Managing Users and Groups

Portions courtesy Ellen Liu

Outline

- What constitutes a user?
  - /etc/passwd, /etc/shadow, /etc/group files
- User management tools
  - Adding users: basic steps, automation, bulk
  - Removing users, disabling logins
  - Pluggable Authentication Modules (PAM) and centralized account management

Introduction

- Adding, removing users is a routine task
  - Centralized servers may have hundreds of accounts
- Identity management
  - Not only adding users to specific machines
  - But also across the entire computing environment
- Security aspect
  - Infrequently used accounts and accounts with easily guessed passwords are prime targets for attacks

What constitutes a user?

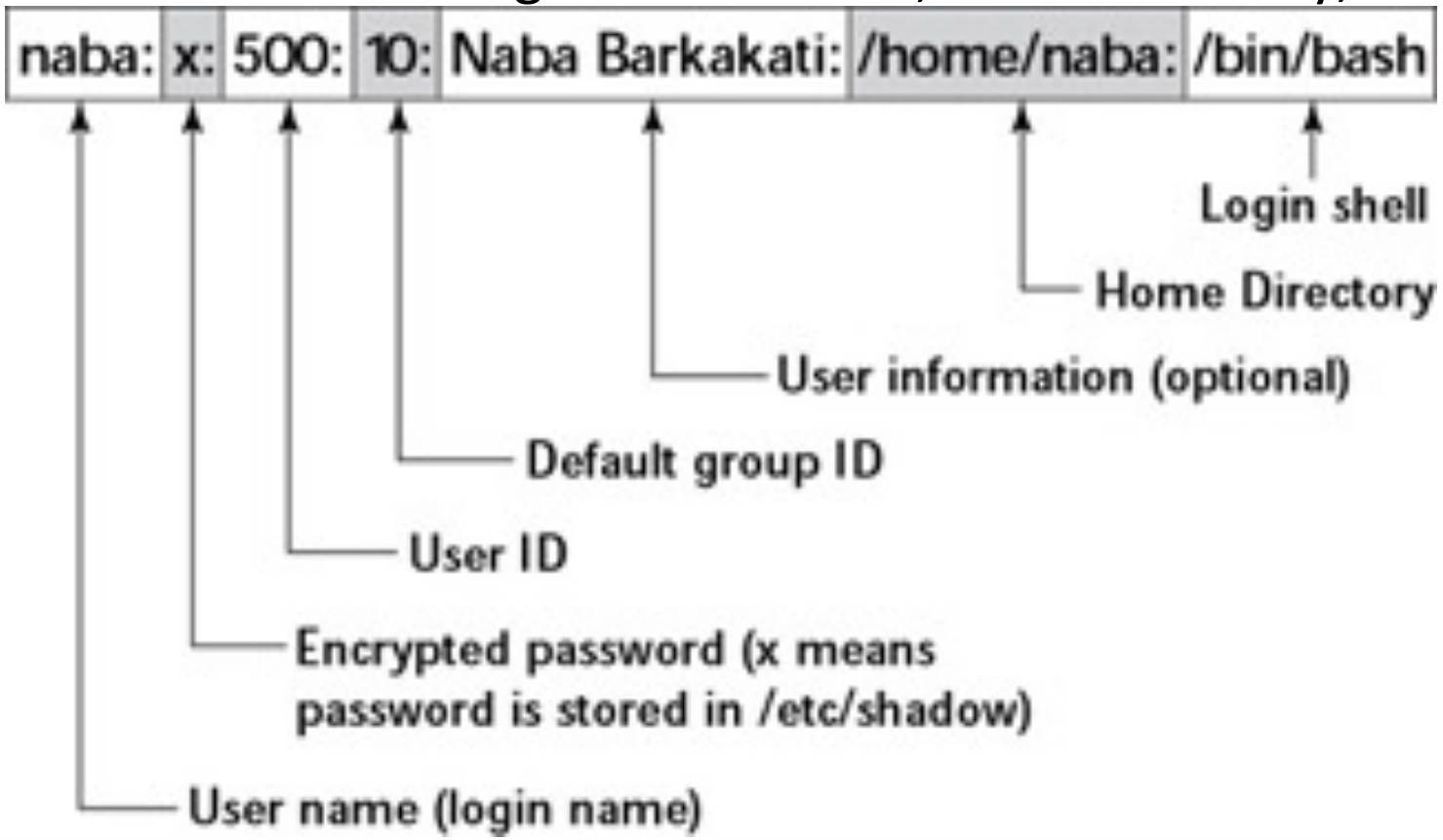
- UNIX was designed to be a multi-user OS
  - Per user "baggage": files, processes, resources, …
- User and group
  - Each user has a unique *user ID (UID),* must belong to at least one *group*. Each group has a unique *GID*
  - Every file and program must be owned by a *user*
  - A running program inherits the permissions of the user who invokes it
- All user information is in text files
  - A simple text editor suffices for management

System internals

- A user is just a number (e.g., 1003)
- Each file the user owns stores only this number as the owner in its metadata
- A system database translates human readable-ish names to numbers
  - E.g., porter == 1003

The /etc/passwd file

- It is a list of users recognized by the system
- It is consulted at login time for UID, home directory, etc.



The seven fields (1/3)

- username: by convention, up to 8 lowercase letters, numbers, and underscores; case sensitive; easy to remember; must be unique
  - A user should have same username on all machines; a username always refers to the same person
  - A naming standard: first names, last names, numbers
  - Used in email addresses
- Encrypted password: passwords are max 8 chars long on legacy systems
  - Encryption schemes: crypt (DES), MD5, Blowfish, …
  - Never ever leave this field empty: that means no password

The seven fields (2/3)

- UID: unsigned 32-bit integers, root has UID 0
  - *pseudo-users* own commands and configuration files, with a fake shell so nobody can login as them
  - UIDs for real user often start at 500 or higher
  - Do **not** recycle UIDs; files in backup may be confused
  - UIDs should be unique within the entire organization
- Default GID: unsigned int, root or system has GID 0
  - Some predefined groups for OS housekeeping: *bin, …*
  - New files/directories are owned by your default GID

The seven fields (3/3)

- *GECOS*: *General electric comprehensive OS*, comma separated personal info: name, office, phone, home phone
  - Try the *finger* and *chfn* commands
- Home directory: default directory at login, stores user specific *configuration files*, *startup scripts*, normal files
- Login shell: the first program to run upon login
  - Popular default: BASH /bin/bash and C shell /bin/tcsh
  - The *chsh* command, or *vipw* the passwd file
  - Available shells are in /etc/shells file

Note on stored passwords

- Your password should never be stored as plaintext
- Most systems store the output of a *one-way function*
  - For example, a cryptographically strong hash
  - Login collects password, passes input through one-way function, compares output

Stored Password Example

- Example: My password is ‘correcthorse’
  - f(‘correcthorse’) = 88c2352e1eb4c0b8f44e4ef596cc5362
  - This is stored in a system database
- If someone tries to log in as me and types ‘batterystaple’, the system computes:
  - f(‘batterystaple’) = d59c5615c874d9a5ca31d6147fd6bfe5
  - != 88c2352e1eb4c0b8f44e4ef596cc5362
- And the login is rejected

Caveat

- In theory, a one-way function implies that, if you know the output, you can't figure out the input
- In practice, one can guess long enough and eventually find an input that produces the output
- Unix used to keep the output in /etc/passwd, which is public
  - Now kept in a read-protected file /etc/shadow

The /etc/shadow file

- The file is readable only to root; keeps encrypted passwords
- Contains 9 fields with last being empty
  - login name (mandatory): same as in /etc/passwd
  - encrypted password (mandatory)
  - date of last password change: #days since 1/1/1970
  - min # days between changes
  - max # days between changes
  - #days in advance to warn about expiration
  - days for which the account can be inactive before being locked
  - account expiration date: use the command *usermod* to change

The /etc/group file

- Contains names of groups and group member lists
- 4 fields: group name, password/placeholder, GID, members

```
root:x:0:root
bin:x:1:root,bin,daemon
daemon:x:2:root,bin,daemon
sys:x:3:root,bin,adm
adm:x:4:root,adm,daemon
```

- Group names, GIDs should be consistent in organization
- Group membership is the union of passwd and group
- Default is to place users in their own personal groups
- Commands to manage groups: *groupadd*, *groupmod*, *groupdel*

Basic Steps to Add Users

- Required
  - Edit the passwd and shadow to define the user account
  - Add the user to /etc/group file
  - Set an initial password
  - Create, *chown*, and *chmod* the user's home directory
  - Configure permissions
- For the user
  - Copy default startup files to user's home directory
  - Configure user's email
- For you:
  - Verify that the account is set up correctly
  - Add user contact info and account status to your database

Notes on Manual Operation

- Use *vipw* to edit passwd and shadow
- *Always set an initial password*, do not leave it to the user
- *Startup files* start with a dot; set terminal type, mesg, environment variables, command aliases, search path…
- Default startup files for shell:
  - bash: .bashrc, .bash_profile
  - tcsh: .login, .cshrc
- Sample startup files are in /etc/skel
- System-wide startup files are processed before user's
  - Depends on shell, e.g., /etc/profile for bash

Final Steps

- To verity correct account setup, first log out, then login as the new user, type
  - *pwd* /* to verify the correct home directory */
  - *ls –la* /* to check owner/group of startup files */
- Notify new users of their username, passwords
  - in person or over the phone
  - Remind them to change passwords immediately
- At a large site, maintain a database to track accounts
  - Who someone is, why they have an account, etc…

Unsolicited Advice

- Do understand where all of the account configurations live and how they work
- Don't configure accounts by hand, use automated tools
  - Configurations are spread across multiple files with invariants across files
  - Files have delicate formats---a typo can break your system!
- Tools greatly reduce these sorts of risks

Automation

- Command-line or GUI based
- *Useradd* implements the basic steps above, it is configurable for customization, uses configuration files
  - Red Hat: /etc/login.defs, /etc/default/useradd
  - Define password aging, encryption scheme, UID/GID ranges. *useradd –D* shows the defaults

```
$sudo useradd –c “David Hilbert” –d /home/
math/dhilbert –g faculty –G famous –m –s /
bin/tcsh dhilbert
```

Adding Users in Bulk

- Command *newusers* creates multiple accounts at one time based on the content of a text file
  - The file is like /etc/passwd with clear text passwords!
- It honors the password aging parameters in /etc/login.defs, but it does not copy in the default startup files
- Often a script is written as the wrapper for *useradd* rather than using *newusers*
  - It reads enrollment roster, forms usernames using local rules, guarantee uniqueness, with strong random passwords, etc.

Steps to Remove a User

- Remove the user from local user databases
- Remove from /etc/aliases or add a forwarding address
- Remove the user's *crontab* and any pending *at* jobs
- Kill any of the user's processes that are still running
- Remove from passwd, shadw, group, gshadow files
- Remove the user's home directory (backup first)
- Remove the user's email spool (queue) (backup first)
- Clean up entries on shared calendars, room reservations
- Delete or transfer ownership of the user-run mailing lists

Automation

- *userdel* command automates the process
- Red Hat has a userdel.local script but no file backing ups

`/usr/sbin/userdel baduser`, delete account and files

`/usr/sbin/userdel -r baduser` also remove the home dir

- A recommendation is to not remove an account right away, but first simply *disable* it
  - That user may come back, may ask for some files, others may ask for some files, etc.

Disabling Logins

- To temporarily disable a user's login
- A straightforward way: add a star or other char in front of the user's encrypted password in /etc/shadow
  - *usermod –L user* to lock, *usermod –U user* to unlock passwords, -L put an ! in method above
  - User login will fail
- To add notification and explain why to the user, can replace the user's shell with a program to do so, the program then exits, terminates the login

Enterprise-Scale Logins

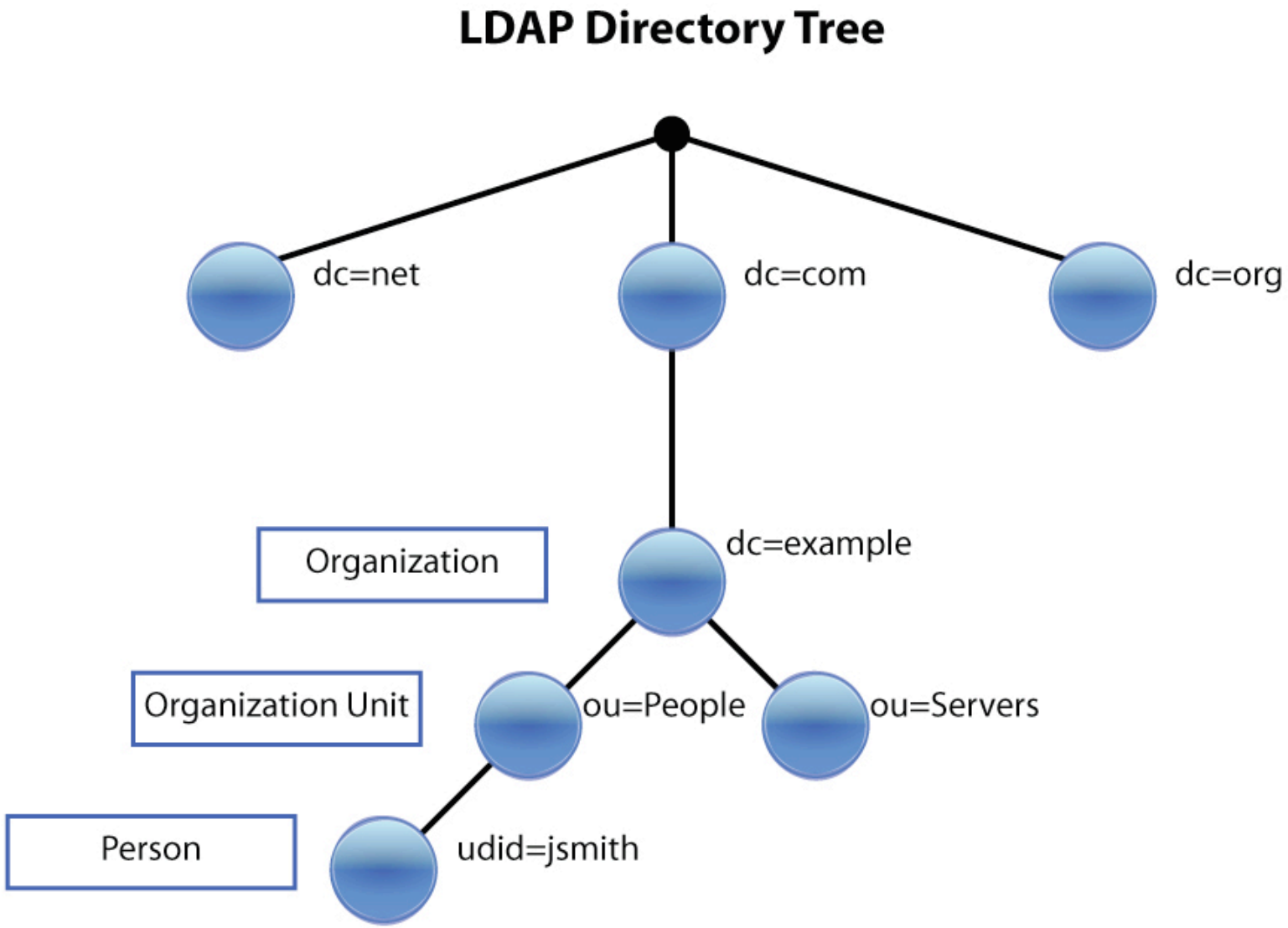
- What if I want a user to be able to log in to all machines in a lab?
- You need identical password databases on each machine
- How?
  - Copy them around? Seems error prone
- Idea: consolidate into a database shared over the network

LDAP

- Lightweight Directory Access Protocol
  - Underlying technology in Microsoft Active Directory
  - Linux/Unix: OpenLDAP
  - Amazingly: all interoperable

But what does LDAP do?

- Basically, you can define some big organizational hierarchy with arbitrary amounts of information (office number, phone, etc)
  - But, importantly, Unix-style credentials information
- So what?
  - Point a machine at part of the tree to get its user information



From: http://blogs.citrix.com/2010/11/05/load-balancing-ldap-authentication/

Key insight

- This tree of information is super-flexible
  - Each node can have arbitrary attributes
- I can create a node that has all of the attributes of an entry in /etc/passwd or /etc/group

Goal

- LDAP server stores all account info
- All machines get user account info from LDAP server(s)
- How do I get the system to use LDAP instead of the local password database?

Pluggable Authentication Modules

- PAM centralizes a system's authentication facilities
  - programs such as *login, sudo, passwd, su*, do not need to include own authentication code any more, they can simply use PAM standard library routines
  - reduces risk inherent in writing secured software
  - allows admin to set site-wide security policies
  - defines an easy way to new authentication methods
- The tools to add and remove users operate under PAM's rules and constraints

PAM Targets

- Can select one or multiple sources
  - And prioritize
- Sources include: local files, LDAP, NIS, etc.
- Configured by /etc/nsswitch.conf and files under /etc/pam.d/*

Integrated Example:

- /etc/nsswitch.conf (use local files, and ldap for user accounts):

passwd: files ldap

group: files ldap

shadow: files ldap

Example, cont:

- Configure the LDAP client to use a particular server and subtree
- /etc/ldap.conf (key entries):

base o=oscar,dc=cs,dc=stonybrook,dc=edu

uri ldap://kermit ldap://miss-piggy

Example, cont

- Configure PAM to accept local or LDAP accounts
- Modify serveral files similarly to /etc/pam.d/common-auth

```
account [success=2 new_authtok_reqd=done default=ignore]        pam_unix.so
account [success=1 default=ignore]      pam_ldap.so
account requisite                       pam_deny.so
account required                        pam_permit.so
```

How to test?

- Command line: getent passwd
- Lots of tools that can connect to LDAP server: jxplorer is good

Replication

- As with other network services, you really want more than one LDAP server
  - Again, primary and replica architecture
- Can be configured using the syncrepl option
  - Replicas periodically get updates from master

Caching

- Going to the LDAP server for every login can get expensive
  - Just like with DNS
- Common system service for caching called nscd
  - Name Service Caching Daemon
- By default, caches lookups for 1 hour

nscd trade-off

- Pros:
  - Reduce latency, network traffic to server
  - Tolerate a server reboot without interruption (most of the time)
- Cons:
  - Takes 1 hr before new users can log in
    - Or to revoke a user's account
- Nonetheless, very commonly used

Summary

- Each system has a user/group/password database
- If you want single-sign-on for many machines, you need to distribute the database
  - LDAP helps

Summary, 2

- Servers: store the database
  - Want multiple servers for redundancy, backup
- Clients (all of the user machines:
  - Get user account info from the server
  - PAM transparently combines the local database with LDAP
  - NSCD caches results of server queries to reduce network traffic and server load