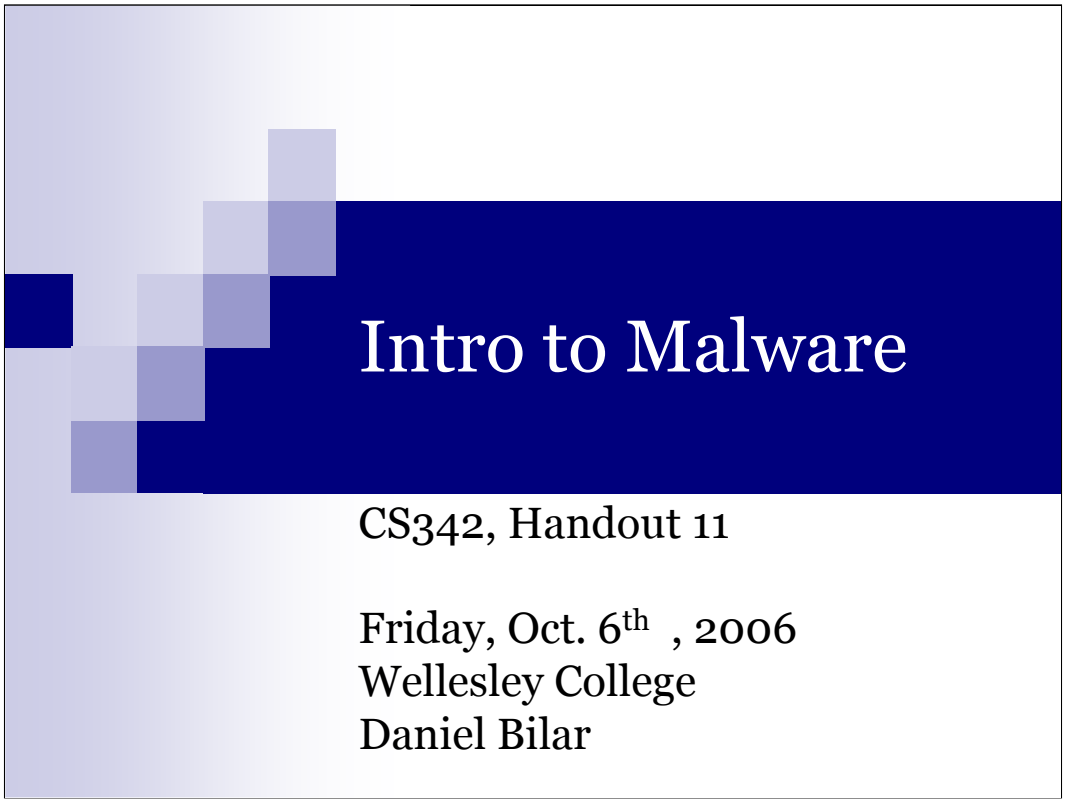

Intro to Malware
CS342, Handout 11
Friday, Oct. 6th , 2006
Wellesley College
Daniel Bilar

# Today's class goals

- Introduction to malware taxonomy
- Appreciating the evolution of viruses/worms
  - Polymorphism/Metamorphism
- General AV detection techniques
- Appreciate again that: If you take your time, have few patterns and use many levels of indirection, you are very hard to beat

# Malware (MALicious softWARE)

- A generic term increasingly being used to describe any form of malicious software like
  - viruses, worms, trojan horses, rootkits, malicious mobile code, backdoors, dialers, ransomware
- Combinations are common
  - Some malware are actually combinations of more basic malware (A rootkit can be viewed as Trojan horse backdoor tool)
- Broadly classified according to
  - Spread technique: “how does it propagate?”
  - Impact on target: “what does it do?”
- Other useful classification is
  - Type of exploited vulnerability: “ How (by violating which assumption) did it get in?”

# Standard Taxonomy of Malware

Based on **mechanisms used to spread**, and as the **impact (payload)** it has on the target. Sometimes all are referred to as 'virus'.

| Type of Malware | Characteristics |
|---|---|
| **Virus** | Infects a host file (e.g., executable, word processing document, etc.) Usually requires human interaction to replicate |
| **Worm** | Spreads across a network. Usually does not require human interaction to spread |
| **Malicious mobile code** | Small programs that are downloaded and executed locally with minimal user intervention. Typically written in Javascript, VBScript, Java, or ActiveX |
| **Backdoor** | Bypasses normal security controls to give an attacker access |
| **Trojan horse** | Disguises itself as a useful program while masking hidden malicious purpose |
| **User-level RootKit** | Replaces or modifies executable programs used by system administrators and users |
| **Kernel-level RootKit** | Manipulates the OS kernel to hide and create backdoors |

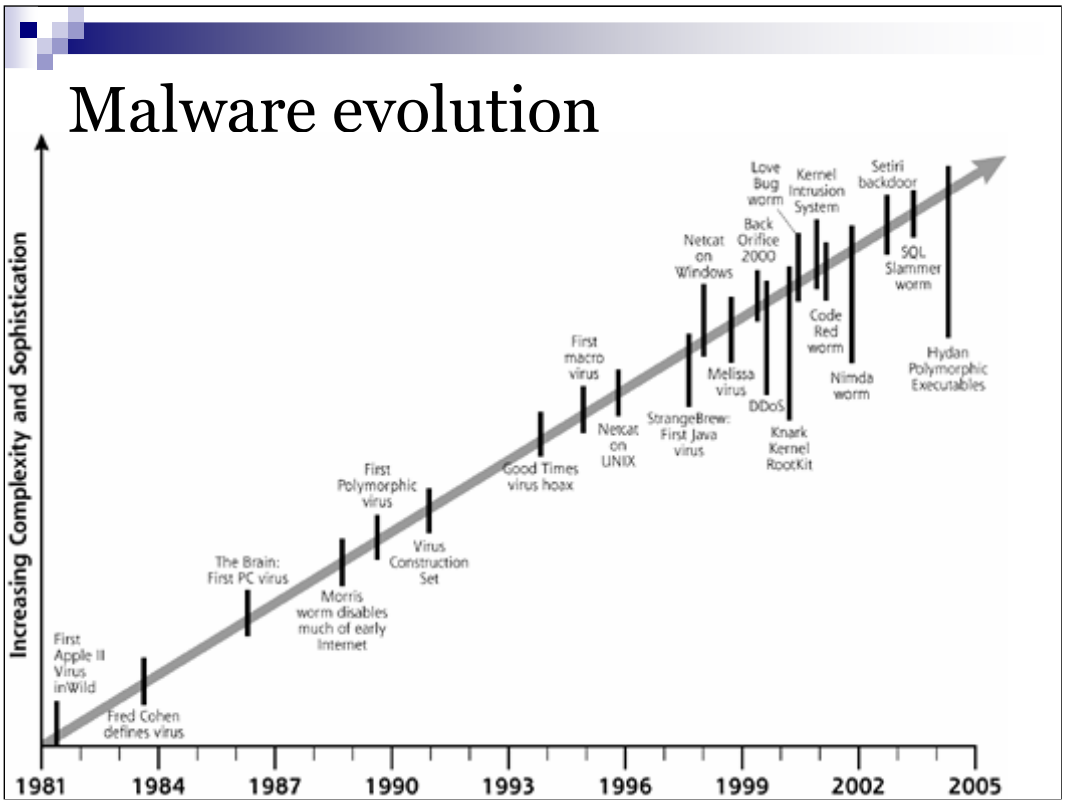
Rootkits are Trojan horse backdoor tools that modify existing operating system software so that an attacker can keep access to and hide on a machine.

# Sample Payloads

- Magistr - trashes the primary hard drive controller, overwrites CMOS RAM, erases flash memory (BIOS), attaches random file when spreads
- VBS_HOMEPAGE.A - randomly opens certain salacious Web sites using Internet Explorer
- PrettyPark – sends victim's name, address book, password files to IRC chat channels
- DoS.Storm – infects vulnerable servers running MS software and then launches DDoS attack against MS website (Code Red similar, but attacked whitehouse.gov)
- Timofonica – sends SMS (Short Messaging Service) message to random cell phone customer of Movistar SMS gate
- 911 worm – dials 911 and erases hard drive

# Example Spread

- Autonomously via network servers
  - Web servers, application servers, network services, open file shares, trusted hosts ..
- Semi-autonomously via network clients
  - Web browser, email clients, ..
- Manually
  - USB key, floppy disk, ..




The increasing complexity and sophistication of malicious software: We went from fairly simple Apple II viruses that infected games to the complex kernel manipulation tools and powerful worms of this new millennium. The newer tools are very crafty in their rapid infection and extreme stealth techniques.

Acceleration of the rate of release of innovative tools and techniques: New concepts in malicious code started slowly, but have certainly picked up steam over time. Especially over the past five years, we've seen the rapid release of amazing new tools, and this trend is only increasing. Just when I think I've seen it all, the computer underground releases an astonishing (and sometimes frightening) new tool.

Movement from viruses to worms to kernel-level exploitation: In the olden days of malicious code, most of the action revolved around viruses and infecting executable programs. Over the past five years, however, we've seen a major focus on worms, as well as exploiting systems at the kernel level.

# Trends and pertinence

Table 1.1: Virus spread profile 1990-2003 [Kes00][Bek03]

| Virus | Year | Type | Time to prevalence [1] | Estimated damages |
|---|---|---|---|---|
| Jerusalem, Cascade, Form, | 1990 | .exe File, Boot Sector | 3 years | $50M |
| Concept | 1995 | Word Macro | 4 months | $50M |
| Melissa | 1999 | E-mail enabled, Word Macro | 4 days | $93M-$385M |
| Love Bug | 2000 | E-mail enabled, .vbs | 5 hours | $700M - $6.7B |
| Slammer | 2003 | self-propagating worm | 10 minutes | $1B |

- Virus/Worms: In 12 years, propagation speed, as well as the estimated damages have increased by five, and two orders of magnitude, respectively
- Damages will get worse and outages more severe as more and more facets of our lives/society are digitally enmeshed

Forecast for email virus infection rates

1 in 100 in 2004

1 in 10 in 2008

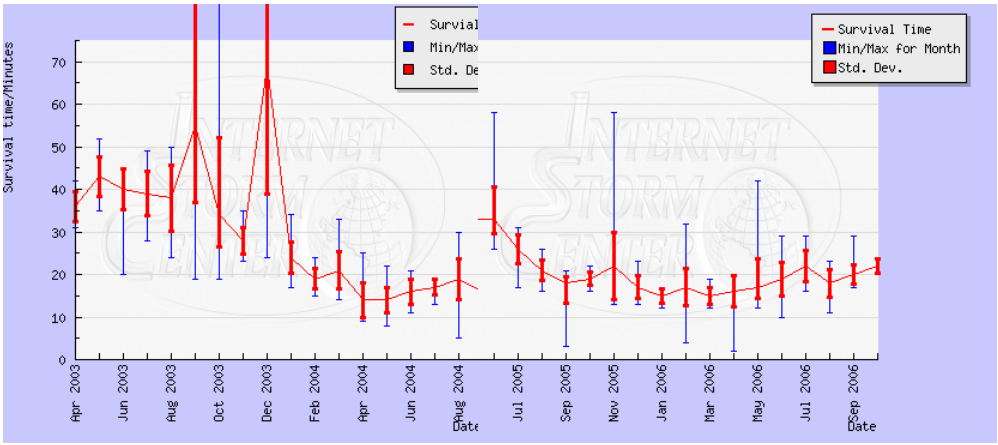
1 in 2 in 2013

3 of 4 in 2015

Source: MessageLabs (www.messagelabs.com)

scans e-mail for >500,000 users

# Survival times unpatched machines




The survival time is calculated as the average time between reports for an average target IP address. If you are assuming that most of these reports are generated by worms that attempt to propagate, an unpatched system would be infected by such a probe.
http://isc.sans.org/survivalhistory.php

# Viruses

- Virus propagates by infecting other programs
  - Automatically creates copies of itself, but to propagate, a human has to run an infected program
    - Self-propagating malicious programs are usually called worms
- Viruses employ many propagation methods
  - Insert a copy into every executable (.COM, .EXE)
  - Insert a copy into boot sectors of disks
    - "Stoned" virus infected PCs booted from infected floppies, stayed in memory and infected every floppy inserted into PC
  - Infect TSR (terminate-and-stay-resident) routines
    - By infecting a common OS routine, a virus can always stay in memory and infect all disks, executables, etc.

# Virus Types (Overlaps exist)

- Stealth viruses
  - Infect OS so that infected files appear normal to user by intercepting the read request to the file and returning the content of the original read request to the uninfected file.
- Macro viruses
  - A macro is an executable program embedded in a word processing document (MS Word) or spreadsheet (Excel)
  - When infected document is opened, virus copies itself into global macro file and makes itself auto-executing (e.g., gets invoked whenever any document is opened)

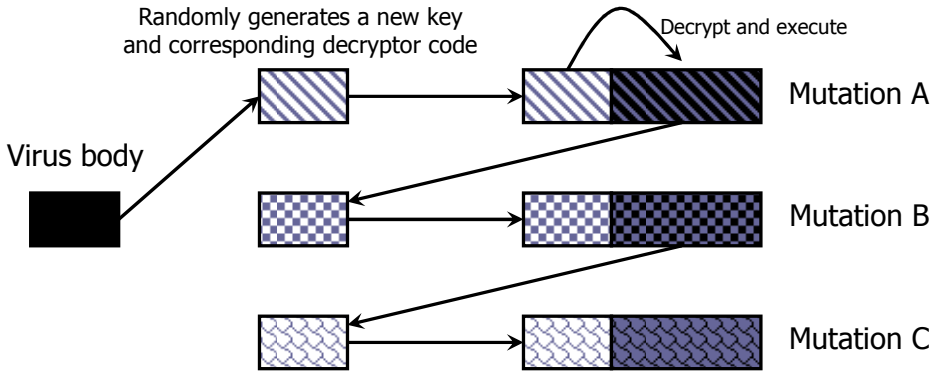
# Anti-Virus Technologies

- Simple anti-virus scanners
  - Look for signatures (fragments of known viruses)
  - Heuristics for recognizing code associated with viruses
    - For example, polymorphic viruses often use decryption loops
  - Integrity checking to find modified files
    - Record file sizes, checksums, MACs (keyed hashes of contents)
- Generic decryption and emulation scanners
  - Goal: detect so-called 'polymorphic' viruses with known body
  - Emulate CPU execution for a few hundred instructions, virus will eventually decrypt, can recognize known body
    - Does not work very well against metamorphic viruses and viruses not located near beginning of infected executable

# Signature-Based Scanning

- Hex strings from virus variants
  - `67 33 74 20 73 38 6D 35 20 76 37 61`
  - `67 36 74 20 73 32 6D 37 20 76 38 61`
  - `67 39 74 20 73 37 6D 33 20 76 36 61`
- Hex string for detecting virus
  - `67 ?? 74 20 73 ?? 6D ?? 20 76 ?? 61`
  - ?? = wildcard

# Virus Detection by Emulation




To detect an unknown mutation  of a known virus  ,
emulate CPU execution of  until the current sequence of
instruction opcodes matches the known sequence for virus body 

# Defeating Signature Scanning

- How do we do it? Signature are really just patterns of code

➔ The antidote to patterns is RANDOMNESS

- Anti-virus scanners detect viruses by looking for signatures (snippets of known virus code)
- Encrypted viruses: virus consists of a constant decryptor, followed by the encrypted virus body

- **Polymorphic** viruses
  - Viruses that mutate and/or encrypt parts of their code with a randomly generated key
- **Metamorphic** viruses
  - Viruses whose (possibly decrypted) body changes from variant to variant

# Evolution of Polymorphic Viruses

- Polymorphic viruses: constantly create new random encryptions of the same virus body
  - Marburg (Win95), HPS (Win95), Coke (Win32)
  - Virus must contain a polymorphic engine for creating new keys and new encryptions of its body
    - Rather than use an explicit decryptor in each mutation, Crypto virus (Win32) decrypts its body by brute-force key search
- Oligomorphic viruses: different versions of virus have different encryptions of the same body
  - Small number of decryptors (96 for Memorial viruses); to detect, must understand how they are generated

# Detecting Polymorphic Viruses

- Steps
  1. Run suspect program in an emulator
  2. Wait until it decrypts
  3. Decrypted code will be identical for various copies
  4. Use signature scanning on decrypted virus body
- Challenges
  - Determining when decryption is complete
  - Decryptor can determine whether its running in an emulator .. What is the potential hiccup here?

# Defeating Anti-Virus Emulators

- To detect polymorphic viruses, emulators execute suspect code for a little bit and look for opcode sequences of known virus bodies
- Some viruses use random code block insertion or insert millions of NOPs at the entry point prior to the main virus body
- Emulator executes code for a while, does not see virus body and decides the code is benign… when main virus body is finally executed, virus propagates

# Metamorphic Viruses

- Obvious next step: mutate the virus body, too!
- Virus can carry its source code (which deliberately contains some useless junk) and recompile itself
  - Apparition virus (Win32)
  - Virus first looks for an installed compiler
    - Unix machines have C compilers installed by default
  - Virus changes junk in its source and recompiles itself
    - New binary mutation looks completely different!
- Many macro and script viruses evolve and mutate their code
  - Macros/scripts are usually interpreted, not compiled

# Metamorphic Virus [Szor, 2001]

- An early generation

```
c7060F000055     mov  dword ptr [esi], 550000Fh
c746048BBC5151   mov  dword ptr [esi+0004],
5151BCBBh
```

- A later generation

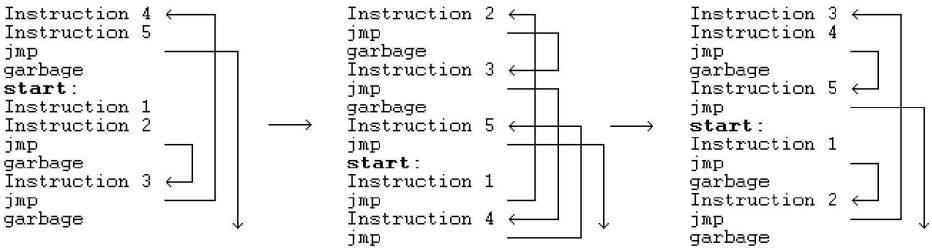
```
BF0F000055       mov  edi, 550000Fh
893E             mov  [esi], edi
5F          pop  edi
52          push edx
B640             mov  dh, 40
BA8BEC5151       mov  edx, 5151EC8Bh
53          push ebx
8BDA             mov  ebx, ebx
895E04           mov  [esi+0004], ebx
```

# Metamorphic Mutation Techniques

- Same code, different register names
  - Regswap (Win32)
- Same code, different subroutine order
  - BadBoy (DOS), Ghost (Win32)
  - If n subroutines, then n! possible mutations
- Decrypt virus body instruction by instruction, push instructions on stack, insert and remove jumps, rebuild body on stack
  - Zmorph (Win95)
  - Can be detected by emulation because the rebuilt body has a constant instruction sequence

# Example of Zperm Mutation

"Hunting for Metamorphic" at
www.symantec.com/avcenter/reference/**hunting**.for.**metamorphic**.pdf




- Body makeup not constant
- Detect by running suspect program in an emulato and analyze behaviour while running
- Can also disassemble and look for virus-like instructions

➔ Hard problem!

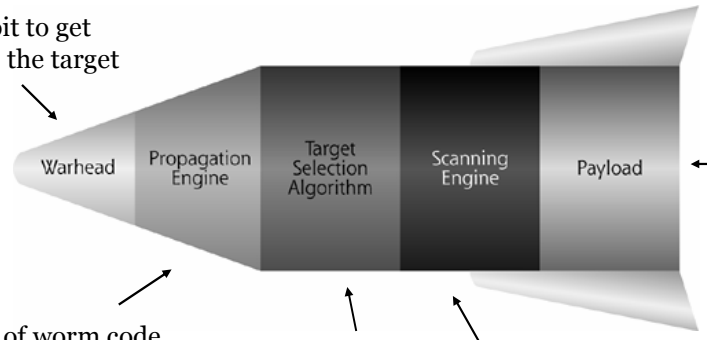
# Mutation Engines

- Real Permutating Engine/RPME (introduced in Zperm virus), ADMutate, many others
- Employ a large set of obfuscating techniques
  - Instructions are reordered, branch conditions reversed
  - Jumps and NOPs inserted in random places
  - Garbage opcodes inserted in unreachable code areas
  - Instruction sequences replaced with other instructions that have the same effect, but different opcodes
    - Mutate SUB EAX, EAX into XOR EAX, EAX or PUSH EBP; MOV EBP, ESP into PUSH EBP; PUSH ESP; POP EBP
- There is no constant, recognizable virus body!

# How Hard Is It to Write a Virus?

- 498 matches for "virus creation tool" in Spyware Encyclopedia
  - Including dozens of poly- and metamorphic engines
- OverWriting Virus Construction Toolkit
  - "The perfect choice for beginners"
- Biological Warfare Virus Creation Kit
  - Note: all viruses will be detected by Norton Anti-Virus
- Vbs Worm Generator (for Visual Basic worms)
  - Used to create the Anna Kournikova worm
- Many others

# Preview: Worms next time




Exploit to get 'in' to the target

Impact of worm, implements some specific action on behalf of attacker on the target

Pull rest of worm code into target (sometimes already contained in warhead)

Selection of new target addresses using different selection techniques

Scan targets and infect vulnerable targets

# Acknowledgements


- Some slides/material from
  - Ed Skoudis, “Malware”
  - Vitaly Shmatikov (U Texas)
  - SANS Storm Center
  - Szor and Ferrie (Symantec)

# For next week

- Read "With Microscope and Tweezers: An Analysis of the Internet Virus of November 1988" at http://cs.wellesley.edu/~cs342/internet_worm1988.pdf
- Read Weaver "How to own the Internet in your spare time" at http://cs.wellesley.edu/~cs342/owninternetinsparetime.pdf