# LVS SYNPROXY概要设计

By 陈建 @360

## 1 背景

LVS当前的主要功能是负载均衡和失败冗余，不能对flood类型的包攻击进行防护，导致攻击流量会被转发到后端RS上，而RS对此类攻击的防护工作TCP层，防护效率不高，同时大流量的包转发也会大大消耗LVS机器的CPU，影响LVS的正常工作。

为了解决上述问题，

1. 将在LVS上增加一种攻击防护功能：Syn-Proxy，该功能借鉴TCP的syn-cookie机制，代理Client和RS之间的TCP连接握手过程，在与Clinet三次握手完成后，才去和RS建立连接，从而实现对synflood攻击的防御；
2. 采用“session不存在直接丢弃”的策略，实现了对ack/fin/rst 其它TCP标志位攻击的防御；
3. 采用“直接丢弃”的策略，实现UDP/IP FRAG DDOS攻击的防御；

注：Syn-Proxy实现较复杂，下文将重点介绍；

## 2目标

Syn-Proxy作为一种攻击防护功能，实现如下功能：

1. 代理与Client的三次握手，包括：
2. Client的Syn报文到来时，采用TCP层的syn-cookie算法，回复Syn/Ack报文；
3. Client三次握手的Ack报文到来时，利用syn-cookie算法进行校验，校验不通过则丢弃报文，校验通过则完成与Client的三次握手；

另外，还实现了TCP层的defer功能，在开启defer功能的情况下，只有客户端发来的Ack报文中带有payload时，才真正完成与Client的三次握手，否则将丢弃该Ack报文。

1. 代理与RS的三次握手，包括：
2. 向RS发送Syn报文；
3. 收到RS的Syn/Ack报文后，回复Ack报文，并记录RS Syn/Ack报文与LVS返回的Syn/Ack报文的Seq之差；
4. 后续报文的序列号修正；

功能要求：Syn-Proxy工作在NAT/FULLNAT转发模式下。

## 3总体思路

1. 增加NF\_INET\_PRE\_ROUTING处HOOK点处理函数ip\_vs\_pre\_routing，用来处理Client发来的Syn报文，回复Syn/Ack;
2. 在tcp\_conn\_schedule函数中，增加对Ack报文的cookie校验，校验通过创建session，把Ack报文保存在Session结构体中，并向RS发送syn报文；
3. 在ip\_vs\_out函数中，增加对RS发来的Syn/Ack报文的处理，更新syn\_proxy\_seq，并向RS发送session中保存的Ack报文。
4. 序列号处理：
5. Packet IN时，在tcp\_dnat\_handler/tcp\_fnat\_in\_handler中，根据syn\_proxy\_seq修正Ack seq，并修正Sack选项中的Ack seq；
6. Packet Out时，在tcp\_snat\_handler/tcp\_fnat\_out\_handler中，根据syn\_proxy\_seq修改Seq；

## 4关键技术点

（注：本次给出的是大部分是IPV4的实现，V6的实现类似）

### 4.1 Syn Cookie处理

Syn Cookie处理包括两个部分：根据收到Syn报文，根据Cookie算法计算返回的Syn/Ack Seq；根据收到的Ack报文，校验Ack seq是否合法；

目前TCP层已经实现了syn-cookie算法，但是syn-cookie和TCP的耦合性比较大：

1）cookie生成算法中需要传入sock对象；

2）cookie校验算法中需要传入sock对象，并且会在cookie校验函数中生成新到请求的sock，这个就需要对现有的cookie校验接口进行修改，才能实现。

3）cookie算法中对Timestamp等Opt的支持函数为cookie\_init\_timestamp，该接口需要request\_sock作为参数，为了使用，也需要对该接口进行修改。

因此，本设计会给出一种新的Cookie计算校验方法和接口，该接口可以方便地被LVS模块调用，同时为了简便起见，不同于TCP层的Syn Cookie算法，本设计会把TCP选项信息放入回复的序列号中，即cookie计算与解析过程中，加入TCP选项信息。

为了实现该方法，需要在现有文件中做如下修改：

1. IPV4支持：在net/ipv4/syncookies.c文件中增加如下函数：
2. Cookie计算函数：ip\_vs\_synproxy\_cookie\_v4\_init\_sequence
3. Cookie校验函数：ip\_vs\_synproxy\_v4\_cookie\_check
4. IPV6支持：在net/ipv6/syncookies.c文件中增加如下函数：
5. Cookie计算函数：ip\_vs\_synproxy\_cookie\_v6\_init\_sequence
6. Cookie校验函数：ip\_vs\_synproxy\_v6\_cookie\_check

### 三次握手报文的处理

Syn Proxy需要对三次握手报文的处理包括如下部分：

1. Client 发来的Syn报文：尝试复用该SKB生成Syn/Ack报文；
2. Client 发来的Ack报文：根据该报文生成发向RS的Syn报文，并保存该Ack报文；
3. RS发来的Syn/Ack报文：把保存的Ack报文发向RS。

#### 4.2.1 Client的Syn报文处理

在函数ip\_vs\_synproxy\_syn\_rcv中实现，该函数在ip\_vs\_pre\_routing中被调用，并且是主要实现逻辑，如下：

1. 判断是否可以复用SKB，不行的话就是用skb\_copy重新创建一个；
2. 设置TCP option：调用自定义的syn\_proxy\_parse\_set\_opts函数，处理Timestamp、Wscale、Sack等。（具体内容见4.4）。
3. 计算Cookie。
4. 设置TCP头部标志位（Syn+Ack）、互换源目的IP、源目的端口、Seq和Ack seq及ttl和tos。
5. 计算IP层checksum；
6. 计算TCP层checksum；
7. 互换源目的MAC；
8. 发送报文：调用dev\_queue\_xmit函数；

#### 4.2.2 Client的Ack报文处理

在函数ip\_vs\_synproxy\_ack\_rcv中实现，该函数在tcp\_conn\_schedule中被调用，并在原有session创建代码的前面执行，其主要逻辑如下：

1. 验证cookie及TCP OPT：调用syn\_proxy\_v4\_cookie\_check：
2. 选择RS，并创建Session，创建成功，则继续处理（具体内容见4.3）。
3. 向RS发送发送Syn报文：在syn\_proxy\_send\_rs\_syn函数中处理。

#### 4.2.2 RS的Syn/Ack报文处理

在函数ip\_vs\_synproxy\_synack\_rcv中实现，在handle\_response中被调用，主要逻辑如下：

1. 获取session锁；
2. 判断是否是RS发来的Syn/Ack报文，如果判断成功，则继续；
3. 计算syn\_proxy\_seq：
4. 向RS发送Ack报文：

### TCP OPT处理

参照函数tcp\_parse\_options；

#### 4.4.1 timestamp等Option过滤

和TCP层保持一致，读取/proc/net/ipv4/下的timestamp、wscale和sack的开关，把不支持的内容设置为NOP。

在函数syn\_proxy\_parse\_set\_opts中实现，在处理Client的Syn报文时被调用。

注：当前不支持timestamp；

#### 4.4.2调整MSS

在Syn Proxy开启时，LVS代理与Client的三次握手，因此需要在回复的Syn/Ack报文中设置MSS。

为了方便实现，我们在此定义proc参数MSS-sysctl\_ip\_vs\_synproxy\_init\_mss，作为预设值；

#### 创建RS Syn报文 Tcp Opt

在完成与Client的三次握手后，LVS需要向RS发送Syn报文，此时需要根据Ack报文中的Option信息，创建Syn报文的Tcp Opt。

在函数syn\_proxy\_syn\_build\_options中实现，该函数在syn\_proxy\_send\_rs\_syn中被调用。

#### Out2In报文的Sack Opt序列号转换

在Syn Proxy开启时，LVS和RS回复的Syn/Ack报文Seq很有可能会不一致，因此在处理Out-In报文时，需要转换Sack Opt中的序列号。

在syn\_proxy\_filter\_opt\_outin函数中实现。

### 序列号转换

Client-LVS-RS的六次握手过程，会将lvs->client和rs->lvs的seq相关信息保存到syn\_proxy\_seq中，根据该信息对Session后续的报文序列号进行转换，以保证Client和RS上序列号的正确性。

### Session复用

在Syn Proxy模式下，如果三次握手的Ack报文命中LVS的已有session（Session可能处理IP\_VS\_TCP\_S\_TIME\_WAIT、IP\_VS\_TCP\_S\_CLOSE等状态），则Ack报文会被简单的转发到后端的RS，LVS不会完成六次握手过程，导致该请求不能被正确的处理。因此，需要增加Session复用功能，对命中Session的Ack报文进行特殊处理。

Session复用主要实现的功能如下：

1）从Client过来的Ack报文数据包在Session命中的情况，判断Session当前的状态，并验证报文的标志位及Cookie校验等，如果验证通过，则重新设置Session的状态及syn\_proxy\_seq等，并向RS发送Syn报文，后续的处理不需要修改；

2）支持CLOSE、TIME\_WAIT、FIN\_WAIT、CLOSE\_WAIT和LAST\_ACK状态的复用，并且对每个状态都提供单独的PROC开关来决定是否进行复用；

3）提供一个PROC开关来控制该功能是否开启。

该功能由ip\_vs\_synproxy\_reuse\_conn实现。

### 向RS发送的SYN报文重传

在Syn Proxy模式下，为了完成六次握手，LVS会向RS发送Syn报文，但没进行Syn报文的重传机制，会导致发送Syn报文丢弃时，该连接不能正常被处理。因此，需要添加Syn报文重传机制。

重传机制实现思想：当session释放时，检查Session所处状态，如果为IP\_VS\_TCP\_S\_SYN\_SENT状态，表明LVS与RS间的握手过程没有完成，于是进行判断，如果重传功能开启并且尚未达到重传次数的阈值，则将该Session保存的syn报文重新发送。

### 4.6 其他技术点

#### 4.6.1 RS回复的RST报文序列号修正

在LVS与RS进行三次握手时，如果RS不回复Syn/Ack报文，而是回复RST报文，LVS也需要进行正确的序列号处理，以便Client能够正确地释放Socket。

实现思路：

在ip\_vs\_synproxy\_synack\_rcv函数中，如果判断RS回复的为RST报文，则同样计算syn\_proxy\_seq.delta，并设置Session状态为IP\_VS\_TCP\_S\_CLOSE，同时更新Session超时时间。

#### 4.6.2 Defer功能实现

TCP协议栈的Defer功能（当第三次握手的Ack报文带有数据时，才建立连接）可有效防护慢连接攻击。在Syn Proxy基础上，可以很方便地实现该功能。

实现思路：

在 syn\_proxy\_ack\_rcv函数中，进行Cookie校验前，先判断该开关，如果开关开启，则判断该Ack报文是否有Payload，有才建立连接。

#### 4.6.3 GET请求超过一个报文的处理

LVS向RS发送Syn报文后，会等待RS发来的Syn/Ack报文，在此期间，会丢掉所有从Client发来的报文，当GET请求超过一个报文时，会导致报文被丢弃，影响请求处理的时间，为此，需要缓存这些报文。

### 4.7控制接口

Syn Proxy增加了syn proxy 开关 1个配置；

1. syn proxy 开关
   1. 该信息维护在svc->flags中，因此，需要在ip\_vs.h头文件中增加

#define IP\_VS\_CONN\_F\_SYNPROXY 0x4000 /\* syn proxy \*/

* 1. 需要修改keepalived和ipvsadm程序

## 5 其它因素考虑

### 5.1 对于其它模块的影响

Template/persistent/ipv6/UDP/ICMP各个模块的影响，处理可能考虑不全；

### 5.2 测试考虑

1. 测试NAT/DR/TUNNEL 兼容性
2. 测试TCP/UDP/ICMP/AH/ESP 数据包
3. 测试IPV6
4. 测试Template/persistent