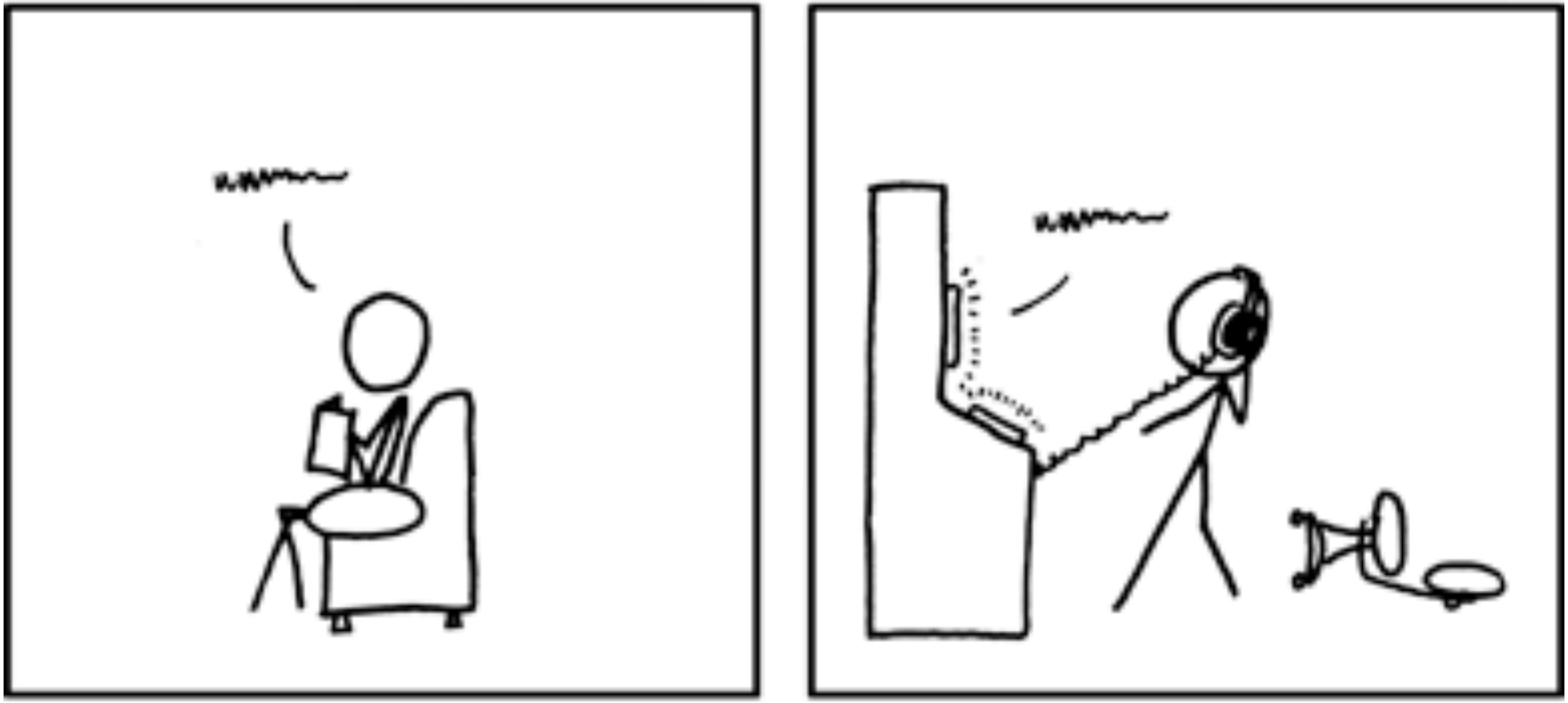
CS 55:
Security and Privacy

Authorization and Multilevel Security (MLS); malware

NOW AND THEN, I ANNOUNCE "I KNOW YOU'RE LISTENING" TO EMPTY ROOMS.



IF I'M WRONG, NO ONE KNOWS.
AND IF I'M RIGHT, MAYBE I JUST FREAKED THE HELL OUT OF SOME SECRET ORGANIZATION.

Source: https://xkcd.com/525/

Big idea: not all information has the same value

Implication: we should treat it differently

Agenda

1. Access control
2. Multilevel Security (MLS)
3. Linux access controls
4. Malware

Much of the early thinking about computer security began with the Cold War military

In the cold war, each side had lots of information to store and process

Much of the information was sensitive

There was a capable adversary that wanted to learn the information

Need a way to reason about security using a model

The ***information confinement*** problem attempts to ensure information doesn't flow to the wrong parties

Adapted from *The Craft of System Security* by Smith and Marchesini

An Access Control Matrix expresses who can do what to whom in a system

Access Control Matrix

Access control matrix often referred to as the *Security Policy*

Objects

Subjects	/home/alice	/home/bob	/home/charlie	/etc/password
Alice	read, write, execute			read
Bob		read, write, execute		read
Charlie	read, write, execute	read, write, execute	read, write, execute	read, write

Access modes can include read (r), write (w), execute (x), but also modify, delete, create, destroy, copy, export, …

Effective separation will keep unauthorized subjects from access to objects

Adapted from *The Craft of System Security* by Smith and Marchesini

A row in the matrix represents the capabilities a subject can perform

Capabilities

Subjects	Objects: /home/alice	/home/bob	/home/charlie	/etc/password
Alice	read, write, execute			read
Bob		read, write, execute		read
Charlie	read, write, execute	read, write, execute	read, write, execute	read, write

Easy to see what rights subject Alice has on each object
Not easy to see everyone who has access to a particular object

Adapted from *The Craft of System Security* by Smith and Marchesini

A column in the matrix represents an Access Control List (ACL) on an object

Access Control List (ACL)

Subjects \ Objects	/home/alice	/home/bob	/home/charlie	/etc/password
Alice	read, write, execute			read
Bob		read, write, execute		read
Charlie	read, write, execute	read, write, execute	read, write, execute	read, write

In practice it is difficult to maintain an Access Control Matrix (sparse)
Normally create Access Control Lists (ACL -- just a column)
Each object keeps a list of access modes subjects are allowed
Common modes – read, write, execute

Adapted from *The Craft of System Security* by Smith and Marchesini

Access control can be either mandatory or discretionary

Mandatory Access Control (MAC)

- A means of restricting access to objects based on:
  - The classification of the information contained in the object
  - Formal authorization (e.g., clearance) of subjects to access information of such sensitivity
- Access set by central authority (system admin)

Discretionary Access Control (DAC)

- A means of restricting access to objects based on the identity of subject and/or groups to which they belong
- The controls are discretionary in the sense that a subject with a certain access permission can pass that permission (perhaps indirectly) to any other subject

Adapted from United States Department of Defense Trusted Computer System Evaluation Criteria and http://www.cs.unc.edu/~dewan/242/f96/notes/prot/node13.html

Systems should check every access, enforce least privilege, and verify acceptable usage

Check every access

- Ensure subject has rights before each object access
- If access revoked, make sure do not use old permissions

Enforce least privilege

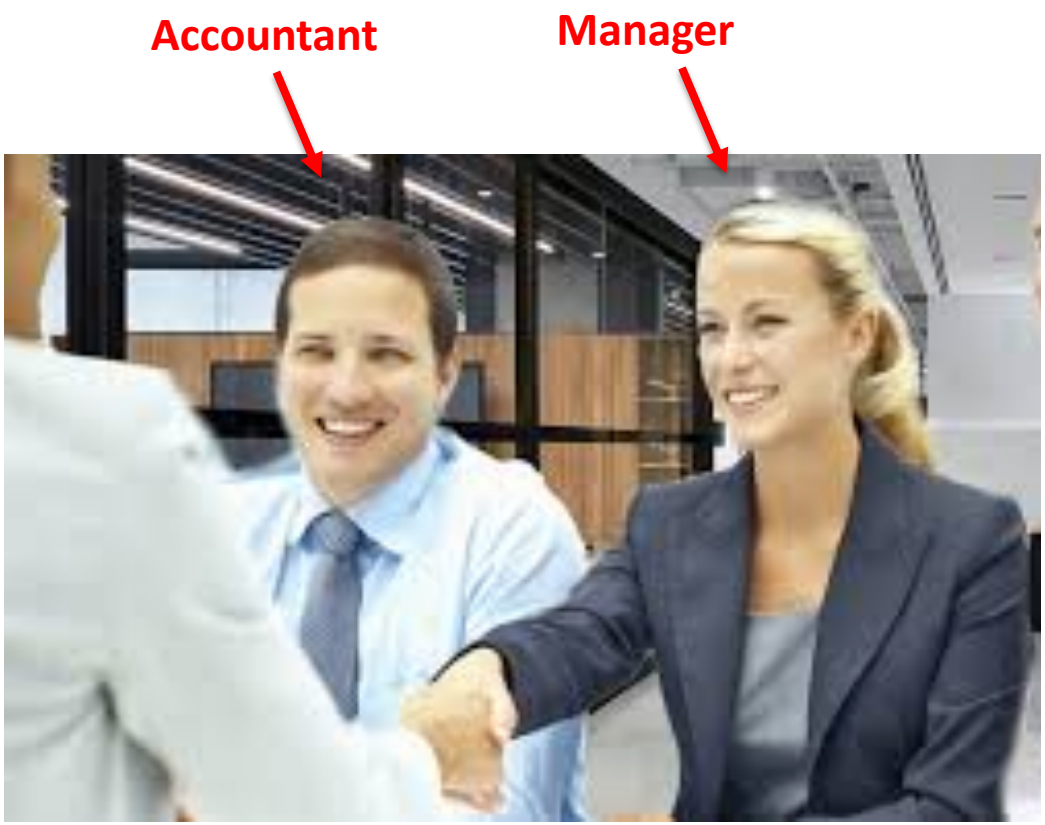
- A subject should have access to the smallest number of objects necessary to perform a task
- Avoid even if extra information would be useless or harmless
- Run applications with minimal privileges
- Guards against compromise

Verify acceptable usage

- Ability to access an object in a mode is a yes or no decision
- Must check if access is appropriate when granting/modifying privileges
- Management concept, not a technical concept

Another approach is to use Role-Based Access Control

Role-based Access Control (RBAC)



My view: RBAC is solved in theory, but not in practice! Too many roles, too many exceptions, and temporary assignments make it worse!

Role-based Access Control

Instead of Access Control Matrix with one entry per subject, create a role and add multiple subjects to the role

Simplifies management

- All employees can access certain information but not others
- Managers have increased access
- Put employees into appropriate group

Plan for creating and terminating employee privileges

Onboarding new employees

- Create Acceptable Use Policies (AUP) detailing what is allowed at work (e.g., is Facebook allowed?)
- New employees sign AUP
- Create new user accounts and assign privileges and groups
- Issue hardware (PC, laptop, tablet, phone, etc)

Terminating access

- Recover issued hardware
- Consider what happens to employee's data (assign to supervisor?)
- Accounts often initially deactivated vs. deleted
- Tricky when terminating a system admin (coordinate with HR)

Perform periodic audits to ensure privileges still match business need

- Things change over time, clear up temporary privileges
- Check time of day access

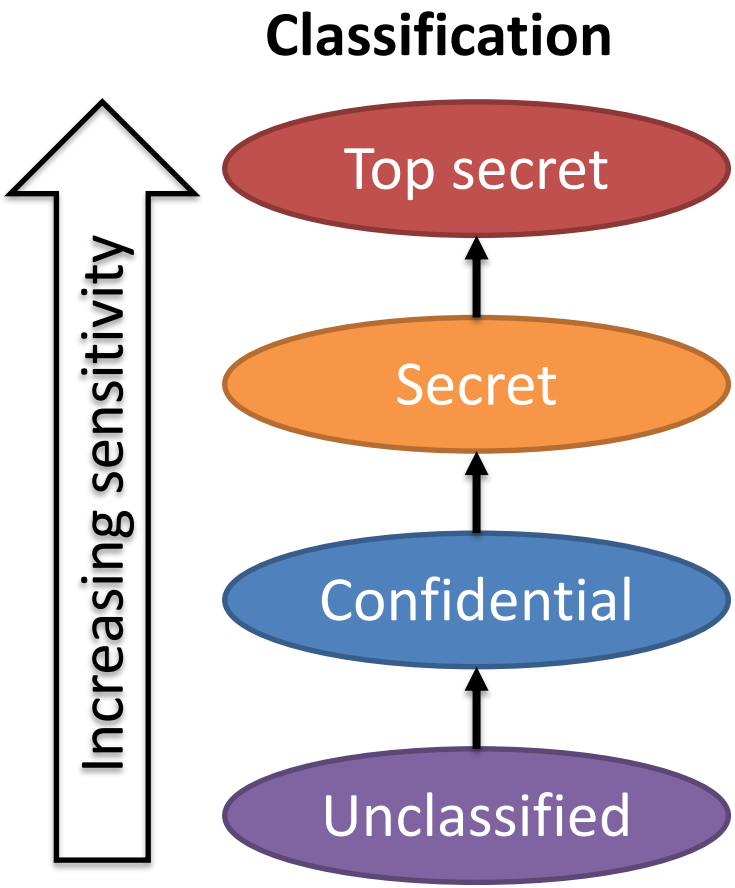
Adapted from https://www.professormesser.com/security-plus/sy0-501/account-management/

Agenda

1. Access control
2. Multilevel Security (MLS)
3. Linux access controls
4. Malware

Multilevel Security (MLS) uses different classification levels and compartments

Directed graph



Compartments

- Nuclear
- Crypto
- Imagery

Label *subjects* with their highest classification and compartments they need to know

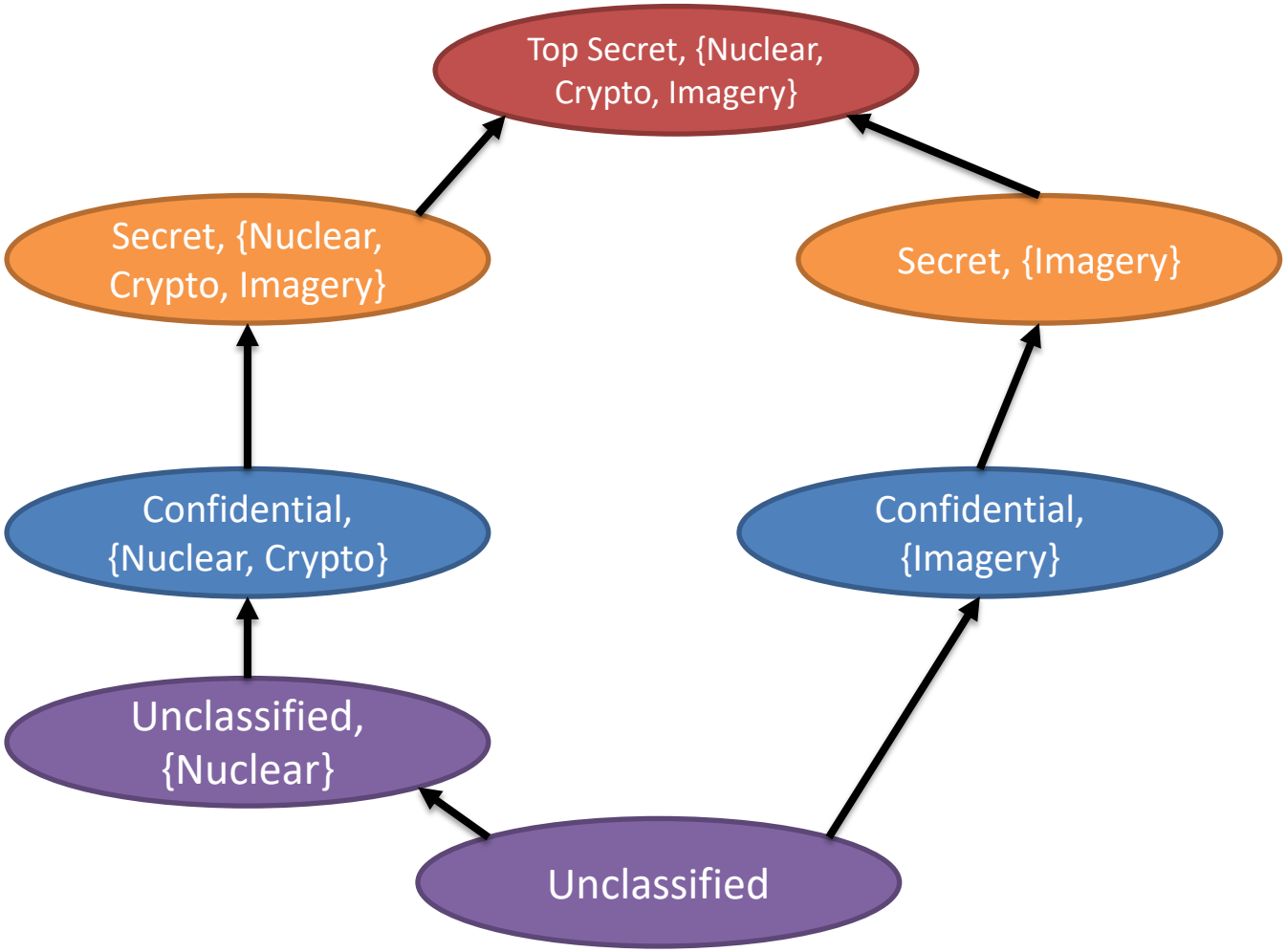
Label *objects* with the highest classification level of information they contain and their compartment

Now we can reason about information flows

Adapted from *The Craft of System Security* by Smith and Marchesini

Bell-LaPadula Model (BLM) formalizes MLS as Finite State Machine obeying three rules

Finite State Machine

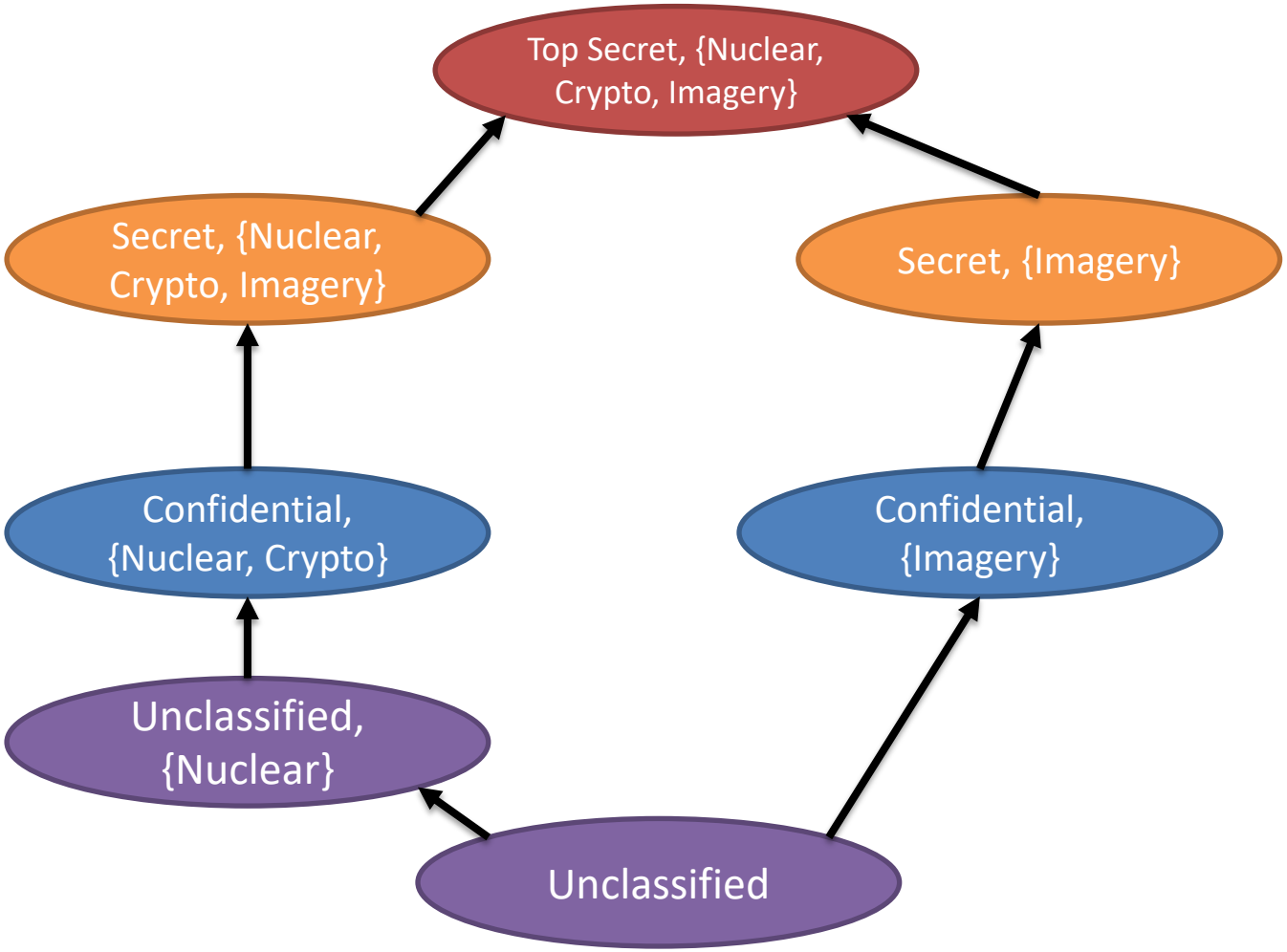


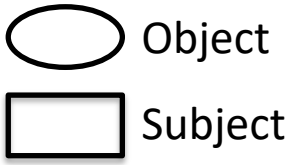
Model's focus is on confidentiality, not integrity or availability

Adapted from *The Craft of System Security* by Smith and Marchesini

Rule 1: No READ up

Finite State Machine





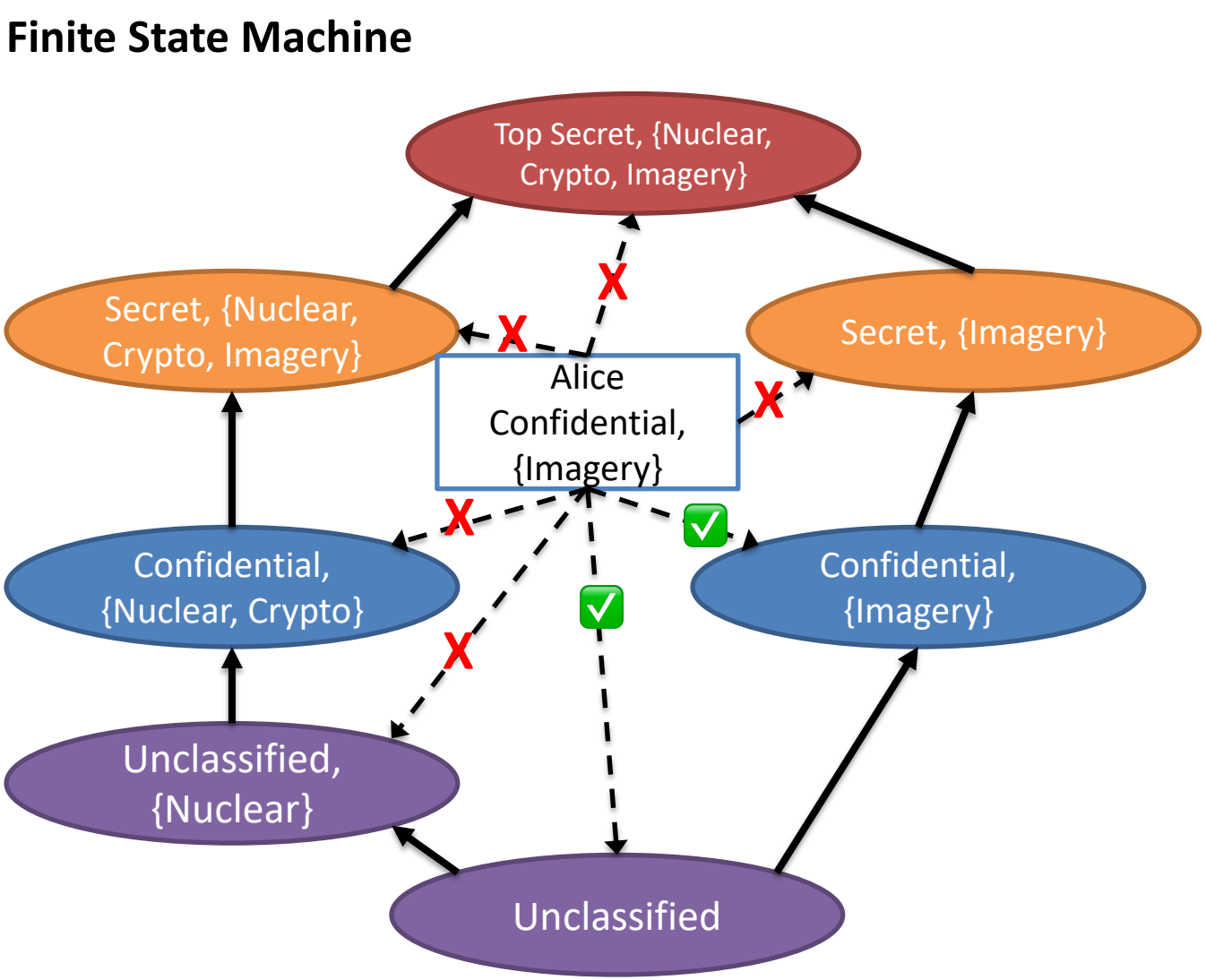
1) Simple Security Property

- A subject can READ an object only if the subject dominates the class of the object
- Dominates
  - $c_o \leq c_s$
  - $d_o \subseteq d_s$
- "No read up" rule

Adapted from *The Craft of System Security* by Smith and Marchesini

Rule 1: No READ up

Finite State Machine



1) Simple Security Property

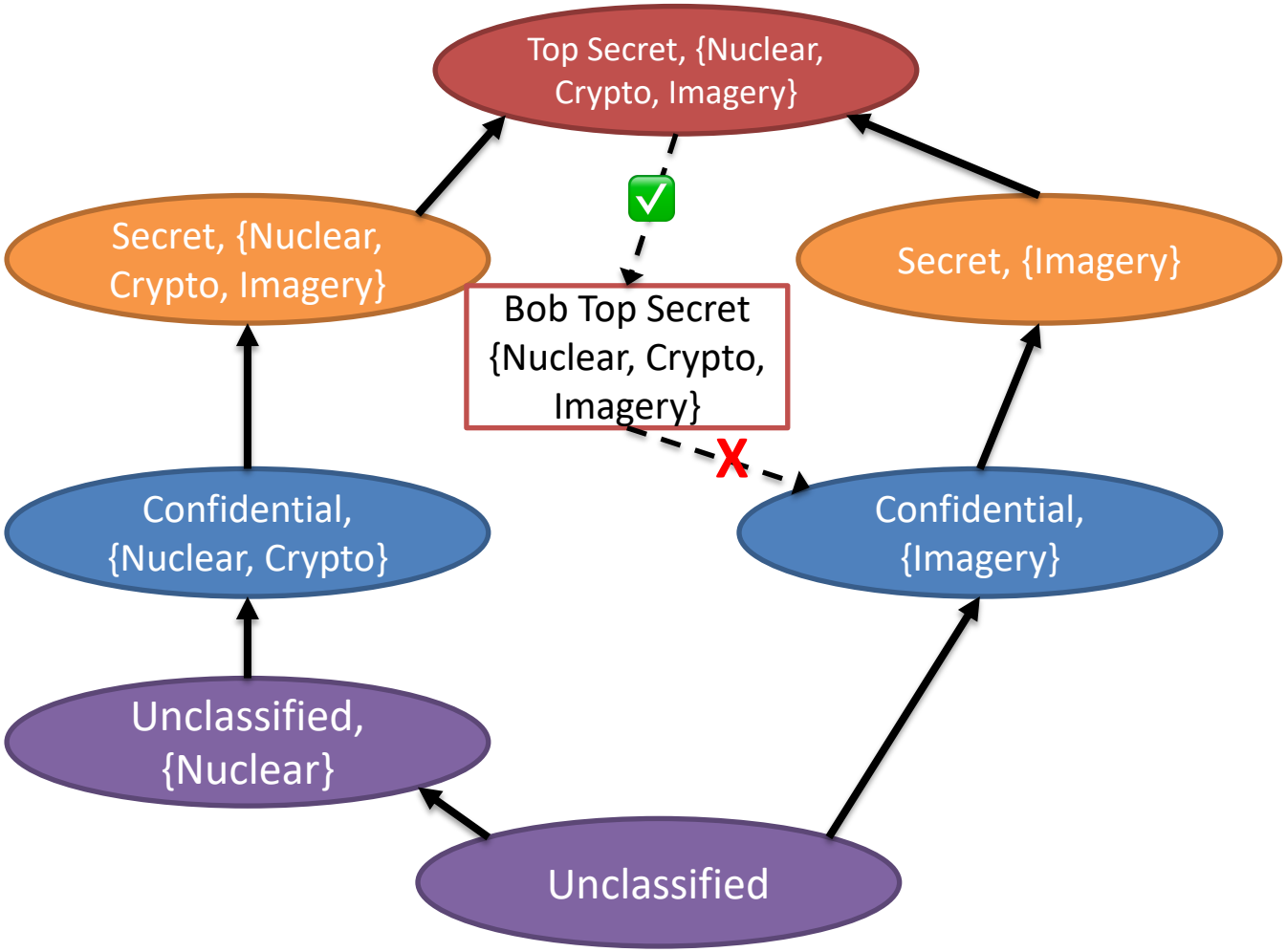
- A subject can READ an object only if the subject dominates the class of the object
- Dominates
  - $c_o \leq c_s$
  - $d_o \subseteq d_s$
- "No read up" rule

Alice can read Confidential Imagery
Cannot read Confidential {Nuclear, Crypto} or Secret { Imagery}

Adapted from *The Craft of System Security* by Smith and Marchesini

Rule 2: No WRITE down

Finite State Machine



2) *-Property

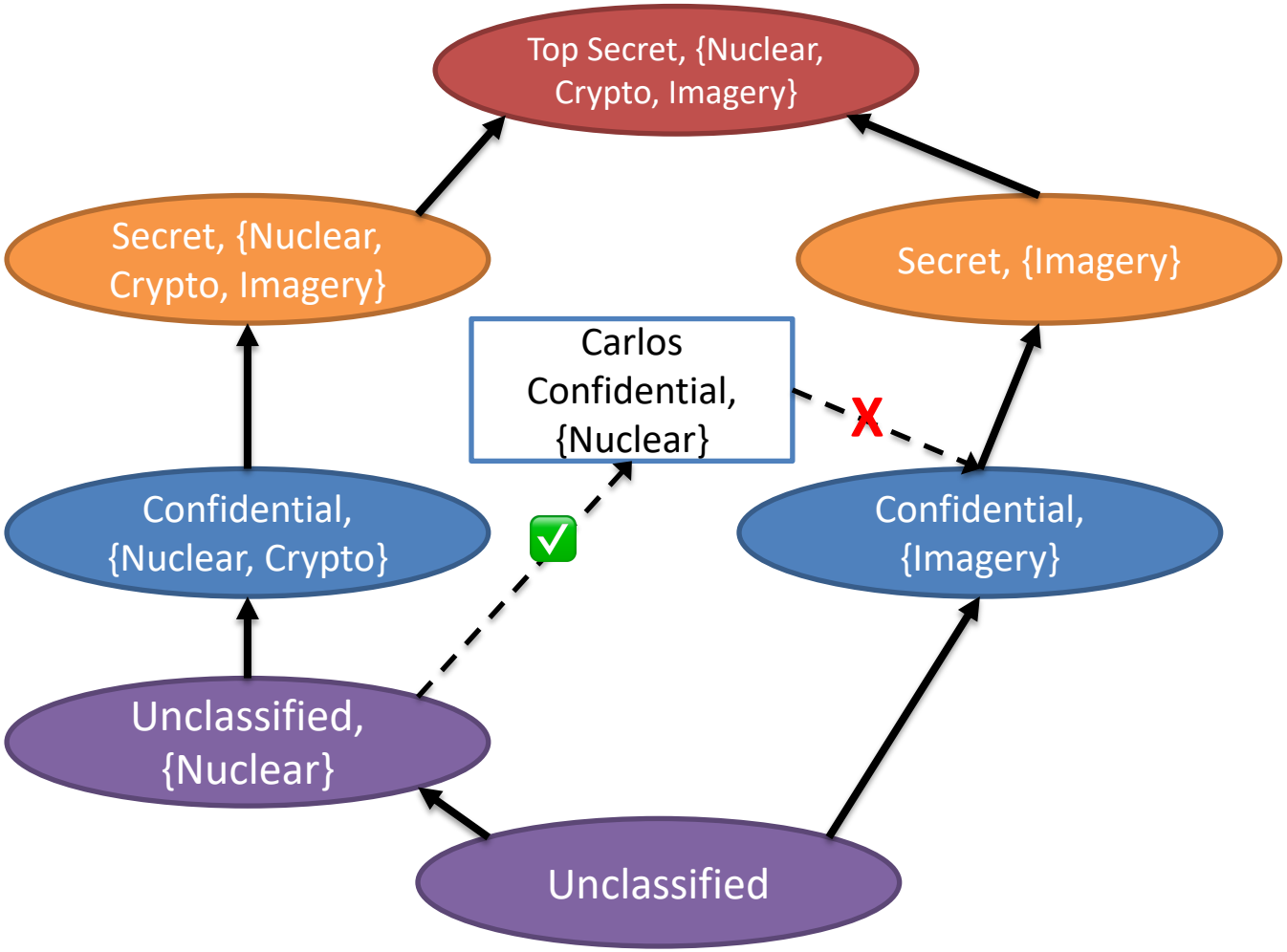
- If subject has READ access on O_1 and WRITE access on O_2, then O_2 must dominate O_1
- Dominates
  - $c_{o1} \leq c_{o2}$
  - $d_{o1} \subseteq d_{o2}$
- "No write down" rule (aka "high water mark")

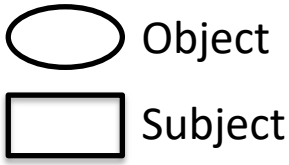
Bob can read Top secret info, but cannot write it to a Confidential object

Adapted from *The Craft of System Security* by Smith and Marchesini

Rule 2: No WRITE down

Finite State Machine





2) *-Property

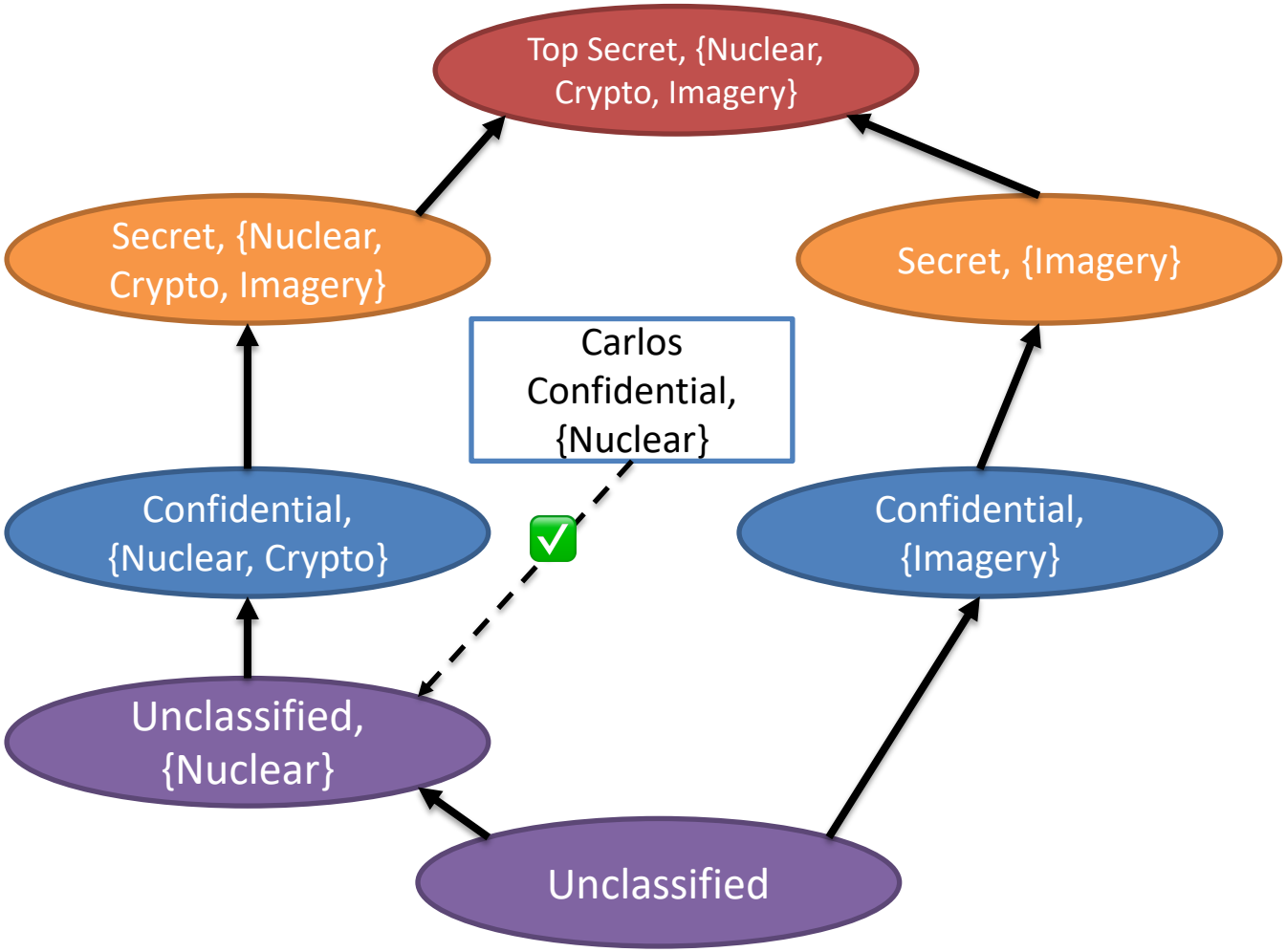
- If Subject has READ access on O_1 and WRITE access on O_2, then O_2 must dominate O_1
- Dominates
  - $c_{o1} \leq c_{o2}$
  - $d_{o1} \subseteq d_{o2}$
- "No write down" rule (aka "high water mark")

Carlos can read Unclassified, Nuclear but cannot write it to a Confidential, Imagery object

Adapted from *The Craft of System Security* by Smith and Marchesini

Rule 3: Discretionary security

Finite State Machine



Object

Subject

3) Discretionary security
A specific subject S can access object O only if that access is permitted in the current Access Control Matrix

Bell and LaPadula used this model to prove the *Basic Security Theorem*

- **If a system starts in secure state**
- **Follows the rules**
- **Then system remains in secure state**

Adapted from *The Craft of System Security* by Smith and Marchesini

The Bell-LaPadula Model has been criticized on several fronts

- Security level of objects remains static (no inherent ability to upgrade or downgrade classification levels)
- Properties of hierarchical access control don't effectively support the "need to know" principle which is often necessary outside the strict military operations in which this idea works best
- Strict emphasis on confidentiality. There are no inherent policies for changing access rights. No focus on integrity or availability
- Even with the emphasis on confidentiality, there exist covert channels by which a subject at a lower clearance may intuit the existence of high-level objects through the simple act of the subject's being denied access to them

Adapated from https://blog.finjan.com/access-control-systems-a-closer-look-at-the-bell-lapadula-model/

Often separate networks are established for information of varying sensitivity

Networks are "air gapped"

Details hard to find

Non-classified Internet Protocol Router Network (NIPRNet) for non-classified but sensitive information (FOUO)

Secret Internet Protocol Router Network (SIPRNET) for Secret data

Joint Worldwide Intelligence Communication System (JWICS) for Top Secret data

https://www.safeandvault.com/index.php/faq/115-gsa-containers/569-siprnet-network-procedure
https://www.bbc.com/news/world-us-canada-11863618
https://en.wikipedia.org/wiki/SIPRNet

Agenda

1. Access control
2. Multilevel Security (MLS)
3. Linux access controls
4. Malware

Every file in Linux has three types of different owners: Owner, Group, and Other

Owner (user)

- By default, the user that created a file is its owner

Group

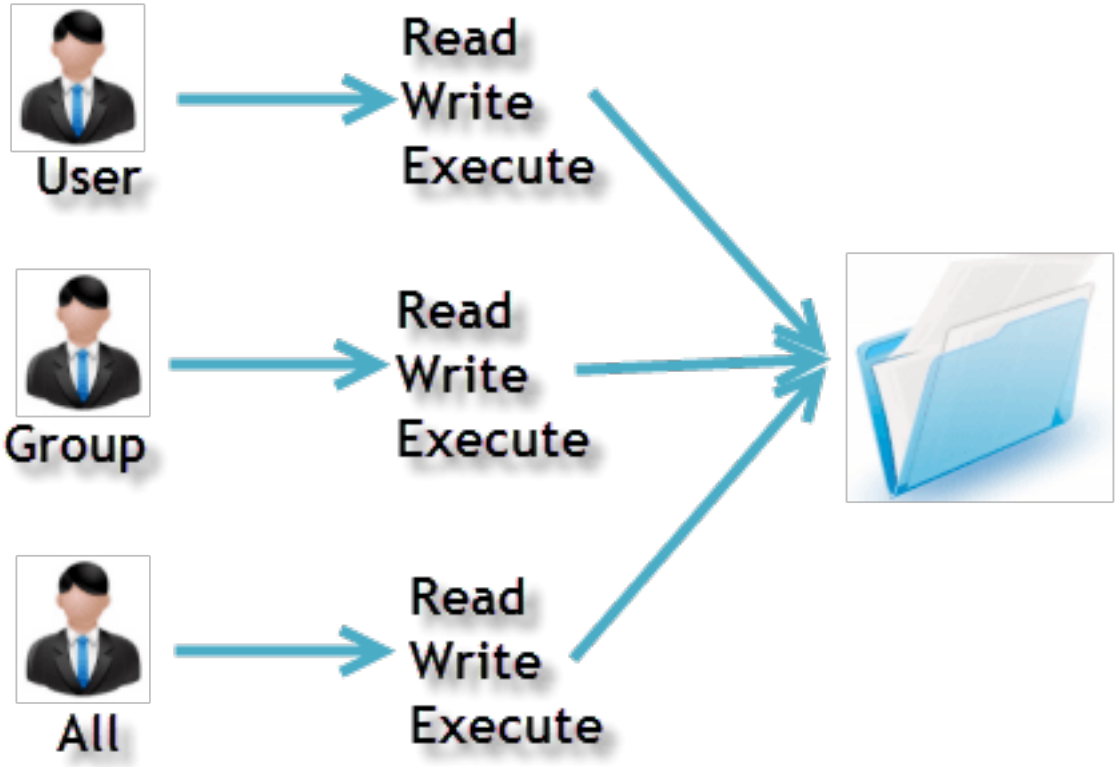
- A user group can contain multiple users
- All users belonging to a group will have the same permissions on a file

Other

- Any other user non-owner, non-group member who has access to file
- Sometimes called “world” because it is everyone else

Adapted from https://www.guru99.com/file-permissions.html

Linux defines three permissions: Read, Write, and Execute



Read

- Open and read a file
- Read on a directory lets you list its content

Write

- Ability to modify a file
- Write on directory lets you add, remove, and rename files If have Write permission on file, but not on directory file is in, can modify file, but not rename, move, or remove file from directory

Execute

- Can run program
- If Execute not set, might still be able to change program
- Can enter directory

Adapted from https://www.guru99.com/file-permissions.html

ls -l shows permissions on files and directories

Directory vs. file

```
[10/18/20]seed@VM:~/src/file_access_demo$ ls -l
total 4
drwxrwxr-x 2 seed seed 4096 Oct 18 20:10 test_directory
-rw-rw-r-- 1 seed seed    0 Oct 18 20:10 test.txt
[10/18/20]seed@VM:~/src/file_access_demo$
```

d indicates this is a directory
- indicates it is a file

<u>Permissions:</u>
r = read
w = write
x = execute
- = no permission

Adapted from https://www.guru99.com/file-permissions.html

Owner permissions are shown after directory indicator

Owner

```
[10/18/20]seed@VM:~/src/file_access_demo$ ls -l
total 4
drwxrwxr-x 2 seed seed 4096 Oct 18 20:10 test_directory
-rw-rw-r-- 1 seed seed    0 Oct 18 20:10 test.txt
[10/18/20]seed@VM:~/src/file_access_demo$
```

After directory flag, next three are read, write, and execute permissions for the owner of the file or directory

The owner has full read, write, execute permissions on the test_directory, but only read and write (no execute) on the text.txt file

The owner of test_directory and text.txt is the user *seed*

Adapted from https://www.guru99.com/file-permissions.html

Group permissions are shown after owner permissions

Group

```
[10/18/20]seed@VM:~/src/file_access_demo$ ls -l
total 4
drwxrwxr-x 2 seed seed 4096 Oct 18 20:10 test_directory
-rw-rw-r-- 1 seed seed    0 Oct 18 20:10 test.txt
[10/18/20]seed@VM:~/src/file_access_demo$
```

Like the owner, the group has full read, write, execute permissions on the test_directory, but only read and write (no execute) on the text.txt file

The group is *seed* (all users initially in their own group named after the user)

Adapted from https://www.guru99.com/file-permissions.html

Other (everyone else) permissions are shown after the group permissions

Other

```
[10/18/20]seed@VM:~/src/file_access_demo$ ls -l
total 4
drwxrwxr-x 2 seed seed 4096 Oct 18 20:10 test_directory
-rw-rw-r-- 1 seed seed    0 Oct 18 20:10 test.txt
[10/18/20]seed@VM:~/src/file_access_demo$
```

Other (everyone else) has read and execute permissions on the test_directory, but only read (no write or execute) on the text.txt file

Adapted from https://www.guru99.com/file-permissions.html

Linux uses a discretionary access control model; permissions changed with chmod

Change mode – chmod permissions filename

#	Permission Type	Symbol
0	No Permission	---
1	Execute	--x
2	Write	-w-
3	Execute+Write	-wx
4	Read	r--
5	Read+Execute	r-x
6	Read+Write	rw-
7	Read+Write+Execute	rwx

Can change permissions using absolute mode

```
[10/18/20]seed@VM:~/src/file_access_demo$ chmod 752 test.txt
[10/18/20]seed@VM:~/src/file_access_demo$ ls -l
total 4
drwxrwxr-x 2 seed seed 4096 Oct 18 20:10 test_directory
-rwxr-x-w- 1 seed seed    0 Oct 18 20:10 test.txt
```

On test.txt

- **Set Owner to 7 (rwx)**

Adapted from https://www.guru99.com/file-permissions.html

chmod can run in absolute mode

Change mode – chmod permissions filename

#	Permission Type	Symbol
0	No Permission	---
1	Execute	--x
2	Write	-w-
3	Execute+Write	-wx
4	Read	r--
5	Read+Execute	r-x
6	Read+Write	rw-
7	Read+Write+Execute	rwx

Can change permissions using absolute mode

```
[10/18/20]seed@VM:~/src/file_access_demo$ chmod 752 test.txt
[10/18/20]seed@VM:~/src/file_access_demo$ ls -l
total 4
drwxrwxr-x 2 seed seed 4096 Oct 18 20:10 test_directory
-rwxr-x-w- 1 seed seed    0 Oct 18 20:10 test.txt
```

On test.txt

- **Set Owner to 7 (rwx)**
- **Set Group to 5 (r-x)**

Adapted from https://www.guru99.com/file-permissions.html

chmod can run in absolute mode

Change mode – chmod permissions filename

#	Permission Type	Symbol
0	No Permission	---
1	Execute	--x
2	Write	-w-
3	Execute+Write	-wx
4	Read	r--
5	Read+Execute	r-x
6	Read+Write	rw-
7	Read+Write+Execute	rwx

Can change permissions using absolute mode

```
[10/18/20]seed@VM:~/src/file_access_demo$ chmod 752 test.txt
[10/18/20]seed@VM:~/src/file_access_demo$ ls -l
total 4
drwxrwxr-x 2 seed seed 4096 Oct 18 20:10 test_directory
-rwxr-x-w- 1 seed seed    0 Oct 18 20:10 test.txt
```

On test.txt

- **Set Owner to 7 (rwx)**
- **Set Group to 5 (r-x)**
- **Set Other to 2 (-w-)**

Adapted from https://www.guru99.com/file-permissions.html

chmod can run with symbolic mode

Change mode – chmod permissions filename

Can change permissions using **symbolic** mode

#	Permission Type	Symbol
0	No Permission	---
1	Execute	--x
2	Write	-w-
3	Execute+Write	-wx
4	Read	r--
5	Read+Execute	r-x
6	Read+Write	rw-
7	Read+Write+Execute	rwx

Op	Description
+	Add a permission
-	Remove a permission
=	Set (overwrite) a permission

User	Description
u	User/owner
g	Group
o	Other
a	All

```
[10/18/20]seed@VM:~/src/file_access_demo$ chmod o=rwx test.txt
[10/18/20]seed@VM:~/src/file_access_demo$ ls -l
total 4
drwxrwxr-x 2 seed seed 4096 Oct 18 20:10 test_directory
-rwxr-xrwx 1 seed seed    0 Oct 18 20:10 test.txt
[10/18/20]seed@VM:~/src/file_access_demo$ chmod g+w test.txt
[10/18/20]seed@VM:~/src/file_access_demo$ ls -l
total 4
drwxrwxr-x 2 seed seed 4096 Oct 18 20:10 test_directory
-rwxrwxrwx 1 seed seed    0 Oct 18 20:10 test.txt
[10/18/20]seed@VM:~/src/file_access_demo$ chmod u-r test.txt
[10/18/20]seed@VM:~/src/file_access_demo$ ls -l
total 4
drwxrwxr-x 2 seed seed 4096 Oct 18 20:10 test_directory
--wxrwxrwx 1 seed seed    0 Oct 18 20:10 test.txt
```

Overwrite other's permissions to rwx

Adapted from https://www.guru99.com/file-permissions.html

chmod can run with symbolic mode

Change mode – chmod permissions filename

Can change permissions using **symbolic** mode

#	Permission Type	Symbol
0	No Permission	---
1	Execute	--x
2	Write	-w-
3	Execute+Write	-wx
4	Read	r--
5	Read+Execute	r-x
6	Read+Write	rw-
7	Read+Write+Execute	rwx

Op	Description
+	Add a permission
-	Remove a permission
=	Set (overwrite) a permission

User	Description
u	User/owner
g	Group
o	Other
a	All

```
[10/18/20]seed@VM:~/src/file_access_demo$ chmod o=rwx test.txt
[10/18/20]seed@VM:~/src/file_access_demo$ ls -l
total 4
drwxrwxr-x 2 seed seed 4096 Oct 18 20:10 test_directory
-rwxr-xrwx 1 seed seed    0 Oct 18 20:10 test.txt
[10/18/20]seed@VM:~/src/file_access_demo$ chmod g+w test.txt
[10/18/20]seed@VM:~/src/file_access_demo$ ls -l
total 4
drwxrwxr-x 2 seed seed 4096 Oct 18 20:10 test_directory
-rwxrwxrwx 1 seed seed    0 Oct 18 20:10 test.txt
[10/18/20]seed@VM:~/src/file_access_demo$ chmod u-r test.txt
[10/18/20]seed@VM:~/src/file_access_demo$ ls -l
total 4
drwxrwxr-x 2 seed seed 4096 Oct 18 20:10 test_directory
--wxrwxrwx 1 seed seed    0 Oct 18 20:10 test.txt
```

Add write permission to group

Adapted from https://www.guru99.com/file-permissions.html

chmod can run with symbolic mode

Change mode – chmod permissions filename

Can change permissions using **symbolic** mode

#	Permission Type	Symbol
0	No Permission	---
1	Execute	--x
2	Write	-w-
3	Execute+Write	-wx
4	Read	r--
5	Read+Execute	r-x
6	Read+Write	rw-
7	Read+Write+Execute	rwx

Op	Description
+	Add a permission
-	Remove a permission
=	Set (overwrite) a permission

User	Description
u	User/owner
g	Group
o	Other
a	All

```
[10/18/20]seed@VM:~/src/file_access_demo$ chmod o=rwx test.txt
[10/18/20]seed@VM:~/src/file_access_demo$ ls -l
total 4
drwxrwxr-x 2 seed seed 4096 Oct 18 20:10 test_directory
-rwxr-xrwx 1 seed seed    0 Oct 18 20:10 test.txt
[10/18/20]seed@VM:~/src/file_access_demo$ chmod g+w test.txt
[10/18/20]seed@VM:~/src/file_access_demo$ ls -l
total 4
drwxrwxr-x 2 seed seed 4096 Oct 18 20:10 test_directory
-rwxrwxrwx 1 seed seed    0 Oct 18 20:10 test.txt
[10/18/20]seed@VM:~/src/file_access_demo$ chmod u-r test.txt
[10/18/20]seed@VM:~/src/file_access_demo$ ls -l
total 4
drwxrwxr-x 2 seed seed 4096 Oct 18 20:10 test_directory
--wxrwxrwx 1 seed seed    0 Oct 18 20:10 test.txt
```

Remove read permission from user

Adapted from https://www.guru99.com/file-permissions.html

In Linux we can change the owner with chown and change the group with chgrp

Change owner – chown

```
[10/18/20]seed@VM:~/src/file_access_demo$ sudo chown root test.txt
[10/18/20]seed@VM:~/src/file_access_demo$ ls -l
total 4
drwxrwxr-x 2 seed seed 4096 Oct 18 20:10 test_directory
-rwxrwxrwx 1 root seed    0 Oct 18 20:10 test.txt
```

Use sudo to change owner to root (does not change group)

Change group - chgrp

```
[10/18/20]seed@VM:~/src/file_access_demo$ sudo chgrp cdrom test.txt
[10/18/20]seed@VM:~/src/file_access_demo$ ls -l
total 4
drwxrwxr-x 2 seed seed  4096 Oct 18 20:10 test_directory
-rwxrwxrwx 1 root cdrom    0 Oct 18 20:10 test.txt
```

Use sudo to change group to cdrom (does not change owner)

<u>Miscellaneous tips</u>

Use command "groups" to get a list of all groups

Two groups cannot own the same file

No nested groups in Linux

x permission on a directory allows you to enter a directory and possible sub directories

Adapted from https://www.guru99.com/file-permissions.html

SetUID allows a program to run with permissions of file's owner

SetUID

```
$ id
uid=1000(seed) gid=1000(seed) groups=1000(seed),4(adm),24(cdrom),27(sudo),<snip>

$ which sudo
/usr/bin/sudo

$ ls -l /usr/bin/sudo
-rwsr-xr-x 1 root root 159852 Jan 20  2017 /usr/bin/sudo
```

User is seed
Owner of sudo is root
Sudo is a SetUID program (has s, not x)
Users can run sudo as file's owner (root)

Adapted from https://www.liquidweb.com/kb/how-do-i-set-up-setuid-setgid-and-sticky-bits-on-linux/

To create a program to run as root first change owner and group to root

SetUID

```
$ touch temp
$ sudo chown root temp
$ sudo chgrp root temp
$ ls -l
-rw-rw-r-- 1 root root     0 Jan 14 17:58 temp
```

Change to owner and group with

sudo chown root temp.txt

Sudo chgrp root temp.txt

Adapted from https://www.liquidweb.com/kb/how-do-i-set-up-setuid-setgid-and-sticky-bits-on-linux/

Next add SetUID to file by setting privileges starting with 4

SetUID

```
$ touch temp
$ sudo chown root temp
$ sudo chgrp root temp
$ ls -l
-rw-rw-r-- 1 root root     0 Jan 14 17:58 temp

$ sudo chmod 4755 temp
$ ls -l
-rwsr-xr-x 1 root root     0 Jan 14 17:58 temp
```

Change to owner and group with
sudo chown root temp.txt
Sudo chgrp root temp.txt

Make SetUID with:
sudo chmod 4755 temp
Note: 4 sets x to s

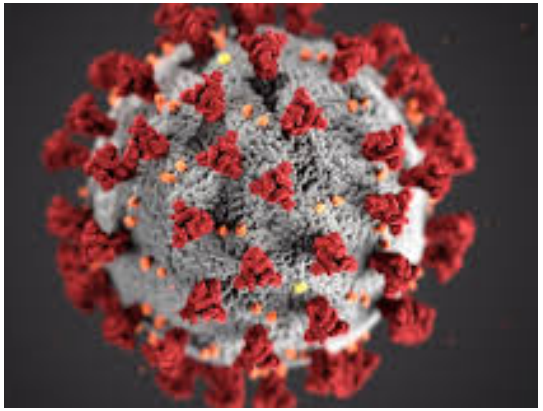
The ability to run with privileges of another user will be key in buffer overflow attacks
Goal: find vulnerable SetUID program owned by root to launch a shell (shell then runs as root)

Adapted from https://www.liquidweb.com/kb/how-do-i-set-up-setuid-setgid-and-sticky-bits-on-linux/

Agenda

1. Access control
2. Multilevel Security (MLS)
3. Linux access controls
4. Malware

Virus and worms are malware that can reproduce themselves



Viruses/worms

- Often sent via email or file upload
- Users must execute the virus program
- On execution
  - Infects computer
  - Reproduces through file system or network
  - Tries to infects other computers
- May not be destructive (used for pentesting)
- New viruses discovered all the time
- Worms can spread itself, without human intervention

Update antivirus!

IPS's can sometimes stop worms and viruses

Some viruses run in:

- Boot sector (doesn't infect OS)
- Scripts in OS or browser
- Macros in Microsoft Office (using VBA) or other programs

Adapted from https://www.professormesser.com/security-plus/sy0-501/viruses-and-worms-3/

Ransomware encrypts data, asks for payment in exchange for decryption key

Ransomware

- Your most valuable asset is your data
- Ransomware tries to make data unreadable by encrypting data
- Does not encrypt operating system (so computer keeps working)
- May say FBI has locked your computer or something official sounding
- Asks for payment (typically bitcoin) for decryption key
- We will see how encryption works next week

If you pay ransom, does it go away?
Are criminals honorable?

Antivirus may stop it

Make sure you have offline backups!

Modern ransomware looks for online backup!

Adapted from https://www.professormesser.com/security-plus/sy0-501/viruses-and-worms-3/

A trojan is malicious software that pretends to be something it is not

Trojan

- Tries to trick you into running it by appearing to be something it is not
- Often reaches out to Command-and-Control server
- Downloads more malware

Remote Access Trojan (RAT)

- Embeds itself into Operating System
- Allows outsider to have remote access to system
  - Files
  - Camera
  - Microphone

Rootkits modify the kernel of the Operating System

Rootkit

- Invisible because part of Operating System
- Task Manager or other won't show it running
- Antivirus cannot see it
- Particularly nasty!

Keyloggers attempt to steal each keystroke

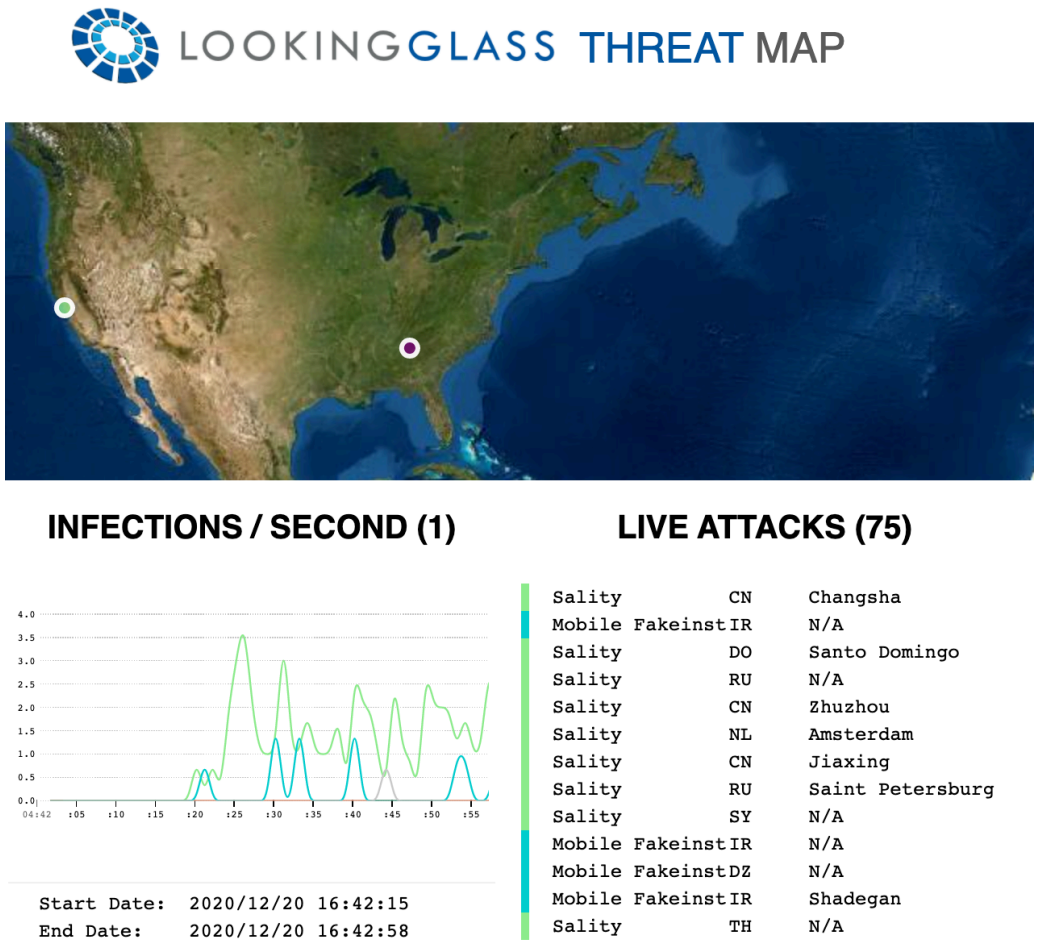
Keylogger

- Capture keystrokes so no encryption
- Can sometimes store screenshots
- Send data to adversaries
- Sometimes hidden as hardware device that goes between keyboard and computer (”bump in the line”)
- Often a software program

Botnets command large numbers of devices via Command-and-Control server

Botnet

- "Robot network"
- Sometimes installed via trojan or virus
- Sometimes install with default passwords
- Wait for command from Command-and-Control server
- Can rent time on botnet for DOS attack
- Example: Mirai botnet
- Real time view: https://map.lookingglasscyber.com/



Logic bombs are malware that execute when an event occurs

- **Supply chain attacks try to embed malware during the manufacture or distribution process**
- **Sometimes embedded in hardware ("chipping")**

Logic bomb

- Software installed (often by admin)
- Waits for a condition
  - Day/time
  - Check that admin is still employed
- Takes action to cause severe damage

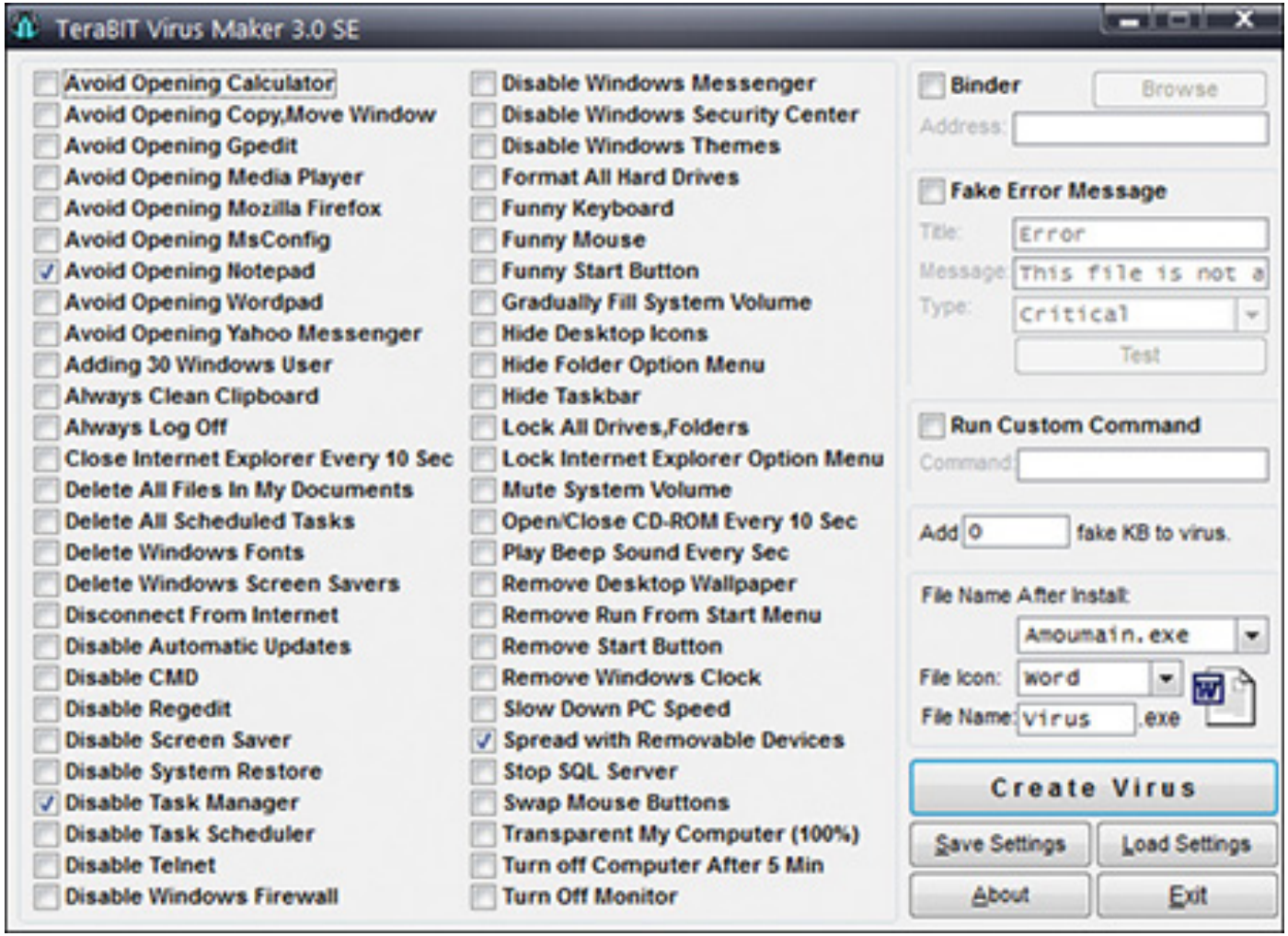
Ukraine power system taken down on Dec 17, 2016 at 11:53pm

- Logic bomb turned off electrical circuits
- Bomb written for SCADA networks

Adapted from https://www.professormesser.com/security-plus/sy0-501/logic-bombs-3/

There are several GUI-based virus creators that require no skill to make new viruses

Terabit Virus Maker



Click the properties you'd like, and virus maker creates a new virus for you

Useful for penetration testing!

Easttom II, William Chuck. *Penetration testing fundamentals: A hands-on guide to reliable security audits*. Pearson IT Certification, 2018.

Pierson's imperfect method of patching an organization

Modified "patch and pray"

- New patches come out frequently
- Patch Tuesday (second Tues in month) -> exploit Wednesday
- Could apply them across your organization and hope updates don't break anything (called "patch and pray")

Instead roll out over a couple of days

- Choose the least time-dependent part of the organization
- Roll out patch to half of that group on day 1
- See if anything breaks
- Next day
  - Complete patch of first part of organization
  - Roll out patches to half of another of another group in organization
- Continue until whole organization covered
- This approach is imperfect to be sure!

<u>Significant</u> problems:

- **Parts of organization exposed for a few days**
- **Problems for one group may not affect another**
- **If patch causes problem, taking down half a group may still be a big issue!**