User Authentication

Tom Kelliher, CS 325

Oct. 20, 2006

1 Administrivia

Announcements

Group milestone written statements due today.

Assignment

None.

From Last Time

Outline

1. User authentication.
2. Pluggable Authentication Modules

Coming Up

Exam I.

2 User Authentication

2.1 Basic Authentication Mechanisms

Authentication can be based upon:

1. A secret you know — a password.
2. Something you have — an unforgeable ID.
3. Something you are — a fingerprint.

We can slow down or thwart an attacker by introducing an artificial delay into a failed authentication, or even locking an account after n failed attempts. *This can have consequences.*

"Loose-lipped" systems.

2.2 Passwords

1. The most common authentication mechanism.
2. Attack methods:
   (a) Try all possible passwords — brute force.

   Optimization: Start with shortest passwords.

   (b) Try most likely passwords for a "general" user.

   (c) Try most likely passwords for a specific user.

   This worked for me once.

   (d) Get the system password file.

   (e) Ask the user — social engineering.

3. Password files

    (a) If plaintext, must be hidden.

    (b) If encrypted, are still usually hidden.

        Originally, encrypted passwords were not hidden. This has become a necessity with the arrival of password cracking software.

        Use of a “salt” to disguise two users choosing the same password.

4. Choosing a good password.

5. One-time passwords via challenge-response systems.

    Your secret is an algorithm, rather than a password.

2.3 Authentication Impersonation

Authentication authenticates you to the system. How is the system authenticated to you?

How do you know that your password is being sent to the system, and not collected by a trojan program?

3 Pluggable Authentication Modules

Generally known as PAM.

3.1 The Idea

1. Separate authentication and other management functions from the applications themselves.

2. Provides an authentication *mechanism*.

3. Various policies are easily achievable.

4. Without PAM, policy changes require re-compilation.

3.2 Provided Management Functions

These are provided on an application-by-application basis:

1. auth: Authenticates a user.
2. account: Performs non-authentication-based account management.

   For example, restricting use to a certain period of the day or according to resource availability.
3. password: Associated with password token updates.
4. session: Carries out system functions that may need to be performed before/after a service is made available to a user.

   For example, ensuring that a home directory on a remote disk is mounted and available.

Policy modules can be *stacked*.

3.3 Policy Examples

1. The default policy, used if a specific policy for a PAM-aware application is not available:

   ```
   auth     required       /lib/security/$ISA/pam_deny.so
   account  required       /lib/security/$ISA/pam_deny.so
   password required       /lib/security/$ISA/pam_deny.so
   session  required       /lib/security/$ISA/pam_deny.so
   ```

2. The policy for passwd:

   ```
   auth       required pam_stack.so service=system-auth
   account    required pam_stack.so service=system-auth
   password   required pam_stack.so service=system-auth
   ```

3. The “catch-all” system-auth policy:

```
auth        required      /lib/security/$ISA/pam_env.so
auth        sufficient    /lib/security/$ISA/pam_unix.so likeauth \
                          nullok
auth        required      /lib/security/$ISA/pam_deny.so

account     required      /lib/security/$ISA/pam_unix.so

password    required      /lib/security/$ISA/pam_cracklib.so retry=3 \
                          type=
password    sufficient    /lib/security/$ISA/pam_unix.so nullok \
                          use_authtok md5 shadow
password    required      /lib/security/$ISA/pam_deny.so

session     required      /lib/security/$ISA/pam_limits.so
session     required      /lib/security/$ISA/pam_unix.so
```

4. The policy for su:

```
auth       sufficient   /lib/security/$ISA/pam_rootok.so
# Uncomment the following line to implicitly trust users in the
# "wheel" group.
#auth       sufficient   /lib/security/$ISA/pam_wheel.so trust use_uid
# Uncomment the following line to require a user to be in the
# "wheel" group.
#auth       required     /lib/security/$ISA/pam_wheel.so use_uid
auth       required /lib/security/$ISA/pam_stack.so service=system-auth
account    required /lib/security/$ISA/pam_stack.so service=system-auth
password   required /lib/security/$ISA/pam_stack.so service=system-auth
session    required /lib/security/$ISA/pam_stack.so service=system-auth
session    optional /lib/security/$ISA/pam_xauth.so
```

The first line allows root to su to some other user without entering the root password.

5. The policy for printconf-gui:

```
auth       sufficient pam_rootok.so
auth       sufficient pam_timestamp.so
auth       required pam_stack.so service=system-auth
session    required pam_permit.so
session    optional pam_xauth.so
session    optional pam_timestamp.so
account    required pam_permit.so
```

Note the used of the “cached” previous authentication token on the second line.