CSE 451: Operating Systems
Winter 2001

Lecture 19:
Some Principles of Security

Steve Gribble
gribble@cs.washington.edu
323B Sieg Hall

Security

(Many of these slides taken from David Wagner's security course at UC Berkeley.)

- Computer Security
  - techniques for computing in the presence of adversaries
    - three categories of security goals:
      - confidentiality: preventing the unauthorized release of information
      - integrity: preventing the unauthorized modification of information
      - availability: preventing denial of service attacks
    - protection is all about providing all three on a single machine
      - usually considered to be the responsibility of operating systems
- Cryptography
  - techniques for communicating in the presence of adversaries

Trusted Computing Base (TCB)

- Think carefully about what you are trusting with your information
  - if you type your password on a keyboard, you're trusting:
    - the keyboard manufacturer
    - your computer manufacturer
    - your operating system
    - the password library
    - the application that's checking the password
  - TCB = set of components (hardware, software, wetware) that you trust your secrets with
- Public web kiosks should *not* be in your TCB
  - should your OS?
    - but what if it is promiscuous? (e.g., IE and active-X extensions)
  - how about your compiler?
    - A great read: "Reflections on Trusting Trust".

Design Principles for Security-Conscious Systems

- Security is much, much more than just crypto
  - if there is a fundamental flaw in the design of a system, then all the crypto in the world won't help you
  - unfortunately, systems design is still as much an art as it is a science…
    - but, decades of building systems the wrong way have helped us cull some learned wisdom
    - we'll cover just a few of these in the rest of this lecture

Principle of Least Privilege

- Figure out exactly which capabilities a program needs to run, and grant it only those
  - not always easy, but: start out granting none, run program, and see where it breaks. add new privileges as needed.
- Unix: concept of root is *not* a good example of this
  - some programs need to run as root just to get a small privilege, such as running with a port < 1024
    - e.g., ftpd

Tractorbeaming wu-ftpd

- wu-ftp tries to run with least privilege
  - but occasionally tries to elevate its privilege with:

```
seteuid(0);
// some privileged "critical section" runs here
seteuid(getuid());
```

  - however, wu-ftp does not disable UNIX signals
    - thus, while it is in critical section, it can be tractor-beamed away to a signal handler
    - remote user can cause signal handler to run by terminating a download in midstream!
      - but need to catch wu-ftp in the critical section
    - wu-ftp doesn't relinquish privileges after signal handler
  - result: abort a download, and if win a race condition, wu-ftp never relinquishes root privileges!
    - get full access to *entire* remote file system

Principle of Least-common Mechanism

- Basic lesson: be careful with shared code
  - assumptions made may no longer be valid
- Eudora/Outlook and Internet Explorer
  - windows exports an API to IE's HTML rendering code
    - eudora and other programs use this to display HTML in email
    - by default, JavaScript and Java parsing is enabled
  - HTML rendering code knows Java{Script} is unsafe
    - disables it when javascript is off of internet
      - "internet is untrusted"
    - but enables it when javascript is loaded off of disk
      - "your own file system is trusted"
  - but…email is loaded off of the disk!
    - oops.

Even more pernicious shared mechanism

- VMS password checking flaw
  - password checking algorithm:

```
for (I=0; I<password.length( ); I++) {
    if password[I] == supplied_password[I]
        return false;
}
return true;
```

  - can you see the problem?
    - hint: think about virtual memory…
    - another hint: think about page faults…
    - final hint: who controls where in memory supplied_password lives?

Principle of Complete Mediation

- Check **every** access to **every** object
  - in rare cases, can get away with less (caching)
    - but only if sure nothing relevant in environment has changed…and there is a lot that's relevant!
- e.g., NFS and file handles
  - NFS is not a good example of complete mediation
  - NFS protocol:
    - client contacts remote "mountd" to get a filehandle to a remotely exported NFS filesystem
      - this is done when remote system is mounted
      - remote mountd checks access control at mount time
    - filehandle is a capability: client presents it to read/write file
      - access control is not checked after mount time!
    - use network sniffer to get filehandle
      - access exported filesystem without access control check
      - this is how I used to mount my directories at Berkeley
        - » faster than going through beurocracy (remember, white-hat)

Fail-Safe defaults

- Start by denying all access, then allow only that which has been explicitly permitted
  - oversights will then show up as "false negatives", I.e. somebody is denied access that should be given it
  - opposite leads to "false positives", I.e. somebody is given access that shouldn't get it
    - bad guys usually don't report this kind of failure…
- Examples:
  - SunOS shipped with "+" in /etc/hosts.equiv
    - `When a user name (or +) is specified in ROOTDIR/etc/hosts.equiv, that user (or all users, in the case of +) may log in to the local host as any local user. User names in ROOTDIR/etc/hosts.equiv should always be specified.`
  - Irix shipped with "xhost +" by default
    - I've used this to help friends debug programs remotely..
      - with their permission, of course

"Security through Obscurity" = bad

- Security through obscurity
  - attempting to gain security by hiding the implementation details of a system
  - claim: a secure system should be secure even if all implementation details are published
    - in fact, a system grows more secure as people scour over implementation details and find flaws
    - rely on mathematics and sound design to keep secure
      - think back to cryptographic protocols in last lecture
- Counterexample: GSM cellphones
  - "impossible to clone"
    - I had the privilege of receiving the first phone call on a cloned GSM cell phone
    - GSM committee designed their own crypto algorithm, but hid it from the world
      - social engineering + reverse engineering revealed the algorithm
      - it turned out to be very weak
        - » could essentially play 20 questions with identity chip on cell phone, and eventually learn its secret key

Security: outlook for the future

- Doesn't look bright…
  - more and more complex systems are being deployed
    - and more and more human lives are being trusted to them
- Bruce Schneier: 3 waves of security attacks
  - 1st wave: physical attacks on wires and hardware
    - physical security to defend against this
  - 2nd wave: syntactic attacks on cryptographic protocols and software implementations
    - e.g., buffer overflows, DDOS attacks. Just beginning to get a handle on this, long way to go.
  - 3rd wave: semantic attacks. Humans and computers trust information that they shouldn't
    - e.g., forged web content
    - e.g., falsified press announcements (Emulex corp stock hoax)
      - CEO resignation announcement, 61% stock drop
      - semantic attack against people with preprogrammed sell orders
    - we have no idea how to handle this
      - yet, networked society is eminently vulnerable to it…
- Bottom line: security research is one of the systems areas!