Win32 Static Analysis in Python

Ero Carrera
Sabre Security GmbH

Objective

- Introduce some basic tools for Windows binary inspection.

- These tools allow to automate a large part of some analysis tasks and I'd like to introduce them to a larger audience with the hope that they will find them useful.

pefile

http://dkbza.org/pefile.html

- pefile is a multi-platform full *Python* module intended for handling PE Files. It should be able to process any file that can be open by the Windows loader.
- pefile requires some basic understanding of the layout of a PE file. Armed with it it's possible to explore nearly every single feature of the file.

Introduction

- Loading a PE file
- Inspecting the Headers
- Sections
- Retrieving data
- Data Directories
- Examples

Loading PE file

- Loading a file is as easy as creating an instance of the PE class with the path to the PE file as argument.

  pe = pefile.PE('path/to/file')

- it's also possible to provide the data in a buffer

  pe = pefile.PE(data=python_string)

Inspecting the Headers

```
>>> import pefile
>>> pe = pefile.PE('notepad.exe')

>>> hex(pe.OPTIONAL_HEADER.ImageBase)
['0x1000000L']

>>> hex(pe.OPTIONAL_HEADER.AddressOfEntryPoint)
['0x6AE0L']

>>> hex(pe.OPTIONAL_HEADER.NumberOfRvaAndSizes )
['0x10L']

>>> hex(pe.FILE_HEADER.NumberOfSections)
['0x3']
```

Sections inspection

```
>>> for section in pe.sections:
...  print (section.Name,
        hex(section.VirtualAddress),
        hex(section.Misc_VirtualSize),
        section.SizeOfRawData )
...
('.text', '0x1000L', '0x6D72L', 28160L)
('.data', '0x8000L', '0x1BA8L', 1536L)
('.rsrc', '0xA000L', '0x8948L', 35328L)
```

Is it packed?

```
import math

def H(data):
   if not data:
      return 0

   entropy = 0
   for x in range(256):
      p_x = float(data.count(chr(x)))/len(data)
      if p_x > 0:
         entropy += - p_x*math.log(p_x, 2)

   return entropy
```

Is it packed? (Unpacked)

```
>>> for section in pe.sections:
...  print section.Name, H(section.data)
...
.text 6.28370964662
.data 1.39795676336
.rsrc 5.40687515641
```

Is it packed? (ASPack)

```
>>> pe2 = pefile.PE('notepad-aspack.exe')
>>> for section in pe2.sections:
...  print section.Name, H(section.data)
...
.text 7.98363149339
.data 4.68226874255
.rsrc 6.09026175185
.aspack 5.90609875421
.adata 0
```

Is it packed? (UPX)

```
>>> pe3 = pefile.PE('notepad-upx.exe')
>>> for section in pe3.sections:
...  print section.Name, H(section.data)
...
UPX0 0
UPX1 7.83028313969
.rsrc 5.59212256596
```

Imports

```
>>> for entry in pe.DIRECTORY_ENTRY_IMPORT:
...   print entry.dll
...   for imp in entry.imports:
...     print '\t', hex(imp.address), imp.name
...
comdlg32.dll
        0x10012A0L PageSetupDlgW
        0x10012A4L FindTextW
        0x10012A8L PrintDlgExW
         [snip]
SHELL32.dll
        0x1001154L DragFinish
        0x1001158L DragQueryFileW
```

pydasm

http://dkbza.org/pydasm.html

- pydasm is a multi-platform Python module wrapping jt's libdasm
- pydasm together with pefile provide with a good tool set to develop mini-IDA wannabes

Disassembling

```
>>> import pydasm
>>> i = pydasm.get_instruction('\x90', pydasm.MODE_32)
>>> pydasm.get_instruction_string(
      i, pydasm.FORMAT_INTEL, 0)

['nop ']

>>> i = pydasm.get_instruction(
     '\x8B\x04\xBD\xE8\x90\x00\x01', pydasm.MODE_32)
>>> pydasm.get_instruction_string(
     i, pydasm.FORMAT_INTEL, 0)

['mov eax,[edi*4+0x10090e8]']
```

The Instruction Object

```
>>> pprint.pprint(dir(i))
['__doc__','__module__',
 'dispbytes', 'extindex',
 'flags', 'fpuindex',
 'immbytes', 'length',
 'mode','modrm',
 'op1', 'op2', 'op3',
 'opcode', 'ptr',
 'sectionbytes', 'sib',
 'type']
```

The Operand Object

```
>>> pprint.pprint(dir(i.op1))
['__doc__', '__module__',
 'basereg', 'dispbytes',
 'displacement', 'dispoffset',
 'flags', 'immbytes',
 'immediate', 'immoffset',
 'indexreg', 'reg',
 'scale', 'section',
 'sectionbytes', 'type']
```

pefile+pydasm

```
>>> ep = pe.OPTIONAL_HEADER.AddressOfEntryPoint
>>> ep_ava = ep+pe.OPTIONAL_HEADER.ImageBase
>>> data = pe.get_memory_mapped_image()[ep:ep+100]
>>> offset = 0
>>> while offset < len(data):
...     i = pydasm.get_instruction(
              data[offset:], pydasm.MODE_32)
...     print pydasm.get_instruction_string(
              i, pydasm.FORMAT_INTEL, ep_ava+offset)
...     offset += i.length
```

pefile+pydasm

```
push byte 0x70
push dword 0x1001888
call 0x1006ca8
xor ebx,ebx
push ebx
mov edi,[0x100114c]
call edi
cmp word [eax],0x5a4d
jnz 0x1006b1d
mov ecx,[eax+0x3c]
add ecx,eax
cmp dword [ecx],0x4550
jnz 0x1006b1d
movzx eax,[ecx+0x18
```

IDA in Python?

< 3k lines

Questions?