Advanced Malware Analysis Training Series

Reversing Automation

Harsimran Walia/Amit Malik

www.SecurityXploded.com

Disclaimer

The Content, Demonstration, Source Code and Programs presented here is "AS IS" without any warranty or conditions of any kind. Also the views/ideas/knowledge expressed here are solely of the trainer's only and nothing to do with the company or the organization in which the trainer is currently working.

However in no circumstances neither the Trainer nor SecurityXploded is responsible for any damage or loss caused due to use or misuse of the information presented here.

Acknowledgement

- Special thanks to **Null** community for their extended support and co-operation.
- Special thanks to **ThoughtWorks** for the beautiful venue.
- Thanks to all the trainers who have devoted their precious time and countless hours to make it happen.

Advanced Malware Analysis Training

This presentation is part of our **Advanced Malware Analysis** Training program. Currently it is delivered only during our local meets for FREE of cost.

For complete details of this course, visit our Security Training page.

Who am I?

Harsimran Walia

- Member, SecurityXploded
- Research Scientist, McAfee Labs
- Reversing, Malware Analysis, Exploit Analysis/Development etc.
- Personal site: http://harsimranwalia.info
- E-mail: walia.harsimran@gmail.com
- Twitter: b44nz0r

Content

- Automation
  - Python scripts
  - Use of modules
- Tools/Modules discussed
  - PEfile
  - PyDbg
  - IDAPython

PEfile

- Python module to read and work with Portable Executable (PE) files
- pefile requires understanding of the layout of a PE file (already covered)
- Tasks that *pefile* makes possible are:
  - Modifying and writing back to the PE image
  - Header Inspection
  - Sections analysis
  - Retrieving data
  - Warnings for suspicious and malformed values
  - Packer detection with PEiD's signatures

Pefile (hands-on)

- Load a PE (create an instance)

```
import pefile
pe =  pefile.PE(r'C:\calc.exe')
```

- Reading important PE header attributes

```
pe.OPTIONAL_HEADER.AddressOfEntryPoint
pe.OPTIONAL_HEADER.ImageBase
pe.FILE_HEADER.NumberOfSections
```

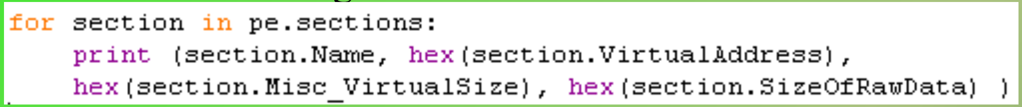
- Modifying values

All PE instance values support assignment followed by a call to write function to write the modified exe to system

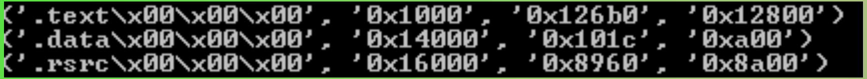
```
pe.OPTIONAL_HEADER.AddressOfEntryPoint = 0xdeadbeef
pe.write(filename=r'C:\calc_modified.exe')
```

Pefile (hands-on)

- PE sections – fetching detail about sections



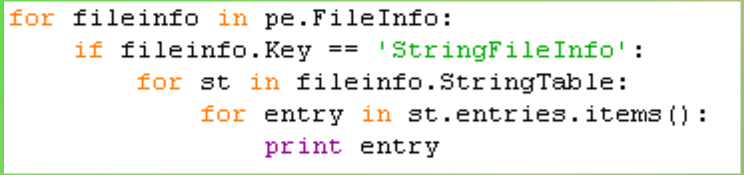
```
for section in pe.sections:
    print (section.Name, hex(section.VirtualAddress),
    hex(section.Misc_VirtualSize), hex(section.SizeOfRawData) )
```



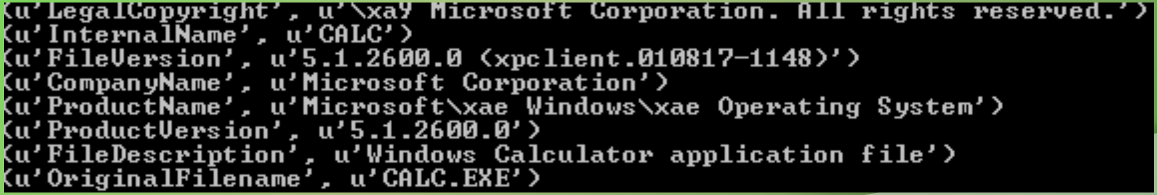
```
('.text\x00\x00\x00', '0x1000', '0x126b0', '0x12800')
('.data\x00\x00\x00', '0x14000', '0x101c', '0xa00')
('.rsrc\x00\x00\x00', '0x16000', '0x8960', '0x8a00')
```

Output

- File Info



```
for fileinfo in pe.FileInfo:
    if fileinfo.Key == 'StringFileInfo':
        for st in fileinfo.StringTable:
            for entry in st.entries.items():
                print entry
```

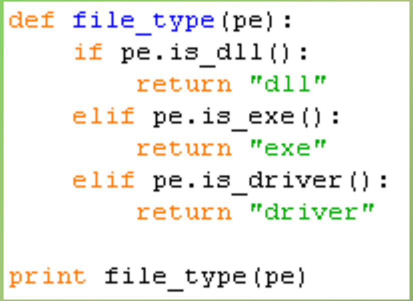


```
(u'LegalCopyright', u'\xa9 Microsoft Corporation. All rights reserved.')
(u'InternalName', u'CALC')
(u'FileVersion', u'5.1.2600.0 (xpclient.010817-1148)')
(u'CompanyName', u'Microsoft Corporation')
(u'ProductName', u'Microsoft\xae Windows\xae Operating System')
(u'ProductVersion', u'5.1.2600.0')
(u'FileDescription', u'Windows Calculator application file')
(u'OriginalFilename', u'CALC.EXE')
```

Output

Pefile (hands-on)

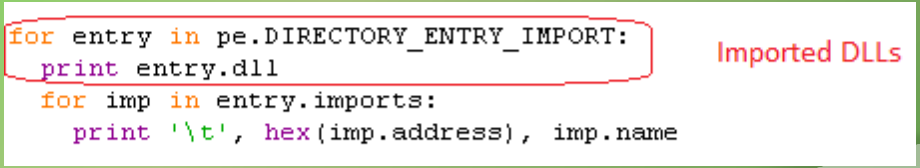
- Type of file (exe/dll/driver)



```
def file_type(pe):
    if pe.is_dll():
        return "dll"
    elif pe.is_exe():
        return "exe"
    elif pe.is_driver():
        return "driver"

print file_type(pe)
```

- List of imported dlls and imported functions



```
for entry in pe.DIRECTORY_ENTRY_IMPORT:
  print entry.dll
  for imp in entry.imports:
    print '\t', hex(imp.address), imp.name
```

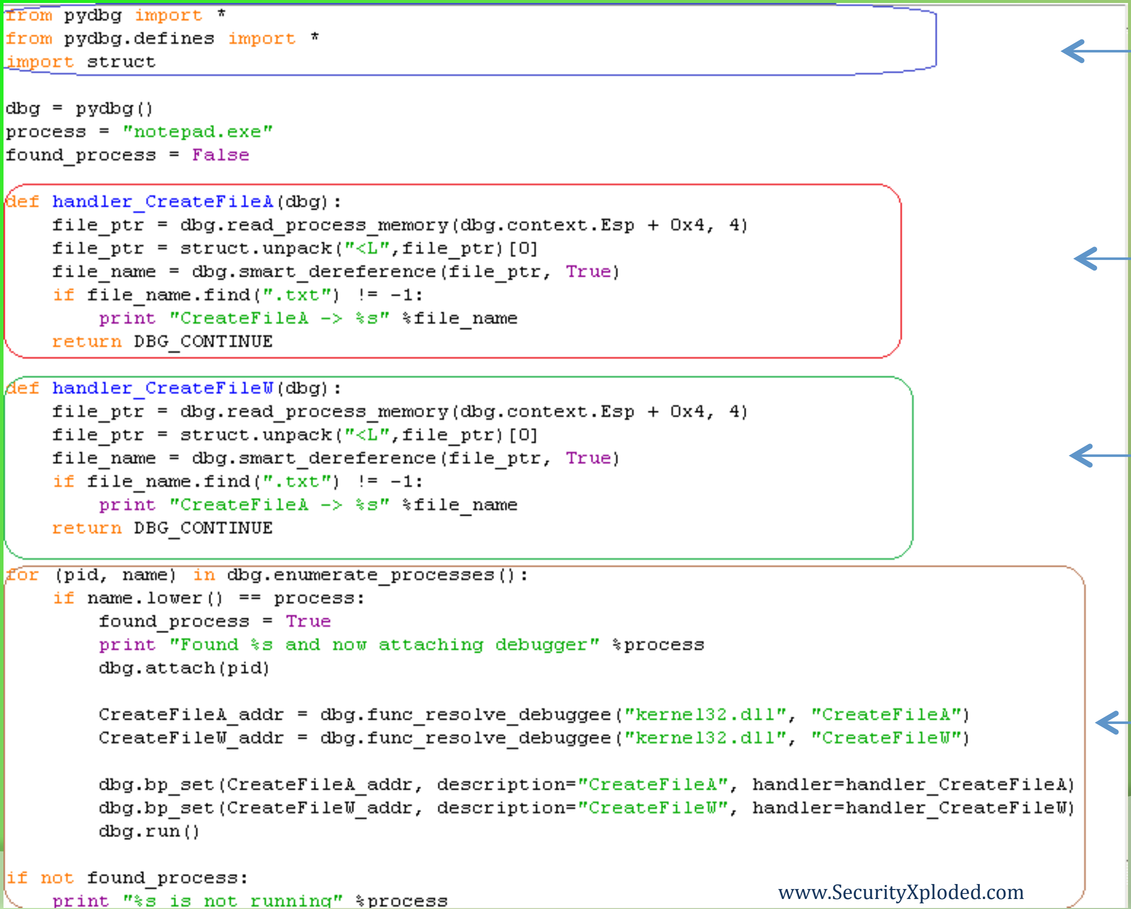
Pydbg

- Open Source Python debugger
- Developed by Pedram Amini as the main component of PaiMei framework
- It uses user-defined callback functions
- These functions can implement actions to take on hitting a breakpoint, exception etc
- Upon execution of the callback function the control is passed back to pydbg to execute the program normally

Pydbg installation

- Download or git clone: https://github.com/OpenRCE/pydbg
- Pre-reqs
  - Python 2.7
  - c-types python library
- Copy the pydbg files to Python-2.7\Lib\site-packages\pydbg
- pydasm.pyd is compiled for Python 2.6, lets fix this!
- Open pydasm.pyd in any hex-editor(010 etc) and search python
  - Change python26.dll to python27.dll
  - Save and replace with original

Pydbg (hands-on)



```
from pydbg import *
from pydbg.defines import *
import struct

dbg = pydbg()
process = "notepad.exe"
found_process = False

def handler_CreateFileA(dbg):
    file_ptr = dbg.read_process_memory(dbg.context.Esp + 0x4, 4)
    file_ptr = struct.unpack("<L",file_ptr)[0]
    file_name = dbg.smart_dereference(file_ptr, True)
    if file_name.find(".txt") != -1:
        print "CreateFileA -> %s" %file_name
    return DBG_CONTINUE

def handler_CreateFileW(dbg):
    file_ptr = dbg.read_process_memory(dbg.context.Esp + 0x4, 4)
    file_ptr = struct.unpack("<L",file_ptr)[0]
    file_name = dbg.smart_dereference(file_ptr, True)
    if file_name.find(".txt") != -1:
        print "CreateFileA -> %s" %file_name
    return DBG_CONTINUE

for (pid, name) in dbg.enumerate_processes():
    if name.lower() == process:
        found_process = True
        print "Found %s and now attaching debugger" %process
        dbg.attach(pid)

        CreateFileA_addr = dbg.func_resolve_debuggee("kernel32.dll", "CreateFileA")
        CreateFileW_addr = dbg.func_resolve_debuggee("kernel32.dll", "CreateFileW")

        dbg.bp_set(CreateFileA_addr, description="CreateFileA", handler=handler_CreateFileA)
        dbg.bp_set(CreateFileW_addr, description="CreateFileW", handler=handler_CreateFileW)
        dbg.run()

if not found_process:
    print "%s is not running" %process
```

Import required pydbg modules and struct

Breakpoint handler for CreateFileA

Extract the parameter from the stack = filename

Breakpoint handler for CreateFileW

- Look for process to debug
- Attach debugger to process
- Set breakpoint on function entry address
- Attach a breakpoint handler

IDA Python

- An IDA Pro plugin
- Integrates Python, allowing scripts to run in IDA Pro
- IDAPython Scripts have access to
  - IDA Plugin API,
  - IDC and all modules available for Python

Installation

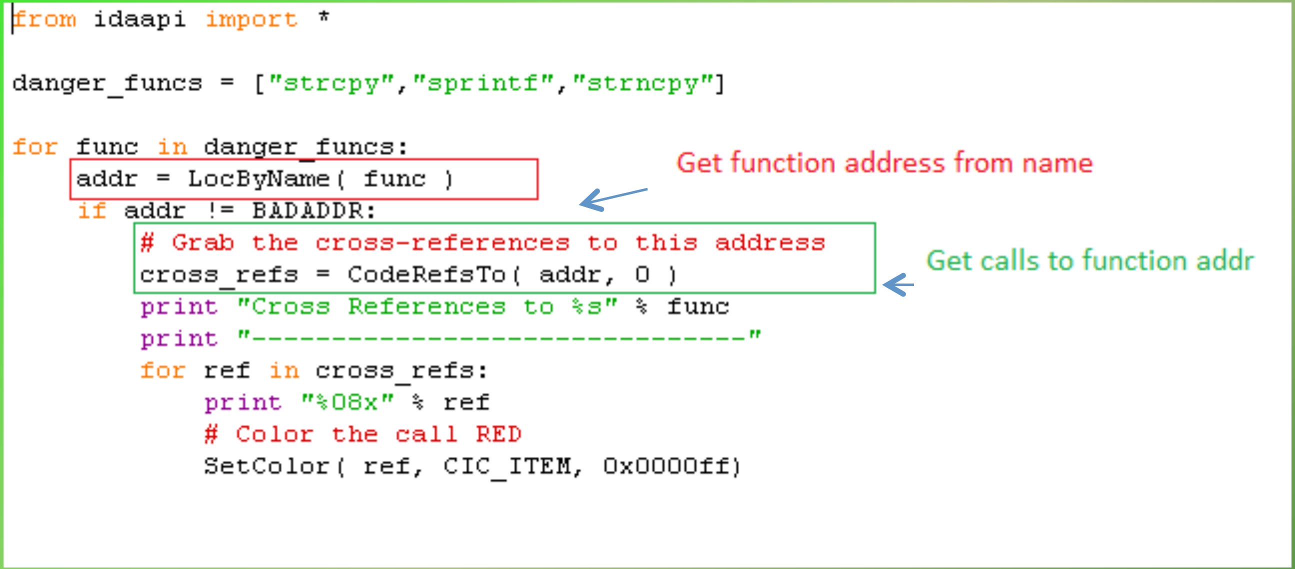
- Download the plugin from https://code.google.com/p/idapython
- Match the IDAPro and python version before downloading
- Copy the "python" directory from the extracted plugin to the IDA Pro install directory (%IDADIR%)
- Copy the plugin executable to "%IDADIR%\plugins\"

Hands-on

- Utility functions
  - **ScreenEA()**
    - Obtains the address of where your cursor is currently positioned on the IDA screen.
  - **GetInputFileMD5()**
    - Returns the MD5 hash of the binary loaded in IDA, which is useful for tracking changes in the binary

- Functions
  - **Functions(long StartAddress, long EndAddress)**
    - Returns a list of all function start addresses contained between StartAddress and EndAddress.
  - **LocByName(string FunctionName)**
    - Returns the address of a function based on its name.
  - **GetFunctionName(long Address)**
    - Given an address, returns the name of the function the address belongs to.

Hands-on



```
from idaapi import *

danger_funcs = ["strcpy","sprintf","strncpy"]

for func in danger_funcs:
    addr = LocByName( func )
    if addr != BADADDR:
        # Grab the cross-references to this address
        cross_refs = CodeRefsTo( addr, 0 )
        print "Cross References to %s" % func
        print "-------------------------------"
        for ref in cross_refs:
            print "%08x" % ref
            # Color the call RED
            SetColor( ref, CIC_ITEM, 0x0000ff)
```

- Try running on *war-ftpd.exe*

Demo..

- ExeScan
  - http://www.securityxploded.com/exe-scan.php

- Malpimp
  - http://www.securityxploded.com/malpimp.php

Reference

Complete Reference Guide for Advanced Malware Analysis Training

[Include links for all the Demos & Tools]

Thank You !

www.SecurityXploded.com