패킷제어를 이용한 무부하 데이터베이스 접근보안

신성철*, 이경석*, 류근호*
*충북대학교 데이터베이스 및 바이오인포매틱스 연구실
e-mail : superior3@naver.com, {kslee, khryu}@dblab.chungbuk.ac.kr

No-load Database Security using Network Packet Control

Sung Chul Sin*, Kyeong Seok Lee*, Keun Ho Ryu*
*Database/Bioinformatics Laboratory, Chungbuk National University

요 약

수많은 데이터를 사용하는 기업 및 기관들은 전용 데이터베이스를 구축하여 모든 정보들을 저장, 관리하고 있다. 개인정보를 포함하여 비인가된 중요 정보들이 저장되어있음에도 불구하고 대부분의 경우 데이터베이스에 대한 보안적용이 되어있지 않거나 아주 미비한 상태이다. 보안사고의 대부분이 데이터베이스에 저장된 중요 정보가 유출되는 점을 보았을 때 데이터베이스 자체에 대한 보안시스템을 구축하여 예방하는 것이 보안사고의 피해를 막을 수 있는 가장 핵심적인 방법이라 할 수 있다.

이에 본 연구에서는 네트워크 패킷을 모니터링하여 데이터베이스의 성능에 영향을 미치지 않고 보안기능을 수행할 수 있는 Sniffing 을 이용한 방안을 제안한다. 제안 보안시스템은 별도의 하드웨어에서 기능을 수행하며 운영 중인 데이터베이스에 서비스의 중단 없이 보안시스템을 구축할 수 있도록 하였다. Sniffing 방식에서 접근제어 기능을 수행 할 수 있도록 알고리즘 설계를 하였으며 감사로그를 기록할 때 가장 많은 부분을 차지하는 SQL 에 대해 MD5 해시함수를 적용하여 해당 로그에 대한 데이터크기를 크게 줄일 수 있었다. 운영중인 데이터베이스 환경에 영향을 주지 않으면서 높은 수준의 감사성능을 제공하고 다양한 보안 정책을 간단하게 적용할 수 있도록 구현하였으므로 데이터베이스에 저장된 중요정보의 유출을 예방하고, 보안사고가 일어났을 때 추적 및 증거자료로 활용할 수 있을 것이다.

1. 서론

오늘날 네트워크의 발달로 인터넷을 통해 거리의 제약을 받지 않고 정보공유가 가능해짐으로 인해 손쉬운 정보교환이 가능해졌다. 이로 인해 정보에 대한 접근이 간단해짐으로 편리함을 얻었지만 원치 않는 정보 유출 및 해킹 등의 문제가 발생함으로써 정보보안에 대한 필요성과 관심이 커지게 되었다.

CIS 와 국가정보원에 통계에 따르면 해킹과 정보유출 건수는 매해 2 배씩 증가하고 있으며, 정보유출의 경우 내부사용자에 의한 사건이 전체 발생 건수의 80% 이상에 이른다는 발표결과가 있다. 이러한 내부사용자에 의한 정보유출은 데이터베이스에 저장되어 있는 핵심정보를 추출하여 외부로 유출시키기 위한 목적이 대부분이다. 해킹의 목적 또한 핵심정보가 담겨져 있는 데이터베이스에 접근하여 외부로 정보를 유출시키는 것이 가장 큰 목표이다.[1]

이에 본 연구에서는 나날이 정보의 중요성이 커지고 있고 이를 보호하고자 하는 효과적인 보안시스템을 구축하고자 한다. 이를 위해 운영중인 데이터베이스 시스템에 성능 영향을 주지 않으면서 간단하게 적용할 수 있고 뛰어난 감사성능을 제공하는 데이터베이스 보안시스템을 구축할 수 있도록 하는 것이 본 연구의 목표이다.

2. 관련연구

2.1 데이터베이스 외부시스템을 이용한 보안

데이터베이스의 보안 개념 및 재량 접근 제어와 필수 접근 제어의 절차에 대해서 살펴보았다. 이렇게 데이터베이스 자체로 강력한 보안 기능을 갖고 있는데 실제 업무에서는 이러한 보안기능들을 적절히 활용하기에는 어려움들이 따른다.

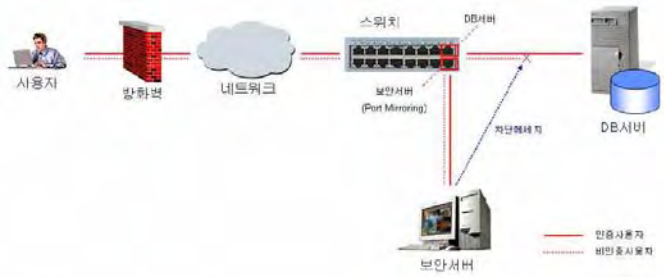
예를 들어 실제 업무시스템에서는 하나의 데이터베이스 계정을 여러 사용자가 공유하여 사용하는 일이 많게 되어 특정 데이터가 삭제되었거나 몰래 정보를 빼내었을 때 어떤 사람이 그러한 행위를 하였는지 찾아낸다는 것은 불가능에 가깝다고 할 수 있다. 또한 데이터베이스 자체의 감사기능도 사용하지 않는 경우가 많다. 이러한 원인으로는 운영중인 시스템에 추가적인 부하를 가하지 않아 서비스에 문제가 없도록 하기 위해서다. 만약 추가된 기능으로 인해 부하가 발생하여 서비스제공에 문제가 발생한다면 그 피해의 규모는 매우 클 수 있음으로 굳이 그러한 위험을 발생시키지 않으려는 것이다.

실제 업무시스템에 성능 영향을 미치지 않으면서

보안성을 얻고자 한다면 데이터베이스 외부 시스템을 사용하는 것이 해결방법이 될 수 있다. 외부 시스템을 이용하는 방법은 크게 암호화 방식과 접근제어 방식이 존재하며 접근 제어 방식의 경우 보안시스템 구성방식에 따라 Sniffing 방식과 Gateway 방식으로 나눌 수 있다. Proxy 방식인 Sniffing 을 이용하면 운영중인 데이터베이스에 부하를 주지 않고 보안시스템을 구축할 수 있으며 동작중인 운영시스템에도 다른 변경을 가하지 않아도 손쉽게 적용할 수 있게 된다[2].

2.1.1 Sniffing 방식

Sniffing 방식은 스위치 장비에서 Port Mirroring 을 통하여 데이터베이스 서버로 송수신되는 모든 패킷을 별도로 구축한 데이터베이스 보안시스템에 보내어 접근제어와 감사를 수행하는 방식이다. 해당 스위치에 Port Mirroring 기능이 없는 장비라면 별도의 TAP 장비를 사용하여 구축 가능하다.



(그림 1) Sniffing 방식을 사용한 접근제어

사용자가 SQL 문을 데이터베이스에 전송하여 데이터를 요청한 경우를 가정해본다면, 이러한 모든 과정은 네트워크 패킷을 이용하여 데이터베이스와 통신하게 된다. 이때 데이터베이스에 송수신 되는 모든 패킷을 보안시스템에 전송하여 저장하거나, 이 패킷을 분석하여 미리 적용된 정책에 따라 차단 패킷을 전송하여 세션을 끊는 역할을 수행한다. 이 방식의 가장 큰 장점으로는 운영중인 데이터베이스에 영향을 미치지 않으면서도 사용자가 요청한 모든 정보를 저장할 수 있다는 것이다. 이미 구축되어 있는 시스템에 손쉽게 적용할 수 있고 보안시스템에 문제가 발생하여도 서비스중인 데이터베이스에 미치는 영향이 없기 때문에 대부분의 경우 가장 현실적인 대안이 되는 방식이다.[3]

2.1.2 Gateway 방식

Gateway 방식의 경우 물리적으로 사용자와 데이터베이스를 연결하는 중간에 위치한 구조이다. 사용자가 데이터베이스에 접근하려면 반드시 보안시스템을 거쳐야만 가능하도록 구성한 것이다. Sniffing 방식과 비교하면 세부적인 보안정책 적용과 완벽하게 접근제어가 가능하지만, 운영중인 데이터베이스 서비스에 미치는 영향이 크다는 단점이 존재한다. 만약 보안시스템에 문제가 발생하게 되면 운영중인 데이터베이스에 대한 서비스 중단이 이루어지게 되는 구조여서 상당한 Risk 를 갖게 된다. 이러한 점을 보안하기 위해 별도의 FOD(Fail Over Device) 장비를 구축하거나 네트워크를 이중화하는 등의 방법을 이용하기도 한다.

(그림 2)에서는 게이트웨이 방식의 구성도를 나타낸다.[3]

(그림 2) Gateway 방식을 사용한 접근제어

2.2 데이터베이스 보안시스템 구성

본 연구에서 제안한 Sniffing 방식의 데이터베이스 보안시스템은 보안서버, 매니저, 저장소, 파싱 모듈, 정책 모듈, 접근제어 모듈, 로깅 모듈 등으로 구성되어 있다. 데이터베이스 보안시스템은 별도의 독립된 하드웨어 장비에서 동작하도록 패키징 되어 있다. 해시 함수 적용, 그룹정책적용, Sniffing 된 패킷들은 Queue 에 담겨 있다가 처리하게 하였다. 이러한 방식은 다음과 같은 장점들을 지니게 되었다.

첫째, 운영중인 데이터베이스에 보안시스템을 적용할 때 기존의 어떠한 환경변화 없이 손쉽게 구축이 가능하다. 운영시스템의 중단 없이 해당 스위치의 Port Mirroring 설정 후 별도로 구축된 데이터베이스 보안시스템과 네트워크 연결만 하면 적용이 마무리된다.

둘째, 데이터베이스 보안시스템을 적용한 후 보안시스템을 관리하는 부담을 감소시킬 수 있다. 대부분의 보안 및 서버 관리자는 새로운 시스템을 도입한 후 해당 시스템 관리에 많은 시간을 할애해야 하는 것이 현실이지만, 본 연구에서 제안한 방식을 적용할 경우 보안정책을 별도로 변경하는 것 이외에는 특별한 관리의 노력을 요하지 않으므로 관리자의 업무부하를 감소시킬 수 있다.

셋째, 실제 운영서비스에 영향을 주지 않는 가장 안전한 방식이다. 만약 데이터베이스 보안시스템에 문제가 발생하더라도 운영시스템에는 영향을 주지 않아야 한다. Sniffing 방식을 이용했을 경우 이러한 조건을 만족시킬 수 있으며 운영시스템의 서비스에 지장을 받지 않고 사용 가능하다.

넷째, 정책 적용 시 해당 항목들을 별도의 그룹에 할당할 수 있도록 하여 정책설정이 간단하도록 하였다. 다섯째, 감사로그 저장할 시 똑같은 SQL 문은 한 번만 저장되도록 해시함수를 사용하여 감사로그의 기록되는 양을 획기적으로 줄일 수 있었다.

<표 1> 데이터베이스 보안시스템의 소프트웨어 구성요소

구분	모듈	역할
데이터베이스 보안서버	sf	네트워크 패킷을 캡쳐하여 분석하는 기능을 수행
	maxsvr	sf 의 상태를 체크하고 제어하는 기능
감사로그 저장소	MySQL 5.5	감사로그를 저장하는 저장소
매니저	Max Manager	보안시스템을 관리하기 위한 C/S 프로그램
OS	CentOS 5.5	Linux Kernel 2.6 기반의 운영체제

2.2.1 감사로그

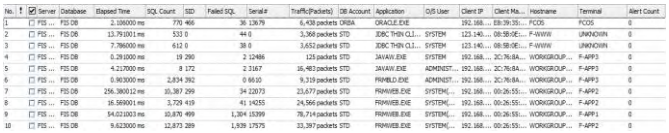
제안 데이터베이스 보안시스템에서 감사로그 모듈은 아래와 같은 특징을 지닌다. 첫째, 데이터베이스에 감사로그를 기록하므로 감사데이터 관리가 편리하며 차후 분석에 있어 다양한 자료로 추출해 내기 용이해지는 장점이 있다. 둘째, 데이터베이스의 느린 insert 작업을 개선하기 위해 별도의 Queue 를 적용시켰다. 데이터베이스에 바로 감사로그를 저장하면 insert 작업의 부하로 인해 파일시스템에 저장하는 것 보다 느리므로 별도의 Queue 를 배치하여 Queue 에 우선적으로 저장한 뒤 Queue 에서 데이터를 추출하여 데이터베이스에 기록하는 방식을 적용하였다. 셋째, 감사로그의 양을 감소시키기 위해 MD5 해시함수를 적용하였다. 감사로그의 가장 많은 용량을 차지하는 것은 SQL 문이다. 같은 SQL 문이 실행될 때는 한번만 기록되는 방식을 선택하여 감사로그의 용량을 줄이고자 하였다. SQL 문은 별도의 테이블에 기록되며 'SQL 로그' 테이블에는 SQL 문을 MD5 해시함수를 적용한 해시값만 기록하도록 한다. 이처럼 같은 SQL 문은 2 번이상 저장되지 않도록 하여 감사로그의 기록량을 크게 줄이도록 하였다. 해시함수를 적용하면 SQL 의 길이에 관계없이 16byte 의 데이터만 필요하게 된다.

2.2.2 접속차단

Sniffing 방식의 가장 제한되는 점이라면 세밀한 접근제어를 하기 힘들다는 부분이다. 패킷들이 물리적으로 보안서버를 거쳐가는 구조가 아니기 때문에 In-line 구조 방식을 지닌 Gateway 방식에 비해 접속차단을 구현하기가 어렵다. 이러한 것들이 Sniffing 방식의 구조적 한계이지만, 본 연구에서 제안한 데이터베이스 보안시스템에서는 TCP Reset 메시지를 전송하는 방법을 사용하여 접속차단까지 구현하였다. TCP Reset 메시지를 전송하여 접속을 차단하기 위해서는 TCP 패킷의 Header 구조를 파악하여야 한다. 주로 참고하게 되는 Sequence Number 와 Ack Number 에 번호를 할당하며 차단 메시지를 보낼 때는 Code Bits 중 RST 부분을 설정하여 Reset 메시지를 전송하게 된다.

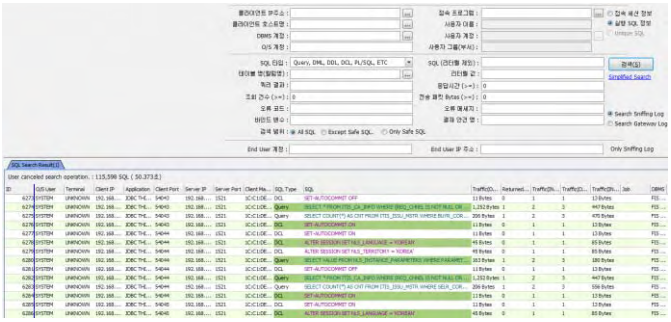
2.2.3 매니저 부분

매니저에서는 등록된 대상 데이터베이스를 모니터링, 정책설정, 감사데이터 조회와 분석하는 부분으로 구성되어 있다. [그림 3]은 대상 데이터베이스를 모니터링 하는 장면이다. 부가적으로 SQL 의 평균응답 시간을 측정할 수 있어 데이터베이스의 성능 평가도 함께 수행이 가능하며 해당 자료를 바탕으로 Tuning 의 포인트로 삼을 수 있다. 성능적으로 부하가 많은 데이터베이스를 발견하게 되면 정책 위반건수를 참고하여 보안적인 측면으로 다양한 분석이 가능하게 된다.



(그림 3) 감시 대상 데이터베이스 모니터링 화면

(그림 4)는 SQL 실행 이력을 조회할 수 있는 화면을 나타낸다. 요청한 SQL 의 내용을 파악할 수 있기 때문에 해당 사용자의 목적을 알 수 있으며 송신지의 각종 정보들 또한 나타내기 때문에 추적 및 분석에 용이하다. 앞서 설명한 것과 같이 SQL 의 응답시간을 알 수 있으므로 성능적인 측면의 분석 또한 가능하다. 만약 어플리케이션 서버가 구축되어 있다면 감사되는 SQL 은 한정적이게 된다. 이러한 점을 이용하여 어플리케이션에서 보내는 SQL 이외의 것들은 사용자가 직접 데이터베이스에 접속하여 실행한 SQL 들 이므로 집중적으로 분석할 소지가 있다.



(그림 4) SQL 실행 이력 조회 화면

3. 평가

Sniffing 방식을 적용한 데이터베이스 보안시스템에서 평가는 감사성능 테스트 항목에 대해 진행하였다.

3.1 데이터베이스 보안시스템 테스트 환경

보안시스템의 성능 및 기능 테스트를 위하여 해당 환경을 구축하였다. 부하테스트를 위해 PerfOne 을 사용하였고 성능측정은 저장소 내의 데이터를 기준으로

하였다. 데이터베이스 보안서버는 CentOS 5.5 64bit Kernel 2.6 을 사용하였고, 저장소(Repository)는 MySQL 5.5 을 사용하였으며, CPU-2.66 Ghz Quad Core, Memory-8GB 를 하드웨어로 사용하였고, 부하 발생기는 Perfone 사용하고, 측정 기준은 감사데이터 건수 카운트로 하였다.

3.2 감사성능 테스트

초당 로깅할 수 있는 SQL 의 최대 건수를 측정하기 위해 Perfone 을 이용하여 점진적인 부하를 가하여 테이블에 저장된 데이터의 초당 감사건수를 카운트하였다. 부하발생기인 PerfOne 을 이용하여 사용자 수와 SQL 문을 설정한 뒤 에이전트로 등록시켜 사용하며 많은 부하를 발생시키기 위해 동시 접속자수를 최대한 늘려 테스트 하였다. 동시접속자 수를 1000 명으로 설정하고 각 접속자의 SQL 실행 횟수를 10 으로 하였을 때 초당 최대 감사 성능에 근접한 결과를 측정할 수 있었다.

-초당 최대 검사 건수 : 781 건

동시 접속자 수를 변경하거나 SQL 실행횟수를 달리하면 결과는 얼마든지 달라질 수 있으며 해당 결과는 보안시스템의 하드웨어 사양 및 네트워크 환경에 따라서도 변경될 수 있다는 것을 감안해야 한다. 구축하고자 하는 데이터베이스의 평균사용량을 측정한 후 그에 맞는 하드웨어 구성과 네트워크 환경을 설정한다면 원하는 성능목표를 달성할 수 있을 것이다.

4. 결론

Sniffing 방식을 사용한 데이터베이스 보안기법은 운영중인 시스템에 설계 및 구현을 어렵지 않게 적용 가능 하며 서비스에 부하를 주지 않는 안정적인 데이터베이스 보안시스템이다. 이러한 보안시스템은 아래와 같이 네가지 측면에서 의미를 갖게 된다고 할 수 있다.

첫째, 앞서 언급된 것처럼 운영중인 시스템에 어떠한 행위를 가하지 않고 Port Mirroring 된 네트워크 장치에 데이터베이스 보안시스템을 연결하기만 하면 간단히 적용이 완료된다.

둘째, 실제 데이터베이스에 영향을 미치지 않는 가장 안전한 보안 방식이다. 동작중인 데이터베이스 보안시스템에 문제가 발생하더라고 실제 서비스에는 전혀 영향을 주면 안되며, Sniffing 기법을 사용했을 경우 이러한 요건을 충족시킬 수 있다.

셋째, 측정 결과에서도 파악할 수 있었듯이 뛰어난 감사성능을 바탕으로 실제 운영서비스에 만족할 수 있는 목표를 달성할 수 있으며 다양한 정책설정으로 원하는 보안정책의 수립이 가능하다.

넷째, 해시 함수의 사용으로 같은 SQL 문에 대해서는 한번만 저장되도록하여 감사로그의 저장량을 크게 줄일 수 있었기 때문에 추가적인 하드웨어의 확장에 따른 부담을 감소시킬 수 있다.

본 연구에서 제안한 Sniffing 기법은 간단하고 안전하지만 일정 부하이상에서는 감사기록이 유실 될 수 있다는 단점이 존재한다. 하드웨어 사양을 향상시켜 해결할 수 도 있지만 근본적인 알고리즘 효율을 개선하여 유실되는 데이터를 줄이는 방안에 대해 지속적인 연구가 필요하다.

사사

이 논문은 2013 년도 미래창조과학부의 재원으로 한국연구재단 (No.2013R1A2A2A01068923) / 미래창조과학부 및 정보통신산업진흥원의 대학 IT 연구센터 지원사업/ IT 융합고급인력과정지원사업(NIPA-2014-H0301-14- 1022)의 연구결과로 수행되었음.

참고문헌

[1] 개인정보보호, www.i-privacy.kr, 2014

[2] Ron Ben Natan, "Implementing Database Security and Auditing, " Digital Press, 2005

[3] 전웅렬, “접근제어형 데이터베이스 보안 시스템의 보호프로파일”, 성균관대학교 정보통신공학부 정보보호연구소 2007