

Tema: Tecnologias para Segurança em Redes
TP: Práticas sobre Detecção de Intrusões em rede (≈8h)

Introdução

Até este momento abordamos um conjunto vasto de tecnologias, na sua grande maioria destinadas a proteger a informação, ou a evitar que algum *hacker* assuma algum tipo de controlo sobre o nosso equipamento informático.

Chegou então o momento de pensarmos de uma forma um pouco mais cautelosa e questionarmo-nos: “*E se eu for alvo de um ataque bem sucedido? O que posso fazer para pelo menos tentar detetar esse ataque?*”. A resposta a estas questões passa por duas alternativas. A primeira é verificar regularmente os *logs* do sistema (quer em Linux quer em Windows isso é fácil de fazer, embora a interpretação dos *logs* possa ser uma tarefa muito complexa ☹). A segunda alternativa é aquela que vamos aqui abordar, passa por instalar e configurar (e usar, claro!) um Sistema de Detecção de Intrusões para redes, ou NIDS – *Network Intrusion Detection System*.

Neste caso em particular vamos utilizar o **Snort** (é um software em *open source*). O Snort é bastante complexo, sendo composto por 4 módulos:

- **Sniffer**, (semelhante ao tcpdump, sendo os ficheiros de captura de tráfego compatíveis); este módulo é responsável pela captura do tráfego que chega à placa de rede local, a funcionar no modo promíscuo; a captura pode ser feita para o ecrã, ou para um ficheiro.
- **Preprocessor**, executa várias funções, sendo uma delas a deteção de pacotes mal formados, como por exemplo, respostas anormais a pedidos ARP; este módulo pode ainda ser usado para preparar os dados que vão ser analisados pelo módulo seguinte.
- **Detection Engine**, verifica os dados que lhe chegam contra um conjunto de regras predefinidas; este conjunto de regras contém as assinaturas dos ataques conhecidos, assim como de comportamentos não autorizados, desde que possam ser caracterizados em termos de características observáveis do tráfego; **MANTER E GERIR O CONJUNTO DE ASSINATURAS É UMA TAREFA MUITO COMPLEXA.**
- **Alerts**, envia uma mensagem de alerta, para um ficheiro, por e-mail, ou para o sistema de logs – conforme for configurado –, sempre que é detectado um pacote suspeito, que verificou alguma das regras.

Esta brevíssima descrição não dispensa a consulta da documentação do projeto, disponível em www.snort.org (e em particular a secção “New to Snort?”, por razões óbvias).

Um dos importantes desafios que se colocam à gestão das regras do Snort (e de qualquer NIDS), é o balanceamento entre as regras que são demasiadamente genéricas e as que são demasiadamente específicas. As primeiras, dada a sua natureza, acabam por gerar um número elevado de falsos alarmes – **falsos positivos** – mas são mais difíceis de contornar, pois caracterizam uma classe de ataques bastante genérica (por exemplo, todo o tráfego TCP que chegue entre as 2h e as 7h da madrugada é considerado anómalo). As regras específicas descrevem ataques particulares e, por conseguinte, não geram tantos falsos positivos. No entanto, uma ligeira variante do mesmo ataque não será detetada, causando **falsos negativos** (por exemplo, uma regra que detete a *string* “/index.html” não detetará a *string* “/index.html”; no entanto, o servidor irá interpretá-la da mesma forma).

Percebe-se assim que não é fácil descobrir a mistura de regras que garante o melhor desempenho do NIDS, em função da rede que estamos a considerar.

Como já foi referido, o Snort é um *software* disponível em regime de código aberto, existindo versões já compiladas para a grande maioria das implementações do Linux e para Windows. Neste trabalho iremos utilizar um ambiente Linux como referência (o Ubuntu, mais especificamente), mas, para além dos detalhes de instalação e das diretorias onde determinados ficheiros estão armazenados, qualquer outra plataforma poderá ser utilizada.

Objectivos

No final deste trabalho deverá estar apto a:

- 1) Instalar o Snort e utilizá-lo como um simples *sniffer* (interpretando os dados obtidos com a sua execução).
- 2) Configurar e testar o pré processador do Snort.
- 3) Configurar e testar a *detection engine* do Snort.
- 4) Criar regras, a partir das características de um dado ataque.
- 5) Analisar o ficheiro de alertas produzido pelo Snort e distinguir falsos positivos.

Material (assume-se que o aluno tem a capacidade para preparar este material, não sendo avaliado o respetivo esforço)

Por questões de segurança e flexibilidade, como habitualmente sugere-se a utilização de um ambiente de virtualização (VMWare, VirtualBox, ou QEMU/Kvm). Para executar este exercício irá necessitar de duas máquinas. Uma, a que chamaremos computador alvo, onde vai instalar e configurar o Snort (é conveniente que, mais tarde, possa instalar nesta máquina alguns outros serviços, tais como o http e o mysql); e uma outra que vai servir apenas para teste (e a que chamaremos computador de teste). Nesta última máquina deve dispor do *software* **ettercap**, que vai ser utilizado para simular a geração de tráfego malicioso (sugere-se a utilização do Kali). No que respeita ao computador alvo, pode utilizar uma imagem já pronta a utilizar, como seja uma versão do Ubuntu LTS (servidor) – daqui para a frente iremos assumir que usa alguma versão do Ubuntu. Deve ter o cuidado de adaptar o exercício à sua configuração. Por isso mesmo e como tem a liberdade de escolher qualquer configuração, **deve descrever no seu relatório a arquitetura que utilizou, assim como as respetivas versões do software.**

Nota 1: na documentação do **Snort** encontra guias de instalação para diversas configurações. É vivamente aconselhado a seguir um desses guias!

Nota 2: caso use um ambiente de virtualização recomenda-se que os interfaces de rede das MVs estejam configurados no modo NAT.

Nota 3: se, como recomendado, utilizar uma distribuição do Ubuntu Server, não terá ambiente gráfico, o que lhe pode colocar algumas dificuldades. Nesse caso recomenda-se que, a partir de uma máquina remota, ou do próprio *host* (seja Windows ou Linux), use o SSH para abrir uma consola – nessa consola poderá mais facilmente operar o servidor, quanto mais não seja porque terá um slide na janela que lhe permite aceder a resultados que já não estão visíveis no ecrã. A alternativa é instalar um interface gráfico, mas isso implica perda de fiabilidade ao nível de um servidor, o que não faz muito sentido!

Sumário das tarefas

- 1) Arrancar com ambas as máquinas e testar a conectividade (trivial).
- 2) Instalar o Snort e utilizá-lo como um simples *sniffer*.
- 3) Criar uma configuração do Snort que usa o pré processador.

- 4) Iniciar o Snort numa das máquinas, executar um ataque a partir da segunda máquina e verificar os *logs*.
- 5) Criar uma configuração do Snort que utilize a *detection engine* e testar essa configuração
- 6) Criar uma configuração com a *detection engine* e uma regra. Testar a eficácia da regra.

Tarefa 2 (o Snort como sniffer)

No computador alvo

- 1) Comece por instalar o Snort na máquina alvo; o processo deverá ser simples (☺ ou não!...). No Ubuntu bastará executar o comando `sudo apt-get install snort` mas não será má ideia seguir o guia de instalação oficial.

Nota 1: atenção que, durante a instalação, poderá ser-lhe pedido que identifique qual a porta de rede que o Snort deverá usar. Normalmente é a porta `eth0`, mas se estiver a utilizar um software de virtualização a porta pode ser outra – verifique qual a porta que é utilizada no *routing*, com o comando **`route -n`**; deverá ainda ter que indicar o endereço da rede em que está a operar.

Nota 2: em Linux o Snort recorre a uma biblioteca de funções, designada por ***libcap***, para colocar a placa de rede no modo promíscuo e fazer a captura do tráfego. O processo de instalação encarrega-se de integrar essa biblioteca e, habitualmente, não precisa de fazer mais nada.

Em Windows, o Snort, para o mesmo efeito, utiliza uma aplicação, designada por

WinPcap. Mas neste caso essa aplicação tem que ser instalada inicialmente. O **Wireshark** utiliza também essa aplicação e instala-a automaticamente, pelo que se instalou previamente aquela aplicação, então não precisa de fazer mais nada.

- 2) Execute o Snort com a opção `-?` e verifique (mesmo que por alto) a lista das opções que pode utilizar na linha de comando. **Em particular anote a função das opções `-v`, `-d`, `-e`, `-l`, `-c` e `-T`.**
- 3) Execute o comando `snort -vde`. Mas antes disso leia com atenção as seguintes notas.

Nota 1: atendendo aos recursos que o Snort utiliza, para o colocar a funcionar como *sniffer* tem que estar no papel de administrador, ou então executar o comando `snort` precedendo-o do comando `sudo`.

Nota 2: se estiver a trabalhar ligado a uma rede com bastante tráfego (sobretudo *broadcast*), pode ter dificuldade em responder à questão seguinte. Para contornar essa dificuldade deve procurar utilizar uma rede isolada (por exemplo, através de uma ligação direta entre os dois computadores).

Nota 3: para terminar a execução do snort deve usar a sequência `Ctrl+C`

a) Anote o que observa no ecrã, procurando justificar a atividade.

No computador de teste

- 4) Execute o comando `ping`, para o endereço do computador alvo.

a) Anote o que observa no ecrã do computador alvo, procurando justificar a atividade.

De novo no computador alvo

- 5) Interrompa o Snort, pressionado `Ctrl+C`

a) Anote o que observa no ecrã.

Tarefa 3 (criar uma configuração que usa o pré processador arpspooft)

No computador alvo

- 1) Vá para a diretoria **/etc/snort** onde é pressuposto estarem os ficheiros de configuração do Snort, assim como a diretoria **rules** onde estão armazenadas todas as regras.

Nota: deve ter à mão o manual do Snort onde todos os aspetos da configuração são discutidos.

- 2) Com o seu editor preferido abra o ficheiro com o nome **snort.conf**, que contém todos os detalhes da configuração do Snort. Este ficheiro tem diversas secções e numa instalação real precisa de ser otimizado de acordo. Neste exercício iremos apenas modificar algumas variáveis, mas poderá ter que explorar bastante mais, sendo, para tal, essencial a consulta da documentação do Snort, tal como é referido no ficheiro **snort.conf**.

- a) Na secção 1 (*Step #1*) altere a linha (adequando ao seu caso):

```
ipvar HOME_NET 192.168.1.0/24
```

e verifique as definições das variáveis

```
ipvar EXTERNAL_NET any
```

```
var RULE_PATH /etc/snort/rules
```

```
var PREPROC_RULE_PATH /etc/snort/preproc_rules
```

as restantes definições podem ficar como estão

Nota: O endereço de rede indicado deve ser, naturalmente, alterado de acordo com a configuração da rede que estiver a utilizar.

Esta secção define essencialmente variáveis globais, que o Snort e todos os seus componentes irão usar; o path indicado pressupõe uma instalação habitual mas, se usar uma instalação em que isso não aconteça, então deve procurar no sistema de ficheiros as diretorias **rules** e **preproc_rules** e fazer as alterações necessárias – tenha em atenção que pode mesmo não ter a diretoria **preproc_rules** e nesse caso terá que a procurar numa distribuição do Snort original, pois os ficheiros que nela estão contidos são essenciais nesta fase do exercício¹.

- b) Na secção 5 (*Step #5*) deverá remover o comentário da linha:

```
preprocessor arpspooft
```

Este é o pré processador do Snort que deteta as anomalias geradas no tráfego ARP durante um ataque por *ARP spoofing* – que força a modificação dos endereços MAC nas caches de ARP, nos computadores da rede – e que, portanto, deve ser carregado na inicialização do Snort. Este pré processador tem algumas “variantes” para detetar alterações mais específicas, como pode ver nos exemplos fornecidos no ficheiro **snort.conf**, mas não precisamos de usar essas variantes neste exercício.

- c) Na secção 6 (*Step #6*) deve comentar todas as opções dos *output plugins*, pois neste exercício pretendemos fazer o registo dos alertas apenas num ficheiro – numa instalação típica “de produção” deverá usar o “output unified2”, que usa um segundo programa para fazer o registo dos alertas na base de dados, aumentando a capacidade do Snort para processar o tráfego, ou ainda um método direto para Syslog, ou para Base de Dados...

Tenha ainda em atenção que as duas últimas linhas desta secção não podem ser comentadas, isto porque incluem no Snort os ficheiros de classificação e de referência para os alertas.

- d) Na secção 7 (*Step #7*) deve remover o comentário da linha:

```
include $PREPROC_RULE_PATH/preprocessor.rules
```

que vai incluir no Snort as regras a utilizar pelos pré processadores.

¹ Isto acontece porque muitas instalações do Snort mais atuais já não fazem uso dos preprocessadores!

Tarefa 4 (iniciar o Snort e testar a configuração)

No computador alvo

- 1) Execute o seguinte comando (sempre com direitos de administrador!) para iniciar o Snort:

```
snort -l /var/log/snort -c /etc/snort/snort.conf
```

Com este comando estamos a executar o Snort com duas opções. A primeira indica que queremos que toda a informação de saída do Snort seja enviada para o ficheiro **/var/log/snort**. A segunda indica que pretendemos que o Snort seja configurado de acordo com os parâmetros contidos no ficheiro **snort.conf**. Este ficheiro, para além das variáveis que caracterizam genericamente a rede, indica que pretendemos usar o pré processador **arpsoof**. Para testar essa configuração vamos lançar um ataque a partir do computador de teste – tenha em atenção que, dependendo da sua configuração, pode ter ainda que usar a opção **-i** na linha de comando que executa o Snort.

Mas antes de prosseguir analise com cuidado a informação que o Snort forneceu durante a inicialização. Poderá verificar que há regras a serem carregadas e que entram em conflito com outras, aparecendo-lhe alguns “warnings”; pode ser boa ideia comentar as regras que causam essas situações, por forma a ter uma instância do Snort mais adequada. Nesse processo a opção **-T** do Snort pode ser muito útil.

No computador de teste

- 2) Execute o **ettercap**. Dependendo da versão que estiver a utilizar, poderá aparecer-lhe inicialmente uma janela (Figura 1) onde deverá seleccionar as opções iniciais: **Options** → **Promisc mode** (deve estar seleccionado por defeito); e **Sniff** → **Unified sniffing**, escolhendo de seguida o interface de rede que pretende utilizar.



Figura 1 – Ecrã inicial do ettercap

- 3) Deverá então aparecer a janela principal do Ettercap, mostrada na figura 2. Do menu **Hosts** selecione a opção **Scan for hosts**. Deverá aparecer-lhe uma mensagem, na parte inferior da janela, a indicar-lhe o número de *hosts* que foram localizados. Se tal não acontecer, o mais provável é ter-se enganado na seleção da placa de rede.
- 4) Ainda do menu **Hosts** selecione a opção **Host list**. Selecione o endereço IP do *gateway* da sua rede e adicione-o como “Target 1”; selecione o endereço IP do computador alvo e adicione-o com “Target 2”. Finalmente, do menu **Mitm** selecione a opção **ARP poisoning** (não ative qualquer dos parâmetros adicionais). Após um intervalo curto (5 a

10 segundos), pare o ataque, através do menu **Mitm**.

Nota: quando executa em máquinas virtuais o Ettercap pode ter dificuldade em identificar corretamente todos os *hosts* e em particular o *gateway*; não se esqueça que nessa configuração existe um *switch* virtual no computador que executa as máquinas virtuais...

Nota: as opções `-T` e `-A` `console` do Snort pode ser muito úteis para verificar eventuais erros. A primeira faz com que o Snort verifique o ficheiro de configuração, sem entrar em operação de análise de tráfego; a segunda indica ao Snort que deve enviar os alertas para a consola e não para o sistema de log, o que permite verificar rapidamente se estão a ser gerados alertas.

- 5) Gere agora algum tráfego normal com a máquina alvo. Se tiver algum dos serviços habituais (telnet, http, ou ftp) abra algumas ligações.



Figura 2 – janela principal do Ettercap

No computador alvo

- 6) Interrompa a execução do Snort, através da sequência de teclas `Ctrl + C`. Observe com atenção o relatório apresentado pelo Snort e anote, em particular:
- a) Quantos pacotes o Snort recebeu e qual a distribuição desses pacotes por protocolos fundamentais (TCP, UDP, ICMP e ARP)? Justifique a observação com base na atividade criada.
 - b) Quantos alertas foram registados (se tudo correu bem, devem ter sido registados vários alertas)?
- 7) Verifique que foi criado o ficheiro **alert** na diretoria `/var/log/snort`. Este ficheiro contém um registo de todos os alertas que foram detetados. Para além desse ficheiro terá ainda um outro, **snort.log.***, com os pacotes que deram origem aos alertas – este ficheiro está no formato tcpdump e pode ser observado com qualquer comando adequado (`tcpdump` ou `snort`, com a opção `-r`, ou o `wireshark`, por exemplo). Analise cuidadosamente o conteúdo do ficheiro **alert** – em princípio, deverá encontrar os registos do tráfego ARP gerado, mas não deverá encontrar registos do restante

tráfego. Pode encontrar outros registos relativamente a outros protocolos que possam ter gerado tráfego anómalo, mas no contexto desta experiência, estamos interessados no tráfego ARP que geramos, naturalmente. A figura 3 ilustra e detalha o tipo de saída que deverá encontrar. **Copie para o relatório o conteúdo do ficheiro de alertas, juntamente com o conteúdo do ficheiro “snort.log.*” e comente o seu conteúdo.**

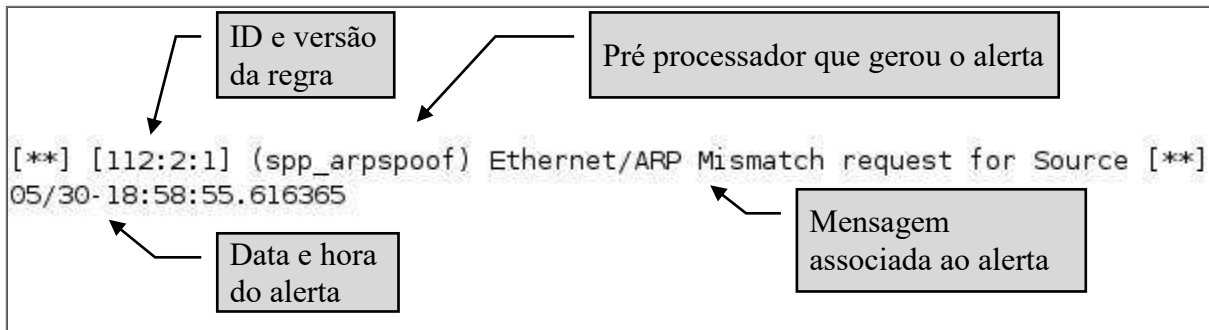


Figura 3 – alerta provocado por ARP *poisoning*

8) Elimine os registos criados na operação anterior, eliminando os ficheiros contidos na diretoria `/var/log/snort` (mas não elimine a diretoria `snort`).

9) **Opcional** – se pretender verificar a eficácia do ataque, execute novamente o comando **MITM → ARP poisoning** na máquina de teste e verifique o estado da cache de ARP do computador alvo, com o comando `arp -a` (no final, não se esqueça de parar o ARP poisoning 😊)

Tarefa 5 (criar uma configuração que utiliza a *detection engine* e testá-la)

No computador alvo

- 1) Em princípio ainda estará a trabalhar na diretoria `/etc/snort` (se por acaso não está, regresse àquela diretoria). Com o seu editor preferido crie um ficheiro com o nome **snort_detection.conf**
- 2) Nesse ficheiro insira as seguintes linhas:

```
1: ipvar HOME_NET 192.168.92.0/24
2: ipvar EXTERNAL_NET any
3: var RULE_PATH /etc/snort/rules
4: ipvar TELNET_SERVERS 192.168.92.136
5: ipvar FTP_SERVERS 192.168.92.136
6: ipvar HTTP_SERVERS $HOME_NET
7: ipvar SMTP_SERVERS $HOME_NET
8: portvar HTTP_PORTS 80
9: preprocessor stream5_global: max_tcp 8192, track_tcp yes, \
10:                                track_udp no
11: preprocessor stream5_tcp: policy first
12: include classification.config
13: include $RULE_PATH/telnet.rules
14: include $RULE_PATH/info.rules
```

Nota: O endereço indicado deve ser, naturalmente, alterado de acordo com a configuração da rede em questão; nas versões do Snort mais recentes e compiladas com a opção de IPv6, as variáveis com endereços deverão ser declaradas com o classificador `ipvar` (como aparece em cima).

As primeiras 3 linhas contém definições globais e já foram analisadas (confirma o *path*

indicado). As linhas 4 a 7 associam a um nome lógico os endereços de servidores de rede, facilitando a sua inclusão nas regras, posteriormente. A linha 8 faz o mesmo relativamente à porta onde estará disponível o serviço HTTP. As linhas 9 a 11 ativam o pré processador “stream5”, necessário nas mais recentes versões do Snort para suportar o controlo mais avançado do tráfego TCP, nomeadamente a capacidade de reconhecer ligações estabelecidas, fluxos de ou para o servidor, etc.. A linha 12 permite incluir na configuração do Snort o conjunto de classificadores dos alertas e as respetivas prioridades² (será útil visualizar o conteúdo desse ficheiro). A linha 13 adiciona o conjunto de regras *standard* disponíveis para o protocolo Telnet (é igualmente útil visualizar esse ficheiro). Finalmente, a linha 14 adiciona um ficheiro de regras proprietárias, produzidas pela equipa da Sourcefire, Inc. (“VRT Certified rules”), que mantém um repositório de regras para o Snort³. É exatamente sobre este ficheiro que irá recair a nossa atenção.

- 3) Abra o ficheiro **info.rules**. Procure a primeira ocorrência do valor 718. Irá encontrar esse valor precedido da sigla “**sid**” – Snort id. Esse valor identifica então a primeira regra do ficheiro e que a seguir se transcreve

```
alert tcp $TELNET_SERVERS 23 -> $EXTERNAL_NET any (msg:"INFO
TELNET login incorrect"; flow:from_server,established; content:"Login
incorrect"; reference:arachnids,127; classtype:bad-unknown; sid:718; rev:9;)
```

Uma regra é composta por um **cabeçalho** e um **corpo**. O cabeçalho contém (por esta ordem):

1. Ação (**alert**) – o que a regra irá produzir, caso se aplique;
2. O protocolo que a regra irá verificar (**tcp**);
3. Os endereços e portas dos computadores origem e destino, a que a regra se irá aplicar. Como se pode observar, no caso anterior são utilizadas as variáveis globais anteriormente definidas (\$TELNET_SERVERS e \$EXTERNAL_NET), sendo ainda referida a porta 23 no computador de origem (serviço telnet) e qualquer porta (**any**) no computador de destino;
4. Os caracteres “->” identificam o sentido do tráfego a observar pela regra.

Quanto ao corpo da regra, apesar de ser opcional, adiciona algum detalhe que permite aumentar a precisão e eficiência da regra. Cada secção do corpo tem o formato

nome_opção:opção

No caso da regra em análise, as opções utilizadas foram:

1. **msg:** → a mensagem que deve aparecer no registo do alerta.
2. **flow:** → define o sentido da comunicação em que a regra se aplica, neste caso “from_server”, com a condição adicional que a ligação esteja estabelecida, isto é, que as três fases iniciais do protocolo TCP tenham sido executadas (apenas acessível se o pré processador stream5 tiver executado).
3. **content:** → especifica um padrão de caracteres que terá que existir no conteúdo do pacote (dados) para que a regra se verifique – neste caso “Login incorrect”⁴

² Como iremos ver mais tarde, ao definir uma regra no Snort temos a possibilidade de “classificar” a regra de acordo com o seu nível de perigosidade e atribuir-lhes uma prioridade. Na prática isto corresponde a um par <classificador><n_prioridade>, que está definido neste ficheiro e que é adicionado a cada alerta, desde que a respetiva regra inclua o classificador. Não tem qualquer outro efeito que não seja informativo!

³ A empresa vende o serviço de acesso a regras com base numa atualização diária, mas também mantém um serviço público, embora com 1 semana de atraso, o que é perfeitamente suportável.

⁴ Esta é a mensagem enviada pelo servidor sempre que alguém tenta fazer *login* sem sucesso.

4. **reference:** → uma referência externa para fornecer mais informação acerca da regra e do ataque que ela tenta detectar; neste caso “arachnids,127” identifica a regra 127 de uma base de dados conhecida por *advanced reference archive of current heuristics for network intrusion detection systems* (ARACHNIDS), disponível em www.whitehats.com
 5. **classtype:** → a classificação do alerta, tal como definida no ficheiro `classification.config` (já referido acima); neste caso a classificação é “bad-unknown”, à qual está atribuída por defeito a prioridade 2 (de 1 a 4).
 6. **sid:** → permite criar um identificador da regra – evite usar um identificador já reservado
 7. **rev:** → indica o número da revisão da regra
- Observe mais algumas regras e encerre o ficheiro.

- 4) Execute novamente o Snort utilizando o comando

```
snort -l /var/log/snort -c /etc/snort/snort_detection.conf
```

Nota: verifique o número de regras que foram carregadas, através do sumário apresentado no ecrã; tenha em atenção à eventual necessidade de utilizar um outro *path*.

No computador de teste

- 5) Execute novamente o **ettercap** (desta vez, aproveite para experimentar outros tipos de interface, nomeadamente com as opções -T ou -C). A ideia é novamente colocar o *gateway* da rede como “Target 1” e o computador alvo como “Target 2”, iniciando depois um ataque Mitm por ARP *poisoning* (os passos são idênticos aos realizados anteriormente). Deixe o ataque decorrer durante 5 a 10 segundos e desligue a operação de ARP *poisoning*.
- 6) De seguida inicie uma sessão Telnet com o servidor. Utilize o nome de utilizador root, ou o que definiu anteriormente e uma palavra-passe errada qualquer. Faça um segundo *login*, mas desta vez com um nome de utilizador e palavra-passe corretos. Finalmente, termine a sessão.

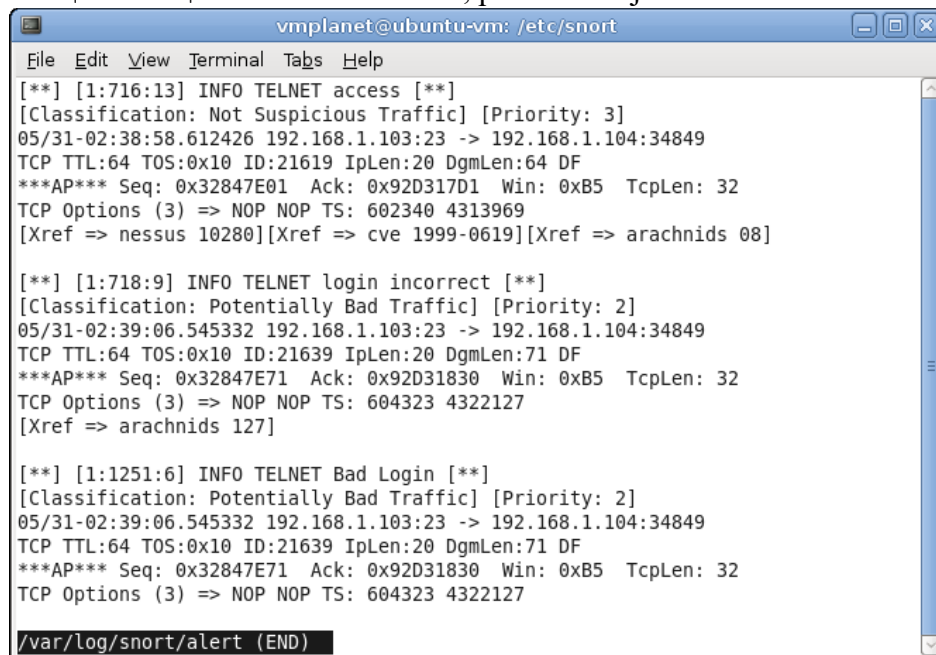
Nota 1: para executar este exercício deverá ter que instalar o serviço Telnet; sendo considerado um serviço não seguro (uma vez que não cifra a comunicação, deixando inclusivamente as credenciais de acesso visíveis), este serviço já não é ativado por defeito, habitualmente; **siga as instruções do seu servidor para o ativar, registando o processo no seu *logbook*.**

No computador alvo

- 7) Interrompa a execução do Snort com Ctrl + C
 - a. **Quantos pacotes o Snort recebeu?**
 - b. **Qual o número de pacotes TCP?**
 - c. **Quantos alertas foram gerados?**
- 8) Observe o conteúdo do ficheiro de alertas `/var/log/snort/alerts`. Deverá identificar aí alguns alertas, incluindo aqueles que são mostrados na figura 4. **Anote todos os alertas detetados e justifique-os à luz da atividade criada. Consegue localizar os alertas gerados pelo ataque ARP *poisoning*? Justifique.**

Nota: Algumas versões do Snort aceitam a regra mas não a conseguem verificar, porque os pacotes TCP/Telnet são truncados (pelo menos em alguns protocolos). Se isso lhe acontecer execute novamente o Snort, mas com o ficheiro de configuração inicial, que inclui alguns alertas associados ao uso do serviço Telnet. Ainda assim poderá não detetar

qualquer alerta... Deve registar no seu *logbook* o facto, anexando o sumário final apresentado pelo Snort.
Em complemento pode ainda alterar no ficheiro **info.rules** um campo da regra para `content:"|0d 0a|"` e verificar o efeito, procurando justificar o resultado obtido.



```

vmplanet@ubuntu-vm: /etc/snort
File Edit View Terminal Tabs Help

[**] [1:716:13] INFO TELNET access [**]
[Classification: Not Suspicious Traffic] [Priority: 3]
05/31-02:38:58.612426 192.168.1.103:23 -> 192.168.1.104:34849
TCP TTL:64 TOS:0x10 ID:21619 IpLen:20 DgmLen:64 DF
***AP*** Seq: 0x32847E01 Ack: 0x92D317D1 Win: 0xB5 TcpLen: 32
TCP Options (3) => NOP NOP TS: 602340 4313969
[Xref => nessus 10280][Xref => cve 1999-0619][Xref => arachnids 08]

[**] [1:718:9] INFO TELNET login incorrect [**]
[Classification: Potentially Bad Traffic] [Priority: 2]
05/31-02:39:06.545332 192.168.1.103:23 -> 192.168.1.104:34849
TCP TTL:64 TOS:0x10 ID:21639 IpLen:20 DgmLen:71 DF
***AP*** Seq: 0x32847E71 Ack: 0x92D31830 Win: 0xB5 TcpLen: 32
TCP Options (3) => NOP NOP TS: 604323 4322127
[Xref => arachnids 127]

[**] [1:1251:6] INFO TELNET Bad Login [**]
[Classification: Potentially Bad Traffic] [Priority: 2]
05/31-02:39:06.545332 192.168.1.103:23 -> 192.168.1.104:34849
TCP TTL:64 TOS:0x10 ID:21639 IpLen:20 DgmLen:71 DF
***AP*** Seq: 0x32847E71 Ack: 0x92D31830 Win: 0xB5 TcpLen: 32
TCP Options (3) => NOP NOP TS: 604323 4322127

/var/log/snort/alert (END)

```

Figura 4 – Exemplo dos alertas gerados pela *Detection Engine*

Tarefa 6 (criar uma configuração que utiliza a *detection engine*, uma regra específica e testá-la)

No computador alvo

- 1) Entre na diretoria `/etc/snort/rules`, onde estão armazenadas as regras do Snort. Aproveite para observar os ficheiros de regras que aí existem e cujo nome sugere o seu objetivo (na maioria dos casos, associado ao protocolo ou a módulos de aplicações que usam os protocolos).
- 2) Utilizando o seu editor preferido crie um ficheiro com o nome **subseven.rules** e acrescente-lhe uma linha com o texto

```

alert tcp any any -> any 27374 (msg:"SubSeven Connection
Attempt";sid:999)

```

Nota: a inclusão do campo `sid` pode ser obrigatória, dependendo da versão do Snort que estiver a utilizar.

Nesta altura deve já compreender perfeitamente qual o objetivo desta regra. **Qual é?**

- 3) Em seguida vamos criar um novo ficheiro de configuração, a que vamos dar o nome **snort_subseven.conf**. Utilize novamente o seu editor preferido, para incluir naquele ficheiro as seguintes linhas

```

var RULE_PATH /etc/snort/rules
include classification.config
include $RULE_PATH/subseven.rules

```

- 4) Execute novamente o Snort utilizando o comando

```
snort -l /var/log/snort -c /etc/snort/snort_subseven.conf
```

Nota: verifique o número de regras que foram carregadas, através do sumário apresentado no ecrã; tenha em atenção a eventual necessidade de utilizar um *path* diferente.

No computador de teste

- 5) Tente abrir uma ligação em Telnet no computador alvo, na porta 27374, executando o comando (tendo o cuidado de colocar o endereço IP correto)

```
telnet 192.168.1.103 27374
```

Em princípio e dependendo da configuração do seu servidor, a ligação deve ser recusada. De qualquer forma, tente mais uma ou duas vezes.

No computador alvo

- 6) Interrompa a execução do Snort com Ctrl + C
- Quantos pacotes recebeu o Snort?
 - Quantos alertas foram gerados?
- 7) Observe o conteúdo do ficheiro de alertas `/var/log/snort/alerts`. Deverá identificar aí um ou mais alertas.
- Anote todos os alertas detetados.
 - Consegue localizar os alertas gerados pela atividade que acabou de gerar?
 - Acha que o alerta gerado está de acordo com o objectivo da regra, anteriormente enunciado (ver 2, acima)?
 - Classificaria este alerta como um “Falso Positivo”?

Notas finais:

A criação e a manutenção das regras revelam-se como as tarefas mais complexas e mais decisivas no desempenho do sistema de deteção. Um excesso de regras e/ou regras demasiadamente genéricas levam à geração de imensos falsos positivos, com a consequente perda de objetividade na análise dos resultados da deteção. Por outro lado, um conjunto limitado de regras e/ou regras demasiadamente específicas, podem resultar na perda de alertas que, de outra forma, poderiam revelar alguma atividade importante. Pode encontrar na *Web* alguns sites com informação relevante sobre esta questão, mas antes disso necessita de aprofundar o seu conhecimento sobre a construção de regras (essa informação está no manual do Snort e em <http://manual.snort.org/node27.html> - do autor do Snort).

Por outro lado, a análise dos alertas é também uma tarefa complexa e decisiva. Para apoiar essa tarefa existem já algumas aplicações, que na sua maioria consistem em aplicações *Web* que acedem a alertas do Snort armazenados numa base de dados (MySQL, geralmente). O Snorby é frequentemente referido como um dos *frontend* mais utilizados, não só porque está em domínio público, mas também porque tem vindo a ser alvo de diversas atualizações por uma comunidade alargada (<https://github.com/Snorby/snorby>). Outra ferramenta importante é o Barnyard (<https://github.com/firnsy/barnyard2>), o qual pode aumentar significativamente o desempenho do Snort, assumindo todas as operações de acesso à BD, o que consome bastante tempo – em algumas distribuições é mesmo possível encontrar o Snort já configurado com o Barnyard. Na Internet poderá facilmente encontrar mais informação acerca destas configurações integradas.