SVEUČILIŠTE U ZAGREBU
**FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA**

DIPLOMSKI RAD br. 705

# Analiza mogućnosti povezivanja alata OSSIM s alatima za strojno učenje i statističku analizu

Dino Sulić

Zagreb, Lipanj 2014.

Zahvala

*Zahvaljujem mentoru doc. dr. sc. Stjepanu Grošu za prijedloge i savjete pri izradi diplomskog rada*

## SAŽETAK

Ovaj diplomski rad proučava alat OSSIM i kako se mogu iskoristiti alati za strojno učenje kako bi se poboljšao njegov rad. Na početku se objašnjava kako ugraditi alat u lokalnu mrežu kako bi počeo nadzirati mrežni promet. OSSIM će na temelju mrežnog prometa generirati alarme kojima pridružuje stupanj rizika. Opisan je način na koji se mogu dohvatiti podatci iz baze podataka kako bi se nad njima moglo primijeniti strojno učenje. Dan je jednostavan primjer algoritma strojnog učenja gdje je cilj dobiti realniji stupanj rizika.

## ABSTRACT

This diploma thesis examines the OSSIM tool and how can the tools for machine learning be used for its improvement. At first it explains how to install it in local area network so it can begin to monitor network traffic. Based on network traffic, OSSIM will generate alarms and assign them risk level. It is described the way to retrieve data which can be used for machine learning algorithms. A simple example is given of machine learning algorithm where the goal is to get a more realistic level of risk.

# Sadržaj

# 1. Uvod

Internet od svojih početaka, iz godine u godinu koristi sve veći broj ljudi. Tako je Internet postao dio svakodnevice većini ljudi u razvijenom svijetu. S velikim brojem korisnika te s velikim brojem funkcija koje se mogu obaviti putem Interneta u pitanje dolazi i sigurnost. U današnje vrijeme postoji veliki broj različitih napada. Kako bi uspjeli prepoznati napad prije nego napad ostavi ozbiljnije posljedice na mrežu, treba pratiti korisnike i promet u mreži. Na temelju tih podataka možemo otkriti zlonamjernog korisnika i vrstu napada te spriječiti napad.

Problem se pojavljuje kada treba odabrati alate kojima se koristiti. Postoji puno alata otvorenog koda koji prikupljaju razne podatke na mreži. Ideja je bila odabrati nekoliko takvih alata te ih integrirati u jedno rješenje koje će nekim pametnim algoritmom obraditi prikupljene podatke te donijeti određene zaključke o sigurnosti mreže. Pretraživanjem postojećih rješenja nađen je alat OSSIM koji integrira veliki broj alata. Koristit će se to rješenje kako se ne bi bespotrebno razvijao alat koji već postoji.

Open Source Security Information Management (OSSIM) je sustav za upravljanje sigurnosnim informacijama i događajima (Security Information and Event Management, SIEM) otvorenog koda koji integrira razne mrežne alate kako bi pomogao mrežnim administratorima u nadzoru mreže i sprječavanju napada na mrežu. Projekt OSSIM je pokrenut 2003. godine suradnjom Dominique Karga i Julio Casala. Par godina kasnije to im postaje temelj za njihovu firmu AlienVault koja danas prodaje komercijaliziranu verziju OSSIM-a koja sadrži neke dodane funkcionalnosti. Projekt se dalje razvija pod pokroviteljstvom te firme. Trenutno se distribuira zajedno sa operacijskim sustavom Debian Linux.

OSSIM se sastoji od 3 komponente:

- Baza podataka
- Poslužitelj
- Agenti

Poslužitelj služi za upravljanje cijelog sustava kao i za prikaz svih važnijih događaja na mreži. U sustavu je dovoljan jedan poslužitelj. Agenata se može postaviti koliko god treba i gdje god u mreži treba. Jedan od agenata može biti i poslužitelj. Agenti snimaju mrežni promet te šalju te podatke na poslužitelj koji ih pohranjuje u bazu podataka.

Alata koji su integrirani unutar OSSIM-a je puno, no neki od značajnijih su:

- Snort
- OpenVas
- Nagios
- OSSEC
- Nmap
- Arpwatch

Svi ti alati generiraju podatke koje OSSIM sakuplja i pohranjuje u bazu podataka. OSSIM koristi svoje mehanizme kako bi prepoznao podatke koji ukazuju na neku potencijalno opasnu situaciju. Postavlja se pitanje može li se iskoristiti kakav algoritam strojnog učenja kako bi prepoznavanje opasnosti bilo što preciznije.

Cilj ovog rada je analizirati OSSIM sustav i proučiti mogućnost nadogradnje sustava s nekim alatom za strojno učenje i statističku analizu.

U prvom poglavlju se opisuje proces ugradnje sustava u lokalnu mrežu. Ugradnja uključuje instalaciju poslužitelja i agenata te njihovo međusobno povezivanje. Drugo poglavlje ukratko opisuje funkcionalnost sustava. U trećem poglavlju se analizira baza podataka na poslužitelju. Proučava se međusobni odnos tablica u bazi kao i njihova interna struktura. Detaljnije se obrađuju tablice u koje se spremaju prikupljeni podatci. U četvrtom poglavlju se opisuje jedan od načina primjene algoritama grupiranja nad prikupljenim podatcima. Na kraju se donosi zaključak ima li alat OSSIM mogućnost nadogradnje alatima za strojno učenje. U poglavlju "Literatura" imamo pregled svih korištenih referenci.

# 2. Instalacija sustava OSSIM

Pod ugradnjom sustava podrazumijeva se instalacija poslužitelja i proizvoljnog broja agenata. Instalacijski paket poslužitelja i agenta se može preuzeti na stranicama projekta [1]. Taj instalacijski paket je zapravo slika DVD medija na kojoj se nalazi instalacija operacijskog sustava koji sadrži potrebne alate, što znači da je potrebno instalirati i novi operacijski sustav. Ukoliko se treba instalirati na postojeći operacijski sustav, za to se može koristiti izvorni kod koji je dostupan na git stranicama projekta [2].

## *2.1. Instalacija korištenjem instalacijskih paketa*

Instalacija ovim putem je puno jednostavnija. Na početku je potrebno preuzeti instalacijski paket sa spomenute stranice. Paket je potrebno snimiti na prijenosni medij i ponovno pokrenuti računalo. Prilikom pokretanja odabire se taj medij za pokretanje operacijskog sustava. Jednom kada se pokrene, na ekranu se prikazuje izbornik u kojem se bira između instalacije poslužitelja ili agenta. Nakon odabira poslužitelja počinje instalacija koja će kroz nekoliko koraka tražiti da se ispune bitni podatci poput regionalnih postavki, IP adrese poslužitelja, IP adrese mreže, IP adrese DNS poslužitelja i lozinke "root" korisnika.

Nakon instalacije poslužitelj se može upravljati direktno na tom računalu, preko web sučelja ili putem SSH protokola . Web sučelju se pristupa na IP adresi koja je prethodno bila unesena. Sučelje pruža uvid u podatke prikupljene na mreži kao i mogućnost dodatnog podešavanja sustava.

## 2.2. Instalacija iz izvornog koda

Ovakav način instalacije je dosta kompliciraniji od prethodnog. Instalacija je podijeljena u tri koraka:

- Instalacija baza podataka
- Instalacija poslužitelja
- Instalacija korištenih alata

U prvom koraku treba kreirati bazu podataka te ju pripremiti za pohranu podataka. OSSIM koristi MySQL bazu podataka koja dolazi instalirana s većinom distribucija Linux operacijskog sustava. Datoteke koje opisuju bazu podataka i skripte za kreiranje te baze se nalaze u izvornom kodu.

Instalacija poslužitelja može prouzrokovati razne probleme ako računalo nema instalirane alate i biblioteke potrebne za rad poslužitelja. Ako su svi preduvjeti zadovoljeni, potrebno je instalirati nekoliko alata među kojima su: phpGACL [3], ACID [4], alat za generiranje PDF dokumenata [5].

Jedan od problema do kojeg se dolazi pri ovakvoj instalaciji je taj da je dokumentacija iz 2004. godine [6]. U dokumentaciji je primjer instalacije zastarjele verzije sustava. Jedan od razloga zbog kojeg je došlo do toga je i jednostavnija instalacija spomenuta ranije.

## 2.3. Instalacija agenta

Agent se može instalirati pomoću distribucijskog diska što je jednostavan proces ili se može povezati računalo koje već ima instalirane neke od alata koji se koriste na agentima. Ukoliko već imamo računalo na kojem se nalazi alat koji se može integrirati sa sustavom, potrebno je to računalo povezati s poslužiteljem kako bi poslužitelj mogao obraditi prikupljene podatke. Jedan od alata koje je moguće na taj način povezati s poslužiteljem je Snort [7].

Snort je jedan od najrasprostranjenijih sustava za otkrivanje napada (Intrusion Detection System, IDS). Njegova zadaća je da prati mrežni promet te javi upozorenje ako smatra da je došlo do napada na mrežu. Snort funkcionira tako da mrežni promet uspoređuje s pravilima koja dolaze s alatom. Ako se utvrdi da paket odgovara nekom pravilu tj. ako postoji zapisano pravilo kojim se prepoznaje napad, onda javlja upozorenje.

Kako bi se Snort povezao s poslužiteljem, potrebno je podatke koje generira Snort preusmjeriti prema poslužitelju. Za preusmjeravanje podataka se koristi standardni alat za bilježenje zapisa u dnevnik, RSYSLOG. To je alat koji u većini slučajeva dolazi s instalacijom Linux operacijskog sustava te ga to čini dobrim izborom za ovaj zadatak. Da bi namjestili rsyslog da nam šalje podatke na poslužitelj, moramo dodati neke naredbe u njegovu konfiguracijsku datoteku. Konfiguracijska datoteka se obično nalazi u direktoriju `/etc` pod nazivom `rsyslog.conf`. U tu datoteku potrebno je dodati:

```
$SystemLogRateLimitInterval 10
```

```
$SystemLogRateLimitBurst 500
local1.info @@192.168.1.180:514
```

Prve dvije linije omogućuju veći protok podataka dok trećom naredbom govorimo alatu da šalje podatke kanala `local1.info` na zadanu IP adresu poslužitelja koja je u ovom slučaju 192.168.1.180 i pristupnu točku 514.

U sljedećem koraku moramo podesiti poslužitelj da primi poslane podatke i pohrani ih u bazu u tablice koje pripadaju Snort komponenti. Kako bi to napravili treba stvoriti novu datoteku koja će se zvati `remote-snort-sensor.conf`. Tu datoteku stvaramo u direktoriju `/etc/rsyslog.d/`. U tu datoteku treba zapisati:

```
$ModLoad imtcp
$InputTCPserverRun 514
if $fromhost-ip == '192.168.1.95' then
/var/log/snort/alert
&~
if $fromhost-ip == '192.168.1.40' then
/var/log/snort/alert
&~
```

Prve dvije linije definiraju da se otvaraju TCP vrata 514. Ostale naredbe provjeravaju s koje je IP adrese došao podatak. Ako se radi o računalu s kojeg želimo dobivati podatke, te podatke preusmjeravamo. Podatci se preusmjeravaju u datoteku u koju ćemo spremati sve podatke generirane od alata Snort sa svih agenata. U ovom primjeru radi se o datoteci `/var/log/snort/alert`. Linijom `&~` alatu se daje do znanja da je taj podatak obrađen tako da ne mora povjeravati ostale uvjete. Na taj je način spriječeno da se zapisuju nepotrebni zapisi u ostale dnevnike.

Kako bi se osiguralo da alat rsyslog prihvati i spremi sve dobivene podatke, potrebno je povećati brzinu primanja podataka. To se može napraviti dodavanjem par linija u konfiguracijsku datoteku tog alata. Slično kao i ranije, datoteku `/etc/rsyslog.conf` nadopunjujemo s naredbama:

```
$SystemLogRateLimitInterval 10
$SystemLogRateLimitBurst 500
```

Nakon dodavanja tih naredbi, potrebno je zaustaviti i ponovno pokrenuti alat rsyslog na svakom računalu na kojem se mijenjala konfiguracijska datoteka. Jednom kada se alat ponovno pokrene, poslužitelj bi trebao biti povezan sa računalima na kojima se izvršava Snort. To se može provjeriti tako da se na računalu agenta pokrene naredba:

```
logger -p local1.info "Test from Snort"
```

koja bilježi zapis na kanalu definiranom za Snort. Ukoliko su računala uspješno povezana, na poslužitelju bi se u datoteci `/var/log/snort/alert` trebao pojaviti zapis:

```
Jun 10 08:04:08 xyzzy bill: Test from Snort
```

koji govori da je alat rsyslog preusmjerio podatak prema poslužitelju koji ga je primio i pohranio u odgovarajuću datoteku.

Ukoliko u OSSIM alatu nije omogućena opcija `snort_syslog` potrebno ju je omogućiti. Kako bi se to provjerilo ili eventualno omogućilo, treba pristupiti poslužitelju preko ssh protokola ili direktno na tom računalu. Jednom kad je dobiven pristup računalu kao root korisnik, naredbom `alienvault-setup` se ulazi u konfiguracijski izbornik poslužitelja. U izborniku je potrebno odabrati opciju `Configure Sensor` te zatim opciju `Configure Data Source Plugins`. Na dobivenom popisu treba pronaći `snort_syslog` te provjeriti da li je uključen taj dodatak tj. da li se nalazi zvjezdica (*) kraj imena. Ukoliko dodatak nije označen pritiskom na razmak se uključuje. Jednom kad su se prihvatile sve promjene OSSIM će pokrenuti potrebni dodatak.

Posljednja promjena koju je potrebno napraviti je podesiti Snort tako da se alat rsyslog obavijesti svaki put kada se generira alarm. Ta se promjena radi u datoteci `/etc/snort/snort.conf` na računalu agenta. Potrebno je pronaći odlomak koji se zove `Step #6: Configure output plugins` i dodati linije:

```
output alert_fast: snort.fast
output alert_syslog: LOG_LOCAL1 LOG_INFO
```

Ovime je postavljeno da Snort sprema generirane alarme u datoteku `snort.fast` i da šalje te alarme alatu rsyslog na kanal `local1.info`.

## 2.4. Kontrola pristupa poslužitelju

Jednom kad je poslužitelj instaliran i pokrenut, može mu se pristupiti od bilo kuda preko njegove IP adrese. Naravno potrebno je znati korisničko ime i lozinku za pristup ali kad se radi o sigurnosti to možda nije dovoljno. Kako bi se ograničio pristup poslužitelju potrebno je poduzeti još neke mjere. Jedno od rješenja koje se nudi je mijenjanje postavka sigurnosne stijene. OSSIM ima ugrađenu sigurnosnu stijenu kojoj se mogu mijenjati IP tablice i postaviti ih kako najbolje odgovara nečijim potrebama. Na taj način se može dozvoliti pristup poslužitelju samo sa određenih IP adresa. Datoteka koju je potrebno mijenjati je `/etc/ossim_firewall`.

U toj datoteci se nalaze pravila koja definiraju akcije za određeni mrežni promet. Svako pravilo specificira jednu od tri moguće akcije:

- dozvoli (*accept*)
- odbaci (*reject*)
- ispusti (*drop*)

Pravilo može provjeravati razne uvjete među kojima su IP adresa i pristupna točka izvorišta i odredišta, protokol, radi li se o dolaznom ili odlaznom prometu. Primjer takvog pravila je:

```
-A INPUT -s (IP adresa) -p tcp -m state --state NEW -m
tcp  --dport (pristupna točka) -j ACCEPT
```

Ovom naredbom se prihvaća dolazni promet koji kreira TCP segment i dolazi sa zadane IP adrese na zadanu pristupnu točku.

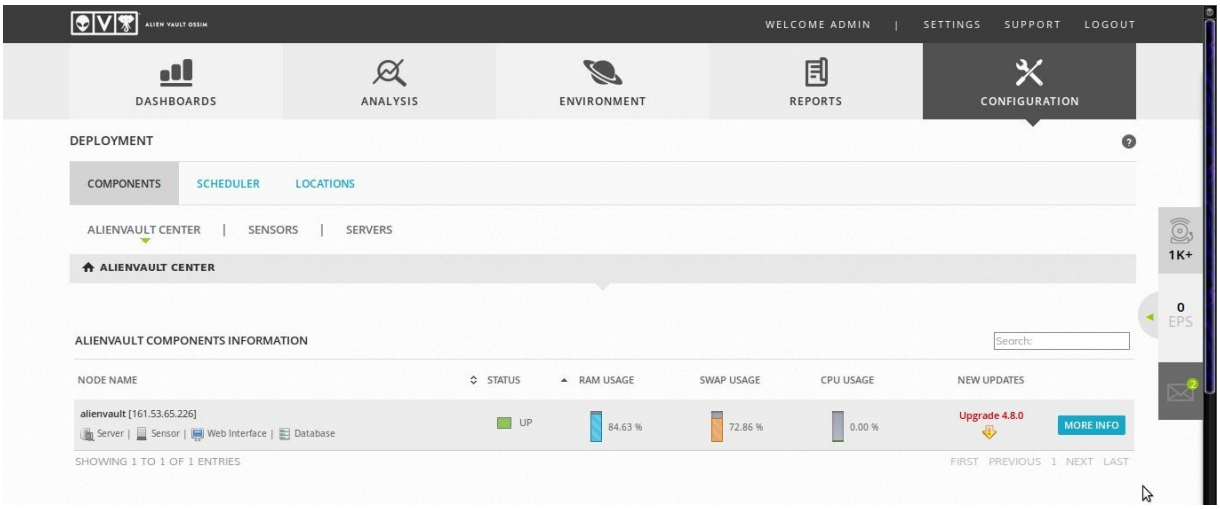
Kako bi se osiguralo da se poslužitelju ne može pristupiti sa drugih IP adresa, potrebno je na kraj datoteke dodati pravilo kojim se odbija promet sa svih drugih IP adresa. Naredba koja to radi je:

```
-A INPUT -p tcp –dport (pristupna točka) -j  DROP
```

Tom se naredbom odbacuje TCP promet sa bilo koje IP adrese na određenoj pristupnoj točci. Ukoliko se pristupna točka ne navede, to pravilo je važeće za sve pristupne točke. Jako je bitno da se takva pravila nalaze na kraju datoteke kako ne bi blokirale i IP adrese kojima želimo dopustiti pristup.

## *2.5. Nadogradnja poslužitelja*

Nadogradnja OSSIM poslužitelja je vrlo jednostavan proces. Jednom kada je dostupna novija verzija OSSIM poslužitelja, u web sučelju će se pojaviti poruka s tom informacijom. Drugi način kako se može pogledati ako je dostupna nova verzija je da se kroz izbornik u web sučelju odabere opcija `Configuration`, i `Deployment` u podizborniku. Tu vidimo informacije o svim komponentama OSSIM sustava. Ukoliko je dostupna nova verzija, biti će ponuđena opcija za nadogradnju s podatcima o tome što se sve promijenilo u novoj verziji kao na Slici 1. Preporučuje se da se dobro pročita što se sve promijenilo prije nego li se obavi nadogradnja.




*Slika 1. Nadogradnja OSSIM komponenti*

# 3. Korelacijski mehanizam OSSIM sustava

OSSIM poslužitelj je samo jedna od komponenti sustava. U sustavu se obično nalazi jedan poslužitelj i jedan ili više senzora ili agenata. Agenti su računala koja imaju ulogu analizirati mrežni promet te se zbog toga preporučuje postavljanje više agenata kroz mrežu. Na tim se računalima nalaze razni alati koji prate promet te generiraju događaje (*events*). Generirani događaji se dalje šalju poslužitelju koji ih pohranjuje u bazu podataka. Jednom kada poslužitelj primi događaje, njih obrađuje korelacijski mehanizam.

Korelacijski mehanizam funkcionira slično kao i logičko stablo sa "ako" i "ili" izrazima. Umjesto tih izraza korelacijski mehanizam koristi direktive. Direktive se definiraju u xml formatu te se sve direktive učitavaju pri pokretanju tog mehanizma. Svaka direktiva se sastoji od jednog ili više pravila s kojima se uspoređuju događaji. Ovo je primjer jedne direktive:

```
<directive id="1" name="Successful Dcom exploit" priority="5">
     <rule type="detector" name="Snort dcom signature"
          reliability="1" time_out="60" occurrence="1"
          from="ANY" to="ANY" port_from="ANY"
          port_to="135,445" plugin_id="1001" plugin_sid="2192">
</directive>
```

Na početku se definiraju informacije vezane za direktivu. Definira se jedinstveni brojčani identifikator, ime direktive i njen prioritet. Unutar oznake direktive zapisuju se pravila. Pravila se definiraju kroz nekoliko parametara među kojima su ime, pouzdanost, izvorišne i odredišne IP adrese, identifikator alata koji je generirao događaj.

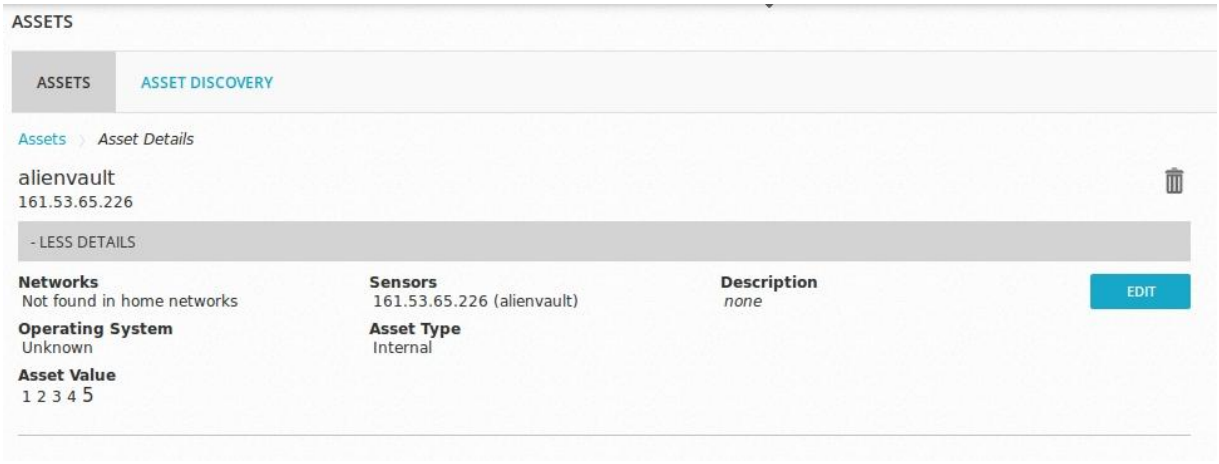
Događaji se provjeravaju na način da ih se uspoređuje sa svakom direktivom po redu. Ukoliko neki događaj zadovolji pravila te direktive, poslužitelj generira alarm i pridružuje mu rizik. Rizik se računa prema formuli:

```
Rizik = (Važnost * Prioritet * Pouzdanost) / 25
```

gdje je:

- Rizik vrijednost između 1 i 10 (1-10)
- Važnost može poprimiti vrijednost između 1 i 5 (1-5)
- Prioritet je u intervalu od 1 do 5 (1-5)
- Pouzdanost može biti između 1 i 10 (1-10)

Važnost je vrijednost koja nam govori koliko nam je neka komponenta bitna. Ta vrijednost se postavlja prilikom dodavanja novih komponenti u sustav. Kroz web sučelje se može provjeriti kolika je vrijednost za pojedinu komponentu te se tamo može i promijeniti. Popis komponenti dobiva se odabirom opcije `Environment` u izborniku i opcije `Assets` u podizborniku. Odabirom na pojedinu komponentu prikazuju se detalji kao na Slici 2.




*Slika 2. Detalji komponente u web sučelju*

Prikazana je komponenta kojoj je dodijeljena važnost 5. To se može vidjeti pod nazivom *Asset Value*. Opcijom *EDIT* mogu se podešavati svojstva komponente.

Prioritet se zadaje prilikom stvaranja direktive te nam on definira koliko je opasan događaj na koji je direktiva reagirala. To se definira specifično za tu direktivu u tom sustavu jer iste prijetnje ne moraju biti podjednako opasne na različitim sustavima. Direktive koje dolaze sa sustavom imaju neku početnu vrijednost koja se po potrebi može promijeniti.

Pouzdanost definira koliko nam je događaj koji je generirao neki alat pouzdan. Npr. ukoliko postoji neki alat koji dnevno generira jako puno događaja od kojih je većina lažno pozitivna, njemu bi se pridružila mala vrijednost pouzdanosti. [8]

Generirani alarm se može vidjeti u web sučelju zajedno sa svim detaljima vezanim za taj alarm. U detaljima možemo vidjeti koji događaji su generirali taj alarm, njegovu vrijednost rizika i druge informacije.

Trenutno jedna od direktiva koja dolazi s alatom opisana je linijom:

```
<directive id="50099" name="AV-FREE-FEED Bruteforce attack,
     SSH authentication attack against DST_IP" priority="4">
```

Ta direktiva ima pravilo:

```
<rule type="detector" name="SSH service authentication
     attempts  failed detected" reliability="6" occurrence="1"
     from="1:SRC_IP" to="1:DST_IP" port_from="ANY"
     time_out="1800" port_to="ANY" plugin_id="7012"
     plugin_sid="5712">
```

Jednom kada se pojavi događaj koji zadovoljava uvjete koji su zapisani u pravilu, računa se rizik. U direktivi je zapisana vrijednost prioriteta i ona iznosi 4, dok je u pravilu definirana pouzdanost s vrijednošću 6. U trenutku kada se obrađivao ovaj događaj vrijednost važnosti je bila 2. Na temelju tih vrijednosti može se izračunati rizik pomoću spomenute formule. Rizik koji se dobije iznosi 1, te se generira alarm.

# 4. OSSIM baza podataka

OSSIM poslužitelj sve prikupljene podatke sprema u SQL bazu podataka. Alat koji se koristi za upravljanjem tom bazom podataka je MySQL. MySQL je jedan od najpoznatijih i najkorištenijih alata za upravljanje bazom podataka. Jedan od razloga zašto je MySQL toliko popularan je taj jer dolazi u paketu sa većinom distribucija Linux operacijskog sustava. Uzimajući to u obzir jasno je zašto je taj alat odabran za upravljanje bazom podataka na poslužitelju.

Na poslužitelju je pripremljena skripta za spajanje na bazu podatak koja pojednostavljuje pristup. Sve što je potrebno za spojiti se na bazu jest upisati naredbu:

```
ossim-db
```

kojom se u komandoj liniji pokreće alat MySQL. Pokrenuti alat je spojen na bazu podataka nazvanu "alienvault" u kojoj se nalaze prikupljeni podatci sa svih agenata.

Skripta `ossim-db` funkcionira na način da iz datoteke `/etc/ossim/ossim_setup.conf` izvlači korisničko ime i lozinku za spajanje na bazu podataka. Ukoliko ne postoji ta datoteka, skripta će javiti pogrešku i završiti izvođenje. Podatci koji su potrebni za spajanje na bazu podataka u datoteci izgledaju ovako:

```
db_ip=127.0.0.1
pass=(password)
user=(username)
```

Naredba kojom se skripta pomoću alata MySQL spaja na bazu podataka je:

```
mysql -A
      -u `grep ^user= /etc/ossim/ossim_setup.conf | cut -f
      2 -d "=" | sed '/^$/d'`
      -h $HOST -p `grep ^pass= /etc/ossim/ossim_setup.conf
      | cut -f 2 -d "=" | sed '/^$/d'`
      $DB
```

Ovo je složena naredba koja se može razdvojiti na više jednostavnijih naredbi. Naredba za dohvat korisničkog imena iz datoteke, s kojim se kasnije spaja na bazu podataka je:

```
grep ^user= /etc/ossim/ossim_setup.conf | cut -f 2 -d
"=" | sed '/^$/d'
```

Ovom se naredbom u datoteci traži redak koji počinje sa `user` te ga se zatim razdvaja i formatira kako bi se izvukli željeni podatci. Slična naredba se koristi i za dobivanje lozinke za pristup bazi podataka.

Varijabla `DB` se ranije u skripti postavlja na vrijednost `alienvault` dok se varijabli `HOST` pridružuje vrijednost zapisana u spomenutoj datoteci naredbom:

```
HOST=`grep db_ip /etc/ossim/ossim_setup.conf | cut -f 2 -d "="`
```

## *4.1.Struktura OSSIM baze podataka*

OSSIM dobivene podatke sprema u MySQL bazu podataka pod nazivom `alienvault`. Baza podataka se sastoji 213 tablica. U te tablice je spremljeno skoro sve potrebno za rad web sučelja. Pod rad sučelja se podrazumijeva prikaz podataka kao i funkcionalnost samog sučelja. To uključuje od dodataka koji se trebaju prikazivati u web sučelju do podataka koje je poslužitelj prikupio i analizirao. Međutim većina baze podataka se odnosi na sakupljene podatke o mrežnom prometu.

U bazi podataka postoji nekoliko grupa tablica koje imaju svoju funkciju. Grupe tablica su namijenjene za:

1. PhpGACL - PhpGACL se koristi na poslužitelju kao skup funkcija koji omogućava kontrolu pristupa različitim komponentama sustava. U bazu podataka se spremaju dopuštenja pojedinih entiteta. Tablicama koje se koristi ovaj alat ime počinje sa "acl_". Neke od tablica su: `acl_assets`, `acl_entities`, `acl_sensors`, `acl_login_sensors`.

2. alarme - Alarmi su događaji koji su aktivirali neka pravila. Oni sadrže najbitnije informacije za korisnika OSSIM sustava. O njima će biti detaljnije napisano kasnije.

3. prikaz dodataka na sučelju - Web sučelje ima opciju da se izaberu razni dodaci koji će se prikazivati na početnoj stranici. To su obično grafovi koji prikazuju stanje sustava. Informacije o tim dodatcima se spremaju u bazu podataka.

4. računala - U bazu podataka se spremaju informacije o različitim računalima. Tablice u koje se ti podaci spremaju možemo prepoznati tako što im ime započinje sa "host_". U tim tablicama se nalaze informacije o računalima koja su dio tog sustava ali i neka druga. Računala o kojima želimo imati više informacija mogu se dodati preko web sučelja. Već postojećim računalima se mogu pridruživati imena i mijenjati određena svojstva. Računala se mogu dodatno skenirati sa alatima poput Nessusa ili Nmapa. Ta skeniranja se mogu izvoditi jednokratno ili se može namjestiti da se izvode periodički.

5. incidente - Unutar OSSIM sustava incidenti podrazumijevaju sredstvo za interni način komuniciranja. To je zapravo jedan sustav zahtjeva (*ticket system*). Zahtjev se može stvoriti putem web sučelja ili ju sustav može automatski stvoriti. Takav sustav je koristan kada preko web sučelja radi više osoba. Na taj način je lakše komunicirati i istaknuti neke bitne događaje. Neke od tablica koje sadrže informacije o tom sustavu su: `incident`, `incident_alarm`, `incident_custom`, `incident_event`, `incident_ticket`, `incident_type`.

6. karte na kojoj se prikazuju računala - OSSIM sustav ima zanimljivi dodatak nazvan Karta rizika (*Risk maps*). Unutar web sučelja se može pronaći nekoliko geografskih karti:

    - svijeta
    - Europe
    - SAD-a
    - Španjolske
    - Kalifornije

- Sjeverne Karoline
- New York-a
- infrastrukture lokalne mreže

Karte se mogu uređivati na način da se na njih postavljaju indikatori. Indikatori mogu predstavljati računala koja su bitna sustavu. Posebno je zanimljiva karta infrastrukture gdje se može namjestiti izgled mreže sa svim bitnim komponentama. Karte trenutno nemaju implementiranu nikakvu funkcionalnost ali nekad je korisno imati vizualni prikaz računala u mreži.

7. mreže - OSSIM sustavu se može definirati više različitih mreža koje treba nadzirati. Popis mreža koje sustav nadzire može se vidjeti na web sučelju i u tablici `net`. Mreže se preko web sučelja mogu pregledavati i dodavati.

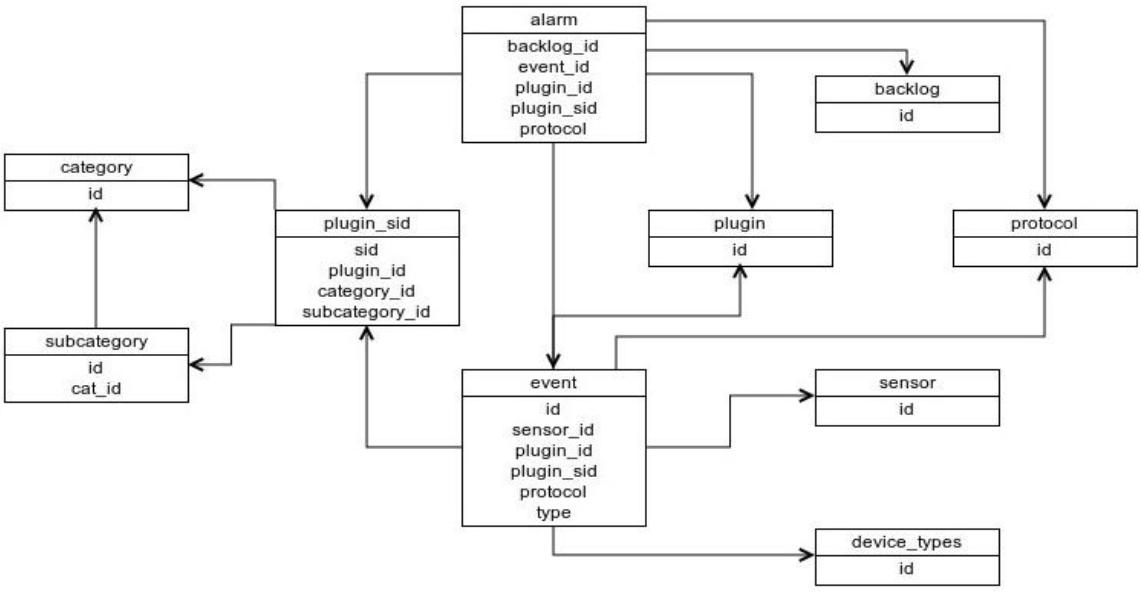
8. dodatne alate - Dodatni alati (*plugins*) su u OSSIM sustavu ključni. Oni su zaslužni za analiziranje mrežnog prometa, spremanje podataka u bazu podataka, generiranje alarma itd. OSSIM je zapravo skup takvih alata složenih da funkcioniraju u jednoj cjelini. Prvotna zadaća OSSIM-a je omogućiti jednostavno korištenje svih tih alata.

9. pravila - OSSIM pruža mogućnost stvaranja svojih pravila po kojima će se generirati alarmi. Pravila se kreiraju preko web sučelja. Pravila su nešto slično kao direktive. Razlikuju se po tome što se direktive koriste kod korelacije dok se pravila koriste za filtriranje podataka. Pomoću pravila se definiraju akcije koje će se poduzeti za određene grupe alarma i događaja. Akcije među ostalim mogu biti ignoriranje pojedinih alarma ili pak mijenjanje neki vrijednosti.

10. Nessus - Nessus je jedan od alata koji je integriran u OSSIM. On se koristi za skeniranje ranjivosti drugih računala. Skeniranja se obično obavljaju tako da se prvo s nekim alatom poput Nmapa skeniraju sve otvorene pristupne točke. Nakon što su otkrivene otvorene pristupne točke, Nessus ih testira na razne načine zlouporabe. Dobiveni podatci se spremaju u bazu podataka te im se može pristupiti preko web sučelja.

## 4.2. Događaji

Događaji su podatci koje su agenti zabilježili. Jednom kad su spremljeni u bazu podataka, korelacijski sustav ih provjerava i eventualno na temelju njih generira alarme. Događaji su spremljeni u tablicu `event`.



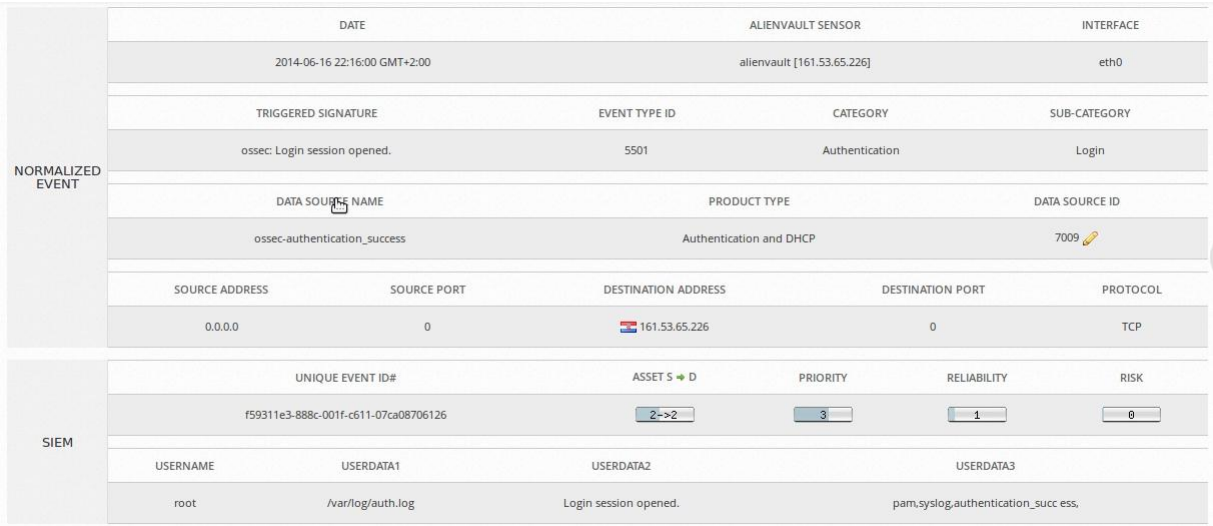

*Slika 3. Dijagram tablica*

Kao što se vidi iz Slike 3. tablica je povezana sa nekim tablicama koje su ujedno povezane i sa tablicom `alarm` pa će biti objašnjene u slijedećem poglavlju i sa tablicama `sensor` i `device_types`. U tablici `sensor` se nalaze svi agenti koji se nalaze u OSSIM sustavu. Za događaj to predstavlja koji agent je zabilježio taj događaj, dok `device_types` govori o kojoj vrsti uređaja se radi. Uređaji među ostalim mogu biti različiti poslužitelji, usmjernici, sigurnosne stijene.

U tablici `event` se nalaze sljedeće informacije:

- vrijeme i datum kad je zabilježen događaj
- na kojem mrežnom sučelju je zabilježen
- IP adrese, MAC adrese, pristupne točke
- imena od izvornog i odredišnog računala
- IP adrese izvorne i odredišne mreže
- prioritet i pouzdanost
- ime pravila koje je generiralo događaj
- dodatni podatci o događaju

Sve te informacije su vidljive kroz web sučelje kao što je prikazano na Slici 4.



| NORMALIZED EVENT | | | | | |
|---|---|---|---|---|---|
| DATE | | | ALIENVAULT SENSOR | | INTERFACE |
| 2014-06-16 22:16:00 GMT+2:00 | | | alienvault [161.53.65.226] | | eth0 |
| TRIGGERED SIGNATURE | | EVENT TYPE ID | CATEGORY | | SUB-CATEGORY |
| ossec: Login session opened. | | 5501 | Authentication | | Login |
| DATA SOURCE NAME | | | PRODUCT TYPE | | DATA SOURCE ID |
| ossec-authentication_success | | | Authentication and DHCP | | 7009 |
| SOURCE ADDRESS | SOURCE PORT | DESTINATION ADDRESS | | DESTINATION PORT | PROTOCOL |
| 0.0.0.0 | 0 | 161.53.65.226 | | 0 | TCP |

| SIEM | | | | | |
|---|---|---|---|---|---|
| UNIQUE EVENT ID# | | ASSET S → D | PRIORITY | RELIABILITY | RISK |
| f59311e3-888c-001f-c611-07ca08706126 | | 2->2 | 3 | 1 | 0 |
| USERNAME | USERDATA1 | USERDATA2 | | USERDATA3 | |
| root | /var/log/auth.log | Login session opened. | | pam,syslog,authentication_succ ess, | |

*Slika 4. Detalji događaja u web sučelju*

## 4.3. Alarmi

Alarm je u OSSIM sustavu jedan od osnovnih podataka. Pomoću njih se daje do znanja administratoru i drugim korisnicima da se dogodilo nešto vrijedno pažnje. Alarm je događaj ili skup događaja kojemu je rizik veći ili jednak 1. Rizik računa korelacijski mehanizam kako je opisano ranije. Korelacijski mehanizam ujedno i brine o tome da grupira slične događaje u jedan alarm kako se ne bi generiralo previše alarma.

Alarmi se u bazi podataka spremaju u tablicu *alarm*. Ta se tablica sastoji od 18 atributa i njena je struktura prikazana u Tablici 1.

| Field | Type | Null | Key | Default |
|---|---|---|---|---|
| backlog_id | binary(16) | NO | PRI | NULL |
| event_id | binary(16) | NO | MUL | NULL |
| corr_engine_ctx | binary(16) | NO | | NULL |
| timestamp | timestamp | YES | | NULL |
| status | enum('open','closed') | YES | MUL | open |
| plugin_id | int(11) | NO | MUL | NULL |
| plugin_sid | int(11) | NO | | NULL |
| protocol | int(11) | YES | | NULL |
| src_ip | varbinary(16) | YES | MUL | NULL |
| dst_ip | varbinary(16) | YES | MUL | NULL |
| src_port | int(11) | YES | | NULL |
| dst_port | int(11) | YES | | NULL |
| risk | int(11) | YES | | NULL |
| efr | int(11) | NO | | 0 |
| similar | varchar(40) | NO | | 0 |
| stats | text | NO | | NULL |
| removable | tinyint(1) | NO | | 0 |
| in_file | tinyint(1) | NO | | 0 |

*Tablica 1. Struktura tablice alarm*

Neki od tih atributa su strani ključevi kojima se ova tablica spaja sa drugim tablicama. Dijagram povezanosti tablica možemo vidjeti na Slici 3. Zbog preglednosti su pod tablicama ispisani samo relevantni ključevi.

Atribut `backlog_id` se spaja sa tablicom `backlog`. U `backlog` tablici se nalaze direktive koje su aktivirane ali nisu još završile korelaciju do kraja ili im nije vrijeme isteklo.

Tablica `protocol` je povezana sa tablicom `alarm` preko svog primarnog ključa atributa `id` i preko atributa `protocol` u tablici `alarm`. U tablici `protocol` se nalazi popis internet protokola. Trenutno u bazi postoji 138 protokola a neki od njih su: TCP, UDP i ICMP. Veza tih dviju tablica označava koji Internet protokol je korišten za mrežni promet za koji je generiran alarm.

Preko atributa `plugin_id` tablica `alarm` je povezana sa tablicom `plugin`, u kojoj su zapisani svi alati koje OSSIM koristi. Tako se može saznati koji je alat generirao određeni alarm.

Tablica `plugin_sid` sadrži detaljnije informacije o razlogu zbog čega je neki alat generirao alarm. Tablica sadrži atribute kao što su naziv, strani ključ koji povezuje sa alatom, prioritet i pouzdanost koji služe za računanje rizika, kategoriju i potkategoriju. `Category` je tablica u kojoj se nalaze kategorije po kojima su raspoređeni uzroci generiranja alarma dok je potkategorija daje malo detaljniji razlog. Neke od kategorija su: *Exploit, Malware, Denial_Of_Service, Suspicious, Antivirus, Application, Voip, Honeypot*.

Tablica `event` u kontekstu alarma označava događaj na temelju kojeg je generiran alarm.

## 4.4. Spremanje podataka iz alata u bazu podataka

Podatci koje generiraju alati kao što je npr. Snort spremaju se u bazu podataka u neke od tablica koje su navedene ranije. U poglavlju 2.3. opisano je kako spojiti na poslužitelja agenta koji koristi Snort, a sad će se prikazati kako ti podatci dolaze do baze podataka.

Za ovaj primjer agent će koristiti IP adresu 192.168.1.5 dok će OSSIM poslužitelj koristiti IP adresu 192.168.1.1. Primjer se može napraviti na jednostavnom protokolu ICMP točnije koristit će se naredba ping:

```
ping 192.168.1.5
```

Ovom se naredbom provjerava dostupnost računala tako da se šalje jednostavan paket te se zatim čeka odgovor.

Ako na agentu postoji snort pravilo za protokol ICMP kao npr. :

```
alert icmp $EXTERNAL_NET any -> $HOME_NET any
(msg:"PROTOCOL-ICMP PING"; icode:0; itype:8;
metadata:ruleset community; classtype:misc-activity;
sid:384; rev:8;)
```

Snort će generirati alarm koji će se pojaviti u datoteci `/var/log/snort/snort.fast.` Ovo pravilo generira alarm za bilo koji ICMP promet u mreži.

Naredbom `tail` možemo provjeriti koji su zadnji unosi u datoteci. Među zadnjim unosima trebao bi biti ispis:

```
06/20-01:32:25.025950  [**] [1:384:8] PROTOCOL-ICMP PING
[**] [Classification: Misc activity] [Priority: 3] {ICMP}
129.82.138.44 -> 192.168.1.5
```

Taj se podatak pomoću alata rsyslog šalje na poslužitelj i sprema u datoteku `/var/log/snort/alert.` Jednom kad je podatak prenesen, alat snort_syslog ga parsira i sprema u bazu podataka. Način na koji to radi se može vidjeti u datoteci `/etc/ossim/agent/plugins/snort_syslog.cfg.` Točne naredbe kojima se parsira podatak su :

```
[01_snort-alert-format]

event_type=event


regexp=^(?P<date>\d\d/\d\d-\d\d:\d\d:\d\d.\d+)  \[\*\*\] \
[(?P<pid>\d+):(?P<sid>\d+):\d\] (?P<msg>.+) \[\*\*\] \
[Classification: .+\] \[Priority: .+\] \{(?P<proto>.+)\}
(?P<src_ip>[\d.]{7,15}):(?P<src_port>\d+) -> (?
P<dst_ip>[\d.]{7,15}):(?P<dst_port>\d+)$


date={normalize_date($date)}
```

```
plugin_id={snort_id($pid)}
plugin_sid={$sid}
protocol={$proto}
src_ip={$src_ip}
src_port={$src_port}
dst_ip={$dst_ip}
dst_port={$dst_port}
```

Na početku je definirano o kakvoj vrsti podataka se radi. To definira kako će se spremiti u bazu tj. koja tablica se koristi. Zatim se pomoću regularnog izraza izvlače svi relevantni podatci i spremaju u varijable. Varijable se moraju zvati isto kao i atributi u tablici u bazi podataka. Na taj način se varijable koriste za upis podataka u bazu.

# 5. Strojno učenje nad podatcima

Strojno učenje je disciplina računarske znanosti kojoj je cilj naći odgovor na pitanje:"Kako možemo napraviti računalne sustave da se poboljšavaju s vremenom i koja su osnovna pravila koja treba poštovati prilikom procesa učenja." [9]. U praksi pod strojnim učenjem se podrazumijevaju algoritmi koji na temelju ulaznih podataka mogu zaključiti nešto o novim podatcima. Algoritmi strojnog učenja se mogu podijeliti u nekoliko kategorija, osnovne kategorije su:

- učenje s nadzorom
- učenje bez nadzora
- učenje s podrškom

Učenje s nadzorom se obično provodi tako da se na ulaz dovedu podatci za koje znamo kako treba izgledati izlaz. Jednom kad algoritam obradi ulazne podatke, uspoređujemo dobiveni rezultat i očekivani rezultat. Na temelju razlike, algoritam uči te poboljšava svoje buduće rezultate. Postoji puno algoritama koji prakticiraju ovakav način učenja, a neki od najpoznatijih su: Naivni Bayesov klasifikator, metoda najbližih susjeda, stablo odluke, regresija.

Učenje bez nadzora se koristi kada za ulazne podatke ne znamo kako treba izgledati izlaz. Cilj ovakvog učenja je pronaći pravilnosti u podatcima koje bi nam mogle koristiti kasnije. Pravilnosti se traže grupiranjem podataka u proizvoljni broj grupa. Za nove podatke će se podrazumijevati da pripadaju onoj grupi koja im najviše odgovara tj. onoj grupi u kojoj se nalaze najsličniji primjeri. Algoritam koji će se u ovom radu koristiti je k-means.

Učenje s podrškom je nešto između prethodna dva učenja. To je učenje bez nadzora ali postoji odgođena nagrada. Kod ovakvog načina učenja se ne zna kakav treba biti izlaz za ulazne podatke, ali se može izračunati koliko je dobiveni izlaz dobar ili loš. Ta povratna informacija djeluje na algoritam kao nagrada ili kazna, te se na temelju toga algoritam uči. Jedna od primjena takvog učenja je igranje raznih igara na ploči. Kod takvih igara ne postoji točan potez, ali može se pretpostaviti ako je potez bio dobar ili loš.

## *5.1. K-means algoritam*

K-means algoritam spada pod algoritme učenja bez nadzora. To je jedan od algoritma grupiranja. Algoritam ulazne podatke grupira u k grupa. Jedno od ograničenja ovog algoritma je što ulazni podatci moraju biti brojevi. Na taj način ulazni podatci se mogu prikazati kao točke u ravnini te ih se može geometrijski grupirati. Grupiranje se odvija tako da se za svaku grupu izračuna centroid koji će predstavljati tu grupu. Na početku algoritma se odabire k centroida. Što je bolji izbor centroida na početku, algoritam će bolje grupirati podatke na kraju. Nakon što su se odabrali centroidi, ulazni podatci se uzimaju jedan po jedan te se pridružuju najbližem centroidu. Jednom kada su se svi podatci grupirali, pozicija centroida se ponovno računa. Nova pozicija centroida se računa tako da se uzme srednja udaljenost od svih podataka koji su pripadali tom centroidu. Za udaljenost se tipično uzima Euklidska udaljenost no može se koristiti bilo koja metoda računanja udaljenosti. Euklidska udaljenost se računa formulom:$d(\mathrm{a},\mathrm{b}) = \sqrt{\sum_{i=1}^{n}(a_i - b_i)^2}$ gdje su a i b koordinate točke koja predstavlja određeni primjer.

Taj proces se ponavlja dok se u jednoj iteraciji ni jedan centroid ne pomakne.

Algoritam se može zapisati u par jednostavnih koraka [10]:

1. Postavi k centroida na neku početnu poziciju.
2. Pridruži svaki podatak najbližem centroidu
3. Kada su pridruženi svi podatci, ponovno računaj poziciju svih centroida.
4. Ponavljaj korake 2 i 3 dok se god neki centroid pomiče

Jednom kada algoritam završi te su određene završne pozicije centroida, nove podatke grupiramo vrlo jednostavno. Pronađe se centroid koji je najbliži novom podatku i podatak se pridružuje grupi koju predstavlja taj centroid.

## 5.2 Primjena strojnog učenja na OSSIM sustav

U OSSIM sustavu postoji puno načina kako se može iskoristiti strojno učenje. Da li će se iskoristiti za neko pametnije filtriranje bitnih podataka ili raniju detekciju napada ovisi samo o implementaciji. Za jednostavniji primjer koristit će se programski jezik Python i Python dodatak Scikit [11]. Scikit je dodatak koji sadrži razne algoritme strojnog učenja. U ovom primjeru koristit će se algoritam K-means te će se pomoću njega pokušati na pametniji način izračunati rizik nekog događaja. Trenutni način računanja rizika je opisan u poglavlju 3. Za parametre algoritma ćemo uzimati iste one koji se i inače koriste za računanje rizika a to su: pouzdanost, prioritet i važnost.

Na početku je potrebno spojiti se na bazu podataka kako bi se dobili podatci koji će se koristiti kao primjeri. Postupak spajanja opisan je u 4. poglavlju. Postupak je potrebno samo prilagoditi programskom jeziku Python. Za spajanje na bazu podataka koristi se biblioteka MySQLdb [12]. Ta biblioteka nudi statičku funkciju pomoću koje se to ostvaruje. Funkciji se predaju potrebni parametri. Primjer te funkcije koji se koristi u ovom primjeru je:

```
db = MySQLdb.connect(host=HOST, user=USER, passwd=PASS, db=DB)
```

Parametri za spajanje na bazu podataka su dobiveni parsiranjem datoteke `/etc/ossim/ossim_setup.conf`.

Iz baze podataka koriste se tablice `event` i `sensor`. Prioritet i pouzdanost se nalaze u tablici `event` dok nam tablica `sensor` služi kako bi dobili važnost. SQL naredba kojom to dohvaćamo je:

```
select event.priority, reliability, sensor.priority, risk_a
     from event join sensor where sensor_id = sensor.id;
```

Dodatno se dohvaća već izračunati rizik kako bi se moglo usporediti dobivene rezultate. U izvornom kodu podatci se dohvaćaju naredbom:

```
cur.execute("select event.priority, reliability,
     sensor.priority, risk_a from event join sensor where
     sensor_id = sensor.id")
```

gdje je `cur` objekt koji služi kao pokazivač na bazu podataka. Pomoću njega se izvode sve SQL operacije nad bazom.

Prije nego se pokrene algoritam, potrebno je definirati nekoliko parametara. Najbitniji parametar je broj grupa u koji se želi grupirati podatke. Kako parametar rizik može biti od 1–10, broj grupa se može postaviti na 10. Svaka grupa će predstavljati jedan stupanj rizika. Taj broj može biti i drukčiji, ali tu se koristi 10 kako bi se moglo usporediti dobivene rezultate sa izračunatim. Ostali parametri koji se mogu postaviti su:

- maksimalni broj iteracija jednog prolaska algoritmom
- broj pokušaja algoritma tj. koliko će se puta algoritam ponoviti sa različitim početnim centroidima
- metoda kojom će se inicijalizirati centroidi grupa

Nakon što su se odabrali parametri algoritam se pokreće naredbom

```
kmeans.fit(data)
```

gdje je `kmeans` objekt koji je stvoren sa prethodno navedenim parametrima naredbom:

```
kmeans = KMeans(init="k-means++", n_clusters=n_digits,
    n_init=10)
```

Objekt `data` je lista sa svim primjerima dohvaćenim iz baze podataka. Za prikaz podataka se koristi polje iz bibiloteke NumPy [13]. Stvaranje tog polja se radi naredbom:

```
data = np.array(raw_data, np.int32)
```

Jednom kada je algoritam završio, pripadnost novog primjera nekoj grupi se može dobiti naredbom:

```
solution = kmeans.predict(new_example)
```

Mogući problem ovakvog pristupa je taj što je potrebno u bazi podataka imati skupinu podataka za svaki stupanj rizika kako bi algoritam dobro pozicionirao centroide. Ukoliko ne postoji raznovrsnost podataka, dolazi do toga da se jedan stupanj rizika razlomi na više grupa.

Izvorni kod ovog primjera se nalazi u dodatku A.

# 6. Zaključak

OSSIM je jedan jako kompleksan sustav koji administratori mreže koriste kao pomoć pri zaštiti računalne mreže. Napravljen je na način da ga je lagano ugraditi u mrežu dok web sučelje omogućuje relativno jednostavan i interaktivan način upravljanja i korištenja sustava.

Problem se pojavljuje kada je potrebno promijeniti nešto što nije dostupno u web sučelju. Kako taj sustav podržava veliki broj alata te ima preko 350 dodataka, mijenjanje nekih stvari se može jako zakomplicirati. Najveći problem dok kojeg dolazi je manjak dokumentacije. Dokumentacija je ili jako zastarjela ili je nema. Kako se o razvoju tog sustava brine firma koja prodaje unaprijeđenu inačicu, jedina dostupna dokumentacija je vezana uz unaprijeđeni sustav.

OSSIM dopušta instalaciju novih alata pa je moguće i koristiti alate za strojno učenje. Podatci koji bi se koristili u tim alatima dostupni su u bazi podataka. Baza podataka je dosta velika te zahtjeva puno truda kako bi se pronašli svi potrebni podatci.

Upotreba alata za strojno učenje nad podatcima može ispasti kao jako korisna stvar. Međutim zbog navedenih nedostatka implementacija nekih kompliciranijih algoritama je jako zahtjevan zadatak.

# Literatura


1. AlienVault Projects, http://www.alienvault.com/open-threat-exchange/projects (25/10/2013)
2. AlienVault os-sim, https://www.assembla.com/code/os-sim/git/nodes (26/10/2013)
3. phpGACL - Generic Access Control Lists, http://phpgacl.sourceforge.net/ (12/5/2014)
4. Analysis Console for Intrusion Databases, http://acidlab.sourceforge.net/ (12/5/2014)
5. FPDF Library, http://www.fpdf.org/ (12/5/2014)
6. D. Gill, S. Fournier, Installing OSSIM on a Debian GNU/Linux, https://www.assembla.com/code/os-sim/git-2/nodes/master/os-sim/doc/INSTALL.Debian.pdf (26/10/2013)
7. Bill Parker, Integrating Snort-2.9.5.x with the AlienVault OSSIM 4.1 SIEM on Linux based systems, URL: http://s3.amazonaws.com/snort-rg/www/assets/224/Integrating_Snort_with_OSSIM.pdf (10/5/2014).
8. Dominique Karg, Correlation engine explained, https://www.alienvault.com/docs/correlation_engine_explained_worm_example.pdf, (12/6/2014).
9. Tom M. Mitchell, The Discipline of Machine Learning, http://www.cs.cmu.edu/~tom/pubs/MachineLearning.pdf (14/6/2014).
10. K-Means Clustering, http://home.deib.polimi.it/matteucc/Clustering/tutorial_html/kmeans.html (14/6/2014)
11. Scikit-learn developers, Documentation of scikit-learn 0.14, http://scikit-learn.org/stable/documentation.html (14/6/2014).
12. MySQL for Python, http://sourceforge.net/projects/mysql-python/ (14/6/2014).
13. NumPy, http://www.numpy.org/ (14/6/2014)

## Dodatak A.

```
#!/usr/bin/python
from time import time
import MySQLdb
import numpy as np
import pylab as pl
from sklearn import metrics
from sklearn.cluster import KMeans
from sklearn.datasets import load_digits
from sklearn.decomposition import PCA
from sklearn.preprocessing import scale

#otvaranje datoteke sa podatcima za pristup bazi podataka
conf = open("/etc/ossim/ossim_setup.conf", "r")
#traže se podatci za pristup bazi podataka
for line in conf:
     if line.startswith("db_ip"):
          HOST = line.split("=")[1].strip()
     if line.startswith("user"):
          USER = line.split("=")[1].strip()
     if line.startswith("pass"):
          PASS = line.split("=")[1].strip()
DB = "alienvault"
#spajanje na bazu podataka
db = MySQLdb.connect(host=HOST, user=USER, passwd=PASS, db=DB)
cur = db.cursor()
#dohvat željenih podataka iz baze podataka
cur.execute("select event.priority, reliability,
sensor.priority, risk_a from event join sensor where sensor_id
= sensor.id")
#spremanje podataka u objekte prikladne za algoritam
raw_data = []
```

```
raw_labels = []
for row in cur.fetchall():
     raw_row = [row[0], row[1], row[2]]
     raw_data.append(raw_row)
     raw_labels.append(int(row[3]))
data = np.array(raw_data, np.int32)

#izračunati rezultat koji se koristi za usporedbu
labels = np.array(raw_labels, np.int32)

#broj grupa
n_groups = 10
#inicijaliziranje objekta koji sadrži algoritme za strojno
#učenje
kmeans = KMeans(init="k-means++", n_clusters=n_groups,
n_init=10)
#zabilježuje se početno vrijeme
t0 = time()
#treniranje algoritma
kmeans.fit(data)

#ispis statistika dobivenog rezultata
print('% 9s   %.2fs    %i   %.3f   %.3f   %.3f   %.3f   %.3f' %
("k_means++", (time() - t0), kmeans.inertia_,
             metrics.homogeneity_score(labels, kmeans.labels_),
             metrics.completeness_score(labels,
kmeans.labels_),
             metrics.v_measure_score(labels, kmeans.labels_),
             metrics.adjusted_rand_score(labels,
kmeans.labels_),
             metrics.adjusted_mutual_info_score(labels,
kmeans.labels_)))
```