Cyber Security

Operating system security & access control

Chris G. Willcocks
Durham University

Lecture Overview

- Access Control
  - ACMs
  - ACLs
- Introduction to *NIX security - we'll cover this in detail due to high-end server popularity
  - https://www.exploit-db.com/search?platform=linux
- Briefly on Windows security:
  - https://www.exploit-db.com/search?platform=windows
- Confidentiality models
- Integrity models
- Briefly on security evaluation
- Protection rings

Access Control

- Your computer contains lots of **subjects** (typically users, people) and lots of **objects** (typically documents, images, programs).
- Who chooses access rights?
    - The file owner?
        - Mandatory access control (MAC)
    - The system owner?
        - Discretionary access control (DAC)
    - Anyone who has rights
- What/how/where do we store access permissions?
    - Multiple approaches

Access Control Matrix (ACM)

+ Easy to define, easy to verify
- Poor scalability, poor handling of changes, could get corrupted.

Objects

Subjects

	bill.doc	**readme.txt**	**edit.exe**	**func.sh**
Alice	-	*{read}*	*{execute}*	*{execute, read}*
Chris	*{read, append}*	*{read}*	*{execute, read, write}*	*{execute}*
Greg	-	*{read}*	*{execute, write}*	-
Jess	*{read}*	*{read, write}*	-	-

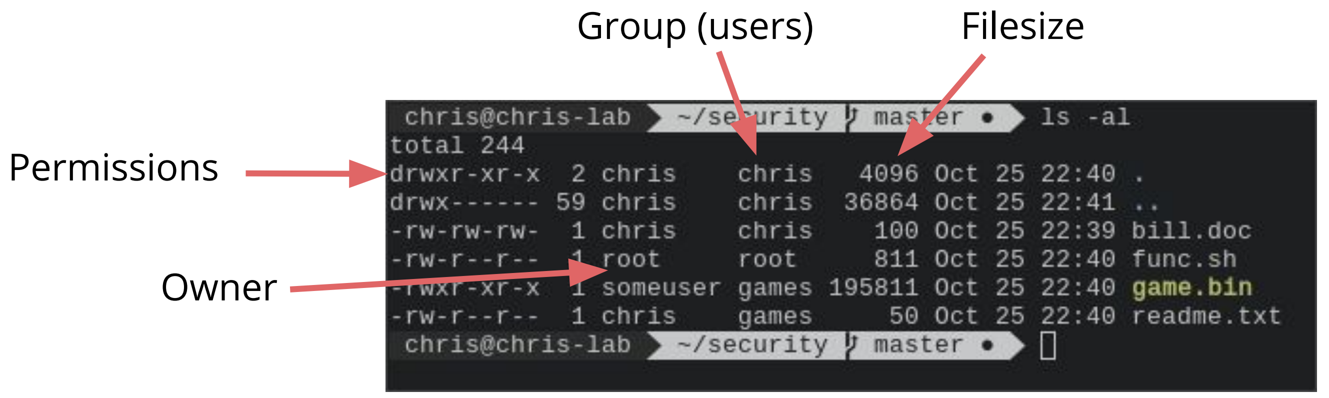
Access Permissions

*NIX has 8 access permission settings for 3 types of users:

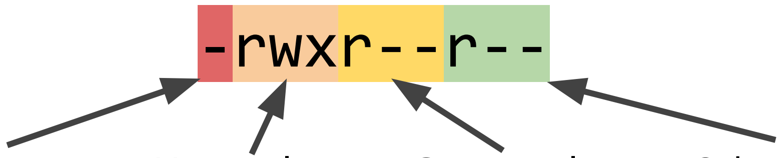
- Owners, Groups, and Others
- Combination of read (r), write (w), and execute (x)
- Represented as numbers in base 8

```
---  all types of access denied
--x  execute access only
-w-  write access only
-wx  write and execute only
r--  read only
r-x  read and execute only
rw-  read and write access only
rwx  everything allowed
```

*NIX Permissions

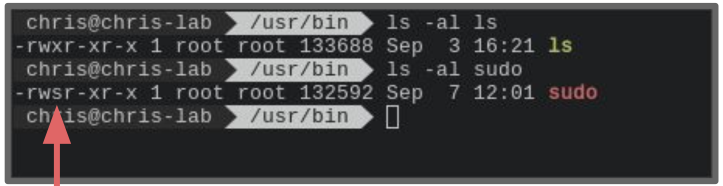


```
 chris@chris-lab  ~/security  master ●  ls -al
total 244
drwxr-xr-x  2 chris    chris    4096 Oct 25 22:40 .
drwx------ 59 chris    chris   36864 Oct 25 22:41 ..
-rw-rw-rw-  1 chris    chris     100 Oct 25 22:39 bill.doc
-rw-r--r--  1 root     root      811 Oct 25 22:40 func.sh
-rwxr-xr-x  1 someuser games  195811 Oct 25 22:40 game.bin
-rw-r--r--  1 chris    games      50 Oct 25 22:40 readme.txt
 chris@chris-lab  ~/security  master ● 
```



Resource type	User rules	Group rules	Other rules
`- file`	`r read`	`r read`	`r read`
`d directory`	`w write`	`w write`	`w write`
`b block device`	`x execute`	`x execute`	`x execute`
`c character device`			
`s socket`			
`l symbolic link`			
`p FIFO`			

setuid, setgid, and sticky bits

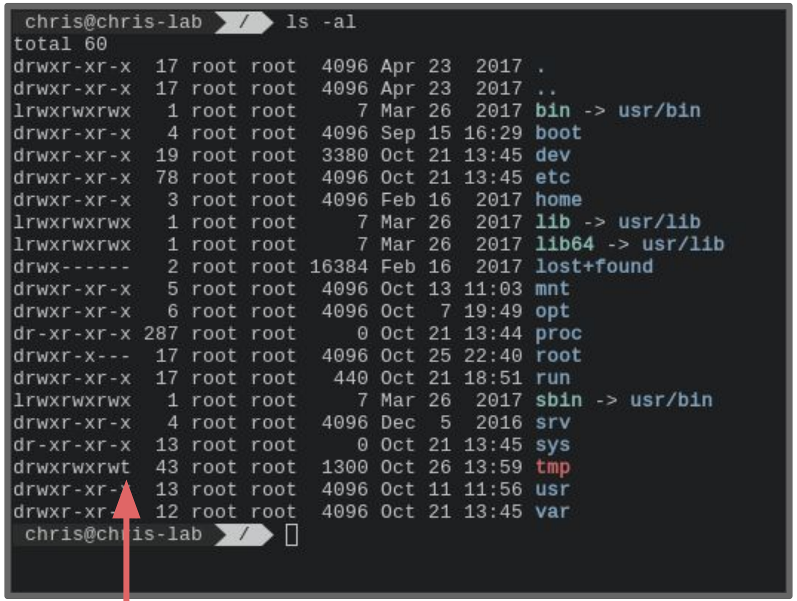


```
 chris@chris-lab  /usr/bin  ls -al ls
-rwxr-xr-x 1 root root 133688 Sep  3 16:21 ls
 chris@chris-lab  /usr/bin  ls -al sudo
-rwsr-xr-x 1 root root 132592 Sep  7 12:01 sudo
 chris@chris-lab  /usr/bin 
```

setuid bit: users run executable with permissions of the executable's owner

Further reading:
https://wiki.archlinux.org/index.php/File_permissions_and_attributes

Setuid hacks:
https://gist.github.com/dergachev/7916152
https://null-byte.wonderhowto.com/how-to/hack-like-pro-finding-potential-suid-sgid-vulnerabilities-linux-unix-systems-0158373/



```
 chris@chris-lab  /  ls -al
total 60
drwxr-xr-x  17 root root  4096 Apr 23  2017 .
drwxr-xr-x  17 root root  4096 Apr 23  2017 ..
lrwxrwxrwx   1 root root     7 Mar 26  2017 bin -> usr/bin
drwxr-xr-x   4 root root  4096 Sep 15 16:29 boot
drwxr-xr-x  19 root root  3380 Oct 21 13:45 dev
drwxr-xr-x  78 root root  4096 Oct 21 13:45 etc
drwxr-xr-x   3 root root  4096 Feb 16  2017 home
lrwxrwxrwx   1 root root     7 Mar 26  2017 lib -> usr/lib
lrwxrwxrwx   1 root root     7 Mar 26  2017 lib64 -> usr/lib
drwx------   2 root root 16384 Feb 16  2017 lost+found
drwxr-xr-x   5 root root  4096 Oct 13 11:03 mnt
drwxr-xr-x   6 root root  4096 Oct  7 19:49 opt
dr-xr-xr-x 287 root root     0 Oct 21 13:44 proc
drwxr-x---  17 root root  4096 Oct 25 22:40 root
drwxr-xr-x  17 root root   440 Oct 21 18:51 run
lrwxrwxrwx   1 root root     7 Mar 26  2017 sbin -> usr/bin
drwxr-xr-x   4 root root  4096 Dec  5  2016 srv
dr-xr-xr-x  13 root root     0 Oct 21 13:45 sys
drwxrwxrwt  43 root root  1300 Oct 26 13:59 tmp
drwxr-xr-x  13 root root  4096 Oct 11 11:56 usr
drwxr-xr-x  12 root root  4096 Oct 21 13:45 var
 chris@chris-lab  / 
```

sticky bit: prevents users with write/execute permissions from deleting the directory contained files

*NIX Permissions to ACM

```
-rw-r----- 2 chris jess    2278  13  Oct 07:40  bill.doc
-rwx-wx--x 2 chris games   340   28  Oct 01:25  game.bin
-r-x--x--- 2 alice fun     748   1   Oct 21:43  func.sh
-rw----r-- 1 jess  jess    170   1   Oct 20:34  readme.txt
```

	bill.doc	game.bin	func.sh	readme.txt
Alice	-	{execute}	{read, execute}	{read}
Chris	{read, write}	{read, write, execute}	{execute}	{read}
Greg	-	{write, execute}	-	{read}
Jess	{read}	{execute}	-	{read, write}

Groups:
```
fun: chris
games: greg
jess: jess
```

Link Vulnerabilities

- Add new path to an inode.
- Multiple names for a single inode.
- For example, to overwrite /etc/passwd:

```
ln -s /etc/passwd file
./trusted_dump file < *passwd-entry*
```

e.g. a command which can read/write root owned files, but doesn't know the file is /etc/passwd

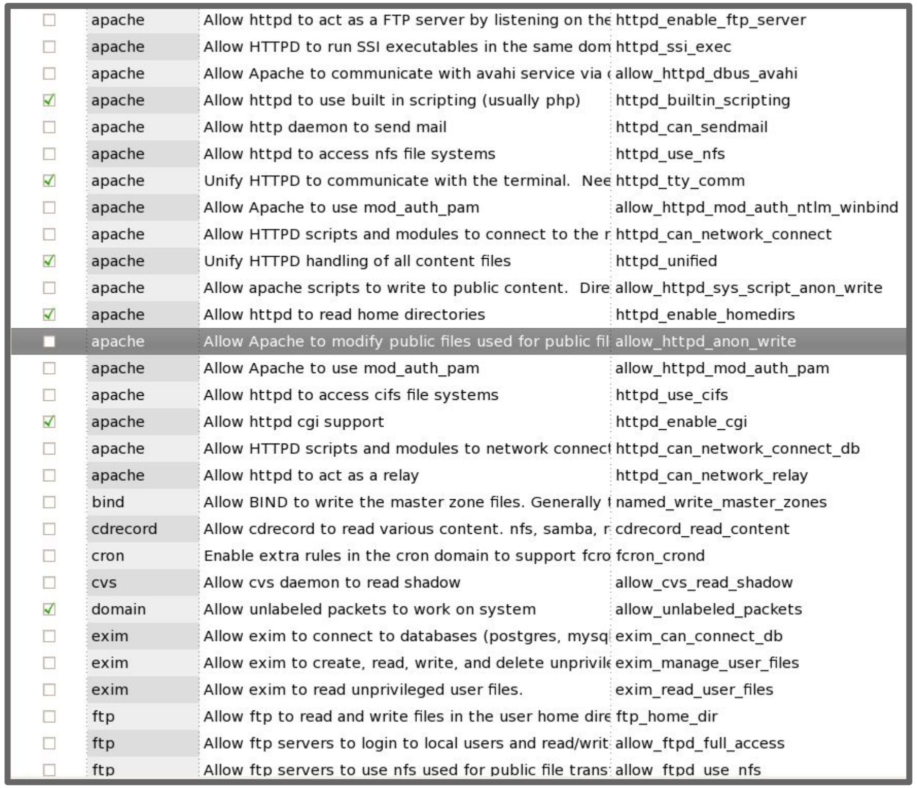
- Programs have to be aware of which files they are using.

`O_NOFOLLOW` flag can be added to prevent following links, e.g. "`open(file, O_NOFOLLOW, mode)`"

Hardening

- SELinux
  - Make sure that programs only access what they're meant to
    - Hard to use in practise
- AppArmor
  - Similar/simpler to SELinux

- Slightly off-topic but will mention here:
  - ASLR
    - Randomize memory address (ret2libc)
  - PaX
    - Executable space protection



☐	apache	Allow httpd to act as a FTP server by listening on the	httpd_enable_ftp_server
☐	apache	Allow HTTPD to run SSI executables in the same dom	httpd_ssi_exec
☐	apache	Allow Apache to communicate with avahi service via (	allow_httpd_dbus_avahi
✓	apache	Allow httpd to use built in scripting (usually php)	httpd_builtin_scripting
☐	apache	Allow http daemon to send mail	httpd_can_sendmail
☐	apache	Allow httpd to access nfs file systems	httpd_use_nfs
✓	apache	Unify HTTPD to communicate with the terminal. Nee	httpd_tty_comm
☐	apache	Allow Apache to use mod_auth_pam	allow_httpd_mod_auth_ntlm_winbind
☐	apache	Allow HTTPD scripts and modules to connect to the r	httpd_can_network_connect
✓	apache	Unify HTTPD handling of all content files	httpd_unified
☐	apache	Allow apache scripts to write to public content. Dire	allow_httpd_sys_script_anon_write
✓	apache	Allow httpd to read home directories	httpd_enable_homedirs
☐	apache	Allow Apache to modify public files used for public fil	allow_httpd_anon_write
☐	apache	Allow Apache to use mod_auth_pam	allow_httpd_mod_auth_pam
☐	apache	Allow httpd to access cifs file systems	httpd_use_cifs
✓	apache	Allow httpd cgi support	httpd_enable_cgi
☐	apache	Allow HTTPD scripts and modules to network connect	httpd_can_network_connect_db
☐	apache	Allow httpd to act as a relay	httpd_can_network_relay
☐	bind	Allow BIND to write the master zone files. Generally t	named_write_master_zones
☐	cdrecord	Allow cdrecord to read various content. nfs, samba, r	cdrecord_read_content
☐	cron	Enable extra rules in the cron domain to support fcro	fcron_crond
☐	cvs	Allow cvs daemon to read shadow	allow_cvs_read_shadow
✓	domain	Allow unlabeled packets to work on system	allow_unlabeled_packets
☐	exim	Allow exim to connect to databases (postgres, mysq	exim_can_connect_db
☐	exim	Allow exim to create, read, write, and delete unprivile	exim_manage_user_files
☐	exim	Allow exim to read unprivileged user files.	exim_read_user_files
☐	ftp	Allow ftp to read and write files in the user home dire	ftp_home_dir
☐	ftp	Allow ftp servers to login to local users and read/writ	allow_ftpd_full_access
☐	ftp	Allow ftp servers to use nfs used for public file trans	allow_ftpd_use_nfs

Device File Vulnerabilities

- Devices are represented as files
  - /dev/tty terminal
  - /dev/mem physical memory
  - /dev/kmem virtual memory
  - /dev/mouse mouse
- Created using mknod (only accessible by root)
  - Can bypass access control by getting access to memory
    - /dev/kmem or /dev/mem used to be “world” (other) accessible
- Can get access to user inputs
  - /dev/tty
    - See passwords, set keys
    - mesg n - prevents write access to current terminal

Access Control Lists (ACL)

- Store by column (object-focused):

+ Easy to view object access control, easy to remove access rights if object removed
- Poor overview of access rights per subject, difficult to remove subject.

	bill.doc	game.bin	func.sh	readme.txt
Alice	-	{execute}	{read, execute}	{read}
Chris	{read, write}	{read, write, execute}	{execute}	{read}
Greg	-	{write, execute}	-	{read}
Jess	{read}	{execute}	-	{read, write}

ACL:

```
bill.doc      {Chris: read, write}, {Jess: read}
game.bin      {Alice: execute}, {Chris: read, write, execute},
              {Greg: write, execute}, {Jess: execute}
func.sh       {Alice: read, execute}, {Chris: execute}
readme.txt    {Alice: read}, {Chris: read}, {Greg: read},
              {Jess: read}
```

Capability-based Security

- Store by row (subject-focused):

+ Easy to transfer ownership, easy inheritance of access rights.

- Poor overview of access rights per object, difficulty of revocation of object.

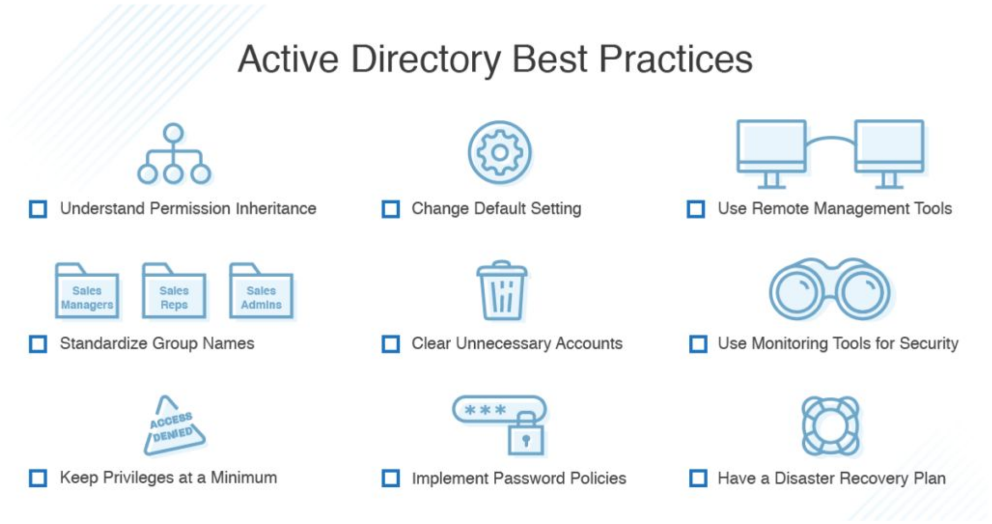
	bill.doc	game.bin	func.sh	readme.txt
Alice	-	{execute}	{read, execute}	{read}
Chris	{read, write}	{read, write, execute}	{execute}	{read}
Greg	-	{write, execute}	-	{read}
Jess	{read}	{execute}	-	{read, write}

Capabilities:

```
Alice         {game.bin: execute}, {func.sh: read, execute},
              {readme.txt: read}
Chris         {bill.doc: read,write}, {game.bin: read, write, execute},
              {func.sh: execute}, {readme.txt: read}
Greg          {game.bin: write, execute}, {readme.txt: read}
Jess          {bill.doc: read}, {game.bin: execute}, {readme.txt: read,write}
```

Windows

- Windows registry
  - Core place for system control
  - Target for hackers
  - Controls multiple computers
- Windows domain + AD
  - Computers sharing things such as passwords
- Principles:
  - SAM format - old but used in most places
  - UPN - more modern
- Login - happens in different ways depending if computer is alone or part of a network
- More levels than *NIX
  - Hardware, System, Administrator, Users


Active Directory Best Practices
Understand Permission Inheritance
Change Default Setting
Use Remote Management Tools
Sales Managers
Sales Reps
Sales Admins
Standardize Group Names
Clear Unnecessary Accounts
Use Monitoring Tools for Security
ACCESS DENIED
Keep Privileges at a Minimum
***
Implement Password Policies
Have a Disaster Recovery Plan

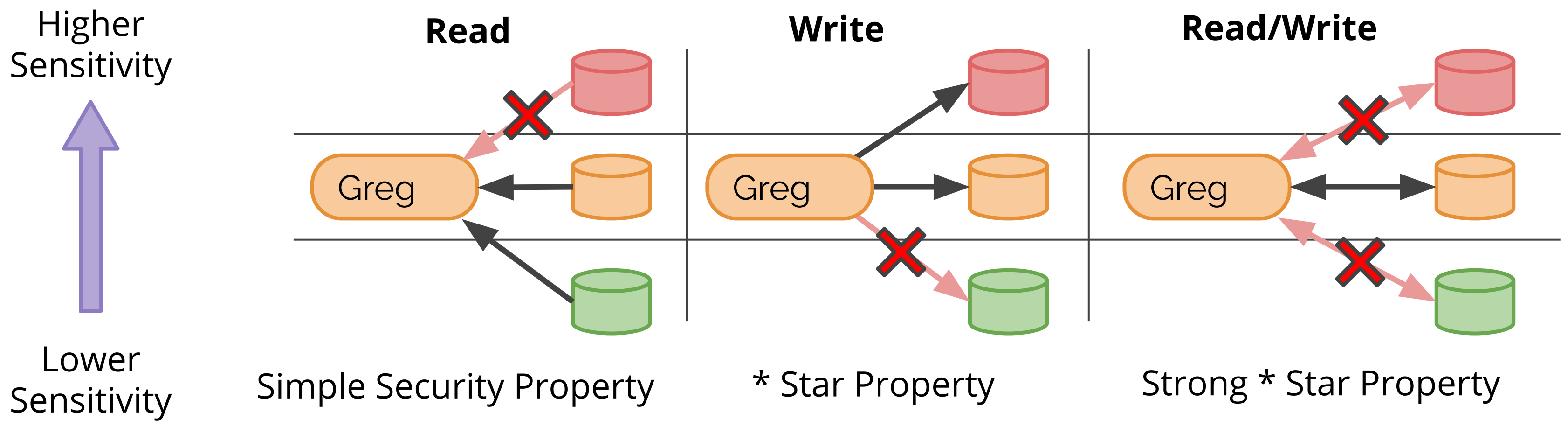
Windows

- Library loading is a problem.
- Viruses are very common and easy.
- Windows adding features to make OS less predictable
  - Image randomization (OS boots in one of 256 configurations)
  - Services restart if failed (not the best practise for security):
    - Vista+ sets some critical services to only restart twice, then manual restart required giving attackers just 2 attempts
- NTFS is much more secure than FAT32 & DOS.
  - Adds two ACLs:
    - DACL: Reading, writing, executing, deleting by which users or groups.
    - SCAL: for defining which actions are audited/logged, e.g. on activity being successful/failed.
  - Compression, encryption.

Link to some Windows security resources and attack vectors for further study

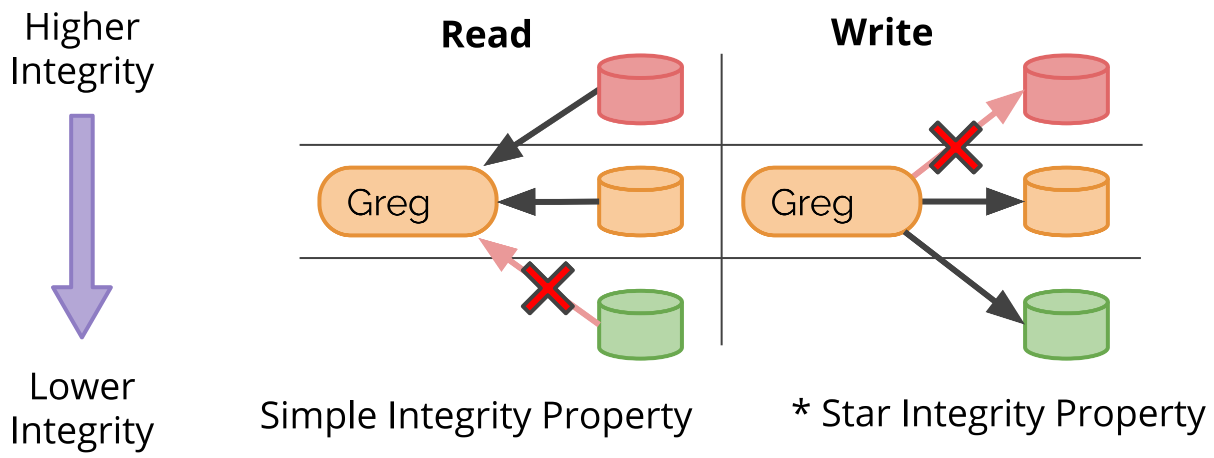
Bell-LaPadula Model

- Bell-LaPadula confidentiality policy, **"read down, write up"**
  - Simple security property
    - Subject (Greg) cannot read object of higher sensitivity
  - Star property (* property)
    - Subject cannot write to object of lower sensitivity.
  - Strong star property (Strong * property)
    - Subject cannot read/write to object of higher/lower sensitivity.



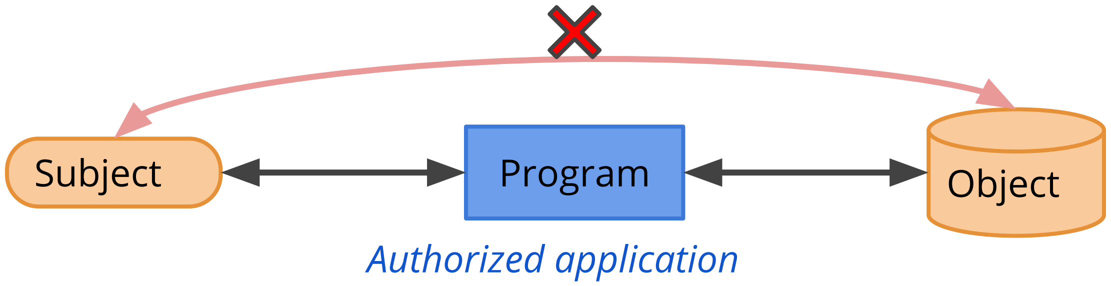
Biba Integrity Model

- Biba integrity model - **“read up, write down”**
  - Simple security property
    - Subject (Greg) cannot read object of lower integrity
  - Star property (* property)
    - Subject cannot write to object of higher integrity.
  - Invocation property
    - Subject/process cannot request higher integrity access.



Clark-Wilson Integrity Model

- Bell-LaPadula is good for **confidential** systems
- Biba is good for **integrity-preserving** systems
- What about businesses/industry processes where you need both?
  - Clark-Wilson Model
    - Limits direct interaction between subjects and objects
    - Prevent unauthorized subjects from modifying objects
    - Prevent authorized subjects from making invalid modifications to objects
    - Maintain internal/external consistency



Other Models

- Brewer and Nash model (Chinese wall model)
    - Allows dynamically changing access permissions.
    - Designed to mitigate conflict of interest.
- Graham-Denning Model
    - Computer security model.
    - Concerned with how subjects/objects are created/deleted securely, how privileges are assigned, and how ownership is assigned.
- Harrison-Ruzzo-Ullman (HRU) model
    - Extends on Graham-Denning model, maps subjects (S) objects (O) and access rights to an access matrix (P) where each cell contains the rights (R).
    - Constrains subjects from access to specific commands that would gain additional privileges, for example restricting access to a command that would grant read access to other documents.

Briefly on Security Evaluation

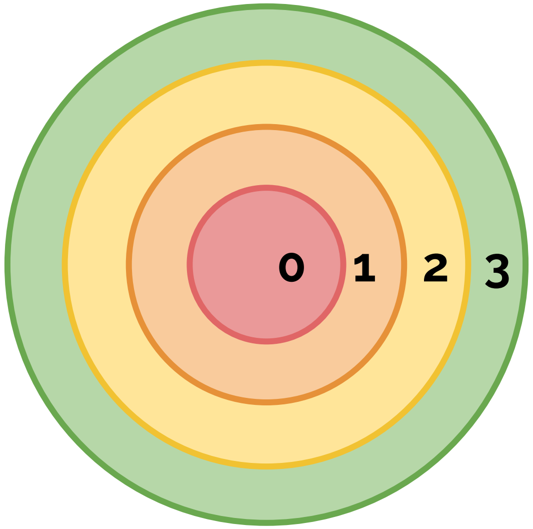
- Common Criteria (CC)
  - Originated with ITSEC, CTCPEC, and TCSEC
  - Concepts for evaluation (TOE, PP, ST, SFRs, SARs, EAL)
  - Often criticized as an expensive (hundreds of thousands £) government-driven process with poor track-record of actually detecting vulnerabilities.
    - Researcher suggests CC discriminates against FOSS-centric organisations.
  - Success stories:
    - Smart cards
  - Failure cases:
    - Operating systems
- UK government uses alternatives to fast track certain scenarios, but these aren't recognised internationally.

THE HIGHEST EVALUATED AND CERTIFIED IT SECURITY PRODUCT IN THE WORLD
CC EAL 7+
INPUT
OUTPUT

Protection Rings

- Hardware based access control.
  - Also used to protect data and functionality from faults.
- Each subject and object are assigned a number based on importance.
- Decisions are made by comparing numbers:
  - If subject < object, disallow access.
- x86 CPUs offer four rings, but typically (Windows/UNIX) only two (0,3) are used.
- ARM implements 3 levels (application, operating system, and hypervisor).

0: Operating system kernel.
1: Operating system.
2: Utilities.
3: User processes.



Granting Root Privileges

Real situation not long ago:

- Phil is a PhD student who has not taken this security course. He’s deploying his mathematical model to the web for the industry that’s funding him.
- His supervisor, Jacob, has a big UNIX server with 30 other PhD projects and lots of highly-sensitive data.
- Phil says “Jacob, I don’t have permission to copy the files to /var/www - can you give me sudo access?”
- Jacob googles “How to add another user as root”, finds the command: “sudo adduser phil sudo”, types it in. Jacob goes back to his office. Done!

<u>This kind of situation is VERY common.</u>

*NIX Recap

- UID 0 & root
- inode data structure & nearly everything is a file
- /etc/passwd
- /etc/shadow
- /etc/group
- File access - RWX
    - Can be converted to ACM
- Link vulnerabilities
    - Link to secure file, run command on linux to make real file insecure
- Devices file
    - /dev/tty
        - Often read/write to all
- Don’t give lots of people root
    - setuid, sudo

Securing BIOS and Bootloader

- BIOS should have a password for changing the settings
  - If you have physical access, then you can reset bios easily by resetting the CMOS
  - So lock the machine physically (require a key)
- Bootloader (e.g. GRUB) should have a password for changing the settings
  - Go into edit mode, then append to the linux kernel options in init=/bin/bash
  - This will directly boot in a shell with root privileges
- On Windows there is a bootable USB that you can make that allows full access to the registry that allows you to edit users/passwords
  - http://www.chntpw.com/burn-to-cd-usb/

Hardware Keylogger