SYN Flood DoS 공격을 차단하기 위한 확장 TCP

(The Extended TCP for Preventing from SYN Flood DoS Attacks)

박 진 원 † 김 명 균 ††
(Zin-Won Park) (Myung-Kyun Kim)

요 약 서비스 거부 공격이란 서비스를 제공하는데 있어 필요한 컴퓨팅 및 통신 자원을 고갈시키는 공격으로 원천적으로 해결하기가 매우 힘든 것으로 알려져 있다. TCP는 연결 설정 과정에 있어 서비스 거부 공격을 당할 수 있는 문제점을 가지고 있다. TCP는 연결 설정이 완료될 때까지 각 연결 설정 단계의 상태를 큐에 저장하고 있다가 연결 설정이 완료되면 연결된 소켓을 어플리케이션으로 전달한다. 공격자는 수많은 연결 요청을 보내고 이 요청에 대한 연결 과정을 완료하지 않음으로 해서 이 큐를 가득 차게 해 다른 정상적인 연결 요청을 받아들일 수 없도록 할 수 있다. 본 논문에서는 이러한 서비스 거부 공격을 차단하기 위한 확장 TCP를 제안하고 이를 리눅스 상에서 구현하였다. 제안한 확장 TCP는 연결 설정이 종료될 때까지는 연결 설정 과정에 대한 상태를 큐에 유지하지 않도록 함으로써 서비스 거부 공격을 막을 수 있도록 하였다. 제안된 확장 TCP를 위해 TCP의 3-way handshake 과정을 일부 수정하고 이를 리눅스 커널에 구현하였으며 그 성능을 실험해 본 결과 정상적인 서비스 환경에서는 수정전의 TCP의 연결 처리 속도에 비해 0.05%정도의 지연이 있었지만 SYN Flood 공격이 이루어지고 있는 상황에서는 아무런 영향을 받지 않았다.

키워드 : TCP, SYN Flood, DoS, 서비스 거부, 보안, 리눅스 시스템

Abstract The Denial of Service(DoS) attacks, which are done by consuming all of the computing or communication resources necessary for the services, are known very difficult to be protected from. TCP has drawbacks in its connection establishment for possible DoS attacks. TCP maintains the state of each partly established connection in the connection queue until it is established completely and accepted by the application. The attackers can make the queue full by sending connection requests repeatedly and not completing the connection establishment steps for those requests. In this paper, we have designed and implemented the extended TCP for preventing from SYN Flood DoS attacks. In the extended TCP, the state of each partly established connection is not maintained in the queue until the connection is established completely. For the extended TCP, we have modified the 3-way handshake procedure of TCP, and implemented the extended TCP in the Linux operating system. The test result shows 0.05% delay more than original TCP, but it shows that the extended TCP is strong for SYN Flood attacks.

Key words : TCP, SYN Flood, DoS, Denial of Service, Security, Linux system

1. 서 론

서비스 거부 공격은 최근 들어 매우 위험하고 치명적인 공격 형태인 것으로 분류되고 있다. 서비스 거부 공격은 사용자가 서버에 접속하여 서비스를 이용하는 것을 방해하는 공격으로 인터넷 쇼핑몰이나 포털 사이트, 온라인 증권 거래 사이트 같은 인터넷 서비스 사업자의 경우 서비스 거부 공격에 대한 피해는 막대한 피해액으로 나타나 많은 손실을 입게 될 수 있다. 한 예로, 지난 2000년 야후(yahoo.com)와 eBay(ebay.com)에 대한 서비스 거부 공격을 들 수 있다.

서비스 거부 공격 중 하나인 SYN Flood 공격은 TCP의 3-way handshake 과정에의 취약점을 이용한

· 본 연구는 산업자원부 지정 울산대학교 네트워크 기반 자동화연구센터 및 울산대학교 교내연구비의 지원에 의한 것입니다.
† 학생회원 : 울산대학교 컴퓨터 · 정보통신공학부
zwsonic@shinbiro.com
†† 종신회원 : 울산대학교 컴퓨터 · 정보통신공학부 교수
mkkim@ulsan.ac.kr
논문접수 : 2004년 10월 28일
심사완료 : 2005년 6월 10일

다. 서버는 클라이언트로부터 연결 요청을 받고 그 정보를 큐에 저장하는데, 공격자는 이 큐를 가득 채워서 더 이상의 연결 요청을 받아들일 수 없게 만드는 것이다. 이 공격에서 사용하는 연결 요청 과정은 정상적인 연결 요청과 구분하기가 힘들고 TCP 자체의 취약점을 이용한 것이기 때문에 예방 또한 어렵다. 뿐만 아니라 공격자는 송신자의 IP 주소를 무작위로 선택된 IP 주소로 위조하여 공격을 할 수 있기 때문에 공격자에 대한 차단도 힘들다.

본 논문에서는 이러한 서비스 거부 공격을 원천적으로 차단할 수 있도록 TCP를 설계하였고 리눅스 커널의 TCP를 수정하여 구현하였다. 이것은 클라이언트의 TCP 구조를 변경하지 않고 적용이 가능하기 때문에 실제 환경에서도 서비스 거부에 대한 예방이 가능하다.

본 논문의 순서로는 2장에서는 기존 연구들에 대해서 알아본 다음 3장에서는 SYN Flood 공격의 원리 및 방법을 설명한다. 4장에서는 SYN Flood 공격을 예방할 수 있는 TCP의 구조에 대한 설명을 하고 5장에서는 리눅스 커널에 구현한 확장 TCP의 동작 방법에 대해서 기술한다. 6장에서는 SYN Flood 공격이 이루어지고 있는 상황에서의 성능을 시험해 보고 보안성을 평가한다. 7장에서는 결론을 맺고 향후 연구 방향에 대해서 언급한다.

2. 기존연구

SYN Flood 공격은 TCP의 연결 설정 과정을 그대로 따르며 많은 운영 체제에서 TCP를 구현하는데 필요한 과정을 악용한 공격이기 때문에 이 공격으로부터 완전히 해방되는 방법은 없는 것이 현실이다[1]. 이러한 SYN Flood공격에 대한 심각성과 그 문제점은 오래 전부터 알려져 왔으며 이에 대한 많은 연구들이 진행되고 있다. 본 절에서는 대표적인 몇 가지의 연구에 대해서만 알아보기로 한다.

[2]에서는 방화벽을 이용한 릴레이 방법을 소개하였는데, 이는 공격자가 방화벽에 대한 SYN Flood 공격을 할 수 있으므로 여전히 문제점을 가지고 있다. [3]에서는 패킷을 분류를 통한 공격을 구분하는 방법을 제시하였는데, 이 방법은 TCP의 근본적인 문제는 해결하지 못할 뿐만 아니라 일시적인 서비스 거부 상태가 될 수 있다는 문제점이 있으며 많은 수의 백로그 큐를 필요로 한다는 문제점이 있다. 이와 유사한 방법으로 [1,4]에서는 별도의 침입 탐지 시스템이나 침입 차단 시스템, 모니터링 시스템을 이용하여 서버의 상태를 점검하며 서비스 거부 공격을 탐지하는 방법들을 제시하고 있다. [2,5]에서는 SYN Flood 공격에 대한 피해를 최소화하기 위하여 백로그 큐의 크기를 크게 설정하고 백로그 큐에 남아있는 엔트리를 제거하는데 걸리는 타임아웃 시간을 줄이는 방법을 제시하였다. 하지만 이 방법은 백로그 큐를 무한으로 늘일 수 없을 뿐만 아니라 백로그 큐를 크게 설정할수록 정상적인 연결을 설정하는데 지연 시간이 커지게 되고 타임아웃 값을 줄임으로 해서 네트워크 환경이 열악한 클라이언트는 연결 설정을 할 수 없을 수도 있다는 문제점을 가지고 있다.

[5]에서는 FreeBSD에 구현한 SYN cache 기법을 제시하였다. 기존의 운영체제는 연결 정보를 저장하기 위하여 큰 크기의 큐를 필요로 했지만 이 방법은 최소의 정보만을 큐에 저장함으로써 작은 크기의 큐 엔트리를 이용하여 많은 연결 요청을 저장할 수 있는 방법이다. 하지만 이 방법은 연결 정보를 저장하기 때문에 다량의 공격 패킷을 이용하여 큐를 오버플로우 시키는 공격이 가능하며, 이 방법 역시 SYN Flood 공격에 대해 완전히 해방될 수는 없다.

리눅스 커널 2.2 이후의 버전에서는 SYN Flood 공격에 대한 SYN cookies라는 기법을 제공한다. SYN cookies는 클라이언트가 연결 요청을 보내오면 서버는 이에 대한 응답 패킷에 서버만이 생성할 수 있는 24비트 해쉬값을 생성하고 클라이언트가 연결 요청 시 전송한 MSS값을 인코딩한 값 3비트와 해쉬할 때 사용한 서버의 비밀값의 인덱스 5비트를 합쳐 초기 순서번호(ISN : Initial Sequence Number)로 설정한다. 클라이언트는 이에 대한 응답 패킷을 보내게 되는데 이 때 응답 번호(acknowledgement number)는 서버의 ISN에 1을 더한(ISN+1) 값을 가진다. 서버는 키값을 다시 계산하여 응답 번호와 비교하고 일치하면 연결 요청을 수락한다. 하지만 SYN cookies에서 사용하는 해쉬 키값은 24비트 길이를 가지므로 너무 짧다는 단점을 가지고 있다. 또한 MSS값을 3비트로 인코딩 하였기 때문에 표현할 수 있는 MSS값은 8가지에 불과하며 클라이언트가 요청한 MSS보다 작거나 같은 값을 가지게 된다. 따라서 TCP의 전송 성능을 저하시킬 수 있다. 또한 데이타 전송 성능을 향상시키기 위해 사용하는 선택 응답 옵션(Selected ACK Permitted)을 표현할 수 없기 때문에 서버는 클라이언트의 이 요청을 무시하게 되고 응답을 계속 보내야하므로 역시 전송 성능이 떨어지게 된다. 만약 서버가 전송하는 SYN/ACK 패킷이 분실된다면 서버는 이를 재전송 해야하지만 SYN cookies에서는 분실을 탐지할 방법이 없다.

3. SYN Flood 공격 원리 및 방법

본 절에서는 TCP에서 수행하는 연결 설정 과정에 대해 알아보고 이 과정의 특성을 이용한 SYN Flood 공

격에 대해 설명한다.

TCP는 3-way handshake 과정을 거쳐 서버와 클라이언트 사이에 연결을 설정한다. 연결 요청을 하는 클라이언트는 서버에 초기 순서번호(ISN)를 가진 SYN 패킷을 전달하고 이것을 수신한 서버는 연결 요청을 잘 받았다는 의미로 자신의 초기 순서번호를 가진 SYN/ACK 패킷을 클라이언트에게 전송한다. 이를 수신한 클라이언트는 응답으로 ACK 패킷을 전달한다. 여기까지의 과정을 거치면 서버와 클라이언트 사이에는 연결 설정이 완료되어 데이타를 주고받을 수 있게 된다. SYN Flood 공격은 서버에게 수많은 SYN 패킷을 전송하는 방법으로 요청을 수신한 서버는 이에 대한 응답으로 SYN/ACK 패킷을 클라이언트에게 전송한 다음 이에 대한 응답인 ACK를 기다리고 있게 된다. 이때 연결 정보는 큐에 저장된다. 이에 대해 공격자는 ACK 패킷을 전송하지 않음으로 해서 서버의 큐가 가득 찬 상태에서 더 이상의 연결 요청을 받아들일 수 없게 만들어버린다. 그림 1은 SYN Flood 공격 과정을 보여주고 있다. 그림에서 보는 바와 같이 클라이언트가 SYN/ACK에 대한 ACK를 전송하지 않음으로 해서 서버측에서는 3-way handshake 과정을 끝마칠 수가 없게 되는 것이다.

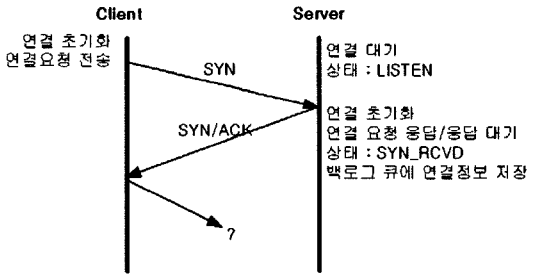

그림 1 SYN Flood 공격 형태

4. 확장 TCP의 설계 및 구조

4.1 SYN Flood 공격을 차단하기 위한 조건

SYN Flood 공격 원리의 핵심은 서버가 SYN/ACK를 보낸 후 클라이언트의 ACK를 기다리고 공격자 클라이언트는 ACK를 보내지 않음으로써 서버의 백로그 큐가 차 오르는데 있다. 이러한 공격 방식으로부터 자유롭기 위해서는 다음의 두 조건을 만족해야 한다[6].

1. 서버는 클라이언트의 요청을 받은 후 응답을 보내고 그것에 대한 상태를 유지하지 않는다.
2. ACK 패킷을 보낸 클라이언트가 연결 요청을 보낸 클라이언트임을 확신할 수 있어야 한다.

확장 TCP는 연결 설정 과정의 상태를 저장하지 않고 서버가 생성한 키 값을 되받아 이 값이 유효한 값인지 검사하고 유효할 경우 연결 설정을 하는 방식을 이용하여 위의 조건을 만족시킨다.

기존 연구들의 다수는 네트워크 환경의 변형을 요구하고 있어 이미 사용중인 시스템에 바로 적용하기는 어렵다는 문제점이 있었다. 따라서 해법으로 제시하는 프로토콜이 가져야할 필요 조건은 아래와 같다[2].

1. 최종 시스템을 보호하기 위한 구현은 운영체제와 네트웍 계층에 독립적이어야 한다.
2. 기존 IP 혹은 TCP 프로토콜 응용과 통신할 수 있어야 한다.
3. 하나의 호스트뿐만 아니라 호스트 그룹도 보호할 수 있어야 한다.
4. 특수한 하드웨어를 요구하지 않아야 한다.
5. 이식성, 확장성을 갖추어야 한다.

본 논문의 확장 TCP는 특정 운영체제에 종속적이지 않고 상/하위 네트워크 계층에 영향을 주지 않는다. 또한 기존의 TCP/IP와 통신할 수 있으며 한 호스트는 물론 호스트 그룹이 모두 확장 TCP로 동작할 경우 모든 호스트가 SYN Flood 공격으로부터 보호된다. 확장 TCP가 동작하기 위해 특수한 하드웨어를 요구하지 않고 필요에 따라 보안성을 강화할 수 있다. 따라서 위 조건들을 만족하고 있으며 기존의 네트워크에 바로 적용이 가능하다.

4.2 확장 TCP의 동작 과정

확장 TCP는 SYN cookies와 유사하게 동작하지만 많은 문제점들을 보완하였다. 접속을 원하는 클라이언트는 서버에게 접속 요청을 하기 위해 C-ISN(클라이언트가 전송하는 초기 순서번호)를 요청 패킷의 SYN 필드에 넣어 전송한다. 요청을 받은 서버는 클라이언트의 IP 주소와 포트번호, C-ISN, 서버의 IP 주소, 포트번호, 비밀값을 조합하여 MD5 해쉬를 적용한 32비트 키값을 생성하여 이 값을 S-ISN(서버가 전송하는 초기 순서번호)로 하고 C-ISN에 1을 더한 값을 ACK 번호로 설정하여 응답한다. MD5 해쉬 알고리즘을 사용하는 이유는 처리 속도가 빠르며 값의 축약효과가 좋다고 검증된 알고리즘이기 때문이다. S-ISN와 ACK를 받은 클라이언트는 S-ISN에 1을 더한 값을 ACK 번호로 하는 응답 패킷을 서버에게 전달한다. 응답을 받은 서버는 앞의 S-ISN와 ACK를 보내는 과정과 같이 SYN-1(=C-ISN)를 구한 다음 다시 해쉬를 하여 키값을 생성한 후 클라이언트가 보내온 ACK-1(=S-ISN)과 비교를 한다. 만약 두 값이 일치한다면 클라이언트는 처음 C-ISN로 요청을 한 클라이언트임을 확신할 수 있기 때문에 연결 요청을 받아들이고 그렇지 않다면 이를 무시한다.

확장 TCP에서 중복된 ISN 값이 생겨날 확률은 매우 낮다. 순서 번호는 같은 서버 포트, 같은 클라이언트 포

트를 사용하는 세션에서 패킷의 순서를 결정하는데 사용된다. 만약 지금 막 TCP 통신을 끝낸 클라이언트가 같은 포트를 이용하여 다시 서버에 접속하여 통신을 시작할 경우 이전 통신에서 사용되던 패킷이 전송이 지연되어 인터넷에서 떠돌다 해당 통신이 끝난 후에 서버에 도달하고 이 패킷의 순서 번호가 지금 생성된 새로운 TCP 세션의 윈도우에 속한다면 에러를 발생시킬 수가 있다. 하지만 이러한 경우가 발생할 확률은 매우 낮으며 TCP 표준에서도 이러한 경우를 무시하고 랜덤 값을 ISN으로 사용하도록 허용하고 있으므로 확장 TCP에서 선택한 ISN이 에러를 일으킬 확률 또한 매우 낮다고 할 수 있다.

그림 2는 확장 TCP의 Finite State Machine을 보여주고 있다. 기존의 TCP와 달리 확장 TCP는 SYN_RCVD(SYN Received) 상태가 없다. SYN_RCVD 상태는 클라이언트가 보낸 SYN을 수신하고 ACK를 기다리면서 연결 정보를 저장하고 있는 상태이다. 확장 TCP에서는 연결 설정 과정의 상태를 저장하지 않기 때문에 LISTEN 상태에서 ACK를 수신하고 확인을 한 후 ESTABLISHED 상태로 전환한다.

클라이언트가 연결 요청시 설정한 TCP 옵션들은 서버가 응답하는 패킷의 타임스탬프 옵션에 표시된다. 타임스탬프 옵션은 호스트가 상대방 호스트와의 RTT (Round Trip Time)를 측정하기 위해 사용되는 옵션으로 서버가 보내는 타임스탬프 값을 클라이언트가 그대로 다시 돌려주는 특성이 있으며 32비트 길이를 가진다. 확장 TCP에서는 이러한 특성을 이용하여 클라이언트가 연결 요청 시 보내오는 MSS값 16비트와 Selective ACK(SACK) Permitted 사용여부를 표시하는 플래그를 설정하여 타임스탬프 필드에 넣는다. 여기에는 17비트만 있으면 충분하다. 그리고 나머지 15비트는 나중을 위하여 남겨 두었다.

이러한 처리를 통해 생성된 서버의 SYN/ACK 패킷의 구조는 그림 3에 나타나 있다.

클라이언트가 응답하는 패킷에는 이 값이 그대로 돌아오기 때문에 서버는 여기에서 MSS 값과 SACK Permitted 여부를 읽어와 TCP 응용에 그대로 적용할 수 있어 TCP 성능이 저하되는 것을 막아준다.

서버는 클라이언트가 보내온 ACK-1값과 자신이 전송한 S-ISN값이 일치하는지 검사하는 과정에서 서버가 보낸 S-ISN값을 유지하고 있는 것은 바람직하지 못하다. 왜냐하면 이를 유지하는데 메모리가 필요하게 되고 이 메모리 공간을 포화시키는 공격이 가능하기 때문이다. 따라서 서버는 값을 유지하는 것이 아니라 클라이언트가 보내온 ACK를 받은 후 다시 키값을 생성하여 비교를 하도록 하였다.

서버가 유지하는 비밀값은 서버 외의 다른 시스템이 서버가 생성해 내는 S-ISN를 유추할 수 없게 하여 ACK 포화공격을 할 수 없게 하는 역할을 한다. 따라서 서버는 주기적으로 비밀값을 새로 생성하고 이전에 사용한 비밀값(oldSecret)과 새로 생성한 비밀값(newSecret)을 유지하면서 ACK 번호를 검사하는데 사용한다. 일정한 시간이 지난 후에 또 새로운 비밀값이 생성되면 newSecret은 oldSecret으로 대체되고 새로 생성되는 비밀값이 newSecret이 된다. 이전에 사용하던 oldSecret은 폐기되기 때문에 이 값으로 생성한 S-ISN에

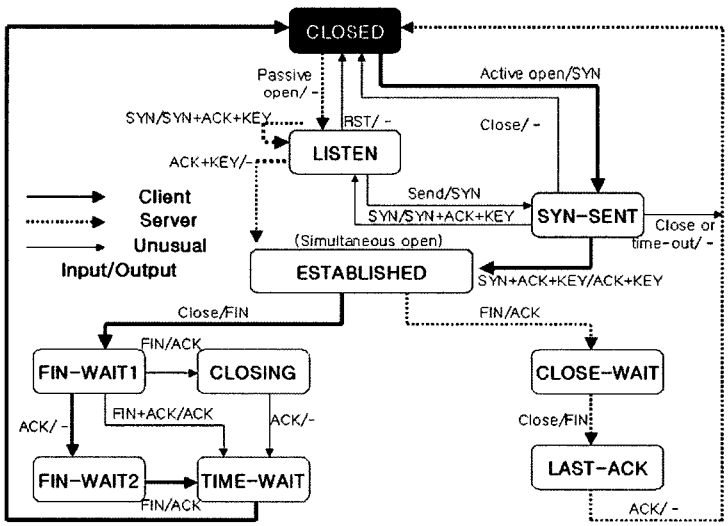

그림 2 확장 TCP Finite State Machine

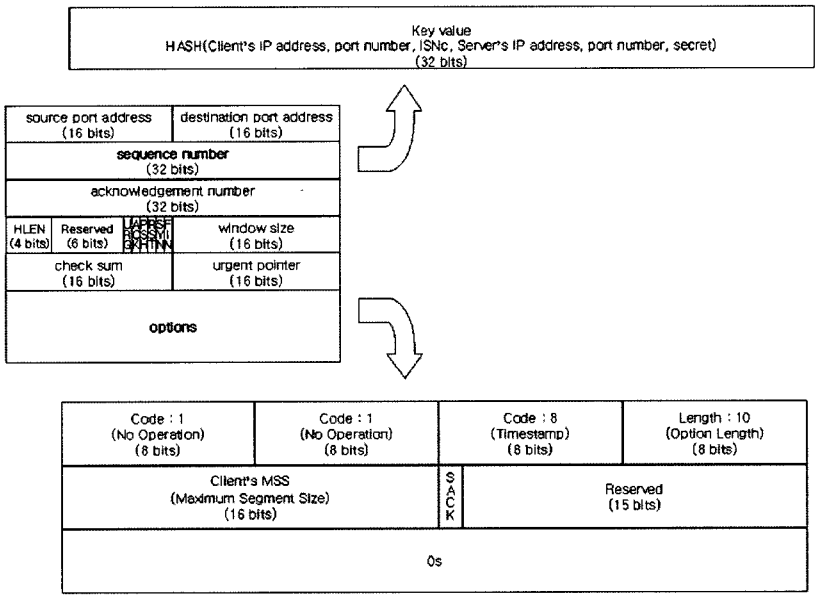

그림 3 확장 TCP의 ISNs/ACK 패킷 구성

대한 응답은 이후에 무시가 된다. 이것은 기존의 TCP에서 응답 대기 타임아웃값과 같은 역할을 할 수 있게 해 준다.

5. 리눅스 커널에서 확장 TCP 구현

본 논문에서 설계한 확장 TCP를 실제 서비스 환경에서 테스트 해 보기 위하여 리눅스 커널의 TCP를 수정하여 구현해 보았다. 여기에 사용된 리눅스 커널의 버전은 안정버전 2.4.21이다. 리눅스는 다양한 네트워크 프로토콜을 지원하지만 본 논문에서는 IPv4기반의 TCP에 대해서만 다룬다.

리눅스 커널 2.4.21에서 TCP는 수신된 패킷 정보를 소켓 버퍼(sk_buff)구조체에 저장하여 각 멤버값들을 핸들링하는 방식으로 처리한다. 패킷이 수신되면 IPv4 핸들러가 이를 수신하여 각 데이타의 유효성을 검사하고 유효성 검사를 통과하면 이 패킷을 처리한 소켓을 찾는다. 만약 처리할 소켓이 존재하지 않는다면 이를 무시하고 찾아낸다면 해당 소켓이 LISTEN 상태인지 ESTABLISHED 상태인지 검사한다. ESTABLISHED 상태인 소켓이라면 해당 어플리케이션으로 패킷을 전달하도록 한다. 만약 LISTEN 상태이고 SYN 플래그가 셋(set)되어 있다면 3-way handshake 과정을 거치게 되므로 새로운 소켓을 생성하고 응답 패킷을 구성한 후 패킷을 전송하고 syn_table[]이라는 구조체 배열에 해당 소켓 포인터를 저장한다. 이 때 새로 생성된 소켓은 SYN_RCVD상태가 된다. 여기서 사용되는 구조체 배열이 백로그 큐(backlog queue) 혹은 불완전 접속 큐(incomplete connection queue)이고 SYN Flood 공격 시 포화상태가 되는 큐이다. 수신된 패킷이 LISTEN 상태인 소켓에 해당하지만 syn_table[]에 해당 소켓이 존재하고 ACK 필드가 셋되어 있을 경우 이는 서버가 전송한 응답 패킷에 대한 응답이므로 syn_table[]에서 소켓 포인터를 가져와 연결 큐(complete connection queue)에 저장하고 소켓을 ESTABLISHED 상태로 만든다.

확장 TCP에서는 위의 syn_table[]을 사용하지 않으며 연결 요청이 있을 경우 새로운 소켓을 생성하지 않는다. 다만 응답 패킷을 생성할 때 4장에서 언급한대로 해쉬 키값을 생성하고 추가적인 정보를 타임스템프 필드에 넣어 응답패킷을 구성하고 전송하기만 한다. 따라서 상태도 변화시키지 않는다. 수신된 패킷에 ACK 필드가 셋되어 있을 경우 이는 서버가 전송한 응답 패킷에 대한 응답이므로 수신한 패킷의 응답 번호에서 1을 뺀 값과 패킷의 정보로부터 새로 생성한 키값이 일치하는지 검사하고 일치한다면 소켓을 생성하여 연결 큐에 해당 소켓 포인터를 저장한다. 만약 일치하지 않는다면 비밀값을 oldSecret을 사용하여 다시 키값을 생성하여 비교한다. 두 번의 비교에서 모두 일치하지 않는다면 이는 잘못 수신된 ACK 패킷이므로 무시한다.

리눅스 커널에 구현된 확장 TCP가 동작하는 과정은 아래의 그림 4에서 순서도로 표현하였다. 짙은 색으로 표현된 부분은 원래 TCP에서 수정된 과정이다.

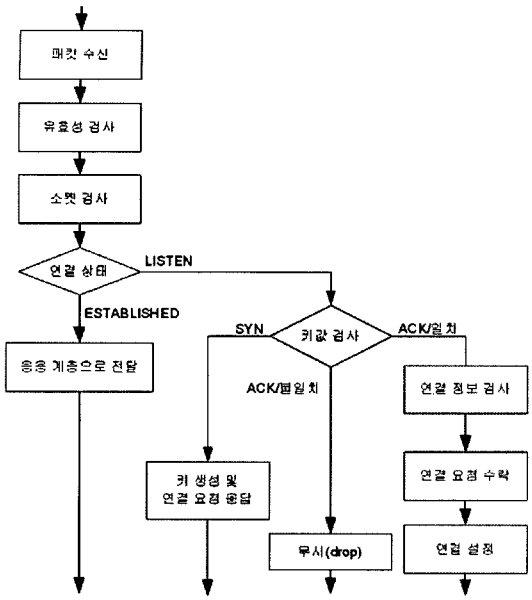

그림 4 확장 TCP의 3-way handshake 과정

6. 성능평가

아래의 표 1은 리눅스에서 사용되는 SYN cookies와 본 논문의 확장 TCP의 특성을 비교한 것이다.

표 1 리눅스 SYN cookies와 확장 TCP의 비교

특성	리눅스 SYN cookies	확장 TCP
키 길이	24-bit	32-bit
키값 파악에 필요한 초당 시도 회수 (64초 키의 경우)	262,144	67,108,864
공격에 필요한 대역폭 (byte)	14,155,776	4,429,185,024
MSS 표현값	3-bit	16-bit
SACK 지원 여부	×	○
유지하는 비밀값의 수	32개	2개
비밀값 유지 시간 설정	불가능	가능

확장 TCP에서는 키 값의 길이가 32비트이기 때문에 브루트 포스(brute force) 공격에 보다 강하다. 타임아웃 시간이 64초로 설정되어 있을 경우 공격자가 브루트 포스 공격을 시도한다면 초당 67,108,864(=232÷64)가지의 키 값을 생성하여 서버에게 전송하여야 하며 ACK 패킷의 크기가 66바이트인 것을 감안한다면 초당 4,429,185,024(=67,108,864×66)바이트의 대역폭을 필요로 하게 되므로 보통의 서버 환경에서는 이러한 시도를 할 수가 없다. 또한 클라이언트가 요구하는 MSS 크기를 그대로 지켜줄 수 있기 때문에 전송 속도 저하가 없으며 Selective ACK Permitted를 준수하기 때문에 재전송 회수를 줄일 수 있다. 서버에서 유지하고 있는 비밀값의 생성 주기를 조절할 수 있기 때문에 서버의 환경에 따라 보안 정도를 조절할 수도 있다.

확장 TCP에서는 연결 설정 과정의 상태를 저장하지 않기 때문에 큐를 가득 채우는 것이 불가능하다. 성능평가에는 기존에 제안되었던 방법인 리눅스 SYN cookies를 이용하는 TCP (SYN cookies), 원래 TCP (original TCP), 그리고 확장 TCP를 이용하여 서버 시스템을 구성하고 공격시스템을 구성하여 공격을 시도하였다. 실험은 공격이 이루어지고 있는 상황에서 정상 연결을 요청하는 클라이언트가 접속에 성공하는데 걸리는 시간을 측정하는 방법으로 수행하였다. 확장 TCP가 실제 서비스 환경에서 사용할만한 가를 알아보기 위해 정상적인 서비스 환경에서 연결 설정 과정의 처리 속도를 측정하는 실험을 수행하고 SYN Flood 공격에 강한지를 알아보기 위해 SYN Flood 공격이 이루어지고 있는 상황에서의 처리 시간을 측정하는 실험만을 수행하였다.

리눅스 SYN cookies를 사용하는 TCP는 백로그 큐를 사용하다가 큐가 가득 찼을 때부터 SYN cookies를 사용한다. 또한 original TCP와 SYN cookies TCP에 사용되는 백로그 큐의 크기는 20개로 설정하였다. 백로그 큐의 크기는 크게 하더라도 SYN Flood 상태가 되는 시간이 더 걸릴 뿐 큰 효과를 볼 수 없으므로 기본 설정값을 그대로 사용하였다. 클라이언트는 100개의 스레드를 생성하여 동시에 서버에 접속 요청을 보낸 후 연결 설정이 되면 서버가 전송하는 1MByte의 데이타를 수신하고 연결을 종료하는 과정을 거친다. 아래에서 행하는 실험들은 3-way handshake 과정의 소요시간을 측정하기 위하여 연결 설정 직전에 마이크로 초 단위의 시간을 측정하고 연결 설정 직후에도 시간을 측정하여 소요 시간을 계산하였다. SYN Flood 공격을 하는 공격 호스트는 libnet 라이브러리를 이용하여 초당 20개의 SYN 패킷을 생성하여 서버 호스트로 보내는 동작을 한다. 공격 패킷의 송신자 주소는 라우팅 할 수 없는 사설망 IP(192.168.0.0)를 무작위로 지정하였다.

아래의 표 2는 세 시스템의 환경을 나타내고 있다.

첫 번째 실험은 네트워크 트래픽이 없는 상태에서 서버가 클라이언트의 요청을 처리하는 성능을 측정하였다. 실험에 사용된 서버 TCP는 original TCP, SYN cookies TCP, 확장 TCP 이다. 아래의 그림 5는 연결 설정에 소요된 시간을 비교하여 보여주고 있다. 확장 TCP는 SYN cookies TCP과 거의 유사한 성능을 보여주며 original TCP 와도 유사한 성능을 가지고 있다는 것을 알 수 있다. 그래프가 나타내는 original TCP와 확장 TCP와의 시간 차이는 확장 TCP가 original TCP

표 2 테스트 환경

항목	서버 호스트	클라이언트 호스트	공격 호스트
CPU	500MHz×1	1GHz×2	500Mhz×1
메모리	192MByte	1GByte	384MByte
대역폭	10Mbps	100Mbps	100Mbps
운영체제	리눅스-2.4.21	리눅스-2.4.21	리눅스-2.4.21
설정	original TCP selected random drop SYN cookies 확장 TCP	pthread 100개의 요청 전송	libnet-1.0

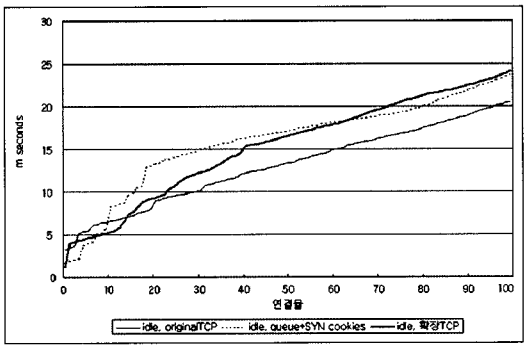

그림 5 idle 상태일 때 original TCP, SYN cookies, 확장 TCP 비교

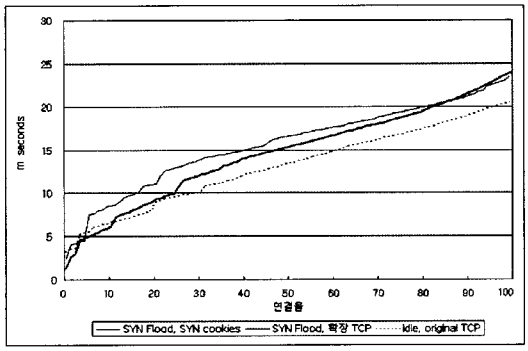

그림 6 SYN Flood 공격 상태에서의 성능 비교

는 수행하지 않는 키값 생성을 하기 때문에 약 0.05%의 지연시간이 더 걸린다는 것을 보여주고 있다.

두번 째 실험은 SYN Flood 공격이 이루어지고 있는 상태에서의 SYN cookies와 확장 TCP, 그리고 original TCP의 정상 상태와의 비교를 해 보았다. 위의 그림 6에서 볼 수 있듯이 SYN Flood 공격이 이루어지고 있는 상태에서 SYN cookies와 확장 TCP는 idle 상태에서의 original TCP와 거의 유사한 성능을 보여주고 있다. original TCP에 SYN Flood 공격을 가할 경우 백로그 큐가 가득 차는데 걸리는 시간인 1초 이후에는 연결에 성공하지 못해 비교자체가 불가능하였다. 따라서 SYN Flood 공격이 가해지고 있는 상황에서도 공격이 가해지지 않을 때와 유사한 성능으로 서비스를 처리할 수 있다는 것을 알 수 있다. 또한 확장 TCP에서는 백로그 큐를 검사하지 않기 때문에 SYN cookies보다 약간 더 빠른 시간에 연결을 처리한다는 것을 알 수 있다.

7. 결론 및 향후 계획

본 논문에서는 서버의 자원을 고갈시켜 서버 프로그램이 더 이상의 클라이언트에게 서비스를 할 수 없게 하는 서비스 거부 공격의 하나인 SYN Flood 공격에 대해서 알아보았다. 또한 SYN Flood 공격에 대한 기존 연구와 공격의 특성 및 원리에 대해서도 알아보았다.

본 논문에서 제시한 확장 TCP는 SYN Flood 공격을 차단하기 위하여 3-way handshake과정에서 고갈되는 연결 요청 큐를 사용하지 않음으로 해서 정상적인 서비스를 할 수 있게 하였다. 또한 리눅스 커널을 수정하여 확장 TCP를 구현하였으며 구현 방법에 대해서도 살펴보았다. 확장 TCP와 유사한 방법을 사용하는 리눅스 SYN cookies가 가지는 여러 가지 한계점도 살펴보았다. 우리는 두 가지 실험을 통하여 성능 평가를 수행하였으며 SYN cookies보다 보안 강도가 높으면서도 SYN Flood 공격이 이루어지고 있는 상황에서도 서비스 처리 시 성능 저하가 거의 없다는 것을 확인하였다.

본 논문에서 언급한 SYN cookies의 문제점 중 서버가 보내는 연결 요청 응답의 분실 시 재전송이 불가능한 문제점은 확장 TCP도 그대로 가지고 있다. 따라서 향후 계획으로는 이러한 문제점을 극복할 수 있는 방안을 연구할 것이다. 또한 보다 많은 실험을 거쳐 확장 TCP의 성능을 평가하고 타임스템프 필드를 이용함에 있어 발생할 수 있는 문제점을 파악하여 문제점이 발견될 경우 이를 해결할 수 있는 방법에 대해 계속 연구를 진행할 것이다.

참 고 문 헌

[1] Lau, F.; Rubin, S.H.; Smith, M.H.; Trajkovic, L.:, "Distributed denial of service attacks," Systems, Man, and Cybernetics, 2000 IEEE International Conference on, Volume: 3, 8-11 Oct. 2000 Page(s): 2275-2280 vol.3

[2] Schuba, C.L.; Krsul, I.V.; Kuhn, M.G.; Spafford, E.H.; Sundaram, A.; Zamboni, D.:, "Analysis of a

denial of service attack on TCP," Security and Privacy, 1997. Proceedings., 1997 IEEE Symposium on, 4-7 May 1997 Page(s): 208-223

[3] Haining Wang; Danlu Zhang; Kang G. Shin;, "Detecting SYN flooding attacks," INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE, Volume: 3, 23-27 June 2002 Page(s): 1530-1539.

[4] Haining Wang; Danlu Zhang; Shin, K.G.;, "SYN-dog: sniffing SYN flooding sources," Distributed Computing Systems, 2002. Proceedings. 22nd International Conference on, 2-5 July 2002 Page(s): 421-428.

[5] Jonathan Lemon;, "Resisting SYN flood DoS attacks with a SYN cache," Proceedings of the BSDCon 2002 Conference, Feb 2002.

[6] Zin-Won Park; Joon-Hyung Lee; Myung-Kyung Kim;, "Design of and Extended TCP for preventing DoS Attacks," Proceedings of 2003KORUS, 2003, Page(s)385-389.

박 진 원

2002년 울산대학교 공학사. 2004년 울산대학교 공학석사. 현재 울산대학교 컴퓨터·정보통신공학과 박사과정. 관심분야는 정보통신, 정보처리, 정보보호

김 명 균

1984년 서울대학교 공학사. 1986년 한국과학기술원 공학석사. 1996년 한국과학기술원 공학박사. 1989년~1997년 우석대학교 교수. 1997년~현재 울산대학교 컴퓨터·정보통신공학과 부교수. 관심분야는 정보통신, 정보처리, 망관리