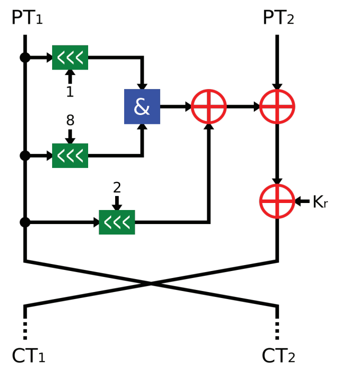
CSE 40567 / 60567: Computer Security



Software Security 2

Homework #3 has been released. It is due 2/18 at 11:59PM

See **Assignments Page** on the course website for details

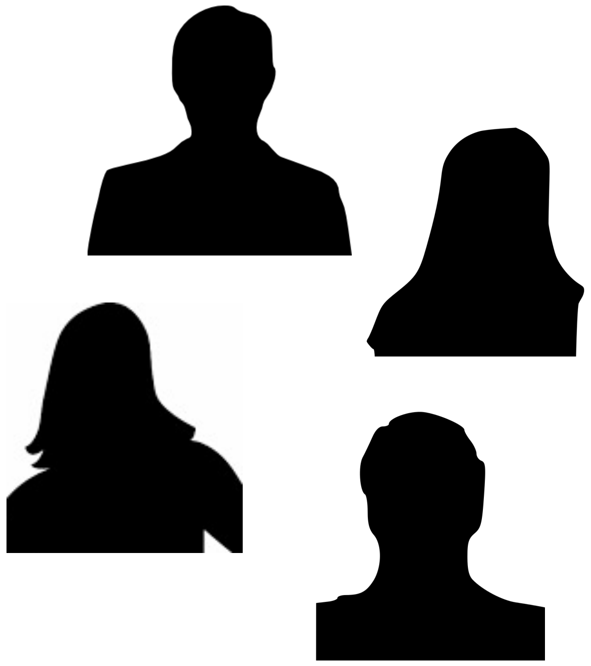
Midterm Exam: 2/27 (In Class)

See Topics Checklist on Course Website

OS Authentication

Users

- A user is an identity tied to a single entity
- Specific systems may add additional constraints
- Systems represent user identity in a number of different ways
- The same system may use different representation of identity in different contexts

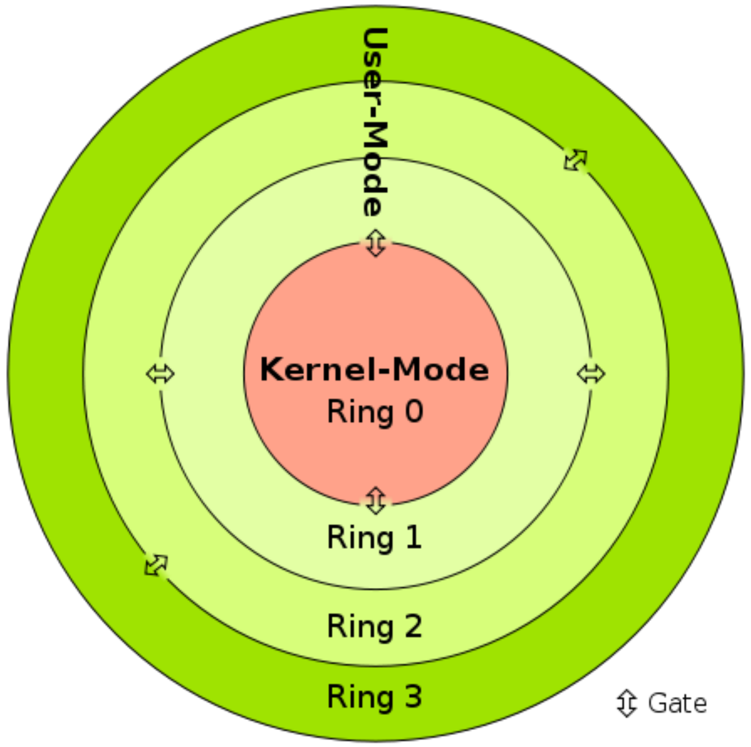


Good design principles for access control

1. Simplicity makes designs and mechanisms easy to understand
2. Simplicity reduces the potential for inconsistencies within a policy or set of policies
3. Restriction minimizes the power of an entity
4. Entities can communicate with other entities only when necessary
5. “Communication” is used in the widest possible sense, including that of imparting information by not communicating

Principle of Least Privilege

The *principle of least privilege* states that a subject should be given only those privileges that it needs in order to complete its task



Example:
userland process

Principle of Separation of Privilege

The *principle of separation of privilege* states that a system should not grant permission based on a single condition.

Example: Ubuntu Linux privilege escalation; user must be in group sudo to use `sudo(8)`

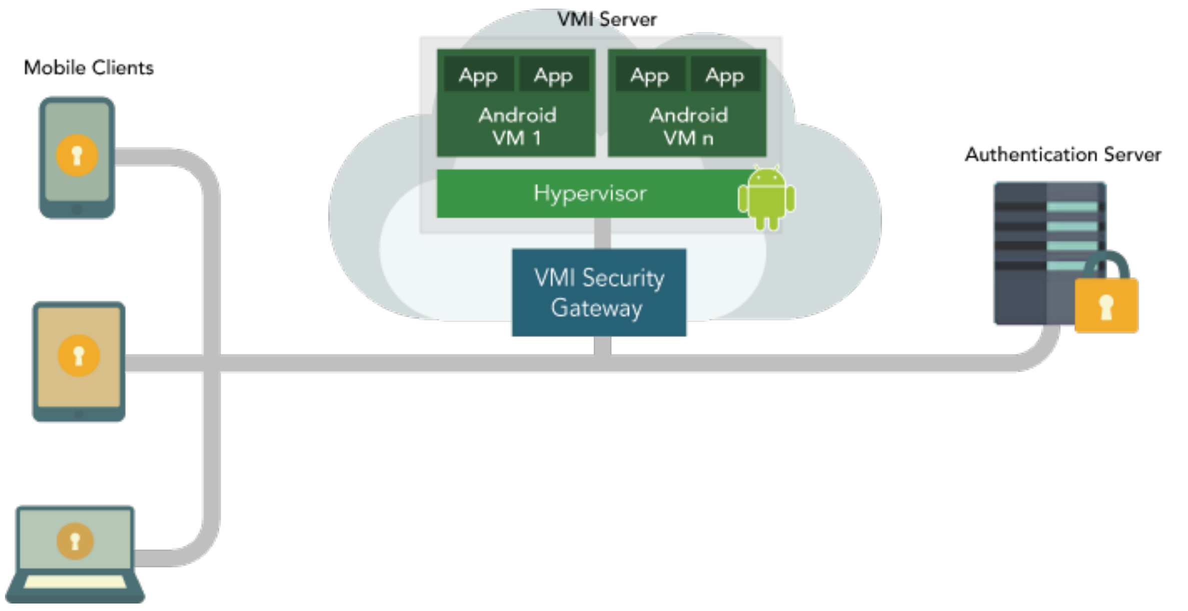
Entry in `/etc/group: sudo:x:27:walter`

```
walter@eve:~$ id
uid=1000(walter) gid=1000(walter)
groups=1000(walter),4(adm),24(cdrom),27(sudo)
```

Principle of Least Common Mechanism

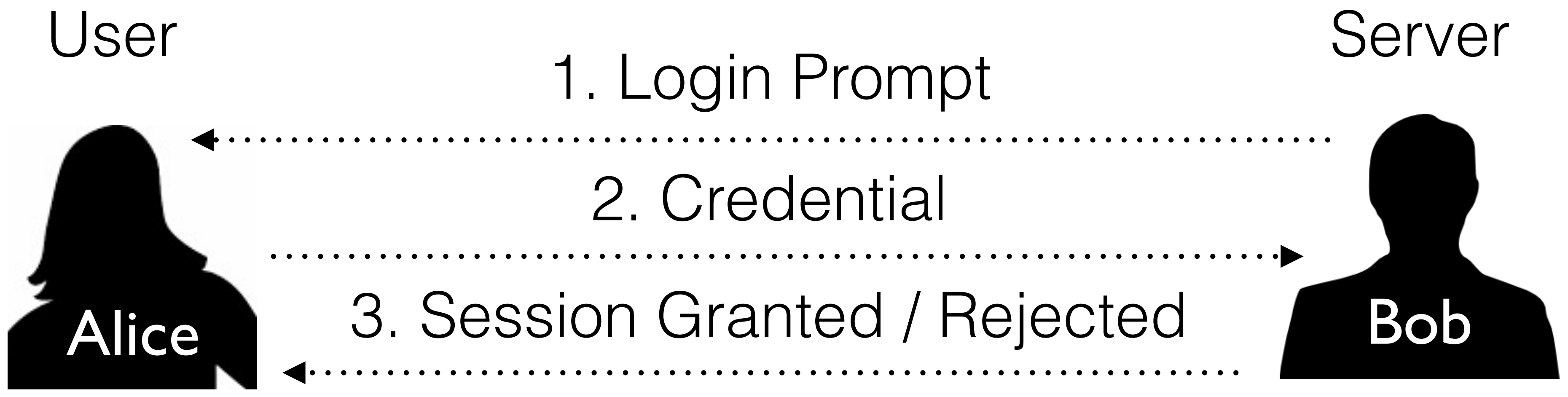
The *principle of least common mechanism* states that mechanisms used to access resources should not be shared.

Example: remote mobile virtualization



Virtual Mobile Infrastructure Deployment 🅭 BY-SA 3.0 Kaseykcross

Basic OS Authentication

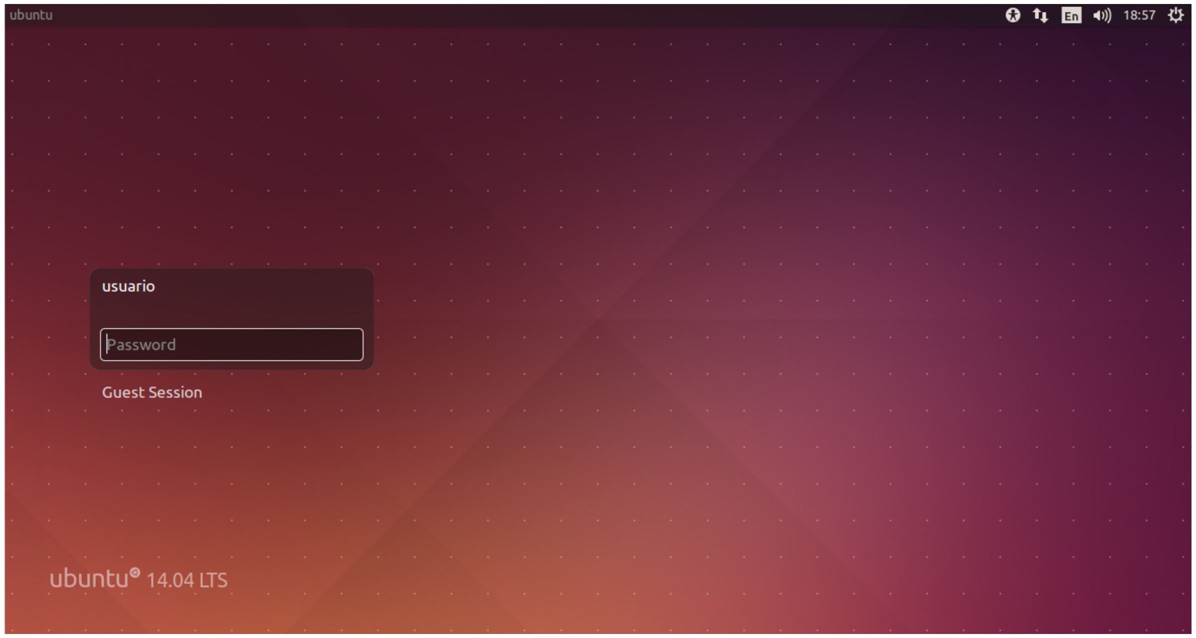


Has:
Password
One-time password
Biometric

Checks:
User list
Credential list
ID list

How we log in — in practice

Local (Unity):



Login ubuntu (cc) BY-SA 4.0 Ricardoborges

Remote (ssh):

```
$ ssh walter@140.247.178.194
walter@140.247.178.194's password: xxxxxxxx
Welcome to Ubuntu 12.04.5 LTS (GNU/Linux 3.2.0-88-generic x86_64)
```

Privilege Levels in Linux

Users have a corresponding ASCII username

`useradd(8)` limits this to 32 characters in Ubuntu

The operating system identifies users by an integer known as a UID

Not all UIDs are created equally:

- root (0), the superuser
- daemon (1) and sys (3), handle some aspects of the network
- lp (7), used for the printer system.
- mail (8), email delivery
- nobody (65534), owns no files and sometimes used as a default user for unprivileged operations

How is privilege enforced?

- OS System Calls
- Linux: `getuid()` returns the real user id of a calling process
- Example: attempt to kill a root owned process as a user

Target process:

```
root      32661  0.0  0.0  21856   380 ?        S    Jan18   0:00 /sbin/udevd --daemon
```

User attempts to kill it:

```
walter@eve:~$ kill -9 32661
-bash: kill: (32661) - Operation not permitted
```

`kill(2)` checks the UID of the calling process and bails out:

```
getpid()                                = 2391
kill(32661, SIGKILL)                    = -1 EPERM (Operation not permitted)
```

`su(1)` and `sudo(8)`

- Logging in as root is considered to be dangerous these days
  - Block root ssh access via `/etc/ssh/sshd_config` `PermitRootLogin no` option
- An alternative: `su` to root
- A better alternative: only run one privileged command at a time
  - example: `sudo service apache2 restart`

Groups

- We saw that being in a special group facilitates sudo access
- Users often need to share resources
  - Systems facilitate this by putting users into groups
  - A group is an alias for a set of users
  - Two models: (1) users are assigned groups for the duration of their login session; (2) users can change from one group to another in the same session

Common group:

```
uid=1000(walter) gid=1000(walter) groups=1000(walter),4(adm),27(sudo),250(cvrl)

uid=1001(gabe) gid=1001(gabe) groups=1001(gabe),250(cvrl)
```

Linux auth model

Authentication looks simple when you do it, but things are more complicated behind the scenes

How user information is stored on the local system:

`/etc/passwd` — All user login information except for the passwords

`/etc/shadow` — The encrypted passwords

/etc/passwd

```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
walter:x:1000:1000:Walter Scheirer,,,:/home/walter:/bin/bash
sshd:x:115:65534::/var/run/sshd:/usr/sbin/nologin
dustin:x:1001:1002:,,,:/home/dustin:/bin/bash
```

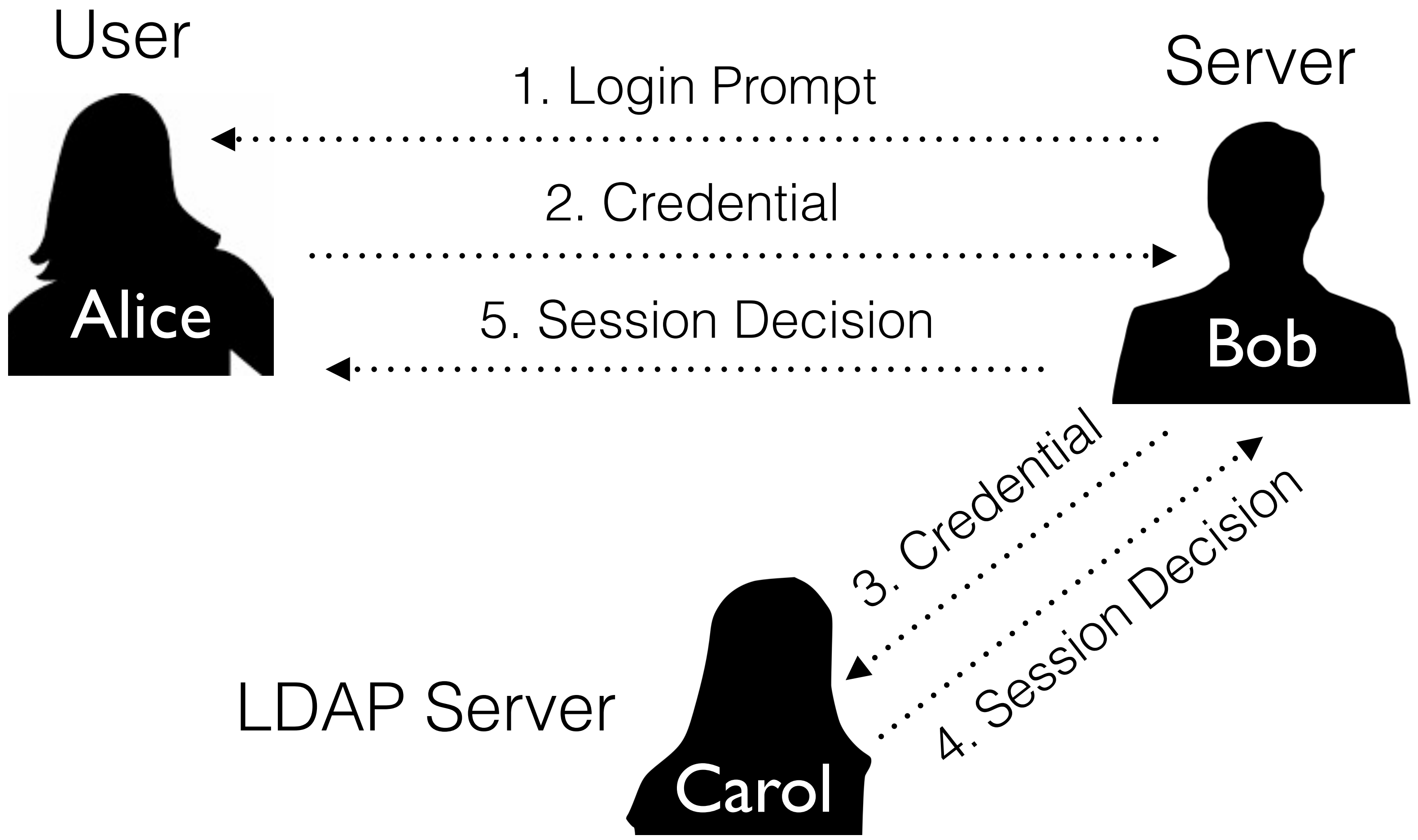
/etc/shadow

```
root:!:15649:0:99999:7:::
daemon:*:15575:0:99999:7:::
bin:*:15575:0:99999:7:::
sys:*:15575:0:99999:7:::
sync:*:15575:0:99999:7:::
games:*:15575:0:99999:7:::
man:*:15575:0:99999:7:::
walter:$6$yKb3N.3f$qC0X9KVHWV6F/
T8b.aYJLXdF4hwp9MtMe/
fLxtCPArX.CgPMRdFw3qwZ1LamMLTpa.p0WplXqGfkPyDrGqBsM
.:15649:0:99999:7:::
sshd:*:15649:0:99999:7:::
dustin:$6$A2AskP/c$vDB3eqaLNH9wwyn/
nhGjKWt5Wb3uSg6RUgmr3RyqGuscg6ampymUNftU/
PvGwN4tNorlUDmzHJv1qwEK.m5c01:16237:0:99999:7:::
```

$6$salt$encrypted format indicates SHA-512

Lightweight directory access protocol (LDAP)

Centralized network authentication simplifies user management:



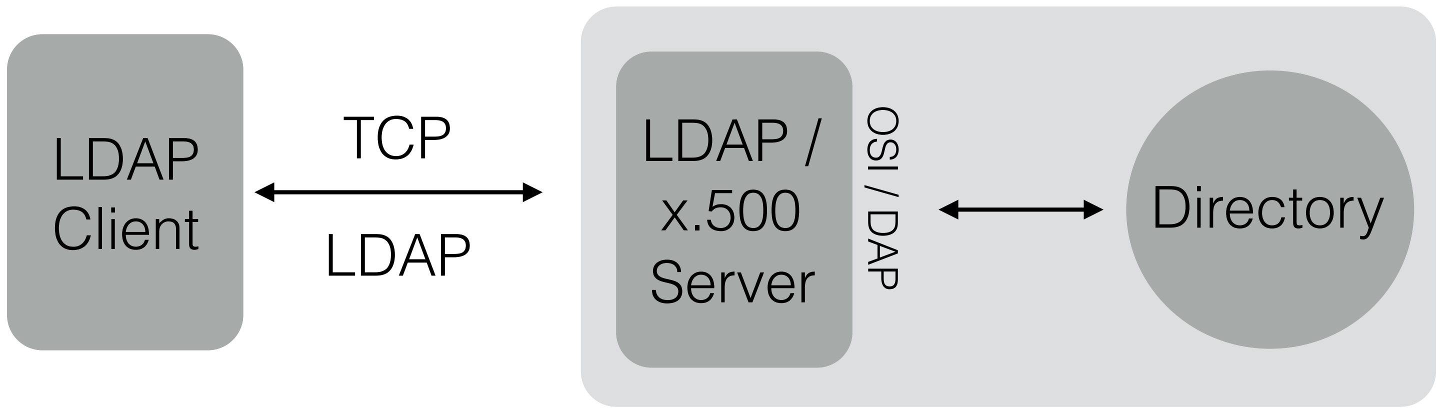
LDAP Protocol

RFC 4511

- Protocol for accessing X.500-based directory services
- Designed to run over TCP/IP networks
- LDAP entry is a collection of attributes that has a globally-unique Distinguished Name (DN)
    - `cn` for common name (Walter Scheirer)
    - `mail` for email address (walter.scheirer@nd.edu)
- Entries are arranged in a hierarchical tree-like structure
    - Useful for arranging user records into an organization's structure

Role of x.500 in LDAP

In essence, LDAP is just an access protocol to an x.500 directory service



LDAP is commonly directly implemented in X.500 servers

Pluggable Authentication Modules (PAM)

- Provides authentication modules for applications
  - Solves problem of developers writing their own authentication modules
  - Suite of shared libraries with common configurations
- During authentication, program invokes library routine `pam_authenticate`
  - The routine accesses configuration files in `/etc/pam.d`
  - Example: `sshd` will access `/etc/pam.d/sshd`

/etc/pam.d/sshd

```
# Disallow non-root logins when /etc/nologin exists.
account    required     pam_nologin.so

# Uncomment and edit /etc/security/access.conf if you need to set complex
# access limits that are hard to express in sshd_config.
# account  required     pam_access.so

# Standard Un*x authorization.
@include common-account

# Standard Un*x session setup and teardown.
@include common-session

# Print the message of the day upon successful login.
session    optional     pam_motd.so #  [1]

# Print the status of the user's mailbox upon successful login.
session    optional     pam_mail.so standard noenv #  [1]

# Set up user limits from /etc/security/limits.conf.
session    required     pam_limits.so

# Set up SELinux capabilities (need modified pam)
# session  required     pam_selinux.so multiple

# Read environment variables from /etc/environment and
# /etc/security/pam_env.conf.
session    required     pam_env.so # [1]
# In Debian 4.0 (etch), locale-related environment variables were moved to
# /etc/default/locale, so read that as well.
session    required     pam_env.so user_readenv=1 envfile=/etc/default/locale
```

PAM Configuration Files

```
   (1)          (2)              (3)
account      required      pam_nologin.so
```

- First field describes the auth. related mechanism treated by the line
- Second field controls the calling of the modules
  - `required` means failure of the module makes authentication fail
- Third field is the name of the module (dynamic library)

PAM Configuration Files

```
account     required     pam_nologin.so
session     optional     pam_motd.so
session     optional     pam_mail.so standard noenv
session     required     pam_limits.so
session     required     pam_env.so
session     required     pam_env.so user_readenv=1 envfile=/etc/default/locale
```

- *Stacking*: modules are invoked successively
- Configuration determines the order
  - Caller can make no assumptions about how the modules work
  - Authentication is in effect hidden from the application using PAM

SELinux

What if we uncomment this option?

```
# Set up SELinux capabilities (need modified pam)
# session  required     pam_selinux.so multiple
```

- Gives users and administrators more access control than the base OS provides
- Access can be constrained on such variables as which users and applications can access which resources
- Access controls are determined by a policy
    - Can’t be changed by careless users or misbehaving applications

http://selinuxproject.org/

SELinux

- Adds finer granularity to access controls
  - Control fundamental OS operations: you specify who can unlink, append only, move a file, etc.
  - Specify access for network resources and interprocess communication (IPC)
- **SELinux users** are not equivalent to Linux users
  - They cannot change via `su` or `sudo`
  - Many Linux users will use the same SELinux user
  - SELinux users that are generic have the suffix "`_u`", such as `user_u`.

SELinux

- **SELinux roles** are defined by the policy
    - Examples: unprivileged user, web administrator, database administrator
    - Objects have the role `object_r`
    - roles have the suffix "`_r`", such as `user_r`
- **SELinux types** are the primary means of determining access
    - a type has the suffix "`_t`", such as `user_t`

SELinux context framework

Contexts are attributes used to determine if access should be allowed between a process and an object

Contexts consist of 3 required fields and 1 optional field:

`user:role:type:range` $\longleftarrow$ optional

Example with required fields:

```
system_u:system_r:xserver_t
```

Adding optional multi-level security:

```
system_u:system_r:xserver_t:SystemLow-SystemHigh
```

SELinux object classes and rules

Object classes are used in the policy and in access decisions to more finely specify what access is allowed

file object class has the permissions *create*, *read*, *write*, and *unlink*

unix_stream_socket object class has the permissions *create*, *connect*, and *sendto*

Rules bring all of these elements together:

```
allow user_t user_home_t:file { create read write unlink };
```

More on file system security coming right up…