제18회 한국정보처리학회 추계학술발표대회 논문집 제9권 제2호 (2002.11)

SAN환경에서의 보안

성준호*, 전성택**
*영산대학교 정보통신대학원
e-mail:daoyou@hanmail.net

Security For Storage Area Networking System

Chun-Ho Sung*, Sung-Tag Jun**
*Graduate School of Computer Engineering & Information Technology, Young-san University

요 약

오늘날 점점 방대해지고 커지는 DATA의 처리를 위해 네트워크 스토리지 시스템의 요구가 증대됨에 따라 네트워크 스토리지의 보안문제가 대두되고 있다. 네트워크 스토리지중 그 성장이 두드러지는 SAN 스토리지의 보안을 위해 정보보호 서비스 구현의 상세한 지식을 필요로 하지 않는 장점을 가진 GSS-API인증 시스템을 이용하여 SAN 스토리지 보안문제에 접근해 보았다.

1. 서론

최근 기업들은 대용량 데이터를 관리하면서 데이터도난, 해커의 공격, 인간의 실수 등에 의하여 치명적인 손실을 입게 되면서 대용량 데이터베이스의 효율적 관리를 위해 SAN등의 네트워크 데이터베이스를 도입하고 있으며, 그중 SAN이 백업 복구기술을 포함한 스토리지 관리비용을 감소시키며, 신속한 재난복구가 가능하고, 장기적인 확장성 및 내장애성 등에 대한 장점 등으로 인하여 그 수요가 증대되고 있다. 하지만 SAN은 스토리지 기능에만 치우쳐 있으며, 데이터 보호에는 대책이 없는 상황이다. 이것은 향후 보안문제를 야기할 가능성이 있으며 SAN의 Security에 많은 투자가 이루어 져야 할 것이다. 본 논문에서는 보안에 관한 지식이 별로 없는 프로그래머라 할지라도 쉽게 적용할 수 있는 GSS-API를 적용하여 SAN의 보안문제를 해결하는데 조금이나마 접근해 보려고 한다.

2. SAN의 기술 및 보안

2. 1 SAN의 기술

호스트 컴퓨터에서 SCSI를 통한 스토리지 서버와 신속한 데이터전송이 한계에 이르고, 복잡한 네트워크환경과 유기적으로 연결되는 서버 / 클라이언트들로 인하여 데이터 전송량이 폭주하고 기존네트워크로는 더 이상 데이터의 이동경로로 사용하기 힘들 정도의 포화에 이름에 따라 SAN이 Fibre Channel을 매체로 해서 해결책으로 등장하게 되었다.

SAN은 스토리지 에 대한 고속 액세스 및 공유를 목적으로 하는 네트워크로서 대규모 네트워크 사용자들을 위하여 서로 다른 종류의 데이터 저장장치를 관련 데이터서버와 함께 연결하여 네트워크상에서 Fibre Channel의 이점인 고속전송과 장거리 연결 및 멀티프로토콜 기능을 활용하는 기술로서 대개 한 기업의 전체컴퓨팅 자원을 연결해놓은 네트웍의 일부로서 대부분 메인프레임과 같은 다른 컴퓨팅 자원에 아주 근접하여 밀집되어 있으며, 광역통신망 기술을 이용하여 원거리에 있는 장소로 확장가능하며, 백업이나 기록의 영구보관 및 저장이 가능한 기술이다.

2. 2 Fibre Channel(ANSI-X3T.11)

Fibre Channel 기본규약이 1994년 미국 표준협회(ANSI)에서의 인증을 계기로 수많은 업체가 SAN 관련 제품을 개발하였고, 현재 SAN 표준화 단체인 SNIA(Storage Networking Industry Association),

FCIA(Fibre Channel Industry Association)등에서 SAN 관련 제품들에 대한 호환성 검증 및 표준화를 수행하고 있으며, 최대 전송거리는 10㎞이며, 최대 전송속도는 200Mbyte/sec에 이르는 고속 I/O 채널로서 100Mbit Fast Ethernet에 비교하여 10~30배 정도의 성능이 향상된 새로운 기술로서 빠른 데이터 액세스와 좋은 확장성을 가지고 있으며, 높은 이용률과 신뢰성이 확보되어 있으며, 스토리지 자원의 공유를 개선하였고, 기존 망으로의 통합이 용이하며, TCO(Total Cost of Ownership)의 감소를 가진 Fibre Channel의 채용으로 SAN은 거리와 속도 면에서 많은 개선점을 가지게 되었다.

파이버채널 프로토콜(Fibre Channel Protocol)은 안전에 대한 위협에 노출되어 있다. 물리적으로 안전한 Fibre Channel fabric에서의 데이터 도난과 파괴를 야기할 수 있으므로 FC 프로토콜의 안전성을 위해 사용자의 동시 접근으로 인한 유효성 유지 등을 지원할 수 있는 보안기술이 요구된다.

2. 3 SAN의 보안관리

SAN은 데이터 센터 자체에 대한 보안이 중요하게 되었다 그 이유는 SAN은 주로 인터넷과 단절되거나 멀리 떨어져 있고 몇몇 방화벽들로 싸여 있기 때문에 대부분의 위협은 내부에서 발생된다. 인력 고용, 보안 정책의 수립에 있어서 신중을 기해야 하는데 SAN안에 배치되어지는 서버와 기기들에 대한 접근은 인가되고, 신뢰성 있는 인원을 배치하여 자체 보안을 철저히 해야하며, SAN으로부터 데이터를 가져가는 요구처리나, SAN에 데이터를 정장하는 등의 작업을 수행하는 서버에 대한 보안이 중요한 것은 만일 공격자가 이들 서버들 중 하나라도 무단 및 인증되지 않은 SAN액세스, 보안되지 않은 관리 액세스, wwn 스푸핑, 다른 액세스 지점에서 허용되는 제어관리의 공격으로 권한 획득 시에는 SAN에 있는 데이터를 제한 없이 취득할 수 있기 때문이다. 서버 관리자는 정례적인 감시와 더불어 보안 기능의 완전한 구현등 서버보안에 각별한 주의를 기울여야 한다. 그러므로 공격자의 불법침입을 막기 위해 SAN서버내부에서 GSS-API의 인증절차를 거쳐 스토리지에 접근하게 한다면 보다 안전하게 데이터를 보호할 수 있을 것이다.

3 GSS-API

3. 1 GSS-API의 특성

GSS-API는 개방형 분산 네트워크 환경에서 응용프로그램 수준의 보안 서비스를 제공하기 위한 네트워크 프로그래밍 인터페이스로 1993년에 IETF(internet Engineering Task Force)의 공통 인증기술(CAT, Common Authentication Technology)에 의해 RFC(Reqiest For Comments) 1507-1511[3],[4]로 표준화되었고 또한 DES 및 RSA와 같은 다양한 암호화 메커니즘을 기반으로 구현되었으며, 통신 프로토콜에 대하여 독립적으로 운영될 수 있도록 상의 수준의 추상화 개념을 제공 및 정보보호 서비스 구현의 상세한 지식을 필요로 하지 않는 장점도 가지고 있다.

3. 2 GSS-API요구사항

하부메커니즘 독립성을 가진 인터페이스를 지원함으로써 정보보호 서비스에 암호서비스 수행의 상세구현과는 독립적으로 제공되어지며 공개키 비밀키 구분과 API 구분 없이 어떤 시스템에 사용하여 구현이 가능하다.

API는 하부 프로토콜환경에 독립적으로 통신프로토콜체제를 지원함으로써 TCP/IP나 기타 통신설정용 응용프로그램 프로토콜과는 독립적으로 동작한다.

응용프로그램의 다양한 프로토콜 이용을 위하여 하나의 인터페이스를 지원한다.

클라이언트 TCB 내부와 외부의 호출자와는 무관하게 구현할 수 있는 환경을 보장한다.[1]

3. 3 GSS-API기능

분산 응용프로그램 환경에서 정보보호 서비스에 대한 범용 인터페이스를 제공한다. 보안서비스들에 대한 폭넓은 인터페이스와 보안서비스를 제공하는 메커니즘은 가진다. 인터페이스에서는 메커니즘과 데이터변환 호출접속 역할을 수행하고 신뢰되어진 제3자의 전형적인 원격지 서버의 보호된 호스트메커니즘을 갖는다. 보안서비스 구현을 위해 사용되는 메커니즘은 세션초기에 선택되고 세션동안 고정되어지며 메커니즘의 선택은 GSS-API에 사용할 수 있는 메커니즘과 응용프로그램에 의해 결정된다 이러한 구조를 GSS-API가 가질 수 있는 것을 분산응용프로그램 서비스로써 보호된 응용프로그램과 서버메커니즘으로 구성되어있기 때문이다.

3. 4 GSS-API 설계 모델

RFC1508에서 제시하는 20개의 기본적 GSS-API 호

신임장 관리 호출(Credential Management Calls)	
gss_acquire_cred()	보안 연결을 시작하거나 얻기 위한 신임장 획득
gss_release_cred()	연결이 끝난 후에 신임장 영역을 해제
gss_inquire_cred()	신임장에 대한 정보를 조회
문맥연결 관리 호출(Context Management Calls)	
gss_init_sec_context()	나가는 암호 연결에 대한 초기화
gss_acccept_sec_context()	들어오는 암호 연결을 승인
gss_delete_sec_context()	더 이상 필요 없을 때 연결 종료
gss_process_context_token()	연결된 토큰의 처리 제어
gss_context_time()	연결에 남은 유효 시간 표기
메시지 암호 호출(Per-Message Calls)	
gss_sign()	메시지에 서명을 붙임
gss_verify()	서명된 메시지를 검증
gss_seal()	메시지를 서명, 암호화 및 캡슐화
gss_unseal()	메시지를 개봉, 필요시 복호화 및 서명 검증
지원 관련 호출함수(Support Calls)	
gss_display_status()	상태 코드를 출력할 수 있는 형태로 변환
gss_compare_name()	2가지 이름이 같은가 비교
gss_indicate_meche()	사용 가능한 메커니즘을 표시
gss_display_name()	이름을 출력 가능한 형태로 변환
gss_import_name()	사람이 읽을 수 있는 이름을 내부적 기호로 변환
gss_release_buffer()	할당된 버퍼 기억장소를 해제
gss_release_name()	이름 기억 장소를 해제
gss_release_old_set()	OID 기억 장소를 해제

<표1>

출은 <표1>과 같이 요약할 수 있다.[2] 신임장 관리 호출(Credential Management Calls)은 통신 실체에게 신임장의 획득 및 해제를 지원하고, 또한 여러 가지의 신임장 정보에 관한 조회를 제공한다. 정보보호 문맥의 연결은 연결의 초기화, 허가 및 삭제, 그리고 연결의 유효시간과 연결 관련된 토큰들을 처리하는 문맥연결 관리 호출(Credential Management Calls)을 사용하여 관리한다.[3]

메시지 암호 호출(Per-Message Calls)이라고 알려진 암호학적 호출은 일단 안전한 세션이 연결되면 메시지 단위로 GSS-API 내에서 무결성 및 기밀성에 대한 접근을 제공한다. 마지막으로 지원 관련 호출(Support Calls)은 할당된 메모리를 해제하고 이름의 비교와 같은 일반적인 관리 및 지원 루틴들을 제공한다.[3]

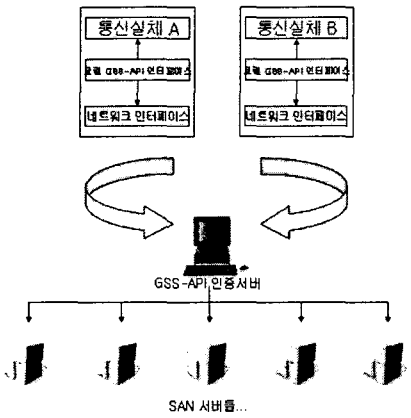
이상과 같은 API 호출을 구현하기 위해서는 다중 레벨의 내부 구조를 갖도록 설계할 수 있다. API의 상위 레벨은 호출 응용프로그램에서 볼 수 있으며 지원 메커니즘을 포함하고 있는 하위레벨과 내부적으로 인터페이스하고 있다. GSS-API는 메커니즘에서 제공될 수 있는 정보보호 서비스에 대한 일관성 있는 인터페이스를 제공하기 위하여 암호 메커니즘과 응용프로그램 사이에 존재하는 접착제 역할을 한다.[1]

구현을 위하여 외형적인 API의 구현에 대한 고려뿐만 아니라 다수의 메커니즘들에 이러한 구조의 통합을 고려해야 한다. 구현상의 어려움은 메커니즘의 데이터 구조에 따라 관련 API 내부 데이터 구조를 변환 및 관리하는 구체적인 절차의 구현에 의존한다. GSS-API를 위한 공개키 기반구조, 특권 허가 서비스 및 보안 정책 등이 향후 구현에 확장하여 고려되어야 할 것이다.[1]

4. DAFS(Direct Access File System)

DAFS는 DAFS Collaborators(www.dafscollaborative.org)라는 기관을 중심으로 DAFS

표준화에 85개 업체가 역할을 분담하여 '성능, 확장성, 안정성 개선' 을 목표로 표준화를 진행중이다.



<그림 1>

DAFS는 네트워크 어플라언스(Network Appliance)에 의해 제안되었으며 TCP 스택의 지연시간을 제거하기 위한 방법으로 서버와 하나의 클러스터 내에 있는 저장장치들 간에 잠재시간이 적고, 고대역폭으로 통신이 가능한 VI를 사용하도록 디자인되었다. 데이터는 등록이 이루어지면 애플리케이션 버퍼에서 NIC로 직접 이동한다. 또한 DAFS는 네트워크에 연결된 서버와의 통신을 위해 표준 TCP/IP를 사용한다. 이러한 IP를 이용한 접근 방식의 장점은 공공 데이터 네트워크를 사용함으로써 사설 전용 회선을 사용하는 데 소요되는 비용을 비약적으로 절감시키는데 있다. TCP/IP의 오버헤드나 불확실성과 같은 부분을 해결하기 위해 RDMA의 접목을 시도하고있는데 이는 스토리지에 있는 DAFS 드라이버가 자신이 전달시켜야 하는 데이터를 최소의 경로로 RDMA에 전달시키기만 하면 이 RDMA는 메인 서버의 메인 메모리까지 스토리지와 메인 서버 어느 쪽의 CPU도 활용하지 않고 단번에 이 데이터를 전달하게 되고 전달된 데이터를 서버 쪽의 DAFS가 또다시 최소 경로를 거쳐 유저 레벨 애플리케이션까지 전달하게 된다는 것이다.

DAFS기술은 iSCSI와 상반되게 오버헤드를 없애는 방향으로 진행되고 있는데, 기존 컴퓨터 내의 대용량 메모리 운영기술 서브시스템간의 커뮤니케이션과 대용량 스토리지들이 저장 데이터를 보다 효율적으로 운영하기 위해 기술을 발전시켜 왔고 또한 10GB의 고속 네트워크시대에 비추어 본다면 이러한 여러 기술들은 오버헤드가 될 수 있는데 DAFS는 이러한 오버헤드들을 없애기 위해 기술을 발전시키고 있다.

5. 결론

본 연구에서는 SAN환경 내에서 취약한 보안을 보완하기 위해 GSS-API를 SAN서버와 연동하여 GSS-API의 장점인 응용프로그래머의 보안지식과는 별개로 GSS-API의 보안메커니즘을 상세히 알지 못하더라도 구현이 가능한 점을 이용하여 SAN서버를 통하여 데이터베이스에 접근을 시도하려할 때 SAN 서버는 GSS-API 서버와 연동하여 접근하는 사용자가 정상적인 사용자인지를 구분해주므로 불법침입을 막을 수 있게 설계할 수 있을 것이다.
또한 GSS-API와 SAN서버 사이에 DAFS 파일시스템을 연결하여 정확한 데이터의 전달과 같은 기능을 제공 할 수 있을 것이다.

그림1은 SAN에서 GSS-API가 응용된 모습을 보여주는 것이다. 앞으로 데이터 센터 자체에 대한 보안과 외부 침입에 대한 모든 메커니즘을 수용할 수 있도록 연구되어야 할 것이다.

참고문헌

[1] 분산 어플리케이션의 정보보호 서비스를 위한 CAPI 설계 모델 (CAPI Design Model for Information Security Services of Distributed Applications) / 이상기 김영덕 최용락 (論文集, Vol.9 No.2, [1998])

[2] 개방형 분산환경의 정보보호를 위한 범용 암호 서비스 모델 (A Generic Cryptographic Model for Information Security in Open Distributed Environments) / 최용락 (論文集, Vol.8 No.2, [1997])

[3] J.Linn. "GSS API", RFC1508. Sept 1993

[4] J. wray, "GSS API : C-Binding", RFC1509, Sept. 1993