POLITECNICO DI MILANO





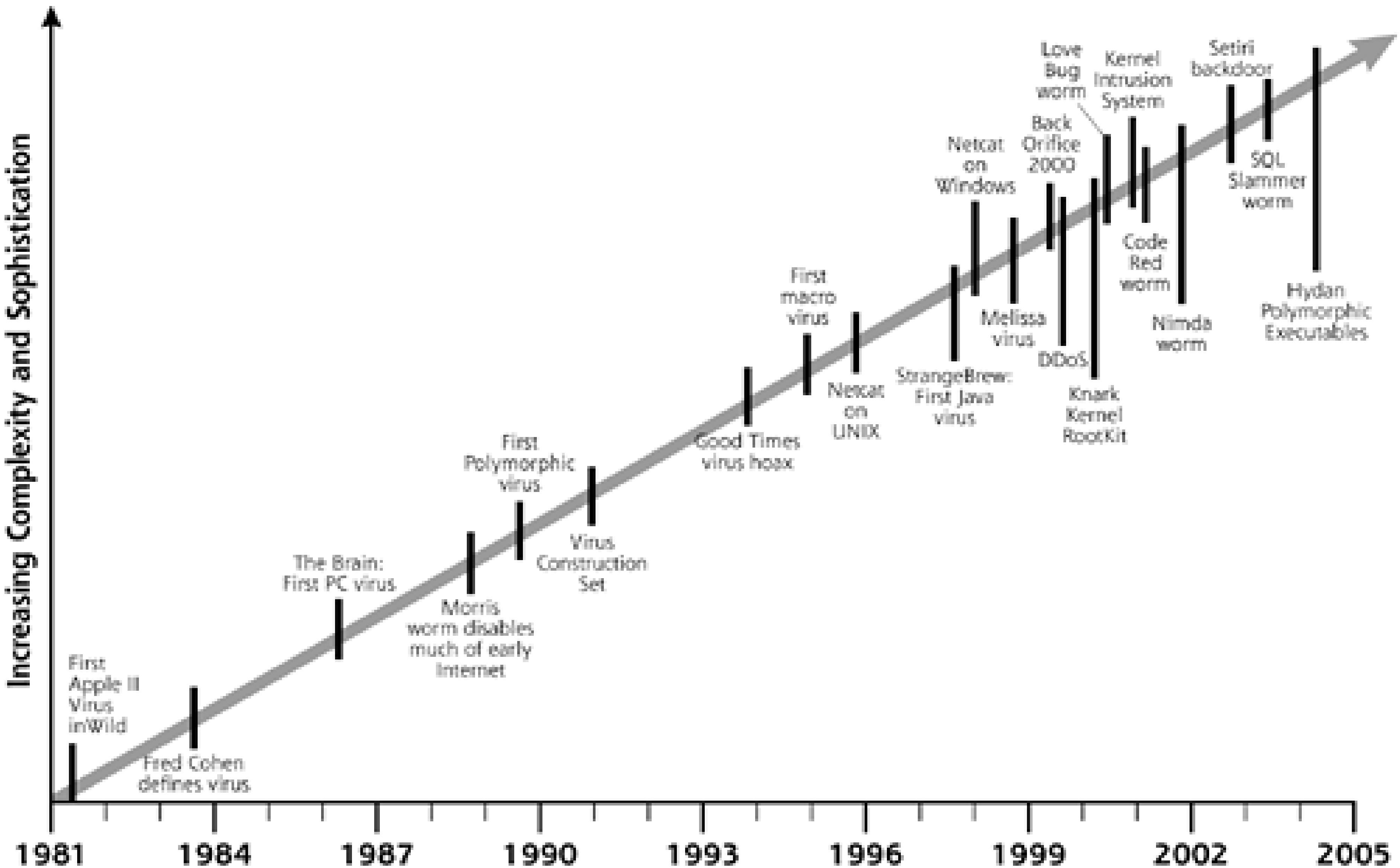
Introduction to malware

Stefano Zanero

What is malware?

- Malware (malicious software), also known as “malicious code”, is code that is intentionally written to violate a security policy.
- Several types of malware (no defined boundaries – no defined taxonomy)
  - Virus: piece of code that **self-propagates** (i.e. copies itself) by infecting other files, usually executables (but also documents with macro capabilities, boot loader code, etc.)
  - Worm: **self-propagating program** which copies itself, often by exploiting host vulnerabilities, or by social engineering (e.g. mail worms)
  - Trojan horses: programs with malicious capabilities, sometimes masqueraded as benign software. Often with **backdoor** capabilities.
  - Rootkits: combinations of trojans and techniques to hide them
  - Scareware/Rogue AVs: trojans that pretend to be anti-viruses or security tools
  - Adware: trojan which is used to display ads

A brief history of malware



A brief history of malware

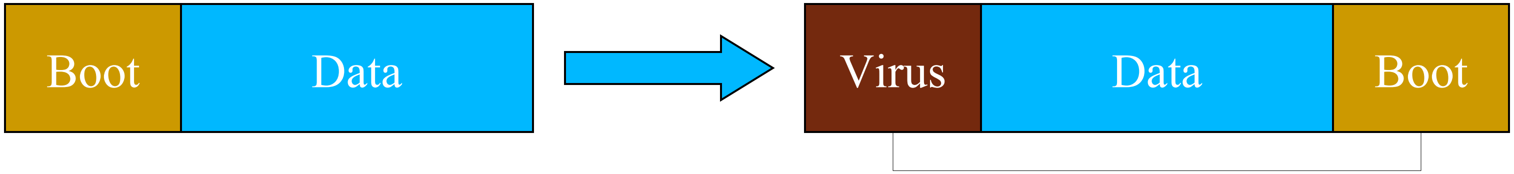
- 1981 First reported virus : Elk Cloner (Apple 2)
- 1983 Virus get defined
- 1986 First PC virus MS DOS
- 1988 First worm : Morris worm
- 1990 First polymorphic virus
- 1998 First Java virus
- 1998 Back orifice
- 1999 Melissa virus
- 1999 Zombie concept
- 1999 Knark rootkit
- 2000 love bug
- 2001 Code Red Worm
- 2001 Kernel Intrusion System
- 2001 Nimda worm
- 2003 SQL Slammer worm

Origins of viruses

- Fred Cohen, in 83, theorized the existence of viruses and produced the first examples
  - From a theoretical computer science point of view, interesting concept of self modifying and self propagating code
  - Soon, the security challenges were understood
- Viruses need a way to infect a program, a way to execute themselves when the program is run, a way to propagate
- Worms followed closely: in 89, Morris jr. propagated a worm on the early Internet, infecting 6000 machines and destroying links because of propagation traffic

Boot Sector Viruses

First sector of disk executed at boot



Worked well back when people traded floppies

- Could come back; “autorun.inf” on CDs

Executables

Attach itself to executable

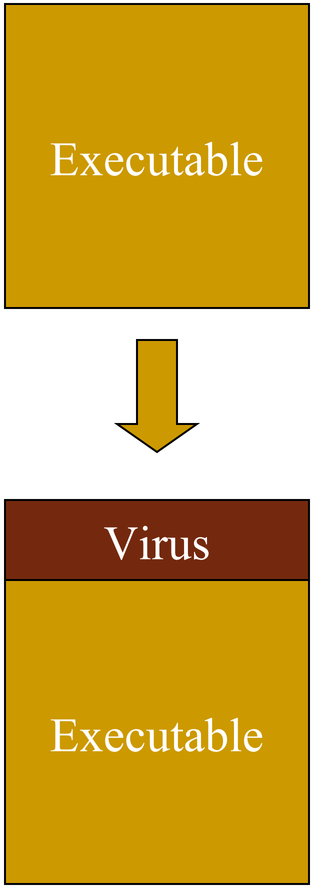
- Virus executes before normal executable is run

Can be multi-platform

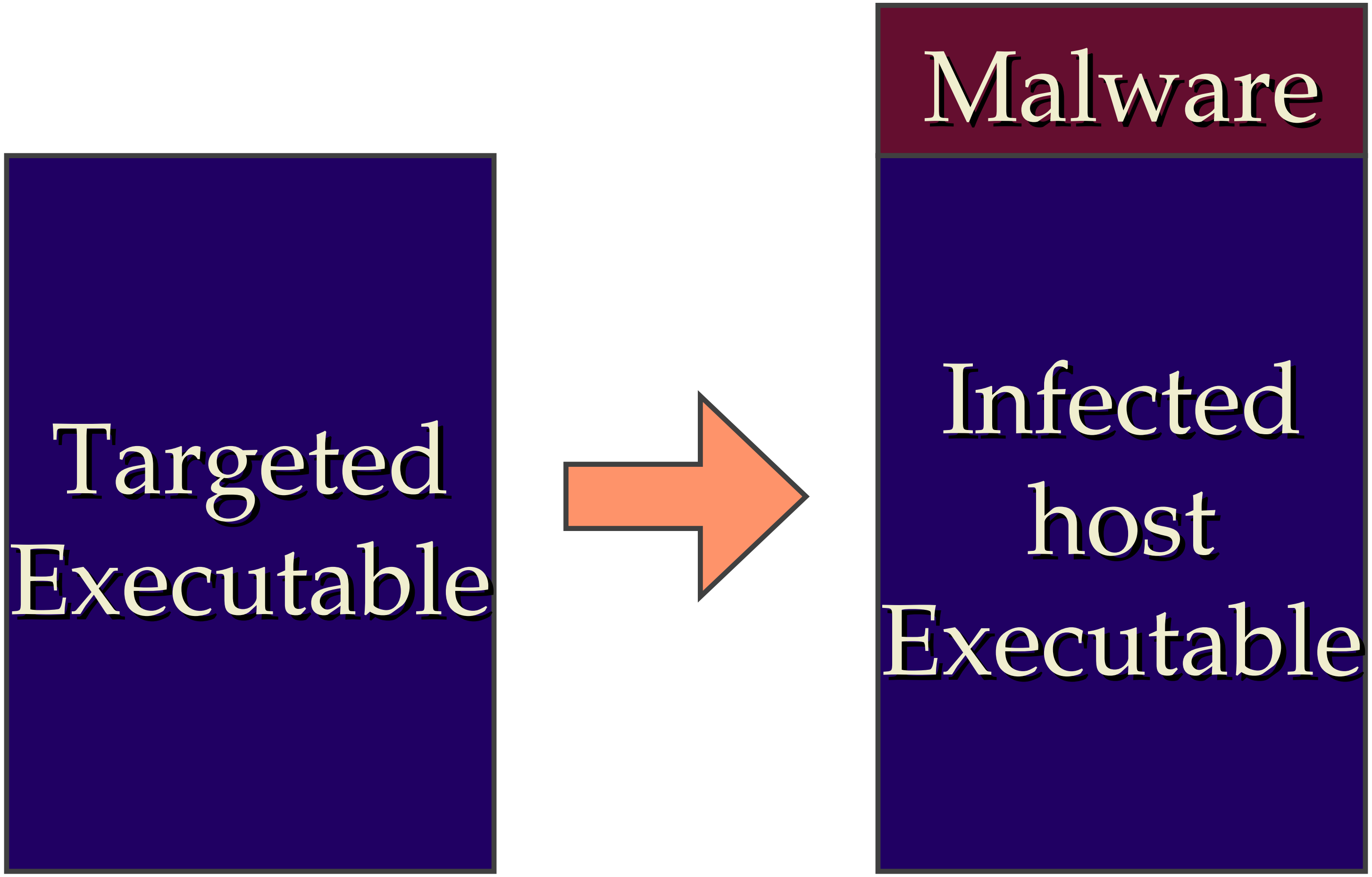
Popular method, esp. when BBS's or floppies were used to trade software

- Also has infected commercial software distributions

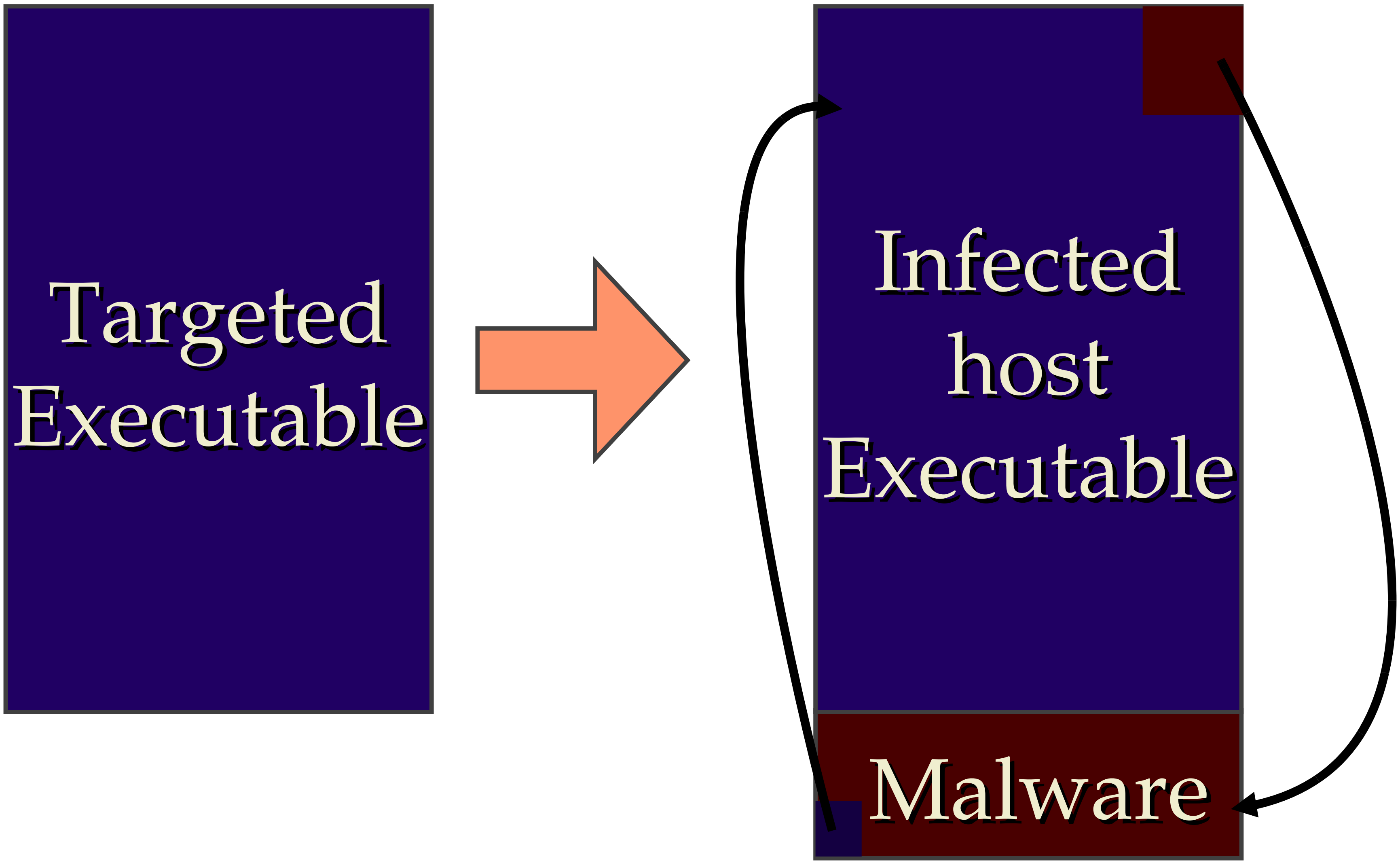
Still in use today



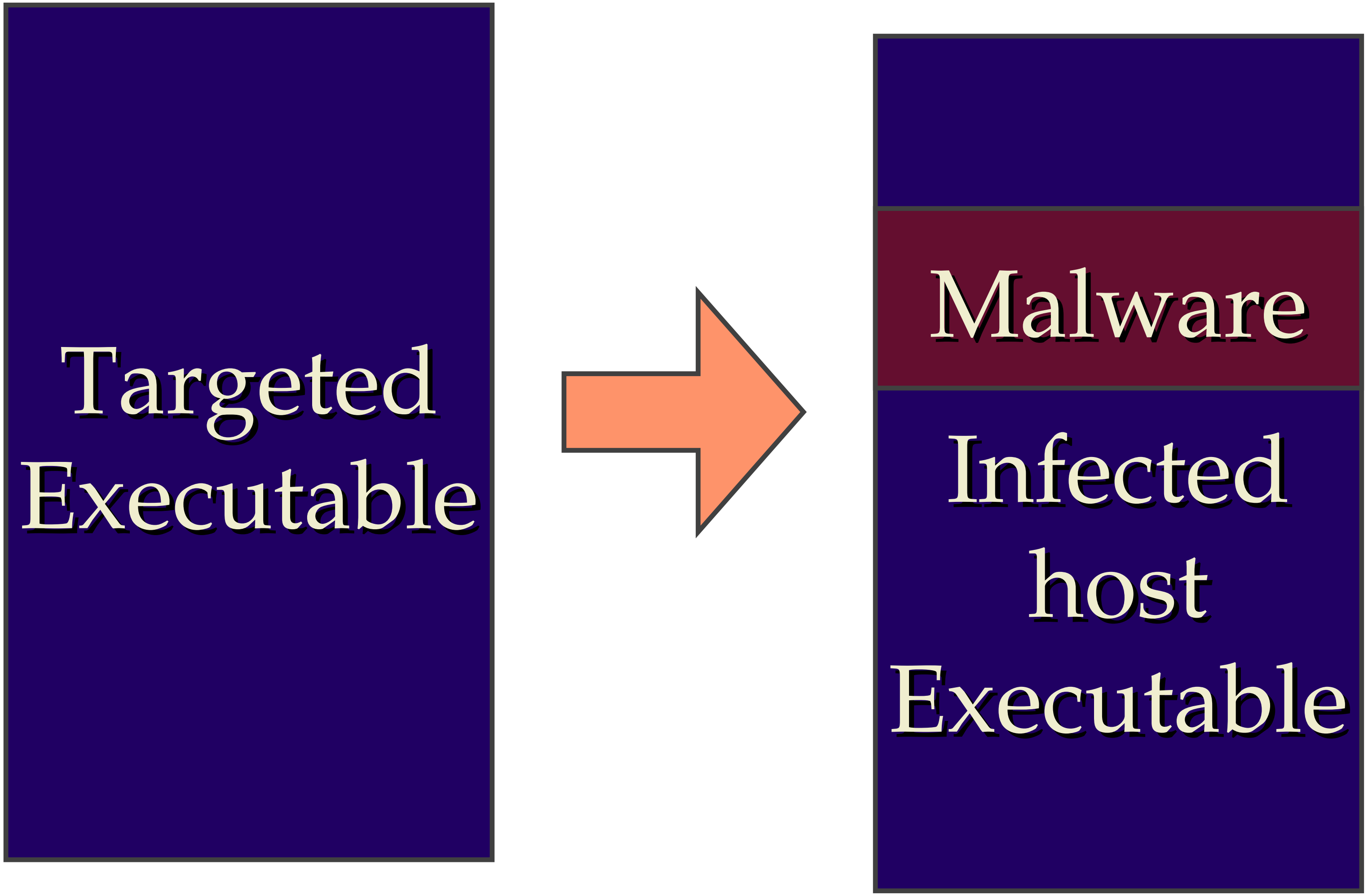
Prepending malware



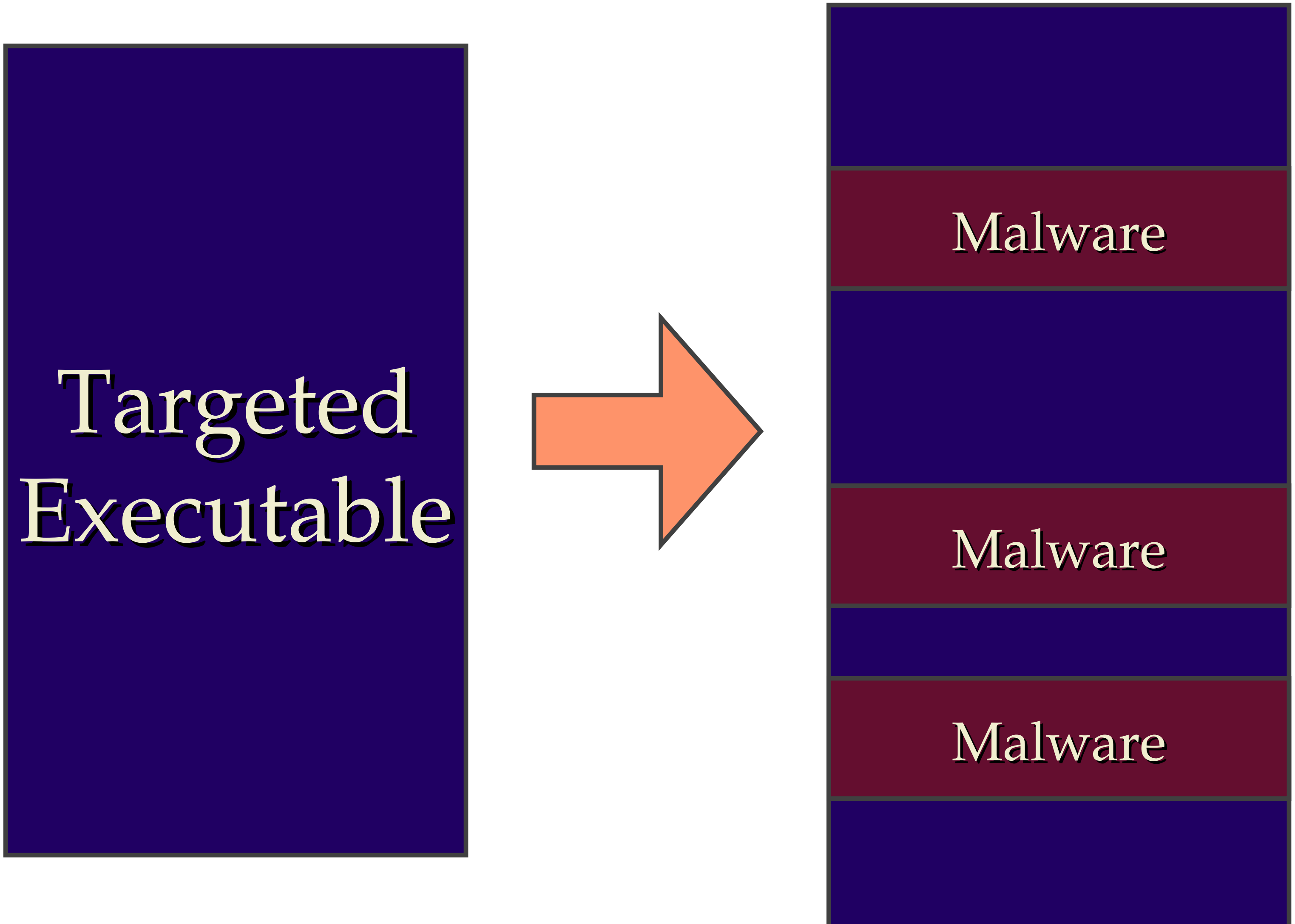
Appending malware



Cavity infector



Multicavity infector



Macros

- Data files traditionally safe from viruses
- Macro functionality blurs the line between data and code
- E.g. spreadsheet macro can:
  - Modify spreadsheet
  - Modify *other* spreadsheets
  - Send email
  - ...
- Example: the Melissa virus

Antiviruses

- Virus scanners are misuse detectors, they detect signatures of files (or memory-resident viruses)
- New viruses, or modified ones, usually escape detection
- It is not possible to build a perfect virus/malware detector (Cohen)
  - Diagonal argument
  - P is a perfect detection program
  - V is a virus
  - V can call P
    - if P(V) = true -> halt
    - if P(V) = false -> spread

Virus Stealth Techniques

Dormant period

Event-triggered payload

Encryption/Polymorphism

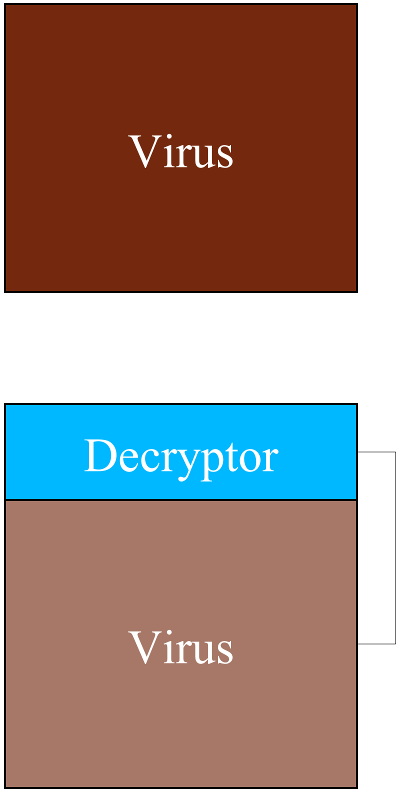
Metamorphism

Encryption

Encrypt virus content

Use small decryption routine with changing key to decrypt prior to execution

Anti-virus: find decryption routine?



Polymorphism

- Equivalent code
  - Insert NO-OP instructions, useless operations
    - x = x+1 ; x = x-1
  - Reorder registers, instructions, control flow
  - ...
- Polymorphic : uses a polymorphic engine to mutate while keeping the original algorithm intact
- Methamorpic : Change after each infection
- Detection problem: check whether code is equivalent to virus
  - How difficult is this?
  - Remember: Cohen's result

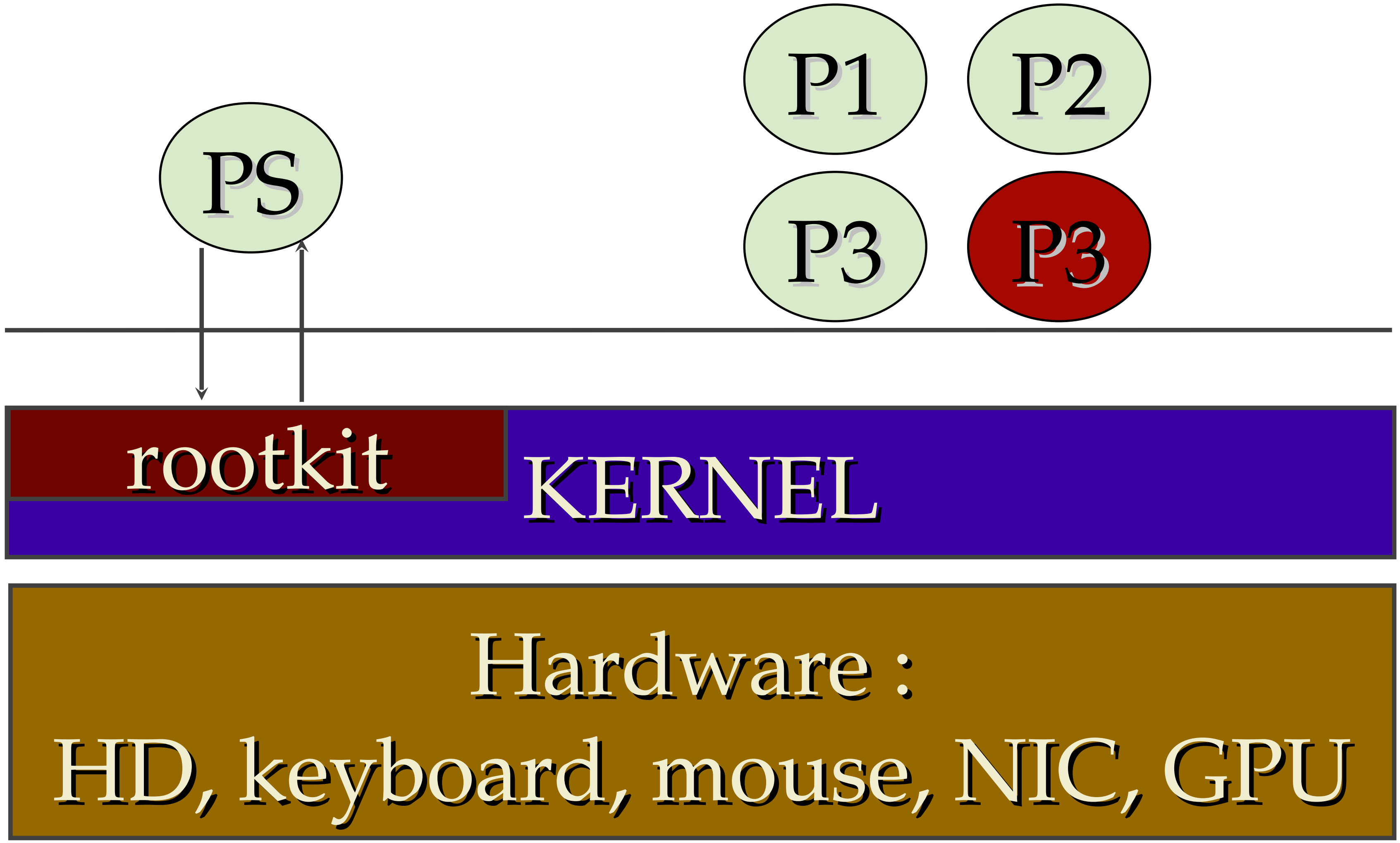
Packing

- Encrypt virus content
- Use small decryption routine with changing key to decrypt prior to execution
- Typical functions:
  - Compress
  - Encrypt
  - Randomize (polymorphism)
  - Anti-debug technique (int / fake jmp)
  - Add-junk
  - Anti-VM
  - Virtualization

Rootkits

- Make files, processes, user and directories disappear
- Make the attacker invisible
- Can be applied to most if not all OS
- Can be either userland or kernel-space
- Linux userland rootkit example:
  - Backdoored login, sshd, passwd
  - Trojanize to hide: ps, netstat, ls, find, du, who, w, finger, ifconfig...
- Windows userland rootkit targets:
  - Task Manager / Process Explorer...
  - Netstat / ipconfig...
- Obviously userland rootkits are complex and can be bypassed (non-trojanized util, recompiled util, util from safe source...)
- Kernel rootkits more versatile and difficult to reveal

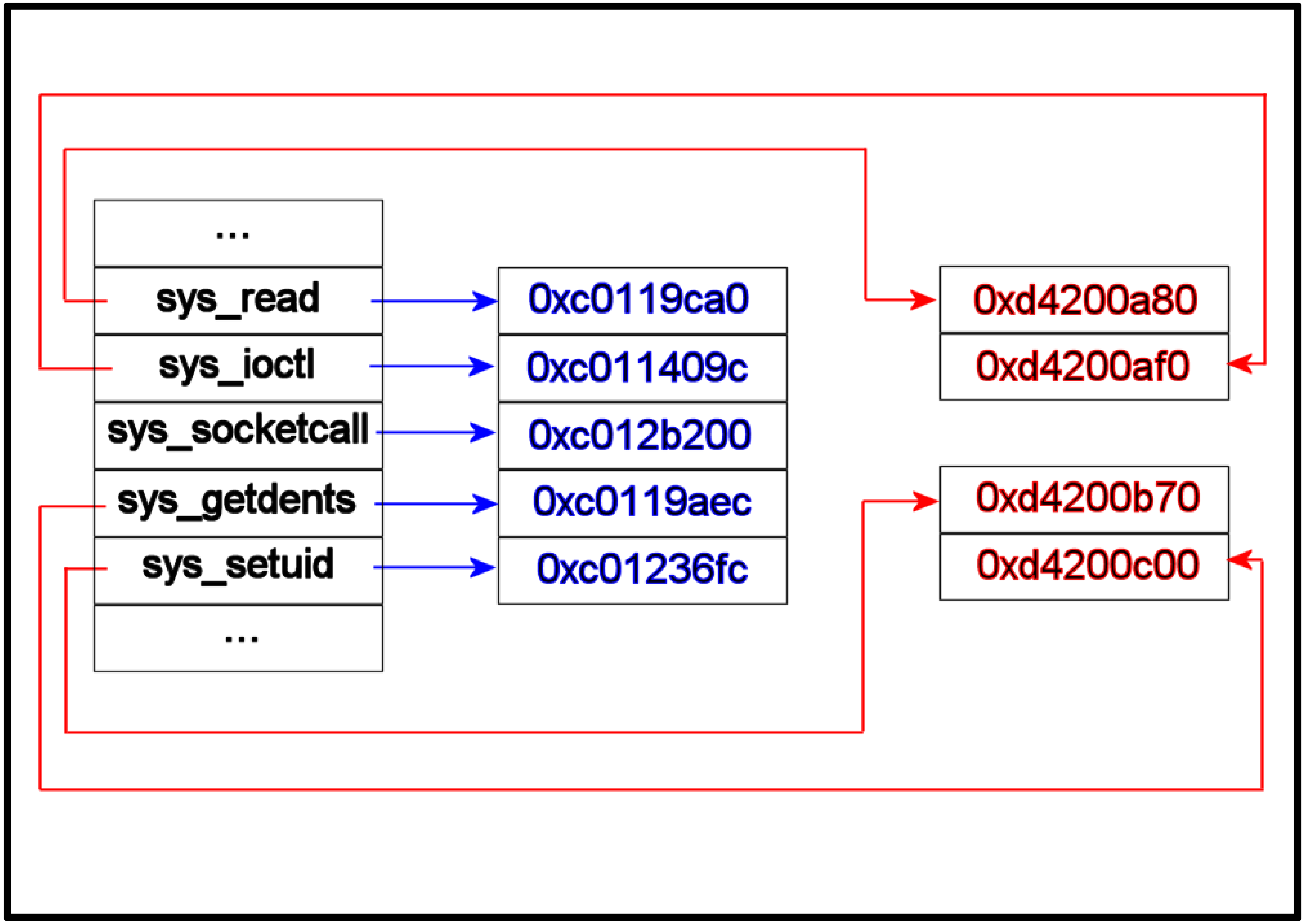
Kernel rootkit / Linux



History of Kernel rootkits

- 1997: Phrack 50, HalfLife “Abuse of the Linux Kernel for Fun and Profit”
- First implementation of syscall hijacking
- 1998, plaguez, “Weakening the Linux Kernel”, first complete LKM rootkit
- Others followed: Itf, heroin, carogna, knark, adore...

Syscall hijacking



A basic linux kernel rootkit

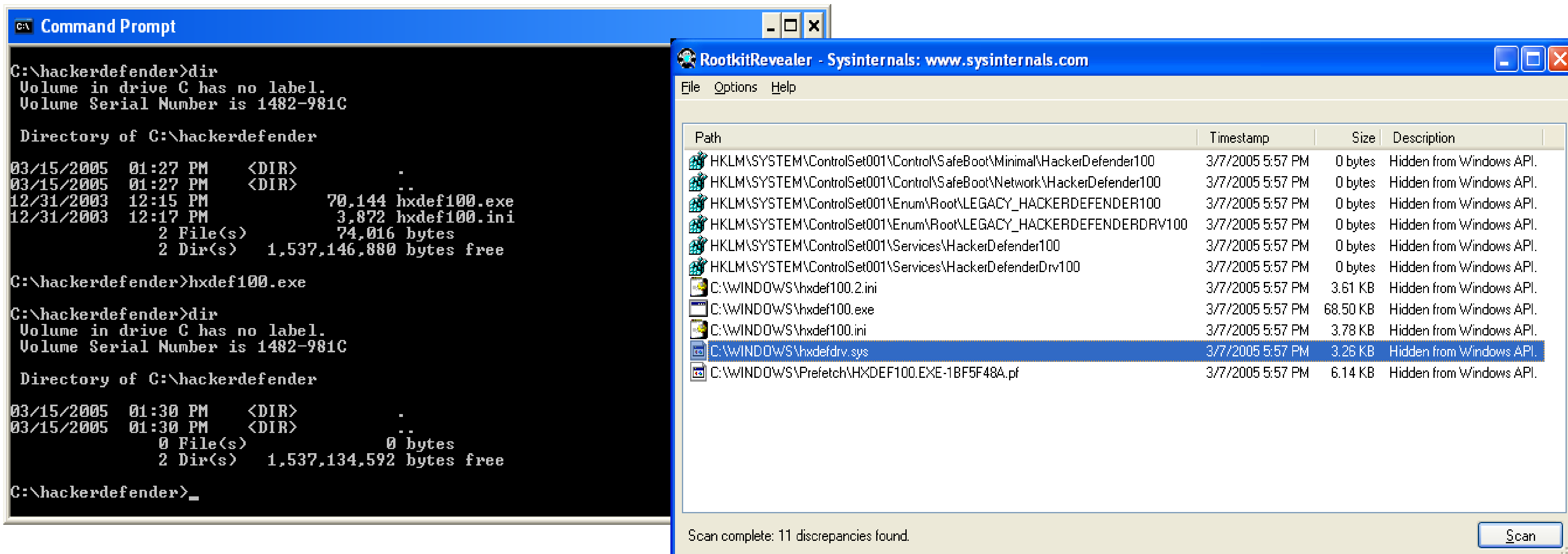
- Methods
  - Hook SYS_CALL Table, Interrupt Descriptor Table, o Global Descriptor Table
    - 2.4 SYS_CALL table is exported
    - 2.6 Kernel – SYS_CALL table is hidden, then SuckIT e.g. scans the IDT looking for a FAR JMP *0xSCT[eax]
  - Detour Patching
  - Directly patch /dev/mem or /dev/kmem
  - How to detect?
- Alternative methods through /dev/kmem
  - Direct patching of syscall code (even more difficult to detect)
  - Can patch kernel even if no LKM support is present (only defense if attacker is root: POSIX capabilities)

Yet other rootkits

- Rootkit in BIOS
  - In ACPI, John Heasman, NGS
  - CMOS, eEye bootloader
- Rootkit on the firmware of NIC or Video Card
- Rootkits in virtualization systems (how do you recognize a rootkit which acts as an hypervisor?)

How to recognize a rootkit

- Intuition (noting abnormalities)
- Cross-layer examination (what I see through an API is different than what I see through another/through the kernel)
- Signature based systems with the usual problems (only for userland kits)
- Explicit Compromise Detection or Trusted Computing Base



Sony Player DRM and Rootkits

- Bad press for Sony in 2005
- To ensure that copy protection is not evaded install rootkit to hide the protection code
- Why is this a **very** bad idea?
  - Available for other attackers to use
  - Non-uninstallable
  - Uses CPU and memory
  - Not adequately noted in EULA

The Morris Worm Incident

How 99 lines of code brought down the Internet (ARPANET actually) in November 1988.

Robert Morris Jr. (at the time a Ph.D student at Cornell), wrote a program that could:

- Connect to another computer, and find and use one of several vulnerabilities (buffer overflow in fingerd, password cracking etc.) to copy itself to that second computer.
- Begin to run the copy of itself at the new location.
- Both the original code and the copy would then repeat these actions in an infinite loop to other computers on the ARPANET (mistake!)

Rebirth of the worms: mass-mailers

- Email software started allowing attached files, including:
  - Executables (dancing bears)
  - Executables masquerading as data
  - E.g. “LOVE-LETTER-FOR-YOU.txt.vbs”
- Spread by emailing itself to others
  - Use address book to look more trustworthy
- Now moving onto mobile phones
  - MMS viruses
  - Bluetooth viruses (maybe)

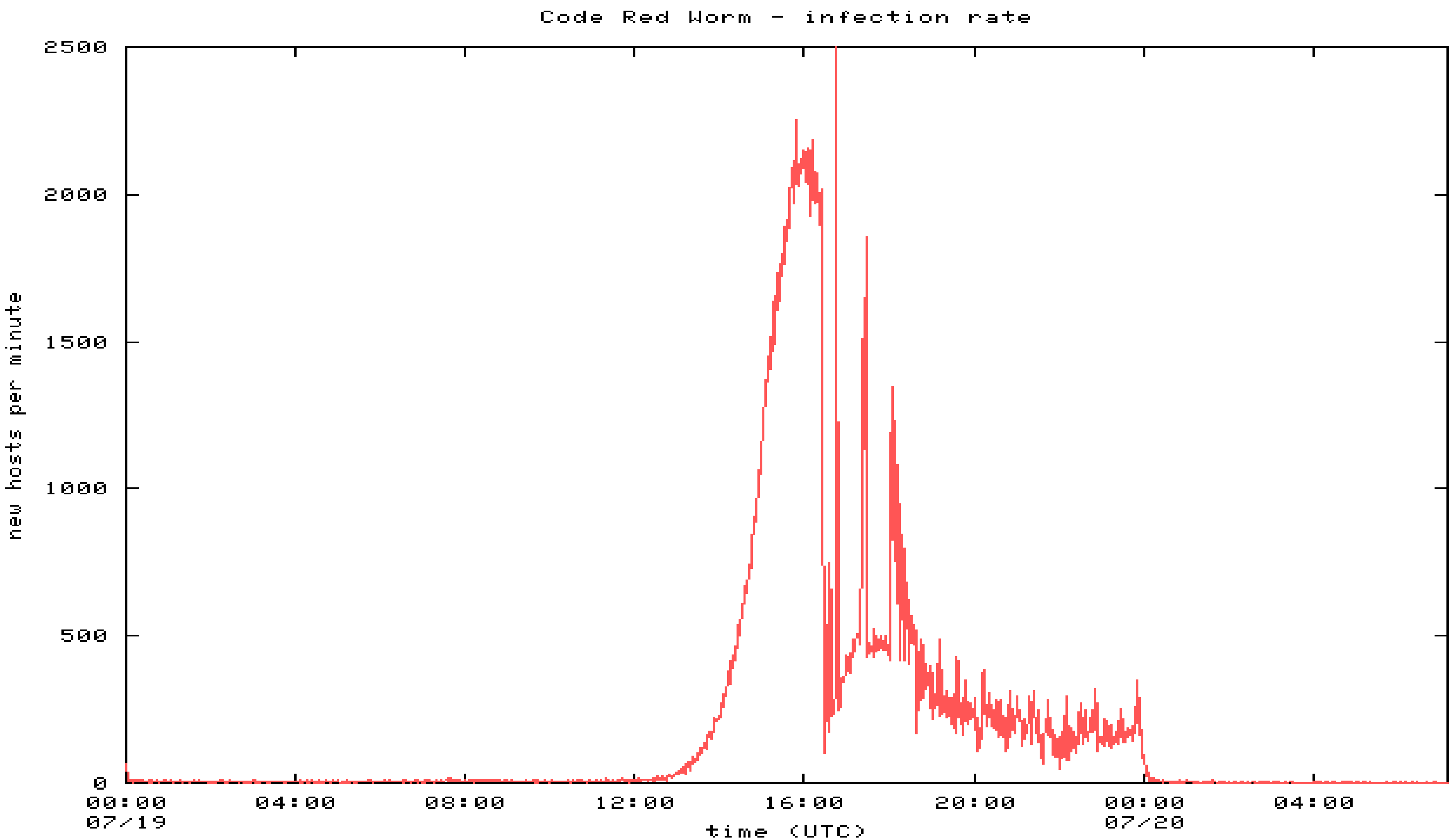
Modern Worms: mass scanners

- Basic pattern the same:
  - Infect computer
  - Seek out new targets
  - Perform something malicious (e.g. deface websites, erase images, install backdoors)
- Faster spread (minutes), larger scale (hundreds of thousands of hosts)
- Scanning
  - Select random addresses - good chance of hitting an existing host
  - Local preference: direct scans towards local network
  - Permutation scanning (divide up IP address space)
  - Hitlist Scanning
  - DNS searches, Spiders, P2P networks, public lists
- Warhol worm - Hit list + permutation
  - How to 0wn the Internet in your spare time http://www.icir.org/vern/papers/cdc-usenix-sec02/

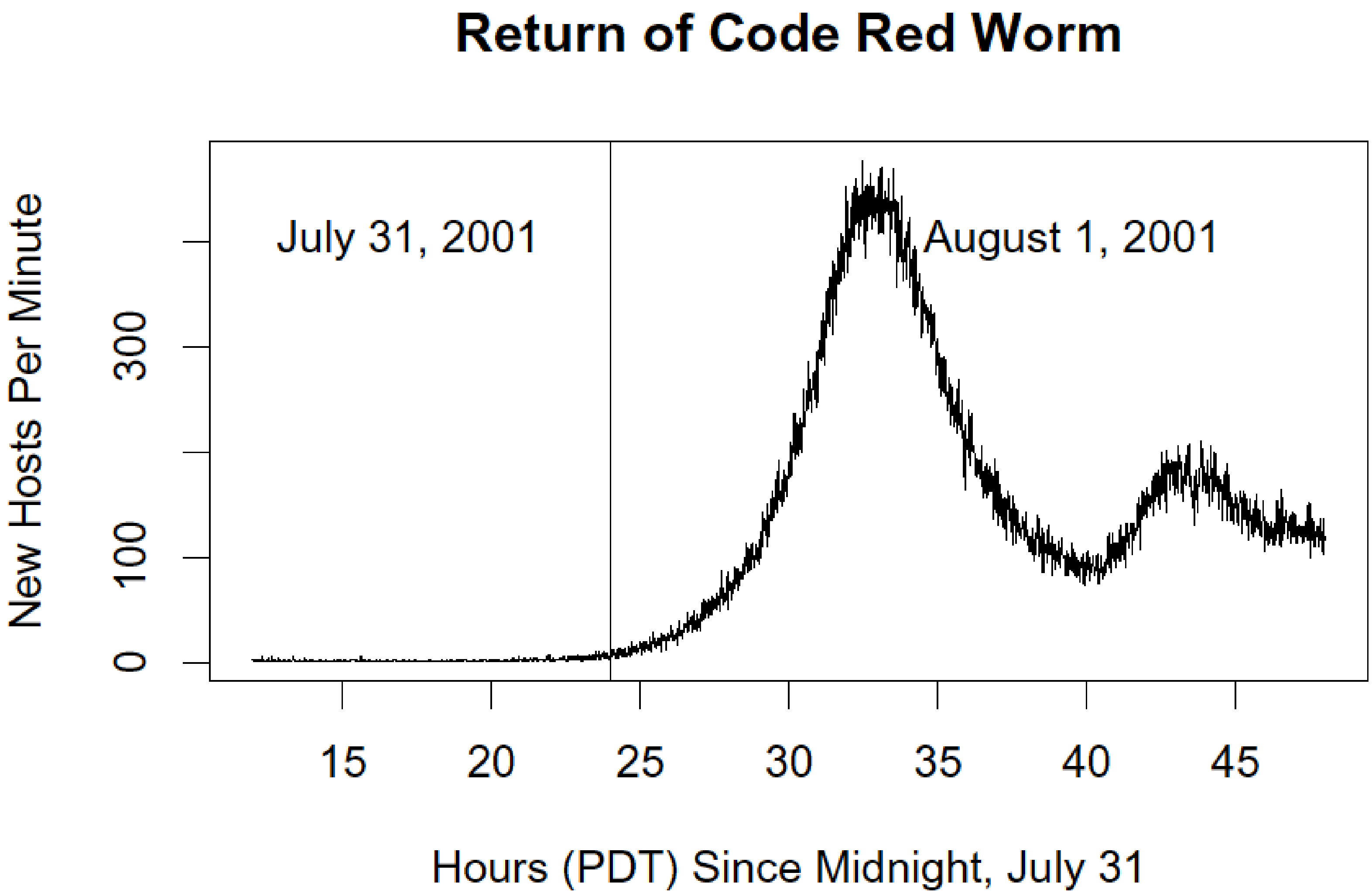
Worm Examples

Worm	Date	Distinction
Morris	11/88	Used multiple vulnerabilities, first one :)
ADM	5/98	Random scanning of IP address space
Lion	3/01	Stealthy, rootkit worm
Cheese	6/01	Vigilante worm that secured vulnerable systems
Code Red	7/01	Completely memory resident
Walk	8/01	Recompiled source code locally
Nimda	9/01	Windows worm: client-to-server, c-to-c, s-to-s
Scalper	6/02	11 days after announcement of vulnerability; peer-to-peer network of compromised systems
Slammer	1/03	Used a single UDP packet for explosive growth

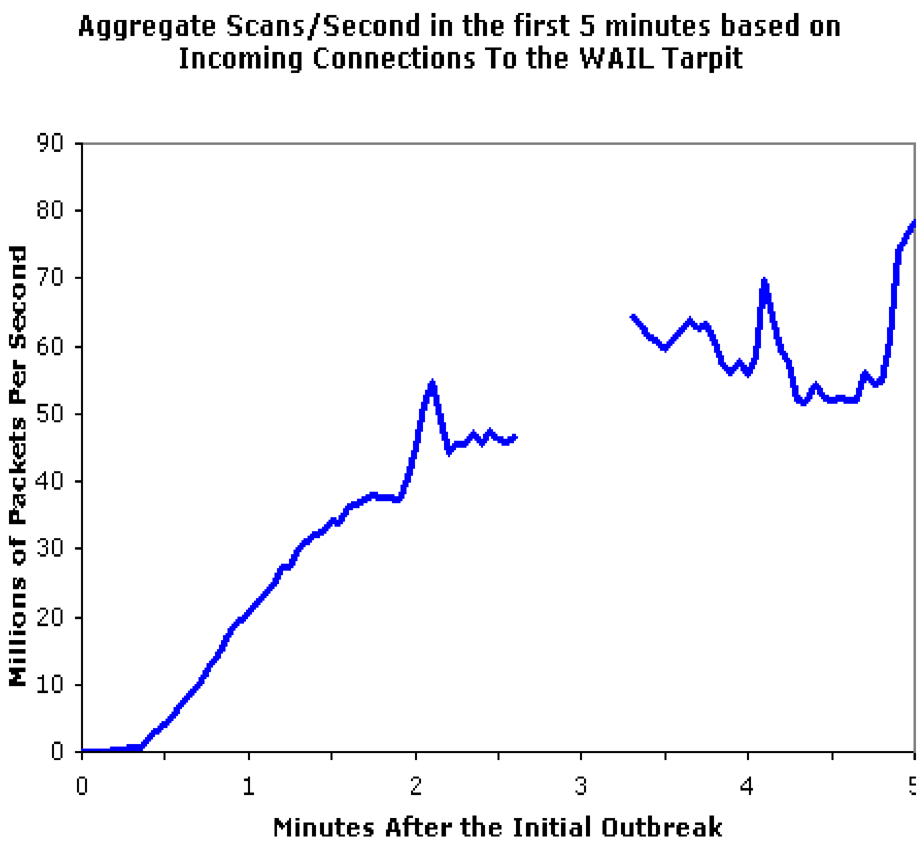
Code red propagation

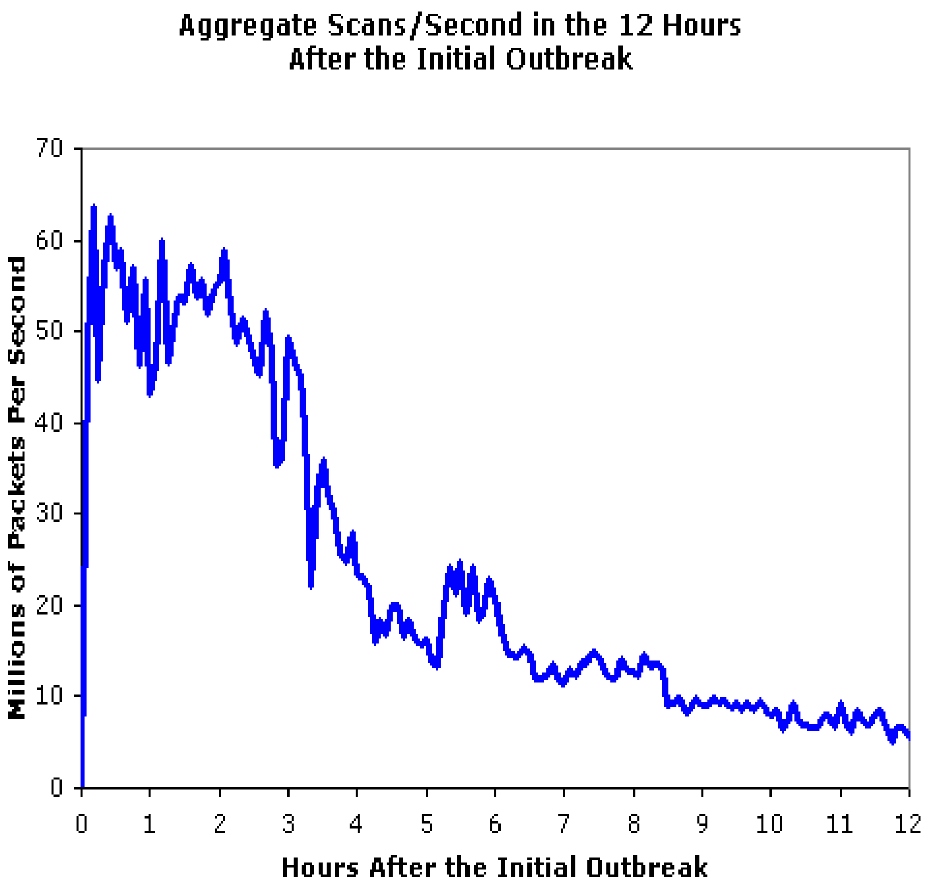


Code red: reactivation



Slammer propagation in UDP





Worms in IPv6

Address space is 2^128 instead of 2^32

- Random address selection will not work

Say ¼ of address in IP4 network run Windows

- 1 in 4 chance of finding a target with each probe

Spread that among 2^128 addresses

- 1 in 2^98 chances of finding a viable target

Worm Defense

Patches

- Most worms exploit *known* vulnerabilities
- Useless against *zero-day* worms

Signatures

- Must be developed automatically
- Worms operate too quickly for human response

Intrusion detection (anomaly)

- Notice fast spreading, suspicious activity, ...
- Can be a driver to automated signature generation
- Still under research...

And by the way... where are the worms ?!

- We *all* thought that the Internet would get wormier
- The trend was clear:
  - 2001: Li0n, Code Red, Nimda
  - 2002: Slapper, Klez
  - 2003: SQL Slammer, Blaster, SoBig
  - 2004: Sober, MyDoom, Witty, Sasser
  - I have even an iDefense t-shirt with this list on it!
- Since then, silence on the wires. No new “major” worm outbreaks
  - Weaponizable vulns were there, we even collectively braced for impact a couple of times
  - Did we get *so better* at defending networks? I bet “not”

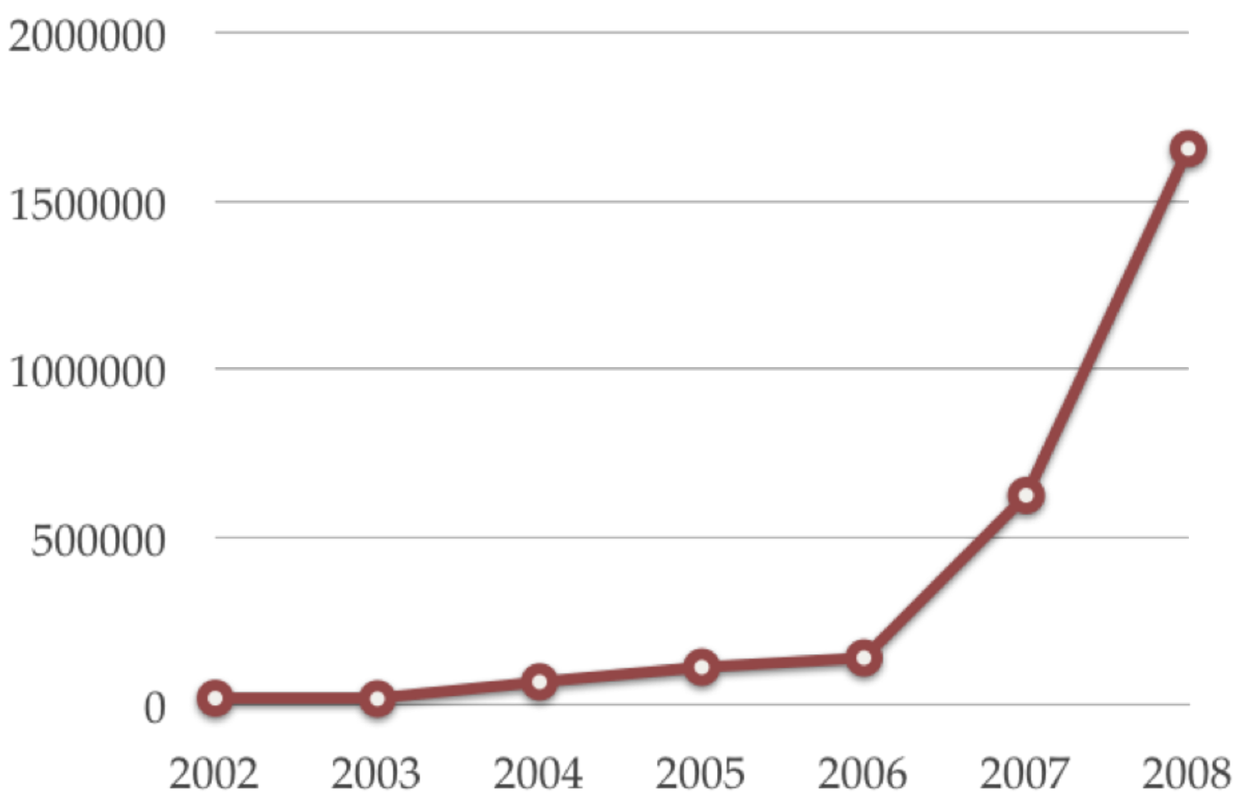
Open wormy questions: example

- Why no worm has ever targeted the infrastructure?
  - (possible exception of Witty, targeting firewalls)
- Possible explanation: routers and the like are a difficult vector to exploit
  - Not really true anymore, see FX's and Michael Lynn's works
  - Can use a traditional worm for propagation + a specialized payload for infrastructure damage
  - Windows of opportunity were there:
    - June 2003: MS03-026, RPC-DCOM Vulnerability (Blaster) + Cisco IOS Interface Blocked by IPv4 Packets
    - April 2004: MS04-011, LSASS Vulnerability (Sasser) + TCP Vulnerabilities in Multiple IOS-Based Cisco Products (resets)
- So why, oh why, the /bin/ladens of the world were not there, grinning and reaping?

Rise of the Bots

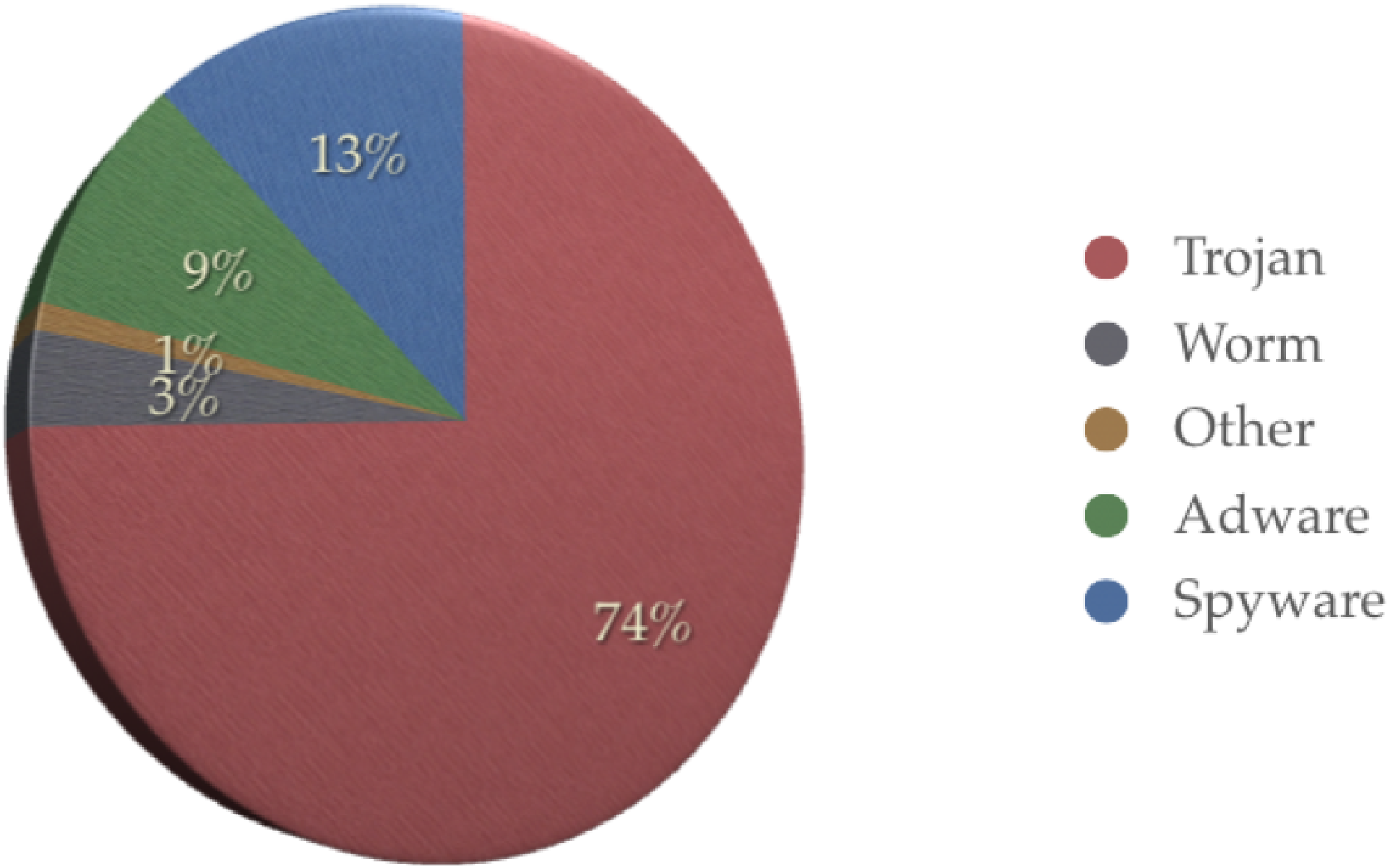
- Bots, bots everywhere
  - When I was a youngster, bots were IRC warriors' stuff (~1999-2000)
  - We used to call remote control trojans “zombies”, and they were usually DDoS tools (2000-2)
- Today's bots are different
  - Intelligent, evolving, with complex C&C infrastructures, difficult to remove as well
  - Larger botnets (10k common, 1M+ seen)
  - Phishing, spamming and pharming bots... more difficult to track than DDoS events
- How do we track them? How do we analyze them?
  - Worm explosive propagation vs. bot slow and steady diffusion: there's no network telescope that can see them

Number of different malware signatures...



Symantec report 2009

Distribution of types of malware



Panda Report 2009