

PV204 Security technologies



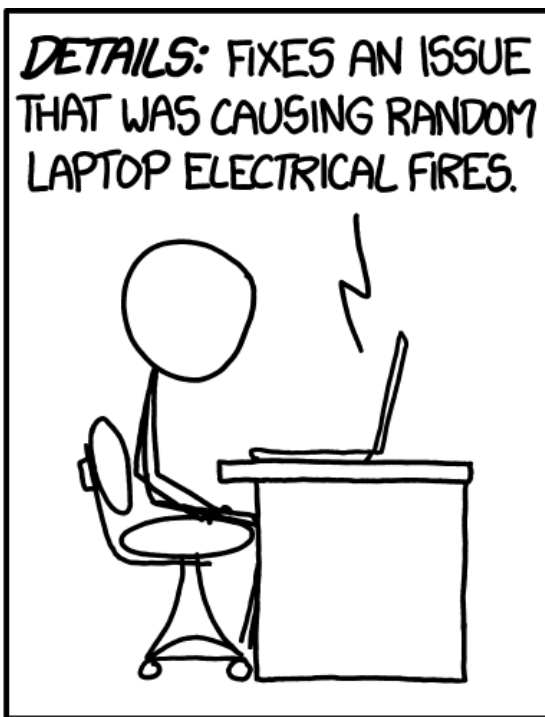
In-Memory Malware Analysis

Václav Lorenc

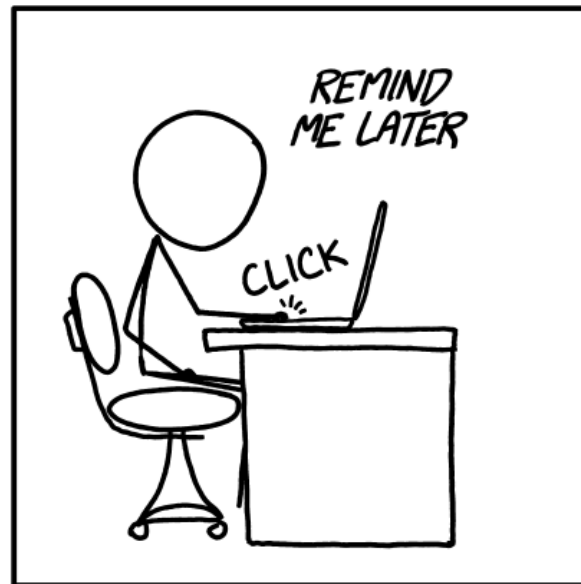
Principal Security Engineer, Here Technologies

CRCS

Centre for Research on
Cryptography and Security



(THIS UPDATE WILL REQUIRE RESTARTING YOUR COMPUTER.)

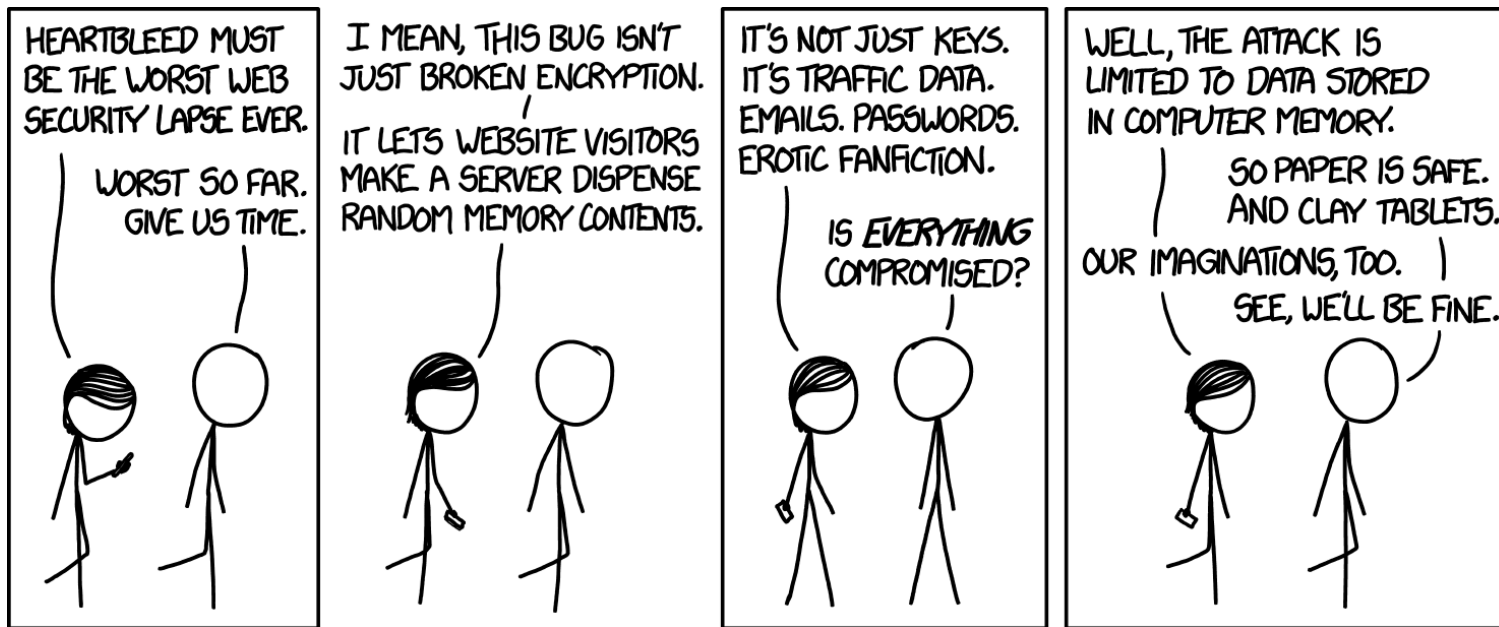


Agenda

- 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



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

'cause reverse engineering ninjas are busy

MEMORY ANALYSIS



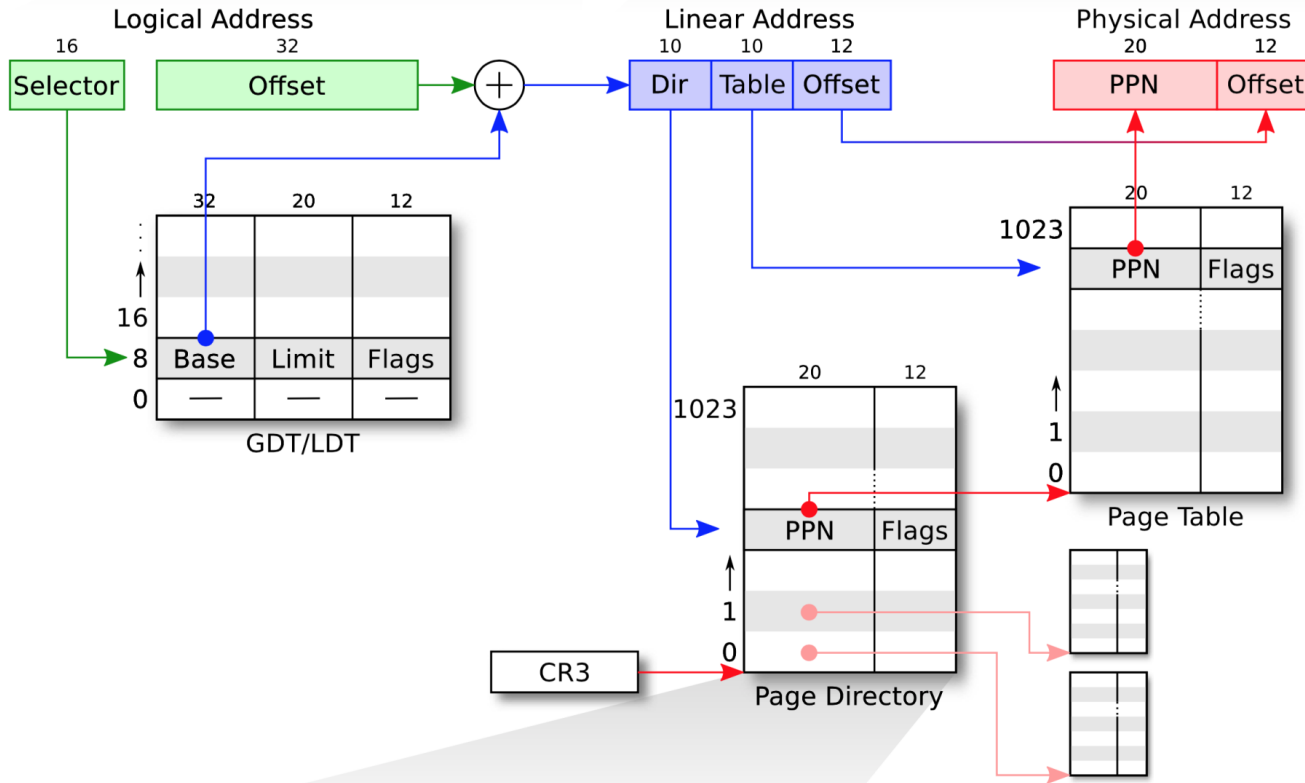
x86/x64 Memory organization

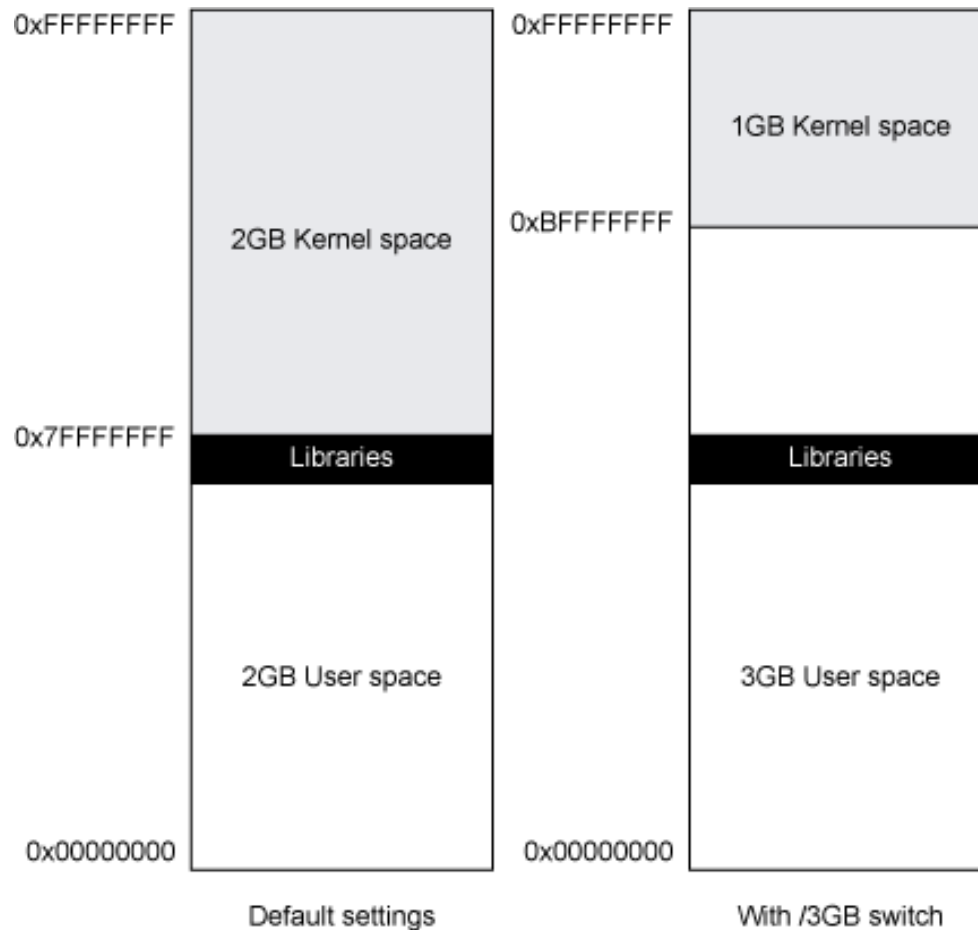
- Physical memory
 - RAM; what we really have installed
- Virtual memory
 - Separation of logical process memory from the physical
 - Logical address space > physical (e.g., swap)
 - Address space shared by several processes, yet separated
- Paging vs. Segmentation
 - Possible memory organization approaches

Segmentation

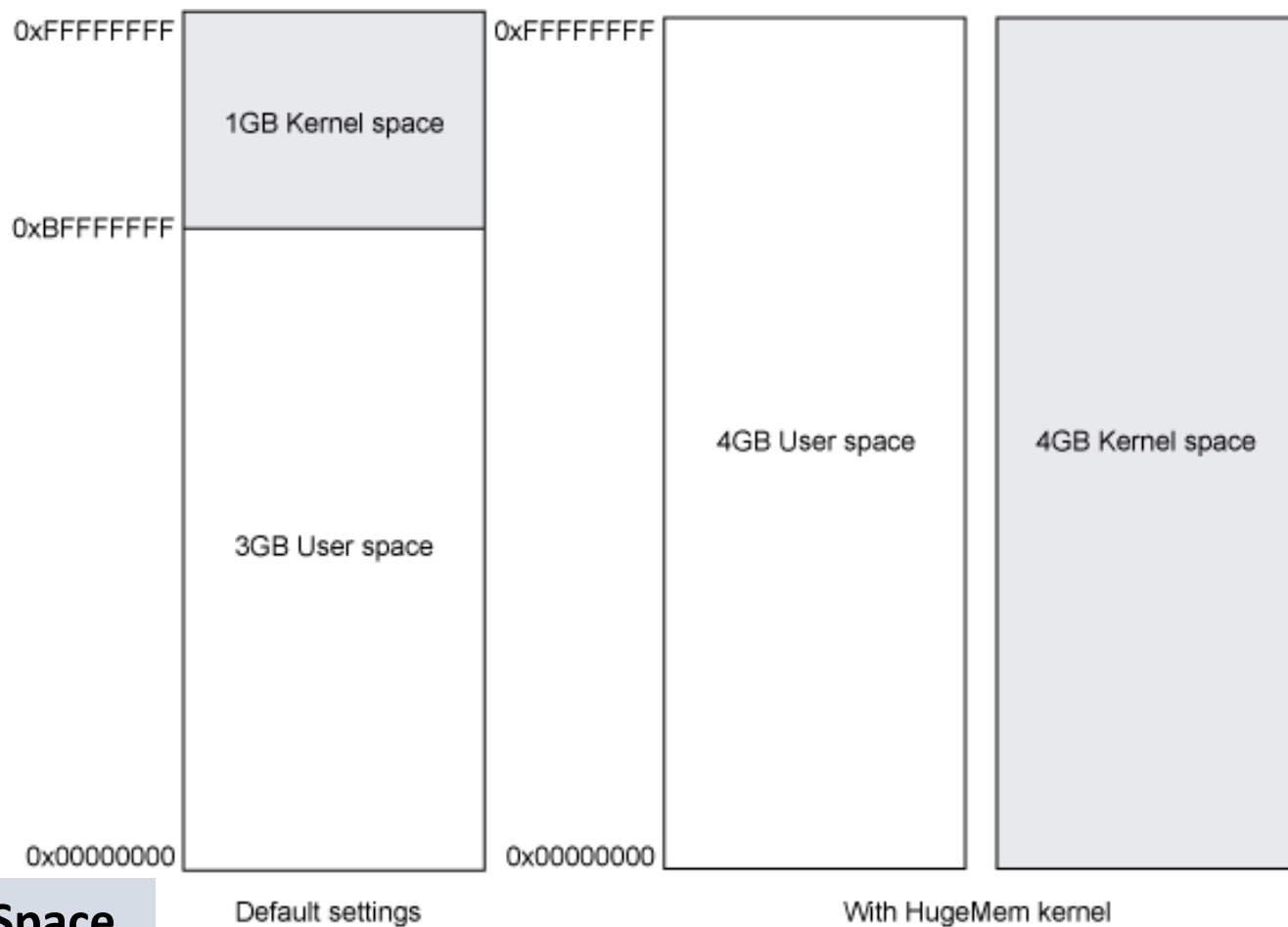
Paging

Physical Address





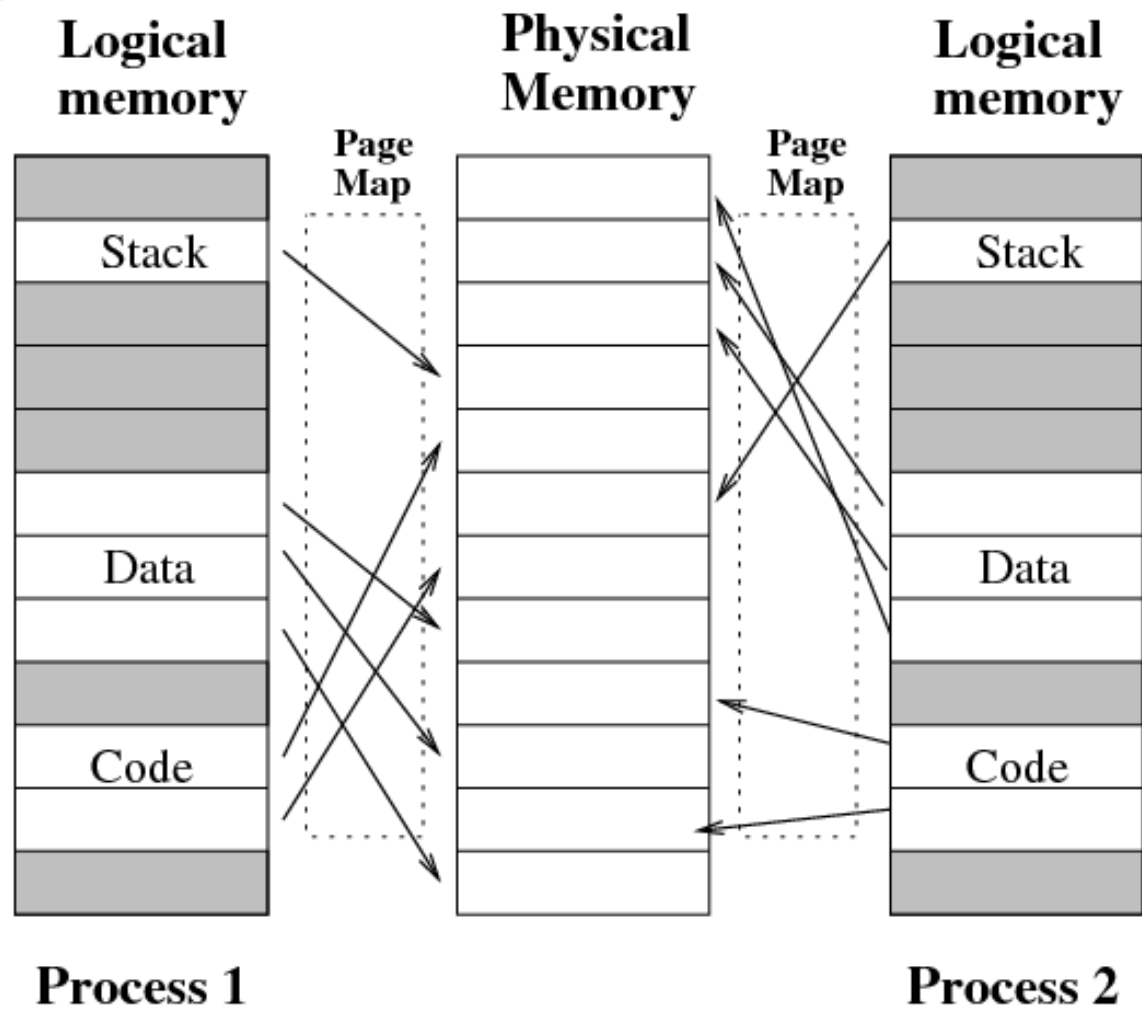
Win32 Address Space



Linux Address Space

Default settings

With HugeMem kernel

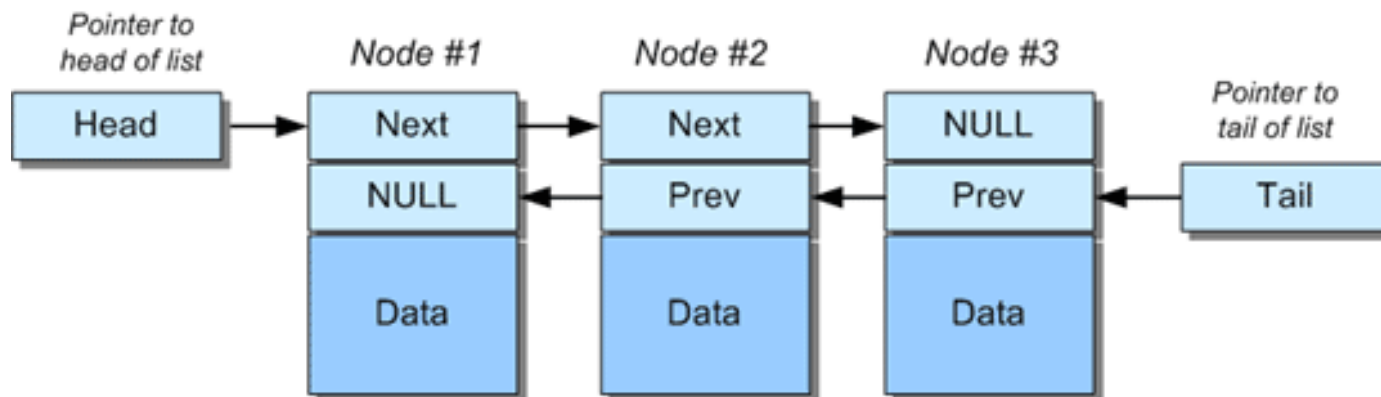




Operating System Data Structures

- 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

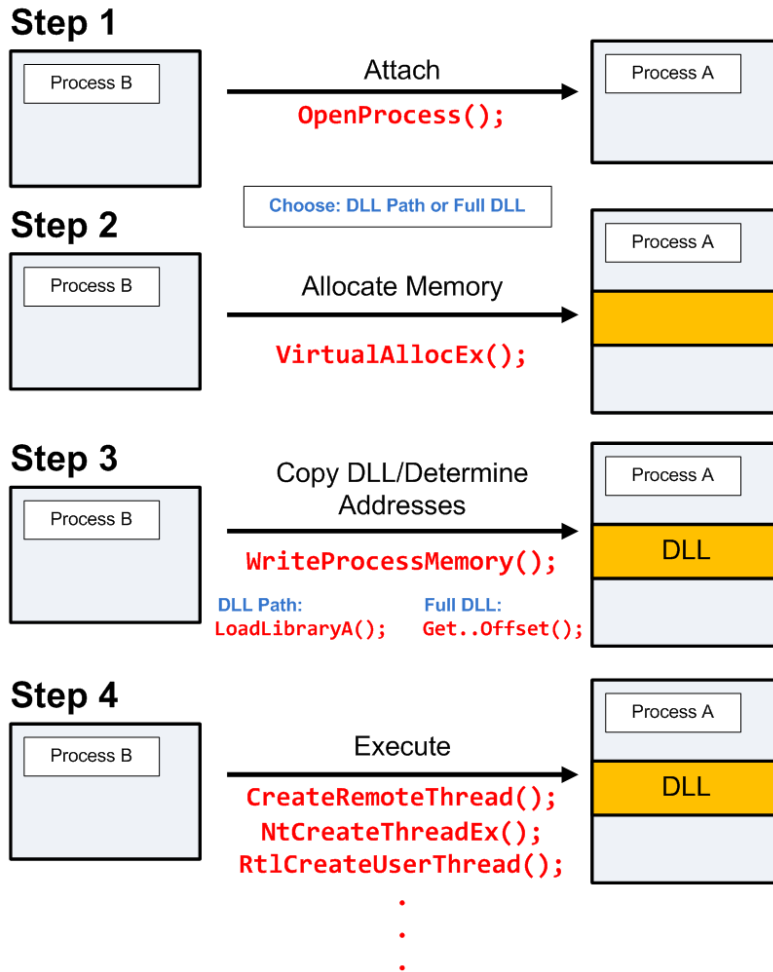
Memory Pages

- 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)

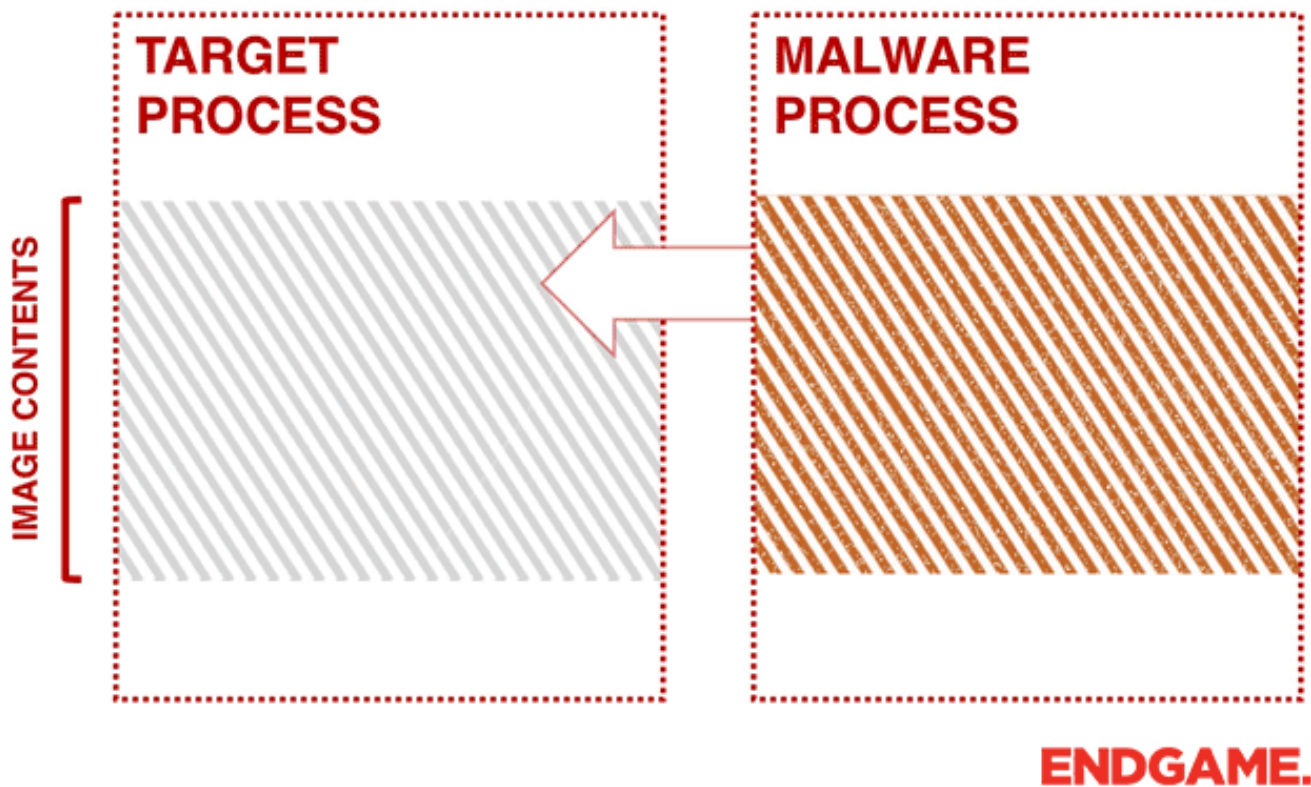
DLL/Process Injection

So that Internet Explorer behaves like a malicious process...

Overview

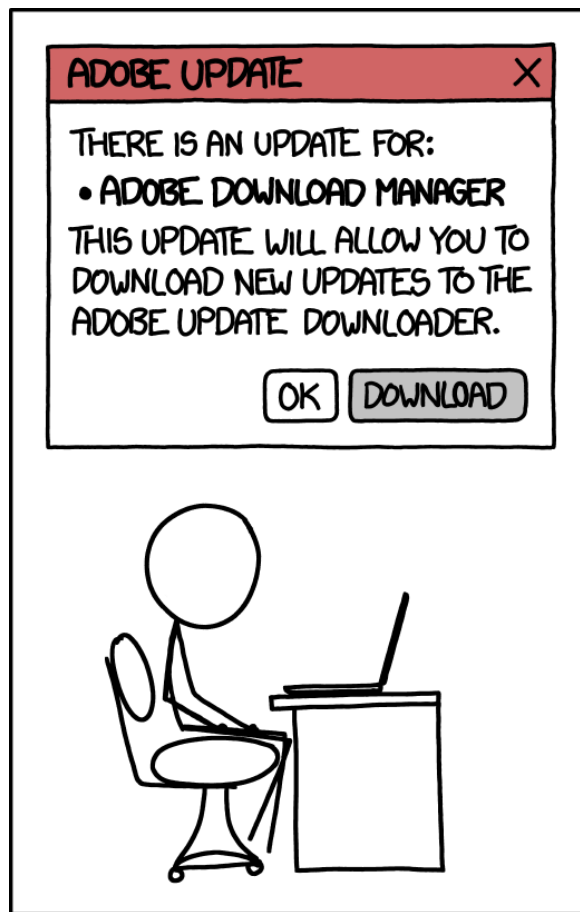


PROCESS HOLLOWING



PRACTICAL

AND NOW SOMETHING COMPLETELY...



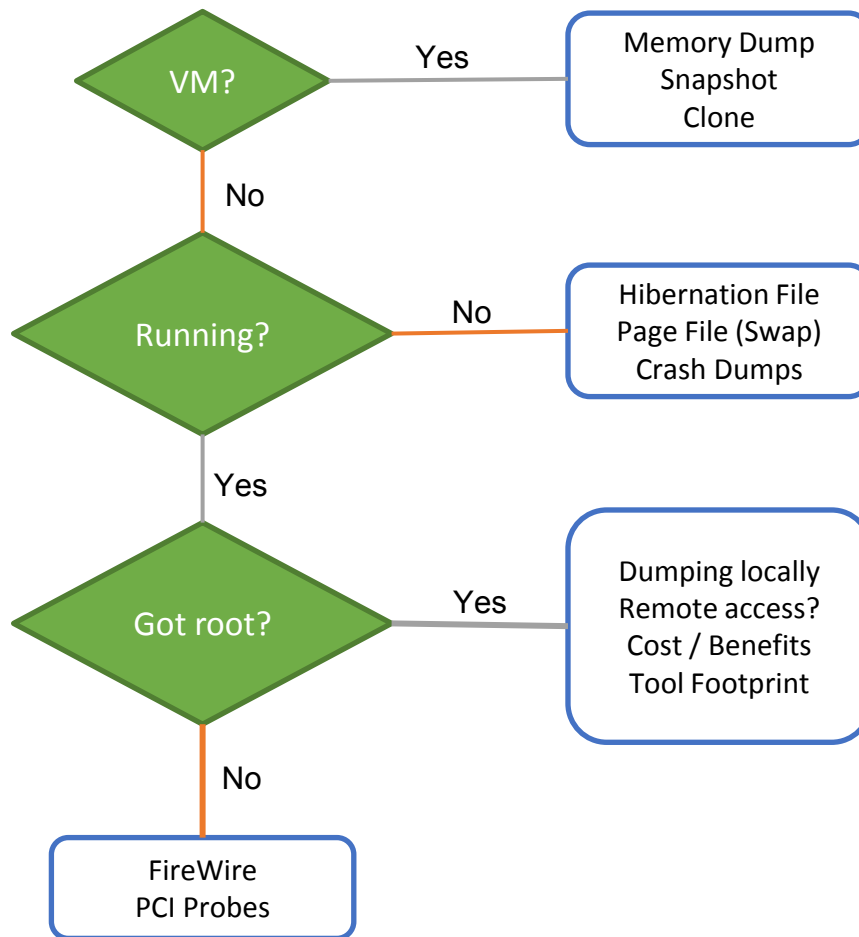
Phase #1

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 than available RAM
- Hibernation file
- Memory crash dumps
 - Limited analysis options

Memory Acquisition



Memory Acquisition

- **Virtual Machines**
 - VMWare, VirtualBox, ...
 - VirtualBox `-dbg -startvm "MalwareVM"` (and `.pgmphysfile` 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

Home ▶ Host ▶ Timeline

Analysis Data

- System Information
- Processes
 - Hierarchical Processes
- File System
 - Imports
 - Exports
 - Strings
 - Alternate Data Streams
 - PEInfo Version Information
 - Resource Data
- Registry
 - Windows Services
 - Persistence
- Users
 - Ports
- DNS Entries
 - Route Entries
- Prefetch
 - Accessed Files
- Volumes
- Browser URL History
- File Download History
- Timeline
- Tags and Comments
- Acquisition History

Host IOC Reports

Not Collected

Timeline Configuration

Show Only Events Associated with Selected Process

- [N/A] (0)
- System (4)
- smss.exe (416)
- FireSvc.exe (456)
- SbClientManager.ex
- [N/A] (516)
- csrss.exe (576)
- wininit.exe (632)
- spoolsv.exe (644)
- services.exe (688)
- lsass.exe (704)
- lsm.exe (712)
- wmiprvse.exe (756)
- svchost.exe (868)
- svchost.exe (948)
- svchost.exe (1004)
- svchost.exe (1072)
- svchost.exe (1112)
- svchost.exe (1144)
- svchost.exe (1152)
- STacSV.exe (1184)
- utilwebget.exe (130)
- Explorer.EXE (1336)
- Dwm.exe (1384)

Processes Tags/Comments

Fields TimeWrinkles™ 0

TimeCrunches™ 1 Users

Reg Ex In All Fields Prev Next

Timestamp	Field	Summary
06/17/2014 18:34:43	Process/StartTime	Name: wmiprvse.exe PID: 6672
06/17/2014 18:33:55	Process/StartTime	Name: wmiprvse.exe PID: 2184
06/17/2014 18:33:52	Process/StartTime	Name: wmiprvse.exe PID: 5440
06/17/2014 18:32:09	Process/StartTime	Name: wmiprvse.exe PID: 756
06/17/2014 18:31:31	Process/StartTime	Name: naPrdMgr.exe PID: 3268
06/17/2014 18:31:01	Process/StartTime	Name: svchost.exe PID: 868

Redline: Timeline

Home ▶ Timeline

Investigative Steps

- Review Processes by MRI Scores
- Review Network Ports / Connections
- Review Memory Sections / DLLs
- Review Untrusted Handles
- Review Hooks
- Review Drivers and Devices

Processes Host IOC Reports

- Processes
 - Handles
 - Memory Sections
 - Strings
 - Ports
- Hierarchical Processes
- Hooks
- Drivers Enumerated by Walking List
 - Device Tree
- System Information
 - Network Adapters
 - Users
 - System Restore
 - Prefetch
- Disks
 - Volumes
- File System
 - Imports
 - Exports
 - Strings
 - Alternate Data Streams
 - PInfo Version Information
 - Resource Data
- Event Logs
- Windows Services
- Registry Hives
 - Registry
 - Tasks
- Network Information
 - Ports
 - ARP Entries
 - DNS Entries
 - Route Entries
- Browser URL History
 - Cookie History
 - Form History
 - File Download History
- Persistence
- Timeline
- Acquisition History

Timeline Configuration

2013-04-23 12:57:27Z

Show: 5 minutes before and after

Timestamp	Field	Summary
2013-02-14 17:23:47Z	File/FilenameChanged	Path: C:\Program Files\ATOM\ActivePresenter\templates\ajax\Ocean.appt MDS:
2013-02-14 17:23:47Z	File/Modified	Path: C:\Program Files\ATOM\ActivePresenter\templates\flash MDS:
2013-02-14 17:23:47Z	File/Changed	Path: C:\Program Files\ATOM\ActivePresenter\templates\flash MDS:
2013-02-14 17:23:47Z	File/Created	Path: C:\Program Files\ATOM\ActivePresenter\templates\flash\Aluminum.aftpl MDS:
2013-02-14 17:23:47Z	File/Changed	Path: C:\Program Files\ATOM\ActivePresenter\templates\flash\Aluminum.aftpl MDS:
2013-02-14 17:23:47Z	File/FilenameCreated	Path: C:\Program Files\ATOM\ActivePresenter\templates\flash\Aluminum.aftpl MDS:
2013-02-14 17:23:47Z	File/FilenameChanged	Path: C:\Program Files\ATOM\ActivePresenter\templates\flash\Aluminum.aftpl MDS:
2013-02-14 17:23:47Z	File/Created	Path: C:\Program Files\ATOM\ActivePresenter\templates\flash\components.swf MDS:
2013-02-14 17:23:47Z	File/Changed	Path: C:\Program Files\ATOM\ActivePresenter\templates\flash\components.swf MDS:
2013-02-14 17:23:47Z	File/FilenameCreated	Path: C:\Program Files\ATOM\ActivePresenter\templates\flash\components.swf MDS:
2013-02-14 17:23:47Z	File/FilenameChanged	Path: C:\Program Files\ATOM\ActivePresenter\templates\flash\components.swf MDS:
2013-02-14 17:23:47Z	File/Created	Path: C:\Program Files\ATOM\ActivePresenter\templates\flash\expressInstall.swf MDS:
2013-02-14 17:23:47Z	File/Changed	Path: C:\Program Files\ATOM\ActivePresenter\templates\flash\expressInstall.swf MDS:
2013-02-14 17:23:47Z	File/FilenameCreated	Path: C:\Program Files\ATOM\ActivePresenter\templates\flash\expressInstall.swf MDS:
2013-02-14 17:23:47Z	File/FilenameChanged	Path: C:\Program Files\ATOM\ActivePresenter\templates\flash\expressInstall.swf MDS:
2013-02-14 17:23:47Z	File/Created	Path: C:\Program Files\ATOM\ActivePresenter\templates\flash\infoBox.swf MDS:
2013-02-14 17:23:47Z	File/Changed	Path: C:\Program Files\ATOM\ActivePresenter\templates\flash\infoBox.swf MDS:
2013-02-14 17:23:47Z	File/FilenameCreated	Path: C:\Program Files\ATOM\ActivePresenter\templates\flash\infoBox.swf MDS:
2013-02-14 17:23:47Z	File/FilenameChanged	Path: C:\Program Files\ATOM\ActivePresenter\templates\flash\infoBox.swf MDS:
2013-02-14 17:23:47Z	File/Created	Path: C:\Program Files\ATOM\ActivePresenter\templates\flash\json.as MDS:
2013-02-14 17:23:47Z	File/Changed	Path: C:\Program Files\ATOM\ActivePresenter\templates\flash\json.as MDS:
2013-02-14 17:23:47Z	File/FilenameCreated	Path: C:\Program Files\ATOM\ActivePresenter\templates\flash\json.as MDS:
2013-02-14 17:23:47Z	File/FilenameChanged	Path: C:\Program Files\ATOM\ActivePresenter\templates\flash\json.as MDS:
2013-02-14 17:23:47Z	File/Created	Path: C:\Program Files\ATOM\ActivePresenter\templates\flash\language.as MDS:
2013-02-14 17:23:47Z	File/Changed	Path: C:\Program Files\ATOM\ActivePresenter\templates\flash\language.as MDS:
2013-02-14 17:23:47Z	File/FilenameCreated	Path: C:\Program Files\ATOM\ActivePresenter\templates\flash\language.as MDS:
2013-02-14 17:23:47Z	File/FilenameChanged	Path: C:\Program Files\ATOM\ActivePresenter\templates\flash\language.as MDS:

New Custom TimeWrinkle

Fields TimeWrinkles™ 1

TimeCrunches™ 0 Users Process

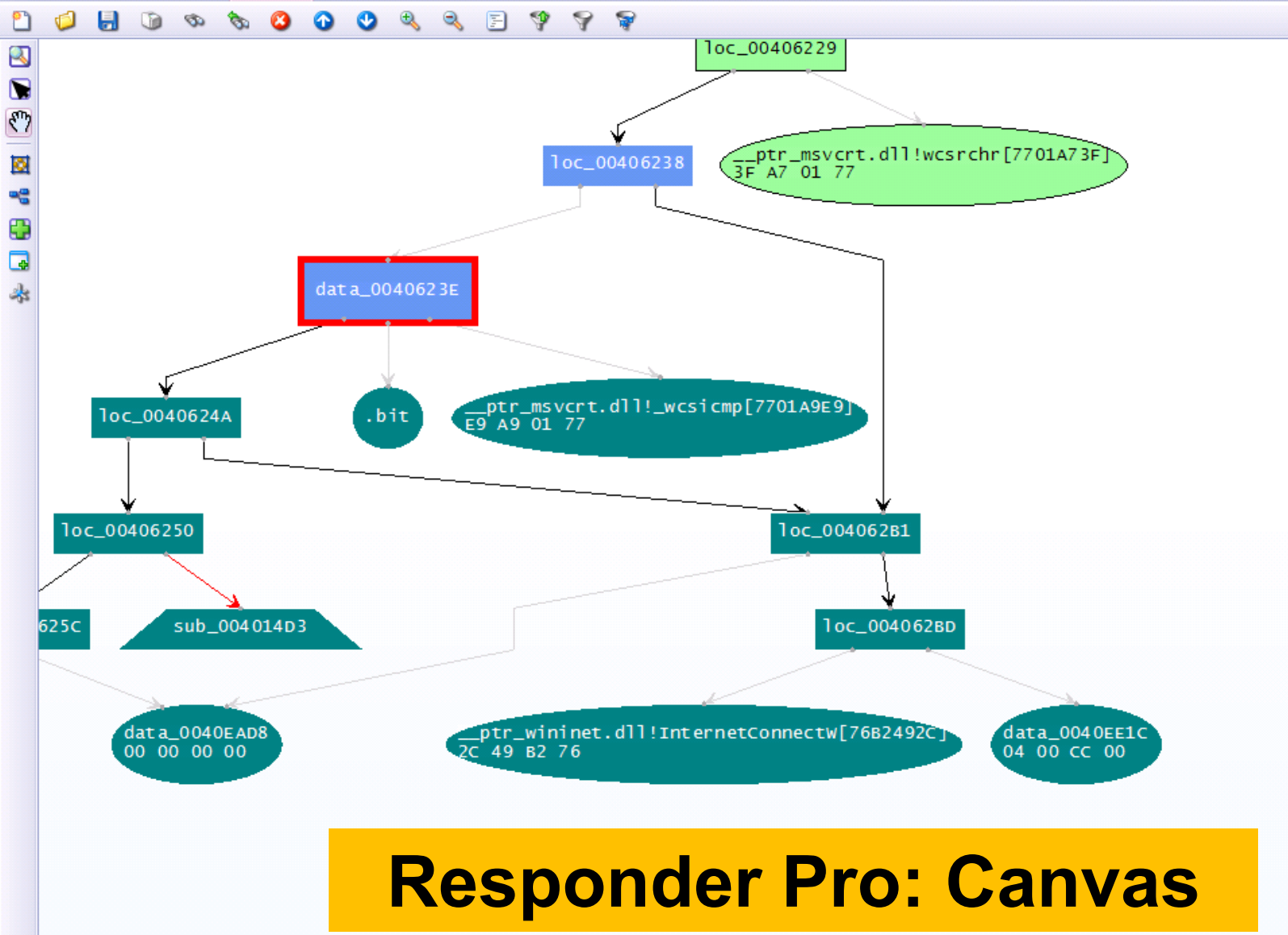
Redline: Time Wrinkles

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?



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 Rekal

- 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
<code>lsass.exe</code>	<code>\windows\system32</code>
<code>services.exe</code>	<code>\windows\system32</code>
<code>csrss.exe</code>	<code>\windows\system32</code>
<code>explorer.exe</code>	<code>\windows</code>
<code>spoolsv.exe</code>	<code>\windows\system32</code>
<code>smss.exe</code>	<code>\windows\system32</code>
<code>svchost.exe</code>	<code>\windows\system32</code>
<code>iexplore.exe</code>	<code>\program files</code> <code>\program files (x86)</code>
<code>winlogon.exe</code>	<code>\windows\system32</code>

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)

A hand is shown placing a white puzzle piece with the word 'ANSWER' on it into a larger, dark blue puzzle piece with the word 'QUESTION' on it. The background consists of many other white puzzle pieces.

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 (URLs, 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