Operating System Security & Access Control

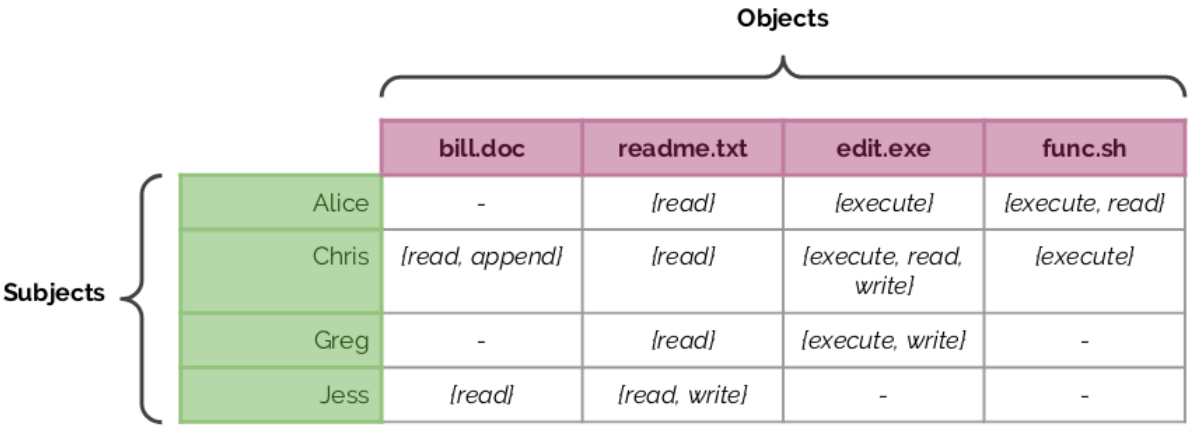
1 Access Control

- 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

2 Access Control Matrix (ACM)

+ Easy to define, easy to verify

- Poor scalability, poor handling of changes, could get corrupted


Objects
Subjects

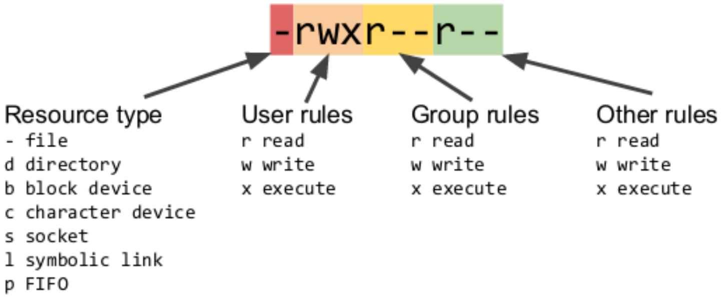
	bill.doc	readme.txt	edit.exe	func.sh
Alice	-	*[read]*	*[execute]*	*[execute, read]*
Chris	*[read, append]*	*[read]*	*[execute, read, write]*	*[execute]*
Greg	-	*[read]*	*[execute, write]*	-
Jess	*[read]*	*[read, write]*	-	-

Dashes represent no access rights
Append typically used for log files

3 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

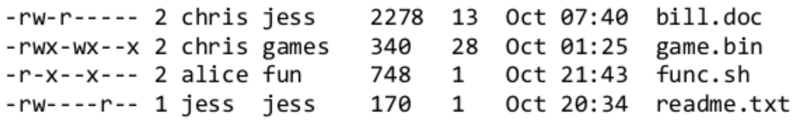


`chown` and `chmod` can be used to modify access permissions

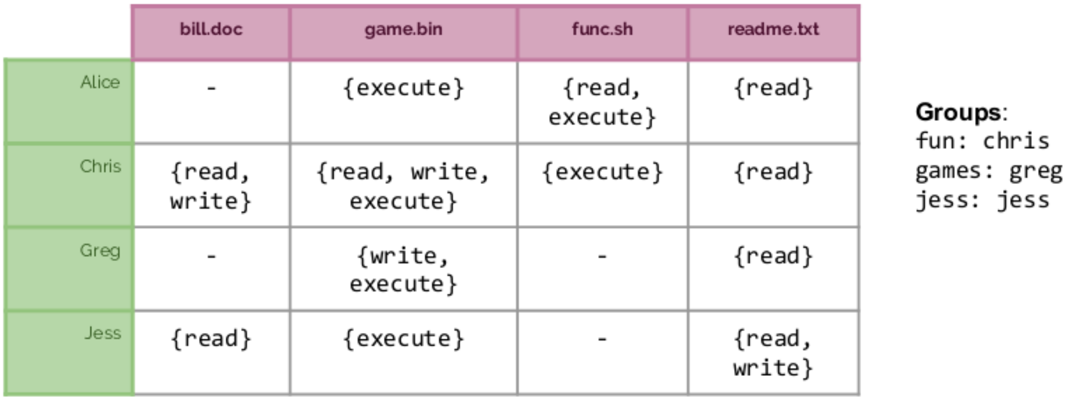
4 setuid, setgid, and sticky bits

setuid bit: Users run executable with permissions of the executable owner
sticky bit: Prevents users with write/execute permissions from deleting the directory contained files (typically on `tmp` folder)

5 *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

6 Link Vulnerabilities

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

  ```
  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)`

7 Hardening (Not examined)

- SELinux - Make sure that programs only access what they're meant to
- AppArmor - Similar but simpler than SE linux

8 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)

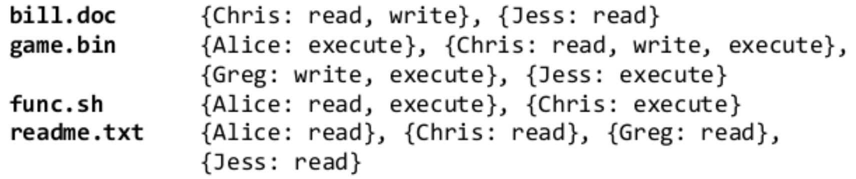
- Can bypass access control by getting access to memory
- Can get access to user inputs

9 Access Control Lists

- 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



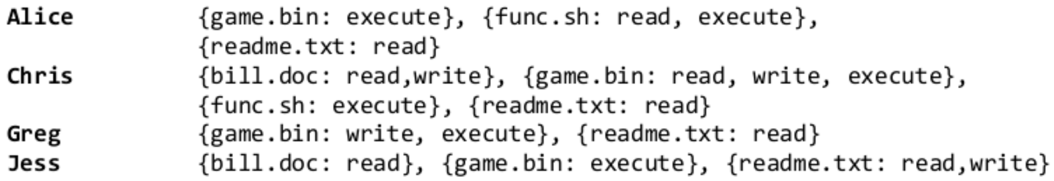
```
bill.doc      {Chris: read, write}, {Jess: read}
game.bin      {Alice: execute}, {Chris: read, write, execute},
              {Greg: write, execute}, {Jess: execute}
func.sh       {Alice: read, execute}, {Chris: execute}
readme.txt    {Alice: read}, {Chris: read}, {Greg: read},
              {Jess: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

10 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



```
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}
```

11 Windows

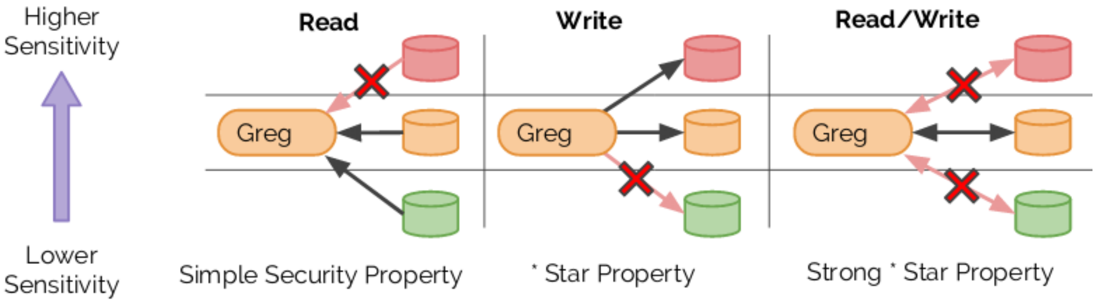
- Windows registry
  - Core place for system control
  - Target for hackers
  - Controls multiple computers
- Windows domain
  - Computers sharing things such as passwords
- Principles
  - SAM format - old but used in most places
  - UPN - more modern
- Login - Happens in different ways depending if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

- NTFS is much more secure than FAT32 & DOS
  - Adds two ACLs:
    - DACL: Reading, writing, executing, deleting by which users or groups
    - SACL: for defining which actions are audited/logged, e.g.on activity being successful/failed
  - Compression, encryption

12 Bell-LaPadula Model

Bell-LaPadula confidentiality policy, ”read down, write up”

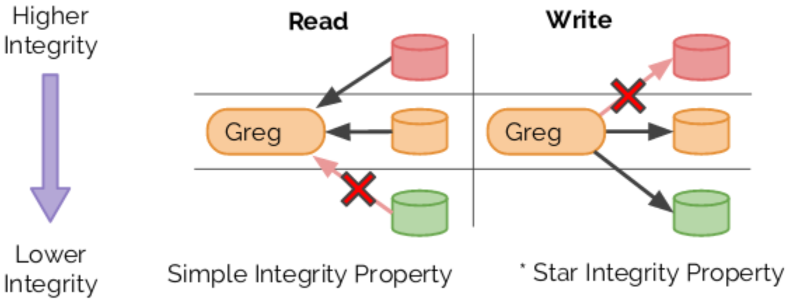
- Simple security property
  - Subject (Greg) cannot read object of higher sensitivity
- Star property (* property)
  - Subject cannot write to object of lower sensitivity. This is because Greg might know things that shouldn’t be able to be accessed by people of lower security
- Strong star property (Strong * Property)



13 Biba integrity model

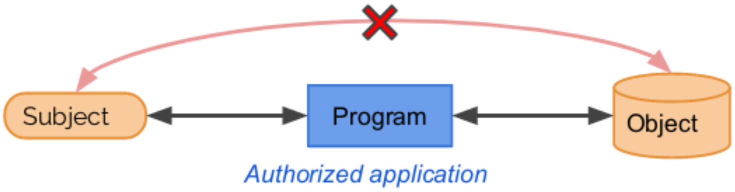
Biba integrity model - ”read up, write down”

- Simple Security property
  - Subject (Greg) cannot read object of lower integrity (can only read data that is as good or better than his)
- Star property (* property)
  - Subject cannot write to object of higher integrity (can only write data that is as good or worse than his)
- Invocation property
  - Subject/process cannot request higher integrity access



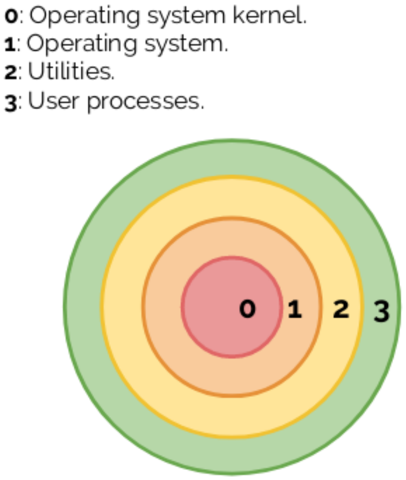
14 Clark-Wilson Integrity Model

- 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



15 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.

16 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