# Active Timeout을 이용한 SYN Flooding 공격의 해결

서정석° 차성덕
한국과학기술원 전자전산학과
{jsseo°, cha}@salmosa.kaist.ac.kr

# Defending against SYN Flooding Attacks based on Active Timeout

Jeongseok Seo° Sungdeok Cha
Div. of Computer Science, Dept. of EECS, KAIST and AITrc/IIRTRC/SPIC

**요 약**

서비스 거부(denial of service) 공격의 일종인 SYN flooding 공격은 TCP/IP 프로토콜의 오류로 인해 발생한다. 기존의 SYN flooding 해결 연구들은 대부분 방화벽이나 라우터에서 패킷을 감시하여 불법적으로 판단된 패킷을 걸러내는 방법을 사용하였다. 따라서 기존의 연구들은 방화벽이나 라우터에 많은 부하를 주게 된다. 본 연구에서는 방화벽이나 라우터의 도움을 받지 않고, 기존의 네트워크 환경이나 운영체제에 큰 변화를 가하지 않으면서, 서버 시스템 자체만으로 SYN flooding 공격에 효율적으로 대응할 수 있는 방법을 제시하고자 한다.

## 1. 서 론

SYN flooding 공격의 개념은 소개된 지 오래되었지만, 그 원인이 시스템의 취약성에 바탕을 둔 것이 아니라 TCP/IP 프로토콜 오류에 있기 때문에 여전히 그 취약점은 해결되지 않은 채 남아있다[3]. 더욱이 SYN flooding 공격은 단순히 서비스 거부 공격 자체로 사용되기 보다는 다른 네트워크 공격들을 시도하는 중간과정을 돕기 위해 많이 사용된다[1]. 예를 들면, IP spoofing 공격에서 spoofing하는 클라이언트의 응답을 마비시키기 위하여 SYN flooding 기법을 사용하며, source IP가 위조된 SYN flooding 공격을 통해 IDS에 무수히 많은 false alarm을 내거나 패킷 손실이 일어나도록 한 뒤 실제 공격을 삽입하여 IDS를 회피하는 등의 다양한 방법으로 사용되기 때문에, SYN flooding 공격을 해결하는 문제는 단순히 DOS 공격의 한 가지 문제를 해결하는 것뿐만 아니라 다양한 네트워크 공격들을 해결하는데 도움을 줄 수 있는 기회를 제공하는 일이 된다.

본 연구에서는 backlog 큐의 사용량에 따라 timeout을 능동적으로 조절(active timeout)하여 SYN flooding 공격을 효과적으로 해결하고자 한다.

## 2. SYN Flooding 공격과 기존의 해결 방법들

그림 1과 같이 위조된 IP 주소를 가진 패킷이 대량으로 SYN 요청을 하게 되면, 서버는 이 요청을 모두 backlog 큐에 저장하고 클라이언트로부터 ACK를 기다리게 된다. 위조된 IP 주소의 호스트가 네트워크에 존재하지 않다면 서버는 timeout이 발생할 때까지 backlog 큐에 세션 정보를 유지하고, 만약 큐가 가득 차게 되면 더 이상의 서비스 요구를 받아들이지 못하고 요청되는 모든 서비스를 거부하게 된다.

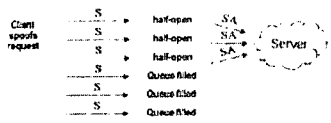


그림 1 SYN flooding 공격

결국 이 공격은 TCP/IP 프로토콜에서 source IP 주소를 임의로 위조를 할 수 있고, 목적지 호스트는 이를 판별할 수 없기 때문에 발생하며, 서버는 개별적인 서비스 요청만으로 SYN flooding 공격을 판별할 수 있는 방법이 없다. 그러나 여러 가지 근거를 종합적으로 추론하여 과도하게 SYN 요청이 발생하는 공격을 판별할 수 있다.

SYN flooding 문제를 해결하기 위한 좋은 방법은 다음과 같은 특징들을 가지고 있어야 한다.

- 기존의 OS나 네트워크 구조를 변경하지 않고 기존의 다양한 OS 시스템에 적용 가능해야 한다.
- 기존의 TCP/IP 프로토콜과 완벽하게 호환해야 한다.
- 부가적인 하드웨어 장치 없이 간단히 시스템에 적용되어 효과적으로 공격에 대응할 수 있어야 한다.
- 확장성이 좋고, 적용되는 시스템에 따라 유동적으로 파라미터를 조절할 수 있어야 한다.

SYN flooding 공격을 해결하기 위한 기존의 방법으로는 OS 패치, 주소 검증(address verification), SYN 감시 기법 등이 존재한다. 가장 널리 이루어지는 방법은 OS 패치를 통하여 backlog 큐의 크기를 늘리고 timeout 시간을 짧게 만드는 것이다[9, 2]. 최근 redhat 8.0, 9.0의 경우, backlog 큐의 크기를 1024로 확장하고 timeout을 10초로 줄일 것을 권장하고 있다. 하지만 이 방법이 근

본적인 해결책은 아니다. 왜냐하면 지속적으로 많은 SYN flooding 공격을 당하게 될 경우, 결국 backlog 큐가 가득 차게 되며, timeout을 줄이면 정상적인 사용자가 round-trip 시간이 timeout보다 긴 네트워크를 - 예를 들면, PPP, SLIP 등 - 사용할 때는 적합한 해결책이 될 수 없다. 더욱이 최근과 같이 네트워크 전송량이 크게 증가하면 더욱 큰 backlog 큐를 사용해야 하므로 효과적인 해결 방법이라고 할 수 없다.

일반적으로 SYN flooding 공격자는 탐지를 회피하기 위해 공격 패킷의 source IP 주소를 속이게 된다. 이 때 네트워크에서 잘못된 source 주소를 가진 패킷을 걸러내어 공격 성공률을 낮출 수 있다[4, 5]. 그러나 이 방법은 ISP(Internet Service Provider) 등과 같이 큰 규모의 네트워크를 관리하는 곳에서 적용가능한 단점이 있다.

마지막으로 서버의 앞단에 존재하는 라우터나 방화벽에서 SYN flooding 공격을 막기 위해 intercepting mode나 watching mode를 제공할 수 있다[6, 8]. 이 두 가지 방법은 라우터나 방화벽이 서비스 전체 세션을 감시하고 비정상적인 세션을 탐지하여 SYN flooding을 차단하게 된다. 그러나 전체 세션을 감시하고 공격자로 판별된 리스트를 관리해야 하는 부담이 따르므로 라우터와 방화벽에 많은 리소스와 비용을 요구하게 된다.

표 1 관련 연구들의 비교

| 관련 연구 | 장 점 | 단 점 |
|---|---|---|
| OS 패치 | ◆ 가장 간단한 방법 | ◆ 큰 용량의 kernel 메모리 사용<br>◆ 느린 네트워크 서비스 봉가 |
| 주소 검증 | ◆ 비정상적인 주소의 패킷 차단 | ◆ 대규모의 ISP에서 적용 가능 |
| SYN 감시 | ◆ 가장 능동적으로 공격을 차단 | ◆ 라우터나 방화벽의 리소스를 사용하여 많은 부하를 줌 |

## 3. 실험 환경

본 연구의 실험에서는 SYN flooding 공격 툴로 C언어로 작성된 SYN flooder를 사용하였다[7]. 표 2는 SYN flooder가 10초당 생성하는 공격 패킷 개수를 통계 낸 것이다. 표 2를 통해 예상할 수 있듯이 100Mbps 서브넷 환경에서 실제 1개의 SYN flooder를 이용하여 wowlinux 7.1(256 backlog)의 웹 서비스를 완전히 마비시켰으며, redhat 9.0(1024 backlog)은 3개의 SYN flooder 분산 공격을 통해 웹 서비스를 완전히 마비시킬 수 있었다. 결국 SYN flooding 공격을 해결하기 위해 단순히 큐의 크기를 늘리고 timeout을 줄이는 것은 한계가 있다. 다음 장에서 효과적인 서비스 제공을 위한 backlog 큐의 크기에 대해 알아보자.

표 2 SYN flooder의 공격 분석 (100Mbps Ethernet)

| SYN flooder 개수 | 공격 패킷 개수 (10초) | 공격 패킷 간의 time interval |
|---|---|---|
| 1개 | 490.5 개 | 20.3 ms |
| 2개 | 977.3 개 | 10.2 ms |
| 3개 | 1478.7 개 | 6.8 ms |
| 4개 | 1933.2 개 | 5.2 ms |
| 5개 | 2475.7 개 | 4.0 ms |

## 4. 적절한 Backlog 큐의 크기

원활한 서비스를 위한 적절한 backlog 큐의 크기를 알아보자. 시스템의 backlog 큐의 크기를 Q라하고, timeout을 T(sec), SYN flooding 공격 패킷의 평균 time interval을 t(sec), 정상적인 서비스 요청 시간을 각각 $n_1, n_2, \ldots\ldots$, 이 때 해당 서비스를 요청하는 클라이언트와의 round-trip 시간을 각각 $rtt_1, rtt_2, \ldots\ldots$라 하자. 총 N개의 정상 서비스 요청을 받았을 때까지 SYN 공격 아래에서 원활한 서비스를 가능하게 하기위하여 필요한 backlog 큐의 크기는 다음과 같이 구해질 수 있다.

$$Q = \left[\frac{T}{t}+1\right] + \left[\frac{\frac{\sum_{i=1}^{N} rtt_i}{N}}{\frac{(n_2-n_1)+(n_3-n_2)+\ldots+(n_N-n_{N-1})}{N}}+1\right]$$

$$= \left[\frac{T}{t}\right] + \left[\left(\sum_{i=1}^{N} rtt_i\right)/(n_N-n_1)\right]+2 \qquad \text{----------(1)}$$

[A] = {a | a is a maximum integer which is not greater than A}

식 (1)에서 $[T/t+1]$은 SYN flooding 공격을 견뎌내기 위한 backlog 큐의 크기이고, $\left[\left(\sum_{i=1}^{N} rtt_i\right)/(n_N-n_1)+1\right]$은 원활한 서비스를 위해 필요한 backlog 큐의 크기이다. 처음 SYN flooding 공격이 발생한 후 timeout이 발생할 때까지 backlog 큐가 가득 차는 것을 막을 수 있으면 SYN flooding 공격에 의해 서비스가 마비되는 현상을 막을 수 있다. 예를 들어 redhat 9.0 시스템이 5개 SYN flooder의 공격을 받고 있다면, redhat 9.0의 timeout T =10(sec)이고, 앞의 표 2에 의해 공격 패킷의 time interval t=0.004(sec)이다. 또 정상 서비스 요청은 대략 100ms당 한 번 발생하며, 해당 클라이언트까지 round-trip 시간이 145ms(yahoo.com의 예)이라고 가정하였을 때, 이 시스템의 원활한 서비스를 위해 필요한 backlog 큐의 크기는 다음과 같다.

$$Q = \left[\frac{10}{0.004}\right] + \left[\left(\sum_{i=1}^{101}(0.145)\right)/(10-0)\right]+2 = 2503 \qquad \text{-----(2)}$$

결국 현재 대부분 OS들의 backlog 큐 크기는 SYN flooding 공격을 근본적으로 막기 부족하며, 서비스 특징에 따라 적절한 backlog 큐 크기를 결정하는 것이 매우

중요하다. 결국 backlog 큐 크기는 네트워크의 전송량과 공격 발생률에 따라서 결정되는데, 큐의 크기가 크면 시스템 성능에 많은 영향을 미치는 커널 메모리의 낭비를 의미할 뿐만 아니라, 크기가 큰 큐를 관리하기 위한 성능 저하도 발생한다. 더욱 최근 giga-bit 이상의 높은 대역폭의 네트워크를 사용하는 서버들은 대규모의 분산 공격을 막기 위해 훨씬 큰 backlog 큐를 가져야 하며, 이는 엄청난 커널 메모리의 낭비를 초래한다. 따라서 본 논문에서는 이를 해결하기 위해 backlog 큐의 사용량에 따라 timeout을 능동적으로 조절하여 적절한 서비스 가용성을 제공할 수 있는 간단하고 효율적인 알고리즘을 제시하고자 한다.

## 5. Active Timeout 알고리즘과 실험

Timeout을 아래 식 (3)과 같이 사용 중인 큐의 크기 $Q_c$에 따라 결정해보자. 여기서 초기 큐의 크기 Q와 threshold $t_c$는 미리 결정되는 변수이다. 이 때 정상 서비스 성공률 S는 아래와 같이 계산될 수 있다.

$$Timeout = \begin{cases} 75 & , Q_c \le t_c \\ -\frac{75}{Q-t_c}(Q_c - Q) & , t_c < Q_c \le Q \end{cases} \quad ---(3)$$

$Q_c : current\ filled\ queue, Q : initial\ queue, Q \neq t_c(threshold)$

$$Service\ rate\ S = \frac{number\ of\ serviced\ request}{total\ normal\ service\ request} \times 100(\%)$$

알고리즘 성능을 확인하기 위해 redhat 9.0(Q=1024) 커널을 수정(timeout=75초)하였다. 2개의 SYN flooder를 통해 공격 패킷을 생성하고, 1초당 10개의 서비스 요청을 하였으며 실험에 사용한 서비스 요청 클라이언트의 round-trip 시간은 10~250ms까지 임의의 값을 가지도록 실험하였다. 아래 표 3은 실험결과를 보여준다.

결과를 보면 active timeout을 적용하지 않은 서비스보다 알고리즘을 적용한 서비스의 성공률이 4배 이상으로 좋은 결과를 보이고 있다. Threshold 값을 작게 할수록 서비스 성공률이 높아지는데, 이는 backlog 큐가 가득 차기 전에 미리 알고리즘을 적용하였기 때문이다. 그러나 round-trip 시간보다 큐의 timeout이 짧아서 서비스가 거부되는 비율은 거의 변화가 없는데, 이는 너무 짧은 timeout에 의해 서비스가 거부되는 현상이 backlog 큐가 가득 차는 시점에 나타나기 때문이며 알고리즘이 적용되는 시점과는 거의 무관하다는 것을 의미한다.

표 3 Active timeout 적용 시 서비스 성공률

| Threshold ($t_c$) | Service rate (S) | 큐가 가득차서 실패한 서비스 비율 | 짧은 timeout으로 실패한 서비스 비율 |
|---|---|---|---|
| 1024(*) | 20.45% | 79.55% | 0% |
| 50 | 94.90% | 2.07% | 3.03% |
| 100 | 94.30% | 2.80% | 2.90% |
| 200 | 93.50% | 3.50% | 3.10% |
| 500 | 89.40% | 7.50% | 3.20% |

(*) $Q=t_c$: active timeout을 적용하지 않았음을 의미

## 6. 결론 및 향후 연구

이 연구는 기존의 연구들과 달리 인터넷 서비스 특징과 네트워크 상황을 반영하여 능동적으로 SYN flooding 공격에 대응할 수 있는 장점이 있다. 또한 라우터나 방화벽 등의 다른 보조적인 도움 없이 간단한 커널 알고리즘을 통해 매우 효과적으로 SYN flooding을 대처하는데 유용하며, 기존의 SYN flooding을 이용한 여러 공격을 해결하는데 좋은 보안 도구가 될 것이다.

성능 개선을 위하여 해시함수를 통해 SYN flooding 공격으로 판명된 IP 주소들을 관리하고 다음 서비스 요청에 패널티를 주는 알고리즘을 같이 적용해 보았으나 active timeout 알고리즘에 비해 복잡하면서 성능은 크게 좋아지지 않았다. 오히려 해시 테이블 관리에 리소스를 사용하므로 간단한 알고리즘의 성능이 더 좋아보였다. 본 연구에서는 적합한 backlog 큐의 크기가 식 (1)에 의해 결정되면 시스템 운용 중에는 큐의 크기를 변경할 수 없다는 단점이 있다. 향후 연구로는 인터넷 서비스의 특성과 서비스 요청 빈도 분석을 통하여 최적의 backlog 큐 크기와 timeout을 능동적으로 적용하는 연구가 많은 도움이 될 것이다.

## 7. 참고 문헌


[1] The Kevin MItnick/Tsutomu Shimomura affair. http://www.gulker.com/ra/hack/

[2] The phrack magazine, issue 48, Sep. 1, 1996. http://www.phrack.org

[3] CERT/CC: Computer Emergency Response Team Coordination Center (Reporting Center for Internet Security Problem). CERT Advisory CA-1996-21.

[4] Cisco Systems Inc. "Defining Strategies to Protect Against TCP SYN DoS Attacks," September 1996.

[5] P. Ferguson, "Network ingress filtering," Internet draft, Cisco Systems Inc. September 1996.

[6] Jim Phillipo, "Preventing SYN Flooding with Cisco Routers," SANS Intrusion Detection, Sep. 6, 2000.

[7] Exploits and Tools, "Digital Information Society," Phreak, http://www.phreak.org/html.

[8] P.Ferguson. Network ingress filtering. Internet draft, Cisco System, Inc. September 1996.

[9] M.Graff. Sun Security Bulletin 00136. Mountain View, CA, Oct. 1996.