Criptografia e Segurança em Redes

ENGENHARIA DE TELECOMUNICAÇÕES E INFORMÁTICA

2023/2024

Trabalho Prático 1

Traffic sniffing & Packet spoofing

David Alexandre Baptista Oliveira – a86732@alunos.uminho.pt

José Pedro Fernandes Peleja – a84436@alunos.uminho.pt

Miguel Fernandes Pereira – a94152@alunos.uminho.pt

Índice

Índice de figuras

Questão 1.

Questão 1.1.

No acesso à página www.scanme.org foi observada a utilização do protocolo HTTP, enquanto que na página www.owasp.org é utilizado o protocolo HTTPS. Com isto, a principal diferença é que no protocolo HTTPS a informação presente no *packet* é encriptada e segura, por outro lado, no HTTP é possível observar toda a informação presente no *packet*.

Questão 1.2.

A propriedade de segurança mais afetada seria a confidencialidade, pois os dados no protocolo HTTP estão em texto simples (*plain text*), tornando os dados facilmente legíveis para qualquer utilizador da rede que intercete o tráfego. Por outro lado, o protocolo HTTPS, devido à criptografia, torna a análise e observação dos dados mais complexa, garantindo assim a confidencialidade dos dados.

Questão 1.3.

Após a reconstrução do fluxo no acesso à página Scanme.org é observável toda a informação presente no site e todas as pesquisas realizadas pelo utilizador em texto simples. Os dados acessíveis após a analise são as datas de acesso, as pesquisas realizadas, o sistema operativo do utilizador e o *browser* em uso. Visto isto, a propriedade de confidencialidade é completamente posta em causa.

Em contrapartida, na reconstrução do fluxo no acesso à página Owasp.org não é possível pois este utiliza a criptografia TLS protegendo assim os dados da página.

Questão 2.

Questão 2.1.

As seguintes figuras representam a captura de pacotes ARP, a figura 1 é a representação e analise do ficheiro .pcap no Wireshark, enquanto que a figura 2 é a analise dos mesmos pacotes através do terminal após o *sniffing* usando a ferramenta Scapy.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	82:c0:4f:b4:d1:b1	22:7b:fe:1d:11:2d	ARP	42	Who has 192.168.29.246? Tell 192.168.29.236
2	0.000058	22:7b:fe:1d:11:2d	82:c0:4f:b4:d1:b1	ARP	42	192.168.29.246 is at 22:7b:fe:1d:11:2d
3	8.708929	PcsCompu_3b:97:10	82:c0:4f:b4:d1:b1	ARP	60	Who has 192.168.29.236? Tell 192.168.29.17
4	8.709716	82:c0:4f:b4:d1:b1	PcsCompu_3b:97:10	ARP	42	192.168.29.236 is at 82:c0:4f:b4:d1:b1
5	33.018165	82:c0:4f:b4:d1:b1	22:7b:fe:1d:11:2d	ARP	42	Who has 192.168.29.246? Tell 192.168.29.236
6	33.018236	22:7b:fe:1d:11:2d	82:c0:4f:b4:d1:b1	ARP	42	192.168.29.246 is at 22:7b:fe:1d:11:2d
7	46.841855	82:c0:4f:b4:d1:b1	PcsCompu_3b:97:10	ARP	42	Who has 192.168.29.17? Tell 192.168.29.236
8	46.842212	PcsCompu_3b:97:10	82:c0:4f:b4:d1:b1	ARP	60	192.168.29.17 is at 08:00:27:3b:97:10
9	61.433896	82:c0:4f:b4:d1:b1	22:7b:fe:1d:11:2d	ARP	42	Who has 192.168.29.246? Tell 192.168.29.236
10	61.433961	22:7b:fe:1d:11:2d	82:c0:4f:b4:d1:b1	ARP	42	192.168.29.246 is at 22:7b:fe:1d:11:2d

Figura 1- Resultado captura ARP com Scapy.

```
###[ Ethernet ]###
  dst       = 22:7b:fe:1d:11:2d
  src       = 82:c0:4f:b4:d1:b1
  type      = ARP
###[ ARP ]###
     hwtype    = Ethernet (10Mb)
     ptype     = IPv4
     hwlen     = 6
     plen      = 4
     op        = who-has
     hwsrc     = 82:c0:4f:b4:d1:b1
     psrc      = 192.168.29.236
     hwdst     = 00:00:00:00:00:00
     pdst      = 192.168.29.246

###[ Ethernet ]###
  dst       = 82:c0:4f:b4:d1:b1
  src       = 22:7b:fe:1d:11:2d
  type      = ARP
###[ ARP ]###
     hwtype    = Ethernet (10Mb)
     ptype     = IPv4
     hwlen     = 6
     plen      = 4
     op        = is-at
     hwsrc     = 22:7b:fe:1d:11:2d
     psrc      = 192.168.29.246
     hwdst     = 82:c0:4f:b4:d1:b1
     pdst      = 192.168.29.236
```

```
###[ Ethernet ]###
  dst       = 22:7b:fe:1d:11:2d
  src       = 82:c0:4f:b4:d1:b1
  type      = ARP
###[ ARP ]###
     hwtype    = Ethernet (10Mb)
     ptype     = IPv4
     hwlen     = 6
     plen      = 4
     op        = who-has
     hwsrc     = 82:c0:4f:b4:d1:b1
     psrc      = 192.168.29.236
     hwdst     = 00:00:00:00:00:00
     pdst      = 192.168.29.246

###[ Ethernet ]###
  dst       = 82:c0:4f:b4:d1:b1
  src       = 22:7b:fe:1d:11:2d
  type      = ARP
###[ ARP ]###
     hwtype    = Ethernet (10Mb)
     ptype     = IPv4
     hwlen     = 6
     plen      = 4
     op        = is-at
     hwsrc     = 22:7b:fe:1d:11:2d
     psrc      = 192.168.29.246
     hwdst     = 82:c0:4f:b4:d1:b1
     pdst      = 192.168.29.236
```

Figura 2- Resultado captura ARP através do Scapy.

Questão 2.2.

Visto que a página Owasp.org utiliza o protocolo HTTPS e como estamos a tentar capturar qualquer pacote TCP com porta de destino 80 , não é possível capturar qualquer pacote. Isto acontece devido ao facto de o HTTPS ter a porta de destino 443.

No caso do Scanme.org de forma geral foi observado o mesmo comportamento, no entanto no que toca aos cabeçalhos html, apenas foram capturados na análise através do Wireshark no *host* na questão 1.

Questão 2.3.

A diferença notável aqui é que ao realizar a captura de todo o tráfego que tem origem na subrede do sistema virtual, é capturada uma ampla variedade de protocolos e pacotes, independentemente de suas portas de origem ou destino. Isso difere da pergunta anterior, em que a captura estava limitada a pacotes TCP destinados à porta 80.

Questão 2.4.

O Scapy é uma ferramenta que permite criar, manipular, observar e enviar pacotes de rede. Esta capacidade pode ser utilizada para realizar ataques de *DoS* e inundar a rede ou *host* com pacotes indesejados. Isto leva a interrupções no serviço e afeta a disponibilidade do mesmo.

Questão 3.

Questão 3.1.

Executando o script de *spoofing* no sistema *host* percebemos que de facto o pacote forjado é enviado para o endereço ip de destino (8.8.8.8) mas com o endereço de origem da VM. Por consequência, o servidor de destino responde com um pacote orientado á VM, ou seja, endereço de origem 8.8.8.8 e endereço de destino da VM. Usando o Wireshark no sistema *host*, conseguimos capturar tanto o pacote enviado, quanto o recebido.

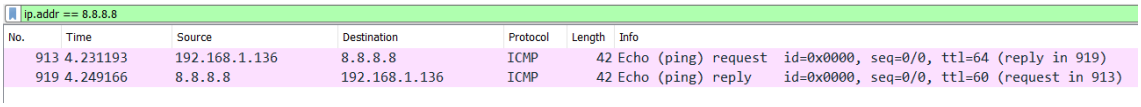


Figura 3- Captura de tráfego após *Spoofing* no *host.*

Questão 3.2.

As propriedades de segurança violadas pelo programa nesta tarefa são a de confidencialidade, a de integridade , a de disponibilidade, a de autenticidade e o não-repudio.

Questão 3.3.

Usando o Wireshark no ambiente virtual (VM), e aplicando no campo ”IP de origem” o endereço 8.8.8.8 percebemos que o pacote foi entregue com sucesso. No entanto, conseguimos entender também que o pacote forjado pelo script de *spoofing* não consegue ser capturado pelo Wireshark na VM.

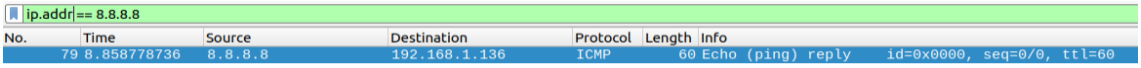
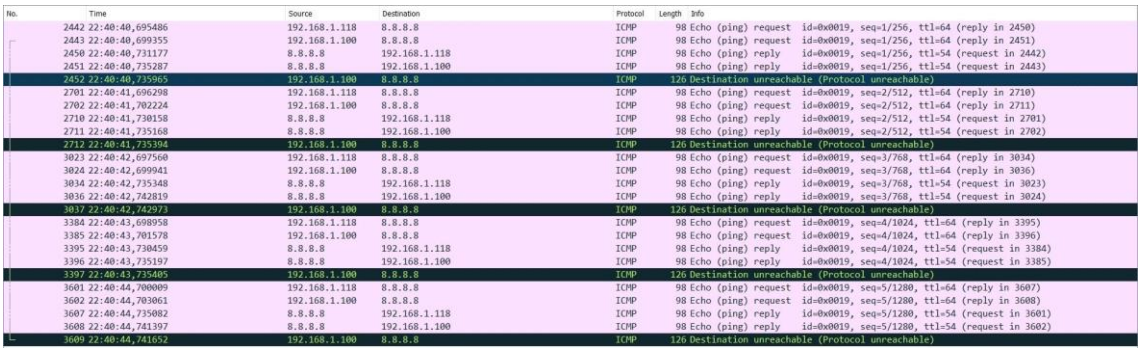


Figura 4- *Echo reply* no sistema de destino.

Questão 3.4.

Para cumprir o objetivo desta questão, primeiramente tivemos de intercetar os pacotes com endereço de origem da máquina virtual e endereço de destino 8.8.8.8 (ou seja, *sniffing*). Feito isto, forjamos um pacote ICMP similar ao intercetado (*spoofing*), alterando apenas o endereço de origem do pacote para corresponder ao endereço IP do sistema *host*, e enviamos o pacote forjado.

Através da análise dos pacotes usando o Wireshark no sistema *host* podemos observar que de facto um novo pacote similar ao original é enviado imediatamente depois do original, mas com as alterações efetuadas no passo acima, e consequentemente o servidor responde enviando dois novos pacotes com endereços de destino diferentes, um com endereço de destino da VM e outro com o do *host*, como é possível verificar na figura 5.



No.	Time	Source	Destination	Protocol	Length	Info
2442	22:40:40,695486	192.168.1.118	8.8.8.8	ICMP	98	Echo (ping) request id=0x0019, seq=1/256, ttl=64 (reply in 2450)
2443	22:40:40,699355	192.168.1.100	8.8.8.8	ICMP	98	Echo (ping) request id=0x0019, seq=1/256, ttl=64 (reply in 2451)
2450	22:40:40,731177	8.8.8.8	192.168.1.118	ICMP	98	Echo (ping) reply id=0x0019, seq=1/256, ttl=54 (request in 2442)
2451	22:40:40,735287	8.8.8.8	192.168.1.100	ICMP	98	Echo (ping) reply id=0x0019, seq=1/256, ttl=54 (request in 2443)
2452	22:40:40,735965	192.168.1.100	8.8.8.8	ICMP	126	Destination unreachable (Protocol unreachable)
2701	22:40:41,696298	192.168.1.118	8.8.8.8	ICMP	98	Echo (ping) request id=0x0019, seq=2/512, ttl=64 (reply in 2710)
2702	22:40:41,702224	192.168.1.100	8.8.8.8	ICMP	98	Echo (ping) request id=0x0019, seq=2/512, ttl=64 (reply in 2711)
2710	22:40:41,730158	8.8.8.8	192.168.1.118	ICMP	98	Echo (ping) reply id=0x0019, seq=2/512, ttl=54 (request in 2701)
2711	22:40:41,735168	8.8.8.8	192.168.1.100	ICMP	98	Echo (ping) reply id=0x0019, seq=2/512, ttl=54 (request in 2702)
2712	22:40:41,735394	192.168.1.100	8.8.8.8	ICMP	126	Destination unreachable (Protocol unreachable)
3023	22:40:42,697560	192.168.1.118	8.8.8.8	ICMP	98	Echo (ping) request id=0x0019, seq=3/768, ttl=64 (reply in 3034)
3024	22:40:42,699941	192.168.1.100	8.8.8.8	ICMP	98	Echo (ping) request id=0x0019, seq=3/768, ttl=64 (reply in 3036)
3034	22:40:42,735348	8.8.8.8	192.168.1.118	ICMP	98	Echo (ping) reply id=0x0019, seq=3/768, ttl=54 (request in 3023)
3036	22:40:42,742819	8.8.8.8	192.168.1.100	ICMP	98	Echo (ping) reply id=0x0019, seq=3/768, ttl=54 (request in 3024)
3037	22:40:42,742973	192.168.1.100	8.8.8.8	ICMP	126	Destination unreachable (Protocol unreachable)
3384	22:40:43,698958	192.168.1.118	8.8.8.8	ICMP	98	Echo (ping) request id=0x0019, seq=4/1024, ttl=64 (reply in 3395)
3385	22:40:43,701578	192.168.1.100	8.8.8.8	ICMP	98	Echo (ping) request id=0x0019, seq=4/1024, ttl=64 (reply in 3396)
3395	22:40:43,730459	8.8.8.8	192.168.1.118	ICMP	98	Echo (ping) reply id=0x0019, seq=4/1024, ttl=54 (request in 3384)
3396	22:40:43,735197	8.8.8.8	192.168.1.100	ICMP	98	Echo (ping) reply id=0x0019, seq=4/1024, ttl=54 (request in 3385)
3397	22:40:43,735405	192.168.1.100	8.8.8.8	ICMP	126	Destination unreachable (Protocol unreachable)
3601	22:40:44,700009	192.168.1.118	8.8.8.8	ICMP	98	Echo (ping) request id=0x0019, seq=5/1280, ttl=64 (reply in 3607)
3602	22:40:44,703061	192.168.1.100	8.8.8.8	ICMP	98	Echo (ping) request id=0x0019, seq=5/1280, ttl=64 (reply in 3608)
3607	22:40:44,735082	8.8.8.8	192.168.1.118	ICMP	98	Echo (ping) reply id=0x0019, seq=5/1280, ttl=54 (request in 3601)
3608	22:40:44,741397	8.8.8.8	192.168.1.100	ICMP	98	Echo (ping) reply id=0x0019, seq=5/1280, ttl=54 (request in 3602)
3609	22:40:44,741652	192.168.1.100	8.8.8.8	ICMP	126	Destination unreachable (Protocol unreachable)

Figura 5- Resposta do sistema de destino aos pacotes enviados.