ISBN : 000-00-000-0000-0

DDoS 공격 유형별 분석보고서

2015. 12.

KiSTi 한국과학기술정보연구원
Korea Institute of Science and Technology Information
www.kisti.re.kr

목 차

표 목차

그림 목차

Ⅰ. 개요

1 목적

□ 2009년 발생한 7·7 DDoS 공격으로 주요 정부기관, 은행 사이트 등에 피해가 발생했으며 이후에도 지속적으로 DDoS 공격 피해가 확인되고 있으며 최근에 급격히 확산된 봇넷(Botnet) 기반의 DDoS 공격 인프라의 확대와 기존의 DDoS 공격 장비를 우회할 수 있는 최신의 공격 기법이 악성코드에 포함되어 세심한 관심과 준비가 필요함

2 적용범위

□ 첨단연구망 내 운영 중인 보안관제 대상 시스템의 DDoS 공격 발생 시 첨단연구망 보안관제 상황실 운영요원에 의한 공격 대응 및 예방활동에 참고함

3 용어정의

□ “분산서비스거부공격” (이하 'DDoS 공격')이라 함은 인터넷 상에 분산되어 있는 다수의 악성코드 감염PC를 이용, 대량의 접속트래픽을 일시에 특정 사이트 또는 시스템에 전송하여 과부하를 유발시킴으로써 정상적인 정보시스템서비스를 할 수 없도록 하는 사이버공격을 말함

□ “악성코드” 라 함은 PC나 서버에 침투하여 피해를 입히는 소프트웨어를 가리키며 컴퓨터바이러스, 웜, 트로이목마, 스파이웨어 등을 포함함

□ “좀비PC” (Zombie PC)라 함은 악성코드에 감염되어 PC이용자도 모르게 DDoS공격 등 사이버공격에 악용되는 일반 이용자 컴퓨터를 말함

□ “봇넷” (Botnet)이라 함은 사이버공격자들에 의해 제어되는 명령제어서버(Command & Control 서버)와 좀비PC들의 집합 또는 네트워크를 말함)

Ⅱ. 유형별 DDoS 공격 및 대응방안

1 개요

□ DDoS 공격의 유형은 공격자의 위치에 따라 내부에서의 공격과 외부에서의 공격으로 나눌 수 있고, 공격대상의 형태에 따라, 특정 네트워크에서 허용하는 대역폭을 모두 소모시키는 방법, 공격대상(Victim)의 시스템 상에서 동작하는 Application 등 프로그래밍 오류에 대한 공격으로 서비스를 거부하게 하는 방법 등이 있음

【 표 Ⅱ.1. 】 DDoS 공격유형 분류

구분	대역폭 소진공격	서비스(어플리케이션) 마비공격
대표 공격유형	UDP/ICMP Flooding, SYN Flooding	HTTP GET Flooding
공격의 형태	○ 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)	○ HTTP Traffic Flooding - GET Flooding, GET with Cache-Control ○ HTTP Header/Option Spoofing - Slowris, Fragmented HTTP Header Attack 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	○ 회선 대역폭 고갈 ○ 동일 네트워크를 사용하는 모든 서비스에 대한 접속장애 발생	○ HTTP 서버과다접속(또는 서비스 부하)으로 인한 장애발생 ○ 공격대상 시스템만 피해

2 대역폭 소진공격

2.1 UDP/ICMP Traffic Flooding 공격

2.1.1 UDP/ICMP Flooding

가. 공격기법

□ 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

나. 대응방안

□ 일반적으로 UDP/ICMP Flooding 공격을 수행할 때 단일 좀비(Zombie)에서 발생시키는 패킷은 그 크기와 전송 간격 또한 다양하며 방화벽으로 모든 패킷이 몰리게 되어 단일 시간(보통 초 단위)에 대량의 패킷이 집중될 수밖에 없음

□ UDP/ICMP 방어 및 완화를 위해서는 네트워크 스위치에서의 ACL기반의 프로토콜 차단이나 방화벽의 임계치(Threshold) 설정 기능을 고려할 수 있음

- (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 extened 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 3.3.3.0/24 any
 seq 7 permit ip 4.4.4.0/24 any
 seq 8 permit icmp host 5.5.5.5 any
```

【 그림 Ⅱ.1. 】 ACL로 UDP/ICMP를 차단하는 설정 예

- **(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)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
  ※ PPS 제안 수치는 서비스에 지장을 주지 않는 범위 내에서 단계적으로 조정하여 설정할 수 있음

2.1.2 DNS Query Flooding

가. 공격기법

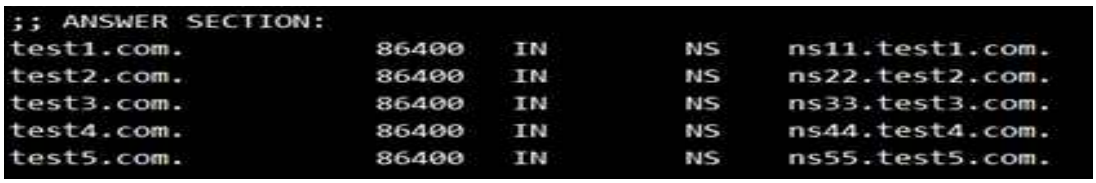
□ 공격자는 UDP 프로토콜 기반의 서비스를 제공하는 DNS에 대해 DNS 쿼리 데이터를 다량으로 서버에 전송하여 DNS의 정상적인 서비스를 방해하는 공격임

※ UDP/ICMP Flooding 공격 형태와 유사함

□ 공격 유형은 크게 1) DNS를 통한 대역폭(Bandwitdh) 공격, 2) 많은 Query를 발생하도록 하여 DNS의 응답을 하지 못하도록 하는 서버 자원 공격(어플리케이션 장애유발)으로 볼 수 있음

나. 대응방안

- **(방안 1) DNS 서버의 다중화를 통한 DNS 공격 트래픽 분산 처리**
  가능한 DNS 서버를 다중으로 구성하여 특정 서버로의 공격이 발생하더라도 다른 서버가 해당 요청에 대해 응답할 수 있도록 구성



【 그림 Ⅱ.2. 】 특정 DNS요청에 대해 다수의 DNS서버 등록

- **(방안 2) IPTABLE을 이용한 ACL 기반의 차단**
  대부분 DNS요청 패킷은 512Byte를 넘을 수 없기 때문에 처리가 가능한 대역폭에서 서비스를 하는 경우라면 iptables등을 이용해 공격 패킷을 차단

(예시) iptables -A INPUT -p udp —dport53 -m length —length 512:1500 -j DROP

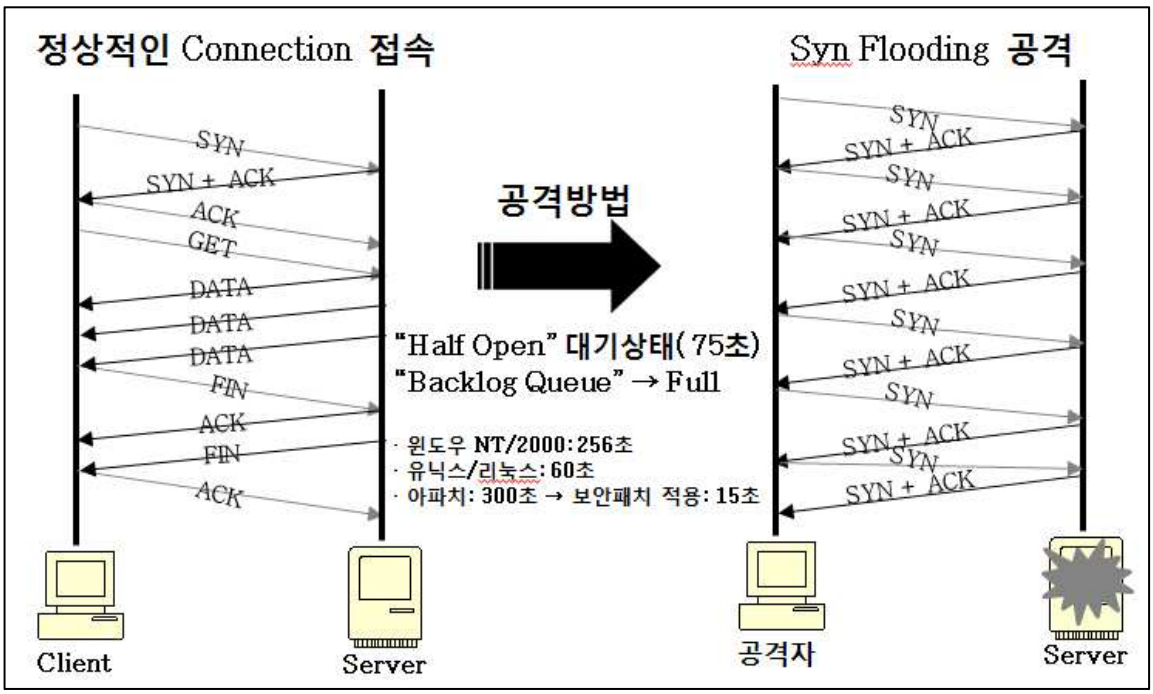
2.2 TCP Traffic Flooding 공격

2.2.1 SYN Flooding

가. 공격기법

□ TCP 연결과정(3-Way Handshaking)의 처음 단계인 SYN 패킷 전송 단계에서 공격자는 대량의 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

※ TCP 프로토콜이 데이터를 보내기 전에 연결을 먼저 맺어야 하는 특징을 이용한 방법임



【 그림 Ⅱ.3. 】 TCP Syn Flooding Attack Flow

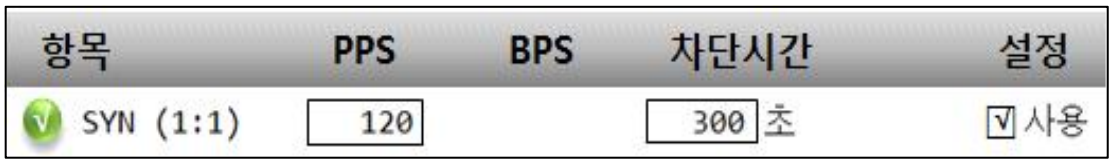
나. 대응방안

□ 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 효과적이며 또한 정상적인 트래픽의 임계치에 의거하여 비정상적으로 많은 트래픽을 유발하는 출발지 IP에 대해 차단하는 임계치 기반의 DDoS 방어

- (방안 1) 임계치 기반의 SYN Flooding 차단

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



【 그림 Ⅱ.4. 】 PPS 설정을 통한 SYN 연결 요청 제한 예

- (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

2.2.2 TCP Flag Flooding

가. 공격기법

□ 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

※ 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

나. 대응방안

□ 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

2.2.3 TCP Session

가. 공격기법

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

나. 대응방안

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

  Connection Timeout에 설정된 시간동안 Client와 웹서버 사이에 데이터신호의 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
```

【 그림 Ⅱ.5. 】 httpd.conf 의 timeout 설정을 통한 세션공격 차단 예

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

- **(방안 2) L7 스위치의 임계치 설정 기능을 이용한 차단**

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

```
when RULE_INIT {
        set :: max_connections_per_ip 200
        array set :: active_clients {}
        array set write_clent {
                10.31.0.230
                10.0.0.5
        }
}
```

【 그림 Ⅱ.6. 】 L7 스위치의 Connection Limit 설정 스크립트 예

2.3 IP Flooding 공격

2.3.1 LAND (IP Header Option 변조)

가. 공격기법

□ 인위적으로 송신지 IP 주소 및 Port를 목적지(대상 웹서버) IP 주소 및 Port와 동일하게 설정하여 트래픽을 전송하는 공격

※ 송신지 IP/Port와 목적지 IPort가 동일하기 때문에 네트워크 장비의 부하를 유발하기도 함

나. 대응방안

□ SYN 이외의 Flooding 공격을 방어하기 위해서는 다음과 같은 다양한 형태의 DDoS 방어 기법을 사용

□ 먼저 정상적인 TCP 세션 연결 이후 정상적인 트랜잭션(Transaction)이 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

2.3.2 Teardrop (IP Fragment Packet Flooding)

가. 공격기법

□ 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

나. 대응방안

□ SYN 이외의 Flooding 공격을 방어하기 위해서는 다음과 같은 다양한 형태의 DDoS 방어 기법을 사용

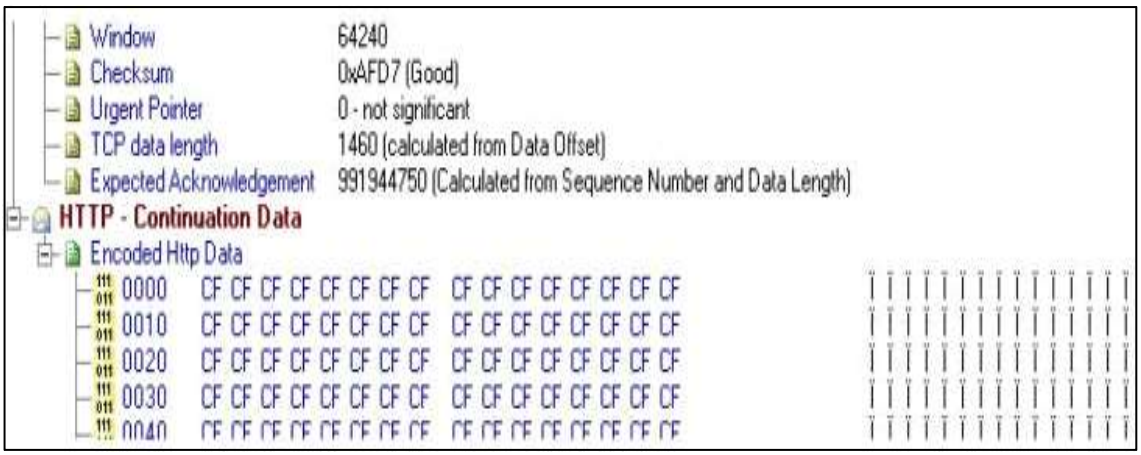
□ 먼저 정상적인 TCP 세션 연결 이후 정상적인 트랜잭션(Transaction)이 수행되는지에 대해 검증하여 만약 정상적인 트랜잭션이 이루어질 경우 서버로 전달해야 하고, 정상적인 트랜잭션 없이 TCP 세션 연결만 수행할 경우에는 서버로 전달하지 않아야 서버에 부하 증가 현상을 막을 수 있음. 또한 네트워크의 정상적인 환경에서 설정된 각 트래픽 유형별 임계치를 통하여 과도한 TCP 세션 연결에 대해 차단

2.3.3 HTTP Continuation (IP Fragment Packet Flooding)

가. 공격기법

□ 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



【 그림 Ⅱ.7. 】 HTTP Continuation Data Flooding 공격 패킷 예

※ 출처 : KrCERT/CC, 「DDoS 공격대응 가이드」, KISA, 2012, p. 27

나. 대응방안

□ 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 필요하며 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

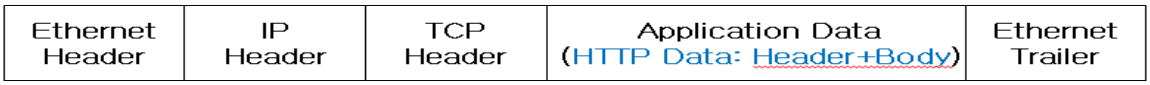
3 서비스(어플리케이션) 마비공격

3.1 HTTP Traffic Flooding 공격

□ HTTP 메시지는 Header와 Body로 구성되며 Header에는 보낼 메시지의 형식을 Body에는 실제 보낼 메시지의 내용을 정의

□ 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



【 그림 Ⅱ.8. 】 데이터 캡슐화에 따른 HTTP 메시지 송/수신 패킷 구조

□ HTTP 메시지는 클라이언트가 요구하는 목적에 맞는 지시자를 이용하여 구성할 수 있는데, 지시자의 종류는 다음과 같음

【 표 Ⅱ.2. 】 HTTP 메시지에 사용하는 지시자

지시자	설명
GET	○ 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.test.com/index.html
POST	○ 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.test.com?login=aaa
HEAD	○ GET과 같은 요청이지만 자료에 대한 정보(meta-information)만을 수신하는 요청
PUT	○ 해당 URL에 자료를 저장하는 요청
DELETE	○ 해당 URL의 자료를 삭제하는 요청
TRACE	○ 이전에 요청한 내용(히스토리)을 요청
OPTIONS	○ 서버가 특정 URL에 대해 어떠한 HTTP 지시자를 지원하는지 질의하는 요청
CONNECT	○ 프록시(Proxy)가 사용하는 요청

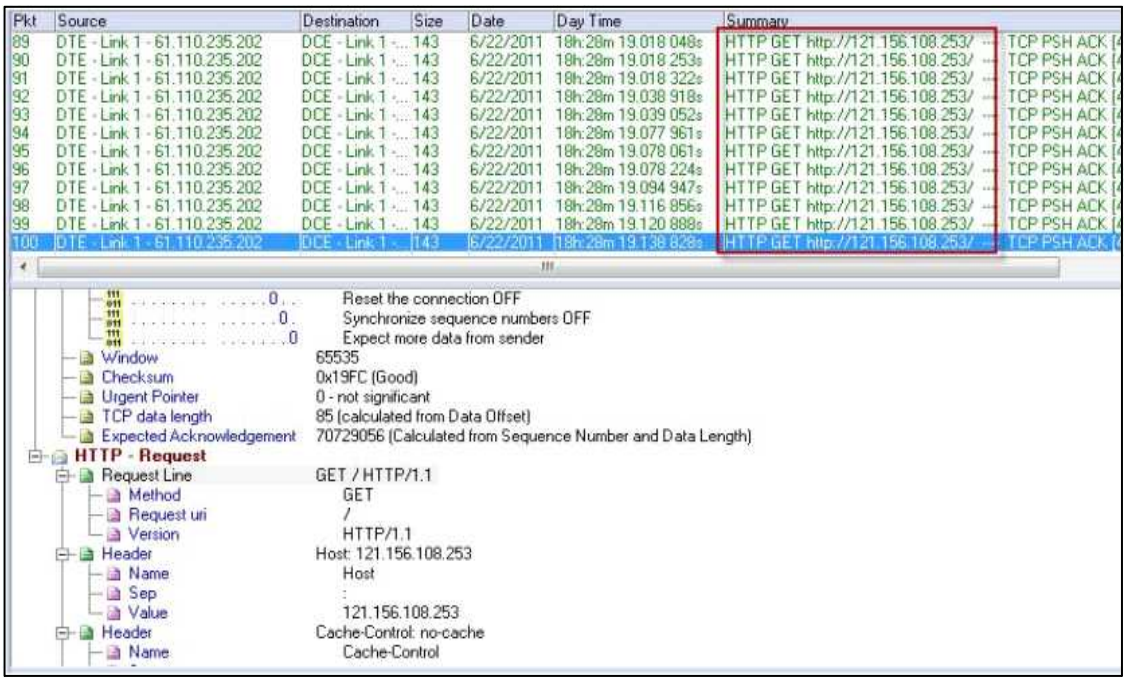
※ 이중 일반적인 웹서비스와 관련된 지시자(Indicator)는 GET과 POST이며, DDoS 공격에 가장 많이 사용되는 지시자임

3.1.1 Get Flooding

가. 공격기법

□ 공격자는 동일한 URL(예:test.com/index.jsp)을 반복 요청하여 웹 서버가 URL에 해당되는 데이터를 클라이언트에게 회신하기 위해 서버 자원을 사용하도록 하는 공격임

※ 웹서버는 한정된 HTTP 처리 Connection 용량을 가지기 때문에 용량 초과시 정상적인 서비스가 어려워짐



【 그림 Ⅱ.9. 】 Get Flooding 공격 패킷 예

※ 출처 : KrCERT/CC, 「DDoS 공격대응 가이드」, KISA, 2012, p. 27

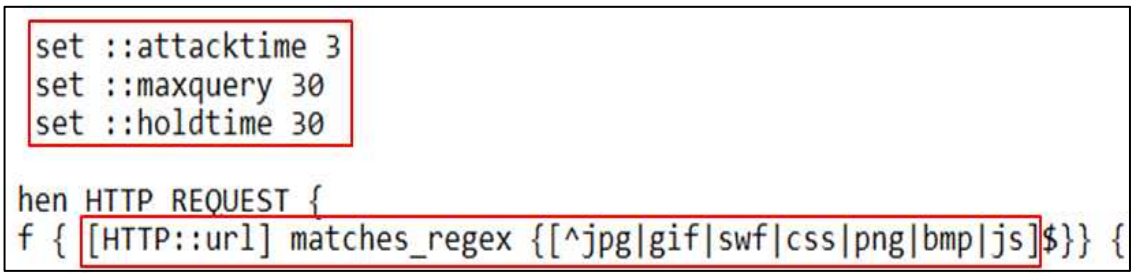
나. 대응방안

□ 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

□ 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

- **(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



【 그림 Ⅱ.10. 】 L7 스위치를 이용한 Get 요청 횟수 제한설정 예

- **(방안 2) 시간별 웹페이지 URL 접속 임계치 설정에 의한 차단**
  URL에 대한 시간별 임계치를 설정하여 임의의 시간 안에 설정한 임계치 이상의 요청이 들어온 경우 해당 IP를 탐지하여 방화벽의 차단목록으로 등록

- **(방안 3) 웹스크래핑(Web-Scraping) 기법을 이용한 차단**
  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

3.1.2 GET Flooding with Cache-Control (CC Attack)

가. 공격기법

□ 일반적으로 웹서버의 부하를 감소시키기 위해 캐싱서버를 운영하여 많이 요청받는 데이터(예:사진파일)는 웹서버가 아닌 캐싱서버를 통해 응답하도록 구축하는 경우

□ HTTP 메시지의 헤더정보에 포함된 Cache-Control 값을 no-store, must-revalidate로 지정하여 캐싱 장비가 응답하지 않고 웹서버가 직접 응답하도록 유도하여 웹서버의 자원을 소진시킴

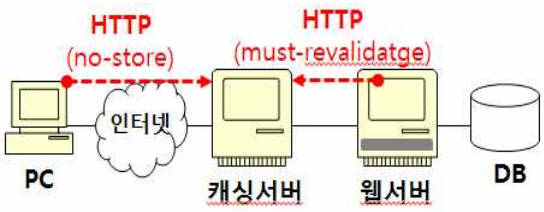
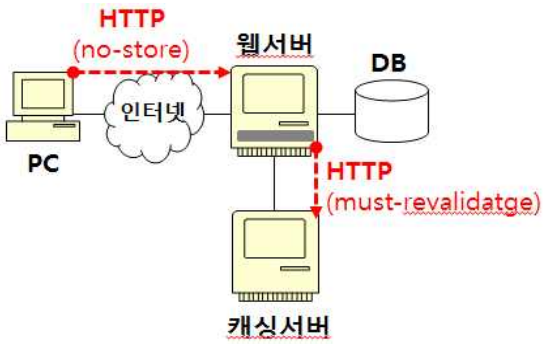
※ must-revalidate는 클라이언트가 보낼 경우, 캐싱장비에 영향을 미치지 않음

□ Cache-Control 옵션을 제외하고는 Get Flooding과 동일한 형태임

【 표 Ⅱ.3. 】 no-store와 must-revalidate

구분	설명
no-store (캐시저장 금지)	○ 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 있음 ※ 웹서버 내에 캐시를 운영하는 경우에는 must-revalidate를 사용하지 않음 ※ 캐싱서버는 웹서버의 원본데이터와 비교하여 최신 데이터로 업데이트함 ※ 클라이언트가 서버에게 보내는 HTTP 메시지에서 사용되는 정보가 아님

【 표 Ⅱ.4. 】 캐싱(Caching) 서버 운영 모델

포워딩(Fowarding) 방식	호스팅(Hosting) 방식
 ○ 캐싱서버를 웹서버 앞에 위치 ○ 캐싱서버가 요청정보를 먼저 처리	 ○ 캐싱서버를 웹서버 뒤에 위치 ○ 웹서버가 요청정보를 분석한 후 캐싱서버를 이용

나. 대응방안

□ 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

□ 하지만 이 공격은 Cache-Control이라는 HTTP 헤더 옵션을 사용하는 공격이므로, HTTP Cache-Control 헤더 옵션 별 임계치 정책으로 방어해야지만 큰 효과를 얻을 수 있음

- **(방안 1) 방화벽에 캐싱공격 문자열을 포함한 IP 차단**
  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
- **(방안 2) L7 스위치를 이용한 캐싱 공격 차단**
  L7 스위치를 이용하면 좀 더 세분화된 차단정책을 적용할 수 있음. 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-Control"
       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
```

【 그림 Ⅱ.11. 】 HTTP 문자열 검색을 통한 IP 차단 설정 예

3.2 HTTP Header/Option Spoofing Flooding 공격

□ 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

□ HTTP Header/Option을 이용한 공격에는 크게 다음과 같이 3가지로 구분되어짐

【 표 Ⅱ.5. 】 HTTP Header/Option 을 이용한 대표적인 공격기법

종류	공격 원리
Slow HTTP POST DoS	○ HTTP POST 지시자를 이용하여 서버로 전달할 대량의 데이터를 장시간에 걸쳐 분할 전송하면 서버는 POST 데이터가 모두 수신하지 않았다고 판단하여 연결을 장시간 유지하게 됨 ○ 만약 이러한 데이터를 전달하는 좀비PC가 많은 경우, 서버는 다른 정상적인 클라이언트에 대한 원활한 서비스가 불가능하게 되는 DoS 상태가 유발됨
Slow HTTP Header DoS (Slowloris)	○ 웹서버는 HTTP 메시지의 헤더부분을 먼저 수신하여 이후 수신할 데이터의 종류를 판단하게 됨 ○ 헤더부분을 비정상적으로 조작하여 웹서버가 헤더정보를 구분 할 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	○ 공격자는 웹서버와 TCP 연결 시, TCP 윈도우 크기 및 데이터 처리율을 감소시킨 후 HTTP 데이터를 송신하여 웹서버가 정상적으로 응답하지 못하도록 DoS 상태를 유발 ○ 웹TCP 윈도우 크기 및 데이터 처리율을 감소시키면 서버는 정상상태로 회복될때까지 대기상태에 빠지게 되어 다른 정상적인 클라이언트의 접속을 방해함

3.2.1 Slow HTTP POST DoS

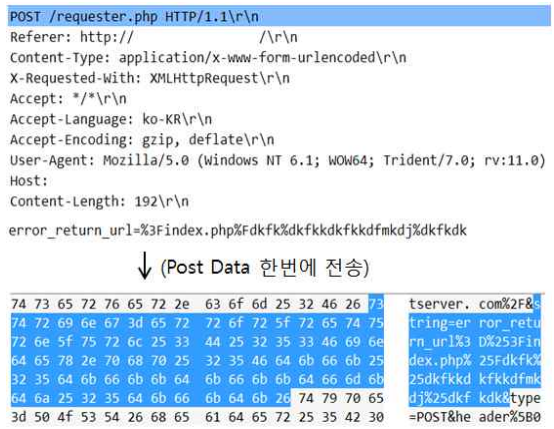
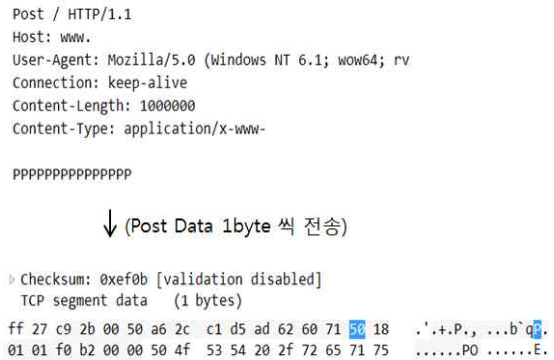
가. 공격기법

□ 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

※ GET 방식과 달리 POST 방식은 클라이언트가 서버로 전송할 데이터의 크기를 설정할 수 있기 때문에 서버는 데이터 수신시까지 대기해야 함

□ Slow HTTP POST 공격 기반의 패킷 형태

【 표 Ⅱ.6. 】 Slow HTTP Post DoS의 특징

정상적인 HTTP 메시지	Slow HTTP POST DoS 메시지
	
< 다량의 Post Data전송 >	< 1byte 씩 전송 >

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

나. 대응방안

- **(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

  (예시) iptables -A INPUT -p tcp --dport 80 -m connlimit —connlimit-above 30 -j DROP

  ※ 30개 이상의 concurrent connection에 대한 차단

- (방안 2) Connection Timeout과 Keepalivetimeout 설정을 통한 차단

Connection Timeout에 설정된 시간동안 Client와 웹서버 사이에 데이터신호의 이동이 전혀 없을 경우 웹서버는 연결된 Connection을 종료 Slow POST 공격 툴은 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
```

【 그림 Ⅱ.12. 】 httpd.conf 의 timeout 설정을 통한 세션공격 차단 예

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

- (방안 3) RequestReadTimeout(mod_reqtimeout Module) 설정을 통한 차단

Apache 2.2.15 버전과 그 이후 버전에서는 클라이언트 요청에 대한 더욱 세부적인 제한을 줄 수 있는 RequestReadTimeout이라는 지시자를 제공 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.
```

【 그림 Ⅱ.13. 】 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
```

【 그림 Ⅱ.14. 】 RequestReadTimeout 지시자 활용 예

3.2.2 Slow HTTP Header DoS (Slowloris)

가. 공격기법

□ 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 유지함

□ 충분히 많은 클라이언트를 이용하여 불완전한 메시지를 전달하면 웹서버는 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

【 표 Ⅱ.7. 】 Slow HTTP Header DoS의 특징

정상적인 HTTP 메시지	Slow HTTP POST DoS 메시지
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 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 < \r \n \r \n 으로 종료 >	▷ Internet Protocol Version 6, Src: ▷ User Datagram Protocol, Src Port: ▷ Hypertext Transfer Protocol X-a: b\r\n 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 < \r \n 으로 종료 >

□ 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

나. 대응방안

□ “2.2.1 Slow HTTP POST DoS 대응방안” 참고

3.2.3 Slow HTTP Read DoS

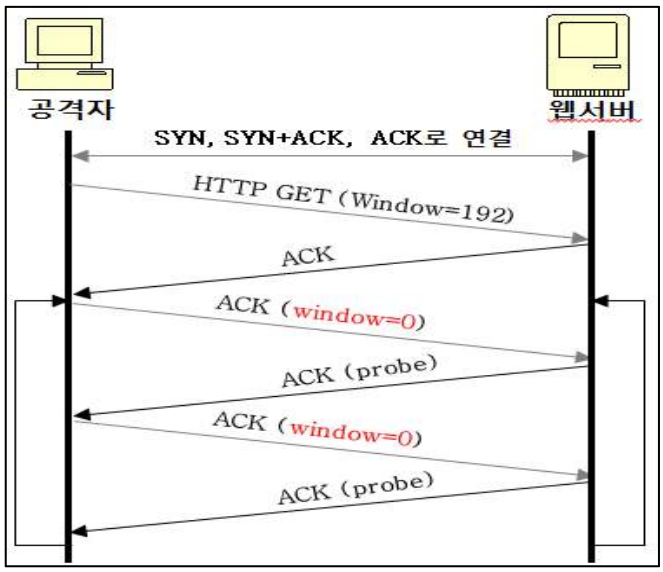
가. 공격기법

□ 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



【 그림 Ⅱ.15. 】 Slow HTTP Read DoS 과정

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

나. 대응방안

□ “2.2.1 Slow HTTP POST DoS 대응방안” 참고

3.3 기타 서비스 마비 공격

3.3.1 해시도스(HashDoS) 공격

가. 공격기법

□ HashDoS는 해시테이블을 이용하는 웹서버를 대상으로 한 공격임

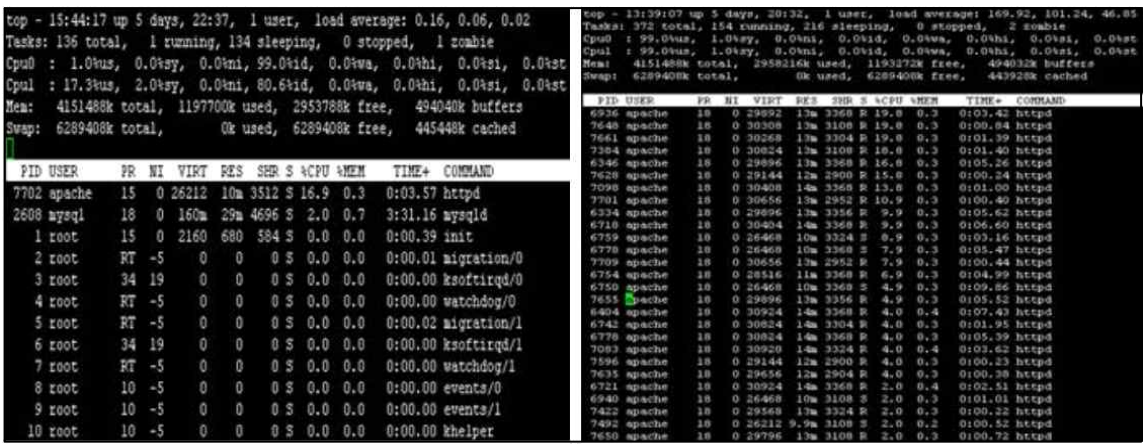
□ 클라이언트에서 HTTP 메시지를 통해 전달되는 각종 매개정보를 관리하는 해시테이블의 인덱스 정보가 중복되도록 유도하여 기저장된 정보 조회시 많은 CPU 자원을 소모하도록 유도하는 공격

※ 웹서버는 GET, POST 방식으로 전송되는 HTTP 메시지에 포함된 매개변수의 효과적인 관리(정보 접근을 쉽고 빠르게 수행)를 위해 해시(Hash) 구조를 사용

□ 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



【 그림 Ⅱ.16. 】 정상적인 상태(좌)와 HashDoS를 이용한 공격시 상태(우)

※ 출처 : KrCERT/CC, 「DDoS 공격대응 가이드」, KISA, 2012, p. 37

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

< 참고 >

▶ 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)
- 해싱함수의 연산에 의해 구해진 위치에 각 정보를 한 개 이상 보관할 수 있도록 구성된 기억 공간

▶ 해시 충돌(Hash Collision)
- 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

나. 대응방안

□ HashDos는 웹서버의 설정값 변경만으로도 차단이 가능

- Tomcat, PHP, Ruby 등 최신 버전에서는 HTTP Post Parameter 개수에 제한을 둘 수 있는 기능을 추가하였으므로 개수 제한 적용 위해 최신버전으로 업데이트

□ 톰캣(Tomcat)에서의 HashDoS 차단 방안

- **(방안 1) 파라미터 개수 제한**

  TOMCAT_HOME/conf/server.xml의 Connector 부분을 다음과 같이 설정

  (예시) 〈Connector port= "8009" protocol="AJP/1.3" maxParameter Count="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
  (예시) 〈Connector port="8009" protocol="AJP/1.3" maxPostSize="xxx" …./〉

□ 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/viewvc?view=revision&revision=321003에서 수정된 소스를 다운받아 재빌드하여 적용
  ※ 대상 : PHP_5_4/main/main.c, PHP_5_4/main/php_globas.h, PHP_5_4/main/php_variables.c

3.3.2 헐크도스(HulkDoS) 공격

가. 공격기법

□ HULK(Http Unbearable Load King) DoS 는 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 차단 정책을 우회한다는 특징을 가짐

※ HULK는 Imperva 사의 Barry Shteiman이라는 연구원이 자신의 블로그인 Nerd에 DDoS 연구나 교육을 위해 만든 도구이지만 해커에 의해 DDoS 공격도구로 악용되고 있음

□ 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

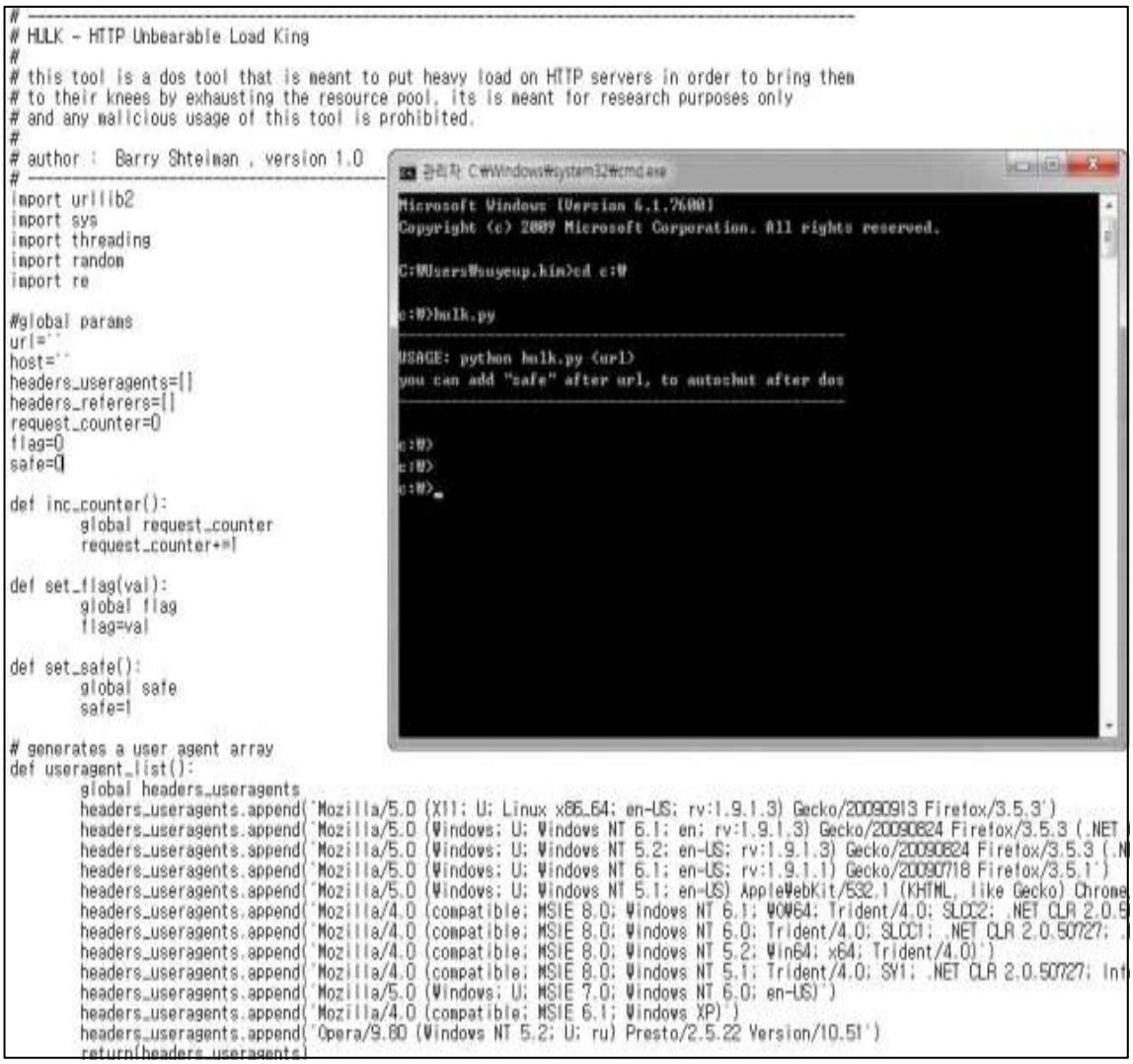
- 파라미터 추가 형식 : ?이름1=값1&이름2=값2&....&이름n=값n
  (예) http://search.yOO.com/search.yOOO ?where=nexearch&
  query=get&fbm=1&ie=utf8

□ HULK DoS는 이러함 파라미터를 지속적으로 변경하여 웹서버로 전달

※ 파라미터가 지속적으로 변경되는 것을 제외하고는 일반적으로 GET Flooding과 동일한 형태임

□ HULK DoS 공격도구 분석

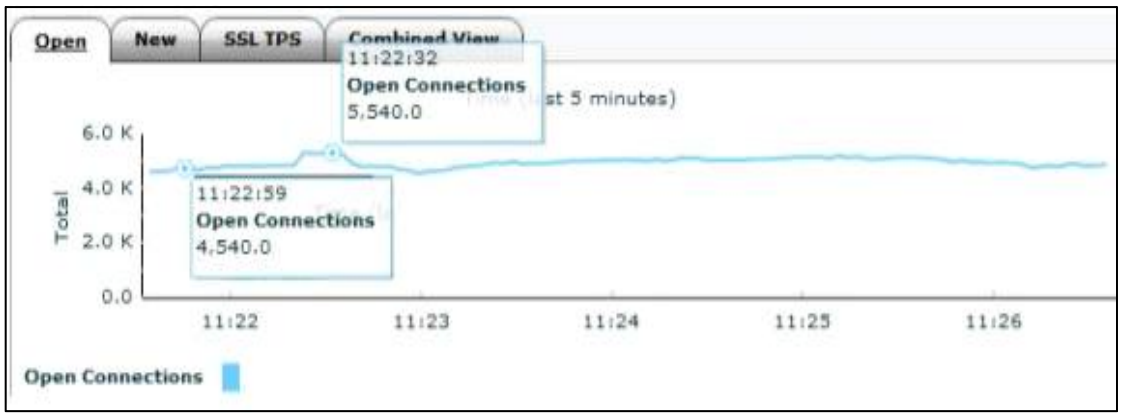
- HULK DoS 공격도구는 python으로 작성되었으며 윈도우즈 환경에서는 Active python을 설치하여 아래 명령어로 간단히 실행할 수 있음
  (예) c:\> hulk.py http://test.com/



【 그림 Ⅱ.17. 】 HULK DoS의 소스파일과 실행화면

※ 출처 : KrCERT/CC, 「DDoS 공격대응 가이드」, KISA, 2012, p. 40

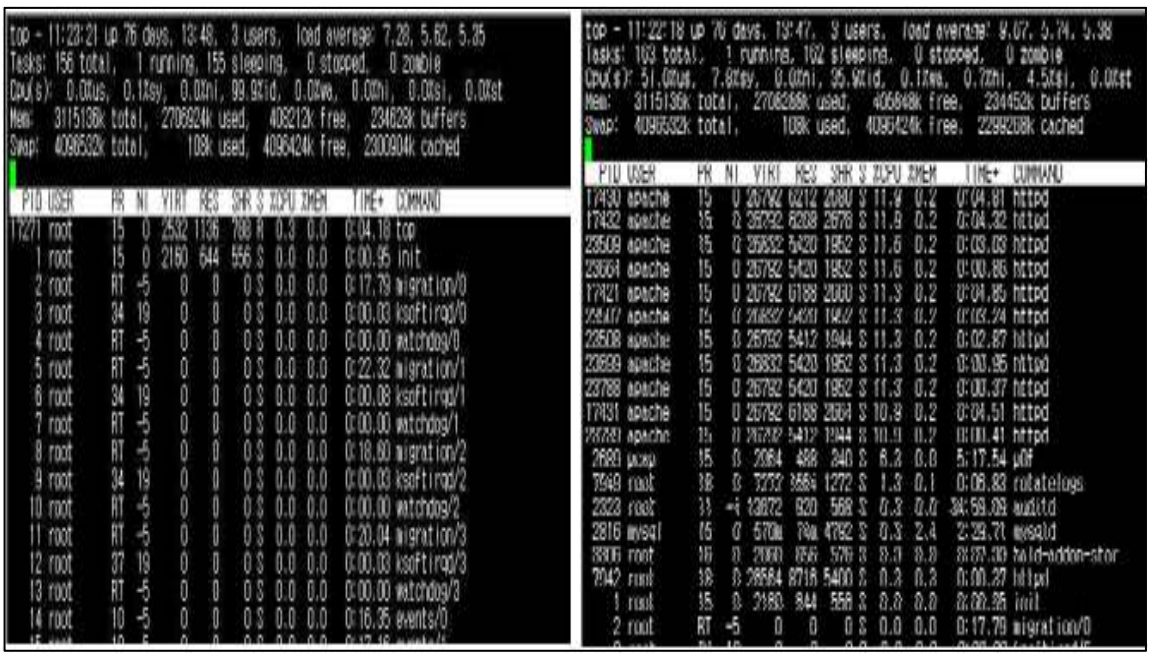
- HULK DoS 공격도구는 서버와 초당 평균 1,000개 정도를 연결하여 약 10M 규모의 공격트래픽을 전달 (아래 그림에서 연결수치 4,540 → 5,540)



【 그림 Ⅱ.18. 】 HULK DoS 툴을 이용한 DoS 공격시 서버 연결 정보

※ 출처 : KrCERT/CC, 「DDoS 공격대응 가이드」, KISA, 2012, p. 41

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



【 그림 Ⅱ.19. 】 정상적인 상태(좌)와 HULK DoS를 이용한 공격시 상태(우)

※ 출처 : KrCERT/CC, 「DDoS 공격대응 가이드」, KISA, 2012, p. 41

- HULK DoS 공격도구에서 생성하는 공격트래픽 형태는 아래 그림과 같이 HOST 뒤에 "/?" 를 입력 후 임의의 문자를 무작위로 입력하면 웹서버는 파라미터 유효검사를 하게 되며, 파라미터가 유효하지 않더라도 "/" URL로 동작하여 클라이언트에게 200ok 응답을 하게 됨

【 그림 Ⅱ.20. 】 Request Header URL, User-Agent, Referer의 지속적 변화

※ 출처 : KrCERT/CC, 「DDoS 공격대응 가이드」, KISA, 2012, p. 41

나. 대응방안

- **(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

  (예시) iptables -A INPUT -p tcp --dport 80 -m connlimit --connlimit -above 30 -j DROP

  ※ 30개 이상의 concurrent connection에 대한 차단

- **(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

- **(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 함

  자동화된 DDoS 공격 tool을 이용한 공격을 사전에 차단하여 웹서버의 부하를 감소시킬 수 있음