스마트 카드와 PAM을 이용한 사용자 인증

강민정* · 강민수* · 박연식**

*경상대학교 · **경상대학교 해양산업연구소

User Authentication by using SMART CARD and PAM

Min-Jung Kang* · Min-Su Kang* · Yeoun-Sik Park**

*National GyeongSang University

E-mail : lundi@dittotec.com

요 약

인터넷 사용의 폭발적인 증가로 대부분의 인터넷 서비스에서 클라이언트와 서버의 인증이 필수적인 요소가 되었다. 유닉스 기반의 서버인 경우에도 기존의 crypt 함수를 이용한 패스워드 인증방법에서 MD5로 암호화한 쉐도우 패스워드(Shadow Password)를 사용함으로써 사용자 인증의 보안성을 한층 강화시켰다. 그러나 쉐도우 패스워드를 사용한다 하더라도 모든 서비스마다 패스워드에 의존한 동일한 인증방식을 사용할 수 밖에 없다는 단점을 가지고 있다. 그러나 PAM(Pluggable Authentication Module)을 사용하여 융통성 있게 각 응용 프로그램마다 개별적으로 사용자를 인증하는 방식을 설정할 수 있다.

본 논문에서는 스마트 카드를 인증 토큰으로 하여 Linux-PAM을 이용한 사용자 인증 시스템을 설계해보고자 한다.

ABSTRACT

Authentication between Server and Client is necessary in most of Internet Service because of increasing of using of Internet. Unix-based Server upgraded security of user authentication using "Shadow Password" instead of "crypt" function. But "Shadow Password" must use same authentication method about all services. But we individually can set user authentication method using PAM(Pluggable Authentication Module).

This paper will propose user authentication system using Linux-PAM that use SMART CARD as authentication token.

키워드

PAM, 사용자 인증, Shadow Password, 스마트 카드

Ⅰ. 서 론

인터넷의 증가와 더불어 클라이언트와 서버간의 인증은 필수적인 요소가 되었다. 인증하는 방법은 크게 인증서를 사용하는 방법과 패스워드 등을 암기하는 지식기반 방법, 그리고 최근 큰 관심의 대상이 되고 있는 사람의 신체적인 특징을 이용하는 방법, 마지막으로 스마트카드 등과 같은 하드웨어 장치를 이용하는 방법으로 나눌 수 있다[1]. 일반적으로 사용자의 아이디와 패스워드를 이용한 로그인(log in) 과정을 통한 지식기반 인증방법이 많이 이용되어지고 있다.

유닉스 기반의 서버들은 이러한 전통적인 인증방법에서 가장 중요한 정보라 할 수 있는 패스워드의 보안성을 위하여 기존의 crypt 함수 대신 MD5를 이용한 쉐도우 패스워드를 이용하고 있지만, 여전히 하나의 인증방식으로 시스템 전체의 인증이 이루어진다는 문제점을 안고 있다. 대용량의 분산시스템 혹은 전용 크랙 하드웨어를 이용한 패스워드 해독, 인증 시스템 공격 혹은 인증 시스템을 우회하는 방법 개발 등을 통한 다양한 공격들의 증가로 미루어 볼 때, 더 이상 이러한 인증방법으로는 안전한 시스템을 유지하기 어렵다[2].

그러나 PAM(Pluggable Authentication Module)은 각 응용 프로그램마다 사용자를 인증하는 방식을 개별적으로 구성할 수 있다. 즉, 로컬시스템에서 기본적인 인증방식을 공통적으로

제공하고 이를 조합하는 방식은 각 응용 프로그램이 개별적으로 설정하여, 하나의 시스템에 여러 개의 인증방법을 이용할 수 있다.

원래 PAM은 시스템 관리자가 응용프로그램들이 사용자를 인증하는 방법을 선택할 수 있도록 해주는 공유 라이브러리들의 묶음으로써, 권한을 부여하는 소프트웨어의 개발과 안전하면서도 적절한 인증방법을 분리하는데 그 목적이 있다[3]. 이는 응용프로그램이 사용자 인증을 처리하기 위해 사용될 함수의 라이브러리를 제공함으로써 가능한데, Linux-PAM 라이브러리는 "/etc/pam.d/xxx "에서 각 시스템에 맞게 설정하여 각 시스템에서 사용가능한 인증모듈을 통해 사용자의 인증요구를 처리한다. 그림1은 이러한 Linux-PAM의 전체 구조를 나타낸 것이다[3].

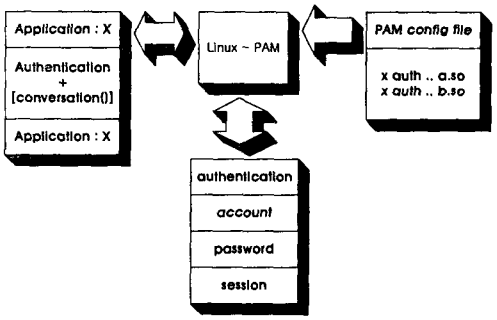


그림 3. Linux-PAM의 구조

II. PAM 설정파일

각 응용프로그램마다 개별적인 사용자 인증방법을 제공할 수 있는 것은 각 응용프로그램에 해당하는 설정파일에서 지정한 모듈들을 통해서 가능해진다. 이러한 모듈들은 "/usr/lib/security"에 위치해 있는데 동적으로 로드 가능한 오브젝트 파일(Object File)의 형태를 가진다.

일반적으로 Linux-PAM 설정파일은 표1과 같은 형식을 가진다.

표 1. Linux-PAM 설정파일 형식과 그 예

module-type	control-flag	module-path	arguments
auth	required	/lib/security/pam_stack.so	service=system-auth

표 1의 1행은 설정파일의 형식이고, 2행은 디폴트로 설정된 SSH 설정파일(/etc/pam.d/sshd)의 내용을 예로 든 것이다.

모듈타입(Module Type)은 표2와 같이 네가지 타입 중 하나이며, 콘트롤 플래그(Control Flag)는 앞에서 지정한 모듈의 성공 또는 실패 결과에 따라 어떻게 PAM 라이브러리가 반응할지를 결정하는데 표3과 같이 네가지 종류를 가진다.

표 2. 모듈 타입(Module Type)의 기능

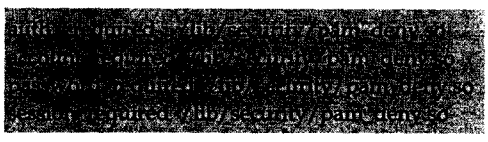
모듈타입	기 능
auth	패스워드 등을 이용하여 정당한 사용자임을 확인
account	현재의 시스템 자원 상황에 따라 서비스 접근을 허용 또는 제한
session	서비스를 사용하기 직전 또는 직후에 필요한 작업
password	사용자와 연관된 인증토큰을 갱신할 때 필요

표 3. 컨트롤 플래그(control flag)의 기능

컨트롤타입	기 능
required	모듈타입이 전체적으로 성공하기 위해서는 이 모듈의 성공이 반드시 필요(나머지 모듈의 실행이 끝나야 이 모듈의 성공여부를 알수 있음)
requisite	required와 비슷하나, 실패 즉시 바로 응용프로그램에게 제어가 넘어감.
sufficient	required 모듈의 실패시 실행되며, 이 모듈의 성공은 해당 모듈타입이 성공했다고 판단할 만큼 충분히 만족스럽다는 것.(이 모듈의 실패는 이 모듈타입의 성공여부 판단에 치명적이지는 않음)
optional	이 모듈이 서비스에 대한 응용프로그램의 성공여부에 중요하지는 않으나, 모든 모듈의 정확한 성공여부를 알 수 없는 경우 이 모듈이 응용프로그램에게 주는 결과를 결정함.

모듈경로(Module Path)는 동적으로 로드될 오브젝트 파일(Object File)의 경로명으로, "/usr/lib/security"가 디폴트로 설정된 모듈경로이다. 아규먼트(arguments)는 모듈이 불려질때 넘겨주는 토큰 리스트로, 유효한 인자들은 특정 모듈에 의해서만 해당되고 이외의 다른 모듈에 의해서는 무시된다.

안전한 시스템 운영을 위하여 보통 OTHER 파일을 두게 되는데, 이 파일은 "/etc/pam.d"에 존재하지 않는 서비스에 대해 공통으로 적용할 수 있는 설정파일로써 디폴트로 다음과 같은 형식으로 설정되어져 있다.



모든 모듈타입에 대하여 "pam_deny.so" 모듈을 사용하고 있는데, 이는 PAM 구조를 통해 응용프로그램에게 항상 실패를 리턴함으로써 접근을 거부하는데 사용된다. OTHER 파일 설정은 엉성하게 설정하게 되면 오히려 보안에 문제를 초래하게 되는 원인이 되므로 다른 설정파일들보다 특별한 신경을 기울여야 한다.

III. PAM 주요 함수들

PAM이 응용프로그램으로부터 호출되어질 때 여러 함수들이 동작한다. PAM API 인증과정에 필요한 중요 함수들은 다음과 같다[4].

1. Linux-PAM 초기화 함수

```
extern int pam_start(const char *service_name,
    const char *user,
    const struct pam_conv *pam_conversation,
    pam_handle_t **pamh);
```

이 함수는 응용프로그램으로부터 처음으로 호출되어져야만 하는 함수이다. 여기서 "service_name"은 특정 서비스 프로그램의 이름을 나타내고, "user"는 인증받을 사용자의 아이디를 의미하며, "pam_conversation"는 응용 프로그램에서 넘어온 pam_conv 구조체에 대한 포인터이며, "pamh"는 PAM 라이브러리에 대한 성공적인 호출을 지속적으로 제공하는 pam_handle_t 구조체에 대한 포인터의 포인터이다.

2. 라이브러리 종료 함수

```
extern int pam_end(pam_handle_t *pamh,
    int pam_status);
```

이 함수는 응용프로그램이 마지막에 반드시 호출해야 하는 함수로서, 호출을 통해 pamh이 더 이상 유효하지 않다는 것을 알린다.

3. PAM 아이템 설정 함수

```
extern int pam_set_item(pam_handle_t *pamh,
    int item_type, const void *item);
```

이 함수는 "item type" 중의 하나의 값을 설정하는데 사용되는 함수다.
아이템은 "PAM_SERVICE", "PAM_USER", "PAM_USER_PROMPT", "PAM_TTY", "PAM_RUSER", "PAM_RHOST", "PAM_CONV", "PAM_FAIL_DELAY" 총 8개의 타입을 가진다.

4. PAM 아이템 데이터 획득 함수

```
extern int pam_get_item(const pam_handle_t
    *pamh, int item_type, const void **item);
```

"item type"이 가리키는 값(데이터)을 가져오는데 사용되는 함수로, 실제적인 데이터에 대한 포인터를 "*item"이 리턴한다.

5. 사용자 인증 함수

```
extern int pam_authenticate(pam_handle_t
    *pamh, int flags);
```

로드되어진 모듈들의 인증 메커니즘에 대한 인터페이스로 제공되어지는 함수이다. "flags"는 사용자가 등록된 인증 토큰을 가지고 있지 않은 경우, "PAM_DISALLOW_NULL_AUTHTOK"를 통해 여러 상황에 적합한 에러 값들을 리턴한다.

6. 사용자 증명서(credentials) 설정

```
extern int pam_setcred(pam_handle_t *pamh,
    int flags);
```

사용자의 특정 모듈 증명서를 설정하기 위하여 사용되는 함수로써, 사용자가 인증되어지고 난 후 그리고 계정(account) 관리가 호출되어지고 난 후 그러나 사용자에 대한 세션(session)이 오픈되기 전에 호출되어진다.

7. 인증 토큰 업데이트 함수

```
extern int pam_chauthtok(pam_handle_t *pamh,
    const int flags);
```

"pamh" 핸들에 의해 지정된 사용자에 대한 인증 토큰을 변경하는 함수로써, "flags"로 "PAM_CHANGE_EXPIRED_AUTHTOK" 아규먼트를 이용하여 이 아규먼트가 실패시 모든 인증 토큰을 변경할 것을 요구한다.

8. 모듈을 구성하는 함수들

위에 소개된 함수들 이외에 PAM 모듈은 반드시 포함해야 하는 함수가 있다. 즉, 모듈은 표4와 같은 여섯 개 함수를 네 가지 모듈 타입에 따라 그룹화하여 제공해야만 하는데, 이들은 Linux-PAM 모듈의 함수를 정의한다[5].

표 4. 모듈이 반드시 포함해야 하는 함수

Authentication management	pam_sm_authenticate()
	pam_sm_setcred()
Account management	pam_sm_acct_mgmt()
Session management	pam_sm_open_session()
	pam_sm_close_session()
Password management	pam_sm_chauthtok()

IV. 스마트카드를 이용한 인증 모듈 작성과 사용자 인증 방법 제안

패스워드가 아닌 스마트카드를 인증 토큰으로 하여 SSH에 접속시 PAM으로 사용자 인증을 한다고 가정하고 작성한 모듈의 일부분을 아래에 나타내었다.

```
#include <security/pam_modules.h>
#include <security/_pam_macros.h>
int pam_sm_authenticate(pam_handle_t *pamh, int
flags, int argc, const char **argv)
{
  int retval=PAM_AUTH_ERR;
  struct stat t2gfileinfo;
  const char *user=NULL;
      ...
 retval=pam_get_user(pamh, &user, NULL);
  if(retval != PAM_SUCCESS)
  {
    _pam_log(LOG_ERR, "get user returned error :
%s", pam_strerror(pamh, retval));
    return retval;
  }
      ...
  outb(0, BASEPORT);
  outb(11, BASEPORT);
  outb(9, BASEPORT);
      ...
  while((fgets(lignefichier,          sizeof(lignefichier)-1,
monfile) != NULL) && retval)
  {
    if(lignefichier[strlen(lignefichier)-1]=='\n')
      lignefichier[strlen(lignefichier)-1]='\0';
    if((strcmp(complet, lignefichier))==0)
      retval=PAM_SUCCESS;
  }
  fclose(monfile);
  if(retval==PAM_SUCCESS)
    _pam_log(LOG_DEBUG, "access allowed for
'%s'", user);
  return retval;
}
int pam_sm_setcred(pam_handle_t *pamh, int flags,
int argc, const char **argv)
{
  return PAM_SUCCESS;
}
```

위에서 작성하여 컴파일한 모듈을 이용하여 "/etc/pam.d/sshd" 설정파일 내용을 다음과 같이 수정하였다.

```
auth       required     /lib/security/pam_
service=system-auth
auth       required     /lib/security/pam_
nologin.so
account    required     /lib/security/pam_marlcard.so
service=system-auth
password   required     /lib/security/pam_marlcard.so
service=system-auth
session    required     /lib/security/pam_marlcard.so
service=system-auth
session    required     /lib/security/pam_limits.so
session    optional     /lib/security/pam_console.so
```

위의 설정은 기본 sshd 설정과 단지 사용하는 인증토큰만 다르고 기타 내용은 동일하다. SSH 설정 파일은 PAM을 이용하는 응용프로그램 중에서 가장 보안에 민감한 파일이므로 다른 설정파일들보다 조건이 까다롭고 파일 내용 확인도 관리자만이 가능하도록 설정되어지는 만큼, 4가지 모듈타입 모두 스마트 카드에 의해 인증이 이루어지고 로그인 과정 없이는 절대로 접속 할 수 없도록 설정했다.

V. 결론 및 향후 연구과제

쉐도우 패스워드를 비롯한 기존의 사용자 인증 방법은 동일한 사용자 인증 방법으로 전체 시스템을 관리해야 한다는 문제점을 안고 있다. 하지만 PAM은 각 응용프로그램마다 원하는 인증방법을 재컴파일할 필요없이 해당 모듈을 호출함으로써 각각 다르게 사용할 수 있다. Linux 배포판 중에서 가장 대표적인 RedHat의 경우 버전 5.0 이후부터 기본적으로 PAM을 지원하고 있기 때문에 앞으로 그 사용률은 더 높아질 것이다. 이에 본 논문에서는 스마트카드를 인증 토큰으로 하여 사용자 인증을 시도하는 모듈을 작성하여, SSH 접속시 이 모듈을 이용하여 사용자 인증을 시도하는 사용자 인증 방법을 제안하고 실제로 테스트해 보았다.

어중간한 안전성을 제공하는 설정파일은 오히려 없는 편이 더 안전할 수 도 있을만큼 PAM 설정파일을 어떻게 설정하느냐가 중요한 사항이다. 본 논문에서 제작한 모듈은 인증 방법을 지식기반 인증 방법에서 단순히 스마트 카드를 이용한 하드웨어 장치를 이용하는 방법으로 변경하여 인증의 과정을 거치도록 작성되었다.

스마트 카드를 통한 사용자 인증이 더 안전하기를 바란다면 앞으로 자바형 스마트 카드를 통한 연구가 계속 이루어져야 할 것이다.

참고문헌

[1] 박윤주, 보안성을 고려한 패스워드기반 인증 컴포넌트의 설계 및 구현, 고려대학교 석사학위논문, 2000

[2] 가디언 세미나 문서, http://panda.snu.ac.kr/windy96/work/security/pam.doc, 2002

[3] Andrew G. Morgan, The Linux-PAM System Administrators' Guide, ttp://www.kernel.org/pub/linux/libs/pam/linux-pam/pam.html, 2001

[4] Andrew G. Morgan, The Linux-PAM Application Developers' Guide, http://www.kernel.org/pub/linux/libs/pam/linux-pam-html/pam_appl..., 2001

[5] Andrew G. Morgan, The Linux-PAM Writers' Guide, http://www.kernel.org/pub/linux/libs/pam/linux-pam-html/pam_modules-3.html