
DDoS 공격대응 가이드

2012. 10.



※ 본 보고서의 전부나 일부를 인용 시, 반드시 [자료:한국인터넷진흥원(KISA)]를 명시하여 주시기 바랍니다.

< 목 차 >

1. DDoS 공격대응 개요	4
2. DDoS 공격대응 절차	5
(1단계) 공격의 인지	5
(2단계) 공격유형 파악	7
(3단계) 차단정책 정의 및 대응	13
(4단계) 공격대응 및 사후조치	16
(추가사항) DNS 공격대응 방안	17
[별첨1] DDoS 공격대응 매뉴얼	19
[별첨2] DDoS 공격차단을 위한 장비 및 현황	21
[별첨3] DDoS 공격유형 분류 및 설명	23
1. DDoS 공격유형 분류	23
2. UDP/ICMP Traffic Flooding 공격	24
3. TCP Traffic Flooding 공격	25
4. IP Flooding 공격	27
5. HTTP Traffic Flooding 공격	28
6. HTTP Header/Option Spoofing Flooding 공격	31
7. 기타 서비스 마비공격	37
[별첨4] 주요 DDoS 공격유형별 대응방안	43
1. UDP/ICMP Traffic Flooding 공격방어	43
2. SYN Flooding 공격방어	44
3. GET Flooding 공격방어	45
4. GET with Cache-Control 공격방어	47
5. HTTP Continuation Data Flooding 공격방어	48
6. TCP Session 공격방어	49
7. URL Redirect 우회 공격방어	50
8. Slow HTTP POST 공격방어	50

9. DNS 공격방어	52
10. Hash DoS 공격방어	53
11. Hulk DoS 공격방어	54
[별첨5] 주요 DDoS 공격도구 분석결과	56
- R.U.D.Y 기반의 Slow POST 공격분석	56
- HTTP DoS Tool 기반의 Slow POST 공격분석	63
- DRDoS (Distributed Reflection DoS) 공격분석	71
- Anonymous WebLoic 공격도구분석	73
- Hash DoS 공격유형 및 도구분석	76
- Hulk DoS 공격유형 및 도구분석	81

1. DDoS 공격 대응 개요

○ DDoS 공격 대응 개요

- DDoS 공격 대응은 공격자와 방어자간의 가용성 확보 싸움임
- 방어자는 자신이 관리하고 있는 웹 서버 및 방어시스템 자원의 한계점을 명확히 알고 있어야 함
 - ※ 자원은 네트워크 대역폭, 웹 서버의 성능 등 물리적인 요소뿐만 아니라 웹서버와 DB가 효율적으로 연동되어 있는가 등에 대한 논리적인 요소들을 포함
- 운영장비의 자원 현황 모니터링 및 끊임없는 차단정책 개선 없이 단순히 장비에만 의존하여 공격을 대응하는 것에는 한계가 존재
 - ※ 공격자는 방어가 이루어질수록 더욱 많은 봇들을 동원하여 다양한 형태의 공격을 수행하기 때문에 장비가 가지는 정적 차단 정책에 의존하는 것은 위험
- 그러므로, 보호하기 위한 시스템과 방어를 위해 사용하는 자원을 항상 모니터링하고 발생하는 DDoS 공격유형에 따른 차단정책을 찾고 적용하는 것이 무엇보다 중요함

○ DDoS 공격 대응 절차 및 목적 ([별첨 1 - 대응매뉴얼] 참조)

- (1단계: 공격 인지를 위한 체크포인트) 웹서비스 관련 이벤트 발생 시 해당 원인이 DDoS 공격으로 인한 것인지에 대한 명확한 판단이 필요
- (2단계: DDoS 공격 유형 파악) DDoS 공격 유형을 명확히 파악하여 차단정책 설정을 위한 근거로 활용
- (3단계: 공격유형에 따른 차단정책 정의 및 대응) 공격의 유형과 목적을 명확히 판단하여 차단정책을 설정함으로써 웹서비스의 가용성 확보
- (4단계: 공격 대응 후, 사후조치) 공격트래픽 분석을 통해 공격 내용을 상세히 규명함으로써 추가 발생할 수 있는 공격 대비를 위해 정책을 업데이트하고 좀비PC IP를 확보

2. DDoS 공격 대응 절차

2.1. (1단계) 공격의 인지 - 공격여부 Check Point

○ 유입 트래픽 크기 (Incoming Traffic Volume)

- 방화벽, IDS 등의 네트워크 장비를 통해 웹서비스 운영 망으로 유입되는 트래픽의 BPS와 PPS 규모를 확인하여 평시와 비교
 - ※ BPS(Bit Per Second)와 PPS(Packet Per Second)는 네트워크 트래픽 규모를 파악하기 위한 기본 단위로 10Mbps = 15,000PPS 임
- 유입 트래픽의 크기가 비정상적인 증감을 나타내는 경우, 공격 발생 여부를 의심할 수 있음
 - ※ 증감기준(임계치)은 특정 값으로 정의할 수 없음. 즉, 방어하고자하는 웹사이트의 하드웨어 성능, 네트워크 대역폭 등을 감안하여 임계치를 정의해야 함

<유입트래픽을 파일로 저장하는 방법>

● Anti-DDoS(예:NXG 10000D)를 운영하는 경우,

- 1) 유입트래픽 확인은 Anti-DDoS장비 GUI/CLI에서하고 CLI gstat으로 Total Inbound/Outbound 트래픽 및 Drop 패킷을 확인 할 수 있으며, pdomain stat2를 통해 보호도메인별 Inbound/Outbound 트래픽 및 Drop 트래픽을 확인
- 2) 유입트래픽을 저장하기위해 해당 Anti-DDoS GUI 장비로 접속 후 모니터링 → 트래픽 현황 → 보호 도메인별 현황에 접속 한다. 보호 도메인별 현황에서는 프로토콜별 트래픽 현황과 수신 차단 트래픽을 확인 하고 xls의 아이콘을 통해 해당 트래픽을 파일로 저장 할 수 있음

CLI명령어	옵션	설명
gstat	없음	실시간 in/out total 트래픽확인
pdomain stat	보호도메인 ID (pdomain stat 1)	해당시간의 보호도메인별 in/out 트래픽 확인
pdomain stat2	보호도메인 ID (pdomain stat2 1)	실시간의 보호도메인별 in/out 트래픽 확인

● 방어장비를 운영하지 않는 경우,

- 1) 트래픽을 저장할 수 있는 네트워크 스위치 또는 장비를 확보하여 패킷을 확보

○ 웹서버 접속 로그 (WebServer Access Log)

- 서버의 접속 로그를 확인하여 비정상 접속 증가여부 확인

※ 일반적인 DDoS 공격은 특정 페이지(예: 메인페이지, 특정 값 요청페이지 등)만을 지속적으로 요청하기 때문에 웹서버 접속로그를 분석하여 비정상 접속 증가 여부를 확인

<웹서버 로그 확보 방법>

구분	Windows 계열(IIS 기준)	Linux 계열(Apache 기준)
로그 파일 경로	1. Default 경로 Windows\System32\LogFiles \inetpub\logs\LogFiles (웹서버 설정에 따라 다름)	1. Default 경로 /usr/local/apache/conf /var/log/httpd (httpd.conf 설정에 따라 다름)
로그 파일명	u_ex110802.log	access_log
파일경로 설정방법	제어판의 관리 도구를 클릭하여 Internet Service Manager 항목을 열어 경로를 변경하고자 하는 웹 서비스의 속성(property)을 클릭하여 log에 대한 경로를 설정 enable logging이 되어 있어야 함 (로그를 남기도록 설정하지 않으면 로그가 남지 않음)	httpd.conf 파일에서 CustomLog로 되어 있는 부분을 수정 CustomLog "logs/access_log" common (httpd.conf 파일의 위치는 Apache 설치시 경로 설정에 따라 다름)

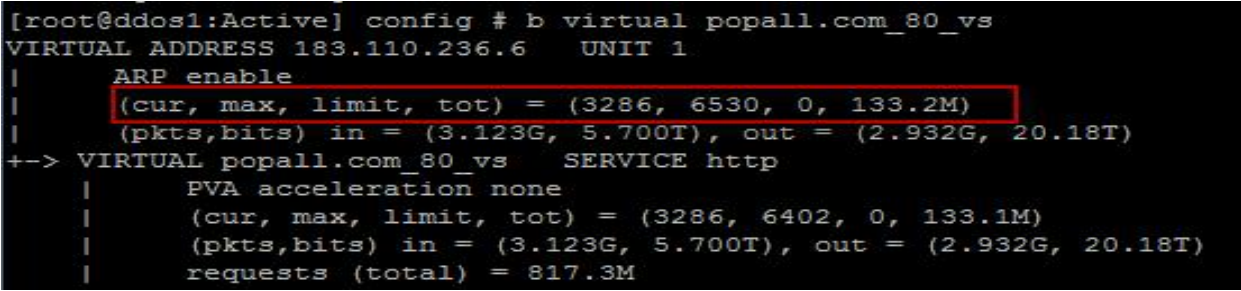
○ 동시접속 정보 (Concurrent Connection)

- 웹서버와 클라이언트가 유지하고 있는 연결(Connection) 규모를 확인하여 평시대비 증감률 비교

※ 웹서버의 연결(Connection) 규모는 웹서버에서 제공하는 기능을 이용하거나 방화벽, IDS, L7 등 네트워크 장비를 통해 확인

<L7 Switch 장비에 접속하여 해당 서비스에 대한 Connection 정보 확인>

```
#b virtual [Virtual name]
```



```
[root@ddos1:Active] config # b virtual popall.com_80_vs
VIRTUAL ADDRESS 183.110.236.6 UNIT 1
|
| ARP enable
| (cur, max, limit, tot) = (3286, 6530, 0, 133.2M)
| (pkts,bits) in = (3.123G, 5.700T), out = (2.932G, 20.18T)
+--> VIRTUAL popall.com_80_vs SERVICE http
| PVA acceleration none
| (cur, max, limit, tot) = (3286, 6402, 0, 133.1M)
| (pkts,bits) in = (3.123G, 5.700T), out = (2.932G, 20.18T)
| requests (total) = 817.3M
```

※ 현재 Connection 정보(cur), 최대치(max), 총 Connection 수(tot)를 확인

○ 유입 트래픽 샘플링 (Incoming Traffic Sampling Capture)

- 웹서버 운영망으로 유입되는 트래픽을 적절히 샘플링하여 실제 트래픽을 분석하여 DDoS 공격 여부를 검증
 - ※ 순간 발생하는 대규모 DDoS 공격 트래픽을 모두 분석하는 데는 한계가 있음
 - ※ DDoS 공격 발생 시 Sampling Capture 만으로도 비정상 여부를 확인할 수 있는 경우가 많이 있음

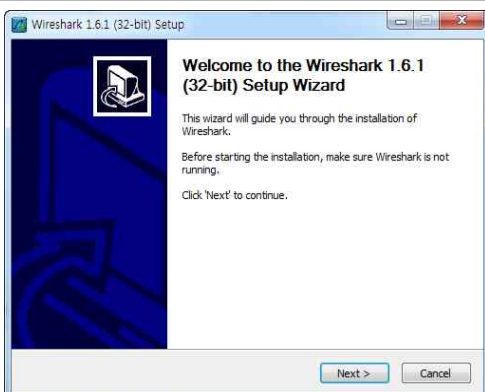
<유입 트래픽 샘플링>
 유입 트래픽 샘플링은 트래픽 양과 접속로그 정보 등을 이용하여 빠른 시간 내에 공격여부, 공격대상 IP, 공격에 사용되는 프로토콜 정보 등 공격 여부를 판단하기 위한 절차로 아래와 같은 방식으로 트래픽을 샘플링
 가) 모든 트래픽이 모여지는 구간을 설정
 나) 샘플링은 될 수 있는 한 짧은 시간(몇 초 단위)으로 설정
 다) Access log 확인 시에는 URL정보와 IP정보 등을 filtering하여 공격여부 판단

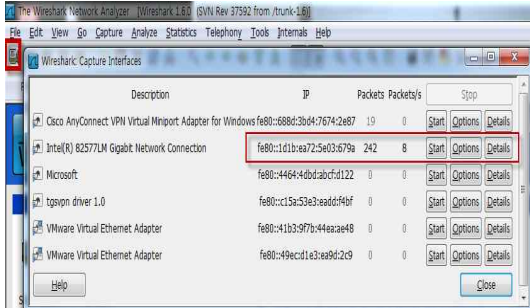
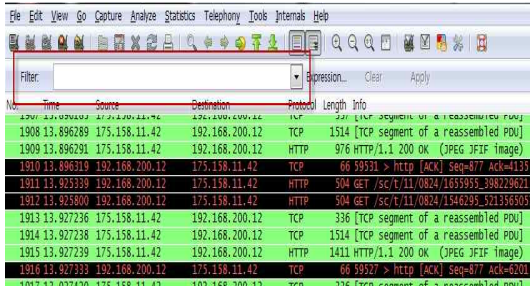
2.2. (2단계) DDoS 공격유형 파악

2.2.1. 유입 트래픽을 이용한 DDoS 공격 유형 파악

○ 패킷 덤프(Packet Dump)를 이용한 유입 트래픽 확보

- tcpdump와 같은 트래픽 캡처 툴을 이용하여 분석하고자 하는 기간 동안의 유입 트래픽 일부를 PCAP 형태로 저장
 - ※ 트래픽 확보는 tcpdump 기능을 하드웨어로 구현한 장비로도 가능
 - ※ PCAP : Packet CAPture의 약자로 네트워크 패킷을 파일로 저장한 것을 의미

<패킷덤프 프로그램 설치 및 사용방법>		
구분	Linux 계열	Windows 계열
Tool 명	TCP Dump	WireShark
설치방법	<ul style="list-style-type: none"> - tcpdump-4.2.0.tar.gz (changelog) (PGP signature) - libpcap-1.2.0.tar.gz (changelog) (PGP signature) <p>1. libcap 설치 <code>gzip -d libpcap-0.7.1.tar.gz</code> <code>tar -xvf libpcap-0.7.1.tar</code> <code>cd libpcap-0.7.1</code> <code>./configure</code> <code>make</code> <code>make install</code></p> <p>2. tcp dump 설치 <code>gzip -d tcpdump-3.9.1-096.tar.gz</code> <code>tar -xvf tcpdump-3.9.1-096.tar</code></p>	

	<pre>cd tcpdump-3.9.1-096 ./configure make make install</pre> <p>Download: http://cvs.tcpdump.org/#latest-release</p>	<p>Download: http://www.wireshark.org/download.html</p>
<p>사용방법 (필터조건)</p>	<pre>[root@R5 Cache 4 ~]# tcpdump -nni eth0 tcpdump: verbose output suppressed, use -v or -vv for full protocol decode listening on eth0, link-type EN10MB (Ethernet), capture size 96 bytes</pre> <p>- Tcpdump의 주요 옵션들</p> <ul style="list-style-type: none"> * -c Number : 제시된 수의 패킷을 받은 후 종료. * -i device : 인터페이스지정. * -n : 모든 주소들을 변환하지 않는다(port, host address 등등) * -nn : protocol 과 port를 이름으로 변환하지 않는다. * -r file : 패킷들을 '-w' 옵션으로 만들어진 파일로 부터 읽어 들인다. * -s length: 패킷들로부터 추출하는 샘플을 default값인 68Byte의 값으로 설정할 때 사용. * -T type : 조건식에 의해 선택된 패킷들을 명시된 형식으로 표시. * -S : TCP sequence번호를 상대적인 번호가 아닌 절대적인 번호로 출력. * -v : 좀 더 많은 정보들을 출력. * -vv : '-v'보다 좀 더 많은 정보들을 출력. * -w : 캡처한 패킷들을 분석해서 출력하는 대신에 그대로 파일에 저장. <p>ex) #tcpdump host 192.168.2.165 192.168.2.165 IP 주소를 가지고 있는 프레임</p> <p>#tcpdump host 192.168.2.165 and port 23 특정 IP 주소와 지정된 포트를 가지고 있는 프레임만 출력.</p> <p>#tcpdump -i eth0 host 192.0.0.1 eth0 인터페이스에서 흐르는 패킷 중 192.0.0.1 IP주소를 가지고 있는 패킷만 출력</p>	<p>1. Tool의 실행(패킷 캡처 시작)</p>  <p>2. Filter 조건</p>  <ul style="list-style-type: none"> * Protocol: 사용 가능한 값: ether, fddi, ip, arp, rarp, decnet, lat, sca, moprc, mopdl, tcp and udp. * Direction: 사용 가능한 값: src, dst, src and dst, src or dst * Logical Operations: 사용 가능한 값: not, and, or. 부정 연산("not") <p>ex) *src host 10.7.2.12 and not dst net 10.200.0.0/16 출발지 IP 주소가 10.7.2.12이면서, 목적지 IP 네트워크가 10.200.0.0/16이 아닌 패킷</p> <p>*ip.src != 10.1.2.3 and ip.dst != 10.4.5.6 출발지 IP 주소가 10.1.2.3이 아니면서, 동시에 목적지 IP 주소가 10.4.5.6이 아닌 패킷</p>

○ 확보된 트래픽 분석(Analysis)

- DDoS 공격 특징을 파악하기 위해서는 프로토콜 정보, HTTP 헤더 정보, 연결 정보를 확인해야 함

분석 도구	설명
tcpdstat	수집된 트래픽의 프로토콜 종류 등에 관한 정보 확인
ngrep, httpry	http header에 관한 정보 확인
argus	concurrent connection에 관한 정보 확인

<분석도구 기능 및 설치 방법>

- 상기 표시된 모든 도구의 필수 패키지

패키지의 의존성 때문에 libpcap 은 필수적으로 설치되어 있어야 함

```
[root@manager_7 ~]# yum search libpcap
Loaded plugins: fastestmirror
Loading mirror speeds from cached hostfile
=====
libpcap.i386 : A system-independent interface for user-level packet capture.
libpcap-devel.i386 : A pcap library.
libnet.i386 : Routines to help with network packet construction and handling
wireshark.i386 : Network traffic analyzer
[root@manager_7 ~]#
```

- tcpdstat

캡처된 pcap 파일의 정보 조회 및 프로토콜별 사용량 확인이 가능하고 파일 내에서의 평균 트래픽과 최대 트래픽 등의 정보를 제공, rpm을 이용하여 설치

```
[root@manager_8 tcpdstat]# rpm -ivh --nodeps tcpdstat-uw-1.0-1.i386.rpm
warning: tcpdstat-uw-1.0-1.i386.rpm: Header V3 DSA signature: NOKEY, key ID 1b5d5459
Preparing...
1:tcpdstat-uw
[root@manager_8 tcpdstat]#
```

- ngrep

실시간 패킷 확인하고 옵션 설정으로 사용자가 필요한 내용만 필터링 가능 Header와 data의 확인 가능, yum을 이용하여 설치

```
[root@R8_Cache_1 ~]# yum search ngrep
Loaded plugins: fastestmirror
Loading mirror speeds from cached hostfile
=====
ngrep.x86_64 : Realtime network grep tool
[root@R8_Cache_1 ~]#
```

```
[root@R8_Cache_1 ~]# yum install ngrep.x86_64
```

- httpry

http protocol(80port)를 사용하는 패킷을 캡처하고 호출되는 콘텐츠의 Method별 확인 가능, source file을 이용하여 설치

```
[root@manager_8 httpry-0.1.6]# make
gcc -Wall -O3 -funroll-loops -I/usr/include/pcap -I/usr/local/include/pcap -o httpry httpry.c format.c methods.c utility.c rate.c -lpcap -lm -lpthread
rate.c: In function ?run_stats? :
rate.c:129: warning: no return statement in function returning non-void
[root@manager_8 httpry-0.1.6]# make install

Installing httpry into /usr/sbin/

You can move the Perl scripts and other tools to
a location of your choosing manually

cp -f httpry /usr/sbin/
cp -f httpry.1 /usr/man/man1/ || cp -f httpry.1 /usr/local/man/man1/
cp: cannot create regular file /usr/man/man1/: No such file or directory
cp: cannot create regular file /usr/local/man/man1/: No such file or directory
make: *** [install] Error 1
[root@manager_8 httpry-0.1.6]#
```

- argus

네트워크 패킷을 모니터링 하는 툴이며 bps, pps, cps, rps 정보 생성

source file 및 rpm 을 이용하여 설치, http://www.qosient.com/argus/index.shtml

<분석도구 사용 시 주요 확인 사항>

- tcpdstat

평균/최대 트래픽, 사용 중인 프로토콜 종류, 프로토콜별 사용량

```
[root@MRTG pcap]# tcpdstat 20110629_final.PCAP
DumpFile: 20110629_final.PCAP
FileSize: 2188.67MB
Id: 201106292023
StartTime: Wed Jun 29 20:23:27 2011
EndTime: Wed Jun 29 20:28:14 2011
TotalTime: 286.44 seconds
TotalCapSize: 2106.59MB CapLen: 1514 bytes
# of packets: 5379020 (2106.59MB)
AvgRate: 62.73Mbps stddev:48.68M PeakRate: 120.14Mbps

### IP flow (unique src/dst pair) Information ###
# of flows: 389364 (avg. 13.81 pkts/flow)
Top 10 big flow size (bytes/total in %):
11.0% 10.9% 10.7% 8.8% 8.3% 8.3% 8.1% 8.1% 0.7% 0.7%

### IP address Information ###
# of IPv4 addresses: 262927
Top 10 bandwidth usage (bytes/total in %):
100.0% 11.7% 11.6% 11.4% 9.5% 9.0% 8.9% 8.7% 8.7% 0.5%

### Packet Size Distribution (including MAC headers) ###
<<<<
[ 32- 63]: 1960227
[ 64- 127]: 2111921
[ 128- 255]: 11028
[ 256- 511]: 14127
[ 512- 1023]: 1236
[ 1024- 2047]: 1280481
>>>>

### Protocol Breakdown ###
<<<<


| protocol      | packets           | bytes                | bytes/pkt |
|---------------|-------------------|----------------------|-----------|
| [*] total     | 5379020 (100.00%) | 2200889224 (100.00%) | 410.66    |
| [*] ip        | 5379020 (100.00%) | 2200889224 (100.00%) | 410.66    |
| [*] tcp       | 4726931 (87.88%)  | 1780885488 (80.62%)  | 376.75    |
| [*] http(s)   | 2075381 (38.58%)  | 1266630731 (57.53%)  | 61.02     |
| [*] http(c)   | 2651550 (49.23%)  | 1654224153 (74.89%)  | 6223.87   |
| [*] udp       | 525637 (9.77%)    | 419215904 (18.98%)   | 797.54    |
| [*] name      | 7 (0.00%)         | 60296 (0.00%)        | 899.43    |
| [*] dns       | 6 (0.00%)         | 60216 (0.00%)        | 1036.00   |
| [*] kerbs     | 7 (0.00%)         | 62296 (0.00%)        | 899.43    |
| [*] sunrpc    | 6 (0.00%)         | 60216 (0.00%)        | 1036.00   |
| [*] rtp       | 6 (0.00%)         | 60216 (0.00%)        | 1036.00   |
| [*] cpmap     | 4 (0.00%)         | 47008 (0.00%)        | 940.40    |
| [*] netb-ns   | 4 (0.00%)         | 46000 (0.00%)        | 1155.50   |
| [*] netb-se   | 4 (0.00%)         | 46296 (0.00%)        | 899.43    |
| [*] ms-ds     | 4 (0.00%)         | 47008 (0.00%)        | 940.40    |
| [*] rtp       | 3 (0.00%)         | 3324 (0.00%)         | 1036.00   |
| [*] kazaa     | 3 (0.00%)         | 6376 (0.00%)         | 797.00    |
| [*] mssql-s   | 3 (0.00%)         | 6376 (0.00%)         | 797.00    |
| [*] realaud   | 3 (0.00%)         | 17456 (0.00%)        | 831.14    |
| [*] hlsif     | 2 (0.00%)         | 14106 (0.00%)        | 940.40    |
| [*] statcrash | 2 (0.00%)         | 16016 (0.00%)        | 1036.00   |
| [*] everque   | 2 (0.00%)         | 23590 (0.00%)        | 907.31    |
| [*] unreal    | 2 (0.00%)         | 6216 (0.00%)         | 1036.00   |
| [*] quake     | 2 (0.00%)         | 18888 (0.00%)        | 899.43    |
| [*] quiceme   | 2 (0.00%)         | 9324 (0.00%)         | 1036.00   |
| [*] other     | 4542 (0.08%)      | 413358014 (18.71%)   | 909.88    |
| [*] icmp      | 126452 (0.23%)    | 8851640 (0.40%)      | 70.00     |
| [*] frag      | 525637 (9.77%)    | 419215904 (18.98%)   | 797.54    |


```

- ngrep

Header 및 데이터 확인(Method, User-agent, Host, Referrer)

```
[root@manager_7 httpry]# ngrep -l 20110629_final.PCAP -tWbyline | more
input: 20110629_final.PCAP
###
T 2011/06/29 20:23:27.936579 116.193.92.132:64328 -> 121.156.108.251:80 [A]
.....
#
T 2011/06/29 20:23:27.936749 116.193.92.132:64328 -> 121.156.108.251:80 [AP]
GET / HTTP/1.1
Accept: image/gif, image/jpeg, image/pipe, application/x-ms-application, application/vnd.ms-xpsdocument,
application/xaml+xml, application/x-ms-xbap, application/x-shockwave-flash, */*.
Accept-Language: ko.
User-Agent: Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.0; Trident/4.0; SLCC1; .NET CLR 2.0.50727; .NE
T CLR 3.5.30729; .NET CLR 3.0.30729).
Accept-Encoding: gzip, deflate.
Host: www.nol-shop.co.cc.
Connection: Keep-Alive.
```

- httpry(SRC IP, DST IP, Method, URL)

```
2011-06-29 20:23:54 222.122.217.193 121.156.108.251 > GET www.nol-shop.co.cc / HTTP/1.1 - -
2011-06-29 20:23:54 222.122.217.193 121.156.108.251 > GET www.nol-shop.co.cc / HTTP/1.1 - -
2011-06-29 20:23:54 222.122.217.193 121.156.108.251 > GET www.nol-shop.co.cc / HTTP/1.1 - -
2011-06-29 20:23:54 222.122.217.193 121.156.108.251 > GET www.nol-shop.co.cc / HTTP/1.1 - -
2011-06-29 20:23:54 222.122.217.193 121.156.108.251 > GET www.nol-shop.co.cc / HTTP/1.1 - -
2011-06-29 20:23:54 222.122.217.193 121.156.108.251 > GET www.nol-shop.co.cc / HTTP/1.1 - -
2011-06-29 20:23:54 222.122.217.193 121.156.108.251 > GET www.nol-shop.co.cc / HTTP/1.1 - -
2011-06-29 20:23:54 222.122.217.193 121.156.108.251 > GET www.nol-shop.co.cc / HTTP/1.1 - -
2011-06-29 20:23:54 222.122.217.193 121.156.108.251 > GET www.nol-shop.co.cc / HTTP/1.1 - -
2011-06-29 20:23:54 222.122.217.193 121.156.108.251 > GET www.nol-shop.co.cc / HTTP/1.1 - -
```

○ 시나리오 기반(Scenario Drawn)의 공격유형 파악

- 대역폭 소진공격, DB 부하 유발공격, 웹서버 자원 공격 등 대표적인 DDoS 공격 유형을 파악

분석 도구	파악 유형
tcpdstat	- 대역폭 소진 공격 유형 분석을 위해 UDP/ICMP Flooding 여부 등 프로토콜별 분포와 트래픽 규모 확인
ngrep, httpry	- Get Flooding 등 DB Connection 부하유발 공격 유형 확인 - 접속자의 요청 페이지(Request Page)에 대한 통계와 특정 시간동안 발생하는 요청 횟수에 대한 통계를 확인
argus	- Syn Flooding 등 웹서버 자원 부하유발 공격 유형 확인 - 특정시간동안 연결된 Connection 규모를 확인

<분석도구별 공격유형을 파악하기 위한 분석 기준>

- tcpdstat

UDP/ICMP의 사용량을 확인하여 대역폭 소진 공격 유형 및 규모 확인

```
[root@MRTG pcap]# tcpdstat 20110629_final.PCAP
DumpFile: 20110629_final.PCAP
FileSize: 2188.67MB
Id: 201106292023
StartTime: Wed Jun 29 20:23:27 2011
EndTime: Wed Jun 29 20:28:14 2011
TotalTime: 286.44 seconds
TotalCapSize: 2106.59MB CapLen: 1514 bytes
# of packets: 5379020 (2106.59MB)
AvgRate: 62.73Mbps stddev:48.68M PeakRate: 120.14Mbps

### IP flow (unique src/dst pair) Information ###
# of flows: 389364 (avg. 13.81 pkts/flow)
Top 10 big flow size (bytes/total in %):
11.0% 10.9% 10.7% 8.8% 8.3% 8.3% 8.1% 8.1% 0.7% 0.7%

### IP address Information ###
# of IPv4 addresses: 262927
Top 10 bandwidth usage (bytes/total in %):
100.0% 11.7% 11.6% 11.4% 9.5% 9.0% 8.9% 8.7% 8.7% 0.5%
### Packet Size Distribution (including MAC headers) ###
<<<<
[ 32- 63]: 1960227
[ 64- 127]: 2111921
[ 128- 255]: 11028
[ 256- 511]: 14127
[ 512- 1023]: 1236
[ 1024- 2047]: 1280481
>>>>

### Protocol Breakdown ###
<<<<
protocol          packets          bytes          bytes/pkt
-----
[0] total          5379020 (100.00%) 2208922428 (100.00%) 410.66
[1] ip              5379020 (100.00%) 2208922428 (100.00%) 410.66
[2] tcp              4726931 ( 87.88%) 1780854884 ( 80.62%) 376.75
[3] http(s)          2075381 ( 38.58%) 126630731 (  5.73%) 61.02
[3] http(c)          2651550 ( 49.29%) 1654224153 ( 74.89%) 623.87
[2] udp              525637 (  9.77%) 419215904 ( 18.98%) 797.54
[3] name              7 (  0.00%) 6296 (  0.00%) 899.43
[3] dns                6 (  0.00%) 6216 (  0.00%) 1036.00
[3] kerb5              7 (  0.00%) 6296 (  0.00%) 899.43
[3] sunrpc             6 (  0.00%) 6216 (  0.00%) 1036.00
[3] ntp                6 (  0.00%) 6216 (  0.00%) 1036.00
[3] epmap              5 (  0.00%) 4702 (  0.00%) 940.40
[3] netb-ns            4 (  0.00%) 4622 (  0.00%) 1155.50
[3] netb-se            7 (  0.00%) 6296 (  0.00%) 899.43
[3] ms-ds              5 (  0.00%) 4702 (  0.00%) 940.40
[3] rip                5 (  0.00%) 4702 (  0.00%) 940.40
[3] kazaa              8 (  0.00%) 6376 (  0.00%) 797.00
[3] mssql-s            8 (  0.00%) 6376 (  0.00%) 797.00
[3] realaud            21 (  0.00%) 17454 (  0.00%) 831.14
[3] halflif            15 (  0.00%) 14106 (  0.00%) 940.40
[3] starcra            6 (  0.00%) 6216 (  0.00%) 1036.00
[3] everque            26 (  0.00%) 23590 (  0.00%) 907.31
[3] unreal            6 (  0.00%) 6216 (  0.00%) 1036.00
[3] quake              21 (  0.00%) 18888 (  0.00%) 899.43
[3] cuseeme            9 (  0.00%) 9324 (  0.00%) 1036.00
[3] other              454298 (  8.45%) 413358214 ( 18.71%) 909.88
[2] icmp              126452 (  2.35%) 8851640 (  0.40%) 70.00
[2] frag              525637 (  9.77%) 419215904 ( 18.98%) 797.54
>>>>
```


- ngrep

Header의 내용 중 특정 문자열을 검색하여 호출 횟수가 많은 URL에 대한 분석

```
[root@manager_7 imsi]# ngrep -l crazy.pcap -tWbyline | grep GET | sort | uniq -c | sort -rn
381 GET /skin/1/images/timer_bg.gif HTTP/1.1.
326 GET /js/jquery-1.3.2.min.js HTTP/1.1.
310 GET /adget/banner.gif HTTP/1.1.
271 GET /adget/adgetBase.js HTTP/1.1.
207 GET /_templates/korean/_imgs/auction/label_10won.png HTTP/1.1.
119 GET /file_57/auctionsid/auction?zix=Wed%20Apr%2027%2023:57:17%202011 HTTP/1.1.
119 GET /file_57/auctionsid/auction?zix=Wed%20Apr%2027%2023:57:16%202011 HTTP/1.1.
118 GET /file_57/auctionsid/auction?zix=Wed%20Apr%2027%2023:57:15%202011 HTTP/1.1.
116 GET /file_57/auctionsid/auction?zix=Wed%20Apr%2027%2023:57:19%202011 HTTP/1.1.
110 GET /file_57/auctionsid/auction?zix=Wed%20Apr%2027%2023:57:18%202011 HTTP/1.1.
103 GET /file_57/auctionsid/auction?zix=Wed%20Apr%2027%2023:57:20%202011 HTTP/1.1.
78 GET /file_57/auctionsid/auction?zix=Wed%20Apr%2027%2023:57:21%202011 HTTP/1.1.
72 GET /_templates/korean/_imgs/label_type03.png HTTP/1.1.
66 GET /js/easySlider1.7.js HTTP/1.1.
66 GET /file_57/auctionsid/auction?zix=Wed%20Apr%2027%2023:57:14%202011 HTTP/1.1.
```

Header의 내용 중 특정 문자열을 검색하여 연결 횟수가 많은 IP에 대한 행위 분석

```
[root@manager_7 pcap]# ngrep -I 20110629_final.PCAP | grep 121.156.108.251 | awk '{print $2}' | sort | uniq -c | sort -rn | more
1905281 121.156.108.251:80
126449 121.156.108.251
230 183.110.246.137:60962
197 222.122.217.193:61634
196 222.122.217.193:61711
196 222.122.217.193:61704
196 222.122.217.193:61638
196 222.122.217.193:61637
```

- httprry

연결 횟수가 많은 IP에 대한 행위 분석

```
[root@manager_7 httprry]# httprry -r 20110629_final.PCAP | awk '{print $4}' | sort | uniq -c | sort -rn
httprry version 0.1.5 -- HTTP logging and information retrieval tool
Copyright (c) 2005-2009 Jason Bittel <jason.bittel@gmail.com>
11634 121.156.108.251
116 183.110.246.137
87 183.110.246.212
87 116.193.92.132
78 183.110.246.205
66 114.111.62.218
64 116.193.90.226

[root@manager_7 httprry]# httprry -r 20110629_final.PCAP | awk '{print $3}' | sort | uniq -c | sort -rn
httprry version 0.1.5 -- HTTP logging and information retrieval tool
Copyright (c) 2005-2009 Jason Bittel <jason.bittel@gmail.com>
1654 61.110.235.201
1484 61.110.235.202
1000 61.110.235.222
1000 61.110.235.221
1000 222.122.217.195
1000 222.122.217.194
1000 222.122.217.193
1000 222.122.217.110
1000 222.122.217.109
1000 222.122.217.108
498 121.156.108.251
116 183.110.246.137
86 183.110.246.212
86 116.193.92.132
78 183.110.246.205
66 114.111.62.218
64 116.193.90.226
```

2.2.2 기타 방법을 이용한 DDoS 공격 유형 파악

○ 웹서버 접속 로그 (WebServer Access Log)

- 서버 접속로그를 확인하여 접속자의 요청 페이지에 대한 통계와 특정 시간동안 발생하는 요청 횟수에 대한 통계를 확인

< 앞의 웹서버 접속 로그 방법 참조 >

2.3. (3단계) 공격유형에 따른 차단정책 정의 및 대응 ([별첨 2] 참조)

○ 대역폭 소진 공격 대응 방안

- 공격 유형 : UDP Flooding, ICMP Flooding
- 대응 방안 : 웹서버 망을 보호하는 방화벽이나 웹서버망 상단에 위치한 라우터에서 해당 프로토콜을 차단하도록 ACL 설정
 - ※ 단, 상단 라우터에서 ACL을 적용하기 위해서는 ISP의 협력이 필요하며, 보호대상 웹사이트에의 UDP, ICMP 서비스 제공 필요여부 확인해야 함

<UDP, ICMP Protocol 전체를 차단하는 ACL 설정(CISCO 장비 기준)>

```
router(config)#access-list 100 deny udp any any
router(config)#access-list 100 deny icmp any any
router(config)#access-list 100 permit ip any any
router(config)#interface g0/0/0
router(config-if)#ip access-group 100 in
```

○ 대역폭 소진 공격 대응 방안

- 공격 유형 : TCP Flooding
- 대응 방안 : 대용량 TCP Flooding 공격은 프로토콜 기준으로 차단하는데 한계가 있어 소스 IP(Source IP)별로 pps 임계치를 설정
 - ※ 임계치 기준은 대상 사이트의 특성을 고려하여 PPS 또는 Connection Count를 활용할 수 있음. 이 경우 확립화된 임계치 기준은 없으므로 사전에 사이트에 대한 특성 고려 필요

<차단방법>

1) L7 Switch(BIG-IP 8900) 장비를 운영하는 경우, 설정방법

- 대용량 TCP 트래픽 발생 시 HTTP Header Size를 설정하여 차단할 수 있으며, 기본적으로 32768로 설정되어 있다. 이 때 공격 트래픽은 Origin Server로 흐르지 않음
- Local Traffic → Profiles → Settings에서 Maximum Header Size 항목에서 사이트 및 공격의 특성에 따라 수치를 적용한다.

Maximum Header Size 32768 bytes

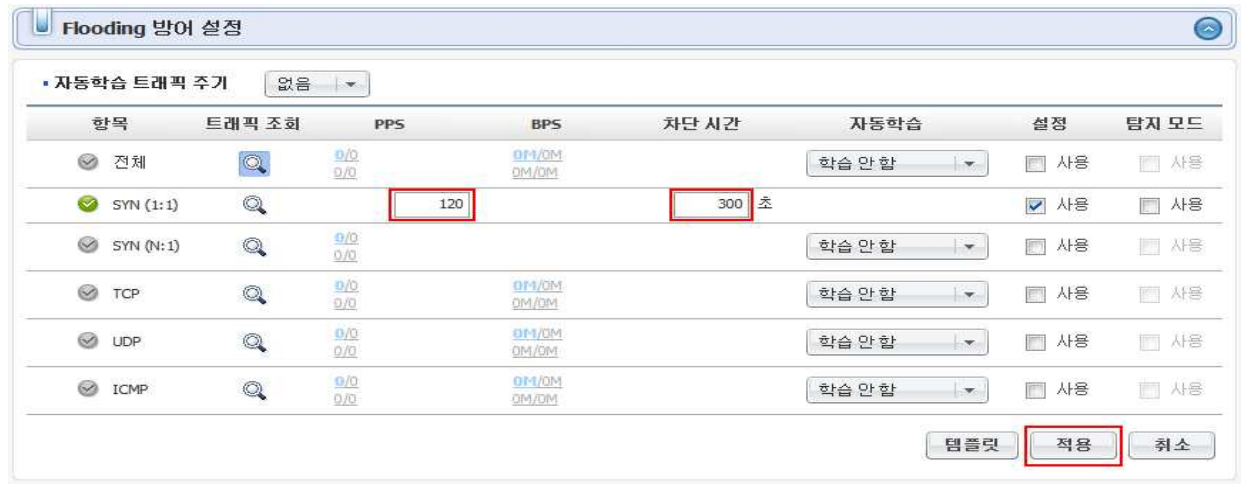
```
HTTP header (33760) exceeded maximum allowed size of 32768 (Client side)
HTTP header (33088) exceeded maximum allowed size of 32768 (Client side)
HTTP header (33332) exceeded maximum allowed size of 32768 (Client side)
HTTP header (33364) exceeded maximum allowed size of 32768 (Client side)
HTTP header (33460) exceeded maximum allowed size of 32768 (Client side)
HTTP header (34080) exceeded maximum allowed size of 32768 (Client side)
```

2) Anti-DDoS(NXG 1000D) 장비를 사용하는 경우, 설정방법

- 메뉴 상단에 보호 도메인을 선택



- pps 임계치를 설정할 대역폭을 선택한 후 Flooding 방어 설정에서 pps 임계치 적용, 사이트 및 공격 특성에 따라 pps 값을 적용



○ 웹서버 자원 소모 공격 대응 방안

- 공격 유형 : Syn(Ack/Fin) Flooding
- 대응 방안 : 웹서버 OS의 TCP 스택(Stack) 자원을 소모하는 특징이 있으므로 ①소스 IP별로 PPS 임계치를 설정하거나 ②패킷 헤더 검사를 통해 정상적인 옵션 필드값을 가지지 않는 비정상 패킷 차단

<차단방법>

Anti-DDoS 장비 설정

- Flooding 방어 설정에서 사이트 및 공격 특성에 따라 pps 값을 적용



○ DB Connection 부하유발 공격 대응 방안

- 공격 유형 : Get Flooding, Post Flooding
- 대응 방안 : 다량의 HTTP 요청으로 웹서버와 DB 연동에 부하를 유발시키는 것이 특징으로 ①클라이언트로부터의 요청 수에 대한 임계치를 설정하여 임계치를 초과하는 소스 IP의 접속 차단하거나 ②HTTP 헤더를 확인하여 HTTP 표준에 맞지 않는 필드 값을 차단 시그너처(Signature)로 설정

<차단방법>

Get Flooding, Post Flooding은 L7 Switch(Big-IP 8900)의 iRule로 차단가능하며, 3초 동안 30회 이상 동일한 URI 요청 시 300초 동안 차단 함, 설정 값은 조절 가능

```

when RULE_INIT {
  array unset ::user
  array set ::user { }
  array set ::blocklist { }
  set ::attacktime 3
  set ::maxquery 30
  set ::holdtime 300
}
when HTTP_REQUEST {
  if { [HTTP::uri] matches_regex {[^jpg|gif|swf|css|png|bmp|js]$} } {
    if { [ info exists ::blocklist([IP::remote_addr]) ] } {
      if { $::holdtime > [ expr [clock seconds] - $::blocklist([IP::remote_addr]) ] } {
        drop
        # log local5. "[IP::remote_addr] is HOLD"
        return
      } else {
        unset ::blocklist([IP::remote_addr])
        # log local5. "[IP::remote_addr] is released"
      }
    }
    if { [info exists ::user([IP::remote_addr],count)] } {
      if { $::attacktime > [expr [clock seconds] - $::user([IP::remote_addr],duration)] } {
        if { $::user([IP::remote_addr],count) > $::maxquery } {
          set ::blocklist([IP::remote_addr]) [clock seconds]
          log local5. "[IP::remote_addr] is blocked"
          drop
          return
        } else {
          incr ::user([IP::remote_addr],count) 1
          return
        }
      } else {
        unset ::user([IP::remote_addr],count)
        unset ::user([IP::remote_addr],duration)
      }
    } else {
      if { 20000 < [array size ::user] } {
        array unset ::user
        array set ::user { }
      }
    }
  }
}

```

```

}
set ::user([IP::remote_addr],count) 1
set ::user([IP::remote_addr],duration) [clock seconds]
}
}
}

```

○ 웹서버 자원 소모 공격 대응 방안

- 공격 유형 : Slow Header Flooding, Slow Data Flooding
- 대응 방안 : 완료되지 않은 연결(Connection) 상태를 지속적으로 유지하는 공격이므로 하나의 요청에 대한 연결 타임아웃을 설정하여 특정 타임아웃이 지나면 연결을 종료시켜 차단

○ 봇 vs 브라우저 식별 대응 방안

- 대응 방안 : 일반적인 봇은 브라우저와 달리 웹서버의 응답코드에 반응하여 행동하지 않으므로 웹서버에서 302 moved temporary와 같은 코드로 응답하여 봇이 발생시키는 요청을 차단

2.4. (4단계) 공격 대응 후, 사후조치

○ 공격 시점의 BPS, PPS, CPS 변화 추이 확인

- 공격 규모를 확인하여 웹서버의 가용성이 침해될 수 있는 지점을 확인하여 정확한 분석정보가 반영된 차단정책 업데이트

○ 공격 유형 확인

- 프로토콜에 대한 통계, 패킷 크기에 대한 통계, 요청 형태에 대한 통계를 상세히 확인하여 시간에 따른 공격 유형의 변경 여부 또는 복합공격 여부를 확인하여 차단 정책 업데이트

○ HTTP 요청 패킷 형태 확인

- ①특정 시간대의 HTTP 요청 횟수(Count)를 확인하여 비정상적인 행위 여부를 규명하고 ②HTTP 헤더의 각 필드 정보를 조사하여, HTTP 표준을 준수하지 않는 비정상 메시지를 차단할 수 있도록 차단정책 업데이트

○ 좀비PC IP 확보

- TCP 기반의 웹서버 가용성 마비 공격은 TCP 3중 연결(3-Way HandShaking) 완료와 함께 시작하므로 실제 공격 IP를 확보하여 차단하도록 조치

※ 웹서버 가용성 마비 공격에는 GET(POST) Flooding, Slow header(data) Flooding이 있음
※ 대역폭 소진 공격의 경우, 공격자는 대부분 Source IP를 위조하므로 IP 위변조 여부를 반드시 확인해야 함

2.5. (추가사항) DNS공격 대응 방안

○ DNS 공격방어를 위한 방어시스템 자원 보유

- DNS는 일반적으로 UDP Protocol을 이용하여 요청/응답을 하는 구조로 백본에서 UDP Protocol을 차단하지 못함
- DNS 정보 또는 IP변경 시 일정시간이 소요되기 때문에 즉각적인 IP변경이 어렵고, 또한 UDP Protocol의 특성 상 Source IP를 변경할 수 있어 공격 IP에 대한 구별이 어려움
- DNS 공격은 대역폭 소진 공격의 특징을 가지므로 대역폭 공격에 대한 방어 자원을 충분히 보유하는 것이 중요함

<DNS공격 시 대응 방안>

1) 대역폭 공격 발생

- Switch에서 차단

- . Switch에서 PBR를 통해 최대 UDP Size인 512bytes 이상의 패킷을 차단
- . PBR적용 시 CPU증가로 현실적으로 사용 불가

```
Router(config)# access-list 111 remark "DNS PBR"  
Router(config)# access-list 111 permit udp any host dns.ip.addr eq 53  
Router(config)# route-map dnsddos permit 10  
Router(config-route-map)# match ip address 111  
Router(config-route-map)# match length 512 1500  
Router(config-route-map)# set interface Null 0  
Router(config-if)# ip route-cache policy  
Router(config-if)# ip policy route-map dnsddos
```

- DNS장비에서 차단

- . 공격량이 크지 않을 경우 DNS서버에서 명령어를 통해 차단

```
iptables -A INPUT -p udp --dport 53 -m length --length 512: -j DROP
```

2) Query 증가

- iptable를 이용 rate-limit 설정으로 공격 패킷 차단

* 5초 동안 5번 query 발생시 해당 패킷 차단

```
iptables -A INPUT -p udp --dport 53 -m recent --name ddos2 --set
iptables -A INPUT -p udp --dport 53 -m recent --name ddos2 --update
--seconds 5 --hitcount 5 -j DROP
```

3) 기타

- 다량의 DNS서버 구축

. 도메인에 대한 NS는 최대 13개까지 등록이 가능하므로 다량의 DNS서버를 구축하여 트래픽 분산 처리

- Anycast 기반의 DNS 구축

. Anycast로 DNS서버를 구축하여 공격 트래픽을 분산, 공격 차단

[별첨 1] DDoS 공격대응 매뉴얼

(1) 공격의 인지 - 공격여부 Check Point

- **incoming traffic volume**

방화벽, IDS등의 네트워크 장비를 통해 웹서비스 운영 망으로 유입되는 트래픽의 bps와 pps 규모를 확인하여 평시와 비교

- **webserver access log**

웹서버의 접속 로그를 확인하여 비정상 접속 증가여부 확인

- **concurrent connection**

방화벽, IDS등의 네트워크 장비를 통해 웹서버가 연결 유지하고 있는 커넥션 규모를 확인하여 평시와 비교

- **incoming traffic sampling capture**

웹서비스 운영 망으로 유입되는 트래픽의 일부를 수집하여 분석

(2) DDoS 공격 유형 파악

[1] incoming traffic을 수집할 수 있는 경우

- **packet dump**

tcpdump와 같은 트래픽 캡처 툴을 이용하여 incoming traffic 일부를 pcap 형태로 저장

- **analysis**

tcpdstat : 수집된 트래픽의 프로토콜 종류등에 관한 정보 확인

ngrep, httprry : http header에 관한 정보 확인

argus : concurrent connection에 관한 정보 확인

- **scenarios drawn**

tcpdstat, tcpdump : 대역폭 소진공격(ex.UDP,ICMP flooding) 여부 판단

ngrep, httprry : DB connection 부하유발 공격(ex. Get flooding)여부 판단

argus : 웹서버 자원(TCP Stack) 부하유발 공격(ex. syn flooding) 여부 판단

[2] incoming traffic을 수집할 수 없는 경우

- **webserver access log**

서버 접속로그를 확인하여 접속자의 request page에 대한 통계와 특정 시간동안 발생하는 request 횟수에 대한 통계를 확인

(3) 공격유형에 따른 공격방안 정의 및 대응

- 대역폭 소진 공격 - ex. UDP, ICMP Flooding**
 웹서버 앞단에 위치한 방화벽이나 상단 라우터(ISP의 협력 필요)에서 해당 프로토콜을 모두 차단하도록 ACL 설정하여 대응
- 대역폭 소진 공격 - ex. TCP Flooding**
 size가 큰 TCP Flooding 공격은 프로토콜 기준으로 차단할 수 없으므로 source ip 별 pps에 대한 임계치 정책을 설정하여 대응
- 웹서버 자원 소모 공격 - ex. Syn(Ack/Fin) Flooding**
 syn flooding 공격은 웹서버 OS의 TCP stack 자원을 소모하는 공격으로서 source ip 별 pps에 대한 임계치 정책을 설정하여 대응하거나 패킷의 헤더를 검사하여 옵션필드가 없는 등의 비정상 패킷을 차단하여 대응
- DB connection 부하유발 공격 - ex. Get(Post) Flooding**
 다수의 HTTP 요청을 유발하여 웹서버와 DB 사이의 연동에 부하를 유발하므로 특정 시간 동안 발생하는 요청 수에 대한 임계치를 설정하여 해당 임계치 이상으로 요청을 발생하는 source ip를 차단
 또한, HTTP header를 확인하여 HTTP 표준에 맞지 않는 field가 설정되었을 경우 해당 field 값을 signature로 설정하여 차단
- 웹서버 자원 소모 공격 - ex. Slow header or Slow data Flooding**
 이 공격은 요청을 완료하지 않고 해당 connection을 지속적으로 유지하는 공격이므로 하나의 요청에 대한 timeout 값을 설정하여 특정 시간동안 요청이 완료되지 않을 경우 connection을 강제 종료시켜서 차단
- 봇 vs 브라우저 식별 방안**
 일반적인 봇은 브라우저와 달리 웹서버의 응답코드에 반응하여 행동하지 않으므로 웹서버에서 302 moved temporary와 같은 코드로 응답하여 봇이 발생시키는 요청을 차단

(4) 공격 대응 후, 사후조치

- 공격 시점의 BPS, PPS, CPS 변화 추이 확인**
 공격 규모를 확인하여 가용성이 침해될 수 있는 지점을 확인하고 정확한 데이터에 따른 차단 정책 업데이트
- 공격 유형 확인**
 프로토콜에 대한 통계, 패킷 크기에 대한 통계, 요청 형태에 대한 통계를 상세히 확인하여 시간에 따른 공격 유형의 변경 여부 또는 복합공격 여부를 확인하여 차단 정책 업데이트
- HTTP 요청 패킷 형태 확인**
 특정 시간에 대한 HTTP 요청 count를 확인하여 비정상적인 행위 여부를 규명하고 HTTP header의 각 field를 조사하여 HTTP 표준 준수 여부를 확인하여 비정상적일 경우 차단 정책 업데이트
- 좀비PC IP 확보**
 GET(POST) Flooding 공격이나 Slow header(data) Flooding 공격의 경우 TCP 삼중 연결 완료와 함께 수행되므로 실제 공격 IP를 확보할 수 있으며 대역폭 소진 공격의 경우 공격자는 대부분 source ip를 위조하므로 IP 위변조 여부를 확인하는 절차가 필요

[별첨 2] DDoS 공격차단을 위한 장비 및 현황

공격 유형	정책 적용 장비	장비별 차단정책 적용 방안
대역폭 소진공격 ex.UDP, ICMP Flooding	Anti-DDoS	<ul style="list-style-type: none"> ●프로토콜 기반으로 차단 설정. 단, 상단 라우터에서 ACL 설정하여 차단
대역폭 소진공격 ex. TCP Flooding	Anti-DDoS, L7 Switch	<ul style="list-style-type: none"> ●비정상적인 헤더 포함여부를 Filtering하여 차단 ●HTTP Continuation 공격 유형은 장비에서 탐지 및 차단하지 못하기 때문에 웹서버의 가용성을 확보(캐싱 장비 이용)하고 공격 트래픽을 전수 혹은 샘플링하여 자세히 분석(운영 장비가 제공하지 않음)하여 공격 IP를 추출하여 차단 ※ HTTP Continuation 공격 : TCP 3-way 연결 후 정상적인 요청 없이 garbage 값을 전송하여 서버의 가용용량을 소진시키는 공격) ※ 효율적인 탐지 및 차단방안을 지속적으로 연구 중이며, 향후에도 고려사항임
웹서버 자원 소모 공격 ex. Syn(Ack/Fin) Flooding	Anti-DDoS	<ul style="list-style-type: none"> ●Source IP 기준 PPS 임계치를 적용하여 차단 ※ 임계치는 웹서버의 성능(서버성능, 네트워크 대역폭 등)을 고려하여 조절하는 값으로 정해진 임계치는 없음
DB connection 부하유발 공격 ex. Get(Post) Flooding	L7 Switch	<ul style="list-style-type: none"> ●Anti-DDoS 장비로는 탐지 및 차단이 불가능하며, L7 Switch에서 Signature를 이용하여 차단 ●Source IP 당 요청가능한 최대 회수를 임계치로 설정하여 차단 ●대부분의 Get(Post) Flooding 공격을 차단하기 위해서는, 트래픽을 상세 분석하여 요청메시지 헤더정보의 비정상 여부를 판단하고 메시지 요청 횟수에 대한 임계치를 유동적으로 조정하여 적용해야 오탐없이 웹서비스의 가용성을 유지할 수 있음

<p>웹서버 자원 소모 공격 ex. Slow header or Slow data Flooding</p>	<p>L7 Switch</p>	<ul style="list-style-type: none"> •Anti-DDoS 장비로는 탐지 및 차단이 불가능 •L7 Switch의 iRule 기능을 이용하여 요청에 대한 차단정책(예: 타임아웃 기능)을 트래픽 분석 데이터를 기초로 도출 및 설정하여 차단 ※ 효율적인 탐지 및 차단방안을 지속적으로 연구 중이며, 향후에도 고려사항임
<p>봇 vs 브라우저 식별</p>	<p>L7 Switch</p>	<ul style="list-style-type: none"> •Anti-DDoS 장비로는 식별이 불가능 •L7 Switch에서 특정 코드로 응답하여 반응하는지 여부를 확인하거나 쿠키에 대해 응답하여 반응하는지 여부를 확인하여 차단 ※ 효율적인 탐지 및 차단방안을 지속적으로 연구 중이며, 향후에도 고려사항임

[별첨 3] DDoS 공격유형 분류 및 설명

1. DDoS 공격유형 분류

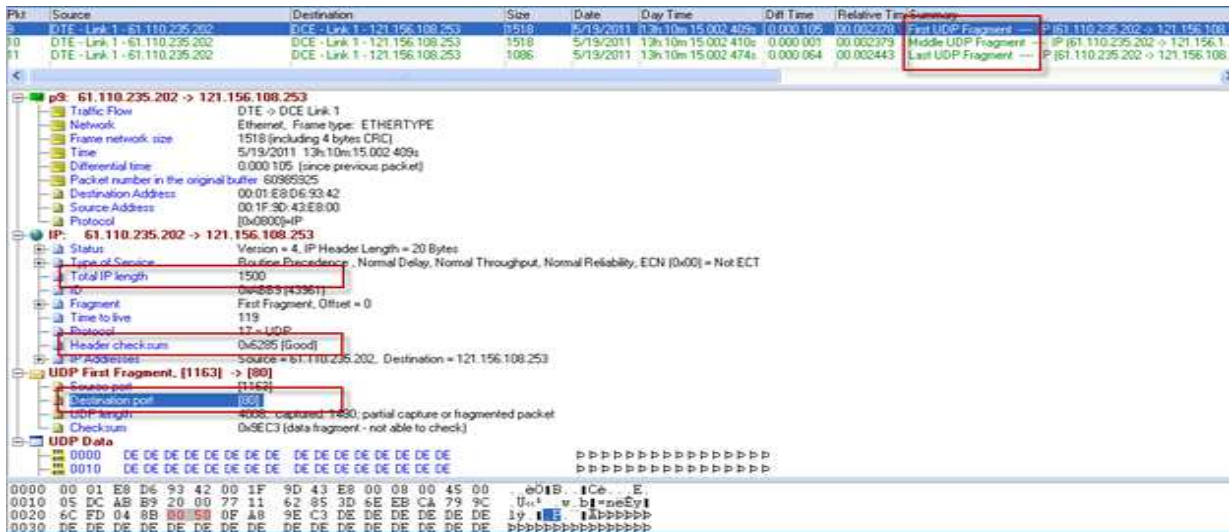
비고	대역폭 소진공격	서비스(어플리케이션) 마비공격
대표 공격유형	UDP/ICMP Flooding, SYN Flooding	HTTP GET Flooding
공격의 형태	<ul style="list-style-type: none"> ○UDP/ICMP Traffic Flooding UDP/ICMP Flooding, DNS Query Flooding 등 ○TCP Traffic Flooding SYN Flooding, SYN+ACK Flooding 등 ○IP Flooding IP Header Option 변조(LAND Attack), IP Fragment Packet Flooding (Teardrop, HTTP Continuation 등) 등 	<ul style="list-style-type: none"> ○HTTP Traffic Flooding GET Flooding, GET with Cache-Control ○HTTP Header/Option Spoofing Slowris, Fragmented HTTP Header Attack(Slowloris/Pyloris) 등 ○TCP Traffic Flooding TCP Session, SYN Flooding, TCP Slow Read 등 ○Other L7 Service Flooding Hash DoS, Hulk DoS, FTP/SMTP Attack 등
프로토콜 (OSI 7-Layer 기준)	3~4계층 (Network, Transport 계층) : IP, ICMP, IGMP, UDP, TCP 등	7계층 (Application 계층) : HTTP, DNS, FTP, SMTP 등
공격대상	네트워크 인프라	웹서버, 정보보호 장비 등
Spoofing 여부	사용/미사용	미사용
증상	<ul style="list-style-type: none"> ○회선 대역폭 고갈 ○동일 네트워크를 사용하는 모든 서비스에 대한 접속장애 발생 	<ul style="list-style-type: none"> ○HTTP 서버과다접속(또는 서비스 부하)으로 인한 장애발생 ○공격대상 시스템만 피해

2. UDP/ICMP Traffic Flooding 공격

2.1. UDP/ICMP Flooding

공격자는 다량의 UDP/ICMP 패킷을 서버로 전송하여 서버가 **보유한 네트워크 대역폭을 가득 채워** 다른 정상적인 클라이언트의 접속을 원활하지 못하도록 유발시키는 공격임
 ※ UDP/ICMP 프로토콜이 비연결지향이라는 특징을 이용하여 소스IP를 변조함

- o Source IP를 변조하거나 실제 IP를 이용하여 UDP/ICMP 패킷을 다량으로 전송하여 네트워크 대역폭을 잠식시켜 DoS 상태를 유발
 - ※ Trinoo라는 DDoS 형태의 UDP Flooding Attack Toolkit이 1996년 미네소타 대학에서 발생했으며 솔라리스 2.x 시스템에서 처음 발견되었고 봇넷 Toolkit(Virut, rBot, IRCbot, Netbot Attacker, 풍운 등)으로 봇넷을 구성하여 대량트래픽을 발생



< UDP 공격 패킷 예 >

2.2. DNS Query Flooding

공격자는 UDP 프로토콜 기반의 서비스를 제공하는 DNS에 대해 **DNS 쿼리 데이터를 다량으로 서버에 전송하여 DNS의 정상적인 서비스를 방해하는 공격임**
 ※ UDP/ICMP Flooding 공격 형태와 유사함

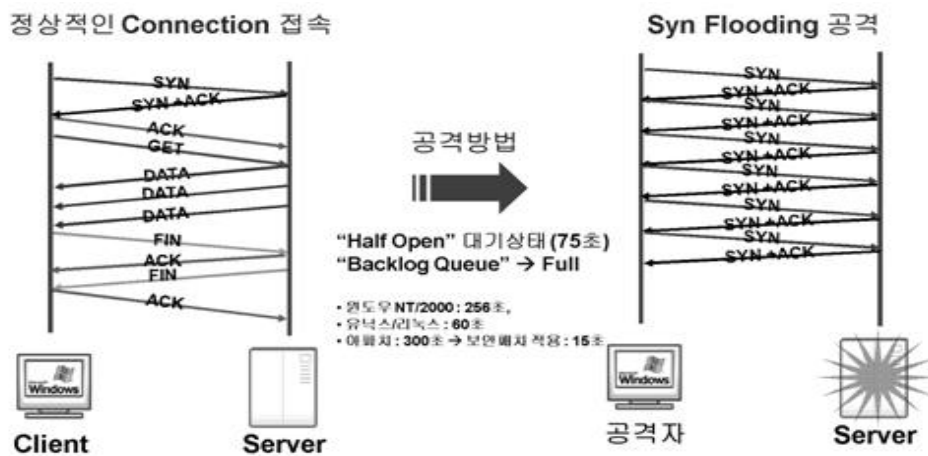
- o DNS는 웹서버에 접속하기 위한 IP를 확인하기 위한 하나의 절차로 웹 서비스를 하기 위한 중요한 요소중 하나임
- o 공격 유형은 크게 1) DNS를 통한 대역폭(Bandwidth) 공격, 2) 많은 Query를 발생하도록 하여 DNS의 응답을 하지 못하도록 하는 서버 자원 공격(어플리케이션 장애유발)으로 볼 수 있음

3. TCP Traffic Flooding 공격

3.1. SYN Flooding

공격자는 다량의 SYN 패킷을 서버로 전달하여 서버의 대기큐(Backlog Queue)를 가득채워 새로운 클라이언트의 연결요청을 무시하도록 하여 장애를 유발시키는 공격
 ※ TCP 프로토콜이 데이터를 보내기 전에 연결을 먼저 맺어야 하는 특징을 이용한 방법임

- TCP 연결과정(3-Way Handshaking)의 처음 단계인 SYN 패킷 전송 단계에서 공격자는 다량의 SYN 패킷을 생성하여 서버로 전달하면,
- TCP 연결요청을 수용할 때 사용하는 서버의 대기큐(Backlog Queue)가 가득 차게(Full) 되어, 이후 들어오는 연결요청을 무시하도록 하는 DoS 상태를 유발함
 - ※ SYN 패킷만 전송하고 웹서버의 SYN-ACK 패킷에 대한 응답으로 ACK 패킷을 전송하지 않으면 Half Open 상태가 되며, 이 정보는 웹서버의 대기큐(Backlog Queue)라는 공간에 쌓여 대기(최대 75초) 하는데 이때 계속하여 SYN 패킷만 전송하여 Half Open 상태를 증가시키면 웹서버는 대기큐(Backlog Queue)가 꽉 차서 더 이상의 TCP 신규 접속을 받지 못하게 됨



< TCP Syn Flooding Attack Flow >

3.2. TCP Flag Flooding

TCP의 Flag 값을 임의로 조작하면 SYN, ACK, FIN, RST과 같이 여러 형태의 패킷을 생성할 수 있으며, 서버는 이러한 패킷을 수신하는 경우 해당 패킷을 검증하기 때문에 서버의 자원을 소진시킴

- 서버는 서버와 연결을 시도하거나 연결되어 있는 클라이언트로부터 수신되는 TCP 패킷이 올바른지 판단하기 위해 검증하는데

o 만약, 다량의 패킷을 서버로 전달하게 되면 서버에 과부하가 발생하여 정상적인 서비스를 제공할 수 없는 DoS 상태에 빠짐

※ ACK Flooding : 공격자가 TCP 세션이 없는 상태에서 TCP 헤더의 Flags를 ACK (0x10)으로 Setting하여 무작위로 보내면 수신측에서 변조된 발신 IP로 RST 패킷을 무작위로 보내게 되고, 동시에 ICMP host Unreachable 패킷을 보내면서 수신측 시스템의 과부하를 초래하는 공격임

※ RST Flooding : 공격대상 서버로 전달되는 클라이언트의 TCP 패킷의 Reset 값을 설정하여 클라이언트가 서버로부터 정상적인 서비스를 받지 못하도록 TCP 연결을 강제로 종료시키는 공격임

3.3. TCP Session

o TCP 3-Way Handshake 과정을 과도하게 유발함으로써 서비스의 과부하를 유발하는 공격 유형으로

o ① TCP 세션 연결을 유지하는 DDoS 공격, ② TCP 세션 연결/해제를 반복하는 DDoS 공격, ③ TCP 세션 연결 후 정상적인 트랜잭션 (Transaction)처럼 보이는 트래픽을 발송하는 DDoS 공격으로 구분



< TCP Session 공격 예 >

4. IP Flooding 공격

4.1. LAND (IP Header Option 변조)

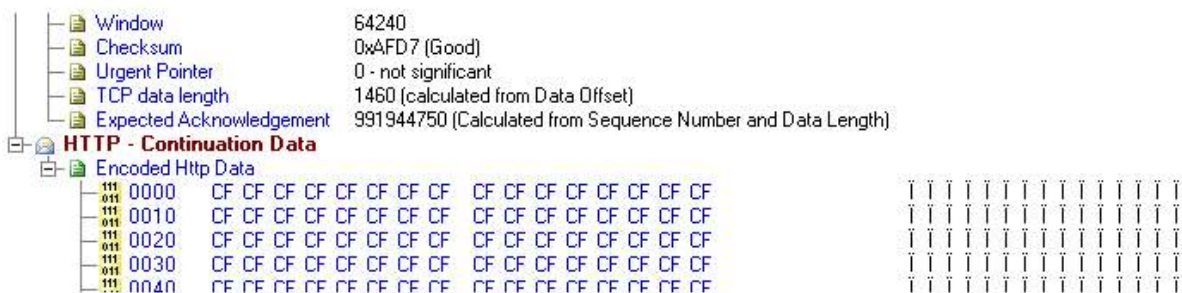
- 인위적으로 송신지 IP 주소 및 Port를 목적지(대상 웹서버) IP 주소 및 Port와 동일하게 설정하여 트래픽을 전송하는 공격
 - ※ 송신지 IP/Port와 목적지 IP/Port가 동일하기 때문에 네트워크 장비의 부하를 유발하기도 함

4.2. Teardrop (IP Fragment Packet Flooding)

- 하나의 IP 패킷은 MTU(Maximum Transmission Unit)라는 이더넷(Ethernet)에서 전송 가능한 IP 데이터그램 크기로 나뉘어 전달되고, 수신자는 나뉘어 전달받은 데이터그램을 하나의 IP 패킷으로 재조합함
- 만약, 공격자가 이러한 IP 데이터그램을 조작하거나 순서를 뒤바꾸어 전송하면 서버는 전달받은 IP 데이터그램의 순서를 알지 못하거나 중복된 IP 데이터그램을 처리하지 못해 시스템 장애가 발생함
- 과거 운영체제 시스템(윈도우 NT, 윈도우 95 등)의 IP 단편화(조각화) 취약점을 이용하였기 때문에 현재 시스템에는 사용할 수 없는 공격 형태임
 - ※ 단편화(fragmentation) : 데이터의 크기가 커서 한 번에 전송할 수 없을 경우 패킷을 나누어 보내는 것을 의미함

4.3. HTTP Continuation (IP Fragment Packet Flooding)

- 서버로 전달하는 패킷에 HTTP Header없이 Data만 채워 웹서버가 지속적으로 데이터 수신을 위해 TCP 자원을 사용하도록 하는 공격
- 패킷 크기를 최대한 크게 하여 보내기 때문에 네트워크 자원도 같이 고갈될 수 있는 공격 형태임



< HTTP Continuation Data Flooding 공격 패킷 예 >

5. HTTP Traffic Flooding 공격

- HTTP(Hypertext Transfer Protocol)는 인터넷에서 브라우저와 웹서버 간에 문서를 전송하기 위해 사용되는 통신규약을 말하며, HTTP 메시지는 Header와 Body로 구성되는데 Header에는 보낼 메시지의 형식을 Body에는 실제 보낼 메시지의 내용을 정의
- 이러한 HTTP 메시지는 TCP를 통해 최소 1개 이상의 패킷으로 분할되어 전송되어짐
 - ※ HTTP 메시지는 TCP의 Payload에 저장되어 전송되어지며, 서버 클라이언트간의 송/수신할 TCP 윈도우 크기(예: 64, 128, 192 등)에 따라 여러 패킷으로 나뉘어 전달
 - ※ TCP 윈도우 : TCP 헤더의 구성요소로써 수신자 측의 수신 가능한 데이터 버퍼용량



< 데이터 캡슐화에 따른 HTTP 메시지 송/수신 패킷 구조 >

- HTTP 메시지는 클라이언트가 요구하는 목적에 맞는 지시자를 이용하여 구성할 수 있는데, 지시자의 종류는 다음과 같음
 - ※ 이 중 일반적인 웹서비스와 관련된 지시자(Indicator)는 GET과 POST이며, DDoS 공격에 가장 많이 사용되는 지시자임

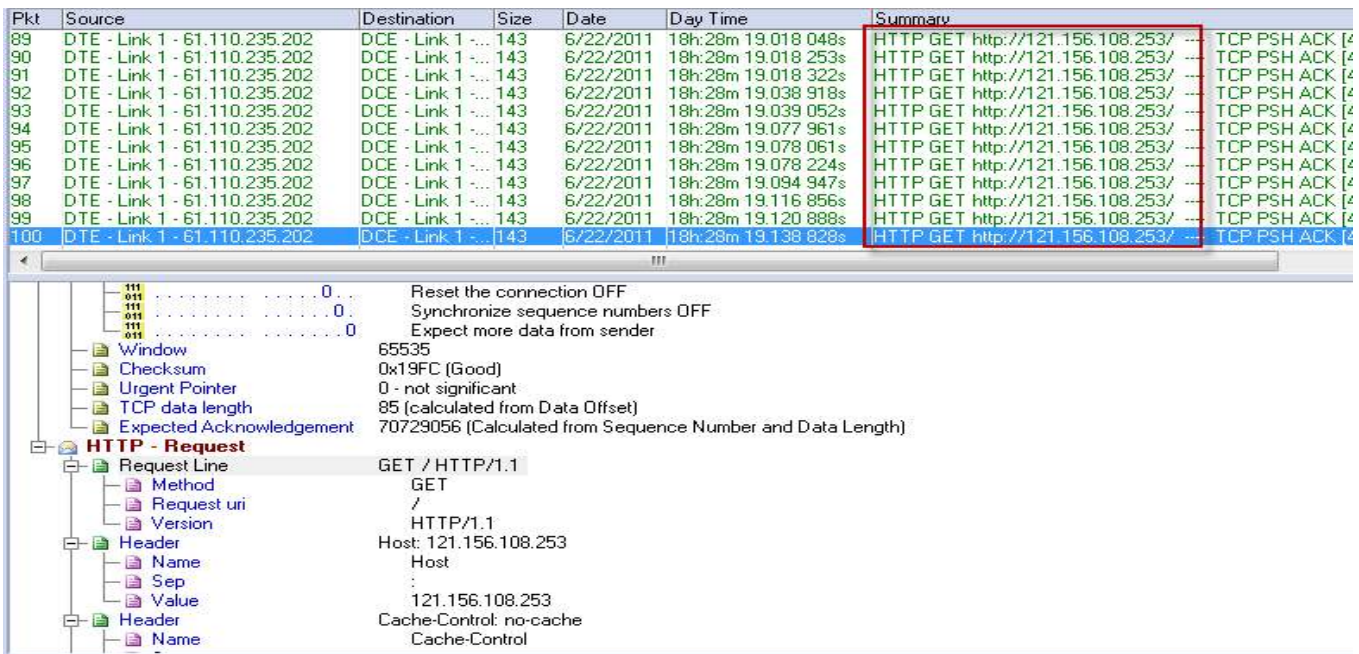
지시자	설명
GET	<ul style="list-style-type: none"> • URL에 해당하는 자료를 제공해 줄 것을 요청 • 웹서버에 저장된 정보를 단순히 요청하기 위해 사용하는 방법으로 클라이언트는 GET 지시자와 함께 URL 정보를 웹서버로 전달하면 웹서버는 해당 정보를 브라우저로 회신하게 됨 ※ 사용 예 : GET HTTP/1.1 op.ddosbunker.com/index.html
POST	<ul style="list-style-type: none"> • 클라이언트에서 웹서버로 데이터를 전송할 때 사용하는 방법으로 웹서버가 처리할 수 있는 자료(예:ID, PWD 등)를 전달할 때 사용 ※ 사용 예 : POST HTTP/1.1 op.ddowbunker.com?login=aaa
HEAD	<ul style="list-style-type: none"> • GET과 같은 요청이지만 자료에 대한 정보(meta-information)만을 수신하는 요청
PUT	<ul style="list-style-type: none"> • 해당 URL에 자료를 저장하는 요청
DELETE	<ul style="list-style-type: none"> • 해당 URL의 자료를 삭제하는 요청
TRACE	<ul style="list-style-type: none"> • 이전에 요청한 내용(히스토리)을 요청
OPTIONS	<ul style="list-style-type: none"> • 서버가 특정 URL에 대해 어떠한 HTTP 지시자를 지원하는지 질의하는 요청
CONNECT	<ul style="list-style-type: none"> • 프록시(Proxy)가 사용하는 요청

< HTTP 메시지에 사용하는 지시자 >

5.1. GET Flooding

공격자는 동일한 URL(예:a.com/index.jsp)을 반복 요청하여 웹서버가 URL에 해당되는 데이터를 클라이언트에게 회신하기 위해 서버 자원을 사용하도록 하는 공격임
 ※ 웹서버는 한정된 HTTP 처리 Connection 용량을 가지기 때문에 용량 초과시 정상적인 서비스가 어려워짐

- 동일한 URL을 반복적으로 요청하는 공격으로 아래 그림과 같이 다량의 GET 요청메시지를 생성하여 서버로 전달하면, 웹서버는 클라이언트의 과도한 응답요구로 인해 DoS 상태에 빠짐

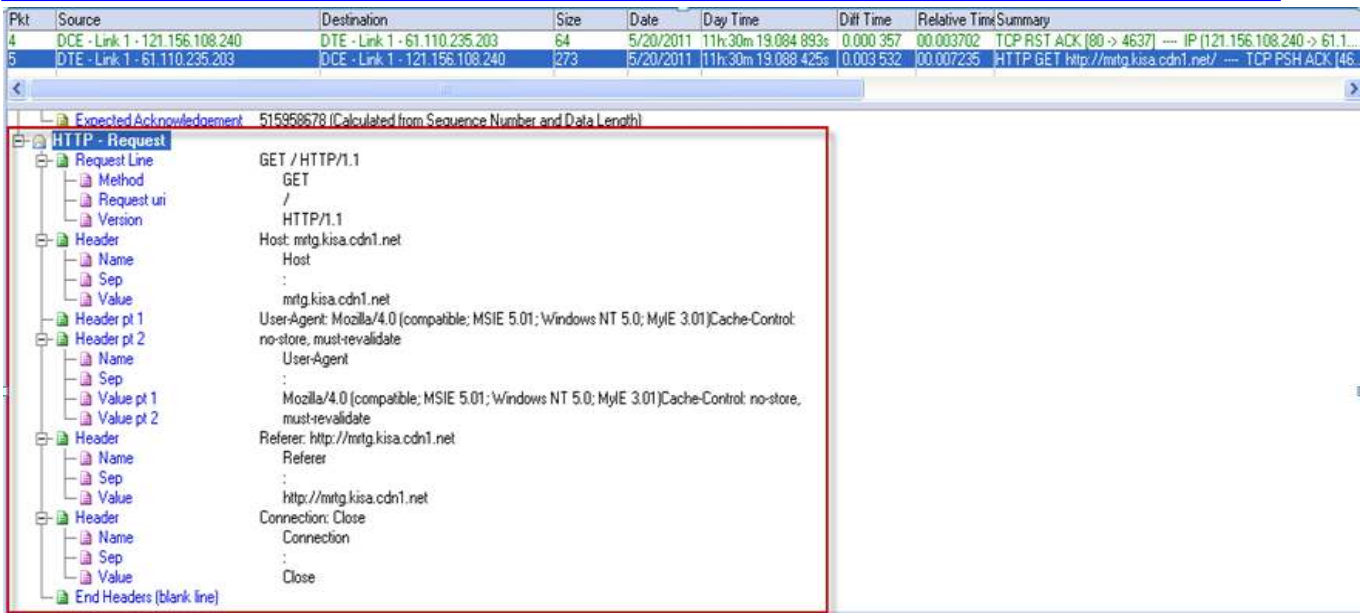


< Get Flooding 공격 패킷 예 >

5.2. GET Flooding with Cache-Control (CC Attack)

- 일반적으로 웹서버의 부하를 감소시키기 위해 캐싱서버를 운영하여 많이 요청받는 데이터 (예:사진파일)는 웹서버가 아닌 캐싱서버를 통해 응답하도록 구축하는 경우,
- 공격자는 HTTP 메시지의 캐시 옵션을 조작하여 캐싱서버가 아닌 웹서버가 직접 처리하도록 유도하여 캐싱서버의 기능을 무력화하고 웹서버의 자원을 소진시키는 공격임

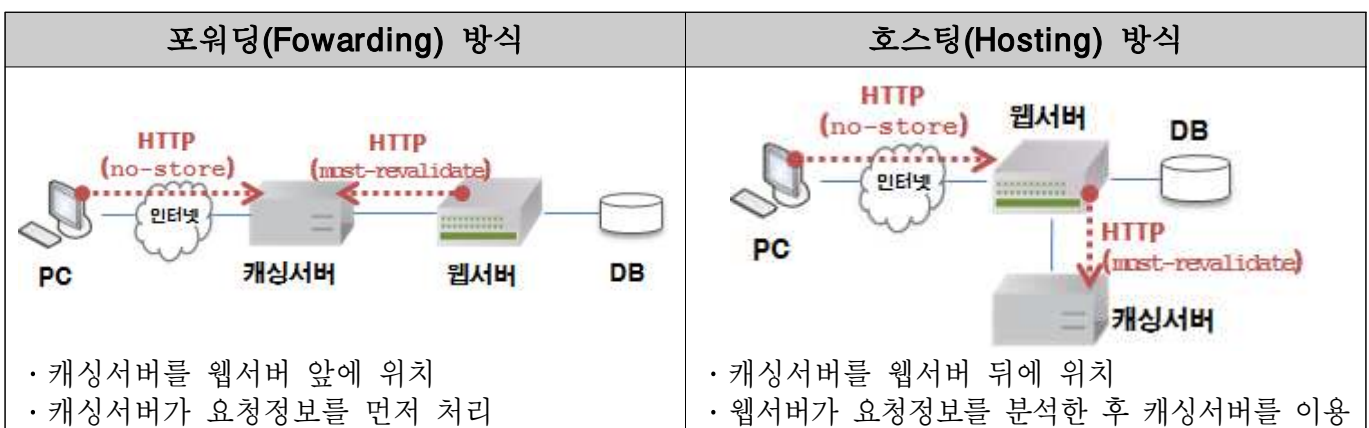
- HTTP 메시지의 헤더정보에 포함된 Cache-Control 값을 no-store, must-revalidate로 지정하여 캐싱 장비가 응답하지 않고 웹서버가 직접 응답하도록 유도하여 웹서버의 자원을 소진시킴
 ※ must-revalidate는 클라이언트가 보낼 경우, 캐싱장비에 영향을 미치지 않음
- Cache-Control 옵션을 제외하고는 Get Flooding과 동일한 형태임



< Cache-Control 공격 패킷 예 >

구분	설명
no-store (캐시저장금지)	- 클라이언트로부터 요청받은 데이터를 디스크나 메모리, 별도의 시스템(캐싱서버)에 저장하는 것을 방지
must-revalidate (캐시검증)	- (별도의 시스템(캐싱서버)를 운영하는 경우) 웹서버는 캐싱서버에 저장된 캐시데이터에 대한 검증을 요구할 수 있는데 ※ 웹서버 내에 캐시를 운영하는 경우에는 must-revalidate를 사용하지 않음 ※ 캐싱서버는 웹서버의 원본데이터와 비교하여 최신 데이터로 업데이트함 ※ 클라이언트가 서버에게 보내는 HTTP 메시지에서 사용되는 정보가 아님

< no-store와 must-revalidate >



< 캐싱(Caching) 서버 운영 모델 >

6. HTTP Header/Option Spoofing Flooding 공격

- o HTTP Header/Option을 이용한 공격은 웹서버의 가용량을 모두 소비시켜 정상적인 웹서비스를 제공하지 못하도록 하는 DoS 상태를 유발하는 공격임
 - ※ HTTP Header/Option을 이용한 공격은 정상적인 TCP 통신을 수행하기 때문에 TCP 전송계층에서의 공격여부를 판단할 수 없음
- o HTTP Header/Option을 이용한 공격에는 크게 다음과 같이 3가지로 구분되어짐

종류	공격 원리
Slow HTTP POST DoS	<ul style="list-style-type: none"> · HTTP POST 지시자를 이용하여 서버로 전달할 대량의 데이터를 장시간에 걸쳐 분할 전송하면 서버는 POST 데이터가 모두 수신하지 않았다고 판단하여 연결을 장시간 유지하게 되는데, · 만약 이러한 데이터를 전달하는 좀비PC가 많은 경우, 서버는 다른 정상적인 클라이언트에 대한 원활한 서비스가 불가능하게 되는 DoS 상태가 유발됨
Slow HTTP Header DoS (Slowloris)	<ul style="list-style-type: none"> · 웹서버는 HTTP 메시지의 헤더부분을 먼저 수신하여 이후 수신할 데이터의 종류를 판단하게 되는데, · 헤더부분을 비정상적으로 조작하여 웹서버가 헤더정보를 구분할 수 없도록 하면, 웹서버는 아직 HTTP 헤더정보가 모두 전달되지 않은 것으로 판단하여 연결을 장시간 유지하게 됨 · 만약 이러한 데이터를 전달하는 좀비 PC가 많은 경우, 서버는 다른 정상적인 클라이언트에 대한 원활한 서비스가 불가능하게 되는 DoS 상태가 유발됨
Slow HTTP Read DoS	<ul style="list-style-type: none"> · 공격자는 웹서버와 TCP 연결 시, TCP 윈도우 크기 및 데이터 처리율을 감소시킨 후 HTTP 데이터를 송신하여 웹서버가 정상적으로 응답하지 못하도록 DoS 상태를 유발 · TCP 윈도우 크기 및 데이터 처리율을 감소시키면 서버는 정상 상태로 회복될때까지 대기상태에 빠지게 되어 다른 정상적인 클라이언트의 접속을 방해함

< HTTP Header/Option 을 이용한 대표적인 공격기법 >

6.1. Slow HTTP POST DoS

공격자는 HTTP POST 지시자를 이용하여 서버로 전달할 대량의 데이터를 장시간에 걸쳐 분할 전송하며, 서버는 POST 데이터가 모두 수신하지 않았다고 판단하여 연결을 장시간 유지하므로 가용량을 소비하게 되어 다른 클라이언트의 정상적인 서비스를 방해함

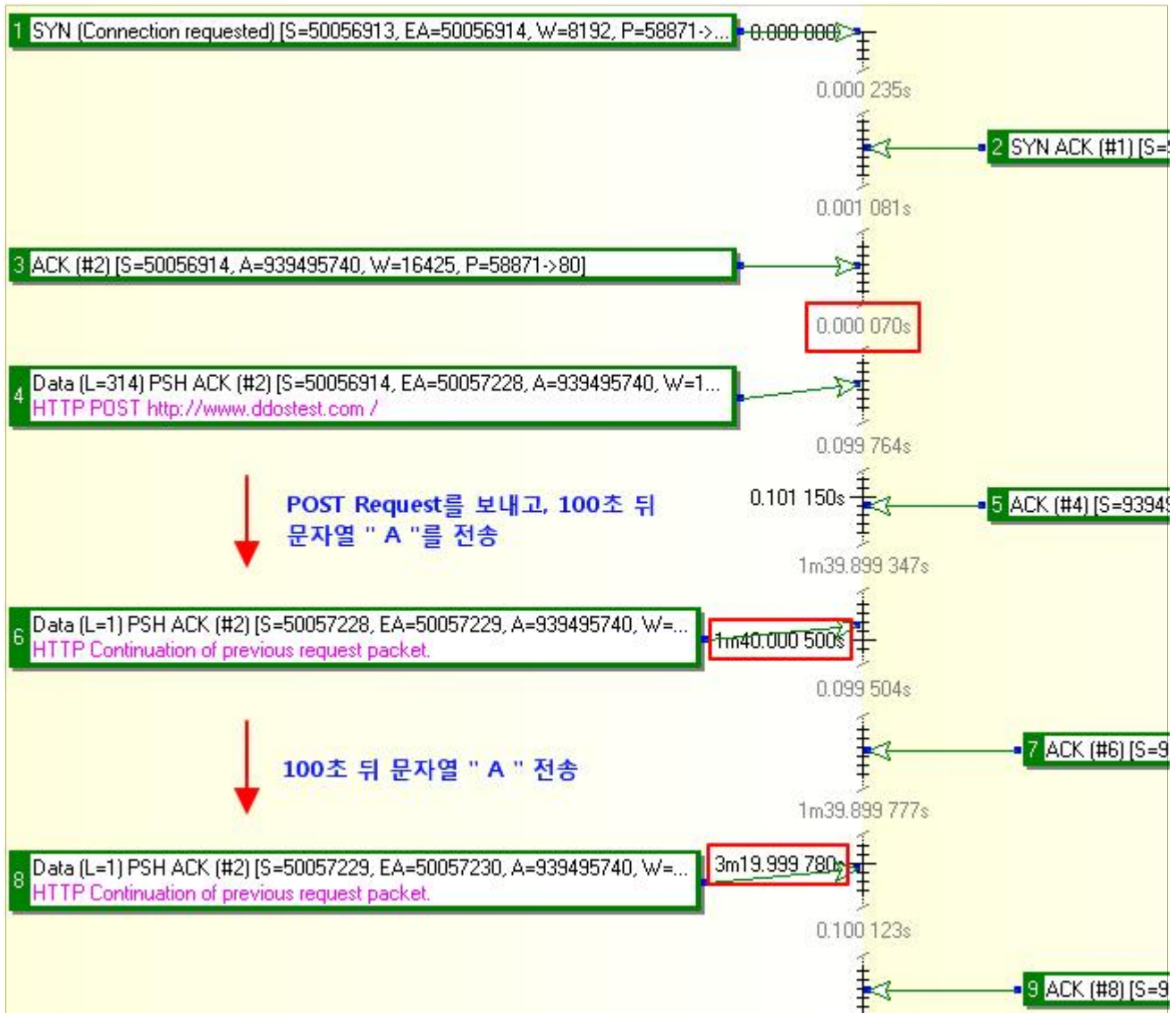
- o 웹서버는 클라이언트가 전달하는 HTTP POST 메시지의 헤더에 정의 되어진 Content-Length 값을 이용하여 데이터를 수신하기 위한 시간을 할당하게 되는데,
- o 만약, 서버가 Max Client(최대 클라이언트 수용)에 도달할 만큼 충분히 많은 클라이언트(좀비PC)를 이용하여 연결을 유지시키면 웹서버는 다른 클라이언트에 대한 정상적인 연결을 제공하지 못하게 되어 서비스 장애가 발생하게 됨
 - ※ GET 방식과 달리 POST 방식은 클라이언트가 서버로 전송할 데이터의 크기를 설정할 수 있기 때문에 서버는 데이터 수신시까지 대기해야 함

o Slow HTTP POST 공격 기반의 패킷 형태

정상적인 HTTP 메시지	Slow HTTP POST DoS 메시지
<pre>POST / HTTP/1.1 Host: User-Agent: Mozilla/5.0 (windows NT 6.1; WOW64; rv Accept: text/html,application/xhtml+xml,application Accept-Language: ko-kr,ko;q=0.8,en-us;q=0.5,en;q=0. Accept-Encoding: gzip, deflate Accept-Charset: EUC-KR,utf-8;q=0.7,*;q=0.7 Connection: keep-alive Referer: http:// Cookie: PHPSESSID=m0depka16sot5rd8ou9avfqi07 Content-Type: application/x-www-form-urlencoded Content-Length: 201 error_return_url=%2Findex.php%3Fmid%3Dcalendar2&mi 40login&act=procMemberLogin&success_return_url=%2F</pre> <p style="text-align: center;">↓ (Post Data 한번에 전송)</p> <pre>67 74 68 3a 20 32 30 31 0d 0a 0d 0a 65 72 72 6f gth: 201erre 72 5f 72 65 74 75 72 6e 5f 75 72 6c 3d 25 32 46 69 6e 64 65 78 2e 70 68 70 25 33 46 6d 69 64 25 33 44 63 61 6c 65 6e 64 61 72 32 26 76 69 64 3d 63 61 6c 65 6e 64 61 72 32 26 76 69 64 3d 26 72 75 6c 65 73 65 74 3d 25 34 30 6c 6f 67 69 6e 26 61 63 74 3d 70 72 6f 63 4d 65 6c 62 65 72 4c 6f 67 69 6e 2d 73 75 63 63 65 73 73 5f 72 65 74 75 72 6e 5f 75 72 6c 3d 25 32 46 69 6e 64 65 78 2e 70 68 70 25 33 46 6d 69 64 25 33 44 63 61 6c 65 6e 64 61 72 32 26 75 73 65 72 5f 69 64 3d 6a 63</pre> <p style="text-align: center;"><다량의 Post Data 전송></p>	<pre>POST / HTTP/1.1 Host: www. User-Agent: Mozilla/4.0 (compat 1.1.4322; .NET CLR 2.0.50313; .! Connection: keep-alive Content-Length: 1000000 Content-Type: application/x-www AAAAAAAAAAAAAAAAAA] ↓ (Post Data 1바이트씩 전송) Checksum: 0xf42b [validation disabled] TCP segment data (1 byte) 00 01 d7 b7 0c 85 00 01 e8 d6 7b 3c 08 00 45 00 ...{< 00 29 77 cf 40 00 37 06 38 29 b7 6e f6 01 79 9c ...)w.@.7. 8).n 6c ca d2 37 00 50 e2 1e cd a0 ba d9 69 7e 50 18 1..7.P. 40 29 f4 2b 00 00 aa 40 29 f4 2b @).+.. @).+</pre> <p style="text-align: center;"><1byte 씩 전송></p>

< Slow HTTP Post DoS의 특징 >

- o Slow HTTP POST DoS는 아래 그림과 같이 POST 데이터를 일정한 간격으로 1바이트씩 분할하여 서버로 전송하여
- o 서버가 해당 데이터를 수신하기 위한 연결상태를 종료하지 못하도록 유지시켜 다른 클라이언트의 연결이 원활하지 못하도록 유도



<Slow HTTP Post DoS를 이용한 HTTP 메시지 요청 형태>

6.2. Slow HTTP Header DoS (Slowloris)

공격자는 서버로 전달할 HTTP 메시지의 **Header** 정보를 비정상적으로 조작하여 웹서버가 헤더 정보를 완전히 수신할 때까지 연결을 유지하도록 하여 가용량을 소비시킴으로 다른 클라이언트의 정상적인 서비스를 방해함

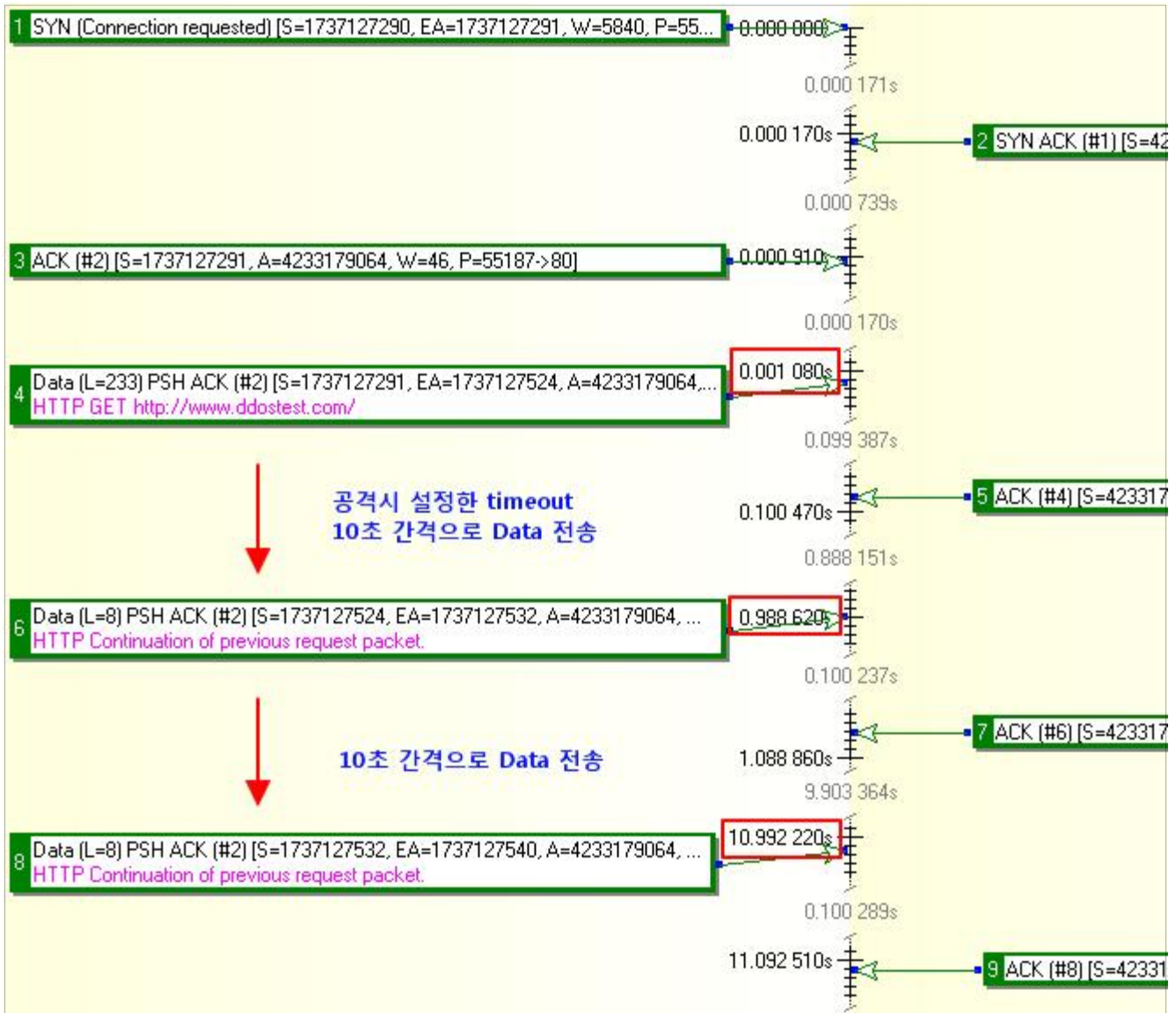
- 웹서버는 HTTP 메시지의 헤더와 바디(데이터)를 개행문자(CRLF, '\r\n\r\n')로 구분
- 만약, 클라이언트가 개행문자 없이 HTTP 메시지를 웹서버로 전달하게 되면, 웹서버는 HTTP 헤더 정보가 다 수신하지 않은 것으로 판단하여 연결을 유지하기 때문에

- 충분히 많은 클라이언트를 이용하여 불완전한 메시지를 전달하면 웹 서버는 다른 클라이언트에 대한 정상적인 연결을 제공하지 못하게 되어 서비스 장애가 발생하게 됨
 - ※ 웹서버는 이러한 불완전한 메시지를 수신하게 되면, 클라이언트로부터의 요청이 끝나지 않은 상태로 인식하기기 때문에 웹로그를 기록되지 않음
- 정상적인 HTTP 메시지의 경우 헤더정보가 '0D0A0D0A'로 종료되지만, Slow HTTP Header (Slowloris) DoS 메시지는 '0D0A0D0A'가 없음

정상적인 HTTP 메시지	Slow HTTP Header DoS 메시지
<pre>Accept-Encoding: gzip, deflate\r\n Accept-Charset: EUC-KR,utf-8;q=0.7,*;q=0.7\r\n Connection: keep-alive\r\n Referer: http:// /\r\n \r\n</pre> <p>72 3a 20 68 74 74 70 3a 2f 2f 77 77 77 2e 6e 61 76 65 72 2e 63 6f 6d 2f 0d 0a 0d 0a</p> <p>< \r\n\r\n으로 종료 ></p>	<pre>Internet Protocol Version 4, Transmission Control Protocol Hypertext Transfer Protocol X-a: t\r\n</pre> <p>00 2e c0 c8 00 00 01 01 08 0a b1 80 d4 6a 19 9b ff 36 58 2d 61 3a 20 62 0d 0a</p> <p>< \r\n으로 종료 ></p>

< Slow HTTP Header DoS의 특징 >

- Slow HTTP Header DoS (Slowloris)는 아래 그림과 같이 불완전한 헤더정보를 가진 HTTP 메시지를 일정한 간격으로 웹서버로 전달하여
- 웹서버가 HTTP 헤더 정보를 완전히 수신하기 위해 연결상태를 종료하지 못하도록 유지시켜 다른 클라이언트의 연결이 원활하지 못하도록 유도



<Slow HTTP Header DoS를 이용한 HTTP 메시지 요청 형태>

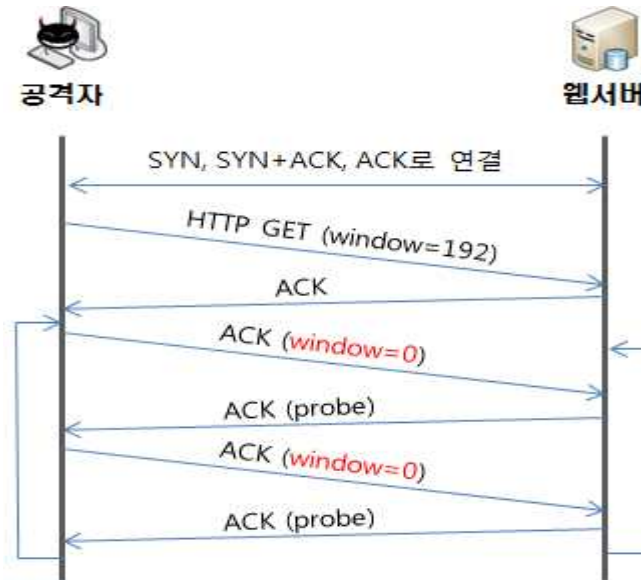
6.3. Slow HTTP Read DoS

공격자는 TCP 윈도우 크기와 데이터 처리율을 감소시킨 상태에서 다수의 HTTP 패킷을 송신하여 웹서버가 정상적으로 응답하지 못하도록 DoS 상태를 유발

- ※ TCP 윈도우 : TCP 헤더의 구성요소로써 수신자 측의 수신 가능한 데이터 버퍼용량
- ※ 데이터 처리율 : 클라이언트가 수신한 데이터를 읽어들이는 단위시간당 처리능력

- 서버와 클라이언트는 연결시 주고받은 TCP 윈도우 크기에 맞게 패킷을 생성하여 주고받게 됨
 - ※ TCP 윈도우 크기를 256byte로 설정하면 1~256byte 범위의 랜덤값(예:192byte)으로 윈도우 크기가 정해져서 SYN 패킷을 전송하게 됨

- 만약, 공격자가 서버와의 연결시, 윈도우 크기를 매우 작은 수치로 설정하여 서버로 전달하게 되면 서버는 클라이언트의 윈도우 크기가 정상으로 환원될 때까지 Pending 상태에 빠지게 되며 연결을 유지시킴
- ※ 공격자는 자신의 TCP 윈도우 크기가 0 byte임을 서버로 전달(ACK) 하면 서버는 공격자의 윈도우 크기가 0 byte임을 인지한 후 더 이상 데이터를 전송하지 않고(pending 상태) 공격자의 윈도우 크기를 점검하는 probe 패킷을 ACK로 전송하며 대기상태에 빠짐



< Slow HTTP Read DoS 과정 >

정상적인 TCP 연결	Slow HTTP Read DoS의 연결
<p><Window Size가 가변적></p>	<p><Window Size가 "0"으로 고정></p>

< Slow HTTP Read DoS의 특징 >

- 서버가 공격자와 정상적인 통신이 이루어지지 않고 지속적으로 연결을 유지하는 상태로 빠지게 하여 서버의 자원을 소진시킴으로 DoS 상태를 유발시킴

7. 기타 서비스 마비 공격

7.1. 해시도스(HashDoS) 공격

웹서버는 클라이언트로부터 전달받는 HTTP 메시지의 매개정보(Parameter) 관리를 위해 해시테이블을 사용함. 조작된 매개정보를 포함한 다량의 메시지는 해시테이블 검색을 위한 인덱스로 사용되는 해시값에 충돌을 발생시켜 정확한 값을 찾기 위해 모든 해시테이블을 검사하게 되는데, 이 때 웹서버의 CPU 자원을 소진하게 되어 정상적인 서비스를 방해함

- o HashDoS는 해시테이블을 이용하는 웹서버를 대상으로 한 공격임
- o 클라이언트에서 HTTP 메시지를 통해 전달되는 각종 매개정보를 관리하는 해시테이블의 인덱스 정보가 중복되도록 유도하여 기저장된 정보 조회시 많은 CPU 자원을 소모하도록 유도하는 공격
 - ※ 웹서버는 GET, POST 방식으로 전송되는 HTTP 메시지에 포함된 매개변수의 효과적인 관리(정보 접근을 쉽고 빠르게 수행)를 위해 해시(Hash) 구조를 사용
- o 많은 수의 매개정보를 전달하면 이러한 정보를 저장하는 해시테이블에서 해시 충돌이 발생하여 정보 조회를 위한 계산시간이 급속도로 증가
 - ※ POST 메시지는 전달되는 파라미터의 길이와 개수에 제한을 두지 않음
 - HashDoS 공격시 해시충돌로 인해 서버의 CPU 사용량이 100%에 도달

```
top - 15:44:17 up 5 days, 22:37, 1 user, load average: 0.16, 0.06, 0.02
Tasks: 136 total, 1 running, 134 sleeping, 0 stopped, 1 zombie
Cpu0 : 1.0%us, 0.0%sy, 0.0%ni, 99.0%id, 0.0%wa, 0.0%hi, 0.0%si, 0.0%st
Cpu1 : 17.3%us, 2.0%sy, 0.0%ni, 80.6%id, 0.0%wa, 0.0%hi, 0.0%si, 0.0%st
Mem: 4151488k total, 1197700k used, 2953788k free, 494040k buffers
Swap: 6289408k total, 0k used, 6289408k free, 445448k cached
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
7702	apache	15	0	26212	10m	3512	S	16.9	0.3	0:03.57	htpd
2608	mysql	18	0	160m	29m	4696	S	2.0	0.7	3:31.16	mysqld
1	root	15	0	2160	680	584	S	0.0	0.0	0:00.39	init
2	root	RT	-5	0	0	0	S	0.0	0.0	0:00.01	migration/0
3	root	34	19	0	0	0	S	0.0	0.0	0:00.00	ksoftirqd/0
4	root	RT	-5	0	0	0	S	0.0	0.0	0:00.00	watchdog/0
5	root	RT	-5	0	0	0	S	0.0	0.0	0:00.02	migration/1
6	root	34	19	0	0	0	S	0.0	0.0	0:00.00	ksoftirqd/1
7	root	RT	-5	0	0	0	S	0.0	0.0	0:00.00	watchdog/1
8	root	10	-5	0	0	0	S	0.0	0.0	0:00.00	events/0
9	root	10	-5	0	0	0	S	0.0	0.0	0:00.00	events/1
10	root	10	-5	0	0	0	S	0.0	0.0	0:00.00	khelper

< 정상적인 상태 >

```
top - 13:39:07 up 5 days, 20:32, 1 user, load average: 169.92, 101.24, 46.88
Tasks: 372 total, 154 running, 216 sleeping, 0 stopped, 2 zombie
Cpu0 : 99.0%us, 1.0%sy, 0.0%ni, 0.0%id, 0.0%wa, 0.0%hi, 0.0%si, 0.0%st
Cpu1 : 99.0%us, 1.0%sy, 0.0%ni, 0.0%id, 0.0%wa, 0.0%hi, 0.0%si, 0.0%st
Mem: 4151488k total, 2958216k used, 1193272k free, 494032k buffers
Swap: 6289408k total, 0k used, 6289408k free, 443928k cached
```

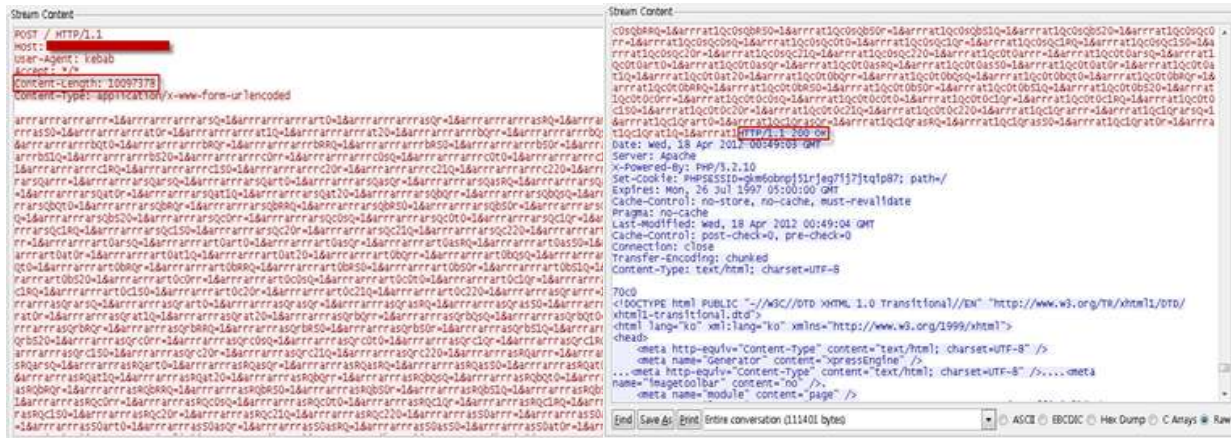
PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
6936	apache	18	0	29892	13m	3368	R	19.8	0.3	0:03.42	htpd
7648	apache	18	0	30308	13m	3108	R	19.8	0.3	0:00.04	htpd
7661	apache	18	0	30268	13m	3304	R	19.8	0.3	0:01.39	htpd
7394	apache	18	0	30824	13m	3108	R	18.8	0.3	0:01.40	htpd
6346	apache	18	0	29896	13m	3368	R	16.8	0.3	0:05.26	htpd
7626	apache	18	0	29144	12m	2900	R	15.8	0.3	0:00.24	htpd
7098	apache	18	0	30408	14m	3368	R	13.8	0.3	0:01.00	htpd
7701	apache	18	0	30656	13m	2952	R	10.9	0.3	0:00.40	htpd
6334	apache	18	0	29896	13m	3356	R	9.9	0.3	0:00.62	htpd
6710	apache	18	0	30404	14m	3368	R	8.9	0.3	0:06.60	htpd
6759	apache	18	0	26468	10m	3324	S	8.9	0.3	0:03.16	htpd
6778	apache	18	0	26468	10m	3368	S	7.9	0.3	0:05.47	htpd
7709	apache	18	0	30656	13m	2952	R	7.9	0.3	0:00.44	htpd
6754	apache	18	0	28316	11m	3368	R	6.9	0.3	0:04.99	htpd
6780	apache	18	0	26468	10m	3368	S	4.9	0.3	0:09.66	htpd
7655	apache	18	0	29896	13m	3356	R	4.9	0.3	0:05.52	htpd
6404	apache	18	0	30924	14m	3368	R	4.0	0.4	0:07.43	htpd
6742	apache	18	0	30824	14m	3304	R	4.0	0.3	0:01.95	htpd
6778	apache	18	0	30824	14m	3368	R	4.0	0.3	0:05.39	htpd
7083	apache	18	0	30920	14m	3324	R	4.0	0.4	0:03.62	htpd
7596	apache	18	0	29144	12m	2900	R	4.0	0.3	0:00.23	htpd
7635	apache	18	0	29656	12m	2904	R	4.0	0.3	0:00.38	htpd
6721	apache	18	0	30924	14m	3368	R	2.0	0.4	0:02.51	htpd
6940	apache	18	0	26468	10m	3108	S	2.0	0.3	0:01.01	htpd
7422	apache	18	0	29568	13m	3324	R	2.0	0.3	0:00.22	htpd
7492	apache	18	0	26212	9.9m	3108	S	2.0	0.2	0:00.52	htpd
7650	apache	18	0	29796	13m	3108	R	2.0	0.3	0:00.72	htpd

< HashDoS를 이용한 공격시 상태 >

- HTTP 메시지와 함께 전달하는 파라미터에 '&' 기호를 이용하여 다량의 파라미터를 전달

1615	DTE	Link 1	61.110.235.194	1518	HTTP Continuation	TCP ACK (1081 → 80)	IP (61.110.235.194 → 121.156.108.253)	ETHER	4/17/2012	18h:56m:55.360.540s	54.447263
1616	DTE	Link 1	61.110.235.194	1518	HTTP Continuation	TCP ACK (1081 → 80)	IP (61.110.235.194 → 121.156.108.253)	ETHER	4/17/2012	18h:56m:55.360.573s	54.447316
1617	DTE	Link 1	61.110.235.194	1518	HTTP Continuation	TCP ACK (1081 → 80)	IP (61.110.235.194 → 121.156.108.253)	ETHER	4/17/2012	18h:56m:55.360.574s	54.447317
1618	DTE	Link 1	61.110.235.194	1518	HTTP POST	TCP ACK (1089 → 80)	IP (61.110.235.194 → 121.156.108.253)	ETHER	4/17/2012	18h:56m:55.360.616s	54.447369
1619	DTE	Link 1	61.110.235.194	1518	HTTP Continuation	TCP ACK (1107 → 80)	IP (61.110.235.194 → 121.156.108.253)	ETHER	4/17/2012	18h:56m:55.360.680s	54.447423
1620	DTE	Link 1	61.110.235.194	1518	HTTP Continuation	TCP ACK (1059 → 80)	IP (61.110.235.194 → 121.156.108.253)	ETHER	4/17/2012	18h:56m:55.360.681s	54.447424
1621	DTE	Link 1	61.110.235.194	1518	HTTP Continuation	TCP ACK (1107 → 80)	IP (61.110.235.194 → 121.156.108.253)	ETHER	4/17/2012	18h:56m:55.360.682s	54.447425
1622	DTE	Link 1	121.156.108.253	70	TCP ACK (80 → 1081)	IP (121.156.108.253 → 61.110.235.194)	ETHERTYPE	4/17/2012	18h:56m:55.360.697s	54.447440	
1623	DCE	Link 1	121.156.108.253	64	TCP ACK (80 → 1081)	IP (121.156.108.253 → 61.110.235.194)	ETHERTYPE	4/17/2012	18h:56m:55.360.701s	54.447444	
1624	DTE	Link 1	61.110.235.194	1518	HTTP Continuation	TCP ACK (1089 → 80)	IP (61.110.235.194 → 121.156.108.253)	ETHER	4/17/2012	18h:56m:55.360.808s	54.447551
1625	DCE	Link 1	121.156.108.253	64	TCP ACK (80 → 1107)	IP (121.156.108.253 → 61.110.235.194)	ETHERTYPE	4/17/2012	18h:56m:55.360.808s	54.447551	
1626	DTE	Link 1	61.110.235.194	1518	HTTP Continuation	TCP ACK (1089 → 80)	IP (61.110.235.194 → 121.156.108.253)	ETHER	4/17/2012	18h:56m:55.360.841s	54.447584
1627	DTE	Link 1	61.110.235.194	1518	HTTP Continuation	TCP ACK (1089 → 80)	IP (61.110.235.194 → 121.156.108.253)	ETHER	4/17/2012	18h:56m:55.360.844s	54.447587
1628	DTE	Link 1	61.110.235.194	1518	HTTP Continuation	TCP ACK (1089 → 80)	IP (61.110.235.194 → 121.156.108.253)	ETHER	4/17/2012	18h:56m:55.360.907s	54.447649
1630	DTE	Link 1	61.110.235.194	1518	HTTP Continuation	TCP ACK (1089 → 80)	IP (61.110.235.194 → 121.156.108.253)	ETHER	4/17/2012	18h:56m:55.360.909s	54.447652
1631	DTE	Link 1	61.110.235.194	1518	HTTP Continuation	TCP ACK (1089 → 80)	IP (61.110.235.194 → 121.156.108.253)	ETHER	4/17/2012	18h:56m:55.360.911s	54.447654
1632	DCE	Link 1	121.156.108.253	64	TCP ACK (80 → 1089)	IP (121.156.108.253 → 61.110.235.194)	ETHERTYPE	4/17/2012	18h:56m:55.360.931s	54.447674	

< HashDoS 도구를 이용한 공격패킷 >



< POST 메시지에 '&'로 구분된 대용량 파라미터를 포함 >

[참고] 해시, 해시테이블, 해시충돌

o 해시(Hash)란?

- 데이터를 저장하고 찾기를 하는데 사용되는 자료 구조의 한 종류로 찾고자 하는 문자열을 특정한 함수로 처리하여 얻은 값으로 데이터의 위치를 찾는 방법임
- 데이터를 찾는 속도가 데이터의 개수의 영향을 거의 받지 않는 특성을 지니고 있어 효율적이고 빠르게 데이터의 위치를 찾을 수 있음

o 해시테이블(Hash Table)은 해시함수의 연산에 의해 구해진 위치에 각 정보를 한 개 이상 보관할 수 있도록 구성된 기억 공간



o 해시 충돌(Hash Collision)은 해시 함수가 서로 다른 두 개의 입력 값에 대해 동일한 출력 값을 내는 상황을 의미

- 대부분의 해시 함수는 상당히 긴 입력값으로부터 고정된 범위의 출력값을 생성하므로 해시 출력 값의 범위보다 훨씬 더 큰 범위 값을 받는 경우 충돌이 발생
- 해시충돌을 일으키는 특정 매개변수를 가진 POST 메시지가 서버로 전달하는 경우, 웹서버는 충돌되는 해시값을 비교해야 하기 때문에 CPU 자원이 고갈됨

7.2. 헐크도스(HulkDoS) 공격

- o HULK(Http Unbearable Load King) DoS는 웹서버의 가용량(웹서버로 접속할 수 있는 최대 클라이언트 수)을 모두 사용하도록 하여 정상적인 서비스가 불가능하도록 유도하는 GET Flooding 공격 유형으로,
 - o 공격대상 웹사이트 주소(URL)를 지속적으로 변경하여 DDoS 차단정책을 우회한다는 특징을 가짐

- o DDoS(Denial of Service, 분산서비스공격)는 대량의 트래픽을 유발하거나 비정상적인 접속을 발생시켜 웹사이트를 마비시키는 공격으로 구분됨
 - ※ 네트워크 대역폭을 잠식하여 트래픽의 흐름을 방해하거나 웹서버 자원에 부하를 유발함으로써 서비스 속도가 느려지게 하여 정상적인 웹서비스를 불가능하도록 유도

- o 웹서버 자원에 부하를 유발하는 공격 기법 중 가장 많이 사용하는 방법은 특정 웹사이트 주소(URL)를 호출하도록 하는 GET Flooding 공격 기법임
 - ※ GET Flooding 공격이란 웹서버와 TCP Connection(3-way handshaking)을 연결한 후 짧은 시간 안에 다수의 동적 콘텐츠를 요청함으로써 웹 서비스의 부하를 유발하는 공격

- o 웹서버 자원 부하유발 공격을 차단하기 위해서는 특정 웹사이트에 접근할 수 있는 회수를 제한하는 임계치(Threshold) 설정방법을 적용할 수 있음
 - ※ 예를 들어, 특정 URL의 임계치를 10으로 설정하면, 클라이언트마다 특정 URL에 동시에 10번 이상 접속이 불가능하게 되어 웹서버의 부하를 감소시키는 효과를 가져옴
 - ※ 임계치는 고정된 URL주소에 대해서만 설정이 가능함

- o 만약, 특정 웹사이트 주소(URL)가 계속 변경된다면 이러한 임계치 설정기반의 방어는 불가능해질 수 있음
 - ※ 웹사이트 주소 뒤에 임의의 파라미터를 포함하도록 하면 DDoS 대응 장비는 임계치가 설정된 웹사이트 주소와 전혀 다른 주소로 인식하게 되어 차단하지 않음

- o HULK DoS는 URL에 임의의 파라미터를 포함하도록 하여 URL 주소를 계속 변경하여 특정 URL에 대한 임계치 기반의 DDoS 차단을 우회하기 위한 공격 기법임
 - ※ HULK는 Imperva 사의 Barry Shteiman이라는 연구원이 자신의 블로그인 Nerd에 DDoS 연구나 교육을 위해 만든 도구이지만 해커에 의해 DDoS 공격도구로 악용되고 있음

□ HULK DoS 공격기법

- o 클라이언트에서 웹서버로 Request URL을 전달하면 웹서버는 해당 URL에 해당하는 웹사이트를 클라이언트에게 전달함

- o Request URL에는 아이디, 비밀번호와 같은 정보를 파라미터로 포함하여 전달할 수 있는데, 이때는 Request URL 뒤에 "?" 기호와 함께 임의의 문자열을 포함할 수 있음

```

· 파라미터 추가 형식 : ?이름1=값1&이름2=값2&....&이름n=값n
(예) http://search.nOOO.com/search.nOOO?where=nexearch&query=get&fbm=1&ie=utf8

```

- o HULK DoS는 이러한 파라미터를 지속적으로 변경하여 웹서버로 전달

```

HTTP GET [redacted]net/?BDXCNW=ZJYwVA --- TCP PSH ACK [50847
HTTP GET [redacted]net/?NBTGABYC=UJBKNOT --- TCP PSH ACK [508
HTTP GET [redacted]net/?XTLAXHMIBS=DSLOZCLRL --- TCP PSH ACK
HTTP GET [redacted]net/?QWDFCZ=UCKQWJFP --- TCP PSH ACK [5085
HTTP GET [redacted]net/?DCXZEYQZYA=SVY --- TCP PSH ACK [50851
HTTP GET [redacted]net/?BJYAYHKDV=MTNSGWSYW --- TCP PSH ACK

```

※ 파라미터가 지속적으로 변경되는 것을 제외하고는 일반적으로 GET Flooding과 동일한 형태임

□ HULK DoS 공격도구 분석

- o HULK DoS 공격도구는 python으로 작성되었으며 윈도우즈 환경에서는 Active python을 설치하여 아래 명령어로 간단히 실행할 수 있음

```
c:\> hulk.py http://test.com/
```

- o HULK DoS의 소스파일과 실행화면

```

# -----
# HULK - HTTP Unbearable Load King
#
# this tool is a dos tool that is meant to put heavy load on HTTP servers in order to bring them
# to their knees by exhausting the resource pool, its is meant for research purposes only
# and any malicious usage of this tool is prohibited.
#
# author : Barry Shteyman , version 1.0
# -----
import urllib2
import sys
import threading
import random
import re

#global params
url=' '
host=' '
headers_useragents=[]
headers_referers=[]
request_counter=0
flag=0
safe=0

def inc_counter():
    global request_counter
    request_counter+=1

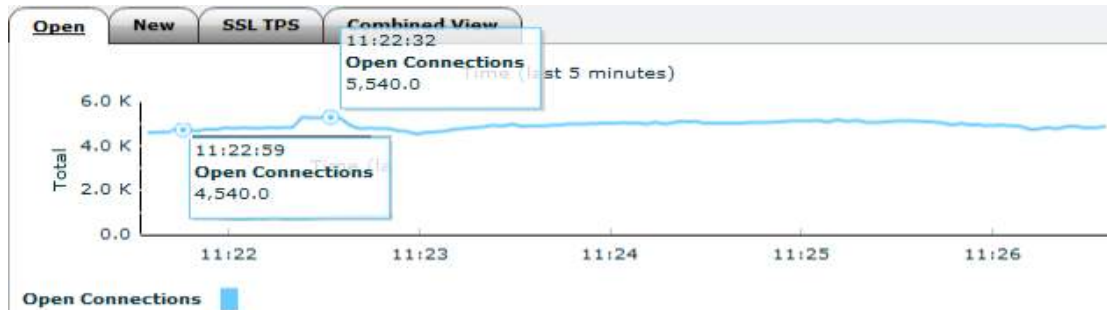
def set_flag(val):
    global flag
    flag=val

def set_safe():
    global safe
    safe=1

# generates a user agent array
def useragent_list():
    global headers_useragents
    headers_useragents.append('Mozilla/5.0 (X11; U; Linux x86_64; en-US; rv:1.9.1.3) Gecko/20090913 Firefox/3.5.3')
    headers_useragents.append('Mozilla/5.0 (Windows; U; Windows NT 6.1; en-US; rv:1.9.1.3) Gecko/20090824 Firefox/3.5.3 (.NET CLR 3.5.30721)')
    headers_useragents.append('Mozilla/5.0 (Windows; U; Windows NT 5.2; en-US; rv:1.9.1.3) Gecko/20090824 Firefox/3.5.3 (.NET CLR 3.5.30721)')
    headers_useragents.append('Mozilla/5.0 (Windows; U; Windows NT 6.1; en-US; rv:1.9.1.1) Gecko/20090718 Firefox/3.5.1')
    headers_useragents.append('Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US) AppleWebKit/532.1 (KHTML, like Gecko) Chrome/4.0.215.1 (Windows NT 5.1)')
    headers_useragents.append('Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; WOW64; Trident/4.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; InfoPath.2)')
    headers_useragents.append('Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.0; Trident/4.0; SLCC1; .NET CLR 2.0.50727; .NET CLR 3.0.30729; .NET CLR 3.5.30729; InfoPath.2)')
    headers_useragents.append('Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 5.2; Win64; x64; Trident/4.0)')
    headers_useragents.append('Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 5.1; Trident/4.0; SV1; .NET CLR 2.0.50727; InfoPath.2)')
    headers_useragents.append('Mozilla/5.0 (Windows; U; MSIE 7.0; Windows NT 6.0; en-US)')
    headers_useragents.append('Mozilla/4.0 (compatible; MSIE 6.1; Windows XP)')
    headers_useragents.append('Opera/9.80 (Windows NT 5.2; U; ru) Presto/2.5.22 Version/10.51')
    return(headers_useragents)

```

- o HULK DoS 공격도구는 서버와 초당 평균 1,000개 정도를 연결하여 약 10M 규모의 공격트래픽을 전달 (아래 그림에서 연결수치 4,540 → 5,540)



< HULK DoS 툴을 이용한 DoS 공격시 서버 연결 정보 >

- o HULK DoS 공격도구를 이용하여 공격시 서버의 CPU 사용량이 50% 도달

```
top - 11:23:21 up 76 days, 13:48, 3 users, load average: 7.28, 5.62, 5.35
Tasks: 156 total, 1 running, 155 sleeping, 0 stopped, 0 zombie
Cpu(s): 0.0%us, 0.1%sy, 0.0%ni, 99.9%id, 0.0%wa, 0.0%hi, 0.0%si, 0.0%st
Mem: 3115136k total, 2706924k used, 408212k free, 234628k buffers
Swap: 4096532k total, 108k used, 4096424k free, 2300904k cached

PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND
17271 root 15 0 2532 1136 788 R 0.3 0.0 0:04.18 top
1 root 15 0 2160 644 556 S 0.0 0.0 0:00.95 init
2 root RT -5 0 0 0 S 0.0 0.0 0:17.79 migration/0
3 root 34 19 0 0 0 S 0.0 0.0 0:00.03 ksoftirq/0
4 root RT -5 0 0 0 S 0.0 0.0 0:00.00 watchdog/0
5 root RT -5 0 0 0 S 0.0 0.0 0:22.32 migration/1
6 root 34 19 0 0 0 S 0.0 0.0 0:00.08 ksoftirq/1
7 root RT -5 0 0 0 S 0.0 0.0 0:00.00 watchdog/1
8 root RT -5 0 0 0 S 0.0 0.0 0:18.60 migration/2
9 root 34 19 0 0 0 S 0.0 0.0 0:00.03 ksoftirq/2
10 root RT -5 0 0 0 S 0.0 0.0 0:00.00 watchdog/2
11 root RT -5 0 0 0 S 0.0 0.0 0:20.04 migration/3
12 root 37 19 0 0 0 S 0.0 0.0 0:00.03 ksoftirq/3
13 root RT -5 0 0 0 S 0.0 0.0 0:00.00 watchdog/3
14 root 10 0 0 0 0 S 0.0 0.0 0:16.35 events/0
```

< 정상적인 상태 >

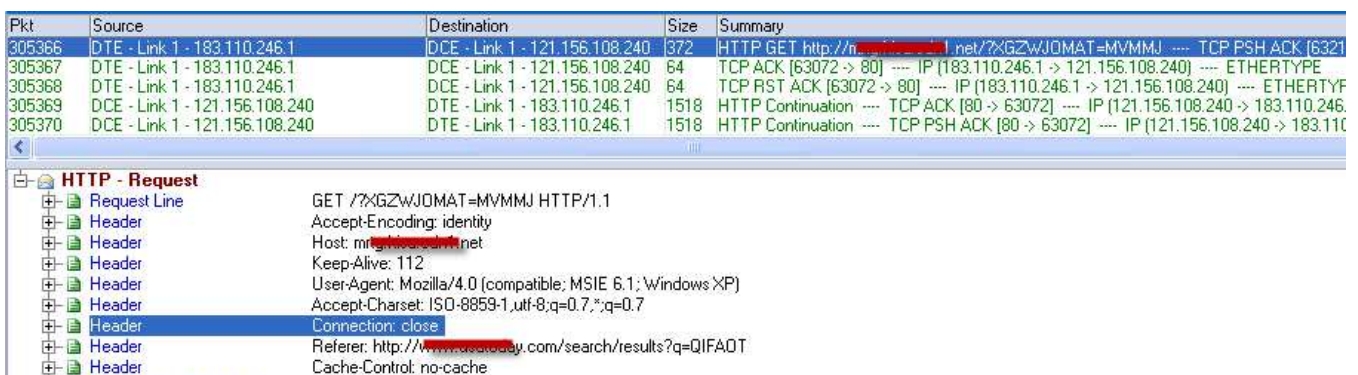
```
top - 11:22:18 up 76 days, 13:47, 3 users, load average: 9.67, 5.74, 5.38
Tasks: 163 total, 1 running, 162 sleeping, 0 stopped, 0 zombie
Cpu(s): 51.0%us, 7.8%sy, 0.0%ni, 35.9%id, 0.1%wa, 0.7%hi, 4.5%si, 0.0%st
Mem: 3115136k total, 2708288k used, 406848k free, 234452k buffers
Swap: 4096532k total, 108k used, 4096424k free, 2299268k cached

PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND
17430 apache 15 0 26792 6212 2680 S 11.9 0.2 0:04.81 httpd
17432 apache 15 0 26792 6208 2676 S 11.9 0.2 0:04.32 httpd
22559 apache 15 0 26832 5420 1952 S 11.6 0.2 0:03.03 httpd
22564 apache 15 0 26792 5420 1952 S 11.6 0.2 0:00.36 httpd
17421 apache 15 0 26792 6188 2660 S 11.3 0.2 0:04.95 httpd
22507 apache 15 0 26832 5420 1952 S 11.3 0.2 0:03.24 httpd
22508 apache 15 0 26792 5412 1944 S 11.3 0.2 0:02.87 httpd
22639 apache 15 0 26832 5420 1952 S 11.3 0.2 0:00.95 httpd
22788 apache 15 0 26792 5420 1952 S 11.3 0.2 0:00.37 httpd
17431 apache 15 0 26792 6188 2664 S 10.9 0.2 0:04.51 httpd
22739 apache 15 0 26792 5412 1944 S 10.9 0.2 0:00.41 httpd
2689 pcap 15 0 2064 468 340 S 6.3 0.0 5:17.54 p0f
7949 root 18 0 7272 1664 1272 S 1.3 0.1 0:06.83 rotatelogs
2323 root 11 -4 13672 920 568 S 0.3 0.0 34:59.09 auditd
2816 mysql 15 0 5708 748 4792 S 0.3 2.4 2:29.71 mysqld
3306 root 16 0 2060 656 576 S 0.3 0.0 0:07.33 hald-addon-stor
7942 root 18 0 26564 8716 5400 S 0.3 0.3 0:00.37 httpd
1 root 15 0 2180 644 556 S 0.0 0.0 0:00.95 init
2 root RT -5 0 0 0 S 0.0 0.0 0:17.79 migration/0
```

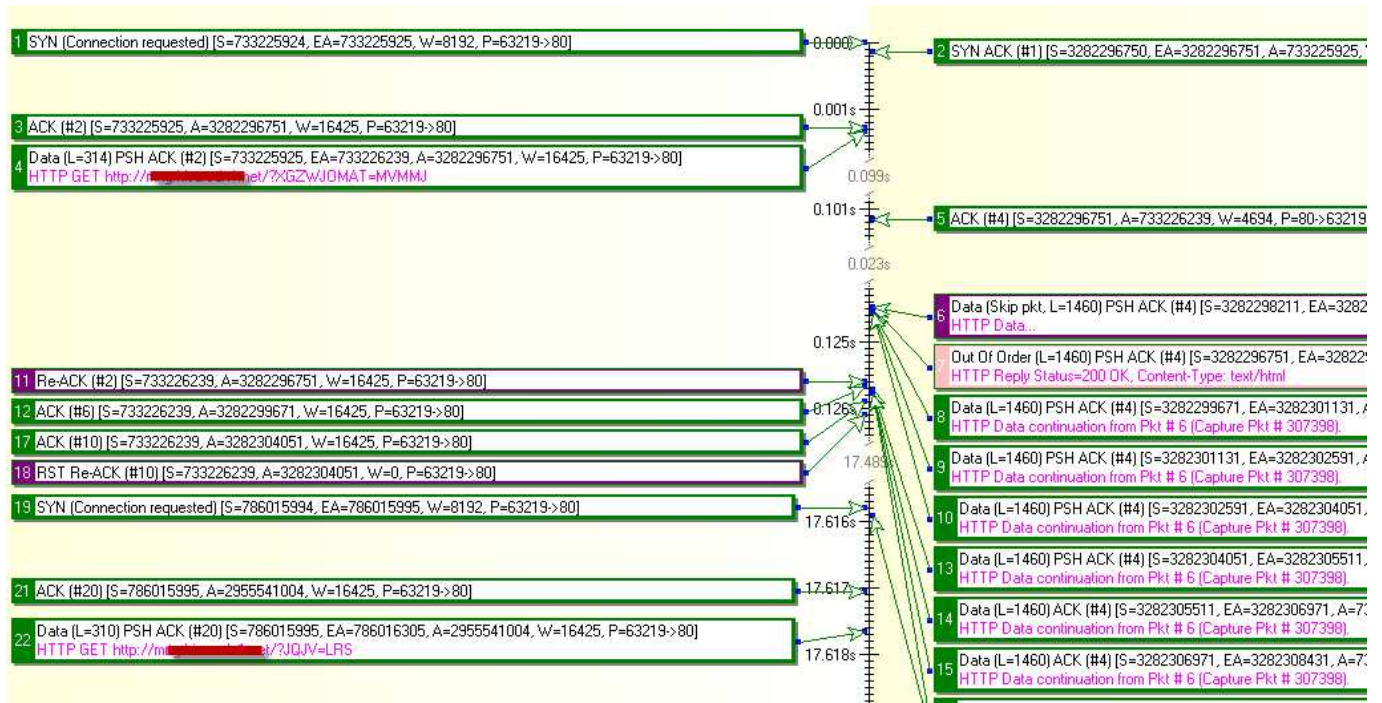
< HULK DoS를 이용한 공격시 상태 >

- o HULK DoS 공격도구에서 생성하는 공격트래픽 형태

- 아래 그림과 같이 HOST 뒤에 “/?”를 입력 후 임의의 문자를 무작위로 입력하면, 웹서버는 파라미터 유효검사를 하게 되는데
- 파라미터가 유효하지 않더라도 “/” URL로 동작하여 클라이언트에게 200ok 응답을 하게 됨



< Request Header URL, User-Agent, Referer의 지속적 변화>



< 정상적인 GET 요청으로 동작함 >

[별첨 4] DDoS 공격유형별 대응방안 (세부)

1. UDP/ICMP Flooding 공격 방어

- 일반적으로 UDP/ICMP Flooding 공격을 수행할 때 단일 좀비 (Zombie)에서 발생시키는 패킷은 그 크기와 전송 간격 또한 다양함 하지만 방화벽으로 모든 패킷이 물리게 되어 단일 시간(보통 초 단위)에 대량의 패킷이 집중될 수밖에 없음
- UDP/ICMP 방어 및 완화를 위해서는 네트워크 스위치에서의 ACL 기반의 프로토콜 차단이나 방화벽의 임계치(Threshold) 설정 기능을 고려할 수 있음

1.1. (방안 1) ACL (Access Control List) 설정을 이용한 차단

웹서버 혹은 운영 장비에 대한 접근 제어 목록에 차단하고자 하는 프로토콜 정보를 다음과 같이 ACL에 UDP/ICMP DROP 정보로 설정하여 차단

```
ip access-list extended acl-Drop-Example
seq 1 deny udp any any
seq 2 deny icmp any any
seq 3 permit ip 1.1.1.0/24 any
seq 4 permit ip 2.2.2.2/29 any
seq 5 permit tcp any any
seq 6 permit ip 3.3.3.3.0/24 any
seq 7 permit ip 4.4.4.0/24 any
seq 8 permit icmp host 5.5.5.5 any
```

<ACL로 UDP/ICMP를 차단하는 설정 예>

1.2. (방안 2) INBOUND 패킷에 대한 임계치 설정을 이용한 차단

운영 장비로 유입되는 INBOUND 패킷을 기준으로 PPS(Packet Per Second) 수치를 유입되는 수치보다 낮게 설정(예: 10)¹⁾하여 임계치 이상의 UDP/ICMP 트래픽의 유입을 차단

※ UDP/ICMP 공격 방어를 위한 1단계/2단계 정책은 UDP 및 ICMP를 이용한 서비스를 제공하지 않는 경우에 적용할 수 있으므로 적용시 주의가 요구됨

1) PPS 제한 수치는 서비스에 지장을 주지 않는 범위 내에서 단계적으로 조정하여 설정할 수 있다.

2. SYN Flooding 공격 방어

- SYN Flooding 공격 발생시 서버는 자신이 연결하고 있지 않은 출발지에서 유입되는 다양한 패킷을 처리하고 확인하는 과정에서 과부하를 야기
- SYN Flooding 공격방어 및 완화를 위해서는 TCP 프로토콜이 연결 지향성(Connection-Oriented)이라는 점을 이용하고 Handshake 과정에 위배된 TCP 패킷이 유입될 경우 비정상적인 트래픽으로 구분하여 차단하는 방법이 가장 효과적
- 또한 정상적인 트래픽의 임계치에 의거하여 비정상적으로 많은 트래픽을 유발하는 출발지 IP에 대해 차단하는 임계치 기반의 DDoS 방어

2.1. (방안 1) 임계치 기반의 SYN Flooding 차단

방화벽의 IP당 SYN 요청에 대한 PPS 임계치를 단계적으로 조정하여 SYN Flooding을 차단

항목	PPS	BPS	차단 시간	설정
SYN (1:1)	120		300 초	<input checked="" type="checkbox"/> 사용

< PPS 설정을 통한 SYN 연결 요청 제한 예 >

2.2. (방안 2) First SYN Drop (Spoofed) 설정에 의한 차단

SYN 패킷을 보내는 클라이언트가 진짜 존재하는지를 파악하여 차단하는 방법으로 클라이언트로부터 전송된 첫 번째 SYN을 Drop하여 재요청 패킷이 도착하는지를 확인하여 Spoofing 여부를 판단하고 차단

2.3. SYN 이외의 Flooding 공격 방어

- SYN 이외의 Flooding 공격을 방어하기 위해서는 다음과 같은 다양한 형태의 DDoS 방어 기법을 사용
- 먼저 정상적인 TCP 세션 연결 이후 정상적인 트랜잭션(Transaction)이 수행되는지에 대해 검증하여 만약 정상적인 트랜잭션이 이루어질 경우

서버로 전달해야 하고, 정상적인 트랜잭션 없이 TCP 세션 연결만 수행할 경우에는 서버로 전달하지 않아야 서버에 부하 증가 현상을 막을 수 있음

- 또한 네트워크의 정상적인 환경에서 설정된 각 트래픽 유형별 임계치를 통하여 과도한 TCP 세션 연결에 대해 차단

3. Get Flooding 공격 방어

- 일반적으로 임계치 기반의 방어 기법을 적용. 즉, HTTP Get Flooding 시 수행되는 TCP 연결 요청의 임계치 값과 HTTP Get 요청의 임계치 값의 모니터링을 통하여 비정상적으로 많은 트래픽을 발생하는 출발지 IP에 대한 선별적인 차단 적용
- 이러한 세션 연결 기반 공격의 경우 출발지 IP는 변조될 수가 없기 때문에 출발지 IP 기준의 임계치를 통하여 방어가 가능
- 향후 이러한 세션 기반 공격의 경우 다수의 사용자를 이용한 DDoS 공격이 이루어지며 하나의 출발지 IP 당 발생하는 DDoS 공격 트래픽 양은 방어 장비에서 설정된 임계치 정책보다도 더 작게 공격할 수 있는 위험을 가지고 있음. 따라서 상세한 DDoS 공격을 방어하기 위해서는 정상적으로 HTTP Get 요청을 하는지에 대한 정밀한 검사 기법이 요구

3.1. (방안 1) 콘텐츠 요청 횟수에 대한 임계치 설정에 의한 차단

- Get Flooding 공격은 특정 동적 콘텐츠 데이터를 다량으로 요청하는 것이 특징이므로 IP마다 콘텐츠를 요청할 수 있는 횟수에 임계치를 설정하여 일정 규모 이상의 Get 요청을 차단하므로 Get Flooding 공격을 방어

```

array_set ::BLOCKLIST { }
set ::attacktime 3
set ::maxquery 30
set ::holdtime 30

when HTTP REQUEST {
  if { [HTTP::uri] matches_regex {[^jpg|gif|swf|css|png|bmp|js]$} } {
    if { [info exists ::BLOCKLIST/$IP::remote_addr] } {

```

< L7 스위치를 이용한 Get 요청 횟수 제한설정 예 >

3.2. (방안 2) 시간별 웹페이지 URL 접속 임계치 설정에 의한 차단

- o URL에 대한 시간별 임계치를 설정하여 임의의 시간 안에 설정한 임계치 이상의 요청이 들어온 경우 해당 IP를 탐지하여 방화벽의 차단 목록으로 등록

3.3. (방안 3) 웹스크래핑(Web-Scraping) 기법을 이용한 차단

- o L7 스위치를 운영하는 경우, 웹스크래핑 기능을 이용하면 요청 패킷에 대해 특정 쿠키(Cookie)값이나 자바스크립트(Javascript)를 보내어 클라이언트로부터 원하는 값이 재요청(혹은 응답) 패킷에 없는 경우 해당 패킷을 차단

```
GET / HTTP/1.1
Accept: image/jpeg, application/x-ms-application, image/gif, application/xaml+xml, image/png, application/x-ms-xap, application/x-shockwave-flash, application/vnd.ms-excel, application/vnd.ms-powerpoint, application/msword, */*
Accept-Language: ko-KR
User-Agent: Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; Trident/4.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0; Tablet PC 2.0; .NET4.0; MALC)
Accept-Encoding: gzip, deflate
Host: mrtg.kisa.cdn.net
Connection: Keep-Alive
```

<Web scraping 기능 이용 후 Client의 요청 패킷>

```
HTTP/1.1 200 OK
Date: Tue, 05 Apr 2011 08:44:19 GMT
X-Powered-By: PHP/5.3.6
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html
Set-Cookie: T546d87a=a68e65aa6b8da462a56a49658043da808e7170d4e3a7d44c4d9ad66a Path=/
Transfer-Encoding: chunked

<script type="text/javascript">
function str_reverse(str) { return ((str.split('')).reverse()).join(''); }
function test1(var table = "00000000 77071096 80E612C 9909318A 0766c419 7064a8ff 9603a135 960493a3 0EDD8B12 796c8BAA E005691E 97620988 0986aC2B 7E817C8D 67B82D07 90BF1091 10871064 6AB020F2
F3697148 84B4110E 1A0A487D 8680C4E8 F4848151 83D385C7 136C9856 6468A8C0 F0D2F97A B085C96C 14015C4F 6306C2D9 FA0F3063 8D8080F5 3B4620C8 4C06105E 056041E4 A2677172 7C03E4D1 AB040447 D26085D0
A5A8E768 1381A88A 4282788C 8880C908 AC8CF04D 120861C1 4387C775 0C0600C7 A8D18039 7409704C 3182D03A C80731B0 8E006116 184F4883 3687C413 CFA26199 8680A03C 7802808E 74038808 C50C7062 83088764
2FA6F7C7 8984C113 C1631048 86667D3D 768C4190 070871D8 98D2308C 8FD5102A 73818199 0686851F 0F8E1E43 E88B811F 7807C9A2 8F00F914 9609A88E E10E9818 7F8A008B 0866102D 93649C97 48635C01 868851E4
3C6C636D 8863308B F262994E 8C06918C 1805A78D 8238F4C1 F709C457 650609C6 37078395 888E888A FC89887C 6300E06E 330A0949 8C03707F F804AC93 A0E36138 2A8131CE A38C01A4 6A8838E2 8A8A4811 10089101
A423C48D D3D8F41F 8186E96A 144ED08C A8478848 DA8088D0 4A842073 11031263 A8A8A83F 08D07C59 5D05713C 3702A1AA 8E0E101D C90C7086 578B5253 20A8F818 8686A809 C61E4A9F 3E8E909C 2909C998 830C9822
C767A888 88118117 2E848081 8785813B C88A8C4D E088823D 9A898188 0186820C 7481D29A E8A84739 9082778F 048D2613 718C1A81 C4180112 94841884 8080A818 7A8A3A88 E40C808B 9388F890 8A80A827 76878811
F00F9148 870A8102 11012268 8080C2FE F762573D B65607C8 108C1671 668896C7 FE41876 8903288D 10DA7A5A 670DAAC2 F08D8F6F 8E8E8F9F 17878E43 60808E05 6080A1E8 A101937E 388C82C4 4E8E8252 038867E1
A88C5767 F8F80666 48827648 8802880A AF0A184C 3681A8F6 A1047A6D D868EFC3 A8769735 318E884F 46098679 C881838C 8C668B1A 2366C2A0 5268E236 C0C27705 80847033 22021899 5505262F C38A1886 828D0828
70845A8D 5C818A84 C2878A87 8308C731 2C898E8F 368EA811 0864C280 8C872256 738AA38C 028D818A 8C808A8F 8081818F 72878783 23805173 938E8A82 E28E7A14 788128A8 0C801818 82C88888 8385888C 7C8E8E8F
0888E21 868D0D84 F18E342 8808918F 1F8A838E 818E18C0 F8892638 8F8897E1 18874777 88881A89 FF80F6A7D 608D38CA 3301887C 8F8198FF F8A2A8A9 038A8F8D 166CF813 A08A8278 0780C8E8 4E848184 198388C2
8F82588D 08818107 4988828D 18E8778E 4E816A8A 08895A8C 4088088C 3F88888F 888CA813 08888E83 478C8F7F 3083F8E9 8088F21C CABAC28A 3383933D 248A4A88 B8D36053 C0D70093 34865279 2389878F 83687A2E
C81A1A8B 56818D02 2A8F2894 84888E37 C38C8EAL 5A83CF18 202E8F8D"]
var s1 = "09y287v8"
var s2 = "v"
var s3 = "v"
var s4 = "v"
var start = s1.charCodeAt(0);
var end = s2.charCodeAt(0);
var arr = new Array(255);
var n = Math.pow((end - start) + 1, 8);
for (var i=0; i<n; i++)
arr[i] = 0;
for (var i=0; i<n; i++)
for (var j=0; j<n; j++)
var t = arr[i].charCodeAt(0);
t += arr[j] * Math.pow(256, j);
if (arr[i].charCodeAt(0) == end) {
break; } else { arr[i] = t; }
var chng = arr.join("");
var str = chng + s1;
var crc = 0;
for (var k = 0; k < str.length; k = k + 1)
for (var i = 0; i < 256; i = i + 1)
for (var j = 0; j < 256; j = j + 1)
crc = (crc + str.charCodeAt(k) * 0x000000FF * j * i) % 256;
```

<Web Scraping 기능 이용 후 L7 Switch의 응답 패킷>

4. Get with Cache-Control 공격 방어

- o HTTP Get Flooding 과 마찬가지로 일반적인 방어 기법인 임계치 기반의 DDoS 공격 방어가 효과적임. 즉, TCP 세션 요청과 HTTP 요청에 대한 임계치 기법으로 방어가 가능
- o 하지만 이 공격은 Cache-Control이라는 HTTP 헤더 옵션을 사용하는 공격이므로, HTTP Cache-Control 헤더 옵션 별 임계치 정책으로 방어해야지만 큰 효과를 얻을 수 있음

4.1. (방안 1) 방화벽에 캐싱공격 문자열을 포함한 IP 차단

- o HTTP Request 메시지를 분석(Parsing) 하여 캐싱 공격에 해당하는 문자열(no-Store, must-revalidate)을 포함하는 경우, 문자열을 포함한 IP 정보를 수집하여 방화벽에 등록하여 공격트래픽을 차단

4.2. (방안 2) L7 스위치를 이용한 캐싱 공격 차단

- o L7 스위치를 이용하면 좀 더 세분화된 차단정책을 적용할 수 있는데, 다음과 같이 HTTP Header의 Cache-Control에 특정 문자열(no-Store, must-revalidate)을 포함하는 경우 해당 IP의 접속을 차단하는 방법을 이용

```
when HTTP_REQUEST {
  if { [[HTTP::header "User-Agent"] contains "must-revalidate" ] } {
    log local2. "[IP::remote_addr] is Drop, Host:[HTTP::host], must-revalidate is using in User-Agent"
    drop
  }
  elseif { [[HTTP::header "Cache-Control"] contains "must-revalidate" ] } {
    log local2. "[IP::remote_addr] is Drop, Host:[HTTP::host], must-revalidate is using in Cache-Contr"
    drop
  }
  elseif { [[HTTP::header "Proxy-Connection"] contains "Keep-Alive" ] } {
    log local2. "[IP::remote_addr] is Drop, Host:[HTTP::host], Proxy-Connection is using in Keep-Alive"
    drop
  }
}
```

< HTTP 문자열 검색을 통한 IP 차단 설정 예 >

5. HTTP Continuation Data Flooding 공격 방어

- HTTP Continuation Data 패킷은 Request Header없이 Data만 보내는 형태로 해당 패킷에 대해 분석하지 않고 해당 패킷을 무시하도록 하여 공격패킷을 차단
- 이러한 방법을 적용하기 위해서는 웹서버 앞단에 캐싱장비가 필요한데, 이는 웹서버가 직접 트래픽을 수신하는 경우 웹서버의 트래픽 수신버퍼가 모두 차게 되어 다른 연결을 원활이 제공할 수 없기 때문에 충분한 연결을 유지할 수 있는 캐싱장비를 통해 효과적으로 대응 가능
- 또한, 이 공격의 가장 큰 특징은 연결 자원고갈과 함께 대역폭을 소진한다는 것이므로 대역폭 소진에 대한 대안도 고려해야 함

6. TCP Session 공격 방어

6.1. (방안 1) Connection Timeout/Keep-Alive/Time-Wait 설정을 통한 차단

- o Connection Timeout에 설정된 시간동안 Client와 웹서버 사이에 데이터 신호의 이동이 전혀 없을 경우 Connection을 종료하도록 설정하거나 웹서버에서 keepalive 기능을 사용하는 경우에는 Keepalivetimeout을 사용하여 세션 공격을 차단

```
#  
# Timeout: The number of seconds before receives and sends time out.  
#  
Timeout 5
```

< httpd.conf 의 timeout 설정을 통한 세션공격 차단 예 >

- o 다만, 공격자는 방어 정책 우회를 위해 Content-Length와 실제 전송하는 데이터의 크기를 조정하고 데이터 전송 term을 짧게 가져가면서도 Connection을 오랫동안 유지할 수 있으므로 이 대응 방안은 일정한 한계가 있음

6.2. (방안 2) L7 스위치의 임계치 설정 기능을 이용한 차단

- o L7 스위치를 운영하는 경우, IP당 Connection Limit을 설정하여 하나의 Client와 Server가 맺을 수 있는 Connection 수치를 조절하여 차단

```
when RULE_INIT {  
    set :: max_connecitons_per_ip 200  
    array set :: active_clients { }  
    array set write_client {  
        10.41.0.610  
        10.0.0.2  
    }  
}
```

< L7 스위치의 Connection Limit 설정 스크립트 예 >

7. URL Redirect 우회 공격 방어

- 2009년 7월 DDoS 공격시 사용되었던 대응 기법중의 하나가 URL Redirect기법으로 좀비 PC에서 특정 URL 요청을 수행할 경우, 좀비 PC에 다른 URL로 리다이렉트 신호를 전송함으로써 공격을 차단
- 특정 URL에 대해서 원하는 URL로 Redirect하는 Rule을 적용시켜 Zombie를 구분하는 Rule을 적용하고 있지만 Redirect를 인식하는 Zombie를 차단하는 대응기법은 보완되어야 함

```
when HTTP_REQUEST {
  if { ([HTTP::host] eq "www.example.com" || [HTTP::host] eq "www.example.net" ) {
    HTTP::redirect http://www.example.com/
  }
}
```

< L7 Switch 의 Redirect 차단 iRule >

8. Slow HTTP POST 공격 방어

8.1. (방안 1) 접속 임계치 설정을 통한 차단

- 특정한 발신지 IP에서 연결할 수 있는 동시 접속수(Concurrent Connection)에 대한 최대값을 설정함으로써 대응이 가능. 이 방법은 한 개의 IP에서 대량의 연결을 시도하는 공격을 차단하기에 적절.
- 유닉스/리눅스 계열의 운영체제를 운영한다면 운영체제의 방화벽 설정 도구인 iptables 명령어를 이용하여 차단

```
iptables -A INPUT -p tcp --dport 80 -m connlimit --connlimit-above 30 -j DROP
# 30개 이상의 concurrent connection에 대한 차단
```

8.2. (방안 2) Connection Timeout과 Keepalivetimeout 설정을 통한 차단

- Connection Timeout에 설정된 시간동안 Client와 웹서버 사이에 데이터 신호의 이동이 전혀 없을 경우 웹서버는 연결된 Connection을 종료.

- o Slow POST 공격 톨은 오랫동안 Connection을 유지하기 위해 데이터 전송 시 일정한 term을 갖고 있어 일정 구간의 패킷을 분석하여 현재 발생하는 DDoS 공격의 패턴을 파악하여 Timeout을 설정하는 방법을 활용

```
#
# Timeout: The number of seconds before receives and sends time out.
#
Timeout 5
```

< httpd.conf 의 timeout 설정 예 >

- o 또한, 웹서버에서 keepalive 기능을 사용하는 경우에는 Keepalivetimeout을 사용하여 Slow POST 공격에 대응

8.3. (방안 3) RequestReadTimeout(mod_reqtimeout Module) 설정을 통한 차단

- o Apache 2.2.15 버전과 그 이후 버전에서는 클라이언트 요청에 대한 더욱 세부적인 제한을 줄 수 있는 RequestReadTimeout이라는 지시자를 제공
- o 이 지시자는 클라이언트의 요청(header and body)이 지정된 시간 내에 완료되지 않을 경우 오류 코드를 클라이언트에게 전송하여 Slow Attack을 차단

RequestReadTimeout header=5 body=8

```
# header=5
# HTTP Header Request가 5초 이내에 완료되지 않으면, FIN 패킷을 보내
  연결을 해제하며, 이 때 Apache Server는 408 Response Code로 응답 한다.
  Timeout이 지난 후 Connection을 종료한다.

# body=8
# POST 요청 이후 8초 동안 데이터가 오지 않을 시 FIN을 보내 연결 해제를
  요청하고 RST로 Connection을 종료한다. 일반적인 POST Request 시
  Apache Server는 3-Way Handshake 이후 POST Request의 Content-Length
  값만큼의 데이터가 수신될 때까지 Connection을 Open 상태로 수신을 기다
  린다.
```

< RequestReadTimeout 지시자의 사용 예 >

```

RequestReadTimeout header=10 body=30
# Client의 요청 header 전송완료는 10초 이내 , 요청 body 전송 완료는 30초
  이내로 제한

RequestReadTimeout body=10,MinRate=1000
# client의 요청 body 전송완료는 10초 이내로 제한, 단 client가 data를 전송
  할 경우 1,000bytes 전송 시점마다 1초씩 증가

```

< RequestReadTimeout 지시자 활용 예 >

9. DNS 공격 방어

9.1. (방안 1) DNS 서버의 다중화를 통한 DNS 공격 트래픽 분산 처리

- o 가능한 DNS 서버를 다중으로 구성하여 특정 서버로의 공격이 발생 하더라도 다른 서버가 해당 요청에 대해 응답할 수 있도록 구성

```

;; ANSWER SECTION:
[redacted] com.      86400    IN       NS       ns55.[redacted] com.
[redacted] com.      86400    IN       NS       ns259.[redacted] com.
[redacted] com.      86400    IN       NS       ns231.[redacted] com.
[redacted] com.      86400    IN       NS       ns33.[redacted] com.
[redacted] com.      86400    IN       NS       ns11.[redacted] com.

```

< 특정 DNS요청에 대해 다수의 DNS서버 등록 >

9.2. (2단계) IPTABLE을 이용한 ACL 기반의 차단

- o 대부분 DNS요청 패킷은 512Byte를 넘을 수 없기 때문에 처리가 가능한 대역폭에서 서비스를 하는 경우라면 iptables등을 이용해 공격 패킷을 차단

```

iptables -A INPUT -p udp --dport 53 -m length --length 512:1500 -j DROP

```

< DNS Query가 512Byte이상일 경우 해당 패킷 차단 >

10. HashDoS 공격 방어

- o HashDos는 웹서버의 설정값 변경만으로도 차단이 가능
 - Tomcat, PHP, Ruby 등 최신 버전에서는 HTTP Post Parameter 개수에 제한을 둘 수 있는 기능을 추가하였으므로 개수 제한 적용을 위해 최신버전으로 업데이트

o 톰캣(Tomcat)에서의 HashDoS 차단 방안

- (방안 1 : 파라미터 개수 제한) TOMCAT_HOME/conf/server.xml의 Connector 부분을 다음과 같이 설정

```
<Connector port="8009" protocol="AJP/1.3" maxParameterCount="xxx" .../>
```

※ 적용가능버전: Tomcat 5.5.35, 6.0.35, 7.0.23

- (방안 2 : POST 메시지 크기 제한) 사이즈를 제한하는 것이 서비스에 문제가 없는 경우 적용하며, TOMCAT_HOME/conf/server.xml의 Connector 부분을 다음과 같이 설정

```
<Connector port="8009" protocol="AJP/1.3" maxPostSize="xxx" .../>
```

o PHP에서의 HashDoS 차단 방안

- PHP 5.4.0 RC4로 업데이트 한 후, php.ini 파일에서 'max_input_var'에서 최대 HTTP POST Parameter 개수를 설정
- PHP 5.4.0 RC4 이하인 경우, PHP Source에서 소스변경부분을 수동으로 설정하고 <http://svn.php.net/viewc?view=revision&revision=321003>에서 수정된 소스를 다운받아 재빌드하여 적용

※ 대상 : PHP_5_4/main/main.c, PHP_5_4/main/php_globals.h, PHP_5_4/main/php_variables.c

11. Hulk DoS 공격 방어

11.1. (방안-1) 접속 임계치 설정을 통한 차단

- 특정한 발신지 IP에서 연결할 수 있는 동시 접속수(Concurrent Connection)에 대한 최대값을 설정하여 한 개의 IP에서 대량의 연결을 시도하는 공격을 차단
- 유닉스/리눅스 계열의 운영체제를 운영한다면 운영체제의 방화벽 설정 도구인 iptables 명령어를 이용하여 차단이 가능하며 예제는 아래와 같음

```
iptables -A INPUT -p tcp -dport 80 -m connlimit --connlimit-above 30 -j DROP
# 30개 이상의 concurrent connection에 대한 차단
```

11.2. (방안-2) HTTP Request의 HOST 필드값에 대한 임계치 설정을 통한 차단

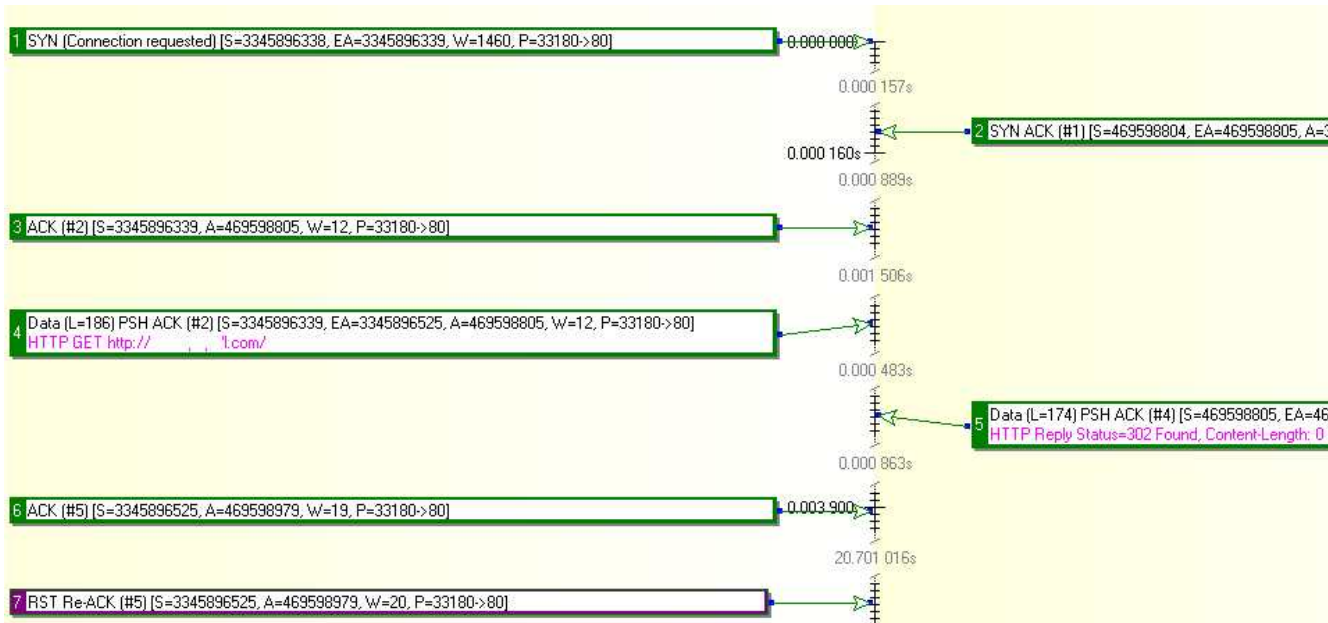
- HULK DoS는 URL을 지속적으로 변경하기 때문에 URL이 아닌 HTTP Request에 포함된 HOST 필드값을 카운트하여 임계치 이상인 경우 차단하도록 설정



< HTTP Request의 HOST 필드 정보 >

o (방안-3) 302-Redirect를 이용한 차단

- 대부분의 DDoS 공격 tool은 302-Redirect 요청에 대해 반응하지 않는 것이 특징이므로 여러 웹사이트 URL 중에 공격당하기 쉬운 웹사이트(예: “/”)에 대한 Redirect 처리를 통해,
- 자동화된 DDoS 공격 tool을 이용한 공격을 사전에 차단하여 웹서버의 부하를 감소시킬 수 있음



< 서버의 302-Redirect 요청시 클라이언트에서 Reset 이 발생함 >

[별첨 5] 주요 DDoS 공격도구 분석결과

R.U.D.Y 기반의 Slow POST 공격분석

2011. 7. 11(월) KISA 해킹대응팀

□ R.U.D.Y 툴 분석

가. R.U.D.Y 소개

R.U.D.Y는 R-U-Dead-Yet의 약어로서 2010년 11월에 Raviv Raz에 의해 개발되고 소개)되었다. 이 도구는 다음의 주소에서 다운로드 받을 수 있다.
<http://www.hybridsec.com/tools/rudy/>

Name	Last modified	Size	Description
Parent Directory	26-Jan-2011 23:03	-	
R-U-Dead-Yet-v2.0.ta...>	26-Jan-2011 23:40	28k	
R-U-Dead-Yet-v2.1.ta...>	26-Jan-2011 23:39	28k	
R-U-Dead-Yet.tar.gz	26-Jan-2011 23:40	23k	
r-u-dead-yet-v2.1.exe	26-Jan-2011 23:42	3.1M	
r-u-dead-yet-v2.2.exe	26-Jan-2011 23:16	3.1M	
r-u-dead-yet-v2.2.ta...>	26-Jan-2011 23:13	28k	

다운받은 파일의 압축을 풀면 아래와 같은 파일을 확인할 수 있다.

```

-rw-r--r-- 1 root root 78871 2011-01-21 23:35 BeautifulSoup.py
-rw-r--r-- 1 root root 68419 2011-07-01 11:11 BeautifulSoup.pyc
-rw-r--r-- 1 root root 1368 2011-01-22 00:10 README
-rw-r--r-- 1 root root 135 2011-07-01 13:50 rudeadyet.conf
-rwxr-xr-x 1 root root 12110 2011-01-22 00:11 r-u-dead-yet-v2.2.py
-rw-r--r-- 1 root root 13785 2011-01-21 23:35 socks.py
-rw-r--r-- 1 root root 15005 2011-07-01 11:11 socks.pyc

```

- BeautifulSoup.py: XML이나Html Source에 대해 parsing 작업 수행
- README: 사용방법
- rudeadyet.conf: r-u-dead-yet-v2.2.py의 기본 환경설정 파일
- r-u-dead-yet-v2.2.py: 공격 실행 파일
- socks.py: SOCKS proxy를 이용 시 사용

2) <http://chaptersinwebsecurity.blogspot.com/2010/11/universal-http-dos-are-you-dead-yet.html>

나. R.U.D.Y 실행 방법 및 공격 유형

점검용 패킷 발생은 interactive한 방법과 config를 사용하는 방법 2가지로 수행할 수 있으며, 2가지 모두 공격 트래픽 및 패턴은 동일하다.

1) interactive한 방법(URL 입력과 간단한 옵션 설정만으로 공격 수행 가능)

#> r-u-dead-yet-v2.2.py <URL(FQDN)>과 같이 입력하면 아래와 같이 질의/응답식으로 수행된다.

```
root@manager2:/script/jeongwoolin/rudy# ./r-u-dead-yet-v2.2.py http://[redacted]
Found 2 forms to submit. Please select number of form to use:
1 ) http://[redacted] /
2 ) http://[redacted] /
> 1
Found 2 parameters to attack. Please select number of parameter to use:
1 ) user_id
2 ) password
> 1
Number of connections to spawn: (default=50)
> 10
Use SOCKS proxy? [yes/no] (Default=no)
>
[*] Attacking: http://[redacted]et/
[*] With parameter: user_id
```

해당 페이지에 대해 submit이 있는지 확인

어떤 field 값을 이용하여 POST 요청을 할 것인지 선택

초당 몇 개의 data를 전송할 것인지 선택

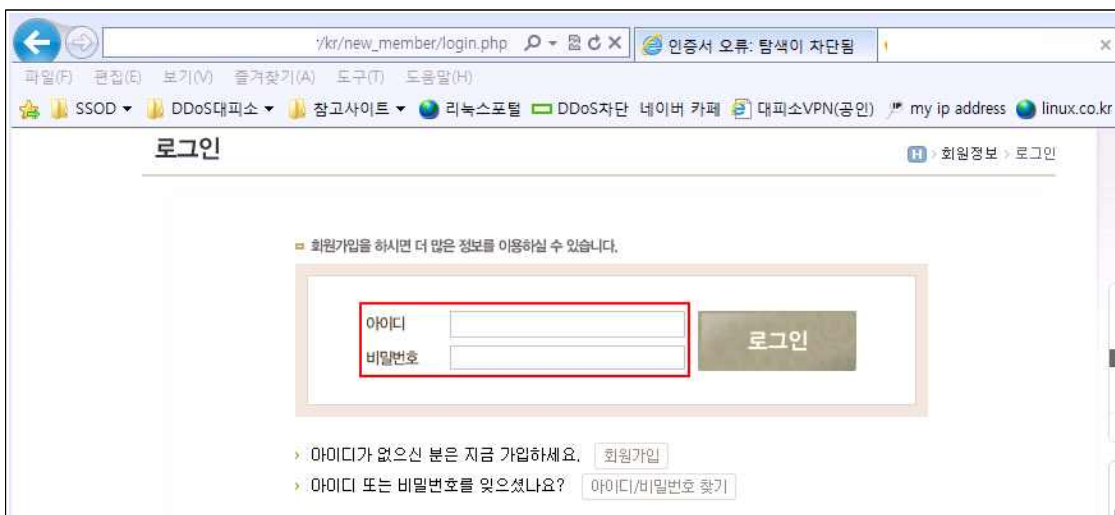
SOCKS proxy를 이용할 것인지 선택

공격 시작

2) Config값을 이용한 방법(실행코드 및 conf 파일 내용을 수정하여 공격 수행)

① Target web page의 submit parameter 찾기

POST 요청을 수행할 페이지의 소스에서 submit 부분의 value를 쉽게 찾을 수 있다.



② rudeadyet.conf 파일 수정

- URL : Target URL 입력
- Number of connection : 연결할 connection 숫자 지정(연결된 connection을 유지하기 위해 10초당 1byte씩 데이터를 전송한다.)
- Attack parameter : form 전송시의 name 지정

```
root@manager2: ~/rudy# cat rudeadyet.conf
[parameters]
URL: http://www. /kr/new_member/login.php
number_of_connections: 3
attack_parameter: user_login
proxy_addr: ""
proxy_port: 0
root@manager2: ~/rudy#
```

③ r-u-dead-yet-v2.2.py 파일의 Content-Length 수정

공격을 유지하고자 하는 만큼의 Content-Length를 지정한다.

- Rudeadyet.conf 파일의 number_of_connection이 "1"일 경우 1개의 connection이 맺어지고, 이 connection을 유지하기 위해 10초에 1byte만큼 전송
- r-u-dead-yet-v2.2.py의 Content-Length가 100,000,000일 경우 전송되는 전체 data의 사이즈가 100,000,000이 될 때 까지 전송

```
POST /(path)s HTTP/1.1
Host: /(host)s
Connection: keep-alive
Content-Length: 100000000
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; .NET CLR 1.1.4322; .NET CLR 2.0.50727)
/(cookies)s
/(params)s=""/(path):path,"host":host,"cookies":cookies,"param":attack_parameter}
else:
headers = ""
POST /(path)s HTTP/1.1
Host: /(host)s
Connection: keep-alive
Content-Length: 100000000
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; .NET CLR 1.1.4322; .NET CLR 2.0.50727)
/(params)s=""/(path):path,"host":host,"param":attack_parameter}
q = Queue()
for i in range(num_of_processes):
p = Client(hostname,port,headers,proxy_addr,proxy_port).start()
q.put(p,False)
```

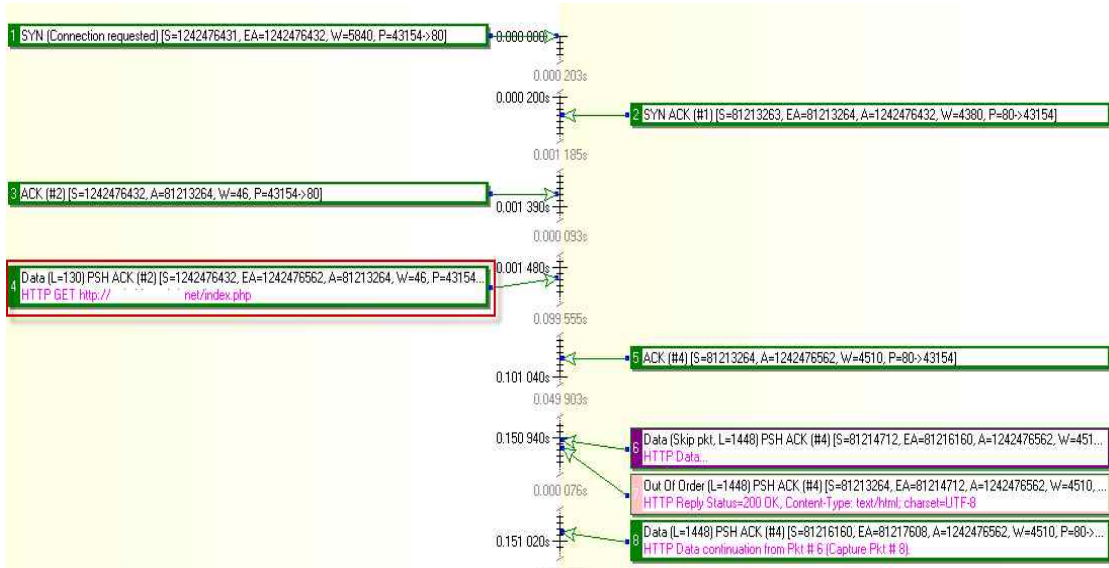
④ r-u-dead-yet-v2.2.py 실행

```
root@manager2:/script/jeongwoolim/rudy# ./r-u-dead-yet-v2.2.py
```

3) 공격 수행 과정

① form name에 대한 scanSubmit Scan

시작 명령을 받은 R.U.D.Y는 먼저 웹페이지를 스캔하여 POST 메소드를 사용하고 있는 입력폼을 찾는다. submit 값 스캔은 아래와 같이 GET 요청을 수행하여 POST 메소드와 파라미터를 찾는다.



공격대상 페이지의 소스코드가 아래와 같을 경우 2개의 form name 값이 스캔된다.

```

<form action="." method="post" onsubmit="return procFilter(this, widget_login)" id="fo_login_widget">
  <div class="idpwWrap">
    <div class="idpw">
      <input name="user_id" type="text" title="user id" />
      <input name="password" type="password" title="password" />
    </div>
  </div>
</form>

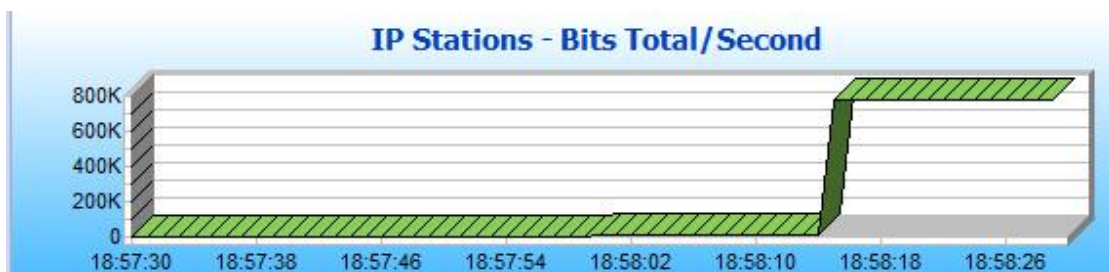
```

공격자가 선택한 submit값을 대상으로 POST 요청을 시작한다.



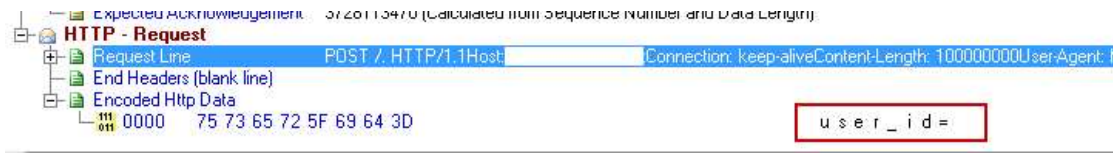
② Slow POST 요청 시작

설정된 connection 수치만큼 연결을 맺고 각 connection 별로 10초에 1개의 continuation data(지속적인 POST 요청)를 발생시킨다. 이때 POST 요청의 body에는 'A'값을 삽입하여 전송한다. connection 수치를 10,000으로 설정할 경우 트래픽 상승량은 아래와 같다.

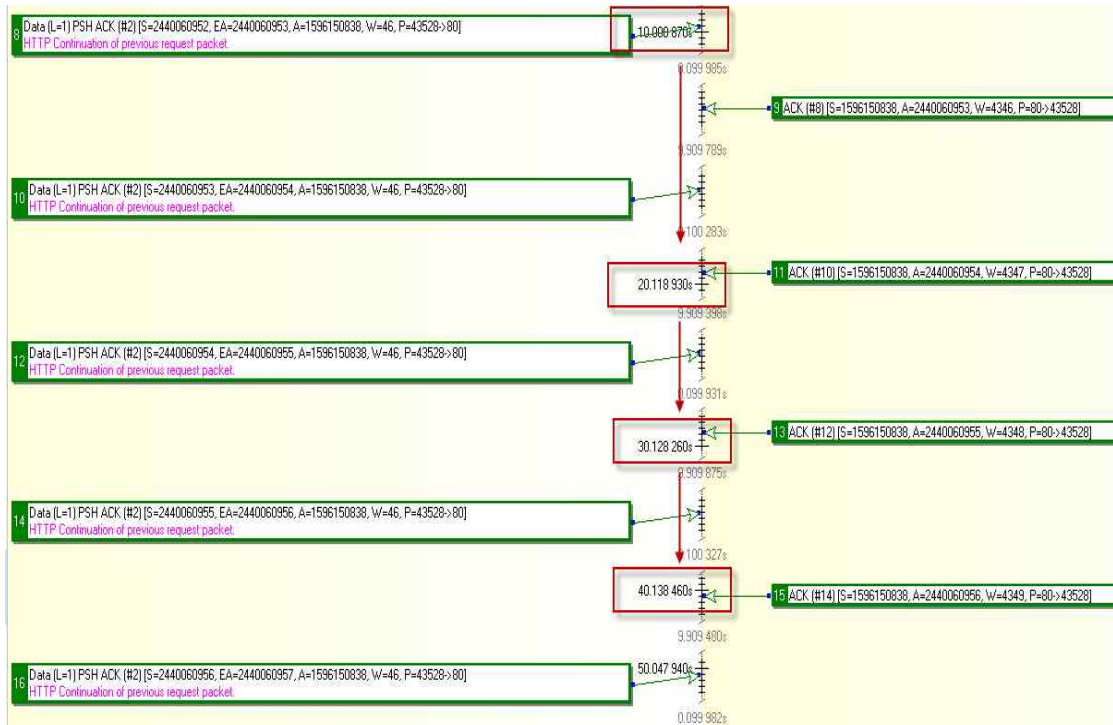


최초의 POST 요청 패킷을 살펴보면 POST 헤더에는 100,000,000 값이 content-length 필드 값으로 설정되어 있고(이 값은 틀에서 제공하는 default 값이며

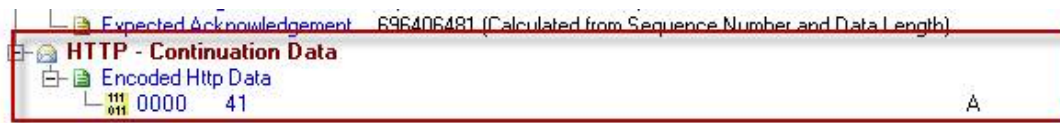
변경 가능하다) POST body에는 파라미터(공격대상 submit 값)이 삽입되어 있다.



하나의 connection은 아래와 같이 10초당 1개의 continuation data(데이터가 여러개의 패킷으로 분할되어 전송될 경우 최초 패킷을 제외한 패킷에는 헤더가 없으며 오직 data(or POST의 body)만을 포함하고 있음)를 발생시킨다.



아래와 같이 continuation data에는 'A'라는 값이 삽입되어 있다.



다. R.U.D.Y 공격 대응 방안

R.U.D.Y는 Slow POST DDoS 공격을 발생시킬 수 있는 툴의 한 종류에 불과하다. 그러므로 R.U.D.Y 툴이 마치 Slow POST 공격의 전부인 것으로 생각해서는 안된다. 응용계층의 공격은 어디까지나 공격 패킷의 형태를 다양화 할 수 있으므로 도구에서에서 생성하는 패킷의 형태에 집중해서는 안되며, 지속적인 테스트를 통해 서버의 장애 지점을 정확히 파악하고 대안을 세우는 데에 집중해야 한다는 점을 먼저 밝힌다.

1) Signature를 이용한 차단

R.U.D.Y는 최초 form name 값을 확인하기 위해 GET 요청을 수행한다. python 언어 기반인 R.U.D.Y는 원하는 웹 page을 가져오기 위해 urllib2라는 모듈을 import 하게 되어있기 때문에 GET 요청 헤더에는 아래와 같이 특정 문자열 (python-urllib/2.6)이 포함된다. 방어자 입장에서 가장 대응하기 쉬운 DDoS공격은 시그니처가 있는 공격이다. R.U.D.Y의 최신 버전에는 이런 시그니처가 존재하는 것이다. 물론 R.U.D.Y라는 툴이 테스트 용도로 배포되었기 때문에 이러한 시그니처가 존재가 가능한 것이며, 임의의 공격자가 소스코드를 수정함으로써 얼마든지 이런 패턴을 제거할 수도 있다. 그러나 R.U.D.Y의 코드가 배포버전에서 수정되지 않았다면 아래 시그니처를 이용하여 공격 차단이 가능하다.

```
Hypertext Transfer Protocol
GET /kr/new_member/login.php HTTP/1.1\r\n
Accept-Encoding: identity\r\n
Host: www      :r\r\n
Connection: close\r\n
User-Agent: Python-urllib/2.6\r\n
\r\n
```

2) Concurrent connection에 대한 임계치 설정

특정한 발신지 IP에서 연결할 수 있는 동시 접속수에 대한 최대값을 설정함으로써 대응이 가능하다. 이 방법은 한 개의 IP에서 대량의 연결을 시도하는 공격을 차단하기에 적절하다. 리눅스 상에서 방화벽을 설정하는 도구인 iptables 명령어를 이용하는 예제는 아래와 같다.

```
Ex. 30개 이상의 concurrent connection에 대한 차단
# iptables -A INPUT -p tcp --dport 80 -m connlimit --connlimit-above 30
-j DROP
```

3) connection timeout과 keepalivetimeout에 대한 설정

connection timeout에 설정된 시간동안 client와 웹서버 사이에 데이터 신호의 이동이 전혀 없을 경우 웹서버는 연결된 connection을 종료한다. R.U.D.Y와 같은 Slow POST 공격 툴은 오랫동안 connection을 유지하기 위해 데이터 전송 시 일정한 term을 갖게된다. 배포된 R.U.D.Y는 기본적으로 10초에 한 번씩 데이터를 전송하고 있다. 이처럼 term을 길게 유지하는 Slow POST 공격 시에는 timeout 값을 유용하게 활용할 수 있다. 그리고 웹서버에서 keepalive 기능을 사용하는 경우에는

keepalivetimeout을 사용하여 Slow POST 공격에 대응 할 수 있다. 다만, 공격자는 방어 정책 우회를 위해 Content-Length와 실제 전송하는 데이터의 크기를 조정하고 데이터 전송 term을 짧게 가져가면서도 connection을 오랫동안 유지할 수 있으므로 이 대응 방안은 일정한 한계가 있다. 또한 네트워크 환경에 따라 작은 값의 timeout 은 정상 사용자의 서비스에도 영향을 미칠 수 있다.

4) Caching 또는 L7 스위치를 이용한 웹 서버의 connection 자원관리

웹 서버의 가용성 확장 및 보안을 위해 웹 서버 앞단에 Caching 또는 L7 스위치를 연동하는 사례가 많다. Caching 또는 L7 스위치는 client의 어떤 요청 하나가 완료된 이후에 웹 서버와 통신할지 여부를 결정하게 된다. 즉, Slow POST 유형은 하나의 요청이 완료되지 않은 상태로 connection을 유지하게 되는데, 이러한 경우 Caching 또는 L7 스위치는 요청이 완료되지 않았기 때문에 해당 요청을 웹 서버로 전송하지 않는다. 예를들어 Content-Length가 10이고 10초당 1byte의 data를 보내는 형태의 공격 이라면 웹 서버로 전송되는 실제 요청은 100초당 한 개가 되는 것이며, 전송이 완료 되기 이전 모든 connection은 Caching 장비가 유지하게 된다. 즉, 하나의 connection 내에서 data 전송을 지연시키는 형태의 공격 발생 시 Caching 또는 L7 스위치는 웹 서버의 부하를 훨씬 저감시킨다. Caching이 연동된 웹서버에서 Slow POST 공격을 테스트 한 결과는 아래와 같다.

client <-->Caching

Time	Source	Destination	Protocol	Length	Info
30	2011-192.168.100.72	114.111.58.181	TCP	74	41185 > 80 [SYN] Seq=0 win=5840 Len=0 MSS=1460 SACK_PERM=1 TSval=166445203 TSecr=850639603
77	2011-114.111.58.181	192.168.100.72	TCP	74	80 > 41185 [SYN, ACK] Seq=0 Ack=1 win=5792 Len=0 MSS=1460 SACK_PERM=1 TSval=850639603 TSecr=166445206
78	2011-192.168.100.72	114.111.58.181	TCP	66	41185 > 80 [ACK] Seq=1 Ack=1 win=5888 Len=0 TSval=166445206 TSecr=850639603
79	2011-192.168.100.72	114.111.58.181	HTTP	372	POST /kr/new_member/login.php HTTP/1.1
80	2011-114.111.58.181	192.168.100.72	TCP	66	80 > 41185 [ACK] Seq=1 Ack=307 win=6880 Len=0 TSval=850639604 TSecr=166445206
81	2011-192.168.100.72	114.111.58.181	HTTP	67	Continuation or non-HTTP traffic[Unreassembled Packet]
82	2011-114.111.58.181	192.168.100.72	TCP	66	80 > 41185 [ACK] Seq=1 Ack=308 win=6880 Len=0 TSval=850639605 TSecr=166445206
84	2011-192.168.100.72	114.111.58.181	HTTP	67	Continuation or non-HTTP traffic[Unreassembled Packet]
85	2011-114.111.58.181	192.168.100.72	TCP	66	80 > 41185 [ACK] Seq=1 Ack=309 win=6880 Len=0 TSval=850642106 TSecr=166446207
86	2011-192.168.100.72	114.111.58.181	HTTP	67	Continuation or non-HTTP traffic[Unreassembled Packet]
87	2011-114.111.58.181	192.168.100.72	TCP	66	80 > 41185 [ACK] Seq=1 Ack=310 win=6880 Len=0 TSval=850644609 TSecr=166447208
88	2011-192.168.100.72	114.111.58.181	HTTP	67	Continuation or non-HTTP traffic[Unreassembled Packet]
89	2011-114.111.58.181	192.168.100.72	TCP	66	80 > 41185 [ACK] Seq=1 Ack=311 win=6880 Len=0 TSval=850647111 TSecr=166448209
90	2011-192.168.100.72	114.111.58.181	HTTP	67	Continuation or non-HTTP traffic[Unreassembled Packet]
91	2011-114.111.58.181	192.168.100.72	TCP	66	80 > 41185 [ACK] Seq=1 Ack=312 win=6880 Len=0 TSval=850649614 TSecr=166449210
92	2011-192.168.100.72	114.111.58.181	HTTP	67	Continuation or non-HTTP traffic[Unreassembled Packet]
93	2011-114.111.58.181	192.168.100.72	TCP	66	80 > 41185 [ACK] Seq=1 Ack=313 win=6880 Len=0 TSval=850652116 TSecr=166450211

Caching <--> 웹서버

```

root@SKB_IN_SSOD_6:~$ tcpdump -nni eth0 host 211.61.51.201 -s 1518 -w rudytest3-org.pcap
tcpdump: listening on eth0, link-type EN10MB (Ethernet), capture size 1518 bytes
0 packets captured
0 packets received by filter
0 packets dropped by kernel
root@SKB_IN_SSOD_6:~$

```

※최초 GET 요청 패킷은 제외

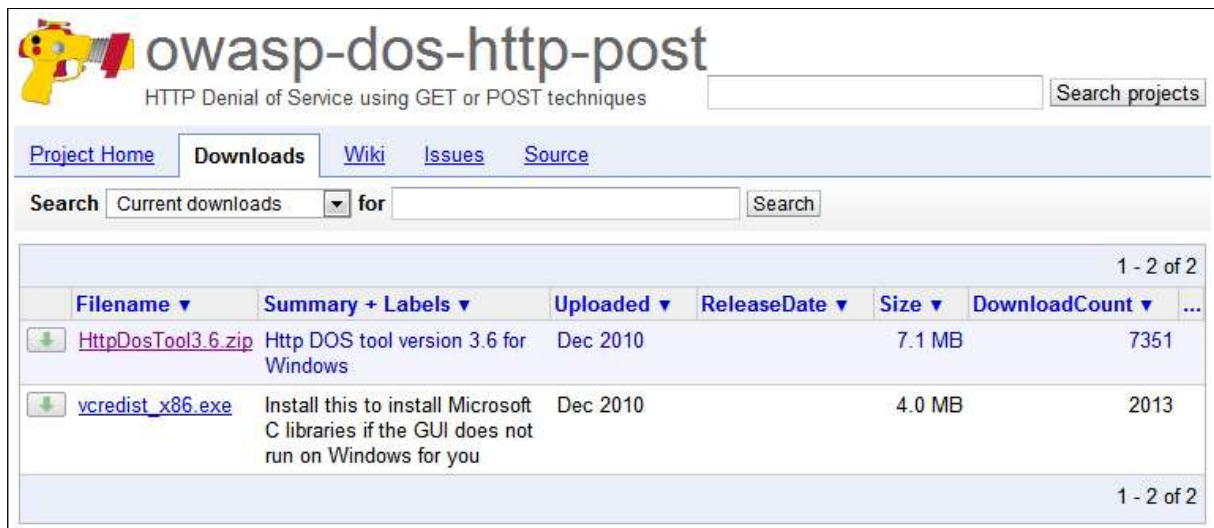
HTTPDoSTool 기반의 Slow POST 공격분석

2011. 8. 12(금) KISA 해킹대응팀

1. HTTP Dos Tool 도구 분석

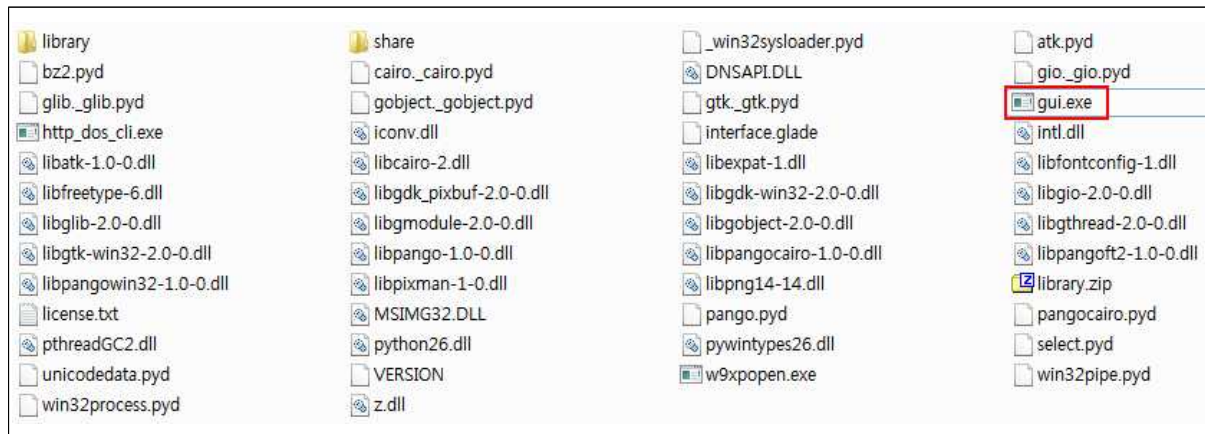
1.1. HTTP Dos Tool 소개

HTTP DoS Tool은 2010년 11월에 처음 공개 되었으며, v3.6까지 공개된 상태이다. 해당 공격툴은 <http://code.google.com/p/owasp-dos-http-post/downloads/list>에 접속해서 무료로 다운로드를 받을 수 있다.



< HTTP DoS Tool download 사이트 >

다운받은 압축파일을 열어보면 다음과 같은 *.dll 파일과 라이브러리로 구성되어 있음을 확인할 수 있다. HTTP Dos Tool의 실행파일은 gui.exe 파일이다.



< HTTP Dos Tool 설치 및 실행 파일 >

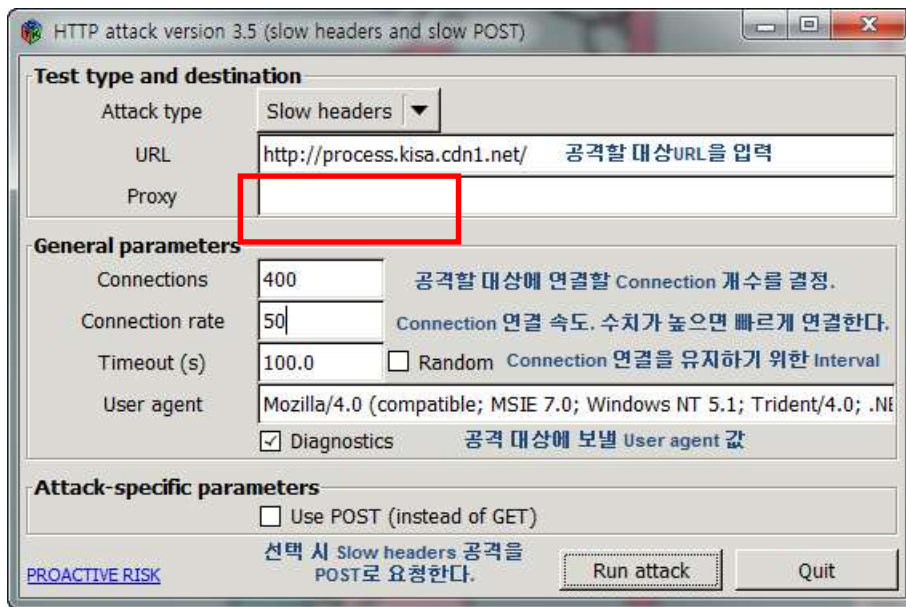
1.2. HTTP Dos Tool 공격의 실행과 형태

HTTP Dos Tool을 이용한 공격은 Slow Header와 Slow POST 2가지 형식(Type)으로 수행할 수 있는데, 2가지 공격 모두 웹 서버의 Connection을 유지하여 정상 사용자가 접속할 수 없도록 만드는 웹서버 자원 고갈형 공격이다.

가. 공격 옵션(Attack Type)

1) Slow Header Attack

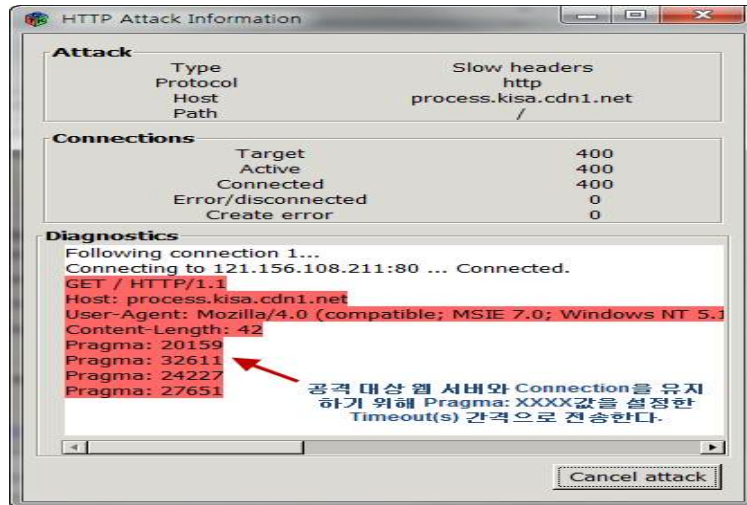
gui.exe를 실행하면 다음 그림과 같은 Slow Headers 공격 화면이 나타난다. 공격 화면에는 여러 선택 옵션을 포함하는데, 공격대상 URL과 사용할 Proxy 정보와 같은 기본적인 사항과 함께 공격할 대상에 대한 연결 수, 연결 빈도, 타임아웃, User-Agent 등의 정보를 설정할 수 있다.



< Slow Header Type 설정 화면 >

위 설정 화면에서 공격 대상 URL 정보를 입력하고 [Run attack] 실행 단추를 클릭하면 새 창이 실행되고 공격진행 현황을 눈으로 확인할 수 있다. 공격진행 현황은 설정 화면에서 'Diagnostics' 옵션을 설정한 경우에만 실행된다.

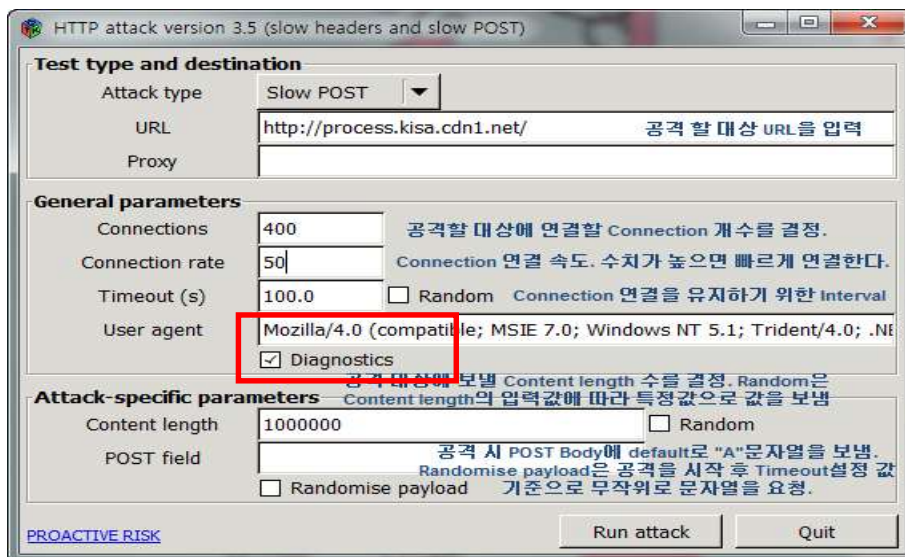
설정된 타임아웃(s) 간격마다 "Pragma:xxxx" 값이 전송됨을 확인할 수 있다.



< Slow Header 공격 진행 화면 >

2) Slow POST Attack

공격 설정 화면에서 상단의 Attack type을 Slow POST로 선택하면 다음 그림과 같이 추가 공격 옵션이 나타난다.

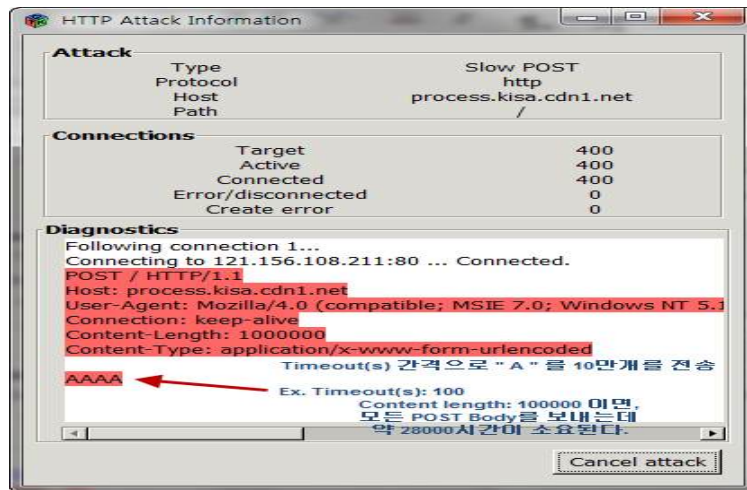


< Slow POST Type 설정 화면 >

위 그림과 같이 Slow POST 공격은 Slow Header 공격과 달리 공격 대상에 보낼 Content-Length 설정 기능과 POST Body에 포함할 문자열 설정 기능이 포함된다. 'POST field'에 아무 문자열을 입력하지 않으면 기본적으로 'A'라는 문자열을 사용하게 되며, 하단의 'Randomise payload'를 선택 시 임의의 문자열을 사용하게 된다.

필요한 공격 정보를 입력한 후 [Run attack] 실행 단추를 클릭하면 새 창이 실행되고 공격 로그를 눈으로 확인할 수 있다. Content-Length값만큼 'A' 문자열을 서버로 전송하게 되며, 모두 전송하는데 28,000시간이 필요하다. 즉, 서버는 28,000 시간동안

공격자의 세션을 계속 유지해야하므로 다른 정상 사용자의 접속이 불가능하게 된다.

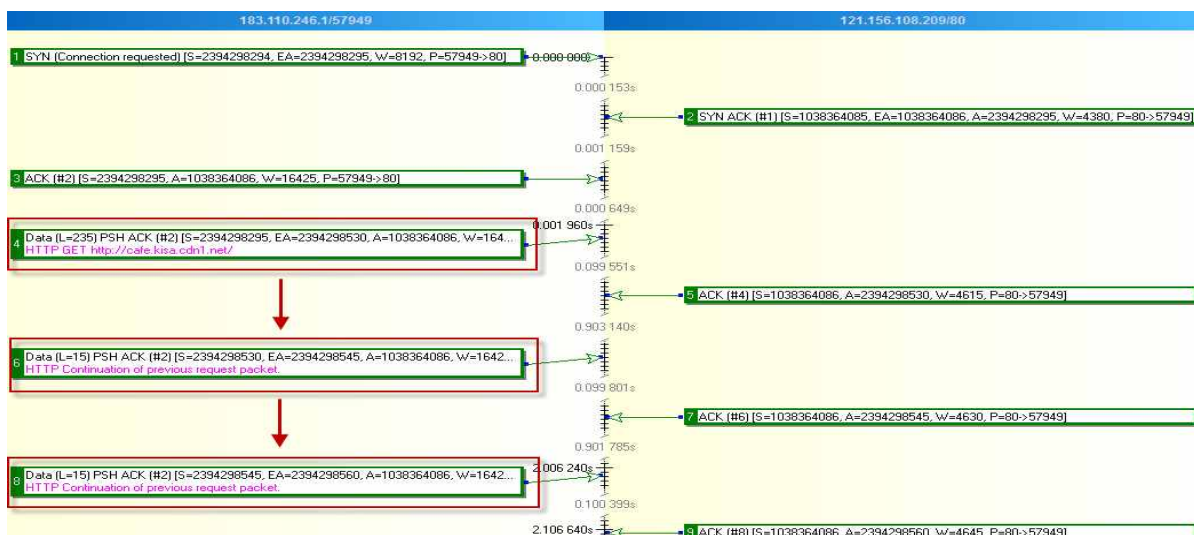


< Slow POST 공격 진행 화면 >

나. 공격 발생

1) Slow Header Attack

Slow Header 공격이 시작되면 먼저 Web Server와 Connection을 맺은 후 GET or POST 메소드를 포함하는 header를 Timeout(s)설정에 따라 "Pragma: XXXX" 문자열 형태의 Continuation Data를 전송하여, 해당 Connection을 지속적으로 Open 상태로 유지 한다.



< Slow Header 공격 실행시 Packet 흐름 >

위 그림을 보면 전송되는 Packet을 보면 Connection을 맺은 후 Request GET 메소드를 포함한 패킷의 header정보에 Pragma: {1 digits} 문자열을 포함한 정보를 주기적으로 전송함을 확인할 수 있다.

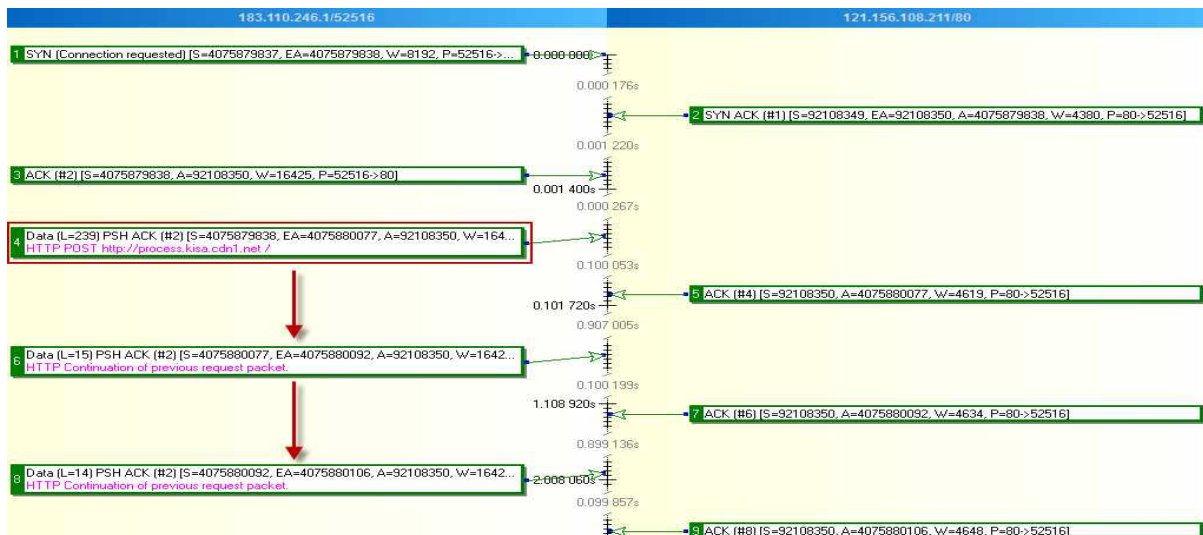
< Slow Header 공격 실행시 Packet 내용 >

아래와 같이 Continuation Data에는 “Pragma: xxxxx”라는 값이 삽입되어 있다.

< Slow header 공격 실행시 전송되는 Data >

2) Slow POST Attack

Slow POST Attack이 시작되면 먼저 Web Server와 Connection을 맺은 후 Content-length의 설정 값을 포함하여 POST 요청을 하게 된다. 요청 후 Timeout(s) 설정에 따라 POST field에 입력된 값이나 별도의 입력을 하지 않을 경우 “A” 문자열로 주기적으로 Request Body에 필요한 Data를 보내며, 해당 Content-length를 전부 전송할 때까지 Connection을 유지 한다.



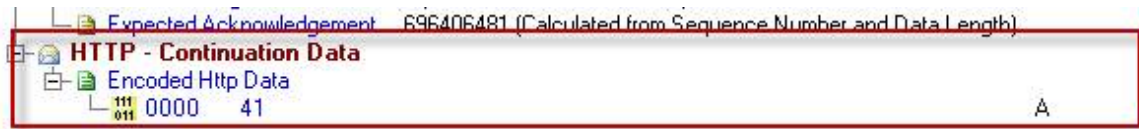
< Slow POST 공격 실행시 Packet 흐름 >

전송되는 Packet을 보면 Connection을 Request Body 메소드를 포함한 패킷의 header정보에 Pragma: {1 digits} 문자열을 포함한 정보를 주기적으로 전송 한다.



< Slow POST 공격 실행시 Packet 내용 >

아래와 같이 Continuation Data에 'A'라는 값이 삽입되어 있음을 확인할 수 있다.



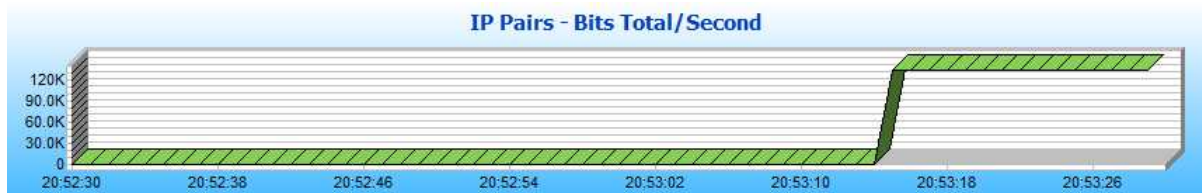
< Slow POST 공격 실행시 전송되는 Data >

3) HTTP DoS Tool 도구를 이용한 공격량

HTTP DoS Tool 도구를 이용하여 Slow DDoS 공격을 최대로 설정한 경우 발생 가능한 트래픽 상승량은 다음과 같다. 최대 연결수를 40,000으로, 연결빈도를 10,000ms, 타임아웃을 1초

설정값	BPS	PPS
Connections: 40,000	5Mbps	9Kpps
Connection rate: 10,000		
Timeout(s): 1		

< HTTP DoS Tool을 이용한 최대트래픽 발생형태 >



< HTTPDoSTool을 이용한 최대발생 트래픽 규모 >

2. HTTP DoS Tool을 이용한 공격대응 방안

HTTP Dos Tool은 Slow Header / POST DDoS 공격을 발생시킬 수 있는 도구의 한 종류에 불과하다. 그러므로 HTTP DoS Tool이 마치 Slow Header / POST 공격의 전부인 것으로 생각해서는 안된다.

응용계층의 공격은 어디까지나 공격 패킷의 형태를 다양화 할 수 있으므로 도구에서 생성하는 패킷의 형태에 집중해서는 안되며, 지속적인 테스트를 통해 웹서버의 장애 지점을 정확히 파악하고 대안을 세우는 데에 집중해야 한다는 점을 먼저 밝힌다.

2.1. 접속 임계치 설정을 통한 차단

특정한 발신지 IP에서 연결할 수 있는 동시 접속수(Concurrent Connection)에 대한 최대값을 설정함으로써 대응이 가능하다. 이 방법은 한 개의 IP에서 대량의 연결을 시도하는 공격을 차단하기에 적절하다. 유닉스/리눅스 계열의 운영체제를 운영한다면 운영체제의 방화벽 설정 도구인 iptables 명령어를 이용하여 차단이 가능하며 예제는 아래와 같다.

```
iptables -A INPUT -p tcp --dport 80 -m connlimit --connlimit-above 30 -j DROP
# 30개 이상의 concurrent connection에 대한 차단
```

2.2. Connection Timeout과 Keepalivetimeout 설정을 통한 차단

Connection Timeout에 설정된 시간동안 Client와 웹서버 사이에 데이터 신호의 이동이 전혀 없을 경우 웹서버는 연결된 Connection을 종료한다. Slow POST 공격 틀은 오랫동안 Connection을 유지하기 위해 데이터 전송 시 일정한 term을 갖게 된다. 배포된 HTTPDoSTool도 주기를 설정하는 옵션이 포함되어 있어 일정 구간의 패킷을 분석하여 현재 발생하는 DDoS 공격의 패턴을 파악하여 Timeout을 설정하는 방법을 활용할 수 있다.

다음 그림은 http.conf에서 5초 동안 연속된 패킷이 오지 않을 경우 tcp 연결을 closed하도록 설정하는 timeout 설정 예를 보여준다.

```
#
# Timeout: The number of seconds before receives and sends time out.
#
Timeout 5
```

< httpd.conf 의 timeout 설정 예 >

또한, 웹서버에서 keepalive 기능을 사용하는 경우에는 Keepalivetimeout을 사용하여 Slow POST 공격에 대응 할 수 있다. 다만, 공격자는 방어 정책 우회를 위해 Content-Length와 실제 전송하는 데이터의 크기를 조정하고 데이터 전송 term을 짧게 가져가면서도 Connection을 오랫동안 유지할 수 있으므로 이 대응 방안은 일정한 한계가 있다. 또한 네트워크 환경에 따라 작은 값의 timeout은 정상 사용자의 서비스에도 영향을 미칠 수 있다.

2.3. RequestReadTimeout(mod_reqtimeout Module) 설정을 통한 차단

Apache 2.2.15 버전과 그 이후 버전에서는 클라이언트의 요청에 대한 더욱 세부적인 제한을 줄 수 있는 RequestReadTimeout이라는 지시자를 제공한다.

이 지시자는 클라이언트의 요청(header and body)이 지정된 시간 내에 완료되지 않을 경우 오류 코드를 클라이언트에게 전송하여 Slow Attack을 차단하게 된다.

지시자를 사용하는 예제는 아래와 같다.

```

RequestReadTimeout header=5 body=8
# header=5
# HTTP Header Request가 5초 이내에 완료되지 않으면, FIN 패킷을 보내 연결을 해제하며, 이 때 Apache Server는 408 Response Code로 응답 한다. Timeout이 지난 후 Connection을 종료한다.
# body=8
# POST 요청 이후 8초 동안 데이터가 오지 않을 시 FIN을 보내 연결 해제를 요청하고 RST로 Connection을 종료한다. 일반적인 POST Request 시 Apache Server는 3-Way Handshake 이후 POST Request의 Content-Length 값만큼의 데이터가 수신될 때까지 Connection을 Open 상태로 수신을 기다린다.

```

< RequestReadTimeout 지시자의 사용 예 >

```

RequestReadTimeout header=10 body=30
# Client의 요청 header 전송완료는 10초 이내 , 요청 body 전송 완료는 30초 이내로 제한

RequestReadTimeout body=10,MinRate=1000
# client의 요청 body 전송완료는 10초 이내로 제한, 단 client가 data를 전송할 경우 1,000bytes 전송 시점마다 1초씩 증가

```

< RequestReadTimeout 지시자 활용 예 >

DRDoS(Distributed Reflection DoS) 공격분석

2012. 1. 20(금) KISA 해킹대응팀

□ DRDoS(Distributed Reflection Denial of Service)

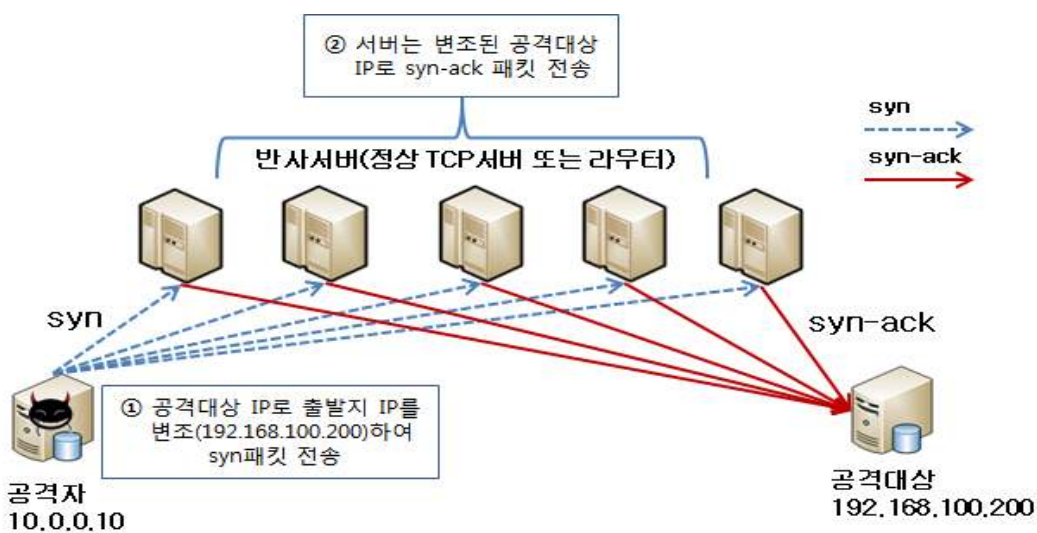
- 공격원리) TCP 3중 연결(3way-handshake)를 이용하는 DDoS 공격으로 공격자는 출발지 IP를 공격대상의 IP로 위조하여 syn 패킷을 다수의 반사서버로 전송하여 공격대상이 이 장비들이 응답하는 syn-ack 패킷을 받아 서비스가 거부 상태가 됨

※ 반사서버(reflection server): DRDoS에 이용되는 라우터 또는 TCP 서버

※ icmp프로토콜의 echo request와 response를 이용하여 동일한 형태의 공격도 가능함

- 공격방식) ①공격자는 송신지 IP를 공격대상의 IP로 위조한 후 대량의 syn패킷을 반사서버로 전송 ②syn패킷을 받은 장비는 위조된 출발지 IP로 정상적인 syn-ack 패킷을 전송함

※ icmp프로토콜을 이용할 경우 공격자는 syn패킷 대신 icmp echo request 패킷을 전송하고 반사서버는 syn-ack 패킷 대신 icmp echo reply 패킷을 응답함



< DRDoS 공격방식 >

- DRDoS 공격의 위협요소(일반 DDoS 공격과의 차이점)

- 공격근원지 파악의 어려움) 출발지 IP를 변조하고 공격트래픽이 수많은 반사서버를 경유하므로 공격의 근원지를 파악하여 역 추적하는 것이 거의 불가능함

- 좀비PC의 공격트래픽 효율 증가) DRDoS에 사용되는 반사서버는 syn-ack 패킷에 대한 응답이 없을 경우 재전송을 하기 때문에 공격자가 전송하는 syn패킷보다 몇 배 많은 syn-ack 패킷이 공격대상 서버에 전송됨, 따라서 일반 DDoS 공격에 비해 적은 좀비PC로 공격 트래픽량을 증가시킬 수 있음

□ 대응방법

- o 네트워크에서의 대응) DRDoS 공격은 출발지 IP 위조하는 공격이므로 IP주소가 위조된 패킷이 인터넷망으로 인입되지 않도록 ISP가 직접 차단 (Ingress Filtering)
 - 금일 네트워크 단(ISP)에서의 대응책 마련을 위해 KT 보안관제팀 실무자와 일부 내용을 협의한 결과
 - 라우터 및 스위치는 자신이 전달하는 패킷의 위변조 여부에 대해 판단할 수 없으므로
 - 네트워크 단에서 대응할 수 있는 유일한 방안은 발생한 공격을 분석하여 공격IP를 블랙리스트로 차단하는 방안이 유일함
- ※ ingress filtering이 국내 모든 ISP에 완벽하게 적용되어 있다면 공격 근원지에서 출발지 IP를 공격대상 IP로 변조하여 트래픽을 전송하는 것이 반사 서버로 도착하기 전에 차단되므로 DRDoS가 대부분 무력화됨
- o 반사 서버에서의 대응) icmp 프로토콜을 이용하는 DRDoS에 악의적으로 이용되지 않기 위해, icmp 프로토콜을 사용할 필요가 없는 시스템인 경우에는 스위치 또는 서버에서 해당 프로토콜 차단
- o 공격대상에서의 대응) icmp 프로토콜을 이용하는 DRDoS에 대응하기 위해 icmp 프로토콜을 사용할 필요가 없는 시스템인 경우에는 스위치 또는 서버에서 해당 프로토콜 차단

□ 추가 진행 할 사항

- o 네트워크 단에서의 DRDoS 대응 방안 마련을 위해 지속적으로 자료 조사 및 전문가 협의를 진행

Anonymous WebLoic 공격도구분석

2012. 2. 20(월) KISA 해킹대응팀

□ 개 요

- o 어나니머스(Anonymous)의 온라인 DDoS 공격페이지(WebLoic) 재개에 따른 DDoS 공격형태 및 대응방안 분석

※ 출처: Anonwiki (<http://www.anonwiki.org/p/webloic.html>)

□ 분석 내용

- o 어나니머스 온라인 DDoS 공격페이지(WebLoic) 메뉴 구성



- o DDoS 공격트래픽 생성 과정

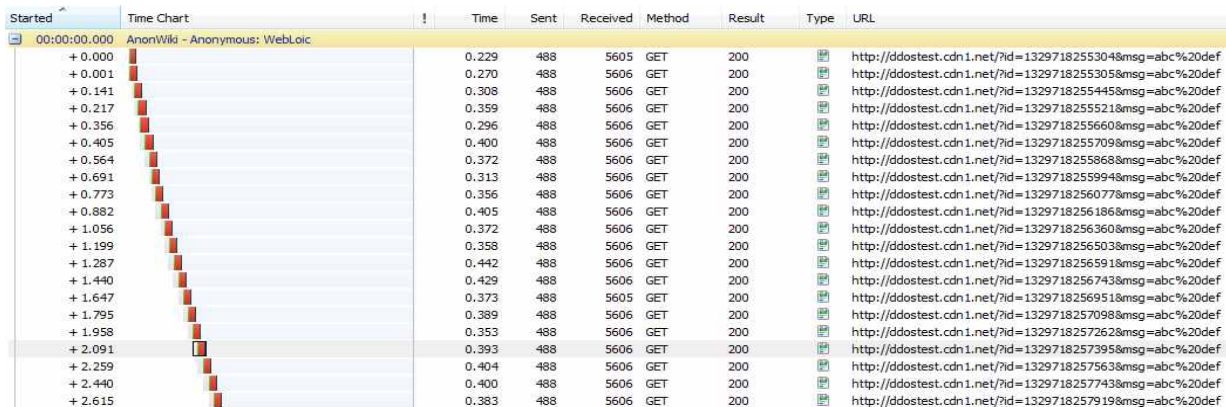
- 공격대상 URL, 쿼리값(Query String)으로 사용할 임의의 문자열, 초당 요청건수를 입력
- WebLoic은 입력받은 문자열을 이용하여, `http://{URL}/?id={Random Number}?msg={문자열}`과 같은 형식으로 Request URI를 생성

항목	설명
URL	- 공격할 URL * http://ddostest.cdn1.net/ 은 사이버대피소의 테스트서버 URL임 URL: http://ddostest.cdn1.net/
문자열	- Query string 문자열 * "abc edf"는 임의의 문자열임 Mensaje: abc edf
생성결과	http://ddostest.cdn1.net/?id=1329718255521&msg=abc%20def http://ddostest.cdn1.net/?id=1329718255660&msg=abc%20def http://ddostest.cdn1.net/?id=1329718255709&msg=abc%20def http://ddostest.cdn1.net/?id=1329718255868&msg=abc%20def http://ddostest.cdn1.net/?id=1329718255994&msg=abc%20def

- 생성한 Request URI를 초당 요청건수만큼 대상서버로 전송

o WebLoic에 의한 DDoS 공격 형태

- URL(http://ddostest.cdn1.net)로 초당 요청건수를 1,000으로 적용 시, 다음 그림과 같은 형식으로 초당 7~8건의 GET Flooding 발생



□ WebLoic 공격 정의

o WebLoic은 Random Number 값에 임의의 문자열을 지속적으로 발생시키는 Circle-CC 공격 형태를 보이지만, 해당 WebLoic에서 발생하는 공격트래픽에는 CC(Cache Control) 문자열을 포함되지 않음

o 즉, 해당 쿼리로 DB의 부하를 발생시켜 서비스 마비를 유도하는 응용계층(L7) 기반의 Get Flooding 공격임

※ 응용계층의 공격은 어디까지나 공격 패킷의 형태를 다양화 할 수 있으므로 도구에서에서 생성하는 패킷의 형태에 집중해서는 안되며, 지속적인 테스트를 통해 웹서버의 장애 지점을 정확히 파악하고 대안을 세우는 데에 집중해야 함

□ WebLoic 공격대응 방안

- 특정한 발신지 IP에서 연결할 수 있는 동시 접속수(Concurrent Connection)에 대한 최대값을 설정함으로써 대응이 가능
 - 특정 IP(183.110.246.1(대피소IP))에서 초당 100회 이상의 접속이 이루어지고 있으며, 이 접속수를 특정 값(예: 20회 이하)으로 제한하여 차단

No	IP	URL	Count	Time	TTL	M	Reason
101	183.110.246.1	ddostest.cdn1.net	145	2012-02-20 15:10:58	15	☞	WIP:ddostest.cdn1.net, Src IP & URL, http://ddostest.cdn1.net/ (Default Profile / ALL //,
100	183.110.246.1	ddostest.cdn1.net	100	2012-02-20 15:10:36	15	☞	WIP:ddostest.cdn1.net, Src IP & URL, http://ddostest.cdn1.net/ (Default Profile / ALL //,
99	183.110.246.1	ddostest.cdn1.net	198	2012-02-20 14:51:58	15	☞	WIP:ddostest.cdn1.net, Src IP & URL, http://ddostest.cdn1.net/ (Default Profile / ALL //,
98	183.110.246.1	ddostest.cdn1.net	100	2012-02-20 14:51:36	15	☞	WIP:ddostest.cdn1.net, Src IP & URL, http://ddostest.cdn1.net/ (Default Profile / ALL //,
97	183.110.246.1	ddostest.cdn1.net	110	2012-02-20 14:50:58	15	☞	WIP:ddostest.cdn1.net, Src IP & URL, http://ddostest.cdn1.net/ (Default Profile / ALL //,
96	183.110.246.1	ddostest.cdn1.net	100	2012-02-20 14:50:48	15	☞	WIP:ddostest.cdn1.net, Src IP & URL, http://ddostest.cdn1.net/ (Default Profile / ALL //,
95	183.110.246.1	ddostest.cdn1.net	261	2012-02-20 14:49:58	15	☞	WIP:ddostest.cdn1.net, Src IP & URL, http://ddostest.cdn1.net/ (Default Profile / ALL //,
94	183.110.246.1	ddostest.cdn1.net	100	2012-02-20 14:49:21	15	☞	WIP:ddostest.cdn1.net, Src IP & URL, http://ddostest.cdn1.net/ (Default Profile / ALL //,
93	183.110.246.1	ddostest.cdn1.net	434	2012-02-20 14:48:58	15	☞	WIP:ddostest.cdn1.net, Src IP & URL, http://ddostest.cdn1.net/ (Default Profile / ALL //,

< 특정 IP에 대한 연결 최대값 설정(최대값=100)에 의한 차단 예 >

- 이 방법은 특정 IP에서 대량의 연결을 시도하는 공격차단에 적절
 - 유닉스/리눅스 계열의 운영체제를 운영한다면 운영체제의 방화벽 설정 도구인 iptables 명령어를 이용하여 차단이 가능
 - 예제는 아래와 같음

```
iptables -A INPUT -p tcp --dport 80 -m connlimit --connlimit-above 30 -j DROP
# 30개 이상의 concurrent connection에 대한 차단
```

< 유닉스/리눅스 계열에서의 IPTABLES를 이용한 차단 방안 >

Hash DoS 공격유형 및 도구분석

2012. 5. 15(화) KISA 해킹대응팀

□ HashDoS란?

- 클라이언트에서 전달되는 각종 파라미터값을 관리하는 해시테이블의 인덱스 정보가 중복되도록 유도하여 기저장된 정보 조회시 많은 CPU 자원을 소모하도록 유도하는 공격
 - ※ 웹서버에서는 HTTP Request 요청 시 GET , POST 방식으로 전송되는 변수를 hash 구조로 관리(변수 값 접근을 쉽고 빠르게 하기 위해, hash 구조 사용)
- 많은 수의 매개변수를 전달하면 매개변수를 저장하는 해시테이블에서 해시 충돌이 발생하여 해시테이블에 접근하는 시간이 급속도로 증가
 - ※ POST 메시지는 전달되는 파라미터의 길이와 개수에 제한을 두지 않음
- HashDoS는 해시테이블을 이용하는 웹서버를 대상으로 한 공격임

□ 해시, 해시테이블, 해시충돌

- 해시(Hash)란?
 - 데이터를 저장하고 찾기를 하는데 사용되는 자료 구조의 한 종류로 찾고자 하는 문자열을 특정한 함수로 처리하여 얻은 값으로 데이터의 위치를 찾는 방법임
 - 데이터를 찾는 속도가 데이터의 개수의 영향을 거의 받지 않는 특성을 지니고 있어 효율적이고 빠르게 데이터의 위치를 찾을 수 있음
- 해시테이블(Hash Table)은 해시함수의 연산에 의해 구해진 위치에 각 정보를 한 개 이상 보관할 수 있도록 구성된 기억 공간



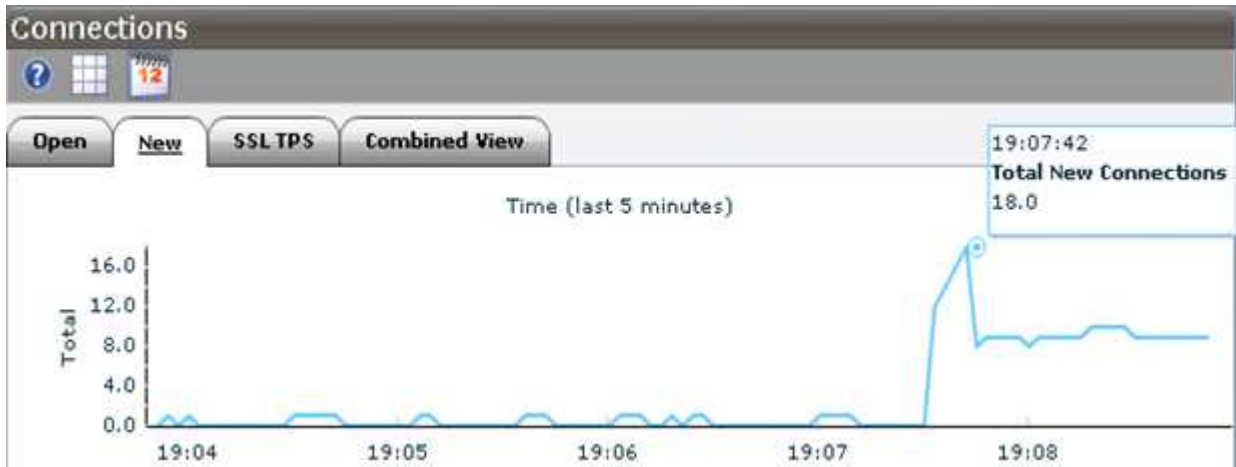
- 해시 충돌(Hash Collision)은 해시 함수가 서로 다른 두 개의 입력 값에 대해 동일한 출력 값을 내는 상황을 의미
 - 대부분의 해시 함수는 상당히 긴 입력 값으로부터 고정된 범위의 출력 값을 생성하므로 해시 출력 값의 범위보다 훨씬 더 큰 범위 값을 받는 경우 충돌이 발생
 - 해시충돌을 일으키는 특정 매개변수를 가진 POST 메시지가 서버로 전달하는 경우, 웹서버는 충돌되는 해시값을 비교해야 하기 때문에 CPU 자원이 고갈됨

□ HashDoS 공격도구 분석

- HashDoS 공격도구는 다음 그림과 같이 명령어셸 기반의 UI를 가짐



- HashDoS 공격도구는 서버와 초당 평균 10개 정도를 연결하여 약 500M 규모의 공격트래픽을 전달



< HashDoS 툴을 이용한 DoS 공격시 서버 연결 정보 >



< HashDoS 툴을 이용한 공격용량 >

o HashDoS 공격도구를 이용하여 공격시 서버의 CPU 사용량이 100% 도달

```
top - 15:44:17 up 5 days, 22:37, 1 user, load average: 0.16, 0.06, 0.02
Tasks: 136 total, 1 running, 134 sleeping, 0 stopped, 1 zombie
Cpu0 : 1.0%us, 0.0%sy, 0.0%ni, 99.0%id, 0.0%wa, 0.0%hi, 0.0%si, 0.0%st
Cpu1 : 17.3%us, 2.0%sy, 0.0%ni, 80.6%id, 0.0%wa, 0.0%hi, 0.0%si, 0.0%st
Mem: 4151488k total, 1197700k used, 2953788k free, 494040k buffers
Swap: 6289408k total, 0k used, 6289408k free, 445448k cached
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
7702	apache	15	0	26212	10m	3512	S	16.9	0.3	0:03.57	httpd
2808	mysql	18	0	160m	29m	4696	S	2.0	0.7	3:31.16	mysqld
1	root	15	0	2160	680	584	S	0.0	0.0	0:00.39	init
2	root	RT	-5	0	0	0	S	0.0	0.0	0:00.01	migration/0
3	root	34	19	0	0	0	S	0.0	0.0	0:00.00	ksoftirqd/0
4	root	RT	-5	0	0	0	S	0.0	0.0	0:00.00	watchdog/0
5	root	RT	-5	0	0	0	S	0.0	0.0	0:00.02	migration/1
6	root	34	19	0	0	0	S	0.0	0.0	0:00.00	ksoftirqd/1
7	root	RT	-5	0	0	0	S	0.0	0.0	0:00.00	watchdog/1
8	root	10	-5	0	0	0	S	0.0	0.0	0:00.00	events/0
9	root	10	-5	0	0	0	S	0.0	0.0	0:00.00	events/1
10	root	10	-5	0	0	0	S	0.0	0.0	0:00.00	khelper

< 정상적인 상태 >

```
top - 13:39:07 up 5 days, 20:32, 1 user, load average: 169.92, 101.24, 46.85
Tasks: 372 total, 154 running, 216 sleeping, 0 stopped, 2 zombie
Cpu0 : 99.0%us, 1.0%sy, 0.0%ni, 0.0%id, 0.0%wa, 0.0%hi, 0.0%si, 0.0%st
Cpu1 : 99.0%us, 1.0%sy, 0.0%ni, 0.0%id, 0.0%wa, 0.0%hi, 0.0%si, 0.0%st
Mem: 4151488k total, 2958216k used, 1193272k free, 494032k buffers
Swap: 6289408k total, 0k used, 6289408k free, 443928k cached
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
6936	apache	18	0	29892	13m	3368	R	19.8	0.3	0:03.42	httpd
7648	apache	18	0	30308	13m	3108	R	19.8	0.3	0:00.84	httpd
7661	apache	18	0	30268	13m	3304	R	19.8	0.3	0:01.39	httpd
7394	apache	18	0	30824	13m	3108	R	19.8	0.3	0:01.40	httpd
6346	apache	18	0	29896	13m	3368	R	16.8	0.3	0:05.26	httpd
7628	apache	18	0	29144	12m	2900	R	15.8	0.3	0:00.24	httpd
7098	apache	18	0	30408	14m	3368	R	13.8	0.3	0:01.00	httpd
7701	apache	18	0	30656	13m	2952	R	10.9	0.3	0:00.40	httpd
6334	apache	18	0	29896	13m	3368	R	9.9	0.3	0:05.62	httpd
6710	apache	18	0	30804	14m	3368	R	9.9	0.3	0:06.60	httpd
6759	apache	18	0	26468	10m	3324	S	8.9	0.3	0:03.16	httpd
6778	apache	18	0	26468	10m	3368	S	7.9	0.3	0:05.47	httpd
7709	apache	18	0	30656	13m	2952	R	7.9	0.3	0:00.44	httpd
6754	apache	18	0	28516	11m	3368	R	6.9	0.3	0:04.99	httpd
6750	apache	18	0	26468	10m	3368	S	4.9	0.3	0:09.86	httpd
7635	apache	18	0	29896	13m	3356	R	4.9	0.3	0:05.52	httpd
6404	apache	18	0	30924	14m	3368	R	4.0	0.4	0:07.43	httpd
6742	apache	18	0	30824	14m	3304	R	4.0	0.3	0:01.95	httpd
6778	apache	18	0	30824	14m	3368	R	4.0	0.3	0:05.39	httpd
7083	apache	18	0	30920	14m	3324	R	4.0	0.4	0:03.62	httpd
7596	apache	18	0	29144	12m	2900	R	4.0	0.3	0:00.23	httpd
7635	apache	18	0	29656	12m	2904	R	4.0	0.3	0:00.38	httpd
6721	apache	18	0	30924	14m	3368	R	2.0	0.4	0:02.51	httpd
6940	apache	18	0	26468	10m	3108	S	2.0	0.3	0:01.01	httpd
7422	apache	18	0	29568	13m	3324	R	2.0	0.3	0:00.22	httpd
7492	apache	18	0	26212	9.9m	3108	S	2.0	0.2	0:00.52	httpd
7650	apache	18	0	29796	13m	3108	R	2.0	0.3	0:00.72	httpd

< HashDoS를 이용한 공격시 상태 >

o HashDoS 공격도구에서 생성하는 공격트래픽 형태

- 서버와 연결을 맺고 POST 메시지에 파라미터값을 '&'를 이용하여 10Mbyte 길이의 스트링 데이터를 전송

1615	DTE	Link 1	- 61.110.235.194	1518	HTTP Continuation	TCP ACK [1081 -> 80]	IP [61.110.235.194 -> 121.156.108.253]	ETHER	4/17/2012	189.56m	55.360.540s	54.447283
1616	DTE	Link 1	- 61.110.235.194	1518	HTTP Continuation	TCP ACK [1081 -> 80]	IP [61.110.235.194 -> 121.156.108.253]	ETHER	4/17/2012	189.56m	55.360.573s	54.447316
1617	DTE	Link 1	- 61.110.235.194	1518	HTTP Continuation	TCP ACK [1081 -> 80]	IP [61.110.235.194 -> 121.156.108.253]	ETHER	4/17/2012	189.56m	55.360.574s	54.447317
1618	DTE	Link 1	- 61.110.235.194	1518	HTTP POST	TCP ACK [1107 -> 80]	IP [61.110.235.194 -> 121.156.108.253]	ETHER	4/17/2012	189.56m	55.360.616s	54.447358
1619	DTE	Link 1	- 61.110.235.194	1518	HTTP Continuation	TCP ACK [1107 -> 80]	IP [61.110.235.194 -> 121.156.108.253]	ETHER	4/17/2012	189.56m	55.360.680s	54.447423
1620	DTE	Link 1	- 61.110.235.194	1518	HTTP Continuation	TCP ACK [1069 -> 80]	IP [61.110.235.194 -> 121.156.108.253]	ETHER	4/17/2012	189.56m	55.360.691s	54.447424
1621	DTE	Link 1	- 61.110.235.194	1518	HTTP Continuation	TCP ACK [1107 -> 80]	IP [61.110.235.194 -> 121.156.108.253]	ETHER	4/17/2012	189.56m	55.360.682s	54.447425
1622	DCE	Link 1	- 121.156.108.253	70	TCP ACK [80 -> 1081]	IP [121.156.108.253 -> 61.110.235.194]	ETHERTYPE	4/17/2012	189.56m	55.360.697s	54.447440	
1623	DCE	Link 1	- 121.156.108.253	64	TCP ACK [80 -> 1081]	IP [121.156.108.253 -> 61.110.235.194]	ETHERTYPE	4/17/2012	189.56m	55.360.701s	54.447444	
1624	DTE	Link 1	- 61.110.235.194	1518	HTTP Continuation	TCP ACK [1089 -> 80]	IP [61.110.235.194 -> 121.156.108.253]	ETHER	4/17/2012	189.56m	55.360.808s	54.447551
1625	DCE	Link 1	- 121.156.108.253	64	TCP ACK [80 -> 1107]	IP [121.156.108.253 -> 61.110.235.194]	ETHERTYPE	4/17/2012	189.56m	55.360.808s	54.447551	
1626	DTE	Link 1	- 61.110.235.194	1518	HTTP Continuation	TCP ACK [1089 -> 80]	IP [61.110.235.194 -> 121.156.108.253]	ETHER	4/17/2012	189.56m	55.360.841s	54.447584
1627	DTE	Link 1	- 61.110.235.194	1518	HTTP Continuation	TCP ACK [1089 -> 80]	IP [61.110.235.194 -> 121.156.108.253]	ETHER	4/17/2012	189.56m	55.360.906s	54.447643
1628	DTE	Link 1	- 61.110.235.194	1518	HTTP Continuation	TCP ACK [1089 -> 80]	IP [61.110.235.194 -> 121.156.108.253]	ETHER	4/17/2012	189.56m	55.360.907s	54.447650
1630	DTE	Link 1	- 61.110.235.194	1518	HTTP Continuation	TCP ACK [1089 -> 80]	IP [61.110.235.194 -> 121.156.108.253]	ETHER	4/17/2012	189.56m	55.360.909s	54.447652
1631	DTE	Link 1	- 61.110.235.194	1518	HTTP Continuation	TCP ACK [1089 -> 80]	IP [61.110.235.194 -> 121.156.108.253]	ETHER	4/17/2012	189.56m	55.360.911s	54.447654
1632	DCE	Link 1	- 121.156.108.253	64	TCP ACK [80 -> 1089]	IP [121.156.108.253 -> 61.110.235.194]	ETHERTYPE	4/17/2012	189.56m	55.360.931s	54.447674	

< HashDoS 도구를 이용한 공격패킷 >

< POST 메시지에 '&'로 구분된 내용량 파라미터를 포함 >

□ HashDoS 공격대응 방안

- o Tomcat, PHP, Ruby 등 최신 버전에서는 HTTP Post Parameter 개수에 제한을 둘 수 있는 기능을 추가하였으므로 개수 제한 적용을 위해 최신버전으로 업데이트
- o 톰캣(Tomcat)에서의 HashDoS 차단 방안

- (방안 1 : 파라미터 개수 제한) TOMCAT_HOME/conf/server.xml의 Connector 부분을 다음과 같이 설정

```
<Connector port="8009" protocol="AJP/1.3" maxParameterCount="xxx" .../>
```

※ 적용가능버전: Tomcat 5.5.35, 6.0.35, 7.0.23

- (방안 2 : POST 메시지 크기 제한) 사이즈를 제한하는 것이 서비스에 문제가 없는 경우 적용하며, TOMCAT_HOME/conf/server.xml의 Connector 부분을 다음과 같이 설정

```
<Connector port="8009" protocol="AJP/1.3" maxPostSize="xxx" .../>
```

o PHP에서의 HashDoS 차단 방안

- PHP 5.4.0 RC4로 업데이트 한 후, php.ini 파일에서 'max_input_var'에서 최대 HTTP POST Parameter 개수를 설정
- PHP 5.4.0 RC4 이하인 경우, PHP Source에서 소스변경부분을 수동으로 설정하고 <http://svn.php.net/viewc?view=revision&revision=321003>에서 수정된 소스를 다운받아 재빌드하여 적용

※ 소스 : PHP_5_4/main/main.c, PHP_5_4/main/php_globals.h, PHP_5_4/main/php_variables.c

Hulk DoS 공격유형 및 도구분석

2012. 7. 31(화) KISA 해킹대응팀

□ HULK DoS란?

- HULK(Http Unbearable Load King) DoS는 웹서버의 가용량(웹서버로 접속할 수 있는 최대 클라이언트 수)을 모두 사용하도록 하여 정상적인 서비스가 불가능하도록 유도하는 GET Flooding 공격 유형으로,
 - 공격대상 웹사이트 주소(URL)를 지속적으로 변경하여 DDoS 차단정책을 우회한다는 특징을 가짐
- DDoS(Denial of Service, 분산서비스공격)는 대량의 트래픽을 유발하거나 비정상적인 접속을 발생시켜 웹사이트를 마비시키는 공격으로 구분됨
 - ※ 네트워크 대역폭을 잠식하여 트래픽의 흐름을 방해하거나 웹서버 자원에 부하를 유발함으로써 서비스 속도가 느려지게 하여 정상적인 웹서비스를 불가능하도록 유도
- 웹서버 자원에 부하를 유발하는 공격 기법 중 가장 많이 사용하는 방법은 특정 웹사이트 주소(URL)를 호출하도록 하는 GET Flooding 공격 기법임
 - ※ GET Flooding 공격이란 웹서버와 TCP Connection(3-way handshaking)을 연결한 후 짧은 시간 안에 다수의 동적 콘텐츠를 요청함으로써 웹 서비스의 부하를 유발하는 공격
- 웹서버 자원 부하유발 공격을 차단하기 위해서는 특정 웹사이트에 접근할 수 있는 회수를 제한하는 임계치(Threshold) 설정방법을 적용할 수 있음
 - ※ 예를 들어, 특정 URL의 임계치를 10으로 설정하면, 클라이언트마다 특정 URL에 동시에 10번 이상 접속이 불가능하게 되어 웹서버의 부하를 감소시키는 효과를 가져옴
 - ※ 임계치는 고정된 URL주소에 대해서만 설정이 가능함
- 만약, 특정 웹사이트 주소(URL)가 계속 변경된다면 이러한 임계치 설정기반의 방어는 불가능해질 수 있음
 - ※ 웹사이트 주소 뒤에 임의의 파라미터를 포함하도록 하면 DDoS 대응 장비는 임계치가 설정된 웹사이트 주소와 전혀 다른 주소로 인식하게 되어 차단하지 않음
- HULK DoS는 URL에 임의의 파라미터를 포함하도록 하여 URL 주소를 계속 변경하여 특정 URL에 대한 임계치 기반의 DDoS 차단을 우회하기 위한 공격 기법임
 - ※ HULK는 Imperva 사의 Barry Shteyman이라는 연구원이 자신의 블로그인 Nerd에 DDoS 연구나 교육을 위해 만든 도구이지만 해커에 의해 DDoS 공격도구로 악용되고 있음

□ HULK DoS 공격기법

- 클라이언트에서 웹서버로 Request URL을 전달하면 웹서버는 해당 URL에 해당하는 웹사이트를 클라이언트에게 전달함
- Request URL에는 아이디, 비밀번호와 같은 정보를 파라미터로 포함하여 전달할 수 있는데, 이때는 Request URL 뒤에 "?" 기호와 함께 임의의 문자열을 포함할 수 있음

· 파라미터 추가 형식 : ?이름1=값1&이름2=값2&....&이름n=값n
 (예) <http://search.n000.com/search.n000?where=nexsearch&query=get&fbm=1&ie=utf8>

- HULK DoS는 이러한 파라미터를 지속적으로 변경하여 웹서버로 전달

```
HTTP GET [redacted] net/?BDXCNW=ZJYWVA --- TCP PSH ACK [50847
HTTP GET [redacted] net/?NBTGABYC=UUBKNOT --- TCP PSH ACK [508
HTTP GET [redacted] net/?XTLAXHMIBS=DSLOZCLRL --- TCP PSH ACK
HTTP GET [redacted] net/?QWDFCZ=UCKQWJFP --- TCP PSH ACK [5085
HTTP GET [redacted] net/?DCXZEYQZYA=SVY --- TCP PSH ACK [50851
HTTP GET [redacted] net/?BJYAYHKDV=MTNSGWSYW --- TCP PSH ACK
```

※ 파라미터가 지속적으로 변경되는 것을 제외하고는 일반적으로 GET Flooding과 동일한 형태임

□ HULK DoS 공격도구 분석

- HULK DoS 공격도구는 python으로 작성되었으며 윈도우즈 환경에서는 Active python을 설치하여 아래 명령어로 간단히 실행할 수 있음

```
c:\> hulk.py http://test.com/
```

- HULK DoS의 소스파일과 실행화면


```

#-----
# HULK - HTTP Unbearable Load King
#
# this tool is a dos tool that is meant to put heavy load on HTTP servers in order to bring them
# to their knees by exhausting the resource pool, its is meant for research purposes only
# and any malicious usage of this tool is prohibited.
#
# author : Barry Shteiman , version 1.0
#-----
import urllib2
import sys
import threading
import random
import re

#global params
url=''
host=''
headers_useragents=[]
headers_referers=[]
request_counter=0
flag=0
safe=0

def inc_counter():
    global request_counter
    request_counter+=1

def set_flag(val):
    global flag
    flag=val

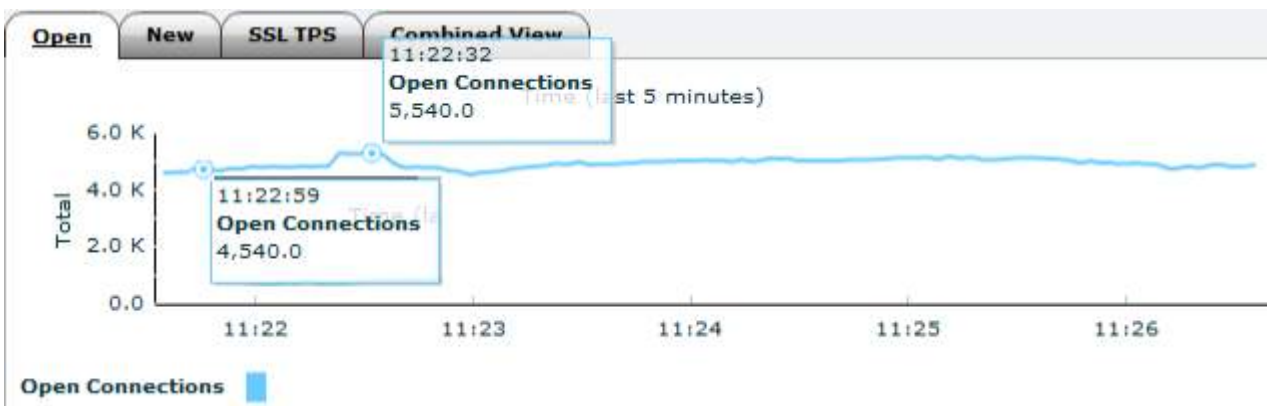
def set_safe():
    global safe
    safe=1

# generates a user agent array
def useragent_list():
    global headers_useragents
    headers_useragents.append('Mozilla/5.0 (X11; U; Linux x86_64; en-US; rv:1.9.1.3) Gecko/20090913 Firefox/3.5.3')
    headers_useragents.append('Mozilla/5.0 (Windows; U; Windows NT 6.1; en; rv:1.9.1.3) Gecko/20090824 Firefox/3.5.3 (.NET CLR 3.5.30721)')
    headers_useragents.append('Mozilla/5.0 (Windows; U; Windows NT 5.2; en-US; rv:1.9.1.3) Gecko/20090824 Firefox/3.5.3 (.NET CLR 3.5.30721)')
    headers_useragents.append('Mozilla/5.0 (Windows; U; Windows NT 6.1; en-US; rv:1.9.1.1) Gecko/20090718 Firefox/3.5.1')
    headers_useragents.append('Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US) AppleWebKit/532.1 (KHTML, like Gecko) Chrome/4.0.215.1 (Windows NT 5.1)')
    headers_useragents.append('Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; WOW64; Trident/4.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; .NET CLR 1.1.4324.2251; .NET CLR 1.0.3745.4245)')
    headers_useragents.append('Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.0; Trident/4.0; SLCC1; .NET CLR 2.0.50727; .NET CLR 3.0.30729; .NET CLR 3.5.30729; .NET CLR 1.1.4324.2251; .NET CLR 1.0.3745.4245)')
    headers_useragents.append('Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 5.2; Win64; x64; Trident/4.0)')
    headers_useragents.append('Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 5.1; Trident/4.0; SV1; .NET CLR 2.0.50727; InfoPath.2)')
    headers_useragents.append('Mozilla/5.0 (Windows; U; MSIE 7.0; Windows NT 6.0; en-US)')
    headers_useragents.append('Mozilla/4.0 (compatible; MSIE 6.1; Windows XP)')
    headers_useragents.append('Opera/9.80 (Windows NT 5.2; U; ru) Presto/2.5.22 Version/10.51')
    return(headers_useragents)

```



o HULK DoS 공격도구는 서버와 초당 평균 1,000개 정도를 연결하여 약 10M 규모의 공격트래픽을 전달 (아래 그림에서 연결수치 4,540 → 5,540)



< HULK DoS 툴을 이용한 DoS 공격시 서버 연결 정보 >

o HULK DoS 공격도구를 이용하여 공격시 서버의 CPU 사용량이 50% 도달

```
top - 11:23:21 up 76 days, 13:48, 3 users, load average: 7.28, 5.62, 5.35
Tasks: 156 total, 1 running, 155 sleeping, 0 stopped, 0 zombie
Cpu(s): 0.0%us, 0.1%sy, 0.0%ni, 99.9%id, 0.0%wa, 0.0%hi, 0.0%si, 0.0%st
Mem: 3115136k total, 2706924k used, 408212k free, 234629k buffers
Swap: 4096532k total, 108k used, 4096424k free, 2300904k cached

PID USER PR NI VIRT RES SHR S ZCPU ZMEM TIME+ COMMAND
17271 root 15 0 2532 1136 788 R 0.3 0.0 0:04.18 top
1 root 15 0 2160 644 556 S 0.0 0.0 0:00.95 init
2 root RT -5 0 0 0 S 0.0 0.0 0:17.79 migration/0
3 root 34 19 0 0 0 S 0.0 0.0 0:00.03 ksoftirq/0
4 root RT -5 0 0 0 S 0.0 0.0 0:00.00 watchdog/0
5 root RT -5 0 0 0 S 0.0 0.0 0:22.32 migration/1
6 root 34 19 0 0 0 S 0.0 0.0 0:00.08 ksoftirq/1
7 root RT -5 0 0 0 S 0.0 0.0 0:00.00 watchdog/1
8 root RT -5 0 0 0 S 0.0 0.0 0:18.60 migration/2
9 root 34 19 0 0 0 S 0.0 0.0 0:00.03 ksoftirq/2
10 root RT -5 0 0 0 S 0.0 0.0 0:00.00 watchdog/2
11 root RT -5 0 0 0 S 0.0 0.0 0:20.04 migration/3
12 root 37 19 0 0 0 S 0.0 0.0 0:00.03 ksoftirq/3
13 root RT -5 0 0 0 S 0.0 0.0 0:00.00 watchdog/3
14 root 10 -5 0 0 0 S 0.0 0.0 0:16.35 events/0
```

< 정상적인 상태 >

```
top - 11:22:18 up 76 days, 13:47, 3 users, load average: 9.67, 5.74, 5.38
Tasks: 163 total, 1 running, 162 sleeping, 0 stopped, 0 zombie
Cpu(s): 51.0%us, 7.8%sy, 0.0%ni, 35.9%id, 0.1%wa, 0.7%hi, 4.5%si, 0.0%st
Mem: 3115136k total, 2706298k used, 408848k free, 234452k buffers
Swap: 4096532k total, 108k used, 4096424k free, 2299268k cached

PID USER PR NI VIRT RES SHR S ZCPU ZMEM TIME+ COMMAND
17430 apache 15 0 26792 6212 2680 S 11.3 0.2 0:04.81 httpd
17432 apache 15 0 26792 6208 2676 S 11.3 0.2 0:04.32 httpd
23509 apache 15 0 26832 5420 1952 S 11.6 0.2 0:03.03 httpd
23564 apache 15 0 26792 5420 1952 S 11.6 0.2 0:00.86 httpd
17421 apache 15 0 26792 6188 2660 S 11.3 0.2 0:04.85 httpd
23507 apache 15 0 26832 5420 1952 S 11.3 0.2 0:03.24 httpd
23508 apache 15 0 26792 5412 1944 S 11.3 0.2 0:02.87 httpd
23508 apache 15 0 26832 5420 1952 S 11.3 0.2 0:00.95 httpd
23788 apache 15 0 26792 5420 1952 S 11.3 0.2 0:00.37 httpd
17431 apache 15 0 26792 6188 2664 S 10.9 0.2 0:04.51 httpd
23739 apache 15 0 26792 5412 1944 S 10.9 0.2 0:00.41 httpd
2689 psat 15 0 2064 498 340 S 6.3 0.0 5:17.54 pof
7949 root 18 0 3272 1664 1272 S 1.3 0.1 0:06.83 rotatelogs
2323 root 11 -4 13672 920 568 S 0.3 0.0 34:59.09 auditd
2816 mysql 15 0 5700 740 432 S 0.3 2.4 2:29.71 mysqld
3306 root 16 0 2060 656 576 S 0.3 0.0 0:07.33 haid-addon-stor
7942 root 18 0 26564 8716 5400 S 0.3 0.3 0:00.37 httpd
1 root 15 0 2160 644 556 S 0.0 0.0 0:00.95 init
2 root RT -5 0 0 0 S 0.0 0.0 0:17.79 migration/0
```

< HULK DoS를 이용한 공격시 상태 >

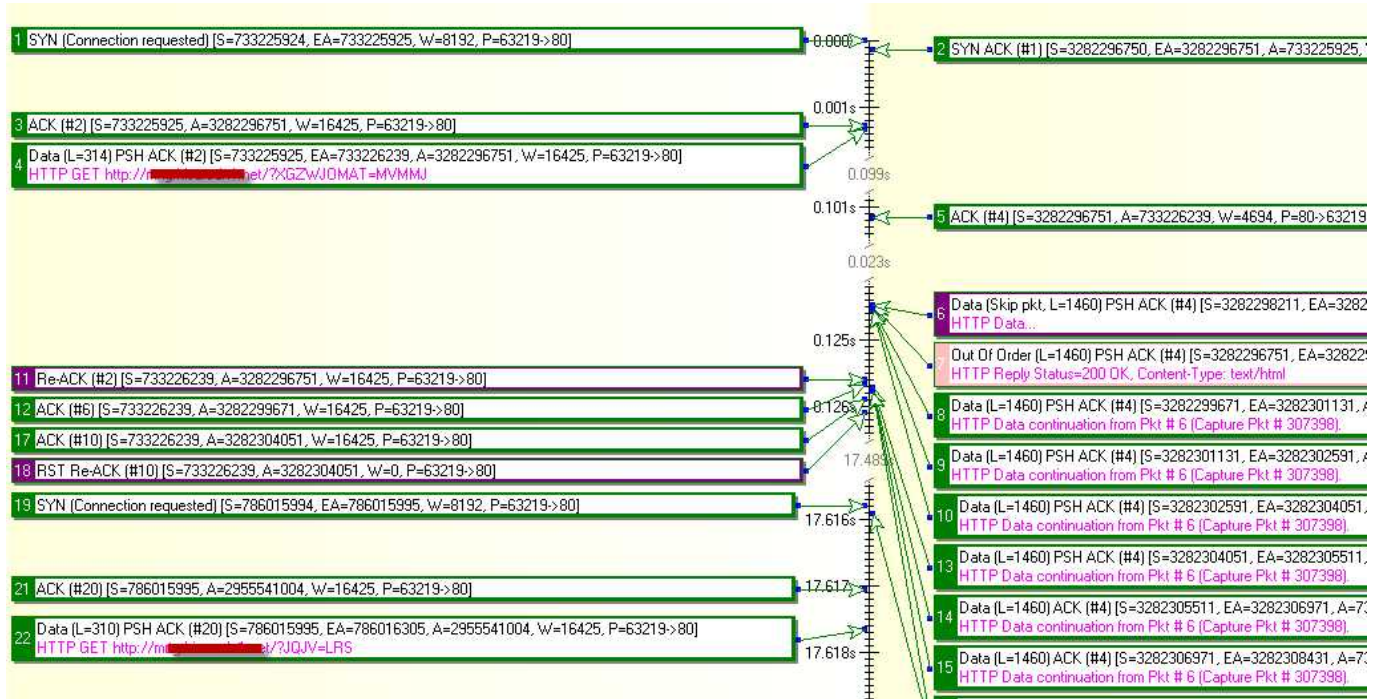
o HULK DoS 공격도구에서 생성하는 공격트래픽 형태

- 아래 그림과 같이 HOST 뒤에 “/?”를 입력 후 임의의 문자를 무작위로 입력하면, 웹서버는 파라미터 유효검사를 하게 되는데
- 파라미터가 유효하지 않더라도 “/” URL로 동작하여 클라이언트에게 200ok 응답을 하게 됨

Pkt	Source	Destination	Size	Summary
305366	DTE - Link 1 - 183.110.246.1	DCE - Link 1 - 121.156.108.240	372	HTTP GET http://n...net/?XGZWJOMAT=MVMMJ --- TCP PSH ACK [6321
305367	DTE - Link 1 - 183.110.246.1	DCE - Link 1 - 121.156.108.240	64	TCP ACK [63072 -> 80] --- IP [183.110.246.1 -> 121.156.108.240] --- ETHERTYPE
305368	DTE - Link 1 - 183.110.246.1	DCE - Link 1 - 121.156.108.240	64	TCP RST ACK [63072 -> 80] --- IP [183.110.246.1 -> 121.156.108.240] --- ETHERTYPE
305369	DCE - Link 1 - 121.156.108.240	DTE - Link 1 - 183.110.246.1	1518	HTTP Continuation --- TCP ACK [80 -> 63072] --- IP [121.156.108.240 -> 183.110.246.
305370	DCE - Link 1 - 121.156.108.240	DTE - Link 1 - 183.110.246.1	1518	HTTP Continuation --- TCP PSH ACK [80 -> 63072] --- IP [121.156.108.240 -> 183.110.


```
HTTP - Request
Request Line GET /?XGZWJOMAT=MVMMJ HTTP/1.1
Header Accept-Encoding: identity
Header Host: m...net
Header Keep-Alive: 112
Header User-Agent: Mozilla/4.0 (compatible; MSIE 6.1; Windows XP)
Header Accept-Charset: ISO-8859-1,utf-8;q=0.7,;q=0.7
Header Connection: close
Header Referer: http://www.usatoday.com/search/results?q=QIFAOT
Header Cache-Control: no-cache
```

< Request Header URL, User-Agent, Referer의 지속적 변화>



< 정상적인 GET 요청으로 동작함 >

□ HULK DoS 공격대응 방안

o (방안-1) 접속 임계치 설정을 통한 차단

- 특정한 발신지 IP에서 연결할 수 있는 동시 접속수(Concurrent Connection)에 대한 최대값을 설정하여 한 개의 IP에서 대량의 연결을 시도하는 공격을 차단
- 유닉스/리눅스 계열의 운영체제를 운영한다면 운영체제의 방화벽 설정 도구인 iptables 명령어를 이용하여 차단이 가능하며 예제는 아래와 같음

```
iptables -A INPUT -p tcp --dport 80 -m connlimit --connlimit-above 30 -j DROP
# 30개 이상의 concurrent connection에 대한 차단
```

o (방안-2) HTTP Request의 HOST 필드값에 대한 임계치 설정을 통한 차단

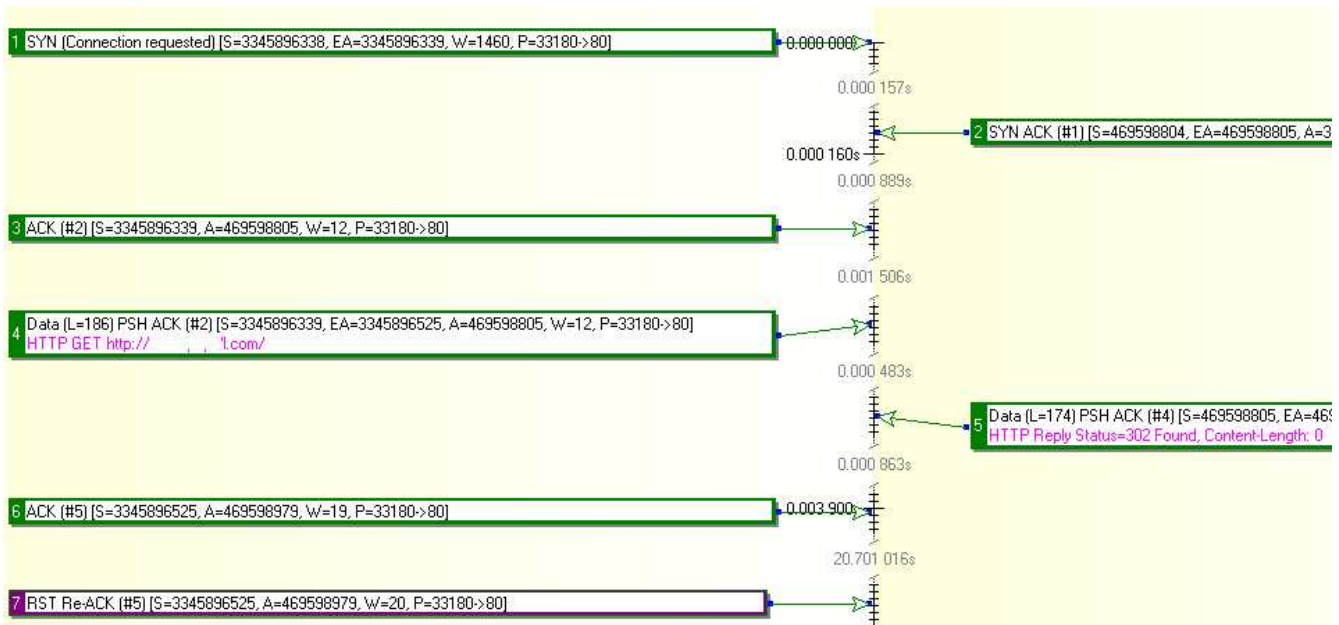
- HULK DoS는 URL을 지속적으로 변경하기 때문에 URL이 아닌 HTTP Request에 포함된 HOST 필드값을 카운트하여 임계치 이상인 경우 차단하도록 설정



< HTTP Request의 HOST 필드 정보 >

o (방안-3) 302-Redirect를 이용한 차단

- 대부분의 DDoS 공격 tool은 302-Redirect 요청에 대해 반응하지 않는 것이 특징이므로 여러 웹사이트 URL 중에 공격당하기 쉬운 웹사이트(예: “/”)에 대한 Redirect 처리를 통해,
- 자동화된 DDoS 공격 tool을 이용한 공격을 사전에 차단하여 웹서버의 부하를 감소시킬 수 있음



< 서버의 302-Redirect 요청시 클라이언트에서 Reset 이 발생함 >