L4 기반 네트워크 스토리지 보안 강화방법

박우람°, 나윤주, 류준길, 박찬익
포항공과대학교
{wizrampa°, thesky, lancer, cipark}@postech.ac.kr

Enhanced Security Network Storage System based L4

Wooram Park° Yunju Na Junkil Ryu Chanik Park
POSTECH

요 약

데이터의 양이 급격히 커지면서, 그에 따라 요구되는 스토리지의 확장 비용 및 관리의 어려움을 해결하기 위하여 네트워크 스토리지에 대한 관심이 증대되고 있다. 네트워크 스토리지는 다수의 사용자가 접근하여 하기 때문에 보안에 대한 심각한 고려가 필요하다. 보안을 강화하기 위하여 페어 키를 이용한 인증 방식을 사용하고 있는데, 이러한 소프트웨어적인 보안 방식은 시스템 보안의 결함 혹은 취약성에 의하여 키의 외부 유출이 가능하다 [11].

본 연구에서는 L4 마이크로커널[1]과 하드웨어적 보안 방식인 TPM (Trusted Platform Module)[2]을 사용하여 네트워크 스토리지 보안을 강화 시킬 방법을 제안하고자 한다. 본 연구를 이용할 경우 authenticated boot 기법[3]을 이용하여 네트워크 스토리지에서 동작할 이미지를 검증하고, 하드웨어적으로 암호화 키 값을 관리함으로써 데이터 패킷의 전송 과정에서 발생할 수 있는 소프트웨어적인 보안 방식의 취약점을 보완할 수 있다.

1. 서론

데이터가 과거의 문서 위주에서 동영상, 사진, 음악 파일과 같은 멀티미디어 데이터로 다양화 됨으로써 필요한 스토리지 용량은 급격하게 증가하고 있다. 이렇게 급속하게 데이터의 양이 증가하면서 스토리지 확장 비용 및 관리에 대한 어려움이 발생하게 되었고, 이에 따라 네트워크 스토리지에 대한 관심은 증가하고 있다.

네트워크 스토리지는 다수의 사용자가 접근하여 사용하기 때문에 보안에 대한 고려가 필요하다. 기존의 네트워크 스토리지는 리눅스나 윈도우와 같은 모놀리식커널 상에서 동작하며, 페어 키를 사용한 소프트웨어적인 암호화 인증 방식을 사용하고 있다.

모놀리식커널 (Monolithic Kernel)은 다양한 기능을 제공해줄 수 있지만, 코드가 방대하고 복잡해짐에 의하여 미처 개발자가 파악하지 못한 보안상의 오류가 자주 발생한다. 실제 네트워크가 발달하면서 이러한 보안상 취약보고와 공격수치가 매년 증가하고 있다[4].

또한 Public/Private 키를 이용한 페어 키 인증 방식이나 암호화 방식은 소프트웨어적인 인증 방식으로, 인증에 필요한 키 값은 스토리지에 저장된다. 보안상의 취약점이나 결함으로 인하여 외부로의 접근, 혹은 웜이나 스파이웨어의 침투로 인해 인증 키가 외부로 유출될 위험성이 존재하며, 실제 보안상 취약보고와 공격수치가 매년 증가하고 있는 최근의 추세에 따라 네트워크 스토리지의 보안 위협은 점차 커지고 있으며, 이에 대한 대응책도 지속적으로 나오고 있다[11].

이에 대한 대안으로 외부로 키 값이 유출되지 않는 하드웨어적인 보안 방식을 활용하여, 스토리지에 저장되어 있는 인증 키 값을 하드웨어 내에 보관되어 있는 키 값을 이용하여 암호화 함으로써 인증 키가 외부로 유출되더라도 해당 인증 키를 복호화 하지 못 하게함으로써 이러한 문제점을 보완한다.

마이크로커널 (Micro Kernel)은 커널의 기능을 최소화하여, 코드를 간결하게 함으로써 모놀리식커널에서 발생하는 보안 상의 취약점을 해결할 수 있다. 그렇기 때문에

본 연구에서는 작년에 제안한 L4 기반의 네트워크 스토리지 플랫폼[12]에 하드웨어적으로 암호화 키를 관리해 주는 TPM (Trusted Platform Module)[2]을 추가하여 네트워크 스토리지 보안을 강화 할 수 있는 방법을 제안하고자 한다.

2. 배경지식

2.1 TPM (Trusted Platform Module)

사용자가 가지고 있던 스토리지의 분실, 혹은 암호화 키의 유출과 같은 소프트웨어적인 보안방식의 취약점을 보완하기 위하여 제안된 하드웨어를 사용한 대표적인 보안방식으로 TCG (Trusted Computing Group)[5]에서 제안되었다.

각각의 TPM 칩은 제조될 때 고유의 키인 EK (Endorsement Key)와 SRK (Storage Root Key)가 주어지며, 이 키 값들은 칩 외부로 나가지 못하게 되어있다. 이러한 키 값에서 파생된 암호화 키를 사용하여, 실제 암호화에 사용된 키 값을 다시 암호화함으로써, 해당 키 값이 저장되어 있는 TPM 칩이 존재하지 않는 시스템에서는 복호화가 불가능하게 함으로써 보안을 강화할 수 있다. 최근 리눅스나 윈도우 같은 범용 운영체제에서 공식적으로 TPM을 지원하고 있으며, TPM이 탑재된 PC의 수도 2006년을 기준으로 급격하게 증가할 것으로 예측되고 있다[6].

2.2 Authenticated Boot

Authenticated Boot[3]는 TPM을 사용하여 컴퓨터의 부팅 단계별로 해시 값을 측정하여 TPM 내의 레지스터인 PCR (Platform Configuration Register)에 저장하고, 해당 값을 remote 서버와 같은 인증된 third party에 레포팅하여 올바른 값이 생성되었는지 검증 받는 부팅 프로세스이다.

이를 이용하여 응용프로그램이 동작할 시스템의 BIOS와 운영체제가 악의적인 프로그램에 의해 변경되지 않고, 정상적으로 부트 과정이 완료되었는지 여부를 판단할 수 있다.

3. 본문

본 연구에서 제안하는 플랫폼은 모놀리식커널에서 발생하는 보안성의 취약점을 최소화 하기 위하여 L4 마이크로커널을 기반으로 한다. 그리고 시스템의 안정성을 높이기 위하여 VDD (Virtual Device Driver) 기법[7][8]을 사용하여 디바이스 드라이버의 오류 시 커널 영역에 영향을 미치지 않도록 하였다.

또한 Authenticated boot를 적용하여 스토리지 시스템에서 동작하는 각 L4 컴포넌트를 검증하고, 검증 결과를 토대로 복구를 수행함으로써 스토리지 시스템이 안전한 환경에서 동작할 수 있도록 플랫폼을 제공하여 준다.

3.1 디자인

네트워크 스토리지 시스템에서 동작하는 컴포넌트들이 안전하지 못 할 경우, 스토리지에 저장되는 데이터의 안전성 또한 보장받지 못한다. 그러므로 부트 과정을 수행하기 전에 네트워크 스토리지 시스템에서 동작할 컴포넌트를 검증하고, 악의적으로 변경되었을 경우 해당 컴포넌트를 복구 해주어야 한다.

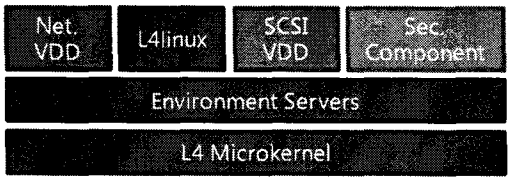
이를 위하여 본 연구의 플랫폼은 *authenticated boot*를 이용하여, 부트 과정에서 사용되는 컴포넌트들의 해시 값을 생성하여, TPM 내의 PCR에 해당 값을 기록하고 별도로 구현한 Security 컴포넌트에서 보유하고 있는 해시 리스트의 값과 비교한다.

검증결과에 따라 Security 컴포넌트는 복구를 진행한다. VDD 형태로 동작하고 있는 디바이스 드라이버 컴포넌트라면, 내려오는 request를 잠시 별도의 큐에 저장한 후, 복구 과정을 거친 뒤 들어온 request를 처리하는 것이 가능하다[7][8][9]. 이를 이용하여 네트워크 스토리지 시스템을 재기동하지 않고, 안전성을 확보할 수 있다는 장점을 가지게 된다.

3.2 구현

제안하는 플랫폼을 구성하기 위하여, 본 연구에서는 TUD의 L4 fiasco 마이크로커널[1]과 Environment servers를 이용하여 L4 환경을 설정하고, iSCSI 프로토콜을 이용하여 네트워크 스토리지를 구축하였다. 또한 TPM 1.2 버전의 칩과 Trusted GRUB[10]이라는 부트로더를 이용하여 authenticated boot를 지원할 수 있게 하였다. 네트워크와 SCSI 드라이버는 시스템의 안정성을 높이기 위하여 VDD 기법을 사용하였다. 네트워크와 SCSI VDD는 L4리눅스 2.6.17 버전을 이용하였다.

3.2.1 구조



< 그림 1 > Secure Network Storage System의 구조

본 플랫폼은 < 그림 1 >에서 보듯이, 크게 L4 마이크로커널, 시스템의 기능을 지원하는 Environment servers, 네트워크와 스토리지를 사용할 수 있도록 지원하는 두 개의 VDD, 그리고 TPM에 접근하여 TPM에 필요한 명령을 내리고, authenticated boot와 복구를 지원하는 security 컴포넌트로 구성되어있다.

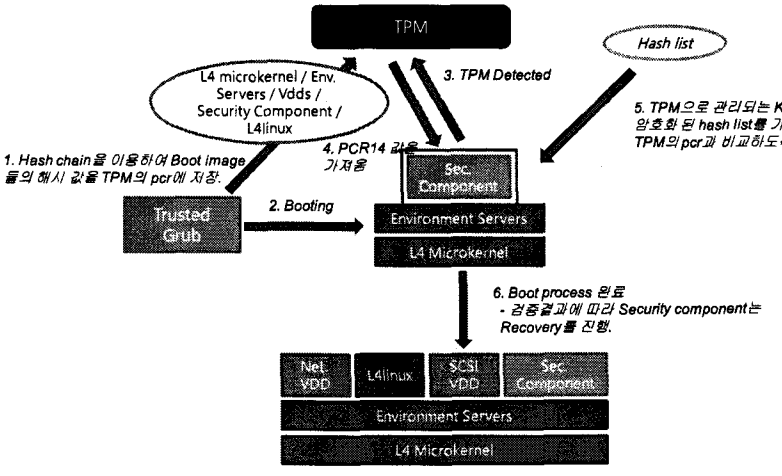
3.2.2 부트로더

본 연구에서 구축한 네트워크 스토리지 시스템은 부트에 필요한 컴포넌트를 불러오기 전에 부트로더에서 menu.lst에 등록된 컴포넌트의 해시 값을 생성하여 TPM의 PCR14에 저장한다. 이 때 각 컴포넌트의 해시 값을 개별로 저장하는 것이 아니라 해시 체인 (Hash Chain)을 이용하여 하나의 해시 값을 생성하여 저장한다.

$$PCR14\ (t+1) = H\ (PCR14\ (t)\ //\ x)$$

즉 위의 식[3]처럼 PCR14에 t+1번째 값을 쓰고자 한다면, PCR14에 t번째 값과 새로운 값인 x를 concatenation한 후, 해시 함수에 해당 값을 넣어 원하는 해시 값을 생성한다.

본 연구에서는 이 과정 이후에 부트로더에서 Security 컴포넌트의 해시 값을 생성하여 해시 리스트의 값과 비교하여 변경되었을 경우 부트과정을 종료하도록 하였다. Security 컴포넌트만 별도로 부트로더에서 검증 과정을 거치는 이유는, 해당 컴포넌트는 TPM에 명령을 내리고, 생성된 해시 값을 검증하고 복구를 담당하는 컴포넌트로 악의적인 변경에 대하여 반드시 보호되어야 하기 때문이다.



< 그림 2 > 시스템의 부트 과정

3.2.3 Security 컴포넌트

Security 컴포넌트는 3.2.2에서 소개한 것처럼 실제로 TPM에 접근하여 명령을 내리는 권한을 가지고, 해시 리스트를 이용하여 각 컴포넌트를 검증하여 복구를 수행하는 컴포넌트이다. Security 컴포넌트가 로드되면 시스템에 설치되어 있는 TPM 칩을 검색한다.

TPM 칩을 찾으면, 암호화 되어 있는 해시 리스트를 읽어온다. 해시 리스트는 AES 알고리즘을 이용하여 암호화 되며, 이 때 사용된 암호화 키는 TPM에서 생성된 키를 이용하여 다시 암호화 되기 때문에 외부로 유출되더라도 열어볼 수 없다.

그 후 TPM으로부터 PCR14에 저장되어있는 해시 값을 가지고 와서 해시 리스트의 값과 비교한다. 검증 결과는 로그에 남기며, fail일 경우 추가적으로 어떠한 컴포넌트에서 문제가 발생하였는지를 검증하게 된다.

각 컴포넌트 이미지는 해시 리스트의 값과 비교하게 되는데, 컴포넌트 이미지 개별로 해시 값을 생성하지 않았기 때문에 SHA1 해시 함수를 사용하여 해시 값을 생성해 주어야 한다.

< 그림 2 >는 네트워크 스토리지 시스템의 부트 과정을 설명한 것이다.

3.2.4 복구

3.2.3에서 설명한 것처럼 PCR14에 저장되어있는 해시 값에 문제가 있을 경우, 변경된 컴포넌트를 파악하여 복구하는 과정이 필요하다. 이를 위하여 security 컴포넌트는 각 컴포넌트 이미지 별로 SHA1 해시 함수를 사용하여 해시 값을 생성하고 해시 리스트의 값과 비교를 하게 된다.

이 과정에서 발견된 컴포넌트는 별도의 암호화된 파티션 혹은 검증된 third party로부터 이미지를 가져와서 변경된 이미지를 대체하도록 한다. Security 컴포넌트는 변경된 컴포넌트에 들어가는 request를 일시적으로 막거나 별도의 큐에 저장해놓고 해당 컴포넌트를 재로드한다.

본 연구에서는 VDD 기법을 사용한 네트워크와 SCSI VDD 컴포넌트에 대한 복구만이 구현되어있으며, Environment server와 같은 컴포넌트의 복구는 시스템의 재구동 하도록 되어있다. 이 부분은 시스템의 재구동 없이 복구가 가능하도록 앞으로 추가되어야 할 부분이다.

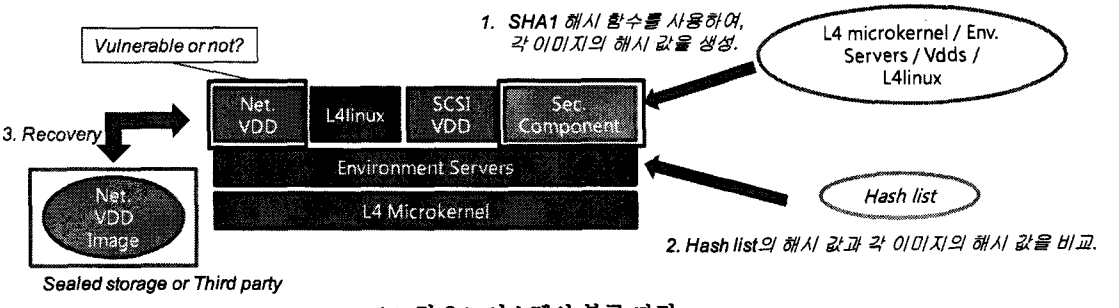
< 그림 3 >은 네트워크 스토리지 시스템의 복구 과정을 설명한 것이다.

3.3 실험

3.3.1 실험환경

본 연구에서는 부트 과정에 추가된 검증 과정이 끼치는 오버헤드와 TPM에 의해 추가된 암호화 과정에 의해서 추가적으로 소요되는 시간을 실험하였다.

이를 위하여 펜티엄 CPU T2300 1.66GHz와 512MB RAM으로 구성된 시스템에 리눅스 커널 2.6.17.14가 적용된 Ubuntu 배포판과 L4 fiasco와 Environment servers (sigma0, roottask, events, names, log, dm_phys, simple_ts, rtc, l4io, l4exec, bmodfs, con, loader), 네트워크와 SCSI VDD, Security 컴포넌트를 설치하여 실험하였다. 부트로더는 GRUB 0.97과 Trusted



< 그림 3 > 시스템의 복구 과정

GRUB 1.1.0을 설치하여 실험하였다.

3.3.2 부트로드 이미지 로드 시간

부트로드 이미지를 로드하는데 걸리는 시간의 측정은 부트로더인 GRUB이 스토리지에 저장되어 있는 운영체제의 이미지를 로드해서 실제 부팅이 진행될 때까지 소요되는 시간을 측정한 것이다. 이를 이용하여 부트로더에 이미지를 검증하는 프로세에 의해 어느정도의 오버헤드가 발생하는지 알 수 있다.

< 표 1 > 부트로드 이미지 로드 시간

Case	로드 시간 (sec)
GRUB 0.97 + Linux 2.6.17.14	0.64
TGRUB 1.1.0 + Linux 2.6.17.14	1.28
TGRUB 1.1.0 + L4 (이미지 27개)	4.92
TGRUB 1.1.0 + L4 (이미지 24개)	3.53

< 표 1 >은 5번의 부팅 시간의 평균값을 구한 것이다. GRUB 0.97은 L4를 부팅시킬 수 없기 때문에 GRUB + L4의 실험결과는 제외하였다. 경우 1은 검증과정 없이 커널 이미지 하나만을 로드할 때 소요되는 시간이다. 경우 2와 경우 3은 해시 값을 생성하여 TPM의 PCR14에 그 값을 기록하는 과정을 가지고 있다. L4가 Linux에 비해서 소요 시간이 많이 발생하는 이유는 L4는 기능별로 컴포넌트가 분할되어있어서 각 컴포넌트의 이미지를 모두 측정하여야 했기 때문이다.

실제로 측정해야 하는 이미지의 개수가 줄어든 경우 4는 경우 3에 비하여 로드 시간이 단축된 것을 확인할 수 있다.

3.3.3 TPM 암호화 소요 시간

본 연구에서는 암호화 키와 AES 알고리즘을 이용하여 데이터를 암호화하고 해당 암호화 키를 TPM에서 관리하는 키로 다시 암호화 하는 방식을 사용한다. 암호화 키의 암호화에 따른 오버헤드를 측정하기 위하여 TPM에서 키를 생성하여 128비트의 키를 암호화 할 때 소요되는 시간을 측정하였으며, 이에 소요된 시간은 0.426sec였다.

3.3.4 실험 정리

3.3.2와 3.3.3에서 나온 실험 결과를 볼 때, TPM에 의한 추가적인 암호화 과정은 크게 영향을 미치지 않을 것으로 판단된다. 다만 부트 이미지를 로드 하기 전에 발생하는 오버헤드는 해시 값을 생성하고 TPM의 PCR에 값을 저장하는 단순한 과정을 거침에도 불구하고 상당한 오버헤드를 보여주고 있다. 이를 해결하기 위해서는 L4에서 부트 시 로드 되는 이미지의 개수를 줄임으로써 가능하다.

3.4 향후 연구과제

현재 구현된 플랫폼은 네트워크 스토리지의 안전성을 확보하기 위하여, 부트 과정에 사용되는 컴포넌트의 이미지를 검증하여 악의적으로 변경된 컴포넌트의 동작을 멈추고 복구 하도록 되어있다. 이 과정에서 전제가 되어야 하는 것은 검증을 하는데 사용되는 security 컴포넌트와 해시 리스트가 악의적으로 변경되지 않아야 한다는 점이다. 이를 위하여 부트로더에서 security 컴포넌트를 부팅을 시작하기 전에 검증을 하여 문제가 발생할 경우 시스템을 종료하도록 구현하였지만, 이 부분은 security 컴포넌트 또한 복구 과정을 거치고 부트 과정이 진행되도록 수정되어야 한다.

또한 복구를 위해 사용되는 이미지를 어떻게 안전하게 보관할 것인지에 대한 고려와 Environment server의 복구 시에 발생하는 시스템 간섭 문제를 해결할 필요가 있다.

현재 네트워크 스토리지 시스템의 검증이 완료된 후, 패킷의 전송과정과 스토리지에 데이터를 저장할 때의 암호화 과정에 대한 연구를 진행 중이며, 이후 시스템 오버헤드를 파악하고 최적화를 하고자 한다.

4. 결론

모놀리식커널 상에서 동작하는 네트워크 스토리지는 모놀리식커널의 방대함과 복잡함에 따라 발생하는 보안의 취약성을 가지고 있다.

이를 해결하기 위하여 본 연구에서는 마이크로커널을 사용하였으며, 또한 authenticated boot를 이용하여 컴

포넌트를 부트 과정에서 검증하고 악의적으로 변경된 컴포넌트를 파악하여 복구를 함으로써 네트워크 스토리지의 안전성을 확보하였다.

또한 TPM을 사용하여 소프트웨어적인 보안방식을 보완함으로써 키의 유출로 인한 피해를 방지하고 있다.

이 플랫폼을 이용함으로써 기존 네트워크 스토리지 시스템의 보안을 강화하고, 시스템의 종료 없이 악의적으로 변경된 컴포넌트를 복구하는 것이 가능하다.

[Acknowledgement]

본 연구는 교육인적자원부의 BK21 사업과 정보통신부 및 정보통신연구진흥원의 대학 IT 연구센터 지원사업 (IITA-2007-C1090-0701-0045), 한국과학재단 (F01-2005-000-10241-0)의 연구결과로 수행되었음.

참고문헌

[1] L4 fiasco microkernel
http://os.inf.tu-dresden.de/fiasco

[2] Trusted Platform Module Work Group
http://www.trustedcomputinggroup.org/groups/tpm

[3] Bernhard Kauer, " Authenticated booting for L4" , 16th USENIX Security Symposium. 2004.

[4] CERT/CC Statistics
http://www.cert.org/stats/cert_stats.html

[5] Trusted Computing Group
http://www.trustedcomputinggroup.org/home

[6] 박우람, 박찬익, " TPM과 네트워크 스토리지 보안" - IDC July 2005 자료, 주간기술동향 통권 제 1279호, p17-p27. 2007.

[7] 강후영, " Virtual Device Driver : A Framework of Improving Device Driver Reliability in L4 Microkernel Environment" , 포항공대 석사학위 논문, 2006.

[8] H. Hartig, J. Loser, F. Mehnert, Reuther, M. Pohlack, and A. Warg. " An I/O Architecture for Microkernel-Based Operating Systems" , Technical Report. TU Dresden, July 2003.

[9] Jorrit N. Herder, Herbert Bos, Andrew S. Tanenbaum. " A Lightweight Method for Building Reliable Operating Systems Despite Unreliable Device Drivers" , Technical Report IR-CS-018, Jan 2006.

[10] Trusted GRUB
http://www.prosec.rub.de/trusted_grub.html

[11] Reiner Sailer, Trent Jaeger, Xianlan Zhang, Leendart van Doorn, " Attestation-based Policy Enforcement for Remote Access" , In Proceeding of the ACM Computer and Communication Security, 2004.

[12] 엄주관, 강후영, 류준길, 박찬익, " iSCSI 기반 스토리지 시스템을 위한 안전한 소프트웨어 플랫폼에 관한 연구" , 대한임베디드공학회, May 2006.