TCP SYN_Flooding 공격의 원인과 해결책

오늘과 내일 넷센터 홍석범(antihong@tt.co.kr)

최근 자신이 운영하는 서버에 특별히 부하가 걸리거나 이상이 있는 것도 아니고
또 데몬도 정상적으로 떠 있는데, 정작 서비스가 작동하지 않는 경우가 종종 있다.
이러한 경우에는 해당 데몬을 완전히 멈추었다가 살리면 다시 작동하는데,
잠시 후에 확인해 보면 똑같은 현상이 다시 나타나곤 한다.
혹시 프로그램을 잘못 설치했나 싶어 지우고 다시 설치해도 마찬가지이다.

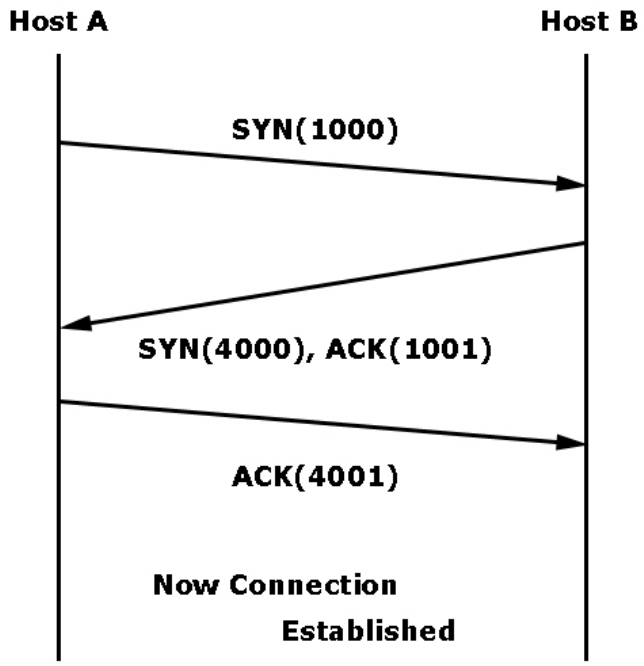
만약 최근 들어 이러한 경험이 있다면 이는 최근 유행하는 DoS(서비스 거부 공격)의 일종인 TCP SYN Flooding 공격을 당했을 가능성이 크다.

SYN Flooding 공격의 개념이 소개된지는 꽤 되었지만 최근 들어 리눅스가 확산되고, 간단하게 실행할 수 있는 공격 소스가 광범위하게 배포되면서 이 공격이 자주 확인되고 있고, 이로 인해 그 피해가 급속히 확산되고 있다. 실제로 현재 가장 많이 사용되고 있는 배포판인 레드햇 6.X 계열에 이 공격을 실행하기만 하면 단 몇 초만에 서비스가 정지해 버리게 된다.

따라서 피해가 확산되고 있는 이 공격의 원리와 대처방법에 대해 알아보도록 하자.

“TCP 의 약점을 이용한 공격원리”

SYN Flooding 공격은 TCP 의 취약점을 이용한 공격의 형태이므로 먼저 TCP 에 대해 알아야 한다. TCP 는 **Transmition Control Protocol** 의 약자로 UDP 와는 달리 신뢰성 있는 연결을 담당한다. 따라서 서버와 클라이언트간에 본격적인 통신이 이루어지기 전에는 다음 그림과 같이 소위 **"3 Way handshaking"** 이라는 정해진 규칙이 사전에 선행되어야 한다.



1 단계. A 클라이언트는 B 서버에 접속을 요청하는 SYN 패킷을 보낸다.
2 단계. B 서버는 요청을 받고 A 클라이언트에게 요청을 수락한다는 SYN 패킷과 ACK 패킷을 발송한다.
3 단계. A 클라이언트는 B 서버에게 ACK 를 보내고 이후로부터 연결이 이루어지고 본격적으로 데이터가 교환된다.

이것이 TCP 의 기본적인 Flow 이다.
그런데, 이 그림에서 악의적인 공격자가 1 단계만 요청(SYN)하고 B 서버로부터 응답을 받은 후(SYN+ACK) 3 단계, 즉 클라이언트에게 ACK 를 보내지 않는다면 어떻게 될까?
SYN+ACK 패킷을 받은 B 호스트는 A 로부터 응답이 올 것을 기대하고 반쯤 열린 이른바 "**Half Open"** 상태가 되어 대기 상태에 머무른 후 일정 시간(75 초) 후에 다음 요청이 오지 않으면 해당 연결을 초기화 하게 되는데, 초기화하기 전까지 이 연결은 메모리 공간인 백로그큐(Backlog Queue)에 계속 쌓이게 된다.

그런데, 이 위조된 연결 시도를 초기화하기 전에 위조된 새로운 요구가 계속 들어오게 된다면 또한 위조된 새로운 요구가 연결을 초기화하는 속도보다 더 빨리 이루어진다면 어떻게 될까? 이러한 경우 SYN 패킷이 어느 정도 백로그큐에 저장이 되다 결국 꽉차게 되어 더 이상의 연결을 받아들일 수 없는 상태, 즉 서비스 거부 상태로 들어가게 되는 것이다. 이처럼 백로그큐가 가득 찼을 경우에 공격을 당한 해당 포트로만 접속이 이루어지지 않을 뿐 다른 포트에는 영향을 주지 않고, 또한 서버에 별다른 부하도 유발하지 않으므로 관리자가 잘 모르는 경우가 많다. 또한 다른 DoS 공격과는 달리 많은 트래픽을 유발하는 공격이 아니므로 쉽게 파악이 되지 않는 공격 형태이다.

그렇다면 이 공격을 당하고 있는지 여부는 어떻게 알 수 있을까?
시스템에 로긴후 **"netstat"** 이라는 명령으로 확인 가능하다.

"그럼, 어떻게 파악하는가?"

netstat 은 시스템의 각종 네트워크 정보를 알려주는 명령어로 네트워크 연결, 라우팅 현황, 인터페이스 통계등의 정보를 확인할 수 있게 해 준다. 여기서 잠깐 netstat 으로 나오는 연결 상태에 대해 알아보자.
netstat -na 로 확인해 보면 Local Address, Foreign Address, State 등의 정보가 출력되는데, 이 중 State 부분에 보이는 메시지를 주목하면 된다.

참고 : State 부분에 가능한 연결상태
LISTEN : 서버의 데몬이 떠서 접속 요청을 기다리는 상태
SYS-SENT : 로컬의 클라이언트 어플리케이션이 원격 호스트에 연결을 요청한 상태
SYN_RECEIVED : 서버가 원격 클라이언트로부터 접속 요구를 받아 클라이언트에게 응답을 하였지만 아직 클라이언트에게 확인 메시지는 받지 않은 상태
ESTABLISHED : 3 Way-Handshaking 이 완료된 후 서로 연결된 상태
FIN-WAIT1 , **CLOSE-WAIT , FIN-WAIT2 :**
서버에서 연결을 종료하기 위해 클라이언트에게 종결을 요청하고
회신을 받아 종료하는 과정의 상태
CLOSING : 흔하지 않지만 주로 확인 메시지가 전송도중 분실된 상태
TIME-WAIT : 연결은 종료되었지만 분실되었을지 모를 느린 세그먼트를 위해 당분간 소켓을 열어놓은 상태
CLOSED : 완전히 종료
##

각각의 연결 상태는 통신 상황에 따라 매우 복잡하게 순간적으로 변화하는데,
이 중 주로 주목하여야 할 상태는 SYN_RECEIVED 이다. 설명에 나온 대로 이 상태는 클라이언트의 확인 메시지를 기다리는 상태이지만 특별히 전용 회선에 장애가 없는 한 이 과정은 순간적으로 일어나므로 실제 netstat 으로 확인되는 경우는 거의 없다.
따라서 **netstat -na|grep SYN_RECV** 로 확인해 보아 많은 메시지가 보인다면
Syn Flooding 공격을 당하고 있는 것으로 판단하면 된다.

“실제 테스트 공격으로 직접 확인해 보자!!”

실제 자신의 시스템이 얼마나 취약한지 자신의 시스템에 테스트해 보도록 하자.
노파심에 이야기하는 것이지만 이 공격은 반드시 자신의 시스템에서만 테스트 용도로 실행해 보기 바란다. 이 공격 소스는 인터넷상에서 쉽게 찾을 수 있다.
http://packetstorm.securify.com/ 나 http://rootshell.com/ 에 접속후 syn 으로 검색해 보면 많은 소스와 문서가 있는데, 이중 관련 파일을 다운로드받아 설치해 보면 된다.

소스에 따라 실행 방법이 다르지만 다운로드 받은 소스파일이 syn_floodinbg_dos.c 라면 **gcc –o syn_flooding_dos syn_flooding_dos.c** 로 컴파일을 한다. 이후
"./syn_flooding_dos 소 IP 공격지 IP 공격할하위포트번호 상위포트번호" 와 같이 실행하면 되는데, 필자는 **./syn_flodding_dos 0 localhost 80 80** 과 같이 테스트해 보았다.

위 명령어의 의미는 공격지 주소를 랜덤하게 무작위 IP 주소로 설정(0) 하여 localhost 서버의 80 번 포트에 Syn_Flooding 공격을 한다는 내용이다.

실제 본인이 테스트한 레드햇 6.2 서버에서는 공격후 2-3 초만에 웹서비스가 중지되었다.
테스트 공격 후 **telnet localhost 80** 으로 접속해 보기 바란다.
분명히 httpd 데몬은 떠 있는데, 접속이 되지 않을 것이다.

아래는 공격을 당한 서버에서 **netstat -na|grep SYN** 으로 SYN 패킷을 잡은 부분이다.

```
[root@net /root]# netstat -na|grep SYN
tcp        0      0 127.0.0.1:80            83.232.136.253:1911     SYN_RECV
tcp        0      0 127.0.0.1:80            199.106.229.9:1308      SYN_RECV
tcp        0      0 127.0.0.1:80            1.81.31.169:2051        SYN_RECV
tcp        0      0 127.0.0.1:80            47.95.74.60:1107        SYN_RECV
tcp        0      0 127.0.0.1:80            57.138.85.69:1697       SYN_RECV
tcp        0      0 127.0.0.1:80            54.125.11.117:2433      SYN_RECV
tcp        0      0 127.0.0.1:80            160.187.111.239:2064    SYN_RECV
tcp        0      0 127.0.0.1:80            167.129.136.239:2043    SYN_RECV
tcp        0      0 127.0.0.1:80            47.183.44.170:1256      SYN_RECV
tcp        0      0 127.0.0.1:80            124.215.160.217:1636    SYN_RECV
tcp        0      0 127.0.0.1:80            76.151.250.211:1673     SYN_RECV
tcp        0      0 127.0.0.1:80            95.168.224.46:1986      SYN_RECV
tcp        0      0 127.0.0.1:80            22.249.49.106:1362      SYN_RECV
tcp        0      0 127.0.0.1:80            21.255.176.15:1610      SYN_RECV
tcp        0      0 127.0.0.1:80            50.110.236.208:2272     SYN_RECV
tcp        0      0 127.0.0.1:80            114.218.24.238:1351     SYN_RECV
tcp        0      0 127.0.0.1:80            88.78.25.148:1215       SYN_RECV
tcp        0      0 127.0.0.1:80            104.254.58.198:2317     SYN_RECV
tcp        0      0 127.0.0.1:80            84.32.128.40:1065       SYN_RECV
tcp        0      0 127.0.0.1:80            244.164.245.106:1360    SYN_RECV
tcp        0      0 127.0.0.1:80            242.112.255.116:1032    SYN_RECV
tcp        0      0 127.0.0.1:80            130.134.146.84:2431     SYN_RECV
tcp        0      0 127.0.0.1:80            170.50.7.22:2020        SYN_RECV
tcp        0      0 127.0.0.1:80            157.180.196.126:1468    SYN_RECV
tcp        0      0 127.0.0.1:80            241.183.146.229:2024    SYN_RECV
tcp        0      0 127.0.0.1:80            105.145.93.35:1891      SYN_RECV
tcp        0      0 127.0.0.1:80            185.105.36.156:1881     SYN_RECV
```

분명히 localhost 에서 공격을 했음에도 위 그림에서처럼 80 번 포트로 SYN 패킷을 요청한 IP 주소는 랜덤하게 보이고 있어 도무지 어떤 IP 에서 공격하고 있는 것인지 알 수 없다. 실제로 공격지 IP 를 확인해 보면 대부분이 현재 인터넷상에 연결되지 않은 존재하지 않는 위조된 IP 들이다.

실제 공격 소스 코드중 소스 IP 를 생성하는 부분을 보면 아래와 같이 0 부터 255 까지 임의의 값을 뽑아 IP 주소로 설정하는 것을 확인할 수 있다.

```
{
    a = getrandom(0, 255);
    b = getrandom(0, 255);
    c = getrandom(0, 255);
    d = getrandom(0, 255);
    sprintf(junk, "%i.%i.%i.%i", a, b, c, d);
    me_fake = getaddr(junk);
 }
```

SYN_Flooding 공격에 대한 대비 및 해결책

그렇다면 이 공격에 대해 어떻게 대비하여야 할까?

1. 백로그 큐를 늘려준다.

직관적으로 보았을 때 서비스 거부에 돌입하게 되는 것은 백로그큐(Backlog Queue)가 가득 차서 다른 접속 요구를 받아들이지 못하기 때문이므로 백로그 큐의 크기를 늘려주면 될 것이다. 실제로 리눅스를 포함해서 많은 운영체제들의 백로그큐값을 조사해 보면 이 값이 필요 이상으로 작게 설정되어 있어 적절히 늘려주는 것이 좋다.
현재 시스템에 설정된 백로그큐의 크기는

[root@net /root]# sysctl -a|grep syn_backlog
net.ipv4.tcp_max_syn_backlog = 128

또는
[root@net /root]# cat /proc/sys/net/ipv4/tcp_max_syn_backlog
128
로 확인가능하며 128kb 인 것을 확인할 수 있다.

일반적으로 시스템의 RAM 이 128M 일 경우에는 128 을 설정하고 그 이상일 경우에는 1024 정도로 설정해 주는 것이 좋다. 이 때 주의할 점은 이 값을 무작정 크게 설정한다고 좋은 것은 아니며 1024 이상으로 설정할 경우는 **/usr/src/linux/include/net/tcp.h** 소스에서 **TCP_SYNQ_SIZE** 변수를 수정 후 커널을 재컴파일하여야 한다. 이 변수를 설정시 **TCP_SYNQ_HSIZE** 에 16 을 곱한 값이 **tcp_max_syn_backlog** 보다는 작거나 같아야 하는데, 그렇지 않을 경우에는 시스템에 문제가 발생할 수 있으니 1024 보다 높은 값으로 설정하지 말기 바란다. 그리고 이 값을 너무 크게 설정하였을 경우에는 경험적으로 아래 설명할 **syncookies** 기능이 잘 적용되지 않는 현상이 가끔 확인되었다.
이와는 별개로 시스템의 부하가 많이 걸릴 경우에도 백로그큐를 늘려주면 일정 정도의 효과를 볼 수 있는 것으로 알려져 있다.

백로그큐의 값을 설정하는 방법은 다음과 같다.

[root@net /root]# sysctl -w net.ipv4.tcp_max_syn_backlog=1024

또는

[root@net /root]# echo 1024 > /proc/sys/net/ipv4/tcp_max_syn_backlog

로 해도 된다.

그러나 이 방법은 임시적인 대책일 뿐, 지속적으로 많은 TCP SYN Flooding 공격을 당할 때는 결국 백로크큐가 가득 차게 되므로 근본적인 해결 방안은 아니다.

2. syncookies 기능을 켠다.

Syncookies("신쿠키" 라고 발음한다.) 는 "Three-way handshake" 진행 과정을 다소 변경하는 것으로 Alex Yuriev 와 Avi Freedman 에 의해 제안되었는데, TCP header 의 특정한 부분을 뽑아내어 암호화 알고리즘을 이용하는 방식으로 Three-way Handshake 가 성공적으로 이루어지지 않으면 더 이상 소스 경로를 거슬러 올라가지 않는다. 따라서 적절한 연결 요청에 대해서만 연결을 맺기 위해 리소스를 소비하게 되는 것이다.

syncookies 기능은 TCP_Syn_Flooding 공격을 차단하기 위한 가장 확실한 방법으로 이 기능을 이용하려면 일단 커널 컴파일 옵션에서 CONFIG_SYN_COOKIES 이 Y 로 선택되어 있어야 한다.

자신의 커널 옵션에 이 기능이 설정되어 있는지 확인하려면
/usr/src/linux 디렉토리로 이동후 make menuconfig 후
Networking options --->
[*] IP: TCP syncookie support (disabled per default)
와 같이 확인하면 된다.

만약 설정이 되어 있지 않다면 선택 후 커널 컴파일을 다시 하여야 하지만 대부분 배포판은 기본적으로 이 옵션이 선택되어 있으므로 걱정할 필요는 없다.
그러나 위와 같이 커널 옵션에 설정되어 있다 하더라도 실제 syncookies 적용은 꺼져 있으므로 이 값을 다음과 같은 방법으로 활성화해야 한다.

[root@control src]# sysctl -a|grep syncookie
net.ipv4.tcp_syncookies = 0

0 으로 설정되어 있으므로 현재 syncookies 는 적용되지 않는다.
따라서 아래와 같이 1 을 설정하여 syncookies 기능을 활성화하도록 한다.

[root@control src]# sysctl -w net.ipv4.tcp_syncookies=1

syncookies 는 백로그큐가 가득 찼을 경우에도 정상적인 접속 요구를 계속 받아들일 수 있도록 해 주므로 SYN_Flooding 공격에 대비한 가장 효과적인 방법중 하나이다.
만약 공격을 당해 syncookies 가 작동할 때에는 /var/log/messages 파일에 아래와 같이 SynFlooding 공격이 진행중이라는 메시지가 출력된다.

Jun 11 18:54:08 net kernel: possible SYN flooding on port 80. Sending cookies.

SYN_Flooding 공격이 지속적으로 매우 심하게 진행중일 때에는 syncookies 기능이 작동한다 하더라도 네트워크가 다운되는 현상이 가끔 확인되었다. 따라서 syncookies 기능 외에 몇 가지 설정도 함께 적용하는 것이 시스템의 안정성을 위해 권장하는 방법이다. 아울러 네트워크가 다운되었을 경우에는 **/etc/rc.d/init.d/network restart** 로 network 를 재설정해 보거나 reboot 를 하여야 한다.

3. 기타 시스템의 네트워크 설정을 최적화한다.

아래 설정은 비단 TCP Syn_Flooding 공격뿐만이 아니라 다른 여타 DoS 공격에도 효과적이으로 방어하므로 적절히 설정할 것을 권장한다.

sysctl -w net.ipv4.icmp_destunreach_rate=1
1/100 초 동안 받아들일 수 있는 "dest unreach (type 3) icmp" 의 개수

sysctl -w net.ipv4.icmp_echo_ignore_broadcasts=1
Broadcast 로부터 오는 ping 을 차단함. (Smurf 공격을 차단함)

sysctl -w net.ipv4.icmp_echoreply_rate=1
1/100 초에 반응하는 ping 의 최대 숫자

sysctl -w net.ipv4.icmp_echo_ignore_all=1
#모든 ping 을 차단함

sysctl -w net.ipv4.icmp_ignore_bogus_error_responses=1
IP 나 TCP 헤더가 깨진 bad icmp packet 을 무시한다.

sysctl -w net.ipv4.icmp_paramprob_rate=1
1/100 초에 받아들이는 param probe packets 의 수

sysctl -w net.ipv4.icmp_timeexceed_rate=1
1/100 초에 받아들이는 timeexceed 패킷의 수(traceroute 와 관련)

sysctl -w net.ipv4.igmp_max_memberships=1
1/100 초에 받아들이는 igmp "memberships" 의 수

sysctl -w net.ipv4.ip_always_defrag=0
항상 패킷 조각 모음을 하지 않는다.

sysctl -w net.ipv4.ip_default_ttl=64
매우 복잡한 사이트에서는 이 값을 늘리는 것도 가능하지만
64 로 두는 것이 적당하며 더 늘렸을 경우에는 큰 문제가 발생할 수도 있다.

sysctl -w net.ipv4.ip_forward=0
게이트웨이 서버가 아닌 이상 패킷을 포워딩 할 필요는 없다.

sysctl -w net.ipv4.ipfrag_time=15
fragmented packet 이 메모리에 존재하는 시간을 15 초로 설정한다.

sysctl -w net.ipv4.tcp_syn_retries=3
일정한 시간과 IP 별로 보내고 받는 SYN 재시도 횟수를 3 회로 제한한다.
이 옵션은 스푸핑된(위조된) 주소로 오는 SYN 연결의 양을 줄여준다.
기본값은 5 이며 255 를 넘지 않아야 한다.

sysctl -w net.ipv4.tcp_retries1=3
무언가 문제가 있을 때 연결을 위해 재시도 할 횟수. 최소값과 기본값은 3 이다.

sysctl -w net.ipv4.tcp_retries2=7
TCP 연결을 끊기 전에 재시도할 횟수.

sysctl -w net.ipv4.conf.eth0.rp_filter=2
sysctl -w net.ipv4.conf.lo.rp_filter=2
susctl -w net.ipv4.conf.default.rp_filter=2

sysctl -w net.ipv4.conf.all.rp_filter=2
이 설정은 자신의 네트워크가 스푸핑된 공격지의 소스로 쓰이는 것을 차단한다.
모든 인터페이스에서 들어오는 패킷에 대해 reply 를 하여 들어오는 인터페이스로 나가지
못하는 패킷을 거부한다.

sysctl -w net.ipv4.conf.eth0.accept_redirects=0
sysctl -w net.ipv4.conf.lo.accept_redirects=0
sysctl -w net.ipv4.conf.default.accept_redirects=0
sysctl -w net.ipv4.conf.all.accept_redirects=0
icmp redirects 를 허용하지 않는다.
만약 ICMP Redirect 를 허용할 경우에는 공격자가 임의의 라우팅 테이블을 변경할 수
있게 되어 자신이 의도하지 않는 경로, 즉 공격자가 의도한 경로로 트래픽이 전달될 수
있는 위험이 있다.

sysctl -w net.ipv4.conf.eth0.accept_source_route=0
sysctl -w net.ipv4.conf.lo.accept_source_route=0
sysctl -w net.ipv4.conf.default.accept_source_route=0
sysctl -w net.ipv4.conf.all.accept_source_route=0
스푸핑을 막기 위해 source route 패킷을 허용하지 않는다.
소스 라우팅을 허용할 경우 악의적인 공격자가 IP 소스 라우팅을 사용해서 목적지의 경로
를 지정할 수도 있고, 원래 위치로 돌아오는 경로도 지정할 수 있다.
이러한 소스 라우팅이 가능한 것을 이용해 공격자가 마치 신뢰받는 호스트나
클라이언트인것 처럼 위장할 수 있는 것이다.

sysctl -w net.ipv4.conf.eth0.bootp_relay=0
sysctl -w net.ipv4.conf.lo.bootp_relay=0
sysctl -w net.ipv4.conf.default.bootp_relay=0
sysctl -w net.ipv4.conf.all.bootp_relay=0
bootp 패킷을 허용하지 않는다.

sysctl -w net.ipv4.conf.eth0.log_martians=1
sysctl -w net.ipv4.conf.lo.log_martians=1
sysctl -w net.ipv4.conf.default.log_martians=1
sysctl -w net.ipv4.conf.all.log_martians=1
스푸핑된 패킷이나 소스라우팅, Redirect 패킷에 대해 로그파일에 정보를 남긴다.

sysctl -w net.ipv4.conf.eth0.secure_redirects=0
sysctl -w net.ipv4.conf.lo.secure_redirects=0
sysctl -w net.ipv4.conf.default.secure_redirects=0
sysctl -w net.ipv4.conf.all.secure_redirects=0
게이트웨이로부터의 redirect 를 허용하지 않음으로써 스푸핑을 막기 위해 설정한다.

sysctl -w net.ipv4.conf.eth0.send_redirects=0
sysctl -w net.ipv4.conf.lo.send_redirects=0
sysctl -w net.ipv4.conf.default.send_redirects=0
sysctl -w net.ipv4.conf.all.send_redirects=0
icmp redirects 를 보내지 않는다.

sysctl -w net.ipv4.conf.eth0.proxy_arp=0
sysctl -w net.ipv4.conf.lo.proxy_arp=0
sysctl -w net.ipv4.conf.default.proxy_arp=0
sysctl -w net.ipv4.conf.all.proxy_arp=0
proxy arp 를 설정하지 않는다. 이 값이 1 로 설정되었을 경우 proxy_arp 가 설정된 인터페
이스에 대해 arp 질의가 들어왔을 때 모든 인터페이스가 반응하게 된다.

sysctl -w net.ipv4.tcp_keepalive_time=30
이미 프로세스가 종료되어 불필요하게 남아 있는 연결을 끊는 시간을 줄이도록 한다.

sysctl -w net.ipv4.tcp_fin_timeout=30
연결을 종료시 소요되는 시간을 줄여준다. (기본 설정값 : 60)

sysctl -w net.ipv4.tcp_tw_buckets=720000
동시에 유지 가능한 timewait 소켓의 수이다. 만약 지정된 숫자를 초과하였을 경우에는
timewait 소켓이 없어지며 경고 메시지가 출력된다. 이 제한은 단순한 DoS 공격을 차단하
기 위해 존재하는데, 임의로 이 값을 줄여서는 안 되며 메모리가 충분하다면 적절하게 늘
려주는 것이 좋은데, 64M 마다 180000 으로 설정하면 된다. 따라서 256M 일 경우에는
256/4=4 4*180000=720000 을 적용하면 된다.

sysctl -w net.ipv4.tcp_keepalive_probes=2
sysctl -w net.ipv4.tcp_max_ka_probes=100
간단한 DoS 공격을 막아준다.

위의 모든 설정은 재부팅 후에 원래의 값으로 다시 초기화되므로 /etc/rc.d/rc.local 에 두어 부팅시마다 실행하도록 하여야 한다. 그리고 리눅스의 버전이 낮아 sysctl 명령어가 없는 경우에는

echo 0 or 1 > /proc/sys/net/* 와 같이 직접 /proc 이하의 값을 직접 설정해 주어도 된다.

echo 명령어 역시 재부팅되면 초기화되므로 /etc/rc.d/rc.local 에 설정해 두어야 재부팅후에도 적용이 된다.

아울러 레드햇 6.2 이상일 경우에는 /etc/sysctl.conf 파일에 **net.ipv4.tcp_syncookies=1** 와 같이 설정한 후 network 를 restart 하는 방법도 있다.

4. 그외 SYN_Flooding 에 대한 보충 설명 몇 가지

(1) 위에서 설명한 방법 외에 추가적으로 설정할 만한 몇 가지 방법이 있다.

RFC 1918 에 의해 내부(Private) IP 를 소스로 들어오는 트래픽을 차단한다.

127.0.0.0, 10.0.0.0, 172.16.0.0, 192.168.0.0 등은 Private IP 로서 내부의 가상 IP 를 사용할 때 쓰이는 주소이며 일반적으로 이러한 IP 를 소스 주소로 라우팅이 될 수 없다.

따라서 아래와 같이 비정상적인 IP 주소를 소스로 해서 들어오는 트래픽을 차단한다.

iptables -A INPUT -s 10.0.0./8 -j DROP

iptables -A INPUT -s 172.16.0.0/12 -j DROP

iptables -A INPUT -s 192.168.0.0/16 -j DROP

사설 IP 를 차단한다.

/8, /16 등은 CIDR 라 하며 /8 은 A Class, /16 은 B Class 를 뜻한다.

iptables -A INPUT -s 255.255.255.255/32 -j DROP

iptables -A INPUT -s 127.0.0.0/8 -j DROP

일반적으로 라우팅이 되지 않는 IP 대역을 차단한다.

iptables -A INPUT -s 240.0.0.0/5 -j DROP

IANA 에 예약된 주소를 차단한다.

iptables -A INPUT -s 211.2.3.4 -j DROP

아울러 자기 자신의 IP 를 소스로 하는 패킷도 필터링한다.(211.2.3.4 대신 자신의 IP 입력)

자신의 IP 를 소스로 해서 패킷이 들어올 수는 없다.

자신의 시스템이 Kernel 2.4 이전 버전의 경우에는 **iptables** 대신 **ipchains** 를 사용하므로 **ipchains -A input -s 10.0.0./8 -j DENY** 와 같은 방법으로 사용하면 된다.
만약 iptables 가 설치되어 있지 않으면 http://netfilter.kernelnotes.org/ 에 접속 후 최신 버전의 iptables.tar 를 다운로드 받아 압축해제 후 make; make install 로 설치하면 된다.
현재 리눅스 시스템의 Kernel 버전은 **uname - r** 을 입력하면 확인할 수 있다.
아울러 아래는 네트워크를 통해 라우팅 될 수 없는 IP 대역이므로 필터링 하여야 할 IP 이다.

0.0.0.0/8 - Historical Broadcast
10.0.0.0/8 - RFC 1918 에 의한 내부 네트워크
127.0.0.0/8 - Loopback
169.254.0.0/16 - Link Local Networks
172.16.0.0/12 - RFC 1918 에 의한 내부 네트워크
192.0.2.0/24 - TEST-NET
192.168.0.0/16 - RFC 1918 에 의한 내부 네트워크
224.0.0.0/4 - Multicast D Class
240.0.0.0/5 - 예약된 E Class
248.0.0.0/5 - 미할당
255.255.255.255/32 - 브로드캐스트

(2) 임의의 IP 가 아닌 특정한 IP 를 소스 주소로 계속적으로 SYN 공격이 이루어 질 경우에는 해당 IP 를 차단하는 것도 좋은 방법이다.
만약 211.2.3.4 에서 지속적으로 공격이 들어올 때는 아래와 같이 차단할 수 있다.

iptables -A INPUT -s 211.2.3.4 -j DROP (Kernel 2.4.x 버전)
ipchains -A input -s 211.2.3.4 -j DENY (Kernel 2.4 이전 버전)

또는
route add -host 211.2.3.4 reject 로 한다.
만약 211.2.3.X 대역 전체를 차단하려면 **211.2.3.0/24** 와 같이 하면 된다.
(/24 는 C Class 를 뜻한다.)
그러나 위와 같이 route 보다는 iptables 나 ipchains 로 차단하는 것이 더 효과적이다.

만약 임의의 IP 로 공격지를 생성한다면 SYN_RECEIVED 로 보이는 IP 중에는 실제 네트워크에 연결되어 있는 IP 도 있을 것이고 그렇지 않은 IP 도 있을 것이다. 그러나 실제 공격을

당할 때 공격지 IP 를 검출해 보면 모두 ping 이 되지 않는 실제 네트워크에 연결되지 않은 IP 주소이다. 어째서 이런 현상이 일어날까? 이는 앞에서 설명한 TCP 의 3 Way-Handshake 원리를 잘 생각해보면 이해가 될 것이다.

즉, 무작위로 생성된 IP 를 소스로 한 SYN 패킷을 받은 서버는, 요청을 받은 모든 IP 로 SYN+ACK 패킷을 보낸다. 그런데, 정작 실제로 해당 IP 를 사용중인 호스트는 SYN 패킷을 보내지도 않았는데, 공격을 받은 서버로부터 영문도 모르는 SYN+ACK 를 받았으므로 이 패킷을 비정상적인 패킷으로 간주하고 해당 패킷을 리셋(RST)하여 초기화 시킨다.

그리고 실제 존재하지 않는 IP 에 대해서 알아보자. 공격을 당한 서버가 해당 IP 로부터 SYN 패킷을 받았다고 판단(실제로는 위조된 패킷이지만) 하여 SYN+ACK 패킷을 발송 후 ACK 패킷을 계속 기다리지만 해당 IP 는 인터넷에 연결되어 있지 않으므로 SYN+ACK 패킷을 받을 수도 없을 뿐더러 이에 대한 응답으로 ACK 패킷을 발송하지 않을 것임은 불을 보듯 뻔한 것이고, 결국 공격을 받는 서버는 존재하지도 않는 IP 로부터 ACK 패킷을 받을 것만을 기다리며 백로그큐는 가득 차게 되는 것이다. 이것이 백로그큐가 가득 차게 되는 이유이며 백로그큐를 가득 채우는 IP 가 모두 실제로는 존재하지 않는 IP 들인 것이다. 따라서 공격자의 입장에서는 인터넷상에서 라우팅이 되지 않는 IP 를 소스 IP 로 하여 공격하는 것이 가장 효과적일 것이다. 즉 인터넷에 연결되어 있는 IP 를 소스 주소로 하여 SYN Flooding 공격하는 것은 의미가 없다.

(3) 실제 공격지 IP 를 추적하는 것은 거의 불가능하다.

대부분의 DoS 공격이 그러하듯이 SYN_Flooding 공격도 소스 IP 를 속여서 들어오기 때문에 netstat 으로 보이는 IP 를 실제 공격지 IP 라고 판단해서 해당 IP 로 역공격을 해서는 안 된다. 공격을 당하는 리눅스 서버에서 공격지를 아는 방법은 없으며 상위 라우터와 해당 라우터가 연결되어 있는 ISP 업체와 긴밀하게 협조가 되었을 때라야 그나마 추척이 가능하다. 그러나 사실상 협조가 이루어져도 추척하기란 매우 어려운데, 만약 라우팅 경로가 20 개이상 되는 곳에서 공격한다면 20 개 라우터를 관리하는 모든 관리자와 동시에 협조가 이루어져야 하고 공격이 실제 이루어지고 있는 당시에 추척이 되어야 하므로 매우 어렵다고 할 수 있다. 결론적으로 공격지 IP 를 추척하는 것은 불가능하다고 할 수 있다.

그리고, 참고적으로 시스템에서 위조된 패킷을 생성하는 것은 오직 root 만이 가능하므로 공격자는 공격지 시스템의 root 소유로 SYN Flooding 공격을 하는 것이라는 사실을 참고하기 바란다.

(4) Virtul-Sever 커널 패치를 하는 방법도 있다.

이 커널 패치를 하였을 경우에는 몇 가지 DoS 공격을 차단할 수 있다. VirtualServer 란 말 그대로 로드 밸랜싱등의 클러스터링 시스템을 구성할 때 필요한 커널 패치로서 패치를 한 후

sysctl -a|grep .vs. 로 확인해 보면 몇 가지 설정이 추가된 것을 확인할 수 있다.
이 방법에 대한 보다 자세한 안내는 http://www.linuxvirtualserver.org/defense.html 를 참고하기 바란다.

(5) 라우터나 방화벽에서 차단 가능하다.

라우터등 네트워크 장비로 유명한 CISCO 에서는 TCP SYN_Flooding 공격을 차단하기 위해 **TCP Intercept** 라는 솔루션을 제안했다. TCP Intercept 는 두 가지 방식으로 구현가능한데 , 첫번째 방식은 "인터셉트 모드" 라 하여 말 그대로 라우터로 들어오는 SYN 패킷 요청을 그대로 서버에 넘겨주지 않고 라우터에서 일단 가로채어(Intercept 하여) 서버를 대신하여 SYN 패킷을 요청한 클라이언트와 연결을 맺고, 연결이 정상적으로 이루어지면 이번에는 클라이언트를 대신하여 서버와 연결을 맺은 다음 두 연결을 투명하게 포워딩하여 연결시켜주는 방식이다. 따라서 존재하지 않는 IP 로부터 오는 SYN 요청은 서버에 도달하지 못하게 되는 것이다. 두번째 방식은 "와치(watch) 모드" 라 하여 "인터셉트 모드"와는 달리 라우터를 통과하는 SYN 패킷을 그대로 통과시키고 일정 시간동안 연결이 이루어지지 않으면 라우터가 중간에서 SYN 패킷을 차단하는 방식이다. 몇몇 방화벽에서도 위의 두 가지 방식으로 SYN Flooding 을 차단하고 있다. 실제로 tcp intercept 를 설정하여 테스트 결과 서버 레벨에는 전혀 스푸핑된 SYN 패킷이 보내지지 않아 SYN_Flooding 공격을 차단하기 위한 가장 확실한 방법이기는 했지만 아쉽게도 라우터의 CPU, Memory 부하가 너무 높아지는 단점이 있었다. 이 설정에 대해 궁금하신 분은 http://www.cisco.com/ 접속후 **"tcp intercept"** 로 검색해 보기 바란다. 이 설정을 했을 경우에는 모든 패킷에 대해 인터셉트를 하므로 트래픽이 많을 경우에는 라우터가 다운되는 경우도 있으니 설정시 각별히 주의하기 바란다.

(6) Windows NT/2000 계열에서는 Registry 값을 수정함으로써 튜닝이 가능하다.

이 값에 대한 튜닝은 Microsoft 의 technical page 나
http://packetstorm.securify.com/groups/rhino9/synflood.doc 를 다운로드 받아 참고하기 바란다.
AIX 나 Solaris 등 다른 UNIX 계열에 대한 튜닝은
http://www.cymru.com/~robt/Docs/Articles/ip-stack-tuning.html 를 참고하기 바란다.

(7) CRON 을 이용해 SYN_Flooding 공격을 감지한다.

아무리 튜닝을 잘 했다 하더라도 집중적으로 SYN Flooding 공격을 받을 때는 네트워크나 서비스 데몬이 이상 작동할 수도 있다. 그래서 이상 현상이 나타나기 전에 일정 시간마다 시스템에 로그인하여 **netstat** 으로 확인할 수 있겠지만 언제 공격이 들어올 줄 알고 지켜보고 있겠는가? 그래서 필자는 SYN Flooding 을 감지하기 위해 다음과 같이 간단한 스크립트를 짜서 공격이 확인되면 메일로 통보되도록 하여 사용중이다.

```
#!/usr/bin/perl

$TASK = `netstat -na|grep SYN_RECV`;
$HOSTNAME = `/bin/hostname`;
$TO_MAIL          = 'antihong@tt.co.kr';
$SUBJECT          = "$HOSTNAME SYN_FLOODING 공격 감지";
$MAIL_PROGRAM    = "/usr/sbin/sendmail";

if ($TASK){
   $TASK_CONFIRM = `netstat -na|grep SYN_RECV|wc -l`;

  if($TASK_CONFIRM > 20){
`/etc/rc.d/init.d/httpd stop`;
`/etc/rc.d/init.d/httpd start`;
  $HTTP_DONE ="httpd was Refreshed!!\n";
  }

open(MAIL, "|$MAIL_PROGRAM -t");
      print MAIL "To: $TO_MAIL \n";
      print MAIL "Subject: $SUBJECT \n\n";
      print MAIL "$HOSTNAME Server is Attacked by SYN_Flooding!!!\n";
      print MAIL "SYN_Flooding Process Number :$TASK_CONFIRM \n";
      print MAIL "$HTTP_DONE\n";
      print MAIL "$TASK \n";
close(MAIL);
  }
```

위 파일의 내용중 **$TO_MAIL** 은 공격 감지시 통보될 메일 주소이므로 자신의 e-mail 주소로 변경하고, 불완전한 SYN 패킷이 20 개 이상일 경우 **($TASK_CONFIRM > 20)** **`/etc/rc.d/init.d/httpd stop`;** 과 **`/etc/rc.d/init.d/httpd start`;** 으로 웹데몬을 멈추었다가 시작하도록 설정하였는데, 이는 자신의 설정에 맞게 적절히 수정하도록 한다.
물론 SYN Flooding 공격이 특정 포트에 대해서만 가능한 것은 아니지만 거의 80 번 포트에 대해 집중적으로 이루어지고 있으므로 웹데몬을 예로 설정한 것 뿐이다.

위 파일의 내용을 /etc/cron.5min/ 이라는 디렉토리에 두고 실행할 수 있도록 700 으로 설정해 둔다. 그리고 /etc/crontab 파일을 열어 아래 내용을 추가하면 5 분마다 SYN_Flooding 여부를 체크하여 공격이 확인시 지정된 메일 주소로 통보해 준다..

59/5 * * * * root run-parts /etc/cron.5min/