

CS574

Computer Security

San Diego State University

Spring 2008

Lecture #7

Today's Structure

- Administrivia
- Questions
- Recent News
- Lecture

Administrivia

- Assignment #1 - ASI_Template.jar
- Alternative Tools
- You sending email to me
- Me sending email to you

Questions?

Recent News

- Dangers of encrypting everything
- Lost laptop and \$54 million lawsuit
- Networked (multifunction) printers
- Security Blackmail

Lecture

On Trust

- Ken Thompson - On Trusting Trust
- Leonid A. Broukis's one-line program:

```
char*a="char*a=%c%s%c;main(){printf(a,34,a,34);}";main(){printf(a,34,a,34);}
```

Terms

- agent
The creator or distributor of malicious programs.
- flaw
An observable departure from specification (typically unexpected / undesired behavior).
- fault
The incorrect step/process/etc. that causes a flaw.

(Malicious) Software

- Intentional versus Inadvertent
- Benign/Beneficial versus Harmful/Destructive
- Violation of “Policy”

Policy Violations

- Intrusion / monitoring
- Data theft
- Data destruction or modification
- System destruction
- Theft / denial of services
- Make vulnerable to future attack
- Use as a base to attack others

Types of Malicious Software

- Bomb
- Back door
- Trojan horse
- Virus
- Rabbit
- Worm

Bombs

- A device set to “go off” under specified conditions.
- A *logic bomb* is a piece of malicious code triggered by some event (after the agent has left).
- A *time bomb* is a specific type of logic bomb triggered by the data or an elapsed interval.

Example Bombs

- A *directory bomb* is a program that exhausts some filesystem resource, such as available inodes or directory-depth limits.
- A *fork bomb* is a program that exhausts the available PIDs for a system.

Back Door

- A *back door* (or *trap door*) is a way in to a system that allows the agent to (re)gain control.
- May be a standalone program, trojan horse, or system modification.
- Often part of a virus or worm payload.
- Many systems (used to) ship with back doors.

Trojan Horse

- A *trojan horse* is a program that does something other than what is advertised.
- Obviously a reference to Greek mythology.
- Often part of a “root kit”.
- Examples

Virus

- A *virus* is a trojan horse that makes other programs into trojan horses.
- Alternatively, a *virus* is a malicious program that can alter non-malicious programs, making them malicious.
- Typically made up of at least one “vector” and a potentially multi-part “payload”.

Virus

- Deliberate analogy to biological viruses.
- Concept described ~ 1970.
- First personal computer viruses in the 1980s.
- Typically infects files on a single computer.

Virus – Infection

- A virus *attaches* itself to a program.
- Can insert itself virtually anywhere in the program (most often beginning or end).
- May run before, after, or concurrently with the program.
- Nearly always changes file size, checksum, digital hash value, etc.

Virus - Vector

- The *vector* of a virus is the means it uses to spread itself.
- Process:
 - Find other (executable) programs locally;
 - Insert “self” in to those programs.
- May or may not know about the program it is infecting.

Virus Types

- Memory-resident
- Application-resident (aka transient)
- Document
- Boot Sector / Auto Run

Virus – Payload

- The *payload* of a virus are those additional functions beyond what is needed to replicate the virus.
- Payloads are frequently time or other logic bombs.
- A “non-malicious” (or ‘proof-of-concept’) virus can easily be made malicious by adding or changing the payload.

Virus – Payload

Example delayed functions:

- Wait for a specified date.
- Wait for some interval after infection.
- Count the reboots.
- Wait for the disk to be full.
- Watch for some specific program to be run.

Virus – Payload

Example payloads:

- Format drives
- Modify or erase files
- Display messages
- Re-infect system

Virus – Design Objectives

- Difficult to detect.
- Not easily destroyed or deactivated.
- Attaches to many other programs.
- Reinfection should not cause problems.
- Easy to create or adapt.
- Machine / Operating System independent.

Non-Malicious Viruses?

- Is it possible to have a non-malicious virus?
- Examples often include:
 - look for malicious viruses to remove
 - compress programs to save disk space
 - fix vulnerabilities in the host system
- Inadvertent effects range from negligible to disastrous (might as well be malicious).

Rabbit

- A piece of malicious code that replicates without bound in order to exhaust resources (e.g., a fork bomb).
- Less commonly used to refer to a piece of malicious code that is difficult to catch.

Aside: Exploits

- An *exploit* (or 'sploit) is a program that takes advantage of known flaws to gain control or access to a system.
- Also refers to the act of using such a program.
- Often via unpublicized back doors, buffer overflows, etc.
- Often packaged for easy/widespread use.

Further Aside: Vulnerability Scanner

- A *vulnerability scanner* is a program that automates the process of locating vulnerable hosts on a network.
- Generally built on a database of known vulnerabilities and way to check for them.
- A legitimate tool for a system administrator.
- And ideal tool for an opportunistic intruder.

Worm

- A *worm* is a program that spreads among computers without human intervention.
- A worm's vector typically includes some network-aware functions:
 - scan networks for vulnerable hosts
 - exploit remote vulnerabilities
 - copy self to victim, and run

Worm – Vectors

- Shares many techniques with viruses.
- Often have a vulnerability scanner.
- Have one or more exploits.
- Can be (but don't have to be) multi-platform.

Worm – Payloads

- Trojan horse(s)
- Back doors
- Time / logic bombs
- Just about any other malicious code

Next Time

- Worm examples
- More malicious code
- Software flaws
- Controls

Lecture References

- Pfleeger, *Security in Computing*, 3rd & 4th edition
- Bishop, *Introduction to Computer Security*

End Lecture

Reading

(For the next few classes)

- Pfleeger, Chapter 3 (finish)
- Pfleeger, Chapter 4 (start)

Finis