%{AUDITD_6}",
                "msg", "%{AUDITD_7}",
                "msg", "%{AUDITD_8}",
                "msg", "%{AUDITD_9}",
                "msg", "%{AUDITD_10}",
                "msg", "%{AUDITD_11}",
                "msg", "%{AUDITD_12}",
                "msg", "%{AUDITD_13}",
                "msg", "%{AUDITD_14}",
                "msg", "%{AUDITD_15}",
                "msg", "%{AUDITD_16}",
                "msg", "%{AUDITD_17}",
                "msg", "%{AUDITD_18}",
                "msg", "%{AUDITD_19}",
                "msg", "%{AUDITD_20}",
                "msg", "%{AUDITD_21}",
                    "msg", "%{AUDITD_22}",
                    "msg", "%{AUDITD_23}",
                    "msg", "%{AUDITD_24}",
                    "msg", "%{AUDITD_25}",
                "msg", "%{AUDITD_26}",
                "msg", "%{AUDITD_27}",
                "msg", "%{AUDITD_28}",
                "msg", "%{AUDITD_29}",
                "msg", "%{AUDITD_30}",
                "msg", "%{AUDITD_31}",
                "msg", "%{AUDITD_32}"
                ]
            }
        mutate {4
                    remove_field => [ "msg", "agent", "ref", "input", "file", "ecs", "host", "log",
"audit_cmd",   "tags",   "audit_dev",   "audit_egid",   "audit_euid",   "audit_fsgid",   "audit_fsuid",
"audit_suid",   "audit_grantors",   "audit_item",   "audit_items",   "audit_mode",   "audit_name",
"audit_ogid", "audit_oldauid", "audit_oldses", "audit_op", "audit_ouid", "audit_sgid" ]
            convert => ["audit_auid", "integer"]
            convert => ["audit_counter", "integer"]
            convert => ["audit_gid", "integer"]
            convert => ["audit_id", "integer"]
            convert => ["audit_pid", "integer"]
            convert => ["audit_ppid", "integer"]
            convert => ["audit_ses", "integer"]
            convert => ["audit_uid", "integer"]
            add_field => {"logstash_time" => "%{@timestamp}" }
```

```
        }
    }
```

Brojevima od 1 do 4 označeni su delovi konfiguracija a njihovo značenje je sledeće:

1. Grok patern koji izdvaja tip audit loga i poruku.
2. Grok patern koji uključuje spolju listu paterna.
3. Lista koja upoređuje izdvojenu poruku sa spoljnim parternima, odgovarajući patern će prihvatiti poruku i indeksirati je. AUDIT_X označava patern u spoljnom fajlu.
4. Mutate dodatak koji dodaje polje logstash_time , menja tip podataka poljima audi_auid, audit_counter, audit_gid, audit_id, audit_pid, audit_ppid, audit_ses, audit_uid u celobrojnu vrednos, izbacuje navedenu listu polja iz rezultata.

Izgled konfiguracionog fajla grok_auditlog koji služi za indeksiranje audit logova uključenog u konfiguraciji se nalazi na putanji /etc/logstash/patterns, izdvojena konfiguracija je u spoljnom fajlu zbog velikog broja paterna koji bi glavni konfiguracioni fajl učinili nepreglednim i nejasnim. Paterni su poređani u redosledu tako da jajčešće korišćeni paterni budu na vrhu kako bi Logstash brže procesirao podatke. Lista paterna je preuzeta sa interneta ali je bilo potrebno izmeniti svaki patern zbog nepoklapanja sa verzijom audit servisa. Skup paterna iz fajla grok_auditlog je dat u nastavku sa napomenom za koji se tip audit loga koristi.

AUDITD_1 se koristi za indeksiranje audit logova tipa: USER_START, USER_END, CRED_ACQ, USER_ACCT, CRED_DISP, CRED_REFR, USER_AUTH, USER_ERR

```
AUDITD_1 pid=%{INT:audit_pid} uid=%{INT:audit_uid} auid=%{INT:audit_auid}
ses=%{INT:audit_ses}( subj=%{DATA:audit_subject})? msg=\'op=%{DATA:audit_op}
grantors=%{DATA:audit_grantors} acct=\"(%{WORD:user}|\?)\" exe=\"%{DATA:audit_exe}\"
hostname=%{DATA} addr=(%{IPV4:src_ip}|\?) terminal=%{DATA:audit_terminal}
res=%{DATA:audit_res}\'
```

AUDITD_2 se koristi za indeksiranje audit logova tipa: LOGIN

```
AUDITD_2 pid=%{NUMBER:audit_pid} uid=%{NUMBER:audit_uid}(
subj=%{DATA:audit_subject})? old-auid=%{NUMBER:audit_oldauid}
auid=%{NUMBER:audit_auid} (tty=%{DATA:audit_tty} )?old-ses=%{NUMBER:audit_oldses}
ses=%{NUMBER:audit_ses} res=%{GREEDYDATA:audit_res}
```

AUDITD_3 se koristi za indeksiranje audit logova tipa: SYSCALL

```
AUDITD_3 arch=%{DATA} syscall=%{NUMBER:audit_syscall} success=%{WORD:audit_success}
exit=%{INT:audit_exit} a0=%{WORD:audit_a0} a1=%{WORD:audit_a1} a2=%{WORD:audit_a2}
a3=%{WORD:audit_a3} items=%{INT:audit_items} ppid=%{INT:audit_ppid}
pid=%{INT:audit_pid} auid=%{INT:audit_auid} uid=%{INT:audit_uid} gid=%{INT:audit_gid}
euid=%{INT:audit_euid} suid=%{INT:audit_suid} fsuid=%{INT:audit_fsuid}
egid=%{INT:audit_egid} sgid=%{INT:audit_sgid} fsgid=%{INT:audit_fsgid}
tty=%{DATA:audit_tty} ses=%{INT:audit_ses} comm=(\")?%{DATA:audit_comm}(\")?
exe=\"%{DATA:audit_exe}\" (subj=%{DATA:audit_subj} )?key=\"%{DATA:audit_key}\"
```

AUDITD_4 se koristi za indeksiranje audit logova tipa: NETFILTER_CFG

AUDITD_4 table=%{WORD:audit_table} family=%{INT:audit_family}
entries=%{INT:audit_entries}

AUDITD_5 se koristi za indeksiranje audit logova tipa: CRYPTO_KEY_USER

AUDITD_5 pid=%{NUMBER:audit_pid} uid=%{NUMBER:audit_uid}
auid=%{NUMBER:audit_auid} ses=%{NUMBER:audit_ses}(subj=%{DATA:audit_subject})?
msg=\'op=%{DATA:audit_op} kind=%{WORD:audit_kind} fp=%{DATA:audit_fp}
direction=%{DATA:audit_direction} spid=%{INT:audit_spid} suid=%{INT:audit_suid}(
rport=%{INT:src_port} laddr=%{IPV4:dst_ip}
lport=%{INT:dst_port})?\s+exe=\"%{DATA:audit_exe}\" hostname=%{DATA}
addr=(%{IPV4:src_ip}|\?) terminal=%{DATA:audit_terminal} res=%{DATA:audit_res}\'

AUDITD_6/7/8/25/26 se koristi za indeksiranje audit logova tipa: AVC

AUDITD_6 avc:\s+%{WORD:audit_avcaction}\s+\{ %{WORD:audit_avcop} \}
for\s+pid=%{INT:audit_pid} comm=\"%{WORD:audit_comm}\" path=\"%{DATA:audit_path}\"
dev=\"%{WORD:audit_dev}\" ino=%{INT:audit_ino} scontext=%{DATA:audit_scontext}
tcontext=%{DATA:audit_tcontext} tclass=%{WORD:audit_tclass}
AUDITD_7 avc:\s+%{WORD:audit_avcaction}\s+\{ %{WORD:audit_avcop} \}
for\s+pid=%{INT:audit_pid} comm=\"%{WORD:audit_comm}\"
capability=%{INT:audit_capability}\s+scontext=%{DATA:audit_scontext}
tcontext=%{DATA:audit_tcontext} tclass=%{WORD:audit_tclass}
AUDITD_8 avc:\s+%{WORD:audit_avcaction}\s+\{ %{WORD:audit_avcop} \}
for\s+pid=%{INT:audit_pid} comm=\"%{WORD:audit_comm}\" name=\"%{DATA:audit_name}\"
scontext=%{DATA:audit_scontext} tcontext=%{DATA:audit_tcontext}
tclass=%{WORD:audit_tclass}
AUDITD_25 avc:\s+%{WORD:audit_avcaction}\s+\{ %{WORD:audit_avcop} \}
for\s+pid=%{INT:audit_pid} comm=\"%{WORD:audit_comm}\" name=\"%{DATA:audit_name}\"
dev=\"%{WORD:audit_dev}\" ino=%{INT:audit_ino} scontext=%{DATA:audit_scontext}
tcontext=%{DATA:audit_tcontext} tclass=%{DATA:audit_tclass}
AUDITD_26 avc:\s+%{WORD:audit_avcaction}\s+\{ %{WORD:audit_avcop} \}
for\s+pid=%{INT:audit_pid} comm=%{WORD:audit_comm}
dest=%{INT:audit_dest}\s+scontext=%{DATA:audit_scontext} tcontext=%{DATA:audit_tcontext}
tclass=%{WORD:audit_tclass}

AUDITD_9 se koristi za indeksiranje audit logova tipa: SERVICE_START, SERVICE_STOP

AUDITD_9 pid=%{NUMBER:audit_pid} uid=%{NUMBER:audit_uid}
auid=%{NUMBER:audit_auid} ses=%{NUMBER:audit_ses}(subj=%{DATA:audit_subject})?
msg=\'(unit=%{DATA:audit_unit})? comm=%{DATA:audit_comm} exe=\"%{DATA:audit_exe}\"
hostname=%{DATA} addr=(%{IPV4:src_ip}|\?) terminal=%{DATA:audit_terminal}
res=%{DATA:audit_res}\'

AUDITD_10 se koristi za indeksiranje audit logova tipa: CRYPTO_SESSION

AUDITD_10 pid=%{INT:audit_pid} uid=%{INT:audit_uid} auid=%{INT:audit_auid}
ses=%{INT:audit_ses}(subj=%{DATA:audit_subject})? msg=\'op=%{DATA:audit_op}
direction=%{DATA:audit_direction} cipher=%{DATA:audit_cipher} ksize=%{INT:audit_ksize}

mac=%{DATA:audit_mac} pfs=%{DATA:audit_pfs} spid=%{INT:audit_spid}
suid=%{INT:audit_suid} rport=%{INT:src_port} laddr=%{IPV4:dst_ip}
lport=%{INT:dst_port}\s+exe=\"%{DATA:audit_exe}\" hostname=%{DATA}
addr=(%{IPV4:src_ip}|\?) terminal=%{DATA:audit_terminal} res=%{DATA:audit_res}\'

AUDITD_11/12 se koristi za indeksiranje audit logova tipa: USER_LOGIN, USER_LOGOUT, USER_CHAUTHOK, ADD_USER, ADD_GROUP, USER_ERR

AUDITD_11 pid=%{NUMBER:audit_pid} uid=%{NUMBER:audit_uid}
auid=%{NUMBER:audit_auid} ses=%{NUMBER:audit_ses}(subj=%{DATA:audit_subject})?
msg=\'op=%{DATA:audit_op} id=%{INT:audit_id} exe=\"%{DATA:audit_exe}\"
hostname=%{DATA} addr=(%{IPV4:src_ip}|\?) terminal=%{DATA:audit_terminal}
res=%{DATA:audit_res}\'
AUDITD_12 pid=%{NUMBER:audit_pid} uid=%{NUMBER:audit_uid}
auid=%{NUMBER:audit_auid} ses=%{NUMBER:audit_ses}(subj=%{DATA:audit_subject})?
msg=\'op=%{DATA:audit_op} acct=\"%{DATA:user}\" exe=\"%{DATA:audit_exe}\"
hostname=%{DATA} addr=(%{IPV4:src_ip}|\?) terminal=%{DATA:audit_terminal}
res=%{DATA:audit_res}\'

AUDITD_13 se koristi za indeksiranje audit logova tipa: USER_ROLE_CHANGE

AUDITD_13 pid=%{NUMBER:audit_pid} uid=%{NUMBER:audit_uid}
auid=%{NUMBER:audit_auid} ses=%{NUMBER:audit_ses}(subj=%{DATA:audit_subject})?
msg=\'pam: default-context=%{DATA:audit_defaultcontext}
selected-context=%{DATA:audit_selectedcontext} exe=\"%{DATA:audit_exe}\" hostname=%{DATA}
addr=(%{IPV4:src_ip}|\?) terminal=%{DATA:audit_terminal} res=%{DATA:audit_res}\'

AUDITD_14 se koristi za indeksiranje audit logova tipa: USER_CMD

AUDITD_14 pid=%{NUMBER:audit_pid} uid=%{NUMBER:audit_uid}
auid=%{NUMBER:audit_auid} ses=%{NUMBER:audit_ses}(subj=%{DATA:audit_subject})?
msg=\'cwd=%{DATA:audit_cwd} cmd=%{DATA:audit_cmd} terminal=%{DATA:audit_terminal}
res=%{DATA:audit_res}\'

AUDITD_15/16 se koristi za indeksiranje audit logova tipa: USER_AVC

AUDITD_15 pid=%{NUMBER:audit_pid} uid=%{NUMBER:audit_uid}
auid=%{NUMBER:audit_auid} ses=%{NUMBER:audit_ses}(subj=%{DATA:audit_subject})?
msg=\'avc:\s+%{WORD:audit_avcaction}\s+\{ %{WORD:audit_avcop} \}\s+for
auid=%{INT:audit_auid} uid=%{INT:audit_uid} gid=%{INT:audit_gid}
path=\"%{DATA:audit_path}\" cmdline=\"%{DATA:audit_cmdline}\"
scontext=%{DATA:audit_scontext} tcontext=%{DATA:audit_tcontext}
tclass=%{WORD:audit_tclass}\s+exe=\"%{DATA:audit_exe}\" sauid=%{INT:audit_sauid}
hostname=%{DATA} addr=(%{IPV4:src_ip}|\?) terminal=%{DATA:audit_terminal}\'
AUDITD_16 pid=%{NUMBER:audit_pid} uid=%{NUMBER:audit_uid}
auid=%{NUMBER:audit_auid} ses=%{NUMBER:audit_ses}(subj=%{DATA:audit_subject})?
msg=\'avc:\s+%{DATA:audit_avcmsg}\s+exe=\"%{DATA:audit_exe}\" sauid=%{INT:audit_sauid}
hostname=%{DATA} addr=(%{IPV4:src_ip}|\?) terminal=%{DATA:audit_terminal}\'

AUDITD_17 se koristi za indeksiranje audit logova tipa: ANOM_PROMISCUOUS

AUDITD_17 dev=%{WORD:audit_dev} prom=%{INT:audit_prom} old_prom=%{INT:audit_oldprom} auid=%{INT:audit_auid} uid=%{INT:audit_uid} gid=%{INT:audit_gid} ses=%{INT:audit_ses}

AUDITD_18 se koristi za indeksiranje audit logova tipa: MAC_STATUS

AUDITD_18 enforcing=%{INT:audit_enforcing} old_enforcing=%{INT:audit_oldenforcing} auid=%{INT:audit_auid} ses=%{INT:audit_ses}

AUDITD_19 se koristi za indeksiranje audit logova tipa: ANON_ABEND

AUDITD_19 auid=%{NUMBER:audit_auid} uid=%{NUMBER:audit_uid} gid=%{INT:audit_gid} ses=%{INT:audit_ses}(subj=%{DATA:audit_subject})? pid=%{INT:audit_pid} comm=\"%{WORD:audit_comm}\" reason=\"%{DATA:audit_reason}\" sig=%{INT:audit_sig}

AUDITD_20 se koristi za indeksiranje audit logova tipa: USER_MGMT

AUDITD_20 pid=%{NUMBER:audit_pid} uid=%{NUMBER:audit_uid} auid=%{NUMBER:audit_auid} ses=%{NUMBER:audit_ses}(subj=%{DATA:audit_subject})? msg=\'op=%{DATA:audit_op} grp=\"%{WORD:audit_id}\" acct=\"%{DATA:user}\" exe=\"%{DATA:audit_exe}\" hostname=%{DATA} addr=(%{IPV4:src_ip}|\?) terminal=%{DATA:audit_terminal} res=%{DATA:audit_res}\'

AUDITD_21 se koristi za indeksiranje audit logova tipa: MAC_POLICY_LOADED

AUDITD_21 policy loaded auid=%{NUMBER:audit_auid} ses=%{NUMBER:audit_ses}

AUDITD_22 se koristi za indeksiranje audit logova tipa: EXECVE

AUDITD_22 argc=%{INT:audit_argc} a0=\"%{DATA:audit_a0}\"%{GREEDYDATA:audit_args}

AUDITD_22 se koristi za indeksiranje audit logova tipa: CWD

AUDITD_23 cwd=\"%{DATA:audit_cwd}\"

AUDITD_24/27 se koristi za indeksiranje audit logova tipa: PATH

AUDITD_24 item=%{INT:audit_item} name=\"?%{DATA:audit_name}\"? node=%{INT:audit_inode}dev=%{DATA:audit_dev}mode=%{INT:audit_mode}ouid=%{INT:audit_o uid} ogid=%{INT:audit_ogid} rdev=%{DATA:audit_rdev} (obj=%{DATA:audit_obj})?nametype=%{WORD:audit_nametype}
AUDITD_27 item=%{INT:audit_item} name=\"?%{DATA:audit_name}\"? nametype=%{DATA:audit_nametype} cap_fp=%{DATA:audit_cap_fp} cap_fi=%{DATA:audit_cap_fi} cap_fe=%{DATA:audit_cap_fe} cap_fver=%{DATA:audit_cap_fver}

AUDITD_28 se koristi za indeksiranje audit logova tipa: DAEMON_END, DAEMON_START

AUDITD_28 op=%{DATA:audit_op} auid=%{INT:audit_auid} pid=%{NUMBER:audit_pid}(subj=%{DATA:audit_subject})? res=%{DATA:audit_res}

AUDITD_29/30 se koristi za indeksiranje audit logova tipa: PROCTITLE

AUDITD_29 proctitle=%{WORD:proctitle}
AUDITD_30 proctitle=\"%{WORD:proctitle}\"

AUDITD_31/32 se koristi za indeksiranje audit logova tipa: SOCKADDR

AUDITD_31 saddr=%{WORD:saddr}
AUDITD_32 saddr=\"%{WORD:saddr}\"

Izgled audit loga pre indeksiranja:

type=SYSCALL msg=audit(1632472510.656:49979): arch=c000003e syscall=42 success=no exit=-115 a0=5 a1=c0006c646c a2=10 a3=0 items=0 ppid=1 pid=97537 auid=4294967295 uid=0 gid=0 euid=0 suid=0 fsuid=0 egid=0 sgid=0 fsgid=0 tty=(none) ses=4294967295 comm="filebeat" exe="/usr/share/filebeat/bin/filebeat" key="aplication_directory"

Izgled audit loga nakon indeksiranja:


```
{
      "audit_ppid" => 1,
      "audit_type" => "SYSCALL",
       "audit_uid" => 0,
       "audit_ses" => 4294967295,
        "@version" => "1",
       "audit_tty" => "(none)",
      "audit_comm" => "filebeat",
       "audit_gid" => 0,
       "audit_key" => "aplication_directory",
        "audit_a0" => "5",
      "audit_success" => "no",
      "audit_syscall" => "42",
      "audit_counter" => 49979,
      "logstash_time" => "2021-09-24T09:15:24.468Z",
         "audit_auid" => 4294967295,
         "fields" => {
      "datatype" => "audit"
   },
        "audit_a2" => "10",
     "@timestamp" => 2021-09-24T09:15:24.468Z,
    "audit_epoch" => "1632472510.656",
       "audit_exe" => "/usr/share/filebeat/bin/filebeat",
        "audit_a3" => "0",
       "audit_a1" => "c0006c646c",
       "audit_pid" => 97537,
      "audit_exit" => "-115"
}
```

Izlaz podataka je konfigurisan tako da šalje podatke Elasticsearch-u uz indeks koji označava tip podatka, odnosno audit logove. Indeks je podešen pomoću elasticsearch dodatka izlaza. Izlaz je konfigurisan tako da podatke šalje celom elasticsearch serveru. Deo konfiguracionog fajla gde je konfigurisan izlaz dat je u nastavku

```
output {
   elasticsearch {
      hosts => ["192.168.0.102:9200"]
   }
}
```

4.2.4 Implementacija Elasticsearch komponente

Elasticsearh je implementiran kao komponenta za čuvanje, skladištenje i pružanje servisa pristupa indeksiranim podacima. Podešen je tako da preko mreže prima podatke od Logstash komponente, skladišti u indekse i preko sistema upita omogući dostupnost podataka Kibana komponenti.

Za potrebe implementiranja Elasticsearch komponente sistema neophodno je bilo instalirati i pokrenuti novi server. Na mreži je dodata nova mašina na kojoj je pokrenut Ubuntu server 20.1 operativni sistem. Podešena mrežna konfiguracija i pridodata IP adresa: 192.168.0.102/24.

Instalacija Elasticsearch sistema je izvršena preko paketa repozitorijuma, pa je na početku potrebno pribaviti javni ključ za potpisivanje svih paketa. Prbavljanje je izvršeno komandom:

wget -qO - https://artifacts.elastic.co/GPG-KEY-elasticsearch | sudo gpg --dearmor -o /usr/share/keyrings/elasticsearch-keyring.gpg

Nakon pribavljanja ključa neophodno je instalirati paket apt-transport-https

sudo apt-get install apt-transport-https

Potrebno je sačuvati definiciju repozitorijuma na putanji /etc/apt/sources.list.d/elastic-8.x.list

echo "deb [signed-by=/usr/share/keyrings/elasticsearch-keyring.gpg] https://artifacts.elastic.co/packages/8.x/apt stable main" | sudo tee /etc/apt/sources.list.d/elastic-8.x.list

Po pribavljanju paketa izvršen je update sistema paketa na operativnom sistemu komandom

sudo apt-get update

Na kraju, instaliran je Logstash komandom

sudo apt-get install elasticsearch

Nakon instalacije potrebno je pokrenuti Elasticsearch servis i omogućiti mrežni saobracaj na portu: 9200.

Elastisearch je pokrenut na serveru da radi kao servis i omogućeno je startovanje prilikom restarta odnosno dodat je u listu za startovanje servisa prilikom startovanja operativnog sistema.

4.2.5 Implementacija Kibana komponente

Kibana je implementirana na sistem kao poslednja komponenta koja omogućava vizualni prikaz informacija. Preko definisanih upita prikuplja podatke sa Elastisearch komponente i prikazuje ih preko vizualizacionih elemenata kao sto su grafikoni, tabele, dijagrami…

Za potrebe implementiranja Kibana komponente sistema neophodno je bilo instalirati i pokrenuti novi server. Na mreži je dodata nova mašina na kojoj je pokrenut Ubuntu server 20.1 operativni sistem. Podešena mrežna konfiguracija i pridodata IP adresa: 192.168.0.103/24.

Instalacija Kibane je izvršena preko paketa repozitorijuma, pa je na početku potrebno pribaviti javni ključ za potpisivanje svih paketa. Prbavljanje je izvršeno komandom:

wget -qO - https://artifacts.elastic.co/GPG-KEY-elasticsearch | sudo apt-key add -

Nakon pribavljanja ključa neophodno je instalirati paket apt-transport-https

sudo apt-get install apt-transport-https

Potrebno je sačuvati definiciju repozitorijuma na putanji /etc/apt/sources.list.d/elastic-8.x.list

echo "deb [signed-by=/usr/share/keyrings/elasticsearch-keyring.gpg] https://artifacts.elastic.co/packages/8.x/apt stable main" | sudo tee /etc/apt/sources.list.d/elastic-8.x.list

Po pribavljanju paketa izvršen je update sistema paketa na operativnom sistemu komandom

sudo apt-get update

Na kraju, instaliran je Logstash komandom

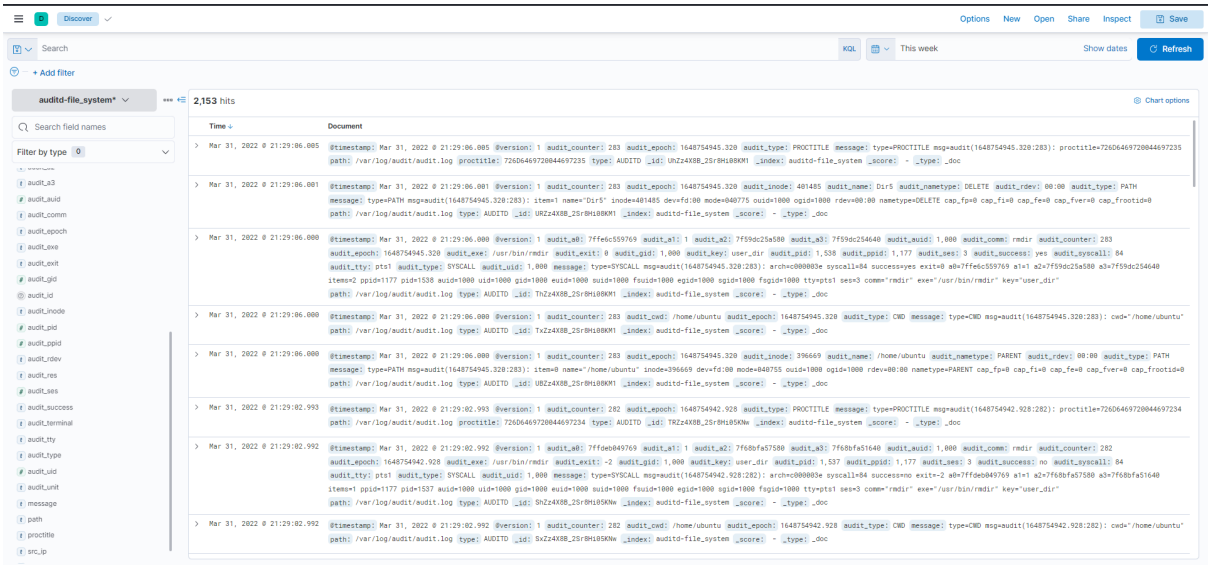
sudo apt-get install kibana

Nakon instalacije potrebno je pokrenuti Kibana servis i omogućiti mrežni saobracaj na portu: 5601. Nakon ovoga moguce je pristupiti Kibana Web interfejsu preko bilo kog internet pregledača.

Kibana je pokrenuta na serveru da radi kao servis i omogućeno je startovanje prilikom restarta odnosno dodata je u listu za startovanje servisa prilikom startovanja operativnog sistema.

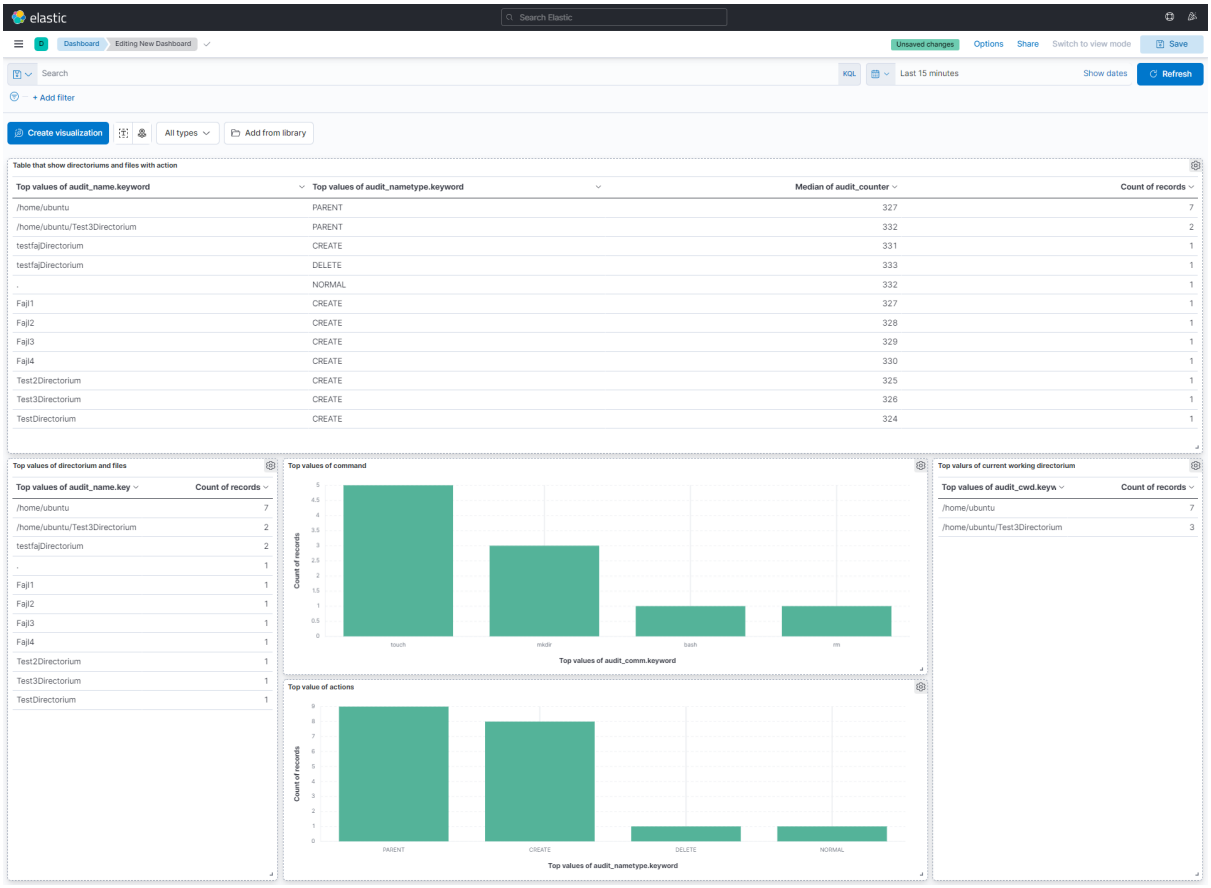
U nastavku će biti prikazano nekolikoliko slika izgleda krajnjih podataka na Kibana vizualizacionom interfejsu:

Slika 4 nam daje prikaz podataka u realnom vremenu. Na slici možemo da vidimo indeksirane podatke audit loga pristiglih redom onako kako su se dešavali na aplikacionom serveru. Ovde mozemo da vidimo celokupan izgled zabeleženog dogadjaja kao i da izvršimo pretragu uz pomoć KQL (Kibana query language) koji nma omogućava pretragu podataka po željenim intervalima i specifičnim zahtevima.



Slika 4. Izgleg indeksiranih logova na Kibana vizualizacionom interfejsu u realnom vremenu

Slika 5 nam predstavlja specifično uredjen prikaz podataka u realnom vremenu sa izdvojenim pojedinim poljima iz logova koji daju sliku o dešavanjima na serveru. Moguće je pratiti promene na fajl sistemu u relnom vremenu sa zakašnjenjem svega 0.025s, takodje moguće je izabrati vreme zadržavanja podataka na prikazu kao i prikaz željenih informacija.



Slika 5. Izgleg podataka na Kibana vizualizacionom interfejsu u realnom vremenu

Ovako implementiran sistem pruža olakšanu kontrolu, praćenje i fonziku sistema dajući potpunu slobodu administratoru prilikom istraživanja podataka.

5. Literatura

[1] NXLog User Guide Documentation, Linux Audit System [ne mreži]
Dostupno na: https://nxlog.co/documentation/nxlog-user-guide/linux-audit.html
[2] SysTutorial Audit, Linux Man Pages [na mreži]
Dostupno na: https://www.systutorials.com/docs/linux/man/7-audit/
[3] Linux Audit Documentation Project [na mreži]
Dostupno na: https://github.com/linux-audit/audit-documentation/wiki
[4] Karen Kent, Murugiah Souppaya, Guide to Computer Security Log Management, National Institute of Standards and Technology Gaithersburg, Septembar 2006
[5] Filebeat Documentation [na mreži] Dostupno na:
https://www.elastic.co/guide/en/beats/filebeat/current/index.html
[6] Elasticsearch Documentation [na mreži] Dostupno na:
https://www.elastic.co/guide/en/elasticsearch/reference/current/index.html
[7] Logstash Documentation [na mreži] Dostupno na:
https://www.elastic.co/guide/en/logstash/current/index.html
[8] Kibana Documentation [na mreži]Dostupno na:
https://www.elastic.co/guide/en/kibana/current/index.html