

PV204 Security technologies



In-Memory Malware Analysis

Václav Lorenc
Incident Response Analyst, CIRT, Honeywell



Agenda

- Basic intro
 - No assembly required
 - No malware (de)obfuscation magic
- How does the 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)

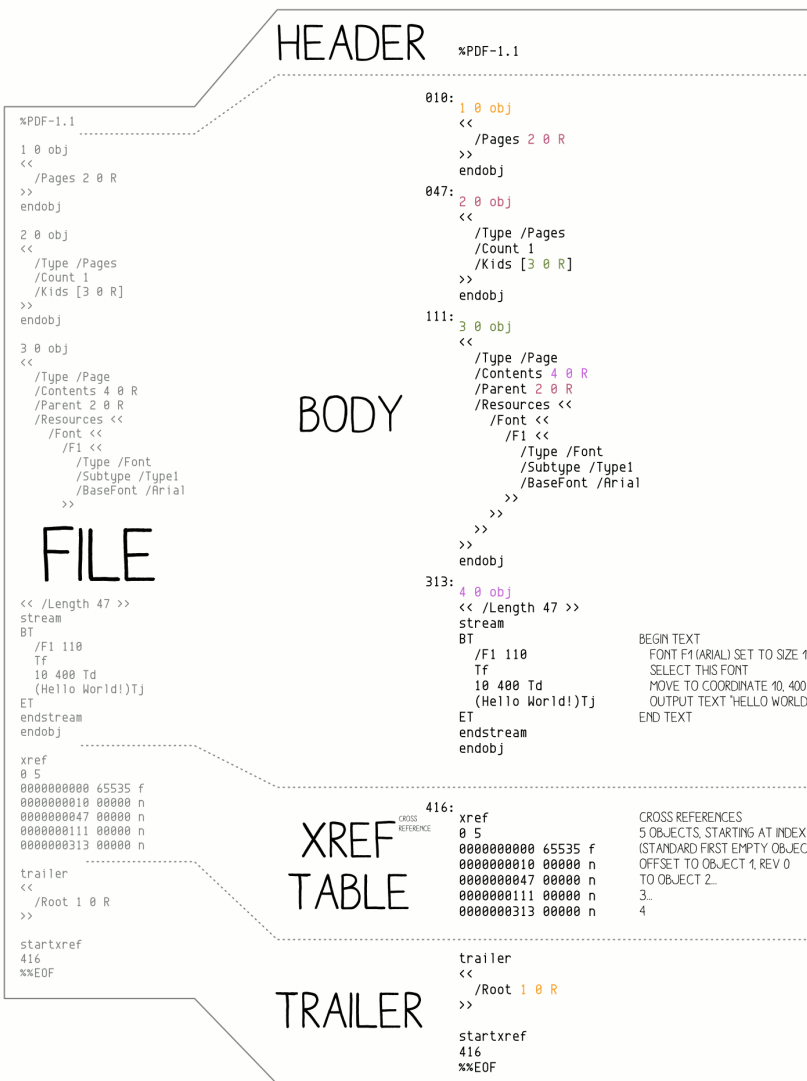
Why memory analysis?

- **It's fun!**
- Acquiring evidence for legal investigations
 - It used to be different in the past
- Incident response activities
 - Easy way how to learn more about the attackers
 - Malicious binary may only be present in memory
- Technical simplification of reverse engineering
 - No binary obfuscation present – the code has to run



Challenges in Reverse Engineering (RE)

- Assembly language (for multiple platforms)
 - Plus 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



BASICS

PDF IS TEXT BASED, WITH BINARY STREAMS

TYPES

0: STRING
EX: (Hello World!)

/NAME (IDENTIFIERS)
EX: /count 1

<<>: DICTIONARY
EX: <</key1 value1 /key2 value2>>

[]: ARRAY
EX: [0 1 2 3 4]

OBJECT REFERENCES

CONTENT IS STORED IN OBJECT
MOST CONTENT CAN BE INLINED OR REFERENCED IN A SEPARATE OBJECT

`/Key1 value` IS EQUIVALENT TO `/Key1 3 0 R`
[...]
`3 0 obj`
`value`
`endobj`

BINARY STREAMS

BINARY STREAMS ARE STORED IN SEPARATE OBJECTS LIKE THIS:

```

<object number> <object revision> obj
<< -STREAM/METADATA> >>
stream
-STREAM CONTENT-
endstream
endobj
    
```

TRIVIA

THE PDF WAS FIRST SPECIFIED BY ADOBE SYSTEMS IN 1993
INITIAL VERSIONS OF ADOBE ACROBAT WERE NOT FREE

FILE STRUCTURE

HEAD OF THE FILE

THE '%PDF-' SIGNATURE IDENTIFIES THE FORMAT AND REQUIRED VERSION

XREF

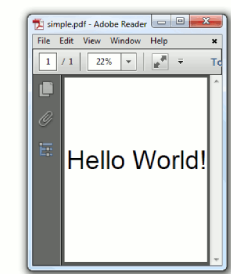
`xref`
-STARTING OBJECT+OBJECT COUNT+ FOLLOWED BY XREF ENTRIES:
F (OBJECT IN USE)
-OFFSET+0+GENERATIONS+ n
ELSE:
-NEXT_FREE_OBJECT+0+GENERATIONS+ f

END OF THE FILE

`startxref`
-XREF OFFSET IN DECODED STREAM+
`%%EOF`

PARSING

THE HEADER '%PDF-1.?' SIGNATURE IS CHECKED TO IDENTIFY THE FILE FORMAT
THE XREF IS LOCATED VIA THE `startxref` OFFSET
THE `xref` TABLE GIVES OFFSET OF EACH OBJECT
THE `trailer` IS PARSED
EACH OBJECT REFERENCE IS FOLLOWED, BUILDING THE DOCUMENT
PAGES ARE CREATED, TEXT IS RENDERED



MEMORY ANALYSIS...

'cause reverse engineering ninjas are busy

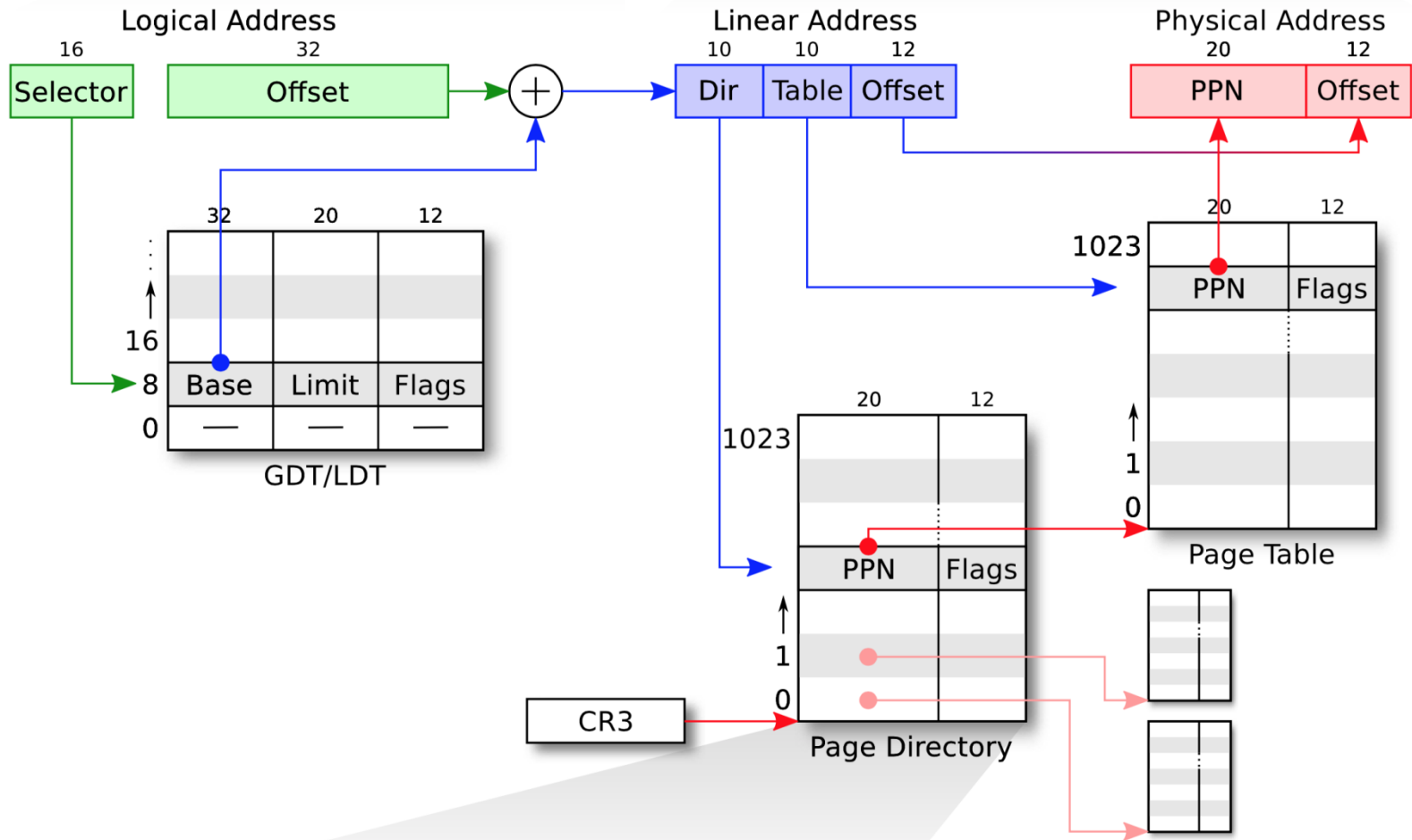
x86/x64 Memory organization

- 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

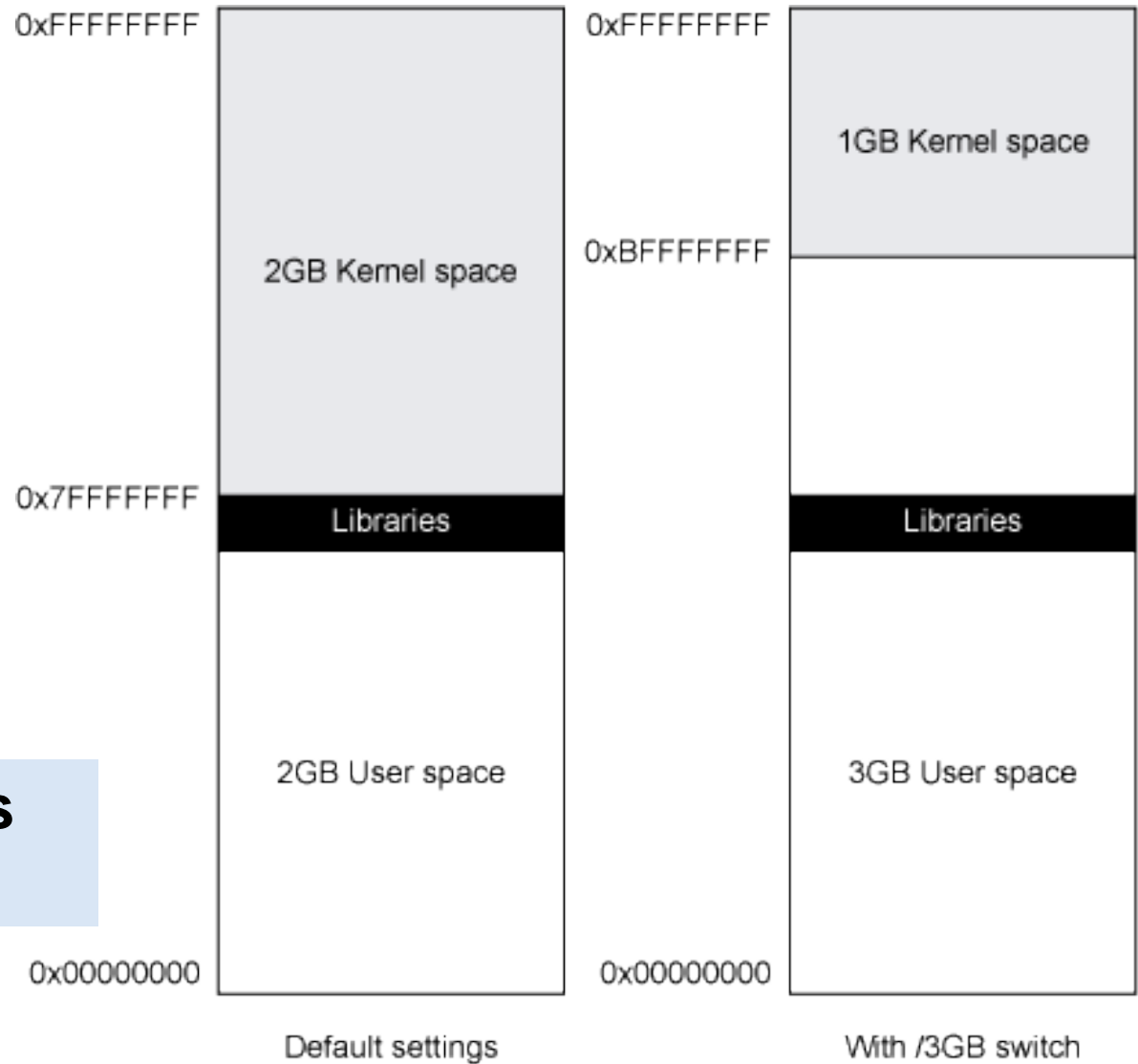
Segmentation

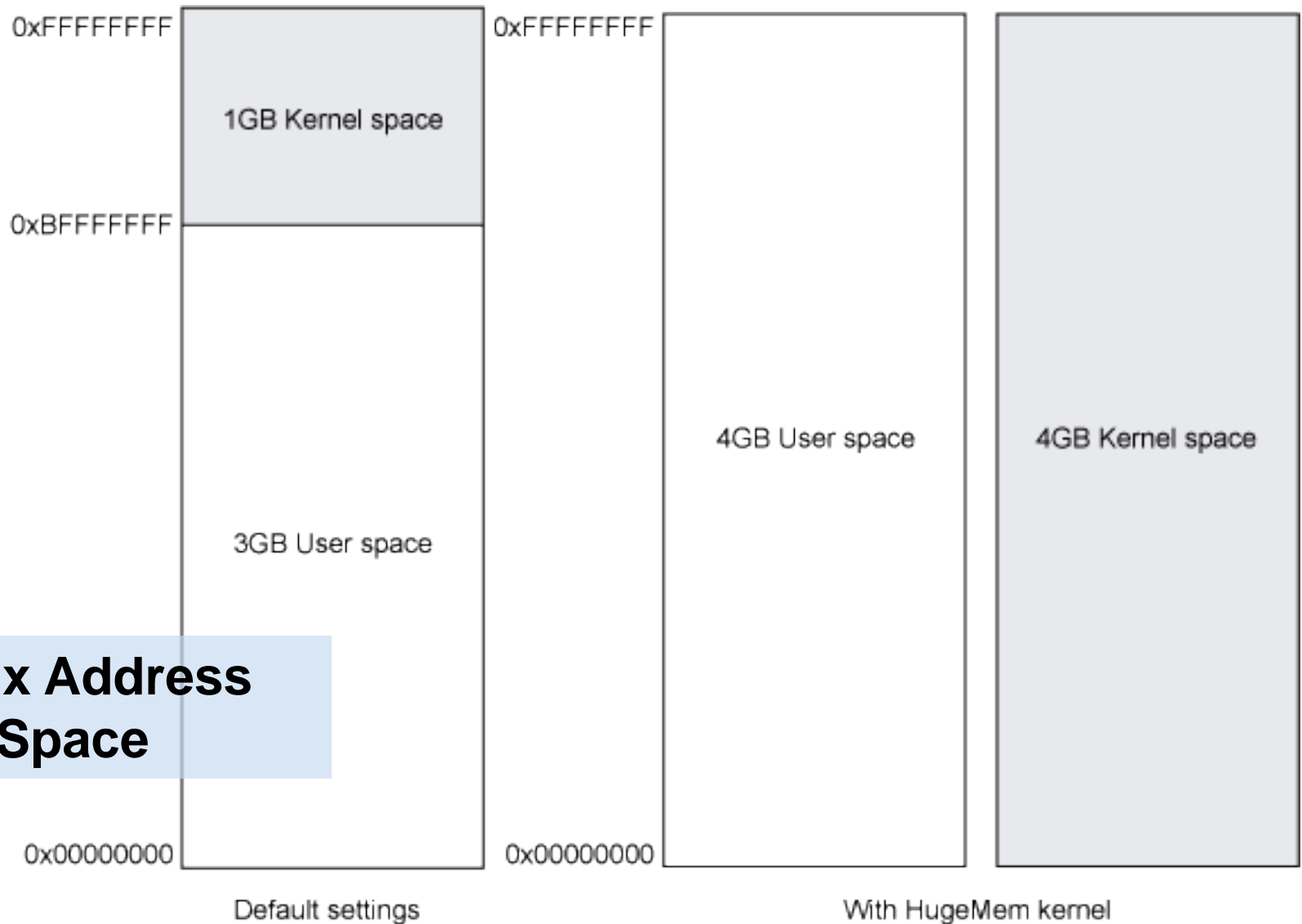
Paging

Physical Address



Win32 Address Space

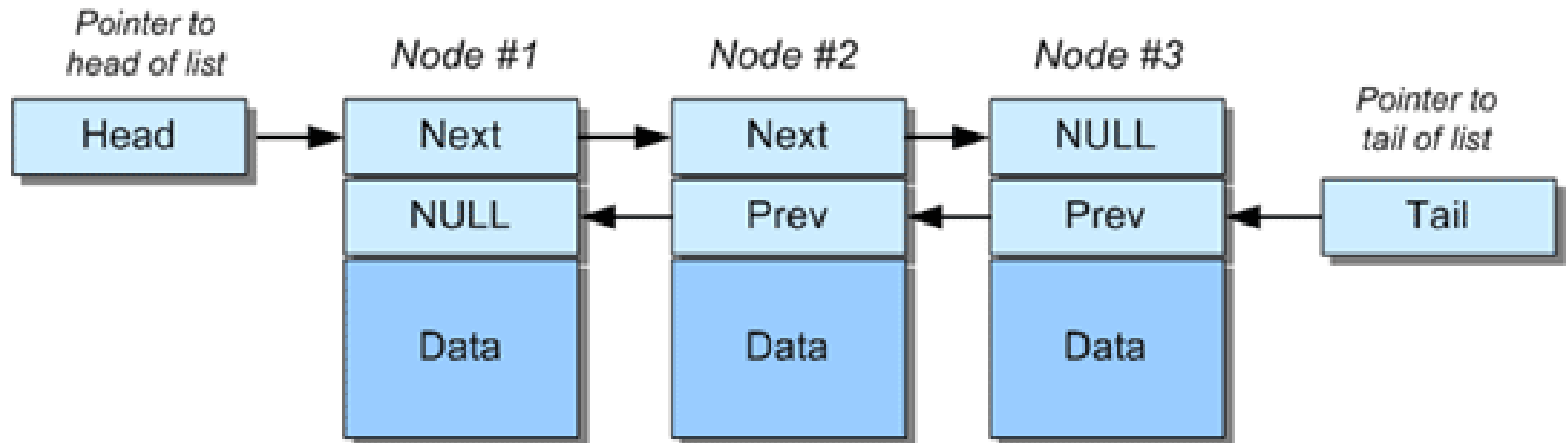




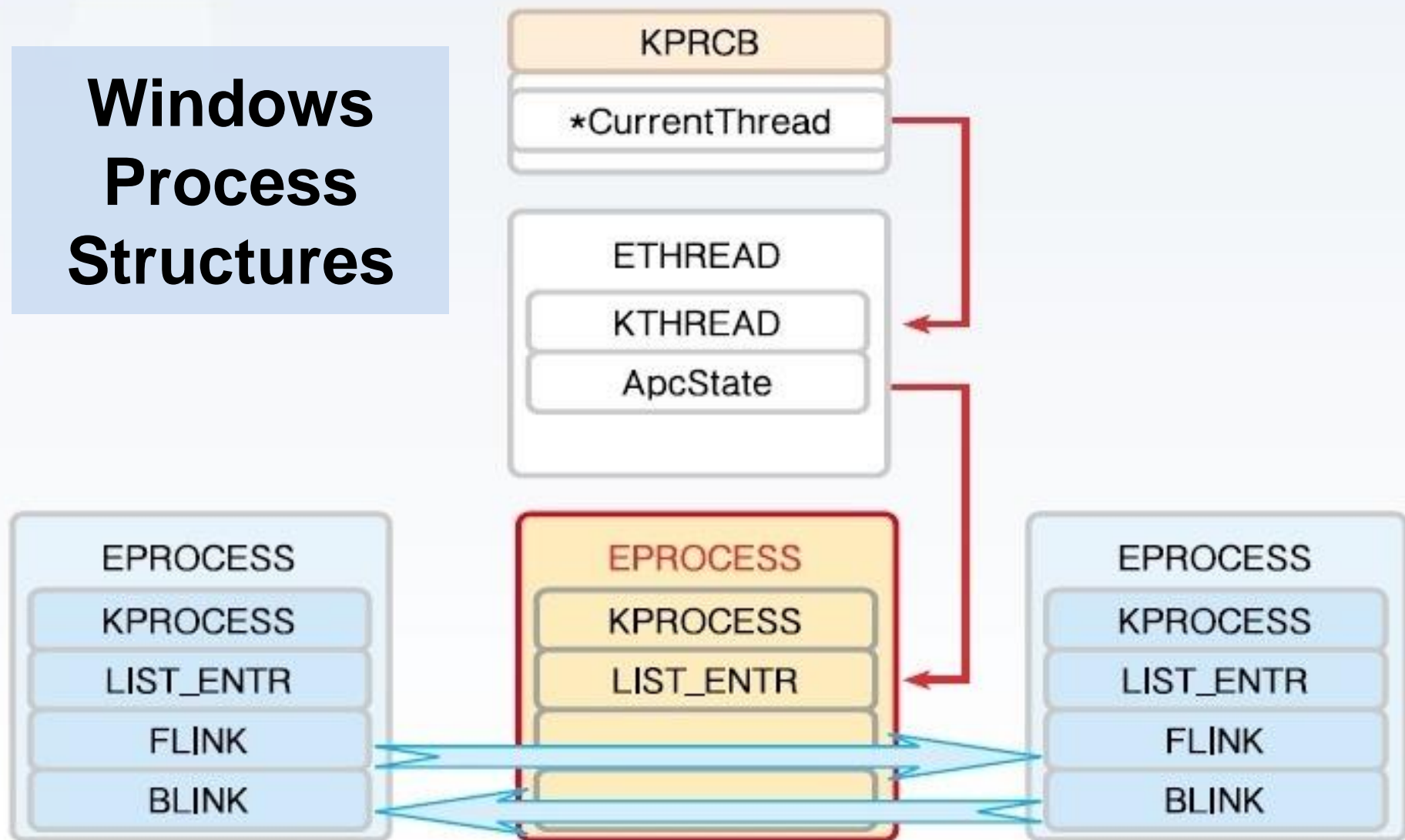
Operating System Data Structures

- 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



Windows Process Structures



Interesting OS Structures

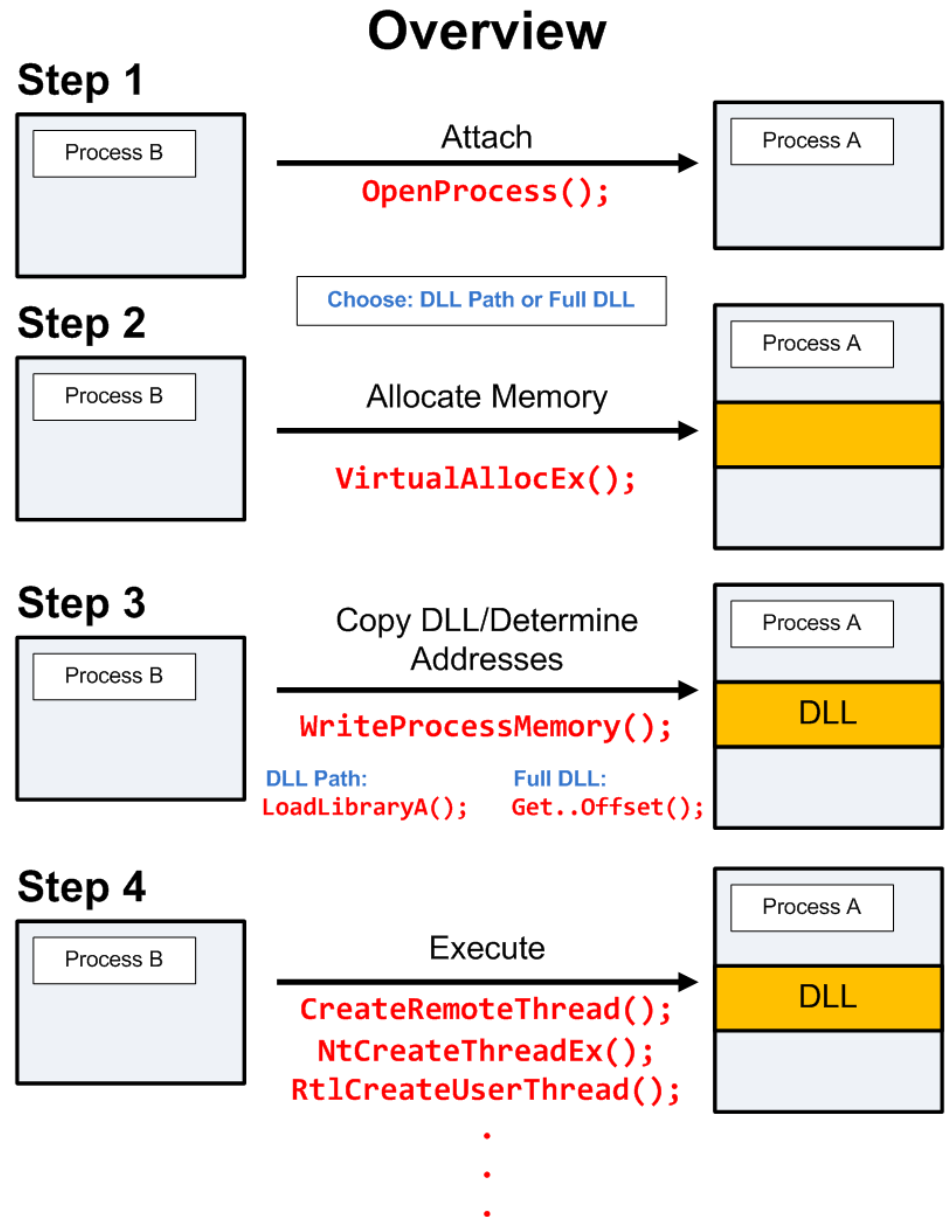
- Suspicious Memory Pages
- Processes
- Threads
- Sockets (Connections)
- Handles (Files)
- Modules/Libraries
- Mutexes
- LSA (Local Security Authority)
- Registry
- ...

Memory Pages

- Various 'flags'
 - Read/write/executable pages
 - Helping OS to organize memory efficiently
- Executable + Writable pages
 - Why is it bad?
- **Process Injection technique**
 - 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...



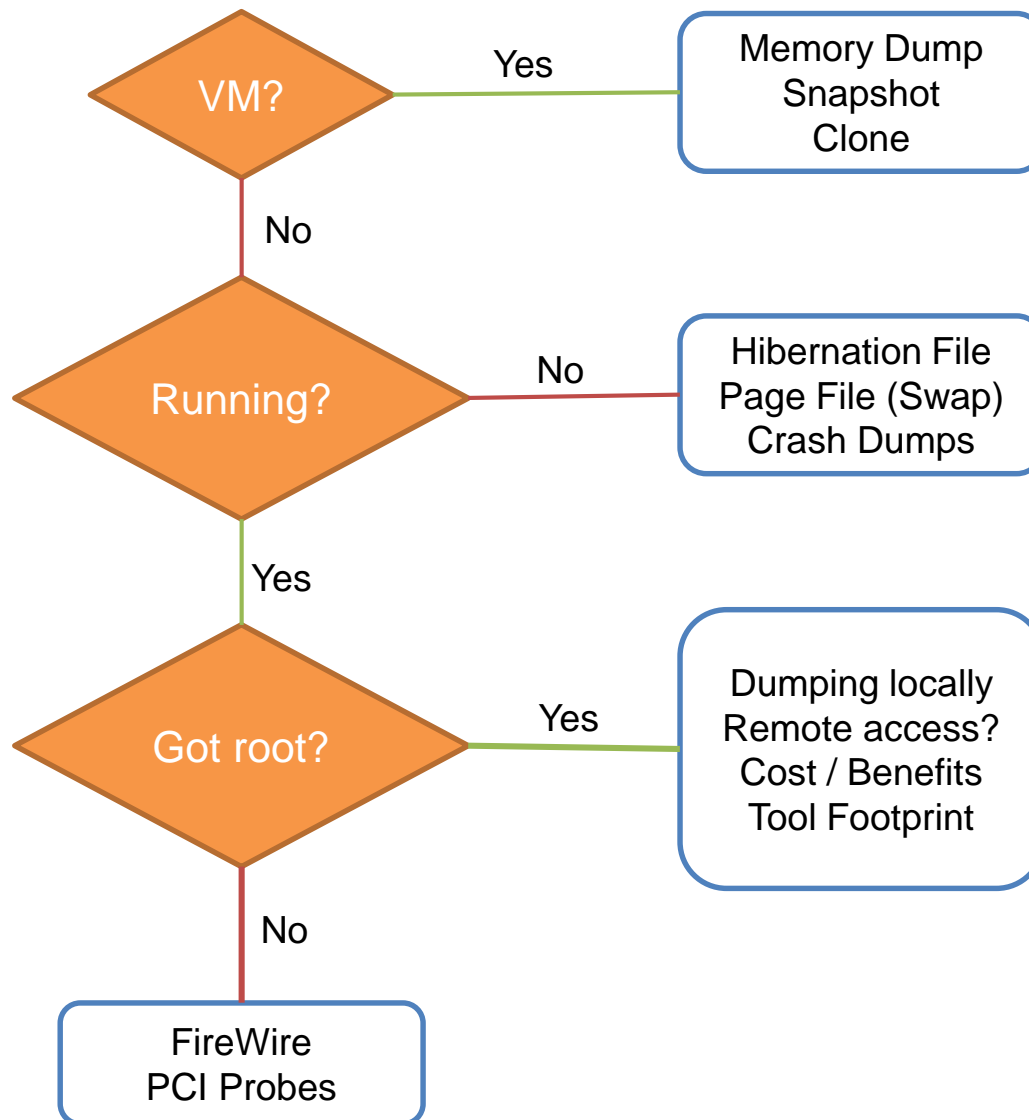
And now something completely...

PRACTICAL

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
 - Very limited analysis options

Memory Acquisition



Memory Acquisition

- **Virtual Machines**
 - VMWare, VirtualBox, ...
 - VirtualBox `-dbg -startvm "MalwareVM"` (and `.pgmphystofile` command)
- Directly from the system! (if we have system rights to do that)
 - `windd`, `fastdump`, `memoryze`
 - 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

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
- Open source alternative?
 - GRR (Google Rapid Response)
 - Still in development, primarily Incident Response tool
 - Allows remote memory acquisition

Memory Analysis Tools

- Mandiant Redline
 - Free, available for Windows
- HBGary Responder (CE/Pro)
 - Community Edition available against registration
- Volatility Framework
 - Open source, no GUI
- Rekall
 - Open source, ‘Volatility done right’, GUI
 - Google supported (part of GRR agent)

Mandiant/FireEye Redline

- Free tool for Incident Response
 - Not open-source, though
 - .NET executable (runs only under Windows)
- 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



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

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 (1300)
 - Explorer.EXE (1336)
 - Dwm.exe (1384)

Processes Tags/Comments
 Fields TimeWrinkles™ 0
 TimeCrunches™ 1 Users

Reg Ex In All Fields [Filter Icon] [Y Icon] 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

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
 - PEInfo 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\ATOMI\ActivePresenter\templates\ajax\Ocean.apt MD5: [N]
2013-02-14 17:23:47Z	File/Modified	Path: C:\Program Files\ATOMI\ActivePresenter\templates\flash MD5: [N]
2013-02-14 17:23:47Z	File/Changed	Path: C:\Program Files\ATOMI\ActivePresenter\templates\flash MD5: [N]
2013-02-14 17:23:47Z	File/Created	Path: C:\Program Files\ATOMI\ActivePresenter\templates\flash\Aluminum.aftpl MD5: [N]
2013-02-14 17:23:47Z	File/Changed	Path: C:\Program Files\ATOMI\ActivePresenter\templates\flash\Aluminum.aftpl MD5: [N]
2013-02-14 17:23:47Z	File/FilenameCreated	Path: C:\Program Files\ATOMI\ActivePresenter\templates\flash\Aluminum.aftpl MD5: [N]
2013-02-14 17:23:47Z	File/FilenameChanged	Path: C:\Program Files\ATOMI\ActivePresenter\templates\flash\Aluminum.aftpl MD5: [N]
2013-02-14 17:23:47Z	File/Created	Path: C:\Program Files\ATOMI\ActivePresenter\templates\flash\components.swf MD5: [N]
2013-02-14 17:23:47Z	File/Changed	Path: C:\Program Files\ATOMI\ActivePresenter\templates\flash\components.swf MD5: [N]
2013-02-14 17:23:47Z	File/FilenameCreated	Path: C:\Program Files\ATOMI\ActivePresenter\templates\flash\components.swf MD5: [N]
2013-02-14 17:23:47Z	File/FilenameChanged	Path: C:\Program Files\ATOMI\ActivePresenter\templates\flash\components.swf MD5: [N]
2013-02-14 17:23:47Z	File/Created	Path: C:\Program Files\ATOMI\ActivePresenter\templates\flash\expressInstall.swf MD5: [N]
2013-02-14 17:23:47Z	File/Changed	Path: C:\Program Files\ATOMI\ActivePresenter\templates\flash\expressInstall.swf MD5: [N]
2013-02-14 17:23:47Z	File/FilenameCreated	Path: C:\Program Files\ATOMI\ActivePresenter\templates\flash\expressInstall.swf MD5: [N]
2013-02-14 17:23:47Z	File/FilenameChanged	Path: C:\Program Files\ATOMI\ActivePresenter\templates\flash\expressInstall.swf MD5: [N]
2013-02-14 17:23:47Z	File/Created	Path: C:\Program Files\ATOMI\ActivePresenter\templates\flash\infobox.swf MD5: [N]
2013-02-14 17:23:47Z	File/Changed	Path: C:\Program Files\ATOMI\ActivePresenter\templates\flash\infobox.swf MD5: [N]
2013-02-14 17:23:47Z	File/FilenameCreated	Path: C:\Program Files\ATOMI\ActivePresenter\templates\flash\infobox.swf MD5: [N]
2013-02-14 17:23:47Z	File/FilenameChanged	Path: C:\Program Files\ATOMI\ActivePresenter\templates\flash\infobox.swf MD5: [N]
2013-02-14 17:23:47Z	File/Created	Path: C:\Program Files\ATOMI\ActivePresenter\templates\flash\json.as MD5: [N]
2013-02-14 17:23:47Z	File/Changed	Path: C:\Program Files\ATOMI\ActivePresenter\templates\flash\json.as MD5: [N]
2013-02-14 17:23:47Z	File/FilenameCreated	Path: C:\Program Files\ATOMI\ActivePresenter\templates\flash\json.as MD5: [N]
2013-02-14 17:23:47Z	File/FilenameChanged	Path: C:\Program Files\ATOMI\ActivePresenter\templates\flash\json.as MD5: [N]
2013-02-14 17:23:47Z	File/Created	Path: C:\Program Files\ATOMI\ActivePresenter\templates\flash\language.as MD5: [N]
2013-02-14 17:23:47Z	File/Changed	Path: C:\Program Files\ATOMI\ActivePresenter\templates\flash\language.as MD5: [N]
2013-02-14 17:23:47Z	File/FilenameCreated	Path: C:\Program Files\ATOMI\ActivePresenter\templates\flash\language.as MD5: [N]
2013-02-14 17:23:47Z	File/FilenameChanged	Path: C:\Program Files\ATOMI\ActivePresenter\templates\flash\language.as MD5: [N]

Redline: Time Wrinkles

New Custom Timeline

Fields TimeWrinkles™ 1

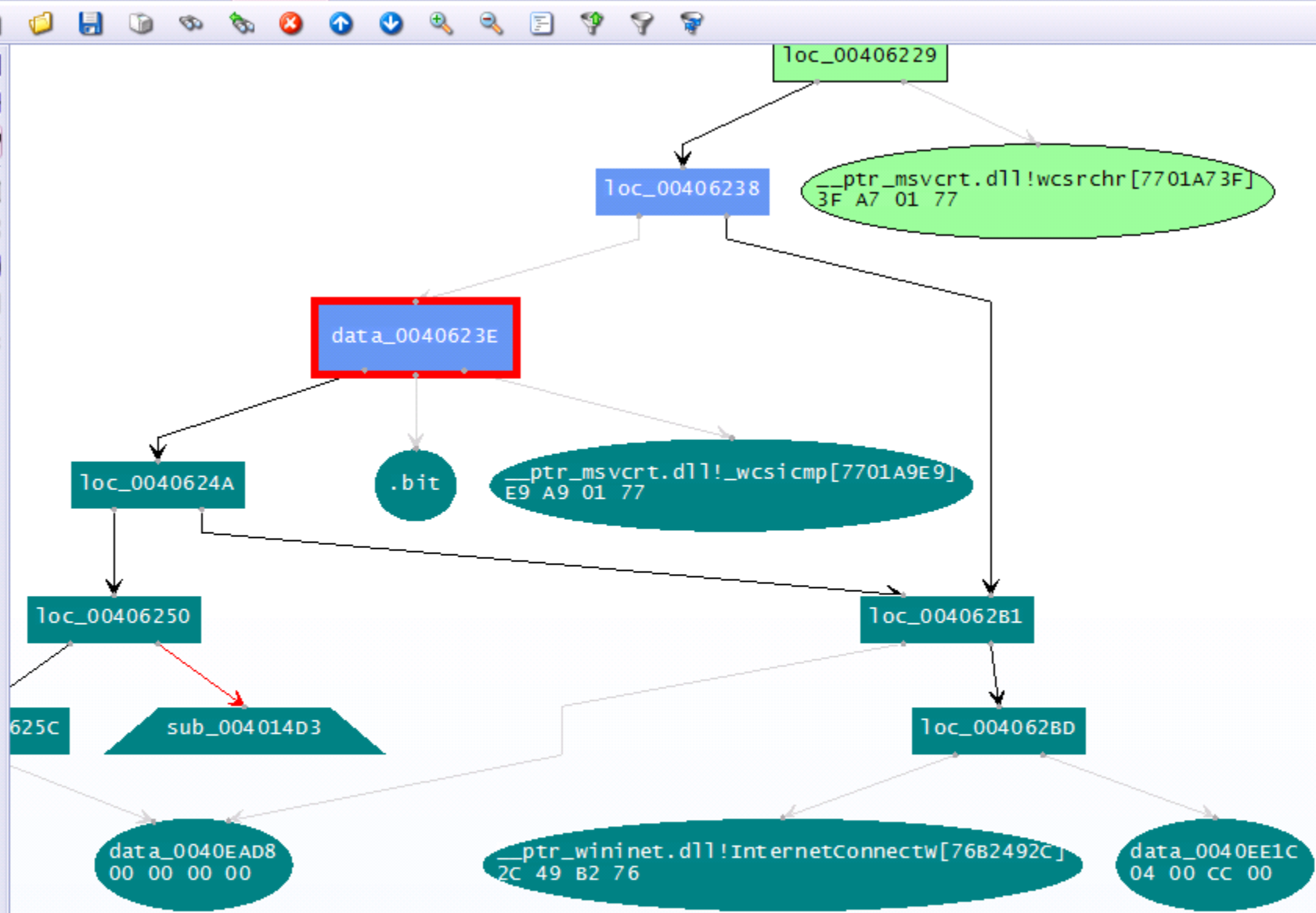
TimeCrunches™ 0 Users Processes

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

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 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

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: `0p1mutx9`
- 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: `35fsdfsdgfd5339`

Operational Security (OpSec)

- Basics of OpSec
 - “Think before you act” mentality
 - Limited information sharing
- Specifics of memory analysis
 - You can often upload dumped executables to VirusTotal
 - md5 of the process 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_2016/
- **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)

Thank you for your attention.

Answers & Questions

LAB



Lab Requirements

- Oracle VirtualBox
 - And enough space on your hard drive (12 GB at least)
- **Volatility Framework**
- Mandiant Redline
- Unix tools
 - strings, foremost
- Your favorite text editor for notes
- 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**

Volatility 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_2016/
- **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)

Thank you for your attention.

Answers & Questions