Protection & Security

- **Security**
  - Prevention of unauthorized access to a system
    - Malicious or accidental access
    - "Access" may be:
      - usher login, a process accessing things it shouldn’t, physical access
    - The access operations may be reading, destruction, or alteration

- **Protection**
  - The mechanism that provides and enforces controlled access of resources to processes
  - A protection mechanism enforces security policies

Threats

Computer security... then

- Colossus at Bletchley Park: breaking codes
- ENIAC at Moore School: ballistic firing tables
- Single-user, single-process systems
- Data security needed
- Physical security

Computer security... now

- Sensitive data of different users lives on the same file servers
- Multiple processes on same machine
- Authentication and transactions over network
  - Open for snooping
- We might want to run other people’s code in our process space
  - Device drivers, media managers
  - Java applets, games
  - Not just from trusted organizations

Systems are easier to attack

**Automation**
- Data gathering
- Mass mailings

**Distance**
- Attack from your own home

**Sharing techniques**
- Virus kits
- Hacking tools
Penetration

Guess a password
- system defaults, brute force, dictionary attack

Crack a password
- Online vs. offline
- Precomputed hashes (see rainbow tables)
  • Defense: Salt

Social engineering
- people have a tendency to trust others
- identify corporate/school organizational structure
- facebook, twitter, blogs, personal home pages
- look through dumpsters for information
- impersonate a user
- Phishing: impersonate a company/service

Malicious Files and Attachments

Masqueraded e-mail

Trojan horse
- program masquerades as another
- Get the user to click on something, run something, enter data

Phishing

Guess a password

Crack a password
- Online vs. offline
- Precomputed hashes (see rainbow tables)
  • Defense: Salt

Page 9 of the Linksys Wireless-N Gigabit Security Router with VPN user guide

Use your Rutgers University username and password to log into any CS iLab and Graduate domain machine.

If you cannot log in please verify that you have acknowledged the Department of Computer Science Academic Integrity Policy at the following address. You must acknowledge this policy once every academic calendar year.
login: pxk
Password: Login incorrect

Phishing

Masqueraded e-mail

Trojan horse
- program masquerades as another
- Get the user to click on something, run something, enter data

Malicious Files and Attachments

Take advantage of:
- Programs that automatically open attachments
- Systems that hide extensions yet use them to execute a program
  - trick the user

love-letter.txt.vbs looks like love-letter.txt
resume.doc.scr looks like resume.doc

© 2014 Paul Krzyzanowski
Exploiting bugs

Exploit software bugs
- Most (all) software is buggy
- Big programs have lots of bugs
  - sendmail, wu-ftp
  - some big programs are setuid programs
    - ftp, uucp, sendmail, mount, mkid, eject

Common bugs
- buffer overflow (blindly read data into buffer)
  - e.g., gets
  - back doors and undocumented options

The classic buffer overflow bug

gets() from OS X © 1990,1992 The Regents of the University of California.

```c
char *buf;
static char w[]="warning: this program uses gets(), which is unsafe.
\n\n"
if (!warned) {
  (void) write(STDERR_FILENO, w, sizeof(w) - 1);
  warned = 1;
}
for (s = buf; (c = getchar()) != '\n';)
  if (c == EOF)
    if (s == buf)
      return (NULL);
    else
      break;
  else
    *s++ = c;
*s = 0;
return (buf);
```

Buffer overflow

More data was input than the programmer expected, causing the local array that was allocated for the data to overflow. The overflow overwrites the return address on the stack. Now, when the function returns, the return address is under the control of the attacker.

Return Oriented Programming (ROP)

- Stack can still be corrupted
- Can overwrite return address with address of a library function
  - Does not have to be the start of the library routine
    - "borrowed chunks"
  - When the library hits the RET instruction, that return address is on the stack, under the attacker’s control
- ROP chains together sequences ending in RET
  - Build together “gadgets” for arbitrary computation
  - Buffer overflow contains a sequence of addresses that direct each successive RET instruction
- Make attacking easier: C compiler that generates gadgets!

Dealing with buffer overflows: No Execute

- Executable space protection
  - Disallow code execution on the stack or heap
  - Set MMU per-page execute permissions to no-execute
  - Intel and AMD added this support in 2004
  - Examples
    - Microsoft DEP (Data Execution Prevention) (since XP SP2)
    - Linux PaX patches
    - OS X ≥10.5

Dealing with buffer overflows: ASLR

- Address Space Layout Randomization
  - Dynamically-loaded libraries used to be loaded in the same place each time … ditto for the stack & memory-mapped files
  - Well-known locations make them branch targets in a buffer overflow attack
  - Position the stack and memory-mapped files (including libraries) to random locations
  - Implemented in
    - OpenBSD, Windows Vista+; Windows Server 2008, Linux 2.6.15, OS X
Dealing with buffer overflows: Canaries

- **Stack canaries**
  - Place a random integer before the return address on the stack
  - Before a return, check that the integer is there and not overwritten: a buffer overflow attack will likely overwrite it

```c
int a, b=999;
char s[5], t[7];
gets(s);
```

Virus

- Does not run as a self-contained process
- Code is attached onto another program or script
- **File infector**
  - primarily a problem on systems without adequate protection mechanisms
- **Boot-sector**
- **Macro** (most common form: visual basic scripting)
- **Hypervisor**
  - install on virtual machines

Virus scanning

- Search for a "signature"
  - Stream of bits in a virus that (we hope!) is unique to the virus and not any legitimate code
  - NOT a cryptographic signature!
- Some viruses are encrypted
  - Signature is either the code that does the decryption or the scanner must be smart enough to decrypt the virus
- Some viruses mutate to change their code every time they infect another system
  - Run the code through an emulator to detect the mutation

Virus scanning

- You don't want to scan through hundreds of thousands of files
  - Search in critical places likely to be infected (e.g., \windows\system32 or removable media)
- **Passive disk scan vs. active I/O scan** ... or both
- "Zero-Day Threats": new virus – signature unknown
- Virus scanning is becoming less effective
  - Estimates are that only 10-30% of new viruses in the network are detectable
Key loggers

- Record every keystroke
- Windows hook mechanism
  - Procedure to intercept message traffic before it reaches a target windows procedure
  - Can be chained
  - Installed via `SetWindowsHookEx`
    - `WH_KEYBOARD` and `WH_MOUSE`
      - Capture key up, down events and mouse events
- Hardware loggers

Rootkits

- Replacement commands (or standard shared libraries or OS components) to hide the presence of an intruder
  - ps, ls, who, netstat, ...
- Hide the presence of a user or additional software (backdoors, key loggers, sniffers)
- Now the OS can no longer be trusted!
- Examples
  - Sony BMG DRM rootkit (October 2005)
    - Creates hidden directory; installs several of its own device drivers; reroutes Windows system calls to its own routines
    - Intercepts kernel-level APIs and disguises its presence with cloaking (hides $sys$ files)
  - Carrier IQ (December 2011)
    - Software for cell phone analytics—designed to be undetectable
    - Installed on Sprint, HTC, Apple (iPhone 4), Samsung, BlackBerry, ...

Dealing With Rootkits

- Hope you don’t get one!
- Restrict permission to modify system files
  - To avoid installing a rootkit in the first place
  - But users often grant permissions during installation
    - And permissions may be needed for drivers
- Signed software and operating system components
  - Microsoft Vista & Windows 7:
    - Requires kernel-mode software to have a digital signature (x64-based systems only)
- Tripwire
  - Software to monitor for changes in files and components in a system

Restricting what an app can do

- Traditional OS approach to protection
  - Privileges are an attribute of the user
  - Protection mechanisms were designed to place restrictions on other users
  - Every process the user runs has the same privileges
    - A program can delete all of your files ...
      - or upload them to a remote server
- How do we protect ourselves from malicious applications?
- Mandatory Access Control (MAC)? Yes! In a way....
  - Traditional forms of MAC (e.g., Bell-LaPadula)
    - Restricts information flow between classes of users
  - Doesn’t solve the problem
  - Enforces ability of processes to access objects
  - Discretionary Access Control allows users/processes to make policy changes
A basic approach: chroot jail

**chroot** system call: change the root directory for a process


\[ chroot [/VAR/POOL/ROOT/files]\]

- Process root gets changed
- Cannot access any other part of the file system

Example: Apple Sandbox

- Create a list of rules that is consulted to see if an operation is permitted

  **Components:**
  - Set of libraries for initializing/configuring policies per process
  - Server for kernel logging
  - Kernel extension using the TrustedBSD API for enforcing individual policies
  - Kernel support extension providing regular expression matching for policy enforcement

  **sandbox** command & **sandbox** init function
  - `sandbox -exec` calls `sandbox_init()` before `fork()` and `exec()`
    - `sandbox_init(0,SBXProfileWrite, Sandbox_NAME, errbuf);`

Apple sandbox setup & operation

- Conversion of human-readable policies into a binary format for the kernel
- Policies passed to the kernel to the TrustedBSD subsystem
- TrustedBSD subsystem passes rules to the kernel extension
- Kernel extension installs sandbox profile rules for the current process

**Operation**
- System calls hooked by the TrustedBSD layer will pass through `Sandbox.kext` for policy enforcement
- The extension will consult the list of rules for the current process
- Some rules require pattern matching (e.g., filename pattern)

Apple sandbox policies

- Some pre-written profiles:
  - Prohibit TCP/IP networking
  - Prohibit all networking
  - Prohibit file system writes
  - Restrict writes to specific locations (e.g., `/var/tmp`)
  - Perform only computation: minimal OS services

Java Sandbox

**Java Virtual Machine**

1. **Bytecode verifier:** verifies Java bytecode before it is run
   - Disallow pointer arithmetic
   - Automatic garbage collection
   - Array-bounds checking
   - Null reference checking

2. **Class loader:** determines if an object is allowed to add classes
   - Ensures key parts of the runtime environment are not overwritten
   - Runtime data areas (stacks, bytecodes, heap) are randomly laid out

3. **Security manager:** enforces protection domain
   - Defines the boundaries of the sandbox (file, net, native, etc. access)
   - Consulted before any access to a resource is allowed

All bets are off if you allow native methods!
The End