# Python & Reverse Engineering Software

*by Alexander Hanel*

# Disclaimer

- Reverse engineering is a very very broad topic.
- A lot of overhead knowledge is required.
- There are no click and it's reverse engineered tools.
- Please ask questions. Some of these topics are esoteric and will need explanations.

# Who Am I.

- Reverse engineer malware at my $dayjob
- Been programming in Python since 2010.
- Wrote the *The Beginner's Guide to IDAPython*, xxxswf.py and author of the blog *Hooked on Mnemonics*.
- Not 1337.
- @nullandnull

# Outline

- Why Python?
  - History from an RE perspective.
  - Why it was adopted.
- What kind of tasks?
- What modules?
- Awesome links!

# Why Python? - History

- Python has been in use by the reverse engineering community for over a decade.
- In 2005 on OpenRCE a number of influential tools were released in Python.
  - IDAPython and Pefile are the most popular.
- In 2005 and 2006 security companies started posting Python as a desired skill.

# Why Python? - Because...

- Python is friendly.
- Prototyping in Python is quick.
  - Continuous recompiling isn’t.
- Adding modules is painless.
- Python is everywhere.
- Batteries included.
  - “import zlib” vs Visual Studio’s Project properties -> Linker -> Input -> Additional dependencies.. etc

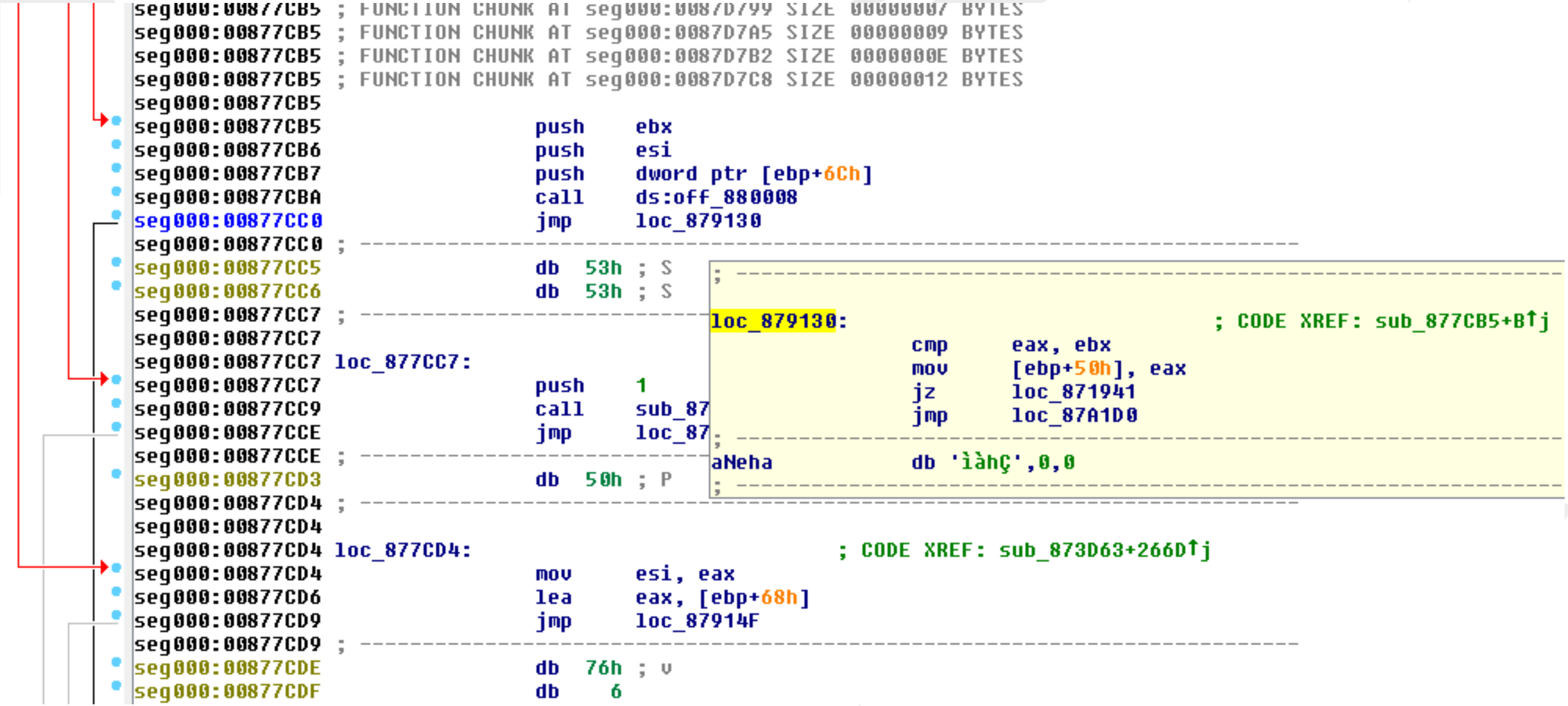
# What kind of tasks?

- It depends of the type of research.
- Python is commonly used for.
  - Binary analysis
  - Forensics
  - Malware analysis
  - Network analysis
  - Exploring file formats
  - Vulnerability and exploit analysis

# Binary Analysis - Tasks

- Disassembling code
  - Binary to Assembly
- Automating the analysis of code
- Deobfuscating code
- Solving Cracks Me
- Full binary analysis frameworks have been written in Python

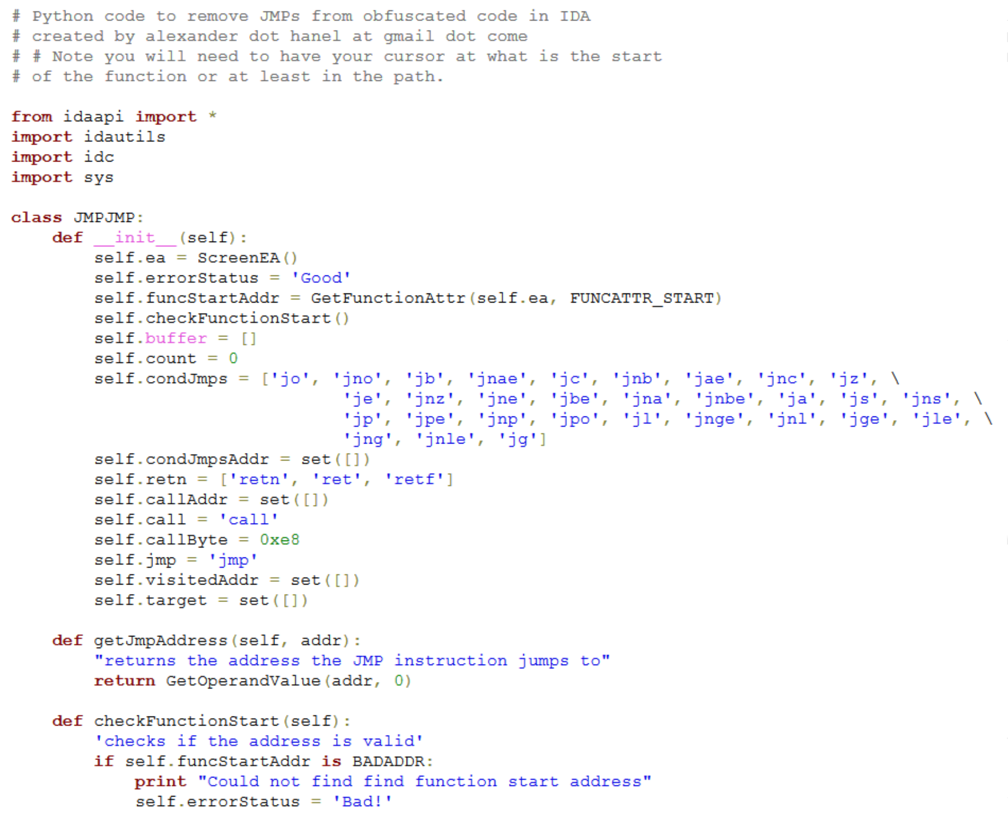
# Binary Analysis - Example



```
seg000:00877CB5 ; FUNCTION CHUNK AT seg000:0087D799 SIZE 00000007 BYTES
seg000:00877CB5 ; FUNCTION CHUNK AT seg000:0087D7A5 SIZE 00000009 BYTES
seg000:00877CB5 ; FUNCTION CHUNK AT seg000:0087D7B2 SIZE 0000000E BYTES
seg000:00877CB5 ; FUNCTION CHUNK AT seg000:0087D7C8 SIZE 00000012 BYTES
seg000:00877CB5
seg000:00877CB5                 push    ebx
seg000:00877CB6                 push    esi
seg000:00877CB7                 push    dword ptr [ebp+6Ch]
seg000:00877CBA                 call    ds:off_880008
seg000:00877CC0                 jmp     loc_879130
seg000:00877CC0 ; ---------------------------------------------------------------------------
seg000:00877CC5                 db  53h ; S
seg000:00877CC6                 db  53h ; S
seg000:00877CC7 ; ----------------------------
seg000:00877CC7
seg000:00877CC7 loc_877CC7:
seg000:00877CC7                 push    1
seg000:00877CC9                 call    sub_87
seg000:00877CCE                 jmp     loc_87
seg000:00877CCE ; ----------------------------
seg000:00877CD3                 db  50h ; P
seg000:00877CD4 ; ---------------------------------------------------------------------------
seg000:00877CD4
seg000:00877CD4 loc_877CD4:                             ; CODE XREF: sub_873D63+266D↑j
seg000:00877CD4                 mov     esi, eax
seg000:00877CD6                 lea     eax, [ebp+68h]
seg000:00877CD9                 jmp     loc_87914F
seg000:00877CD9 ; ---------------------------------------------------------------------------
seg000:00877CDE                 db  76h ; v
seg000:00877CDF                 db    6
```

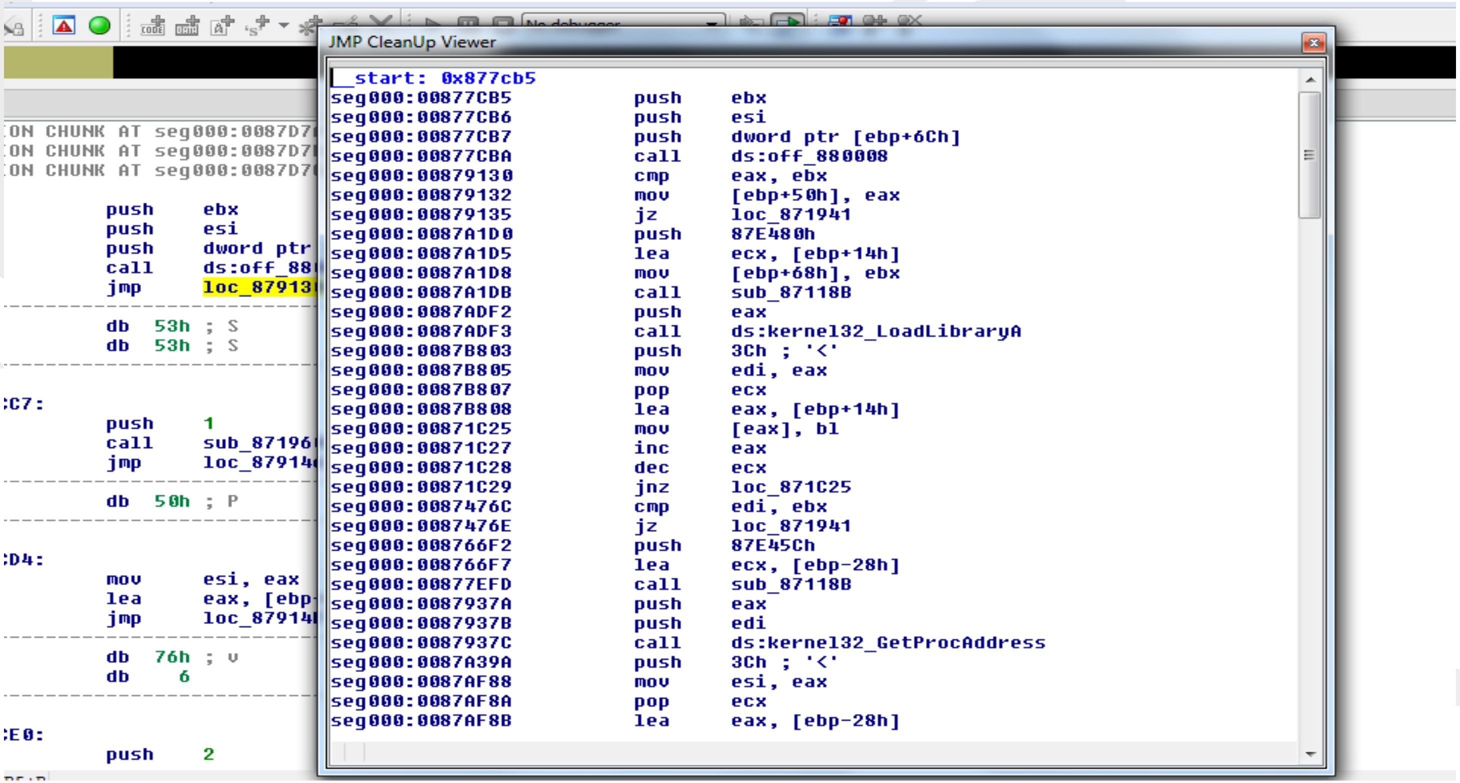
```
; ---------------------------------------------------------------------------

loc_879130:                             ; CODE XREF: sub_877CB5+B↑j
                cmp     eax, ebx
                mov     [ebp+50h], eax
                jz      loc_871941
                jmp     loc_87A1D0
; ---------------------------------------------------------------------------
aNeha           db 'ìàhÇ',0,0
; ---------------------------------------------------------------------------
```

# Binary Analysis - Example



```
# Python code to remove JMPs from obfuscated code in IDA
# created by alexander dot hanel at gmail dot come
# # Note you will need to have your cursor at what is the start
# of the function or at least in the path.

from idaapi import *
import idautils
import idc
import sys

class JMPJMP:
    def __init__(self):
        self.ea = ScreenEA()
        self.errorStatus = 'Good'
        self.funcStartAddr = GetFunctionAttr(self.ea, FUNCATTR_START)
        self.checkFunctionStart()
        self.buffer = []
        self.count = 0
        self.condJmps = ['jo', 'jno', 'jb', 'jnae', 'jc', 'jnb', 'jae', 'jnc', 'jz', \
                                'je', 'jnz', 'jne', 'jbe', 'jna', 'jnbe', 'ja', 'js', 'jns', \
                                'jp', 'jpe', 'jnp', 'jpo', 'jl', 'jnge', 'jnl', 'jge', 'jle', \
                                'jng', 'jnle', 'jg']
        self.condJmpsAddr = set([])
        self.retn = ['retn', 'ret', 'retf']
        self.callAddr = set([])
        self.call = 'call'
        self.callByte = 0xe8
        self.jmp = 'jmp'
        self.visitedAddr = set([])
        self.target = set([])

    def getJmpAddress(self, addr):
        "returns the address the JMP instruction jumps to"
        return GetOperandValue(addr, 0)

    def checkFunctionStart(self):
        'checks if the address is valid'
        if self.funcStartAddr is BADADDR:
            print "Could not find find function start address"
            self.errorStatus = 'Bad!'
```

# Binary Analysis - Example



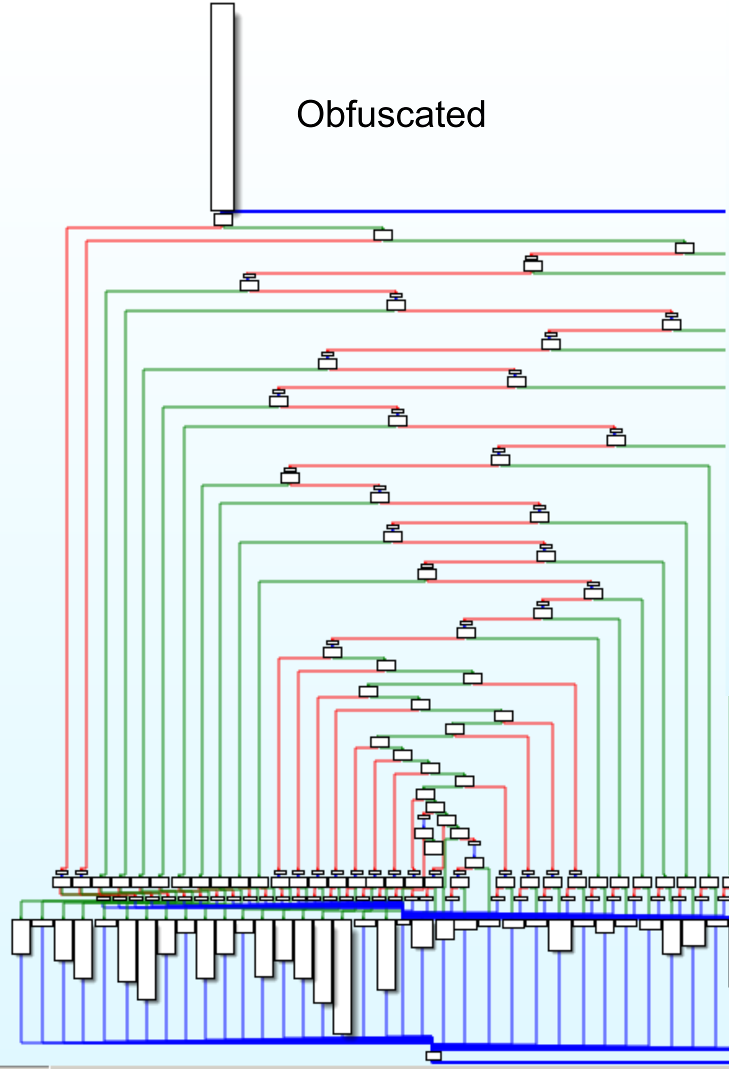
```
ON CHUNK AT seg000:0087D7
ON CHUNK AT seg000:0087D7
ON CHUNK AT seg000:0087D7

        push    ebx
        push    esi
        push    dword ptr
        call    ds:off_88
        jmp     loc_87913
---------------------------------
        db  53h ; S
        db  53h ; S
---------------------------------
:C7:
        push    1
        call    sub_87196
        jmp     loc_87914
---------------------------------
        db  50h ; P
---------------------------------
:D4:
        mov     esi, eax
        lea     eax, [ebp
        jmp     loc_87914
---------------------------------
        db  76h ; v
        db    6
---------------------------------
:E0:
        push    2
```

```
__start: 0x877cb5
seg000:00877CB5          push    ebx
seg000:00877CB6          push    esi
seg000:00877CB7          push    dword ptr [ebp+6Ch]
seg000:00877CBA          call    ds:off_880008
seg000:00879130          cmp     eax, ebx
seg000:00879132          mov     [ebp+50h], eax
seg000:00879135          jz      loc_871941
seg000:0087A1D0          push    87E480h
seg000:0087A1D5          lea     ecx, [ebp+14h]
seg000:0087A1D8          mov     [ebp+68h], ebx
seg000:0087A1DB          call    sub_87118B
seg000:0087ADF2          push    eax
seg000:0087ADF3          call    ds:kernel32_LoadLibraryA
seg000:0087B803          push    3Ch ; '<'
seg000:0087B805          mov     edi, eax
seg000:0087B807          pop     ecx
seg000:0087B808          lea     eax, [ebp+14h]
seg000:00871C25          mov     [eax], bl
seg000:00871C27          inc     eax
seg000:00871C28          dec     ecx
seg000:00871C29          jnz     loc_871C25
seg000:0087476C          cmp     edi, ebx
seg000:0087476E          jz      loc_871941
seg000:008766F2          push    87E45Ch
seg000:008766F7          lea     ecx, [ebp-28h]
seg000:00877EFD          call    sub_87118B
seg000:0087937A          push    eax
seg000:0087937B          push    edi
seg000:0087937C          call    ds:kernel32_GetProcAddress
seg000:0087A39A          push    3Ch ; '<'
seg000:0087AF88          mov     esi, eax
seg000:0087AF8A          pop     ecx
seg000:0087AF8B          lea     eax, [ebp-28h]
```

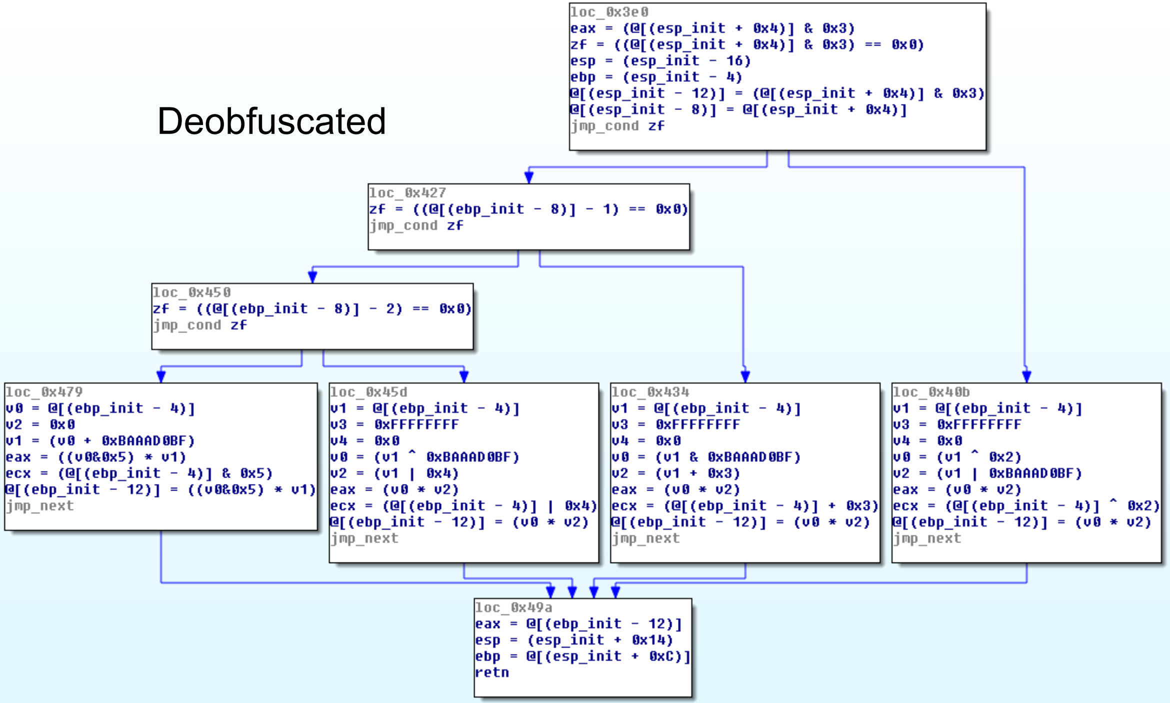
# Binary Analysis - Cool Examples!

- Using Microsoft's Z3 Theorem Prover to solve a CrackMe **.
- Deobfuscation: recovering an OLLVM-protected program **.
- Breaking Kryptonite's Obfuscation: A Static Analysis Approach Relying on Symbolic Execution **.

Obfuscated



Deobfuscated



```
loc_0x3e0
eax = (@[(esp_init + 0x4)] & 0x3)
zf = ((@[(esp_init + 0x4)] & 0x3) == 0x0)
esp = (esp_init - 16)
ebp = (esp_init - 4)
@[(esp_init - 12)] = (@[(esp_init + 0x4)] & 0x3)
@[(esp_init - 8)] = @[(esp_init + 0x4)]
jmp_cond zf
```

```
loc_0x427
zf = ((@[(ebp_init - 8)] - 1) == 0x0)
jmp_cond zf
```

```
loc_0x450
zf = ((@[(ebp_init - 8)] - 2) == 0x0)
jmp_cond zf
```

```
loc_0x479
v0 = @[(ebp_init - 4)]
v2 = 0x0
v1 = (v0 + 0xBAAAD0BF)
eax = ((v0&0x5) * v1)
ecx = (@[(ebp_init - 4)] & 0x5)
@[(ebp_init - 12)] = ((v0&0x5) * v1)
jmp_next
```

```
loc_0x45d
v1 = @[(ebp_init - 4)]
v3 = 0xFFFFFFFF
v4 = 0x0
v0 = (v1 ^ 0xBAAAD0BF)
v2 = (v1 | 0x4)
eax = (v0 * v2)
ecx = (@[(ebp_init - 4)] | 0x4)
@[(ebp_init - 12)] = (v0 * v2)
jmp_next
```

```
loc_0x434
v1 = @[(ebp_init - 4)]
v3 = 0xFFFFFFFF
v4 = 0x0
v0 = (v1 & 0xBAAAD0BF)
v2 = (v1 + 0x3)
eax = (v0 * v2)
ecx = (@[(ebp_init - 4)] + 0x3)
@[(ebp_init - 12)] = (v0 * v2)
jmp_next
```

```
loc_0x40b
v1 = @[(ebp_init - 4)]
v3 = 0xFFFFFFFF
v4 = 0x0
v0 = (v1 ^ 0x2)
v2 = (v1 | 0xBAAAD0BF)
eax = (v0 * v2)
ecx = (@[(ebp_init - 4)] ^ 0x2)
@[(ebp_init - 12)] = (v0 * v2)
jmp_next
```

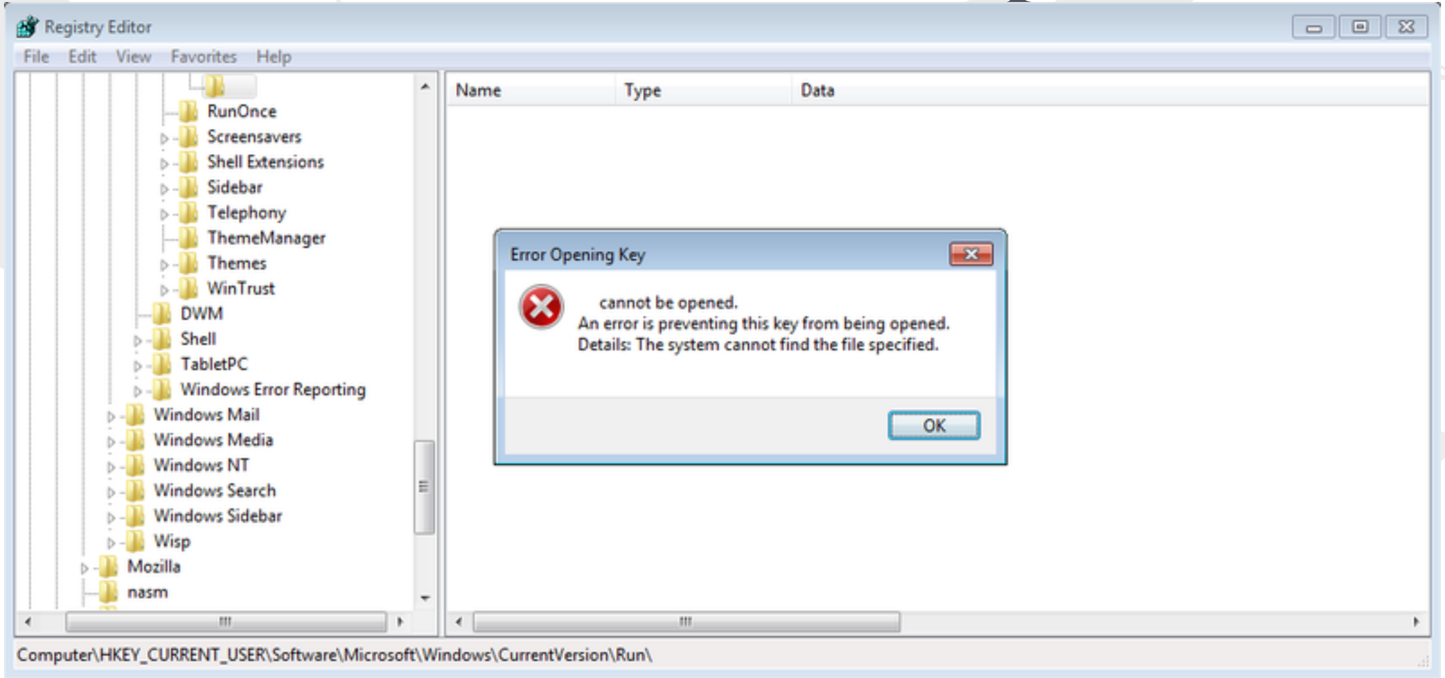
```
loc_0x49a
eax = @[(ebp_init - 12)]
esp = (esp_init + 0x14)
ebp = @[(esp_init + 0xC)]
retn
```

Via Quarkslab blog

# Forensics - Why & Tasks

- Most tools are *not* platform dependent.
    - Example: Analyzing a Windows memory dump in Linux
- File and disk analysis.
    - Timelines
- Parsing the registry.
    - Extracting shellbag data to see folders accesses.
- Memory analysis.
    - Analyzing memory dumps

# Forensics - Example



cannot be opened.
An error is preventing this key from being opened.
Details: The system cannot find the file specified.

Computer\HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run\

Image via G-Data blog

- Error is caused by regedit unable to display invalid characters written by poweliks. An analyst could copy %userprofile%/ntuser.dat to a separate machine and parse the hive using python-registry.

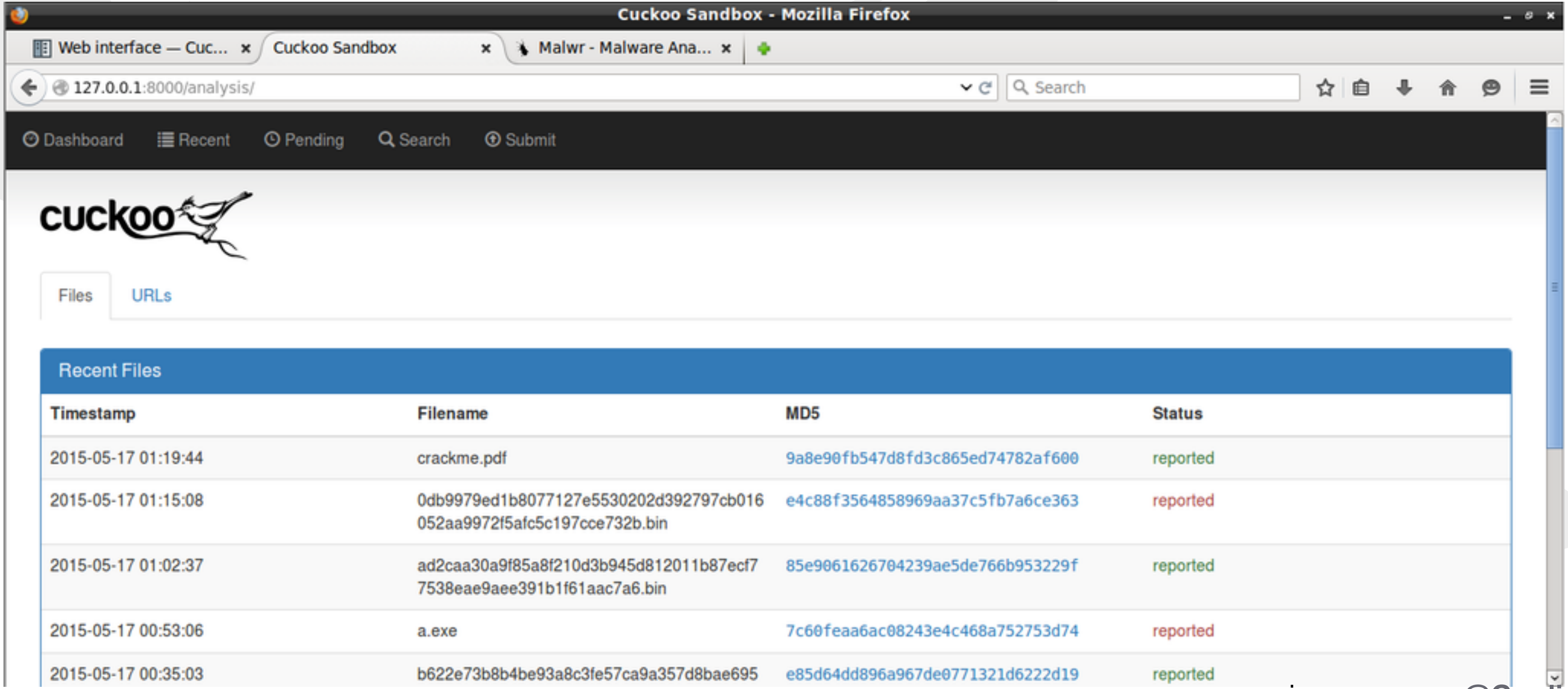
# Forensics - Cool Examples!

- python-registry Introduction by Willi Ballenthin **.
- Automating DFIR Series by David Cowen **.
- Stuxnet’s Footprint in Memory with Volatility 2.0 **.
- Extracting the Powelik's DLL from the Registry **.

# Malware Analysis- Tasks

- Automating the analysis of samples in a sandbox environment.
- Deobfuscation, decompressing and decoding data.
- Debugging and disassembling.
- Scanning files.
- Extracting data.
- Hard to describe so many amazing projects.

# Malware Analysis - Sample Automation

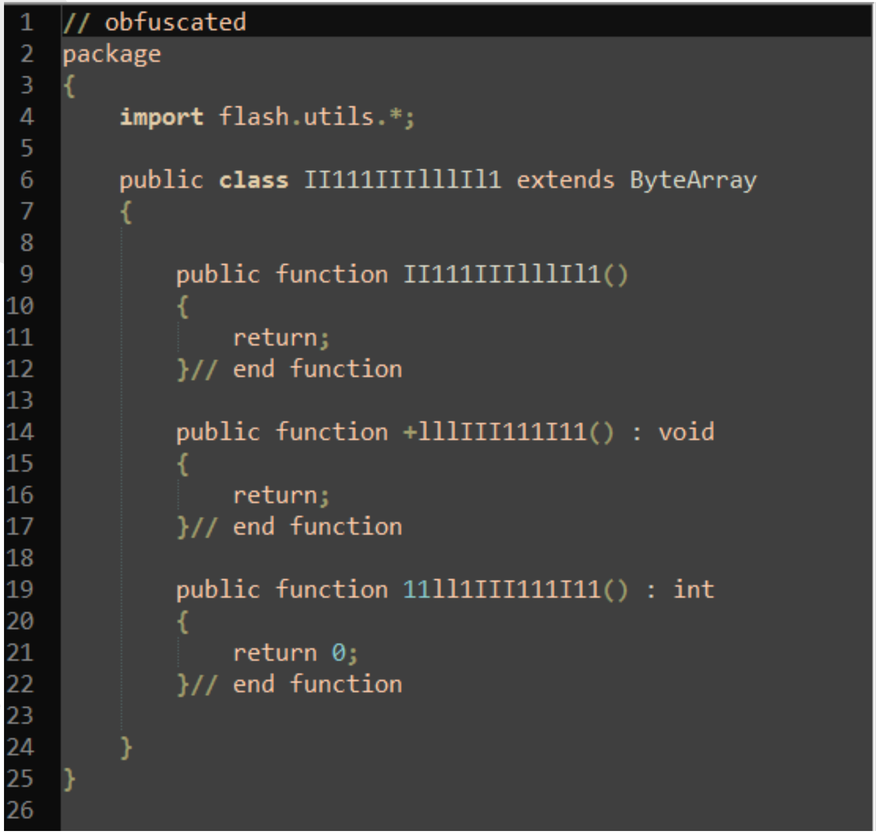


Recent Files

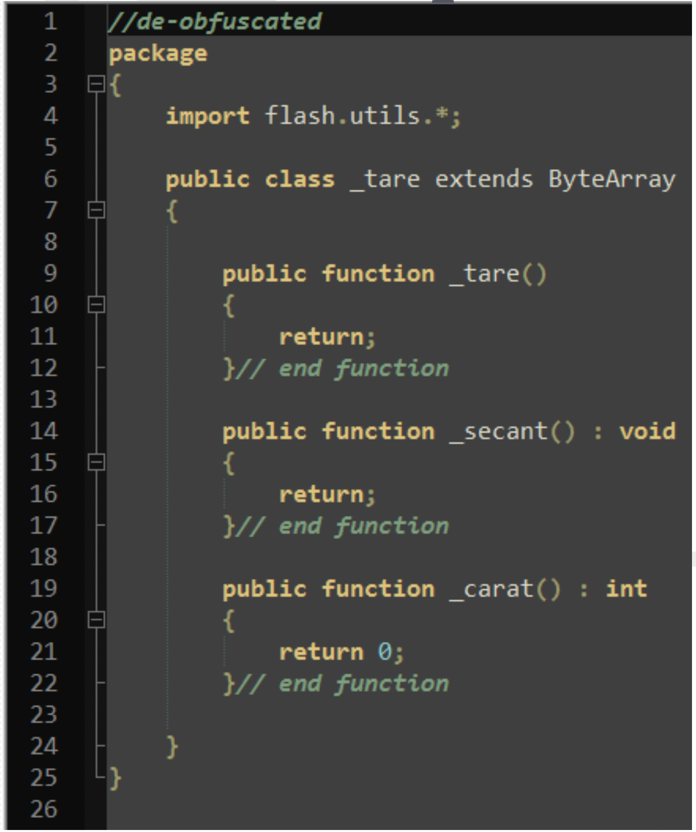
| Timestamp | Filename | MD5 | Status |
|---|---|---|---|
| 2015-05-17 01:19:44 | crackme.pdf | 9a8e90fb547d8fd3c865ed74782af600 | reported |
| 2015-05-17 01:15:08 | 0db9979ed1b8077127e5530202d392797cb016052aa9972f5afc5c197cce732b.bin | e4c88f3564858969aa37c5fb7a6ce363 | reported |
| 2015-05-17 01:02:37 | ad2caa30a9f85a8f210d3b945d812011b87ecf77538eae9aee391b1f61aac7a6.bin | 85e9061626704239ae5de766b953229f | reported |
| 2015-05-17 00:53:06 | a.exe | 7c60feaa6ac08243e4c468a752753d74 | reported |
| 2015-05-17 00:35:03 | b622e73b8b4be93a8c3fe57ca9a357d8bae695 | e85d64dd896a967de0771321d6222d19 | reported |

image source @GradiusX

# Malware Analysis - Deobfuscation Example

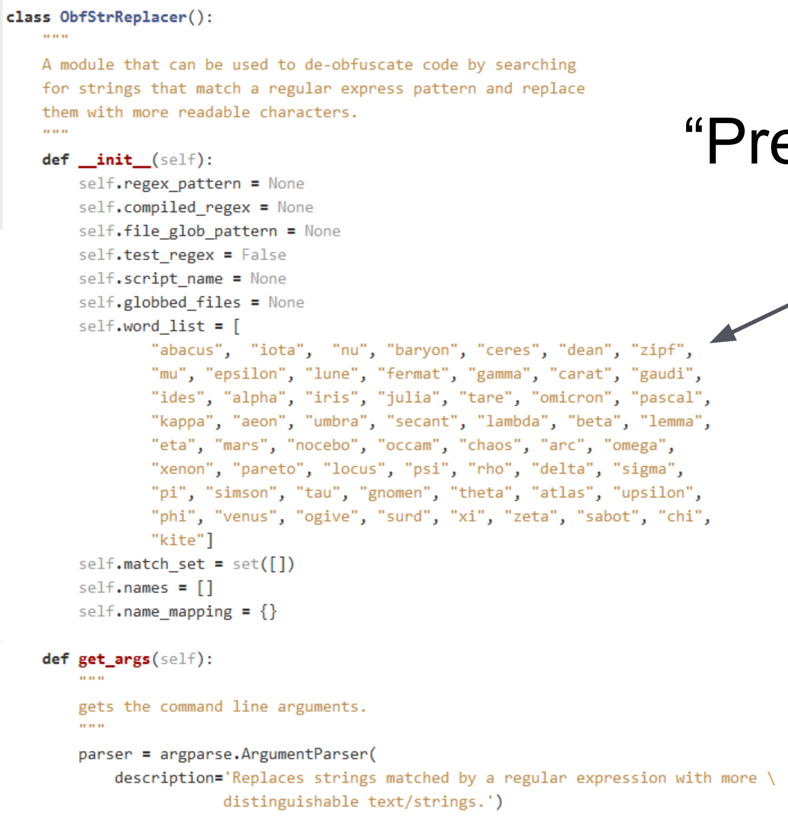


```
// obfuscated
package
{
    import flash.utils.*;

    public class II111IIIlllIl1 extends ByteArray
    {

        public function II111IIIlllIl1()
        {
            return;
        }// end function

        public function +lllIII111I11() : void
        {
            return;
        }// end function

        public function 11ll1III111I11() : int
        {
            return 0;
        }// end function

    }
}
```



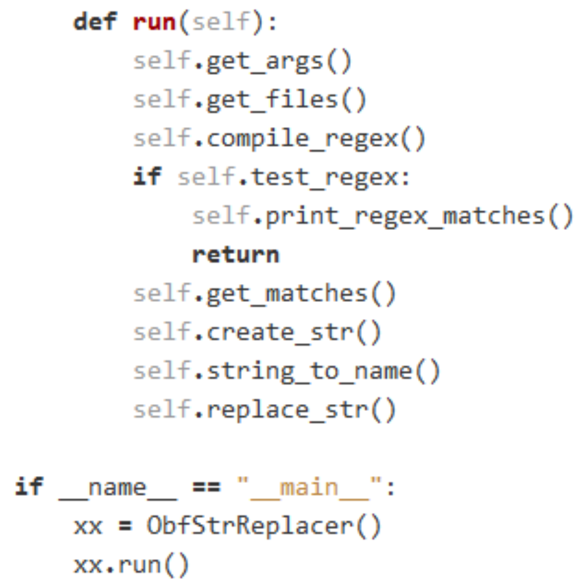
```
//de-obfuscated
package
{
    import flash.utils.*;

    public class _tare extends ByteArray
    {

        public function _tare()
        {
            return;
        }// end function

        public function _secant() : void
        {
            return;
        }// end function

        public function _carat() : int
        {
            return 0;
        }// end function

    }
}
```

# Malware Analysis - Example



```
class ObfStrReplacer():
    """
    A module that can be used to de-obfuscate code by searching
    for strings that match a regular express pattern and replace
    them with more readable characters.
    """
    def __init__(self):
        self.regex_pattern = None
        self.compiled_regex = None
        self.file_glob_pattern = None
        self.test_regex = False
        self.script_name = None
        self.globbed_files = None
        self.word_list = [
                "abacus",  "iota",  "nu", "baryon", "ceres", "dean", "zipf",
                "mu", "epsilon", "lune", "fermat", "gamma", "carat", "gaudi",
                "ides", "alpha", "iris", "julia", "tare", "omicron", "pascal",
                "kappa", "aeon", "umbra", "secant", "lambda", "beta", "lemma",
                "eta", "mars", "nocebo", "occam", "chaos", "arc", "omega",
                "xenon", "pareto", "locus", "psi", "rho", "delta", "sigma",
                "pi", "simson", "tau", "gnomen", "theta", "atlas", "upsilon",
                "phi", "venus", "ogive", "surd", "xi", "zeta", "sabot", "chi",
                "kite"]
        self.match_set = set([])
        self.names = []
        self.name_mapping = {}

    def get_args(self):
        """
        gets the command line arguments.
        """
        parser = argparse.ArgumentParser(
            description='Replaces strings matched by a regular expression with more \
                        distinguishable text/strings.')
```

“Pretty” replacement words



```
    def run(self):
        self.get_args()
        self.get_files()
        self.compile_regex()
        if self.test_regex:
            self.print_regex_matches()
            return
        self.get_matches()
        self.create_str()
        self.string_to_name()
        self.replace_str()

if __name__ == "__main__":
    xx = ObfStrReplacer()
    xx.run()
```

# Malware Analysis- Projects!

- My Favorites or at least should be mentioned
  - Cuckoo Sandbox
  - Yara
  - winappdbg, pydbg, pykd and vivisect
  - Capstone Project
  - IdaPython
  - pefile
  - IDAScope

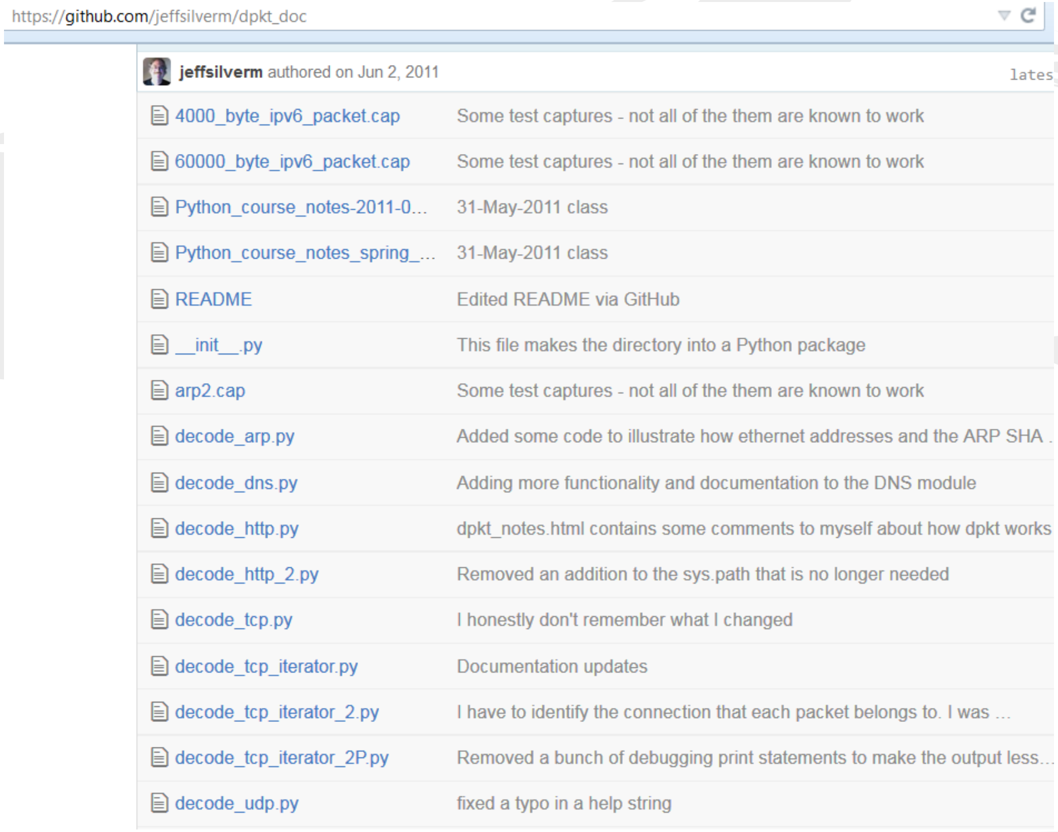
# Network Analysis- Tasks

- Protocol and decoding analysis
- Network and browser emulation.
- PCAP parsing.
- Packet creation, sniffing and manipulation.
- Custom passive DNS tool.
- Automating URL lookups

# Network Analysis- Projects!

- Scapy or Dpkt
- Chopshop built on top of Pynids
- fakedns.py or Fakenet (python bindings)
- jsunpack
- Malcom

# Network Analysis- Example



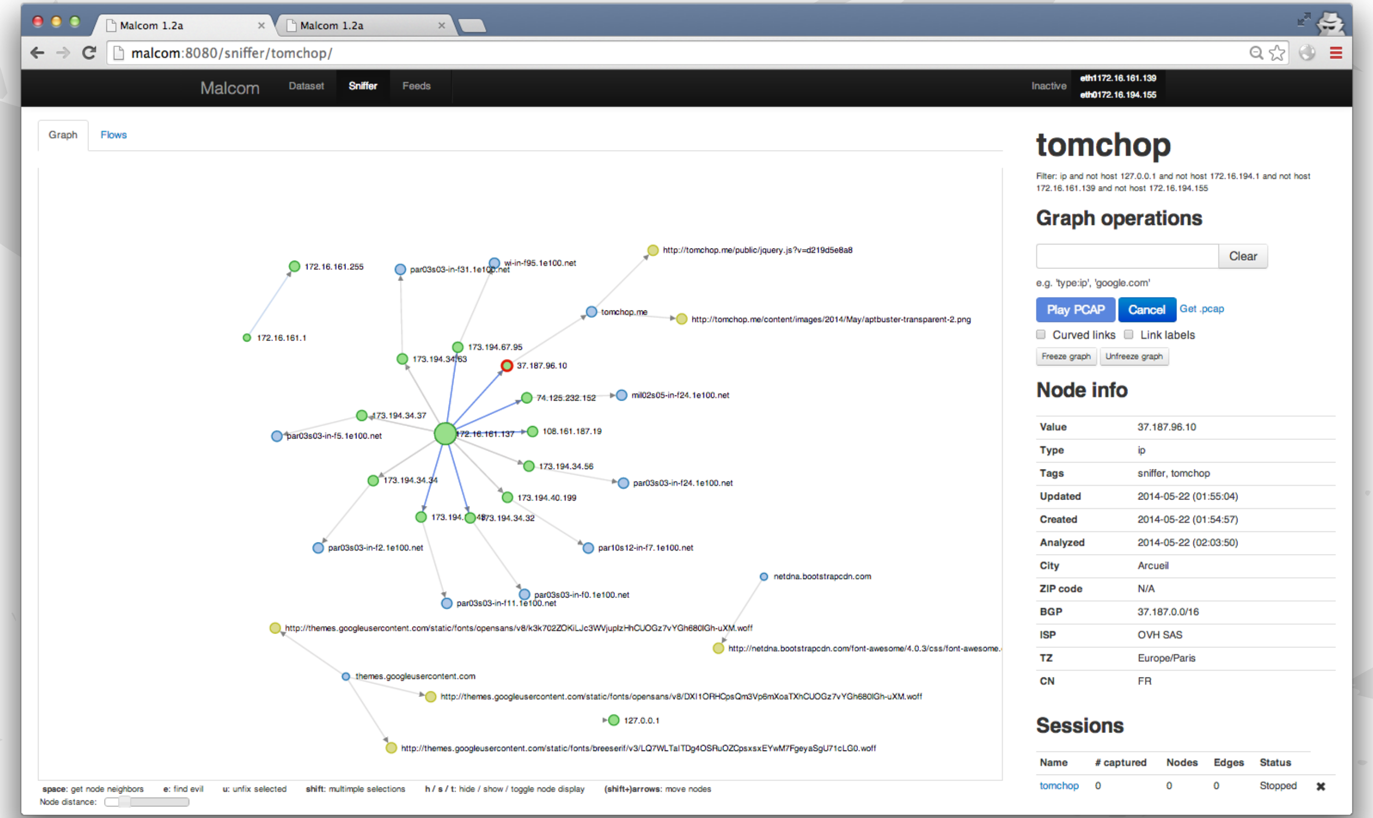
https://github.com/jeffsilverm/dpkt_doc

**jeffsilverm** authored on Jun 2, 2011 lates

| | |
|---|---|
| 4000_byte_ipv6_packet.cap | Some test captures - not all of the them are known to work |
| 60000_byte_ipv6_packet.cap | Some test captures - not all of the them are known to work |
| Python_course_notes-2011-0... | 31-May-2011 class |
| Python_course_notes_spring_... | 31-May-2011 class |
| README | Edited README via GitHub |
| __init__.py | This file makes the directory into a Python package |
| arp2.cap | Some test captures - not all of the them are known to work |
| decode_arp.py | Added some code to illustrate how ethernet addresses and the ARP SHA . |
| decode_dns.py | Adding more functionality and documentation to the DNS module |
| decode_http.py | dpkt_notes.html contains some comments to myself about how dpkt works |
| decode_http_2.py | Removed an addition to the sys.path that is no longer needed |
| decode_tcp.py | I honestly don't remember what I changed |
| decode_tcp_iterator.py | Documentation updates |
| decode_tcp_iterator_2.py | I have to identify the connection that each packet belongs to. I was ... |
| decode_tcp_iterator_2P.py | Removed a bunch of debugging print statements to make the output less... |
| decode_udp.py | fixed a typo in a help string |

dpkt - example code and documentation.

# Network Analysis- Example


Malcom 1.2a
Malcom 1.2a
malcom:8080/sniffer/tomchop/
Malcom
Dataset
Sniffer
Feeds
Inactive
eth1172.16.161.139
eth0172.16.194.155
Graph
Flows
172.16.161.255
172.16.161.1
par03s03-in-f31.1e100.net
wi-in-f95.1e100.net
http://tomchop.me/public/jquery.js?v=d219d5e8a8
tomchop.me
http://tomchop.me/content/images/2014/May/aptbuster-transparent-2.png
173.194.67.95
173.194.34.63
37.187.96.10
74.125.232.152
mil02s05-in-f24.1e100.net
173.194.34.37
par03s03-in-f5.1e100.net
172.16.161.137
108.161.187.19
173.194.34.56
par03s03-in-f24.1e100.net
173.194.34.34
173.194.40.199
173.194.
173.194.34.32
par03s03-in-f2.1e100.net
par10s12-in-f7.1e100.net
netdna.bootstrapcdn.com
par03s03-in-f0.1e100.net
par03s03-in-f11.1e100.net
http://themes.googleusercontent.com/static/fonts/opensans/v8/k3k702ZOKiLJc3WVjuplzHhCUOGz7vYGh680lGh-uXM.woff
http://netdna.bootstrapcdn.com/font-awesome/4.0.3/css/font-awesome.
themes.googleusercontent.com
http://themes.googleusercontent.com/static/fonts/opensans/v8/DXI1ORHCpsQm3Vp6mXoaTXhCUOGz7vYGh680lGh-uXM.woff
127.0.0.1
http://themes.googleusercontent.com/static/fonts/breeserif/v3/LQ7WLTaITDg4OSRuOZCpsxsxEYwM7FgeyaSgU71cLG0.woff

**space**: get node neighbors **e**: find evil **u**: unfix selected **shift**: multimple selections **h / s / t**: hide / show / toggle node display **(shift+)arrows**: move nodes

Node distance:

## tomchop

Filter: ip and not host 127.0.0.1 and not host 172.16.194.1 and not host 172.16.161.139 and not host 172.16.194.155

## Graph operations

Clear

e.g. 'type:ip', 'google.com'

Play PCAP Cancel Get .pcap

□ Curved links □ Link labels

Freeze graph Unfreeze graph

## Node info

| | |
|---|---|
| Value | 37.187.96.10 |
| Type | ip |
| Tags | sniffer, tomchop |
| Updated | 2014-05-22 (01:55:04) |
| Created | 2014-05-22 (01:54:57) |
| Analyzed | 2014-05-22 (02:03:50) |
| City | Arcueil |
| ZIP code | N/A |
| BGP | 37.187.0.0/16 |
| ISP | OVH SAS |
| TZ | Europe/Paris |
| CN | FR |

## Sessions

| Name | # captured | Nodes | Edges | Status | |
|---|---|---|---|---|---|
| tomchop | 0 | 0 | 0 | Stopped | ✖ |

Malcom - graphical view

# Exploring file formats- Tasks

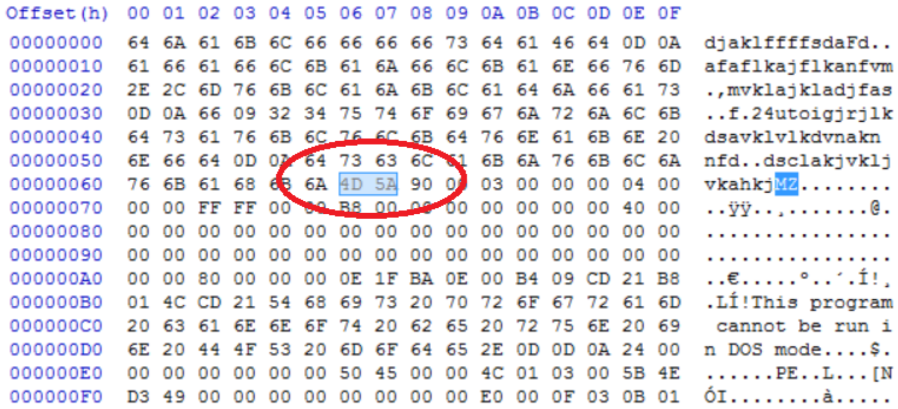
- Carving out embedded files in a data streams.
- Exploring structured data.
- Decompressing files.
    - SWF files are compressed zlib.
- Writing binary parsers
- Analyzing and extracting firmware.

# Exploring file formats- Projects

- pdf-parser.py or peepdf
- oletools or oledump.py
- xxxswf.py
- pe-carv.py
- hachoir
    - hachoir-urwid
    - hachoir-subfile
- construct & vstruct

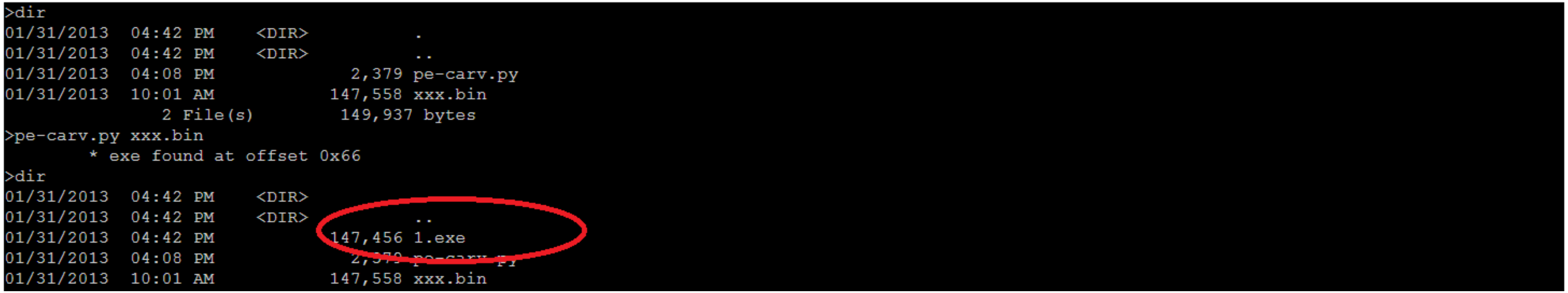
# Exploring file formats- Example

File carving with pe-carv.py



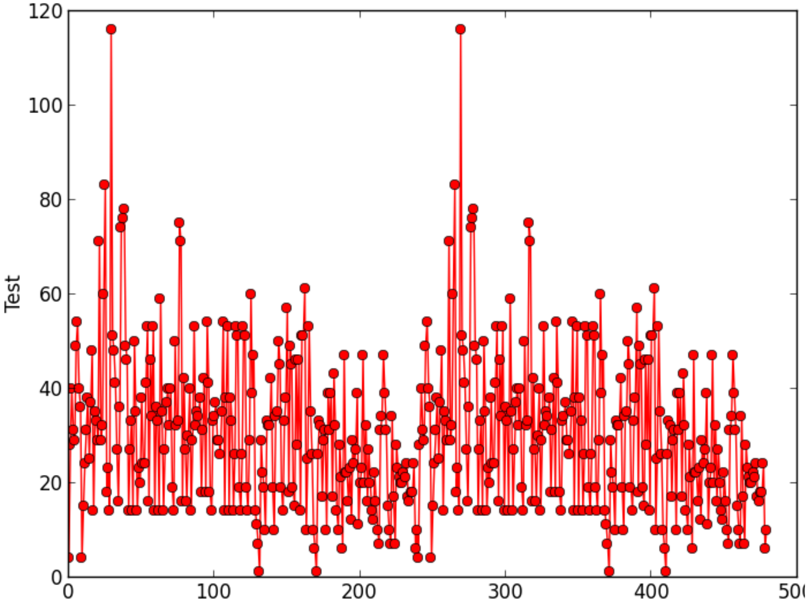
```
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

00000000  64 6A 61 6B 6C 66 66 66 66 73 64 61 46 64 0D 0A  djaklffffsdaFd..
00000010  61 66 61 66 6C 6B 61 6A 66 6C 6B 61 6E 66 76 6D  afaflkajflkanfvm
00000020  2E 2C 6D 76 6B 6C 61 6A 6B 6C 61 64 6A 66 61 73  .,mvklajkladjfas
00000030  0D 0A 66 09 32 34 75 74 6F 69 67 6A 72 6A 6C 6B  ..f.24utoigjrjlk
00000040  64 73 61 76 6B 6C 76 6C 6B 64 76 6E 61 6B 6E 20  dsavklvlkdvnakn
00000050  6E 66 64 0D 0A 64 73 63 6C 61 6B 6A 76 6B 6C 6A  nfd..dsclakjvklj
00000060  76 6B 61 68 6B 6A 4D 5A 90 00 03 00 00 00 04 00  vkahkjMZ........
00000070  00 00 FF FF 00 00 B8 00 00 00 00 00 00 00 40 00  ..ÿÿ..¸.......@.
00000080  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000090  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000000A0  00 00 80 00 00 00 0E 1F BA 0E 00 B4 09 CD 21 B8  ..€.....º..´.Í!¸
000000B0  01 4C CD 21 54 68 69 73 20 70 72 6F 67 72 61 6D  .LÍ!This program
000000C0  20 63 61 6E 6E 6F 74 20 62 65 20 72 75 6E 20 69   cannot be run i
000000D0  6E 20 44 4F 53 20 6D 6F 64 65 2E 0D 0D 0A 24 00  n DOS mode....$.
000000E0  00 00 00 00 00 00 50 45 00 00 4C 01 03 00 5B 4E  ......PE..L...[N
000000F0  D3 49 00 00 00 00 00 00 00 00 E0 00 0F 03 0B 01  ÓI........à.....
```

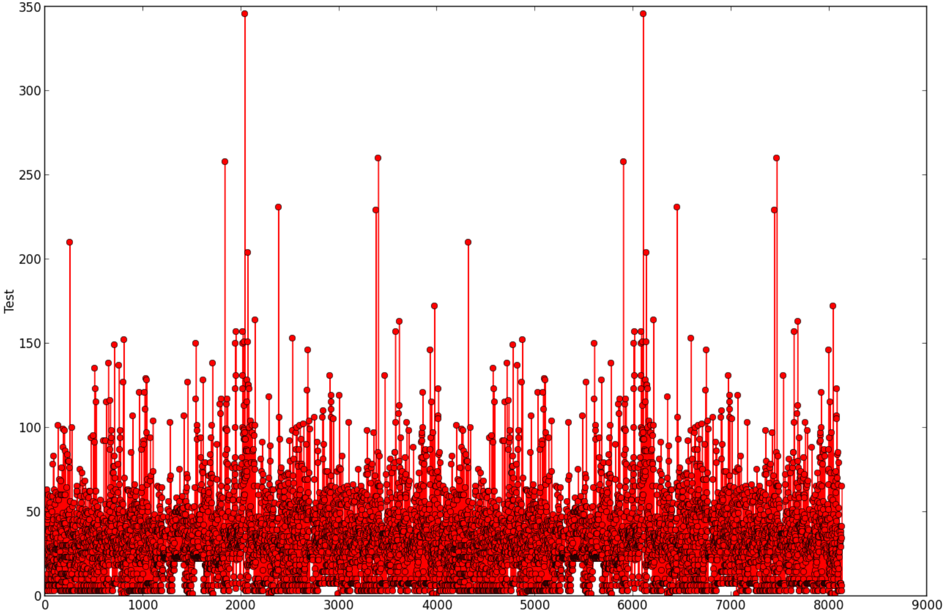
Data Stream with Embedded Portable Executable



```
>dir
01/31/2013  04:42 PM    <DIR>          .
01/31/2013  04:42 PM    <DIR>          ..
01/31/2013  04:08 PM             2,379 pe-carv.py
01/31/2013  10:01 AM           147,558 xxx.bin
               2 File(s)        149,937 bytes
>pe-carv.py xxx.bin
        * exe found at offset 0x66
>dir
01/31/2013  04:42 PM    <DIR>          .
01/31/2013  04:42 PM    <DIR>          ..
01/31/2013  04:42 PM           147,456 1.exe
01/31/2013  04:08 PM             2,379 pe-carv.py
01/31/2013  10:01 AM           147,558 xxx.bin
```

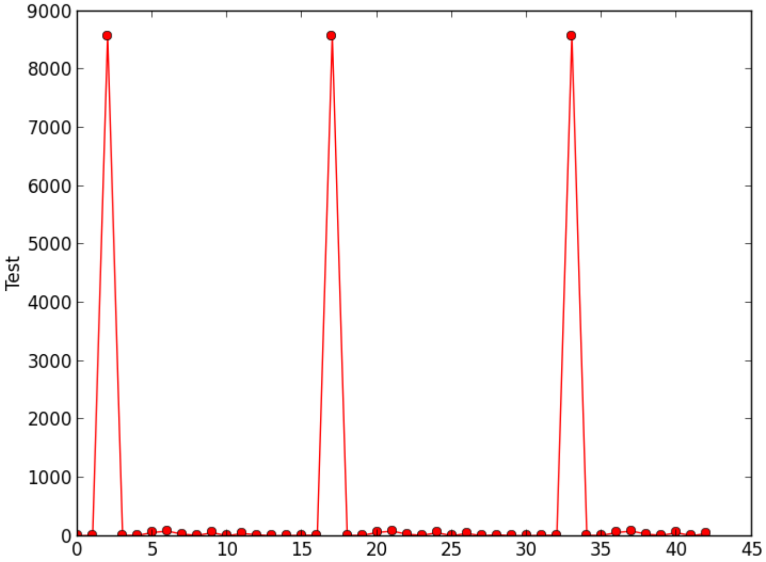
# Exploring file formats- Example

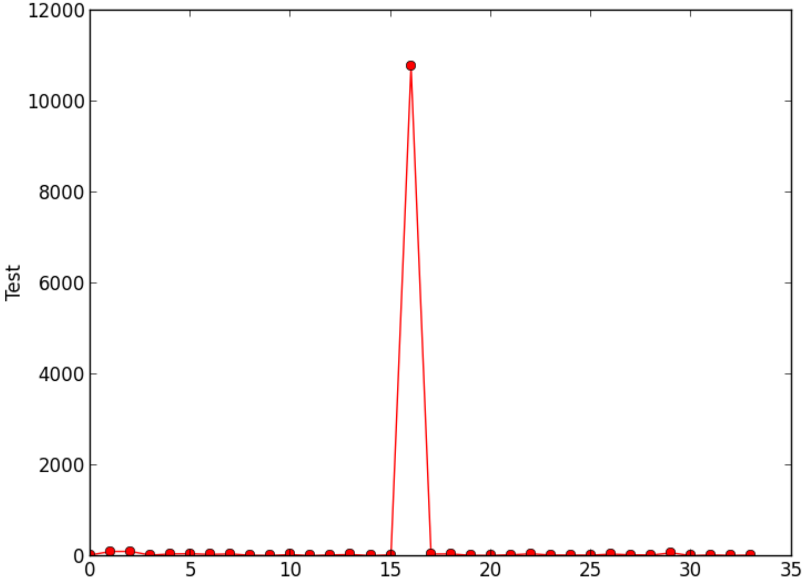




Non-obfuscated JavaScript

# Exploring file formats- Example





Obfuscated JavaScript

# Task -Vulnerability & Exploit Analysis

- Small subset of a very complex area.
- Fuzzing
  - Providing invalid, unexpected or random data as input to see if the data invokes exceptions or crashes
  - Projects
    - Sulley
    - Peach 2
    - python-afl (for fuzzing python code)

# Task -Vulnerability & Exploit Analysis

- Auditing Binaries
    - Scripts in IDAPython commonly used.
    - Simple search for commonly buggy functions
        - Enumerating suspicious function calls (strcpy, fgets, etc)
        - Enumerating file and network input and output
    - Diffing the assembly of patched and vulnerable executables
    - Analyzing data flow and allocation of variables

Links

- Simple Deobfuscation of Code Transformation
- - http://hooked-on-mnemonics.blogspot.com/2012/10/simple-deobfuscation-of-code.html
- Using Z3 to solve a crack me - http://wiremask.eu/hackingweek-2015-reverse-4/
- Deobfuscation: recovering an OLLVM-protected program
- - http://blog.quarkslab.com/deobfuscation-recovering-an-ollvm-protected-program.html
- Breaking Kryptonite's Obfuscation: A Static Analysis Approach Relying on Symbolic Execution - https://doar-e.github.io/blog/2013/09/16/breaking-kryptonites-obfuscation-with-symbolic-execution/
- python-registry Introduction by Willi Ballenthin - https://github.com/williballenthin/python-registry
- Automating DFIR - http://www.hecfblog.com/2015/02/automating-dfir-how-to-series-on.html
- Stuxnet's Footprint in Memory with Volatility 2.0 - http://mnin.blogspot.com/2011/06/examining-stuxnets-footprint-in-memory.html
- Extracting the Powelik's DLL from the Registry - http://sketchymoose.blogspot.com/2015/08/extracting-poweliks-dll-from-registry.html
- ObfStrReplacer - http://hooked-on-mnemonics.blogspot.com/2015/06/obfstrreplacer-extractsubfile-snippets.html
- Cuckoo Sandbox - https://github.com/cuckoobox/cuckoo
- dpkt - https://github.com/jeffsilverm/dpkt_doc
- malcom - https://github.com/tomchop/malcom
- The Very Unofficial Dummies Guide To Scapy - https://theitgeekchronicles.files.wordpress.com/2012/05/scapyguide1.pdf
- Using Python to Fight Cybercrime - https://speakerdeck.com/krmaxwell/using-python-to-fight-cybercrime

Books!

- Hacking Secret Ciphers with Python
- Gray Hat Python: Python Programming for Hackers and Reverse Engineers
- The Art of Memory Forensics: Detecting Malware and Threats in Windows, Linux, and Mac Memory
- Black Hat Python: Python Programming for Hackers and Pentesters
- Python Forensics: A workbench for inventing and sharing digital forensic technology
- The Beginner’s guide to IDAPython :)

# Questions?

*Thanks!*