

# Tópicos Especiais em Segurança da Informação

## TP6 - Ataques de Negação de Serviço

Arthur do Prado Labaki

21-06, 2022

GBC 235

## Informações adicionais

Nem todos os exercícios estão com imagem aqui no relatório, mas todas as imagens integradas nesse relatório, quanto códigos, planilhas ou gifs de demonstração estão em meu repositório no GitHub abaixo.

[Link do meu GitHub](#)

## Resolução do item 1)

Foi possível estabelecer conexão entre as máquinas, utilizando o comando ping ou ssh.

## Resolução do item 2)

O SYN Cookie é uma técnica usada para resistir a ataques de inundação (SYN flood attacks). Ao ativar o recurso, o sistema passa a responder ao pacote SYN inicial com um cookie, que identifica o cliente. Com isso, o sistema aloca espaço para a conexão apenas após receber o pacote ACK de resposta, tornando o ataque inefetivo.

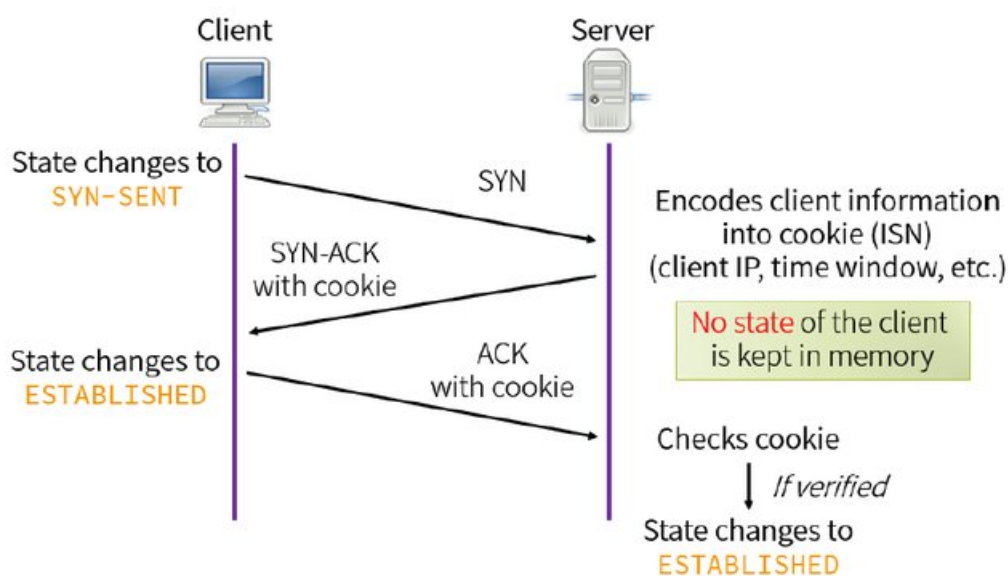


Figura 1: Exemplo de uso da técnica do SYN Cookie

## Resolução do item 3)

Verificando o alvo, temos que o tamanho da fila de recursos alocado é de 128. O SYN cookie está com valor 1, significando que está ativado. Utilizando o comando para desativa-lo, temos o valor do SYN cookie como 0.

```
[06/10/22]seed@VM:~$ sysctl -q net.ipv4.tcp_max_syn_backlog
net.ipv4.tcp_max_syn_backlog = 128
[06/10/22]seed@VM:~$ sudo sysctl -a | grep cookie
net.ipv4.tcp_syncookies = 1
net.netfilter.nf_conntrack_sctp_timeout_cookie_echoed = 3
net.netfilter.nf_conntrack_sctp_timeout_cookie_wait = 3
[06/10/22]seed@VM:~$ sudo sysctl -w net.ipv4.tcp_syncookies=0
net.ipv4.tcp_syncookies = 0
```

*Figura 2: Utilização dos comandos do item 3*

## Resolução do item 4)

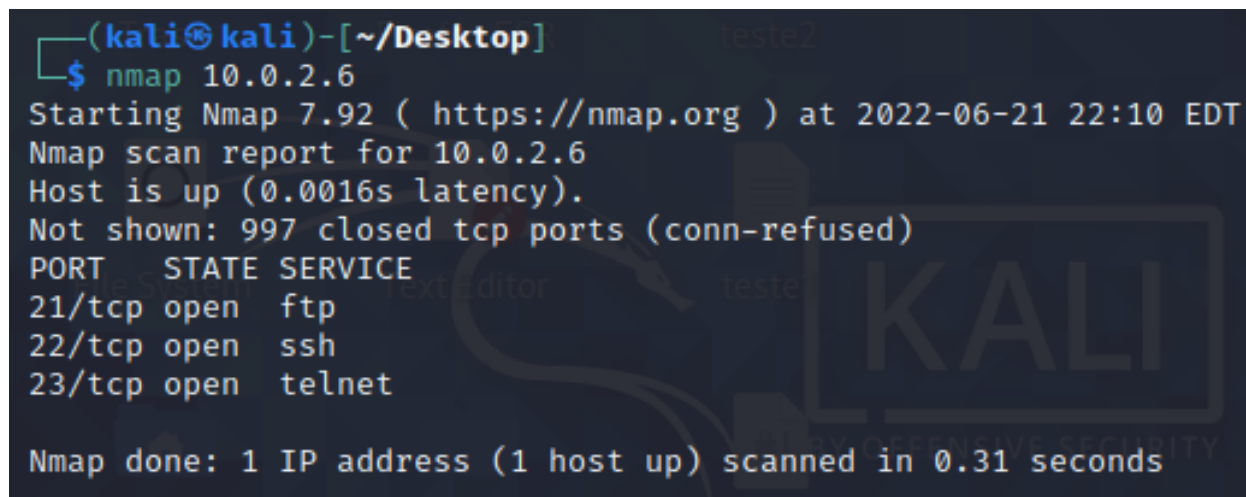
Utilizando o comando, pode verificar que não existe nenhuma conexão TCP semi-aberta (SYN-RECV).

```
[06/10/22]seed@VM:~$ netstat -na | grep tcp
tcp        0      0 127.0.0.1:33841      0.0.0.0:*            LISTEN
tcp        0      0 192.168.122.1:53     0.0.0.0:*            LISTEN
tcp        0      0 127.0.0.53:53        0.0.0.0:*            LISTEN
tcp        0      0 0.0.0.0:22           0.0.0.0:*            LISTEN
tcp        0      0 127.0.0.1:631        0.0.0.0:*            LISTEN
tcp        0      0 0.0.0.0:23           0.0.0.0:*            LISTEN
tcp        0      0 10.0.2.6:53726       52.43.98.120:443     ESTABLISHED
tcp        0      0 10.0.2.6:34626       34.122.121.32:80     TIME_WAIT
tcp        0      0 10.0.2.6:39234       66.203.125.12:443    ESTABLISHED
tcp        0      0 10.0.2.6:22          10.0.2.15:32790      ESTABLISHED
tcp6       0      0 :::21                :::*                 LISTEN
tcp6       0      0 :::22                :::*                 LISTEN
tcp6       0      0 :::1:631             :::*                 LISTEN
```

*Figura 3: Utilização do comando para mostrar conexões TCP*

## Resolução do item 5)

As portas abertas da máquina alvo são as 21, 22 e 23 (ftp, ssh e telnet). Em 1 minuto de utilização do comando ping, foram transmitidos 60 pacotes com uma média de 0.561 ms.

A terminal window from a Kali Linux machine. The prompt is (kali㉿kali)-[~/Desktop]. The user has entered the command \$ nmap 10.0.2.6. The output shows the Nmap scan report for 10.0.2.6, indicating the host is up with a latency of 0.0016s. It lists three open ports: 21/tcp (ftp), 22/tcp (ssh), and 23/tcp (telnet). The scan was completed in 0.31 seconds. A large, semi-transparent 'KALI' watermark is visible in the background of the terminal output.

```
(kali㉿kali)-[~/Desktop]
$ nmap 10.0.2.6
Starting Nmap 7.92 ( https://nmap.org ) at 2022-06-21 22:10 EDT
Nmap scan report for 10.0.2.6
Host is up (0.0016s latency).
Not shown: 997 closed tcp ports (conn-refused)
PORT      STATE SERVICE
21/tcp    open  ftp
22/tcp    open  ssh
23/tcp    open  telnet
Nmap done: 1 IP address (1 host up) scanned in 0.31 seconds
```

*Figura 4: Utilização do nmap na máquina alvo*

## Resolução do item 6 e 7 )

Foi disparado ataques na máquina alvo na porta 22. Foi possível perceber que existe um ataque ocorrendo, usando o comando para verificar conexões TCP semi-abertas, é possível encontrar mais de 30 existentes. O comando ping também consegue evidenciar, passando de 0.561 ms de média para 37.398 ms, com alguns pacotes passando de 100 ms. Também foi possível analisar que chega uma quantidade anormal de pacotes que a máquina alvo recebe utilizando o Wireshark, além da máquina alvo ficar relativamente mais lenta que o normal. Por fim, não foi possível estabelecer conexão via ssh de uma terceira máquina com a máquina alvo, demorando muito para conectar ou até mesmo não conectando, demonstrando que o ataque realmente está ocorrendo.

## Resolução do item 8)

Refazendo o ataque com o SYN Cookie ativado na máquina alvo, é possível identificar que existe alguma diferença. Existe apenas 16 conexões TCP semi-abertas. No comando ping, o tempo médio dos pacotes é de 3.729 ms, muito diferente do exercício anterior. No Wireshark é possível ver que o ataque ocorre e também que o envio de um cookie de confirmação, mas não afeta a máquina. Tentando conectar no ssh por outra máquina, é possível notar uma demora quase irrelevante (4 segundos), porém a conexão foi estabelecida e funcionando sem travamentos.

## Resolução do item 9)

É possível evidenciar que, quanto menor a fila de registros acumulada, pior será o desempenho do *host* contra o ataque, pois está diminuindo o número máximo de conexões entreabertas possíveis, chegando rapidamente em seu limite, sobrecarregando o alvo. Porém caso a fila for muito grande, se o sistema não tiver memória suficiente para lidar com esse aumento de tamanho da fila de registros acumulados, seu desempenho será afetado negativamente, também prejudicando o alvo.

## Resolução do item 10)

O comando ping envia um *ECHO REQUEST* do *Internet Control Message Protocol* para obter um *ECHO RESPONSE* de um *host*. Diferente do SYN Flood normal, o ping não realiza o *handshake* e, conseqüentemente, não faz a requisição para o servidor de alocar um espaço no disco. Com isso, o ping é inferior ao ataque normal, quase nem prejudicando a máquina alvo. Para inundar o alvo com inúmeras requisições ping, foi utilizado o comando "ping -i 0.002 -s1500 10.0.2.6", sendo:

- **-i 0.002** o tempo entre as requisições, sendo 0.002 segundo o menor valor possível;
- **-s1500** o tamanho das requisições, no caso 1500 (1528) bytes;
- **10.0.2.6** o IP do alvo.

```

File Actions Edit View Help
File System Text Editor teste1
kali@kali: ~
File Actions Edit View Help
64 bytes from 10.0.2.6: icmp_seq=44 ttl=
64 bytes from 10.0.2.6: icmp_seq=45 ttl=
64 bytes from 10.0.2.6: icmp_seq=46 ttl=
64 bytes from 10.0.2.6: icmp_seq=47 ttl=
64 bytes from 10.0.2.6: icmp_seq=48 ttl=
64 bytes from 10.0.2.6: icmp_seq=49 ttl=
64 bytes from 10.0.2.6: icmp_seq=50 ttl=
64 bytes from 10.0.2.6: icmp_seq=51 ttl=
64 bytes from 10.0.2.6: icmp_seq=52 ttl=
64 bytes from 10.0.2.6: icmp_seq=53 ttl=
64 bytes from 10.0.2.6: icmp_seq=54 ttl=
64 bytes from 10.0.2.6: icmp_seq=55 ttl=
64 bytes from 10.0.2.6: icmp_seq=56 ttl=
64 bytes from 10.0.2.6: icmp_seq=57 ttl=
64 bytes from 10.0.2.6: icmp_seq=58 ttl=64 time=0.212 ms
64 bytes from 10.0.2.6: icmp_seq=59 ttl=64 time=0.185 ms
64 bytes from 10.0.2.6: icmp_seq=60 ttl=64 time=0.464 ms
^C
— 10.0.2.6 ping statistics —
60 packets transmitted, 60 received, 0% packet loss, time 60089ms
rtt min/avg/max/mdev = 0.152/0.561/3.405/0.635 ms
(kali@kali)-[~/Desktop]
$

File Actions Edit View Help
1508 bytes from 10.0.2.6: icmp_seq=30339 ttl=64 time=0.471 ms
1508 bytes from 10.0.2.6: icmp_seq=30340 ttl=64 time=1.02 ms
1508 bytes from 10.0.2.6: icmp_seq=30341 ttl=64 time=0.730 ms
1508 bytes from 10.0.2.6: icmp_seq=30342 ttl=64 time=1.51 ms
1508 bytes from 10.0.2.6: icmp_seq=30343 ttl=64 time=0.331 ms
1508 bytes from 10.0.2.6: icmp_seq=30344 ttl=64 time=0.335 ms
1508 bytes from 10.0.2.6: icmp_seq=30345 ttl=64 time=0.367 ms
1508 bytes from 10.0.2.6: icmp_seq=30346 ttl=64 time=0.354 ms
1508 bytes from 10.0.2.6: icmp_seq=30347 ttl=64 time=0.486 ms
1508 bytes from 10.0.2.6: icmp_seq=30348 ttl=64 time=2.72 ms
1508 bytes from 10.0.2.6: icmp_seq=30349 ttl=64 time=0.946 ms
1508 bytes from 10.0.2.6: icmp_seq=30350 ttl=64 time=1.20 ms
1508 bytes from 10.0.2.6: icmp_seq=30351 ttl=64 time=0.291 ms
1508 bytes from 10.0.2.6: icmp_seq=30352 ttl=64 time=0.744 ms
1508 bytes from 10.0.2.6: icmp_seq=30353 ttl=64 time=1.29 ms
1508 bytes from 10.0.2.6: icmp_seq=30354 ttl=64 time=0.465 ms
1508 bytes from 10.0.2.6: icmp_seq=30355 ttl=64 time=0.257 ms
^C
— 10.0.2.6 ping statistics —
30356 packets transmitted, 30355 received, 0.00329424% packet loss, time 63692ms
rtt min/avg/max/mdev = 0.144/0.489/16.489/0.507 ms, pipe 2
(kali@kali)-[~/Desktop]
$

```

Figura 5: Utilização do ping como SYN Flood

## Resolução do item 11)

### 1) Comportamento do Alvo

Mesmo com mais memória, SYN Cookie ativado e a fila com o tamanho *default*, duas máquinas atacantes é superior todos os ataques feitos nesse relatório, com 1289,469 segundos de tempo médio e mais de 80% de *packet loss*. Foi possível perceber também que, menos memória principal dos atacantes, aparentemente o ataque fica melhor.

### 2) Proteção para Empresas

Contra ataques de SYN Flood, empresas podem utilizar alguns métodos. Alguns desses métodos podem ser o que foi visto aqui, como aumentar a fila de registros acumuladas, ou o SYN Cookies, ou ainda reciclar a conexão TCP entreaberta mais antiga assim que o registro estiver cheio. Porém esses métodos são precários para empresas, que podem receber muitos

ataques de diferentes máquinas. A melhor maneira disponível hoje no mercado é contratar um serviço externo, como a Cloudflare que funciona como um *firewall* entre os pacotes e o servidor, lidando com o processo de *handshake*.