제21회 한국정보처리학회 춘계학술발표대회 논문집 제11권 제1호 (2004. 5)

내부자에 의한 정보 유출 방지를 위한 안전한 데이터 관리 시스템

김경태, 윤희용
성균관대학교 정보통신 공학부
k1k1t1@hamail.net, youn@ece.skku.ac.kr

Secure Data Management System for Prevention Leakage by Insider

Kyung Tae Kim and Hee Yong Youn
School of Information and Communications Engineering, Sungkyunkwan University

요 약

최근 정보 시스템의 활용도가 높아지고 인터넷을 통한 공간의 제약 없이 네트워크를 통해 먼 거리의 컴퓨터에 원격으로 접속하고 상호간에 자유롭게 데이터를 공유하게 됨으로써 정보유출 위험성이 날로 증가하고 있다. 현재 정보 유출 방지를 위해 다양하고 효율적인 방법들이 연구되어 오면서, 수많은 정보 시스템들은 사용자 인증 및 암호화 기술을 사용하여 보안성이 크게 강화되었다. 그러나 여전히 내부관리자나 사용자에 의한 정보 유출에 대해서는 취약한 실정이다. 본 논문에서는 커버로스 인증 시스템 및 접근제어목록을 이용하여 대부분의 암호화 파일시스템에서 문제로 제기되는 파일 공유 문제를 해결하고, 사용자 단위로 관리할 필요가 없는 시스템수준의 키 관리를 하는 안전한 데이터 관리 시스템을 제안함으로써 내부자에 의한 정보 유출 방지 및 외부자 침입을 탐지할 수 있는 방안을 제시한다.

1. 서론

정보 통신 기술과 대용량의 정보 저장 기술이 발달하면서 우리는 언제 어디서나 많은 정보들을 쉽게 접할 수 있다. 이로 인해 우리의 생활들이 많이 윤택해 졌지만, 부작용 또한 만만치 않다. 최근 들어 정보 시스템의 활용도가 높아지고 인터넷을 통한 공간의 제약 없이 네트워크를 통해 먼 거리의 컴퓨터에 원격으로 접속하고 상호간에 자유롭게 데이터를 주고 받으며 공유하게 됨으로써 정보 시스템의 정보 유출 위험성이 날로 증가하고 있다. 특히 정보 보안 기술의 발달로 외부 공격자에 의한 정보 유출은 줄어드는 추세이지만, 내부 관리자나 사용자에 의한 정보 유출이 날로 증가하고 있다.

많은 스토리지 정보 시스템들은 정보 유출을 방지하기 위해서 사용자 인증 및 암호화 기술을 사용하고 있다. 대표적인 스토리지 정보 시스템으로 CFS(Cryptographic File System), NFS(Network File System), Cepheus, NASD(Network-Attached Secure Disks) 등이 있다 [1-4]. 이런 시스템들이 이용하는 사용자 인증 추세는 PKI(Public Key Infrastructure) 환경에서 인증이 이루어지는 방향으로 나아가고 있으며, 또한 커버로스(Kerberos)처럼 제삼자가 인증을 해주는 방향으로 나아가고 있다. 암호화 기술은 DES나 AES을 이용하는 추세이다. 이렇게 정보 시스템들이 보안성을 강화는 하였지만 여전히 내부자에 의한 정보 유출이 가능하다.

스토리지 정보 시스템들의 키 관리는 두 가지 방법이 있다. 첫 번째는 스토리지 서버나 다른 제삼자인 인증 서버에서 관리하는 방법이다. 두 번째는 파일의 소유자가 관리하면서 사용자에게 분배하는 방법이다. 이 경우는 사용자가 키를 요구하는 경우 소유자는 사용자를 인증한 후 분배한다. 두 번째 방식은 사용자 권한 취소시 문제가 된다. 왜냐하면 사용자가 받은 키를 은닉할 수 있기 때문이다. 이러한 이유 때문에 스토리지 정보 시스템들은 사용자 권한 취소시 그 사용자가 read나 write 권한을 가지고 있던 파일들을 새로운 키로 다시 암호화를 한다.

이 두 가지 방식 모두 소유자들이 파일을 암호화한 키들은 알고 있다. 또한 다른 사용자들도 read나 write시에 키를 받아 알고 있다. 이에 소유자가 직장을 옮긴다거나 마음먹기에 따라 정보 유출을 시도할 수 있다. 따라서 본 논문에서는 이러한 문제점을 해결하는 방법을 제안한다. 본 논문에서 제안하는 안전한 데이터 관리 시스템은 소유자나 사용자가 암호화한 파일을 유출하여도 암호화 키를 모르기 때문에 열어 볼 수 없다. 또한 사용자 권한 취소시, 그 사용자가 권한을 가지고 있던 파일들을 새로운 키로 다시 암호화하는 오버헤드 같은 것은 존재하지 않는다. 그리고 추가적인 장점으로 공격자의 침입탐지를 수행한다 [5].

본 논문의 구성은 다음과 같다. 2 장에서는 대표적인 스토리지 시스템인 CFS에 대해서 알아본 다음 커버로스(Kerberos) 인증 시스템에 대해서도 알아 본다. 3 장에서는 본 논문에서 제안하는 안전한 데이터 관리 시스템을 제안하고 기존 시스템들과 차이점에 대해 알아본다. 마지막으로 4 장에서 결론을 맺는다.

2. 관련연구

2.1 암호화 파일 시스템(CFS)

파일시스템 수준에서의 암호화는 사용자로 하여금 데이터 암호화를 위한 특별한 조작을 필요 없게 하는 사용상의 투명성을 제공하고, 시스템 침입자에 대해서 그들이 원하는 정보를 감춤으로써 데이터에 대한 보안성을 제공해 준다. CFS는 NFS 파일시스템 내에 암호화 기능을 추가한 것으로, 암호화된 파일에 대해 표준 유닉스 파일시스템 인터페이스를 적용함으로써 시스템 수준에서의 보안 저장장치(secure storage)를 제공한다[1]. 사용자 수준의 구현이기 때문에 많은 문맥 교환 등으로 시스템 성능에 한계가 있다. Transparent CFS는 원격 NFS 서버와 통신하는 수정된 클라이언트 쪽의 NFS 커널 모듈이다[6]. 커널 영역에서 구현되어 암호화 서비스와 파일시스템 사이에 더 깊은 통합을 제공함으로써 CFS를 개선하였다. 사용자키는 /etc/tcfspasswd 파일에 저장되는 데, 이는 차후에 보안성을 감소시킨다. Cryptfs는 Stackable Vnode Layer loadable kernel module로써 설계 및 구현된 파일시스템이다[7]. 사용자에게 클라이언트 파일 시스템을 '캡슐화' 함으로써 투명한 암호화 기능을 수행하며 커널 수준에서 동작한다. 암호화 키는 사용자 ID 뿐만 아니라 프로세스 세션 ID 에 기반하고, 커널 메모리에 보관하므로 좀 더 강한 보안성을 제공하지만, 암호화를 수행하는 추가된 층으로 인해 시스템에 과부하를 주고, 다른 사용자들간의 공유 문제가 어렵다는 단점이 있다. 그리고 사용자가 암호키를 제공하고 그 키를 알고 있기 때문에 유출 등 파일 시스템에 피해가 발생할 수 있다는 단점이 있다

2.2 커버로스(Kerberos)

커버로스 인증 메커니즘은 인증 서버(AS: Authentication Server)와 티켓 허가 서버(TGS: Ticket Granting Server)로 나누어져 있어 사용자가 여러 번 반복하여 패스워드를 입력해야 하는 불편을 없앴으며 또한 클라이언트에 대한 인증뿐 아니라 인증 서버와 티켓 허가 서버에 대한 인증도 동시에 수행되는 상호 인증(Mutual Authentication)을 제공한다.

커버로스는 버전 5까지 나왔으며, 버전 5의 기본 서비스 과정은 클라이언트-인증 서버간의 교환, 클라이언트-티켓 허가 서버간의 교환, 그리고 클라이언트와 서버간의 교환으로 나누어 생각할 수 있다.

1) 클라이언트-인증 서버간의 교환

<메시지 1>

$C \rightarrow AS: Options||ID_C||Realm_C||ID_{TGS}||Times_1||Nonce_1$

<메시지 2>

$AS \rightarrow C: Realm_C||ID_C||Ticket_{TGS}||E_{Kc}[K_{C,TGS}||Times_1||Nonce_1||Realm_{TGS}||ID_{TGS}]$

$- Ticket_{TGS}=E_{K_{TGS}}[Flags||K_{C,TGS}||Realm_C||ID_C||AD_C||Times_1]$

메시지 1에서는 서비스를 받고자 하는 클라이언트가 인증 서버에게 요청을 하게 된다. 커버로스에서 클라이언트-서버의 마스터키는 사용자의 패스워드를 이용한다. 메시지 1에는 자신의 ID와 자신이 속해있는 영역(realm), 서비스를 받고자 하는 티켓 허가 서버를 명시해야 한다. 또, 티켓이 유효한 시간을 나타내는 Times, 오류를 검사하기 위해, 혹은 보낸 메시지가 여러 개일 때, 이를 구별해주기 위한 Nonce를 포함하여 보내게 된다. Times와 Nonce는 커버로스 버전5에서 새롭게 제안된 구조로 버전4에서의 Timestamp의 역할을 나누어 하게 된다. 메시지 1의 요청을 받은 인증 서버는 클라이언트의 신원을 확인하여 티켓 허가 서버와의 통신을 가능하게 하는 인증서와 같은 역할을 하는 티켓을 클라이언트에게 전달한다. 이 단계에서 클라이언트가 전달받는 티켓은 티켓 허가 서버의 마스터키로 암호화되어 있기 때문에 클라이언트는 그 내용을 볼 수 없다. 그리고 클라이언트가 티켓 허가 서버와의 교환에 쓸 세션 키는 클라이언트의 마스터키로 암호화하여 전달된다.

2) 클라이언트-티켓 허가 서버간의 교환

<메시지 3>

$C \rightarrow TGS: Options||ID_V||Times_2||Nonce_2||Ticket_{TGS}||Authenticator1_C$

<메시지 4>

$TGS \rightarrow C: Realm_C||ID_C||Ticket_V||E_{Kc,TGS}[K_{C,V}||Times_2||Nonce_2||Realm_V||ID_V]$

$- Ticket_V= E_{Kv}[Flags||K_{C,V}||Realm_C||ID_C||AD_C||Times_2]$

$- Authenticator1_C= E_{K_{C,TGS}}[Realm_C||ID_C||TS_1]$

메시지 3은 인증 서버로부터 받은 티켓을 티켓 허가 서버에게 전달한다. 또한 클라이언트가 서비스를 요청할 서버의 ID, 시간, Nonce와 클라이언트를 인증할 수 있는 인증자를 전달한다.

메시지 4단계에서는 인증을 확인한 티켓 허가 서버가 클라이언트와 서버가 사용할 세션 키를 서버에게 전달될 티켓에, 그리고 클라이언트와 티켓 허가 서버 사이의 세션 키로 암호화된 곳에 포함시켜 전

달한다. 두 Times는 티켓 허가 서버가 받은 메시지의 요청이 옳은 것인지를 판단하는데 사용된다.

3) 클라이언트-서버 인증 교환

<메시지 5>

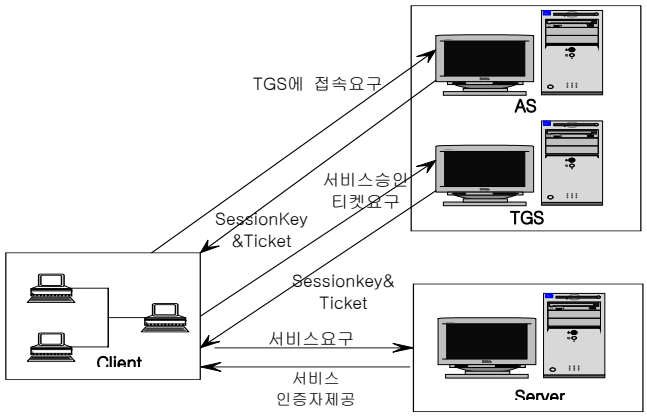
$C \rightarrow V$: Options||$Ticket_V$||$Authenticator2_C$

<메시지 6>

$V \rightarrow C$: $E_{Kc,v}$[TS_2||Subkey|| Seq#]

- $Ticket_V = E_{Kv}[Flags||K_{C,V}||Realm_C||ID_C||AD_C||Times_2]$
- $Authenticator2_C = E_{K_{C,v}}[Realm_C||ID_C||TS_2||Subkey||Seq\#]$

메시지 5단계에서는 클라이언트가 티켓 허가 서버로부터 받은 티켓을 서버에게 전달하게 된다. 또한 클라이언트를 인증할 수 있는 인증자도 전달된다. 마지막으로 메시지 6단계에서는 서버와 클라이언트 사이의 세션 키로 암호화된 메시지를 전달함으로써 클라이언트가 서버를 인증하게 된다. 이 두 단계에서 보조키(Subkey)는 인증 과정이 끝난 후 클라이언트와 서버 사이의 메시지 전달 과정에서 암호화에 필요한 키이며 Seq#는 메시지에 대한 일련의 번호로 전달 과정에서 오류가 생겼을 때 다시 보내야 하는 부분을 구별하기 위한 것이다. 그림 1은 커버로스 인증 시스템을 나타내고 있다.



(그림 1) 커버로스 인증 시스템.

3. 안전한 데이터 관리 시스템

3.1 시스템 체계

이번 장에서는 2장에서 설명한 커버로스 인증 시스템 및 접근제어목록(ACL: Access Control Lists)을 이용하여 대부분의 암호화 파일시스템에서 문제로 제기되는 파일 공유 문제를 해결하고, 사용자 단위로 관리할 필요가 없는 시스템 수준의 간단한 키 관리를 하는 안전한 데이터 관리 시스템을 제안한다.

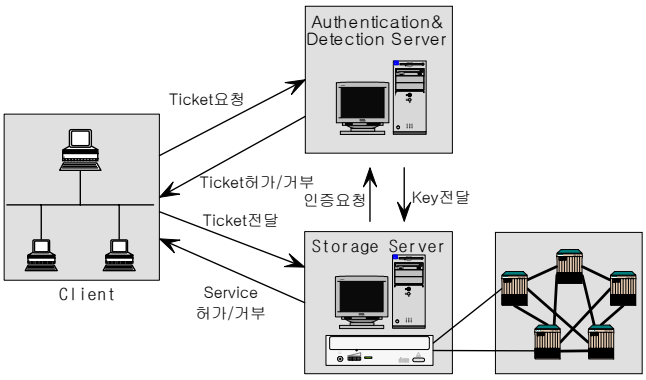
안전한 데이터 관리 시스템은 인증 서버, 스토리지 서버, 클라이언트로 구성된다. 여기서 본 논문에서 제안하는 인증 서버는 커버로스 인증 시스템에서의 인증 서버로 추가적으로 데이터 별로 사용자들의 접근제어목록 및 데이터를 암호화하는 키를 저장 관리한다. 다른 점은 TGS가 클라이언트에게 주는 티켓 $Ticket_V$에 일련번호(SN: Serial Number)가 추가된다는 것이다.

$Ticket_V = E_{Kv}[SN||Flags||K_{C,V}||Realm_C||ID_C||AD_C||Times_2]$

그리고 인증서버가 그 티켓의 일련번호와 사용자 ID을 티켓의 라이프타임 동안 보관하게 된다는 것이다.

안전한 데이터 관리 시스템 사용 절차를 세부적으로 나누어 보면 3 가지 단계로 나누어 볼 수 있다. 첫 번째 단계는 데이터를 read하거나 write하기는 원하는 클라이언트가 2장에서 설명한 커버로스의 절차처럼 티켓을 얻는 것이다.

두 번째 단계로 티켓을 받은 클라이언트가 그 티켓을 스토리지 서버에 전달하고 스토리지 서버의 티켓 확인 후에 데이터에 대한 read나 write을 요구하면 스토리지 서버는 클라이언트로 받은 티켓과 클라이언트가 요구한 파일명을 인증서버에게 전송한다.



(그림 2) 안전한 데이터 관리 시스템.

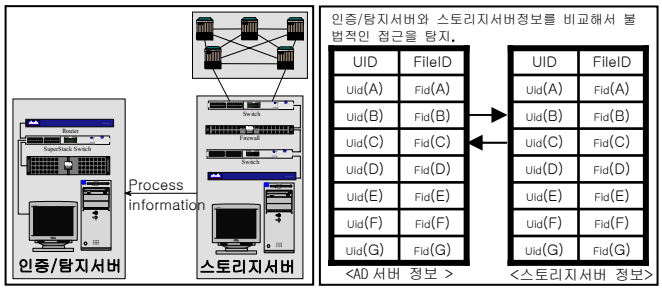
안전한 데이터 관리 시스템은 데이터의 안전한 관리를 위해 인증 서버와 스토리지 서버가 데이터의 암호화를 전적으로 관리한다. 즉 클라이언트는 단지 데이터에 대한 read나 write을 요구할 뿐이다. 마지막 세 번째 단계가 이에 관한 단계가 되겠다. 세 번째 단계로 인증서버는 스토리지 서버로부터 받은 티켓의 이상유무를 확인한다. 이상이 없다면 인증서버는 두 가지 경우로 나누어 행동한다. 먼저 첫 번째 경우로 최초의 파일 생성에 관한 요구라면 인증서버는 그 파일에 대한 접근제어목록을 만들고 암호키를 랜덤하게 생성하여 보관과 동시에 스토리지 서버에게 전송한다. 두 번째 경우로 생성되어 있는 파일에 대한 read나 write에 관한 요구라면 먼저 접근제어목록을 확인하여 클라이언트의 적합 여부를 확인한 후 read의 경우는 기존의 암호키를 스토리지 서버에 전송하고 write 경우는 새로운 암호키를 랜덤하게 생성하여 스토리지 서버에 전송한다. 그러면 스토리지 서버는 인증서버에게 받은 암호키를 클라이언트가 알 수 없도록 커널상에서 데이터를 암호화하거나 복호화한 후 암호키를 삭제하고 암·복호화된 데이터를 저장매체에 저장하거나 클라이언트에 전달한다. 이 단계는 보안성을 어느 정도에 두느냐에 따라 달라진다. 만일 강력한 보안성을 가진 시스템을 원한다면 매번 데이터가 read되거나 write될 시 암호키를 변경하여 해당 데이터를 다시 암호화한 후 저장시킨다. 반면에 낮은 보안성을 가진 시스템을 원한다면 최초로 저장될 때 생성된 암호키를 그 파일이 삭제될 때까지 사

용할 수 있다. 스토리지 서버와 인증서버간 암호키 전달 시 키 노출을 막기 위해 RPC 등과 같은 암호화 전송 프로토콜을 이용한다.

그림 2는 전반적인 안전한 데이터 관리 시스템의 체계도를 나타내고 있다.

3.2 침입탐지

본 논문에서 제안하는 안전한 데이터 관리 시스템은 안전한 데이터 관리 외에도 공격자의 시스템 침입을 탐지할 수 있다. 여기서 인증 서버는 사용자 인증, 접근제어목록 관리 외에도 침입탐지 에이전트 역할을 수행한다. 인증 서버는 데이터의 암호키를 요구한 사용자와 데이터 정보를 최대 티켓의 라이프타임 동안 유지하면서 주기적으로 스토리지 서버에서 실행되고 있는 프로세스 정보를 보고 받는다. 그리고 그 정보와 유지하고 있는 정보를 비교하여 침입탐지를 수행한다. 안전한 데이터 관리 시스템은 주기적인 검사 시 암호키가 요구되지 않은 데이터가 실행되고 있을 경우에는 침입으로 간주할 것이다. 그림 3은 침입탐지 체계를 나타내고 있는데 (A)는 스토리지 서버에서 인증/탐지서버로 Process 정보를 전달하는 과정, (B)는 인증/탐지서버에서의 정보비교를 나타내고 있다.



(A) (B)

(그림 3) 침입 탐지 체계

위에 안전한 데이터 관리 시스템을 제안하였다. 이 시스템은 2 장에서 제기한 기존 시스템의 문제점을 보완한다. 먼저 기존 시스템은 사용자들이 자기의 소유한 데이터이던지 또는 다른 사용자가 소유한 데이터이던지 해당 암호키를 알고 있기 때문에 유출 시 심각한 문제가 발생한다. 안전한 데이터 관리 시스템은 위의 설명에서 나타나듯 어떠한 클라이언트도 자신이 관련된 어떠한 데이터의 암호키도 알 수 없다. 따라서 수단과 방법을 가리지 않고 자신이 생성한 데이터를 유출시켜도 암호키를 모르기 때문에 스토리지 시스템 외부에서는 그 데이터의 내용을 볼 수 없다. 만일 시스템 관리자도 인증서버에 저장된 암호키를 볼 수 없도록 시스템이 설계된다면 관리자에 의해 유출되어도 데이터는 무용지물이 된다. 이와 같은 데이터 관리 시스템으로 기존 시스템에서 발생하는 데이터 유출문제는 해결될 수 있다. . 비상시를 위해 암호키들을 백업시켜 비상시에만 시스템 관리자들이 열람할 수 있도록 설계하면 시스템은 더욱 안전할 것이다.

기존 시스템의 다른 문제점은 기존 사용자의 권한 취소 시 해당 사용자가 접근했던 데이터의 안전성을 위해 재암호화 하는데 시스템 오버헤드가 크다는 것이었다. 본 논문에서 제안하는 시스템은 데이터를 재암호화할 필요 없이 접근제어목록에서 사용자 이름을 삭제만 하면 된다. 왜냐하면 사용자는 권한이 취소 전이나 후나 암호키를 알 수 없기 때문이다. 기존 시스템의 또 다른 문제점인 데이터 공유문제는 접근제어목록을 사용함으로써 해결된다.

기존 시스템은 별도의 침입탐지 시스템을 두어야 했으나 안전한 데이터 관리 시스템은 별도의 침입탐지 시스템을 둘 필요 없이 키가 요구된 데이터와 및 실행 되고 있는 데이터 정보를 비교함으로써 쉽게 침입 탐지를 수행한다.

4. 결론 및 계획

본 논문에서 제안한 안전한 데이터 관리 시스템은 데이터를 암·복호화하는데 사용되는 키를 사용자가 모르게 시스템 레벨에서만 관리함으로써 기존 시스템의 내부자에 의한 데이터 유출 문제. 공유문제 등을 해결하였다. 그리고 추가적으로 침입탐지 기능까지 수행하는데, 여기에서 알 수 있듯 이 시스템은 기업이나 국가 기관에 적합할 것이다. 그러나 침입탐지는 운영체제와 많은 관련이 있는데 이 부분에 미약한 점이 많다. 따라서 앞으로 이 부분에 대해서 연구 발전시킬 것이다.

참고문헌

[1] M. Blaze, “A cryptographic file system for UNIX”, Proceedings of 1st ACM Conference on Communications and Computing Security, 1993

[2] J. Hughes and D. Corcoran, “A universal access, smart-card-based, secure file system”, Atlanta Linux Showcase, October 1999

[3] K. Fu, “Group sharing and random access in cryptographic storage file systems”. MIT Masters Thesis, June 1999 <Stor-

[4] H. Gobioff, “Security for high performance commodity Subsystem”, Ph.d. Thesis CMU-CS-99-160, July 1999

[5] Erik Riedel, Mahesh Kallahalla, and Ram Swaminathan, “A framework for evaluating storage system security”, Proceedings of 1st ACM Conference on File and Storage Technologies. January 2002

[6] G. Cattaneo and G. Persiano. “Design and Implementation of a Tansparent Cryptographic File System for Unix”, Unpublished Technical Report. Dip. Informatica ed Appl, Universita di Salerno, 8 July 1997

[7] E. Zadok, I. Badulescu, and A. Shender, “Cryptfs: A Stackable Vnode Level Encryption File System”, Technical Report CUCS-021-98. Computer Science Department, Columbia University, July 1998