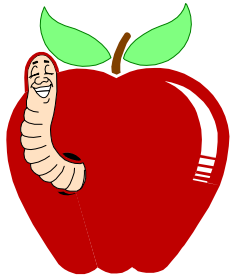
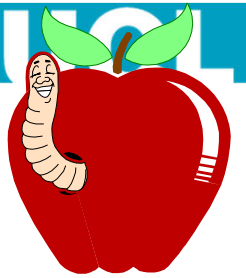
Malicious Software

Nicolas T. Courtois
- University College London

Sources of Problems



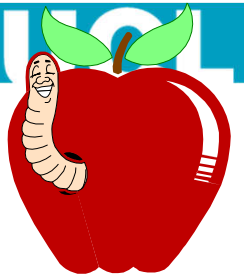


Technology Failures

Security as an afterthought.

- Programming developed with absence of security.
  - C/C++ is unsafe (Microsoft is blacklisting big chunks of standard C: could have happened 30 years ago).
  - Security/cryptography research developed with obsession with security. Both never met.
- Windows developed first,
  - networking developed later…
- Structural defects in the OS design
  - lack of multi-layer defense strategy

    [also in Unix: seems even worse than Windows]



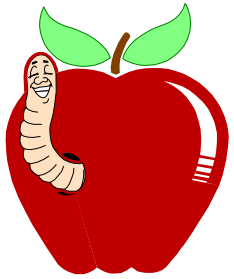
+ Failures In Operation

- Over-privileged users
  - Windows 98: all users can modify system files and system memory
  - Unix – over-powerful root, >21 critical capabilities in one entity
- Over-privileged code
  - code executed by a user to access all rights of that user
  - Windows Vista: worse than that: built-in privilege escalation:
    - if name contains setup, will run with many admin-level capabilities

Human Cognitive Failures

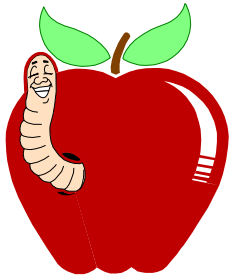
- Mystified:
  - security issues are probably always exaggerated and distorted, one way or another (downplayed OR exaggerated,
    Ross Anderson: “hypertrophy” of security
  - Also a huge demand, but both don’t meet to frequently).

- Lack of people that would defend the public interest

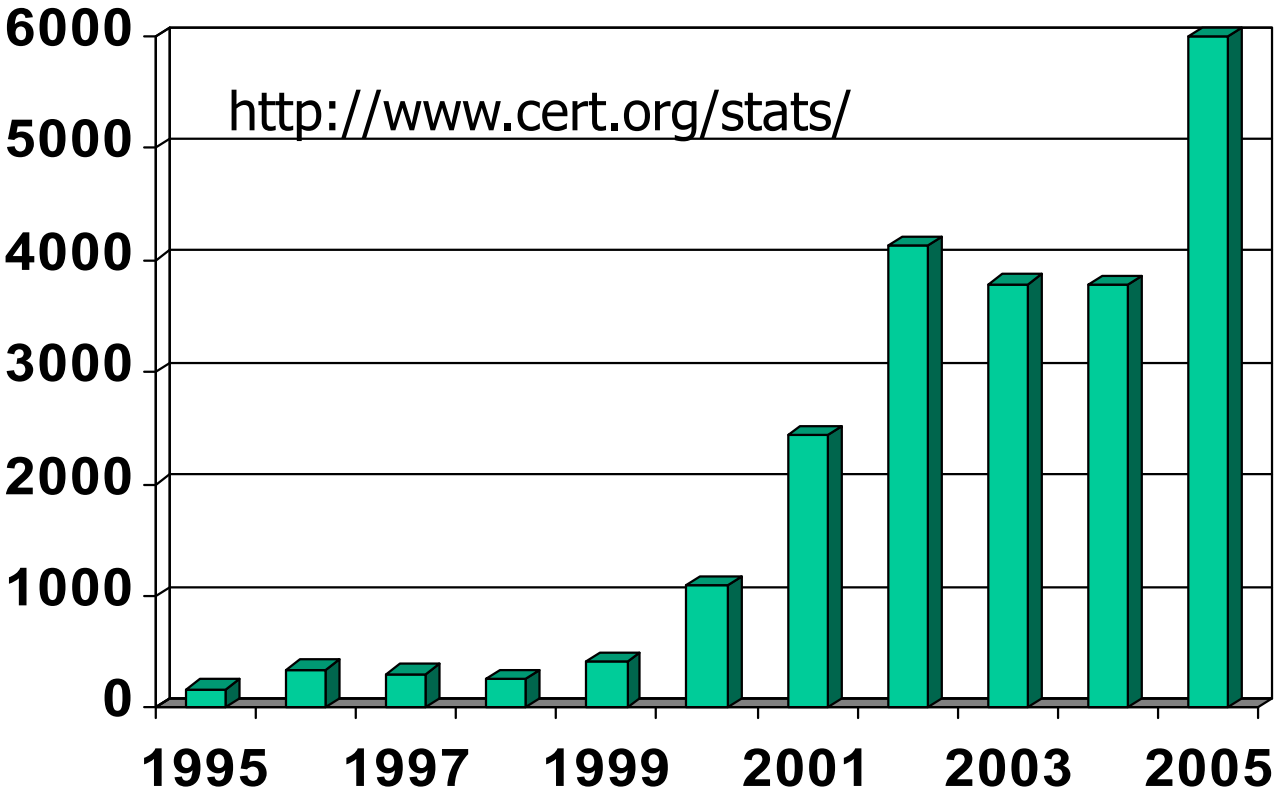


Market Failures

- Economics/Business:
  - many things just don't matter at all!
  - customers do not see => do not care about security
    - "market for lemons"
  - externalities, cost shifting
    - losses affect many "small" people that don't react…
      - people will not even switch to another software…
      - unable to defend themselves
      - 1 billion x very small loss
    - usability: user burden, businesses don't care

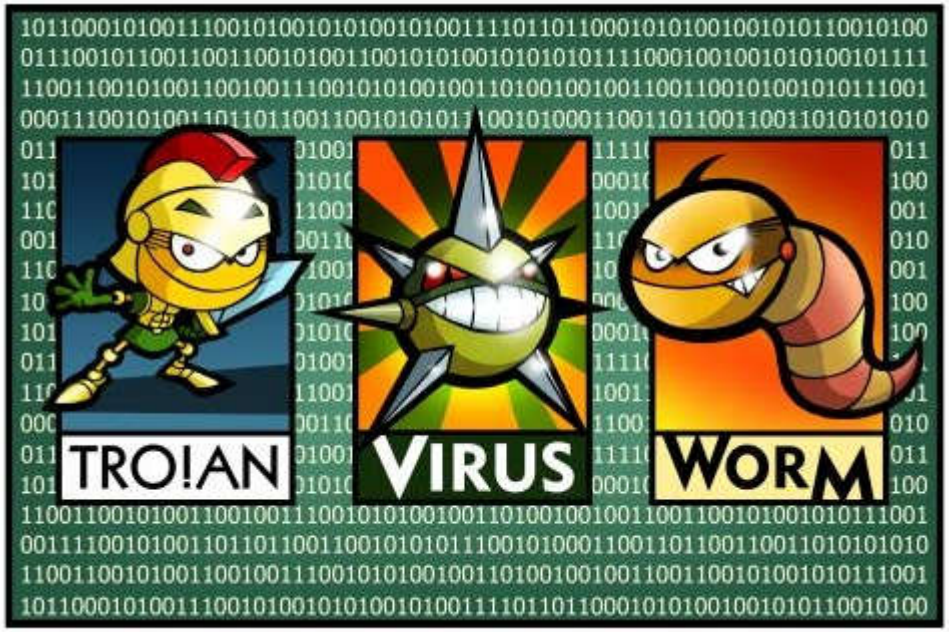


Explosion of Known Vulnerabilities

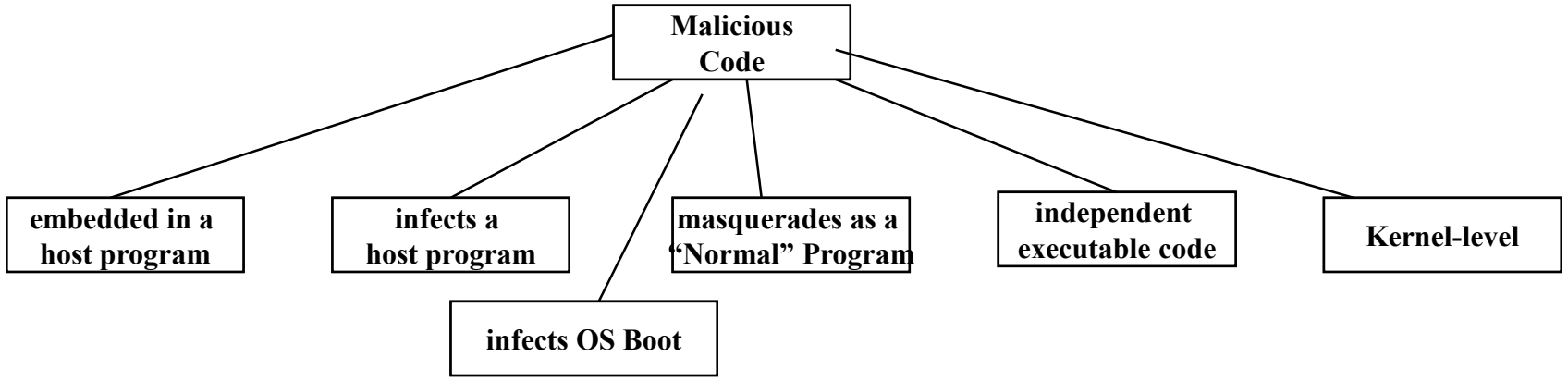


What about the unknown ones?

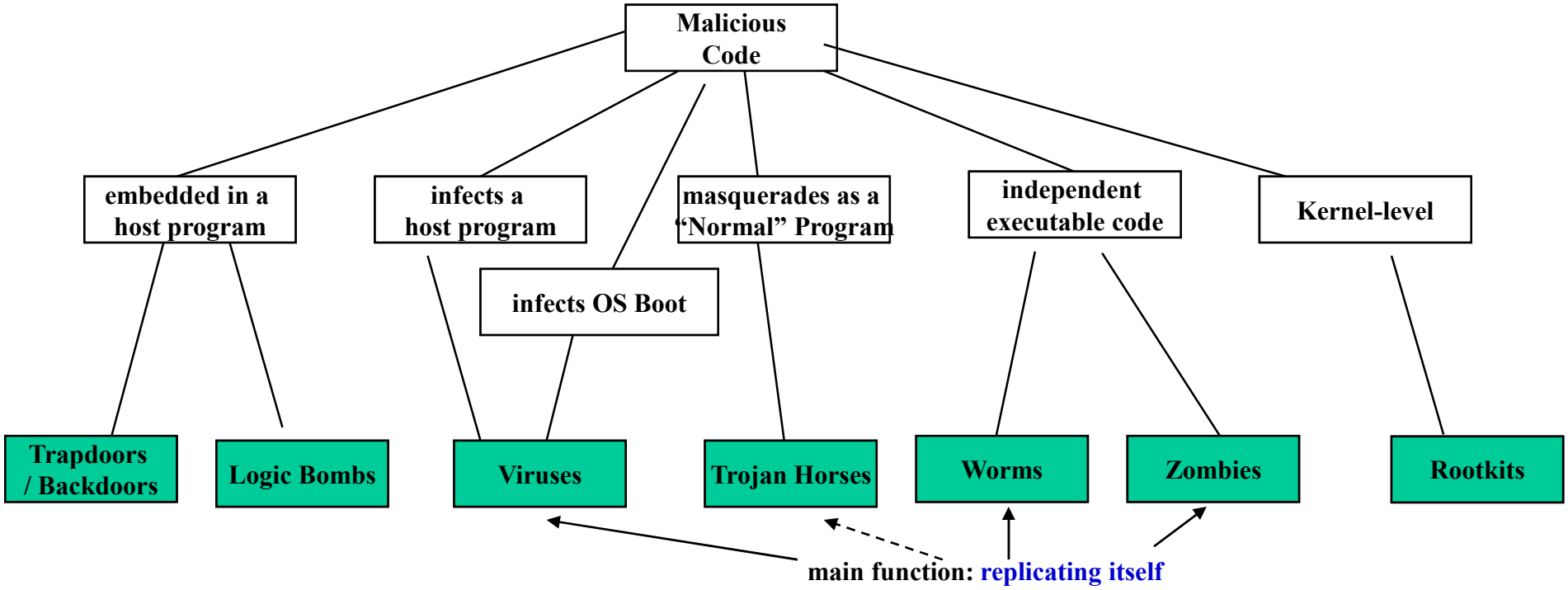
Taxonomy of Malicious Software



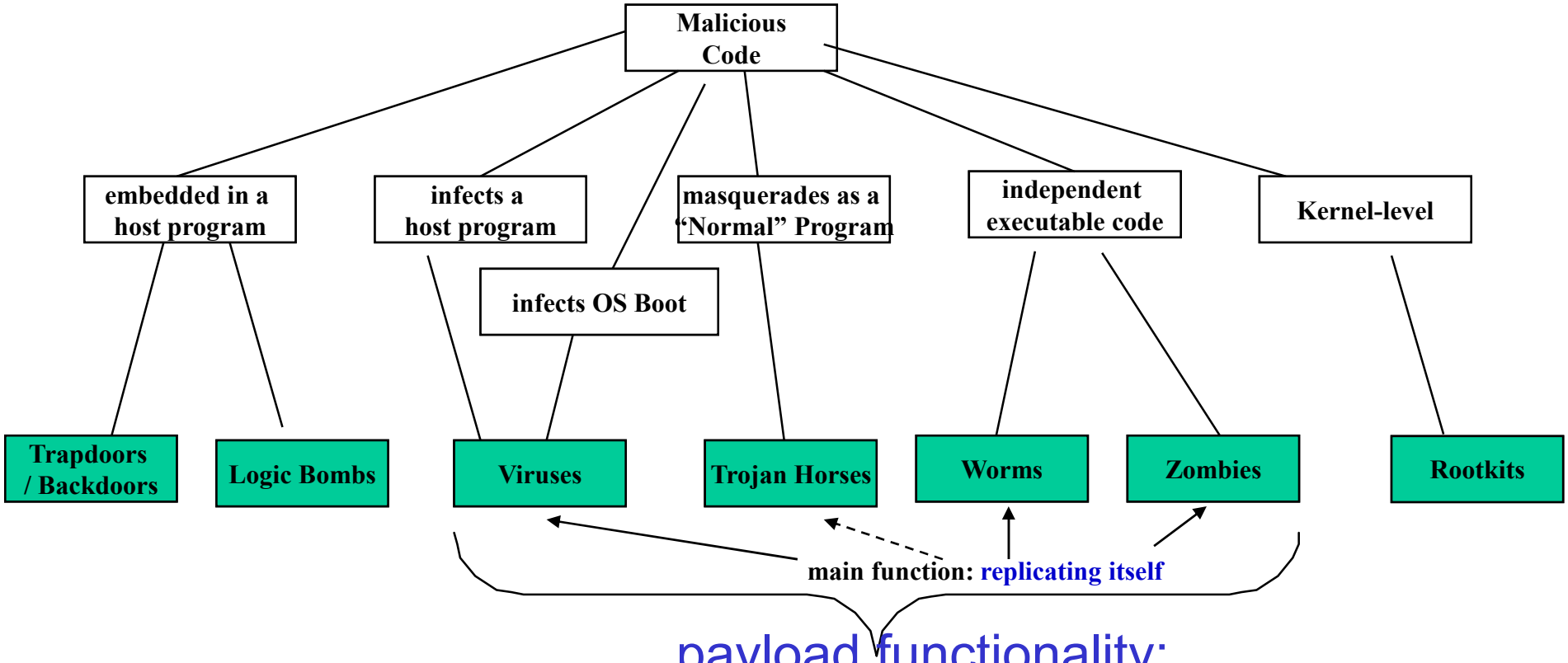
Vectors of Infection



Software-Borne Threats



Infection + Payload



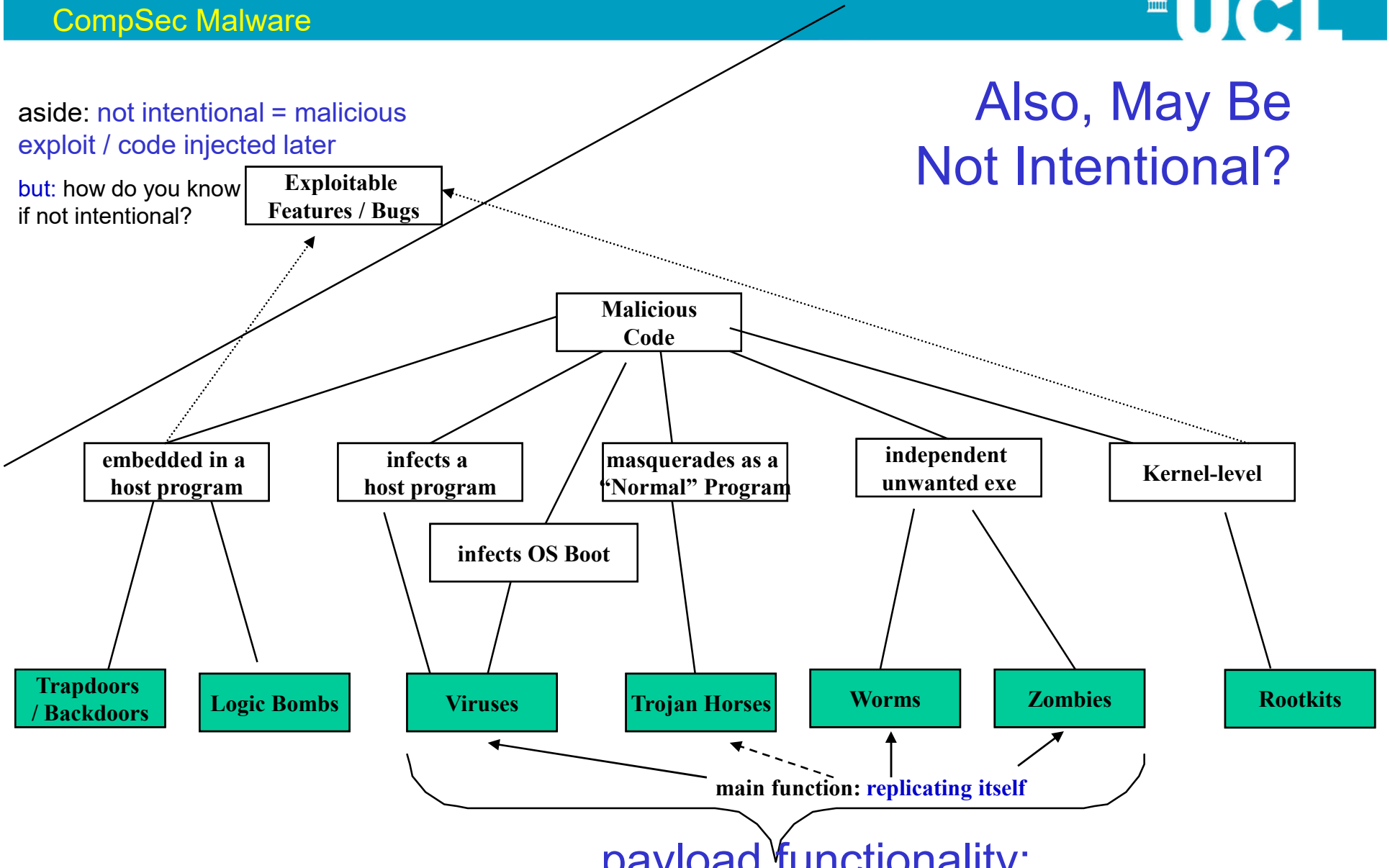
payload functionality:

Harmful	Stealing data	Key Loggers	DDOS/ Bots	Dialers	CPU cycles	download /install more malware

Also, May Be Not Intentional?

aside: not intentional = malicious exploit / code injected later

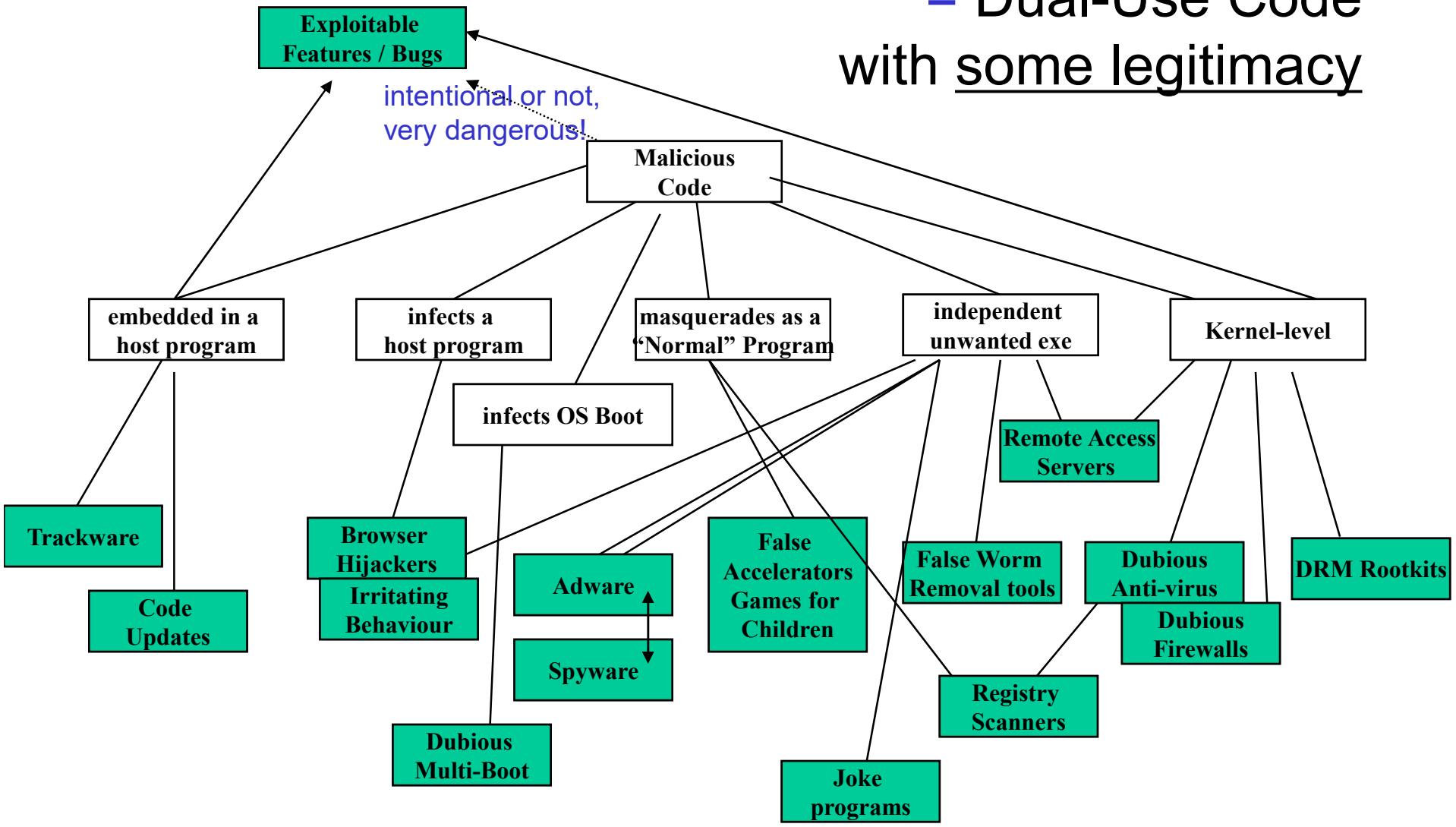
but: how do you know if not intentional?



payload functionality:

Harmful | Stealing data | Key Loggers | DDOS/ Bots | Dialers | CPU cycles | download /install more malware

More "Grayware" = Dual-Use Code with some legitimacy



Crimeware

Malware vs. Crimeware:

1. same infection methods,
2. different goals,

⇒ more specific forms of payload,

⇒ automation of crime,

⇒ malware as illegal business venture:

Example: Keyloggers and Spyware

⇒ tailored for stealing passwords [e.g. Bitcoin wallet] and credit card numbers

Detailed Definitions

Hidden Mechanisms Embedded in Original Software

Trapdoor, Backdoor

Hidden function that can be used to circumvent normal security. A hidden entry point into a system.

- also, can be a hidden feature leaking some data… (backdoor).

- Examples:
  - Special user id or special password
  - Special instruction / option / keyboard sequence
  - Etc…
- Commonly used by developers
  - "insecurity by obscurity" ☺
  - hard to distinguish legitimate reasons (testing, debugging, circumventing some bug, jokes and Easter Eggs) from intentional security compromise
  - beware: can be included in a compiler as well…
    - source code will not help then…
    - Rice Theorem : source code will not always help…

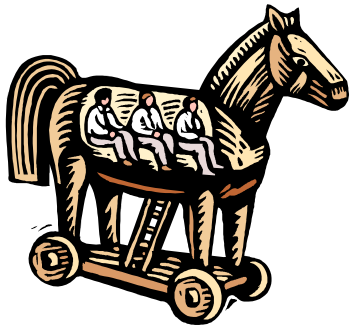
this way

Logic Bomb

- A malicious feature that will be activated when certain conditions are met
  - e.g., presence/absence of some file;
  - particular date/time
  - particular user
- when triggered, typically will do some harm
  - modify/corrupt/delete files/OS, etc.

Trojan Horse

- Program has an overt (expected) and covert (malicious and unexpected) effect such that
  - works / appears to be a normal program,
  - covert effect violates the given security policy
- User is tricked into executing a Trojan
  - does the usual (overt) job
  - covert effect is performed with user's rights/authorization level.

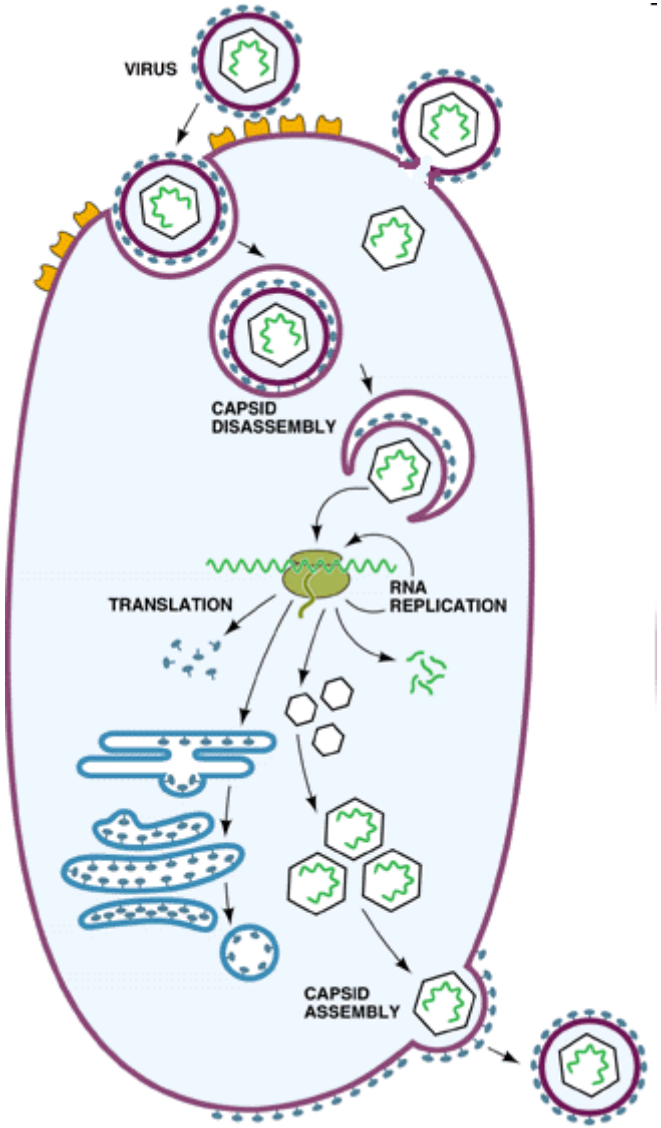


Virusology

Viruses

In biology, a virus is a piece of DNA/RNA+some proteins.

- once present in the cell, it will force the cell to produce copies of itself.
  - not a living creature, cannot survive alone.



Computer Virus:
term coined in 1984
by prof. Leonard Adleman
(A from RSA).

Viruses – Main types

1. Add-On Virus = Appending Virus => most viruses
2. Shell Virus (nothing to do with Unix shell)
3. Intrusive Virus

Common Features of Viruses

- no overt action
  - tries to remain totally invisible
- self-replicates
  - potentially unlimited spread
  - can have some predefined strategy and predefined target

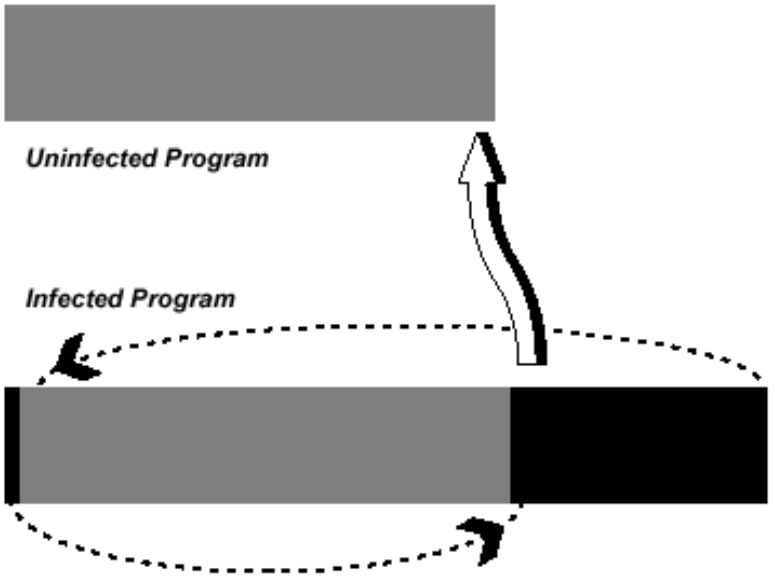


Figure 2: Add-on Virus Infection

Payload

Payload: frequently a virus performs additional malicious actions

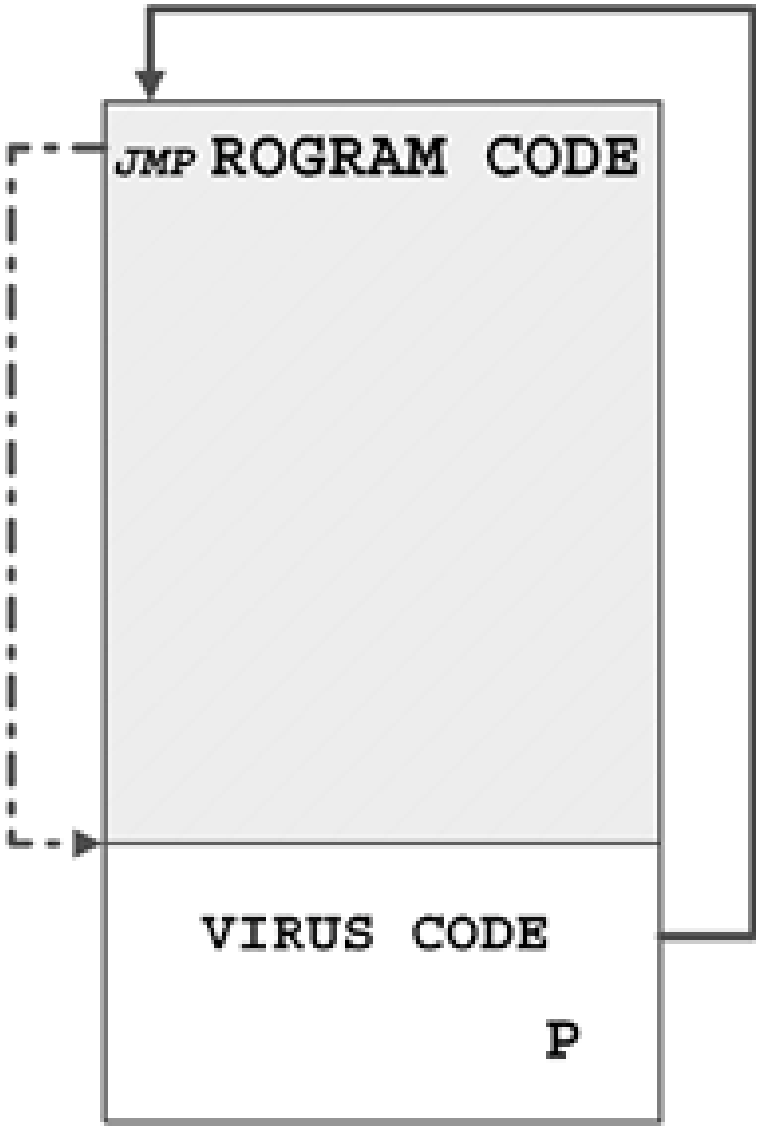
- except “zero payload” viruses
- just harmful actions
- execute / download additional code

Virus Life Cycle Elements

- Dormant phase: idle
- Propagation phase
- Triggering phase: the virus is activated to:
- Execution phase: perform the payload functions

Add-On Virus = Appending Virus

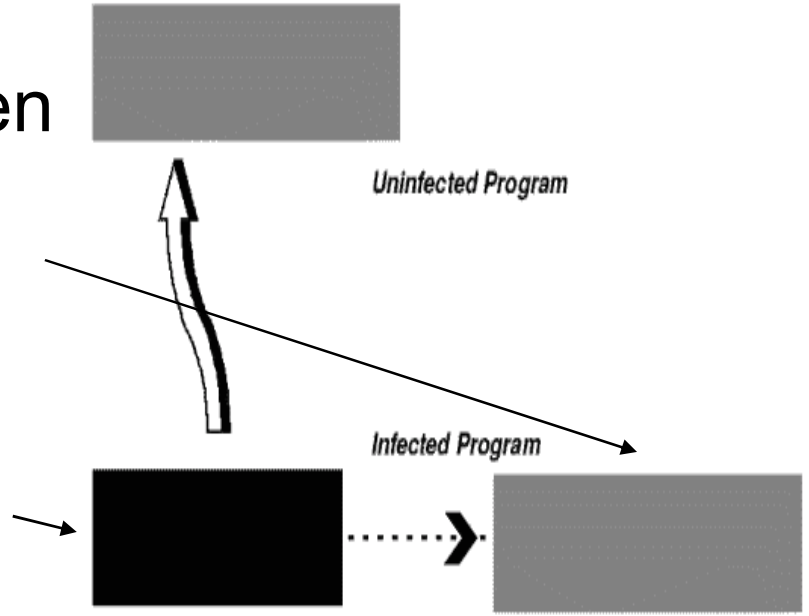
- attaches itself to (any) other exe program (host program)
  - typically 200-4000 bytes
- operates when infected exe file is executed



Shell Virus

ambiguous misleading name:

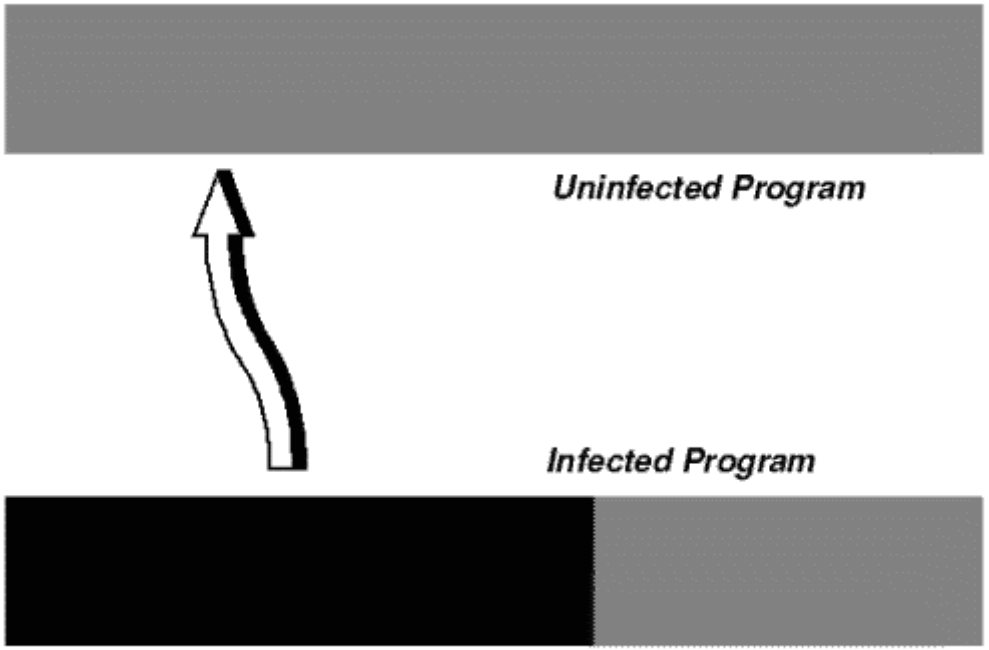
- little to do with Unix shell
- “wrapping around” a given program or system call



Intrusive Virus

Does not append, rather modifies the program itself and changes the functionality of this program.

Cannot be removed if we don't have the original copy…



Viruses by Medium of Infection

- Exe file infectors
- Boot infectors
  - hard drive boot
    - master boot record (MBR), OS indep
    - OS loader hijack, 1 partition
  - CDROM autorun hijackers
  - USB stick autorun hijackers
- Half way before system starts:
  - OS libraries hijack (e.g. some dll loaded early)
  - driver hijackers
- Data file infectors
  - Macro Viruses
  - format string exploits (not called viruses)

load before any anti-virus software

Additional Infection Mechanisms

- Terminate and Stay Resident = TSR
  - since MS-DOS..
  - stays active in memory after application exits
  - can then infect other targets, for example
    - can trap OS calls that execute any program…

Virus Defenses

More about this later.

Oldest methods:

- Black-list:
  - signature-based detection.
- Track changes to executables:
  - Tripwire, hash functions, MACs etc…

Virus Self-Defense

"Stealth" Viruses – avoid detection

- conceal code:
  - Pack/compress/encrypt virus
  - Polymorphism
    - constantly change virus code
- conceal actions
    - mimicry: imitate other programs
    - associated rootkit prevents detection
    - watchdog program
    - disable or disturb anti-virus software
    - remove itself after job done, such as creating 2 copies elsewhere

Self-defence technology	Relative frequency
Packers	47.8%
Code obfuscation	14.3%
Rootkits	14.3%
Disrupt security software	9.6%
Virtual machine detection	4.8%
Auto-click	4.8%
Anti-debugging	1.9%
Polymorphism	1.0%
Block file access	1.0%
Modify hosts file	0.5%
Using streams	0.1%

Table 6. Relative prevalence of malware self-defence technolgies identified by Shevchenko [13].

Macro Viruses

- infected a data file (e.g. word)
  - relies on macros interpreted by some application
    - application-dependent
    - can be OS-independent
      - Example: Microsoft Word: MAC and Windows

Independent / “More Sophisticated Forms of Life”

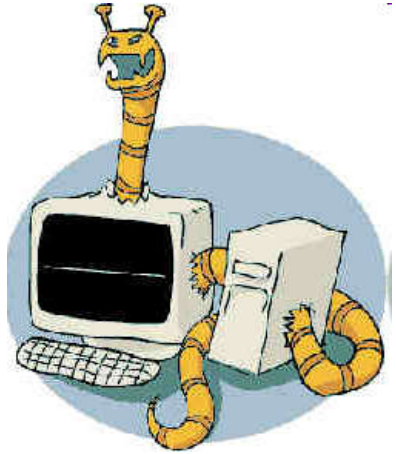
*Bacteria

Bacteria:

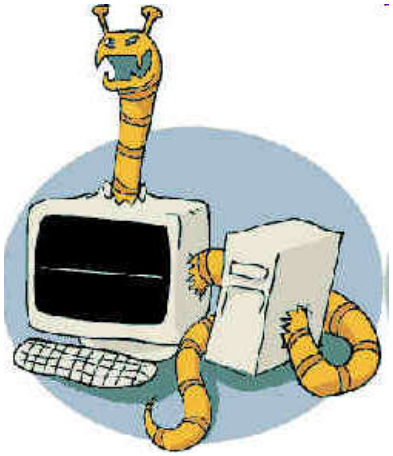
simple functionality,
program that replicates until it fills all disk space, all memory, all CPU cycles

Worms

- introduced by Shoch and Hupp in 1982.
- runs independently,
  - no host program
  - infects a host machine
  - propagates in a network
    - a fully working version of itself copied to another host machine
    - spreads totally without human intervention ≠ virus



more worms



A worm has two main components:

- an exploit
  - usually exploits web servers
    - or other exposed "DMZ-style" components
- a "payload" of hidden tasks
  - backdoors, spam relays, DDoS agents, etc.

Life cycle phases:

probing → exploitation → replication → running the payload

Zombie Network = Botnet

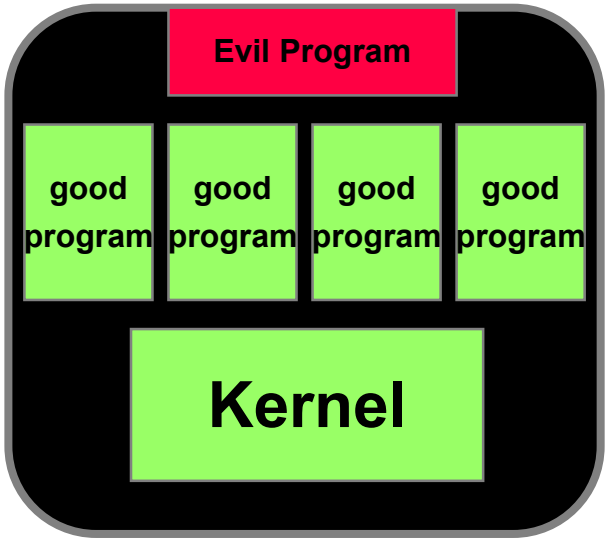
- Secretly takes over another networked computer by exploiting software flaws
- Connect the compromised computers into a zombie network or botnet =
  - a collection of compromised machines
    - running programs such as worms, Trojan horses, or backdoors,
    - under a common command and control infrastructure.
- Uses it to indirectly launch attacks
  - e.g., spamming, phishing , DDoS, password cracking etc.
- very frequently sold or rented,
  - about 0.05 $ / host / week

Rootkits

- Software used after system compromise to:
  - Hide the attacker's presence
  - Provide backdoors for easy reentry
- Simple rootkits:
  - Modify user programs (ls, ps)
  - Modify a compiler
  - Detectable by tools like Tripwire (stores hashes of files).
- Sophisticated rootkits:
  - Modify the kernel itself
  - Hard to detect from userland

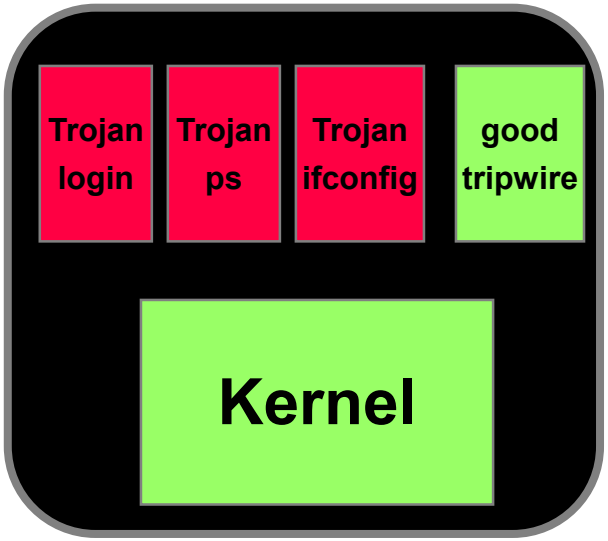
Rootkit Classification (1)

Application-level Rootkit



Hxdef, NTIllusion

Traditional RootKit

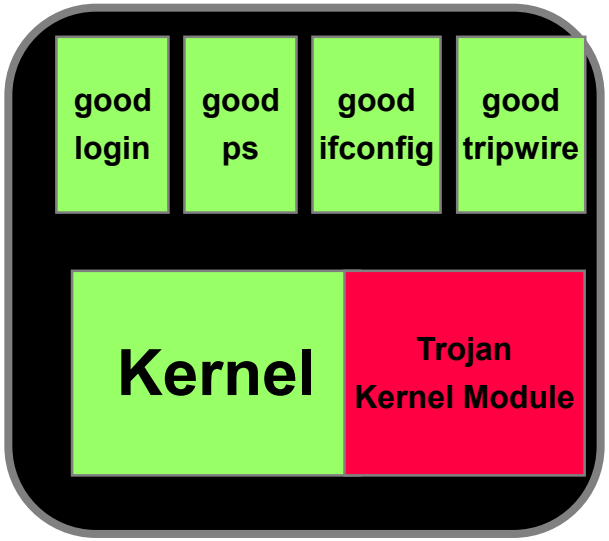


Lrk5, t0rn

Tripwire: detected! / maybe not detected

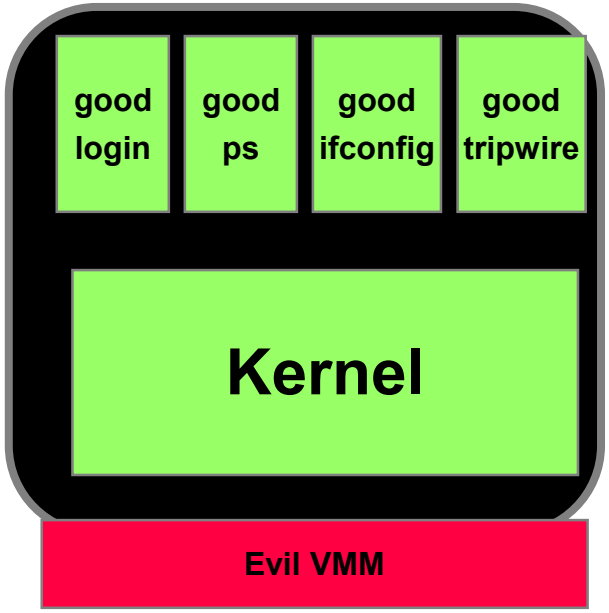
Rootkit Classification (2)

Kernel-level RootKit



Shadow Walker, adore

Under-Kernel RootKit

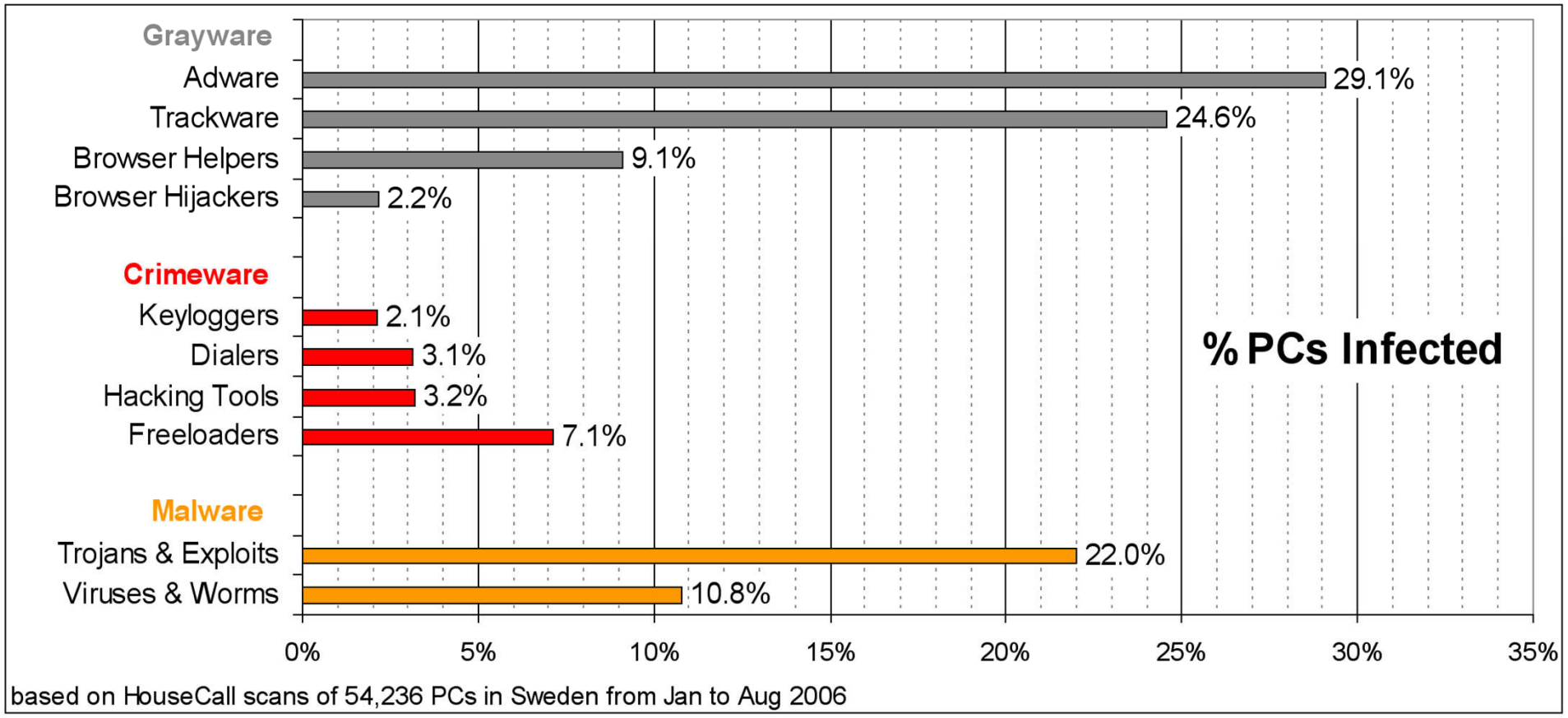


SubVirt, ``Blue Pill''

What's Going On?

Is My PC Infected?

- Swedish large scale study



Another Study

PCs infected
[source: Trend Micro]

1. Adware	43.9%
2. Trojan	18.8%
3. Browser helper	13.8%
4. Freeloader	9.8%
5. Trojan spyware	9.6%
6. Trackware	9.3%
7. Cracking app	5.0%
8. Worm	4.3%
9. Java script	3.9%
10. Dialler	3.1%
11. Keylogger	2.9%
12. Hacking tool	2.7%
13. Backdoor	2.1%
14. Portable executable	1.9%
15. Downloader	1.8%
16. Remote access app	1.2%
17. Exploit	1.1%
18. HTML script	0.7%
19. Joke program	0.6%
20. Browser hijacker	0.4%

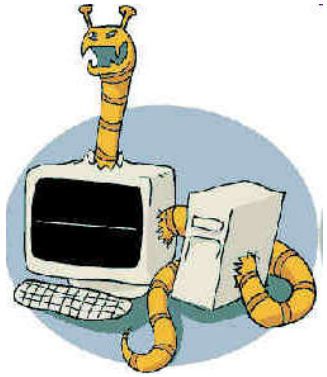
According to experts:

- Today's malware is designed to remain undetected for months.
  - do not get famous, get rich!
  - zero-day malware

Our Focus

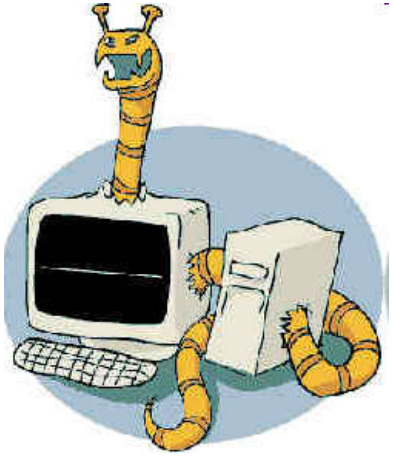
Worms are very dangerous. Why?

- Viruses and Trojans require some human actions, such as sharing a USB stick, clicking on web sites, opening ;doc email attachments etc.
  - spread is slow.
- Worms don't require human presence.
  - Spread is MUCH faster



Moreover worms do hack/break into computers, viruses and Trojans they use legitimate access channels and just abuse these privileges.

Famous Historical Worms - Unix



Morris Worm (first major network attack)

- Released November 1988
  - spreading on Digital and Sun workstations
  - exploited several Unix security vulnerabilities

- Consequences
  - no immediate damage
  - replication
    - load on network,
    - load on CPUs
  - many systems were shut down
    - fearing damage (only later people found it was not harmful)

* BTW.

Who was Morris?

- A criminal?
- A hacker?
- A university researcher?

* BTW.

Who was Morris?

- A criminal?
- A hacker?
- A university researcher?
  - and a well connected one…

*Morris - Author

Robert T. Morris, released it November 1988

- His father, another Robert Morris was
  - a cryptologist and code-breaker (broke codes for the FBI),
  - worked for the NSA “National Computer Security Center”,
  - wrote a book about UNIX Operating System Security (1984).

Morris Worm

- program to spread worm
  - looks for other machines that could be infected, several methods used: 'netstat -r -n‘, /etc/hosts,
  - when worm successfully connects, forks a child to continue the infection while the parent keeps trying new hosts
- vector program (99 lines of C)
  - re-compiled and run on the infected machines

Three ways the worm spread

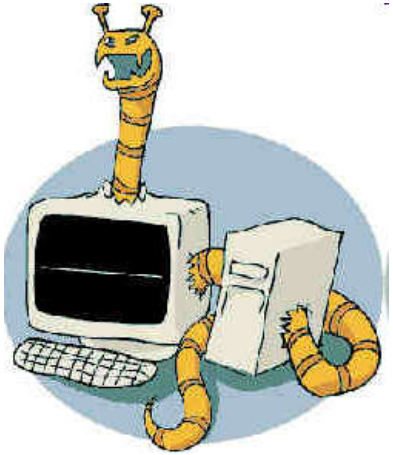
- Sendmail
  - exploited debug option in sendmail to allow shell access
- Fingerd
  - exploited a buffer overflow in the fgets function
- Remote shell
  - reading list of trusted hosts known to local OS
  - password cracking

Detecting Morris Internet Worm

- Files
  - Strange files appeared in infected systems
  - Strange log messages for certain programs
- System load
  - Infection generates a number of processes
  - Password cracking uses lots of resources
  - Systems were reinfected => number of processes grew and systems became overloaded
    - Apparently not intended by worm's creator

Thousands of systems were shut down

Famous Windows Worms



Increasing propagation speed

- Code Red, July 2001
  - fascinating story, see Brad Karp slides
    - Released AFTER Microsoft released the patch
    - affecting like 500 000 hosts in hours
- SQL Slammer, January 2003
  - See Brad Karp slides.
    - vulnerable population infected in less than 10 minutes
    - its growth was limited… by the speed of the Internet

Remark: both exploited an already known and already patched buffer overflow vulnerability!

Nimda worm

- Spreads via 5 methods to Windows PCs and servers
  - e-mails itself as an attachment (every 10 days)
    - runs once viewed in preview pane (due to bugs in IE)
  - scans for and infects vulnerable MS IIS servers
    - exploits various IIS directory traversal vulnerabilities
  - copies itself to shared disk drives on networked PCs
  - appends JavaScript code to Web pages
    - surfers pick up worm when they view the page.
  - scans for the back doors left behind by the "Code Red II" and "sadmind/IIS" worms

Nimda worm

- Nimda worm also
  - enables the sharing of the c: drive as C$
  - creates a "Guest" account on Windows NT and 2000 systems
  - adds this account to the "Administrator" group.

Malware Defences

Virus Defenses

Today's "anti-virus software":
Just a name.
Virus + firewall + etc…
Defends against all sorts of malware.

classical viruses are only about 5% nowadays…

Tips

- do not execute programs obtained by email
- maybe do not install any new software
  - ??? YES!
    - very few software companies can be trusted to care about their customers,
      - lack of liability, lack of legal obligations, culture of irresponsibility, need to renew products range etc etc…
      - do they even understand how secure is their own software?

Automated Virus Defenses

White list:

accept only trusted digitally signed programs.
Examples: Apple iPad, Nokia, Microsoft update, drivers, anti-virus install and updates, etc…

Prevalent methods in PCs:

- black-list, "signature"-based detection.* ≠ digital signatures
- network firewalls
- control application calls and IPC
- monitor and prevent "privileged" system calls, e.g. registry modification, plug-ins install etc.
- track changes to executables (hash + MAC/sign)

digital signatures!

*Code Signing = Digital Signatures

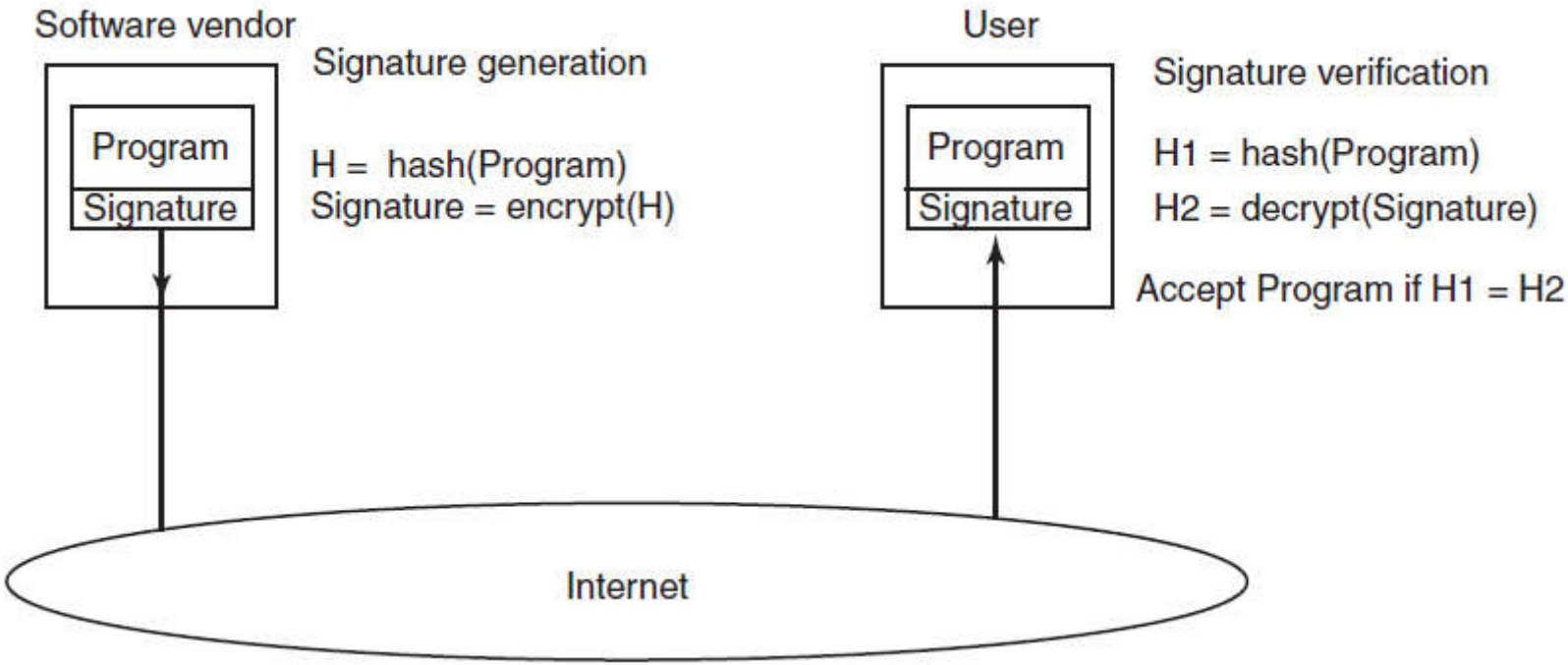


Figure 9-35. How code signing works.