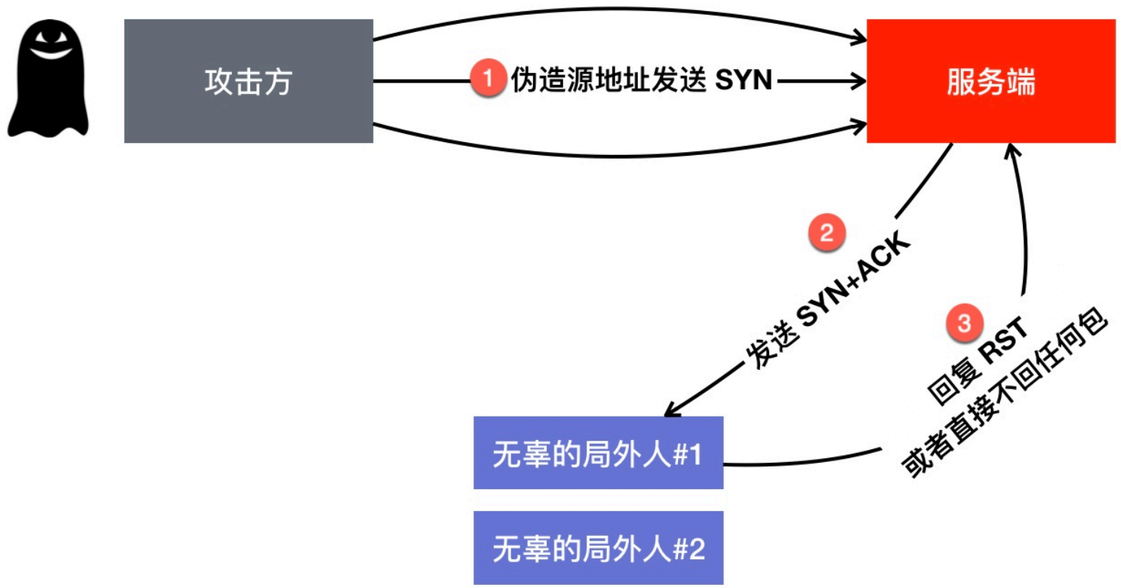
SYN Flood（SYN洪泛）攻击原理

SYN flood 攻击

SYN Flood 是一种广为人知的 DoS（拒绝服务攻击），想象一个场景：客户端大量伪造 IP 发送 SYN 包，服务端回复的 ACK+SYN 去到了一个「未知」的 IP 地址，势必会造成服务端大量的连接处于 SYN_RCVD 状态，而服务器的半连接队列大小也是有限的，如果半连接队列满，也会出现无法处理正常请求的情况。



Scapy 工具介绍

Scapy是一个用 Python 写的强大的交互式数据包处理程序。它可以让用户发送、侦听和解析并伪装网络报文。官网地址：https://scapy.net/ ，安装步骤见官网。

安装好以后执行sudo scapy就可以进入一个交互式 shell

```
$ sudo scapy
>>>
```

发送第一个包

在服务器（10.211.55.15）开启 tcpdump 抓包

```
sudo tcpdump -i any host 10.211.55.3 -nn
```

在客户端（10.211.55.3）启动sudo scapy输入下面的指令

```
send(IP(dst="10.211.55.15")/ICMP())
```

```
Sent 1 packets.
```

服务端的抓包文件显示服务端收到了客户端的ICMP echo request

```
06:12:47.466874 IP 10.211.55.3 > 10.211.55.15: ICMP echo request, id 0, seq 0, length 8
06:12:47.466910 IP 10.211.55.15 > 10.211.55.3: ICMP echo reply, id 0, seq 0, length 8
```

scapy 构造数据包的方式

可以看到构造一个数据包非常简单，scapy 采用一个非常简单易懂的方式：**使用/来「堆叠」多个层的数据**

比如这个例子中的 IP()/ICMP()，如果要用 TCP 发送一段字符串 hello，world，就可以这样堆叠：

```
IP(src="10.211.55.99", dst="10.211.55.10") / TCP(sport=9999, dport=80) / "hello, world"
```

如果要发送 DNS 查询，可以这样堆叠：

```
IP(dst="8.8.8.8") / UDP() /DNS(rd=1, qd=DNSQR(qname="www.baidu.com"))
```

接收数据

如果想拿到返回的结果，可以使用sr（send-receive）函数，与它相关的有一个特殊的函数sr1，只取第一个应答数据包

```
>>> res = sr1(IP(dst="10.211.55.15")/ICMP())
>>> res
<IP  version=4 ihl=5 tos=0x0 len=28 id=65126 flags= frag=0 ttl=64 proto=icmp chksum=0xf8c5
src=10.211.55.10 dst=10.211.55.5 |<ICMP  type=echo-reply code=0 chksum=0xffff id=0x0 seq=0x0 |>>
```

scapy 模拟

在客户端用 scapy 执行的 sr1 函数向目标机器（10.211.55.5）发起 SYN 包

```
sr1(IP(src="23.16.*.*", dst="10.211.55.15") / TCP(dport=80, flags="S") )
```

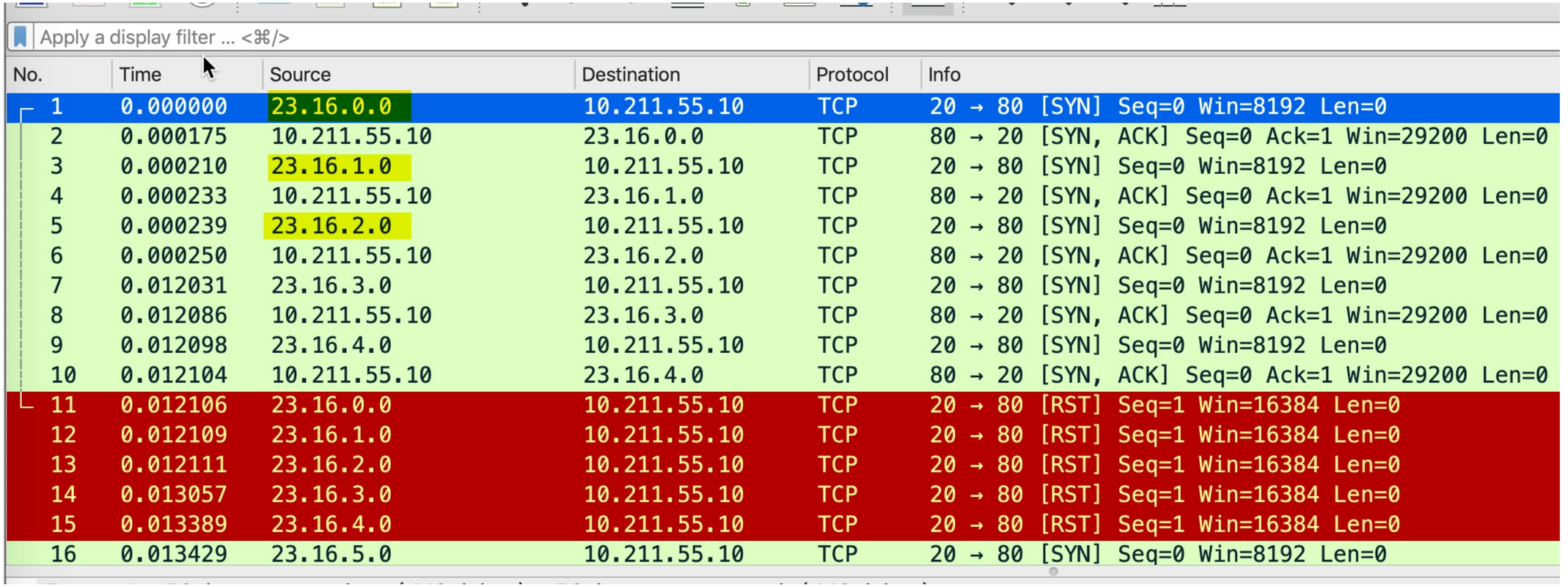
其中服务端收到的 SYN 包的源地址将会是 23.16 网段内的随机 IP，隐藏了自己的 IP。

服务端状态

```
netstat -lnpat | grep :80

tcp        0      0 0.0.0.0:80              0.0.0.0:*               LISTEN      -
tcp        0      0 10.211.55.10:80         23.16.63.3:20           SYN_RECV    -
tcp        0      0 10.211.55.10:80         23.16.64.3:20           SYN_RECV    -
tcp        0      0 10.211.55.10:80         23.16.62.3:20           SYN_RECV    -
```

在服务端抓包看到下面的抓包



可以看到短时间内，服务端收到了很多虚假 IP 的 SYN 包，马上回复了 SYN+ACK 给这些虚假 IP 的服务器。这些虚假的 IP 当然一脸懵逼，我都没发 SYN，你给我发 SYN+ACK 干嘛，于是马上回了 RST。

使用 netstat 查看服务器的状态

```
netstat -lnpat | grep :80
tcp        0      0 0.0.0.0:80              0.0.0.0:*               LISTEN      -
tcp        0      0 10.211.55.10:80         23.16.63.3:20           SYN_RECV    -
tcp        0      0 10.211.55.10:80         23.16.64.3:20           SYN_RECV    -
tcp        0      0 10.211.55.10:80         23.16.62.3:20           SYN_RECV    -
```

服务端的 SYN_RECV 的数量偶尔涨起来又降下去，因为对端回了 RST 包，这条连接在收到 RST 以后就被从半连接队列清除了。如果攻击者控制了大量的机器，同时发起 SYN，依然会对服务器造成不小的影响。

而且 SYN+ACK 去到的不知道是哪里的主机，是否回复 RST 完全取决于它自己，万一它不直接忽略掉 SYN，不回复 RST，问题就更严重了。服务端以为自己的 SYN+ACK 丢失了，会进行重传。

模拟不回复 RST 的场景

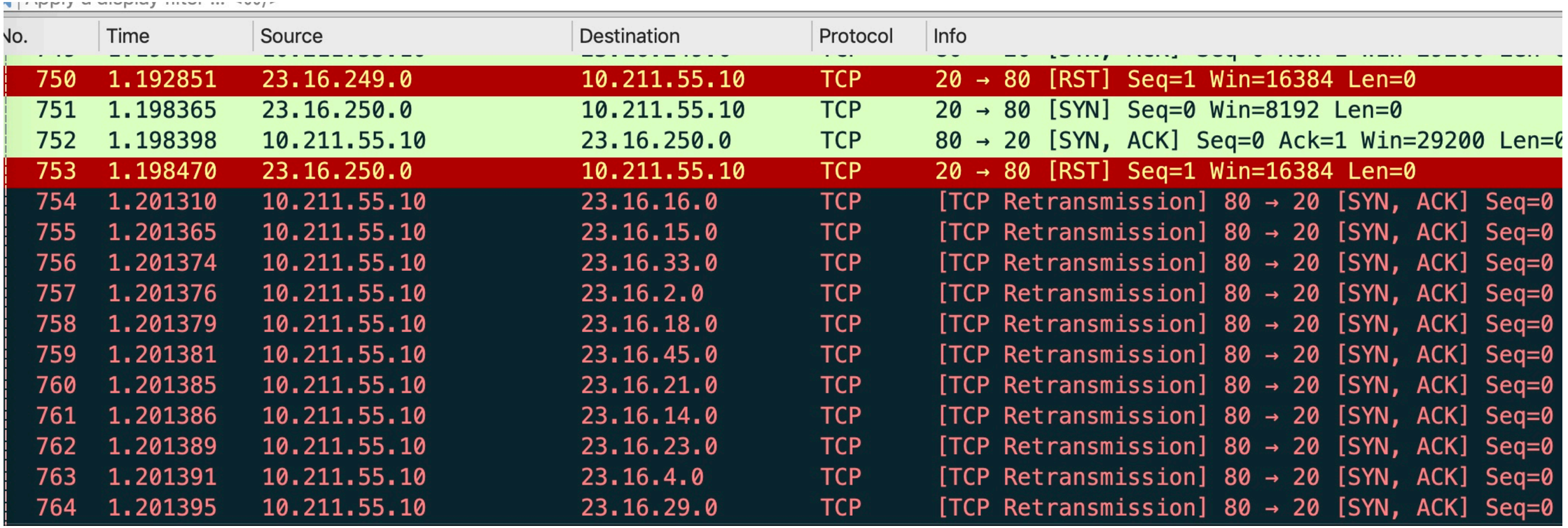
在服务器用 iptables 墙掉主机发过来的 RST 包，模拟主机没有回复 RST 包的情况。

```
sudo  iptables --append INPUT  --match tcp --protocol tcp --dst 10.211.55.15 --dport 80 --tcp-flags RST RST --jump DROP
```

这个时候再次使用 netstat 查看，满屏的 SYN_RECV 出现了

```
 will not be shown, you would have to be root to see it all.)
tcp        0      0 0.0.0.0:80              0.0.0.0:*               LISTEN      -
tcp        0      0 10.211.55.10:80         23.16.201.0:20          SYN_RECV    -
tcp        0      0 10.211.55.10:80         23.16.235.0:20          SYN_RECV    -
tcp        0      0 10.211.55.10:80         23.16.203.0:20          SYN_RECV    -
tcp        0      0 10.211.55.10:80         23.16.136.0:20          SYN_RECV    -
tcp        0      0 10.211.55.10:80         23.16.215.0:20          SYN_RECV    -
tcp        0      0 10.211.55.10:80         23.16.176.0:20          SYN_RECV    -
tcp        0      0 10.211.55.10:80         23.16.221.0:20          SYN_RECV    -
tcp        0      0 10.211.55.10:80         23.16.147.0:20          SYN_RECV    -
tcp        0      0 10.211.55.10:80         23.16.254.0:20          SYN_RECV    -
tcp        0      0 10.211.55.10:80         23.16.178.0:20          SYN_RECV    -
tcp        0      0 10.211.55.10:80         23.16.238.0:20          SYN_RECV    -
tcp        0      0 10.211.55.10:80         23.16.194.0:20          SYN_RECV    -
tcp        0      0 10.211.55.10:80         23.16.140.0:20          SYN_RECV    -
tcp        0      0 10.211.55.10:80         23.16.230.0:20          SYN_RECV    -
tcp        0      0 10.211.55.10:80         23.16.146.0:20          SYN_RECV    -
tcp        0      0 10.211.55.10:80         23.16.222.0:20          SYN_RECV    -
tcp        0      0 10.211.55.10:80         23.16.192.0:20          SYN_RECV    -
tcp        0      0 10.211.55.10:80         23.16.187.0:20          SYN_RECV    -
tcp        0      0 10.211.55.10:80         23.16.189.0:20          SYN_RECV    -
tcp        0      0 10.211.55.10:80         23.16.172.0:20          SYN_RECV    -
tcp        0      0 10.211.55.10:80         23.16.167.0:20          SYN_RECV    -
tcp        0      0 10.211.55.10:80         23.16.170.0:20          SYN_RECV    -
tcp        0      0 10.211.55.10:80         23.16.225.0:20          SYN_RECV    -
tcp        0      0 10.211.55.10:80         23.16.239.0:20          SYN_RECV    -
tcp        0      0 10.211.55.10:80         23.16.159.0:20          SYN_RECV    -
```

通过服务端抓包的文件也可以看到，服务端因为 SYN+ACK 丢了，然后进行重传。重传的次数由/proc/sys/net/ipv4/tcp_synack_retries文件决定，在我的 Centos 上这个默认值为 5。



No.	Time	Source	Destination	Protocol	Info
750	1.192851	23.16.249.0	10.211.55.10	TCP	20 → 80 [RST] Seq=1 Win=16384 Len=0
751	1.198365	23.16.250.0	10.211.55.10	TCP	20 → 80 [SYN] Seq=0 Win=8192 Len=0
752	1.198398	10.211.55.10	23.16.250.0	TCP	80 → 20 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0
753	1.198470	23.16.250.0	10.211.55.10	TCP	20 → 80 [RST] Seq=1 Win=16384 Len=0
754	1.201310	10.211.55.10	23.16.16.0	TCP	[TCP Retransmission] 80 → 20 [SYN, ACK] Seq=0
755	1.201365	10.211.55.10	23.16.15.0	TCP	[TCP Retransmission] 80 → 20 [SYN, ACK] Seq=0
756	1.201374	10.211.55.10	23.16.33.0	TCP	[TCP Retransmission] 80 → 20 [SYN, ACK] Seq=0
757	1.201376	10.211.55.10	23.16.2.0	TCP	[TCP Retransmission] 80 → 20 [SYN, ACK] Seq=0
758	1.201379	10.211.55.10	23.16.18.0	TCP	[TCP Retransmission] 80 → 20 [SYN, ACK] Seq=0
759	1.201381	10.211.55.10	23.16.45.0	TCP	[TCP Retransmission] 80 → 20 [SYN, ACK] Seq=0
760	1.201385	10.211.55.10	23.16.21.0	TCP	[TCP Retransmission] 80 → 20 [SYN, ACK] Seq=0
761	1.201386	10.211.55.10	23.16.14.0	TCP	[TCP Retransmission] 80 → 20 [SYN, ACK] Seq=0
762	1.201389	10.211.55.10	23.16.23.0	TCP	[TCP Retransmission] 80 → 20 [SYN, ACK] Seq=0
763	1.201391	10.211.55.10	23.16.4.0	TCP	[TCP Retransmission] 80 → 20 [SYN, ACK] Seq=0
764	1.201395	10.211.55.10	23.16.29.0	TCP	[TCP Retransmission] 80 → 20 [SYN, ACK] Seq=0

重传 5 次 SYN+ACK 包，重传的时间依然是指数级退避（1s、2s、4s、8s、16s），发送完最后一次 SYN+ACK 包以后，等待 32s，服务端才会丢弃掉这个连接，把处于SYN_RECV 状态的 socket 关闭。

危害

在这种情况下，一次恶意的 SYN 包，会占用一个服务端连接 63s（1+2+4+8+16+32），如果这个时候有大量的恶意 SYN 包过来连接服务器，很快半连接队列就被占满，不能接收正常的用户请求。

如何应对 SYN Flood 攻击

常见的有下面这几种方法

- 增加 SYN 连接数：tcp_max_syn_backlog（心理安慰）
- 减少SYN+ACK重试次数：tcp_synack_retries（用处不大）
- 启用 tcp_syncookies（诸多局限）

SYN Cookie 机制

SYN Cookie 技术最早是在 1996 年提出的，最早就是用来解决 SYN Flood 攻击的。

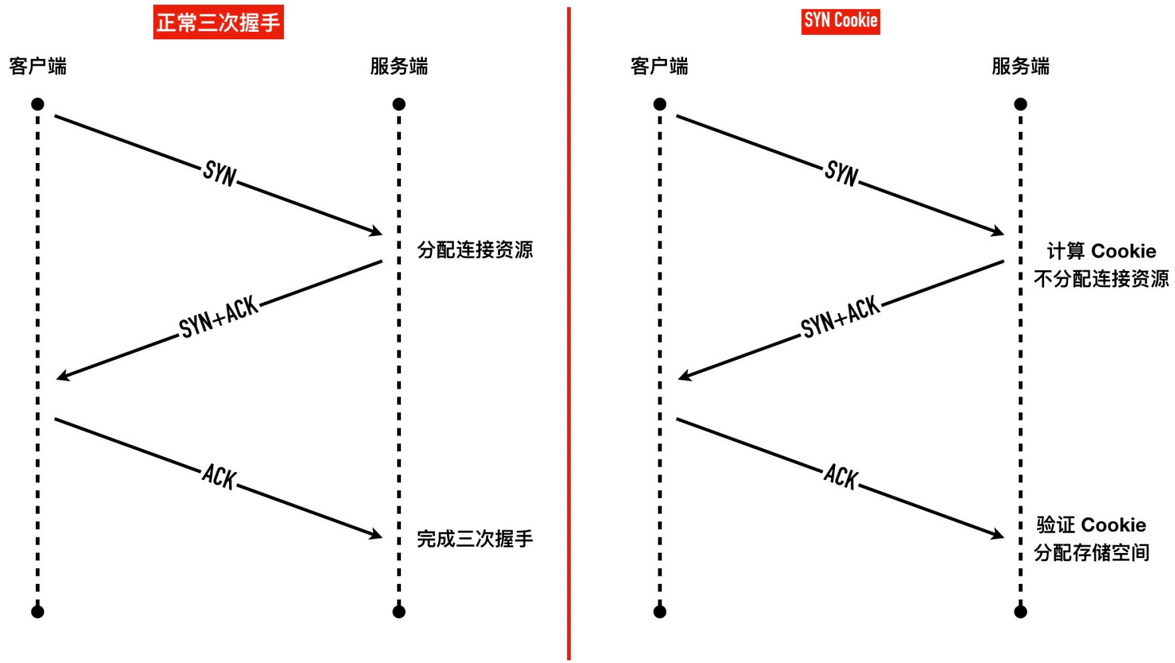
SYN Cookie 参数配置

由/proc/sys/net/ipv4/tcp_syncookies控制

- 1 表示连接队列满时启用（默认）
- 0 表示禁用
- 2 表示始终启用

SYN Cookie 原理

SYN Cookie 的原理是基于「无状态」的机制，服务端收到 SYN 包以后不马上分配为 Inbound SYN分配内存资源，而是根据这个 SYN 包计算出一个 Cookie 值，作为握手第二步的序列号回复 SYN+ACK，等对方回应 ACK 包时校验回复的 ACK 值是否合法，如果合法才三次握手成功，分配连接资源。



Cookie 值的计算规则

Cookie 值的计算规则是怎么样的呢？Cookie 总长度是 32bit。这部分的源码见 Linux 源码：syncookies.c

```
static __u32 secure_tcp_syn_cookie(__be32 saddr, __be32 daddr, __be16 sport,
                   __be16 dport, __u32 sseq, __u32 data)
{
    u32 count = tcp_cookie_time(); // 系统开机经过的分钟数
    return (cookie_hash(saddr, daddr, sport, dport, 0, 0) + // 第一次 hmac 哈希
        sseq + // 客户端传过来的 SEQ 序列号
         (count << COOKIEBITS) + // 系统开机经过的分钟数左移 24 位
        ((cookie_hash(saddr, daddr, sport, dport, count, 1) + data)
         & COOKIEMASK)); // data 包含了 MSS，增加 MSS 值做第二次 hmac 哈希然后取低 24 位
}
```

其中 COOKIEBITS 等于 24，COOKIEMASK 为 低 24 位的掩码，也即 0x00FFFFFF，count 为系统的分钟数，sseq 为客户端传过来的 SEQ 序列号。

SYN Cookie 的缺陷

SYN Cookie 看起来比较完美，但是也有不少的问题。

- MSS 值只能是少数的几种，由数组 msstab 值决定
- 因为 syn-cookie 是一个无状态的机制，服务端不保存状态，不能使用其它所有 TCP 选项，比如 WScale，SACK 这些（可以通过其它的 hack 的方式来实现）

小结

- SYN Flood 攻击是什么
- 用 Scapy 工具构造 SYN Flood 攻击
- 缓解 SYN Flood 攻击的几种方式