

Sessão 5: Sistema de detecção/prevenção de intrusos



Todas as atividades desta sessão serão realizadas na máquina virtual *FWGW1-G*, com pequenas exceções destacadas no enunciado de cada exercício.

1) Instalação do Snort

1. A seção 1.5 do manual oficial do Snort, *Packet Acquisition*, alerta para o fato que duas características de placas de rede e de processamento do kernel Linux podem afetar negativamente o funcionamento do IDS: LRO (*large receive offload*) e GRO (*generic receive offload*). Em particular, o fato de que as placas de rede podem remontar pacotes antes do processamento do kernel pode ser problemático, pois o Snort trunca pacotes maiores que o *snaplen* de 1518 bytes; em adição a isso, essas *features* podem causar problemas com a remontagem de fluxo orientada a alvo [1] do Snort.

Na máquina *FWGW1-G*, instale o pacote **ethtool** e desative as *features* **lro** e **gro** da interface **enp0s3**. Se houver algum erro desativando as características, não se preocupe; siga para o próximo passo.

```
# hostname
FWGW1-A
```

```
# apt-get install ethtool
```

```
# ethtool -K enp0s3 gro off
# ethtool -K enp0s3 lro off
Cannot change large-receive-offload
```

2. Agora, vamos instalar o Snort. Execute:

```
# apt-get install snort
```

Durante a configuração do pacote, responda as perguntas como se segue. Se sua máquina for do grupo **B**, customize as faixas de endereços IP mostradas na tabela.

Tabela 1. Configurações do Snort durante a instalação

| Pergunta | Parâmetro |
|---------------------------------------|---------------------------|
| Interface(s) que o Snort deve escutar | enp0s3 |
| Gama de endereços para a rede local | 172.16.1.0/24,10.1.1.0/24 |

O arquivo de configuração principal do Snort é o `/etc/snort/snort.conf`. No Debian, em particular, há também o arquivo `/etc/snort/snort.debian.conf` que define algumas variáveis em particular, como mostrado abaixo:

```
# cat /etc/snort/snort.debian.conf | grep -v '^#'  
  
DEBIAN_SNORT_STARTUP="boot"  
DEBIAN_SNORT_HOME_NET="172.16.1.0/24,10.1.1.0/24"  
DEBIAN_SNORT_OPTIONS=""  
DEBIAN_SNORT_INTERFACE="enp0s3"  
DEBIAN_SNORT_SEND_STATS="true"  
DEBIAN_SNORT_STATS_RCPT="root"  
DEBIAN_SNORT_STATS_THRESHOLD="1"
```

3. Teste o funcionamento do Snort.

```
# snort -V  
  
,,_      -*> Snort! <*-  
o" )~    Version 2.9.7.0 GRE (Build 149)  
'''     By Martin Roesch & The Snort Team: http://www.snort.org/contact#team  
         Copyright (C) 2014 Cisco and/or its affiliates. All rights reserved.  
         Copyright (C) 1998-2013 Sourcefire, Inc., et al.  
         Using libpcap version 1.8.1  
         Using PCRE version: 8.39 2016-06-14  
         Using ZLIB version: 1.2.8
```

2) Configuração inicial do Snort

1. Primeiramente, vamos desabilitar (via comentários) todas as regras padrão do Snort instaladas pelo gerenciador de pacotes. Iremos, em um passo futuro, usar o PuledPork para atualizar as regras pela Internet.

```
# sed -i 's/^\(include \$RULE_PATH.*\)/#\1/' /etc/snort/snort.conf
```

2. Descomente a linha que habilita regras customizadas locais, que usaremos em breve para testar o funcionamento do Snort.

```
# sed -i 's/^\#\(\include \$RULE_PATH/local.rules\)/\1/' /etc/snort/snort.conf
```

```
# grep '^include \$RULE_PATH/local.rules' /etc/snort/snort.conf  
include $RULE_PATH/local.rules
```

3. Remova a palavra-chave **nostamp** da saída de eventos do Snort, de forma que os arquivos de log sejam identificados pelo *timestamp* de criação do arquivo.

```
# sed -i 's/^(output unified2.*\ ) nostamp,\(.*\)/\1\2/'g /etc/snort/snort.conf
```

```
# grep '^output unified2' /etc/snort/snort.conf
output unified2: filename snort.log, limit 128, mpls_event_types, vlan_event_types
```

4. Teste o arquivo de configuração do Snort procurando por erros de sintaxe. Se tudo estiver correto, a penúltima linha deverá dizer **Snort successfully validated the configuration!**.

```
# snort -T -c /etc/snort/snort.conf
```

```
(...)
Snort successfully validated the configuration!
Snort exiting
```

5. Vamos criar uma regra customizada no Snort para testar se tudo está a contento. No arquivo **/etc/snort/rules/local.rules**, insira a linha:

```
alert icmp any any -> any any (msg:"ICMP packet from all, to all"; sid:10000001; rev:001;)
```

Esta regra irá simplesmente levantar um alerta se o Snort detectar um pacote ICMP vindo de qualquer IP, qualquer porta, para qualquer IP, qualquer porta.

6. Descubra o IP público da máquina *FWGW1-G*:

```
# ip a s enp0s3 | grep '^ *inet ' | awk '{ print $2 }'
192.168.29.103/24
```

Agora, vamos rodar o Snort em modo console e testar o funcionamento da regra.

```
# snort -A console -q -g snort -u snort -c /etc/snort/snort.conf -i enp0s3
```

Em sua máquina física, envie alguns pacotes ICMP para o IP público da máquina *FWGW1-G*:

```
C:\>ping 192.168.29.103
```

```
Pinging 192.168.29.103 with 32 bytes of data:
```

```
Request timed out.
```

```
Request timed out.
```

```
Request timed out.
```

```
Request timed out.
```

```
Ping statistics for 192.168.29.103:
```

```
Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),
```

De volta à máquina *FWGW1-G*, note que o Snort gerou registros para cada um dos pacotes recebidos, como esperado:

```
09/04-09:10:33.691493  [**] [1:10000001:1] ICMP packet from all, to all [**]
```

```
[Priority: 0] {ICMP} 192.168.29.102 -> 192.168.29.103
```

```
09/04-09:10:38.278164  [**] [1:10000001:1] ICMP packet from all, to all [**]
```

```
[Priority: 0] {ICMP} 192.168.29.102 -> 192.168.29.103
```

```
09/04-09:10:43.279523  [**] [1:10000001:1] ICMP packet from all, to all [**]
```

```
[Priority: 0] {ICMP} 192.168.29.102 -> 192.168.29.103
```

```
09/04-09:10:48.283261  [**] [1:10000001:1] ICMP packet from all, to all [**]
```

```
[Priority: 0] {ICMP} 192.168.29.102 -> 192.168.29.103
```

Observe, ainda, que os ICMP **echo-reply** enviados por sua máquina física não foram respondidos porque o firewall interno permite tráfego ICMP oriundo apenas das redes 172.16.1.0/24 e 10.1.1.0/24, como configurado na sessão 3.

```
# iptables -vn -L INPUT | grep ' prot\|icmp '
```

| pkts | bytes | target | prot | opt | in | out | source | destination |
|------|-------|--------|------|-----|----|-----|---------------|-------------|
| 1 | 84 | ACCEPT | icmp | -- | * | * | 172.16.1.0/24 | 0.0.0.0/0 |

```
icmp type 255
```

| | | | | | | | | |
|---|---|--------|------|----|---|---|-------------|-----------|
| 0 | 0 | ACCEPT | icmp | -- | * | * | 10.1.1.0/24 | 0.0.0.0/0 |
|---|---|--------|------|----|---|---|-------------|-----------|

```
icmp type 255
```

Finalize o Snort com CTRL+C, e comente a regra inserida no arquivo `/etc/snort/rules/local.rules`.

3) Configurando atualizações de regras de forma automática com o PuledPork

1. O programa PuledPork nos permite receber definições de regras atualizadas periodicamente pela Internet, sempre que novas vulnerabilidade e *exploits* forem descobertos e divulgados.

Primeiro, vamos instalar as dependências do PuledPork:

```
apt-get install git \
                libcrypt-ssleay-perl \
                liblwp-useragent-determined-perl
```

2. Crie o diretório `/root/src`, se não existir, e faça o download do código-fonte do PuledPork. Em seguida, copie seus binários e arquivos de configuração para os locais apropriados.

```
# mkdir ~/src
# cd ~/src
```

```
# git clone https://github.com/shirkdog/pulledpork.git
Cloning into 'pulledpork'...
remote: Counting objects: 1323, done.
remote: Total 1323 (delta 0), reused 0 (delta 0), pack-reused 1323
Receiving objects: 100% (1323/1323), 331.28 KiB | 343.00 KiB/s, done.
Resolving deltas: 100% (884/884), done.
Checking connectivity... done.
```

```
# cd pulledpork/
```

```
# cp pulledpork.pl /usr/local/bin/
# chmod +x /usr/local/bin/pulledpork.pl
```

```
# cp ./etc/*.conf /etc/snort
```

3. Crie os diretórios e arquivos de configuração padrão do PuledPork, vazios.

```
# mkdir /etc/snort/rules/iplists
# touch /etc/snort/rules/iplists/default.blacklist
```

4. Teste o funcionamento do PuledPork, verificando sua versão.

```
# pulledpork.pl -V
PulledPork v0.7.4 - Helping you protect your bitcoin wallet!
```

5. Vamos agora configurar o PuledPork. O primeiro passo é a obtenção de um *Oinkcode*, que é basicamente um número de registro com o [snort.org](https://www.snort.org) que nos permitirá o download de listas de regras geradas pela comunidade.

1. Acesse <https://www.snort.org/>, e clique em *Sign In* no canto superior direito.

2. Se você não possuir uma conta, clique em *Sign up*.
 3. Preencha os campos *Email* (use um email válido e acessível), *Password* e *Password confirmation*, marque a caixa *Agree to Snort license* e finalmente clique em *Sign up*.
 4. Acesse o e-mail informado no passo (3). Dentro de algum tempo, você deverá receber uma mensagem com o título *Confirmation instructions*. Abra-a e clique no link *Confirm my account*.
 5. Com a conta confirmada, faça login no site <https://www.snort.org/> usando os dados informados anteriormente.
 6. No canto superior direito da página, clique no seu e-mail cadastrado, logo ao lado do ícone de logout.
 7. Na nova página, clique no menu *Oinkcode*. Deverá aparecer uma *string* de cerca de 40 caracteres no centro da tela. Copie-a, pois a usaremos em seguida.
6. Com o *Oinkcode* em mãos, vamos configurar o PulledPork. No comando abaixo, substitua o valor **OINKCODE** no começo do comando pelo código que você copiou no item (7) do passo anterior. Em seguida, execute-o no terminal.

```
# oc="OINKCODE" ; sed -i "s/^\(rule_url=https:\\\\www\\.snort\\.org\\/reg-rules\\/|snortrules-snapshot\\.tar\\.gz|\\).*\\1${oc}/" /etc/snort/pulledpork.conf ; unset oc
```

Se tudo deu certo, você deverá ver seu *Oinkcode* ao final da linha de regras baixadas do site <https://www.snort.org>, como mostrado a seguir (nota: o *Oinkcode* abaixo é fictício):

```
# grep 'rule_url=https://www.snort.org/reg-rules' /etc/snort/pulledpork.conf
rule_url=https://www.snort.org/reg-rules/|snortrules-
snapshot.tar.gz|13eba036f37e80d0efb689c60af9e6daae810763
```

+rm / Substitua todas as instâncias de **/usr/local/etc** por **/etc**, e **/usr/local/lib** por **/usr/lib**, para refletir corretamente o diretório de armazenamento de configurações e bibliotecas do Snort:

```
# sed -i 's/\\usr\\/local\\/etc\\/etc/g' /etc/snort/pulledpork.conf
# sed -i 's/\\usr\\/local\\/lib\\/usr\\/lib/g' /etc/snort/pulledpork.conf
```

Corrija o local do binário do Snort, de **/usr/local/bin/snort** para o valor correto, que é **/usr/sbin/snort**:

```
# sed -i 's/\\usr\\/local\\/bin\\/snort\\/usr\\/sbin\\/snort/g'
/etc/snort/pulledpork.conf
```

Finalmente, falta substituir a distribuição-alvo padrão do PulledPork:

```
# sed -i 's/^\(distro=\).*\/1Debian-6-0/' /etc/snort/pulledpork.conf
```

```
# grep '^distro=' /etc/snort/pulledpork.conf
distro=Debian-6-0
```

7. Vamos testar as configurações do PuledPork, e fazer o download das listas de regras mais atualizadas.

```
# pulledpork.pl -c /etc/snort/pulledpork.conf -l
```

```
https://github.com/shirkydog/pulledpork

  -----
  \-----,\      )
  \---==\ \ /      PuledPork v0.7.4 - Helping you protect your bitcoin wallet!
  \---==\ \ /
  .-~~~~-.Y|\ \_   Copyright (C) 2009-2017 JJ Cummings, Michael Shirk
@_/_      / 66\_   and the PuledPork Team!
|   \   \   _(")
 \  /-| ||'--'  Rules give me wings!
  \_ \  \_ \
~~~~~

(...)

Rule Stats...
  New:-----34178
  Deleted:---0
  Enabled Rules:----10999
  Dropped Rules:----0
  Disabled Rules:---23179
  Total Rules:-----34178
IP Blacklist Stats...
  Total IPs:-----1382

Done
Please review /var/log/sid_changes.log for additional details
Fly Piggy Fly!
```

Se tudo deu certo, o PuledPork deve ter consolidado as regras baixadas no arquivo `/etc/snort/rules/snort.rules`. Verifique o tamanho e o número de linhas desse arquivo.

```
# du -sk /etc/snort/rules/snort.rules
18432   /etc/snort/rules/snort.rules
```

```
# wc -l /etc/snort/rules/snort.rules
38437 /etc/snort/rules/snort.rules
```

8. Finalmente, basta indicar ao Snort que esse arquivo seja usado em sua inicialização. Insira a linha `include $RULE_PATH/snort.rules` ao final do arquivo `/etc/snort/snort.conf`.

```
# echo 'include $RULE_PATH/snort.rules' >> /etc/snort/snort.conf
```

Pare todas as instâncias do Snort, e remova os arquivos de log antigos. Em seguida, inicie-o, e verifique seu uso de memória.

```
# systemctl stop snort
# ps auxwm | grep '^snort'
```

```
# rm /var/log/snort/snort.log*
```

```
# systemctl start snort
```

```
# ps -eo 'rss,comm' | grep 'snort$'
995976 snort
```

9. Para que as regras se mantenham atualizadas, é necessário atualizá-las periodicamente. Crie um novo arquivo no diretório `/etc/cron.daily` que atualize as regras diariamente, com o seguinte conteúdo:

```
#!/bin/sh

test -x /usr/local/bin/pulledpork.pl || exit 0
/usr/local/bin/pulledpork.pl -c /etc/snort/pulledpork.conf -l
```

Verifique que o usuário/grupo dono e permissões do arquivo estão corretos.

```
# chown root.root /etc/cron.daily/pulledpork
# chmod 0755 /etc/cron.daily/pulledpork
```

4) Processando arquivos de log do Snort com o Barnyard2

1. Note que, por padrão, o Snort está fazendo o log de eventos registrados no arquivo

/var/log/snort/snort.log:

```
# grep 'snort.log' /etc/snort/snort.conf | grep -v '^#'
output unified2: filename snort.log, limit 128, mpls_event_types, vlan_event_types
```

Este arquivo está no formato **unified2** que, como documentado no manual oficial do Snort (<http://manual-snort-org.s3-website-us-east-1.amazonaws.com/node21.html>), é um formato de log binário que permite ao Snort maior nível de performance ao registrar os eventos em disco. O problema, evidentemente, é que esse arquivo de log não é legível diretamente.

2. Iremos instalar e configurar o *Barnyard2* para lidar com esses arquivos de log. Como de costume, o primeiro passo é instalar as dependências do pacote:

```
# apt-get install autoconf \
                    build-essential \
                    libdaq-dev \
                    libdumbnet-dev \
                    libmariadb-dev \
                    libmariadb-dev-compat \
                    libpcap-dev \
                    libprelude-dev \
                    libtool \
                    mariadb-server
```

Em adição a isso, é necessário criar um link simbólico para a biblioteca **dumbnet.n**, como se segue:

```
# ln -s /usr/include/dumbnet.h /usr/include/dnet.h
```

```
# ldconfig
```

3. Agora, volte ao diretório de download de códigos-fonte (**/root/src**), baixe o Barnyard2, compile-o e instale:

```
# cd ~/src
```

```
# git clone https://github.com/firnsy/barnyard2.git
```

```
# cd barnyard2/
```

```
# autoreconf -fvi -I ./m4
```

```
# ./configure --with-mysql --with-mysql-libraries=/usr/lib/x86_64-linux-gnu
```

```
# make
```

```
# make install
```

4. Vamos agora proceder à configuração do Barnyard2. Primeiramente, vamos criar arquivos e diretórios padrão com as permissões corretas:

```
# touch /var/log/snort/barnyard2.waldo
```

```
# mkdir /var/log/barnyard2 /var/log/snort/archive
```

```
# chown snort.snort /var/log/barnyard2/ /var/log/snort/archive/  
/var/log/snort/barnyard2.waldo
```

Em seguida, crie e edite o arquivo de configuração `/etc/snort/barnyard2.conf`, com o seguinte conteúdo:

```
config sid_file:          /etc/snort/sid-msg.map  
config gen_file:          /etc/snort/gen-msg.map  
config reference_file:    /etc/snort/reference.config  
config classification_file: /etc/snort/classification.config  
  
config hostname:          localhost  
config interface:         enp0s3  
  
input unified2  
  
output alert_fast  
output database: log, mysql, user=snorby password=snorby dbname=snorby host  
=localhost
```

5) Visualizando eventos com o Snorby

1. Precisamos de um método conveniente para visualizar e tratar os eventos registrados pelo Snort. Iremos instalar e configurar o *Snorby*, uma aplicação web criada em *Ruby on Rails* para

auxiliar no trabalho de monitoramento de ferramentas IDS populares como o Snort, Suricata e Sagan. Como de costume, o primeiro passo é instalar as dependências do pacote:

```
# apt-get install --no-install-recommends \
    bundler \
    imagemagick \
    libmariadbclient-dev \
    libmariadbclient-dev-compat \
    libpq-dev \
    libreadline-dev \
    libssl-dev \
    libxml2-dev \
    libxslt1-dev \
    libyaml-dev \
    postgresql-server-dev-9.6 \
    ruby \
    ruby-dev \
    wkhtmltopdf \
    zlib1g-dev
```

2. Agora, volte ao diretório de download de códigos-fonte (`/root/src`), baixe o Snorby e instale suas dependências (no Ruby, conhecidas como *gems*):

```
# cd ~/src
```

```
# git clone https://github.com/Snorby/snorby.git
```

```
# cd snorby/
```

```
# bundle install
```

3. Vamos agora proceder à configuração do Snorby. Copie e renomeie os arquivos de configuração de conexão com o banco de dados (`config/database.yml`), editando-o como se segue:

```
# cp config/database.yml.example config/database.yml
```

```
# nano config/database.yml
(...)
```

```
# cat config/database.yml | grep '^ *username\|password' | grep -v '^#'
username: snorby
password: "snorby"
```

Faça o mesmo para a configuração principal do Snorby (`config/snorby_config.yml`), como mostrado abaixo:

```
# cp config/snorby_config.yml.example config/snorby_config.yml
```

```
# nano config/snorby_config.yml
(...)
```

```
# cat config/snorby_config.yml | grep '^
*baseuri\|domain\|wkhtmltopdf\|mailer_sender\|time_zone' | head -n5
baseuri: ''
domain: 'snorby.FWGW1-A.intnet'
wkhtmltopdf: /usr/bin/wkhtmltopdf
mailer_sender: 'snorby@FWGW1-A.intnet'
time_zone: 'America/Sao_Paulo'
```

Edite apenas as configurações da seção `production:` do arquivo `config/snorby_config.yml`. As demais seções (`development:` e `test:`) não serão usadas.

4. Vamos configurar o banco de dados. Primeiro, crie uma base de dados vazia e configure suas permissões:

```
# mysql -u root -e 'create database snorby'
```

```
# mysql -u root
```

```
MariaDB [(none)]> grant all on snorby.* to 'snorby'@'localhost' identified by
'snorby';
Query OK, 0 rows affected (0.00 sec)
```

```
MariaDB [(none)]> flush privileges;
Query OK, 0 rows affected (0.00 sec)
```

```
MariaDB [(none)]> quit
Bye
```

5. Agora, invoque o script de *setup* do Snorby para popular a base de dados:

```
# bundle exec rake snorby:setup
```

6. O Snorby irá rodar, por padrão, na porta TCP/3000. Como não há regra permitindo esse tipo de acesso direto ao firewall, crie-a:

```
# iptables -A INPUT -p tcp --dport 3000 -j ACCEPT
```

```
# iptables-save > /etc/iptables/rules.v4
```

6) Integração dos serviços com o sistema

Note que ainda não iniciamos nem o Barnyard2 nem o Snorby, recentemente instalados e configurados. Muito embora seja possível iniciá-los manualmente e gerenciar seus processos, isso rapidamente se torna bastante inconveniente à medida que o sistema aumenta em complexidade. Para facilitar a tarefa, iremos criar dois *scripts* de inicialização para os serviços junto ao **systemctl**. Siga os passos:

1. Crie um arquivo novo, **/etc/systemd/system/barnyard2.service**, com o seguinte conteúdo:

```
[Unit]
Description=Barnyard2 Snort log spooler
After=snort.service

[Service]
Type=simple
ExecStart=/usr/local/bin/barnyard2 -D -c /etc/snort/barnyard2.conf -d
/var/log/snort -w /var/log/snort/barnyard2.waldo -l /var/log/snort -a
/var/log/snort/archive -f snort.log -u snort -g snort
Restart=always

[Install]
WantedBy=multi-user.target
```

2. Faça o mesmo para o arquivo **/etc/systemd/system/snorby.service**:

```
[Unit]
Description=Snorby Snort web monitoring
Requires=barnyard2.service

[Service]
Type=forking
PIDFile=/root/src/snorby/tmp/pids/server.pid
WorkingDirectory=/root/src/snorby
ExecStart=/bin/bash -lc 'bundle exec rails server -e production -d'

[Install]
WantedBy=multi-user.target
```

3. Recarregue a lista de serviços do sistema com o comando:

```
# systemctl daemon-reload
```

4. Agora, inicie o serviço do Barnyard2, como se segue:

```
# systemctl start barnyard2.service
```

Em seguida, monitore seu início com o comando:

```
# journalctl -u barnyard2.service -f
-- Logs begin at Mon 2018-10-15 15:59:29 -03.
out 15 16:22:58 FWGW1-A systemd[1]: Started Barnyard2 Snort log spooler.
out 15 16:22:58 FWGW1-A barnyard2[3739]: Running in Continuous mode
(...)
```

O Barnyard pode demorar um certo tempo para iniciar, até cerca de três minutos. Observe os logs até que uma mensagem parecida com a que se segue apareça na tela:

```
out 15 16:25:53 FWGW1-A barnyard2[3739]: Opened spool file
'/var/log/snort/snort.log.1539630818'
out 15 16:25:53 FWGW1-A barnyard2[3739]: Waiting for new data
```

Feito isso, encerre o monitoramento dos registros com **CTRL + C**—o Barnyard2 iniciou corretamente.

5. Inicie o Snorby como mostrado abaixo:

```
# systemctl start snorby.service
```

6. Usando o navegador web em sua máquina física, acesse a URL <http://FWGW1-G:3000>,

substituindo a palavra **FWGW1-6** pelo endereço IP externo do seu firewall (atrelado à interface **enp0s3**). Você verá a tela de login do Snorby; acesse com o usuário **snorby@example.com**, e senha **snorby**.

Você verá o *dashboard* principal do Snorby, como mostrado na imagem a seguir:

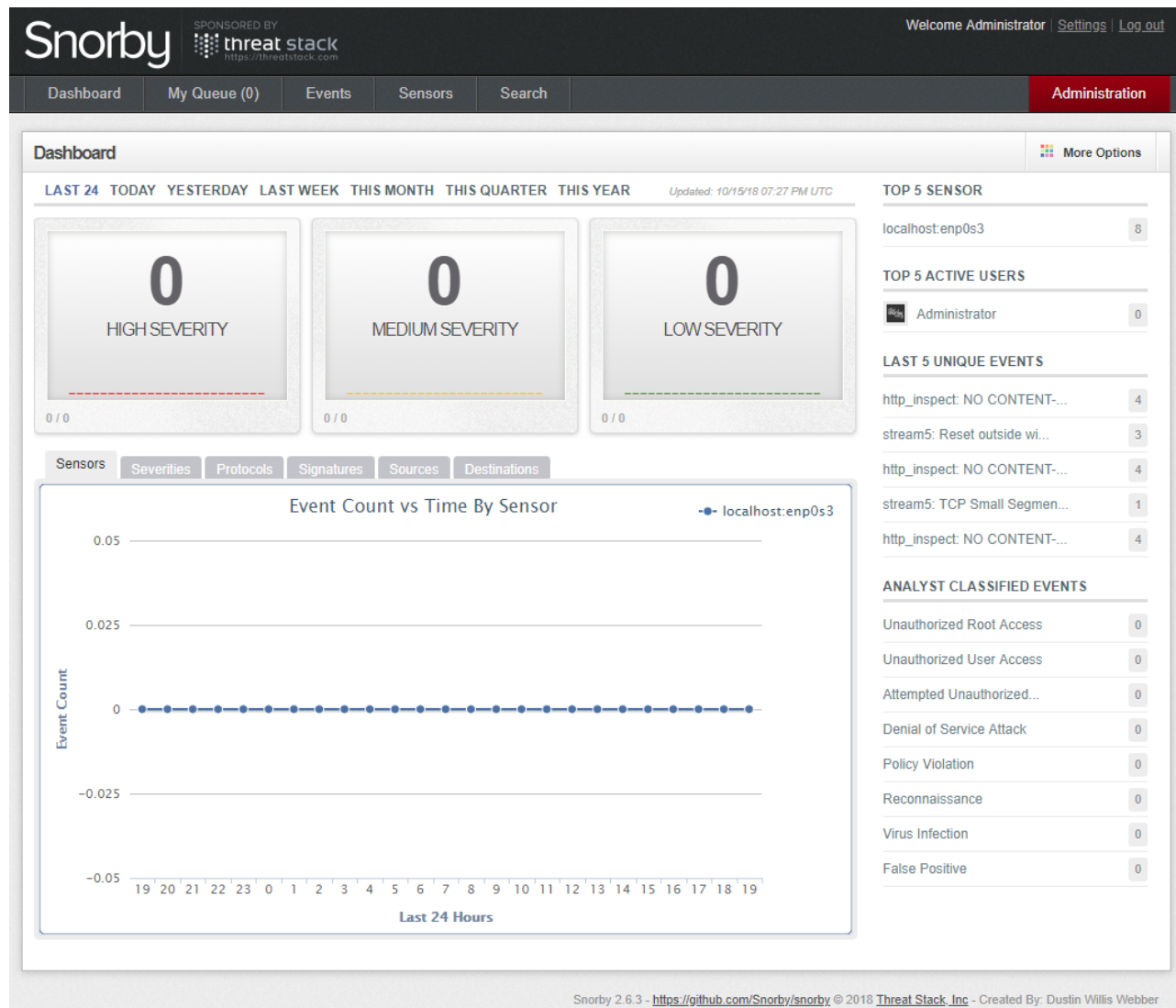


Figura 1. Dashboard do Snorby

7) Gerando alertas para o IDS

1. Vamos gerar alguns alertas para testar o funcionamento da solução. No Virtualbox, conecte a placa de rede da máquina *KaliLinux-G* à rede externa, acessando *Settings > Network 1 > Adapter 1 > Attached to: Bridged Adapter*. Em seguida, altere a configuração da interface para DHCP editando o arquivo `/etc/network/interfaces`:

```
# hostname
KaliLinux-A
```

```
# nano /etc/network/interfaces  
(...)
```

```
# cat /etc/network/interfaces | grep '^iface eth0'  
iface eth0 inet dhcp
```

```
# systemctl restart networking
```

Ao final do processo, sua máquina *KaliLinux-G* deverá ter um endereço IP da rede externa, como:

```
# ip a s eth0 | grep '^ *inet ' | awk '{print $2}'  
192.168.29.105/24
```

2. Execute alguns comandos para gerar tráfego suspeito na direção da máquina *FWGW1-G*. Nos exemplos abaixo, o IP público **192.168.29.103** está sendo usado como o endereço da interface **enp0s3** da máquina *FWGW1-G*.

Primeiro, verifique que a máquina *LinServer-G* está ligada e em seguida rode o **nikto**, um *scanner* de servidores web:

```
# nikto -h 192.168.29.103
```

Use o **nmap** para realizar um *scan* do tipo *Xmas tree* na máquina *FWGW1-G*:

```
# nmap -sX 192.168.29.103 -p1-65535
```

Use o **nmap** com a opção de fragmentar pacotes para realizar um *scan* na máquina *FWGW1-G*:

```
# nmap -Pn -sS -A -f 192.168.29.103
```

Existem vários outros testes que podem ser realizados para gerar tráfego suspeito e testar a eficácia do IDS. A página <https://www.aldeid.com/wiki/Suricata-vs-snort> contém uma excelente lista de ataques e ferramentas sugeridos para realizar uma inspeção nesse sentido.

3. Vamos ver o que foi observado pelo Snort, através do Snorby. Acessando a aba *Events*, podemos observar que vários eventos foram gerados ao rodar a ferramenta **nikto**, como mostrado na imagem a seguir:

Listing Sessions (12 unique unclassified sessions)

| Sev. | Sensor | Source IP | Destination IP | Event Signature | Timestamp | Sessions |
|------|------------------|----------------|----------------|---|-----------|----------|
| 2 | localhost:enp0s3 | 192.168.29.106 | 192.168.29.108 | stream5: Bad segment, overlap adjusted size less than/equal 0 | 7:28 PM | 6 |
| 3 | localhost:enp0s3 | 192.168.29.105 | 192.168.29.108 | http_inspect: UNKNOWN METHOD | 7:27 PM | 4 |
| 3 | localhost:enp0s3 | 192.168.29.105 | 192.168.29.108 | http_inspect: UNESCAPED SPACE IN HTTP URI | 7:27 PM | 13 |
| 2 | localhost:enp0s3 | 192.168.29.105 | 192.168.29.108 | http_inspect: LONG HEADER | 7:27 PM | 3 |
| 2 | localhost:enp0s3 | 192.168.29.105 | 192.168.29.108 | stream5: TCP Small Segment Threshold Exceeded | 7:27 PM | 7 |
| 2 | localhost:enp0s3 | 192.168.29.105 | 192.168.29.108 | http_inspect: NON-RFC DEFINED CHAR | 7:27 PM | 5 |
| 3 | localhost:enp0s3 | 192.168.29.108 | 192.168.29.105 | http_inspect: POST W/O CONTENT-LENGTH OR CHUNKS | 7:27 PM | 1 |
| 3 | localhost:enp0s3 | 192.168.29.108 | 192.168.29.105 | http_inspect: NO CONTENT-LENGTH OR TRANSFER-ENCO... | 7:27 PM | 17 |
| 3 | localhost:enp0s3 | 200.236.31.3 | 192.168.29.108 | http_inspect: NO CONTENT-LENGTH OR TRANSFER-ENCO... | 7:17 PM | 3 |
| 3 | localhost:enp0s3 | 151.101.92.204 | 192.168.29.108 | http_inspect: NO CONTENT-LENGTH OR TRANSFER-ENCO... | 7:17 PM | 1 |
| 2 | localhost:enp0s3 | 192.168.29.108 | 192.30.253.113 | stream5: Reset outside window | 7:14 PM | 3 |
| 2 | localhost:enp0s3 | 200.236.31.3 | 192.168.29.108 | stream5: TCP Small Segment Threshold Exceeded | 7:13 PM | 1 |

Snorby 2.6.3 - <https://github.com/Snorby/snorby> © 2018 Threat Stack, Inc - Created By: Dustin Willis Webber

Figura 2. Listagem de eventos no Snorby

4. É possível clicar em um evento para explorar mais detalhes sobre o mesmo, incluindo hosts de origem/destino do ataque, assinatura que causou o *match* no pacote, e até mesmo detalhes de seu *header* e *payload*. A imagem a seguir ilustra a inspeção de um evento de baixa criticidade, identificado como uma inspeção HTTP usando método desconhecido:

Listing Sessions (12 unique unclassified sessions)

| Sev. | Sensor | Source IP | Destination IP | Event Signature | Timestamp | Sessions |
|------|------------------|----------------|----------------|------------------------------|-----------|----------|
| 3 | localhost:enp0s3 | 192.168.29.105 | 192.168.29.108 | http_inspect: UNKNOWN METHOD | 7:33 PM | 6 |

IP Header Information

| Source | Destination | Ver | Hlen | Tos | Len | ID | Flags | Off | TTL | Proto | Csum |
|----------------|----------------|-----|------|-----|-----|-------|-------|-----|-----|-------|-------|
| 192.168.29.105 | 192.168.29.108 | 4 | 5 | 0 | 205 | 51147 | 0 | 0 | 64 | 6 | 46649 |

Signature Information

| Generator ID | Sig. ID | Sig. Revision | Activity (5/4595) | Category | Sig Info |
|--------------|---------|---------------|-------------------|----------|------------------------------------|
| 119 | 31 | 1 | 0.11% | unknown | Query Signature Database View Rule |

TCP Header Information

| Src Port | Dst Port | Seq | Ack | Off | Res | Flags | Win | Csum | URP |
|----------|----------|------------|------------|-----|-----|-------|-----|-------|-----|
| 57230 | 80 | 2645660303 | 1601495387 | 8 | 0 | 24 | 229 | 11096 | 0 |

Payload

```

000000: 52 44 56 55 20 2f 20 48 54 54 50 2f 31 2e 31 0d 0a 43 6f 6e 6e 65 63 74 69 6f  RDVU./..HTTP/1.1..Connectio
000010: 6e 3a 20 63 6c 6f 73 65 0d 0a 48 6f 73 74 3a 20 31 39 32 2e 31 36 38 2e 32 39  n:.close..Host:192.168.29
000020: 2e 31 30 38 0d 0a 55 73 65 72 2d 41 67 65 6e 74 3a 20 4d 6f 7a 69 6c 6c 61 2f  .108..User-Agent:Mozilla/
000030: 35 2e 30 20 28 63 6f 6d 70 61 74 69 62 6c 65 3b 20 4e 6d 61 70 20 53 63 72 69  5.0.(compatible;.Nmap.Scri
000040: 70 74 69 6e 67 20 45 6e 67 69 6e 65 3b 20 68 74 74 70 73 3a 2f 2f 6e 6d 61 70  pting.Engine;.https://nmap
000050: 2e 6f 72 67 2f 62 6f 6f 6b 2f 6e 73 65 2e 68 74 6d 6c 29 0d 0a 0d 0a  .org/book/nse.html)....
  
```

Notes

This event currently has zero notes - You can add a note by clicking the button below.

Add A Note To This Event

Figura 3. Listagem de eventos no Snorby

5. Retorne a máquina *KaliLinux-G* para a rede **DMZ**, restaurando suas configurações de rede originais.



Nesta sessão fizemos a instalação de todas as ferramentas (Snort, PulledPork, Barnyard2 e Snorby) dentro da mesma máquina, *FWGW1-G*. É muito comum, no entanto, não sobrecarregar a máquina responsável pela tarefa de IDS/IPS também com a tarefa de operação do banco de dados e do *frontend* web, delegando essas tarefas a uma outra máquina.

Nesse cenário, instalaríamos o Snort, PulledPork e Barnyard2 na máquina *FWGW1-G*, e o Barnyard2, Snorby e a base de dados MariaDB em outra máquina (dedicada e segmentada para este fim). A máquina *FWGW1-G* se conectaria à base de dados remota para escrever os registros de eventos, sem ser necessário instalar nela uma série de dependências potencialmente perigosas, como o Ruby, compiladores e bibliotecas de desenvolvimento.

A configuração desse cenário mais complexo fica a cargo do leitor, como um exercício avançado.

Referências

- [1] Novak, J. e Sturges, S. (2007). Target-Based TCP Stream Reassembly. [online] Pld.cs.luc.edu. Disponível em: http://pld.cs.luc.edu/courses/447/sum08/class5/novak,sturges.stream5_reassembly.pdf [Acessado em 4 Set. 2018].