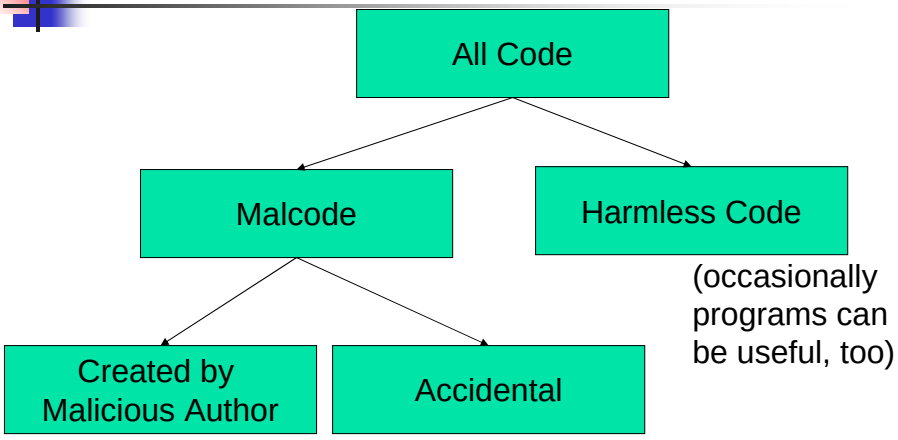
Programmed Threats and Program Security

- Program faults and flaws
- Controls Against Program Threats
- Targeted Malicious Codes, Viruses, worms, etc.

1

Taxonomy of Code



(occasionally programs can be useful, too)

2

Program Security

- Protection of programs is at the heart of security in computing. Why?
- Two types of program flaws:
  - Inadvertent human errors
    - Domain error
    - Aliasing
    - Boundary conditions
  - Malicious, intentionally induced flaws
- Why can't we stop all program flaws?

3

Big problem: buffer overflows

Has contributed to as much as 50% of security problems; is it getting any better?

- What is a buffer overflow?
  - *Buffer* is a memory area where contiguous chunks of the same data type are allocated.
  - *Buffer overflow* occurs when a program writes past the bounds of a buffer.
- Types of buffer overflows: heap and stack overflow
- Defense against buffer overflows

4

Main culprit: the language

- It is difficult to write secure C programs :
  - C is an inherently unsafe language.
  - There is no bounds checking on array and pointer references.
  - It is the programmer's responsibility to do the checking.
  - Unsafe string operations in the standard C lib.
- The best approach is to use a "safe" language.
- The challenge is there exist lots of C/C++ codes out there.

5

Stack versus heap overflows

Compared to other parts of a process (such as data segment and program segment, which are static), the stack and the heap are dynamic.

- Stack is used for allocating the context of the current function call (non-static local variables, parameters passed by value, return address, ...)
  - → *activation record* (or *stack frame*)
- Heap is for allocating data requested dynamically by a user program, such as via *malloc( )* in C or *new* in C++.

6

Heap Overflows

A less likely attack than stack overflows, because Heap overflows are generally much harder to exploit .

- The attacker needs to know many things:
  1. Which variable(s) are security critical;
  2. How the variables are allocated in the heap;
  3. The number of bytes allocated to a variable;
  4. A buffer that can overflow the target variable;
  5. ...

7

Stack Overflows

Steps:

1. Find a stack-allocated buffer that allows us to overwrite the return address in a stack frame;
2. Place some hostile code in memory to which control can jump when the function we are attacking completes execution and returns;
3. Write over the return address on the stack with a value that causes the program to jump to our hostile code.

8

Controls Against Program Threats

- Programming controls

  Typical software engineering methods: peer reviews, walk-through, information hiding, independent unit/integration testing, configuration control/management, formal methods
- Process controls
  - 1990: ISO 9000 – to specify actions to be taken when **any** system has quality goals and constraints
  - 1993: CMM (Capability Maturity Model) – to assess the quality of a software development company (replaced in 2000 by CMMI)
  - 1995: SSE CMM (System Security Engineering CMM) – to assess the quality of security engineering development practices (http://www.sse-cmm.org/)

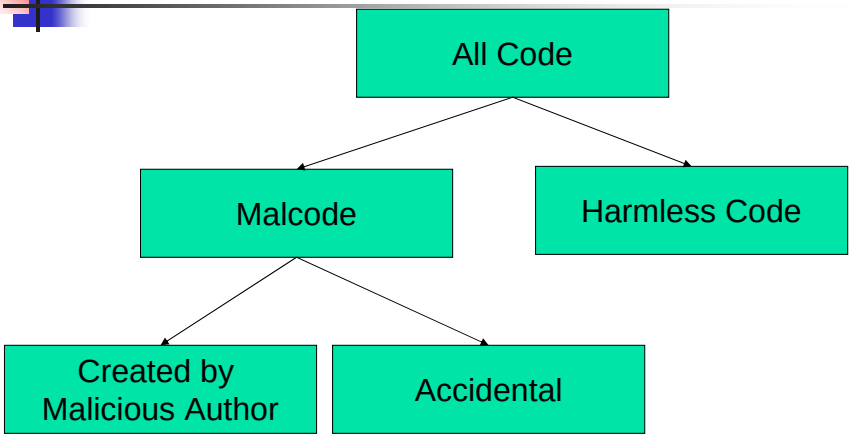
9

Controls Against Program Threats

- OS controls
  - Trusted OS
  - Confined programs
  - Access logs for auditing
- Administrative controls
  - Enforcing standards of design, documentations, programming, testing, configuration management ...
  - Security audits
  - Separation of duties among employees

10

Taxonomy of Code



11

Malicious Code

- **Malicious code**: *Instructions designed to violate the security policy of a site.*
- Sometimes called ***malware***: *any piece of hardware, software or firmware that is intentionally included or introduced into a computer system for unauthorized purposes, usually without the knowledge or consent of the user*
- "Malicious Code" is not an appropriate name
  - Code has no intent
  - Programmer's intent does not matter
  - What the code does matters
- Malcode ?

12

It includes

Malicious code which moves itself (active)

- **Virus**: code that replicates itself into another program/file/system
- **Worm**: migrating program.

Malicious code which stays in one place (static)

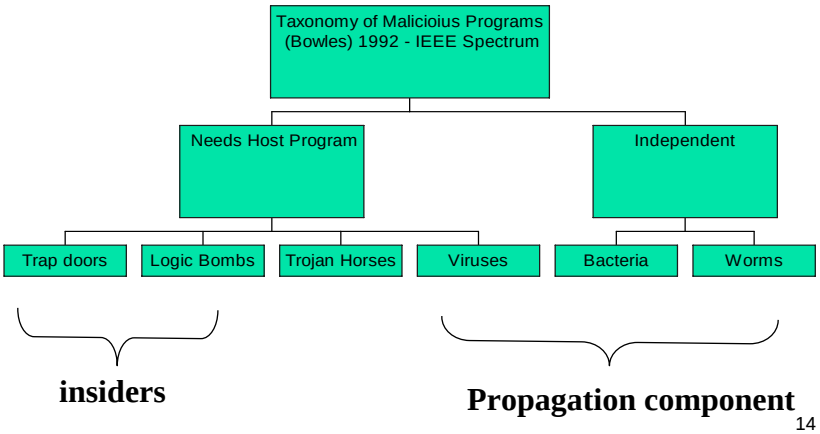
- **Trojan Horse**: covert code in a 'known' function.
- **Trap Door**: bypass authentication
- **Logic Bomb**: performs (destructive) action upon external condition.

Other

- **Rabbit, bacterium**: a code for denial of service-style attack by continuously tries to use resources until all absorbed.
- **Black Widows**: malicious applets.
- **Hoaxes**: message with false warning
- **Joke programs**: a kind of Trojan horse not intended to be malicious

13

Malicious code classification



14

Trap Doors

- A secret undocumented entry point into a program/module used to grant access bypassing usual methods of access authentication.
- Used for many years in legitimate program development - e.g., debugging mode
- Difficult to implement OS controls against trap doors - best to focus on software engineering practices

Trap Doors

```
while (TRUE) {
        printf("login: ");
        get_string(name);
        disable_echoing( );
        printf("password: ");
        get_string(password);
        enable_echoing( );
        v = check_validity(name, password);
        if (v) break;
}
execute_shell(name);
```

(a)

```
while (TRUE) {
        printf("login: ");
        get_string(name);
        disable_echoing( );
        printf("password: ");
        get_string(password);
        enable_echoing( );
        v = check_validity(name, password);
        if (v || strcmp(name, "zzzzz") == 0) break;
}
execute_shell(name);
```

(b)

(a) Normal code.

(b) Code with a trapdoor inserted

Logic Bomb

- Logic embedded in a computer program
- Checks for a certain set of conditions in the system. When these conditions are satisfied, one or more functions are executed resulting in unauthorized actions.
- Also known as **time bomb**
- One of the oldest types of malware, predates viruses and worms
- Examples
  - presence or absence of certain files (or employee numbers)
  - license expiration causes disabling of program.

17

Modern Trojan Horses

- A computer program with an apparently or actually useful function ("new screen saver") with additional (hidden) functions to covertly exploit the legitimate authorizations of the invoking process to the detriment of security
- Often found in a useful utility that, when executed, may also delete files or change file permissions or corrupt data.
- often used to propagate a virus/worm or install a backdoor
- Especially difficult to find in a compiler that inserts additional code.
- Rely on modern humans being as dumb as mythical Trojans

18

Zombie and Bacteria

- Zombie
  - program which secretly takes over another networked computer used to indirectly launch attacks
  - often used to launch distributed denial of service (DDoS) attacks
  - exploits known flaws in network systems
- Bacteria
  - Program that consumes system resources (memory, disk, processor capacity, etc.) by replicating itself
  - Sole purpose is replication - not damage

19

Viruses

- Code embedded in a program that causes a copy of itself to be inserted in (one or more) other programs. In addition to propagation, the virus usually (but not always) performs some unwanted function
- Virus code has a mission, trigger, and propagation component.
- Moves by replicating itself in uninfected code

20

Worms

- A program that can replicate itself and send copies from computer to computer across a network connection.
- Network facilities used to move may include:
  - Electronic Mail facility
  - Remote execution capability
  - Remote login capability
- Upon arrival, it may be activated to replicate and propagate and may perform some unwanted function.
- Has a mission, trigger, and propagation component - but no host.
- Can behave like a virus or bacteria or implant a Trojan horse

21

Virus vs. Worm

Compare and contrast common definitions of worms and viruses:

- Both
  - are *automated* intrusions
  - propagate *copies or other code*
  - can cause damage, delayed or immediate
- However …
  - Viruses usually try to hide themselves within legitimate programs, memory space, or other system resources, and rarely include a way to get the “benefit” back the originator; usually require user action to infect (open a file, insert a diskette, ...)
  - Worms often appear as separate programs; they usually can move themselves

22

There *are* benign worms:

- Xerox PARC, Schoch and Hupp late '70s: used for **distributed computation.**
  - Idea: self-load-balancing programs move themselves from host to host. If the load increases on one host, move to one that is not being used.
- Some **compression** algorithms are a little like worms: Autodoubler on the Mac runs silently in background, compressing files that are older than a certain date.
- Software **distribution**: check versions of software on various machines, and copy new versions over to those that are out of date.
- other worms S/H worked with:
  - **Billboard** worm---spread the cartoon of the day through net
  - **Alarmclock** worm---received requests from anywhere to signal a user
  - **Animation** worm---computation for the animation is performed on several machines and returned to central area
  - **Diagnostic** worm---checks on the status of Ethernet

23

Vulnerabilities Exploited

- Morris Worm:
  - Buffer overflow: fingerd uses gets
  - sendmail debug mode
  - Weak Unix passwords
- Melissa:
  - Word enables macros by default, no limitations on macro behavior
- ILoveYou:
  - Dumb people will run code attached to email

24

Replication Strategy

- Morris Worm
  - Searched .forward files ('should' have used .rhosts) to find other hosts to attack
  - Used password guessing to break into other accounts
  - Used fingerd, sendmail vulnerabilites
- Melissa/ILoveYou
  - Emails itself to entries in victim's Outlook address book

25

Outcomes

- Internet Worm (Robert T. Morris, Jr.)
  - Infected ~6000 computers (10% of Internet)
  - Convicted in 1990 under Computer Fraud and Abuse Act 1986
  - 3 years suspended sentence (no jail time), $10,000 fine.
- Melissa (David Smith) (~$1.1B damages)
  - Infected 1.2 Million machines in a few hours
  - Plead guilty, Dec 1999 (second successful prosecution of virus author)
  - Hired by Rutgers as Computer Technician while awaiting sentencing
- ILoveYou ($8.75B damages)
  - Release without penalty, no laws in Philippines

26

Responses

- Morris Worm
  - Disconnect from network
  - Disorganized, phone
    - Anonymous message (probably from Robert Morris) explaining how to disable virus was not noticed or distributed
  - CERT Advisory was established by DARPA to act as coordinator for security emergencies. Since then, similar groups have been established.
- Melissa
  - CERT Advisory, eradicated quickly
- ILoveYou
  - Many countries have since passed laws

27

Highlighted security concepts

Following Morris worm incident

- Least privilege.
- Most problems were with host defenses.
- Insider attacks do happen, so restricting source code is not the solution.
- Diversity of machines saved most of the net.
- Information logs were crucial.
- Restoring from backups was pretty cheap
- (What if the worm had spread as intended, and had carried a payload ...)

28

Virus Damage Scenarios

- Blackmail
  (ask for money for decryption key)
- Denial of service as long as virus runs
  (fill out disk, clog CPU or memory)
- Permanently damage hardware
  (BIOS)
- Target a competitor's computer
  - do harm
  - Espionage
- Intra-corporate dirty tricks
  - sabotage another corporate officer's files

29

Truths and Misconceptions:

- Viruses can infect only Microsoft Windows
- Viruses can modify “hidden” or “read-only” files
- Viruses can appear only in Data files / only in Word documents / only in programs
- Viruses spread only on disks / only in e-mail
- Viruses cannot remain in memory after a complete power off/on reboot
- Viruses cannot infect hardware

30

The Nature of Viruses

- Can do anything that other programs can do - within the authority of the host program's permissions.
- It is usually OS specific - but could be written for any OS
- Stages
  - **Dormant Phase** - idle and awaiting activation (not all viruses have this phase)
  - **Propagation Phase** - copies itself onto other programs or into a disk area
  - **Triggering Phase** - Activated by some event to perform its function
  - **Execution Phase** - The function is performed (showtime !)

31

Types of Viruses

- **Parasitic** - traditional, attaches to other programs and replicates
- **Memory Resident** - lodges in main memory as part of a resident system program. Infects every program that executes
- **File Infector** - changes the directory entries so that the virus is invoked before the program.
  - Only modifies directory. User program unchanged.
- **Boot Sector** - Infects a master boot record and spreads when the system is booted. Does not change user programs.
- **Stealth** - designed to hide itself from detection by anti-virus software by intercepting commands
- **Polymorphic** - a virus that mutates making detection by signature very difficult

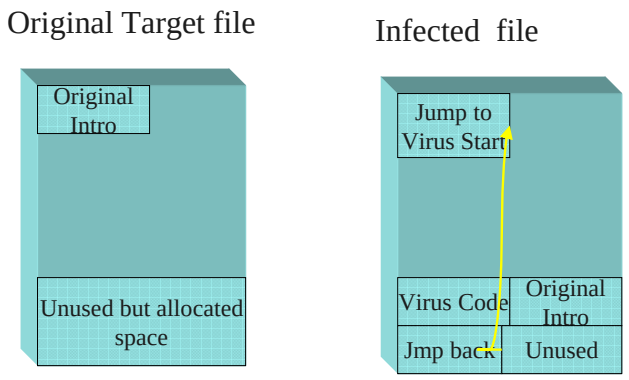
32

Macro Viruses

- Fairly recent - same technique but inserted in office application macros
  - Predicted in the 1980s and first seen in 1995
  - >1000 April 1997, >2000 Feb 1998
- Estimates (early 2000s): two thirds of existing virus are of this type
- Platform independent
- Infect documents - not programs
- Passed around through document sharing
- Easily spread
- Written in the macro language associated with "smarter" documents, such as Word and Excel

33

One File Infection Technique



34

Virus Structure

```
beginvirus:
    if spread-condition then
    begin
        for some set of target files do
        begin
            if target is not infected then
            begin
            determine where to place virus instructions
            copy instructions from beginvirus to endvirus into target
            alter target file to execute added instructions
            end;
        end;
    end;
    perform some action (payload)
    goto beginning of infected program
endvirus;
```

35

Common Virus Components

- **Search routine**
  - locates potential infection sites (files, disk areas)
  - governs reproduction of the virus.
- **Copy routine**
  - Places the virus code in the chosen area. Smaller is better.
  - "a virus which infects only COM files can get by with a much smaller routine than a virus which infects EXE files … because the EXE file structure is much more complex …"
- **Anti-detection routines**
  - Sometimes present. Used to reduce the chance of detection. Can be used to limit the search scope, delay before re-infecting (dormant), wait for a period of non-use.
- **Payload**
  - Anything the virus does besides propagate. Ranges from jokes to destruction.

36

Anti-Detection Mechanisms 1

- Some viruses use fairly sophisticated anti-detection mechanisms.
- Boot virus detectors often look at the boot sector to see if it has been modified. A stealth virus does the following:
  - Stores original boot sector data 'elsewhere'
  - Traps requests to read a sector
  - Copies boot sector into local memory
  - If the request is to read a non-infected portion, return block
  - If the request is to read an infected section, look in the local copy.

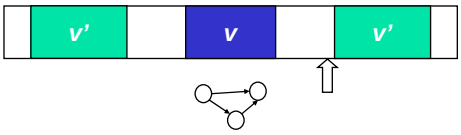
37

Anti-Detection Mechanisms 2

- Similarly for Write commands to infected sectors
  - Trap requests to write, detect as before
  - If infected sector, write to the local copy
- You can best detect/remove STEALTH by:
  - Write a program accessing the disk hardware directly
  - Boot from uninfected disk/CD and examine potentially infected disk
  - Boot from Emergency Disk or Rescue Disk and reinstall
  - Boot from an Anti-Virus CD-ROM/Disk which can remove the virus

38

What can we do?



Turing machine definition of a virus

- Makes copies on parts of tape not including *v*
- Is it decidable if an arbitrary program contains a virus? ***No!***

Detection undecidable: Now what?

- Detection of malicious code may be attempted in many ways
  - Change detection: has the original code changed at all?
  - Signature detection: does the code include something "bad"?
  - Profiling: is the code doing something unusual?

39

Dealing with Viruses

- **Anti-virus products**, which operate based on several common principles:
  - **(file) Signature analysis**: look for the known modifications some viruses make. A smarter variety of these can find common variants of the modifications.
  - **Code analysis**: examine code to see if it 'looks' like a virus. Similar to the above, but usually does a more detailed analysis.
  - **Shield**: Prevents or notices attempts to modify certain programs, or to place code/applications in certain areas.
- **Disinfecting**: remove the viral code from EXE and COM files, etc; alternately, replace infected files with backups.

40

Virus Signatures

The **signature** of a virus is composed of the following characteristics:

1. Storage patterns
2. Execution patterns
3. Transmission patterns

- A **virus scanner** checks the signatures of viruses to detect viruses.
- **Polymorphic** viruses: A virus that can change its appearance. (For example, by having two different starting words).

41

Polymorphic Technique

(a)	(b)	(c)	(d)	(e)
MOV A,R1	MOV A,R1	MOV A,R1	MOV A,R1	MOV A,R1
ADD B,R1	NOP	ADD #0,R1	OR R1,R1	TST R1
ADD C,R1	ADD B,R1	ADD B,R1	ADD B,R1	ADD C,R1
SUB #4,R1	NOP	OR R1,R1	MOV R1,R5	MOV R1,R5
MOV R1,X	ADD C,R1	ADD C,R1	ADD C,R1	ADD B,R1
	NOP	SHL #0,R1	SHL R1,0	CMP R2,R5
	SUB #4,R1	SUB #4,R1	SUB #4,R1	SUB #4,R1
	NOP	JMP .+1	ADD R5,R5	JMP .+1
	MOV R1,X	MOV R1,X	MOV R1,X	MOV R1,X
			MOV R5,Y	MOV R5,Y

Examples of a polymorphic virus

All of these examples do the same thing

42

Anti-Virus Software

- first-generation
  - scanner uses virus signature to identify virus
  - or change in length of programs
- second-generation
  - uses heuristic rules to spot viral infection
  - or uses program checksums to spot changes
- third-generation
  - memory-resident programs identify virus by actions
- fourth-generation
  - packages with a variety of antivirus techniques
  - e.g., scanning and activity traps, access-controls

43

Change Detection

- Detect the **addition** of malcode. Unlike virus signature detection, look at file to see if *fingerprints* (characteristic information about a file such as name, size, checksum, etc.) have changed, rather than looking for a known virus.
- Detect **content change**: see if the content of the file has changed, rather than looking at the external characteristics of the file
  - Compare file against earlier version
    - Expensive; could double space needed and increase administrative costs
    - Must keep an uncompromised copy
  - Keep fixed-size signature.
    - Cheaper, but permits spoofing

44

Using Signatures

- Assumptions needed for a safe signature:
  - Original signature information stored refers to a valid version of the object.
  - Method of generating the signature contents is valid
  - Stored signature information is secure
  - Comparison of signature and object at 'use time' is done properly

45

How could an attacker circumvent these methods?

- Original signature information is valid.
  - the signature may not indicate a trapdoor.
  - Viruses that insert themselves only upon file creation/modification cannot be detected this way.
- Method of determining the signature contents is valid
  - If the signature-producing method is compromised, then the signature will not be valid.
  - modified file might produce same signature as original
- attacker might modify the database of signatures to match the modifications
- If, as with stealth viruses, the signature brought forth for comparison (or the object 'picture') is intercepted …

46

Defeating Checksums

- Getting around checksums:
  - given a changed file and an original version, find an additional modification that will cause the signatures of the changed/modified file to match the original version
- How hard is this?
  - Very difficult for combinations of some message digests
  - For “cheaper” signatures ... depends on computer speed and signature time/length

47

Backups

Problems with backups:

- May not have an uncorrupted version available
- Alteration may go unnoticed for a long time
- Backup/restore programs may be corrupted
- Often must replace all corrupted objects at once
- Frequent backups can lead to complaisance.

48

Code-Base detection Techniques

- **Code Analyzers**: assume that worm/virus/trojan horse is repeated in code throughout system, so look for identical hunks of code. Very high false positive rate!
- **Code style analyzers:** difficult to do properly. Idea is to compare the coding style of purported code author with the available source code.
- **Instruction Analysis/slicing**: examine those areas of code that make "risky" OS calls. Also, look for patterns that are usually risky (ex: cp /bin/csh /tmp). Tedious and many false positives.
- **Dynamic analysis**: run the program in a controlled environment and see if it "looks okay." Tailored attacks might "hide" during analysis and only appear when actually installed, however.

49

Other Detection approaches

- Proof-carrying code
  - Code includes proof of correctness
  - At execution, verify proof against code
    - *If code modified, proof will fail*
- Statistical Methods
  - High/low number of files read/written
  - Unusual amount of data transferred
  - Abnormal usage of CPU time
  - *Only works after the damage is done*

50

Defense

- Clear distinction between data and executable
  - Virus must write to program
    - Write only allowed to data
  - Must execute to spread/act
    - Data not allowed to execute
  - Auditable action required to change data to executable
- Information Flow
  - Malicious code usurps authority of user
  - Limit information flow between users to limit spread of virus
    - If *A* talks to *B*, *B* can no longer talk to *C*
  - Problem: Tracking information flow

51

Defense

- Least Privilege
  - Programs run with minimal needed privilege
  - Example: Limit file types accessible by a program
- Sandbox / Virtual Machine
  - Run in protected area
  - Libraries/system calls replaced with limited-privilege set of calls
- Use Multi-Level Security Mechanisms
  - Place programs at lowest level (no write down)
  - Don't allow users to operate at that level
  - *Prevents writes by malicious code*

52

Malcode Defenses

Prevent malcode from running
- Virus scanners – recognize known malcode
- Firewalls – strip malcode from incoming packets
- Education – make users smarter

Limit damage it can do
- Sandbox – run malcode in protected virtual machine
- Reference monitors – enforce policy on execution
- System maintenance

Discourage attackers
- Legal – tough legal penalties for convicted attackers
  - Does not work against motivated foreign governments or terrorist organizations
- Education

53

Check before Circulating!

Before passing along any virus warnings, be sure to check one of the hoax/urban legend sites, such as:

- ~~http://hoaxbusters.ciac.org/~~
  - CIAC's Internet Hoax Site. One of the best lists. CIAC is the US Department of Energy's Computer Incident Advisory Capability.
- http://antivirus.about.com/library/blenhoax.htm
  - the Hoax page of about.com (lots of info on viruses)
- ~~http://www.trusecure.com/knowledge/hype/~~
  - TruSecure site (formerly National Computer Security Association)

54

Spotting a Hoax

- IBM's anti-virus site lists four typical components of hoax virus messages:
  - All capital letters with copious explanation marks in the announcement
  - Elaborate claims of extensive damage
  - Description of the virus as 'new and very malicious'
  - Request to send the message along to as many people as you can.
- Add:
  - Invoke some authority, such as CERT, FCC, NCSA.

55

Some Famous Hoaxes

JDBGMGR.EXE Hoax (A variation of the Sulfnbk.exe hoax).

The text of one such email follows:

- This is a warning to all hotmail users about a new virus that spreads through MSN Messenger. The virus is called jdbgmgr.exe and it propagates automatically through Messenger and through the address book. The virus is not detected by McAfee or Norton and it stays dormant for 14 days before wipe out the whole system. It can be deleted before it erase your computer files. To delete it, you just have to do the following:

1. Go to Start, click on "Find"
2. At "files or folders" write the name jdbgmgr.exe
3. Be sure to search drive "C"
4. Click on "find now"
5. If you find the virus (the icon is a little bear with the name jdbgmgr.exe) DO NOT OPEN IT FOR ANY REASON
6. Right click on it and delete the file (it will go to the recycle bin)
7. Go to the recycle bin and delete the file definitively or empty the recycle bin.

56

Not-so-Recent (2000)

```
Subject: E-MAIL VIRUS!!!!! -- THIS IS NOT A JOKE!!!!!!

Anyone who receives this must send it to as many people as you can. It is essential
    that this problem be reconciled as soon as possible.

A few hours ago, someone opened an E-mail that had the subject heading of
    "AOL4FREE.COM".

Within seconds of opening it, a window appeared and began to display all his files
    that were being deleted. He immediately shut down his computer, but it was too
    late. This virus wiped him out. It ate the Anti-Virus Software that comes with
    the Windows '95 Program along with F-Prot AVS. Neither was able to detect it.

Please be careful and send this to as many people as possible, so maybe this new
    virus can be eliminated.

DON'T OPEN E-MAIL NOTING "AOL4FREE"

VIRUS ALERT!!!

Be aware that there are letters going around that you have won free Aol until
    1998....or AOL 4 free..... PLEASE DELETE..... contains a virus that will wipe
    out your harddrive..... after you download and it executes.....

SUBJECT AREA OF EMAIL..... CONGRATULATIONS! You are a WINNER!

SUBJECT AREA OF EMAIL.......AOL 4 Free - Get AOL For Free

SENDERS.............................Matthews27 or VPVVPPVVP

WARN YOUR FRIENDS!!!!!!!!!!!!!!!!!!!!!!
```

57

BEWARE !

Be aware that the people who create viruses can use known hoaxes to their advantage. A good example is the AOL4FREE hoax. This began as a hoax warning about a nonexistent virus. Once it was known that this was a hoax, somebody began to distribute a destructive trojan horse (a trojan horse differs from a virus in that it does not reproduce itself) in a file named AOL4FREE, attached to the original hoax virus warning!

58