CS615 - Aspects of System Administration

Multiuser Fundamentals

Department of Computer Science
Stevens Institute of Technology
Jan Schaumann
jschauma@stevens.edu
https://stevens.netmeister.org/615/

Multiuser

UNIX was designed from the beginning (1970s) as a portable, multi-tasking, *multi-user* system.

Windows gained this functionality with WindowsNT in 1993.

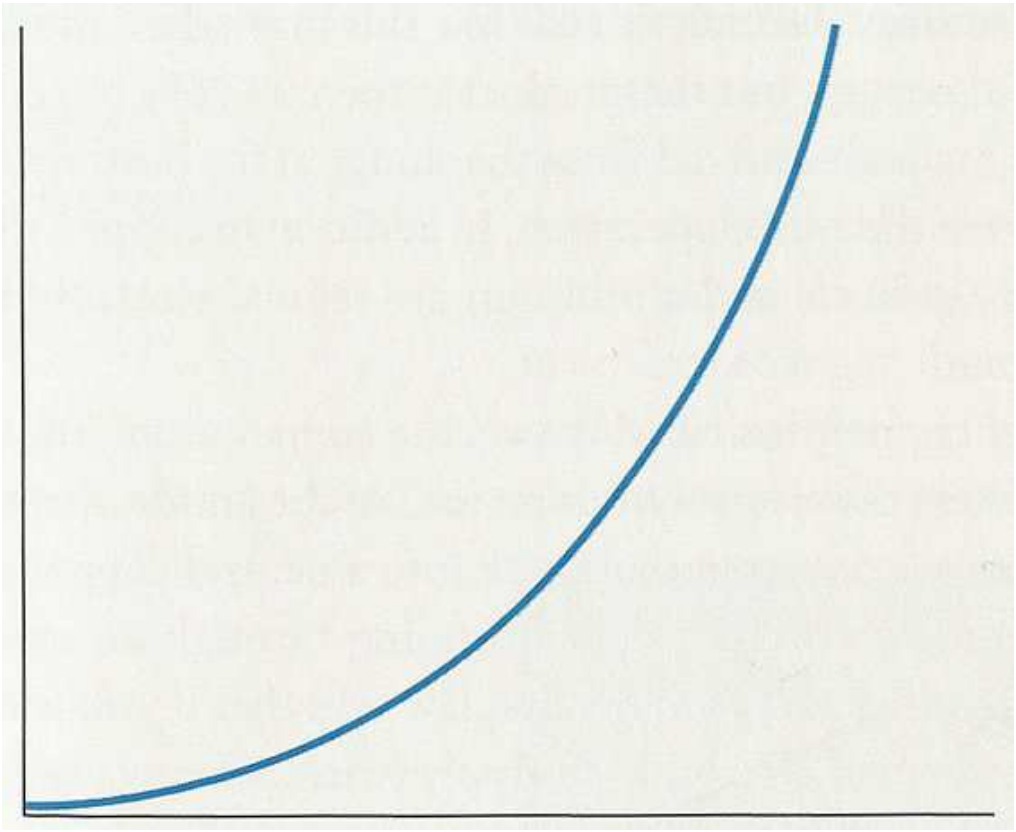
Mac OS followed in 2001 with OS X.

Implications of a Multi-User System

Implications of a Multi-User System

Consider Scalability

Things to consider:



Granting Privileges requires Trust

- different environments have different trust models
- human interactions in small groups strengthen trust
- larger groups are divided into smaller, close-nit groups
- the more groups you have, the weaker their trust bonds are

Granting Privileges requires Trust

- different environments have different trust models
- human interactions in small groups strengthen trust
- larger groups are divided into smaller, close-nit groups
- the more groups you have, the weaker their trust bonds are

Trust does not scale.

Granting Privileges requires Trust

Implement *Zero Trust* principles.

Granting Privileges requires Trust

Implement *Zero Trust* principles.

For computers, apply *Least Privilege*, time-based access expiration, and automated renewal with strong audit cabilities.

Granting Privileges requires Trust

Implement *Zero Trust* principles.

For computers, apply *Least Privilege*, time-based access expiration, and automated renewal with strong audit cabilities.

For humans, apply *Least Privilege*, time-based access expiration, and automated renewal with strong audit capabilities.

Granting Privileges requires Trust

Implement *Zero Trust* principles.

For computers, apply *Least Privilege*, time-based access expiration, and automated renewal with strong audit cabilities.

For humans, apply *Least Privilege*, time-based access expiration, and automated renewal with strong audit capabilities.

But beware getting in people's ways - they *will* find ways to circumvent your controls!

Implications of a Multi-User System

- users may want to keep files private

Implications of a Multi-User System

- users may want to keep files private
- users may want to share files

Implications of a Multi-User System

- users may want to keep files private
- users may want to share files
- users may (try to gain) access to files they shouldn't have access to

Implications of a Multi-User System

- users may want to keep files private
- users may want to share files
- users may (try to gain) access to files they shouldn't have access to
- users may (want to) do things that affect other users

Implications of a Multi-User System

- users may want to keep files private
- users may want to share files
- users may (try to gain) access to files they shouldn't have access to
- users may (want to) do things that affect other users
- different users may require different privileges

Users and User-IDs

alice

bob

claire

dennis

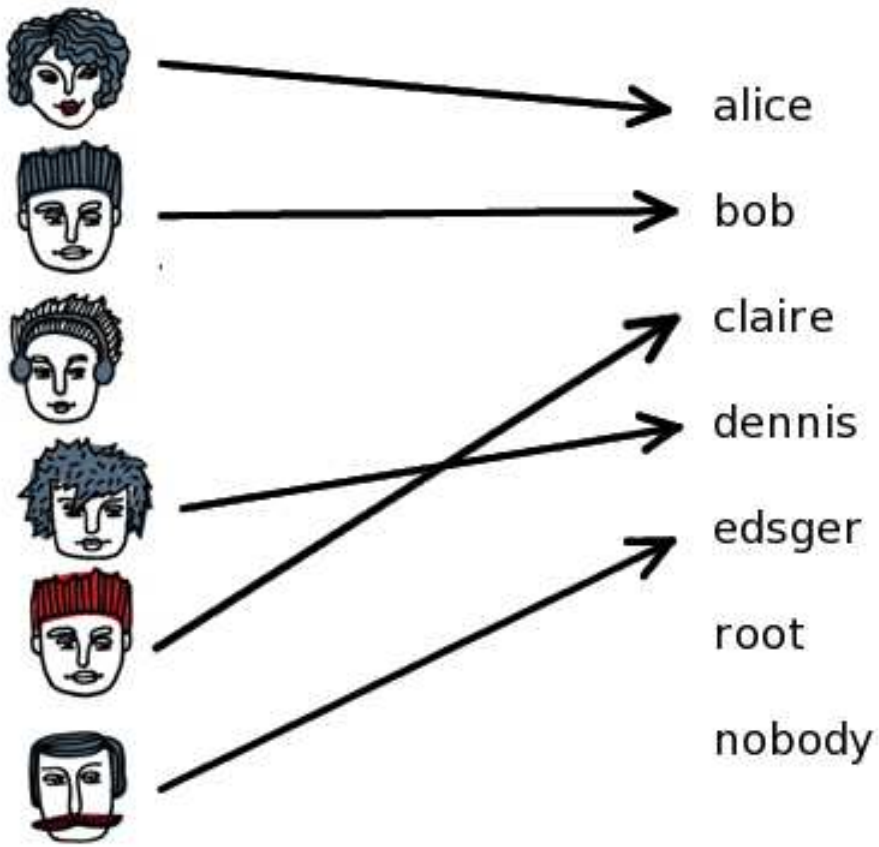
edsger

root

nobody

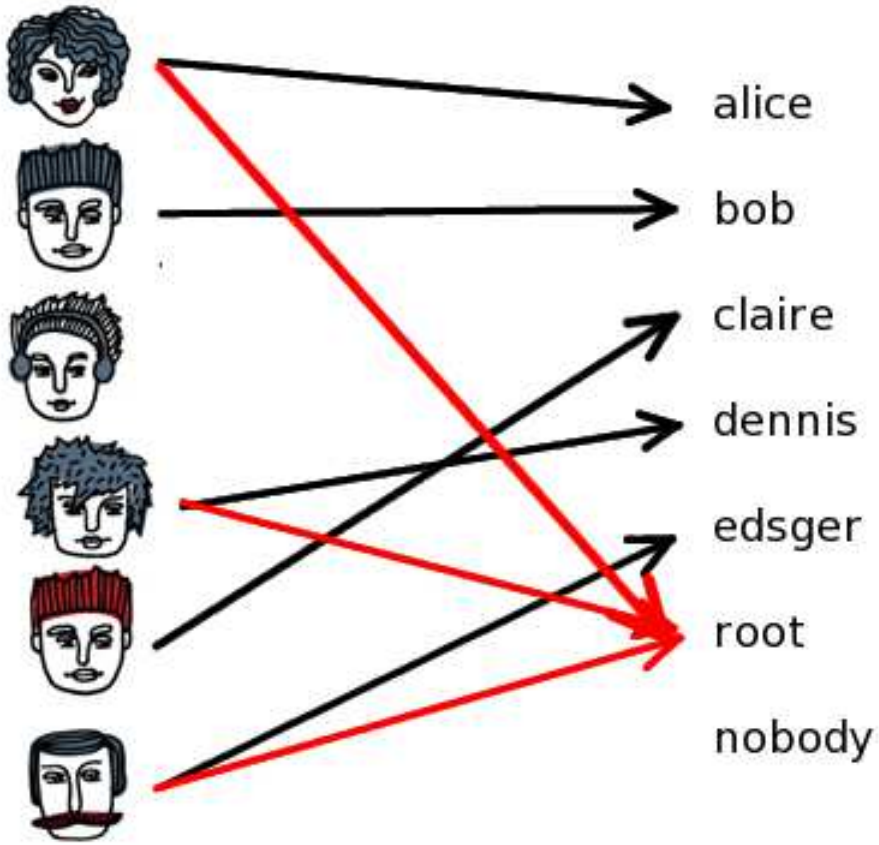
Bijective?

Users and User-IDs



Not surjective!

Users and User-IDs



Not injective, either!

Users and User-IDs

nobody

Authentication

- proof of identity, not proof of *authorization*

Authentication

- proof of identity, not proof of *authorization*
- something you know, something you have, something you are

Authentication

- proof of identity, not proof of *authorization*
- something you know, something you have, something you are
- multi-factor authentication combines these to help protect against different threats

Authentication

- proof of identity, not proof of *authorization*
- something you know, something you have, something you are
- multi-factor authentication combines these to help protect against different threats
- mutual authentication may be a requirement

Authentication

Common examples:

```
NetBSD/amd64 (SERVER) (console)

login: jschauma
password: ********************************
NetBSD 7.0.2 (SERVER) #2: Tue Jan 24 02:33:13 EST 2017

Welcome to NetBSD!
hostname$
```

Authentication

Common examples:

```
$ ssh-keygen -l -f /dev/stdin <<<$(aws ec2 get-console-output \
        i-0990f1eb069c853c4 | grep ^ecdsa)
256 19:af:35:01:0b:2a:ee:3d:30:0f:69:11:cc:55:7c:20 (ECDSA)
$ ssh -i ~/.ssh/myawskey ec2-54-227-16-184.compute-1.amazonaws.com
The authenticity of host 'ec2-54-227-16-184.compute-1.amazonaws.com
(54.227.16.184)' can't be established.
ECDSA key fingerprint is 19:af:35:01:0b:2a:ee:3d:30:0f:69:11:cc:55:7c:20.
Are you sure you want to continue connecting (yes/no)?  yes
NetBSD 7.0.2 (SERVER) #2: Tue Jan 24 02:33:13 EST 2017

Welcome to NetBSD!
hostname$
```

Authentication

Common examples:

```
$ kinit
Password for jschauma@DOMAIN: **********************************
$ klist
Ticket cache: /tmp/krb5cc_ttypa
     Default principal: jschauma@DOMAIN

     Valid starting     Expires            Service principal
     02/13/17 13:50:21  02/13/17 21:50:20  krbtgt/KDC@DOMAIN
$ ssh somehost
somehost$
```

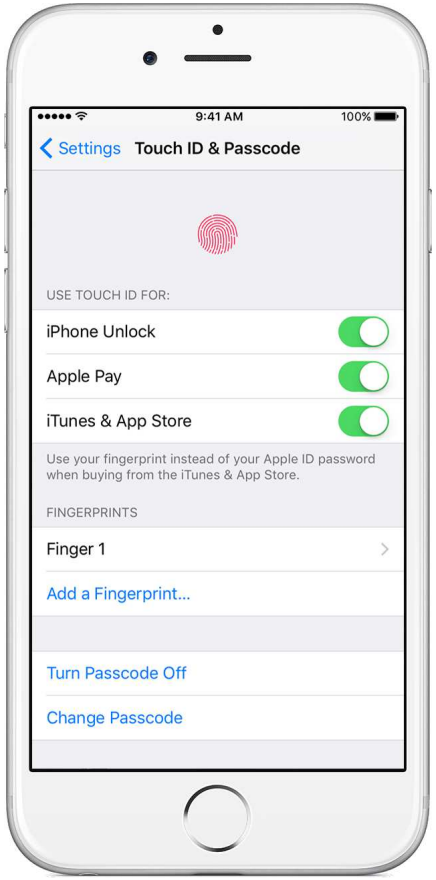
Authentication

Common examples:

```
localhost$ ssh sshca
YubiKey for ‘jschauma’: **********************************
Password: **********************************
localhost$ ssh-add -l
2048 SHA256:TzwuHGc5BKBe+VJSnGoVyh92J8XKBUkaL7MGQn8ML0Y (RSA)
2048 SHA256:TzwuHGc5BKBe+VJSnGoVyh92J8XKBUkaL7MGQn8ML0Y (RSA-CERT)
localhost$ ssh somehost
Duo two-factor login for jschauma

Enter a passcode or select one of the following options:

 1. Duo Push to XXX-XXX-0712
 2. Phone call to XXX-XXX-0712
 3. SMS passcodes to XXX-XXX-0712

Passcode or option (1-3): 1
Success. Logging you in...
Last login: Thu Jan 26 17:39:30 2017 from 10.1.2.3

somehost$
```

Authentication

Common examples:



Authentication

Common examples:

- passwords, PINs
- ssh keys, PGP keys, X.509 certificates
- security tokens: OTPs in hardware or software, RFIDs
- physical biometrics: fingerprint, retina scan, facial recognition
- behavioral biometrics: speech pattern, gait, keystroke dynamics...

Mix and match the above to yield multi-factor authentication:

- password + PIN via e.g. SMS
- ssh key + TOTP from e.g. mobile device
- fingerprint + security token
- ...

UNIX Fundamentals: User Accounts and File Permissions

Every account

- has a *unique* ID
- belongs to at least one group
- may or may not be password protected
- may or may not have a valid login program
- may or may not be allowed to escalate privileges

UNIX Fundamentals: User Accounts and File Permissions

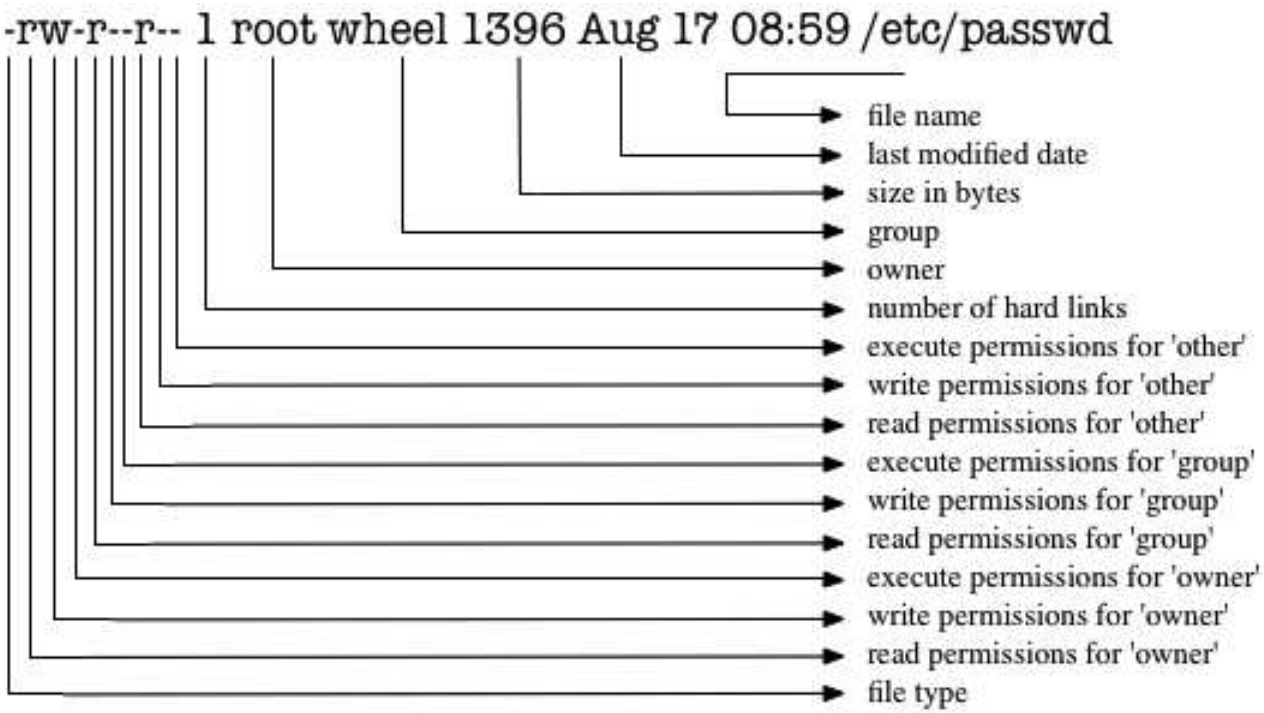
Every account

- has a *unique* ID
- belongs to at least one group
- may or may not be password protected
- may or may not have a valid login program
- may or may not be allowed to escalate privileges

Every file

- is associated with a *uid* and a *gid*
- has a number of protection bits

UNIX Fundamentals: User Accounts and File Permissions



Raising privileges

Some tasks require special privileges:

- binding a port < 1024 (e.g. 22, 25, 80, 443)
- operating on raw sockets (e.g. `ping(1)`, `traceroute(8)`)
- changing local passwords
- accessing files/directories without explicit permissions
- just about anything involving file systems
- ...

Raising privileges

Options:

```
$ ls -l command
-rwsr-xr-x  1 daemon  wheel  12556 Feb 17 21:45 command
$ man setuid
```

Raising privileges

Options:

```
somehost$ exit
$ ssh root@somehost
#
```

Raising privileges

Options:

```
$ su user2 -c 'some command'
Password:
$ su - root
Password:
#
```

Raising privileges

Options:

```
somehost$ sudo bash
jschauma is not allowed to run sudo on somehost.  This incident will be reported.
```

Raising privileges

Options:

```
jschauma@somehost$ ls dir
ls: cannot open directory dir: Permission denied
jschauma@somehost$ sudo bash
Sorry, user jschauma is not allowed to execute '/bin/bash' as root on somehost.
jschauma@somehost$ sudo ls dir
Sorry, user jschauma is not allowed to execute '/bin/ls' as root on somehost.
jschauma@somehost$ sudo -u otheruser ls dir
Password: ******************************
file1   file2
jschauma@somehost$
```

Unix Groups

- enables *arbitrary* collections of users to share resources
- information stored in `/etc/group`, format is:
  `name:*:GID:user1,user2,...`
- most Unix systems impose a limit of 16 or 32 group memberships per user
- most Unix systems have a common default group for new users (some Linux versions deviate)
- some Unix systems have/had group shadow files

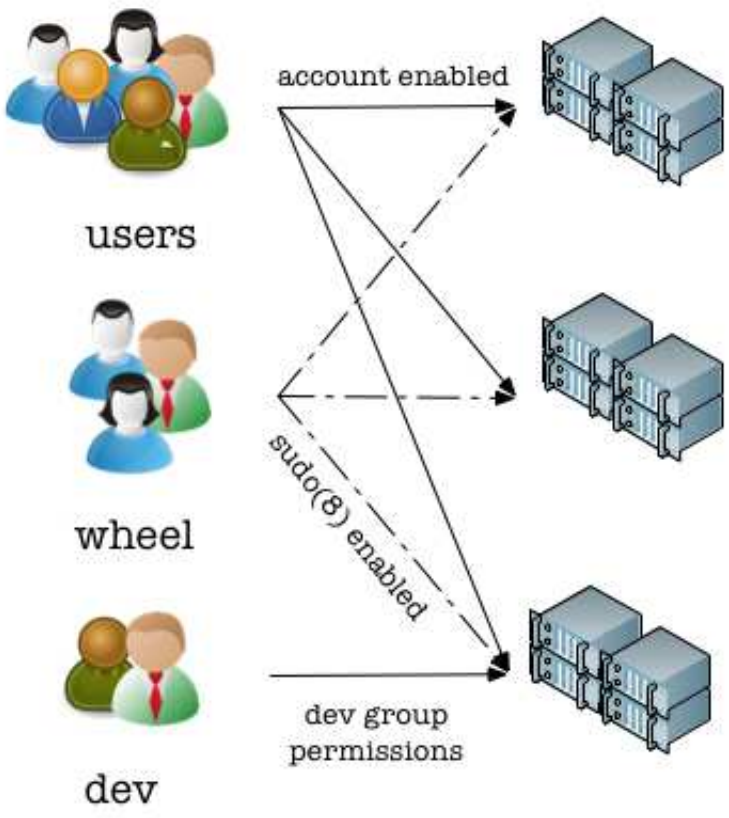
Group Access

At any but the smallest environments, we find:

- a central user database
- users divided into different access groups
- access to systems is granted primarily by such group membership
- privileges on a system are also granted by such group membership

The privileges granted in this manner are commonly broken down and controlled via *role-based access control* (RBAC).

Group Access



Multiuser Truths

- *All users are equal.*

Multiuser Truths

- *All users are equal.*
- *Some users are more equal than others.*

Multiuser Truths

- *All users are equal.*
- *Some users are more equal than others.*
- *The principle of least privilege applies to all.*

Multiuser Truths

- *All users are equal.*
- *Some users are more equal than others.*
- *The principle of least privilege applies to all.*
- *Trust does not scale.* (Think "*Zero Trust*")

Multiuser Truths

- *All users are equal.*
- *Some users are more equal than others.*
- *The principle of least privilege applies to all.*
- *Trust does not scale.* (Think "*Zero Trust*")
- *You will always face trade-offs.*

Adding and Removing Accounts

Recommended exercise:

`https://stevens.netmeister.org/615/useradd-exercise.html`

Reading

User Management:

- *Frisch*: Ch 6; *Burgess*: Ch 5;

- `https://is.gd/wg5OsE`
- `https://www.netmeister.org/book/06-users-and-groups.pdf`