제19회 한국정보처리학회 춘계학술발표대회 논문집 제10권 제1호 (2003. 5)

리눅스 기반에서의 이중 패스워드 설계

우연옥*, 김행욱*, 강홍식**
*인제대학교 전산학과
**인제대학교 컴퓨터공학과
e-mail:poppi99@nate.com, hukim@nice.inje.ac.kr
hskang@nice.inje.ac.kr

Design of the Second Password Receiver System in Linux

Yeaon-Ok Woo*, Haeng-Uk Kim*,
Heung-Seek Kang**
*Dept of Computer Science, Inje University
**Dept of Computer Engineering, Inje University

요 약

시스템에 대한 보안은 네트워크에 연결되어진 대부분의 시스템이라면 한번쯤은 고려 해봐야 할 것이다. 시스템에 대한 보안이 필요한 가장 큰 이유는 침입으로부터 시스템과 사용자들의 정보를 보호하기 위한 것이다. 특히 일반 사용자들의 계정에 대한 패스워드는 크래킹 당할 위험요소를 많이 가지고 있으므로 이에 대한 보안을 철저히 해야한다. 본 논문에서 제안하는 시스템은 사용자들의 계정에 대하여 임의의 사용자가 로그인하였을 경우 또 다른 패스워드, 즉 SP(Second Password)를 로그인한 사용자로부터 얻어와 재확인 과정을 거쳐 해커가 일반 사용자의 계정으로 쉘을 획득하지 못하도록 하여 사용자 계정과 시스템의 정보를 보호하도록 설계하였다.

1. 서론

하나의 운영체제를 사용하고 있는 각 시스템들은 보다 강력하고 효율적인 시스템 보안 프로그램을 요구하며, 향후 사용이 급증할 것으로 생각되어지는 특정 운영체제에 대해서는 그 필요성이 절실히 요구되고 있다. 특히 리눅스의 경우 다중 사용자모드를 기본으로 하며, 네트워크에 연결되어진 서버의 형식으로 많이 사용되어 지는데 이 같은 경우 앞에서 언급했던, 보다 강력하고 효율적인 보안 프로그램의 필요성이 더욱 증가하고 있다. 보안의 여러 가지 형태 중 가장 본질적이고 효과적인 방법은 실제 계정의 주인과 시스템의 자원을 사용하기 위하여 로그인된 사용자가 동일한 인물임을 해당 시스템에서 확인할 수 있어야 하는 것이다. 현재의 대부분의 시스템에서는 단 한번의 로그인 과정을 통해 시스템의 자원 사용을 허락하고 있는데, 인터넷 뱅킹과 VPN(가상사설망)이 대중화되면서 이중 인증장치에 대한 관심도 높아지고 있다. 이중으로 신원을 확인하는 프로그램에 대한 개발이 이루어진 예를 들면, 드림시큐리티사의 MASTM이 있다. 이것은 이중 신원확인 방식을 적용해 휴대용 저장장치인 USB 토큰을 PC에 연결하고 아이디, 패스워드를 입력하는 두가지 조건을 충족해야 중앙통제에 의해 PC로그인이 가능하고 인증키의 분실, 파손 시에도 중앙에서 키복구 방식으로 대처할 수 있도록 한다. 또 다른 방식으로 사용하는 예는 아이디와 패스워드 외에 계정 소유자가 개인적인 질문을 선택하고 그 질문에 대한 답을 입력하도록 한 후에 패스워드를 분실했을 경우 질문에 대한 답을 알고 있다면 임의로 패스워드를 계정 소유자의 이메일로 발송해주도록 하여 패스워드 재발급을 제한한다. 리눅스 시스템 또한 단 한번의 로그인 과정만 거치면 쉘을 획득할 수 있게되는데, 일반 사용자들의 계정과 패스워드는 쉽게 크래킹 할

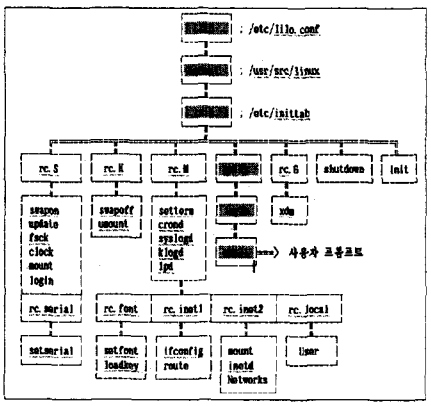
수 있다는 점을 고려한다면 단 한번의 로그인만으로는 시스템의 정보와 시스템을 사용하는 일반 사용자들의 정보가 안전할 수 없다.

따라서 본 논문에서는 계정의 소유자와 해당 계정으로 로그인되어 시스템을 사용하는 인물이 동일함을 다시 한번 확인하기 위하여 SPR(Second Password Receiver) 프로그램을 설계하고자 한다. 본 논문의 2장에서는 현재의 리눅스시스템 로그인 과정과 문제점에 대해서 알아보고, 3장에서는 SPR 시스템에 대한 설계 후 4장에서는 결론 및 향후 과제를 제시한다.

2. 리눅스 시스템의 로그인 과정과 문제점

2.1 리눅스 시스템의 로그인 과정

우선 init는 getty 프로그램을 각각의 터미널(혹은 콘솔)에 실행시킨다. getty는 터미널에서 로그인하려는 사용자가 있는지 살펴보면서 기다린다(즉, 사용자가 뭔가를 타이핑하지 않는지 살펴본다). 사용자가 있다면, getty는 환영 메시지를 출력하고(이 메시지는 /etc/issue에 들어있다) "login:" 같은 프롬프트를 띄운 뒤 마지막으로 로그인 프로그램을 실행시킨다. 로그인 프로그램은 "user-name"을 매개변수로 전달받고, 해당 password를 묻기 위해 "password:" 같은 프롬프트를 띄운다. password가 정확하면, 로그인 프로그램은 설정되어 있는 쉘을 실행시킨다. password가 틀리다면, 로그인 프로그램은 단순히 종료된다(보통은 몇 번 정도 기회를 더 준 뒤에 종료된다). init는 로그인 프로그램이 종료된 것을 감지하고, 터미널에 새로운 getty를 띄워 놓는다. [1]



[그림1] 리눅스 로그인 과정

2.2 로그인 프로그램의 역할

로그인 프로그램은 우선 사용자를 인증하며(즉, user-name과 password가 맞는지 확인한다), 시리얼 라인에 퍼미션을 주고 쉘을 시작시켜 사용자의 초기 환경을 만들어 준다. 또한 초기 설정의 일부로서, /etc/motd('오늘의 메시지' 같은 짧은 정보를 넣어둔다)의 내용을 화면에 뿌려주며 또한 전자 우편이 도착하였는지를 확인시켜준다. 만일 이런 것들을 보고 싶지 않다면, 사용자의 홈 디렉토리에 .hushlogin 파일을 만들어 두면 된다.

로그인 프로그램은 모든 실패한 로그인에 대한 기록을 시스템 로그 파일에 기록하여 둔다(이 일은 syslog를 통해서 이루어진다). 현재 로그인해 있는 사용자는 /var/run/utmp에 나열되어 있다. 이 내용은 시스템이 부팅될 때 지워지므로, 단지 시스템이 가동 중일 때만 유효하다. 이 파일에는 현재 로그인한 사용자의 이름과 사용중인 터미널 등의 정보가 수록되어 있는데, who나 w 같은 명령들이 바로 이 utmp 파일을 들여다보고 누가 로그인해 있는지 알아낸다. 모든 성공적인 로그인은 /var/log/wtmp에 기록된다. 이 파일은 끝없이 크기가 커지므로 주기적으로 그 내용을 지워 주어야 하는데, 예를 들면 cron을 사용해서 일주일에 한번 정도 지워주는 것이 좋다. [1]

2.3 리눅스 시스템의 로그인 문제점

일반적으로 해킹은 루트의 패스워드를 직접 알아내서 시스템에 침입하는 방법보다는 패스워드의 보안에 상대적으로 관심을 갖고 있지 않은 일반 사용자의 계정을 도용해서 시스템에 침입한 후에 su명령어를 직접 이용하거나 su와 SUID로 구성된 스크립트를 이용해서 루트의 권한을 획득하는 방법을 사용한다.

따라서 일반 사용자의 계정과 패스워드에 대한 철저한 보안이 필요할 것이다. 대부분의 리눅스에서는 DES(Data Encryption Standard : 데이터 암호화 표준)라고 하는 단방향 암호화 연산법(One-Way Encryption Algorithm)을 사용해서 패스워드를 암호화한다. 이렇게 암호화된 패스워드는 /etc/passwd 또는 /etc/shadow(쉐도우 패스워드를 사용하는 경우)에 저장된다. 사용자가 로그인 할 때 입력한 패스워드는 먼저 암호처리가 된 후에, 이 처리된 값이 다시 passwd 문서에 저장되어 있는 패스워드 처리 값과 비교가 되게끔 되어 있다. 둘이 일치하면 같은

패스워드임이 분명하므로 엑세스가 허가된다. 그러나 사용자들의 패스워드가 암호화되었다고 해서 완전한 보안이 이루어진 것은 아니다. 어떠한 이유로 인하여 시스템의 계정을 소유하고 있는 사용자들의 일부가 자신이 소유한 계정의 패스워드를 크랙킹 당했다고 가정한다면 해당 시스템의 루트 권한마저 위험에 처해지기 때문이다. [2]

3. SPR 프로그램의 설계

3.1 SPR의 설계 필요성

SPR의 기본취지는 이중으로 된 패스워드를 통하여 이미 로그인한 사용자에 대해서 실제 사용자와 계정의 주인이 동일한지에 대한 여부를 재확인함으로 해커의 침입을 차단하는데 있다. 그렇기 때문에 두 번째 패스워드(SP)는 반드시 계정 소유자의 신상에 관련된 정보를 이용하는 것을 요구하며, 그 이유는 해커가 임의의 계정 소유자에 대한 신상정보까지 획득하기는 매우 힘들기 때문이다. 효과적인 SP의 예는 계정 소유자의 주민등록번호 뒷자리 또는 핸드폰 번호의 일부분 등일 것이다. 따라서 SPR의 디폴트 SP값은 계정 소유자만이 알 수 있도록 계정 소유자의 신상정보에 대한 값으로 한다.

3.2 SPR을 위한 기본 지식

3.2.1 wall

상대방의 터미널로 간단한 메시지를 보내는 기능을 가진 명령어이며, 시스템에 로그인해 있는 모든 사용자에게 메시지를 전달한다. wall 명령어는 루트 사용자가 시스템을 종료시키기 전에 모든 사용자들에게 메시지를 전달하는 브로드캐스트 기능으로 많이 사용되며 일반 사용자들이 wall 명령어를 사용할 수 없도록 설정하고 있다. 루트 사용자가 wall을 실행했을 때에는 mesg의 설정에 관계없이 모든 터미널에 메시지가 출력된다.

3.2.2 cron

cron은 특정한 시간에 명령어에 대해 반복 실행을 제어할 수 있는 기능을 제공한다. 따라서 파일시스템에 대한 점검이나 시스템파일의 분석과 같이 정기적으로 실행되어야 하는 작업은 cron으로 설정해 놓으면, 매번 동일한 작업을 실행할 필요 없이 반복적인 작업을 보다 효율적으로 처리 할 수 있게된다. crond는 cron을 위한 데몬이다. [3]

3.3 SPR의 구성요소

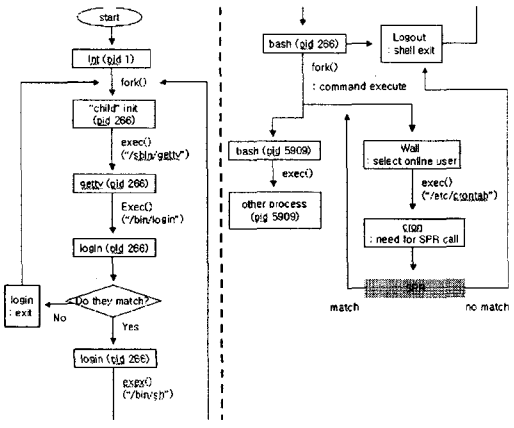
3.3.1 SP의 패스워드 테이블

SP의 기본 성격은 리눅스의 쉐도우 패스워드 성격을 동일하게 지닌다. 그러나 리눅스의 패스워드 테이블과는 물리적으로 다른 또 하나의 패스워드 테이블을 가진 파일로 구성된다. SP의 패스워드 테이블은 /usr/계정명/spr에 저장된다. 각 계정에 대해 SP의 패스워드 테이블이 필요하므로 이를 각 사용자의 홈 디렉토리 아래에 두도록 한다. SP의 패스워드 테이블 구조는 리눅스 사용자의 아이디, 패스워드, SP 세 개의 필드로 구성된다. 후에 사용자의 아이디와 패스워드뿐만 아니라 SP까지 비교하게 된다.

3.4 SPR 호출 단계

SPR의 호출을 위해서는 한번의 wall과 cron을 사용한다. wall의 사용은 로그인이 끝난 후 사용자가 쉘을 획득한 직후에 시스템에 의해서 호출되며, 사용자의 컴퓨터 시스템 수행능력 차이로 나타나는 쉘 획득 시간의 차이를 시스템에서 확인하기 위해 사용한다. 또한 wall의 성격이 지니고 있는 온라인 상태인 사용자를 확인하기 위해서도 사용된다.

cron의 사용시간은 wall 프로세스가 시작된 다음 특정시간 경과 후이며, wall 프로세스의 시작 시간은 프로세스의 상태를 나타내는 ps 명령의 U옵션에서 얻을 수 있다. cron을 사용하는 또 다른 이유는 SPR의 호출을 위해서이다. SPR의 호출시점은 사용자가 쉘을 획득하고 특정시간이 지나서이며, 이 특정시간은 cron의 설정파일인 crondtab에서 원하는 시간에 설정이 가능하다. [4]



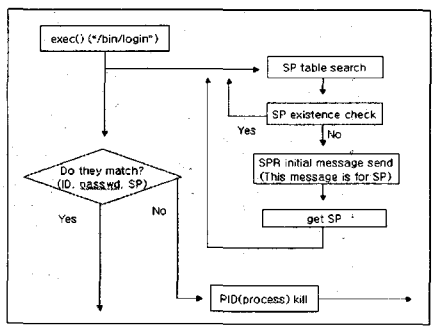
[그림2] SPR의 호출 시점

3.5 SPR의 알고리즘

SPR의 알고리즘은 간단하지만, 전체적인 시스템의 보안에 대해서는 더욱 향상된 성능을 기대할 수 있다. SPR에서 수행하는 알고리즘은 두 가지 경우로 나눌 수 있다.

첫 번째로 SP의 패스워드 테이블에 입력되어져 있는 SP의 값과 사용자가 입력한 SP를 비교하는 것이다. 로그인한 사용자가 SP를 소유하고 있다면 사용자의 아이디와 SP의 패스워드 테이블의 아이디 필드, 사용자가 입력한 SP와 SP의 패스워드 테이블의 SP필드가 동일한지에 대해서 비교를 하게된다. 만약 로그인한 사용자가 SP를 소유하지 않고 있다면 SPR에서는 SP를 얻기 위해 메시지를 사용자에게 보내게 된다. 사용자는 시스템에서 보낸 메시지에 대한 답으로 원하는 SP를 입력한다. SP의 디폴트 값은 위에도 언급했듯이 계정 소유자의 신상에 대한 것으로 한다. 이렇게 얻게 된 SP는 앞서와 같은 방법으로 비교한다.

SPR의 두 번째 알고리즘은 SPR의 핵심 기능으로 사용자가 입력한 SP와 SP의 패스워드 테이블의 SP값이 동일하지 않을 경우 사용자의 PID를 kill 하게 되는 부분이다. 로그인되어진 사용자가 입력한 SP값과 SP의 패스워드 테이블의 SP값이 다를 경우는 현재 계정을 사용하고 있는 사용자가 실제 계정 소유주가 아님을 뜻하므로 쉘의 PID와 loged의 PID를 kill하여 더 이상 시스템을 사용할 수 없도록 한다. 이 기능을 통해서 실제 온라인 되어져 있는 사용자가 계정 소유자인지 혹은 해커인지에 대한 필터링을 하게된다.



[그림 3] SPR의 구성도

4. 결론 및 향후 과제

본 논문에서는 기존의 리눅스 시스템 사용 시에 발생할 수 있는 침입을 보다 효과적으로 필터링하며, 사용자의 계정과 시스템의 정보보호를 위한 SPR이라는 프로그램을 설계하였다. SPR은 단 한번의 SP를 사용자에게 입력하게 함으로써 보다 확실하고 정확하게 침입을 방지를 할 수 있으며, 기존의 리눅스 시스템에서는 볼 수 없는 이중패스워드 방식을 취하고 있다. 하지만 crond의 자동 실행과 wall 프로세스가 끝남과 동시에 cron이 실행되어야 하는데 이에 대한 crondtab의 시간 설정에 대해서는 향후 수행과제로 남겨 두었다.

참고문헌

[1]http://kldp.org/Translations/html/SysAdminGuide-KLDP/log-in-and-out.html

[2]http://kldp.org/HOWTO/html/Security/Security-HOWTO-6.html

[3]송창훈 저 "레드햇 리눅스 완벽가이드" 사이버출판사 1999.

[4]http://lachesis.pe.kr/etc/bootprocess/bootprocess.html

[5]전승협 역 "리눅스 보안의 모든 것" 인포북 2000

[6]Maurice J. Bach. "The design of the UNIX operating system" Prentice-Hall c1986.

[7]AGETTY(8), INIT(8), INITTAB(5), Linux MAM PAGE.

[8]StevensRichard "Advanced Programming in UNIX environment" Addison Wesley.