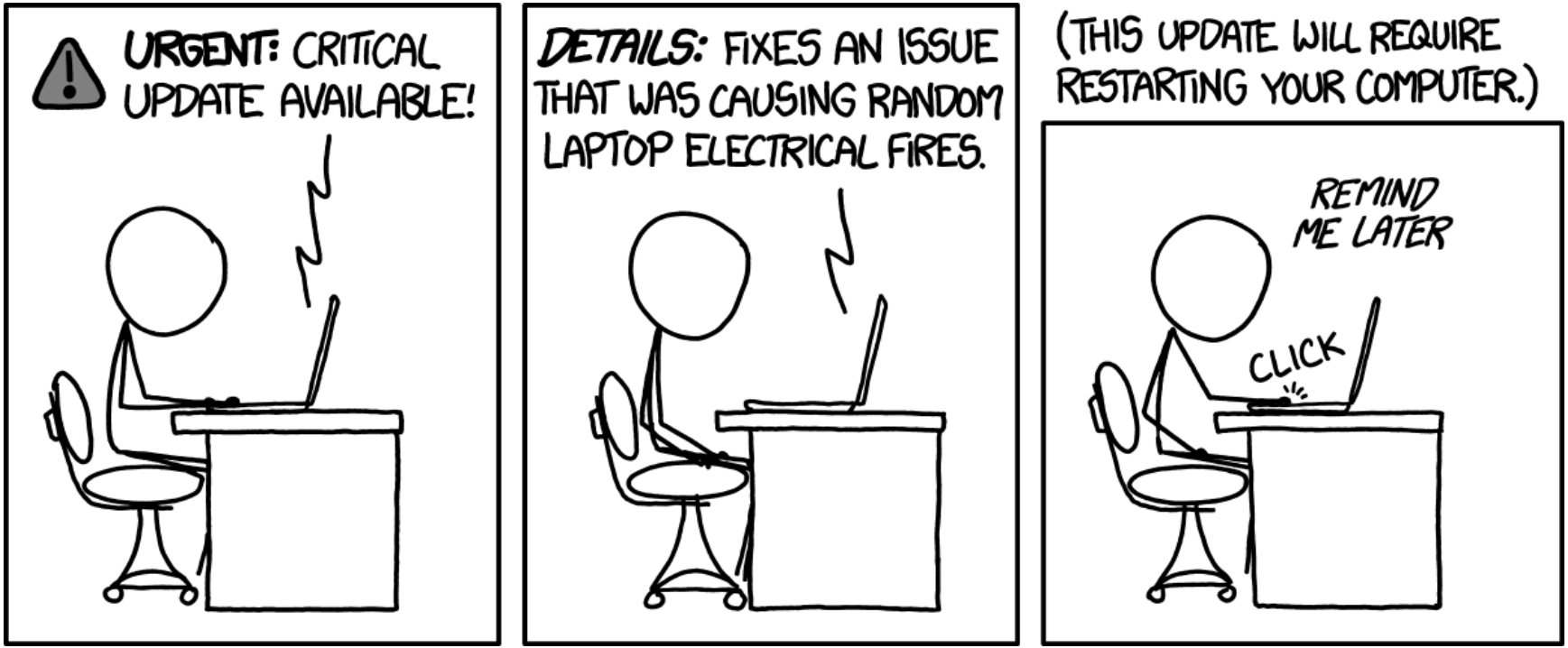
PV204 Security technologies

In-Memory Malware Analysis

Václav Lorenc
Principal Security Engineer, Here Technologies

CROCS
Centre for Research on Cryptography and Security

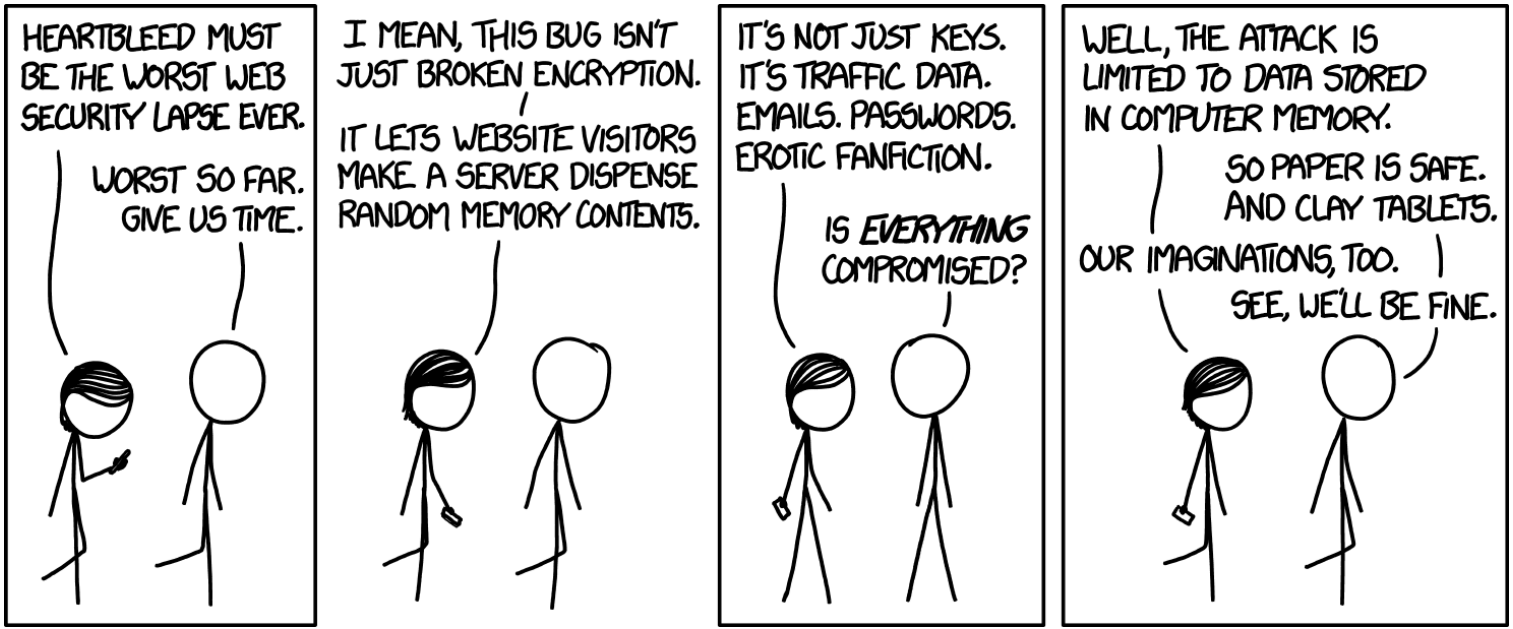

URGENT: CRITICAL UPDATE AVAILABLE!
DETAILS: FIXES AN ISSUE THAT WAS CAUSING RANDOM LAPTOP ELECTRICAL FIRES.
(THIS UPDATE WILL REQUIRE RESTARTING YOUR COMPUTER.)
REMIND ME LATER
CLICK

Agenda

- Motivation!
  - No x86 assembly required
  - No malware (de)obfuscation magic
- How does an OS look “inside”?
  - Processes and other data structures
  - How the memory is organized
- Common tools used for analysis
- Searching for system “oddities”
  - What are the important system indicators?
- Real samples discussed and analyzed! (Labs)

Why memory analysis?

- **It's fun!**
- Acquiring evidence for legal investigations
  - It used to be different in the past
- Technical simplification of reverse engineering
  - No binary obfuscation present – the code has to run
- Incident response activities
  - Easy way how to learn more about the attackers
  - **Malicious binary may only be present in memory**
  - **Fast**: RAM is (usually) smaller than full hard-drive images


HEARTBLEED MUST BE THE WORST WEB SECURITY LAPSE EVER.
WORST SO FAR. GIVE US TIME.
I MEAN, THIS BUG ISN'T JUST BROKEN ENCRYPTION.
IT LETS WEBSITE VISITORS MAKE A SERVER DISPENSE RANDOM MEMORY CONTENTS.
IT'S NOT JUST KEYS. IT'S TRAFFIC DATA. EMAILS. PASSWORDS. EROTIC FANFICTION.
IS EVERYTHING COMPROMISED?
WELL, THE ATTACK IS LIMITED TO DATA STORED IN COMPUTER MEMORY.
SO PAPER IS SAFE. AND CLAY TABLETS.
OUR IMAGINATIONS, TOO.
SEE, WE'LL BE FINE.

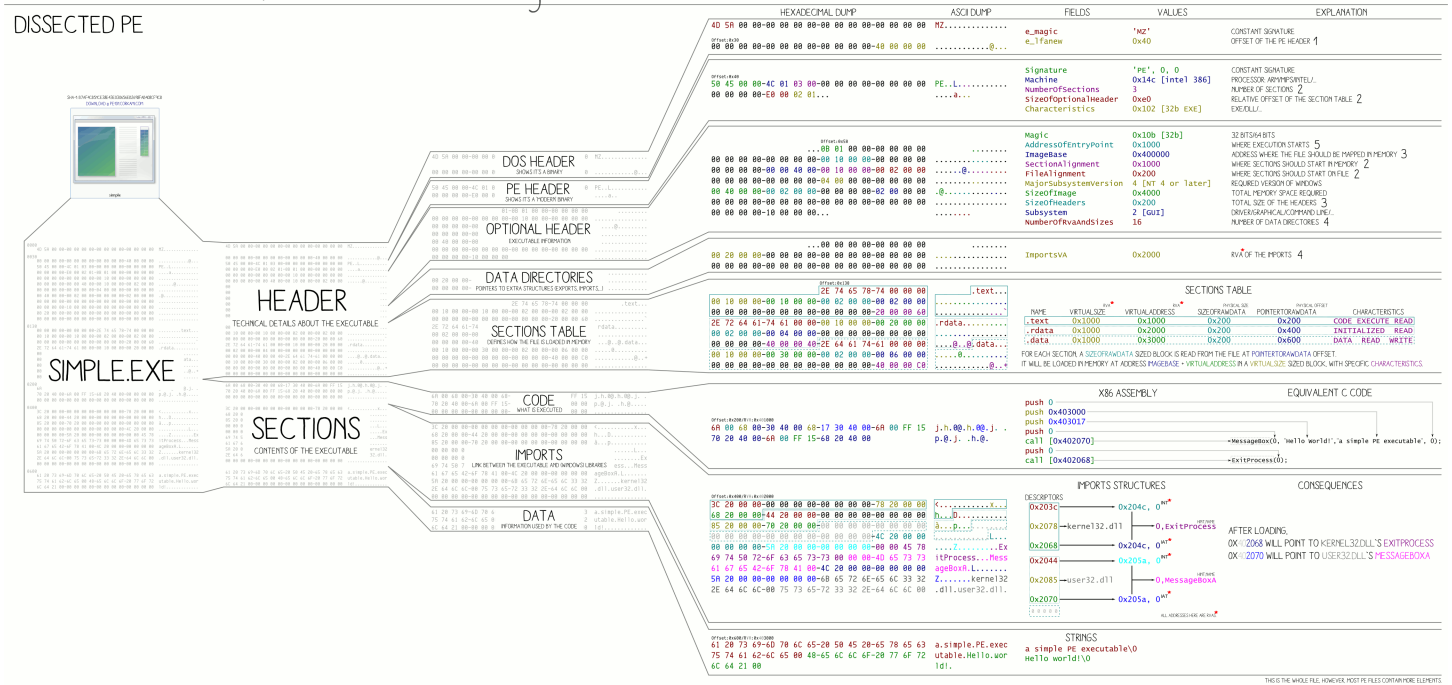
Challenges in Reverse Engineering (RE)

- Assembly language (for multiple platforms)
  - Along with undocumented instructions (or behavior)
- Anti-debugging tricks
  - Exceptions, interrupts, PE manipulations, time checking, ...
- Anti-VM tricks
  - Uncommon behavior of known instructions
  - Registry detections, HW detections
- Code obfuscation/packing
  - The most challenging to overcome, mostly

PE 101 a windows executable walk-through

ANGE ALBERTINI
CORKAMI.COM

DISSECTED PE



LOADING PROCESS

1 HEADERS

THE DOS HEADER IS PARSED
THE PE HEADER IS PARSED
- IT'S OFFSET (DOS HEADER'S E_LFANEW)

THE OPTIONAL HEADER IS PARSED
- IT FOLLOWS THE PE HEADER

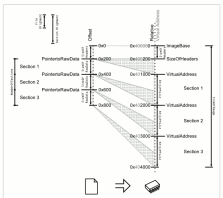
2 SECTIONS TABLE

SECTIONS TABLE IS PARSED
- IT IS LOCATED AT: OFFSET (OPTIONALHEADER + SIZEOFOPTIONALHEADER)

IT CONTAINS NUMBEROFSECTIONS ELEMENTS
IT IS CHECKED FOR VALIDITY WITH ALIGNMENTS
- FILEALIGNMENTS AND SECTIONALIGNMENTS

3 MAPPING

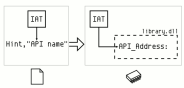
THE FILE IS MAPPED IN MEMORY ACCORDING TO:
- THE IMAGEBASE
- THE SIZEOFHEADERS
- THE SECTIONS TABLE



4 IMPORTS

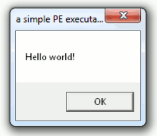
DATADIRECTORIES ARE PARSED
- THEY FOLLOW THE OPTIONALHEADER
- THEIR NUMBER IS NUMBEROFRVAANDSIZES

IMPORTS ARE ALWAYS #2
IMPORTS ARE PARSED
- EACH DESCRIPTOR SPECIFIES A DLL NAME
  - THIS DLL IS LOADED IN MEMORY
  - IAT AND INT ARE PARSED SIMULTANEOUSLY FOR EACH API IN INT
    - ITS ADDRESS IS WRITTEN IN THE IAT ENTRY



5 EXECUTION

CODE IS CALLED AT THE ENTRYPOINT
THE CALLS OF THE CODE GO VIA THE IAT TO THE APIS



NOTES

MZ HEADER AKA DOS_HEADER
STARTS WITH 'MZ' (INITIALS OF MARK ZBIKOWSKI MS-DOS DEVELOPER)

PE HEADER AKA IMAGE_FILE_HEADERS / COFF FILE HEADER
STARTS WITH 'PE' (PORTABLE EXECUTABLE)

OPTIONAL HEADER AKA IMAGE_OPTIONAL_HEADER
OPTIONAL ONLY FOR NON-STANDARD PES BUT REQUIRED FOR EXECUTABLES

RVA RELATIVE VIRTUAL ADDRESS
ADDRESS RELATIVE TO IMAGEBASE (AT IMAGEBASE, RVA = 0)
ALMOST ALL ADDRESSES OF THE HEADERS ARE RVAS
IN CODE, ADDRESSES ARE NOT RELATIVE

INT IMPORT NAME TABLE
NULL-TERMINATED LIST OF POINTERS TO HINT, NAME STRUCTURES

IAT IMPORT ADDRESS TABLE
NULL-TERMINATED LIST OF POINTERS
ON FILE, IT IS A COPY OF THE INT
AFTER LOADING IT POINTS TO THE IMPORTED APIS

HINT
INDEX IN THE EXPORTS TABLE OF A DLL TO BE IMPORTED
NOT REQUIRED BUT PROVIDES A SPEED-UP BY REDUCING LOOK-UP

PE File Format

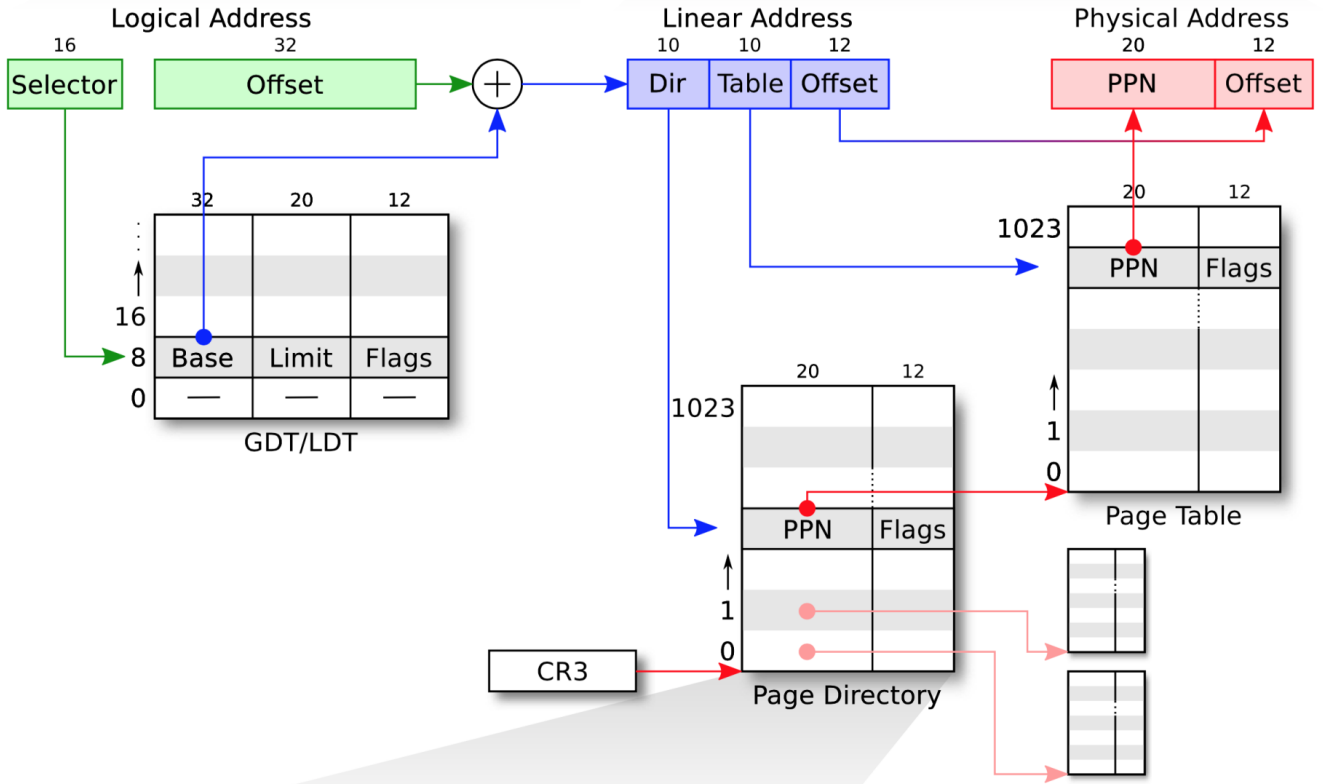
‘cause reverse engineering ninjas are busy

MEMORY ANALYSIS

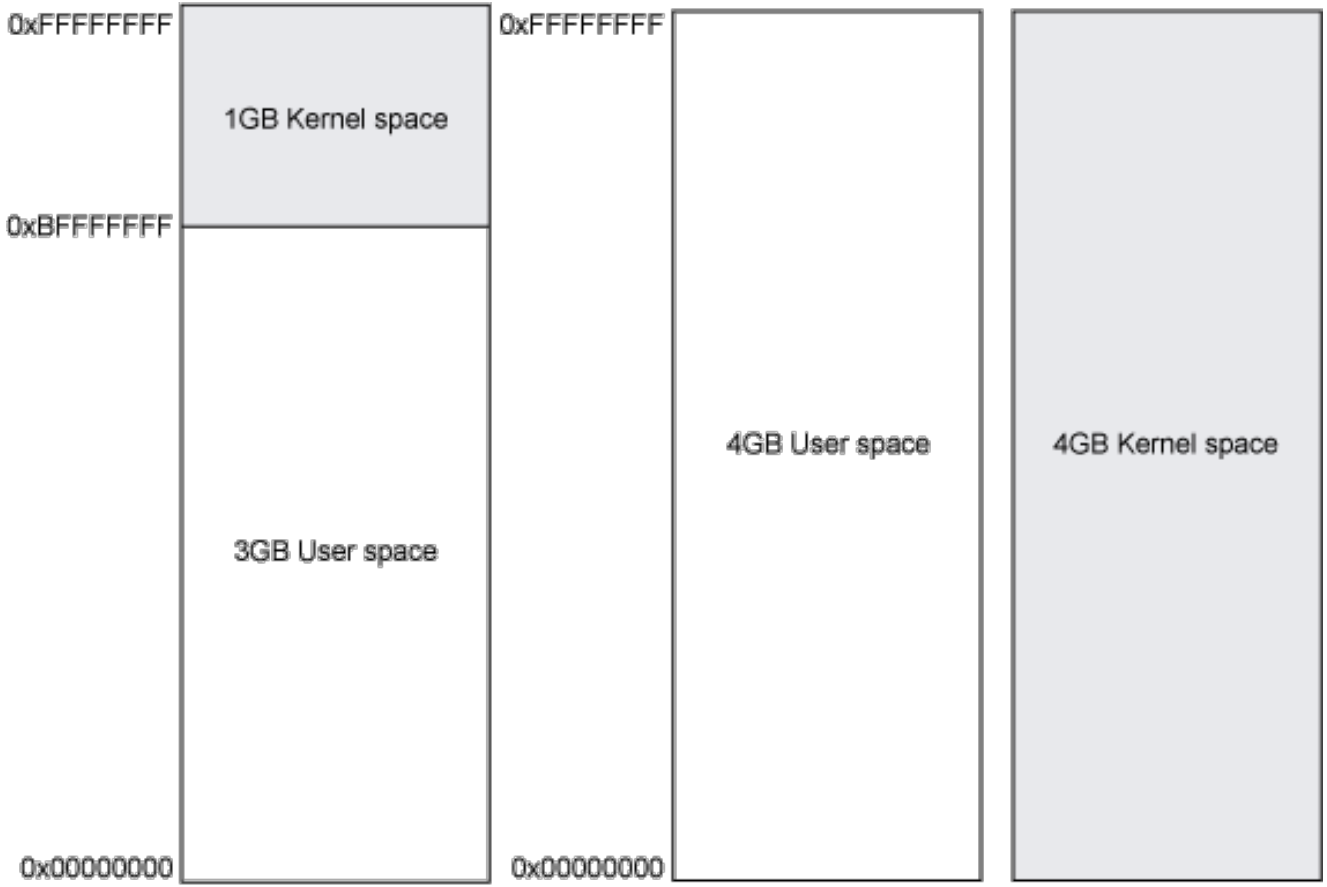
x86/x64 Memory organization

- Physical memory
  - RAM; what we really have installed
- Virtual memory
  - Separation of logical process memory from the physical
  - Logical address space > physical (e.g., swap)
  - Address space shared by several processes, yet separated
- Paging vs. Segmentation
  - Possible memory organization approaches

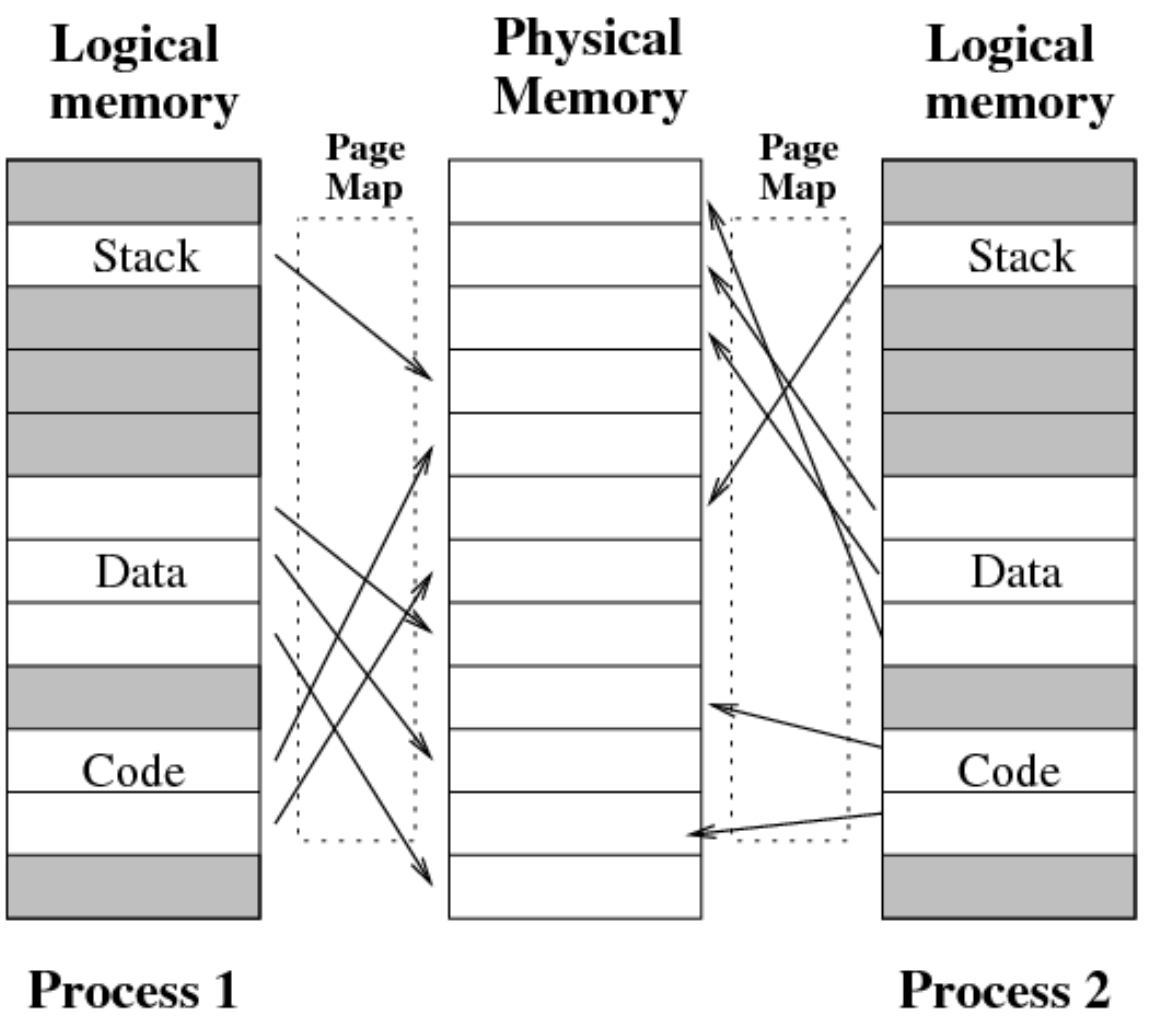
Segmentation | **Paging** | **Physical Address**

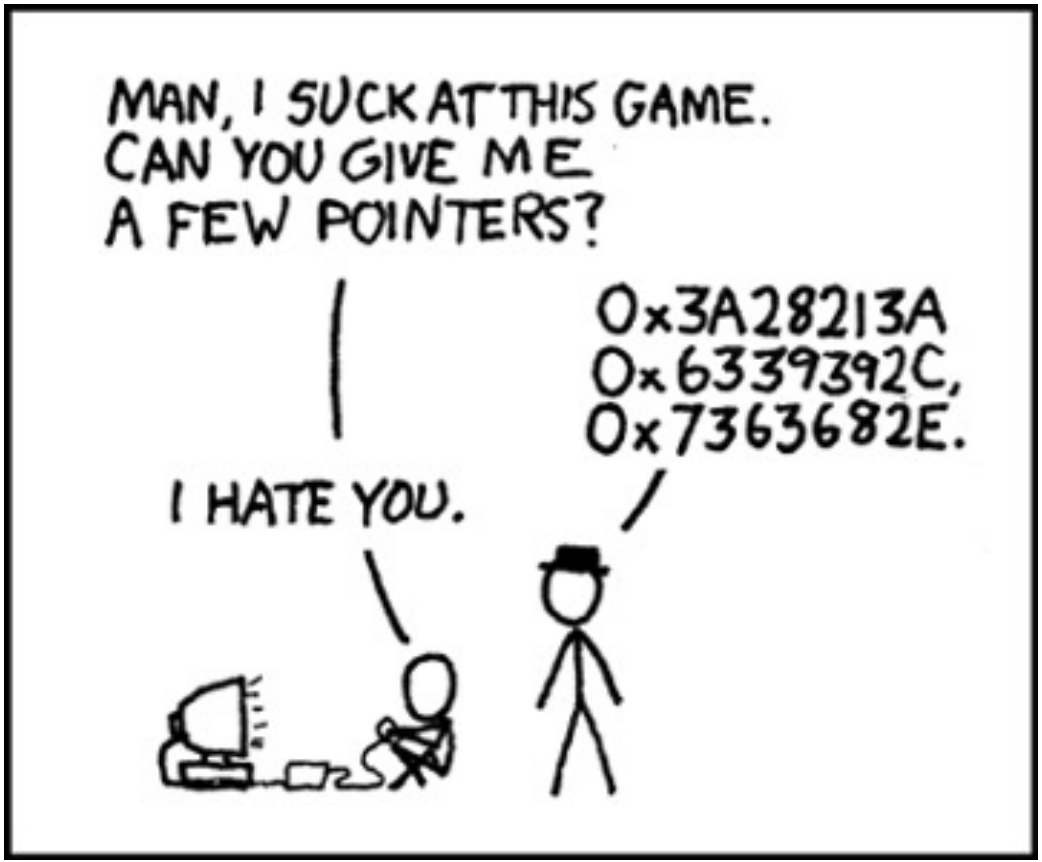


Win32 Address Space



Linux Address Space

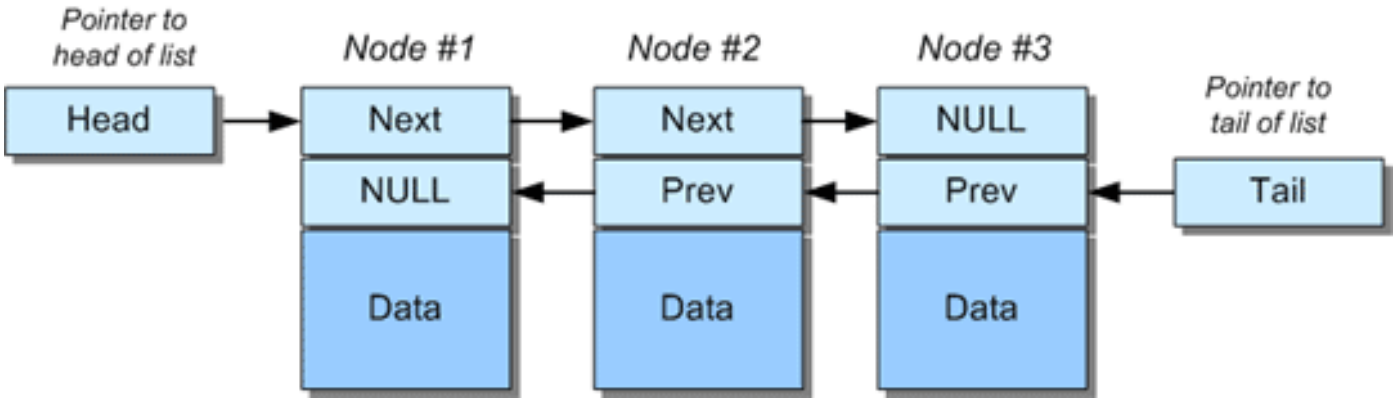

Logical memory
Physical Memory
Logical memory
Page Map
Page Map
Stack
Data
Code
Stack
Data
Code
Process 1
Process 2


MAN, I SUCK AT THIS GAME.
CAN YOU GIVE ME
A FEW POINTERS?
0x3A28213A
0x6339392C,
0x7363682E.
I HATE YOU.

Operating System Data Structures

- How the OS knows about processes, files, …?
  - A lot of ‘metadata’ for important data
  - Based on C/C++ data structures (see MSDN documentation)
- (Double-)linked list
  - Another common data structure (not only in OS)
  - Method for implementing lists in computer memory
- Direct Kernel Object Manipulation (DKOM)
  - Used for manipulating the structures to hide malicious stuff

Double Linked Lists



DKOM – Direct Kernel Object Manipulation

- Dozens of various (double-)linked lists in Windows
  - Maintained by kernel
  - Processes, threads, opened files, memory allocations, …
- DKOM is used by rootkits
  - Hiding from the sight of the user
- Rootkit paradox
  - Rootkits need to run on the system
  - … and need to remain hidden at the same time
- Memory analysis can help to discover DKOM
  - Anti-analysis techniques are known as well

Interesting OS Structures

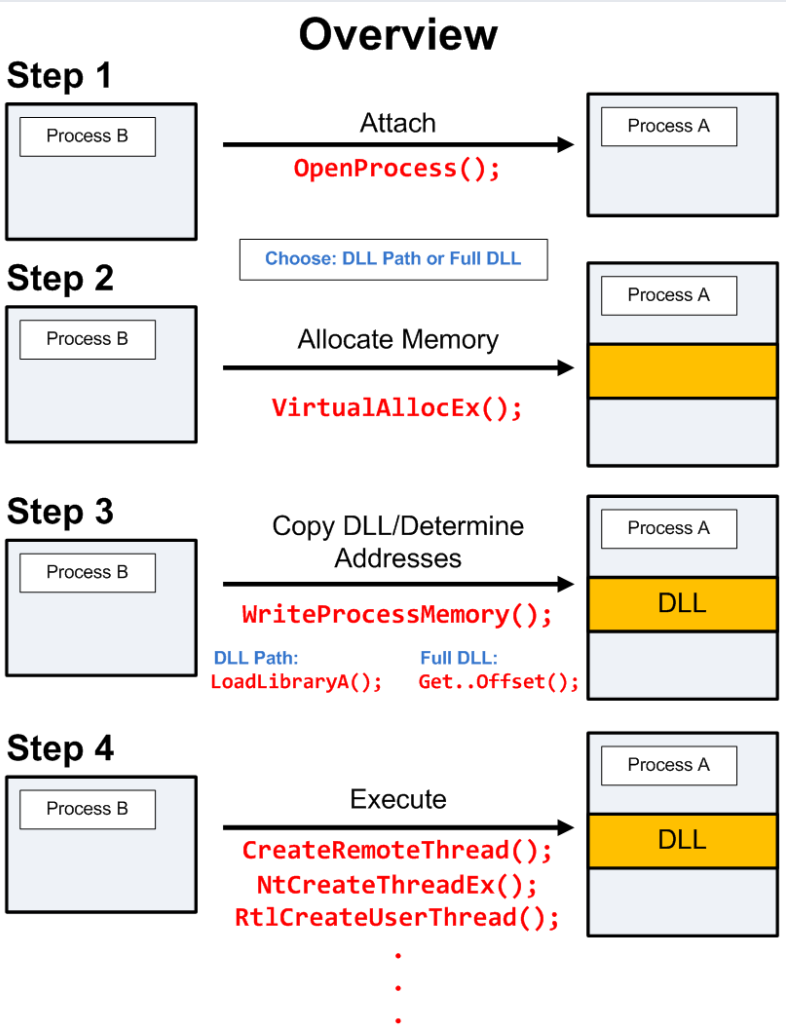
- Suspicious Memory Pages
- Processes
- Threads
- Sockets (Connections)
- Handles (Files)
- Recently executed binaries
- Modules/Libraries
- Mutexes
- LSA (Local Security Authority)
- Registry
- …
- Files
- Caches

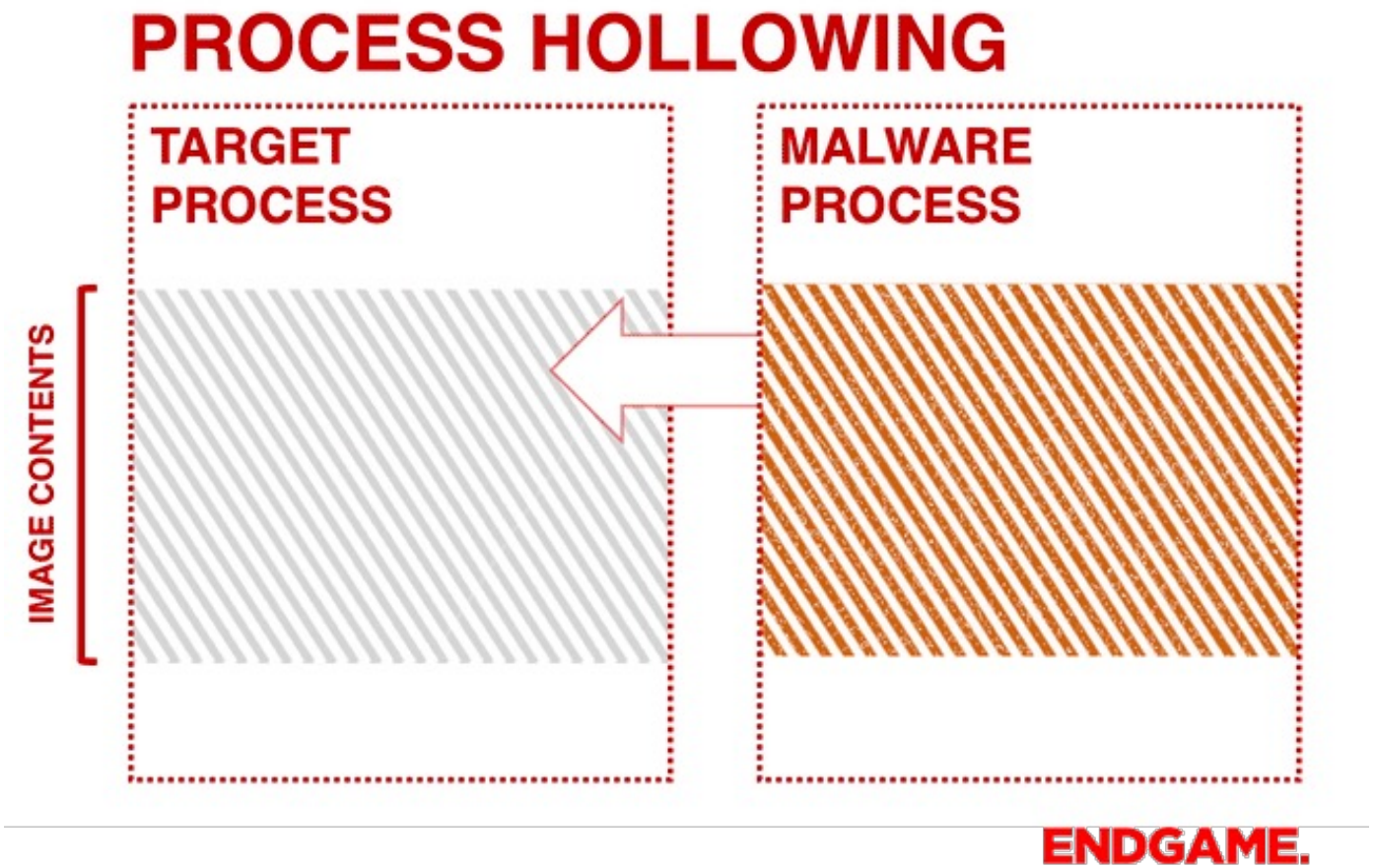
Memory Pages

- Various ‘flags’
  - Read/write/executable pages
  - Helping OS to organize memory efficiently
- Executable + Writable pages
  - Why is it bad?
- **Process Injection Technique(s)**
  - Allocating a memory that can be modified (unpacked, decoded, decrypted) and executed.
  - Used by legitimate processes too (Windows OLE)

DLL/Process Injection

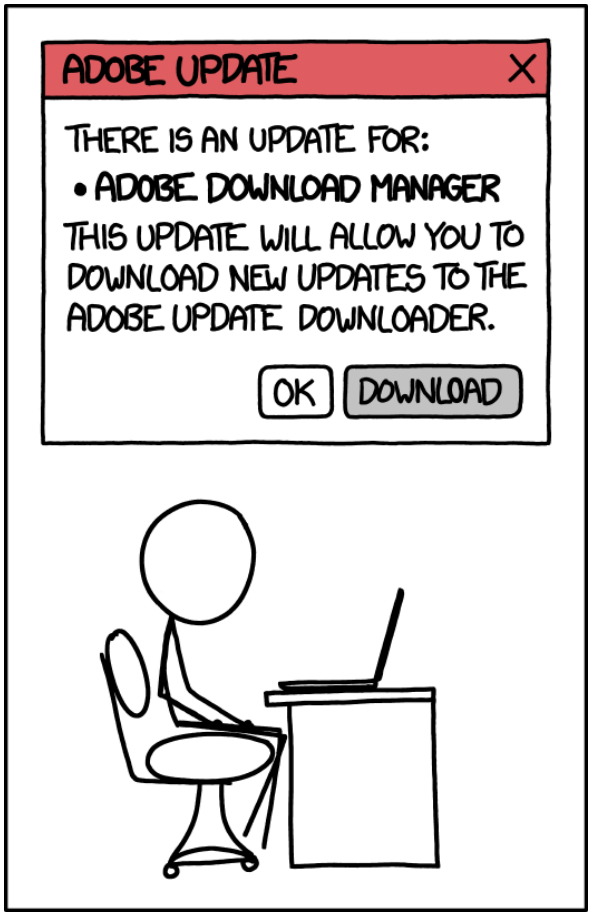
So that Internet Explorer behaves like a malicious process…




PROCESS HOLLOWING
TARGET PROCESS
MALWARE PROCESS
IMAGE CONTENTS
ENDGAME.

PRACTICAL

AND NOW SOMETHING COMPLETELY…


ADOBE UPDATE
THERE IS AN UPDATE FOR:
• ADOBE DOWNLOAD MANAGER
THIS UPDATE WILL ALLOW YOU TO DOWNLOAD NEW UPDATES TO THE ADOBE UPDATE DOWNLOADER.
OK
DOWNLOAD

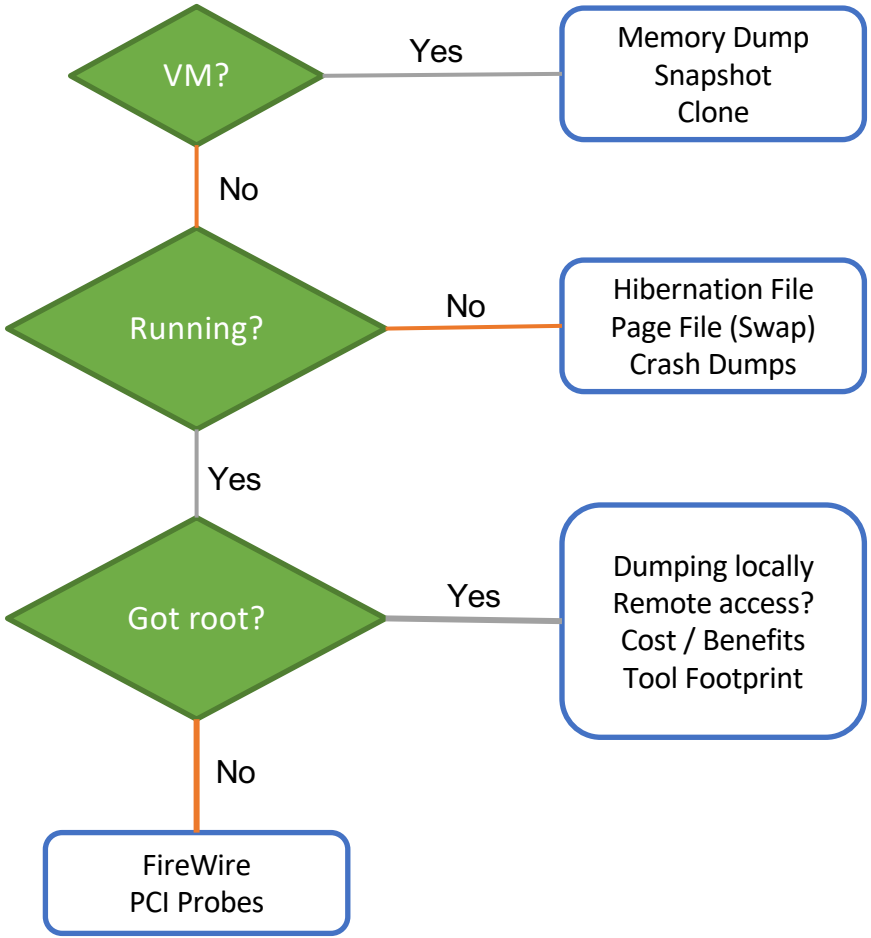
Phase #1

MEMORY ACQUISITION

Memory (re)sources

- Live RAM
  - The most common source for analysis
  - Easier to obtain from virtualized hosts
- Paging file/Swap
  - Used by operating systems to allocate more memory then available RAM
- Hibernation file
- Memory crash dumps
  - Limited analysis options

Memory Acquisition



Memory Acquisition

- **Virtual Machines**
  - VMWare, VirtualBox, …
  - `VirtualBox -dbg -startvm "MalwareVM"` (and `.pgmphystofile` command or `vboxmanage debugvm`)
- Directly from the system! (if we have permissions to do that)
  - `windd`, `fastdump`, dumpit, `memorize`, **`winpmem`**
  - Or we can hibernate the system (hiberfil.sys)
- Remotely
  - Encase Enterprise, Mandiant Intelligent Response, Access Data FTK
- Common issues
  - Unsupported OS (Linux, MacOS; 32bit/64bit)
  - Swap (portions of memory on drive)
  - Malware not running inside a virtual machine

Memory Acquisition (2)

- **Local memory acquisition** notes
  - Unless you have plenty of money, try to get root/admin access to the host
  - Better to acquire to external storage (USB, network)
  - The lower tool's memory footprint, the better
  - If you run malware in VM, better have less RAM
    - Faster analysis
    - .. And configure no swap for the system too
    - However: malware can check for the available memory

Memory Acquisition (3)

- **Remote memory acquisition**
  - Very useful for fast Incident Response
  - Requires enterprise licenses for the commercial tools
  - Acquisition is done over network
  - Agents already in memory, no extra memory demands
  - Modern EDR/XDR solutions support this too
- Open-source alternative?
  - GRR (Google Rapid Response)
    - Still in development, primarily Incident Response tool
    - Allows remote memory acquisition

Phase #2

MEMORY ANALYSIS

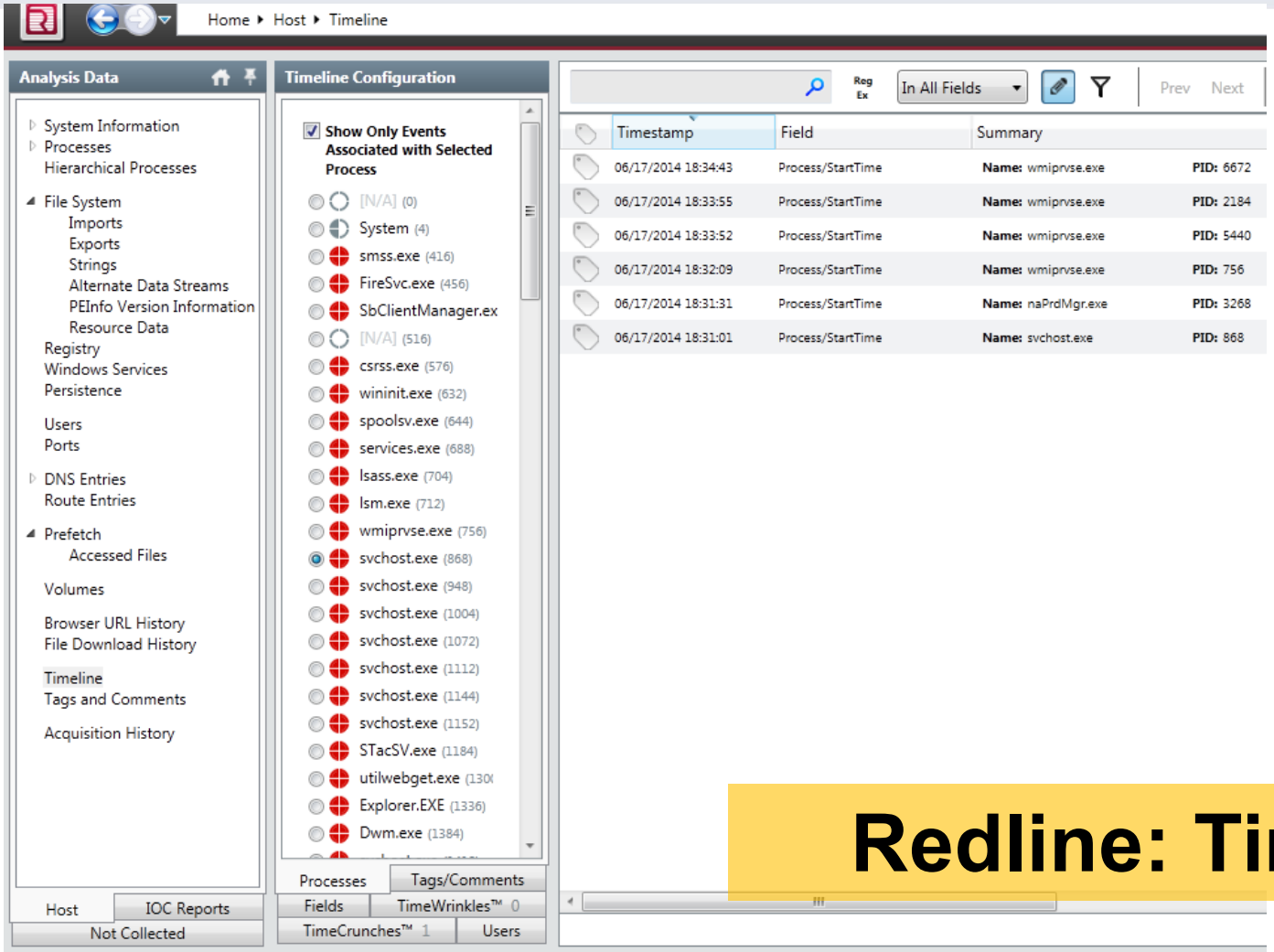
Memory Analysis Tools

- FireEye Redline
  - Free, available for Windows
- HBGary/GoSecure Responder Pro
  - Community Edition used to be available
- **Volatility Framework**
  - Open source, no GUI

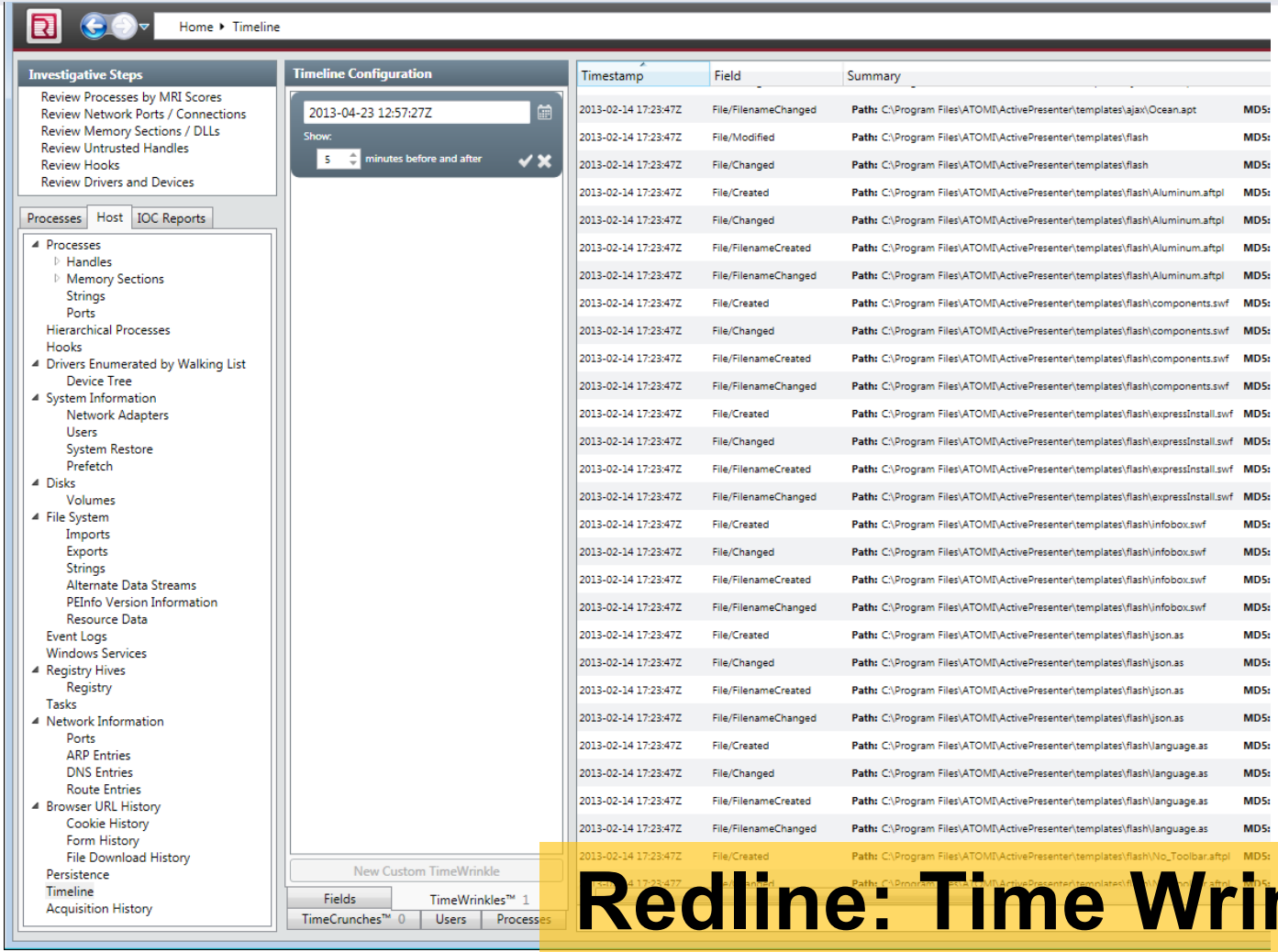
FireEye Redline

- Free tool for Incident Response
  - Not open-source, though
  - .NET executable (runs only under Windows)
  - Support OS X and Linux artifacts too
- Nice and simple user interface
  - Very nice analysis workflow
  - Perfect for searching for string information
  - Rates the level of suspiciousness over processes
- Sad things
  - Memory analysis not reliable, process rating as well


Redline®
Collect Data
Create a Standard Collector >
Create a Comprehensive Collector >
Create an IOC Search Collector >
Analyze Data
From a Saved Memory File >
Open Previous Analysis >
Recent Analysis Sessions
AnalysisSession4.mans >
AnalysisSession3.mans >
AnalysisSession2.mans >
AnalysisSession1.mans >
Redline: Start



Redline: Timeline



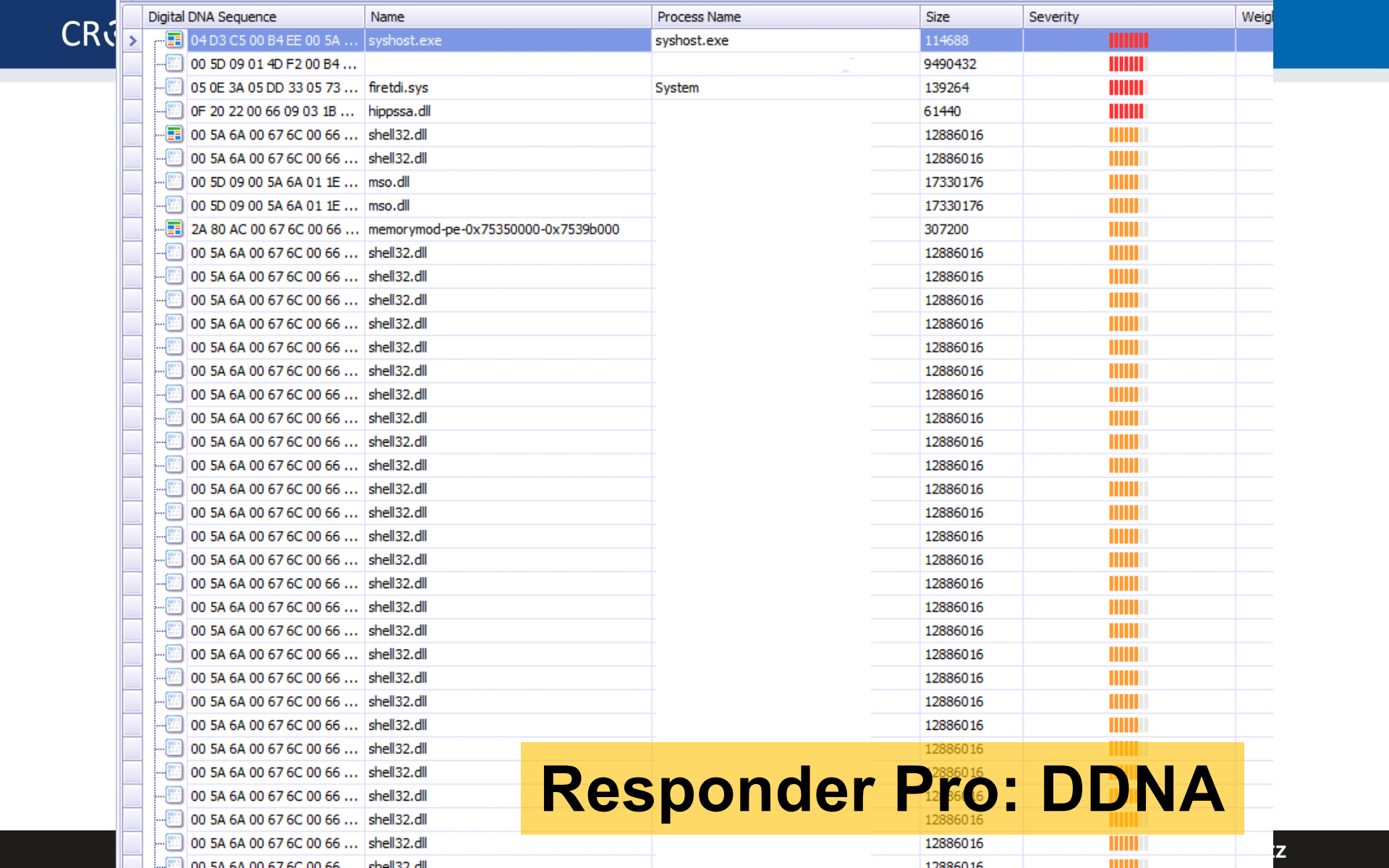
Redline: Time Wrinkles

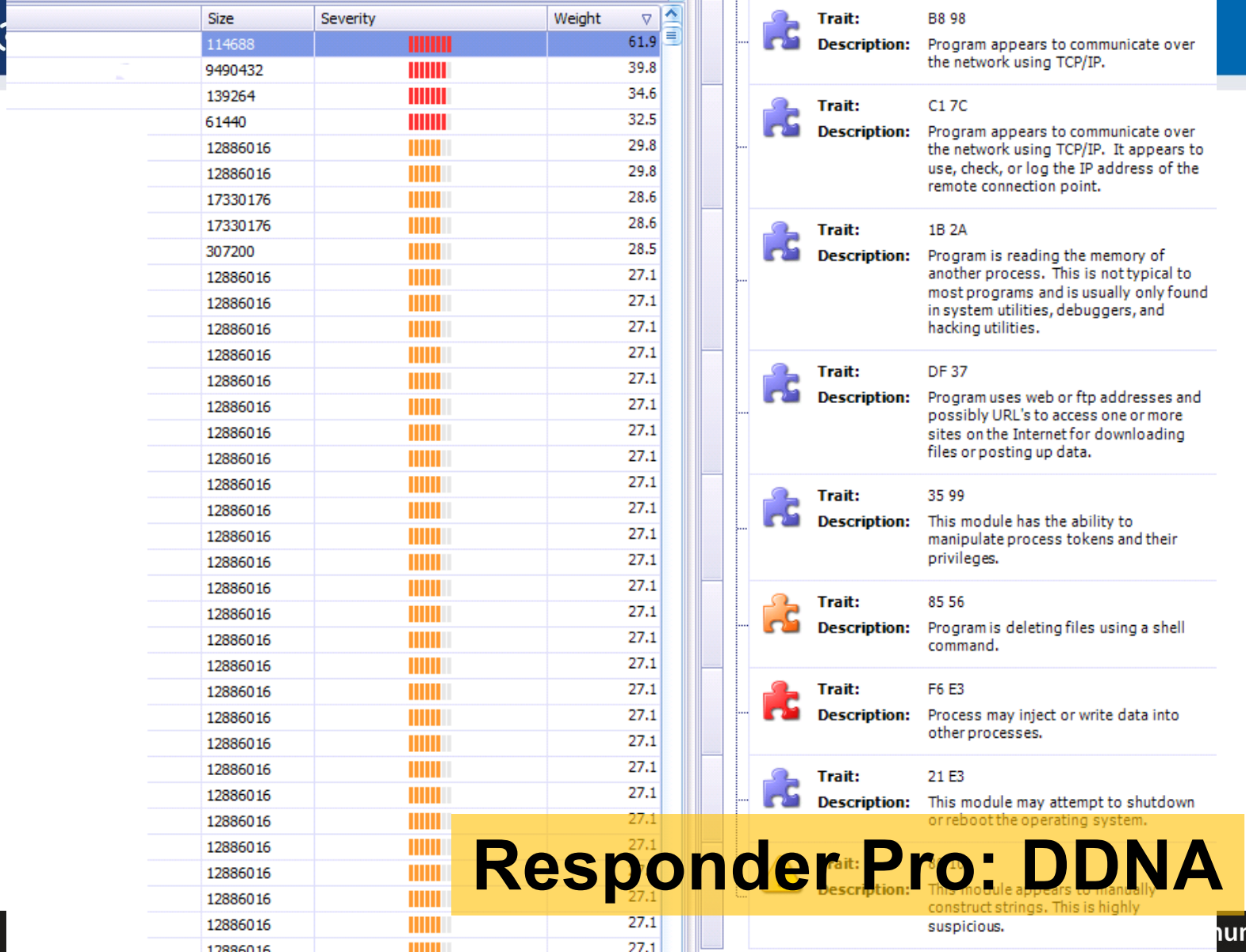
HBGary Responder (Pro/CE)

- Professional Tool
  - Very expensive
  - Yet not very well maintained in the last few years
- Windows only
  - .NET written, supports only Windows images
- ‘Killer’ features
  - Digital DNA
    - automatic rating of suspicious processes
  - Visual ‘Canvas’ debugger
- Supports the analysis of (unpacked) binaries
- Replaced with CounterTack Responder Pro

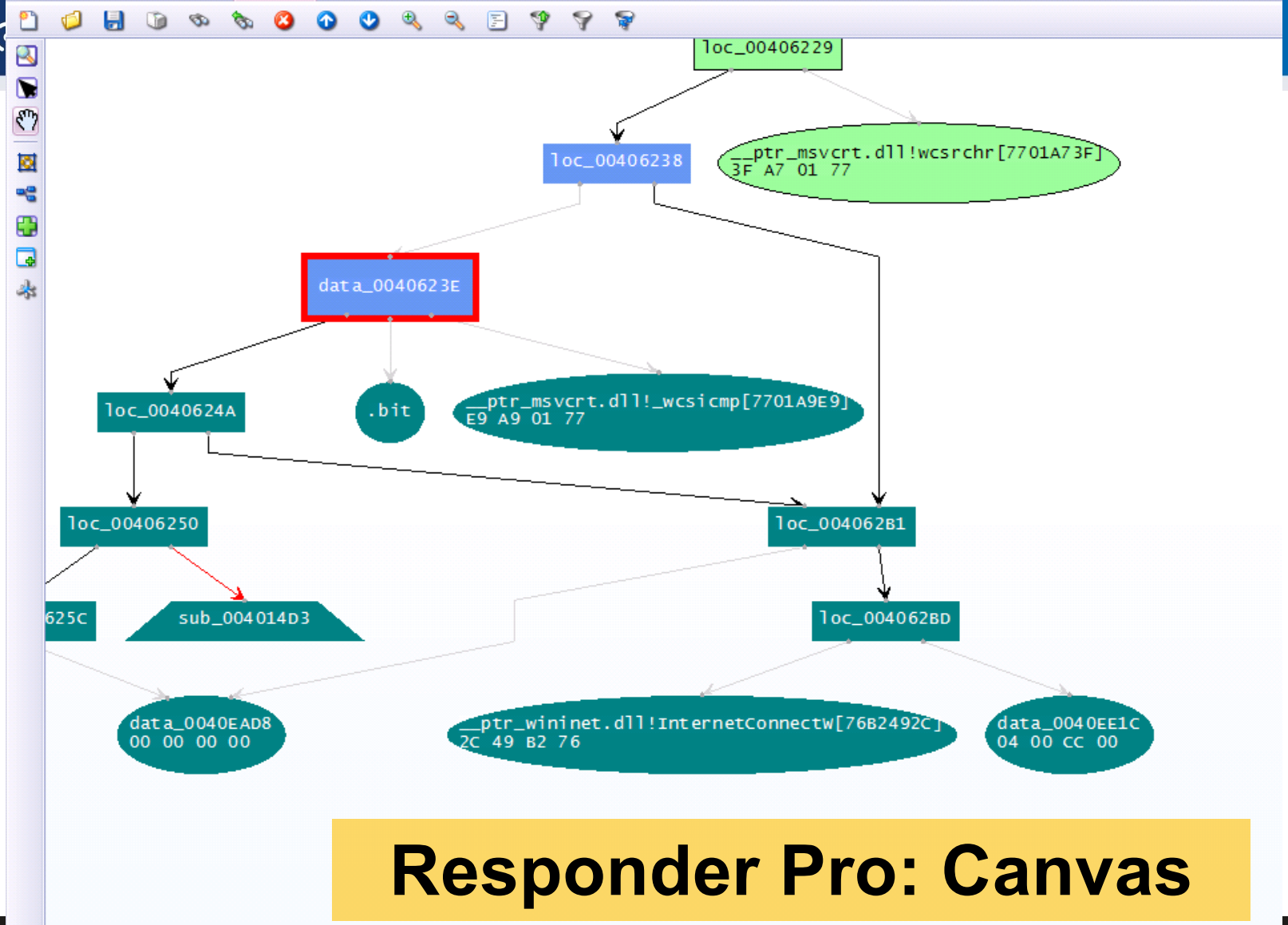
HBGary Responder Pro -- DDNA

- Examples of the ‘reasoning’ behind DDNA
  - Does the process communicate over TCP/IP?
  - Does it manipulate with registry?
  - Did the analysis reveal any known bad stuff (strings, IPs, mutexes?)
  - Does the process access any other process in the system?
  - Does it access some system-critical process?
  - Did the analysis find any evidence of obfuscation?


Digital DNA Sequence
Name
Process Name
Size
Severity
04 D3 C5 00 B4 EE 00 5A ...
syshost.exe
syshost.exe
114688
00 5D 09 01 4D F2 00 B4 ...
9490432
05 0E 3A 05 DD 33 05 73 ...
firetdi.sys
System
139264
0F 20 22 00 66 09 03 1B ...
hippssa.dll
61440
00 5A 6A 00 67 6C 00 66 ...
shell32.dll
12886016
00 5D 09 00 5A 6A 01 1E ...
mso.dll
17330176
2A 80 AC 00 67 6C 00 66 ...
memorymod-pe-0x75350000-0x7539b000
307200
Responder Pro: DDNA



Responder Pro: DDNA



Responder Pro: Canvas

Volatility Framework

- Open-source tool
  - GPL licensed
- Written in Python
  - Available for variety of platforms (Linux, Windows, Mac OS)
  - Can be automated; many contributed plugins
- Supports analysis of memory dumps from various OSs
  - Windows, Linux, MacOS, Android
  - Both 32-bit and 64-bit versions
- Command-line driven
- Two (experimental) web GUIs

Google Rekall

- Another open source tool
- Supported by Google
  - Included as a part of GRR (Google Rapid Response) agent
- Originally based on the code of Volatility
  - Shared commands
  - Different architectural concepts
- Proof-of-concept GUI
  - Better workflows
- **Discontinued since 2020**

Additional Important Tools

- **Strings**
  - Both *nix and Windows
  - Extracts strings information from the file
  - Can be used in cooperation with Volatility/Rekall
  - Beware of text encoding! (ascii, utf-8, …)
- **Foremost**
  - Forensic tool
  - Can extract various data files from an image (or process)
    - Images, executables, documents, …

Forensic analysis of RAM?

- Are there any benefits?
- Collecting forensic evidence
  - Executable images
  - PDF/Doc documents
    - Possible origin of the infection?
  - Images
  - URLs
- Getting approximate timeline
  - Works better on servers (always online, higher uptime, way more RAM)

What to search for in Operating System?

- Command & Control (C2) communication
- Hidden processes
- Process/DLL injection evidence
- Non-standard/infamous binaries/mutexes
- Open sockets and files
- Registry records
- Command-line history
- Encryption keys!

Known Bad Mutexes

- *Conficker*: `.*-7 and .*-99`
- *Sality.AA*: `Op1mutx9`
- *Flystud.??*: `Hacker.com.cn_MUTEX`
- *NetSky*: `'D'r'o'p'p'e'd'S'k'y'N'e't'`
- *Sality.W*: `u_joker_v3.06`
- *Poison Ivy*: `)!VoqA.I4` (and 10 thousand others)
- *Koobface*: `35fsdfsdfgfd5339`

Known Good Processes/Locations

Process Name	Expected Path
lsass.exe	\windows\system32
services.exe	\windows\system32
csrss.exe	\windows\system32
explorer.exe	\windows
spoolsv.exe	\windows\system32
smss.exe	\windows\system32
svchost.exe	\windows\system32
iexplore.exe	\program files \program files (x86)
winlogon.exe	\windows\system32

Operational Security (OpSec)

- Basics of OpSec
  - "Think before you act" mentality
  - Limited information sharing
- Specifics of memory analysis
  - You can often upload acquired executables to VirusTotal
    - MD5/SHA1 of the dump is different from the executable
    - This doesn't apply for documents/HTML pages!
  - **However, incomplete binaries still can infect your system!**
    - Running in VM or other OS is recommended

Recommended Analysis Process

- **Use Internet!** (Google, VirusTotal, …)
- **Make notes!**
  - What OS is being analyzed? (imageinfo)
  - Network connections? (+ whois records, …)
  - Processes (hidden, odd, non-standard; timestamps, …)
  - Mutexes (+ files open)
  - Dump processes when needed (OpSec!)
  - Strings (URIs, C-like strings %s %d, domains, …)
- **Summarize your findings in final report**

More information

- Web pages of this course
  - https://dior.ics.muni.cz/~valor/pv204
- **Additional resources**
  - Public memory images for analysis
  - Reverse Engineering for Beginners (amazing PDF doc)
  - REMnux: All you need to start with RE
  - ContagioDump blog (for additional malware samples)
  - Malware Traffic Analysis (both traffic & samples)

Thank you for your attention.

ANSWERS & QUESTIONS

LAB

Lab Requirements

- Oracle VM VirtualBox
  - And enough space on your hard drive (12 GB at least)
- **Volatility Framework**
  - **Version 2** (version 3 is available in the VM too)
- Unix tools
  - `strings`, `foremost`
- Your favorite text editor for notes
- Voluntary:
  - Javascript/PDF analysis tools

Recommended Analysis Process

- **Use Internet!** (Google, VirusTotal, …)
- **Make notes!**
  - What OS is being analyzed?
  - Network connections? (+ whois records, …)
  - Processes (hidden, odd, non-standard; timestamps, …)
  - Mutexes (+ files open)
  - Strings (URIs, C-like strings %s %d, domains, …)
  - …
- **Summarize your findings in final report**

Volatility2 Framework – cheat sheet

- `psxview` (search for hidden processes)
- `apihooks`
- `driverscan`
- `ssdt` / `driverirp` / `idt`
- `connections` / `connscan` (WinXP, active network connections)
- `netscan` (Win7, opened network sockets and connections)
- `pslist` / `psscan` (process listing from WinAPI vs. EPROCESS blocks)
- `malfind` / `ldrmodules` (code injection + dump / DLL detection)
- `hivelist` (registry lookup and parsing) / `hashdump`
- `handles` / `dlllist` / `filescan` (filelist / DLL files / FILE_OBJECT handles)
- `cmdscan` / `consoles` (`cmd.exe` history / console buffer)
- `shimcache` (application compatibility info)
- `memdump` / `procmemdump` / `procexedump`

Analysis: xp-infected.vmem

- Recommended tools
  - Volatility, Rekall (or Redline)
- Objectives:
  - Get familiar with memory of your first infected system

Analysis: win7_x64.vmem

- Recommended tools
  - Volatility, Rekall (or Redline)
- Objectives:
  - Get familiar with memory of Win7 x64 system
  - Can you see any differences from the previous sample?

Analysis: zeus.vmem

- Recommended tools
  - Volatility, Rekall
- Objectives:
  - Find suspicious network connections
  - Find process responsible for the network activity
  - Can you figure out what infections this

Analysis: zeus2x4.vmem

- Recommended tools
  - Volatility, Rekall
- Objectives:
  - Find suspicious network connections
  - Find process responsible for the network activity
  - Can you figure out what infections this
  - Can you dump the virus configuration?

Analysis: bob.vmem

- Recommended tools
  - Volatility, Rekall, Foremost, Strings
- Objectives:
  - Find suspicious network connections
  - Find process responsible for the network activity
  - Can you figure out what caused the infection?
  - Can you dump the initial source vector?
  - What known vulnerability (CVE) has been exploited?

More information

- Web pages of this course
  - https://dior.ics.muni.cz/~valor/pv204
- **Additional resources**
  - Public memory images for analysis
  - Reverse Engineering for Beginners (amazing PDF doc)
  - REMnux: All you need to start with RE
  - ContagioDump blog (for additional malware samples)
  - Malware Traffic Analysis (both traffic & samples)

Thank you for your attention.

Answers & Questions