Università degli Studi di Pisa

Dipartimento di Informatica
Corso di Laurea in Informatica (Classe L-31)

Utilizzo degli allarmi di sicurezza nel monitoraggio del traffico di rete

Laureando:
Marco Bianucci
Matricola 464134

Relatore
Luca Deri

A.A. 2018/2019

Ringraziamenti

Arrivato alla fine di un percorso faticoso intrapreso e raggiunto in età ormai adulta
ringrazio tutti quelli che mi hanno supportato e sopportato in questa avventura.

Ringrazio il Prof. Luca Deri
per avermi portato con molta comprensione del mio status inconsueto
alla conclusione di un percorso che mi gratifica molto
sia dal punto di vista professionale che morale.

Ringrazio la mia famiglia, mia moglie e i miei figli
per la pazienza dimostratami durante il percorso di studio

Ringrazio i miei colleghi di lavoro
per il supporto che mi hanno dato ed in particolare i Professori
Rolla Pierangelo, Lucchesi Mauro, Capaccioli Simone, Labardi Massimiliano,
Prevosto Daniele, per avermi invogliato a riprendere lo studio e ad avermi spronato.

Ringrazio i vari studenti compagni di avventura che mi hanno dato un valido
aiuto e una bella spinta dettandomi in qualche modo le scadenze da rispettare
trattandomi come un loro coetaneo quasi come se le nostre differenze di età non ci fossero.

Ringrazio il Team di sviluppo di NTOPNG e in particolar modo Alfredo Cardigliano
per gli aiuti ed il supporto che mi hanno dato.

Grazie a tutti
Marco Bianucci

Utilizzo degli allarmi di sicurezza nel monitoraggio del traffico di rete

Sommario

1 Introduzione

Nell'ambito del monitoraggio del traffico dati sulla rete LAN, WAN e Internet è ormai fondamentale controllare i volumi di dati oltre che la qualità del traffico per prevedere malfunzionamenti di dispositivi di rete che non fanno il loro lavoro, sia intercettare le attività malevoli che vanno dal furto di informazioni ad attacchi software [43] con l'obiettivo di mettere in crisi le normali funzionalità dei dispositivi di gestione e l'attività degli utenti.

Gli attacchi sono di molti tipi, alcuni dei più comuni sono la diffusione di virus, spyware, worm [32] (codice autoreplicante), ransomware fra cui Cryptolocker [4, 49], azioni di saturazione del traffico per far decadere i servizi (DoS - Denial of Service) fra cui molto frequenti quelli di tipo SYN Scan e SYN Flooding sui messaggi con protocollo TCP [38] o quelle che utilizzano risorse non autorizzate ad es. mail spamming.

Tradizionalmente i sistemi di misura del traffico di rete e quelli per la rilevazione di attacchi di rete più inclini a lanciare allarmi diretti per eventi sospetti non si integrano fra loro rendendo necessario ai gestori di utilizzare più di una "consoles" per il monitoraggio e che solitamente non sono correlate fra loro.

Altro aspetto diffuso è che molti dei software specializzati nel verificare il corretto funzionamento della rete detti "Intrusion Detection System" (**IDS** - sistema di individuazione di intrusione) e "Instrusion Prevention System" (**IPS** - sistema di prevenzione di intrusione) per ragioni di performance ed efficienza non dispongono di interfaccia grafica nativa, operano in background e spesso inviano o registrano in log file report degli eventi rilevati in modo indipendente.

1.1 Obiettivo, mettere insieme visibilità e sicurezza

Visibilità della rete intesa come poter monitorare il traffico dati per consentire di vedere tutto ciò che accade e valutarne:

- Performance

- Prestazioni delle applicazioni
- Rilevamento dispositivi connessi
- Qualità e tipologia dei dati che transitano

Tipicamente il monitoraggio avviene controllando l'intestazione (header) e solo i primi byte dei pacchetti di dati in modo simile a come opera la DPI (vedi par. 2.3), che è un'operazione necessaria per determinare, mittente, destinatario, protocollo e quanto necessario per identificare e catalogare in "flussi" le comunicazioni intransito.

Sicurezza vista dal lato degli amministratori come:

- Avere bisogno di una chiara immagine del traffico che fluisce nella loro rete e collocare gli eventi di sicurezza nel giusto contesto.
- Poter correlare gli eventi di sicurezza con il traffico di rete per avere una migliore visibilità di ciò che sta accadendo e della causa principale delle minacce.
- Potere accorgersi che singoli eventi sull'immediato considerati individualmente innocui sono in realtà piccoli pezzi di fenomeni più gravi.

La sicurezza tipicamente avviene decodificando il contenuto dei flussi, operazione in cui sono specializzati gli IDS/IPS e confrontandolo con delle regole, che possono essere anche arbitrariamente definite, che servono per identificare i malware e altri problemi, quando sono violate.

L'idea è quindi che sarebbe interessante correlarle visibilità e sicurezza essendo due tematiche complementari dello stesso contesto.

Si potrebbero poi confrontare gli attacchi con le "liste nere" degli IP noti dal rilevamento del malware con gli allarmi stessi ed i dispositivi connessi rilevati con l'analisi del monitoraggio della rete per avere quadri di riscontro più efficaci degli eventi indesiderati.

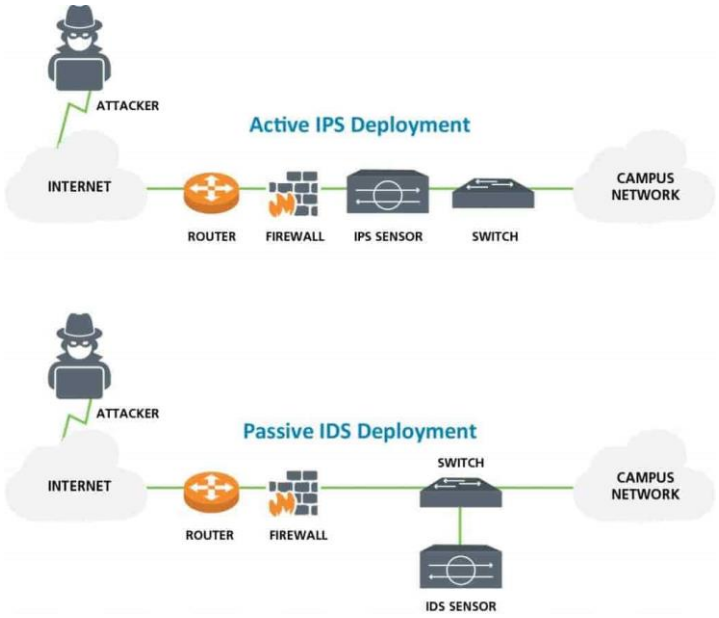
L'obiettivo quindi che ci proponiamo è di riuscire ad accorpare gli allarmi del monitoraggio del traffico forniti da applicativi IDS/IPS a quella di altri tool di "network

monitoring” preposti al rilevamento di anomalie e catalogazione dei flussi di dati, il tutto magari raccolto in un'unica veste grafica, cioè “un cruscotto” riepilogativo (dashboard) unico che riepiloga tutti i risultati nell’idea di renderne più semplice la visione per chi si occupa di controllare la rete.

1.2 IDS, IPS, IDPS in breve

“Intrusion Detection System” (**IDS** - sistema di individuazione di intrusione) e “Instrusion Prevention System” (**IPS** - sistema di prevenzione di intrusione) [21], sono tecnologie complementari nell'ambito della sicurezza delle reti e sono in grado di lavorare in sinergia. Entrambe sono accomunate dall'analisi del traffico di rete ed entrambe entrano in funzione (di alert o di prevention) sulla base del matching fra determinate regole predefinite e/o fornite dall'amministratore ed i pacchetti in transito facendone un’analisi approfondita a livello di contenuto e non solo di intestazione.

Proprio per questi motivi, diversi software implementano sia la funzione di “detection” che di “prevention” in un unico “engine”, dando così origine a prodotti ibridi noti con l'acronimo di **IDPS** o Instrusion Detection and Prevention System.



(Immagine tratta da: https://www.digitalplanet.ie/ips-ids/)

Ormai tutti i nostri sistemi di rete sono più o meno dotati di firewall realizzati in software e/o hardware che eseguono azioni come il blocco ed il filtraggio del traffico non

richiesto o che non soddisfi criteri specifici, ma i firewall, sono solo una prima barriera di difesa della nostra rete locale e non sono in grado di catalogare ed analizzare il traffico.

Un sistema IPS/IDS, magari coadiuvato da tool di analisi dei protocolli, in più è in grado di rilevare e avvisare un amministratore di sistema che un attacco è in atto o prevenirlo analizzando e classificando il contenuto del traffico sia a livello di protocollo che a livello di traffico anomalo che in genere è sintomo di problemi in atto o di tentativo di violazione di un sistema da parte di malintenzionati.

Un sistema IPS/IDS è in genere una seconda linea di difesa molto più "fine" che opera di solito dopo un firewall e può anche essere distribuita nei vari nodi (switch) di smistamento locale con modalità di intervento sia attivo, bloccando il traffico indesiderato istantaneamente che passivo, con invio di allarmi e log eventi.

1.3 Descrizione dell'attività svolta per correlare gli allarmi di Suricata in NTOPNG via Syslog

Recentemente grazie alla definizione comune di alcuni codici univoci che mettono in correlazione i vari pacchetti dei singoli flussi dati in rete fra cui il "COMMUNITY_ID [1]" è possibile, anche se questo meccanismo è ancora poco utilizzato, collegare informazioni di rete e sicurezza cercando di mettere in correlazione i "flussi" delle comunicazioni dati monitorate.

L'obiettivo è proprio quello di cercare di mettere in correlazione informazioni di rete e di sicurezza cercando di usare proprio tali ID univoci dei flussi di dati in transito, ricalcolandoli al bisogno, in modo da creare un unico sistema di monitoraggio, una consolle unica, che sia arricchita dalle informazioni di sicurezza generate dagli IDS.

- Suricata è un ottimo strumento IDS/IPS per analizzare i protocolli selezionati, estrarre le metriche chiave ed emettere avvisi basati sul contenuto del flusso guidato da regole esterne.

- NTOPNG è un tool di monitoraggio che è in grado di raccogliere informazioni da varie fonti (pacchetti, NetFlow, sFlow), analizzarle in un formato completo ed emettere avvisi. Tutto in un'unica applicazione, con requisiti minimali.
- E se potessimo unificare questi due strumenti open source in un unico strumento in grado di fornire la migliore soluzione per integrare sicurezza e visibilità?

Si è quindi provato ad utilizzare uno strumento per il monitoraggio di rete NTOPNG open-source esistente, realizzato dalla comunità ntop [20], che è stato esteso nelle sue funzionalità per essere messo in grado di ricevere le informazioni di un altro strumento sempre open-source per il rilevamento di attacchi di rete "Suricata" [3], un IDS che appartiene alla categoria di quelli che operano con l'analisi del contenuto dei pacchetti "per signatures" (per firme).

Le due piattaforme sono state messe, con opportuni aggiustamenti, in comunicazione affinché fosse possibile da NTOPNG ricevere i report di potenziali (o reali) attacchi alla rete rilevati da SURICATA che opera per "signatures", per firme, cioè facendo una decodifica esplicita dei pacchetti analizzandone il contenuto.

Come tramite, la comunicazione è avvenuta attraverso Syslog, un applicativo noto normalmente disponibile nelle distribuzioni Linux che utilizza un protocollo di interscambio che nelle funzionalità di base è ormai abbastanza standardizzato.

Quello che ci aspettiamo di ottenere è creare vantaggi per entrambe i tool.

Per Suricata:

- Fornire una GUI Web che non richieda database esterni o strumenti di terze parti per dotarsi di visibilità di cosa accade sulla rete.
- Migliorare Suricata con metriche di rete di cui non dispone
- Fornire agli utenti Suricata le funzionalità esistenti e disponibili in ntop (ad es. uno storage efficiente dei flussi basato su nIndex [51] o allarmi basati su Slack [50]).

Per NTOPNG:

- Aggiungere i vantaggi dell'analisi del traffico basata sul controllo delle “signatures”.
- Unire gli avvisi sul traffico Suricata con quelli che già gestisce per ricavare il meglio di entrambi.

NTOPNG è normalmente in grado di ricevere dati di misurazione del traffico accumulati e monitorati da opportuni probe (fra cui ad esempio nPROBE [26]) esterni di monitoraggio dislocati nella rete che gli vengono trasmessi verso opportuni “collezionatori” software per essere opportunamente processati.

Quindi l’idea è stata quella di estendere NTOPNG con un ulteriore opportuno “collezionatore” di messaggi per ricevere dati anche da SURICATA l’impiego di SYSLOG si è reso utile per snellire il lavoro di interfacciamento fra protocolli e l’invio automatico in rete degli allarmi generati.

Operativamente in breve, dapprima si è cercato di familiarizzare con Suricata facendolo funzionare in modalità nativa in cui salva le informazioni fornite sui classici file di log generati (“eve.log”, etc.), dopo averlo tenuto un poco in esecuzione e dopo avergli fatto analizzare alcuni file pcap con traffico malevole si sono prelevati i log-file ottenuti e si sono analizzati.

Verificati i contenuti in formato JSON [7] dei log eventi (eve-log) ottenuti abbiamo cominciato a definire quali erano le informazioni che ci interessavano e quali no.

Di seguito si è cercato di trovare di far dialogare suricata con Syslog facendo una serie di personalizzazioni dei rispettivi file di configurazione monitorando la messaggistica fra i due applicativi mediante “Wireshark” [39] che è un software di libero uso specializzato nel monitorare i pacchetti “grezzi” che transitano nel traffico di rete visibile dalla postazione di lavoro adottata.

Fatte le opportune verifiche e messe a punto si è stato poi modificato il codice di NTOPNG per permettergli di ricevere i messaggi rilanciati via Syslog da Suricata cercando

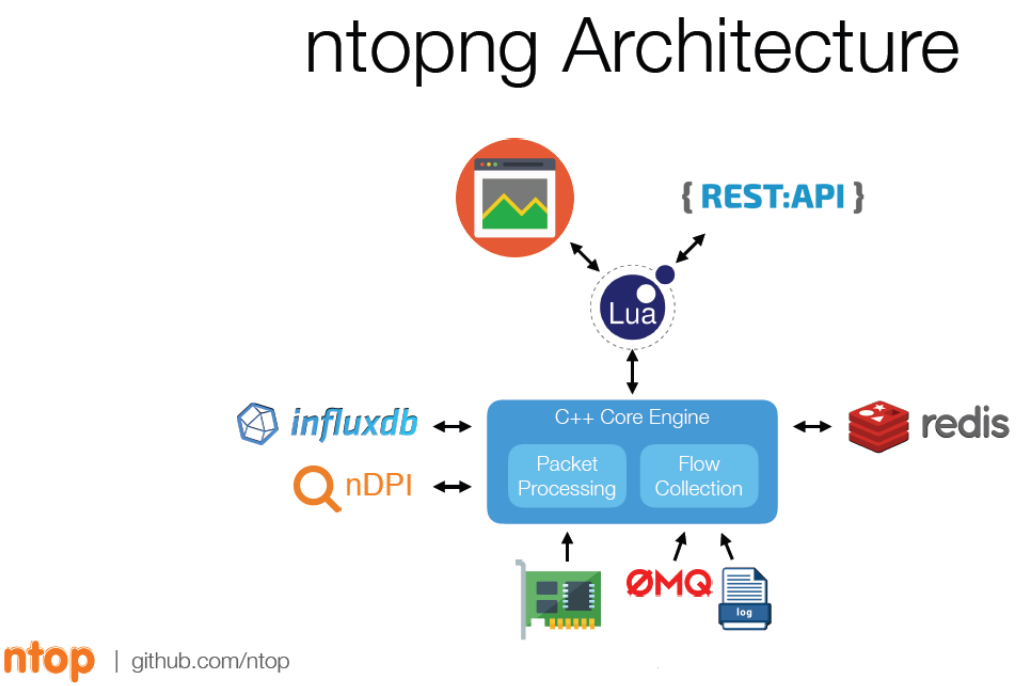
di correlarli alla normale analisi di flusso rilevata e cercando di visualizzarli insieme alle altre informazioni normalmente “captate” da NTOPNG.

Il risultato finale è stato poi validato sia utilizzando tracce di attacchi di rete liberamente disponibili e mediante la verifica degli elenchi di IP che si sono rivelati come “minacce emergenti” o “IP catalogati come pericolosi” da enti specializzati [5, 6] che mediante verifiche di cattura dei normali pacchetti di traffico in transito rilevati in una rete locale con volumi di traffico abbastanza consistenti.

2 NTOPNG

NTOPNG di ntop [20] è un noto programma di monitoraggio del traffico open source e di uso gratuito nelle funzionalità di base che può lavorare in locale gestendo una anche più schede ethernet simultaneamente e/o ricevere conteggi e campionamento del traffico registrati in punti sensibile della rete ed inviati da opportune sonde software (vedi nProbe [26]) installate e dislocate negli "snodi" (in genere switch) da monitorare ad un server di "collezionamento" ed analisi del traffico.

NTOPNG in virtù degli obiettivi preposti ha subito delle integrazioni software opportune e quindi per utilizzare le nuove funzionalità introdotte che sono ad oggi in fase di testing e soggette a continuo aggiornamento oltre a quelle già contenute nella sua architettura volte al continuo miglioramento dell'efficienza e della velocità di processo dei pacchetti, utilizzando nella sua architettura tecnologie come i data-store "in-memory" REDIS [19] , il linguaggio multi paradigma di scripting LUA [29] ed altre strategie.



2.1 Installazione di NTOPNG in ambiente Linux

Per L'installazione di NTOPNG è sempre meglio riferirsi alla pagina web ufficiale del gruppo (https://www.ntop.org/) dei suoi sviluppatori per ottenere il codice dal repository ufficiale più aggiornato disponibile dell'applicativo

NTOPNG per funzionare necessita di avere preinstallato e correttamente compilato anche altri due moduli software anche essi open source "nDPI" e "PF_RING" forniti e scaricabili sempre dal sito degli autori, quindi per essere certi di utilizzare l'ultimo repository corretta è meglio seguire le istruzioni suggerite dal sito ufficiale [2], vedi la voce del menu "GET STARTED" ed ai relativi link fra cui:

https://www.ntop.org/get-started/download/

L'ambiente di lavoro preferenziale per NTOPNG è Linux ed i test svolti in questo lavoro sono tutti stati effettuati in ambiente Linux, l'applicativo è cmq disponibile per funzionare anche in altre piattaforme HW/SW normalmente diffuse fra cui in ambiente Windows, ARM, MAC ed altri.

Per effettuare il download del codice di ntop, la procedura più consueta e prelevarne il sorgente dal servizio applicativo GitHub [22, 23] che se non è già presente sul nostro sistema è necessario installare il modulo di libreria “git”. Lo si può fare ad esempio anche direttamente da shell con il comando:

```
sudo apt install git
```

Una volta scaricati codici sorgente delle tre applicazioni è necessario in sequenza prima compilare e configurare nDPI [25, 26] e PF_RING [27], solo successivamente compilare NTOPNG perché quest’ultimo ne utilizza alcune funzionalità che dovranno essere già disponibili nella macchina su cui lo stiamo installando già al momento della sua compilazione.

Riporto qui di seguito la lista dei comandi shell Linux per la compilazione e configurazione dei tre applicativi presi dal sito ufficiale che è sempre bene consultare in quanto potrebbero esserci variazioni o modifiche alle procedure dovute ai continui aggiornamenti implementativi del codice.

nDPI:

```
git clone https://github.com/ntop/nDPI.git
cd nDPI
./autogen.sh
./configure –with-pic
make
```

PF_RING:

```
git clone https://github.com/ntop/PF_RING.git
cd PF_RING/kernel
make
sudo insmod ./pf_ring.ko
cd ../userland
make
```

NTOPNG:

```
git clone https://github.com/ntop/ntopng.git
cd ntopng
./autogen.sh
```

```
./configure
make
sudo make install
```

seguendo le istruzioni suggerite dal sito ufficiale in sostanza i tre tool, NTOPNG compreso, sono dotati dagli autori di una serie di script "./autogen.sh" e "./configure" che eseguite in sequenza analizzano il sistema in uso e personalizzano auto-generandolo dopo alcuni passaggi il meta-codice del file "MAKEFILE" che servirà per compilare il codice C++ sorgente quando verrà invocato il comando "make" e cioè:

```
autogen.sh -> produce meta-istruzioni per lo script configure
configure -> produce il "MAKEFILE" definitivo per compilare correttamente il relativo
             pacchetto software sulla nostra macchina.
```

In fine l'ultima istruzione "make install" avvia le sotto istruzioni specifiche del makefile per installare operativamente NTOPNG e quant'altro a lui necessario sulla macchina dove lo stiamo compilando e dove andrà in esecuzione.

Alcuni "Warning" …

Può accadere che durante la fase di esecuzione dello script "configure" il processo si interrompa perché mancano delle librerie di sistema che saranno segnalate opportunamente con una serie di messaggi, esse dovranno essere aggiunte eventualmente a mano da shell tramite la procedura standard:

```
sudo apt-get install <nome_libreria>
```

2.2 Esecuzione di NTOPNG in ambiente Linux

Il programma è a linea di comando ed opera in background similmente ad un demone fornendo un servizio server sulla porta 3000. Secondo le opzioni richieste all'avvio e/o in fase di compilazione può anche stampare nella shell alcuni messaggi tipicamente per funzionalità di debug o log-data per supervisionarne in fase di testing le funzionalità

Per avviare il programma nella modalità base seve l'autorizzazione "super user" e quindi generalmente occorre il comando di avvio seguente, a cui seguirà la richiesta della password dell'utente "root" della macchina in uso:

```
sudo /etc/init.d/ntopng start
```

Volendo verificare che il programma sia andato effettivamente in esecuzione in background lo si può fare anche da shell filtrando il report del comando di sistema "netstat" se e quali servizi sono attivi sulla porta locale 3000 con il comando:

```
sudo netstat -tulpn | grep :3000
```

NTOPNG quando è in esecuzione apre un servizio server sulla porta locale 3000 dove con un comune browser è possibile interagire con una "dashboard" (un cruscotto, una console) con pagine web generate da ntop stesso all' indirizzo web locale.

```
http://"your-server-ip":3000
```

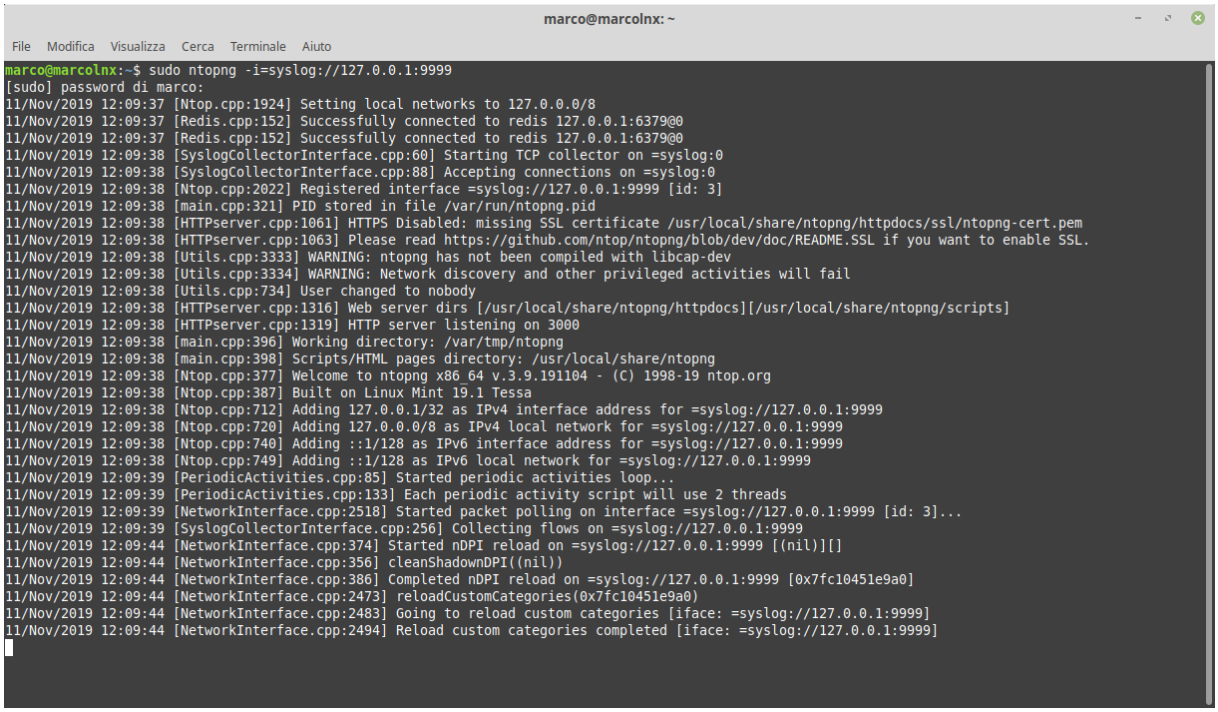
Dove [your-server-ip] è il "localhost name" della macchina in uso che si può ottenere digitando con il comando "hostname" da shell

A riguardo dell'interfacciamento di NTOPNG con Suricata come già accennato lo scambio di messaggi avverrà sfruttando le funzionalità di un altro demone di sistema presente nell'ambiente Linux che farà da tramite "Syslog" [14, 37] infatti fra le varie opzioni di Suricata è già previsto dai suoi sviluppatori l'interfacciamento "Suricata-Syslog".

A sua volta Syslog andrà configurato per fare da tramite verso NTOPNG come verrà poi più dettagliato del capitolo 4.2 a tal proposito sono state pubblicate delle note sul sito ufficiale di ntop sull'integrazione effettuata per la "collaborazione" dei tre software di interesse al seguente link [15]:

https://www.ntop.org/guides/ntopng/advanced_features/suricata.html

Di conseguenza per operare con Syslog, NTOPNG andrà avviato su un'interfaccia apposita, vedi il parametro "-i", tipo ad esempio "ntopng -i syslog://127.0.0.1:9999" (i dettagli più avanti).



2.3 Note su nDPI e PF_RING

nDPI e PF_RING sono moduli sviluppati dallo stesso "Ntop-Team" utilizzati dal progetto globale NTOPNG e riusabili anche per terze applicazioni.

nDPI è basato sulla tecnica Deep Packet Inspection (DPI) [24, 25] (ispezione in profondità dei pacchetti) che è un tipo di elaborazione dei dati che controlla in dettaglio i messaggi che transitano a livello di protocollo tipicamente a livello "Rete" (Network/Internet - layer 2) e "Trasporto" (Transport – layer 3) [16, appendice 9.3] catalogandoli ed analizzandone il contenuto, registrandoli di conseguenza.

La DPI viene spesso utilizzata per garantire che i dati (in genere gli header e parte del messaggio) siano nel formato corretto, per verificare la presenza di pacchetti danneggiati o dannoso, intercettarlo ed eventualmente bloccarlo analizzando i pacchetti stesi a partire dalle loro intestazioni sia a livello IP che TCP-UDP.

La libreria nDPI è un “super set” cioè una rielaborazione ed estensione gestita dal team di sviluppo di NTOPNG della popolare libreria OpenDPI, sotto licenza LGPL, il suo obiettivo è quello di estendere la libreria originale aggiungendo nuovi protocolli che sarebbero altrimenti disponibili solo sulla versione a pagamento di OpenDPI.

nDPI è utilizzato sia da NTOPNG che dalle sonde nProbe [26], sviluppate che esse dal ntop-team, per aggiungere il rilevamento a livello di applicazione per una consistente lista di protocolli indipendentemente dalla porta utilizzata.

Ciò significa che è possibile rilevare sia protocolli noti su porte standard e non standard, ad es. rilevare http utilizzato su porte diverse da 80 oppure rilevare il traffico Skype che occasionalmente effettua sulla porta 80 per ingannare i router ed avviare i suoi servizi.

Questo perché al giorno d'oggi il concetto di “port = application” non è più valido, molte porte che storicamente erano standard per un determinato protocollo e ad esso associate in modo pressoché esclusivo sono sempre più spesso utilizzate per più di un servizio in modo esclusivo.

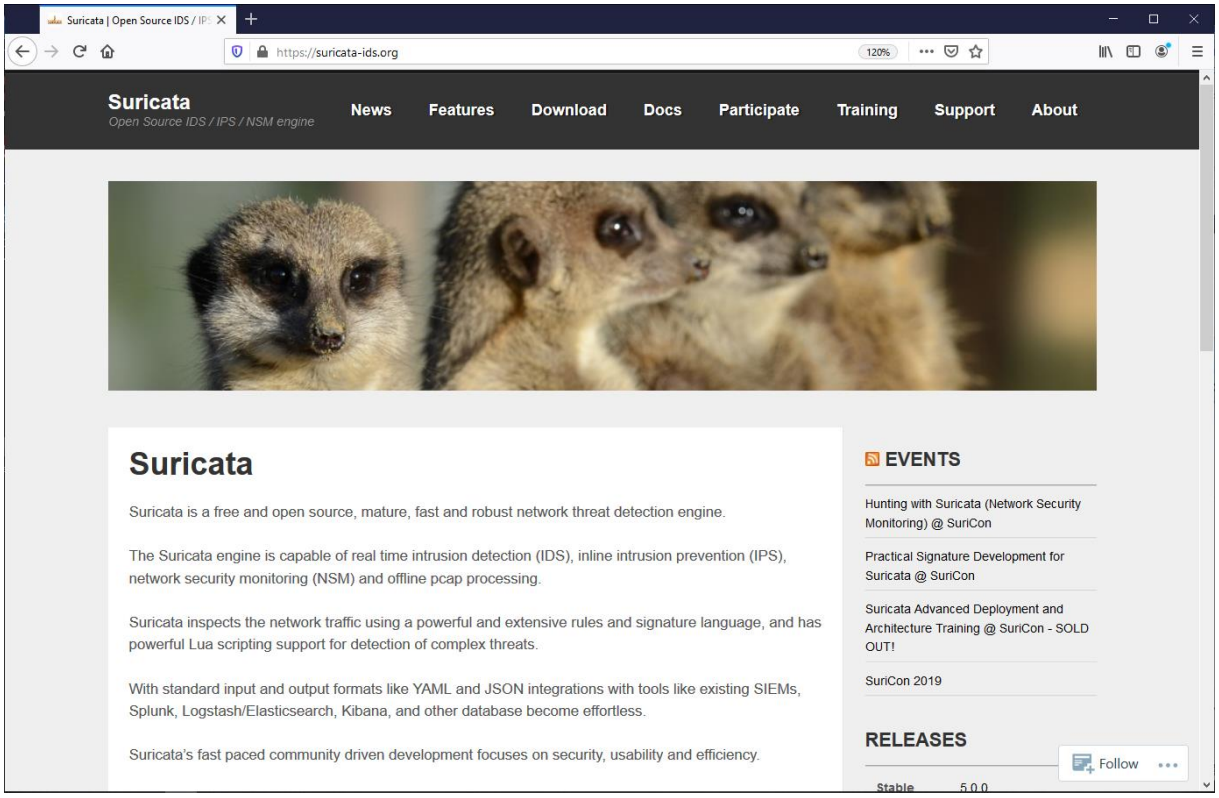
PF_RING [27] è un nuovo tipo di socket di rete che fornisce una libreria di acquisizione di pacchetti ad alta velocità che trasforma un PC in un efficiente ed economico strumento di misurazione della rete adatto sia all'analisi dei pacchetti sia all'analisi attiva del traffico che ne migliora notevolmente la velocità di acquisizione e si propone come uno strumento per tutti coloro che devono gestire molti pacchetti al secondo compatibilmente con l’hardware adottato.

3 Suricata

Suricata [3] è un applicativo open source e di proprietà di una fondazione no profit gestita dalla comunità, la Open Information Security Foundation (OISF).

Suricata è sviluppato dall'OISF, da segnalare SuriCon [36], una conferenza annuale sponsorizzata proprio da OISF dove vengono proposte novità e sviluppi correlati anche sviluppati da terze parti [45].

Suricata è un software di rilevamento delle minacce di rete gratuito e open source, veloce e robusto che è in grado di rilevare e prevedere in tempo reale le intrusioni (IDS - IPS) di effettuare il monitoraggio della sicurezza della rete (NSM -Network Security Monitoring) ed eseguire l'elaborazione offline analizzando file di campionamenti di rete salvati secondo il formato ormai standard “pcap” [9, 11, 12, 18].



Suricata controlla il contenuto traffico di rete dei protocolli più comuni utilizzando regole prefissate e/o custom grazie ad un ricco linguaggio di riconoscimento della “signature” dei messaggi utilizzato per il rilevamento di minacce complesse e dispone di

molti formati di output standard fra cui JSON [7, 10] che è quello su cui si è concentrato il lavoro di interfacciamento Suricata-NTOPNG.

In Suricata per i parametri di configurazione sono utilizzati file in formato “YAML” [13]. È necessario porre attenzione alla versione SW di suricata da utilizzare che deve essere la 4.0 o superiore. Nelle versioni precedenti nella messaggistica di output non contiene alcune informazioni necessarie agli scopi preposti mentre noi ci attendiamo da analizzare messaggi con struttura JSON simili a quello riportato qua sotto sia nel formato che nei contenuti ma nelle versioni più vecchie molti campi attesi non sono presenti.

Attualmente sono molto diffuse le versioni 4.1.3 e 4.1.4 ma è stata rilasciata anche la versione 5.0.0.

```
Mar 29 16:16:23 pc-monitoring1 suricata[32748]: {"timestamp": "2019-03-
29T16:16:23.000591+0100", "flow_id": 868314455276614, "event_type": "flow", "src_ip":
"192.12.193.65", "src_port": 5353, "dest_ip": "224.0.0.251", "dest_port": 5353, "proto":
"UDP", "app_proto": "failed", "flow": {"pkts_toserver": 1, "pkts_toclient": 0,
"bytes_toserver": 117, "bytes_toclient": 0, "start": "2019-03-29T16:15:52.523334+0100",
"end": "2019-03-
29T16:15:52.523https://www.windfinder.com/weatherforecast/castiglione_della_pescaia334+0100
", "age": 0, "state": "new", "reason": "timeout", "alerted": false}}
```

Fra i vari messaggi inviati da Suricata si ha particolare interesse per quelli del gruppo “event_type” di tipo NETFLOW ed ALERT, associati di norma a dei flussi intercettati che hanno violato una qualche regola di sicurezza o di preallarme prefissate.

Tali messaggi in genere contengono campi informativi del flusso di pacchetti del tipo del dettaglio seguente.

```
NetFlow_Parse!

{ "pkts": 26, "bytes": 2392, "start": "2019-02-23T20:38:25.482560+0100", "end": "2019-02-
23T20:50:50.201112+0100", "age": 745, "min_ttl": 128, "max_ttl": 128 }

note: “age” è il tempo di durata del flusso osservato in secondi dato dalla differenza.
“start-end” e pkts è il numero di pacchetti che ha interessato quel flusso.
```

L’interesse è proprio per questi campi che ci servono per ricalcolare il parametro univoco “COMMUNITY_ID” associato al flusso stesso che ci occorre per correlare le informazioni catalogate da NTOPNG con quelle inviate da Suricata.

3.1 Installazione di Suricata in ambiente Linux

Come già accennato anche Suricata è un programma di monitoraggio del contenuto del traffico su cui è possibile definire delle regole di monitoraggio eventualmente attive in grado non solo di verificare ma anche di bloccare il traffico dei flussi ritenuti “malevoli”.

Come NTOPNG lavora a linea di comando e non ha interfaccia grafica nativa ci sono però molte applicazioni sviluppate da terze parti che attingono dai file di LOG su cui registra gli eventi accaduti e i conteggi del traffico fornendone una veste grafica più “user friendly”.

Fra le varie modalità di reportistica Suricata può lavorare in locale o inviare dati a terze applicazioni in particolar modo può interagire con il servizio SYSLOG generalmente presente nelle distribuzioni Linux.

Syslog è un applicativo demone locale che può ricevere messaggi da processi locali ed eventualmente instradarli via rete ad altri applicativi remoti.

Syslog può essere presente in alcune varianti a seconda della distribuzione Linux che stiamo usando.

Per default Suricata opera in modalità di logging locale salvando su disco diversi file di reportistica. In ambiente Linux quelli più rilevanti sono rilocati in:

```
-> /var/log/suricata/….
```

e sono:

```
eve.json
fast.log
```

```
stats.log
suricata.log
```

Per l'installazione di Suricata si possono seguire le istruzioni dei seguenti link o più facilmente si può usare il gestore applicazioni ormai presente in ogni distribuzione Linux, la differenza su cui fare attenzione è che spesso il gestore automatico non permette di personalizzare le librerie e le funzionalità accessorie e non è garantito che installi l'ultima versione del codice disponibile ma una meno recente che non faccia al caso nostro.

Sul sito ufficiale Suricata sono disponibili molte informazioni ed anche una sessione in formato web del manuale d'uso molto completa versione per versione rilasciata, non che le istruzioni per ottenere dal repository l'ultima versione utile ufficiale del programma

```
https://redmine.openinfosecfoundation.org/projects/suricata
https://suricata.readthedocs.io/en/suricata-4.1.2/install.html
….
https://suricata.readthedocs.io/en/suricata-5.0.0/install.html
```

Riportiamo per esempio alcune note per l'installazione prelevate dalle pagine web ufficiali della versione 4.1.4 ("https://suricata.readthedocs.io/en/suricata-4.1.4/install.html#install-binary-packages") ma che in modo molto simile si trovano per tutte le altre versioni rilasciate per un ambiente Ubuntu o distribuzione da esso derivata come ad esempio Mint.

Per ottenere sempre l'ultima copia aggiornata stabile di l'Open Information Security Foundation (OISF) che si occupa dello sviluppo di suricata mantiene sempre un repository "`suricata-stable`" che si può installare come segue:

```
sudo add-apt-repository ppa:oisf/suricata-stable
sudo apt-get update
sudo apt-get install suricata
```

Se invece si dispone del codice sorgente esplicito (ad es. della versione 4.1.0) si può anche procedere nel seguente modo:

```
tar xzvf suricata-4.1.0.tar.gz
cd suricata-4.1.0
./configure
make
make install
```

A conclusione dell'installazione i vari file di interesse di Suricata saranno localizzabili nelle seguenti tre directory:

Suricata file in:	`/usr/local/bin/`
File di configurazione:	`/usr/local/etc/suricata/`
File di output	`/usr/local/var/log/suricata`

Il programma tuttavia ha varie dipendenze da moduli di terze parti, quindi per una corretta installazione si rimanda alla consultazione delle istruzioni pubblicate sul sito ufficiale che riporta dei set di librerie da installare secondo le necessità:

Minimal:

```
apt-get install libpcre3 libpcre3-dbg libpcre3-dev build-essential libpcap-dev   \
                libyaml-0-2 libyaml-dev pkg-config zlib1g zlib1g-dev \
                make libmagic-dev
```

Recommended:

```
apt-get install libpcre3 libpcre3-dbg libpcre3-dev build-essential libpcap-dev   \
                libnet1-dev libyaml-0-2 libyaml-dev pkg-config zlib1g zlib1g-dev \
                libcap-ng-dev libcap-ng0 make libmagic-dev libjansson-dev        \
                libnss3-dev libgeoip-dev liblua5.1-dev libhiredis-dev libevent-dev \
                python-yaml rustc cargo
```

Extra for iptables/nftables IPS integration:

```
apt-get install libnetfilter-queue-dev libnetfilter-queue1  \
                libnetfilter-log-dev libnetfilter-log1      \
                libnfnetlink-dev libnfnetlink0
```

For Rust support:

```
apt-get install rustc cargo
```

3.2 Esecuzione di Suricata in ambiente Linux

Se non necessario personalizzare le regole di intervento sul controllo del traffico suricata si può avviare con le regole di default appena installate mediante esplicite opzioni a linea di comando, a tal proposito si possono consultare anche le note rilasciate dagli sviluppatori di ntop [15].

Fra i parametri richiesti è necessario indicare su quale ID della porta LAN Hardware dove si vuole che deva essere monitorato il traffico, ad esempio l'ID "enp0s3" il tag tipico della risorsa HW ethernet utilizzato nelle distribuzioni di Linux Mint.

Negli esempi seguenti le opzioni (-c) indicano dove e quale file di configurazione "yaml" caricare, l'opzione (-s) indica a directory dove è l'elenco delle regole e può essere omesso se si desidera utilizzare quelle di default mentre l'opzione (-i) indica l'interfaccia LAN da usare.

```
sudo suricata -c /etc/suricata/suricata.yaml -s signature.rules -i enp3s0
sudo suricata -c /etc/suricata/suricata.yaml -i ens33
sudo suricata -c /usr/local/etc/suricata/suricata.yaml -i eth1
```

Come già accennato Suricata analizza, classifica, e prende decisioni anche attive sui i pacchetti controllati in base ad una serie di regole di azione, all'installazione ne sono presenti alcune di default ma è possibile crearne delle proprie o personalizzare quelle esistenti, nel manuale d'uso sono indicate le sintassi apposite, come pure indicare in quale directory sono costudite nel caso non si vogliano utilizzare i path di default. Nelle prove da me effettuate comunque ho sempre utilizzato le regole di default.

Spesso nei parametri e nel file di configurazione "suricata.yaml" ricorre il tag:

"signature.rules" e/o "suricata.rules"

Questi tag corrispondono al "path" ed ai singoli nomi delle regole di analisi del traffico e di dove sono rilocate, esse, che tipicamente si trovano in: "/var/lib/suricata", possono essere

salvate in altra directory comunicandolo al programma o modificando come di consueto i parametri in suricata.yaml o aggiungendo fra i parametri a linea di comando mediante l'opzione "-s [path_regole].

Le regole di default di suricata sono nella directory "etc/suricata/rules.

Una volta avviato, il programma con le configurazioni di default effettuerà azioni di monitoring drop, etc. etc. riportando tutto nei log file fra cui uno dei più importanti è "eve.json" (event json file), è facile notare che questi file crescono di dimensione man mano che si fa traffico di rete, le azioni intraprese dipendono dalle regole predefinite o non di controllo dei pacchetti.

Si osserva che all'avvio possono comparire dei warning che si è riscontrato in genere sono dovuti da un errato o incompleto settaggio del file "suricata.yaml" che al più ne limitano alcune funzionalità ma il programma funziona ugualmente.

Per fare test sull'operato di suricata occorre tenerlo in funzione su una rete locale con un certo traffico visibile meglio sarebbe se si disponessero i permessi di poter accedere a dei veri e propri nodi principali dove c'è molto traffico, oppure è anche possibile fargli analizzare pacchetti di rete in off-line sotto forma di file "pcap" [11, 12, 18] che sono un formato tipico in cui molti sniffer comuni come ad esempi Wireshark possono salvare quanto intercettato.

3.2.1 Macchine AMD e modulo Hyperscan

Hyperscan è una tecnologia nata per i processori di Intel, utilizzata se disponibile, da Suricata per il rapido riconoscimento di espressioni regolari all'interno di un testo che consente di velocizzare in particolar modo in processi in streaming il riconoscimento dei contenuti [31]. Occorre prestare attenzione quando si installa Suricata su una macchina con processori AMD.

Fra le varie opzioni di installazione anche in base alla procedura utilizzata può capitare che il processo di installazione chieda di attivare il modulo "Hyperscan" [30] anche se in realtà non è disponibile perché questa tecnologia non è perfettamente compatibile con i processori AMD con cui può generare mal funzionamenti.

Solitamente è Suricata stesso che avverte del problema appena mandato in esecuzione stampando a shell diversi warning opportuni.

Qualora si sia attivato per errore l'uso di Hyperscan e ci si accorge di stranezze nel funzionamento di Suricata è comunque possibile rimediare modificando le impostazioni di avvio modificando opportunamente alcune voci del file "suricata.yaml" (vedi il prossimo paragrafo).

3.3 Suricata.yaml

Il formato YAML è un linguaggio di serializzazione dei dati che non appartiene alla categoria dei formati di markup ma vuol essere molto più simile come struttura al formato JSON, è spesso utilizzato per finalità di file di configurazione, messaggistica e altri scopi di organizzazione dei dati.

Suricata dispone di un file di configurazione principale abbastanza articolato e complesso: "suricata.yaml" che in genere è salvato nella directory:

"/etc/suricata/suricata.yaml".

Ci sono molte voci all'interno di questo file, facilmente modificabile con un comune text-editor, che consentono di personalizzare il funzionamento con ampio margine. Ad esempio, per l'eventuale disattivazione del modulo Hyperscan sopra citato si possono modificare come segue alcune voci:

mpm-algo: auto	➔ modificare in ..	#mpm-algo: auto mpm-algo: ac

spm-algo: auto	➔ modificare in ..	# spm-algo: auto spm-algo: bm

Affinché si ricevano correttamente gli allarmi da suricata e non solo i warning e gli alert è necessario configurare anche il parametro Home Network (HOME_NET) presente subito nelle prime righe del file di configurazione indicando quale è la rete a cui si desidera inviare i report che altrimenti verranno reindirizzati in modo errato e molti risulteranno persi.

Per i test da effettuare potremmo inserire l'IP:0.0.0.0/0 (equivalente all' impostazione HOME_NET: "any") in modo che si indichi a suricata di inviare gli allarmi a "tutti gli indirizzi IPV4 possibili sulla rete locale" nel mio caso però con questa impostazione seppur prevista Suricata restituisce un errore e si blocca. Quindi ho proceduto ad inserire esattamente la rete locale su cui si trova la macchina con cui sono stati effettuati i test.

```
## Step 1: inform Suricata about your network
##

vars:

  # more specifc is better for alert accuracy and performance
  address-groups:

    HOME_NET: "[192.168.0.0/16,10.0.0.0/8,172.16.0.0/12]"
    #HOME_NET: "[192.168.0.0/16]"
    #HOME_NET: "[10.0.0.0/8]"
    #HOME_NET: "[172.16.0.0/12]"
    #HOME_NET: "any"

    EXTERNAL_NET: "!$HOME_NET"
    #EXTERNAL_NET: "any"

……
…..
```

Modificato in:

```
#HOME_NET: "[192.168.0.0/16,10.0.0.0/8,172.16.0.0/12]"
 HOME_NET: "[131.114.0.0/16,192.168.1.0/16]"
#HOME_NET: "[192.168.0.0/16]"
#HOME_NET: "[10.0.0.0/8]"
#HOME_NET: "[172.16.0.0/12]"
#HOME_NET: "any"
```

Altre impostazioni di interesse da modificare/personalizzare per i nostri scopi le vedremo più avanti nella trattazione via via che se ne renderà necessario.

3.4 Suricata ed i files pcap

Come già accennato poco sopra Suricata è in grado di analizzare alche il traffico off-line facendogli scansionare dei file secondo lo standard “pcap” [11, 12, 18]. passati in input, qui sotto riportato sotto uno dei tanti link che affrontano l’argomento:

https://401trg.com/using-emergingthreats-suricata-ruleset-to-scan-pcap/

Ci sono dei siti web specifici che collezionalo o sintetizzano file “pcap” con registrazioni di traffico con contenuto malevole e non, particolarmente utili per fare testing del sistema e del funzionamento di eventuali regole di controllo che ci siamo definiti embedded, una sintassi “minima” standard a linea di comando che ho utilizzato per eseguire questo tipo di funzionalità e far leggere i file pcap al posto della cattura del traffico di rete online è la seguente:

```
“sudo suricata -r <path_to_pcap_file>/<pcap_file_name.pcap>”
```

Qui ancora una volta si nota la necessità dei privilegi “super user”, in quanto come già detto in precedenza per NTOPNG, anche Suricata che utilizza funzionalità di basso livello necessita delle necessarie autorizzazioni di sistema.

4 Syslog

Come già accennato Syslog (System Log) [14, 37] è un protocollo di rete appartenente alla suite di protocolli Internet utilizzato per trasmettere attraverso una rete semplici informazioni di log implementato in un processo che viene eseguito sottoforma di "demone di sistema".

Normalmente presente in Linux in una delle possibili implementazioni disponibili e dovrebbe essere attivo di default semmai dormiente, Suricata è in grado di interagirvi.

Cambiando distribuzione Linux può cambiare l'applicativo Syslog presente ma le varianti sono tipicamente solo cinque ed hanno funzionamenti simili e comunque a equivalenti nella parte base, generalmente si trova il programma e/o i suoi file di configurazione nella directory di sistema "/etc/init.d".

- **syslogd** - logs system messages [14]
- **syslog-ng** - logs system messages but also suports TCP, TLS, and other enhanced enterprise features [35]
- **rsyslogd** - logs system messages but also support TCP, TLS, multi-threading, and other enhanced features
- **klogd** - logs kernel messages
- **sysklogd** - basically a bundle of syslogd and klogd

Il protocollo SYSLOG a differenza di altri protocolli non ha specifiche rigidamente definite e solo in tempi recenti è stato in parte standardizzato dall'IETF [40].

Syslog usa tipicamente il protocollo UDP su porta 514, vedi RFC 5424 [14, 37], ma in caso di necessità può utilizzare TCP su porta 1468.

Il protocollo UDP è utilizzato nella maggior parte degli impieghi di logging eventi mentre in particolari applicazioni dove il monitoraggio è fondamentale oppure quando certi eventi possono innescare azioni da parte del server ed è necessario un miglior controllo della certezza dell' invio dei messaggi si ricorre ad implementazioni che usano il protocollo TCP eventualmente associati a metodi crittografici.

Generalmente un processo client che vuol sfruttare questo servizio invia un certo messaggio di testo, se si usa UDP è consigliabile non superare i 1024 caratteri, al processo server opportuno comunemente definito come "syslogd", "syslog daemon" o "syslog server" incaricato di smistarlo nella rete o eventualmente nella stessa macchina, ad esempio in localhost su 127.0.0.1.

La semplicità del protocollo fa sì che il processo server possa gestire messaggi provenienti da una variegata tipologia di macchine, da computer, stampanti, dispositivi di rete, macchinari, ecc. limitandosi a registrare l'evento, per avere un archivio centrale degli avvenimenti, oppure reagire a particolari livelli di severità chiamando programmi, inviando e-mail, o altro.

SYSLOG è particolarmente diffuso in Unix e conseguentemente sotto Linux, mentre non è molto diffuso in ambiente Windows dove non è un componente standard ma può essere installato a parte.

Per scoprire quale demone Syslog è disponibile nel nostro sistema si possono seguire i seguenti suggerimenti, tipicamente però su Ubuntu o suoi derivati è presente l'implementazione rsyslogd:

https://suricata.readthedocs.io/en/suricata-4.0.4/output/syslog-alerting-comp.html

dove in particolare il paragrafo 12.3.2 riporta:

"There are many ways to find out what syslog daemon you are using but here is one way:

```
cd /etc/init.d
ls | grep syslog
```

You should see a file with the word syslog in it, e.g. "syslog", "rsyslogd", etc. Obviously if the name is "rsyslogd" you can be fairly confident you are running rsyslogd. If unsure or the filename is just "syslog", take a look at that file. For example, if it was "rsyslogd", run:

```
less rsyslogd
```

At the top you should see a comment line that looks something like this:

```
# rsyslog        Starts rsyslogd/rklogd.
```

Locate those files and look at them to give you clues as to what syslog daemon you are running. Also look in the *start()* section of the file you ran "less" on and see what binaries get started because that can give you clues as well."

Per verificare se "Syslog" è già attivo fra i processi di sistema, sull'applicazione standard "monitor di sistema" può essere necessario attivare l'opzione "visualizza tutti i processi", infatti talvolta le impostazioni default mantengono nascosti gli applicativi di sistema a basso livello anche se sono in effetti attivi.

In alternativa si può usare da shell il comando "top" ma senza opportuni filtraggi stampa a video (fra cui " | grep..")il demone potrebbe rimanere fuori videata essendo in genere fra i servizi meno onerosi di risorse si troverà a fine lista e difficoltoso da visualizzare.

Nota:

Qualora Syslog fosse disattivo(attivo) per farlo riattivare(disattivare) vi si può interagire da shell con i comandi:

…> /etc/init.d ./rsyslog start

…> /etc/init.d ./rsyslog stop

Altre informazioni su rsyslogd si possono trovare a questo link:

https://www.rsyslog.com/doc/v8-stable/configuration/index.html

NTOPNG per ricevere i dati da suricata via Syslog è stato dotato di un servizio server collezionatore apposito da attivare su un'apposita porta di ascolto, typ. la 9999, molte prove di interfacciamento sono state dapprima effettuate localmente in loop-back locale sulla 127.0.0.1 dopo di che si è passati ad una comunicazione vera e propria fra macchine distinte.

4.1 Interfacciamento Syslog a Suricata

Suricata è in grado di interfacciarsi abbastanza facilmente a Syslog [33], come tante altre funzionalità si segnala a Suricata di utilizzarlo mediante opportune modifiche alle impostazioni del file "suricata.yaml". Partendo dal presupposto che il demone "rsyslog" è

uno dei più diffusi nelle distribuzioni Linux ed è anche quello presente nelle macchine con cui abbiamo effettuato i test e che in genere il file "suricata.yaml" si trova in "/etc/suricata/…" è necessario editarlo ed individuare e modificare alcune delle seguenti voci tenuto conto che con "#" all'inizio di una riga si commenta tale opzione rendendola inattiva.

Di particolare interesse per i nostri scopi è l'impostazione sulle informazioni sui flussi esportati da suricata, se si sta già elaborando traffico con NTOPNG (mirroring dello stesso traffico verso Suricata e NTOPNG), non è necessaria l'esportazione integrale delle informazioni dei poiché NTOPNG le sta già calcolando.

Volendo quindi utilizzare Suricata come sensore per esportare i metadati del flusso su NTOPNG è necessario abilitare l'esportazione delle informazioni "netflow" al posto di quelle "flow" nel file di configurazione suricata.yaml commentando e decommentando le opportune voci:

```
…..
- ssh
- stats:

      totals: yes       # stats for all threads merged together
      threads: no       # per thread stats
      deltas: no        # include delta values

    # bi-directional flows

    - flow                          "questa direttiva va commentata per ridurre i messaggi sui flussi mandati da suricata che è molto verboso. inizialmente si è osservato che la mancanza di info sui flussi non consentirà di avere alcune informazioni ma molte di esse sono già valutate da ntopng e sarebbero doppioni.

    # uni-directional flows

    #- netflow                      "questa andrà decommentata perché i messaggi netflow al contrario di quelli flow hanno info più utili per il collezionatore per ricevere da Syslog creato per ntong"

    #- dnp3

….etc etc…
```

```
…..
- ssh
- stats:

      totals: yes       # stats for all threads merged together
      threads: no       # per thread stats
      deltas: no        # include delta values

    # bi-directional flows

    # - flow                        .

    # uni-directional flows

    - netflow

    #- dnp3

….etc etc…
```

Continuando a “navigare” il testo del file, esiste quindi più di un punto in cui è necessario indicare di abilitare l’opzione Syslog per specifiche funzionalità che ci servono.

```
…..
 # a line based alerts log similar to fast.log into syslog
  - syslog:
      enabled: no
…….
```

```
…..
 # a line based alerts log similar to fast.log into syslog
  - syslog:
      #enabled: no
      enabled: yes
…….
```

```
…..
- syslog:
    enabled: no
      facility: local5
      format: "[%i] <%d> -- "
      # type: json …….
……..
```

```
…...
- syslog:
    #enabled: no
      enabled: yes
      facility: local5
      format: "[%i] <%d> -- "
      # type: json …….
……
```

Ed ancora altre parti d’interesse sono:

```
…..
# Extensible Event Format (nicknamed EVE) event log in JSON format
  - eve-log:
      enabled: yes
      #filetype: regular #regular|syslog|unix_dgram|unix_stream|redis
      filename: eve.json
      #prefix: "@cee: " # prefix to prepend to each log entry
      # the following are valid when type: syslog above
      identity: "suricata"
      facility: local5
      level: Info ## possible levels: Emergency, Alert, Critical,  <-qui metteremo alert
                  ## Error, Warning, Notice, Info, Debug
……..
```

```
…...
# Extensible Event Format (nicknamed EVE) event log in JSON format
  - eve-log:
      enabled: yes
      #filetype: regular #regular|syslog|unix_dgram|unix_stream|redis
      filetype: syslog #regular|syslog|unix_dgram|unix_stream|redis
      filename: eve.json
      #prefix: "@cee: " # prefix to prepend to each log entry
      # the following are valid when type: syslog above
      identity: "suricata"
      facility: local5
      level: Info ## possible levels: Emergency, Alert, Critical,  <-qui metteremo alert
                  ## Error, Warning, Notice, Info, Debug
……
```

Nota: se non si mette “filetype: syslog” ma si lascia regular, le funzionalità di logging di suricata potrebbero non essere correttamente inviate a Syslog e potrebbe non partire il traffico fra suricata e rsyslogd sulla porta 514

4.1.1 Note sull'interazione suricata-Syslog

Inizialmente le prime prove di interazione fra Suricata e Syslog sono state fatte modificando le configurazioni seguendo i suggerimenti del sito ufficiale di Suricata:

https://suricata.readthedocs.io/en/suricata-4.1.2/output/syslog-alerting-comp.html#example

Effettuate quindi le modifiche cui sopra facendo ripartire sia il demone che suricata e monitorizzando con Wireshark [39] si nota subito molta attività sulla porta su 127.0.0.1:514, questo ha consentito di catturare gli effettivi messaggi inviati da suricata sul traffico intercettato che sono stati particolarmente utili per capirne l'effettiva struttura.

Subito dopo, appena verificato che in effetti c'era scambio di messaggi ci si è concentrati su come e in che formato reindirizzarli a NTOPNG.

4.2 Interfacciare Syslog a NTOPNG

Anche SYSLOG ha un proprio file di configurazione su cui dovremo fare alcune modifiche si trova in generalmente dentro "/etc/" e in genere si chiama "rsyslog.conf or syslog.conf" (/etc/rsyslog.conf), per far dialogare Syslog con NTOPNG occorre quindi fare alcuni aggiustamenti ai parametri di default anche in questo file si rimanda i prossimi paragrafi 4.2.1 e 4.2.2 per i dettagli.

4.2.1 Note sull'interazione suricata-Syslog-NTOPNG via UDP

Il formato di interscambio fra Suricata e Syslog inizialmente è stato provato a gestire mediante formattazione dei dati attraverso opportuni template secondo le regole descritte dal protocollo stesso, ad esempio nel riquadro sono riportare alcuni dettagli utilizzati durante i test del contenuto del file "rsyslog.conf" e di seguito si riportano alcune note riscontrate durante i lavori eseguiti:

```
$template sysklogd, "<%PRI%>%syslogtag:1:32%%msg:::sp-if-no-1st-sp%%msg%"
```

```
#questo template non invia alcuna info utile ai nostri scopi:
$template TraditionalFormat,"%timegenerated% %HOSTNAME%
%syslogtag%%msg:::drop-last-lf%\n"

#
# Where to place spool and state files
#

$WorkDirectory /var/spool/rsyslog

#
# Include all config files in /etc/rsyslog.d/
#

$IncludeConfig /etc/rsyslog.d/*.conf

#alcune opzioni per il tipo di messaggio da mandare sono  (*.* = tutti di ogni tipo) e ache
server @...:port; ed eventuale template da inviare

#user.alert @127.0.0.1:514;sysklogd
#user.* @127.0.0.1:514;sysklogd
#*.* @127.0.0.1:514;sysklogd
#*.* @127.0.0.1:514;TraditionalFormat

*.* @127.0.0.1:514;sysklogd
```

Esistono delle apposite regole per poter personalizzare i vari template corrispondenti ai messaggi inviabili da Syslog secondo quanto riceve da suricata.

Si può dichiarare un template ed assegnargli un nome con la seguente sintassi:

```
$template <nome del template>, <…regole formattazione del messaggio di template .>
```

Dopo di che è possibile con sintassi apposite come quelle riportate nel riquadro seguente fare una sorta di pre-scelta della tipologia di messaggi che interessano da reindirizzare così composte: "scelta del tipo messaggio (user.alert, *.alert, etc.) seguito con il simbolo "@<indrizzo:porta>" facendo anche notare che è possibile specificare una porta

di uscita diversa dalla 514 di default ";" <nome template di formattazione del messaggio di uscita>"

```
#user.alert @127.0.0.1:514;sysklogd
#user.* @127.0.0.1:514;sysklogd
#*.* @127.0.0.1:9999;sysklogd
#*.* @127.0.0.1:514;TraditionalFormat
*.* @127.0.0.1:514;sysklogd
```

Notare che il simbolo "*.*" indica una sintassi del tipo "tipo_processo.informazione" per cui associare "*.*" indica di ritrasmettere tutte le informazioni di tutte le tipologie di processo i processi che richiedono il logging via Syslog di ogni tipo

Si possono però ad esempio con "*.alert" limitare i messaggi della soltanto della tipologia alert, oppure, oppure se si è associato un processo al servizio "LOCAL5" (opzione possibile anche per suricata vedi suricata.yaml) con "local5.*" → syslog invierà solo i messaggi di chi si è registrato verso Syslog come servizio "local 5" e così via, alcune informazioni su come strutturare i template possono essere trovate a questi link:

```
https://www.rsyslog.com/doc/v8-stable/configuration/templates.html
https://www.rsyslog.com/doc/master/configuration/templates.html
```

Una volta messa a punto la comunicazione Suricata-Syslog monitorizzando il traffico sulla porta 514 loacalhost con Wireshark [39] si ottengono adesso messaggi molto interessanti, in formato JSON, come quelli riportati nel riquadro successivo, con informazioni sui flussi dei pacchetti, alert, e vari dati del traffico nonché un codice particolare "FLOW_ID" che ci tornerà utile sia per riconoscere i pacchetti di una stessa comunicazione che insieme ad altri dati classici fra cui "indirizzo IP e porta - destinatario/mittente", che per calcolare il codice univoco COMMUNITY_ID menzionato nelle premesse ad introduzione necessario per mettere in correlazione il traffico monitorizzato da Suricata con quello monitorizzato da NTOPNG.

Esempio di messaggi di flusso standard di tipo "flow" replicati inviati da suricata e replicati sa Syslog:

Mar 29 16:16:23 pc-deri suricata[32748]: {"timestamp": "2019-03-29T16:16:23.000591+0100", "flow_id": 868314455276614, "event_type": "flow", "src_ip": "192.12.193.65", "src_port": 5353, "dest_ip": "224.0.0.251", "dest_port": 5353, "proto": "UDP", "app_proto": "failed", "flow": {"pkts_toserver": 1, "pkts_toclient": 0, "bytes_toserver": 117, "bytes_toclient": 0, "start": "2019-03-29T16:15:52.523334+0100", "end": "2019-03-29T16:15:52.523334+0100", "age": 0, "state": "new", "reason": "timeout", "alerted": false}}

203 30.179207 131.114.134.54 131.114.134.51 Syslog 562 LOCAL5.EMERG: marcolnxApr 1 15:25:12suricata[9563]: {"timestamp": "2019-04-01T15:25:12.000303+0200", "flow_id": 469652813216956, "event_type": "flow", "src_ip": "131.114.132.165", "src_port": 138, "dest_ip": "131.114.133.255", "dest_port": 138, "proto": "UDP", "app_proto": "failed", "flow": {"pkts_toserver": 1, "pkts_toclient": 0, "bytes_toserver": 243, "bytes_toclient": 0, "start": "2019-04-01T15:24:41.687292+0200", "end": "2019-04-01T15:24:41.687292+0200", "age": 0, "state": "new", "reason": "timeout", "alerted": false}}

Esempio di messaggi di flusso malevole replicati inviati da suricata e replicati sa Syslog:

90 118.884451 131.114.134.54 131.114.134.51 Syslog 240 LOCAL5.ERR: marcolnxApr 1 15:50:30suricata[10349]: [1:2025637:3] ET TROJAN Remcos RAT Checkin 23 [Classification: A Network Trojan was detected] [Priority: 1] {TCP} 10.0.90.215:49205 -> 103.1.184.108:2404ù

101 118.936655 131.114.134.54 131.114.134.51 Syslog 980 LOCAL5.EMERG: marcolnxApr 1 15:50:30suricata[10349]: {"timestamp": "2019-03-19T02:53:55.841544+0100", "flow_id": 507765172211669, "pcap_cnt": 1795, "event_type": "alert", "src_ip": "10.0.90.215", "src_port": 49205, "dest_ip": "103.1.184.108", "dest_port": 2404, "proto": "TCP", "alert": {"action": "allowed", "gid": 1, "signature_id": 2025637, "rev": 3, "signature": "ET TROJAN Remcos RAT Checkin 23", "category": "A Network Trojan was detected", "severity": 1, "metadata": {"updated_at": ["2019_03_22"], "malware_family": ["Remcos"], "created_at": ["2018_07_03"], "signature_severity": ["Major"], "tag": ["RAT"], "deployment": ["Perimeter"], "attack_target": ["Client_Endpoint"], "affected_product": ["Windows_XP_Vista_7_8_10_Server_32_64_Bit"], "former_category": ["TROJAN"]}}, "app_proto": "failed", "flow": {"pkts_toserver": 30, "pkts_toclient": 54, "bytes_toserver": 4366, "bytes_toclient": 3730, "start": "2019-03-19T02:49:44.659413+0100"}}

23 114.780281 131.114.134.54 131.114.134.51 Syslog 328 LOCAL5.WARNING: marcolnxApr 1 15:50:26suricata[10349]: {"timestamp":"2019-04-01T15:50:26.501915+0200","log_level":"Warning","event_type":"engine","engine":{"error_code":306,"error":"SC_WARN_FLOWBIT","message":"flowbit 'ET.wininet.UA' is checked but not set. Checked in 2021312 and 0 other sigs"}}

1263 121.743133 131.114.134.54 131.114.134.51 Syslog 260 LOCAL5.NOTICE: marcolnxApr 1 15:50:33suricata[10349]: {"timestamp":"2019-04-01T15:50:33.458233+0200", "log_level":"Notice", "event_type":"engine","engine":{"message":"Pcap-file module read 1 files, 13186 packets, 7563934 bytes"}}

Fra le etichette json specifiche “chiave-valore” si osservi particolare attenzione ai campi “event_type” che danno informazioni sulla tipologia del messaggio contenuto e nel messaggio del tipo: "event_type": "alert" ed il sotto campo “metadata” che contiene molte informazioni aggiuntive specifiche dell’alert con alcune parti standard documentate dagli sviluppatori di Suricata ed altre che possono variare anche molto in base al dettaglio dello specifico evento segnalato nel messaggio.

Inoltre, nel preambolo del messaggio si noti anche l’informazione del tipo “tipo_processo.informazione” da cui “*.*” già accennata in precedenza e nel caso dei messaggi del riquadro si nota che sono stato inviati del servizio LOCAL5 (servizio a cui per fare i test era stato da noi registrato Suricata) con specificato dopo il punto la tipologia del messaggio che come si nota può essere:

Local5.NOTICE..

Local5.ERR….

Local5.WARNING…

Local5.EMERG

Tuttavia, però, sebbene inizialmente si sia tentato di cominciare ad instradare i messaggi Syslog verso NTOPNG gestendoli con traffico UDP sulla porta 514 molto velocemente si è arrivati alla conclusione che c'è poco spazio di modifica sui parametri dei template di Syslog per aggiungere informazioni utili ai nostri scopi rispetto a quelle che Suricata fornisce nel complesso.

Altro aspetto non secondario spesso i dati in arrivo da suricata superavano i 1024 byte e canalizzandoli con UDP se la nostra rete adotta la dimensione standard dei pacchetti da 1500 byte (MTU = 1500) si rischia di perderne delle parti o riceverli disallineati per la caratteristica intrinseca del protocollo stesso.

Infine, si è notato che il servizio Syslog su UDP tende comunque a tagliare messaggi più grandi di 2048 byte, quindi velocemente sapendo che Syslog può inviare messaggi anche in formato TCP si è passato a riconfigurarlo appositamente cambiando un poco l'idea iniziale di funzionamento che si è pensato d'interazione con NTOPNG

4.2.2 Note sull'interazione suricata-Syslog-NTOPNG via TCP

Come accennato nel paragrafo precedente a questo punto si è iniziato a modificare il codice di NTOPNG per implementare un apposito collezionatore TCP che riceve i messaggi inviati da Suricata via Syslog non più riorganizzati tramite i template ma direttamente come stream TCP [15].

Quindi, a seguito delle implementazioni e modifiche software realizzate per operare nella nuova modalità di comunicazione Ntop-Syslog, occorre attivare un'opportuna "interfaccia di ascolto" di NTOPNG sulla porta 9999, vedi il parametro "-i", realizzata opportunamente per Syslog, specificando ovviamente l'indirizzo della macchina da cui si

riceveranno i dati replicati da Suricata che può essere anche la stessa e quindi in tal caso l'IP di ascolto sarà il classico localhost 127.0.0.1.

Conseguentemente anche il file di configurazione di Syslog va leggermente modificato e per farlo si possono seguire due strade:

1) creare un nuovo file (es: cat /etc/rsyslog.d/99-remote.conf) di configurazione apposito e segnalare al processo di caricarlo al posto di "rsyslog.conf" ad esempio chiamandolo "99-remote.conf" con all'interno indicate direttive come la seguente:

```
*.* action(type="omfwd" target="192.168.2.222" port="9999" protocol="tcp"
action.resumeRetryCount="100" queue.type="linkedList" queue.size="10000")
```

Dettagli e note della sintassi e dei parametri qui sopra utilizzati la si può trovare ai seguenti link:

```
https://www.rsyslog.com/doc/v8-stable/configuration/actions.html
https://www.rsyslog.com/doc/v8-stable/rainerscript/queue_parameters.html
```

2) In modo simile alla traccia precedente modificare, rispetto all'esempio precedente, dentro "rsyslog.conf" alcuni parametri.

"*.* @127.0.0.1:9999;sysklogd"

in

. @@(o)127.0.0.1:9999;sysklogd

Il prefisso "@@(o)" indica adesso a Syslog che i messaggi vengono trasmessi in TCP e non più in UDP e sysklogd continua ad essere il template che specifica la struttura/sintassi con cui i messaggi vengono inviati ad esempio:

$template sysklogd, "<%PRI%> %HOSTNAME% %timegenerated% %syslogtag:1:32% %msg:::sp-if-no-1st-sp% %msg%"

dove il tag "%msg%" è relativo all'effettivo messaggio con i contenuti di suricata e tutte le altre parti che compongono il template fanno parte

dell’intestazione a cui possiamo aggiungere alcune informazioni su da quale macchina è partito “%HOSTNAME%” in quale momento “%timegenerated%”, e altro.

5 Dettagli sulla struttura dei messaggi JSON di Suricata

Oltre quanto già riportato nel capitolo 4, elenchiamo adesso alcuni dettagli sulle informazioni che si possono ottenere dai messaggi inviati da Suricata che sono tipicamente ordinati secondo la tipologia protocollo, software o messaggi di sistema che hanno scatenato il warning o l'errore limitandoci alle tipologie più comuni che sono segnalate nel campo "event_type":"<tipo di evento scatenante>" che possono essere:

flow
stats
tls
dns
alert
netflow
engine
http
……
e altri

Gli event-type a cui si è prestato più interesse all'inizio sono quelli di tipo "netflow" ed "alert", poi come sviluppo successivo il software di NTOPNG è stato esteso per riconoscere e gestire tutti gli altri eventi segnalati sa Suricata.

I gruppi "netflow" contengono informazioni sul flusso dei pacchetti analizzati e sono più specifici di quelli di tipo "flow" che possono contenere informazioni molto più generiche e di conseguenza possono risultare in mole anche troppo frequenti e in parte già valutate da ntop stesso.

I due gruppi di messaggi normalmente è bene non farli generare in contemporanea si avrebbero potenziali duplicazioni delle informazioni, è quindi necessario disattivare i messaggi flow e lasciare attivi quelli netflow commentando i primi e de-commentando i secondi nell'apposita sezione di configurazione del file "suricata.yaml".

```
# bi-directional flows
#- flow
```

```
# uni-directional flows
- netflow
```

Fra i campi json ricevuti, come già accennato c'è sempre una prima parte generale contenente una serie di informazioni uguali per ogni tipo di evento ricevuto e relativa al flusso stesso che saranno utilizzati per riconoscere a quale flusso appartiene il singolo pacchetto ricevuto affinché sia possibile fare un poco di statistica mediante opportuni contatori.


```
…{"timestamp": "2019-03-29T16:16:23.000591+0100", "flow_id": 868314455276614, "event_type": "flow", "src_ip": "192.12.193.65",
"src_port": 5353, "dest_ip": "224.0.0.251", "dest_port": 5353, "proto": "UDP", "app_proto": "failed",…
```


Di particolare interesse sono gli eventi di tipo "alert" che contengono al loro interno informazioni di anomalie rilevate sulla rete dovuti anche ad attacchi di vario genere, intrusione, trojan, malware, etc. fra cui la gravità dell'attacco "severity", il tipo di attacco "category", l'azione intrapresa in base alla regola che è stata violata "action", che può essere consentire il transito del messaggio ed avvertire soltanto "allow" oppure bloccarlo ed evitarne la diffusione "blocked".

È poi normalmente presente nei messaggi di alert una sottosezione con chiave "metadata" che contiene informazioni di dettaglio con parti non standard o quantomeno variabili a seconda del tipo di evento scatenante con cui Suricata cerca di informare, e in cui se ci sono dettagli a sufficienza per determinarlo, suricata potrebbe anche fornire già il dato a riguardo di chi è stato l'attaccante:

```
"alert": {
  "action": "allowed" or "blocked"
  "gid": 1,
  "signature_id": 1,
  "rev": 1,
  "app_proto": "http",
  "signature": "HTTP body talking about corruption",
  "severity": 3,
  "category" : " descrizione testuale del tipo di alert inviato"
  "metadata": {
                    ...informazioni aggiuntive sull'evento .. specifiche di un troian
                       linee http sospette, etc.
                }
   //i campi source e target sono previsti ma molto rari nei messaggi
  "source": {
```

```
                "ip": "192.168.43.32",
                "port": 36292
              },
 "target": {
                "ip": "179.60.192.3",
                "port": 80
              },
```

(per alcuni dettagli è possibile consultare anche [42])

Fra i campi cui sopra possiamo dettagliare che:

"action"

indica quali decisioni Suricata ha preso sull'alert che ci sta inviando

"gid, signature_id,rev"

Sono 3 indici collegati in qualche modo fra loro e si propongono di fare una sorta di catalogazione relativa allo stesso suricata

"signature, category"

Sono in qualche modo collegati fra loro e forniscono dei messaggi sulla tipologia del problema.

"severity"

È un indice numerico che segnala la gravità del problema, più è alto il valore meno grave è il problema, sebbene suricata segnali che questo indice possa assumere valori da 1 a 255 all'atto pratico ad oggi analizzando sia i messaggi ricevuti che un po' il codice nativo di suricata si riscontra che vengono utilizzati solo valori da 1 a 4.

Nel riquadro seguente è riportato, come esempio, un blocco di definizioni estratto da del codice in cui si possono osservare alcune associazioni fra valore e la tipologia di evento.

https://github.com/OISF/suricata/blob/master/classification.config

```
//https://github.com/OISF/suricata/blob/master/classification.config
/*
 #
# config classification:shortname,short description,priority
```

```
#
config classification: not-suspicious,Not Suspicious Traffic,3
config classification: unknown,Unknown Traffic,3
config classification: bad-unknown,Potentially Bad Traffic, 2
config classification: attempted-recon,Attempted Information Leak,2
config classification: successful-recon-limited,Information Leak,2
config classification: successful-recon-largescale,Large Scale Information Leak,2
config classification: attempted-dos,Attempted Denial of Service,2
config classification: successful-dos,Denial of Service,2
config classification: attempted-user,Attempted User Privilege Gain,1
config classification: unsuccessful-user,Unsuccessful User Privilege Gain,1
config classification: successful-user,Successful User Privilege Gain,1
config classification: attempted-admin,Attempted Administrator Privilege Gain,1
config classification: successful-admin,Successful Administrator Privilege Gain,1

# NEW CLASSIFICATIONS
config classification: rpc-portmap-decode,Decode of an RPC Query,2
config classification: shellcode-detect,Executable code was detected,1
config classification: string-detect,A suspicious string was detected,3
config classification: suspicious-filename-detect,A suspicious filename was detected,2
config classification: suspicious-login,An attempted login using a suspicious username was detected,2
config classification: system-call-detect,A system call was detected,2
config classification: tcp-connection,A TCP connection was detected,4
config classification: trojan-activity,A Network Trojan was detected, 1
config classification: unusual-client-port-connection,A client was using an unusual port,2
config classification: network-scan,Detection of a Network Scan,3
config classification: denial-of-service,Detection of a Denial of Service Attack,2
config classification: non-standard-protocol,Detection of a non-standard protocol or event,2
config classification: protocol-command-decode,Generic Protocol Command Decode,3
config classification: web-application-activity,access to a potentially vulnerable web application,2
config classification: web-application-attack,Web Application Attack,1
config classification: misc-activity,Misc activity,3
config classification: misc-attack,Misc Attack,2
config classification: icmp-event,Generic ICMP event,3
config classification: kickass-porn,SCORE! Get the lotion!,1
config classification: policy-violation,Potential Corporate Privacy Violation,1
config classification: default-login-attempt,Attempt to login by a default username and password,2

# Update
config classification: targeted-activity,Targeted Malicious Activity was Detected,1
config classification: exploit-kit,Exploit Kit Activity Detected,1
config classification: external-ip-check,Device Retrieving External IP Address Detected,2
config classification: domain-c2,Domain Observed Used for C2 Detected,1
config classification: pup-activity,Possibly Unwanted Program Detected,2
config classification: credential-theft,Successful Credential Theft Detected,1
config classification: social-engineering,Possible Social Engineering Attempted,2
config classification: coin-mining,Crypto Currency Mining Activity Detected,2

 */
```

"metadata" [44]

Questo blocco contiene informazioni di dettaglio sull'evento dell'alert.

Talvolta è presente un campo particolare molto interessante che si chiama "attack_target" che purtroppo non è sempre presente mentre sarebbe invece molto utile come dato, in special modo c'è quando gli alert sono relativi a malware e che specifica da chi è partito l'attacco stesso, purtroppo questa informazione non è sempre presente in tutti i messaggi.

Un esempio di contenuto del campo metadata è il seguente:

"metadata": { "updated_at": ["2019_03_22"], "malware_family": ["Remcos"], "created_at": ["2018_07_03"], "signature_severity": ["Major"], "tag": ["RAT"], "deployment": ["Perimeter"], "attack_target": ["Client_Endpoint"], "affected_product": ["Windows_XP_Vista_7_8_10_Server_32_64_Bit"], "former_category": ["TROJAN"] }…..

“source & target”

Parametri previsti nella documentazione di Suricata ma di fatto quasi mai presenti nel messaggio. Potrebbero sembrare un duplicato delle informazioni generali in intestazione del messaggio ma che in realtà sono specifici dell’alert su chi è l’attaccato e l’attaccante formazioni di dettaglio sull’evento dell’alert.

Oltre alla “sezione metadata” talvolta possono trovarsi nel messaggio altre sezioni di dettaglio specifiche molto simili contenenti dettagli aggiuntivi per protocolli noti o virus noti come ad esempio per http:

…..""http"": {… ""http_user_agent"": … ,… , ""http_content_type"":…, ...}….

Possibili informazioni che possono essere contenute nelle eventuali sezioni json di alert o di altra tipologia di evento tipo problemi sul DNS, sul traffico HTTPS, TLS contengono quantomeno dettagli specifici che a titolo di esempio potrebbero essere catalogate con strutture (in codice C/C++) di questo tipo:

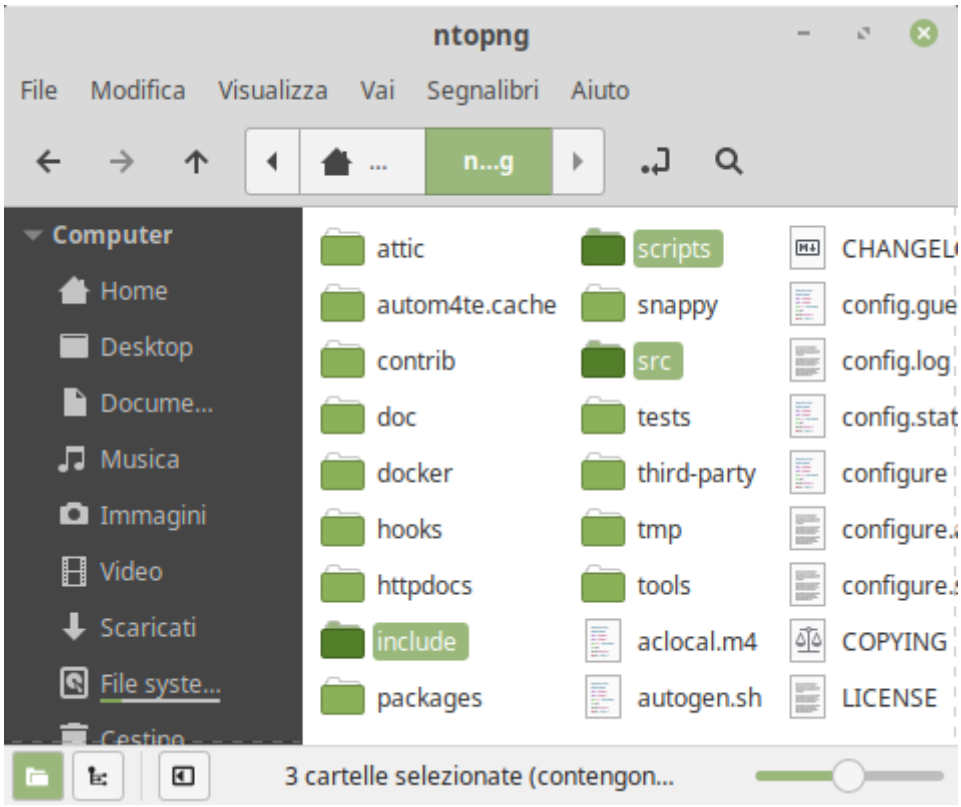
```
typedef struct suricataAlert {
        const char *suricata_FlowId;
        const char *community_id;
        AlertSuricataAction action;
        int32_t gid;
        int32_t signature_id;
        int32_t rev;
        const char *app_proto;
        const char *signature;
        int32_t severity
        const char *category;
        json_object *Metadata;
        attackTargetSuricataAlert attack_target;
```

```
   attackSignatureSeverity  signature_severity;
        IpAddress src_ip, trg_ip;
        u_int16_t src_port, trg_port;
} Parsed_SuricataAlert;


typedef struct suricataDNSevt{
        const char *Type;
        int      ID;
        const char *Rcode;
        const char *Rrname;
        const char *Rrtype;
        int      TTL;
        const char *Rdata;
        int      TxID;
} Parsed_suricataDNSevt;

typedef struct suricataHTTPEvt{
        const char *Hostname;
        const char *URL;
        const char *HTTPUserAgent;
        const char *HTTPContentType;
        const char *HTTPMethod;
        const char *Protocol;
        int Status;
        int      Length;
} Parsed_suricataHTTPEvt;

typedef struct suricataFfileinfoEvt {
        const char *Filename;
        const char *Magic;
        const char *State;
        const char *Md5;
        bool Stored;
        int Size;
        int TxID;
} Parsed_suricataFfileinfoEvt;

typedef struct suricataTLSEvt {
        const char *Subject;
        const char *Issuerdn;
        const char *Fingerprint;
        const char *Sni;
        const char *  Version ;
} Parsed_suricataTLSEvt;
```

6 Cenni sul codice NTOPNG sviluppato per l'integrazione con Suricata

Per l'integrazione delle nuove funzionalità insieme al team di sviluppo di NTOPNG sono state sia riorganizzate parti di codice esistenti che implementati alcuni nuovi file soprattutto per la gestione del parsing dei messaggi in arrivo da Suricata e la relativa messa in correlazione con il traffico di rete da esso monitorizzato elenchiamo qui con un piccolo dettaglio quelli che hanno funzionalità che si riflettono sulla strategia di integrazione ntop-suricata.

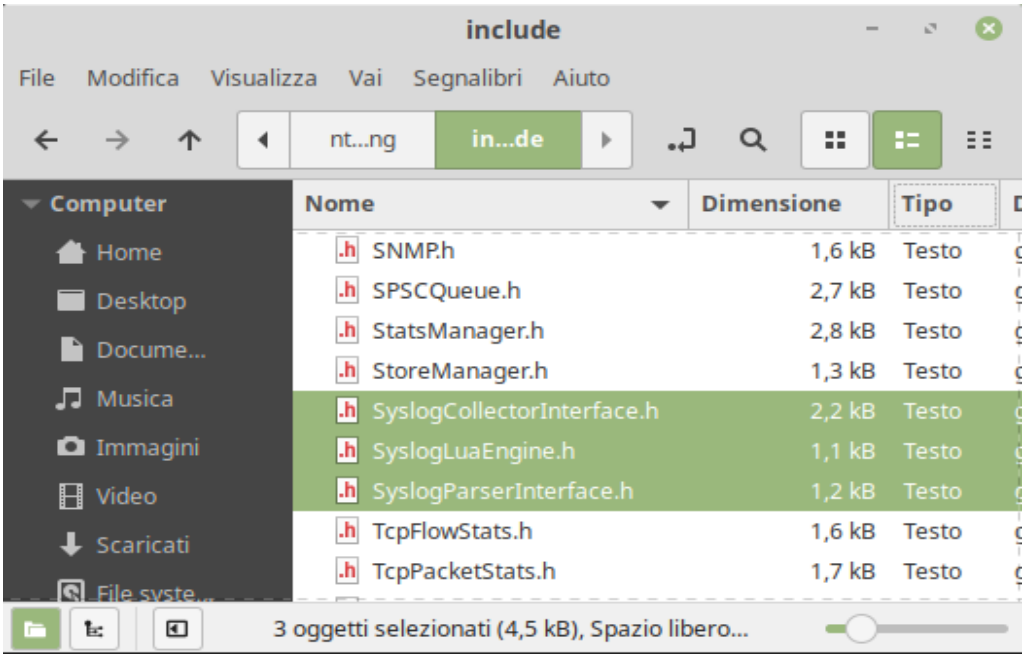
Nella struttura della distribuzione [8] del codice le directory di interesse per il codice Syslog sono: include, src, e scripts:

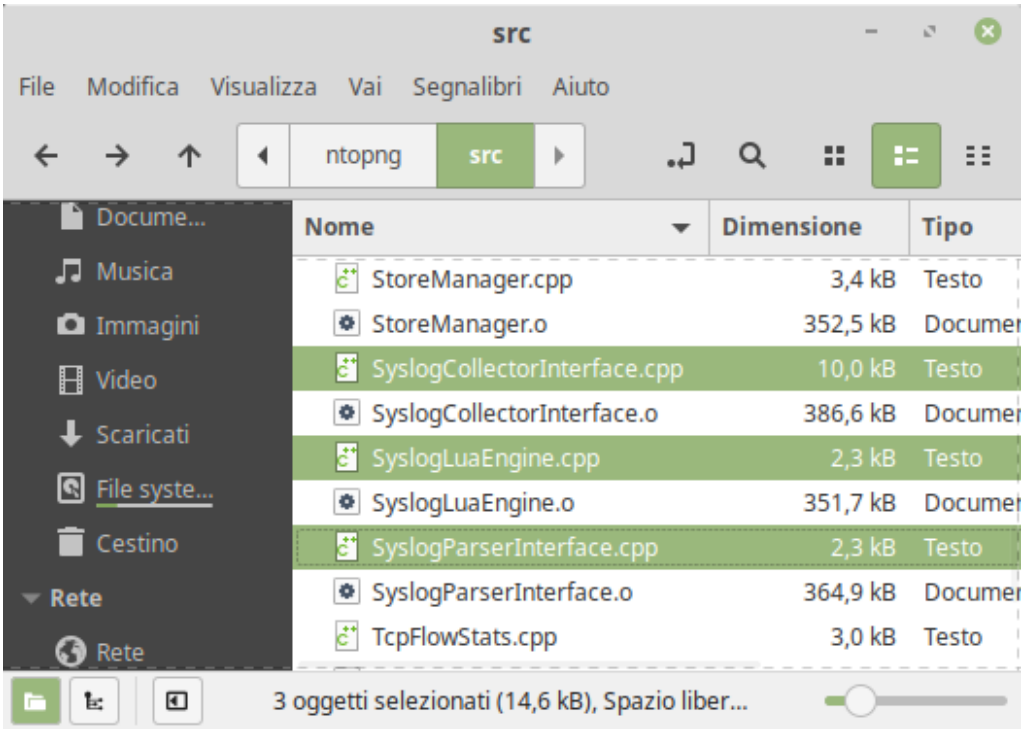


occorre comunque considerare che questo è un applicativo in costante sviluppo e che quindi è normale che in futuro questi sorgenti possano subire ulteriori modifiche ed aggiornamenti, è disponibile sul sito un repository con la documentazione di supporto per gli sviluppatori [34].

I pacchetti dati in transito sulla rete monitorati da ntop genericamente vengono identificati in flussi e per ogni flusso le informazioni relative vengono pre-elaborate e parzialmente conservate in oggetti di tipo “flow” che hanno una struttura abbastanza complessa (vedi Flow.h/.cpp).

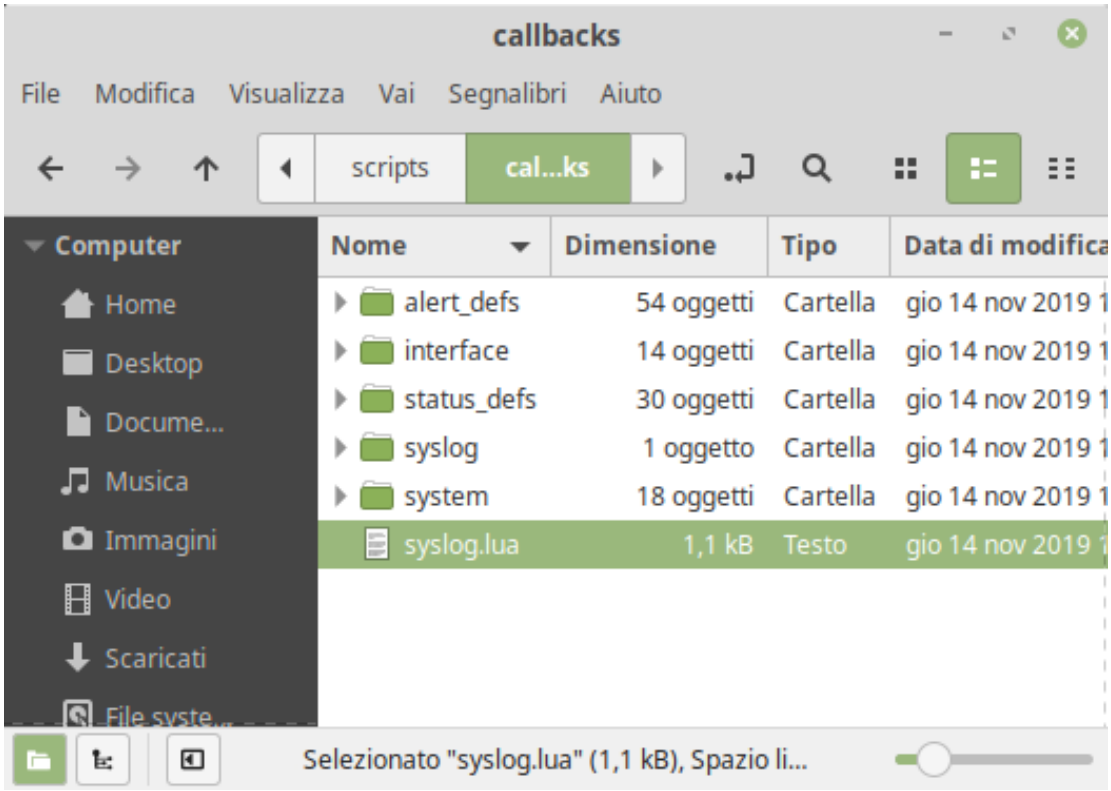
Questi “flussi” genericamente vengono ricevuti con un collezionatore e un “parser” apposito che ne preconfezionano le informazioni e ne fanno una pre-analisi quindi parallelamente, come già affermato in precedenza, sono quindi stati realizzati un apposito collezionatore e parser anche per ricevere da suricata via Syslog (vedi: SyslogCollectorInterface.h/.cpp, SyslogParserInterface.h/.cpp, SyslogLuaEngine.h/.cpp) in modo parallelo a quelli già esistenti.

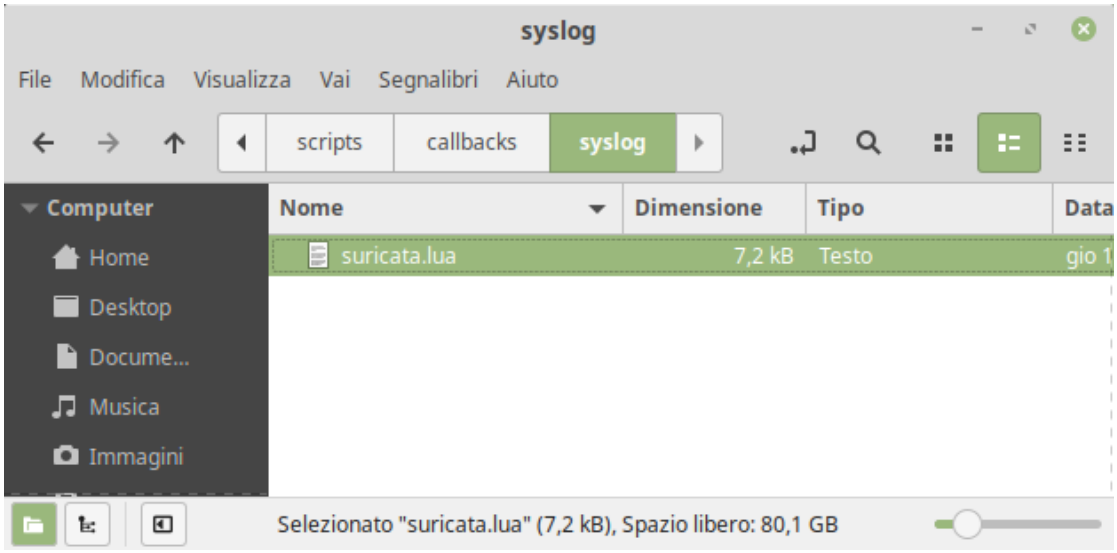




In SyslogCollectorInterface.cpp/SyslogParserInterface.cpp il pacchetto ricevuto viene parzialmente elaborato con il suo corrispettivo oggetto “flow” lasciando cmq una parte del messaggio “agganciato” al flusso stesso nel suo formato json nativo per essere utilizzato nelle fasi successive dove vi verrà estratto un po' di messaggistica fra quelle fornite nei campi json accessori di Suricata come ad esempio dal blocco “metadata” (vedi capitolo 5) e fare un po’ di statistica sugli eventi interessanti che via via si presentino. Il file SyslogLuaEngine.cpp predispone i dati ricevuti da Syslog per essere poi inviati agli script LUA preposti alla gestione grafica della “dashboard” di interazione.

Il flusso dei messaggi segue poi delle post elaborazioni fino alla GUI che in vari modi ne presenta i report elaborati intersecandole con la normale grafica del software NTOPNG che è implementata in linguaggio LUA. Grafica presentata sottoforma di pagina web in modo che gli utenti “Network-Supervisor” autorizzati possano interagirvi sia localmente che da remoto con un normale browser (vedi: scripts/callbacks/syslog.lua, scripts/callbacks/syslog/suricata.lua).





Dal punto di vista operativo, infine, tutte le parti attive del software preposte alla ricezione ed alla gestione dei messaggi ricevuti si aggiungono a quelle normalmente processate dal core di ntopng che è basato su Thread-Pull di task cooperanti.

6.1 Note su alcuni file sorgente di interesse

I file sorgenti di NTOPNG e tutto il supporto per lo sviluppo (development) si può trovare ai link riportati sul sito ufficiale oppure in via diretta agli indirizzi web indicati al punto [2] della bibliografia.

Riportiamo qui alcune note sui file sorgenti di diretto interesse all'integrazione effettuata tenuto conto che il software è molto complesso e fortemente ottimizzato quindi oltre ai moduli specifici ci sono molte interazioni indirette anche con altri file di codice non espliciti e per questo lavoro si è reso necessario un diretto coinvolgimento anche del team ufficiale di sviluppo:

SyslogCollectorInterface.h/.cpp, SyslogParserInterface.h/.cpp

> Contiene il codice che fa una prima analisi dei messaggi eseguito run-time mentre lo si riceve riconoscendo le tipologie di interesse ed estraendo i dati generali del pacchetto per associarlo al flusso che l'ha generato.

SyslogLuaEngine.h/.cpp

> Contiene il codice per attivare gli appositi script Lua quando si presentano eventi in arrivo da Syslog da inviare alla GUI

Flow.h/.cpp

> Questi file contengono le classi specifiche di uso globale per tutti i pacchetti di dati processati da NTOPNG che vengono catalogati in "flussi" come oggetti ottenuti da istanze di una classe "Flow" che ne cataloga tutte le caratteristiche e le peculiarità.

ZMQCollectorInterface.h/.cpp, ZMQParserInterface.h/.cpp

> Contengono il codice relativo al collezionatore per la ricezione dei conteggi del traffico di rete in arrivo dai vari probe installati nella rete da monitorare.

SyslogCollectorInterface.h, SyslogParserInterface.h

> Contengono il codice relativo al collezionatore specifico per ricevere messaggi via Syslog da Suricata, ha molti parallelismi con "ZMQ collector" in quanto ne riflette alcune funzionalità opportunamente specializzate ad OK.

ntop_typedefs.h

contiene un insieme di strutture e definizioni di tipo necessarie al “parsing” dei messaggi ricevuti nei vari protocolli di rete.

scripts/callbacks/syslog.lua

Script generic LUA per processare i messaggi da Syslog nell’ottica di uno sviluppo futuro tale che Syslog possa essere utilizzato anche per ricevere messaggi da altri applicativi oltre a suricata.

scripts/callbacks/syslog/suricata.lua

Script LUA specializzati per la gestione degli eventi ricevuti da Suricata via Syslog e come detto in precedenza come sviluppo questi script sono già stati pensati per eventuali altri moduli atti a integrare altri applicativi nelle funzionalità di ntopng.

6.2 Link GitHub codice NTOPNG e Suricata

- https://github.com/ntop
- https://github.com/ntop/ntopng
- https://github.com/ntop/nDPI
- https://github.com/ntop/PF_RING
- https://github.com/ntop/nProbe

- https://github.com/OISF/suricata

7 Avvio del sistema dei 3 programmi cooperanti

Effettuate quindi le modifiche indicate ai vari file di configurazione, compilato ed installato sia NTOPNG che Suricata può essere necessario occorre procedere con il riavvio del Syslog in modo che vengano ricaricate le nuove impostazioni [15]:

```
systemctl restart rsyslog
```

Dopo di che si può procedere con far partire Suricata specificando o verificando che nel file di configurazione suricata.yaml se non già fatto in precedenza siano presenti almeno le seguenti impostazioni nella sezione che indica le proprietà di esportazione flussi e alert su syslog.

```
- eve-log:
enabled: yes
filetype: syslog
```

Alla fine di queste operazioni è anche possibile, se lo si desidera, verificare l'attività dei processi che ci interessano sulla porta 9999 con NETSTAT digitando da shell:

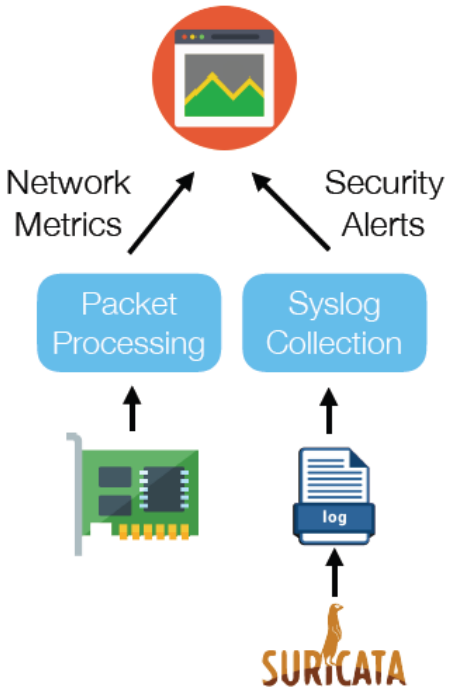
```
sudo netstat -tulpn | grep --color :9999
```

da cui potremmo avere dei report di questo tipo:

```
tcp 0 0 127.0.0.1:9999 0.0.0.0:* LISTEN 19498/ntopng
```

Ed infine passiamo ad avviare NTOPNG…

Possiamo farlo attivandolo "solo" sull'interfaccia di Suricata nel modo seguente secondo che sia attivo su un generico PC nella nostra sottorete locale oppure sullo stesso PC di NTOP e quindi operando in localhost:




```
> ntopng -i syslog://192.168.2.222:9999
> ntopng -i syslog://127.0.0.1:9999
```

A questo punto visto che in suricata è stato attivato il traffico netflow già riceveremo non solo gli alert ma anche in copia campionamenti del traffico reale che NTOPNG elaborerà, ad ogni modo è tuttavia interessante indicargli di operare su doppia interfaccia e lasciarlo indipendente di analizzare il traffico in base alle proprie funzionalità standard ed in più collezionare ed integrare le informazioni con quelle ricevute dal dispatcher di Syslog

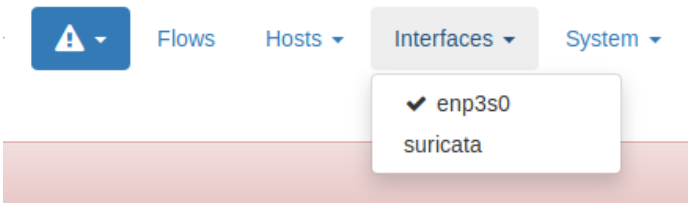
```
> ntopng -i enp0s3   -i syslog://127.0.0.1:9999
> ntopng -i eth0   -i syslog://127.0.0.1:9999
```

8 Validazione analisi traffico di rete

Le registrazioni del traffico sono state effettuate con un PC periferico sulla rete UNIPI presso il complesso della Facoltà di Scienze Matematiche, Fisiche e Naturali (SMFN) con sistema operativo Linux Mint V18.4 denominato “Sarah” e dotato di una scheda ethernet integrata nella main board da 1GBit/sec, 32GB di RAM e CPU quad-core.

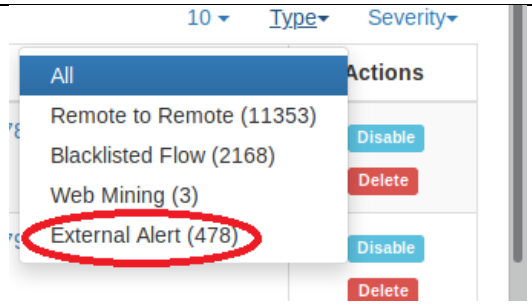
Un secondo PC con sistema operativo windows localizzato nella stessa sottorete di quello di test è stato utilizzato al bisogno per catturare i pacchetti grezzi in transito con Wireshark per avere quando possibile una controverifica empirica dei dati catturati

Le operazioni di test del traffico sono durate circa 2 settimane mantenendo attivi Suricata e NTOPNG sullo stesso PC dove per l’intercambio SYSLOG è stato configurato per operare in localhost su porta 127.0.0.1:9999. Come si può verificare dal menu delle interfacce di ascolto dichiarate all’avvio di ntop, dove “enp3s0” è l’ID ethernet tipico di Linux Mint e la voce “suricata” corrisponde allo pseudo-nome che è stato dato al canale localhost 127.0.0.1:9000 di ricezione da Syslog

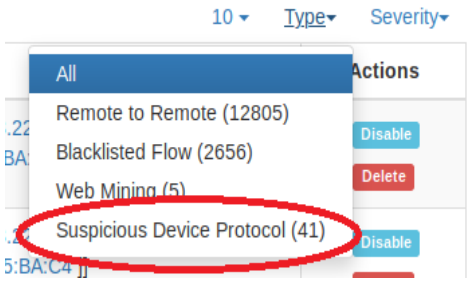


Nonostante la postazione di test utilizzata sia periferica e localizzata in una ramificazione locale all’interno dei laboratori, i volumi di traffico locale pubblico e broadcast sono risultati comunque consistenti e tali da essere riusciti a catturare e visionare molti “tipi di eventi” di varia tipologia.

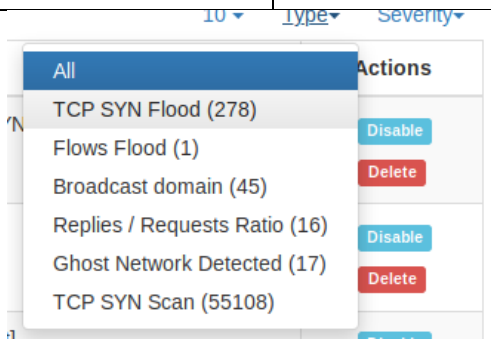
Ad’ esempio si riportano le voci di riepilogo della tipologia degli alert rilevati singolarmente dai “collettori” ntop e suricata in una delle fasi di test. Emergono subito alcuni aspetti interessanti dell’operazione di integrazione dei due applicativi.



"Flow alert" sull'interfaccia Suricata!



"Flow alert" sull'interfaccia Ntop!



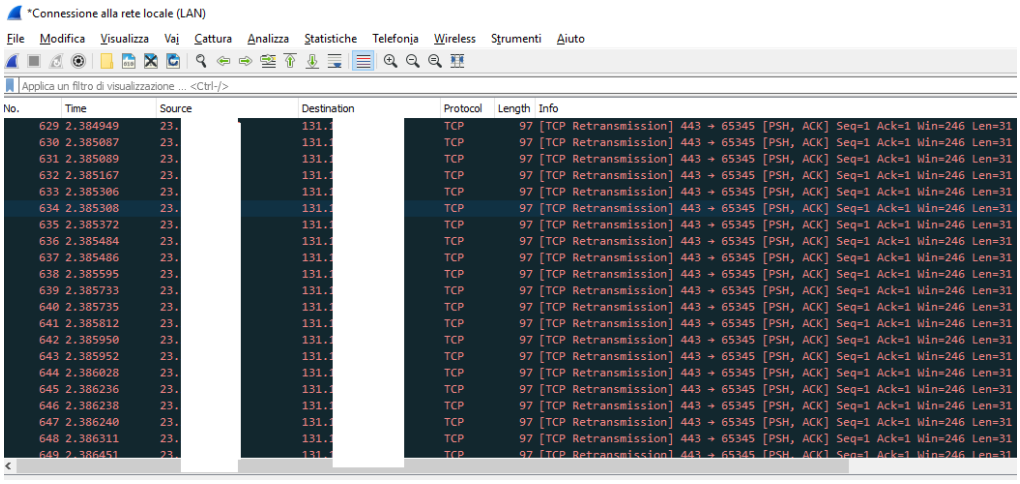
"Engaged alert" sull'interfaccia NTOPNG!

Si osserva subito che a livello di quantità di alert sui flussi (Flow alert) le differenze fra quanto rilevato sull'interfaccia Suricata e quella Ntop sono piccole e potrebbero essere imputabili alla velocità della scheda di rete del pc in uso, momenti di sovraccarico della CPU o altre cause tecniche, ma quello che più spicca sono le voci diverse:

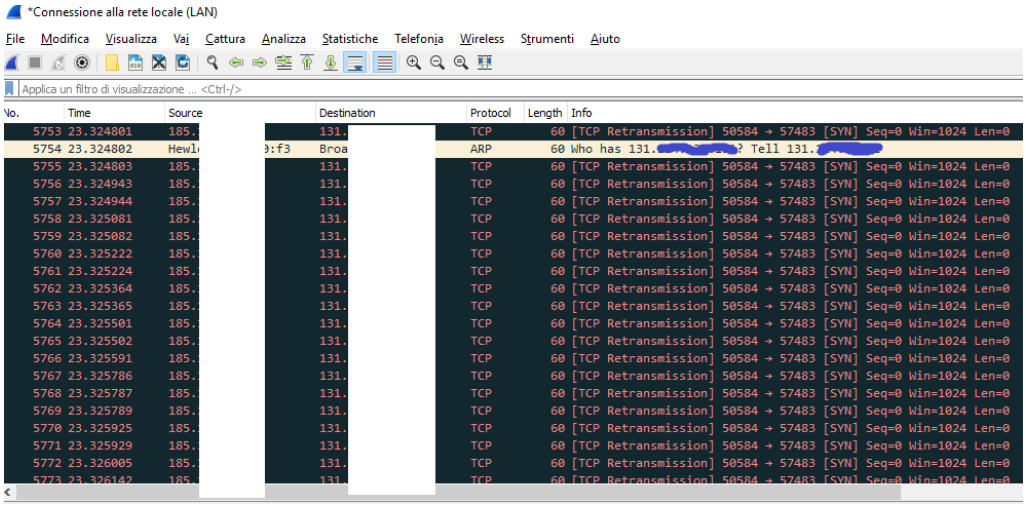
- "External Alert": sono la parte specifica dell'analisi in profondità IDS di Suricata "sul contenuto" dei pacchetti e di cui daremo qualche dettaglio fra poco.
- "Suspicius device Protocol": l'implementazione all'interno di NTOPNG della DPI, vedi il modulo nDPI, che lo rende in grado di analizzare e fare ipotesi sulla reale tipologia di molti tipi di protocolli di rete dei flussi rilevati che potrebbero essere offuscati.
- "Engaged alert": è un'altro dettaglio sulla tipologia di flussi di dati e/o sulla scansione dei device di rete che è tipico di ntopng e che Suricata non è in grado di fornire.

Senza entrare nello specifico di ogni singola tipologia di attacco e nel suo significato perché sarebbero eccessivi da trattare in questo contesto si rimanda al link: https://www.ntop.org/guides/ntopng/basic_concepts/alerts.html#misbehaving-flows un dettaglio più completo delle voci di "Warning ed Error" che possono comparire nella messaggistica di NTOPNG. Riportiamo qui a titolo di esempio solo qualche immagine di dettaglio degli allarmi più frequenti rilevati.

Si notano essere frequentissimi i problemi di categoria DoS (Denial of Service) tipo "TCP SYN" flood e "TCP SYN scan". Proprio perché molto frequenti è facile trovarne riscontro immediato anche confrontando quanto rilevato dal PC usato come "sniffer" con Wireshark in esecuzione.



(esempio di attacco SYN flood)



(esempio di attacco SYN scan)

Date/Time	Duration	Severity	Alert Type	Drilldown	Description	Actions
02:08 ago	01:00	Error	TCP SYN Scan		Host [redacted] is a SYN Scan attacker [122 > 50 SYN sent]	Disable Delete
02:08 ago	01:00	Error	TCP SYN Scan		Host [redacted] is a SYN Scan attacker [128 > 50 SYN sent]	Disable Delete
03:08 ago	02:00	Error	TCP SYN Scan		Host [redacted] is a SYN Scan attacker [126 > 50 SYN sent]	Disable Delete
02:08 ago	01:00	Error	TCP SYN Scan		Host [redacted] is a SYN Scan attacker [124 > 50 SYN sent]	Disable Delete
04:18 ago	03:00	Error	TCP SYN Scan		Host [redacted] is under SYN Scan [174 > 30 SYN received]	Disable Delete
03:18 ago	01:00	Error	TCP SYN Scan		Host [redacted] is a SYN Scan attacker [122 > 50 SYN sent]	Disable Delete
03:18 ago	01:00	Error	TCP SYN Scan		Host [redacted] is a SYN Scan attacker [123 > 50 SYN sent]	Disable Delete
04:18 ago	01:00	Error	TCP SYN Scan		Host [redacted] is a SYN Scan attacker [123 > 50 SYN sent]	Disable Delete
08:45:00	01:00	Error	TCP SYN Scan		Host [redacted] is a SYN Scan attacker [120 > 50 SYN sent]	Disable Delete
08:45:00	01:00	Error	TCP SYN Scan		Host [redacted] is a SYN Scan attacker [121 > 50 SYN sent]	Disable Delete

(dettaglio TCP SYN scan)

Date/Time	Duration	Severity	Alert Type	Drilldown	Description	Actions
11/11/2019 16:07:00	01:00	Error	TCP SYN Flood		Host [redacted] is under SYN flood attack [122 > 50 SYN received]	Disable Delete
11/11/2019 16:09:00	01:02	Error	TCP SYN Flood		Host [redacted] is under SYN flood attack [120 > 50 SYN received]	Disable Delete
11/11/2019 16:54:00	01:02	Error	TCP SYN Flood		Host [redacted] is under SYN flood attack [121 > 50 SYN received]	Disable Delete
11/11/2019 17:17:00	01:00	Error	TCP SYN Flood		Host [redacted] is under SYN flood attack [52 > 50 SYN received]	Disable Delete
11/11/2019 17:21:01	00:59	Error	TCP SYN Flood		Host [redacted] is a SYN flooder [123 > 50 SYN sent]	Disable Delete
11/11/2019 17:32:01	01:00	Error	TCP SYN Flood		Host [redacted] is a SYN flooder [122 > 50 SYN sent]	Disable Delete
11/11/2019 17:33:01	01:00	Error	TCP SYN Flood		Host [redacted] is a SYN flooder [123 > 50 SYN sent]	Disable Delete
11/11/2019 17:37:01	01:00	Error	TCP SYN Flood		Host [redacted] is under SYN flood attack [124 > 50 SYN received]	Disable Delete
11/11/2019 17:37:01	01:00	Error	TCP SYN Flood		Host [redacted] is a SYN flooder [124 > 50 SYN sent]	Disable Delete
11/11/2019 19:02:01	01:00	Error	TCP SYN Flood		Host [redacted] is under SYN flood attack [67 > 50 SYN received]	Disable Delete

(dettaglio TCP SYN flood)

Lo stesso confronto è stato fatto anche per alcune altre tipologie di eventi riscontrati con più frequenza dove era più facile farlo ma esula dal contesto di questo lavoro, quindi ne tralasciamo dettagli ulteriori.

Flow Alerts

10 Suspicious Dev

Date/Time	Duration	Count	Severity	Alert Type	Drilldown	Description
12/11/2019 13:41:50	-	1	Error	! Suspicious Device Protocol		Client application 'RX' is not allowed by the configured Printer application policy [Flow: [redacted] [UDP] [Application: RX]
12/11/2019 14:01:50	-	1	Error	! Suspicious Device Protocol		Client application 'RX' is not allowed by the configured Printer application policy [Flow: [redacted] [UDP] [Application: RX]

(attività sospetta che il flusso dati non sia offuscato)

00:39 ago	-	1	Error	! Remote to Remote		Remote client and remote server [Flow: Info [redacted]
01:14 ago	-	1	Error	! Remote to Remote		Remote client and remote server [Flow: [redacted] [TCP] [Application: Unknown]
01:19 ago	-	1	Error	! Remote to Remote		Remote client and remote server [Flow: [redacted] [UDP] [Application: NetBIOS]

(attività sospetta determinata da un flusso avente server e client nella stessa rete)

14:29:15	-	1	Error	฿ Web Mining		The website is known for mining cryptocurrencies on client devices [Flow: [redacted] [TCP] [Application: Mining]
14:29:05	-	1	Error	฿ Web Mining		The website is known for mining cryptocurrencies on client devices [Flow: [redacted] [TCP] [Application: Mining]
06:31:35	-	1	Error	฿ Web Mining		The website is known for mining cryptocurrencies on client devices [Flow: [redacted] [TCP] [Application: Mining]
19/11/2019 23:40:15	-	1	Error	฿ Web Mining		The website is known for mining cryptocurrencies on client devices [Flow: [redacted] [TCP] [Application: Mining]

(attività sospetta che un sito noto raccolga informazioni a nostra insaputa)

00:36 ago	-	1	Error	! Blacklisted Flow		Blacklisted Client [Flow: [redacted] [TCP] [Application: HTTP_Proxy]
00:51 ago	-	1	Error	! Blacklisted Flow		Blacklisted Client [Flow: [redacted] [TCP] [Application: Unknown]
01:26 ago	-	1	Error	! Blacklisted Flow		Blacklisted Client [Flow: [redacted] [TCP] [Application: HTTP_Proxy]
02:06 ago	-	1	Error	! Blacklisted Flow		Blacklisted Client [Flow: [redacted] [TCP] [Application: Unknown]
02:16 ago	-	1	Error	! Blacklisted Flow		Blacklisted Client [Flow: [redacted] [TCP] [Application: Unknown]

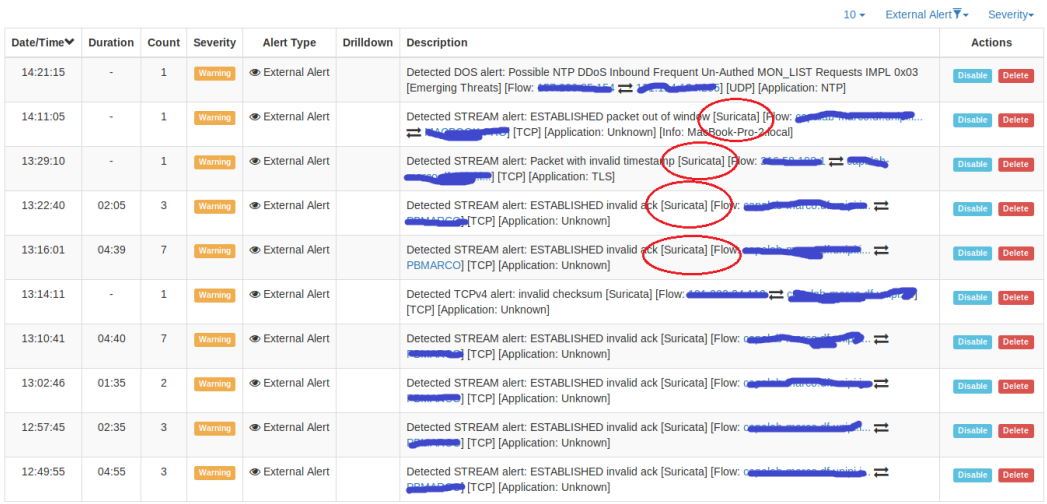
(rilevato un flusso fra client locali e siti appartenenti a liste nere ufficiali di IP con contenuti malevoli)

Torniamo quindi a verificare alcuni dettagli degli “External Alert” derivanti dalla messaggistica in arrivo da Suricata.

Come già riportato nei capitoli 4 e 5 Suricata trasmette dei file di log testuali in formato JSON che contengono informazioni sul singolo pacchetto ma non esegue nessuna forma di correlazione diretta ad informazioni “logistiche” degli stessi.

Server Name	configuration.apple.com
Additional Flow Elements	
Community ID	1:VIJm/8PMBYqQONmKOGGL29z99cI=
Suricata Flow ID	691166213991477

Questi messaggi però rianalizzati all’interno di NTOPNG possono essere rivalutati in base alle sue strategie di mapping dopo aver tramutato, ricalcolandolo, il “flow_id” in “community_ID” si riesce a ripresentarne i report arricchiti da dettagli sugli host coinvolti come dimostrano le seguenti immagini che riportano alcuni degli eventi rilevati da Suricata nelle fasi di test.



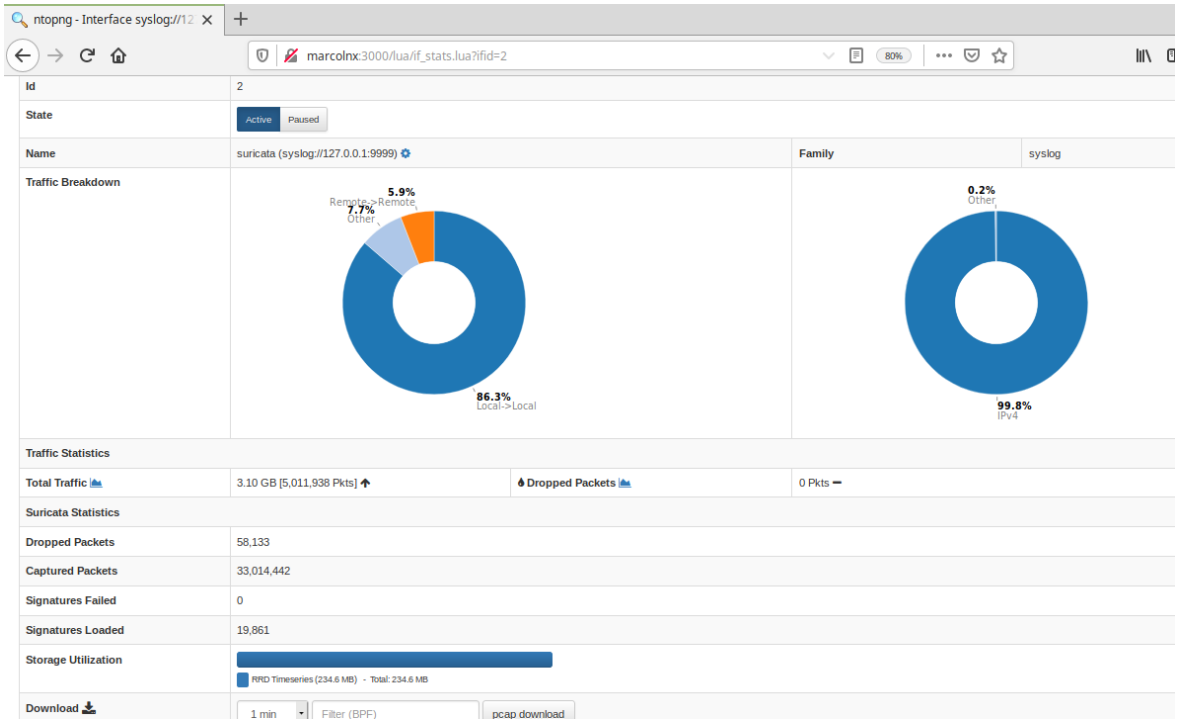
Date/Time	Duration	Count	Severity	Alert Type	Drilldown	Description	Actions
14:21:15	-	1	Warning	External Alert		Detected DOS alert: Possible NTP DDoS Inbound Frequent Un-Authed MON_LIST Requests IMPL 0x03 [Emerging Threats] [Flow: [redacted]] [UDP] [Application: NTP]	Disable Delete
14:11:05	-	1	Warning	External Alert		Detected STREAM alert: ESTABLISHED packet out of window [Suricata] [Flow: [redacted]] [TCP] [Application: Unknown] [Info: MacBook-Pro-2.local]	Disable Delete
13:29:10	-	1	Warning	External Alert		Detected STREAM alert: Packet with invalid timestamp [Suricata] [Flow: [redacted]] [TCP] [Application: TLS]	Disable Delete
13:22:40	02:05	3	Warning	External Alert		Detected STREAM alert: ESTABLISHED invalid ack [Suricata] [Flow: [redacted]] [TCP] [Application: Unknown]	Disable Delete
13:16:01	04:39	7	Warning	External Alert		Detected STREAM alert: ESTABLISHED invalid ack [Suricata] [Flow: [redacted] PBMARCO] [TCP] [Application: Unknown]	Disable Delete
13:14:11	-	1	Warning	External Alert		Detected TCPv4 alert: invalid checksum [Suricata] [Flow: [redacted]] [TCP] [Application: Unknown]	Disable Delete
13:10:41	04:40	7	Warning	External Alert		Detected STREAM alert: ESTABLISHED invalid ack [Suricata] [Flow: [redacted]] [TCP] [Application: Unknown]	Disable Delete
13:02:46	01:35	2	Warning	External Alert		Detected STREAM alert: ESTABLISHED invalid ack [Suricata] [Flow: [redacted]] [TCP] [Application: Unknown]	Disable Delete
12:57:45	02:35	3	Warning	External Alert		Detected STREAM alert: ESTABLISHED invalid ack [Suricata] [Flow: [redacted]] [TCP] [Application: Unknown]	Disable Delete
12:49:55	04:55	3	Warning	External Alert		Detected STREAM alert: ESTABLISHED invalid ack [Suricata] [Flow: [redacted]] [TCP] [Application: Unknown]	Disable Delete

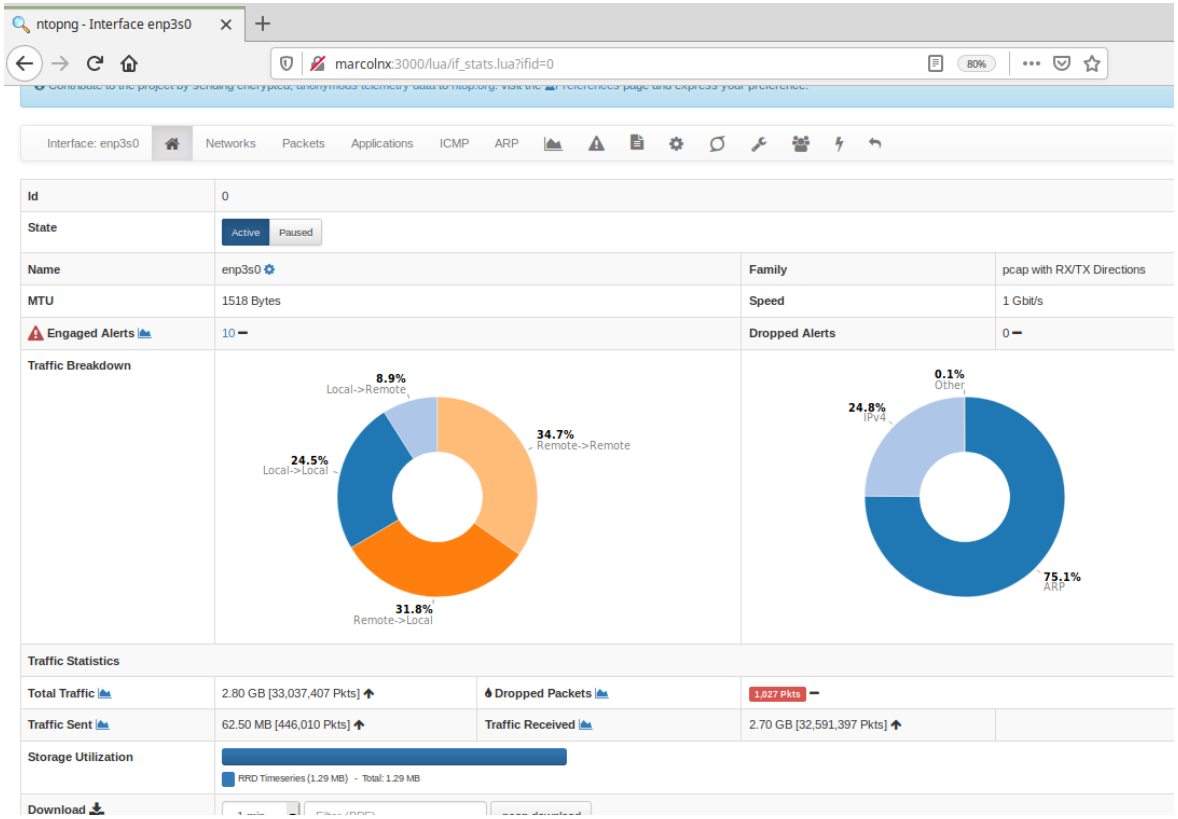
Date/Time	Duration	Count	Severity	Alert Type	Drilldown	Description	Actions
02:46 ago	-	1	Warning	External Alert		Detected SCAN alert: Suspicious inbound to MSSQL port 1433 [Emerging Threats] [Flow: [redacted]] [TCP] [Application: MsSQL-TDS]	Disable Delete
07:41 ago	-	1	Error	External Alert		Detected POLICY alert: Dropbox.com Offsite File Backup in Use [Emerging Threats] [Flow: [redacted]] [TCP] [Application: TLS] [Info: d.dropbox.com]	Disable Delete
17:50:55	-	1	Warning	External Alert		Detected SCAN alert: Suspicious inbound to MSSQL port 1433 [Emerging Threats] [Flow: [redacted]] [TCP] [Application: MsSQL-TDS]	Disable Delete
17:47:55	-	1	Error	External Alert		Detected POLICY alert: Dropbox.com Offsite File Backup in Use [Emerging Threats] [Flow: [redacted]] [TCP] [Application: TLS]	Disable Delete
17:46:00	-	1	Warning	External Alert		Detected POLICY alert: GNU/Linux APT User-Agent Outbound likely related to package management [Emerging Threats] [Flow: [redacted]] [TCP] [Application: HTTP] [Info: archive.canonical.com/ubuntu/dists/bionic/InRelease]	Disable Delete
17:46:00	-	1	Warning	External Alert		Detected POLICY alert: GNU/Linux APT User-Agent Outbound likely related to package management [Emerging Threats] [Flow: [redacted]] [TCP] [Application: HTTP] [Info: packages.linuxmint.com/dists/tessa/Release]	Disable Delete
17:45:55	00:05	3	Warning	External Alert		Detected POLICY alert: GNU/Linux APT User-Agent Outbound likely related to package management [Emerging Threats] [Flow: [redacted]] [TCP] [Application: HTTP]	Disable Delete
17:45:55	-	1	Warning	External Alert		Detected POLICY alert: GNU/Linux APT User-Agent Outbound likely related to package	Disable

Date/Time	Duration	Count	Severity	Alert Type	Drilldown	Description	Actions
14:44:00	-	1	Warning	External Alert		Detected STREAM alert: Packet with invalid timestamp [Suricata] [Flow: [redacted]] [TCP] [Application: TLS]	Disable Delete
14:38:55	-	1	Warning	External Alert		Detected STREAM alert: Packet with invalid timestamp [Suricata] [Flow: [redacted]] [TCP] [Application: TLS]	Disable Delete
09:44:10	-	1	Warning	External Alert		Detected STREAM alert: ESTABLISHED packet out of window [Suricata] [Flow: [redacted]] [TCP] [Application: Unknown] [Info: MacBook-Pro-2.local]	Disable Delete
09:28:00	00:25	2	Warning	External Alert		Detected STREAM alert: ESTABLISHED invalid ack [Suricata] [Flow: [redacted]] [TCP] [Application: Unknown]	Disable Delete
09:20:15	-	1	Warning	External Alert		Detected STREAM alert: ESTABLISHED packet out of window [Suricata] [Flow: [redacted]] [TCP] [Application: Unknown]	Disable Delete
09:11:05	-	1	Warning	External Alert		Detected STREAM alert: ESTABLISHED packet out of window [Suricata] [Flow: [redacted]] [TCP] [Application: Unknown] [Info: MacBook-Pro-2.local]	Disable Delete
09:07:35	-	1	Warning	External Alert		Detected DOS alert: Possible NTP DDoS Inbound Frequent Un-Authed MON_LIST Requests IMPL 0x03 [Emerging Threats] [Flow: [redacted]] [UDP] [Application: NTP]	Disable Delete
08:29:05	-	1	Warning	External Alert		Detected DOS alert: Possible NTP DDoS Inbound Frequent Un-Authed MON_LIST Requests IMPL 0x03 [Emerging Threats] [Flow: [redacted]] [UDP] [Application: NTP]	Disable Delete
06:32:50	-	1	Warning	External Alert		Detected DOS alert: Possible NTP DDoS Inbound Frequent Un-Authed MON_LIST Requests IMPL 0x03 [Emerging Threats] [Flow: [redacted]] [UDP] [Application: NTP]	Disable Delete
02:31:41	-	1	Warning	External Alert		Detected TCPv4 alert: invalid checksum [Suricata] [Flow: [redacted]] [TCP] [Application: Unknown]	Disable Delete

Per completezza infine si riportano 2 immagi con il riepilogo dei pacchetti analizzati sulle due interfacce in una fase di test e si noti che all'incirca entrambe le interfacce hanno "parsato" circa lo stesso numero di pacchetti (oltre 33milioni) e quindi la convivenza dei due tool sullo stesso PC non ha avuto alcun problema di risorse di sistema confermando

quanto dichiarato sia da ntop che suricata, verificabile facilmente dal “monitor risorse sistema”, che i due applicativi nella configurazione standard richiedono risorse di calcolo medie contenute.





8.1 Test con file pcap contenenti malware

Dopo le prove “ON-LINE” si è proceduto “replicando” in rete dei pacchetti TCP/IP campionati contenenti malware prelevati da sito Malware-Traffic-Analysis.net [6] (https://www.malware-traffic-analysis.net) e “rilanciati” sulla LAN sfruttando l’applicativo Linux “tcpreplay” che consente di ritrasmettere in rete il contenuto di una campionatura di traffico salvata in un file pcap.

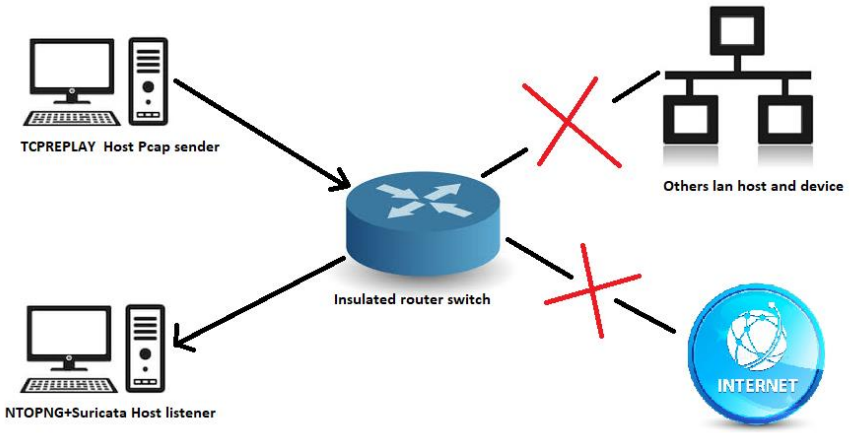
```
> tcpreplay -i eth0  malicious_test_file_.pcp
 più genericamente:
> tcpreplay -i <interfaccia_lan_da_usare> <file_pcap_da_trasmettere>
```

L’idea è quella di testare gli applicativi nel reimmettere il contenuto di registrazioni di pacchetti malevoli in modo che questo traffico problematico venga ricevuto simultaneamente da ntop e da suricata per confrontare i singoli report sulle due interfacce di ascolto distinte “verificando chi si accorge di cosa”.

Per fare questa operazione però, sebbene si tratti di file registrati reintrodurli nella rete potrebbe far allertare alcuni sistemi di sicurezza, quindi è opportuno effettuare questa operazione creando una mini-rete isolata apposita.

Per l’esattezza però ci attendiamo che con questo test la reattività massima avvenga sull’interfaccia Suricata/Syslog proprio per la tipologia di traffico che andiamo a testare.

Mediante un piccolo router-switch come quelli che ormai tutti hanno in casa per il traffico internet, si realizza una mini-rete a 2 pc tenuta disconnessa sia da internet che dal resto della rete LAN (di casa o aziendale che sia).



Malicious pcap test strategy!

La stessa operazione può essere fatta all'occorrenza anche continuando ad operare su un unico PC, ad esempio lo stesso dove risiede Ntopng+Suricata attivando tcpreplay in una shell separata. Il piccolo router esterno è comunque necessario per attivare correttamente l'hardware della/e interfacce ethernet di comunicazione soprattutto se è attivo il DHCP.

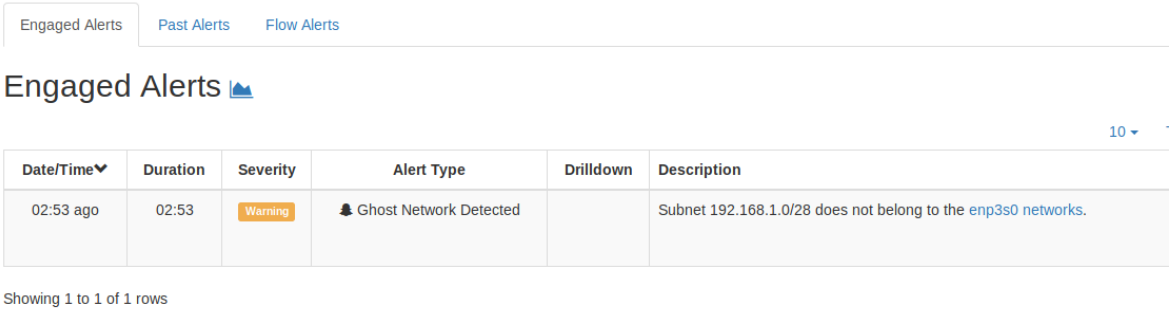
Siamo adesso in grado di fare una simulazione realistica e con un pc di servizio sono stati re-inviati in questa mini-rete isolata rete i contenuti dei pcap malevoli sintetizzati con tcpreplay.

Come ci aspettavamo essendo i pacchetti inviati essenzialmente malevoli nel contenuto ma non nel protocollo a livello di "flussi" sia l'interfaccia ntop che quella Syslog/Suricata ne hanno registrato il traffico ognuno con alcuni dettagli propri, ma a livello di "alert" solo Suricata è stato in grado analizzando il contenuto di generare warning ed error.

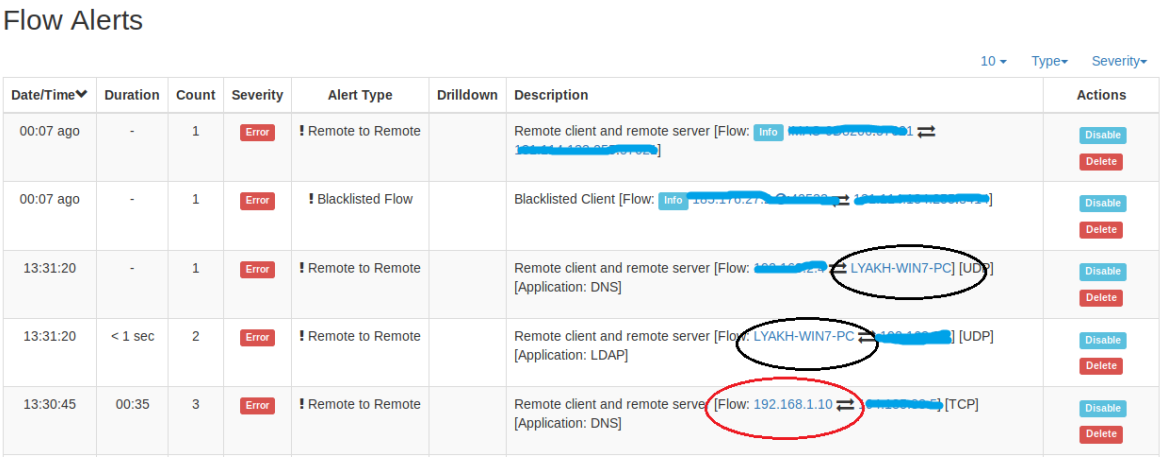
Le corrispondenze sui flussi si verifica sia controllando da entrambe le interfacce chi li ha generati e registrati visto che l'IP del PC che ha ritrasmesso i pcap malevoli è noto appartenete 192.168.1.x (tipico del DHCP della sottorete NAT di piccoli router) ed esattamente nel nostro caso abbiamo 192.168.1.10, sia perché abbiamo a grandi linee un'idea del contenuto dei pcap sintetizzati e possiamo fare ipotesi sulla reportistica ottenuta.

A Livello di alert dopo l'invio completo di alcuni blocchi di pcap con differente contenuto malevole o corrotto NTOPNG, come atteso, non ne ha registrato nessuno mentre Suricata ha prodotto svariati report coerenti.

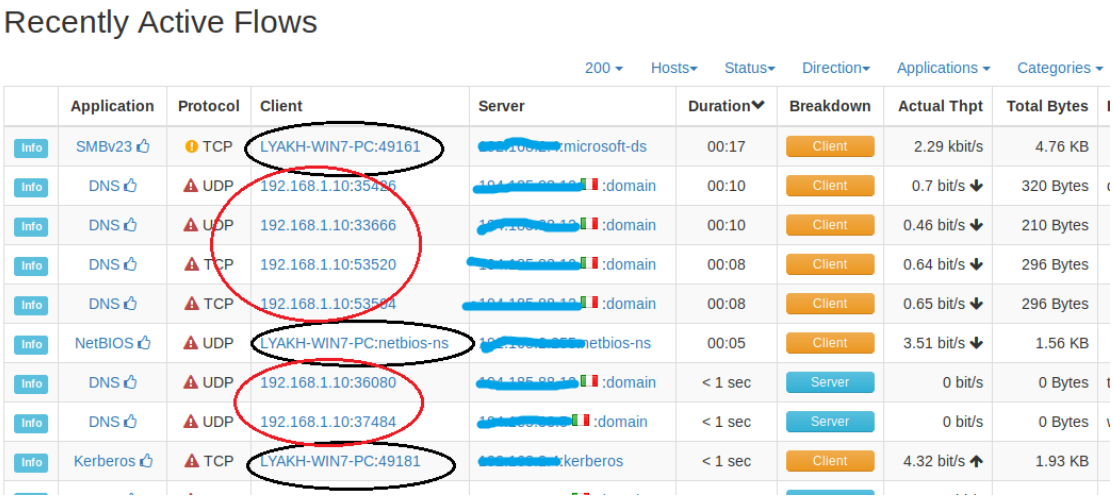
Riportiamo qui a titolo di esempio le immagini di uno solo dei test mettendo in evidenza alcune corrispondenze



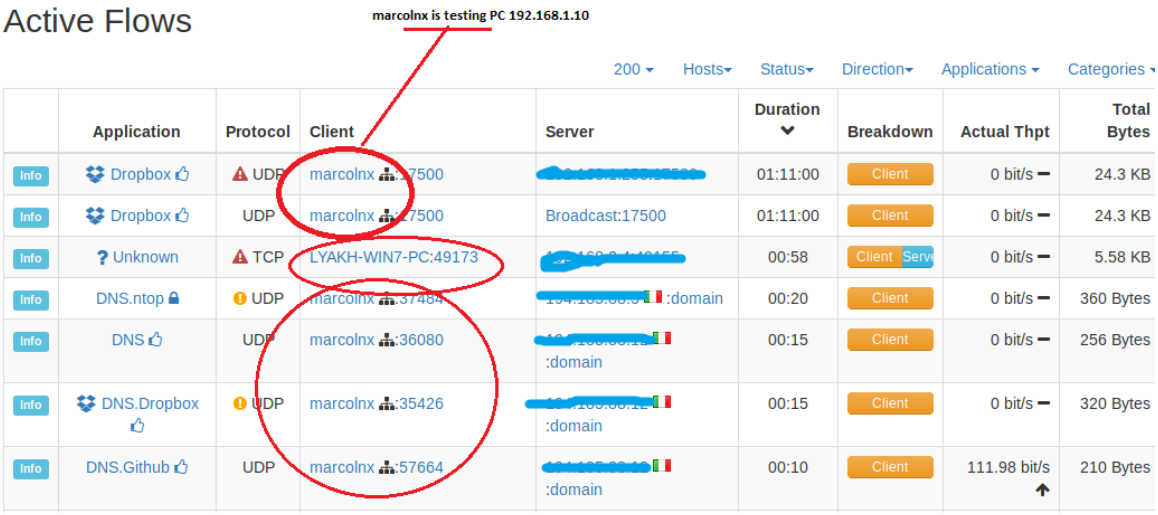
(alert rilevati dall' interfaccia ntop, quasi nessuno)



(alcuni alert rilevati dall' interfaccia Suricata)



(alcuni flussi rilevati dall' interfaccia Suricata)



(alcuni flussi rilevati dall' interfaccia ntop)

9 Idee e possibili ulteriori sviluppi

Visti gli sviluppi ottenuti risulta vi sia effettivamente un utile vantaggio di avere un'unica consolle di controllo convergendo report di applicativi che si integrano fra loro perché specializzati e capaci nel monitoraggio simultaneo dello stesso traffico di rete di fornire dettagli che opportunamente correlati danno molta più efficacia di supervisione che non i singoli report analizzati in modo indipendente, quindi idee per possibili futuri sviluppi della piattaforma NTOPNG potrebbero essere:

*) Sviluppare l'integrazione per altri IDS/IPS.

Sarebbe interessante proseguire il lavoro di integrazione fatto per Suricata realizzando opportuni moduli di integrazione per altri IDS/IPS come ad esempio "Zeek" (aka Bro) o Snort che sono molto spesso messi a confronto e risultano avere per avere ciascuno alcune peculiarità specifiche ad esempio vedi in bibliografia [9, 17, 46].

Zeek (https://www.zeek.org/) è il nuovo nome per il sistema noto con il nome "Bro" con supporto garantito per lungo termine è un sistema IDS open source che consente una grande flessibilità dell'analisi del contenuto dei pacchetti in modo arbitrario oltre agli script già forniti di serie, non prevede l'invio di allarmi in modo diretto ma è in grado di salvare uno storico retroattivo degli eventi che possono essere analizzati a posteriori e fra le funzionalità interessante adotta politiche di verifica sulle firme digitali.

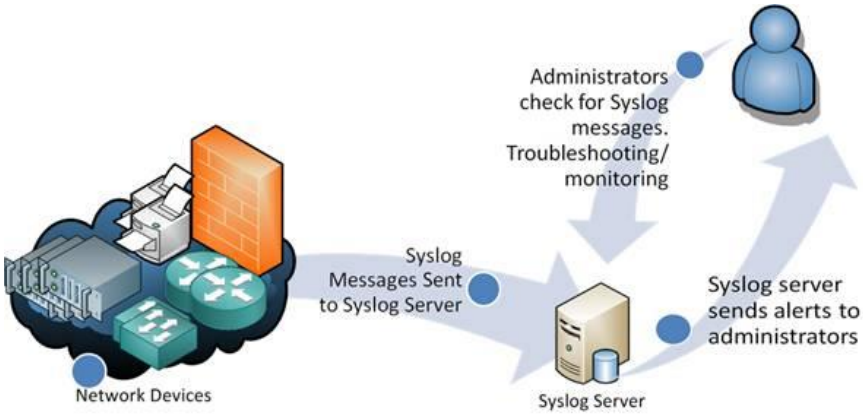
Snort (https://www.snort.org/) è un IDS sviluppato a partire dagli anni 90 e acquisito da Cisco nel 2013. Anche Snort è un software libero "open source" per l'analisi dei pacchetti all'interno di una rete distribuito sotto i termini della licenza libera GNU General Public License, è stato il precursore di Suricata che è stato sviluppato in secondo tempo dopo l'acquisizione di Snort da parte di Cisco ma che è tuttora sviluppato aggiornato ed utilizzato ed ha funzionalità simili a Suricata.

*) Implementare un servizio Server Syslog.

Si potrebbero realizzare moduli di ricezione in NTOPNG in modo da estenderlo con funzionalità di server Syslog per la gestione di report, allarmi e warning di malfunzionamento da dispositivi commerciali che "scrivono" in Syslog dato che fra gli apparati di fascia medio alta ve ne sono ormai molti in grado di inviare messaggistica di log in formato Syslog.

Questi sistemi sono noti con l'acronimo di SIEM, Security Information and Event Management [28, 47, 48], basta fare una semplice ricerca in rete per vedere che l'argomento è abbastanza diffuso e recente "fra gli addetti ai lavori":

http://www.ciscopress.com/articles/article.asp?p=426638&seqNum=3
https://www.loggly.com/docs/network-devices-and-routers/
https://help.papertrailapp.com/kb/configuration/configuring-remote-syslog-from-routers-switches-network-devices/
https://docs.bmc.com/docs/NetworkAutomation/89/troubleshooting/configuring-syslog-on-network-devices
https://www.networkmanagementsoftware.com/what-is-syslog/
https://www.gns3.com/news/article/configuring-remote-syslog-from-r



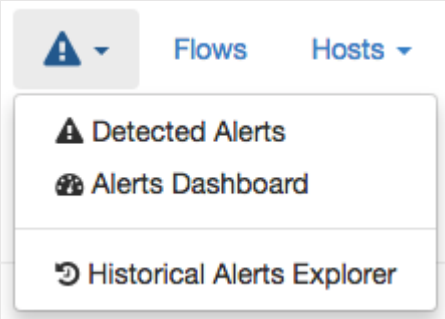
(Immagine tratta da: https://www.networkmanagementsoftware.com/what-is-syslog/)

10 Conclusioni

Il lavoro effettuato ha consentito effettivamente di raccogliere in un'unica piattaforma i riscontri di due applicativi che si complementano aumentando per l'operatore l'opportunità di avere un quadro più completo della rete che amministra.

La possibilità poi di essere avvertiti in automatico al sopraggiungere di eventi importanti derivanti anche da l'intervento di Suricata sfruttando i meccanismi già disponibili in ntop, ad esempio per e-mail rende più sinergico ed efficiente il monitoraggio.

(vedi: https://www.ntop.org/guides/ntopng/web_gui/alerts.html)



…….

12.1. Alert Endopints

Generated alerts can also be sent to third-party endpoints. Currently supported endpoints are:

- Email
- Slack
- Syslog
- Nagios
- Webhook

Endpoints can be enabled and configured from the ntopng preferences page.

……

All'atto pratico si è ottenuto quindi un'unione delle informazioni degli allarmi di Suricata con analisi del traffico di NTOPNG che può portare vantaggi per l'intera comunità open source, nonché per le comunità NTOPNG e Suricata.

Si potrebbe dire che la sicurezza e la visibilità della rete simultanee sono ora possibili o quantomeno si è cercato di aprire la strada a questa strategia utile su cui potrebbero essere fatti altri passi in avanti come ad esempio integrare la DPI in Suricata per ampliarne il riconoscimento di molti altri protocolli, infatti ad oggi Suricata è in grado di gestirne pochi e solo quelli più comuni, oppure convogliare anche alti tool per realizzare una piattaforma unica sempre più completa.

Quindi, integrare il monitoraggio di NTOPNG con suricata significa aggiungere alla parte di visibilità di rete l'introspezione e l'analisi dei contenuti confrontandoli con delle "firme" che servono per identificare i malware tipiche dell'operato della tecnologia IDS/IPS.

Di fatto Suricata non aggiunge direttamente alert a ntop ma gli permette di emettere allarmi basandoli sui contenuti degli eventi dei flussi ritenuti malevoli che gli trasmette quando vengono violate le opportune regole, tale che, ciascuno dei due applicativi mantiene la propria identità ma viene ora proposto un supporto dove possano interagire.

11 Appendici

11.1 “rsyslog.conf” modificato usato per i test

```
#  /etc/rsyslog.conf	Configuration file for rsyslog.
#
#			For more information see
#			/usr/share/doc/rsyslog-doc/html/rsyslog_conf.html
#
#  Default logging rules can be found in /etc/rsyslog.d/50-default.conf


#################
#### MODULES ####
#################

module(load="imuxsock") # provides support for local system logging
#module(load="immark")  # provides --MARK-- message capability

# provides UDP syslog reception
#module(load="imudp")
#input(type="imudp" port="514")

# provides TCP syslog reception
#module(load="imtcp")
#input(type="imtcp" port="514")

# provides kernel logging support and enable non-kernel klog messages
module(load="imklog" permitnonkernelfacility="on")

###########################
#### GLOBAL DIRECTIVES ####
###########################

#
# Use traditional timestamp format.
# To enable high precision timestamps, comment out the following line.
#
$ActionFileDefaultTemplate RSYSLOG_TraditionalFileFormat

# Filter duplicated messages
$RepeatedMsgReduction on

#
# Set the default permissions for all log files.
#
$FileOwner syslog
$FileGroup adm
$FileCreateMode 0640
$DirCreateMode 0755
$Umask 0022
$PrivDropToUser syslog
$PrivDropToGroup syslog

#$template sysklogd, "<%PRI%>%syslogtag:1:32%%msg:::sp-if-no-1st-sp%%msg%"
$template sysklogd, "<%PRI%> %HOSTNAME% %timegenerated% %syslogtag:1:32% %msg:::sp-if-no-1st-sp% %msg%"


#A template that resembles traditional syslogd file output:
$template TraditionalFormat,"%timegenerated% %HOSTNAME% %syslogtag% %msg:::drop-last-lf%\n"

#A template that tells you a little more about the message:
$template precise,"%syslogpriority%,%syslogfacility%,%timegenerated%,%HOSTNAME%,%syslogtag%,%msg%\n"

#A template for RFC 3164 format:
$template RFC3164fmt,"<%PRI%>%TIMESTAMP% %HOSTNAME% %syslogtag%%msg%"

#A template for the format traditonally used for user messages:
$template usermsg, " XXXX%syslogtag%%msg%\n\r"

#And a template with the traditonal wall-message format:
$template wallmsg, "\r\n\7Message from syslogd@%HOSTNAME% at %timegenerated%

#A template that can be used for the database write (please note the SQLtemplate option)
$template MySQLInsert,"insert iut, message, receivedat values ('%iut%', '%msg:::UPPERCASE%', '%timegenerated:::date-mysql%') into systemevents\r\n", SQL

#The following template emulates WinSyslog format (it's an Adiscon format, you do not feel bad if you don't know it ;)). It's interesting to see how it takes different parts out of the
date stamps. What happens is that the date stamp is split into the actual date and time and the these two are combined with just a comma in between them.

$template WinSyslogFmt,"%HOSTNAME%,%timegenerated:1:10:date-rfc3339%, %timegenerated:12:19:date-rfc3339%,%timegenerated:1:10:date-rfc3339%,
%timegenerated:12:19:date-rfc3339%,%syslogfacility%,%syslogpriority%,%syslogtag%%msg%\n"

#
# Where to place spool and state files
#
$WorkDirectory /var/spool/rsyslog

#
# Include all config files in /etc/rsyslog.d/
```

```
#
$IncludeConfig /etc/rsyslog.d/*.conf

#user.alert @127.0.0.1:514;sysklogd
#user.* @127.0.0.1:514;sysklogd
#*.* @127.0.0.1:9999;sysklogd

*.* @@(o)127.0.0.1:9999;sysklogd


#*.* @127.0.0.1:514;TraditionalFormat
#*.* @131.114.134.51:514;sysklogd
#*.*  action(type="omfwd" target="127.0.0.1" port="9999" protocol="tcp" action.resumeRetryCount="100" queue.type="linkedList" queue.size="10000")

#local5.* @131.114.134.51:514;sysklogd
```

11.2 "suricata.yaml" modificato usato per i test

Visto la dimensione consistente di questo file sono state riportate solo le parti che è stato necessario modificare.

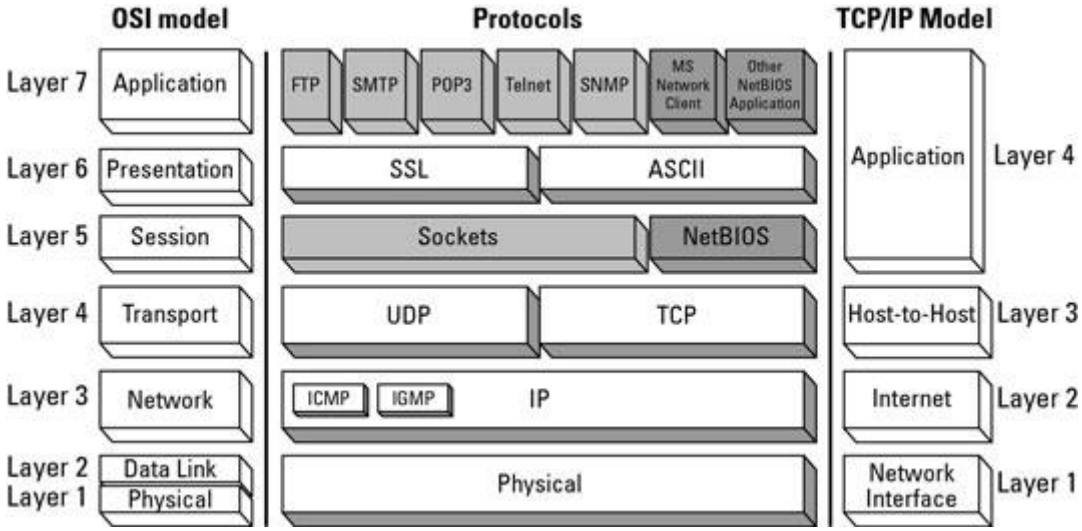
```
%YAML 1.1

# Suricata configuration file. In addition to the comments describing all
# options in this file, full documentation can be found at:
# https://redmine.openinfosecfoundation.org/projects/suricata/wiki/Suricatayaml

##
## Step 1: inform Suricata about your network
##


vars:


 # more specifc is better for alert accuracy and performance
 address-groups:


   HOME_NET: "[0.0.0.0/0]"
   #HOME_NET: "[192.168.0.0/16]"
   #HOME_NET: "[10.0.0.0/8]"
   #HOME_NET: "[172.16.0.0/12]"
   #HOME_NET: "any"


   EXTERNAL_NET: "!$HOME_NET"
   #EXTERNAL_NET: "any"


.....
.....

 # Extensible Event Format (nicknamed EVE) event log in JSON format
 - eve-log:
     enabled: yes
     filetype: syslog #regular|syslog|unix_dgram|unix_stream|redis
     filename: eve.json
     #prefix: "@cee: " # prefix to prepend to each log entry
     # the following are valid when type: syslog above
     identity: "suricata"
     facility: local5
     level: Emergency, Alert, Critical, Error, Warning, Notice, Info, Debug

     #level: Alert  ## possible levels: Emergency, Alert, Critical,
                ## Error, Warning, Notice, Info, Debug

.....
.....

       # bi-directional flows
       #- flow
       # uni-directional flows
       - netflow
       #- dnp3

.....
.....

 # a line based alerts log similar to fast.log into syslog
 - syslog:
     #enabled: no
     enabled: yes
     # reported identity to syslog. If ommited the program name (usually
```

```
    # suricata) will be used.
    identity: "suricata"
    facility: local5
    level: Emergency, Alert, Critical, Error, Warning, Notice, Info, Debug

    #level: Alert ## possible levels: Emergency, Alert, Critical,
              ## Error, Warning, Notice, Info, Debug

.....
.....

 # Define your logging outputs.  If none are defined, or they are all
 # disabled you will get the default - console output.
 outputs:
 - console:
    enabled: yes
    # type: json
 - file:
    enabled: yes
    level: info
    filename: /var/log/suricata/suricata.log
    # type: json
 - syslog:
    #enabled: no
    enabled: yes
    facility: local5
    format: "[%i] <%d> -- "
    type: json

.....
.....

##
## Include other configs
##

# Includes.  Files included here will be handled as if they were
# inlined in this configuration file.
#include: include1.yaml
#include: include2.yaml
```

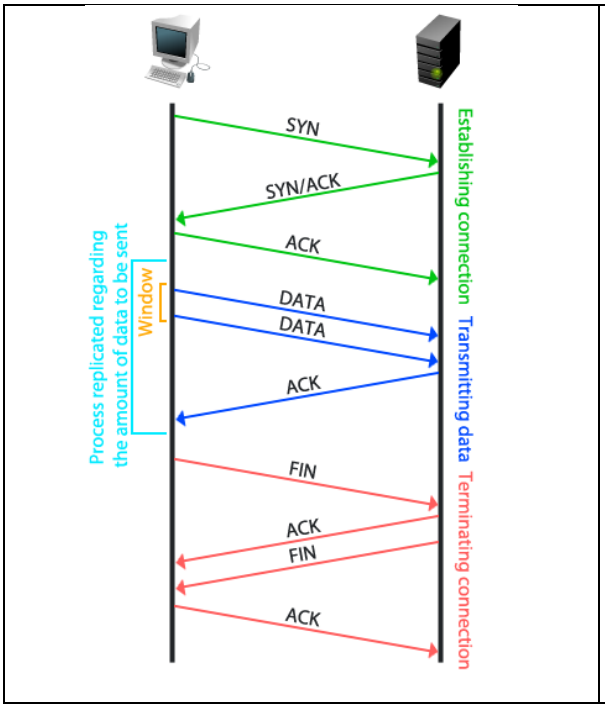
11.3 Layer protocolli di rete OSI e TCP-IP

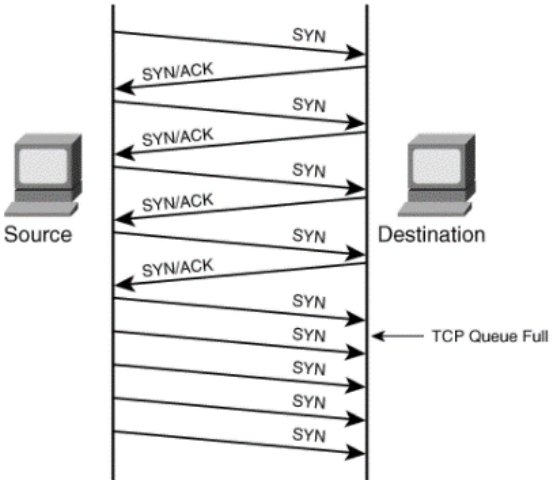
Come noto i protocolli di comunicazione sulla rete operano a livelli inter-operanti che possono essere 4 o 7 secondo la catalogazione OSI (Open Systems Interconnection) o quella più comune normalmente adottata TCP-IP (Transmission Control Protocol - Internet Protocol).

Ecco un dettaglio comparativo dei layer OSI/TCP-IP con l'elenco i protocolli più comuni operanti in rete ed in quale livello si collocano:



11.4 Confronto sequenza standard TCP-IP e sequenza di SYN attack





service denied for overload requests

11.5 Tabella associazione porta-protocollo di alcuni servizi di rete più comuni

7	ECHO
9	DISCARD
13	DAYTIME
17	QUOTD (Quote of the day)
19	CHARGEN (Character generator)
20	FTP-DATA (FTP data transfer)
21	FTP (File Transfer Protocol)
22	SSH (Secure Shell)
23	TELNET
25	SMTP (Simple Mail Transfer Protocol)
42	WINS (Windows Internet Naming Service)
53	DNS (Domain Name Server)
69	TFTP (Trivial File Transfer Protocol)
79	FINGER
80	HTTP (Hyper Text Transfer Protocol)
110	POP3 (Post Office Protocol 3)
113	IDENT/AUTH
119	NNTP (Network News Transfer Protocol)
135	EPMAP (DCE Endpoint Mapper)
137	NETBIOS-ns (name service)
138	NETBIOS-dgm (datagram service)
139	NETBIOS-ss (session service)
143	IMAP (Internet Message Access Protocol)
161	SNMP (Simple Network Management Protocol)
389	LDAP (Lightweight Directory Access Protocol)
443	HTTPS (Secure HTTP)
445	Microsoft-ds (Microsoft Directory Service)

12 Bibliografia

[1] Christian, Kreibich. Corelight Inc., An open standard for hashing network flows into identifiers, a.k.a "community IDs", https://github.com/corelight/community-id-spec.

[2] Luca Deri, Alfredo Cardigliano, Simone Mainardi, ntop, ntopng general repository GitHub, https://github.com/ntop/ntopng.

[3] OISF, Open Information Security Foundation, Suricata official website, https://suricata-ids.org.

[4] Ng, Chee Keong, Lei Pan, and Yang Xiang. "Ramsonware and Honeypot." *Honeypot Frameworks and Their Applications: A New Framework*. Springer, Singapore, 2018. 75-78.

[5] Proofpoint, Company, Cyber Security Solution, Emerging Threats, rules server, Raw IPs for the firewall block lists, https://rules.emergingthreats.net/fwrules/emerging-Block-IPs.txt .

[6] Malware-Traffic-Analysis.net, A source for pcap files and malware samples, https://www.malware-traffic-analysis.net.

[7] Douglas, Crockford. Introduzione a JSON, https://www.json.org/json-it.html

[8] Luca Deri, Alfredo Cardigliano, Simone Mainardi, ntop, Ntopng, repository C++ source code, https://github/ntop/ntopng/tree/dev/src .

[9] Alessio Trevisonno, Università degli Studi di Bologna, Tesi: "Studio di un Intrusion Detection System basato su Packet Sampling", a.a. 2016-2017.

[10] OISF, Open Information Security Foundation, Suricata official website, User Guide, https://suricata.readthedocs.io/.

[11] Wikipedia, Intro file pcap, https://en.wikipedia.org/wiki/Pcap.

[12] Reid, Gilman, opensecuritytraining.info, "Intro to pcap", Open Security Training, Approved for Public Release: 13-0979. Distribution Unlimited, http://opensecuritytraining.info/Pcap_files/intro-to-pcap-public-release.pdf

[13] Tina Müller, YAML, The Official YAML Web Site, https://yaml.org/

[14] Rainer, Gerhards, IETF, RFC 5424 The Syslog Protocol, March 2009, https://tools.ietf.org/pdf/rfc5424.pdf

[15] Ntop, Official Web Site, Doc Advanced Features, Suricata Integration, https://www.ntop.org/guides/ntopng/advanced_features/suricata.html

[16] Hunt, Craig. *TCP/IP network administration*. Vol. 2. " O'Reilly Media, Inc.", 2002.

[17] Mirko Pazzaglia. Università degli Studi di Camerino, Tesi, "Suricata: caso di studio di Intrusion Detection and Prevention System" a.a. 2011-2012.

[18] McCanne, Steve. *"libpcap: An Architecture and Optimization Methodology for Packet Capture"*, 2013, https://sharkfestus.wireshark.org/sharkfest.11/presentations/McCanne-Sharkfest'11_Keynote_Address.pdf

[19] Macedo, Tiago, and Fred Oliveira. *Redis Cookbook: Practical Techniques for Fast Data Manipulation*. " O'Reilly Media, Inc.", 2011.

[20] Ntop, Official Web Site, https://www.ntop.org/

[21] Fuchsberger, Andreas. "Intrusion detection systems and intrusion prevention systems." *Information Security Technical Report* 10.3 (2005): 134-139.

[22] GitHub, Official Web Site, development platform, https://github.com/.

[23] Wikipedia, GitHub intro, https://it.wikipedia.org/wiki/GitHub

[24] Wikipedia, DPI intro, https://en.wikipedia.org/wiki/Deep_packet_inspection

[25] Ntop, Official Web Site, nDPI library product, https://www.ntop.org/products/deep-packet-inspection/ndpi/

[26] Ntop, Official Web Site, nprobe library product, https://www.ntop.org/products/netflow/nprobe/

[27] Ntop, Official Web Site, pf_ring modules, https://www.ntop.org/products/packet-capture/pf_ring/

[28] Kufel, Lukasz. "Security event monitoring in a distributed systems environment." *IEEE Security & Privacy* 11.1 (2012): 36-43.

[29] Wikipedia, LUA, Introduzione al linguaggio, Wikipedia, https://it.wikipedia.org/wiki/Lua

[30] Xiang, Wang. Intel Software Development Zone, Hyperscan Intel, “developer zone” articles, https://software.intel.com/en-us/articles/introduction-to-hyperscan.

[31] Geoff, Langdale. Intel corporation, SuriCon conference 2016, “Hyperscan In Suricata: State of the union”, https://suricon.net/wp-content/uploads/2016/11/SuriCon2016_GeoffLangdale.pdf

[32] Spafford, Eugene H. "The Internet worm program: An analysis." *ACM SIGCOMM Computer Communication Review* 19.1 (1989): 17-57.

[33] OISF, Open Information Security Foundation, Suricata official website, Suricata docs, Syslog Alerting Compatibility, https://suricata.readthedocs.io/en/suricata-5.0.0/output/syslog-alerting-comp.html.

[34] Ntop, Official Web Site, developer support documentation, https://github.com/ntop/ntopng/tree/dev/doc.

[35] Peter, Czanik. Syslog-ng Community, SuriCon Conference 2018 in Vancouver, Analize your Suricata logs in real-time using syslog-ng, https://www.syslog-ng.com/community/b/blog/posts/analyze-your-suricata-logs-in-real-time-using-syslog-ng, Syslog-ng Community

[36] OISF, Official SuriCon Conference web site, https://suricon.net/

[37] Wikipedia, Syslog introduzione, https://it.wikipedia.org/wiki/SysLog

[38] Wang, Haining, Danlu Zhang, and Kang G. Shin. "Detecting SYN flooding attacks." *Proceedings. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies*. Vol. 3. IEEE, 2002.

[39] Wireshark©, Free tool for network traffic analysis and log protocols, https://www.wireshark.org

[40] IETF, Internet Engineering Task Force ONG, https://it.wikipedia.org/wiki/Internet_Engineering_Task_Force.

[41] Wikipedia, Lista di porte standard di protocolli di rete noti, https://it.wikipedia.org/wiki/Lista_di_porte_standard

[42] OISF, Suricata User Guide, 2019, https://buildmedia.readthedocs.org/media/pdf/suricata/latest/suricata.pdf

[43] Alfredo De Santis, Diego Radica, Università degli Studi di Salerno - introduzione a tipologie di attacchi di rete, http://www.di-srv.unisa.it/~ads/corso-security/www/CORSO-0203/Cisco/tipi_attacchi_htm/tipi_attacchi_intro.htm.

[44] OISF, Open Information Security Foundation, Suricata official website, Suricata doc, Meta Keywords, https://suricata.readthedocs.io/en/suricata-5.0.0/rules/meta.html

[45] Luca Deri, Alfredo Cardigliano, SuriCon 2019, Conference Amsterdam: “Ntopng e Suricata: unificare visibilità con sicurezza, https://www.ntop.org/ntopng/ntopng-suricata-unifying-visibility-with-security/

[46] Ridho, M. Faqih. *Analysis and evaluation snort, bro, and suricata as intrusion detection system based on linux server*. Diss. Universitas Muhammadiyah Surakarta, 2014.

[47] Wikipedia, SIEM intro, https://it.wikipedia.org/wiki/SIEM

[48] University of Houston Clear Lake, Security Information and Event Monitoring (SIEM), https://www.uhcl.edu/computing/information-security/tips-best-practices/siem

[49] Hansberry, Ashley, A. Lasse, and Andrew Tarrh. "Cryptolocker: 2013’s Most Malicious Malware." *Retrieved February* 9 (2014): 2017.

[50] Wikipedia, Slack, https://it.wikipedia.org/wiki/Slack

[51] Ntop, Official Web Site, nIndex Intro, https://www.ntop.org/ntopng/say-hello-to-nindex-personal-big-data-system-for-network-flows/