Agregacija heterogenih alarma različitih IDS sistema upotrebom IDMEF formata

Lazar Ignjatović, 17124
Elektronski fakultet, Niš
Email: lazar.ignjatovic@elfak.rs

Sadržaj—Cilj ovog rada je projektovanje i implementacija rešenja koje vrši agregaciju heterogenih alarma različitih IDS sistema u jedinstven skup alarma formatiranih po IDMEF standardu. Konkretni sistemi čiji se alarmi prikupljaju su Snort i Suricata IDS paketi podignuti i odgovarajuće konfigurisani na Linux mašini. Za agregaciju i konverziju alarma iskorišćen je Logstash serverski alat za obradu podataka, sa proširenjem u vidu instaliranog *plugin*-a za automatsku konverziju podataka u IMDEF format. U radu je detaljno opisana implementacija ovog sistema, kao i proces prikupljanja, filtriranja, agregacije i konverzije u IDMEF standad heterogenih alarma koje generišu Snort i Suricata IDS sistemi. Takođe će biti dat vizuelni prikaz mogućnosti samog sistema i primer rada.

I. Uvod

U našem svetu sve veće Internet konektivnosti, postoji stalna pretnja od upada, napada uskraćivanja usluga i bezbrojnih drugih zloupotreba računarskih i mrežnih resursa. Kao efektivna linija odbrane protiv mrežnih napada usmerenih na računarske sisteme smatraju se sistemi za detekciju napada (*Intrusion Detection Systems*). Oni su, s obzirom na sve veću ozbiljnost i verovatnoću napada, primenjeni u skoro svim većim IT infrastrukturama [1].

Cilj sistema za detekciju napada (IDS) je da posmatra mrežne resurse kako bi detektovao njihovu zloupotrebu ili neadekvatno ponašanje istih. Kao proizvod detekcije zloupotrebe ili neuobičajenog ponašanja, IDS generiše alarm koji sadrži informacije o napadu i njime informiše administratora, odnosno ostatak sistema [1].

Za preciznu i efikasnu detekciju napada koriste se kolaborativni sistemi za detekciju napada (*Collaborative Intrusion Detection Systems*). CIDS koristi više specijalizovanih detektora na različitim nivoima – mrežnom, kernel i aplikacionom – kao i okvir za agregaciju alarma različitih detektora koji bi obezbedio jedinstveni alarm prilikom napada. Premisa je da pažljivo dizajniran i konfigurisan CIDS može da poveća preciznost detekcije napada u poređenju sa samostalnim IDS-ovima, bez značajne degradacije u performansama [2].

Cilj ovog rada je projektovanje i implementacija sistema za agregaciju heterogenih alarma različitih IDS sistema, konkretno Snort i Suricata IDS sistema, u uniformne alarme formatirane po IDMEF standardu Na mašini čija je namena detekcija napada, podignut je Linux operativni sistem i na njemu su podignuti i odgovarajuće konfigurisani Snort i Suricata IDS sistemi, kao sistemi nad kojima će se vršiti agregacija alarma. Alarmi koje generišu Snort i Suricata šalju se do Logstash alata koji ih dalje parsira i filtrira na osnovu napisanog konfigracionog fajla i korišćenjem instaliranog *codec plugin*-a konvertuje u alarm formatiran po IDMEF standardu. Svi prikupljeni alarmi se skladište u log fajlu na Linux mašini i mogu se koristiti za dalju potencijalnu analizu i donošenje odluka o daljim akcijama radi poboljšanja bezbednosti celokupne IT infrastrukture u kojoj se detekcija napada vrši. U radu će biti opisana implementacija predloženog sistema za agregaciju alarma, kao i proces prikupljanja heterogenih alarma, njihovog parsiranja i filtriranja i krajnje konverzije u IDMEF format. Takođe će biti dat i vizualni prikaz mogućnosti *codec*-a za konverziju i prikaz implementiranih IDMEF polja u konkretnom sistemu, kao i primer Snort i Suricata alarma i njihove odgovarajuće reprezentacije po IDMEF standardu kao rezultat rada sistema.

Nakon uvoda, u drugom poglavlju biće data teoretska osnova neophodna za implementaciju predloženog sistema. Biće diskutovan princip rada i format alarma Snort i Suricata IDS Sistema, IDMEF formata i njegovog značaja, kao i Logstash alata koji se koristi za prikupljanje i filtriranje alarma i instaliranog *codec plugin*-a za konverziju u IDMEF format. U trećem poglavlju biće detaljno opisan implementirani sistem. U četvrtom poglavlju biće dat prikaz mogućnosti i rada predloženog sistema. Zaključak će biti dat u petom poglavlju.

II. Teoretska osnova

A. Snort

Snort je *cross-platform* alat koji služi za detekciju mrežnih napada i koji se najčešće koristi tako da vrši monitoring malih TCP/IP mreža sa mogućnošću detektovanja različitih varijanti sumnjivog mrežnog saobraćaja, kao i direktnih napada na mrežu. Može da administratoru dostavi više nego dovoljno podataka kako bi doneo informisanu odluku o toku daljih akcija u vezi sa sumnjivim aktovnostima na mreži. Snort takođe može brzo da se podigne kako bi popunio potencijalne sigurnosne rupe u mreži, kao što su nove vrste napada koji se pojavljuju dok komercijalni dobavljači sigurnosnih softvera ne izbace potpise za prepoznavanje istih [3].

Snort se zasniva na *libpcap* biblioteci za osluškivanje saobraćaja i odlikuje se logovanjem zasnovanim na pravilima da bi izvršio pronalaženje definisanih šablona u sadržaju i detekciju širokog opsega malicioznih aktivnosti. Takođe, Snort poseduje mogućnost obaveštavanja u realnom vremenu, gde alarmi mogu biti prosleđeni *syslog*-u, *Server Message Block (SMB) ''WinPopup''* porukama, ili poslate u zaseban "alert" fajl. Snort se konfiguriše korišćenjem "prekidača" komandne linije i opcionim *Berkeley Packet Filter* komandama. Detekcioni sistem Snort-a se programira jednostavnim programskim jezikom koji opisuje testove paketa i odgovarajuće akcije. Lakoća korišćenja pojednostavljuje i ubrzava proces razvijanja pravila za detekciju novih napada koji se pojavljuju [3].

Pravila u Snort-u su prosta za pisanje, a dovoljno moćna da izvrše detekciju širokog opsega sumnjivih i malicioznih aktivnosti na mreži. Postoje tri osnovne direktive akcija koje Snort može da primeni kada paket zadovolji definisane uslove: *pass*, *log* ili *alert*. *Pass* pravila jednostavno odbacuju paket. *Log* pravila beleže ceo paket u log rutinu izabranu od strane korisnika u trenutku izvršenja. *Alert* pravila generišu notifikaciju na način specifikovan od strane korisnika kroz *command line,* a zatim loguju ceo paket korišćenjem

selektovanog mehanizma logovanja radi omogućavanja dalje analize. Snort pravila formalno se definišu kao [3]:

<akcija><protokol><izvorna_IP_adresa><izvorni_port>
<smer><odredišna_adresa><odredišni_port><opcije>

U polju za opcije mogu se kombinovati više opcija u bilo kom maniru radi detekcije i klasifikacije paketa od interesa [3].

B. Suricata

Suricata je *open-source Intrusion Detection System, Intrusion Prevention System* i *Network Security Monitoring* alat visokih performansi razvijen i u vlansištvu OSIF (*Open Information Security Foundation*) non-profit organizacije. Koristi se za analizu mrežnog saobraćaja u realnom vremenu, kao i za analizu *pcap* fajlova. Suricata generiše log mrežnog saobraćaja i alarme zasnovane na prethodno napisanim pravilima. Potpisi, odnosno pravila, se sadrže od 3 glavna dela: *Action* deo koji definiše šta se dešava ako paket odgovara potpisu, *Header* deo koji definiše protokol, IP adrese, portove i smer pravila i *Options* deo koji definiše specifičnosti pravila [4].

```
drop tcp $HOME_NET any -> $EXTERNAL_NET
any (msg:"ET TROJAN Likely Bot Nick in IRC (USA
+..)"; flow:established,to_server;
flowbits:isset,is_proto_irc; content:"NICK ";
pcre:"/NICK .*USA.*[0-9]{3,}/i";
reference:url,doc.emergingthreats.net/2008124;
classtype:trojan-activity; sid:2008124; rev:2;)
```

Slika 1. Primer Suricata pravila

Na slici 1. prikazan je primer Suricata pravila. U ovom primeru, crvenom bojom označen je deo akcije, zelenom header deo, a plavom bojom označene su opcije [4].

Iako je moguće samostalno pisanje pravila, ili preuzimanje gotovih pravila i njihova ručna instalacija, to se ne preporučuje. Oficijalni način za unapređivanje i menadžment pravila je korišćenjem suricata-update alata koji dolazi u paketu zajedno sa Suricata IDS-om [4].

C. IDMEF

Intrusion Detection Message Exchange Format (IDMEF) je model reprezentacije podataka koji ima za cilj definiciju formata podataka i procedure za razmenu i deljenje informacija od interesa između *Intrusion Detection and Response* sistema, kao i sa menadžment sistemima sa kojima komuniciraju. IDMEF je projektovan sa namerom da bude standardni format podataka koji automatizovani IDS sistemi koriste u svojim alarmima. Standardizacija bi omogućila interoperabilnost različitih sistema, dozvoljavajući korisnicima da iste uklapaju i kombinuju radi optimalne implementacije sigurnosnog sistema [5].

IDMEF predstavlja objektno-orijentisanu reprezentaciju podataka koje, putem alarma, generišu IDS sistemi i koje šalju *Intrusion Detection* analizatorima. Implementiran je korišćenjem XML (*Extensible Markup Language*) jezika s obzirom da fleksibilnost XML-a najpogodnija za implementaciju IDMEF-a [5].

Primer alarma formatiranog po IDMEF standardu, kao i glavna struktura IDMEF formata biće data u četvrtom poglavlju.

D. Logstash

Logstash je prosleđivač događaja zasnovan na *plugin*-ovima koji pruža dosta mogućnosti. Podaci se simultano mogu uzimati iz više izvora, transformisati i prosleđivati dalje. Svaki događaj se obrađuje kroz tri faze: *inputs →filters→outputs* [6].

Najpre, input plugin omogućava rukovanje specifičnim ulaznim tokovima podataka. Logstash podržava različite tipove ulaza i dozvoljava hvatanje događaja u svim uobičajenim tipovima podataka kao što su tekstualni fajlovi, CSV fajlovi, TCP/UDP soket, HTTP API endpoint, *Elasticsearch* i mnogi drugi [6].

Dalje, obrađeni događaj može se mutirati radi izmene, uklanjanja suvišnih podataka ili dodavanja dodatnih informacija o događaju. Logstash obezbeđuje *plugin*-ove koji omogućavaju različite operacije nad podacima. Filteri se uglavnom primenjuju kondicionalno i vrše kompleksne operacije. Na primer, korišćenjem *grok* filtera moguće je izvršiti ekstrakciju podataka iz string polja koje sadrži tekst formatiran po poznatom šablonu, *ruby* filter omogućava izvršavanje proizvoljnog Ruby koda radi obrade događaja [6].

Na kraju, sledi output plugin koji podržava širok spektar tipova destinacija. Uglavnom se svi događaji šalju *Elasticsearch* sistemu, ali je takođe moguće Logstash koristiti nezavisno i podatke slati u tekstualni fajl, CSV fajl, SQL bazu podataka, algoritmu za analizu podataka (npr. Azure Machine Learning) ili ih jednostavno prikazati u konzoli. Za svaki nezavisan izvor podataka (jedinstven tip senzora podataka) potrebno je kreirati poseban Logstash konfiguracioni fajl. Delovi ovog konfiguracionog fajla odnose se na pomenute faze u obradi podataka. Operacije unutar ovih sekcija se primenjuju u redosledu u kom su deklarisane, ali obrada celokupnih sekcija se vrši u striktno definisanom redosledu: *input→filter→output*. Nije moguće specifikovati dodatan filter nakon output sekcije, u slučaju da je potrebno kreirati dve različite output akcije od jedinstvenog ulaza, potrebno je duplirati događaj, markirati ga i u skladu sa tim ga adekvatno dalje obraditi. Markiranje se takođe vrši i prilikom unosa iz više različitih izvora podataka i dalja obrada se vrši na osnovu tog markiranja [6].

Osim specifikacje izvora i destinacije podataka, potrebno je definisati i format ovih podataka. U najvećem broju slučajeva koristi se plain text ili JSON kao *codec*, ali u određenim slučajevima može se javiti potreba za kompleksnijim parsiranjem (npr. *gzip* enkodiran sadržaj, *base64*, itd) [6].

U ovom radu, korišćen tip input-a i output-a je *file*. Obzirom da se koristi jedan konfiguracioni fajl za prikupljanje podataka iz više izvora, izvršeno je markiranje podataka i kondicionalna obrada istih. Od filtera korišćen je *grok* za ekstrakciju podata iz tekstualnih fajlova u kojima se beleže alarmi, *mutate* za promenu imena pojedinih polja i izbacivanje podataka koji nisu od značaja i date za obradu *timestamp*-ova. Korišćen je IDMEF *codec* za parsiranje podataka po IDMEF standardu u output sekciji. O ovom *codec*-u biće više reči u četvrtom poglavlju

III. PREGLED IMPLEMENTIRANOG SISTEMA

Sistem predložen u ovom radu nalazi se na Linux mašini koja ima mogućnost osluškivanja mrežnog saobraćaja IT sistema za koji se ovo rešenje primenjuje. Tokom izrade korišćena je virtuelna mašina na kojoj je kao operativni sistem

podignut Ubuntu 20.4. Mrežni adapter virtuelne mašine postavljen je u promiskuitetni mod, što dalje omogućava potencijalno osluškivanje saobraćaja na mreži radi detekcije napada. Omogućen je pristup mašini putem SSH radi vernije simulacije rada na udaljenom realnom serveru u potencijalnom IT sistemu.

Na Linux mašini su dalje instalirani Snort i Suricata IDS sistemi, kao IDS sistemi čiji će se alarmi prikupljati. Nakon instalacije Snort-a, preuzeta su i instalirana Snort community pravila, na osnovu kojih će se vršiti prepoznavanje napada. Moguće je koristi i druge skupove pravila, ili čak pisati pravila od nule, ali obzirom da to nije fokus ovog rada iskorišćena su već gotova pravila. Snort je konfigurisan da radi u pozadini i da svoje alarme beleži u *fast* formatu u /var/log/snort/alert fajlu. Na slici 2. prikazan je primer alarma koji Snort generiše u *fast* formatu, dok su na slici 3. prikazane promenljive koje se koriste za ekstrakciju podataka iz alarma.

```
12/12-00:17:57.772654  [**] [1:384:5] ICMP PING
[**] [Classification: Misc activity] [Priority: 3]
{ICMP} 192.168.56.1 -> 192.168.56.101
```

Slika 2. Primer Snort fast alarma (iz /var/log/snort/alert fajla)

```
$timestamp  [**] [$n1:$n2:$n3] $desc [**]
[Classification: $event] [Priority: $level]
{$protocol} $srcip -> $dstip
```

Slika 3. Promenljiva polja Snort fast alarma

Značenje promenljivih je sledeće: **$timestamp** označava vreme kada je generisan alarm, **$n1** označava Generator ID, odnosno informaciju o tome koja je komponenta Snort-a generisala alarm, **$n2** je Signature ID i identifikuje pravilo koje je generisalo alarm, **$n3** predstavlja Revision ID koji se uvećava sa svakom revizijom pravila, **$desc** sadrži kratak opis događaja, **$event** prestavlja klasifikaciju događaja, odnosno sadrži podatke o tipu događaja koji je izazvao generisanje alarma, **$level** je broj koji označava prioritet alarma, **$protocol** označava protokol koji se koristio prilikom razmene potencijalno malicioznog saobraćaja i na kraju **$srcip** i **$dstip** označavaju izvorišnu i odredišnu IP adresu saobraćaja respektivno.

Nakon izvršene instalacije i konfiguracije Snort-a, prešlo se na istalaciju i kofiguraciju Suricata IDS sistema. Suricata je, kao i Snort, instalirana preko *package manager*-a. Ovaj način instalacije ujedno pojednostavljuje proces instalacije i rešava zavisnosti paketa koji se instalira. Po instalaciji pokrenut je *suricata-update* alat koji omogućava preuzimanje i ažuriranje javno dostupnih gotovih pravila za detekciju napada, a koji dolazi zajedno sa Suricata sistemom. Kao i kod Snort-a, moguće je korišćenje drugih skupova pravila ili ručno pisanje pravila. Suricata je takođe konfigurisana da radi u pozadini i da svoje alarme beleži u sličnom *fast* formatu u /var/log/suricata/fast.log fajlu. Primer alarma koji generiše Suricata dat je na slici 4. dok je prikaz promenljivih polja ovakvog alarma dat na slici 5. Osim ovog, Suricata paralelno generiše alarme i u EVE JSON formatu, ali nam za ovaj rad on nije od značaja. Ukoliko se od sistema zahtevaju visoke performanse, moguće je ovaj tip logovanja isključiti modifikacijom Suricata konfiguracionog fajla, čime se blago podižu performanse sistema.

```
11/19/2021-15:54:14.414183  [**] [1:2027397:1] ET
POLICY Spotify P2P Client [**] [Classification: Not
Suspicious Traffic] [Priority: 3] {UDP}
192.168.56.1:57621 -> 192.168.56.255:57621
```

Slika 4. Primer Suricata fast alarma (/var/log/suricata/fast.log)

```
$timestamp  [**] [$n1:$n2:$n3] $desc [**]
[Classification: $event] [Priority: $level]
{$protocol} $srcip:$sport -> $dstip:$dpost
```

Slika 5. Promenljiva polja Suricata fast alarma

Iz priloženog može se uočiti sličnost između formata koje koriste Snort i Suricata, međutim, postoje i razlike koje onemogućavaju prostu agregaciju ovih alarma. Prva razlika je u fomatu datuma *timestamp*-a, za razliku od Snort-a, Suricata u svoj datum uključuje i godinu. Druga glavna razlika je u tome što Suricata dodaje izvorišni i odredišni port (promenljive **$sport** i **$dport** respektivno). Sva ostala polja imaju isto značenje i sintaksu kao i u Snort-u.

Kao alat za agregaciju alarma iskorišćen je Logstash upravo zbog svoje proširljivosti. Napisan je konfiguracioni fajl koji, najpre u svom input delu preuzima alarm po alarm iz oba fajla (/var/log/snort/alert i /var/log/suricata/fast.log). Obzirom da se u jednom konfiguracionom fajlu vrši prikupljanje različitih alarma, izvršeno je markiranje svakog događaja i kasnija kondicionalna obrada istih. Zatim je u filter sekciji upotrebljen *grok* filter za ekstrakciju podataka iz string polja (slike 2 i 4) i smeštanje korisnih podataka u promenljive (slike 3 i 5). *Date* filter je iskorišćen za parsiranje datuma i vremena, odnosno *timestamp*-a, što omogućava kasniju konverziju u ISO8601 format koji zahteva IDMEF standard. Iskorišćen je još i *mutate* filter pomoću kojeg je markica iz input sekcije iskorišćena kao polje koje nosi informaciju o tome koji IDS je generisao alarm i pomoću kojeg su uklonjena polja koja nisu od značaja (n1,n2,n3 i level). Na kraju u output sekciji, svi obrađeni događaji šalju se u /var/log/logstash/idmef pri čemu se koristi *logstash-codec-idmef* za automatsku konverziju podataka u IDMEF format.

Važno je napomenuti da nazivi promenljivih (slike 3 i 5) nisu nasumično odabrani. Naime, *logstash-codec-idmef* funkcioniše tako što promenljive sa određenim imenima smešta u odgovarajuća polja u IDMEF poruci, a svaki podatak promenljive čije ime ne prepozna smešta se u *AdditionalData* polje, što je iskorišćeno za polja koja potencijalno mogu biti od značaja, a nisu podržana od strane *codec*-a i/ili IDMEF-a. Više o mogućnostima ovog *codec*-a biće rečeno u narednom poglavlju.

IV. MOGUĆNOSTI IMPLEMENTIRANOG SISTEMA

U prethodnom poglavlju opisana je implementacija i princip rada predloženog sistema. Cilj ovog poglavlja je predstavljanje mogućnosti (i ograničenja) implementiranog sistema.

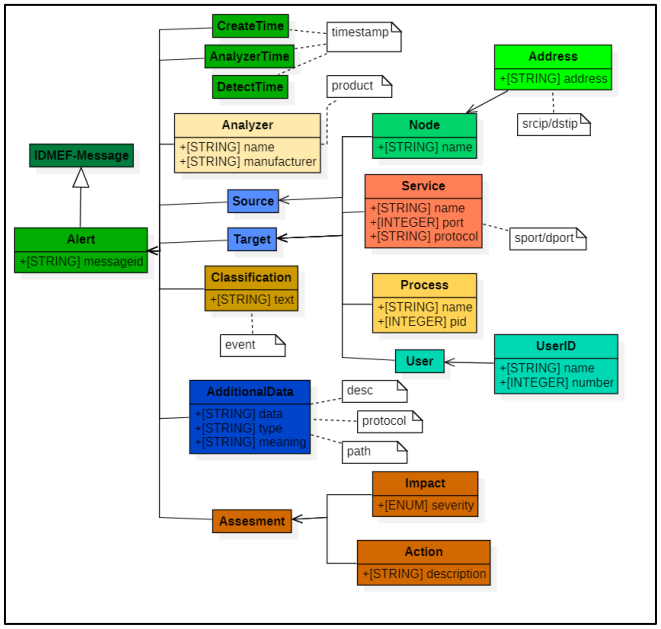
Slika 6. tabelatno prikazuje mogućnosti *logstash-codec-idmef plugin*-a, podatke koje sadrže logovi Snort i Suricata IDS sistema, kao i šta je od svega toga implementirano u predloženom sistemu. Izuzetno je važno pomenuti da ovo nisu sva polja koja se mogu popunjavati u IDMEF poruci, koja može sadržati i do 200 polja u zavisnosti od kompleksnosti poruke i potrebe za tolikom količinom detaljnih podataka. Iz

priloženog se vidi da *codec* podržava tek dvadesetak najbitnijih polja IDMEF standarda, što potencijalno može ograničiti njegovu primenu u kompleksnijim sistemima.

IDMEF Fields					Codec-supported	Contained in Log output		Implemented
						Snort	Suricata	
IDMEF-Message	Alert	Analyzer		name	Yes	No	No	Yes
				manufacturer	Yes	No	No	No
				CreateTime	Yes	Yes	Yes	Yes
				DetectTime	Yes	Yes	Yes	Yes
				AnalyzerTime	Yes	Yes	Yes	Yes
		Source	Node	name	Yes	No	No	No
				Address: address	Yes	Yes	Yes	Yes
			Service	name	Yes	No	No	No
				port	Yes	No	Yes	Yes
				protocol	No	Yes	Yes	No
		Target	Node	name	Yes	No	No	No
				Address: address	Yes	Yes	Yes	Yes
			Process	name	Yes	No	No	No
				pid	Yes	No	No	No
			Service	name	Yes	No	No	No
				port	Yes	No	Yes	Yes
				protocol	No	Yes	Yes	No
			User	UserID: name	Yes	No	No	No
				UserID: number	Yes	No	No	No
		Classification		text	Yes	Yes	Yes	Yes
		Assesment	Impact	severity	Yes	No	No	No
			Action	description	Yes	No	No	No
		AdditionalData		data	Used for data not covered in codec and/or IDMEF standard			
				type				
				meaning				

Slika 6. Tabelarni prikaz mogućnosti sistema

Takođe, može se primetiti da postoje pojedina polja podržana u IDMEF-u koja odgovaraju poljima u logovima, ali koja nisu podržana od strane *logstash-codec-idmef plugin*-a. *Source code*, kao i podržana polja i adekvatni nazivi promenljivih pri upotrebi ovog *codec*-a mogu se naći u [7].



Slika 7. Deo arhitekture IDMEF-a i iskorišćena polja

Na slici 7. grafički je predstavljen deo arhitekture IDMEF-a podržan od strane *logstash-codec-idmef plugin*-a, kao i koja polja logova su upisana i gde u okviru ove arhitekture. Pošto *codec* ne podržava polje *port* u okviru *Service* klase, podaci o portu se upisuju u *AdditionalData* segment. Celokupan klasni dijagram dostupan je na [8].

Za kraj, na slici 8. prikazan je primer konverzije izvučen iz /var/log/logstash/idmef fajla. Konkretan alarm koji je konvertovan u IDMEF formatiran alarm na slici, je alarm generisan od strane Suricata IDS sistema i prikazan je na slici 4.

```
<?xml version="1.0"?>
<idmef:IDMEF-Message xmlns:idmef="http://iana.org/idmef">
  <idmef:Alert messageid="afdf7afb-98ee-4796-b30e-a46e74cad893">
    <idmef:Analyzer analyzerid="lazar-VirtualBox" name="Suricata"/>
    <idmef:CreateTime ntpstamp="0xe54237ea.0x683126e9">2021-11-19T14:49:14+00:00</idmef:CreateTime>
    <idmef:DetectTime ntpstamp="0xe54237ea.0x683126e9">2021-11-19T14:49:14+00:00</idmef:DetectTime>
    <idmef:AnalyzerTime ntpstamp="0xe54237ea.0x683126e9">2021-11-19T14:49:14+00:00</idmef:AnalyzerTime>
    <idmef:Source spoofed="unknown">
      <idmef:Node category="unknown">
        <idmef:Address category="unknown">
          <idmef:address>192.168.56.1</idmef:address>
        </idmef:Address>
      </idmef:Node>
      <idmef:Service>
        <idmef:port>57621</idmef:port>
      </idmef:Service>
    </idmef:Source>
    <idmef:Target decoy="unknown">
      <idmef:Node category="unknown">
        <idmef:Address category="unknown">
          <idmef:address>192.168.56.255</idmef:address>
        </idmef:Address>
        <idmef:name>lazar-VirtualBox</idmef:name>
      </idmef:Node>
      <idmef:Service>
        <idmef:port>57621</idmef:port>
      </idmef:Service>
    </idmef:Target>
    <idmef:Classification text="Not Suspicious Traffic"/>
    <idmef:AdditionalData meaning="desc" type="string">
      <idmef:string>ET POLICY Spotify P2P Client</idmef:string>
    </idmef:AdditionalData>
    <idmef:AdditionalData meaning="protocol" type="string">
      <idmef:string>UDP</idmef:string>
    </idmef:AdditionalData>
    <idmef:AdditionalData meaning="path" type="string">
      <idmef:string>/var/log/suricata/fast.log</idmef:string>
    </idmef:AdditionalData>
  </idmef:Alert>
</idmef:IDMEF-Message>
```


Slika 8. Primer IDMEF formata

ZAKLJUČAK

U ovom radu predložen je sistem za agregaciju heterogenih alarma različitih IDS sistema upotrebom IDMEF formata. Konkretni IDS sistemi čiji se alarmi agregiraju i konvertuju su Snort i Suricata. Agregacija se vrši korišćenjem Logstash alata, a za konverziju iskorišćen je *logstash-codec-idmef plugin*. Glavna prednost sistema je jednostavna implementacija i mogućnost relativno brzog podizanja istog. Mana ovog sistema je sam *codec* za konverziju u IDMEF koji pruža dosta ograničene mogućnosti u odnosu na pun potencijal IDMEF formata. Generalno, predloženo rešenje je korisno i praktično u *lightweight* sistemima, dok bi za ozbiljnije sisteme većeg obima i zahteva trebalo naći adekvatnije rešenje za konverziju u IDMEF umesto pomenutog *codec*-a radi većeg iskorišćenja potencijala koji pruža IDMEF format.

ZAHVALNICA

Autor ovog rada bi želeo da se zahvali prof. dr Vladimiru Ćiriću i mast. inž. Nađi Gavrilović na uloženoj energiji, vremenu, savetima i podršci.

LITERATURA

[1] Luigi V. Mancini, Roberto Pietro, “Intrusion Detection Systems”, Springer US, 2008.

[2] Wu, Yu-Sung, et al. "Collaborative intrusion detection system (CIDS): a framework for accurate and efficient IDS." 19th Annual Computer Security Applications Conference, 2003. Proceedings.. IEEE, 2003.

[3] Roesch, Martin. "Snort: Lightweight intrusion detection for networks." Lisa. Vol. 99. No. 1. 1999.

[4] https://suricata.readthedocs.io

[5] https://datatracker.ietf.org/doc/html/rfc4765

[6] Bajer, Marcin. "Building an IoT data hub with Elasticsearch, Logstash and Kibana." 2017 5th International Conference on Future Internet of Things and Cloud Workshops (FiCloudW). IEEE, 2017.

[7] https://www.rubydoc.info/gems/logstash-codec-idmef/0.9.3/LogStash/Codecs/IDMEF

[8] https://www.secef.net/idmef-schema/IDMEF/