ESERCIZIO INTRUSION DETECTION – ESAME 27 LUGLIO 2023

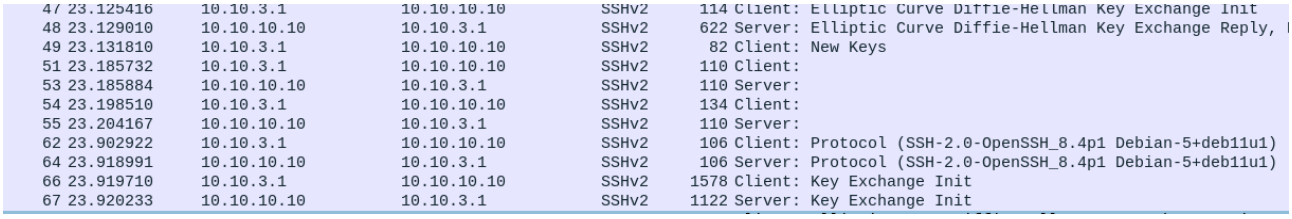
0) Spiegare che differenza c'è tra Suricata in modalita IPS e suricata in modalità IDS. Evidenziare in particolare in quale modalità è stato da noi utilizzato, evidenziandone pregi e svantaggi
1) Scrivere una regola suricata in modalità alert per il traffico icmp SOLTANTO IN ENTRATA sulla rete 192.168.56.X
2) Scrivere una regola suricata in modalità alert per qualsiasi richiesta che viene eseguita accedendo al portale virtuale.unibo.it
3) Ricavare dal file del tracciato in allegato le seguenti informazioni: Protocollo del traffico (non è SSH), I 2 Indirizzi Ip in gioco (considerate che ovviamente il traffico è in due direzioni), La/Le porte del protocollo in gioco

0) Suricata è in grado di eseguire sia rilevamento delle intrusioni in tempo reale (IDS intrusion detection system) che prevenzione delle intrusioni (IPS intrusion protection system). Noi lo abbiamo utilizzato nella modalità IDS anche se ci siamo prevalentemente concentrati in esercitazioni in cui l'elaborazione dei pcap era offline.
1) Prima di tutto è necessario modificare il file /etc/suricata/suricata.yaml. Nello specifico va modificata la variabile HOME_NET in HOME_NET: "[192.168.56.0/24]"
   alert icmp any any -> $HOME_NET any (msg:"logged icmp traffic"; itype:8; sid:1009876; rev:1;)
2) alert dns any any -> any any (msg:"logged dns_query; content:virtuale.unibo.it"; sid:1009877; rev:1;)
3) Utilizzo wireshark per analizzare il tracciato contenuto nel file .pcapng. Il protocollo del traffico è ircu, sulla porta 6666.
   Gli indirizzi ip in gioco sono 192.168.56.1 e 192.168.56.3
   Le porte in gioco sono 54968 per 192.168.56.1 e 6666 per 192.168.56.3
   Acquisite queste informazioni è necessario scrivere la regola nel file /etc/suricata/rules/seclab.rules
   alert tcp 192.168.56.1 54968 -> 192.168.56.3 6666 (msg:"logged ircu traffic"; flow:to_server; sid:1009481; rev:1;)
   per far partire l'analisi del file .pcapng con il comando
   ```
   sudo suricata -c /etc/suricata/suricata.yaml -r file.pcapng -k none
   ```
   il risultato sta nel file eve.json. Utilizziamo jq per parsarlo. Prima filtriamo tutti i pacchetti che hanno atteribuito event_type = "alert" e tra quelli filtriamo quelli che possiendono l'attributo payload_printable
   ```
   cat eve.json | jq 'select(.event_type=="alert") | .payload_printable'
   ```
   la flag risulta spezzettata su più pacchetti differenti, ricomponendola otteniamo: SEC{this_is_the_first_part_this_is_the_second_part_third_part}

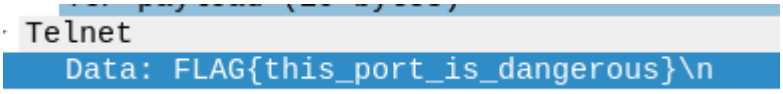
ESERCIZIO SURICATA 11 GENNAIO 2024

1) Scaricare traffic e analizzarlo con wireshark, vedere quali sono i 4 tipi di interazione verso il server 10.10.10.10
2) Scrivere una regola suricata che permette di registrare la FLAG nel file degli eventi

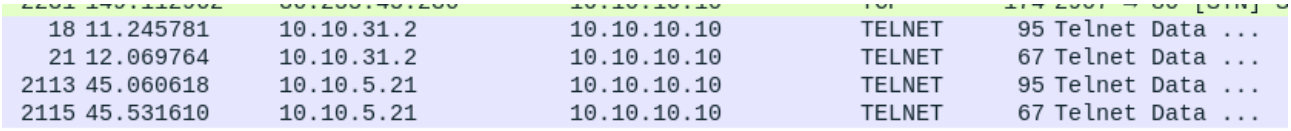
1) Ordinando per indirizzo sorgente si notano grandi quantità di pacchetti da 10.10.10.10 per porte variabili -> port scan
   Ordinando per protocollo notiamo: una normale connessione SSH da 10.10.10.10



   E una connessione TELNET con payload contenente una flag



   Ordinando i pacchetti per indirizzo destinazione vediamo una grande quantità di pacchetti TCP che derivano da indirizzi diversi e vanno alla porta 80. Questo vuol dire DDos



2) Vediamo che i pacchetti TELNET (quelli che contengono la flag) hanno destination port 23, quindi è li che dovremmo creare l'alert. Derivano da indirizzi 10.10.X.X quindi sarà:
   alert tcp 10.10.0.0/16 any -> 10.10.10.10 23 (msg:"Flag"; content:"FLAG"; sid:30202030; rev:1;)
   Poi andiamo a stampare la flag con i passaggi qua sotto

PASSAGGI DA FARE

Andiamo a fare `sudo nano /etc/suricata/rules/seclab.rules` dove andiamo ad inserire la nostra regola e salviamo

Andiamo a metterci in ascolto su un altro tab sui log facendo `tail -f /var/log/suricata/fast.log`

Facciamo partire suricata in modalità offline con `sudo suricata -r [path del tracciato.pcap] -c /etc/suricata/suricata.yaml -l /var/log/suricata/`

Così riusciamo a vedere gli alert in fast.log

Per vederli in eve.json con il payload modifichiamo suricata.yaml attivando payload-printable

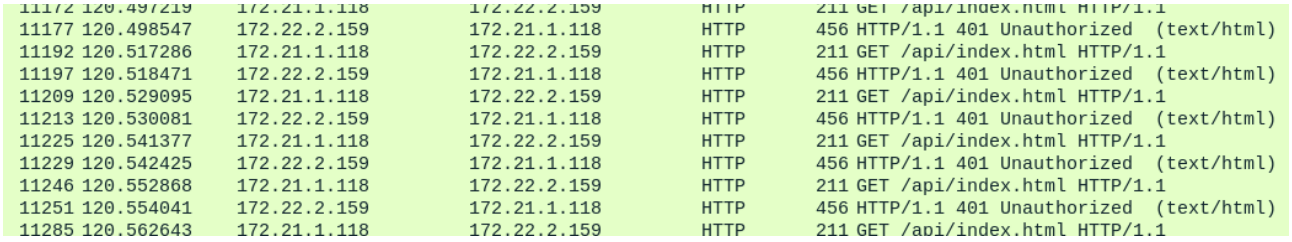
Dopo di che al posto di fast.log andiamo a fare `tail -f /var/log/suricata/eve.json | jq 'select(.event_type="alert")' | grep FLAG`

In questo caso facciamo una grep con FLAG perché era il messaggio che avevamo inserito nell'alert

ESERCIZIO SURICATA 8 FEBBRAIO 2024

1) Analizzare traffico con wireshark, il tracciato è raccolto su un router che ha indirizzo con byte finale = 1 su tre subnet diverse. Usare ordinamento per distinguere i diversi tipi di traffico. Concentratevi sui protocolli applicativi ignorando i numerosi TCP degli handshake. Un solo dei tipi di traffico è evidentemente un attacco
2) Scrivere due regole suricata per registrare nel file eventi il passaggio dei pacchetti dall'attaccante al bersaglio, il passaggio dei pacchetti di risposta dal bersaglio all'attaccante. Inserire nella consegna anche il file eve.json prodotto analizzando il tracciato.

1) C'è del traffico DHCP con degli invii broadcast, sapendo il suo funzionamento non è strano questo comportamento. Poi c'è del traffico DNS, può essere usato per nascondere informazioni mettendo comandi nelle query.
   Si vedono tantissime richieste di tipo http, si vedono richieste del tipo,
   se andiamo a vedere nei dettagli dei pacchetti GET vediamo dei tentativi di accesso dove in Authorization: .... Ci sono stringhe che cambiano ad ogni richiesta. È chiaro che si tratta di un tentativo di bruteforcing.



   Quindi una richiesta http GET alla macchina 172.22.2.159 che prova ad accedere a /api/index.html (non so come capire la macchina dal quale provengono)
   Vediamo anche le relative risposte. Quindi il server che non autorizza rispondendo con uno status 401

2) Per identificare i pacchetti dell'attaccante scriviamo: **alert tcp 172.21.1.118 any -> 172.22.2.159 80 (msg:"Attacco";content:"/api/index.html";sid:10000001;rev:1;)**

   Per i pacchetti di risposta: **alert tcp 172.22.2.159 any -> 172.21.1.118 80 (msg:"Risposta";content:"401";sid:10000002;rev:1;)**

ESERCIZIO SURICATA 10 FEB 2023

1) Scrivere una regola suricata in modalità alert per TUTTO il traffico icmp SOLTANTO IN ENTRATA sulla rete 192.168.X.X
   Soluzione:
   - andiamo a modificare /etc/suricata/suricata.yaml per cambiare la variabile HOME_NET e quindi gli diciamo quale rete vogliamo monitorare: **HOME_NET: "[192.168.0.0/16]"**
   - **alert icmp any any -> $HOME_NET any (msg:"Ping detected"; sid:100001; rev:1;)** -> regola che monitora il traffico da qualsiasi rete alla nostra home net di tipo icmp (i ping), le opzioni non so cosa significhino.

2) Scrivere una (o più) regola suricata in modalità alert per qualsiasi richiesta a evilcorp.com. Nota bene NON è possibile utilizzare il protocollo http o la porta 80 per creare questa regola.
   Soluzione:
   - **alert dns any any -> any any (msg:"DNS query for evilcorp.com"; dns_query; content:"evilcorp.com"; sid:100002; rev:1;)**
     ➔ produciamo alert per qualsiasi tipo di richiesta DNS
   - **alert ip any any -> evilcorp.com any (msg:"Request for evilcorp.com"; sid:100003; rev:1;)**
     ->Alert a qualsiasi richiesta ip a evilcorp.com

3) È stato rilasciato un file pcap esame_10_febbraio_2023.pcapng, il file rappresenta il tracciato di traffico di diversi tentativi di autenticazione su di un noto protocollo. Compito dello studente è identificare:

-- Protocollo del traffico

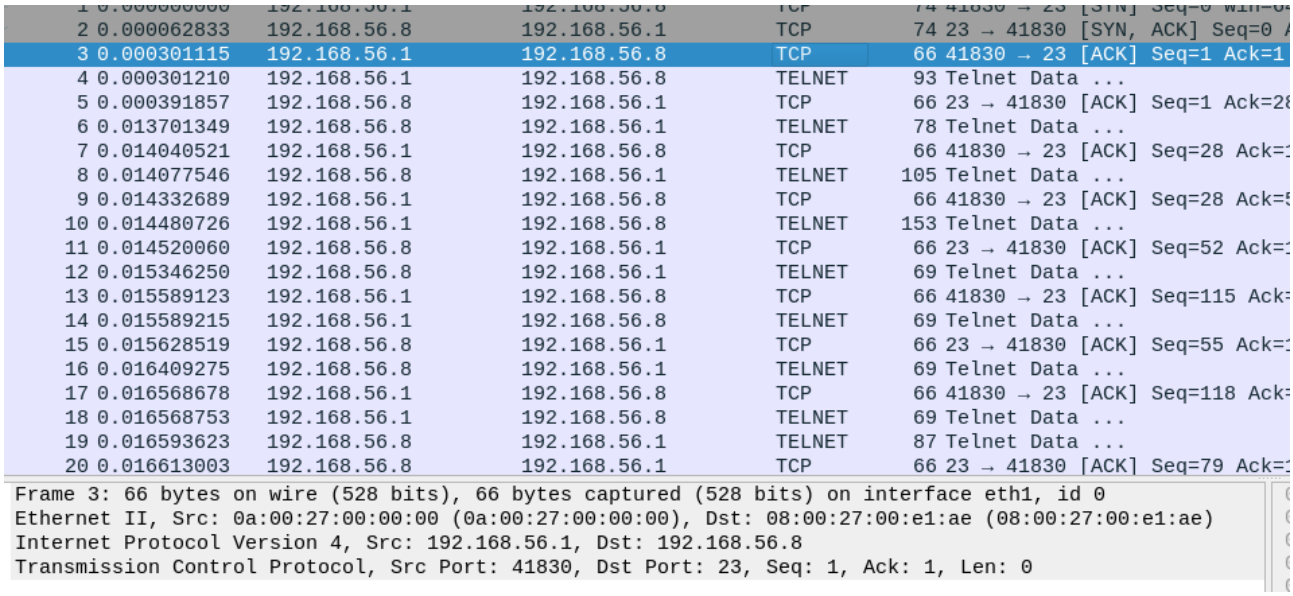
-- I 2 Indirizzi Ip in gioco (considerate che ovviamente il traffico è in due direzioni)

-- La/Le porte del protocollo in gioco

Hint: Avete un tool valido per recuperare queste informazioni

Soluzione:

Apriamo in wireshark il file e vediamo il traffico



2	0.000062833	192.168.56.8	192.168.56.1	TCP	74 23 → 41830 [SYN, ACK] Seq=0
3	0.000301115	192.168.56.1	192.168.56.8	TCP	66 41830 → 23 [ACK] Seq=1 Ack=1
4	0.000301210	192.168.56.1	192.168.56.8	TELNET	93 Telnet Data ...
5	0.000391857	192.168.56.8	192.168.56.1	TCP	66 23 → 41830 [ACK] Seq=1 Ack=28
6	0.013701349	192.168.56.8	192.168.56.1	TELNET	78 Telnet Data ...
7	0.014040521	192.168.56.1	192.168.56.8	TCP	66 41830 → 23 [ACK] Seq=28 Ack=
8	0.014077546	192.168.56.8	192.168.56.1	TELNET	105 Telnet Data ...
9	0.014332689	192.168.56.1	192.168.56.8	TCP	66 41830 → 23 [ACK] Seq=28 Ack=
10	0.014480726	192.168.56.1	192.168.56.8	TELNET	153 Telnet Data ...
11	0.014520060	192.168.56.8	192.168.56.1	TCP	66 23 → 41830 [ACK] Seq=52 Ack=
12	0.015346250	192.168.56.8	192.168.56.1	TELNET	69 Telnet Data ...
13	0.015589123	192.168.56.1	192.168.56.8	TCP	66 41830 → 23 [ACK] Seq=115 Ack=
14	0.015589215	192.168.56.1	192.168.56.8	TELNET	69 Telnet Data ...
15	0.015628519	192.168.56.8	192.168.56.1	TCP	66 23 → 41830 [ACK] Seq=55 Ack=
16	0.016409275	192.168.56.8	192.168.56.1	TELNET	69 Telnet Data ...
17	0.016568678	192.168.56.1	192.168.56.8	TCP	66 41830 → 23 [ACK] Seq=118 Ack=
18	0.016568753	192.168.56.1	192.168.56.8	TELNET	69 Telnet Data ...
19	0.016593623	192.168.56.8	192.168.56.1	TELNET	87 Telnet Data ...
20	0.016613003	192.168.56.8	192.168.56.1	TCP	66 23 → 41830 [ACK] Seq=79 Ack=

```
Frame 3: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface eth1, id 0
Ethernet II, Src: 0a:00:27:00:00:00 (0a:00:27:00:00:00), Dst: 08:00:27:00:e1:ae (08:00:27:00:e1:ae)
Internet Protocol Version 4, Src: 192.168.56.1, Dst: 192.168.56.8
Transmission Control Protocol, Src Port: 41830, Dst Port: 23, Seq: 1, Ack: 1, Len: 0
```

Vediamo subito che gli indirizzi coinvolti sono 192.168.56.1 e 192.168.56.8, e la porta dove si tenta di accedere è la 23.

L'alert da creare è quindi: **alert tcp 192.168.56.1 41852 -> 192.168.56.8 23 (msg:"Flag detected"; flow:from_client; flowbits:set,logged_in; content:"sec:sec"; sid: 100005; rev:1;)**

Abilitando l'opzione paylaod-printable in /etc/suricata/suricata.yaml si ottiene in /var/log/suricata/eve.json il payload dei pacchetti catturati. Questi eve log verranno esaminati per recuperare la flag.

DA PROVARE

ESERCIZIO 25 GIUGNO 2021

1. Scrivere una regola suricata in modalità alert per il traffico icmp SOLTANTO IN ENTRATA sulla rete 192.168.56.X
2. Scrivere una regola suricata in modalità alert per qualsiasi richiesta a evilcorp.com. Nota bene NON è possibile utilizzare il protocollo http o la porta 80 per creare questa Regola
3. Ricavare da questo file di tracciato di traffico nc_esame_25_giugno.pcapng le seguenti informazioni: protocollo del traffico (non è ssh), i due indirizzi ip in gioco (traffico è in due durezioni), le porte del protocollo in gioco. Successivamente lo studente deve scrivere una (o più di una se necessario) regola suricata in modalità alert per il traffico del protocollo in questione, specificandone gli IP trovati precedentemente.

alert icmp any any -> $HOME_NET any (msg:"Ping detected"; itype:8; sid:1000477; rev:1;)

in /etc/suricata/suricata.yaml è stata modificata la variabile HOME_NET in

HOME_NET: "[192.168.56.0/24]"

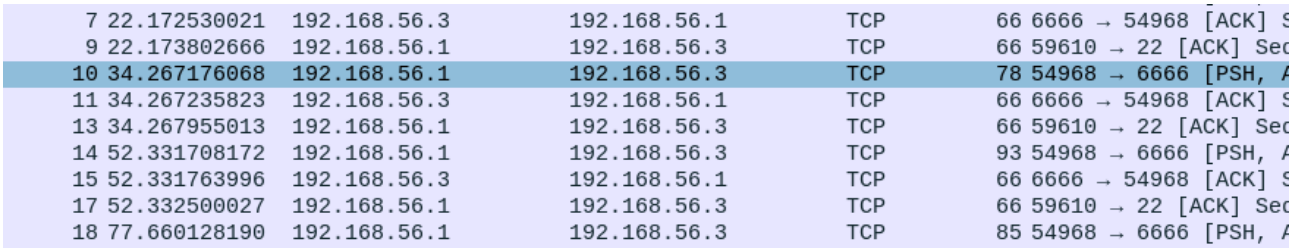
2

alert dns any any -> any any (msg:"DNS Query for evilcorp.com; dns_query; content:"evilcorp.com"; sid:1234; rev:1;)

e/o

alert ip any any -> evilcorp.com any (msg:"SURICATA TRAFFIC-ID: evilcorp.com"; sid:300000001; rev:1;)

3



7	22.172530021	192.168.56.3	192.168.56.1	TCP	66 6666 → 54968 [ACK] S
9	22.173802666	192.168.56.1	192.168.56.3	TCP	66 59610 → 22 [ACK] Seq
10	34.267176068	192.168.56.1	192.168.56.3	TCP	78 54968 → 6666 [PSH, A
11	34.267235823	192.168.56.3	192.168.56.1	TCP	66 6666 → 54968 [ACK] S
13	34.267955013	192.168.56.1	192.168.56.3	TCP	66 59610 → 22 [ACK] Seq
14	52.331708172	192.168.56.1	192.168.56.3	TCP	93 54968 → 6666 [PSH, A
15	52.331763996	192.168.56.3	192.168.56.1	TCP	66 6666 → 54968 [ACK] S
17	52.332500027	192.168.56.1	192.168.56.3	TCP	66 59610 → 22 [ACK] Seq
18	77.660128190	192.168.56.1	192.168.56.3	TCP	85 54968 → 6666 [PSH, A

gli Ip in gioco erano 192.168.56.1 e 192.168.56.3, la porta la 6666

quindi la regola per loggare il traffico era semplicemente

alert tcp 192.168.56.1 any -> 192.168.56.3 6666 (msg:"Flag Detected"; sid:30000006; rev 1;)

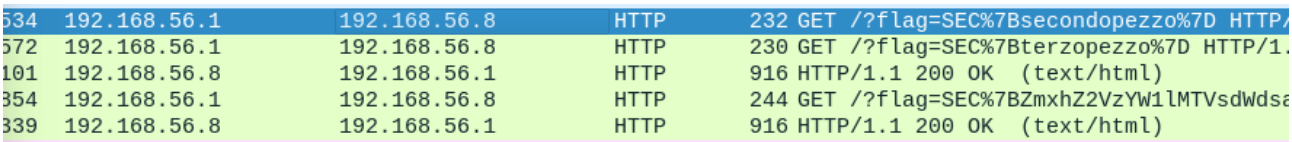
Avendo abilitato l'opzione payload-printable in /etc/suricata/suricata.yaml si ottiene in /var/log/suricata/eve.json si ha il payload dei pacchetti catturati; gli eve log devono esser esaminati per recupeare la flag che era spezzettata su più pacchetti, con risultato finale

SEC{this_is_the_first_part_this_is_the_second_part_third_part}

ESERCIZIO 15 LUGLIO 2022

Estraiamo il file .pcapng e lo analizziamo con wireshark

Vediamo delle richieste http GET con delle flag



534	192.168.56.1	192.168.56.8	HTTP	232	GET /?flag=SEC%7Bsecondopezzo%7D HTTP/
572	192.168.56.1	192.168.56.8	HTTP	230	GET /?flag=SEC%7Bterzopezzo%7D HTTP/1.
101	192.168.56.8	192.168.56.1	HTTP	916	HTTP/1.1 200 OK (text/html)
354	192.168.56.1	192.168.56.8	HTTP	244	GET /?flag=SEC%7BZmxhZ2VzYW1lMTVsdWdsa
339	192.168.56.8	192.168.56.1	HTTP	916	HTTP/1.1 200 OK (text/html)

Vanno da 192.168.56.1 (porta variabile) a 192.168.56.8 (porta 80)

Andiamo a creare una regola

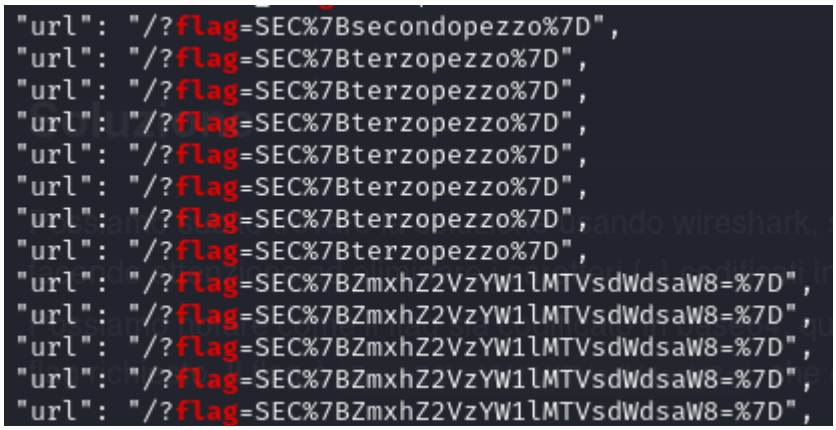
Avevo fatto una regola specifica ma alla fine essendo l'unico traffico http del tracciato bastava una regola generica su http

alert http any any -> any any (msg:"Flag_fragm"; sid:300000301; rev:1;)

SERVE ATTIVARE ANCHE L'OPZIONE METADATA E QUELLE PER HTTP DEL FILE DI CONFIGURAZIONE

Andiamo ad usare suricata con questo comando

```
tail -f /var/log/suricata/eve.json | jq
'select(.event_type="alert")' | grep flag
```



Vediamo che la flag è protetta quindi dobbiamo renderla leggibile

I caratteri { e } sono codificati come %7B e %7D

Vediamo se è base64 con il comando

```
echo 'ZmxhZ2VzYW1lMTVsdWdsaW8=' | base64 -d
```

e abbiamo la flag