

Projeto 2 Segurança e Confiabilidade 22/23

Parte II: snort

Grupo SegC-003

Regras Definidas

1. Deve ser gerado um alerta para a consola quando forem recebidas na máquina segura 5 ou mais ligações TCP para portos inferiores a 1024 durante um intervalo de um minuto (pode indicar um varrimento de portos) (NOTA: neste intervalo de um minuto deve ser gerado apenas um alarme, qualquer que seja a máquina que inicia as ligações, i.e., as ligações não têm todas de ter origem na mesma máquina).

Para esta situação definimos a seguinte regra snort:

```
alert tcp any any -> 10.101.204.4 0:1023 (msg: "Possível varredura de
portas"; threshold: type threshold, track by_src, count 5, seconds
60; sid:10001; rev:1;)
```

Esta regra tem como objetivo detetar uma "Possível varredura de portas". Específicamos de ínicio que esta regra se aplica ao tráfego TCP de de qualquer endereço e porta de origem para o endereço da nossa máquina segura (MServer) e portas de destino entre 0 a 1023. De seguida define a mensagem de alerta e depois configura o que acionará o alerta. Neste caso temos que 5 conexões no intervalo de 60 segundos irá fazer com surja o alerta.

Testagem

Ora para conseguirmos testar o funcionamento desta regra podemos simplesmente usar conexões SSH de 5 máquinas diferentes para o nosso MServer todas dentro de 1 minuto. Como vemos abaixo, com outras máquinas a ligarem-se ao MServer em menos de 1 minuto é imprimida a mensagem e informação sobre o sucedido.

```
MServer: 10.101.204.4/8

Og
Verifying Preprocessor Configurations!

[ Port Based Pattern Matching Memory ]
pcap DMQ configured to passive MServer-eth0".

Reload thread starting...
Reload thread starting...
Reload thread started, thread 0x7fc19a684700 (9394)

Decoding Ethernet

--== Initialization Complete ==--

Of Of Office Memory Offic
```

 Com o snort a correr na nossa máquina segura vamos então fazer as ligações para ver se a mensagem é impressa

```
Werifying Preprocessor Configurations!

[ Port Based Pattern Matching Memory ]
peap DRQ configured to passive.
Requiring network traffic from "MServer-eth0".
Reload thread starting...
Reload thread started, thread 0x7fba9c6b4700 (10090)
Becoding Ethernet

--== Initialization Complete ==--

-** Snort! (*-

o' )** Version 2.9.6.0 GRE (Build 47)

By Martin Roesch & The Snort Team; http://www.snort.org/snort/snort-team

Copyright (C) 2014 Cisco and/or its affiliates. All rights reserved.
Copyright (C) 1938-2013 Sourcefire, Inc., et al.
Using libpeap version 1.5.3

Using PCRE version; 8.31 2012-07-06

Using ZLIB version; 8.31 2012-07-06

Using 2LIB version; 1.2.8

Commencing packet processing (pid=10081)
05/26-12:07;44,848371 [**] [1:10001:1] Possivel varredura de portas [**] [Prior ittg: 0] (TCP) 10.121.72.23:50324 -> 10.101.204.4:22
```

2. Conseguimos ver que depois de efeutadas as ligações SSH o alerta aparece.

2. Deve ser gerado um alerta para a consola quando for detetado tráfego de rede que indique um possível ataque causado pela aplicação NoTintol. Esta situação deve ser diferenciada de um acesso considerado normal a partir de um cliente Tintolmarket (considera-se como acesso normal a ocorrência de até 3 ligações/tentativas de ligação num intervalo de 15 segundos) e que, portanto, não deve gerar alarme. Deve ocorrer no máximo um alerta a cada 15 segundos, caso a ação causada pela aplicação NoTintol persista. A aplicação NoTintol pode ser executada, por exemplo, na máquina Outsider e deve ser iniciada com os seguintes parâmetros: java NoTintol <IP do TintolMarketServer> <m_threads>. Onde n_threads representa o número de threads lançadas e deve ser igual a 2000.

Para esta situação definimos a seguinte regra snort:

```
alert tcp any any -> <IP do TintolMarketServer> <Porto do
TintolMarketServer> (msg: "Possível ataque NoTintol"; threshold: type
limit, track by_src, count 3, seconds 15; sid:10002; rev:1;)
```

Esta regra permite detetar ataques do **NoTintol**, assumindo que um cliente normal tem o limite de 3 tentativas de ligação num intervalo de 15 segundos. Isto é definido pelo valor de *count*(3) e pelo valor de *seconds*(15).

Testagem

Para testar esta regra vamos correr o **NoTintol** na máquina *Outsider* do sistema. Iremos executar com os seguintes parâmetros: java NoTintol 10.101.204.4 12345 2000. Se a regra estiver bem definida iremos receber desde logo o aviso do ataque na consola.

```
MServer: 10.101.204.4/8

Og
Verifying Preprocessor Configurations!

[Port Based Pattern Matching Memory ]
peap IAR configured to passive.
Reload thread starting...
Reload thread started, thread 0x7fdf0c225700 (10645)

IBecoding Ethernet

---= Initialization Complete ==-

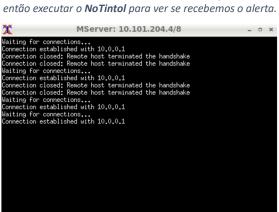
-** Snort! (*-

o' )"
Version 2.9.5,0 GRE (Build 47)
By Martin Roesch & The Snort Team; http://www.snort.org/snort/snort-team

Copyright (C) 2014 Cisco and/or its affiliates. All rights reserved.
Copyright (C) 2015 Sourcefire. Inc., et al.
Using libpeap version 1.5.3
Using PCRE version; 8,31 2012-07-06
Using ZLIB version; 1,2.8

Commencing packet processing (pid=10636)
```

 Com o snort a correr na nossa máquina segura vamos então executar o NoTintol para ver se recebemos o alerta.



5. Estado do Servidor depois da execução do **NoTintol**



4. Servidor iniciado na nossa máquina segura.

```
MServer: 10.101.204.4/8

cap INO configured to passive.

Requiring network traffic from "MServer-etho".

Reload thread starting.

Reload thread started, thread 0x7fe43aeea700 (20093)

Becoding Ethernet

--== Initialization Complete ==--

-*> Snortl (*-

o* )* Version 2.9.6.0 GRE (Build 47)

Wersion 2.9.6.0 GRE (Build 47)

Was ylartin Rosech & The Snort Team; http://www.snort.org/snort/snort-team

Copyright (C) 2014 Cisco and/or its affiliates. All rights reserved.

Copyright (C) 1939-2013 Sourcefire, Inc., et al.

Using PDCR version: 8.31 2012-07-06

Using ZLIB version 1.2.8

Commencing packet processing (pid=20084)

05/26-12:36:00.638693 [**] [1:10002:2] Possivel ataque NoTintol [**] [Priority: 0) [TDP] 10.0.0.147352 -> 10.101.04.412345

05/26-12:36:00.638653 [**] [1:10002:2] Possivel ataque NoTintol [**] [Priority: 0) [TDP] 10.0.0.147368 -> 10.101.04.412345

05/26-12:36:00.638653 [**] [1:10002:2] Possivel ataque NoTintol [**] [Priority: 0) [TDP] 10.0.0.147368 -> 10.101.04.412345

05/26-12:36:00.638653 [**] [1:10002:2] Possivel ataque NoTintol [**] [Priority: 0) [**]
```

6. O snort imprimiu os alertas, logo a regra funciona

Efeitos do NoTintol no TintolmarketServer

Para vermos as diferenças de recursos consumidos pelas duas aplicações iremos usar um filtro do comando *top*: top -b -n 1 -c | grep mininet | grep java

Com este filtro vemos o seguinte durante a execução das duas aplicações:

```
MServer: 10.101.204.4/8
mininet@mininet-vm:"/Desktop$ top -b -n 1 -c | grep mininet | grep java
23796 mininet 20 0 2511380 68860 19572 S 13.0 6.8 0:06.64 java
23812 mininet 20 0 4381888 85672 18288 S 13.0 8.4 0:05.67 java |
23796 mininet
23812 mininet
mininet@mininet
                                 /Desktop$ top -b -n 1 -c | grep
0 2521660 69784 19572 S 19.5
0 4381888 85936 18288 S 13.0
                                                                                      mininet | grep
6.9 0:06.93
8.5 0:05.85
  3796 mininet
3812 mininet
 ininet@mininet
         mininet
                          mininet@mininet
                                                                                                                       NoTint+
nininet@mininet
                          vm:~/Desktop$ top -b -n 1
23796 mininet
23812 mininet
                         20
20
                                 0 2529884
0 4381888
                                                                                                                       -jar T+
NoTint+
```

7. Vários usos do filtro no comando **top**. Temos que o processor 23812 é o **TintolMarketServer** e o 23796 é o **NoTintol**

Como fica fácil de concluir, com a execução do **NoTintol** este consome muito mais recursos quer de memória (virtual e RAM) quer a percentagem de uso do CPU. Isto pode ser devido ao facto de estamos a executar a aplicação com um grande número de threads e cada *thread* destas consome recursos.

Por outro lado também verificamos que o **TintolmarketServer** usa uma grande percentagem de memória, isto deverá ser pelo facto de o **NoTintol** colocar uma elevada carga de trabalho na aplicação causada pelo elevado número de solicitações.