리눅스에서 사전공격방지를 위한 이중 암호 인증 시스템

최종혁, 허기택, 주낙근
동신대학교 컴퓨터학과

Double Encryption Authentication System for Resisting Dictionary Attack in Linux

Jong-Hyuk Choi, Gi-Teak Hur, Nak-Keun Joo
Dept. of Computer Science, DongShin University
E-mail : bigdream@kornet.net, gthur@blue.dongshinu.ac.kr, nkjoo@blue.dongshinu.ac.kr

요 약

기존의 리눅스 패스워드 인증 시스템에서는 사용자의 계정을 만들어 처음으로 로그인하여 패스워드를 생성할 때에는 사용자의 프로세스 번호와 패스워드를 생성한 시각에 해당하는 값을 seed로 하여 난수를 발생시켜서 salt값을 만든다. 이 salt값은 사용자가 서로 같은 암호를 사용할 경우 암호가 같은 값으로 저장되는걸 막기 위해서 사용하는데 시스템은 salt값과 사용자 패스워드를 단방향 DES 알고리즘을 사용하여 패스워드 파일을 암호화하여 저장한다. 그러나 패스워드 파일에 사용자암호는 암호화되어 저장되지만, salt값이 그대로 저장되기 때문에 패스워드 파일을 가져가게 된다면 사전공격 해킹툴인 John-the-ripper나 Crack 프로그램 등을 이용하여 쉬운 패스워드는 공격자에 의해 간단하게 풀려버린다. 이러한 사전공격에 대한 취약점을 해결하기 위해 암호화된 사용자 패스워드들을 시스템의 또다른 비밀키를 사용하여 암호화하는 방법을 도입함으로써 사전공격에 강한 패스워드 인증 시스템을 설계 및 구현한다.

1. 서론

세계의 컴퓨터들이 하나로 연결되는 인터넷의 물결 속에서 정보의 공유가 점차 확대되고 컴퓨터 이용에 의한 많은 역기능들이 컴퓨터 범죄, 바이러스와 같은 형태로 사회전반에 많은 부작용을 낳고 있다. 사용자 인증 기능은 인터넷과 같은 전산망에서 컴퓨터 보안의 위협을 예방하는 차원에서 합법적인 사용자만 컴퓨터 시스템에 접근할 수 있게 해준다. 사용자 인증을 위해 사용되는 방법으로는 사용자 고유의 물리적인 특성.즉 음성, 지문, 그리고 홍채의 혈관 구조 등을 이용하여 개인의 신원을 확인하거나 패스워드와 같이 사용자가 알고 있는 정보를 이용하는 방법과 사용자가 소지하고 있는 메모리 카드나 스마트 카드와 같은 것을 이용하는 방법이 있다.

본 연구는 정보통신부 지원 대학 S/W 연구센터의 지원에 의해 연구되었음

이 중 가장 널리 사용되는 방법은 패스워드를 사용한 인증방법이다. 특히 리눅스 기반의 시스템에서는 전통적인 사용자 아이디와 패스워드를 사용한 인증기법을 사용하고 있다. 패스워드를 이용한 개인식별은 그 운영 메커니즘이 매우 단순하고 사용자들도 단지 아이디와 패스워드만을 기억하면 되기 때문에 가장 보편적인 개인식별방식으로 인식되고 있지만 다음과 같은 문제점을 내포하고 있다.

먼저 패스워드에 대한 공격자의 불법적인 도청이고, 다음으로 일반사용자들이 패스워드를 자신이 기억하기 쉬운 유형의 문자열을 선택함으로써 공격자에게 추측을 가능하게 한다는 것이다. 특히 패스워드의 길이가 짧은 경우에는 모든 가능한 문자열을 시도해 보는 사전공격을 이용하여 특정 사용자의 패스워드를 추측해 낼 수도 있다.

기존의 패스워드 인증 시스템에서는 사용자의 패스워드를 생성할 때에 사용자의 프로세스 번호와 패스

워드를 생성한 시각에 해당하는 값을 seed로 하여 난수를 발생시켜서 salt라고 하는 값을 만들고, 이 값과 사용자 패스워드를 단방향 DES 알고리즘을 사용하여 암호화한 다음 저장한다. 그러나 패스워드 파일에 사용자암호는 암호화되어 저장되지만, salt값이 그대로 저장되기 때문에 패스워드 파일을 해킹해서 가져가게 된다면 사전공격 해킹툴인 John-the-ripper나 Crack 프로그램 등을 이용하여 쉬운 패스워드는 공격자에 의해 간단하게 풀려버린다[1].

따라서 본 논문에서는 이러한 사전공격에 강한 패스워드 인증 시스템을 제안한다. 본 논문의 구성은 다음과 같다. 먼저, 2장 관련연구에서는 기존의 패스워드 인증 시스템에 대한 개념과 취약점 그리고 암호 알고리즘에 대해서 기술하고, 3장에서는 본 논문에서 제안한 이중암호 시스템을 설계 및 구현하였다. 그리고 4장에서는 제안한 시스템의 실험 및 결과를 기술하고 마지막으로 5장에서는 결론을 제시한다.

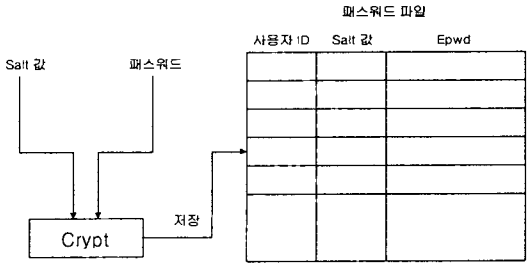
2. 관련연구

2.1 기존의 패스워드 인증 시스템

현재 가장 널리 이용되는 운영체제중의 하나인 리눅스 시스템에서 각 이용자의 패스워드는 DES를 사용해서 만들어진 단방향성 함수로 암호화되어 패스워드 파일에 저장되어진다. 엄밀히 말하면 리눅스 시스템에 로그인하기 위해 사용되는 패스워드가 암호화되는 것이 아니라 64 개의 '0'을 암호화하기 위해 패스워드를 키로 사용한다. 사용자가 입력한 8개의 문자들이 평문으로서 암호화되어 /etc/shadow 파일에 저장되는 것처럼 보이지만 암호화의 비밀키로 사용되는 것이다.

하지만 어떤 두 사용자가 동일한 패스워드를 가지면 동일한 암호가 /etc/shadow 파일에 저장이 된다. 이에 대한 보완책으로 사용자의 계정을 만들어 처음으로 로그인하여 패스워드를 생성할 때에는 사용자의 프로세스 번호와 패스워드를 생성한 시각에 해당하는 값을 seed로 하여 난수를 발생시켜서 salt라고 하는 값을 만든다. 그리고 이 값을 사용자가 입력한 패스워드와 함께 암호화함으로 임의의 두 사용자가 우연히 동일한 패스워드를 선정했다고 할지라도 패스워드 파일에는 서로 다른 값으로 나타나게 된다.

리눅스 시스템에서 로그인하기 위해 사용되는 패스워드 파일 구성은 [그림 1]과 같다.



[그림 1] 패스워드 파일 구성도

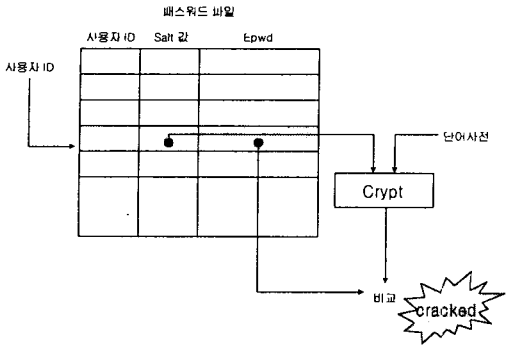
2.2 사전공격

패스워드의 생성과정과 패스워드를 이용한 인증과정을 흉내내서 사용자 패스워드를 알아내는 소프트웨어가 있다. 현재까지는 DES 알고리즘은 역연산(reverse operation)을 통해 복호화할 수 없다고 알려져 있기 때문에 암호화된 패스워드와 DES 알고리즘만을 갖고 패스워드를 알아낼 수는 없다. 따라서 Crack 프로그램에서는 소위 말하는 "사전공격"이라는 방법을 사용한다[5].

Crack을 구동시키기 위해서는 사전 파일이 필요한데 이 파일은 인터넷에서 Crack 프로그램과 함께 손쉽게 얻을 수 있다. 이러한 사전파일에는 사전에 있는 수 만개(2만 5천개 정도)의 단어들이 한 줄에 하나씩 기록되어 있어 Crack 프로그램의 입력 값으로 사용된다.

Crack 프로그램을 실행시키면 /etc/shadow 파일을 읽어서 그 파일에 등록된 모든 사용자들의 password entry를 만들고 각 사용자의 salt를 읽어 들인 다음 사전 파일에 있는 단어들을 하나하나 대입하여 앞에서 설명한 암호화 과정을 거친다.

그런 다음 생성된 암호화된 패스워드를 /etc/shadow 파일에서 얻은 암호화된 패스워드와 비교하여 이것이 같으면 입력했던 단어를 그 사용자의 패스워드로 간주한다. Crack 프로그램은 단순히 패스워드 유추작업을 자동화한 것에 지나지 않지만 사전파일의 내용을 만드는 사람의 노력여하에 따라서 굉장한 성과를 얻을 수 있는 해킹툴이 될 수 있다.



[그림 2] 패스워드 사전 공격

2.3 DES 알고리즘

DES(data encryption standard)는 1974년 컴퓨터 보안의 필요성에 의해 제안된 미 연방 정보처리 표준 46(FIPS PUB46)으로 채택된 대칭키 암호 알고리즘이다[2][3]. 현재 ISO의 표준(DEA-1)으로 제정되어 있으며, 지난 20년동안 세계적인 표준으로 사용되어 왔다[4].

DES는 64비트 블록 암호 시스템으로 64비트 평문 입력을 통해 64비트 암호문을 출력한다. 이때 사용되는 64비트의 키는 안전성을 위해서 128비트로 변환하여 사용할 수 있고 키값의 일부는 검사용으로 사용된다. 대부분의 대칭키 알고리즘은 라운드 방식을 통해 같은 서브알고리즘을 반복하는 암호화 과정을 이용하며 DES는 한 입력 블록에 대해 총 16 라운드의 암호문을 반복한다. 이때 각 라운드는 치환과 전치 및 대치의 과정을 거치면서 평문과 키에서 나온 48비트의 키가 섞여 암호문을 만든다. 복호화는 키를 암호화의 역순으로 적용시키기 때문에 복호화를 위한 알고리즘은 동일하다.

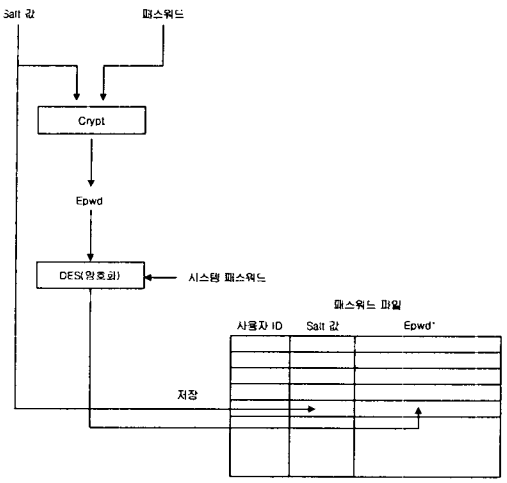
3. 이중 암호 인증 시스템

리눅스 시스템은 패스워드가 shadow 파일에 저장될 때 salt값이 암호화되지 않은 상태로 패스워드 파일에 저장이 되고, 또한 사용자들이 패스워드에 대한 보안의식이 약하기 때문에 패스워드를 입력하기 좋은 숫자들의 조합이나 간단한 단어를 사용함으로써 쉽게 보안상으로 노출되어 진다. 패스워드가 저장된 파일을 공격자가 가져간다면, 쉬운 패스워드들은 사전공격을 통해 쉽게 해독되어지는 취약성을 보이고 있다.

이러한 취약점을 보완하기 위해서 본 논문에서 제안하는 방법은 리눅스에서 shadow 파일에 암호화된 사용자의 패스워드를 저장할때 시스템의 패스워드가 키로 사용되어 다시 한번 DES로 암호화함으로써 사전공격에 강한 새로운 패스워드 인증 시스템을 제안한다.

3.1 패스워드 생성

사용자의 계정을 만들고 그 패스워드를 생성할 때 기존의 salt값 생성 방법인 사용자의 프로세스 번호와 패스워드를 생성한 시각에 해당하는 값을 seed로 하여 난수를 발생시켜서 salt값을 생성하고, 그 값과 함께 기존 단방향 DES로 암호화한 패스워드를 시스템의 패스워드와 DES를 통해 다시 암호화한다. 이중 암호화된 파일은 salt값과 함께 패스워드 파일에 저장된다.



[그림 3] 패스워드 생성

3.2 시스템 패스워드 생성

사전공격에 강한 이중 암호 인증 시스템을 구현하기 위해서는 시스템 관리자만이 알고 있는 시스템 패스워드를 생성해야 하는데, 본 논문에서는 salt값 생성하는 것과 유사하게 그 시스템 패스워드를 생성할 때 사용자의 프로세스 아이디 번호와 패스워드를 생성한 시각에 해당하는 값을 seed로 하여 난수를 발생하여 시스템 패스워드로 사용된다. 이 값은 기존의 암

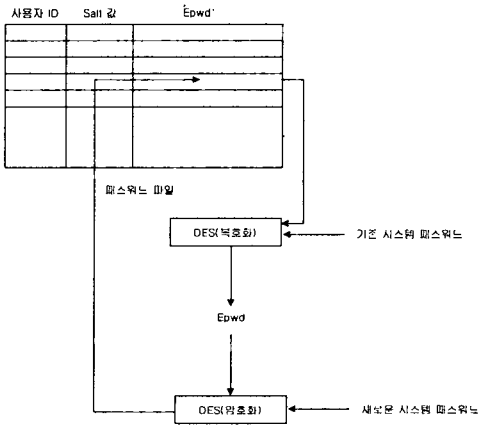
호 패스워드를 이중 암호화할 때 키 값으로 이용된다.

관리자는 시스템 패스워드를 주기적으로 또는 필요시 언제든지 변경할 수 있다. 시스템 패스워드가 변경될 경우 패스워드 파일에 저장된 사용자 패스워드들은 다시 복호화되어 새로운 시스템 패스워드에 맞게 암호화한 다음 패스워드 파일에 저장된다.

3.3 패스워드 파일 변경

시스템 패스워드는 자동으로 일정주기가 되면 변경되거나 또는 필요에 따라 관리자에 의해 변경된다. 이때 패스워드 파일도 새로운 시스템 패스워드에 해당하는 암호값으로 변경해야만 한다.

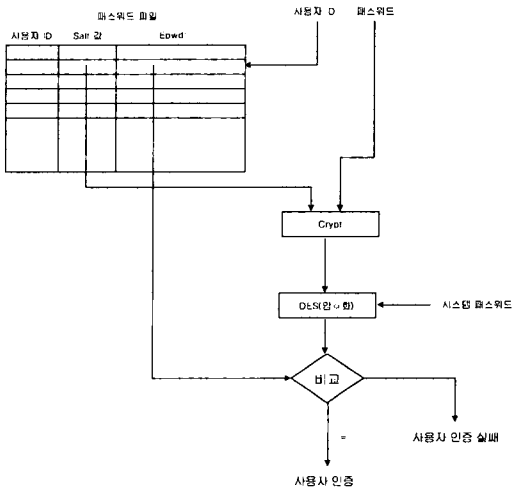
기존의 시스템 패스워드를 이용해 생성된 패스워드 파일내에 있는 패스워드들을 복호화 하여 새로운 시스템 패스워드 값과 다시 암호화한 다음 패스워드 파일에 저장한다.



[그림 4] 패스워드 파일 변경

3.4 사용자 인증

본 논문에서 제안한 시스템에서 사용자 인증은 사용자 입장에서는 인증을 위해서 기존의 패스워드만 입력하지만 시스템에서는 사용자 패스워드와 시스템 패스워드를 같이 사용하여 암호화한 다음 패스워드 파일과 비교해서 같으면 인증에 성공하게 되고, 다르면 인증에 실패하게 된다.



[그림 5] 사용자 인증

4. 실험 및 결과

기존의 패스워드 파일을 사전공격 해킹툴인 John-the-ripper를 실행하면 [그림 6]과 같다. 여기서 "gon"이라는 사용자의 패스워드가 "love"라는 것을 쉽게 알 수 있다.

```
# ./john -wordfile:./password.lst  -rules shadow
Loaded 3 passwords with 3 different salts (FreeBSD MD5 [32/32])
guesses: 0  time: 0:00:00:39 0% (2)  c/s: 524  trying: McCloy
guesses: 0  time: 0:00:00:40 0% (2)  c/s: 537  trying: NixPub
guesses: 0  time: 0:00:00:48 0% (2)  c/s: 532  trying: VHK-f
guesses: 0  time: 0:00:00:50 0% (2)  c/s: 531  trying: XX-Tag
guesses: 0  time: 0:00:00:52 0% (2)  c/s: 531  trying: abater
guesses: 0  time: 0:00:00:54 0% (2)  c/s: 530  trying: accute
-------------
love             (gon)
guesses: 0  time: 0:00:00:56 0% (2)  c/s: 529  trying: adapis
guesses: 0  time: 0:00:00:58 0% (2)  c/s: 529  trying: aedile
guesses: 0  time: 0:00:01:00 0% (2)  c/s: 528  trying: agname
--------
```

[그림 6] 패스워드 파일을 사전공격 한 화면

본 논문에서 제안한 시스템의 패스워드 파일을 동일한 사전공격 해킹 툴인 John-the-ripper를 사용하여 해킹해 본 결과 "gon"이라는 사용자의 패스워드 "love"를 발견하지 못함을 알 수 있다.

```
# ./john -wordfile:./password.lst  -rules shadow
Loaded 3 passwords with 3 different salts (FreeBSD MD5 [32/32])
guesses: 0  time: 0:00:00:39 0% (2)  c/s: 524  trying: McCloy
guesses: 0  time: 0:00:00:40 0% (2)  c/s: 537  trying: NixPub
guesses: 0  time: 0:00:00:48 0% (2)  c/s: 532  trying: VIHK-E
guesses: 0  time: 0:00:00:50 0% (2)  c/s: 531  trying: XX-Tag
--------
guesses: 0  time: 0:00:01:00 0% (2)  c/s: 528  trying: agname
guesses: 0  time: 0:00:00:56 0% (2)  c/s: 529  trying: adapis
guesses: 0  time: 0:00:00:58 0% (2)  c/s: 529  trying: aedile
guesses: 0  time: 0:00:01:00 0% (2)  c/s: 528  trying: agname
--------
```

[그림 7] 본 논문에서 제안한 시스템의
패스워드 파일을 해킹한 화면

5. 결론

본 논문에서는 기존의 패스워드 시스템이 사전공격에 취약한 점을 해결하기 위해서 새로운 인증 시스템을 제안했다. 제안한 패스워드 인증 시스템을 리눅스에서 사용할 경우 패스워드 파일이 공격자에게 노출이 된다고 하더라고 시스템 패스워드 파일이 노출되지 않는 이상 사전공격으로 인해서 패스워드가 해독되어지는 것은 불가능하다.

리눅스에서 텔넷 서비스를 사용하지 않고 SSH를 사용해 스니퍼 공격을 막고, 본 논문에서 제안한 이중 패스워드 인증시스템을 사용한다면 사전공격에 강한 시스템이 구성되어 시스템이 보다 더 안전하게 사용자 인증을 할 수 있게 된다.

향후 연구 과제로는 패스워드 변경시 시간을 효율적으로 아끼는 방법과 시스템 패스워드의 효율적인 생성과 관리 등에 대한 연구가 필요하다.

[참고문헌]

[1] http://cnscenter.future.co.kr.

[2] ANSI X3.92, "American National Standard for Data Encryption Algorithm(DEA)", American National Standard Institute, 1981.

[3] National Bureau of Standards(U.S.), "Data Encryption standard", Federal Information Processing Standards Publication 46, National Technical Information Services, Springfield, VA(1977).

[4] 박희운, 이임영, "암호기술", 정보처리 제7권 제2호, pp.7~19, 2000. 3.

[5] ETRI 부설 국가보안기술연구소, 현대암호학, 경문사, 2000.