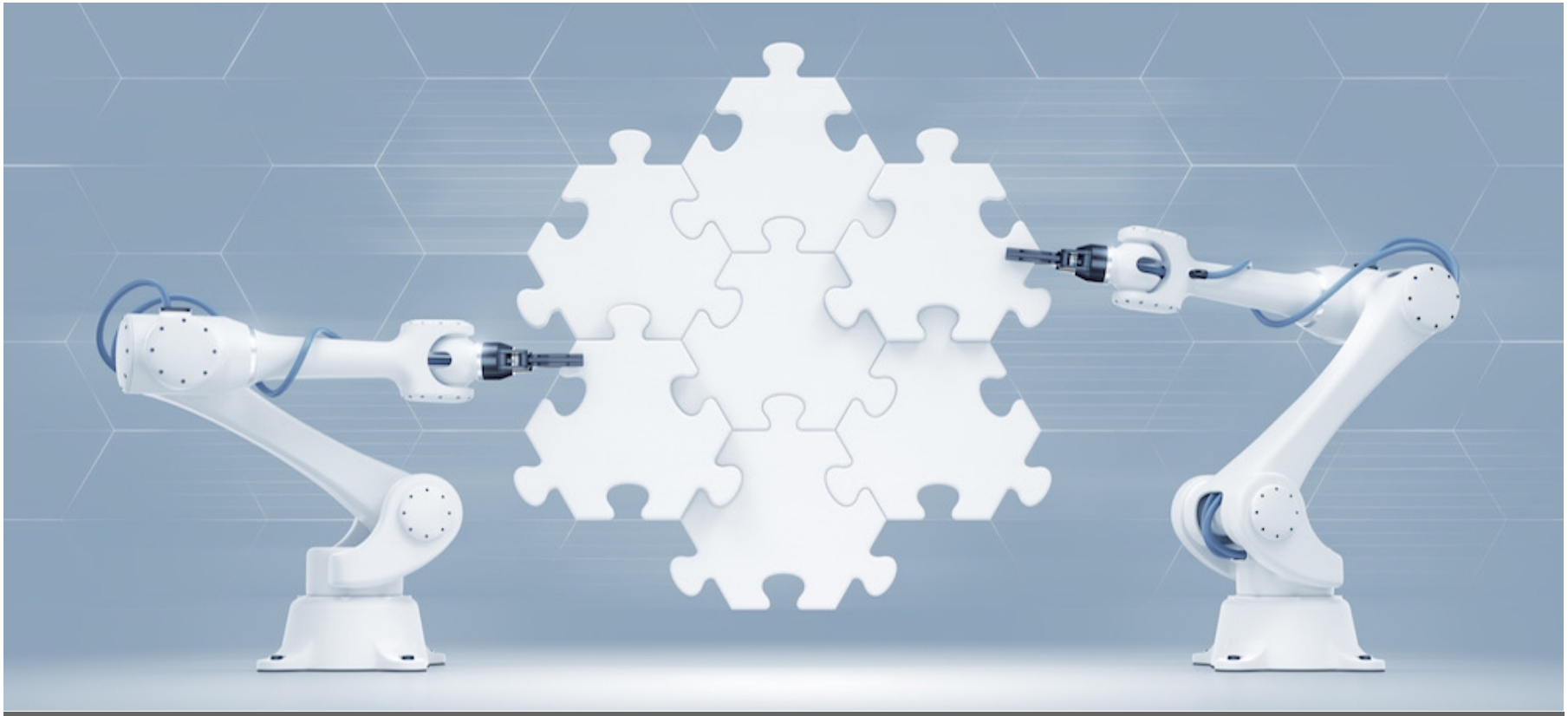

WE HOPE YOU HAD A NICE WINTER BREAK
WELCOME BACK. EVERYTHING IS FINE.
imgflip.com

Software Reverse Engineering

COMP 272 | Spring 2022 | University of the Pacific | Jeff Shafer



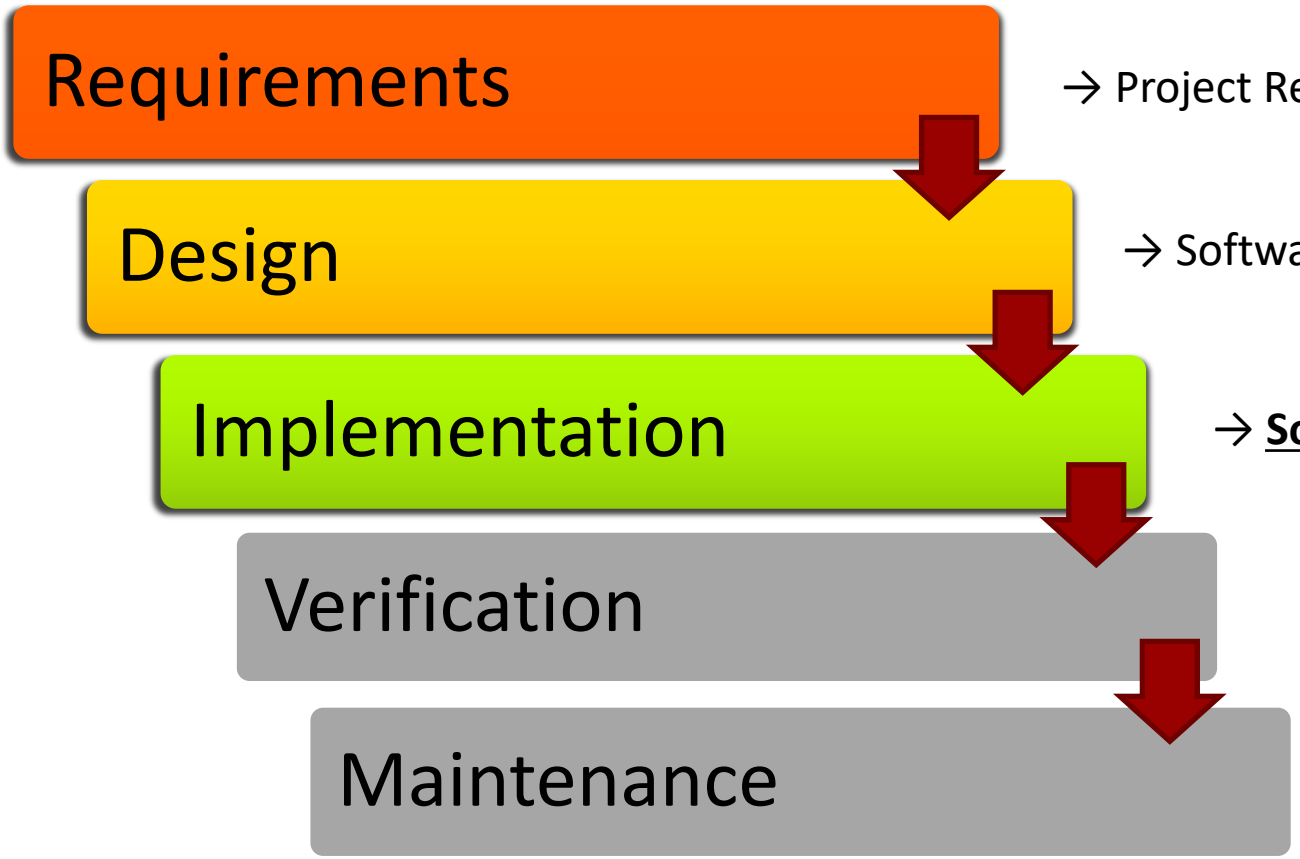
"Disintegrating" by Fabian Oefner
http://fabianoefner.com/

"Disintegrating" by Fabian Oefner
http://fabianoefner.com/

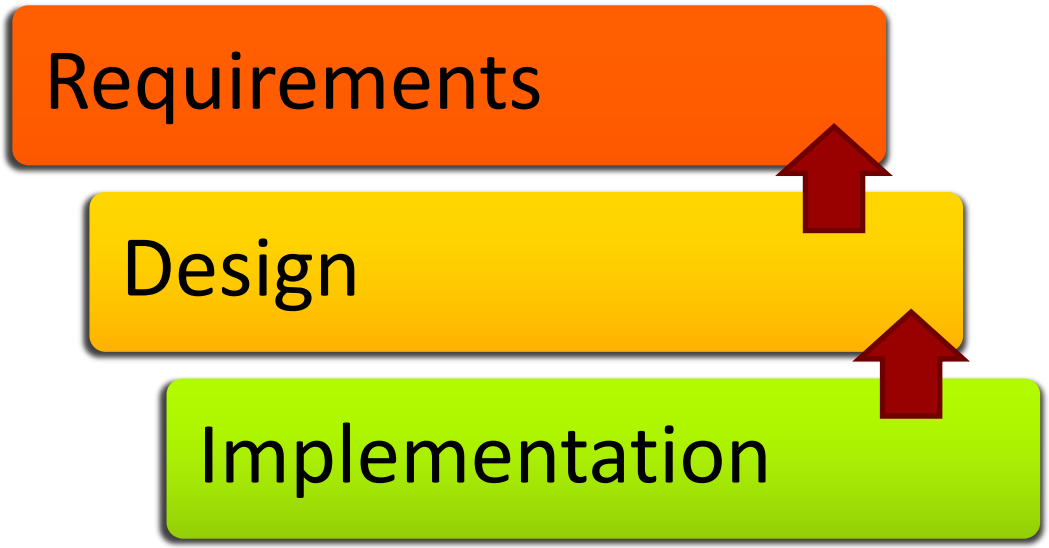

K CT 1263

Software Engineering

Classic Waterfall Model



Software Reverse Engineering



→ **Project Requirements Document**

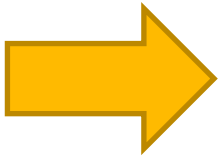
→ **Software Architecture**

→ Software Implementation

Software Reverse Engineering

Binary Code

0100110011…



Specifications

- *Algorithms?*
- *Design/Architecture?*
- *File I/O formats?*
- *Network protocols?*
- *Usage Instructions?*

+

- *Any available public documentation?*
- *Any available source code? (even if incomplete)*

Why Reverse Engineer Software?

- **Produce a competing product**
- Phoenix Technologies Ltd
  - Reverse engineered IBM PC BIOS in 1980's and sold IBM-compatible BIOS to PC clone manufacturers
- "Clean Room Design" method
  - Avoids copyright law (but not patent law)
    - The original IBM implementation was copyrighted – can't copy!
  - Team A examines original software and writes **detailed specification**
  - Lawyers review specification – any copyrighted material present?
  - Team B implements new software based *only* on specification (they never saw original software)

Why Reverse Engineer Software?

- **Provide interoperability with existing systems**
  - Network protocols, file types, operating systems, databases, …
- Samba
  - Compatible with Microsoft Windows
  - Open source (Linux, BSD, Mac OS, …)
  - SMB/CIFS file sharing protocol
  - Windows Server Domain Controller or member

Why Reverse Engineer Software?

- **Provide interoperability with existing systems**
  - Network protocols, file types, operating systems, databases, …
- WINE project
  - Windows API
- ReactOS
  - Windows API and ABI (Application Binary Interface)
- OpenOffice / LibreOffice
  - Microsoft Office (and many other) file types

Why Reverse Engineer Software?

- Customization of embedded systems
  - Replace stock firmware with enhanced firmware
    - Car engines, WiFi routers, etc…
- Generate documentation for orphan code (and possible code modifications, e.g. Y2K)
  - Consultant wrote app for your company 20 years ago but has gone out of business?
  - Documentation was lost / never produced in first place?
  - Business app was written 3 corporate reorganizations ago?
- Removal of copy protection
- Removal of gameplay restrictions

Why Reverse Engineer Software?

…. and?

Cybersecurity

Motivating Question for Class

- **What do you, as a security professional, need to know to answer two questions about a new binary?**
  - Is this binary *malicious*?
  - If it is malicious, *what does it do?*
- Same skillset needed as reverse engineering for other purposes

- **Lenny Zeltser**
- CISO at Axonius
  - Cybersecurity Asset Management
- Developer of **REMnux**
  - Linux distro for reverse engineering and analyzing malware

https://zeltser.com/

https://remnux.org/

Reverse Engineering for Security

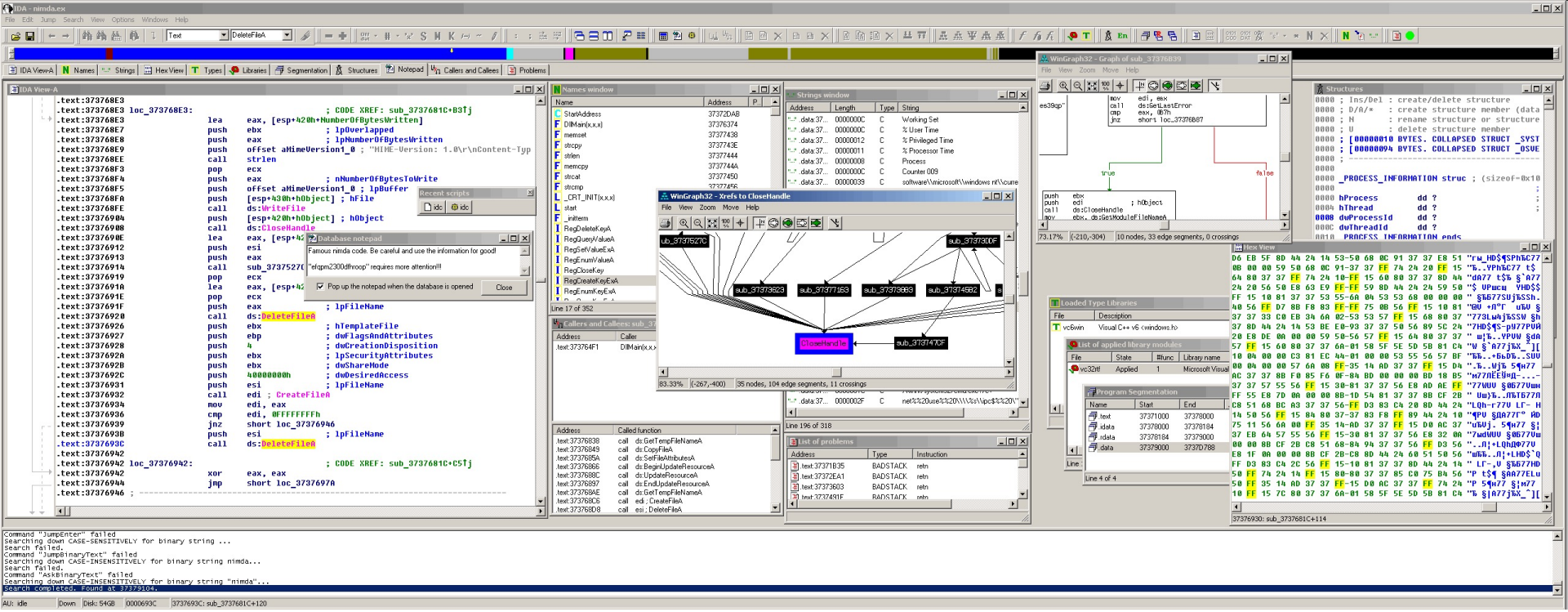
- **What can be learned by examining malicious software?**
  - Does the program pose a threat to your organization?
  - What are its capabilities?
  - How can the program be detected across enterprise systems? (Servers, desktops, middleware/network boxes)
  - What would data exfiltration (if purpose) look like on our network monitoring infrastructure?
    - Can we see *if* data was taken? Can we see *what* data was taken? (huge difference in HIPPA or PCI fines and penalties!)
  - Does the program reveal anything about your adversaries?
    - Are they targeting *you* specifically?
    - What are their capabilities?
    - Origins of code? (Did *they* develop it?)
    - Similarities to earlier malicious programs?

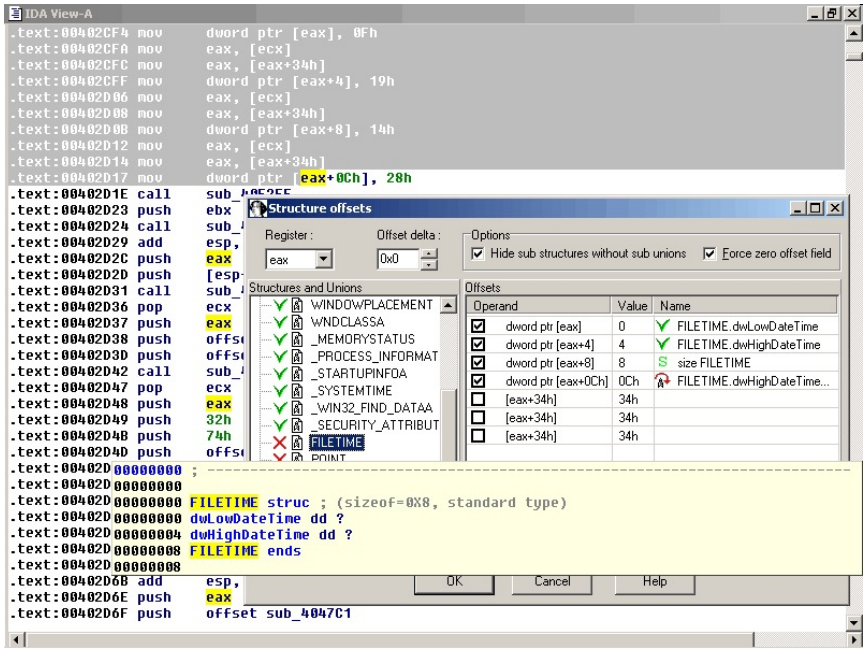
Malware Analysis Report

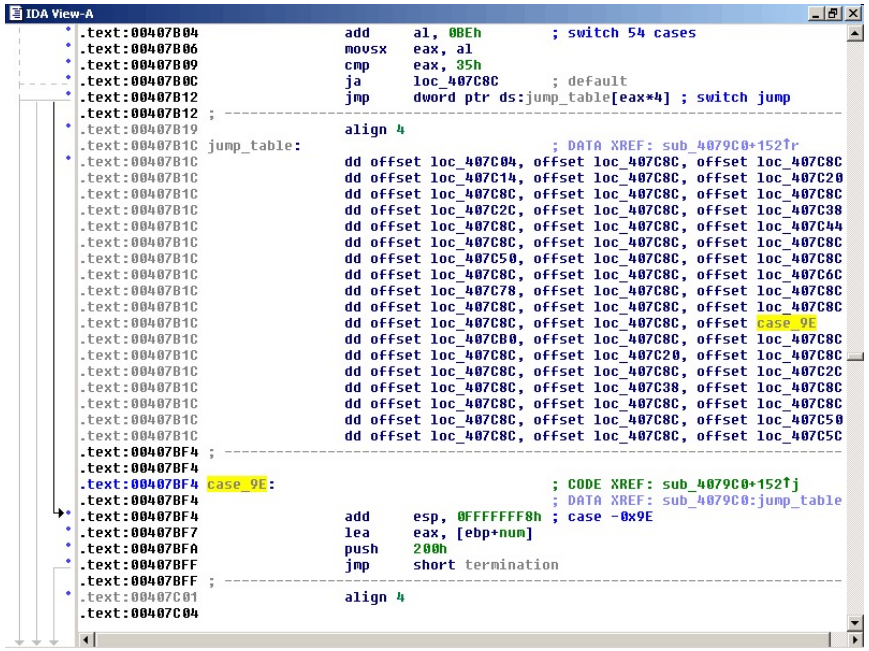
- What to include in an *Analysis Report* for your boss?
  - Capabilities
  - Identification (name, size, hashes, …)
  - Characteristics
    - Infection, exfiltration, anti-analysis, …
  - Dependencies for operation
  - Behavioral and code analysis
  - Tables and Figures (lots!)
  - Indicators of Compromise (IOC) for NOC

https://zeltser.com/malware-analysis-report/

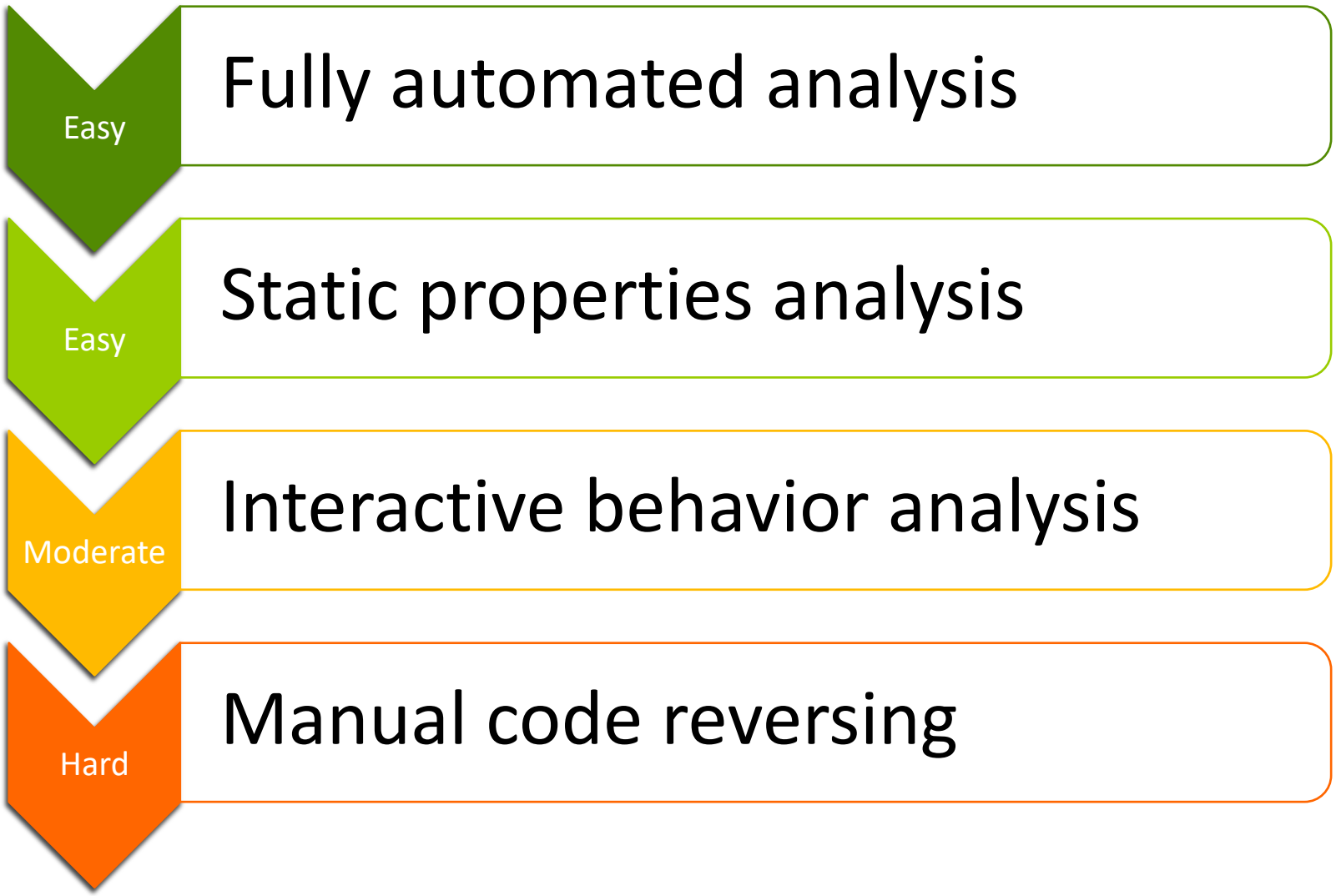
Sounds great, so let’s dive into the assembly code!


IDA - nimda.exe
IDA View-A
Names window
Strings window
WinGraph32 - Graph of sub_37376B39
Structures
WinGraph32 - Xrefs to CloseHandle
Database notepad
Famous nimda code. Be careful and use the information for good!
"efqpm2300dfhroop" requires more attention!!!
Pop up the notepad when the database is opened
Close
Recent scripts
idc
Hex View
Loaded Type Libraries
List of applied library modules
Program Segmentation
List of problems
Callers and Callees: sub_37
Command "JumpEnter" failed
Searching down CASE-SENSITIVELY for binary string ...
Search failed.
Command "JumpBinaryText" failed
Searching down CASE-INSENSITIVELY for binary string nimda...
Search failed.
Command "AskBinaryText" failed
Searching down CASE-INSENSITIVELY for binary string "nimda"...
Search completed. Found at 37379104.


IDA View-A
.text:00402CF4 mov dword ptr [eax], 0Fh
.text:00402CFA mov eax, [ecx]
.text:00402CFC mov eax, [eax+34h]
.text:00402CFF mov dword ptr [eax+4], 19h
.text:00402D06 mov eax, [ecx]
.text:00402D08 mov eax, [eax+34h]
.text:00402D0B mov dword ptr [eax+8], 14h
.text:00402D12 mov eax, [ecx]
.text:00402D14 mov eax, [eax+34h]
.text:00402D17 mov dword ptr [eax+0Ch], 28h
Structure offsets
Register :
eax
Offset delta :
0x0
Options
Hide sub structures without sub unions
Force zero offset field
Structures and Unions
WINDOWPLACEMENT
WNDCLASSA
_MEMORYSTATUS
_PROCESS_INFORMAT
_STARTUPINFOA
_SYSTEMTIME
_WIN32_FIND_DATAA
_SECURITY_ATTRIBUT
FILETIME
POINT
Offsets
Operand
Value
Name
dword ptr [eax] 0 FILETIME.dwLowDateTime
dword ptr [eax+4] 4 FILETIME.dwHighDateTime
dword ptr [eax+8] 8 size FILETIME
dword ptr [eax+0Ch] 0Ch FILETIME.dwHighDateTime...
[eax+34h] 34h
[eax+34h] 34h
[eax+34h] 34h
.text:00000000 ; ---
.text:00000000
.text:00000000 FILETIME struc ; (sizeof=0X8, standard type)
.text:00000000 dwLowDateTime dd ?
.text:00000004 dwHighDateTime dd ?
.text:00000008 FILETIME ends
.text:00000008
OK
Cancel
Help
.text:00402D6B add esp,
.text:00402D6E push eax
.text:00402D6F push offset sub_4047C1


IDA View-A
.text:00407B04 add al, 0BEh ; switch 54 cases
.text:00407B06 movsx eax, al
.text:00407B09 cmp eax, 35h
.text:00407B0C ja loc_407C8C ; default
.text:00407B12 jmp dword ptr ds:jump_table[eax*4] ; switch jump
.text:00407B12 ; ---
.text:00407B19 align 4
.text:00407B1C jump_table: ; DATA XREF: sub_4079C0+152↑r
.text:00407B1C dd offset loc_407C04, offset loc_407C8C, offset loc_407C8C
.text:00407B1C dd offset loc_407C14, offset loc_407C8C, offset loc_407C20
.text:00407B1C dd offset loc_407C8C, offset loc_407C8C, offset loc_407C8C
.text:00407B1C dd offset loc_407C2C, offset loc_407C8C, offset loc_407C38
.text:00407B1C dd offset loc_407C8C, offset loc_407C8C, offset loc_407C44
.text:00407B1C dd offset loc_407C8C, offset loc_407C8C, offset loc_407C8C
.text:00407B1C dd offset loc_407C50, offset loc_407C8C, offset loc_407C8C
.text:00407B1C dd offset loc_407C8C, offset loc_407C8C, offset loc_407C6C
.text:00407B1C dd offset loc_407C78, offset loc_407C8C, offset loc_407C8C
.text:00407B1C dd offset loc_407C8C, offset loc_407C8C, offset loc_407C8C
.text:00407B1C dd offset loc_407C8C, offset loc_407C8C, offset case_9E
.text:00407B1C dd offset loc_407CB0, offset loc_407C8C, offset loc_407C8C
.text:00407B1C dd offset loc_407C8C, offset loc_407C20, offset loc_407C8C
.text:00407B1C dd offset loc_407C8C, offset loc_407C8C, offset loc_407C2C
.text:00407B1C dd offset loc_407C8C, offset loc_407C38, offset loc_407C8C
.text:00407B1C dd offset loc_407C8C, offset loc_407C8C, offset loc_407C50
.text:00407B1C dd offset loc_407C8C, offset loc_407C8C, offset loc_407C5C
.text:00407BF4 ; ---
.text:00407BF4
.text:00407BF4 case_9E: ; CODE XREF: sub_4079C0+152↑j
.text:00407BF4 ; DATA XREF: sub_4079C0:jump_table
.text:00407BF4 add esp, 0FFFFFFF8h ; case -0x9E
.text:00407BF7 lea eax, [ebp+num]
.text:00407BFA push 200h
.text:00407BFF jmp short termination
.text:00407BFF ; ---
.text:00407C01 align 4
.text:00407C04

Hmmn, perhaps *you* should jump first…

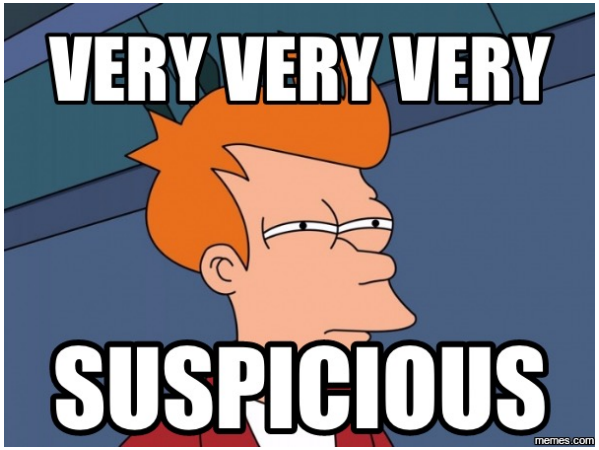

Easy
Fully automated analysis
Easy
Static properties analysis
Moderate
Interactive behavior analysis
Hard
Manual code reversing

Fully Automated Analysis

- Try this first, but don't get your hopes up!
  - The more sophisticated the malware, the more resistant it will be to generic automated tools
- Automated sandbox tools
  - Hybrid Analysis
  - Cuckoo Sandbox

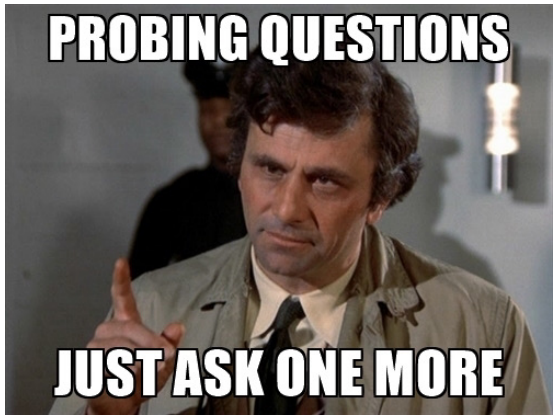
Static Properties Analysis

- The program is *not running* – just inspect binary file on disk
- Triage step – Is this file *even* malware? Is there anything interesting about it to warrant further investigation?
  - *Always more unknown binaries than engineer time available to analyze them*
  - Are there suspicious strings? Or a total lack of un-obfuscated strings?
  - Are there suspicious libraries? Or a total lack of un-obfuscated libraries?
  - Are there suspicious API calls? Or a total lack (??) of un-obfuscated API calls?
- Tools
  - PEStudio
  - Detect It Easy
  - pescanner.py
  - strings



Interactive Behavior Analysis

- Binary file *looks suspicious*, but we have a lot of unanswered questions!
  - You *could* dive into the assembly code, but some important questions might be answerable via an easier method
  - **Time to run it (safely!) and see what it does**
- Carefully observe
  - API calls made (user and kernel space)
  - Filesystem accessed (read/write)
  - Registry accessed (read/write) – *Windows-only*
  - Network communication
- Tools
  - Process Hacker
  - RegShot
  - Wireshark (+ other tools, e.g. fakedns)
  - API Monitor



Manual Code Reversing

Course Overview

Websites

Main website

- https://cyberlab.pacific.edu/

Canvas CMS (gradebook, labs, …)

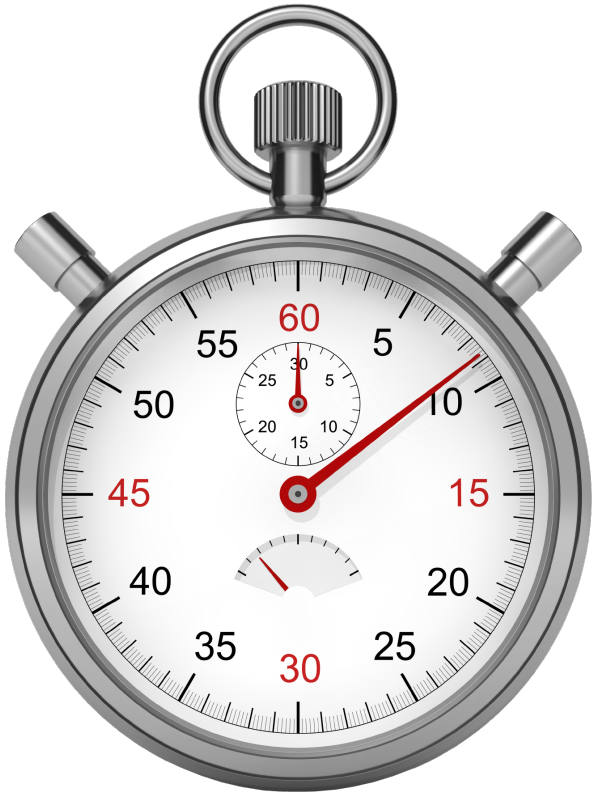
- http://canvas.pacific.edu

Textbook

- **No official textbook**
- May require technical paper readings prior to class lectures/discussion
  - Will announce in advance

Courseware Version 0.1a

- Only 15 weeks in a semester
- 29 classes, including today
  - **The clock is ticking now!**
- What to cover?



Class Topics

- **Windows** malware
  - Not focusing on Linux
  - Not focusing on OS X
  - Not focusing on mobile
  - *Different tools and different critical APIs, but identical skills and thought process*
- **Document file** malware
- **Web (JavaScript)** malware
- Tools
  - More Tools…
  - Yet More Tools….
  - Tool A doesn't work with Malware Z? Try Tool B….
- **x86, x86-64 assembly code**
- Anti-RE and obfuscation techniques
- Communication protocol recovery

Course Components

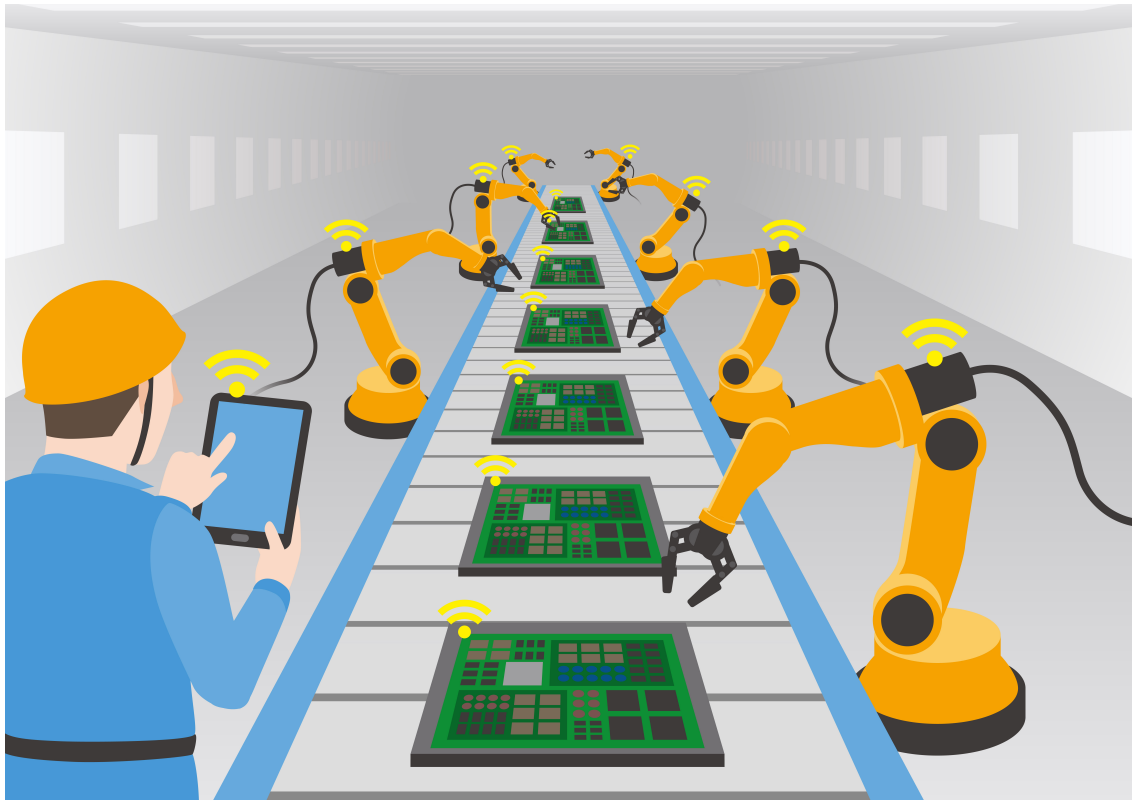
- **50%** - **Labs**
  - Begin in-class, finish at home
  - Hands-on experience using analysis tools and working with real malware
- **25%** - **Lab Practical Exams** (2)
  - Demonstrate that you learned something in the labs!

Course Components

- **25%** - **Projects** (Malware analysis)
  - *Here's a binary that the IT intern pulled off our Exchange server. I need an analysis report on my desk by 5pm Friday…*
  - Will start early in the semester and expand reports as our skills develop

This Week

- Malware Analysis Basics – Automated Tools



Questions?

- Questions?
- Concerns?