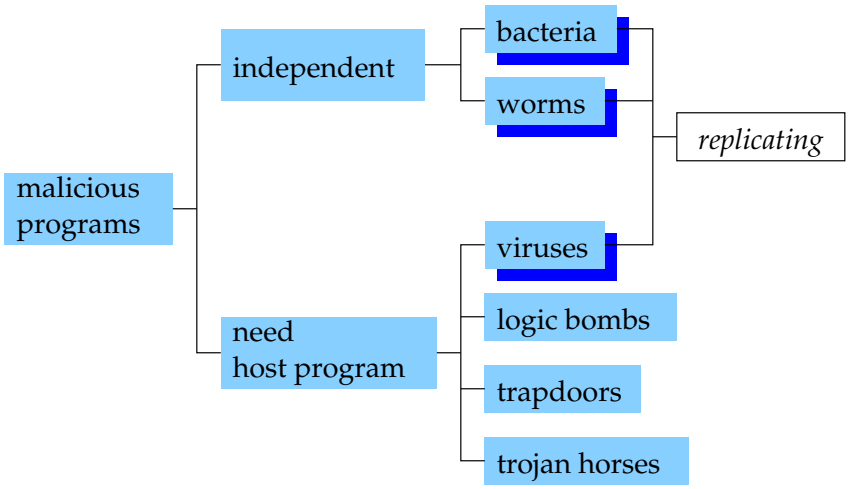
CSCI 7000-001 — Malicious Programs

November 13, 2001

A *malicious program* is a piece of software that

- destroys or consumes valuable resources, or
- exposes, creates, or installs vulnerabilities in a computing system

1 Taxonomy of malicious programs



2 Trapdoors

Remember the movie "War Games"…
A trapdoor is a secret (or simply undocumented) entry point into the *host* program.

- used by developers to test and debug their programs
- introduced to allow remote or direct management of the program
- introduced with malicious intentions

3 Logic Bomb

Logic bombs predate viruses and worms. They are integrated parts of the *host* program that are normally executed with it. Logic bombs are set to "explode" (i.e., destroy some valuable computing resources) when a certain condition is met.

4 Trojan Horses

- useful (or attractive) program
- contains hidden malicious code
- perform destructive actions, or
- expose or create vulnerabilities

5 Viruses

A *virus* is a malicious piece of code that

- can replicate itself, and attach itself to (or *infect*) other programs or documents
- usually performs some more or less destructive actions

Like other forms of malicious code, viruses can do anything that the host program can do, including destroying or modifying files, and exposing or creating vulnerabilities.
The main difference is that viruses implement an infection mechanism.

6 Bacteria (or *Rabbits*)

Bacteria are also self-replicating programs, however they execute independently of any *host* program.
Bacteria usually execute as separate threads/processes on multi-threaded systems. By forking off new copies, bacteria end up consuming all system resources.

7 Worms

- sophisticated self-replicating programs
- spread across networked machines
- use network services and *daemons* to infect other systems

8 Virus life-cycle

A typical virus goes through the following stages:

- *dormant phase*: the virus is idle. The virus does not even replicate itself in this phase. The propagation may be activated by some time-dependent conditions or other system conditions. Not all viruses have this phase
- *propagation phase*: the virus infects other executable programs or documents
- *triggering phase*: the is activated to execute its function on the victim system. Again, any condition may be used to activate the virus
- *execution phase*: the virus executes its function on the victim system.

9 Anatomy of a virus

The replication function of viruses exploits specific features of the host system (e.g., binary formats, scripting languages, specific directories where other executable programs can be found), therefore very tightly bound to a specific platform.
Here we examine the generic structure of a virus:

```
proc virus() {
        goto main
        // virus signature (guard)
                12345678901234567
        main:
                // infection routine
                loop:
                        file := select_random_executable
                        if file contains 12345678901234567
                        then goto loop
                        else prepend virus to file
                // trigger condition
                        if trigger_condition = true
                // malicious function
                        remove all your files
}
```

10 Anti-virus approach

- *detection*: determines which executables have been infected
- *identification*: identifies the specific virus that has infected a program

- *removal*: restoring a functioning copy of the host program

First-generation virus scanners performed detection and identification by looking for virus *signatures* in executables. Some also provided some form of repair of the executable.

11 Camouflage techniques

- *stealth viruses*: use compression to maintain original size for the host program.

  CRCs (or cryptographic hash) can be used as an integrity code to defeat compression methods. However other more sophisticated techniques are possible: the virus could (on some operating systems) intercept some I/O system calls, and present the file in its original form to every reader.

- *polymorphic viruses*: the virus infection process creates functionally identical mutations of the virus. Even better, the virus could use some simple cryptography to create ever-changing encrypted versions of itself.

12 Type of host environment

The most common platform for viruses is now executable *macros* embedded in documents (e.g., MSWord documents).

- it is relatively more platform-independent (although it affects

E-mail creates a great vehicle for the exchange of infected programs and documents. Notice that e-mail technology creates two different classes of vulnerabilities:

- e-mail is a very cheap "push" mechanism for attackers, therefore it is an inherent a vulnerability

- e-mail clients may implement naive security policies. In particular, clients may make implicit and automatic decisions on how to handle specific e-mail attachments, that may result in serious threats

13 Advanced anti-virus techniques

- using heuristics in addition to exact pattern matching, for example look for specific code fragments in crucial areas of the host program

- separate integrity checks using cryptographic hash functions. Example: `rpm -V`

- memory-resident monitors. The idea is to detect the presence of an active virus from its actions rather than from looking at possible host programs

- ...and of course, a decent operating system would help a lot too