CSE 543 - Computer Security

Lecture 9 - Malware
September 25, 2007
URL: http://www.cse.psu.edu/~tjaeger/cse543-f07/
The Morris Worm

- Robert Morris, a 23 doctoral student from Cornell
  - Wrote a small (99 line) program
  - November 3rd, 1988
  - Simply disabled the Internet

- How it did it
  - Reads /etc/password, they tries the obvious choices and dictionary, /usr/dict words
  - Used local /etc/hosts.equiv, .rhosts, .forward to identify hosts that are related
    - Tries cracked passwords at related hosts (if necessary)
    - Uses whatever services are available to compromise other hosts
  - Scanned local interfaces for network information
  - Covered its tracks (set is own process name to sh, prevented accurate cores, re-forked itself)
Vulnerabilities

- Network daemon vulnerabilities
  - Buffer overflows
- Insecure programs
  - Remote logins allowed
- User errors
  - Poor passwords
- Administration errors
  - Trust in other machines (hosts.equiv)
- Network information
  - Information about next likely victims (propagation)
Are those vulnerabilities a problem today?

- What was the problem with the threat model at the time of the Morris worm?

- **Which of these vulnerabilities are still a problem today?**

- Have we fixed the threat model?
Buffer Overflows

- One means by which the bad guys take over a host
  - install root kits
  - use as SPAM bots
  - use as zombies
  - launch other attacks

- There are many attacks, but this is most prevalent

- It all starts with some programmer mistake
  - e.g., bad software
Buffer Overflow

• How it works
Defenses

- "Canary" on the stack
  - Random value placed between the local vars and the return address
  - If canary is modified, program is stopped
  - Will this address the "basic" buffer overflow?

- Alternative:
  - Non-executable stack

- Are we done?
Code Injection

• So, the problem is solved then?
Address Space Randomization

• Problem: return-to-libc
• Attack
  • Overflow buffer
  • Instead of running code on stack
    • Call an existing libc function

• Randomization moves function locations to counter this attack
Commercial System Defenses

- **Buffer overflow prevention is now common**

- **Linux**
  - StackGuard
  - Non-executable stack
  - Address space randomization (for the stack)

- **Windows Vista**
  - Data execution prevention
  - Address space randomization (for all code)
  - Change local variable order on stack
  - Verify exception handlers
  - Function pointer obfuscation
Other Ways to Gain Entry

- Unfortunately, there are still lots of attack paths...

- Email Attachments
  - Many viruses

- Web
  - Cross-site scripting, Javascript, flawed plug-ins

- Trojan horse
  - What does that free program do?

- Others?
What do the bad programs do?

- **Virus**
  - A program that inserts itself into one or more executable files in which it will be executed with that file

- **Worm**
  - A program whose code aims to propagate itself to other machines

- **Trojan horse**
  - A program with an overt (documented) effect and a covert (undocumented) effect (often malicious)

- **Rootkits**
  - A virus that embeds itself in the trusted computing base of the system in such a way that it cannot be detected
Botnets

• A set of infected systems under the control of one authority
  • Usually, to distribute spam
  • Purpose of compromise has changed...
    • Bots may implement antivirus

• Estimated: 10 million bots
  • Botmasters trade/sell resources