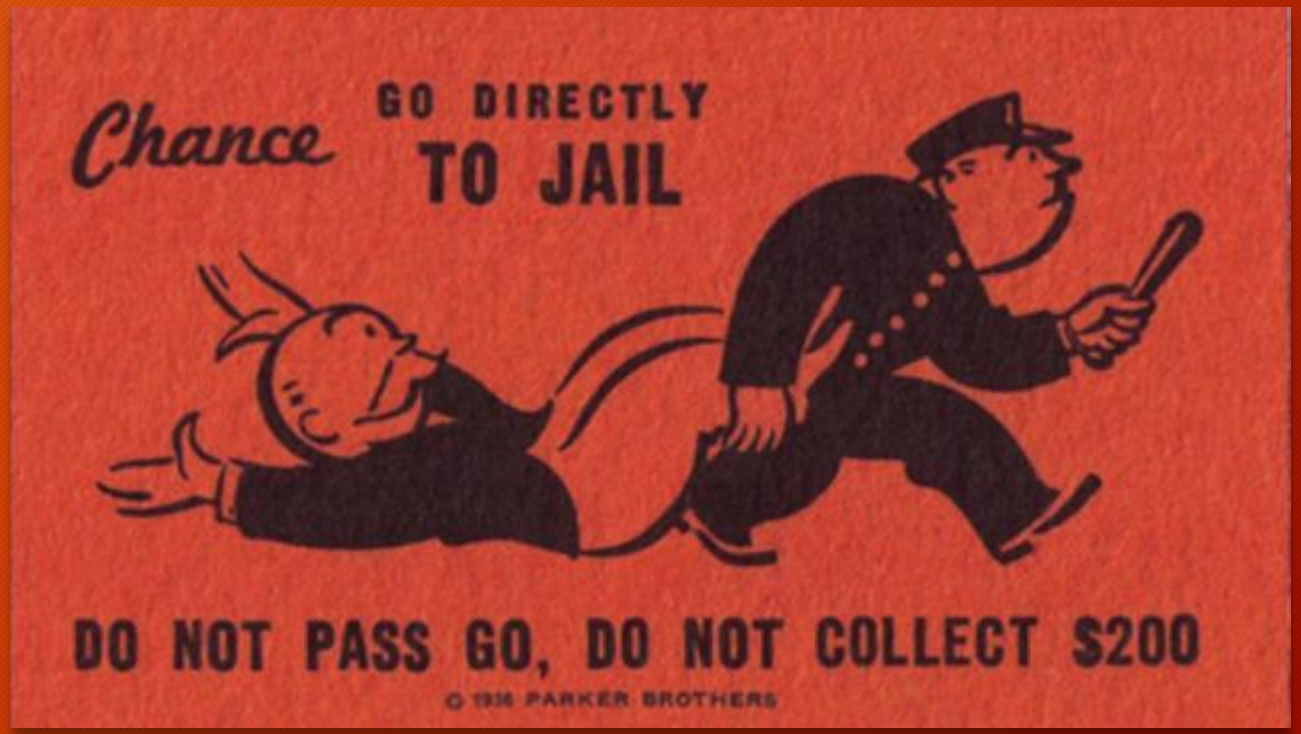
Malware Data Science

Norberto Limon

Disclaimer: Patents & Copyright

- Reverse Engineering patented or copyrighted software is illegal in the United States and EU
- Also some of this information may be a little outdated, incomplete, or just wrong, etc.
- If you break something, don't blame me, you guys know the drill
- So don't try this at home, yet…



Why Malware Data Science…?

- Security Professionals - MA helps supplement and at times automate vulnerability detection and identification of exploits
- Incident Response – Gain insight into methodologies used by attackers and understand how malicious code operates to develop solutions/patches.
- AV supplement – Anti-Virus Detection cannot always keep up with malware development trends, Data Science and Reverse Engineering can help

Security and Malware Analysis

- MA and MDS can be used to gather information, identify exploits, patch vulnerability and mitigate damage potential and impact, as well as visual trends and relationships
- How can Python help?
  - Scripts for Reverse Engineering:
    - Python modules for NSA's Ghidra RE Software
      - https://github.com/NationalSecurityAgency/ghidra/tree/master/Ghidra/Features/Python/ghidra_scripts
    - Python Integration for Hex-Rays' IDA PRO RE Software
      - https://github.com/idapython/src
  - Static Analysis
  - Dynamic Analysis
    - Visualizing trends and behaviors
- Automation tools
  - Capstone
  - Data Science w/ Python + Jupyter

Static Analysis

- Static Analysis aims to gather useful statistics and identifying information about the binary without triggering it to run.
- Extracting Binary metadata for Triage
- File Structure/Resource Allocation
- Identify allocation tables (IATs)
- Some sites will do static analysis for you:
  - Malwr.com
  - Submit file, returns info, similar to metadata websites

Back to Basics: Assembly

- Most Commonly Used Assembly Instructions:
  - CALL
  - MOV
  - ADD
  - JMP
  - CMP
- Step Into vs Step Over

```
    :   top of stack
    :   frame pointer
    :   source
    :   destination

    :   argument 1
    :   argument 3
11  :   for

    :   Must be preserved
    :   Must be preserved

    :   Jump to previous position
L   :   Jump by name
    :   xref

D   :   Next data
I   :   Next immediate value
T   :   Next text
B   :   Next sequence of bytes

F12 :   Function calls

+F3 :   Functions
+F7 :   Segments

F2  :   Stop process
    :   Step over
alt+B:  List breakpoints

    :   Data
    :   Rename
    :   Enter repeatable comment
    :   Edit function
    :   Declare function type
+F2 :   Run script

    :   run
    :   Step over
    :   Pause
    :   Open CPU window
    :   Open log window
    :   Open option window
```

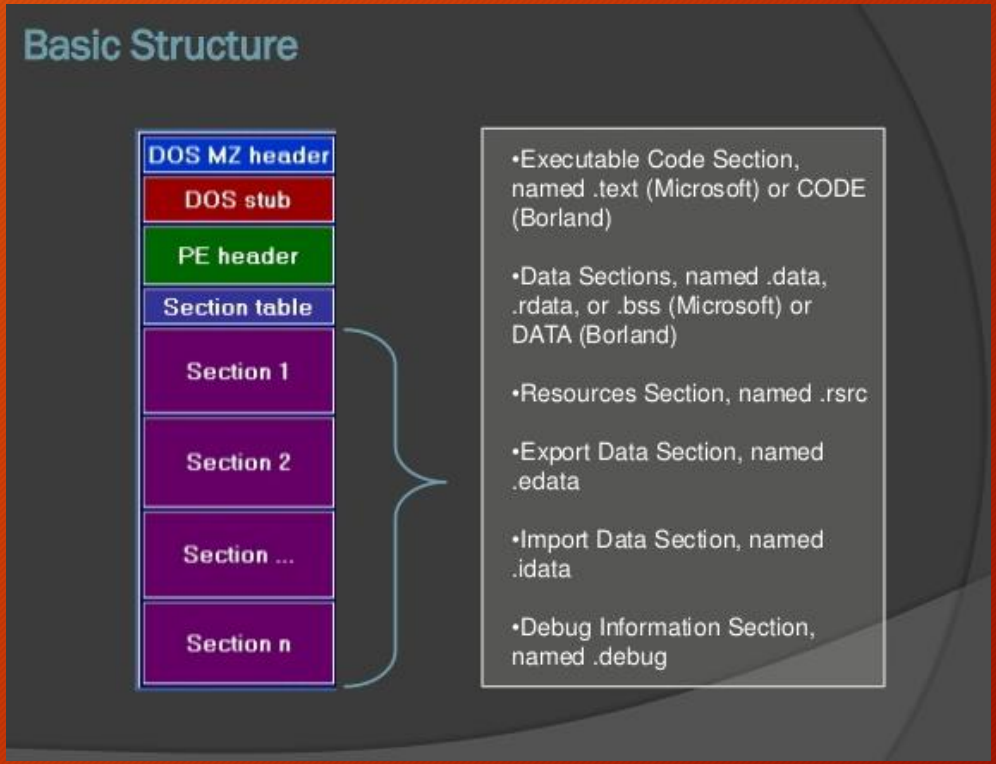
```
[ ASM ]
MOV  : Move (copy)
XCHG : Exchange
PUSH : Push onto stack
POP  : Pop from stack
ADD  : Add
SUB  : Subtract
DIV  : Divide
IDIV : Signed integer divide
MUL  : Multiply
IMUL : Signed integer multiply
INC  : Increment
DEC  : Decrement
SAL  : Shift left
SAR  : Shift right
ROL  : Rotate left
ROR  : Rotate right
NOT  : Invert each bit
AND  : Logical and
OR   : Logical or
XOR  : Logical exclusive or
SHL  : Shift logical left
SHR  : Shift logical right
NOP  : No operation (0x90)
INT  : Interrupt
CALL : Call subroutine
JMP  : Jump
JE   : Jump if equal
JZ   : Jump if zero
JCXZ : Jump if CX zero
JNE  : Jump if not equal
JNZ  : Jump if not zero
JECXZ: Jump if ECX zero
RET  : Return from subroutine
JA   : Jump if above
JAE  : Jump if above or equal
JB   : Jump if below
JBE  : Jump if below or equal
JNA  : Jump if not above
JNAE : Jump if not above or equal
JNB  : Jump if not below
JNBE : Jump if not below or equal
JC   : Jump if carry
JNC  : Jump if no carry
JG   : Jump if greater
JGE  : Jump if greater or equal
JL   : Jump if less
JLE  : Jump if less or equal
JNG  : Jump if not greater
JNGE : Jump if not greater or equal
JNL  : Jump if not less
JNLE : Jump if not less or equal
JO   : Jump if overflow
JNO  : Jump if no overflow
JS   : Jump if sign (= negative)
JNS  : Jump if no sign (= positive)

[ IDA Pro plugins ]
IDA Scope
Diaphora
IDA Tool Bag
IDA Signsrch

[ WinDBG plugin ]
pykd
```

```
[ Win32 memory map ]
0x00000000   Low memory addresses
stack
heap
0x00400000
program image
dlls
TEB
0x7ffd0000
PEB
0x7fffffff
kernel land
0xffffffff   High memory adresses
```

```
[ Python ]
>>> a="A"
>>> print ord(a)
65
>>> print hex(ord(a))
0x41

>>> b=0x42
>>> print str(b)
66
>>> print chr(b)
B

>>> c="\x41"
>>> print c
A
>>> print ord(c)
65

>>> c="\x90"
>>> print c
>>> import sys
>>> sys.stdout.write(c)

>>> int("0x100", 16)
256

#XOR
>>> d="\xBE"
>>> x="\xFF"
>>> print hex(ord(d)^ord(x))
0x41
>>> print chr(ord(d)^ord(x))
A

def rol32(num, count):
  num1 = (num << count) & 0xFFFFFFFF
  num2 = (num >> (0x20-count)) &0xFFFFFFFF
  return num1 | num2

def ror32(num, count):
  num1 = (num >> count) & 0xFFFFFFFF
  num2 = (num << (0x20-count)) &0xFFFFFFFF
  return num1 | num2

def ror8(num, count):
  num1 = (num >> count) & 0xFF
  num2 = (num << (0x08 - count)) & 0xFF
  return num1 | num2

def rol8(num, count):
  num1 = (num << count) & 0xFF
  num2 = (num >> (0x08 - count)) & 0xFF
  return num1 | num2

def shl(dest, count):
  return hex(dest << count)
```

The PE Header

- The PE format is the structure of a Windows program file
  - .exe, .dll, .sys extension
- By knowing your file structure you can begin to extrapolate a lot of useful information
- Section Headers
  - .text, .idata – IAT, dynamically linked library imports
  - Data Sections
    - .rsrc, .data, .rdata



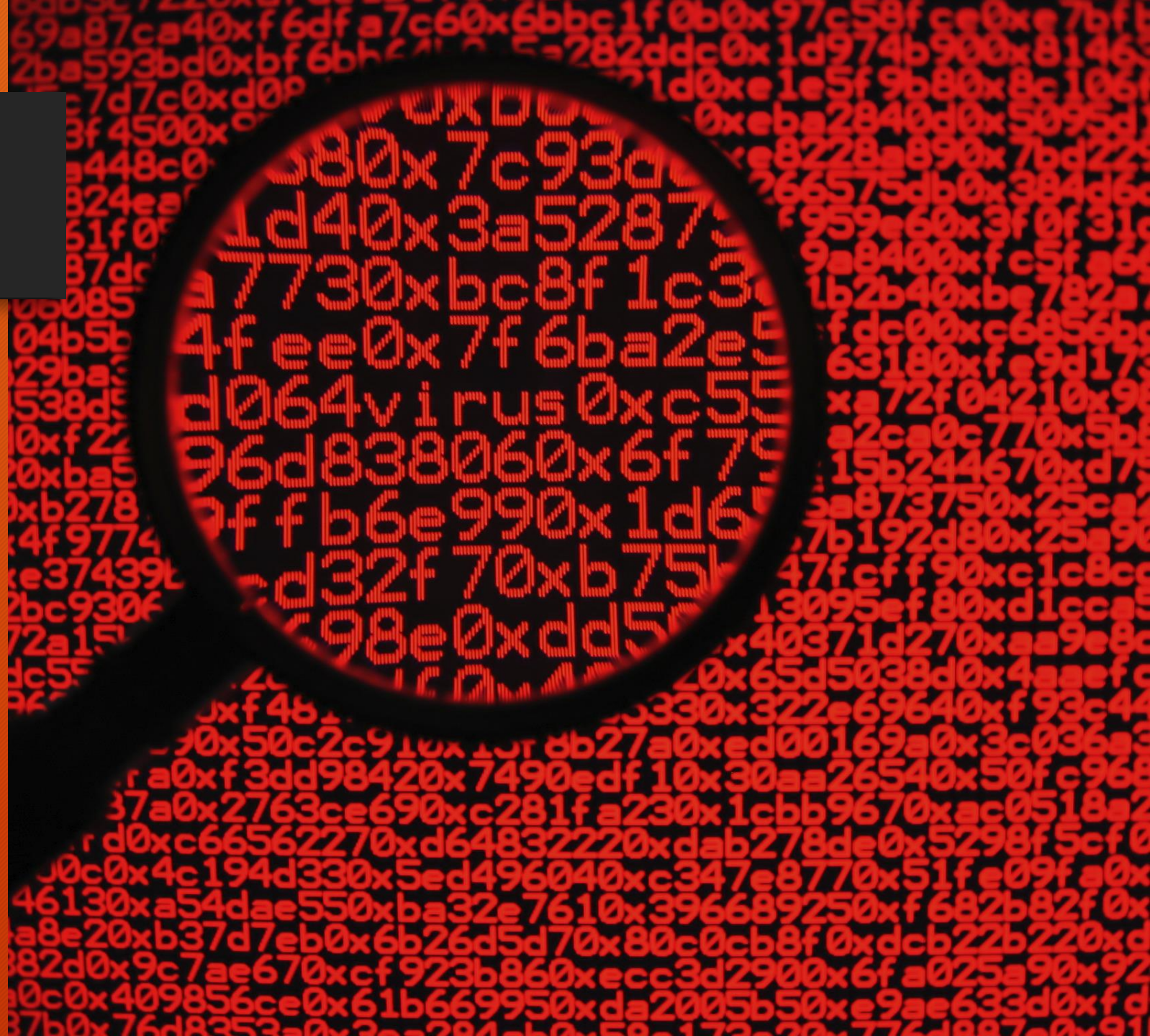
Dynamic Analysis

- The goal of Dynamic Analysis is to measure the malware's different features and behaviors.

Exploit Techniques to watch for:

> 'Code Obfuscation' – encryption, scrambling, goal is to waste as much time as possible. Use lots of jumps to lead them down rabbit holes, dead end etc
>
> 'Packing' – compression for purposes of hiding code inside other program files.
>
> 'Evasion' – Counter to fully Automated Analysis workflows, temporary dormant state, avoid detection

- Virtual Machines, Isolation, Backups

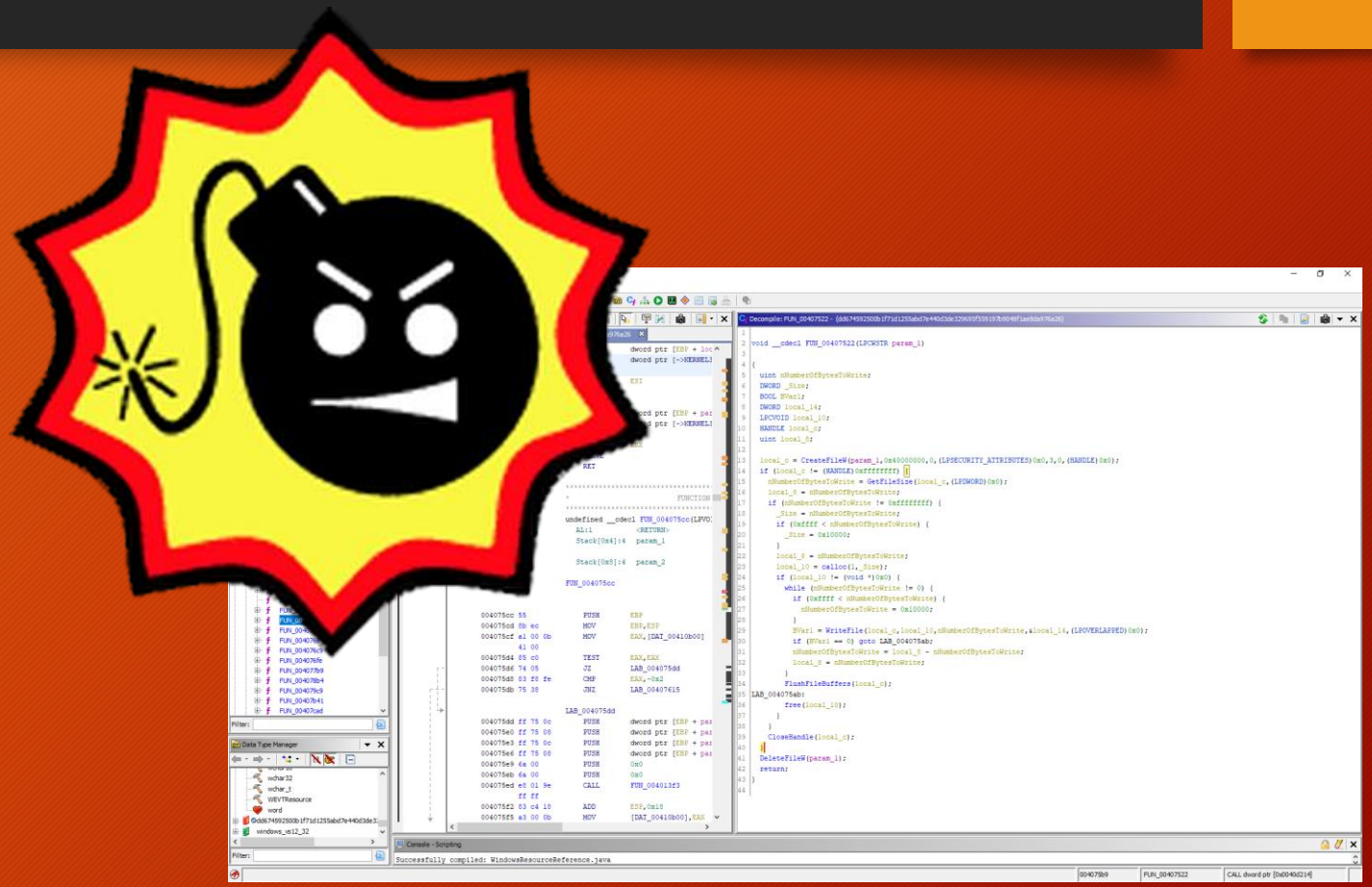


Python

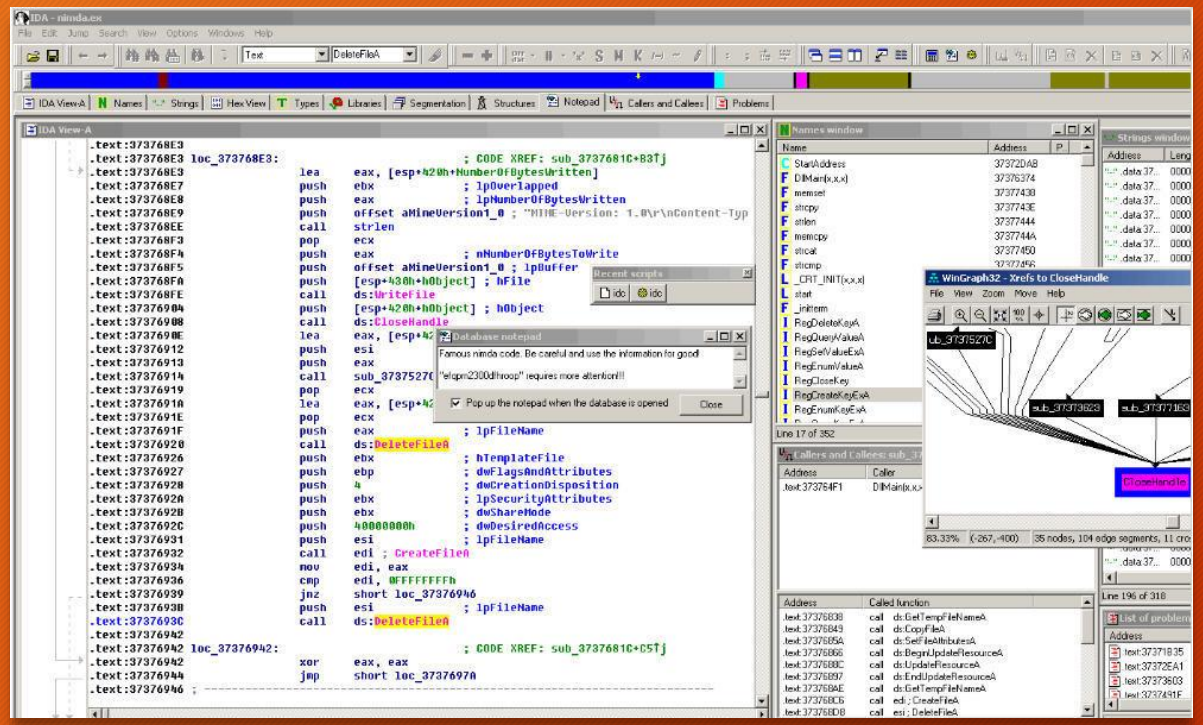
Python can be used to perform Static and Dynamic Analysis, as well as identify trends and patterns

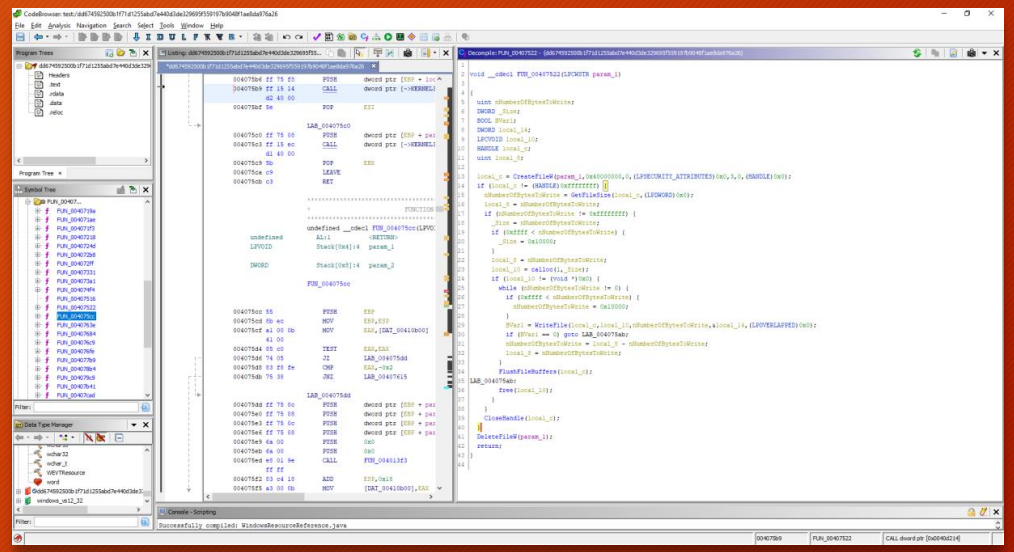
Lab Environment Tips

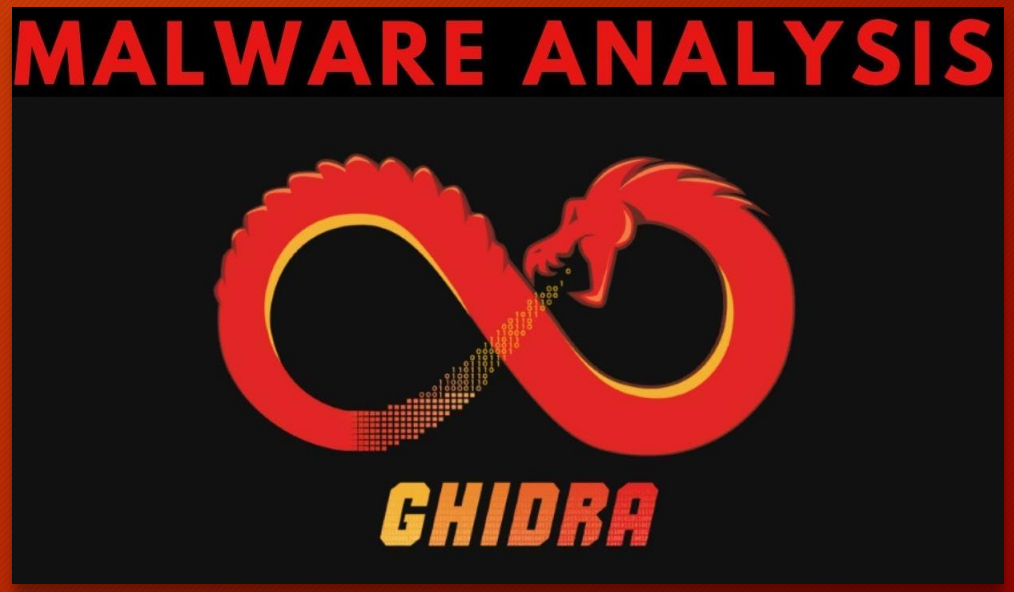
- VirtualBox + Snapshots
- Dedicated Malware Analsis VMs
  - https://malwareunicorn.org/workshops/re101.html#0Backups, and More Backups
- Useful Sites
  - Free Comprehensive Text Book pdf
    - https://beginners.re/RE4B-EN.pdf
  - CookooSandBox.org
  - Reversinghero.com
  - AnubisISEclab.org
  - www.malwr.com
  - www.virustotal.com
  - Flare-on Challenge:
    - www.flare-on.com



Disassembler







Quick Demo

Jupyter-Notebook

IDA Free

Ghidra