情報保護學會論文誌
第 21 卷 第 2 號, 2011. 4

TCP 플러딩 공격 방어를 위한 네트워크 인터페이스용 고성능 TCP 프락시 제어 로직 구현

김 병 구,[12†] 김 익 균,[2‡] 김 대 원,[2] 오 진 태,[2] 장 종 수,[2] 정 태 명[1]
[1]성균관대학교, [2]한국전자통신연구원

Implementation of High Performance TCP Proxy Logic against TCP Flooding Attack on Network Interface Card

Byoungkoo Kim,[12†] Ikkyun Kim,[2‡] Daewon Kim,[2] Jin-tae Oh,[2]
Jong-soo Jang,[2] Tai-Myoung Chung[1]
[1]Sungkyunkwan University, [2]ETRI

요 약

본 논문은 인터넷 서버의 정상적인 TCP 연결을 방해하는 형태의 DDoS(Distributed Denial of Service) 공격으로부터 서버를 보호하고 원활한 서비스를 제공하기 위한 DDoS 공격 차단 방법에 관한 것이다. 즉, 유효한 TCP 연결만을 전달해주는 기능을 가진 네트워크 인터페이스 카드에서의 고속 TCP 연동 로직(NIC_Cookie)에 대하여 기술한다. NIC_Cookie의 장점은 플러딩 공격 상태에서도 실시간으로 공격 패킷을 차단하고 정상적인 패킷만이 메인 CPU로 전달될 수 있도록 하기 때문에, 서버의 CPU 성능과 외부 네트워크의 구성 등에 영향을 받지 않는다는 것이다. 또한, 패킷 단위로 공격 TCP 세션에 대하여 실시간 차단 대응을 하기 때문에 정상적인 연결 시도에 대하여 잘못된 대응을 할 가능성이 없다. 이와 더불어, NCI_Cookie 로직 추가로 발생하는 지연시간은 패킷 길이와 상관없이 $7x10^{-6}$초 이하의 성능을 보장하며, 본 논문에서는 구현된 NIC_Cookie가 일반적인 플러딩 공격을 실시간으로 방어하면서 그 성능을 보장함을 확인하였다.

ABSTRACT

TCP-related Flooding attacks still dominate Distributed Denial of Service Attack. It is a great challenge to accurately detect the TCP flood attack in hish speed network. In this paper, we propose the NIC_Cookie logic implementation, which is a kind of security offload engine against TCP-related DDoS attacks, on network interface card. NIC_Cookie has robustness against DDoS attack itself and it is independent on server OS and external network configuration. It supports not IP-based response method but packet-level response, therefore it can handle attacks of NAT-based user group. We evaluate that the latency time of NIC_Cookie logics is $7x10^{-6}$ seconds and we show 2Gbps wire-speed performance through a benchmark test.

Keywords: DDoS attack, NIC_Cookie, TCP Proxy

I. 서 론

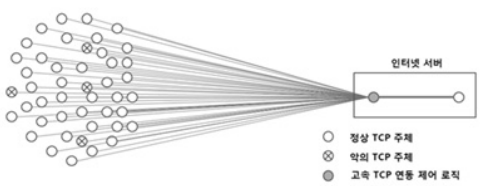
인터넷 기반 서버의 가용성을 위협하는 형태의 서비스 거부(DoS: Denial of Service) 공격이 최근 급증하고 있으며, 이러한 공격의 형태는 주요 웹사이트나 루트 DNS 서버에 대한 공격처럼 국가나 인터넷

접수일(2011년 1월 21일), 게재확정일(2011년 4월 4일)
† 주저자, bkkim05@etri.re.kr
‡ 교신저자, ikkim21@etri.re.kr

기반체계를 대상으로 매우 광범위하게 전개되고 있는 실정이다. 과거에는 DDoS 공격의 대상으로서 네트워크 인프라 마비를 목표로 하는 사례도 있었으나 최근에는 금전적인 갈취나 정치적 목적으로 특정 서비스를 제공하는 서버를 직접 공격하는 사례가 대세를 이루고 있다. 특정 서버에서 발생되는 DDoS 공격의 유형은 TCP 프로토콜을 이용하는 공격이 대부분을 차지하고 있으며, 불필요한 TCP 연결 시도와 TCP 프로토콜의 비정상적인 접근, 사용하지 않는 연결 설정 유지 등을 이용하고 있으며, 또한 HTTP GET 플러딩과 같이 정상적인 TCP 프로토콜 연결 상에서 정상적인 패킷을 이용하는 경우도 많이 발생한다.

서버를 공격하는 DDoS 공격이 빈번히 발생하는 기술적인 이유는 정상적인 TCP 주체들과 악의의 TCP 주체들이 동시에 서버에 접속하는 상황에서 TCP 프로토콜이 보안성을 고려하지 않은 상태에서 서비스를 제공함으로써 DDoS와 같은 공격에 매우 취약하기 때문이다. 이런 단점을 보완하기 위하여 IETF에서는 SCTP(Stream Control Transmission Protocol)[1]과 같은 보안성이 강화된 프로토콜을 새롭게 개발하여 그 규격을 배포하고 있지만, 기존 TCP의 점유율이 워낙 지배적이기 때문에 SCTP로 대체되기에는 불가능하다고 보는 게 정설이다. 따라서 불특정 다수가 이용하는 인터넷 서버의 경우 기존의 TCP의 보안상의 취약점과 그로 인한 성능 문제 해결이 함께 보완되기 위한 연구들이 진행되어 왔다.

본 논문은 서버의 정상적인 TCP 연결을 방해하는 형태의 DDoS 공격으로부터 서버를 보호하고 원활한 서비스를 제공하기 위한 TCP 기반 DDoS 공격 차단 기술로써, 악의적인 TCP 사용자의 접속 시도에 대하여 서버의 성능 감소 없이 원천적으로 차단하는 방법에 관한 것이다. 본 논문은 [그림 1]과 같이 악의적인 사용자의 TCP 연결을 근본적으로 차단하여 유효한 TCP 연결만을 전달해주는 기능을 가진 네트워크 인터페이스 카드에서의 고속 TCP 연동 로직 구현에 대하여 기술한다. 이는 악의적인 TCP 사용자의 행위에 대하여 네트워크 인터페이스 카드에 내장된 고속 TCP 연동 제어 기법 (NIC_Cookie)을 통하여 비정상적인 TCP 행위를 근본적으로 차단함으로써 서버의 CPU 부하 증가가 전혀 없도록 서버를 보호하게 한다. 따라서, 정상적인 TCP 사용자의 응용 서비스 사용은 원활하게 제공하면서, 악의적인 TCP 사용자의 서버 접근을 식별, 실시간으로 차단하는 방법을 제공한다.



[그림 1] 인터넷 서버를 이용하는 TCP 주체의 성격

본 논문에서 제안한 방법의 주요 장점은 통상 네트워크 로드 밸런싱 기능으로 인한 비대칭적 연결 설정 환경에서도 TCP 플러딩 공격에 대해 최종적인 공격 방어책이 될 수 있고, 리눅스 및 윈도우 등 OS 제약사항에 관계없이 투명하게 제공될 수 있으며, TCP 연결 설정 연동에 있어 자원 고갈 문제가 근본적으로 해결되는 방식이므로 자체적으로 DoS 공격에 내구성을 가지고 있고, TCP 계층의 플러딩 공격뿐만 아니라 이를 이용한 응용 계층 DDoS 공격 방어에도 확장될 수 있는 구조이며, 무엇보다도 가장 중요한 점은 서버의 고유 서비스를 제공함에 있어 성능적 오버헤드가 전혀 없는 구조로 고성능 공격 방어 처리를 보장한다는 것이다.

이 후 논문의 구성은 제 2장에서 TCP 플러딩 방어 기술과 관련된 연구에 대해 기술하고 제 3장에서는 본 논문에서 제안한 고성능 네트워크 인터페이스용 TCP 연동 제어 로직 구조에 대해 기술하며, 그 장점과 성능 향상을 위해 사용된 구현 특징 및 구현 환경에 대해 설명하고, 제 4장에서는 구현 로직의 성능 평가와 그 활용에 대해 기술하며, 마지막으로 결론 및 향후 계획에 대해 다룬다.

II. TCP Flooding 방어 관련 연구 및 동기

DDoS 공격의 종류를 구분해 보면, 크게 네트워크 프로토콜 취약점을 이용하는 네트워크 레벨의 공격과 응용 계층의 취약점을 이용하는 공격으로 나눌 수 있다. 응용계층 레벨의 공격의 대표적인 것이 HTTP GET 플러딩 공격[2]이며, 네트워크 레벨 공격 중 TCP와 관련된 공격은 TCP SYN 플러딩, TCP 플래그 플러딩(ACK, FIN, RST 필드 등) 공격, TCP 연결 설정 공격 등이 있으며, 이는 전체 DDoS 공격의 90%를 차지한다[3]. 이 중에서도 TCP SYN 플러딩 공격은 대표적인 공격 방법으로서, 단순한 공격 기법임에도 불구하고 명확한 대응방법이 없는 게 현실이다. 최근에는 UDP/ICMP 플러딩 공격 또한 많은

이슈로 등장하고 있지만 UDP, ICMP 트래픽의 대역폭 조절[7], Random Dropping[13] 이외에 뚜렷한 방안이 제시되지 못하고 있는 실정이다. 2009년 7.7 DDoS 공격 사고와 같이 최근 DDoS 공격은 그 형태가 다양한 유형의 공격 조합을 이용하기 때문에 한 가지 대응 방법이 DDoS 공격을 방어할 수 있는 경우는 없다고 할 수 있고, 여전히 TCP 프로토콜을 이용한 네트워크 공격이 큰 비중을 차지하고 있는 현실이다. 본 절에서는 네트워크 레벨의 DDoS 공격 중 대표적인 공격인 TCP SYN 플러딩 공격 방어에 대한 연구에 대하여 살펴보도록 한다.

SYN 공격은 DoS의 하나로 공격할 시스템의 모든 자원을 소비하게 해서 재 부팅하게 만든다. 두 호스트 사이에 TCP 연결이 이루어지는 것은 클라이언트 호스트가 TCP 헤더에 SYN 플래그를 활성화한 상태로 서버 호스트로 연결 요청하면서 시작된다. 연결 요청을 받은 서버는 클라이언트로 SYN/ACK를 보내고 이를 받은 클라이언트는 서버에게 ACK를 보낸다. 그런데 클라이언트에서 의도적으로 ACK를 보내지 않음으로써 문제를 야기 시킨다. 여기에서 문제는 TCP에서 하나의 소켓에서 동시에 SYN 요청을 처리하기에는 한계가 있으며 이 한계가 백로그(backlog)라고 한다. 이 백로그 큐가 꽉 차게 되면 이후 들어오는 SYN 요청은 무시되며 이러한 공격을 SYN 플러딩이라고 한다. 다른 유닉스 OS 중에서는 들어오는 소켓 요청에 대하여 두 개의 큐를 만드는 경우도 있다. 이 경우 하나는 half-open 소켓 (SYN을 받고 SYN/ACK를 보낸 상태)을 위한 큐이고 또 하나는 완전히 연결이 되었지만 애플리케이션에서 accept() 콜을 기다리는 큐이다. 백로그도 무조건 늘리는 것이 좋은 것은 아니라는 보고가 있으며 1024 이상으로 설정 시에는 커널 소스를 수정해야 한다.

SYN 프락시[11]는 라우터 레벨에서의 TCP SYN 플러딩 방어 방법으로써, TCP SYN 패킷을 목적지 서버에 직접 전달하지 않고 중간 라우터 위치에서 마치 목적지 서버 역할을 하면서 연결 설정 클라이언트에게 SYNACK패킷을 응답하는 방법이다. SYN 프락시가 클라이언트로부터 TCP 연결 설정의 마지막 ACK 패킷을 수신했을 때 목적지 서버로 새로운 TCP 연결 설정을 시도하여 연동시킨다. SYN 프락시는 BSD 계열의 PF[9]에 선택기능으로 구현되어 있고, NetScreen[8] 같은 상용제품에도 고성능으로 구현되어 있으며, 다른 리눅스에서는 SYN Cookie[10,11]로 설치되어 있는 기능이다.

SYN 프락시는 클라이언트와의 연결설정을 완결할 때까지 목적지 서버로 새로운 연결 설정을 시도를 기다리는 반면, 체크포인트사의 FW-1에서 구동되고 있는 SYNDefendr[4]는 릴레이모드로 동작할 경우 SYN 프락시와 동일하게 동작을 하고, 게이트웨이 모드로 동작할 경우, 클라이언트로부터 SYN 패킷이 수신되면 즉시 목적지 서버와 완벽한 연결 설정을 완성하게 되는데, 이는 목적지 서버가 half-open 상태로 자원을 낭비하게 되는 것을 보호해 주게 된다.

네트워크 레벨에서 SYN 플러딩을 탐지하는 대표적인 방법으로는 'first mile' 과 'last mile'의 라우터에서 TCP SYN-FIN쌍의 트래픽 분포를 간단히 변곡점 분석을 통하여 판단하는 방법으로, 이 방법이 SYN 패킷과 SYN-ACK 패킷의 트래픽 분포를 분석하는 것보다 더 정확하다고 주장한다.

서버에서의 해결책은 SYN Cookie[10,11] 기법과 SYNCache[6] 방법이 있다. SYNCache는 half-open 연결 리스트에서 특정 연결을 찾기 위해 필요한 메모리를 제한시키도록 설계되었다. 즉, 각 포트별로 half-open의 리스트를 분리토록 하여 SYN 플러딩 공격에 의해 목표가 되는 포트를 분리할 수 있도록 하여 다른 포트에서는 정상적인 서버 기능이 동작되도록 보장하는 방법이다. 이는 윈도우, 리눅스와 BSD OS에서 채택된 방법으로써 현재 대부분의 OS에서 기본적으로 이 기능을 제공하고 있다. SYN Cookie는 SYN 플러딩 공격을 막을 때 사용하는 TCP 초기 시퀀스 값입니다. 이 기법은 1996년 9월에 Daniel J. Bernstein[10]과 Eric Schenk에 의해 고안되었다. 원래 SYN 큐가 가득 차면 서버는 더 이상 접속을 처리하지 못하고 되는데, 이렇게 하는 대신에 SYN Cookie를 사용하면 서버는 SYN+ACK 응답에 접속 관련 정보를 써넣은 다음 전송하고 SYN 큐에서 삭제합니다. 이후에 클라이언트로부터 ACK 응답을 받았을 때, 서버는 TCP 시퀀스 번호에 인코드 된 정보를 이용하여 SYN 큐 접속 정보를 복원할 수 있습니다. SYN 쿠키는 프로토콜 명세를 위반하지 않기 때문에 모든 TCP 구현과 호환 가능하지만, 서버가 사용할 수 있는 MSS 값이 8가지로 제한된다는 점과 TCP 옵션을 거부해야 한다는 단점이 있습니다. 이 때문에 성능이 최적으로 나오지 않을 가능성이 있지만, 사실상 클라이언트에 미치는 영향이 미미한데다 서버가 공격 당하고 있는 상황에서만 이 기법을 적용하는 것도 가능하기 때문에 별다른 문제가 되지 않는다.

이렇듯 SYN 플러딩 공격을 방어하기 위한 다양한 대안이 오래 전부터 연구 발표되었으나, 많은 DDoS 공격 기법 중인 SYN 플러딩 공격조차도 완벽히 차단하지 못하고 있는 게 현실이다. 첫째 이유는 네트워크 레벨에서의 트래픽 분석을 통한 탐지 방법은 오탐율 및 미탐율이 존재하여 대규모 네트워크 규모로 볼 때에는 탐지율이 높지 않다는 점이다. 둘째로는, SYN 프락시 같은 기능이 라우터 레벨로 동작할 경우에는 IP 트래픽의 특성상 경로가 보장될 수 없고, 특히, 네트워크 트래픽이 모이는 병목점에서 조차도 로드밸런싱 기능을 운용하기 때문에 SYN 프락시 기능이 완벽하게 동작하지 않을 가능성이 크다는 것이다. 셋째로는 네트워크레벨의 접근이나 서버레벨의 접근에 있어 공통적으로 고려되는 사항으로써, SYN 프락시, SYN Cookie 기능 등이 최대한의 성능을 보장 할 수 있어야 한다는 점이다. 최근 연구[5]에 따르면, 비교적 성능이 좋은 Free BSD와 Debian OS에서 SYN Cookie 기능을 설정 후에도 초당 1100개 이상의 SYN 패킷을 처리하기가 힘들다고 성능 평가 결과를 보여주고 있다. 마지막으로는 대응 방법에 관한 것이다. 정확한 플러딩 공격에 있어 정확한 탐지가 된다고 하더라도, 차단을 어떻게 하느냐에 따라 DDoS 공격 대응에 정확도가 달라진다. Rate limiting과 같은 패킷 기반의 mitigation 방법은 정상 사용자도 차단될 수 있을 뿐만 아니라, 플러딩 공격 패킷까지 통과될 수 있는 형태이고, 또한 IP 기반 차단 같은 경우에는 NAT(Network Address Translation)를 이용하는 클라이언트 환경에서는 정상적인 클라이언트도 서비스가 차단될 수 있다.

이와 같이, 본 논문에서는 로드 밸런싱 등 네트워크 환경상의 근본적인 제약점으로 인한 탐지 오탐율, 차단(대응) 방법론, 성능 이슈 등을 모두 해결하면서 SYN 플러딩 공격, TCP 플래그 플러딩 공격, 미사용과다 TCP 연결 설정 공격뿐만 아니라 응용 계층 공격 방어에도 확장될 수 있는 구조를 제안한다.

III. NIC_Cookie 설계

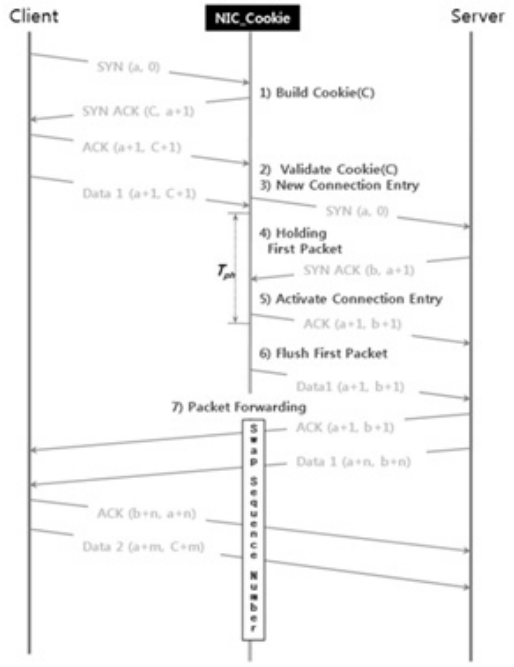
3.1 설계 개념

본 논문에서 제안하는 TCP 기반 플러딩 공격 방어 기법의 설계 개념은 앞 절에서도 언급된 바와 같이 i) TCP 플러딩 공격에 대해 미탐과 오탐없이 최종적인 대응 방법으로써, ii) 플러딩 공격 상태에서도 메인 CPU 점유율이 0% 상태로 1G bps 선로 속도를 보장하면서, iii) 공격 패킷 차단의 즉시 대응이 가능한 구조를 목표로 한다. 위의 설계 개념을 만족하기 위해서는 네트워크의 환경에 지배받지 않는 서버 레벨의 방어 접근이어야 될 것이고, 메인 CPU 점유율이 없도록 하기 위해서는 NIC 계층에서 TCP 프로토콜의 처리가 필요하다고 할 수 있다. 상기의 목적을 달성하기 위해서 TCP 기반 서버에 대한 부하공격을 탐지하고 차단하기 위한 하드웨어 기반의 DDoS 대응 기법을 제안한다. 이는 TCP를 이용하는 서버에 대한 TCP 레벨의 공격뿐만 아니라 응용 계층 과부하 공격도 차단할 수 있는 기법을 제공하며, 정상적인 사용자와 악의적인 사용자를 구분하여 공격 상황에서도 서버가 원활한 서비스를 제공할 수 있도록 한다. 우리는 이런 형태를 NIC_Cookie로 명명한다. NIC_Cookie는 정상 TCP 주체들 사이에서 악의의 TCP 주체를 분별함으로써 서버의 서비스를 보장하게 한다. 즉, 서버에 접속되는 모든 TCP 세션 설정 시도에 대하여 유효성 검사를 수행하고, 유효한 세션 시도에 대해서만 데이터 전달 서비스를 수행하도록 한다.

3.2 NIC_Cookie의 TCP 제어 연동 흐름

NIC_Cookie의 동작은 시간 축을 기준으로 TCP 3-way 연결 관리를 수행한다. TCP 유효성 검사는 NIC_Cookie의 주요 기능으로써 서버로의 새로운 TCP 연결 시도에 대하여 TCP SYN Cookie 기능 규격을 이용하여 유효성 검사를 수행한다. 일반적인 TCP SYN Cookie는 SYN 플러딩 공격에 대하여 서비스를 지속하기 위하여 Daniel J. Bernstein에 의해 고안된 방법론이다. NIC_Cookie는 TCP SYN Cookie의 기본 개념을 이용하지만, 유효성이 검증된 TCP 연결에 대해서만 서버로 전송될 수 있도록 한다.

원격 TCP 클라이언트와 NIC_Cookie는 [그림 2]와 같은 3-way 핸드 쉐이킹을 통하여 연결 설정을 수행하고, TCP 클라이언트로부터 첫 번째 패킷의 수신을 기다린다. 이 과정은 데이터 전송 없이 과도한 수의 TCP 연결을 open하는 경우를 대비한 방어 전략 중의 하나이다. 첫 번째 패킷이 수신되면, 고속 프락시 TCP 연결 관리에서는 적어도 TCP 수준에서는 정상적인 연결 설정으로 간주하고, 관리하는 3) 연결 관리 테이블에 엔트리를 추가한다. 이 연결 관리 테이블에는 연결 설정에 관련된 상태 정보와 TCP 시퀀스



〔그림 2〕 NIC_Cookie의 TCP 연결 제어 연동 흐름

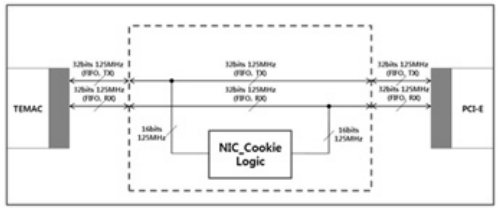
넘버를 치환할 수 있는 정보 등을 기록한다.

TCP 클라이언트로 수신된 첫 번째 패킷은 즉시 전달하지 않고, 고속 프락시 TCP 연결 관리가 서버측의 TCP 주체와 정상 연결을 설정하는데 걸리는 시간(T_{ph})동안 4) 패킷 버퍼에 보관하여야 한다. 클라이언트 측으로부터 수신된 첫 번째 패킷을 매개로 하여, NIC_Cookie는 새로운 연결 설정을 위한 SYN 패킷을 서버로 송신한다. 이때 사용되는 초기 시퀀스 넘버(ISN: Initial Sequence Number)는 첫 번째 데이터 패킷에 기록된 (시퀀스 넘버 - 1) 값을 사용함으로써 한 방향(클라이언트 -〉 서버)에 대한 시퀀스 넘버 동기화가 자동으로 이루어지도록 한다. 고속 프락시 TCP 연결 관리와 서버 간의 정상적인 3-way 핸드 쉐이킹이 이루어지는 시점에 연결 테이블의 해당 엔트리는 5) "connected" 로 정상적인 연결 설정 완료 상태로 전환되고, 저장된 6) 첫 번째 패킷을 서버 측으로 송신하게 된다. 이후 전달되는 모든 패킷은 연결 관리 테이블 해당 엔트리의 상태에 따라 전송여부가 결정될 것이며, 패킷 전달 시에는 시퀀스 넘버 동기화 정보를 이용하여 TCP 헤더 내에 시퀀스 넘버 치환과정이 수반된다. 이 모든 과정은 1Gbps 선로 속도에 준하는 성능을 보장할 수 있는 자료 구조로 설계된다.

3.3 NIC_Cookie 설계 주요 이슈

3.3.1 NIC_Cookie 로직 인터페이스

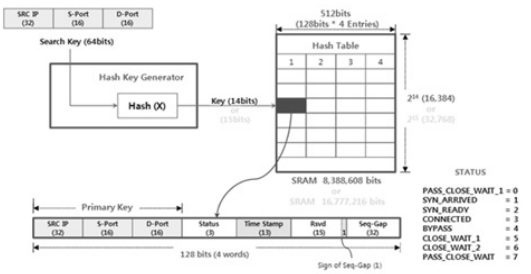
〔그림 3〕은 네트워크 인터페이스 카드에서의 NIC_Cookie 로직이 기존의 네트워킹 로직과 연동되는 인터페이스를 도식화한 것 이다. 일반적인 기가비트 이더넷 MAC 프로토콜 기능을 수행하는 로직과 호스트 서버와 접속되는 PCI-Express 접속 로직 사이에 NIC_Cookie 로직이 위치하며, 의미적으로는 MAC 로직과 PCI-E 로직 사이에서 인라인(in-line)방식으로 구성되어 패킷 차단 등의 즉각적인 대응 기능을 수행할 수 있어야 한다. 또한 송/수신 양방향 데이터에 대한 TCP 세션을 모니터링하여야 하므로 NIC_Cookie 로직으로 수신되는 패킷은 수신 MUX 로직으로 병합된다. 이때, 기가비트의 성능을 보장하기 위하여 적어도 패킷 데이터의 전달은 2Gbps 처리율(16데이터 비트 x 125Mhz clock)을 보장한다. NIC_Cookie 로직 내의 패킷 저장 FIFO는 패킷 처리를 할 수 있는 시간, 즉, 1500바이트 기준으로 800clock 이상 동안 패킷이 임시로 저장되는 공간이다. 이 시간은 NIC_Cookie가 네트워크 인터페이스 카드에서 동작하면서도 최소한의 패킷 지연시간을 보장해주는 역할을 한다.



〔그림 3〕 네트워크 인터페이스 카드에서의 NIC_Cookie 연동 인터페이스

3.3.2 NIC_Cookie 연결 테이블 구조

NIC_Cookie의 연결 테이블은 TCP 연결 정보들을 저장하는 가장 중요한 자료구조이다. 이 테이블은 기본적으로 〈소스 IP 주소, 소스 TCP 포트, 목적지 TCP 포트〉을 단위로 연결 상태, TCP 시퀀스 넘버 차이 값, 타임 스탬프 등이 저장된다. 연결 테이블은 빠른 검색을 위해 해쉬 함수를 사용하고, 4-set associative cache 구조를 이용하여 해쉬 충돌을 줄이고 있다. 해쉬 함수는 CRC, Shift-Add등의 함수



〔그림 4〕 TCP 연결 관리 테이블

를 사용하며, 64비트(소스 IP 주소, 소스 TCP 포트, 목적지 TCP 포트) 입력으로 14비트 메모리 주소를 생성한다. 그림과 같이 하나의 연결 관리 엔트리는 128비트로 구성되며, 8메가비트의 SRAM을 사용할 경우 16K개의 엔트리를 저장할 수 있다.

테이블에 저장되는 하나의 연결은 그림과 같이 4워드로 구성된다. SRAM은 한 clock에 32비트 단위로 처리할 수 있으므로, 하나의 패킷에 대해 읽고 쓴다면 4clock(read) + 1clock(write) = 5 clock이 필요하다. 패킷 처리를 파이프 라인 방식으로 하므로 최소 패킷인 64바이트(데이터 depth = 16비트이므로 32clock 필요)을 처리하는데도 성능상의 문제가 없다.

3.3.3 Cookie 값 관리

Cookie 값 관리는 클라이언트의 SYN 패킷을 검증하는 수단으로 이용된다. Cookie 에 해당되는 ISN과 ACK의 시퀀스 넘버를 해쉬 값으로 확인을 하는 절차가 타이밍 성능에 제약을 받을 수 있기 때문에 연결 관리와 병렬로 동작시켜 그 결과 값을 동기화하도록 한다. 해쉬 함수의 선택은 기본적으로 MD5 해쉬를 기반으로 하도록 하며 고속화의 타이밍 문제가 발생될 시에는 유사 해쉬 함수로 대신하는 방안으로 설계한다.

〔그림 2〕처럼 원격 TCP 클라이언트로부터 SYN 패킷을 수신하였을 경우, 1) ISN (Cookie)을 생성한다. Cookie 생성을 위하여 시스템에는 기본적인 정보 즉, Secret Key1, Secret Key2, 8개 엔트리를 가진 MSS 테이블과 천천히 증가하는 카운터를 유지한다. MSS 테이블은 통신상의 MTU를 결정하기 위한 TCP 헤더옵션인 MSS(Maximum Segment Size) 필드를 선택적으로 수용하기 위함이다.

NIC_Cookie 에서는 생성된 ISN을 Cookie로 이용하여 SYN_ACK 패킷을 TCP 클라이언트로 송신한다. 이후, ACK 패킷이 수신되었을 때, 2) Cookie 검증 과정을 수행한다. 이 검증 과정은 Cookie 생성 동작과 동일하다. (ACK 시퀀스 넘버 - 1) 값이 소스 IP 주소, 소스 TCP 포트, 목적지 IP 주소, 목적지 TCP 포트 등의 패킷 정보와 일치하면 유효하다고 판단한다.

3.3.4 첫 번째 패킷 지연 시간 (T_{ph})

TCP 클라이언트로 수신된 첫 번째 패킷은 즉시 전달하지 않고, 고속 프락시 TCP 연결 관리가 서버 측의 TCP 주체와 정상 연결을 설정하는데 걸리는 시간(T_{ph}) 동안 패킷 버퍼에 보관하여야 한다. 이는 〔그림 2〕의 6)을 수행하기 위한 것으로써, 패킷 지연 전달 기능은 클라이언트로부터 생성된 연결을 통해 전달되는 첫 번째 패킷을 임시로 저장하였다가, 서버와의 3-way 핸드 쉐이킹이 완료되는 시점에 저장된 패킷을 서버로 송신한다. 패킷 지연 전달 기능을 설계함에 있어 가장 주의해야 할 사항으로는 패킷 지연 FIFO의 크기와 패킷 지연 시간의 설정이다. 서버의 상황에 따라, 3-way 핸드 쉐이킹의 시간이 지연될 수 있으며, 이에 따른 FIFO full이 발생할 수 있으므로, 클라이언트로부터의 첫 번째 패킷이 유실될 경우 서비스의 지연이 불가피해 지게 된다. 특히, 응용 계층 DDoS 공격은 공격 패킷의 대부분이 연결 후, 첫 번째 패킷임을 고려해 볼 때에 주요한 설계 개념이 필요하다.

즉, 패킷 지연 전달 기능은 패킷 지연 FIFO를 통하여 패킷을 임시로 저장할 있어야 한다. 이는 FPGA 내의 블록 메모리(RAMB16)을 사용하여 FIFO를 구성한다. 예를 들어, 패킷 지연 FIFO에서 out 될 ISN 값이 A라고 하고, SYN_ACK에서 수신된 acknowledge 시퀀스 넘버(ISN+1)가 A' 이라고 할 때, 패킷 지연 전달 기능에서 문제가 발생하는 경우는 다음과 같다.

정리 1. A' 의 SYN_ACK 패킷이 도착하지 않을 경우(서버의 무응답)

정리 2. A' 의 SYN_ACK 패킷이 주어진 시간(패킷 지연 시간)에 도착하지 않을 경우(서버의 지연)

정리 3. A' 의 SYN_ACK 패킷이 도착했을 경우, FIFO의 값이 A가 아닌 경우(FIFO의 강제 flush-out)

위의 3가지 경우 모두, 패킷 지연 시간의 값이 FIFO가 full이 나지 않을 만큼의 적정한 값으로 정해져야 하고, FIFO-out 될 ISN값이 SYN_ACK의 A' 값과 다를 경우는 이미 flush-out 된 경우(패킷 유실)이므로, 해당 SYN_ACK에 대한 패킷 지연 전달 과정은 생략된다. 이 경우 클라이언트의 TCP 재전송 메커니즘에 의해 동작될 것이다.

3.3.5 시퀀스 넘버 동기화 및 Checksum 재계산

NIC_Cookie의 동작에서 성능 저하에 치명적인 영향을 미칠 수 있는 부분이 TCP 시퀀스 넘버의 치환과 이로 인해 발생되는 TCP Checksum의 재계산이다. NIC_Cookie에서는 클라이언트의 유효성 검증으로 사용되었던 Cookie 값이 서버에서 생성한 ISN과 동일하지 않기 때문에, 이 값들의 차이 값을 연결 테이블에 저장하고, 이 차이 값을 바탕으로 해당 연결에서 전달되는 모든 TCP 패킷의 시퀀스 넘버와 ACK 넘버는 치환되어야 한다. 클라이언트에서 서버로의 데이터일 경우, 〈ACK 넘버 = ACK 시퀀스 넘버 - 차이 값〉으로, 서버에서 클라이언트로 데이터 일 경우, 〈시퀀스 넘버 = 시퀀스 넘버 - 차이 값〉으로 새롭게 계산되고, 계산된 값이 패킷 전달 시점에 치환된다. 이와 더불어, Checksum 계산 로직이 병렬 프로세스로 동작하여, 치환되는 시퀀스 넘버와 함께 계산되어, 최종적으로는 TCP Checksum 역시 포워딩 시점에 치환된다. 이 모든 과정은 주어진 시간(800clock)내에 계산이 끝나고 전달 시점에 파이프라인 선로 속도로 치환이 이루어진다.

IV. NIC_Cookie 구현 및 평가

4.1 NIC_Cookie 구현

NIC_Cookie 의 기능의 구현환경은 Xilinx 사의 FPGA(XC5VFX70T_FF1136 패키지)를 사용하여 로직 디자인 되었다. Xilinx FPGA에서 제공하는 TEMAC을 이용하여 Gigabit Ethernet MAC 계층을 접속하였고, 이는 PHY 기능과 MAC 계층을 모두 지원하며 타 기능 블록과 FIFO로 인터페이싱한다. 외부 물리접속 규격이 optic type의 SFP 인 경우, PHY 계층이 TEMAC에 포함되어 있고, copper type인 경우에는 독립된 PHY Chip을 FPGA 외부에 배치를 한다. TEMAC은 Xilinx Core Generator로

〔그림 5〕 NIC_Cookie가 구현된 PCI-E 타입의 네트워크 인터페이스 카드

Ethernet MAC wrapper를 구성하여 사용한다. 이렇게 생성된 TEMAC library는 NIC_Cookie 로직과 FIFO로 인터페이싱한다. NIC_Cookie에서 Host CPU와 고속 데이터 전달을 위하여 PCI Express 규격으로 접속하며, 이는 Xilinx에서 제공하는 Hard Core library를 이용하여 PCI-E 4x 규격을 활용한다. PCI-E Hard Core library와의 접속은 FIFO 접속을 통하여 데이터 전달을 수행한다. 2Gbps 성능을 처리하기 위하여 SRAM 접속은 125Mbps의 32bit 데이터 버스를 사용하였으며, NIC_Cookie내의 주요 로직은 SRAM과 동일한 Clock 도메인으로 구현되었다.

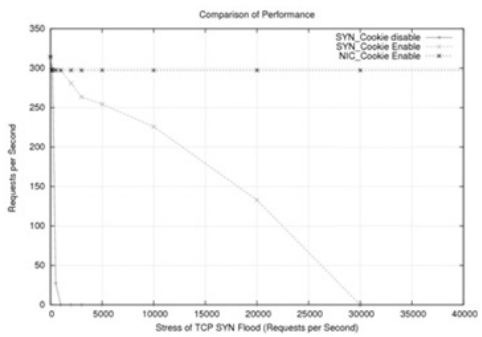
4.2 NIC_Cookie의 평가

앞 절에서도 언급한 바와 같이 NIC_Cookie의 설계 개념은 i) TCP 플러딩 공격에 대해 미탐과 오탐이 거의 존재하지 않는 최종적인 대응 방법으로써, ii) 플러딩 공격 상태에서도 메인 CPU 점유율이 0% 상태로 1Gbps 선로 속도를 보장하면서, iii) 즉시 공격 패킷 차단 대응이 가능할 수 있도록 하는 것이다.

이와 같은 NIC_Cookie의 첫 번째 장점으로는 TCP 연결 관리를 위한 자원 재 할당이 일어나지 않기 때문에, 자체적으로 DoS 공격에 강하다. NIC_Cookie에서 관리하고 있는 연결 관리 테이블은 4-set associated 테이블로써 인덱스 해쉬 값이 충돌될 경우, 4개의 관련 엔트리 중 한 개가 선택이 되고 4개의 엔트리 모두 사용될 경우 LRU(Least Recently Used) 개념으로 가장 최근까지 사용되지 않았던 엔트리를 Cache-out 하는 방식이므로 해쉬 인덱스 충돌 문제와 엔트리 고갈 문제를 해결하고 있다. 두 번째 장점으로는, 메인 서버의 OS의 종류와

상관없이 네트워크 인터페이스 카드 레벨에서TCP 보호 기능과 TCP 연동 기능을 수행하고 있으므로 서버 OS에 성능 저하없이 투명하게 그 기능을 제공하고 있고, 네트워크의 구성 등에 영향을 전혀 받지 않는 장점이 있다. 세 번째 장점으로는 실시간 대응을 함에 있어 패킷 단위로 공격 시도 TCP 연결을 차단하므로 NAT 사용자 그룹을 포함한 정상적인 연결 시도에 대하여 잘못된 대응을 할 가능성이 없어진다. 또한 SYN 플러딩 등을 대역폭 제한 등으로 대응을 접근하는 방식에 비해 오탐율이 zero에 가깝다고 할 수 있다. 네 번째 장점으로는 무엇보다도 중요한 성능적인 이슈이다. 네트워크 인터페이스 카드에서 NCI_Cookie 기능 추가로 발생하는 로직 지연시간은 패킷 길이와 상관없이 상수 값으로 6.8 micro-sec (850Clock * 8ns)이다. 앞 절에서 언급된 기능들을 이용하여 위의 세 가지 장점을 유지하면서 추가적으로 발생하는 기능 로직 지연 시간이 $7x10^{-6}$초 이하를 보장한다는 점이다.

아래 실험치는 일반적인 리눅스 CentOS에 Apache Web 데몬을 설치한 후, SYN 플러딩 공격 상황에서 SYN Cookie 비활성화 및 활성화 상태와 NIC_Cookie 활성화 상태에서의 서버의 처리 성능에 대한 결과치이다. 서버의 Apache 버전은 2.2.3 버전이고 성능 측정은 Apache benchmark (#ab -n 1000) 툴을 이용하였다. 〔그림 5〕와 같이 서버에서 SYN Cookie를 비활성화 할 경우, 초기에 평균 300여개의 HTTP request를 처리하던 Apache 서버가 초당 500개정도의 SYN 플러딩 공격에 서비스가 줄어 초당 1000개의 SYN 플러딩 공격에는 서비스 마비가 되었다. 리눅스 OS에서 제공하는 커널 레벨의 SYN Cookie를 활성화 한 경우는 초당 30000개의 SYN 플러딩 공격에서 서비스가 마비되었지만, 그림과 같이 NIC_Cookie를 활성화하면 SYN 플러딩 공격에 상관없이 웹 서버가 일정하게 서비스를 제공할 수 있음을 볼 수 있다.



〔그림 6〕 SYN 플러딩 공격 상황에서 SYN_Cookie와 NIC_Cookie의 성능 비교

V. 결 론

본 논문은 악의적인 사용자의 TCP 연결을 근본적으로 차단하여 유효한 TCP 연결만을 전달해주는 기능을 가진 네트워크 인터페이스 카드에서의 고속 TCP 연동 로직 구현에 대하여 기술하였다. 이는 악의적인 TCP 사용자의 행위에 대하여 네트워크 인터페이스 카드에 내장된 고속 TCP 연동 제어 기법(NIC_Cookie)을 통하여 비정상적인 TCP 행위를 근본적으로 차단함으로써 서버의 CPU 부하 증가가 전혀 없도록 서버를 보호하게 한다. 본 논문에서 제안한 방법의 주요 장점은 통상 네트워크 로드 밸런싱 기능으로 인한 비대칭적 연결 설정 환경에서도 TCP 플러딩 공격에 대해 최종적인 공격 방어책이 될 수 있고, 리눅스 및 윈도우 등 OS 제약 사항에 관계없이 투명하게 제공될 수 있으며, TCP 연결 설정 연동에 있어 자원 고갈 문제가 근본적으로 해결되는 방식이므로 자체적으로 DoS 공격에 내구성을 가지고 있고, TCP 계층의 플러딩 공격뿐만 아니라 이를 이용한 응용 계층 DDoS 공격 방어에도 확장될 수 있는 구조이며, 무엇보다도 가장 중요한 점은 서버의 고유 서비스를 제공함에 있어 성능적 오버헤드가 전혀 없는 구조로 고성능 공격 방어 처리를 보장한다는 것이다.

메인 CPU의 부하를 최대한 줄이기 위하여 다양한 형태의 Offload 엔진 기술들이 개발되고 있는 시점에서 DDoS 공격 방어를 위한 NIC_Cookie 기능이 네트워크 인터페이스 카드 상에서 구현된 것은 Security Offload 엔진 구현이라는 개념적인 의미를 부여할 뿐만 아니라 향후, 다양한 보안 엔진을 Offload 하기위한 기반 기술로 활용이 가능하리라 판단된다.

참고문헌

〔1〕 R. Stewart and Q. Xie, "Stream control transmission protocol(SCTP): a reference guide," Addison Wesley Professional, New York, NY, 2001.

〔2〕 Wei Zhou Lu, and Shun Zheng Yu, "A

HTTP flooding detection method based on browser behavior," 2006 International Conference on IEEE Computational Intelligence and Security, vol 2, pp. 1151-1154, Nov. 2006.

〔3〕 D. Moore, G. Voelker, and S. Savage, "Inferring internet denial of service activity," Proceedings of USENIX Security Symposium '2001, Aug. 2001.

〔4〕 Check Point Software Technologies Ltd. "TCP syn flooding attack and the firewall-1 SYNdefender," 1997.

〔5〕 C. Smith and A. Matrawy, "Comparison of operating system implementations of SYN flood defenses (cookies)," Communications, 2008 24th Biennial Symposium on, pp. 243-246, 2008.

〔6〕 J. Lemon, "Resisting SYN flood DoS attacks with a SYN cache," In Proceedings of the BSDCon 2002, Feb. 2002.

〔7〕 J. Udhayan and R. Anitha, " Demystifying and rate limiting ICMP hosted DoS/DDoS flooding attacks with attack productivity analysis," Advanced Computing Conference, IACC 2009. IEEE International, pp. 558-564, Mar. 2009.

〔8〕 Netscreen 5400 Firewall Appliance, http://www.juniper.net/us/en/products-services/security/netscreen/

〔9〕 OpenBSD Packet Filter Manual, ftp://ftp.openbsd.org/pub/OpenBSD/doc/pf-faq.pdf

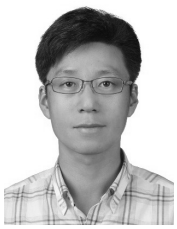
〔10〕 D.J. Bernstein, SYN Cookies. http://cr.yp..to/syncookies.html, 2007.

〔11〕 Eddy, W. "Defenses against TCP SYN flooding attacks," Cisco Internet Protocol Journal vol.8, no. 4, Dec. 2006.

〔12〕 W. Eddy, "TCP SYN flooding attacks and common mitigation," RFC 4987, 2007.

〔13〕 L. Ricciulli, P. Lincoln, and P. Kakkar, "TCP SYN flooding defense," In Comm. Net. and Dist. Systems Modeling and Simulation Conf. (CNDS′ 99), 1999 Western MultiConf. (WMC′ 99), Jan. 1999.

〈著者紹介〉



김 병 구 (Byoungkoo Kim) 정회원
1999년 2월: 성균관대학교 정보공학과 졸업
2001년 8월: 성균관대학교 전기전자및컴퓨터공학과 석사
2001년 8월~현재: 한국전자통신연구원 선임연구원
2010년 3월~현재: 성균관대학교 전자전기컴퓨터공학과 박사과정
〈관심분야〉 네트워크보안, 악성코드 분석 및 탐지, 침입탐지 및 대응

김 익 균 (Ikkyun Kim) 정회원
1994년: 경북대학교 컴퓨터공학과 졸업
1996년: 경북대학교 컴퓨터공학과 석사
2009년: 경북대학교 컴퓨터공학과 박사
1996년~1999년: 한국전자통신연구원 연구원
2000년~2001년: (주) 팍스콤 선임연구원
2004년~2005년: 미국 Purdue University 객원연구원
2001년~현재: 한국 전자통신 연구원 책임연구원, 보안관제기술연구팀 팀장
〈관심분야〉 네트워크보안, 컴퓨터네트워크, 클라우드컴퓨팅, 스마트그리드 보안

김 대 원 (Daewon Kim) 정회원
2003년 2월: 경북대학교 전자공학과 졸업
2005년 2월: KAIST 전자공학과 석사
2005년 3월 ~ 현재: 한국전자통신연구원 선임연구원
〈관심분야〉 네트워크보안, 악성코드 분석 및 탐지, 침입탐지 및 대응

오 진 태 (Jintae Oh) 정회원
1990년 2월: 경북대학교 전자공학과 졸업
1992년 2월: 경북대학교 전자공학과 석사
2011년 2월: 충남대학교 컴퓨터공학과 박사
1992년 2월~1998년 2월: 한국전자통신연구원 선임연구원
1998년 3월~1999년 1월: 미국 MinMax Tech. 연구원
1999년 2월~2001년 10월: 미국 Engedi Networks. Director
2001년 10월~2003년 1월: 미국 Winnow Tech. Co-founder, CTO 부사장
2005년~2009년: 한국전자통신연구원 보안게이트웨이연구팀 팀장
현재: 한국전자통신연구원 책임연구원
〈관심분야〉 네트워크보안, 공격 시그니처 자동생성기술, 보안하드웨어기술

〈著者紹介〉

장 종 수 (Jong-Soo Jang) 종신회원
1984년: 경북대학교 전자공학과 졸업
1986년: 경북대학교 대학원 전자공학과 석사
2000년: 충북대학교 대학원 컴퓨터공학과 박사,
2000년~2003년: 한국전자통신연구원 네트워크보안구조팀팀장
2004년~2008년: 한국전자통신연구원 그룹장
1989년~현재: 한국전자통신연구원 책임연구원
2006년~현재: 대검찰청 디지털수사자문위원회 위원
2007년~현재: OSIA 이사
2008년~현재: 방송통신위원회 인터넷 정보보호협의회 안전인터넷분과 위원
〈관심분야〉 네트워크 보안, 정책기반보안관리, 비정상트래픽탐지, 유해정보차단

정 태 명 (Tai-Myoung Chung) 종신회원
1984년 9월~1985년 5월: 일리노이주립 대학교, 연구조교 (Chicago IL, 미국)
1985년 8월~1987년 12월: Waldner and Co., 엔지니어 (Oak Brook, IL, 미국)
1987년 12월~1990년 8월: BBN 연구소, 연구원 (Cambridge MA, 미국)
1991년 8월~1995년 8월: 퍼듀 대학교, 연구조교 (W. Lafayette IN, 미국)
2005년 2월~2007년 1월: 성균관 대학교 정보통신처, 처장
1995년 9월~현재: 성균관 대학교, 교수
2010년 ~ 현재: 한국정보처리학회 학회장
2011년 3월~현재: 성균관대학교 소프트웨어공학과 학과장
〈관심분야〉 네트워크 보안, 컴퓨터 네트워크, 실버인터넷, 모바일 시큐리티