情報保護學會論文誌
第20卷 第5號, 2010. 10

웹서비스 대상 경량화 된 응용계층 DDoS 공격 대응 메커니즘*

이 태 진,† 임 채 수, 임 채 태, 정 현 철
한국인터넷진흥원

Light-weight Defense Mechanisms for application layer DDoS Attacks in the Web Services*

Tai-jin Lee,† Chae-su Im, Chae-tae Im, Hyun-chul Jung
Korea Internet & Security Agency

요 약

네트워크 대역폭 소모 위주로 발생했던 DDoS 공격이 최근에는 웹서비스 대상 응용계층에서의 DDoS 공격 위주로 발생하고 있다. 응용계층 DDoS 공격은 Source IP당 발생하는 트래픽이 감소하고 정상 사용자의 트래픽과 유사한 형태로 진화하고 있어 기존 Threshold 기반 탐지기법으로 대응하기 어렵다. 본 논문에서는 발생 가능한 Get Flooding의 모든 유형을 도시하고, 간단하면서도 강력한 3가지 대응 메커니즘을 제시한다. 특히, 제안된 메커니즘은 현재까지 발생한 모든 DDoS 공격도구를 탐지 및 차단할 수 있으며, 성능분석 결과 상용 환경에 적용할 수 있을 것으로 예상된다.

ABSTRACT

Recently, network based DDoS attacks have been changed into application layer DDoS attacks which are targeted at the web services. Specially, an attacker makes zombie PCs generate small traffic and its traffic pattern has been similar to the normal user's pattern. So, existing HTTP PPS based Threshold cannot defend the DDoS attacks effectively. In this paper, we displayed all the GET Flooding attack types and propose three DDoS attack defense mechanisms which are simple and very powerful. Proposed mechanisms can defend all the existing GET Flooding DDoS attacks and be deployed in the real environment immediately with little resource consumption.

Keywords: DDoS, Web Service, Behavior, Challenge

I. 서 론

네트워크 대역폭 소모위주의 DDoS(Distributed Denial of Service) 공격은 수십년간 끊임없이 발생해왔으나, 최근에는 세션 고갈 형 공격, GET Flooding 등을 통한 시스템의 자원을 고갈시키는 공격이 많이 발생하고 있다. 세션 고갈 형 공격으로는 부하분산 및 서버 보호 목적의 L4스위치, 방화벽 등 해당 장비의 세션 관리 능력의 유한성을 악용하여 SYN Flooding, Slowloris 등이 있으며, Get Flooding 등 서버 자원 고갈 형 공격으로 서버 자원의 트랜잭션 처리 한계를 악용하여 정상적 접속 후 대량의 요청을 발생시켜 네트워크 모니터링으로 파악이

접수일(2010년 4월 8일), 수정일(2010년 6월 24일), 게재확정일(2010년 8월 13일)
* 본 연구는 지식경제부 및 한국산업기술평가관리원의 IT산업원천기술개발사업의 일환으로 수행되었음. [2009-S-038-01, 분산서비스거부공격대응기술 개발]
† 주저자, tjlee@kisa.or.kr

불가능한 공격으로 7.7 대란의 Cache Control attack이 해당된다.

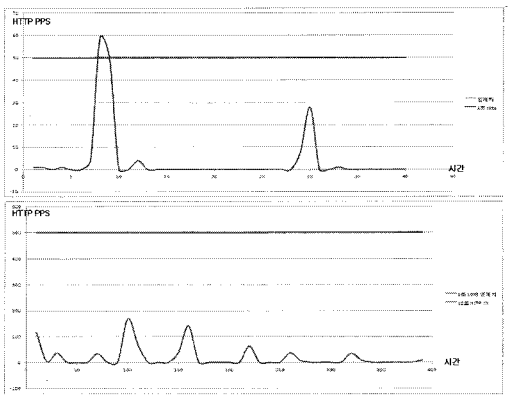
특히, 최근 발생하는 GET Flooding 공격은 각각의 좀비PC가 최소한의 HTTP 트래픽을 발생시키면서, 대량의 좀비PC를 동원하여 시스템을 마비시키는 형태의 공격이 주를 이루고 있다. CC Attack 같은 경우도 일반적인 웹 접속 시에는 1회 요청하는 페이지를 한대의 좀비 PC에서 초당 6~10여회를 요청하는 패턴을 보였으며 지속적으로 웹 서버에 부하를 발생시킨다. 그러나, 주요 DDoS 대응장비에서는 HTTP PPS(Packet Per Second : 초당 발생한 패킷 수), CPS(Character Per Second : 초당 발생량 트래픽 량)에 대한 Threshold 기반 대응방법을 적용하고 있는데, 이는 좀비PC당 소규모 트래픽을 발생시키는 최근 DDoS 공격동향을 비추어 볼 때, 효과적으로 동작하지 못한다. 본 논문에서는 발생 가능한 Get Flooding의 모든 공격유형을 도시하고, 간단하면서도 강력한 대응 메커니즘을 제안한다. 2장에서는 상용환경에서 사용하고 있는 Volume 기반 대응 메커니즘의 한계점을 웹서비스 특징에 비추어 분석 및 개선방안을 제시하고, 3장에서는 웹 서비스에 대한 다양한 통계를 기반으로 3개의 DDoS 공격 대응 메커니즘을 제안한다. 4장에서는 DDoS 공격대응시스템 구축을 통해 제안한 메커니즘을 성능분석 결과를 제시하고, 5장에서는 성능분석 결과를 토대로 본 대응 메커니즘이 가지는 의미를 분석하고 결론을 제시한다.

II. 관련 연구

응용계층 DDoS 공격에 대한 여러 가지 연구가 진행되고 있다[1,2,3]. 웹 서비스 특성상 이용자 IP 리스트는 flat하게 분포하지 않고 기존에 들어왔던 이들이 계속 들어오는 특징을 이용하여, 그 비율을 이용한 DDoS 공격 탐지방안[4], 웹 서비스 이용패턴 정보를 분석하여 공격자로 의심스러운 Source IP 들을 대상으로 Greylist로 분류하고, 이들에 대해 적은 리소스를 할당하는 방법에 대한 연구[5], 각 URL page hit distribution 등을 통계적 방식에 기반하여 학습한 후, 일시적인 request 폭주와 DDoS 공격을 구별하는 방법[6], 웹 서비스 사용자 page 이동경로에 따른 분석[7], 허용된 사용자에 한해 웹 서비스 접속을 허용하는 Admission Control을 통해 대응하는 방법 등이 제안되어 있다[8]. 위 연구동향 중 URL page hit distribution은 많은 연산량을 필요로 하며 웹서비스에 올라가는 컨텐츠 및 시기에 따라 크게 변하여 Threshold 설정이 어려운 단점이 있고, Admission Control 기법은 out of path 방식으로는 동작할 수 없고, 인라인으로 동작해야 하는데 세션 관리가 필요한 단점이 있다. 본 논문에서는 상용 환경에 바로 적용할 수 있는 대응기술을 제안하고자 하므로, 상용 DDoS 공격대응장비에서 사용하고 있는 HTTP PPS 트래픽에 대한 Threshold 기반 대응방법의 문제점을 기반으로 간단하면서도 강력한 대응 메커니즘을 제시하고자 한다. HTTP PPS 트래픽에 대한 Threshold는 초 단위로 트래픽을 수집하고, HTTP 전체에 대한 Threshold를 설정하는데 다음의 문제점이 있다.

2.1 1초 단위의 Volume 기반 탐지

웹서비스에서 사용되는 HTTP 트래픽은 일정한 특징을 가지고 있다. 정상 사용자의 경우, 대부분의 시간은 트래픽이 발생하지 않는 상태로 있다가 웹 브라우저를 실행하거나 메뉴·링크 클릭을 통한 웹서핑, refresh 등을 통해 어떤 행위를 하게 되면, HTTP Request가 발생하는데 한번의 행위는 수십에서 수백개의 HTTP Request를 한순간에 발생시킨다. 즉, 정상 사용자의 웹서핑에서도 평소에는 트래픽 발생량이 매우 저조하지만, 특정 순간에는 대량의 HTTP Request가 발생하게 된다. 따라서 1초 단위의 Volume 기반 탐지 방법은 해당 클라이언트의 비정상 행위 여부를 탐지하는데 한계를 가지게 된다. [그림 1]은 정상 사용자의 1초 단위의 HTTP PPS와 10초 단위의 HTTP PPS를 나타낸다. 정상 사용자의 웹



[그림 1] 정상 사용자의 1초, 10초 단위의 HTTP 트래픽 분포

서핑 시, 1초 단위로 트래픽을 보면 사용자의 행위가 발생한 시점에서 대량으로 발생하여 보통 초당 Threshold로 설정하는 50을 넘어서지만, 그 외의 시점에서는 트래픽 발생량이 미미하다. 즉, 정상 사용자조차 트래픽 발생량의 변화가 극심하므로, 1초 단위의 Volume 기반 탐지기법은 한계가 존재한다. 그러나, 10초 단위로 트래픽을 보면 사용자가 행위를 어느 정도 했는지 판단할 수 있다. 10초 단위 분석기법의 효과에 대해서는 3장에서 상세히 기술한다.

2.2 HTTP 트래픽 전체에 대한 Threshold

앞서 기술한대로, 웹 서비스 특성 상 정상 사용자의 단 한번의 행위가 적게는 수십, 많게는 수백개의 HTTP Request를 발생시키게 된다. 따라서, 정상 사용자를 공격자로 오탐지하지 않도록 하기 위해, Threshold는 보통 매우 큰 값으로 사용하게 되어 다수의 미탐지가 발생한다. 특히, 다수 좀비PC를 사용하고 각 좀비PC 당 소규모의 트래픽을 발생시키는 최근 DDoS 공격추세에 비추어보면 더욱 그렇다. 여기서 HTTP Request들을 분석하면 사용자가 웹브라우저 상에서 어떤 행위를 했을 때 발생되는 HTTP Request와 이에 수반하여 발생하는 HTTP Request로 구분됨을 알 수 있다. 예를 들어, 특정 페이지 요청 시 200 OK로 해당 페이지에 대한 정보를 받고, 웹브라우저에 해당 페이지를 도시하기 위한 여러 iframe, jpg, gif 등 다양한 HTTP Request가 발생하게 된다[9,10]. 여기서, 처음 발생한 HTTP Request는 사용자의 행위에 의해 발생했지만, 그 뒤에 수반되는 Request는 해당 페이지 도시를 위한 수반되는 Request들이다. 만일, 수많은 HTTP Request 중에서, 사용자 행위에 의한 발생한 HTTP Request를 식별하고, 그들에 대해 Threshold를 설정할 수 있다면 DDoS 공격에 효과적인 대응이 가능하다. 왜냐하면 실제 사용자의 행위는 예를 들어 10초에 30번 이상 발생하는 것은 웹브라우저 상에서 사실상 불가능하기 때문에, 10초당 사용자 행위에 대한 Threshold를 30으로 설정하면 효과적으로 대응할 수 있다.

지금까지 주요 DDoS 전용장비에 적용되어 있는 HTTP에 대한 Threshold 기반 대응 메커니즘의 한계점과 개선하기 위한 접근방안을 제시하였다. 1초 단위 트래픽 분석 이외에 10초 단위의 트래픽 분석이 필요하며, HTTP 전체에 대한 Threshold 기반 대응 외에 사용자의 실제 행위에 대한 Threshold를 통해 정교한 대응이 가능하다. 이러한 접근방안을 토대로, 3장에서는 본 논문에서 제안하는 3가지 대응 메커니즘을 제안한다.

III. DDoS 공격대응 메커니즘

[그림 2]는 본 논문에서 제안하는 동작 메커니즘을 나타낸다. Source IP들로부터의 HTTP Request 발생 시, 2개의 Behavior 기반 DDoS 공격대응 메커니즘을 통해 정상 Source IP와 의심스러운 Source IP로 분류하고, 의심스러운 행위를 하는 Source IP에 대해 Challenge 기반 DDoS 공격대응 메커니즘을 적용하여 악성행위를 하는 Source IP를 blacklist 처리한다.

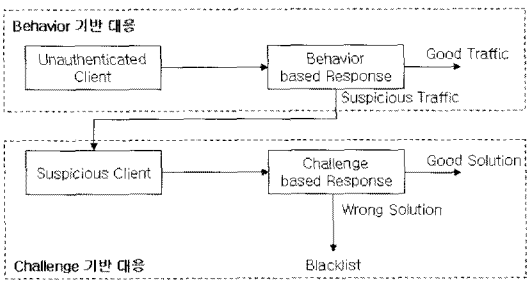
3.1 Direct Request에 대한 URL별 Threshold 기반 대응 메커니즘

3.1.1 Direct Request, URL별 Request 수와의 상관관계

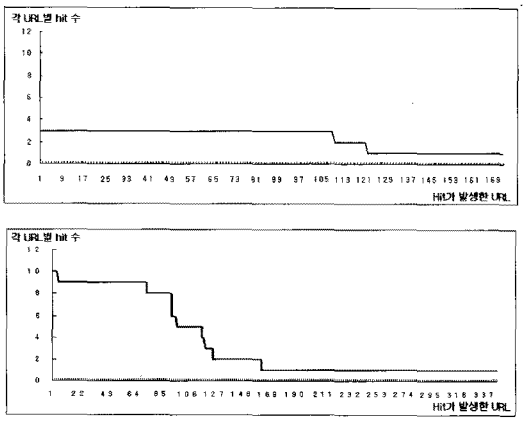
사용자 행위에 기반한 HTTP Request 구분을 위해 다음과 같이 용어를 정의한다.

- Direct Request : 웹브라우저 실행, 특정 메뉴 클릭, refresh 등 사용자가 직접 어떤 행위를 했을 때, 발생하는 HTTP Request
- Indirect Request : Direct Request에 의해 발생하게 되는 image, iframe 등에 대한 HTTP Request

HTTP 트래픽 분석결과, 사용자가 한번의 행위(메뉴클릭, 웹브라우저 실행 등)를 했을 때, 발생하는 수많은 HTTP Request 중에서 동일한 URL(Uniform Resource Locator)을 요청하는 경우는 거의

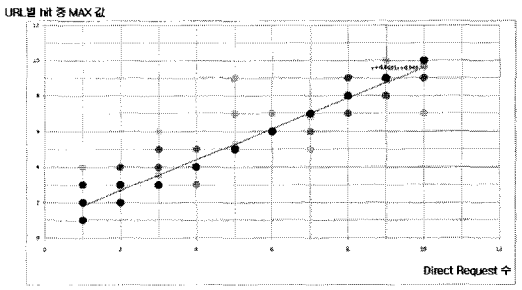


[그림 2] DDoS 공격대응 메커니즘



〔그림 3〕 Direct Request 횟수 3, 10에서의 URL별 HTTP Request 분포

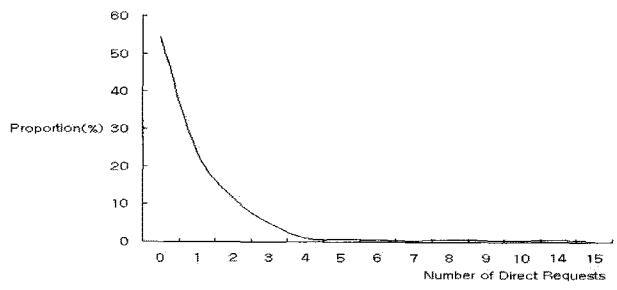
없다(하나의 웹 페이지에 배너 등에서 같은 이미지가 중복 사용될 수 도 있으나 이는 관리자가 파악할 수 있는 내용이므로, 예외로 함). 예를 들어, 한번의 Direct Request가 100개의 Indirect Request를 발생시킨다면, 100개의 Indirect Request는 각각 다른 URL을 요청한다. 즉, 1개의 Direct Request에 대해 발생하는 Indirect Request가 n개라면($Request_k$: $Request_k$는 k번째 URL을 요청한 Request 횟수, $k=1...n$), $\max(Request_1, Request_2 .... Request_n) \leqq 1$이며 m개의 Direct Request가 발생할 경우, $\max(Request_1, Request_2 .... Request_n) \leqq m$이 된다. 따라서, 각 URL별 Request 수는 Direct Request 수 이상으로 존재할 수 없다. 〔그림 3〕은 특정 포털사이트를 대상으로 10초간 발생한 Direct Request의 수가 3과 10일 때, 각 URL별 Request 수를 큰 순서대로 정렬한 분포를 나타낸다. X축은 hit이 발생한 URL들을 나타내는데, 편의상 숫자로 표시하였다.



〔그림 4〕 Direct Request와 URL별 최대 Request 수와의 상관관계

관측결과, 각 URL별 Request 수는 Direct Request 수인 3과 10을 초과하여 존재하지 않는다. 이를 검증하기 위해 국내 주요 포털사이트 5개를 대상으로 10초간 각 사이트별로 웹브라우저 실행, 메뉴 클릭, refresh 등 Direct Request 수를 1부터 10까지 증가시킬 때, URL별 Request 수 중 최대 값을 도식화하였다. 각각의 점은 100번의 실험데이터 평균값이다. 〔그림 4〕는 10초간의 Direct Request 수와 최대 URL Request 수와의 상관관계를 나타낸다.

관측결과, 사용자의 실제 행위 횟수를 나타내는 Direct Request 수와 각 URL별 Request 수는 유사한 값을 나타낸다. 그러므로, 사용자의 실제 행위 횟수에 대한 Threshold가 S라면, 각 URL별 Request 수에 대한 Threshold를 S로 설정하여 이상 트래픽을 탐지 및 대응할 수 있다. 〔그림 4〕에서 Direct Request 수를 벗어난 일부 사례들 및 동일한 배너 이미지 사용 등의 경우가 존재하므로, 이상 트래픽으로 탐지한 경우 Challenge 기반 대응 메커니즘 등을 통해 DDoS 공격 발생여부를 최종 판단한다.



k	proportion	Direct Request 수
2	75%	4.169
3	88.88%	5.753
4	93.75%	7.337
5	96%	8.921
6	97.22%	10.505
7	97.95%	12.089
8	98.43%	13.673
9	98.76%	15.257
10	99%	16.841

〔그림 5〕 정상 사용자의 Direct Request 분포 및 Chebyshev's Inequality

3.1.2 Direct Request에 대한 Threshold

여기서는 Direct Request에 대한 Threshold를 어떻게 설정할 것인지 분석한다. 이를 위해, 주요 포털 사이트 5개에 대해 정상적인 웹 서비스를 이용할 때 10초당 Direct Request가 몇 번이나 되는지 관측하고, Direct Request 수 별 분포 비율로 나타냈다. [그림 5]는 10초 단위의 정상 사용자의 Direct Request에 대한 분포와 이 분포에 대한 Chebyshev's Inequality를 나타낸다.

[그림 5]와 같이, 10초 동안 정상 사용자의 Direct Request 수는 2 이하가 89.4%, 4 이하가 95.5%를 차지한다. 예상했던 대로, 대부분 Direct Request가 4 이하에서 존재한다. Chebyshev's Inequality에 따르면, 분포모양에 상관없이 $(1-\frac{1}{k^2})\times 100(\%)$의 데이터는 적어도 $average-k\sigma$와 $average+k\sigma$ 사이에 위치한다. 본 분포에 대해 Chebyshev's Inequality에 따라 k가 6인 경우, 10초 동안의 Direct Request 수 중 10.505 이하에 최소 97.22%의 데이터가 존재한다. 그러므로, Direct Request에 대한 Threshold를 10으로 설정하면, 오탐지율 최대값은 2.78% 이하가 된다. Threshold를 초과하는 Source IP에 대해서도 무조건 필터링 처리하는 것이 아니라 약 5초~10초정도 필터링 처리하고, 반복될 경우 DDoS 공격이 발생한 것으로 간주하는 등 DDoS 공격징후를 토대로 다양한 DDoS 공격대응정책을 적용하여 오탐지율 문제를 해결할 수 있다.

3.1.3 대응 알고리즘

Direct Request에 대한 Threshold 기반 대응 메커니즘 알고리즘을 나타낸다.

- $p_{t,s,k}$: 특정 시간구간 t, IP s에서 k번째 URL에 대한 HTTP PPS
- $p_{T,s,k}$: 전체 시간구간 T에서의 $p_{t,s,k}$의 합
- n : 전체 시간구간 T동안 요청이 발생한 URL 개수
- $threshold_1$: $p_{T,s,k}$에 대한 Threshold

[표 1] Direct Request에 대한 Threshold 기반 대응 알고리즘

```
packet received, calculate p_{t,s,k}, n
while ( t != T ) {
  p_{T,s,k} = p_{T,s,k} + p_{t,s,k}
  t = t + 1
}
for k=1 to n
  if p_{T,s,k} > threshold_1,
    Source IP s is regarded as zombie client
    and is blocked
p_{T,s,k} = 0, t = 0, n = 0
```

3.2 URL 당 평균 HTTP에 대한 Threshold 기반 대응 메커니즘

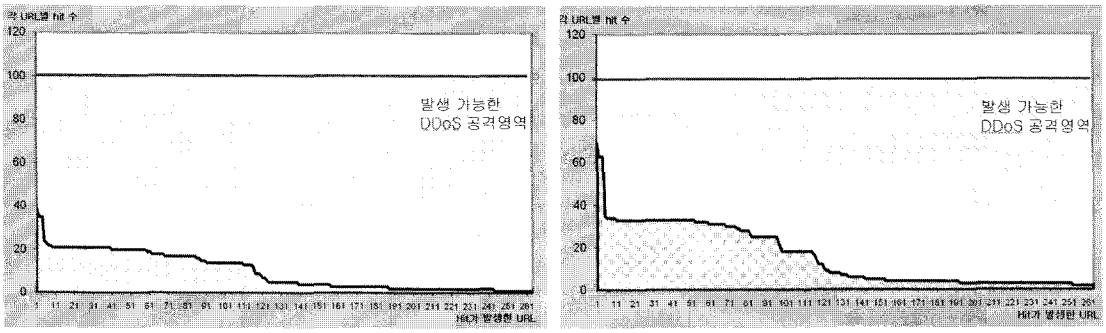
3.2.1 발생 가능한 DDoS 공격유형

앞서 기술한 Direct Request에 대한 Threshold는 기존 대응기법보다 정교한 대응을 할 수 있으나, 이를 우회할 수 있는 DDoS 공격형태가 발생할 수 있다. [그림 6]은 DDoS 공격이 발생할 수 있는 유형을 나타낸다.

[그림 6]은 주요 포털 사이트 5개를 대상으로 100초간 30번, 70번의 Direct Request를 발생시켰을 때, URL별 hit 수를 높은 순으로 정렬한 결과를 나타낸다. X축은 hit이 발생한 각각의 URL을 나타내며, 편의상 숫자로 표기하였다. 정상 사용자의 웹 서비스 이용패턴에서 100초간 30번의 Direct Request를 발생시키기는 쉽지 않으며, 70번은 지속적으로 F5를 눌러서 refresh 할 때나 가능한 수준이다. 앞서, 10초 단위에서 Direct Request 기준이 10이므로 Threshold 100 기준으로 통계결과를 보면, Direct Request 30의 경우 9.66% 영역만 차지하고 있어 DDoS 공격 이상 징후로 탐지되지 않으면서 지금 발생한 트래픽 대비 최대 1,035%의 트래픽을 더 발생시킬 수 있고, Direct Request 70의 경우 15.2% 영역만 차지하고 있어 지금 발생한 트래픽 대비 최대 658%의 트래픽을 더 발생시킬 수 있다. 이와 같은 DDoS 공격을 탐지 및 대응하기 위해서는 HTTP 트래픽 패턴을 세부적으로 분석해야 한다.

3.2.2 정상 사용자의 HTTP 트래픽 패턴 분석

정상 사용자가 어떤 행위를 했을 때 발생하는 HTTP Request가 S개, 그때까지 시간이 T라면, 최초 행위 발생 후 T 이내인 T_1 시점에서 신규 행위가 발생하게 되면, 그때까지 발생한 HTTP 개수 및 요청한 URL 개수는 $S\times\frac{T_1}{T}$이며, 신규 행위 발생시점인



〔그림 6〕 정상 사용자의 URL별 HTTP 분포 및 발생 가능한 DDoS 공격

T_1이 T보다 클 경우에는 HTTP 개수 및 요청 URL 개수는 S가 된다. 정상 사용자의 행위횟수가 많아지게 되면, T_1이 점차 작은값을 가지게 됨에 따라, $S \times \frac{T_1}{T}$에 해당하는 HTTP Request 개수도 크게 증가하지 않는다. 따라서, URL 당 평균 HTTP 개수는 정상 사용자의 경우, 일정치 이상으로 증가하기 어렵다. 〔그림 7〕은 10초간 사용자의 Direct Request가 증가함에 따라, 전체 HTTP 개수와 URL 당 HTTP 개수를 비교하여 나타낸다.

〔그림 7〕은 10초간 Direct Request 수를 1부터 10까지 증가시킬 때, 발생한 트래픽에 대해 좌측은 HTTP PPS, 우측은 URL당 평균 HTTP 개수를 나타낸다. 여기서 발생한 트래픽은 정상 사용자가 웹브라우저를 통해 사용자의 행위 횟수를 증가시킬 때의 데이터임에도 불구하고 10초당 850개가 넘는 HTTP가 발생하게 되어, 정상과 공격 트래픽을 구분하기 어렵다. 그러나, URL 당 평균 HTTP 개수를 보면, 10초간 10번의 Direct Request 발생, 즉 1초마다 계속 페이지 refresh를 할 때조차도 URL 당 HTTP 개수는 3.11이상으로 증가하지 않는 것을 확인할 수 있다.

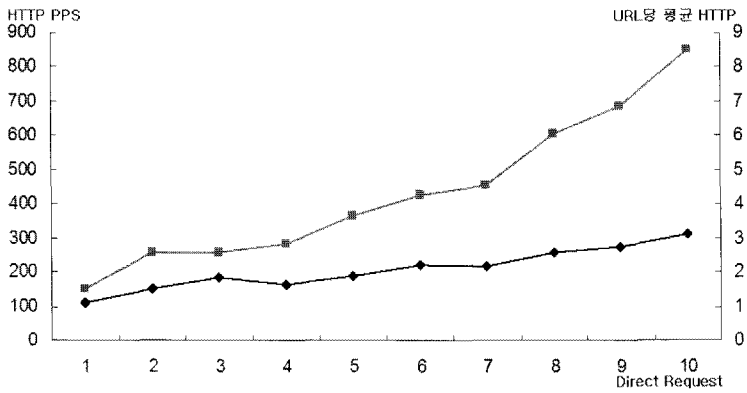
3.2.3 대응 알고리즘

URL 당 평균 HTTP에 대한 Threshold 기반 대응 알고리즘을 나타낸다.

- $w_{t,s}$: 특정 시간구간 t, IP s에서의 HTTP PPS
- $w_{T,s}$: 전체 시간구간 T에서의 $w_{t,s}$의 합
- n : 전체 시간구간 T동안 요청이 발생한 URL 개수
- $http_{uri,T}$: $w_{T,s}$ division to n for time interval T
- $threshold_2$: threshold of $http_{uri,T}$

3.3 Challenge 기반 DDoS 공격대응 메커니즘

앞서, 웹서비스 트래픽 패턴에 기반한 DDoS 공격



Direct Request (10sec)	HTTP	HTTP/URL
1	149.72	1.10
2	258.81	1.51
3	258.63	1.84
4	281.13	1.63
5	363.95	1.89
6	426.04	2.22
7	452.50	2.17
8	607.50	2.58
9	686.95	2.75
10	851.00	3.11

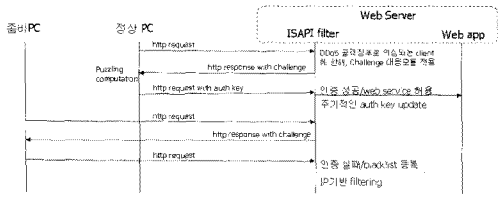
〔그림 7〕 HTTP 개수와 URL당 평균 HTTP 개수 상관관계

〔표 2〕 URL 당 평균 HTTP에 대한 Threshold 기반 대응 알고리즘

```
packet received, calculate w_{t,s}
while ( t != T ) {
  w_{T,s} = w_{T,s} + w_{t,s}
  calculate n
  t = t + 1
}
if http_{w_i,T} > threshold_2, source ip s is regarded
as Zombie Client and is blocked
  w_{T,s} = 0, w_{t,s}=0, t = 0, n = 0
```

대응 메커니즘을 제안하였다. 여기서는 의심스러운 트래픽을 발생시키는 Source IP를 대상으로 DDoS 공격여부를 판단할 수 있는 Challenge 기반 대응 메커니즘을 제안한다. Netbot Attacker, Blackenergy, DosHTTP, 7.7 DDoS 등 지금까지 발생했던 응용계층에 대한 모든 DDoS 공격은 대량의 HTTP Request를 발생시키거나 TCP Connection을 고갈시키는 형태로 발생했으나, 이들은 웹서버로부터의 Response는 모두 무시한다[11,12]. 그러나, 정상 웹브라우저의 경우 HTTP Request 후, 웹서버로부터 받은 HTTP Response를 토대로 Indirect Request가 발생한다. 따라서, 본 논문에서 제시하는 메커니즘은 HTTP Response에 Challenge를 삽입하고, 이후 발생하는 Request에 Challenge에 대한 적절한 Response 여부에 따라 정상 사용자의 트래픽과 좀비PC에 의한 트래픽을 구별한다. 〔그림 8〕은 Challenge 기반 DDoS 공격대응 절차를 나타낸다.

위 그림과 같이, 웹브라우저를 통한 정상 사용자의 HTTP Request를 수신한 웹서버는 Response의 payload에 Challenge 값을 클라이언트에게 보내고, 웹브라우저는 수신한 payload 내용을 해석하는 과정에서 Challenge에 대한 Authentication Key를 계산하고 HTTP Cookie에 삽입하여 redirection 형태로 HTTP Request를 다시 발생시키게



〔그림 8〕 Challenge 기반 DDoS 공격대응 절차

된다. 웹서버는 HTTP Cookie에 포함된 Authentication Key를 검증하여 DDoS 공격 발생여부를 탐지하게 된다. 지금까지 발생한 DDoS 공격도구는 웹서버의 Response를 무시하므로, 이후 발생하는 HTTP Request는 Authentication Key를 포함하고 있지 않게 되므로, 웹서버는 이를 탐지하고 해당 Source IP를 Blacklist 처리한다. 아래 표는 java script 중 Challenge 코드의 핵심부분을 나타낸다.

- S_t : Server의 현재시간
- T : Server가 Challenge 값을 보낼 때의 시간
- $[ABC]r$: 비교하고자 하는 자리 수(r=2일 경우, 하위 2자리(32bit)만 비교)
- S_{key} : Server측이 보유한 비밀 키
- $H(T,r,S_{key})$: T,r,S_{key}를 hash 한 값

현재 DDoS 공격방어기술 중 SYN Cookie 기법이 널리 사용되고 있는데, 이는 IP Spoofing을 통해 SYN Flooding 공격 등을 발생시키는 공격에 대응할 수 있다. 그러나, 위와 같이 정상적인 TCP 세션을 맺고, 발생하는 GET Flooding 공격에 대응할 수 없는 반면, Challenge 기반 대응 메커니즘은 대응 가능하다. 또한, HTTP Proxy를 사용하는 경우, 웹방화벽 등 보안기능을 제공하지만, 정상 사용자와 동일한 request에 대해 탐지 및 대응할 수 없지만,

〔표 3〕 Challenge 기반 DDoS 공격대응 알고리즘

```
Client receives T,r,H(T,r,S_key) from Challenge
Code in HTTP Response
Initializes X
while ([H(T,r,X)]r != [H(T,r,S_key)]r) Increase X
writes T,r,X,H(T,r,X) in HTTP Cookie
Client sends HTTP Request to Server
Server receives HTTP Request
if ( [H(T,r,X)]r = [H(T,r,S_key)]r) {
  if ( T < S_t - 60) start Reauthentication //
  Auth Key timeout
  else Source IP is regarded as Normal Client
  // 정상 Response
}
else if (no Authentication Key in HTTP
Cookie)
  Source IP is regarded as Zombie Client and
  blocked
  // No Auth Key
else Replay Attack may be generated
  // Wrong Auth Key
```

Challenge 방식은 대응 가능하다. Challenge 기반 대응 메커니즘은 모든 Source IP 대상 트래픽에 적용되는 것이 아니라, 앞서 Behavior 기반 대응 메커니즘에 의해 의심스러운 HTTP 트래픽 패턴을 보이는 Source IP에 한정하여 적용한다. 의심스러운 Source IP로부터 HTTP Request를 수신한 웹서버는 Challenge 기반 DDoS 대응 메커니즘을 시작한다. 이와 같이, 적용대상을 한정하여 성능 문제를 해결할 수 있다.

IV. 성능 분석

4.1 Implementation

대응 메커니즘 검증을 위해 DDoS 공격대응 시스템을 개발하였다. 네트워크 인터페이스 카드(NIC)를 통해 들어오는 패킷은 NIC 드라이버를 거쳐 NDIS(Network Driver Interface Specification)로 오게 된다. DDoS 공격대응을 위해 개발한 NDIS Intermediate 드라이버를 통해 패킷을 후킹하며 Packet Pool로 넘기고, 각 패킷에 대한 포인터는 NDIS Driver API Helper로 넘긴다. PacketShield Main Engine에서는 이 정보들을 토대로 패킷을 Layer 3, 4, 7에서 파싱하고, 필터링 처리한다. 실제 트래픽에 대한 제어는 PacketShield Main Engine에서 이루어진다. [그림 9]는 DDoS 공격대응시스템 아키텍처 및 패킷 파싱, 필터링 구조를 나타낸다.

4.2 Behavior 기반 대응 메커니즘 성능시험 결과

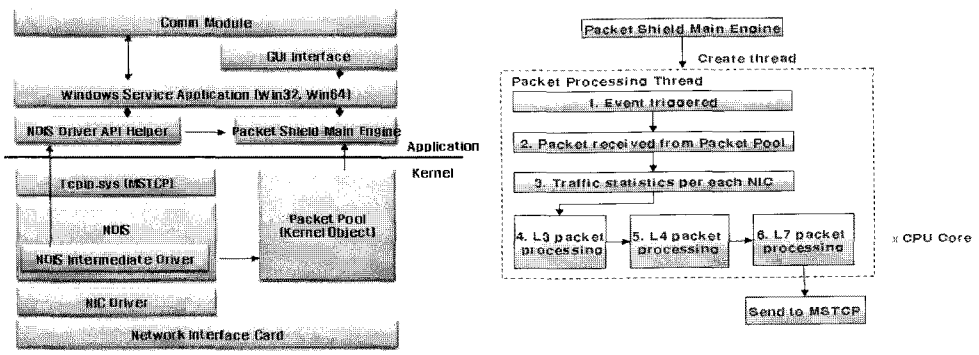
4.2.1 기존 DDoS 공격도구에 따른 성능평가

주요 DDoS 공격도구인 Netbot Attacker와 Blackenergy를 통한 GET Flooding 공격 발생 시, Behavior 기반 메커니즘을 적용했을 때와 적용하지 않았을 때의 CPU 이용률, 네트워크 이용률, 웹서버의 lifetime을 분석하였다. 이를 위해, 좀비PC 및 C&C 서버, 웹서버 등으로 구성된 테스트베드 환경을 구축하였다. [그림 10]은 Netbot Attacker는 좀비PC당 Thread를 5개, Blackenergy는 Thread를 10개로 설정하고, HTTP GET Flooding을 발생시키는 좀비PC의 수를 점차 증가시킬 때의 성능분석 결과를 나타낸다.

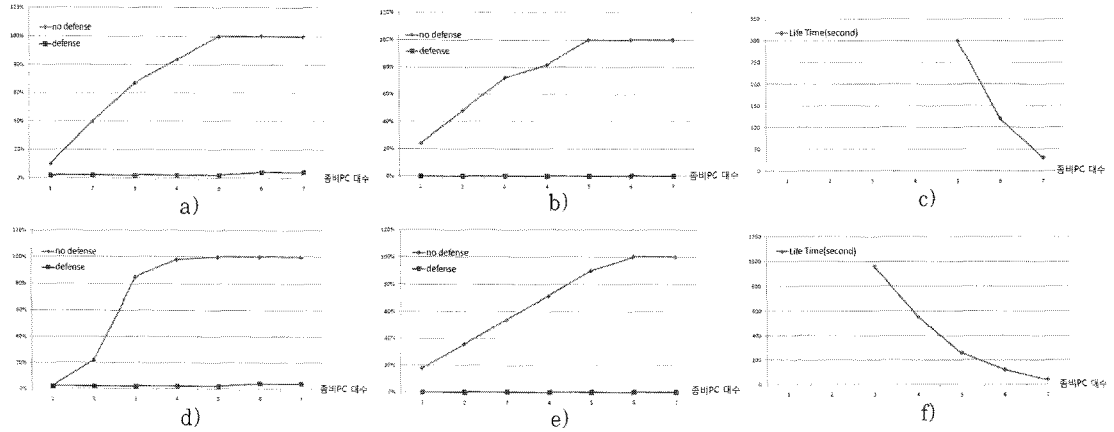
a),b),c)는 Netbot Attacker, d),e),f)는 Blackenergy를 통한 DDoS 공격발생 시, CPU, 네트워크 대역폭, 웹서버 lifetime을 나타낸다. Netbot Attacker의 경우, 대응 메커니즘이 동작하지 않을 때, CPU의 경우 좀비 PC 4~5대면 100%에 이르고(a), 네트워크 대역폭 역시 좀비 PC 5~6대면 100%를 이르고(b), 좀비PC가 7대의 경우 웹서버는 50여초 이상을 버티지 못한다(c). 그러나, Behavior 기반 DDoS 공격대응 메커니즘 적용 시, 안정적으로 동작함을 확인할 수 있다.

4.2.2 Behavior 기반 대응 메커니즘에 의한 성능평가 저하

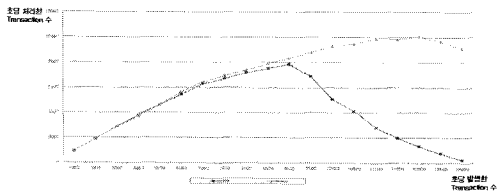
여기서는 Behavior 기반 대응 메커니즘을 적용할 때와 적용하지 않을 때의 성능저하가 얼마큼 발생하는지 분석한다. 이를 위해, Behavior 기반 대응 메커니즘은 모든 연산을 수행하지만, blacklist로 도출된 IP를 필터링하지 않을 때(모니터링 모드)에서의 성능저하 여부를 측정하였다. X축은 어플리케이션 성능측정 전문장비인 Avalanche 2900 장비를 통해 초당



[그림 9] DDoS 공격대응 시스템 아키텍처



〔그림 10〕 Netbot Attacker, Blackenergy를 통한 DDoS 공격발생 시, 성능평가



〔그림 11〕 Behavior 기반 대응모듈 Enable/Disable에서의 TPS 처리성능

발생시킨 Transaction 수를 나타내고, Y축은 Behavior 기반 대응모듈 Enable, Disable 상황에서 초당 처리한 Transaction 수를 나타낸다.

Behavior 기반 대응모듈을 동작하더라도 8,000 TPS(Transaction Per Second)까지 비슷하게 동작하며, Disable의 경우 최대 10,043 TPS까지 처리 가능한 반면, Enable의 경우 최대 7,848 TPS까지 처리 가능하다. 즉, 최대 Transaction 처리 성능 대비 78.14% 선에서 운영할 수 있다. 이는 모니터링 모드로 운영했을 때의 성능저하를 나타내므로, DDoS 공격정보에 대해 필터링 모드로 동작하게 되면 더욱 좋은 결과를 나타낼 것이다.

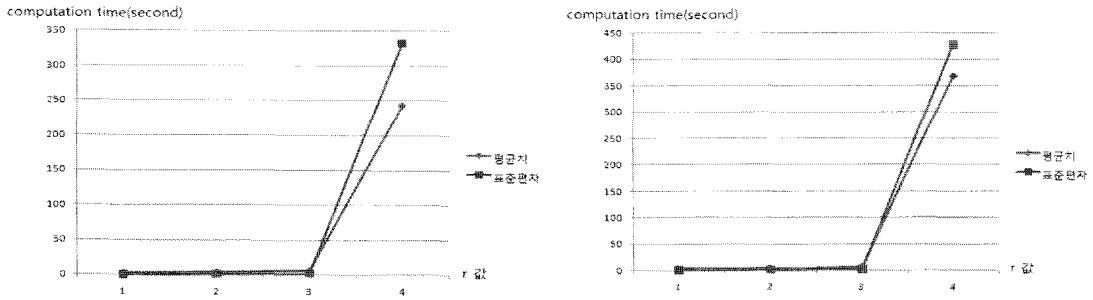
4.2.3 Challenge 기반 대응 메커니즘 성능시험 결과

Challenge 기반 DDoS 공격대응 메커니즘은 모든 Source IP에 대해 적용하지 않고 DDoS 공격으로 의심스러운 Source IP에 한정하여 적용함으로써, Challenge 공격대응 모듈에 따른 부하를 줄일 수 있다. Challenge 기반 대응모듈이 적용되는 Source IP는 Challenge에 따른 Authentication Key를 계산해야 하는데, 클라이언트가 받게 되는 부하는 r 값을 통해 정책적으로 조정할 수 있다. 〔표 7〕은 Internet Explorer 7에서 r 값에 따른 클라이언트의 Response Time을 나타낸다.

분석결과, r 값이 2, 3인 경우 각각 평균 3.16ms, 6.45ms이지만, r 값이 4가 될 경우 평균 242.67ms가 되는 등 기하급수적으로 증가함을 확인할 수 있다. 따라서, r 값을 3이하로 운영하면 정상 사용자의 성능 저하 없이 운영할 수 있다. 또한, Response Time은

〔표 7〕 Internet Explorer 7에서의 r에 따른 Response Time

r값	Response Time																					
	1차	2차	3차	4차	5차	6차	7차	8차	9차	10차	11차	12차	13차	14차	15차	16차	17차	18차	19차	20차	평균	표준편차
0자리	3.03	2.02	2.14	3.10	2.31	1.97	2.00	2.41	1.89	2.79	2.03	2.26	1.81	2.18	1.95	2.24	2.52	1.89	1.95	1.98	2.22	0.38
1자리	2.96	2.18	2.27	2.24	2.63	2.81	2.44	2.22	2.27	2.16	2.62	2.82	2.53	2.90	2.52	2.41	2.23	2.50	2.23	2.10	2.45	0.27
2자리	2.92	3.61	3.93	2.34	2.48	3.16	2.87	3.89	3.20	2.76	4.17	2.41	2.74	4.27	3.20	4.33	2.47	2.43	2.52	3.58	3.16	0.68
3자리	3.14	7.94	4.08	4.53	6.05	5.23	5.51	7.42	4.75	6.97	7.73	11.89	7.62	6.22	5.79	7.54	6.34	4.24	6.36	7.66	6.35	1.91
4자리	137.45	35.64	97.34	198.63	29.35	1000.0	109.43	85.20	207.52	21.31	115.48	1000.0	85.91	69.28	43.73	197.25	19.08	175.31	1000.0	225.54	242.67	332.71



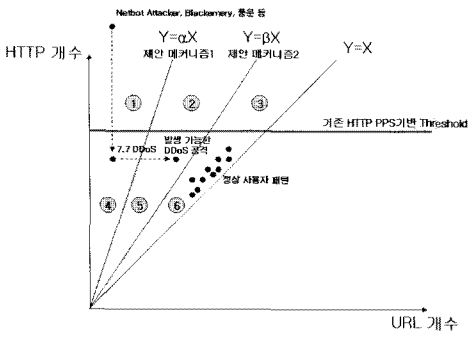
〔그림 12〕 PC 성능에 따른 Response Time

다양한 클라이언트 PC 성능에 따라 달라질 수 있다. 〔그림 12〕는 고사양 PC(Intel Core2 Duo CPU 3.0GHz, 3.37GB RAM)와 저사양 PC(Intel Core2 CPU 1.86GHz, 1.0GB RAM)에서 측정한 Response Time을 나타낸다.

분석 결과, PC의 사양과 관계없이 3이하의 r 값을 운영할 때 효과적으로 동작하는 반면, r 값 4에서는 고사양 PC와 저사양 PC에서 차이가 발생한다. 따라서, Challenge 기반 DDoS 공격대응 메커니즘 적용 시, r 값은 2~3정도로 조정하여 사용하면, PC 성능과 상관없이 성능 부하를 최소화하면서 좀비PC에 의한 DDoS 공격 트래픽을 차단할 수 있다.

V. 결 론

지금까지 웹서비스를 대상으로 발생하는 응용계층 DDoS 공격에 대한 Direct Request 기반 Threshold, URL 당 평균 HTTP 기반 Threshold, Challenge 기반 대응 메커니즘을 제안하고, 성능분석 결과를 제시하였다. 〔그림 13〕은 DDoS 공격유형 및 본 논문에서 제안된 메커니즘을 통해 대응 가능한 영역을 나타낸다.



〔그림 13〕 DDoS 공격유형 및 대응가능 영역

X축은 Source IP별 10초간 요청한 URL 개수를 나타내고, Y축은 해당 시간동안 발생한 HTTP 개수를 나타낸다. 따라서, Source IP별 10초 단위마다 정상 및 DDoS 공격의 모든 트래픽 패턴은 하나의 점으로 표현된다. X축은 요청한 URL 개수를 나타내므로 $Y \leq X$ 영역에는 데이터가 존재할 수 없다. 기존 HTTP에 대한 Threshold 기반 대응 메커니즘은 URL 개수는 고려하지 않으므로, ①+②+③ 영역의 트래픽 유형에 대해서는 DDoS 공격으로 탐지 및 차단할 수 있다. 이를 우회할 수 있는 트래픽은 ④+⑤+⑥ 영역에 존재할 수 있는데, 본 논문에서 제안한 Direct Request 기반 Threshold 대응기법은 ④ 영역을 대응할 수 있고, URL당 평균 HTTP Request에 대한 대응기법은 ④+⑤ 영역을 탐지 및 대응할 수 있다. Challenge 기반 대응기법은 의심스러운 트래픽 패턴을 보이는 ④+⑤ 영역 뿐 아니라, 정상 사용자 패턴에 해당하는 ⑥ 영역에서도 DDoS 공격발생 여부를 탐지할 수 있다. α는 $\frac{Y}{X}$으로 URL 당 평균 HTTP Request 수를 의미하므로, 각 URL 별 HTTP Request들 중 α이상의 Request 수를 기록한 경우가 존재한다. 따라서, Direct Request에 대한 Threshold가 α라면, $Y \geq \alpha X$에 해당하는 최소한 ①+④ 영역은 DDoS 공격으로 탐지 및 대응할 수 있다. 앞서, 웹서비스 이용패턴 분석 결과, 앞서 기술한 바와 같이 Threshold를 10으로 설정할 경우 $Y \geq 10X$ 영역에 대해 대응할 수 있다. β 역시 URL 당 평균 HTTP Request 수를 의미하므로 $\beta = \frac{\text{전체 } HTTP \text{개수}}{\text{요청된 } URL \text{개수}}$이며, $Y \geq \beta X$에 해당하는 ①+②+④+⑤ 영역은 DDoS 공격으로 탐지 및 대응할 수 있다. 앞서, 웹서비스 이용패턴 분석 결과, URL 당 평균 HTTP에 대한 Threshold를 3으로 설정할 수 있으므로, $Y \geq 3X$ 영역에 대해 대응할 수 있다. 일반

적인 웹서핑 사용자 대부분의 트래픽은 10초당 Direct Request 개수가 2이하인 경우가 89.4%이므로 $Y \geq X \sim Y \geq 1.5X$ 사이에 존재하게 되며, 웹브라우저 refresh 등을 통해 고의적으로 트래픽을 많이 발생시킬 경우, $Y \geq 2X \sim Y \geq 3X$ 영역에 위치하게 된다.

최근까지 큰 피해를 발생시킨 Netbot Attacker, Blackenergy, 풍운, DoSHTTP 등의 DDoS 공격 도구는 특정 URL에 대해 Get Flooding을 발생시키므로, 주로 ①영역에 위치하며, 7.7 DDoS 공격의 20 PPS 이하로 특정 URL을 대상으로 발생하므로, ④영역에 위치한다. Netbot Attacker 5.5 버전이상에 존재하는 Circle CC Attack의 경우, 사전에 설정된 URL들을 대상으로 DDoS 공격이 발생하므로, ③영역에 해당하여 탐지 및 대응이 가능하다. 응용계층에 대한 DDoS 공격은 점차 정상 사용자와 유사한 패턴으로 발전하고 있으므로, 향후 DDoS 공격은 ①⇒④⇒⑤⇒⑥영역으로 발전할 것으로 예상된다. 본 논문에서 제안한 메커니즘은 ④, ⑤ 영역을 탐지 및 대응할 수 있고, ⑥ 패턴에 해당되더라도 Challenge 기반 대응 메커니즘을 통해 대응할 수 있다. 본 논문에서 제안한 메커니즘은 간단하면서도 강력한 성능을 내므로, 상용 환경에 바로 적용할 수 있을 것으로 예상된다. 최근, DDoS 공격은 점차 정상 사용자의 유사한 형태로 발전하고 있으므로, 본 논문에서 제안한 메커니즘을 더욱 발전시켜 정교한 DDoS 공격에 대응할 수 있어야 할 것이며, 본 논문에서 다루지 않은 TCP Connection을 고갈시키는 DDoS 공격유형에 대한 분석이 추후 연구되어야 한다.

참 고 문 헌

[1] Alefiya Hussain, John Heidemann, Christos Papadoopoulos, "A Framework for Classifying Denial of Service Attacks", SIGCOMM pp. 99-110 , 2003.

[2] Yi Xie and Shun-Zheng Yu, "Monitoring the Application-Layer DDoS Attacks for Popular Websites", IEEE/ACM Transactions on Networking, vol. 17 issue. 1, pp.15-25 , Feb. 2009

[3] Song Huang, Ling Zhang, Shou-Ling Dong, "A Behavior-based Ingress Rate-Limiting Mechanisms against DoS/DDoS Attacks", LNCS, vol. 3783, pp. 231-242 , 2005.

[4] 최해권, 네트워크 보안위협에서 DDoS 공격분석 및 차단방안에 관한 연구, 석사학위논문, 전북대학교, 2008년 2월

[5] http://sniper.nowcom.co.kr/?page_no=38

[6] http://www.radware.com/Thank_you_download.aspx?ID=6949

[7] Takeshi Yatagai, "Detection of HTTP GET Flood Attack Based on Analysis of Page Access Behavior", PACRIM pp232-235, 2007

[8] M. Srivatsa et al., "Mitigating Application Level Denial of Service Attacks on Web Servers", ACM Transactions on WEB, Vol. 2 Issue. 3, July. 2008

[9] HTTP, http://www.ietf.org/rfc/rfc2616.txt

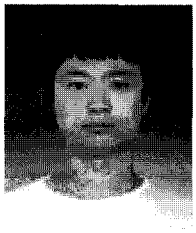
[10] TCP, http://www.faqs.org/rfcs/rfc793.html

[11] http://www.nbddos.com/

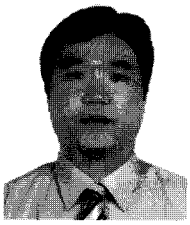
[12] Jose Nazario, "BlackEnergy DDoS Bot Analysis", Arbor Networks, Nov. 2007

〈著者紹介〉

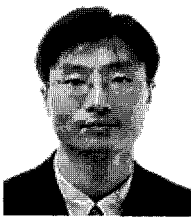
이 태 진 (Tai-jin Lee) 정회원
2003년 2월: POSTECH 컴퓨터공학과 졸업
2008년 2월: 연세대 컴퓨터공학과 석사 졸업
2003년 1월~현재: 한국인터넷진흥원 선임연구원
〈관심분야〉 네트워크 보안, VoIP 보안, 스마트그리드



임 채 수 (Chae-su Im)
2010년 2월: 원광대학교 정보전자상거래학부 졸업
2009년 9월~현재: 한국인터넷진흥원 주임연구원
〈관심분야〉 정보보호, 네트워크 보안



임 채 태 (Chae-tae Im) 정회원
2000년 8월: 충남대학교 컴퓨터과학과 졸업
2003년 2월: POSTECH 컴퓨터공학과 석사 졸업
2009년 2월~ 전남대학교 정보보호협동과정 박사과정
2003년 1월~ 한국인터넷진흥원 융합보호R&D팀 파트장
〈관심분야〉 악성코드, 네트워크보안, VoIP 보안, 이동통신



정 현 철 (Hyun-chul Jung) 정회원
1996년 2월: 서울시립대학교 전산통계학과 졸업
1999년 8월: 광운대학교 전자계산학과 석사 졸업
1996년 7월~ 현재: 한국인터넷진흥원 팀장
〈관심분야〉 네트워크보안, 침해사고대응, 포렌식