CSci 4271W
Development of Secure Software Systems
Day 12: Ethics and law in security

Stephen McCamant

University of Minnesota, Computer Science & Engineering

Outline

Exercise: using Unix permissions

Ethics and security

Legal context for security

More Unix permissions

Octal digits represent access

- 7 = rwx
- 6 = rw
- 5 = rx
- 4 = r
- 0 = no access

Setting: files related to this class

- Student and course staff materials
- Imagine everything is in Unix files on CSE Labs
  - Versus reality of a mixture of Unix with web-based systems like Canvas

Users and groups

- Users: smccaman (instructor), wang8330 (TA), stude003 (student)
- Groups: csci4271staff (instructor and TAs), csci4271students, csci4271all (staff and students)

What I want from you

- Brainstorm sets of octal permissions bits that could be used
- For each permission bits set, give user, owner, and file/directory contents/use that would be sensible

Outline

Exercise: using Unix permissions

Ethics and security

Legal context for security

More Unix permissions

Don’t be evil

- Broadly, ethics are principles for distinguishing good from bad actions
- Most people try to be good most of the time
  - But there are hard cases
- Topics important enough for security are usually also important for ethics
  - But adversaries often arise from ethical disagreement

Principles and consequences

- Ethical reasoning tends to be a mix of:
- Principles for categorizing actions as good or bad
  - Religions and laws provide many examples
- Attention to the consequences of actions
  - E.g., actions are evil because of their negative effects
- Another meta-principle: people's ethical intuitions vary

Ethics and laws

- The legal system is a primary way societies enforce ethical guidelines
  - But the law is an imperfect consensus approximation of ethics
- Following the law and being ethical can be separate constraints
  - You should try to satisfy both

Beyond white and black hats

- In describing techniques, we posit a clear distinction of attackers and defenders
- But in real scenarios, you can't assume that attacker = bad and defender = good
- What follows are some specific situations showing more complexity

Ethics of security research

- Why do good people research (and teach) about attack techniques?
  1. In order to effectively defend, you have to be able to anticipate attacker strategies
  2. In some cases, attacks may be ethically justified
- Common example: finding vulnerabilities so they can be fixed

Responsible disclosure

- If you find a vulnerability in software, who should you tell about it? Two extremes:
  - Only the author/vendor ever needs to know
  - Make the information fully public right away (full disclosure)
- Security researchers often push on vendors for more and faster disclosure
- A common compromise is to give vendors a head start, but with a deadline
  - E.g., Google uses 90 days (or 7 days if being used)

Nation states

- Many governments would argue they need to break the security of criminals or foreign spies
  - "justice", "public safety", "national security", etc.
- "Cyber-warfare" has both offensive and defensive aspects
  - Compare with various ethical perspectives on killing in war

Interoperability and repair

- Vendors of devices can have economic desires to control how the devices interact with other devices or can be repaired
  - Classic example: expensive proprietary ink cartridges
- If vendors use security and cryptography techniques to implement these restrictions, is it ethical to attack them?

Copy protection and DRM

- Vendors of software and media would prefer you can't make copies to give to your friends
  - Many generations of attempts to implement such restrictions
  - Fundamentally hard, because the data must be decoded to be used
  - Keeping software from being reverse engineered is also hard
- Do the ethics depend on how competent the technique is?

Malware analysis

- Labeling software as malicious is defining it to be the evil side
  - E.g., viruses, botnet clients
- Leads to many software security concerns being inverted
- Preventing reverse engineering is a common goal of DRM software and malware

Outline

Exercise: using Unix permissions

Ethics and security

Legal context for security

More Unix permissions

Mostly US federal law

- In the US, federal law is most important in computing
  - State laws are hard to enforce across the Internet
- Other countries have their own laws that differ in details
- Treaties and international effects are sometimes also important

Benefits and costs of law/regulation

+ Enforce ethical norms on otherwise reluctant parties
  - Especially: criminals, large corporations

- Interested parties lobby for laws favorable to them

- Laws can easily fall behind technology development

- Extra costs of complying with laws

Intellectual property

- Patents: useful inventions, ~20 years
- Copyrights: fixed expressions, ~100 years
- Trademarks: business identifiers, unlimited
- Trade secrets: supplementing contracts, unlimited

Privacy?

- No law provides general protection of personal privacy
  - Gap partially filled by agency regulation
- Two major industries have specific laws:
  - FERPA in education
  - HIPAA in health care (the P doesn't stand for privacy)

CFAA

- Computer Fraud and Abuse Act of 1986
- Civil and criminal liability for "unauthorized access" to a computer
- Gradually extended to cover any computer, and many related activities
- Potentially applied to any contract or terms-of-service violation
  - Not always successfully

Example: Randal Schwartz

- Schwartz worked as a contract sysadmin several Intel divisions
- He ran a password cracking program and moved password files between machines in a division he no longer worked for
- He was convicted of three felonies under an Oregon state law
  - Similar to the CFAA, somewhat more vague

DMCA

- Digital Millennium Copyright Act of 1998
- Legally reinforces DRM by criminalizing "circumvention" and tools that perform it
- But, can violate without violating copyright
  - App stores, video game bots, garage door openers
- A narrow exemptions process is growing in application

Example: Sony BMG "rootkit"

- In 2005, sold CDs with software that modified a Windows or Mac OS to interfere with copying
- To prevent removal, the software used techniques usually used by malicious software
  - A "rootkit" is backdoor software installed on a compromised machine
  - Common techniques include hiding files and processes
- Led to a recall, class action suits, FTC settlement, etc.

Outline

Exercise: using Unix permissions

Ethics and security

Legal context for security

More Unix permissions

Process UIDs and `setuid(2)`

- UID is inherited by child processes, and an unprivileged process can't change it
- But there are syscalls root can use to change the UID, starting with `setuid`
- E.g., login program, SSH server

Setuid programs, different UIDs

- If 04000 "setuid" bit set, newly exec'd process will take UID of its file owner
  - Other side conditions, like process not traced
- Specifically the *effective UID* is changed, while the *real UID* is unchanged
  - Shows who called you, allows switching back

More different UIDs

- Two mechanisms for temporary switching:
  - Swap real UID and effective UID (BSD)
  - Remember *saved UID*, allow switching to it (System V)
- Modern systems support both mechanisms at the same time

Setgid, games

- Setgid bit 02000 mostly analogous to setuid
- But note no supergroup, so UID 0 is still special
- Classic application: setgid `games` for managing high-score files

Special case: `/tmp`

- We'd like to allow anyone to make files in `/tmp`
- So, everyone should have write permission
- But don't want Alice deleting Bob's files
- Solution: "sticky bit" 01000

Special case: group inheritance

- When using group to manage permissions, want a whole tree to have a single group
- When 02000 bit set, newly created entries with have the parent's group
  - (Historic BSD behavior)
- Also, directories will themselves inherit 02000

Other permission rules

- Only file owner or root can change permissions
- Only root can change file owner
  - Former System V behavior: "give away `chown`"
- Setuid/gid bits cleared on `chown`
  - Set owner first, then enable setuid