제 18 회 한국정보처리학회 추계학술발표대회 논문집 제 9 권 제 2 호 (2002.11)

리눅스 커널에서 구현한 웹서버 암호화 가속 기법에 대한 연구

황준*, 민병조*, 남의석*, 김학배*, 장휘**
*연세대학교 전기전자공학과
**㈜엔버젼스
e-mail : hbkim@yonsei.ac.kr

A study on Secure Socket Layer WEB Acceleration using Linux Kernel Thread

Jun Hwang*, Byungjo Min*, Euiseok Nahm*, Hagbae Kim*, and Whie Chang**
*Dept. of Electrical and Electronic Engineering, Yon-Sei University
**Nvergence, Inc.

요 약

인터넷 전자 상거래의 폭발적 증가와 더불어 개인 및 기업의 정보가 온라인 상으로 유출되는 경우가 증가하고 있다. 이에 따라, 새로운 하드웨어의 추가 없이 프로토콜 및 알고리즘의 변화에 유연한 인터넷 보안방법이 요구되고 있다. 본 논문에서는 사용자 영역과 상관없는 커널 스레드를 사용하고 커널 영역으로 포팅된 라이브러리를 참조하여 사용자의 웹 페이지 요청을 처리함으로써 응답시간과 서버 부하를 감소시키는 새로운 SSL(Secure Socket Layer) 처리 구조를 제안한다.

1. 서론

최근 인터넷 상에서의 온라인 주식 거래, 전자 상거래 등이 크게 증가함에 따라 관련된 기업, 기관, 그리고 개인들에게 웹 트랜잭션에 관한 보안성 확보가 중대한 문제로 대두되고 있다. 그 결과, IPSec (IP Security), SET (Secure Electronic Transaction), SSL (Secure Socket Layer) 과 같은 다양한 인터넷 보안 관련 프로토콜이 제안되어 왔다. 그 중, SSL [1]이 가장 널리 사용되고 있다. 이 프로토콜은 웹 브라우저 회사였던 넷스케이프에 의해 만들어졌으며 익스플로러 및 넷스케이프등의 주요 웹 브라우저가 이 프로토콜의 스택을 완벽하게 지원하여 전세계 전자 상거래에 관한 웹 트랜잭션의 근간이 되어 왔다. 그러나 SSL 을 이용한 웹 서버의 경우 사용자들이 서버에 접속할 때, 서버부하가 크게 증가되어 전반적인 웹 서비스 속도가 저하되는 문제가 야기된다. 평문의 HTTP (Hyper Text Transfer Protocol)이 2 번의 메시지 핸드쉐이킹을 거치는 요청/응답과정이, SSL 세션을 거칠 경우 최소 3 번에서 6 번까지 증가하기 때문이다[1]. 또한, SSL 의 암호화에는 기밀성과 무결성을 높이기 위해 RSA 등 공개키 암호 시스템(PKI)에 기반한 암호화 알고리즘이 사용되는데 이때, 매우 큰 수($2^{561231024}$)에 관한 나머지 연산 및 소인수 연산을 필요로 하기 때문에 웹 서비스 속도 저하의 중요한 원인이 된다[2].

SSL 처리시 성능을 높이는 기존의 방법으로 SSL 연산작용을 특정 하드웨어에 전담시키는 방법[3]과 독립형 SSL 장비를 서버에 추가하는 기술 등이 많이 연구되어 왔다. 첫번째 방법은 관리 비용 및 방법에서 효율적이지 못하며, 미래의 프로토콜의 변화와 알고리즘의 추가, 변경 및 업데이트 등에 유연하지 못한 단점이 있다. 두번째 방법은 SSL 장비 자체가 전체 웹 서비스 속도의 병목이 될 수 있으며 웹 서버에게 인증 정보등이 공개되지 않아 서비스가 투명하지 못하다는 단점이 있다.

본 논문에서는 리눅스 커널에서 구현한 웹 서버 암호화 가속기법에 관해 논의한다. 그 구조는 커널 스레드를 통하여 커널 영역으로 포팅된 SSL 라이브러리를 이용하고 요청을 처리하는 것이다. 결론에서는 Webbench[5]를 통한 성능평가를 통하여 기존에 비하여 응답시간과 서버의 부하가 감소함을 알 수 있다.

2. 기존 웹 서버의 SSL 처리방식

2.1. 기존 SSL 단의 구조적 문제

대부분의 웹 서비스에서의 SSL 처리 구조는 그림 1

에서 보이는 것처럼, 실제 웹 요청을 처리하는 아파치(Apache)와 SSL 요청처리 아파치 모듈 프로그램인 modSSL[7], 그리고 실제 SSL 에 관한 암/복호화 알고리즘 라이브러리인 OpenSSL[5]로 이루어 진다. 일반적인 HTTPS 서버는 SSL 을 지원하기 위해서 유저레벨에서 TCP/IP 소켓을 열고 그 위에서 SSL 라이브러리를 로딩하고, 그 위에 HTTP 를 제공하는 웹 서버를 올리는 방법으로 구현된다. 이 경우, HTTP 보다 훨씬 복잡한 HTTPS 는 커널 스위칭이 많아지기 때문에 CPU 오버헤드가 발생한다. 또한 유저레벨에서는 OS 의 멀티태스킹으로 SSL 의 복잡한 알고리즘 연산 수행도중 다른 프로세스에 의해 우선순위를 빼앗겨 스케줄링을 당할 수도 있기 때문에 캐시의 효용성을 잃어버릴 수 있다.

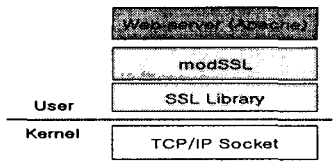

그림 1. 기존의 SSL 프로그램 계층 구조

2.2. 사용자 영역 스레드 사용 문제

기존의 웹서버 처리방식에서 브라우저를 통한 사용자들의 요청이 서버에 도달하면 각 요청에 대해 처리 스레드를 실행하여 각 스레드가 하나의 요청을 전담한다. 이 구조에서는 동시 접속자 수가 많아지면 스레드 수가 증가하여 서버의 메모리 할당이 증가되고 각 스레드 마다 할당되는 CPU 점유시간은 감소하게 된다. 결과적으로, 서버의 요청 처리 능력이 감소하고, 사용자의 응답시간은 늦어지게 된다[6]. 이때 사용되는 스레드는 리눅스 커널 스레드와는 달리 사용자 영역에서 런칭되어, CPU 에 대한 독점적, 비선점적 사용이 불가능하다. 그러므로 같은 아파치 프로그램에서 만들어진 스레드들이 CPU 에 대한 같은 수준의 우선순위를 갖게 되어 CPU 에 의해 모두 같은 시간을 할당받게 되고 많은 스레드가 동작중인 경우 CPU 스위칭 시간이 증가하여 처리속도에 큰 저해요소로 작용한다. 또한, SSL 세션 기능을 사용할 경우, 사용자 레벨에서 최소 2 번의 receive 시스템 콜과 2 번의 send 시스템 콜을 수행해야만 접속이 이루어 지기 때문에 커널모드 스위칭 오버헤드가 존재한다[2].

3. 제안된 SSL 처리 방식

3.1. SSL 요청 처리구조의 변화

본 논문에서 제안한 구조는 사용자 영역 멀티 스레드 처리 구조를 커널 영역 단일 스레드 구조로 변화시키고 중요한 코드 블록은 비선점화 하여 코드 수행중 CPU 를 독점하여 요청을 처리한다. 각 요청마다 새로운 스레드를 실행시키는 구조 대신 요청을 받아들이는 커널 스레드와 요청을 처리하는 커널 스레드 하나씩 갖는 구조로 구현된다. 즉, 각 요청이 TCP 포트를 거쳐 커널로 들어오면 받아들이는 커널 스레드가 큐에 저장하고, 저장순으로 처리 커널 스레드가 라이브러리를 이용하여 처리하게 된다. 직관적으로, 단일 처리 스레드에 하나의 큐를 사용해서 멀티 스레드보다 처리 속도가 느려 보일 수 있지만, 사용자 스레드 간의 문맥(context) 스위칭의 지연시간보다 제안된 구조의 지연시간이 짧기 때문에 실제로 속도는 빨라진다. SSL 요청 처리단은 한 개 이상의 CPU 를 가진 경우 최적의 성능을 발휘할 수 있도록 각 CPU 당 하나의 커널 스레드를 할당하고 SSL 접속을 관리한다. 결과적으로, SSL 요청 처리단은 SSL 처리를 담당하는 커널 스레드의 개수가 CPU 수 만큼이기 때문에 많은 요청이 쇄도해도 너무 많은 프로세스 동작으로 인한 CPU 및 메모리 오버헤드가 없다. 그림 2 는 기존방식과 제안된 방식의 SSL 요청 처리 구조의 차이를 표현한 것이다.

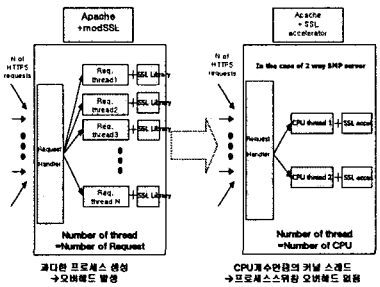

그림 2. 기존 SSL 처리 방식과 본 논문에서 제안된 방식의 비교

3.2. 사용자 영역 스위칭 감소

시스템 콜은 사용자 영역에서 커널 영역으로 특정 동작을 수행시킬 때 사용하는 운영체제의 구현 방식을 의미한다. 읽기/쓰기 등과 같은 파일/네트워크 연산에 관련된 시스템 콜의 경우, 커널과 사용자 각 영역에서의 메모리 접근 방식 차이로 인해 2,3 회의 버퍼 복사가 추가되어 커널 레벨에서 직접 읽기/쓰기 연산을 하였을 때보다 메모리 오버헤드와 처리시간 지연 등이 발생한다. 기존의 SSL 처리구조가 시스템 콜을 사용한데 반해, 제안된 구조는 직접 커널 함수를 사용하므로 시스템 콜을 사용할 필요가 없다.

3.3. 커널 스레드 웹 가속 구조

웹 서버의 사용자 영역 스레드 사용시 발생하는 문제를 없애기 위하여, 아파치의 전단에 별도의 웹 서버 가속 구조를 추가한다. 이 가속 구조물은 사용자 영역의 멀티 스레드대신 커널 영역의 단일 스레드를 사용하고 클라이언트 요청을 받아 단일 큐에 저장하는 구조를 갖는다. CPU 개수당 하나의 커널 스레드가 큐에 저장된 요청을 할당받고 기존에 캐시된 요청인지 판단하여 사용자 영역에 있는 아파치와의 통신을 하게 된다[8]. 별도의 웹 서버 가속 구조는 제안된 SSL 요청 처리구조와 커널 스레드, 큐구조 사용과 유사한 방식을 따르며 또한 유사한 성능 향상을 가져온다.

4. 프로그램 구조

SSL 커널 가속 모듈은 사용자 인증, 암/복호화, 메시지 무결성 등을 지원하는 SSL 라이브러리 모듈과

클라이언트와의 통신, 웹 서버와의 연동, 쓰레드 및 세션 관리 등의 일을 하는 SSL 요청 처리 모듈로 구성된다. SSL 라이브러리 모듈은 보안 알고리즘을 수행하기 위한 코드들이며 오픈 소스인 OpenSSL[5]을 기반으로 구성된다. SSL 라이브러리 모듈은 OpenSSL 이 커널에서 정상적으로 동작하게 하기 위해 사용자 레벨에서는 존재하지만 커널 영역에서 존재하지 않는 외부 함수나 시스템 콜들의 구현을 포함하고 있다. SSL 요청 처리 모듈은 HTTPS 접속을 받아서 서로의 데이터를 협상(negotiation)하고, 클라이언트에게 온 메시지를 복호화하며 웹서버로부터 전해진 데이터를 암호화해서 다시 클라이언트에게 보내는 역할을 한다. 다음 그림은 동작 구조를 계층별로 요약한 것이다.

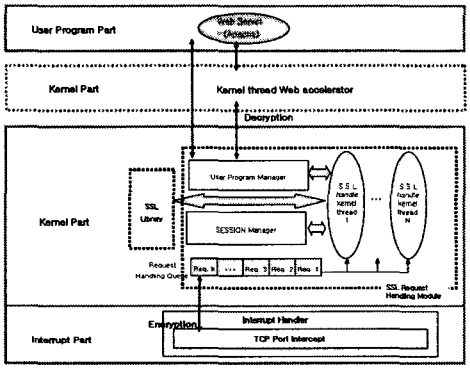

그림 3. SSL 커널 스레드 가속기의 동작 계층 구조

OpenSSL 은 SSLeay 를 기반으로 한 공개용 암호화 라이브러리이다. 미국에서 상용으로 개발된 RSA 알고리즘이 완벽하게 호환가능하며 프리웨어로 소스코드 및 각 암호화 알고리즘 자체가 완벽히 공개되어 있다[5]. 본 연구에 사용된 이유도, 오픈 소스라는 점과 많은 아파치와 같이 사용되면서 안정성과 호환성에 대한 검증을 받았다는 것이다. 약 40 가지 이상의 암호화 알고리즘과 4 가지 이상의 디지털 서명 알고리즘, 3 가지 이상의 메시지 인증 코드(MAC) 알고리즘이 구현되어 있다. 본 연구에서 사용한 OpenSSL 은 다음 표와 같다.

표 1. OpenSSL 의 세부 사양

이 름	OpenSSL-0.9.6d
Protocol Support	SSLv2, SSLv3, TLSv1
PKI 알고리즘	RSA, EDH, DH, etc
Conventional Encryption Algorithm	DES,3DES,RC4,RC2,RC5,etc
Message Authentication Code Algorithm	MAC, HMAC, SHA, SHA1,etc

본 연구의 커널 SSL 라이브러리 모듈은 보안 알고리즘을 수행하기 위한 코드들이다. 이 모듈은 OpenSSL 을 기반으로 구성되고 OpenSSL 이 커널에서 정상적으로 동작하게 하기 위해 커널 레벨로 구현된 것이다.

실제 사용자 레벨에서의 SSL 라이브러리 프로그램은 read, write, send, recv 와 같은 시스템 콜을 통해 커널과 인터페이스를 갖는데 반해, 커널 레벨의 SSL 라이브러리를 구현할 경우, 이러한 시스템 콜을 사용하지 않아도 되고 CPU 오버헤드를 제거하며 그것을 독점적으로 사용하여 입출력 동작의 속도를 크게 증가시킬 수 있다. 그러므로 CPU 집중적인 SSL 프로토콜의 특성상 CPU 및 네트워크 데이터 처리 속도에 중요한 recv, send 등의 함수를 커널 레벨에서 수행함으로써 전체 성능향상에 도움이 된다. 이러한 구현 방식을 선택할 경우, 가장 중요한 문제는 커널에서 완벽하게 동작할 수 있도록 기존의 SSL 라이브러리를 커널 레벨로 포팅해야 하는 것이다. 본 논문에서는 기존 사용자 레벨에서 gcc 라이브러리를 통해서 사용되는 OpenSSL 의 라이브러리를 커널 레벨에서 동작할 수 있도록 바꾸었다. 이 구현을 위해서는, 기존 OpenSSL 라이브러리 소스에서 사용자 영역의 여러 라이브러리와의 관련성을 없애고 라이브러리 전체가 모두 커널에서 동작하도록 커널 레벨로 수정해야 한다. 또한, 사용자 영역 라이브러리와의 연관성(dependency)이 없기 때문에, 메모리/파일/시간/네트워크 오퍼레이션에 대한 함수 등이 커널에서는 존재하지 않으므로 이 기능들이 커널에서 동작되도록 직접 구현하여야 한다.

결과적으로 표준(standard) 입출력, 시간에 관한 라이브러리(time, localtime, gmtime), 메모리 할당 및 해제에 관한 함수호출 그리고 네트워크 소켓에 관한 함수호출들을 커널레벨에서 구현하여 SSL 라이브러리에서 사용할 수 있도록 해야한다. 이러한 커널 레벨의 SSL 라이브러리 구현 과정은 다음과 같이 요약된다.

- 표준 헤더 파일들을 커널 헤더 파일로 변환한다.
- 표준 라이브러리 및 공유 라이브러리와의 연결(linking) 및 의존도(dependency)를 없앤다.
- 커널에 구현되어 있는 API 만을 사용하여 SSL 라이브러리를 구현한다.
- Apache 와 같이 멀티 프로세스에 맞춰진 SSL 라이브러리 구현 방식을 커널 스레드방식으로 구현한다.

5. 성능 평가

제안된 SSL 가속 커널 모듈의 성능평가를 위해 기존 SSL 웹 서버(Apache + modSSL + OpenSSL)와 본 가속 모듈을 탑재한 웹 서버(Apache + SSL 가속 커널 모듈)의 클라이언트 요청 처리량을 비교하였다.

성능 평가에 사용된 서버의 주요 사양은 표 2 와 같다.

표 2. 서버 주요 사양

비교 항목	서버 사양
CPU	Dual Pentium3 (933Mhz)
Memory	512 M
Hard Disk	SCSI Ultra160 LVD/SE
LAN Card	3Com Giga LAN
배포판	Redhat 7.2
SSL 라이브러리	OpenSSL0.9.6d / 본 웹가속기의 SSL 라이브러리
리눅스 커널	2.4.18

테스트 환경은 다음과 같다. 동일 하드웨어의 서버에 기존 SSL 처리구조와 본 연구에서 제안된 SSL 가

속 커널 모듈을 번갈아 탑재하고 10 대의 클라이언트를 1 대의 서버에 동일 위상(topology)으로 연결하였다. 웹서버 프로그램으로는 아파치가 사용되었고, 버전은 레드햇(Redhat) 7.2 에 기본으로 설치되어있는 1.3.20 이다. 테스트 프로그램은 ZDnet 의 Webbench 4.1 이 사용되었으며, 암호화 비트는 128bit 이다[4]. 이 프로그램의 협상 암호문(Negotiation cipher)은 EDH-DSS-DES-CBC3-SHA, 테스트한 SSL 버전은 TLS 1.0 이다.

태스트는 각각의 클라이언트가 SSL 세션에서 다음의 HTTP 요청을 보내는 경우에 대해서 이루어졌다.

1) HTTP 1.0 100%
2) HTTP 1.1 지속연결(persistent connection) 100%
3) HTTP 1.1 연속요청(pipeline request) 100%
4) HTTP 1.0/1.1 혼합

HTTP 1.0 에 비교하여 지속 연결과 연속 요청 기능이 추가된 HTTP 1.1 상에서 본 가속 모듈이 어느 정도 유효한 지에 대하여 테스트 하였다. 클라이언트를 1 기부터 10 기까지 증가시키며 부하를 주고 그에 따라 1 초당 처리되어 나온 요청의 수를 기록하였다.

1) HTTP 1.0 100% 경우

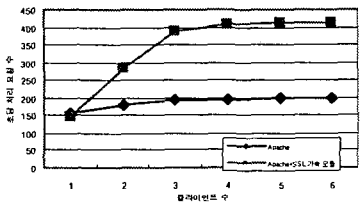

그림 4. HTTP1.0 100%의 요청시 초당 처리 요청수

2) HTTP 1.1 지속연결 100%경우

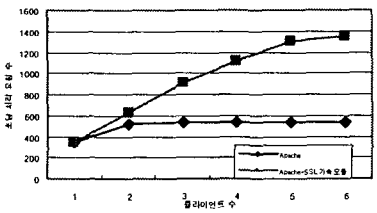

그림 5. HTTP 1.1 persistent connection 요청시 초당 처리 요청수

3) HTTP 1.1 연속요청 100% 경우

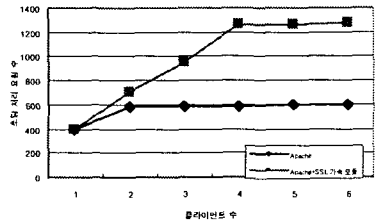

그림 6. HTTP 1.1 Pipeline 요청시 초당 처리 요청수

4) HTTP 1.0/1.1 혼합된 경우

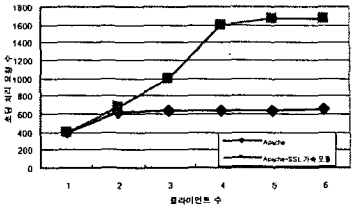

그림 7. HTTP 1.0/1.1 혼합 요청시 초당 처리 요청수

테스트 결과, 각각 최고 125%~200% 정도의 초당 처리 요청수가 증가했으며 제안된 SSL 가속 커널 모듈의 처리 성능이 기존 웹 서버 보다 뛰어남을 알 수 있다. 결국 기존 웹 가속기에 HTTPS 요청 처리 핸들러 및 SSL 라이브러리 기능을 추가함으로써, HTTPS 연결시 처리 성능을 향상시킬 수 있다.

6. 결론

본 논문에서는 복잡한 암호화 과정 및 접속시 메시지 핸드쉐이킹 증가로 웹 서버의 처리 성능이 현격히 감소하는 기존 SSL 처리의 문제점을 밝혔다. 또, 그 해결책으로 부가적인 하드웨어나 장비의 도움 없이 내부 구조를 변화시켜, SSL 웹서버의 성능을 증대시키는 커널 레벨의 SSL 가속 모듈을 제시하고, 이를 구현하였다. 주요 코드의 비선점형 실행, 시스템 콜이 필요없는 커널 스레드 및 단일 처리 큐 사용 등의 방법은 사용자와 웹 서버에 투명하게 SSL 처리 속도를 증대시켰다. 또한 일반 SSL 웹 서비스와 비교한 테스트에서도 그 이득을 명확히 확인할 수 있었다.

참고문헌

[1] A.O. Freier, P. Karlton and P.C. Kocher, "The SSL Protocol, V3.0", IETF draft at www.netscape.com/eng/ssl3/draft302.txt.

[2] K. Kant, R. Iyer and P. Mohapatra, "Architectural Impact of Secure Socket Layer on Internet Servers", Proc. IEEE 2000 International Conference on Computer Design, pp. 7-14

[3] A.J. Elbirt, W. Yip, B. Chetwynd and C. Paar, "An FPGA-based performance evaluation of the AES block cipher candidate algorithm finalists", IEEE Transacions on VLSI Systems

[4] http://www.etestinglabs.com, Ziff Davis Media Web Site.

[5] http://www.openssl.org

[6] H. Yiming, A. Nanda and Q. Yang, "Measurement, Analysis and Performance Improvement of the Apache Web Server", Proc. IEEE International Conf. Performance, Computing and Communications, 1999, pp. 261-267

[7] http://www.modssl.org

[8] J. Park, H. Lim and H. Kim, "Development of kernel thread web accelerator", IEE Electronic Letters, vol.38, no.13, June 2002-09-09, pp.672-673