CSE 451: Operating Systems

Security

Outline

- "Classic" security topics
  - goal: safe sharing
  - general principles
  - Trusted Computing Base (TCB)
- Contemporary security problems
  - worms
  - botnets
  - spyware

Safe sharing

- Protecting a non-networked PC with one user is easy
  - Nobody can access the data on your computer
  - Nobody can install new code
  - Nobody can attack you over the network
- Sharing resources safely is hard
  - Prevent some users from reading private data
  - yet allow authorized users to access it
  - e.g., grades, keystrokes
  - Prevent some users from using too many resources
  - e.g., disk space
  - Prevent users from interfering with others' programs
  - spoofing displays, replacing programs with malicious code, killing off processes...

Much of security is art, not science

- Difficult to "prove" a system secure
- Security is based on principles and best practices
  - experience reveals commonly occurring types of flaws
  - but clearly we need to do better...

Principle of Least Privilege

- Figure out exactly which capabilities a program needs to run, and grant it only those
  - start out by 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 root just to get a small privilege
    - e.g., FTP daemon requires root:
      - to listen on network port < 1024
      - to change between user identities after authentication
    - but root also lets you read any file in filesystem

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!
- A TLB caches access control information
  - page table entry protection bits
  - is this a violation of the principle?
“Security through Obscurity” = bad

- Security through obscurity
  - “gain security” by hiding system implementation details
  - should be secure even if implementation is open!
    - in fact, publishing makes it more secure, since people can scour implementation and find/fix flaws
- Counterexample: GSM cell phones
  - GSM committee designed own crypto algorithm, but hid it
    - “impossible to clone”
    - social + reverse engineering revealed the algorithm
      - It turned out to be very weak

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
      - including the keyboard device driver
      - the password library
      - the application that’s checking the password
    - what about the computer that compiled all of this software (!!)
- TCB = set of components (hardware, software, wetware) that you must trust to preserve your secrets
  - should be as small as possible
    - public web kiosks should “not” be in your TCB
    - how about your web browser?

Modern security problems

- Internet experiencing a plague of attacks
  - remote exploits: attackers breaking into your system
  - worms: self-replicating attack code
  - botnets: armies of compromised machines
  - spyware: software that tries to steal information from you
- Underlying issues
  - most of our code is buggy
  - the Internet was designed to be “open”
    - easy to build new services, but easy to find/attack victims
  - understanding security is hard
    - haven’t found simple conceptual models or usable UIs
    - e.g., what does the lock icon in IE really mean?

Remote exploit

- An exploitable bug in network-facing software
  - e.g.: buffer overflow attack
    ```c
    int main(int argc, char *argv[]){
    char buffer[10];
    strcpy(buffer, argv[1]);
    return 0;
    }
    ```
  - exploit this bug, smash the stack, run code of your choice
  - e.g.: SQL injection attack
    - typing the following into a bookstore web search form:
      ```sql
      "book tipping point;  SELECT * FROM CREDITCARDS"
      ```

Using remote exploits -- worms

- Pseudocode for a simple worm
  ```c
  for (i = 0.0.0.0; i < 255.255.255.255; i++) {
    open network connection to IP address “i”;
    if succeed {
      try to exploit vulnerability x on “i”;
      if succeed {
        send code for self to victim and run it;
      }
      close connection to “i”;
    }
  }
  ```
  - Will this worm propagate?
    - how quickly?

A “better” worm

```c
while (1) {
  open network connection to random IP address “i”;
  if succeed {
    try to exploit vulnerability x on “i”;
    if succeed {
      send code for self to victim and run it;
    }
    close connection to “i”;
  }
}
```
Even better worms…

- **Local scanning**
  - probe nearby IP addresses preferentially
- **Increased scan rate ==> faster spread**
  - Code Red: approximately 5 scans per second
  - Sapphire worm: approximately 4000 scans per second
    - single UDP packet contains worm

- **Sapphire worm data**
  - worm doubled in size every 8.5 seconds
  - saturated susceptible population of ~75,000 hosts in about 5-10 minutes (!!!)

Sapphire fallout

- It propagated too fast for its own good!
  - no per-host damage
  - but massively clogged Internet backbones with scans
  - self-interference slowed its propagation rate

Using remote exploits - Botnets

- **Step 1:** compromise a remote computer
- **Step 2:** upload “botnet” software
  - sits silently in the background
  - gives attacker remote control of the “zombie” computer
- **Step 3:** repeat steps 1 and 2 10,000 times
  - amass a giant “zombie” army
- **Step 4:** control army using botnet “controller”
  - rent out time on botnet army
  - use zombies to perform spam relay, click spam
  - perform “denial of service” attack on a victim
  - flood it with requests

Example: Phatbot

- Some of its features:
  - polymorphs on install to evade anti-virus signature
  - sends email probes to test for spam relay capability
  - can steal windows product keys
  - runs an FTP server to distribute itself to other hosts
  - runs a redirection service for TCP connections
    - (launders network traffic)
  - can scan and spread using many exploits
    - (worm-like behavior)
  - kills worms, other bots to defend turf
  - kills anti-virus processes
  - steals various website account passwords
  - harvests email addresses for spam purposes

Recent, local example

- **UW Medical Center**
  - some unpatched machines were compromised
  - attackers used foothold to get UWMC password database
  - things started to fall apart
    - some key cards would no longer open operating room doors
    - some computers in the ICU stopped working
    - some doctors’ pagers stopped working
  - Impossible to know which accounts got compromised
    - 20,000 people had to change their UW NetID passwords!
    - hopefully no confidential data was taken…

Spyware

- Software that is installed that collects information and reports it to third party
  - key logger, adware, browser hijacker, …
- **Installed one of two ways**
  - piggybacked on software you choose to download
  - “drive-by” download
    - your web browser has vulnerabilities
  - web server can exploit by sending you bad web content
- **Estimates**
  - majority (50-90%) of Internet-connected PCs have it
  - 1 in 20 executables on the Web have it
  - about 0.5% of Web pages attack you with drive-by-downloads
kingsofchaos.com

- A benign web site for an online game
  - earns revenue from ad networks by showing banners
  - but, it relinquishes control of the ad content

Incident

- kingsofchaos.com was given this "ad content"
  ```javascript
  \u003cscript type="text/javascript"\u003e\document.write('...');</script>...etc.
  ```

- This "ad" ultimately:
  - bombarded the user with pop-up ads
  - hijacked the user's homepage
  - exploited an IE vulnerability to install spyware

What's going on?

- The advertiser was an ex-email-spammer
  - His goal:
    - force users to see ads from his servers
    - draw revenue from ad "affiliate programs"
    - Apparently earned several millions of dollars
- Why did he use spyware?
  - control PC and show ads even when not on the Web

Parting thoughts...

- Security is hard
  - fundamentally an adversarial, escalating game
  - we're getting better, but so are the "bad guys"
- Our systems are insecure
  - OS software one of the most complex artifacts of humankind
  - no surprise it has flaws!
- Current trends
  - reduce TCB to exclude OS
  - develop stronger sandboxes to contain flaws
  - virtual machine software (e.g., Vmware)
  - program with safer languages than C