Linux 下基于 SYN Cookie 的防 SYN Flood 攻击的实现与改进①

黄晗辉[1] 陈蜀宇[2] 石晶翔[1](1.重庆大学 计算机学院 重庆 400030;
2.重庆大学 软件学院 重庆 400030)

摘 要: DoS 的思想是用大量的数据包来攻击服务器，降低服务器的性能。SYN Flood 攻击是 DoS 攻击的一种重要形式,它是利用 TCP 协议 3 次握手时的漏洞对服务器进行攻击。介绍了 SYN Flood 的攻击原理，研究了 Linux 2.6 内核下 SYN Cookie 的实现，并在此基础上提出了一种改进方法。

关键词: SYN Flood; SYN Cookie; DoS 攻击; 3 次握手

Implementation and Improvement Against SYN Floody Attack Based on SYN Cookie Under Linux

HUANG Han-Hui[1], CHEN Shu-Yu[2], SHI Jing-Xiang[1]
(1.Computer College, Chongqing University, Chongqing 400030, China; 2. Chongqing University, Chongqing 400030, China)

Abstract: The principle of DoS is to attack servers to degrade their capability with a lot of data packets. SYN Flood attack is an important way of DoS attack, which uses leaks to attack servers during TCP's three-way handshake. The paper introduces the attack principle of SYN Flood and the implementation of SYN Cookie based on Linux 2.6 kernel. Finally, it proposes an improved method.

Keywords: SYN Flood; SYN Cookie; DoS attack; three-way handshake

引言

拒绝服务(DoS)攻击基于这样的思想：用大量的数据包对目标系统进行洪水攻击，用这种方式来破坏或严重降低 Internet 连接，将本地服务器消耗到极限以至于无法响应合法的请求，或者最糟糕的情况就是让目标系统崩溃。在目前网络环境中，SYN Flood 是一种非常危险而常见的 DoS 攻击方式。到目前为止，能够有效防范 SYN Flood 攻击的手段并不多，而 SYN Cookie 就是其中最著名的一种。SYN Cookie 原理由 D. J. Bernstain 和 Eric Schenk 发明[1]，在很多操作系统上都有各种各样的实现。

1 TCP/IP三次握手规则

TCP/IP 的连接和建立都是采用客户服务器方式。主动发起连接建立的应用进程叫做客户，而被动等待连接建立的应用进程叫做服务器[2]。设主机 B 中运行一个服务器进程(如图 1)，它先发出一个被动打开命令，告诉它的 TCP 要准备接受客户进程的连接请求。客户进程运行在主机 A 中，它先向其 TCP 发出主动打开命令，表明要向某个 IP 地址的某个端口建立运输连接。A 和 B 之间的通信遵守 TCP/IP 的三次握手规则。其操作步骤为：

(1) 主机 A 向主机 B 发出连接请求报文段，其首部中的同步比特 SYN 应置为 1，同时选择一个序号 X，表明在后面传送数据时的第一个数据字节的序号是 X+1。

(2) 主机 B 在收到连接请求报文段后，如同意，则发回确认。在确认报文段中应将 SYN 和 ACK 都置

① 基金项目:科技部科技型中小企业技术创新基金(05C26215111378)
收稿时间:2009-06-13

1，确认号为 X+1,同时也为自己选择一个序号 Y。

(3) 主机 A 收到 B 的确认后，要向 B 给出确认，其 ACK 置 1，确认号为 Y+1,而自己的序号为 X+1。

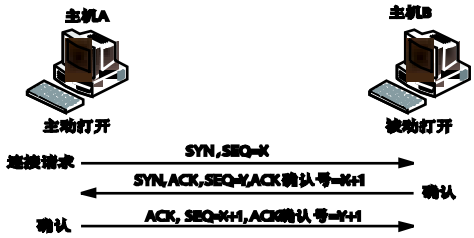

图 1　TCP/IP 三次握手

2　SYN Flood攻击原理

SYN Flood 攻击是一种典型的拒绝服务型(Denial of Service,DoS)攻击。所谓拒绝服务型攻击就是通过进行攻击，使受害主机或网络不能够良好的提供服务，从而间接达到攻击的目的。

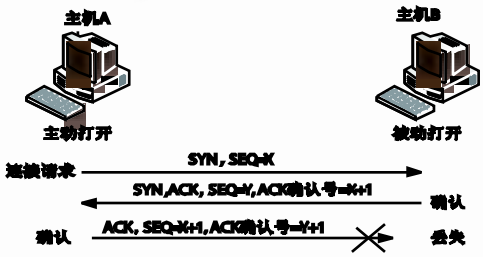

图 2　半开连接状态

SYN Flood 攻击就是利用了“三次握手”的漏洞。假设一个用户向服务器发送了SYN报文后突然死机或掉线，那么服务器在发出 SYN+ACK 应答报文后是无法收到客户端的 ACK 报文的(第三次握手无法完成)，一般把服务器收到 SYN 包而还未收到 ACK 包时的连接状态成为半开连接[3](如图 2)。这时服务器端一般会重试(再次发送 SYN+ACK 给客户端)并等待一段时间后丢弃这个未完成的连接，这段时间的长度我们称为 SYN Timeout，一般来说这个时间是分钟的数量级(大约为 30 秒-2 分钟)。如果有一个恶意的攻击者大量模拟这种情况(如图 3)，服务器端将为了维护一个非常大的半连接列表而消耗非常多的资源----数以万计的半连接，即使是简单的保存并遍历也会消耗非常多的 CPU 时间和内存，何况还要不断对这个列表中的 IP 进行 SYN+ACK 的重试。服务器端将忙于处理攻击者伪造的 TCP 连接请求而无暇理睬客户的正常请求，此时从正常客户的角度看来，服务器失去响应，这种情况就称作：服务器端受到了 SYN Flood 攻击(SYN 洪水攻击)。

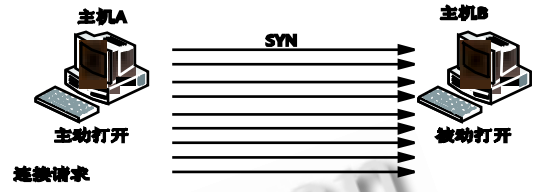

图 3　SYN Flood 攻击

3　SYN Cookie原理及实现

3.1 SYN Cookie 原理

SYN cookie 就是用一个 cookie 来响应 TCP SYN 请求的 TCP 实现，根据上面的描述，在正常的 TCP 实现中，当服务器接收到一个 SYN 数据包，它返回一个 SYN+ACK 包来应答，然后进入 TCP_SYN_RECV (半开放连接)状态来等待最后返回的 ACK 包[4]。服务器用一个数据空间来描述所有未决的连接，然而这个数据空间的大小是有限的，所以攻击者将塞满这个空间。SYN Cookie 是对 TCP 服务器端的三次握手协议作一些修改，专门用来防范 SYN Flood 攻击的一种手段。它的原理是，在 TCP 服务器收到 TCP SYN 包并返回 TCP SYN+ACK 包时，不分配一个专门的数据区，而是根据这个 SYN 包计算出一个 cookie 值，并将这个 cookie 作为将要返回的 SYN+ACK 包的初始序列号。在收到 TCP ACK 包时，TCP 服务器再根据报头信息获取 cookie 值，并通过这个 cookie 值检查这个 TCP ACK 包的合法性。如果合法，再分配专门的数据区进行处理未来的 TCP 连接。

由此可见，cookie 的实现要求 cookie 必须与每次 TCP 连接紧密对应，攻击者无法伪造 cookie，同时 cookie 中包含连接的状态信息。

cookie 的生成验证算法如下[5]：

$H_1=hash_{32-61}(S_{addr}|S_{port}|D_{addr}|D_{port}|K_1)$

$H_2=hash3_{2-61}(S_{addr}|S_{port}|D_{addr}|D_{port}|counter|K_2)$

cookie 的生成：

$cookie=H_1+ISN_{client}+(counter \times 224)+ (H_2+data) mod\ 2^{24}$

cookie 的验证：

$counter_{cookie}=(cookie-H_1-ISN_{client}))\div 2^{24}$

$\triangle counter=counter_{current}-counter_{cookie}$

$data=(cookie-H_1-ISN_{client}) \bmod 2^{24}-H_2 \bmod 2^{24}$

其中 $hash_n(x)$:n 表示位数,利用 MD5 或 SHA-1 生成

S_{addr}，S_{port}，D_{addr}，D_{port} ：表示源/目的 IP 地址和端口

K_1 ，K_2 ：表示密钥

ISN_{client} ：表示客户端进行连接时使用的序列号

counter:用来表示时间记数

△counter：用来表示记数差，必须小于 4

data:24 位的值，用来表示 MSS 值对应的 3 位编码的索引值。

在这个算法中，cookie 是通过两个密钥 K_1 和 K_2 生成和验证的。这两个密钥必须足够长来覆盖所使用哈希函数的输入缓冲。其中 data 的值是服务器定义的值，它用来存储预定义 MSS 值的 3 位编码的索引值。密钥 K_1 和 K_2 以及 ISN 值都是通过随机数生成器产生的。

3.2 Linux 2.6 内核下 SYN Cookie 的实现分析

Linux 内核对 TCP 流程的处理主要在 net/ipv4/tcp_ ipv4.c 和 syncookies.c 文件中的函数实现。目前的 cookie 是一个__u32 的值，相关的算法实现主要包括两部分：

具体的，当收到 TCP SYN 包时，系统进入 tcp_v4_conn_ request()函数，其中调用 cookie_v4_init_sequence ()生成一个 ISN(Initial Sequence Number)。Linux 内核把它作为 SYN Cookie 流程中的 cookie。cookie_v4_init_ sequence()函数在 syncookies.c 文件中定义，它又调用 random.c 文件中的 secure_tcp_syn_ cookie()函数。cookie 的实质计算是在这个函数中进行的。

在服务器收到 TCP ACK 包时，相应的要进行 SYN Cookie 的检查。这个检查过程在函数 tcp_v4_hnd_req()中的 cookie_v4_check()函数开始。cookie_v4_check()调用 cookie_check()函数，cookie_check()函数调用 check_tcp_syn_cookie()函数，该函数在 random.c 中定义。

此外，在初始化时，需要调用 init_syncookie()初始化数组 syncookie_secret[2][16-3+HASH_BUFFER_SIZE]，其中所有的比特值在 secure_tcp_syn_cookie()中用 get_ random_bytes()函数随机的赋予，它们成为制作 cookie 的密钥。

3.3 Linux 2.6 内核下 SYN Cookie 的计算

当收到 TCP SYN 包，根据 secure_tcp_syn_cookie()函数中的代码，我们可以得出，cookie 值主要是由一些信息及其 hash 值相加得到的。关键代码内容如下：

return (cookie_hash(saddr, daddr, sport, dport, 0, 0)

+sseq + (count << COOKIEBITS)

+((cookie_hash(saddr, daddr, sport, dport, count, 1) + data)& COOKIEMASK))

其中，COOKIEBITS 为 24，COOKIEMASK 为低 24 位的掩码，即((__u32)1 << COOKIEBITS)- 1。而 cookie_hash()的前四个参数为源/目的地址和端口信息，第五个参数 count 是系统的分钟数，用 jiffies/(HZ*60)计算得到。第六个参数为 0 或 1，指明使用 syncookie_secret[0] 还是 syncookie_secret[1]来计算 hash 值,data 是从前往后最后一个小于 skb 中携带的 MSS 值的值的索引。

由此可知，Cookie 一共由三部分相加而成：

① 以源/目的地址和端口、0 以及 syncookie_secret[0]为输入的 hash 值

② sseq 序列号，sseq = ntohl(skb->h.th->seq) 这里的 skb 是携带 TCP SYN 的那个 skb。

③ 高 8 位为当前分钟数，即 jiffies/(60*HZ)，低 24 位为 hash 值+data 后的 24 位，其中，hash 值以源/目的地址和端口、当前分钟数和 syncookie_secret[1]为输入。在 cookie 中加入的 data 只包括了低 24 位信息，data 其实只保存了协商后的 MSS 值在 msstab 中的索引值。

3.4 Linux 2.6 内核下 SYN Cookie 的校验

当服务器收到 TCP ACK 包时,就能获取到该包的 cookie 值，此时需要对该值进行检查,主要采取的方法是：

(1) 检查时间值

首先去掉 cookie 值的①②两部分,计算如下：

cookie -= cookie_hash(saddr, daddr, sport, dport, 0, 0) + sseq;

其次取出 cookie 的高八位，即发送 SYN+ACK 时编码进去的当时的系统分钟数，用当前分钟数去减，

即为前后时间的差值，精确到分钟，即

diff = (count - (cookie >> COOKIEBITS)) & ((__u32) - 1 >> COOKIEBITS) 。

如果 diff 值过大，则认定 cookie 非法，其中 maxdiff 的值为 COUNTER_TRIES(4 分钟)

if (diff >= maxdiff)　return (__u32)-1

(2) 如果 diff 值合理，则检验 data 值

首先计算出 data 的低 24 位：(cookie-cookie_hash(saddr, daddr, sport, dport, count - diff, 1)) & COOKIEMASK

然后判断 data 的低 24 位是否合法，即判断 data 的低 24 位是否属于 msstab 的合法索引，如果合法，则通过校验，否则，校验失败，判断 ACK 报文非法。

4　SYN Cookie存在的问题及改进

SYN Cookie 思想从根本上解决了三次握手带来的 SYN Flood 攻击，但是却引进新的问题：服务器会对发送出的 SYN+ACK 包和收到的 ACK 包进行复杂的计算。

首先是 SYN Cookie 会引起攻击方发起一种 ACK Flood 攻击。服务器方要想完成对一个 ACK 报文的核查必须要根据四元组信息和时间信息以及密钥来计算，然后再进行比较，同时还要兼顾时间的判断，造成了一定的延迟，扩大了连接建立时间。如果攻击方跳过发送 SYN 报文这一步骤，而是伪造了大量的 ACK 报文直接发往服务器，不管 ACK 是否有效，服务器都要进行处理，运行一个非常复杂的算法，造成了 CPU 资源的极大浪费，同时还会影响正常的客户端连接。

其次是 SYN Cookie 思想由于没有预先分配资源，所以 TCP 的超时重传定时器这里就用不到了。因为服务器端并没有给这次半连接分配任何资源。服务器不能判断这次连接请求是否已经发送了 SYN，只能通过复杂的算法计算后才能确定该 ACK 报文是否有效。

在正常的 TCP 连接中，一个 SYN 包总是对应着一个 FIN 包或者一个 RESET 包，也就是说 SYN 包的数量应该与 FIN 和 RESET 包的数量总和有一种较稳定的联系，我们可以统计 SYN 包与 FIN 包和 RESET 包的数目之差，将其归一化后再与某个门限值(可根据实际情况设定)比较，就可判定攻击的存在与否，进而决定是否采取 SYN Cookie 机制。

在内核中先建立两个模块：监测模块和过滤模块(如图 4)。其中过滤模块中包含了合法 IP 表和非法 IP 表：一张用来存放最近一段时间访问服务器通过 cookie 验证合法的 IP,另一张用来存放被过滤掉的不合法的 IP 记录。由于 Hash 算法在快速寻找目标中的优势，采用 Hash 算法来存储两张表。

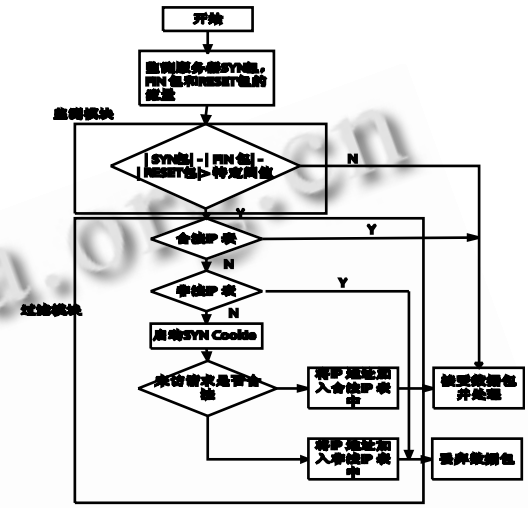

图 4　改进后的流程图

5　实验

按照图 5 所示的网络拓扑图组建局域网，并在局域网内进行了一系列攻击实验，其中利用 20 台傀儡机同时开多个线程对服务器 80 端口进行攻击来模拟高流量、高强度的 SYN Flood 攻击。

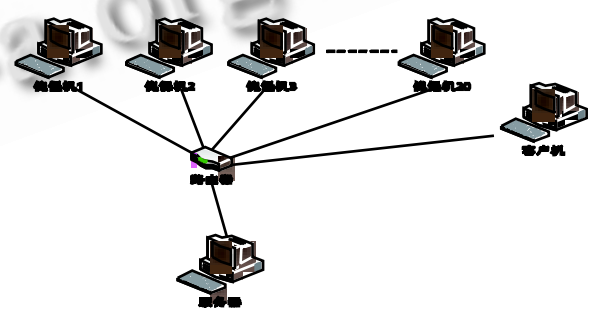

图 5　网络拓扑图

实验中服务器的 CPU 为 P4 2.8G，内存 1G，采用 Red Hat Linux 9 操作系统。图 6 为 SYN Flood 攻击时服务器 CPU 使用率分布图，实验结果表明在 SYN Flood 攻击状态下,当百兆带宽被占满时,服务器的 CPU 也没有被耗尽,服务器仍能够正常工作，客户机仍能够正常的访问服务器 Web 页面。实验证明，对

SYN Cookie 的改进在工作效率上有一定的优越性。

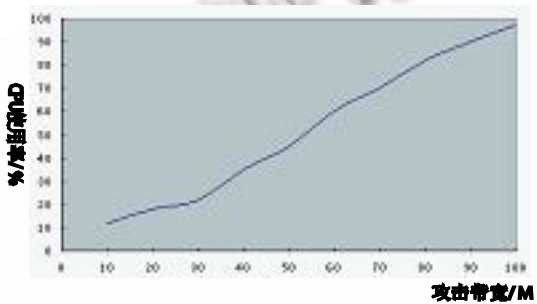

图 6 攻击时 CPU 使用率

6 结语

现在普遍使用的网络协议带有很多安全上的问题，其中面对 SYN Flood 攻击的软弱就是一点。在不变 TCP 三次握手流程的情况下，TCP Server 几乎不可能有效的防范 SYN Flood 的攻击。要保证完全防范 SYN Flood，必须修改三次握手协议。SYN Cookie 是一种很有效的方法。本文通过研读 Linux2.6 内核中的代码，了解了 Linux 内核中 SYN Cookie 的实现，并在 SYN Cookie 机制上增加了包过滤模块，能够更好的防卸 SYN Flood 攻击。

参考文献

1 沈清,金心宇,周绮敏.基于SYN Cookie下防分布式拒绝服务攻击算法的分析与实现.计算机应用, 2005,25(12):2745－2747.

2 吴虎,刘云超,陈挺.对 DDoS 攻击防范策略的研究及若干实现.计算机应用研究, 2002,19(8):34－36.

3 SCHUBA C.Analysis of a Denial of Service Attack on TCP. IEEE Security and Privacy Conference. 1997. 208－230.

4 Denial of service Attacking with TCP SYN flooding [2009-01].http://www.cert.org.tw

5 Chang R. Defending against flooding-based distributed denial-of-service attacks: A tutorial. IEEE Communica