MazeWalker

Enriching static malware analysis and more

Yevgeny Kulakov
@p_h_0_e_n_i_x
About Me

• Malware RE @ Trusteer, IBM, Seculert
  • binary analysis automation
  • sandbox development
• Now in vEYE Security on software container problems
Agenda

• Malware vs Reverser
• General idea behind MazeWalker Tool
• How and What MazeWalker solves
• Demo
• Future work
Malware vs Reverser

Prevent or slowdown manual analysis

Make me suffer
Some examples of annoying behaviour
Code (un)packing

- New executable areas introduced
- Runtime code change
- Stack-based execution
Code (un)packing - PiC

- Runtime CF change - Indirect Calls & Jumps

```assembly
  call    eax
  inc     ecx
  call    eax
  xchg    ecx, edx
  neg     edx
  jmp     short loc_2D531F3
```
Environment Detection

- Anti-VMs
  - API based
    - device enumeration
    - api monitoring detection (cuckooobox hooks)
  - ASM based
    - elapsed time diff
Code dispersion

- Hard to follow - several debug sessions
- Attaching debugger may freeze the UI

Sample
operational deployment

svchost.exe
watchdog

eplorer.exe
CnC, rootkit

Kernel
Permission
Elevation
Obfuscate at rest

- Encrypt all the things - cfg, code, etc
- Obfuscate API calling or resolve it on each API call
- Own API resolution - use own DLLs copies
- Abuse asm and mix code with data
No Run No Fun
A word on code amount
There is a lot of code

- Malware is taken as a serious software project
  - release cycles, test labs, dev teams
  - copy & paste from other malware projects too
<table>
<thead>
<tr>
<th>Folder</th>
<th>Description</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>BC</td>
<td>import Carberp Botnet</td>
<td>a year ago</td>
</tr>
<tr>
<td>BJWJ</td>
<td>import Carberp Botnet</td>
<td>a year ago</td>
</tr>
<tr>
<td>BSS</td>
<td>import Carberp Botnet</td>
<td>a year ago</td>
</tr>
<tr>
<td>BinToHex</td>
<td>Import Carberp Botnet</td>
<td>a year ago</td>
</tr>
<tr>
<td>BlackJoeWhiteJoe</td>
<td>import Carberp Botnet</td>
<td>a year ago</td>
</tr>
<tr>
<td>BootkitDropper</td>
<td>import Carberp Botnet</td>
<td>a year ago</td>
</tr>
<tr>
<td>Demo_Cur2</td>
<td>import Carberp Botnet</td>
<td>a year ago</td>
</tr>
<tr>
<td>Demo_Cur3</td>
<td>import Carberp Botnet</td>
<td>a year ago</td>
</tr>
<tr>
<td>Demo_cur</td>
<td>import Carberp Botnet</td>
<td>a year ago</td>
</tr>
<tr>
<td>DillLoaderHook</td>
<td>import Carberp Botnet</td>
<td>a year ago</td>
</tr>
<tr>
<td>DillLoaderHook1</td>
<td>import Carberp Botnet</td>
<td>a year ago</td>
</tr>
<tr>
<td>DropSploit</td>
<td>import Carberp Botnet</td>
<td>a year ago</td>
</tr>
<tr>
<td>DropSploit1/src</td>
<td>import Carberp Botnet</td>
<td>a year ago</td>
</tr>
<tr>
<td>FakeDllAutorun</td>
<td>import Carberp Botnet</td>
<td>a year ago</td>
</tr>
<tr>
<td>GrabberIE_FF</td>
<td>import Carberp Botnet</td>
<td>a year ago</td>
</tr>
<tr>
<td>InjectDLL</td>
<td>import Carberp Botnet</td>
<td>a year ago</td>
</tr>
<tr>
<td>Locker</td>
<td>import Carberp Botnet</td>
<td>a year ago</td>
</tr>
<tr>
<td>Mini</td>
<td>import Carberp Botnet</td>
<td>a year ago</td>
</tr>
<tr>
<td>NodInject</td>
<td>import Carberp Botnet</td>
<td>a year ago</td>
</tr>
<tr>
<td>OCR</td>
<td>import Carberp Botnet</td>
<td>a year ago</td>
</tr>
</tbody>
</table>
## Gozi

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Added Gozi/ISFB Source</td>
<td></td>
</tr>
<tr>
<td>AcDll</td>
<td>Added Gozi/ISFB Source</td>
<td>a year ago</td>
</tr>
<tr>
<td>BcClient</td>
<td>Added Gozi/ISFB Source</td>
<td>a year ago</td>
</tr>
<tr>
<td>Builder</td>
<td>Added Gozi/ISFB Source</td>
<td>a year ago</td>
</tr>
<tr>
<td>Common</td>
<td>Added Gozi/ISFB Source</td>
<td>a year ago</td>
</tr>
<tr>
<td>Handle</td>
<td>Added Gozi/ISFB Source</td>
<td>a year ago</td>
</tr>
<tr>
<td>KeyLog</td>
<td>Added Gozi/ISFB Source</td>
<td>a year ago</td>
</tr>
<tr>
<td>Lib32</td>
<td>Added Gozi/ISFB Source</td>
<td>a year ago</td>
</tr>
<tr>
<td>Lib64</td>
<td>Added Gozi/ISFB Source</td>
<td>a year ago</td>
</tr>
<tr>
<td>Rsakey</td>
<td>Added Gozi/ISFB Source</td>
<td>a year ago</td>
</tr>
<tr>
<td>SocksLib</td>
<td>Added Gozi/ISFB Source</td>
<td>a year ago</td>
</tr>
<tr>
<td>ZipLib</td>
<td>Added Gozi/ISFB Source</td>
<td>a year ago</td>
</tr>
<tr>
<td>apdepack</td>
<td>Added Gozi/ISFB Source</td>
<td>a year ago</td>
</tr>
<tr>
<td>client</td>
<td>Added Gozi/ISFB Source</td>
<td>a year ago</td>
</tr>
<tr>
<td>crypto</td>
<td>Added Gozi/ISFB Source</td>
<td>a year ago</td>
</tr>
<tr>
<td>cryptor</td>
<td>Added Gozi/ISFB Source</td>
<td>a year ago</td>
</tr>
<tr>
<td>dname</td>
<td>Added Gozi/ISFB Source</td>
<td>a year ago</td>
</tr>
<tr>
<td>release(builder)</td>
<td>Added Gozi/ISFB Source</td>
<td>a year ago</td>
</tr>
<tr>
<td>x64/release(builder)</td>
<td>Added Gozi/ISFB Source</td>
<td>a year ago</td>
</tr>
<tr>
<td>zconv</td>
<td>Added Gozi/ISFB Source</td>
<td>a year ago</td>
</tr>
<tr>
<td>Config.exe</td>
<td>Added Gozi/ISFB Source</td>
<td>a year ago</td>
</tr>
</tbody>
</table>
There is a lot of code (cont)

- Culminates in large codebase over time
- Takes substantial amount of time to analyze
Time is Money
both are at most insufficient
Ideas behind MazeWalker
**MazeWalker - Main Ideas**

- It must save time !!!!
- Maximize time spent in IDA vs time in Debugger
- Work with non modified VMs
- Retrieve all runtime info and push into IDA
- Help with overall malware understanding
  - dig into asm on an interest - basis
  - enable research focusing
Architecture

MazeWalker Tool

PinTool
- Memory Track
- Python Engine
- Code Analysis

IDA Plugin
Intel’s Pin Framework

Pin is a dynamic binary instrumentation framework for the IA-32, x86-64 and MIC instruction-set architectures that enables the creation of dynamic program analysis tools.

- Callbacks on everything
  - instructions
  - API calls
  - Image loading
  - Threads, Exceptions
  - memory read/writes

- VM in essence
- Multi-platform
Code unpacking - memory

- Rely on allocated page analysis
- Tracks all executed memory by comparing executing BBL to older copy
  - detect new PEs
  - identify known (dynamically) loaded DLLs

```json
"whitelist": {
  "imphash": [
    {
      "name": "wow64cpu.dll",
      "hash": "99760ef4e9750fe74c20aa23cc71b9b6"
    },
    {
      "name": "kerne132.dll",
      "hash": "51d53c5e0b0dd0eb29d977440ba62d9"
    },
    {
      "name": "cryptsp.dll",
      "hash": "ebc7b47d85441b0f3dce38e782316e8c"
    },
    {
      "name": "advapi32.dll",
      "hash": "56357721d7fd0b68c7be9d465e71475"
    }
  ],
  "exphash": [
    {
      "name": "ntdll.dll",
      "hash": "302ce81fcc0c08531dd6637cd5e81f"
    },
    {
      "name": "ntdll.dll",
      "hash": "4a40a87fd83bebb5f83fd4e5be70262e"
    }
  ],
  "path": ["C:\\Windows\"]
}
```
Code unpacking - PiC

- Pin helps to do Call/Jump site analysis
- Logging call-site <-> target pair

```assembly
    call    eax        ; GetProcAddress
    inc     ecx
    call    eax        ; RtlDecompressBuffer
    xchg    ecx, edx
    neg     edx
    jmp     short loc_2D531F3
```
System API monitoring

• Pin’s Routine Objects
  • Harder to detect
• Configurable
• API Agnostic monitor interface
• Scriptable
System APIs - CreateFileW

```python
import ctypes
import json

def pre_analyzer(LPCTSTR lpFileName,
                 DWORD dwDesiredAccess,
                 DWORD dwShareMode,
                 LPSECURITY_ATTRIBUTES lpSecurityAttributes,
                 DWORD dwCreationDisposition,
                 DWORD dwFlagsAndAttributes,
                 HANDLE hTemplateFile,
                 **kwargs):

    FileName = ctypes.c_wchar_p.from_address(LPCTSTR lpFileName)
    res = []
    if (FileName and FileName.value):
        result = {'name': 'lpFileName', 'data': FileName.value}
        res.append(result)
    return json.dumps(res)
```
Environment Detection

```python
def post_analyzer(HDEVINFO_DeviceInfoSet, 
PSP_DEVINFO_DATA_DeviceInfoData, 
pProperty, 
PDWORD_PropertyRegDataType, 
PBYTE_PropertyBuffer, 
pPropertyBufferSize, 
PDWORD_RequiredSize, 
**kwargs):
    Property = ctypes.c_ulong.from_address(pProperty)
    if (Property.value == 0x0C):
        PropertyBufferSize = ctypes.c_ulong.from_address(pPropertyBufferSize)
        if (PropertyBufferSize.value > 0):
            res = []
            pPropertyBuffer = ctypes.c_ulong.from_address(PBYTE_PropertyBuffer)
            PropertyBuffer = ctypes.cast(pPropertyBuffer.value, ctypes.c_char_p)
            buffer = (c_char * PropertyBufferSize.value).from_address(pPropertyBuffer.value)
            res.append({'name': 'PropertyBufferSize', 'data': PropertyBufferSize.value})
            res.append({'name': 'original_PropertyBuffer', 'data': PropertyBuffer.value})
            replace_string(buffer, PropertyBuffer, ['vmware', 'virtual'], ['b'NewTek', 'b'Digital'])
            res.append({'name': 'fixed_PropertyBuffer', 'data': PropertyBuffer.value})
    return json.dumps(res)
return None
```
MazeWalker

- Code Unpacking
- Memory Tracking
- Position Independent Code
- System APIs
- Environment Detection
- Code Dispersion
Code dispersion

- Use scriptable APIs monitoring for code injection tracking
  - this helps Pin to find target process
- Use Pin’s existing ability to track child processes
def pre_analyzer(dwDesiredAccess, bInheritHandle, dwProcessId, **kwargs):
    pid = ctypes.c_int.from_address(dwProcessId)
    if (pid and pid.value and os.getpid() != pid.value):
        if "pin_dir" in kwargs and "out_dir" in kwargs:
            process = subprocess.Popen(kwargs["pin_dir"] + "/pin.exe -unique_logfile -pid " + str(pid.value) + " -t " + kwargs["pin_dir"] + "/MazeWalker.dll -cfg " + kwargs["pin_dir"] + "/config.json" + " -out " + kwargs["out_dir"] + " -unique_logfile")
    res = []
    result = {
        'name': 'dwProcessId',
        'data': pid.value
    }
    res.append(result)
    return json.dumps(res)
Control Flow Graph

With PIN’s BBL callbacks
Covers all memory regions
Covers across different processes
Threads everywhere

All execution metadata is on thread basis
Demo
<table>
<thead>
<tr>
<th>Name</th>
<th>PID</th>
<th>CPU</th>
<th>I/O total</th>
<th>Private bytes</th>
<th>User name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>System Idle Process</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>NT AUTHORITY\SYSTEM</td>
<td>NT Kernel &amp; System</td>
</tr>
<tr>
<td>System</td>
<td>4</td>
<td>49.22</td>
<td>0</td>
<td>0</td>
<td>NT AUTHORITY\SYSTEM</td>
<td>Windows NT Session Manager</td>
</tr>
<tr>
<td>smss.exe</td>
<td>540</td>
<td>180</td>
<td>1.89 MB</td>
<td>0</td>
<td>NT AUTHORITY\SYSTEM</td>
<td>Client Server Runtime Process</td>
</tr>
<tr>
<td>csrss.exe</td>
<td>604</td>
<td>1.89</td>
<td>0</td>
<td>0</td>
<td>NT AUTHORITY\SYSTEM</td>
<td>Windows NT Logon Application</td>
</tr>
<tr>
<td>winlogon.exe</td>
<td>628</td>
<td>7.25</td>
<td>0</td>
<td>0</td>
<td>NT AUTHORITY\SYSTEM</td>
<td>Services and Controller app</td>
</tr>
<tr>
<td>services.exe</td>
<td>672</td>
<td>1.67</td>
<td>0</td>
<td>0</td>
<td>NT AUTHORITY\SYSTEM</td>
<td>VMware Activation Helper</td>
</tr>
<tr>
<td>vmmcthlp.exe</td>
<td>844</td>
<td>564</td>
<td>0</td>
<td>0</td>
<td>NT AUTHORITY\SYSTEM</td>
<td>Generic Host Process for Win32</td>
</tr>
<tr>
<td>svchost.exe</td>
<td>860</td>
<td>2.95</td>
<td>0</td>
<td>0</td>
<td>NT AUTHORITY\SYSTEM</td>
<td>Windows Security Center Notify</td>
</tr>
<tr>
<td>wmiiprsvs...</td>
<td>1496</td>
<td>3.18</td>
<td>0</td>
<td>0</td>
<td>NT AUTHORITY\SYSTEM</td>
<td>WMI</td>
</tr>
<tr>
<td>svchost.exe</td>
<td>928</td>
<td>1.76</td>
<td>0</td>
<td>0</td>
<td>NT AUTHORITY\SYSTEM</td>
<td>Generic Host Process for Win32</td>
</tr>
<tr>
<td>svchost.exe</td>
<td>972</td>
<td>14.71</td>
<td>0</td>
<td>0</td>
<td>NT AUTHORITY\SYSTEM</td>
<td>Generic Host Process for Win32</td>
</tr>
<tr>
<td>wsntfy,...</td>
<td>268</td>
<td>456</td>
<td>0</td>
<td>0</td>
<td>TOM-S22\Administrator</td>
<td>Windows Security Center Notify</td>
</tr>
<tr>
<td>cmd.exe</td>
<td>136</td>
<td>1.07</td>
<td>0</td>
<td>0</td>
<td>NT AUTHORITY\LOCAL SERVICE</td>
<td>Application Layer Gateway Service</td>
</tr>
<tr>
<td>lsass.exe</td>
<td>684</td>
<td>3.8</td>
<td>0</td>
<td>0</td>
<td>NT AUTHORITY\SYSTEM</td>
<td>LSA Shell (Export Version)</td>
</tr>
<tr>
<td>vmtoolsd.exe</td>
<td>256</td>
<td>760</td>
<td>18.4 MB</td>
<td>0</td>
<td>TOM-S22\Administrator</td>
<td>VMware Tools Core Service</td>
</tr>
<tr>
<td>Process Hacker.exe</td>
<td>688</td>
<td>0.78</td>
<td>16.44 MB</td>
<td>0</td>
<td>TOM-S22\Administrator</td>
<td>Process Hacker</td>
</tr>
<tr>
<td>explorer.exe</td>
<td>1680</td>
<td>10.63</td>
<td>0</td>
<td>0</td>
<td>TOM-S22\Administrator</td>
<td>Windows Explorer</td>
</tr>
<tr>
<td>cmd.exe</td>
<td>1732</td>
<td>1.94</td>
<td>0</td>
<td>0</td>
<td>TOM-S22\Administrator</td>
<td>Windows Command Processor</td>
</tr>
<tr>
<td>pin.exe</td>
<td>2120</td>
<td>608</td>
<td>0</td>
<td>0</td>
<td>TOM-S22\Administrator</td>
<td>Pin Executable</td>
</tr>
<tr>
<td>sample.exe</td>
<td>1196</td>
<td>50.00</td>
<td>39.04 MB</td>
<td>0</td>
<td>TOM-S22\Administrator</td>
<td>Windows Command Processor</td>
</tr>
<tr>
<td>cmd.exe</td>
<td>1972</td>
<td>2.18</td>
<td>0</td>
<td>0</td>
<td>TOM-S22\Administrator</td>
<td>Windows Command Processor</td>
</tr>
</tbody>
</table>
Collected Data

```
[{
    "process": {
        "name": "",
        "pid": 1196,
        "threads_num": 2,
        "threads": [
            {
                "tid": 0,
                "tfunc": 4572402,
                "bbs": [],
                "calls": [],
                "api_parameters": []
            }
        ],
        "mem_areas": [
            {
                "id": 0,
                "start": 4194304,
                "end": 7255112,
                "size": 3060808,
                "entry": 4572402,
                "tids": []
            }
        ]
    }
}]
```
Hierarchy matters

Navigate the execution flow

Original IDA

Maze Walker
Hierarchy matters

Wrapped functions get different meaning with context
Focus

Work on Memory
Part Only
Focussing on Registry only
ToDo...
Further development

• Stability and Memory consumption reduction
• Memory dumps consolidation
• Custom IDA Loader
• “Maze Walk” in kernel space
• Implement anti-instrumentation prevention logic
  • Dynamic Binary Instrumentation Frameworks: I know you’re there spying on me (ReCon 2012)
Thank you!

@p_h_0_e_n_i_x