TPM에 기반한 iSCSI 네트워크 스토리지의 안전한 접근 방안

최종욱○ 박우람 박찬익

포항공과대학교

whiteugi@postech.ac.kr, wizrampa@postech.ac.kr, cipark@postech.ac.kr

A Framework of Secure Access to iSCSI Network Storage based on TPM

Jongwook Choi○ Wooram Park Chanik Park

Pohang University of Science and Technology

요 약

iSCSI[1]기반의 네트워크 스토리지는 IPSec을 사용함으로써, 전송하는 데이터의 기밀성과 무결성을 보장하고, 인증을 지원하고 있다. IPSec[2][3]과 함께, iSCSI에서는 CHAP(Challenge-Handshake Authentication Protocol)[4]라는 사용자 인증 프로토콜을 제공함으로써 iSCSI Target 서버에 접근하는 사용자를 인증할 수 있다. 하지만, CHAP프로토콜은 서버에 접근하는 사용자만을 인증할 뿐, 사용자 호스트 플랫폼 자체에 대한 인증은 수행하지 않는다. 이러한 상황에서 사용자 호스트에 Root-kit이나, 악의적인 소프트웨어가 설치되어 있다면 사용자가 접근하는 Target Disk의 데이터나, 사용자가 입력하는 패스워드와 같은 정보가 유출 될 수 있다. 본 논문에서는, TCG(Trusted Computing Group)에서 제안하는 Trusted Platform Module(TPM) 하드웨어 칩을 사용하여서 iSCSI로 구성된 네트워크 스토리지 환경에서 서버가 클라이언트 호스트의 플랫폼을 인증하는 방법에 대해서 디자인 하고 구현한다.

1. 서 론

많은 기업과 기관들은 사용자가 네트워크를 통해서 사용할 서버의 저장 장치(Storage Device)를 사용할 수 있도록, SAN(Storage Area Network)과 NAS(Network Attached Storage) 시스템을 구축하고 있다. 이러한 환경에서, 컴퓨팅 시스템과 애플리케이션이 네트워크를 통해 서버의 저장 장치에 있는 데이터에 접근할 수 있는 네트워크 스토리지를 사용한다. 특히, iSCSI는 IETF에서 개발한 IP에 기초한 네트워크 스토리지 프로토콜로써, 이더넷이 구축되어 있으면 어디에서나 네트워크 스토리지 환경을 만들 수 있다. 그렇기 때문에, 고비용의 복잡한 파이버 채널(Fibre Channel) 없이도, 네트워크 스토리지의 유연한 데이터 관리 기능을 충분히 활용할 수 있다는 장점이 있다. 하지만, iSCSI는 IP를 기반으로 동작하기 때문에, 네트워크 환경에서 존재하는 제3자 공격(Man-In-the-Middle Attack), 위장 공격(Masquerade Attack), 변조 공격(Spoofing Attack) 등에 취약하다. 그래서, 네트워크 보안은 네트워크 스토리지 구축에 중요한 요소이다.

현재 iSCSI에서는 네트워크 스토리지 보안을 위해서 CHAP(Challenge-Handshake Authentication Protocol), 사용자 인증 프로토콜과 IPSec(IP Security) 프로토콜을 사용하고 있다. IPSec 프로토콜은 IP 계층의 프로토콜로써, 전송되는 패킷을 암호화 하여서 정보의 기밀성(Confidentiality)을 유지하며, MD5/SHA1과 같은 해시 알고리즘을 사용해서 전송하고자 하는 패킷의 해시 코드를 생성하고, 이 정보를 바탕으로 송신 데이터에 대한 무결성(Intgrity)과 인증(Authentication)을 제공한다.

또한, CHAP 프로토콜을 통해서 iSCSI 서버(Target)에 접근하려고 하는 사용자를 인증할 수 있다. CHAP 프로토콜은 iSCSI 클라이언트(Initiator)에게 인증 정보를 요청하고 이를 확인하는 방식으로, 사용자를 인증하는데 사용한다. 그러나, 이 프로토콜에서는 클라이언트 호스트 플랫폼에 대한 인증 절차는 수행되지 않으며, 사용자가 알고 있는 정보만 확인 하는 과정이다. 이러한 상황에서, 클라이언트 호스트에 Root-kit이나 악의적인 소프트웨어가 존재한다면, 클라이언트가 접근하는 서버의 저장 장치 내용이 유출되거나, 클라이언트에서 사용하는 중요한 데이터 등을 유출할 가능성이 존재한다[5]. 이러한 취약점을 해결하기 위해서, 본 논문에서는 TPM(Trusted Platform Module)[6] 기반의 원격 검증(Remote Attestation)[7]을 활용해서 클라이언트의 플랫폼을 검증하고, 이를 통해서 클라이언트가 신뢰할 만한 플랫폼인지를 확인한다.

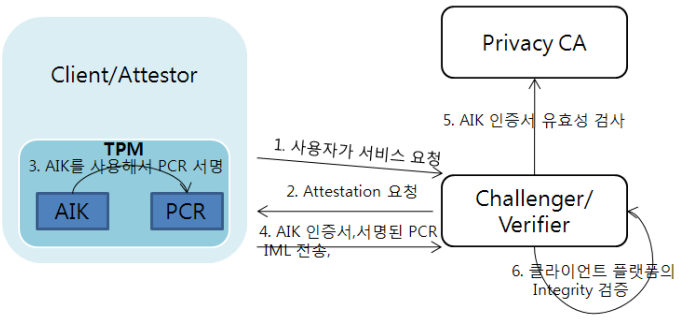

그림 1. 원격 검증(Remote Attestation)

본 논문에서 다루고자 하는 것은, 네트워크 상에서 발생하는 공격을 막는 것이 아니라, iSCSI로 구성된 네트워크 스토리지 환경에서 서버가 클라이언트 호스트 플랫폼을 인증하는 것이다.

본 논문의 구성은 다음과 같다. 2장은 TPM과 함께 사용되는 기술들에 대해서 설명, 3장은 TPM을 적용한 iSCSI 서버의 디자인 및 구현, 4장은 구현에 대한 평가, 5장에서는 결론을 맺도록 한다.

2. TPM(Trusted Platform Module)

사용자 호스트의 보안을 위해서 소프트웨어적인 보안 방식만을 사용하는 것에는 한계점이 존재하기 마련이다. 안티-바이러스 프로그램이나 보안 프로그램을 사용하는 것의 전제 조건은 운영체제(OS)가 안전하다는 가정을 두고 있기 때문에 보안 문제가 발생할 가능성이 있다. 또한, 운영체제나 소프트웨어의 취약점을 이용한 Zero-Day Attack이 발생하는 상황에서는 더욱 더 심각한 문제를 발생시킬 수 있다[4]. 이러한 한계점을 보완하기 위해서 하드웨어를 사용한 보안 방식이 제안되었고, 그 대표적인 예가 TCG(Trusted Computing Group)[8]에서 제안한 TPM(Trusted Platform Module) 하드웨어 칩이다. TPM 칩의 가장 큰 특징은 키(Key)나 패스워드와 같은 중요한 데이터 등을 저장할 수 있는 비휘발성 및 휘발성 저장공간을 제공하고 이러한 저장공간에 접근할 수 있는 명령어들을 제공하는 것이다. 이러한 TPM 칩을 사용하여서 상대방 플랫폼의 신뢰성 여부를 파악할 수 있는 원격 검증을 수행 할 수 있다.

아래에서, 본 논문에서 제안하는 기법을 구현하기 위해 필요한 기술들에 대해서 소개하도록 하겠다.

2.1 IMA(Integrity Measurement Architecture)

IMA[9]는 SHA-1 알고리즘을 사용하여서, 시스템 상에서 실행되는 모든 프로세스와 라이브러리의 코드에 대한 해시 값을 계산하고, 그 결과를 TPM 내부에 존재하는 특정 PCR(Platform Configuration Register)에 업데이트 하는 코드이다. 그리고 수행된 프로세스와 라이브러리 코드의 해시 값은 실행 순서대로, 로그 파일형태로 저장이 된다. 로그 파일에 저장되는 각각의 항목을 IML(Integrity Measurement Log)라고 하며, 프로세스(혹은 라이브러리) 경로, 해시 값, 해시 값이 업데이트 된 PCR 번호로 구성되어 있다. 추후에 각 IML의 해시 값을 계산(Extend) 하여서 특정 PCR 값과 동일한지를 비교하여 무결성 (Integrity) 검사를 수행할 수 있다.

2.2 원격 검증(Remote Attestation)

원격 검증이란 상대방의 플랫폼이 신뢰할 수 있는 상태인지 검증하는 것을 말한다. 그림 1은 원격 검증 절차를 보여 주고 있다. 클라이언트가 서버(Challenger)에게 어떠한 서비스를 요청하면, 서버는 서비스를 제공 하기 전에 클라이언트 플랫폼의 신뢰성을 검증하기 위한 원격 검증을 수행한다. 서버는 새롭게 생성한 난수와 함께 원격 검증 요청 메시지를 전송한다. 클라이언트는 수신한 난수와 현재 PCR 값, IML을 자신의 AIK로 서명하고, 이것을 서버에 전송한다. 서버는 TTP(Trusted Third Party)로부터 AIK 인증서를 확인하고, AIK로 서명한 값들의 무결성을 검증한다. 그리고 IML의 해시 값들을 차례로 Extend 하여서 수신한 PCR값과 일치하는지 확인함으로써 클라이언트가 신뢰할 수 있는 플랫폼인지 증명한다.

3. 디자인 및 구현

네트워크 스토리지 환경에서 클라이언트에 대해서 원격 검증을 수행함으로써, 클라이언트 플랫폼 자체를 신뢰할 수 있다. 이를 통해서, 클라이언트는 자신이 서버로부터 읽고 서버에 쓰는 데이터들과 그 이외의 중요한 데이터들이 자신이 의도하지 않는 소프트웨어(공격자가 설치한 소프트웨어)에 의해서 유출되는 것을 막을 수 있다. 아래에서는, 네트워크 스토리지 환경에서, 클라이언트를 검증하고 신뢰할 수 있는 플랫폼을 디자인 하고, 구현 하는 것에 대해서 소개하도록 하겠다.

iSCSI 기반의 네트워크 스토리지 환경에서, 저장장치를 외부에 제공하는 서버를 iSCSI 타겟(Target)이라 하고, 서버에 있는 저장장치를 사용하는 클라이언트를 iSCSI initiator라고 한다. 여기서 각각을 서버와 클라이언트라고 칭하도록 한다. 구현은 open-iscsi-2.0-870.3과 iscsi-enterprise-target(IET) 0.4.17 을 사용하였다.

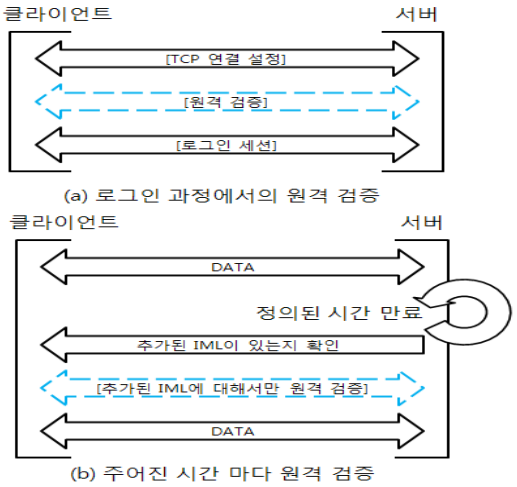

그림 2. 원격 검증 시점 (a)는, 로그인 과정에서 원격 검증을 수행한다. (b)는서버와 클라이언트가 세션을 생성 후 데이터를 입출력하는 도중, 정의된 시간이 만료가 되어서 발생하는 원격검증이다.

3.1 클라이언트 플랫폼의 원격 검증 시점

iSCSI 서버에서 클라이언트의 플랫폼을 검증하는 시점은 크게 두 부분으로 나눌 수 있다. 첫 번째는, 클라이언트가 서버에 로그인을 수행하기 전에 플랫폼을 검증하는 것이다. 두 번째는, 로그인을 마치고 iSCSI 세션을 생성 후, 서버와 클라이언트가 통신을 하는 과정에서 정해진 시간 주기 마다 서버가 클라이언트의 플랫폼을 재검증 하는 과정을 수행한다. 첫 번째 시점에서는, 부팅 과정부터 수행된 모든 프로세스와 라이브러리의 IML 전체를 전송하고 무결성을 계산 하지만, 두 번째 시점부터는 새롭게 수행된 프로세스의 IML만을 전송하고 무결성을 재계산 하게 된다. 그림 2의 (b)는 정의된 시간이 만료 될 때마다, 이전에 원격 검증 후 발생한 IML에 대해서만 추가로 원격 검증을 수행하는 것을 보여준다.

3.2 시스템 아키텍처 및 디자인

그림 3의 (a)와 (b)는 원격 검증을 위한 iSCSI 클라이언트(Initiator)와 서버(Target)의 구조를 보여준다. 클라이언트의 Attestor Module은 원격 검증 과정에서 TPM을 사용하여 플랫폼 인증을 위한 정보를 서버에 전달하는 역할을 수행한다. 서버의 검증 매니저(Attestation Manager)는 현재 서버에 연결된 iSCSI 세션 정보를 유지하며, 그 정보를 바탕으로 각 클라이언트 별로 원격 검증을 수행하는 Verifier 프로세스를 생성한다. 또한 검증 매니저는 클라이언트로부터 연결 요청 및 로그인 요청, 세션 등을 관리하는 IETD(iSCSI Enterprise Daemon)와 커뮤니케이션 한다. 각 Verifier는 클라이언트에 대해서 원격 검증을 수행 한다.

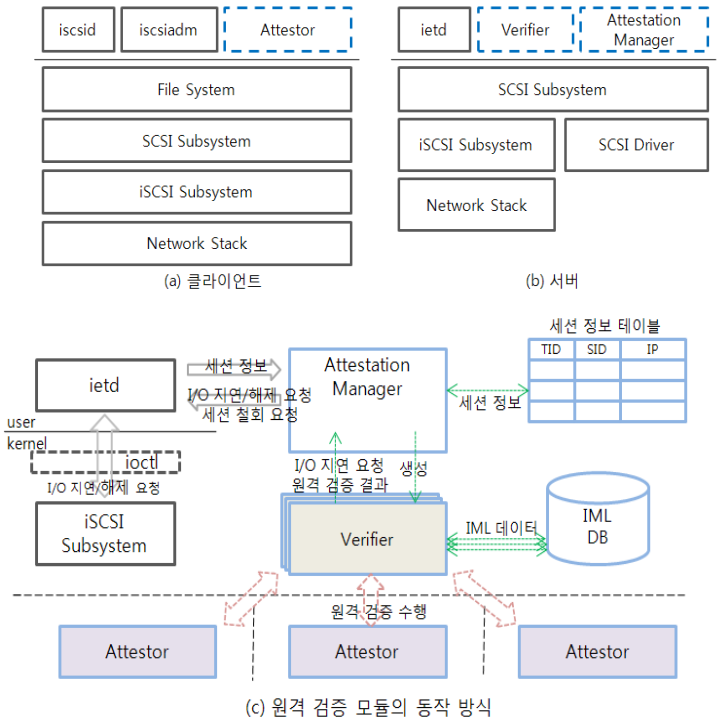

그림 3. 클라이언트와 서버의 구조

그림 3의 (c)는 각 모듈이 동작하는 과정을 설명하고 있다. 새로운 세션이 생성 혹은 해제 되거나, 클라이언트로부터 연결 요청이 발생하면, 그것을 관리하는 IETD는 해당 정보를 검증 매니저에게 전달한다. 이때, 검증 매니저는 자신이 유지하고 있는 세션 정보 테이블에 Target ID(TID), Session ID(SID), 클라이언트 IP 주소를 저장하게 된다. 그리고 새로운 클라이언트에 대한 Verifier 프로세스를 생성하고, 클라이언트에 대한 원격 검증을 수행하도록 한다. Verifier는 클라이언트의 Attestor 모듈과 커뮤니케이션을 하면서 원격 검증을 수행하게 되고, 서버가 유지하고 있는 IML DB를 원격 검증에 사용한다. IML DB에는 유효한 프로세스와 라이브러리 코드에 대한 해시 값들이 저장되어 있으며, Attestor로부터 수신한 IML들의 해시 값과 비교하기 위해서 사용된다.

Verifier는 정해진 시간마다 Attestor와 원격 검증을 수행하는데, 이때, 검증 매니저를 통해서 해당 세션의 I/O 처리를 지연 하도록 요청 한다. 그 이유는, 원격 검증 과정이 수행되는 동시에 서버와 클라이언트 사이에서 데이터 입출력이 발생하게 되면, 원격 검증이 끝나기 전에 서버의 데이터가 읽히거나, 서버에 새로운 데이터가 쓰여질 가능성이 존재하기 때문이다. 그래서, 원격 검증시 마다 Verifier는 검증 매니저에게 I/O 지연 요청을 하고, 검증 매니저는 IETD를 통해서 해당 세션의 I/O 지연 요청을 수행한다. Verifier가 검증을 모두 마치게 되면 검증 결과를 검증 매니저에게 전달하고, 그 결과에 따라서 검증 매니저는 I/O 지연을 해제 할 것인지, 아니면 해당 세션을 철회할 것인지를

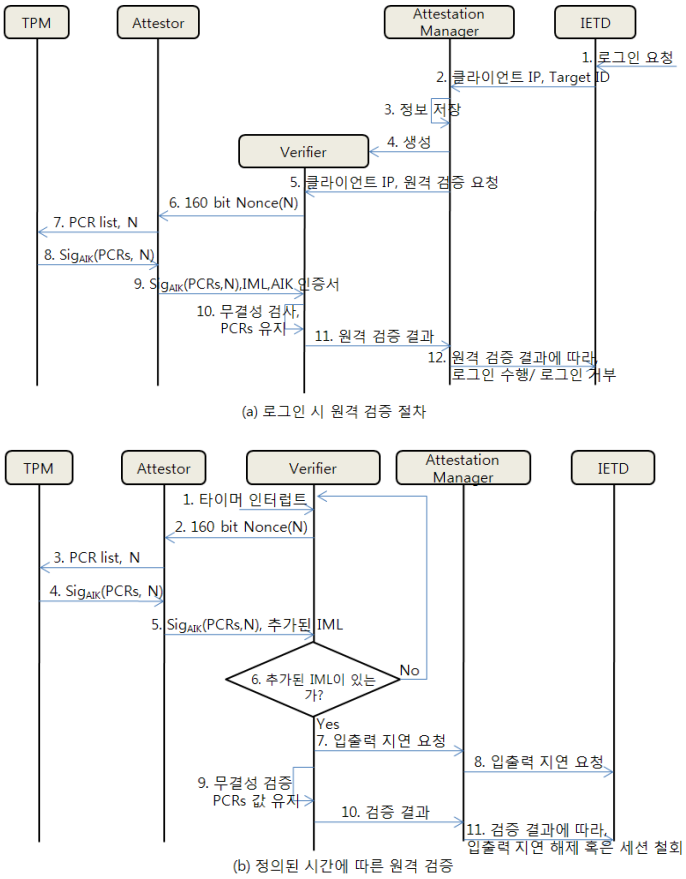

그림 4. 원격 검증 수행 과정

결정한다.

3.3 원격 검증에 대한 구현

원격 검증 구현을 위해서, IETD와 iSCSI Subsystem을 수정 하였다. 로그인 시 원격 검증을 위해 IETD 모듈에서 로그인을 지연하기 위한 코드를 추가하고 수정 하였다. 또한, 특정 시간 주기마다 원격 검증을 수행하는 과정에서 iSCSI 입출력 지연을 발생시키기 위해서 IETD와 iSCSI Subsystem을 수정하였다.

그림 4의 (a)는 로그인 시 원격 검증 절차에 대한 순차 다이어그램이다. 서버가 로그인 요청을 수신하면, 로그인을 지연하고, Verifier와 Attestor가 원격 검증 절차를 수행한다. 검증 과정의 10번에서 클라이언트로부터 수신한 PCR값과, Nonce, IML, AIK 인증서를 사용해서 클라이언트 플랫폼에 대한 무결성 검증을 수행한다. 그리고 추가적으로 수신한 PCR 값을 따로 저장해 둔다. 그 이유는, Attestor가 차후에 수행될 원격 검증에서는 전체 IML을 수신하는 것이 아니라, 이전에 검증 과정에서 전송한 IML을 제외하고 새롭게 생성된 IML값만을 전송하기 때문이다. 이전에 저장 해 두었던 PCR 값에 SHA-1 함수를 적용함으로써, 검증을 위해 새롭게 수신한 PCR값을 생성할 수 있다. 이것을 식으로 표현하면 다음와 같다.

oldPCR = 이전 검증에서 수신한 PCR 값
newPCR = 검증을 위해 수신한 PCR 값
h_i = 수신한 i번째 IML의 해시 값
n = 수신한 IML의 개수

수식 1 $newPCR = SHA\text{-}1(SHA\text{-}1(\ldots SHA\text{-}1(SHA\text{-}1(oldPCR \mid h_1) \mid h_2)\ldots) \mid h_{n-1}) \mid h_n)$

수식 1에서, newPCR을 생성하기 위해서 oldPCR에 차례로 IML의 해시 값을 OR 연산 후, SHA-1 해시 값을 계산 하는 것을 볼 수 있다. 이러한 방식으로, 새롭게 생성된 IML에 대해서 무결성 검사를 수행할 수 있으며, 전체 IML을 재전송하는 오버헤드를 줄일 수 있다.

그림 4의 (b)는 정해진 시간 주기마다 새롭게 생성된 IML에 의해서만 원격 검증하는 것을 보여 주고 있다. 6번 과정에서는 클라이언트가 추가로 생성된 IML이 있는지를 검사하고 있다. 이는 간단히 이전에 저장해 두었던 PCR 값과 새롭게 수신한 PCR 값을 비교함으로써 가능하다. 새롭게 수신한 PCR값은 TPM 내부에서 서명된 값이므로, 이를 통해서 현재 클라이언트의 정확한 PCR 값을 알 수 있다. 즉, 전송되는 PCR값의 무결성이 보장된다. 사인된 PCR값을 수신한 Verifier는 이전에 수신하였던 AIK키를 사용해서 PCR 값을 복원하게 되며, 저장해 두었던 PCR 값이 새롭게 수신한 PCR 값과 일치한다면 클라이언트에서 추가로 생성된 IML이 없다는 것을 의미하며, 원격 검증이 수행되지 않는다. 하지만 두 PCR 값이 불일치 할 경우, 즉, 추가로 생성된 IML이 있다면 Verifier는 원격 검증 과정을 수식 1을 사용하여서 수행하게 된다. 그리고 원격 검증 과정에서 발생하는 클라이언트의 Data 입출력을 막기 위해서, Verifier는 입출력 지연 요청을 수행한다. 원격 검증이 완료되면, 그 결과에 따라서 입출력 지연이 해제 되거나, 혹은 세션을 철회하는 과정을 수행한다.

4. 평가

제안된 기법이 시스템에 어떠한 영향을 미치는지에 대한 실험을 위해서, Xeon CPU, 2GB 메모리, SCSI 디스크로 구성된 서버와 Dual core CPU, 1GB 메모리, SATA로 구성된 클라이언트를 100MB LAN으로 연결하였다.

실험은 크게 두 가지로 구성하였으며, 첫 번째는 로그인시 수행되는 원격검증에 의한 로그인 지연 시간을 측정하였다. 두 번째는, 주기적으로 발생하는 원격 검증에 의해서 클라이언트의 I/O 입출력 지연을 측정하였다.

표 1 원격검증에 의한 로그인 지연시간

Boot IML의 개수	141
Post-Boot IML의 개수	648
IML 전송 및 로그 재계산 시간(초)	1.17
IML DB 검색 시간(초)	5.65
총 로그인 지연 시간(초)	6.82

표 1은 원격검증에 의한 로그인 지연시간을 보여주고 있다. 클라이언트에서 IMA에 의해서 생성되는 IML의 종류는, 부팅 과정에서 BIOS와 부트로더에 의해서, 실행되는 코드나 라이브러리의 해시 값을 측정해서 생성된 Boot IML과 부팅 후에 커널에 의해서, 실행되는 코드나 라이브러리의 해시 값을 측정해서 생성된 Post-Boot IML 두 가지가 있다. 위의 표 1은 각각의 IML이 141개, 648개로 구성되어 있음을 보여 주고 있다. 이 IML을 서버로 전송하고, 그 값들을 재계산하는데 걸리는 시간이 1.17초이며, 모든 IML의 로그 값이 IML DB에 존재하고, 올바른 값인지 검사하는데 5.65초가 소요된 것을 보여 주고 있다. 이 둘에 의해 발생하는, 로그인 지연시간은 6.82초이며, 대부분의 지연은 IML DB를 검색하고 확인하는데 소요되는 것을 볼 수 있다. 위와 같은 로그인 지연시간은 클라이언트와 서버의 통신과정에서 단 한번만 수행되므로, 시스템의 전체 수행과정에 영향을 미치지 않는다.

제안된 기법에서는 특정 시간 주기마다, 새롭게 생성된 IML에 대해서만 원격 검증을 수행하는 방식을 사용하고 있다. 이러한 검증 과정에서 입출력을 지연시킴으로써 클라이언트가 iSCSI 입출력을 수행하지 못하도록 하였다. 그림 5는 IML 개수에 따른 원격 검증으로 인한 입출력 지연 시간을 나타낸다. 구현에서 재검증 하는 시간의 주기는 10초로 설정하였다. 10초마다, 서버는 클라이언트에게 재검증을 요청하면, 클라이언트는 업데이트 된 PCR값과 새로 생성된 IML을 전송한다. 만약 새로 생성된 IML이 없을 경우에는 재검증 과정이 발생하지 않으며 입출력 지연 또한 발생하지 않을 것이다. 하지만, 클라이언트에서 새롭게 생성된 IML이 존재할 경우, 그에 대한 재검증 과정이 발생하게 된다. 표 2에서는 새롭게 수행된 프로세스의 IML 개수에 따른 재검증시 발생하는 입출력 지연시간을 보여주고 있다. 1개의 IML이 새롭게 생성되었을 때는 약 48ms의 입출력 지연시간이 발생하고, 20개의 IML이 새롭게 생성되었을 때는 약 175ms의 지연시간이 발생한다. 위 실험은 IML 개수가 정해진 상태에서 입출력 지연 시간을 보여준다. 하지만, 실제 워크로드에서의 입출력 지연시간을 측정하기 위해서는, 특정 시간 주기마다 사용자가 얼마나 많은 IML을 생성하는지 고려해 볼 필요할 것이다.

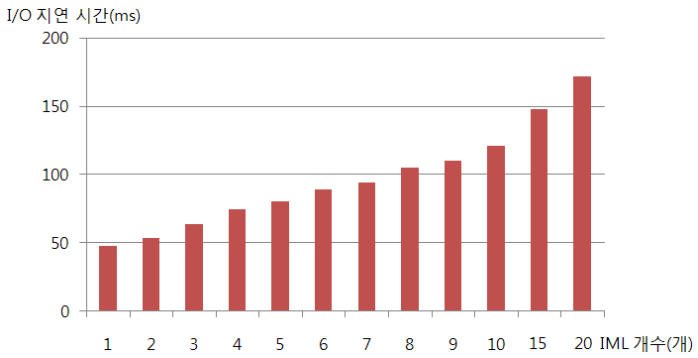

그림 5 IML 개수에 따른 I/O 지연시간

5. 결론

본 논문에서는 iSCSI 네트워크 스토리지 환경에서 클라이언트 플랫폼을 인증하고, 클라이언트의 연결을 제한하는 시스템을 디자인하고 구현하였다. 로그인 과정에서 한번의 원격 검증이 수행이 되고, 정해진 주기마다 반복해서 재검증 함으로써 클라이언트 플랫폼의 신뢰성을 높일 수 있다. 또한 재검증 과정에서는 새롭게 생성된 IML만 전송함으로써, 전체 IML을 전송하는데 발생하는 오버헤드를 줄일 수 있다. 그리고, 이러한 시스템을 적용할 때 발생하는 지연시간에 대해서 측정함으로써, 전체 시스템에 원격 검증이 미치는 오버헤드에 대해서 고려하였다.

참고 문헌

[1] Julian Satran, Kalman Meth, Costa Sapuntzakis, Mallikarjun Chadalapaka, Efri Zeidner, “iSCSI Draft”

[2] S. Kent, R. Atkinson, “IP Authentication Header”, RFC 2402, Nov. 1998.

[3] S. Kent, R. Atkinson, “IP Encapsulating Security Payload”, RFC 2406, Nov. 1998.

[4] RFC 1994 - PPP Challenge Handshake Authentication Protocol (CHAP)

[5] 박우람, 박찬익, “TPM과 네트워크 스토리지 보안”, 주간기술동향 제1279호, Jan. 2007.

[6] Trusted Platform Module Work Group Web Site: http://www.trustedcomputinggroup.org/developers/trusted_platform_module

[7] Dries Schellekens, Brecht Wyseur, Bart Preneel, "Remote Attestation on Legacy Operating Systems With Trusted Platform Modules", Science of Computer Programming Vol74, Dec. 2008.

[8] Trusted Computing Group Web Site : http://www.trustedcomputinggroup.org/

[9] Reiner Sailer, Xiaolan Zhang, Trent Jaeger, and Leendert van Doorn, "Design and Implementation of a TCG-based Integrity Measurement Architecture", USENIX Security Symposium, Vol13, 2004.