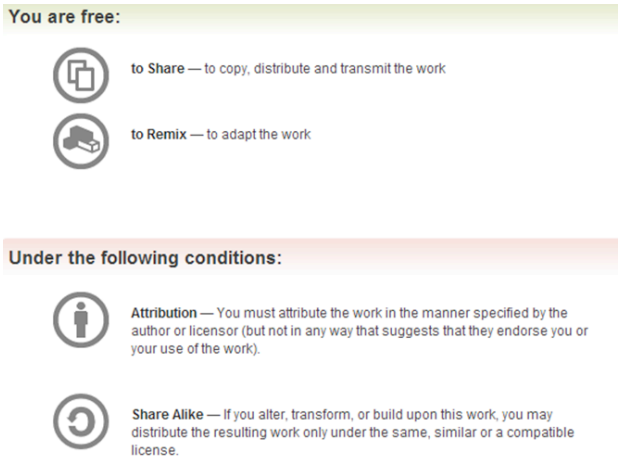
Malware Dynamic Analysis

Part 4

Veronica Kovah
vkovah.ost at gmail

All materials is licensed under a Creative Commons “Share Alike” license

http://creativecommons.org/licenses/by-sa/3.0/



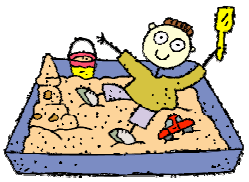
Outline

- Part 3
  - Malware functionality
    - Keylogging, Phone home, Security degrading, Self-destruction, etc.
- Part 4
  - **Using an all-in-one sandbox – Cuckoo Sandbox**
  - **Malware Attribute Enumeration and Characterization (MAEC)**
  - Actionable output
    - Detection – Snort and Yara

See notes for citation

Malware Analysis Sandbox

- Provides file system, registry keys, and network traffic monitoring in controlled environment and produces a well formed report
- Using a sandbox is more efficient and sometimes more effective
- Configure your own sandbox such as Joebox, GFI Sandbox, and Cuckoo Sandbox.
- Use public sandbox such as ThreatExpert, GFI ThreatTrack, and Anubis
  - Do not submit malware to a public sandbox if it reveals sensitive information about your organization and/or customer.



See notes for citation

[References]

• Joe Sandbox, http://www.joesecurity.org/index.php/joe-sandbox-standalone
• GFI Sandbox, http://www.gfi.com/malware-analysis-tool
• Cuckoo Sandbox, http://www.cuckoosandbox.org
• ThreatExpert, http://www.threatexpert.com/submit.aspx
• GFI ThreaetTrack, http://www.threattrack.com/
• Anubis, http://anubis.iseclab.org/

[Image Sources]

• http://plannerwire.net/wp-content/uploads/2011/02/Playing-Sandbox_meeting_planners.gif

Cuckoo Sandbox

- Open source automated malware analysis system
- At the time v1.0
- Analyzes PE, PDF, MS Office, PHP scripts, etc.
- Outputs JSON/HTML/MAEC reports
- Customization
  - Machinery Modules: virtualization software
  - Analysis Package: how to conduct the analysis procedure
  - Processing Modules: how to analyze raw results
  - Signatures
  - Reporting Modules
  - Auxiliary Modules: to be executed in parallel to every analysis

See notes for citation

[References]

• Cuckoo Sandbox Book, http://docs.cuckoosandbox.org/en/latest

[Image Sources]

• http://www.cuckoosandbox.org/graphic/cuckoo.png

Poison Ivy

- Open three terminals
- #1 terminal, run inetsim
  - $ sudo inetsim
- #2 terminal, run Cuckoo Sandbox
  - $ cd ~/MalwareClass/tools/cuckoo
  - Edit conf/auxiliary.conf (to sniff on vboxnet1)
  - $ python ./cuckoo.py
- #3 submit piagent.exe to Cuckoo
  - $ cd ~/MalwareClass/tools/cuckoo/utils
  - $ python ./submit.py ~/MalwareClass/samples/PoisonIvy/piagent.exe

See notes for citation

Results

- Task results are generated under {Cuckoo Root}/storage/analysis/[task number]/
  - {Cuckoo Root} = ~/MalwareClass/tools/cuckoo
  - *reports* directory includes reports in different formats
  - *logs* directory includes raw data named <process id>.bson
  - *shots* directory includes screen shots
  - *files* directory includes dropped files. You can then run dropped executables through on their own
- Submitted sample will be copied to {Cuckoo Root}/storage/binaries/MD5NAME, where MD5NAME is the md5 of the submitted sample
  - A symbolic link (named *binary*) exists under the task result directory

Poison Ivy Results

- $ cd ~/MalwareClass/tools/cuckoo/storage/analysis/1/reports
- $ firefox report.html &
- $ gedit report.json &
- $ firefox report.maec-4.0.1.xml &

See notes for citation

MAEC (Malware Attribute Enumeration and Characterization)

- “a standardized language for encoding and communicating high-fidelity information about malware based upon attributes such as behaviors, artifacts, and attack patterns”

  https://maec.mitre.org/about/index.html

- A standard is necessary to provide a common way to share malware analysis results among organizations to avoid duplicate, inaccurate work

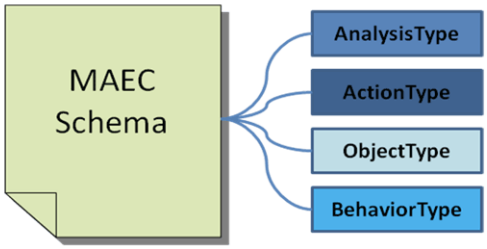
See notes for citation

[References]

• MAEC, https://maec.mitre.org

MAEC (Malware Attribute Enumeration and Characterization)

- Standard communication method among
  - human ↔ human
  - human ↔ tool
  - tool ↔ tool
- MAEC Schema
  - Defines syntax



Key MAEC Schema Components

http://maec.mitre.org/language/schema.html

- Would be very useful to search openmalware.org samples based on attributes, could make a new search engine: “Ask MAEC!”

See notes for citation

[References]

• Ivan Kirillov et al. Malware Attrribute Enumeration and Characterization, https://maec.mitre.org/about/docs/Introduction_to_MAEC_white_paper.pdf

[Image Sources]

• https://maec.mitre.org/images/schema.gif

Parite

- Submit parite sample to Cuckoo Sandbox
  - $ cd ~/MalwareClass/tools/cuckoo/utils
  - $ python submit.py ~/MalwareClass/samples/parite/malware.exe

Q1. Does this drop files with randomized names?

Q2. How does it maneuver?

Q3. How does it persist?

Q4. Does it have self-avoidance?

Q5. Does it self-destruct?

Q6. Where does it try to connect to?

See notes for citation

Answers for Parite Lab

A1. Yes

- C:\DOCUME~1\student\LOCALS~1\Temp\?ta1.tmp

A2. OpenProcess (PID=1760) → VirtualAllocEx → NtWriteVirtualMemory → CreateRemoteThread

- Now you are interested in the process name of PID 1760 :D

Answers for Parite Lab

A3. Set a registry value Software\Microsoft\Windows\CurrentVersion\RUN\fmsiocps
  - we don't know exact data in the value based on cuckoo result

A4. Yes, mutex “Residented” is created

A5. Not explicitly

A6. No explicit network activity, we don't know if the malware is waiting for an event or just sleeping

See notes for citation

Nitol

- Submit nitol sample to Cuckoo Sandbox
  - $ cd ~/MalwareClass/tools/cuckoo/utils
  - $ python submit.py ~/MalwareClass/samples/nitol/malware.exe

Q1. Does this drop files with randomized names?

Q2. How does it maneuver?

Q3. How does it persist?

Q4. Does it have self-avoidance?

Q5. Does it do self-destruction?

Q6. Where does it try to connect to?

See notes for citation

Answers for Nitol

A1. Yes
- One file name is random:
  - C:\WINDOWS\system32\??????.exe

A2. We cannot identify maneuvering technique from the cuckoo's result
- SetWindowsHookEx? Nope, the hooks (WH_MSGFILTER (-1) and WH_CBT (5)) are for its own process

A3. Registered an auto-start service
- HKLM\System\CurrentControlSet\Services\Distribuijq
- CreateSerivce(), StartService() API calls

See notes for citation

Answers for Nitol

A4. No, false positive
- ShimCacheMutex is opened by side effect

A5. Yes, it moves itself to
- C:\DOCUME~1\student\LOCALS~1\Temp\SOFTWARE.LOG

A6. tutwl.3322.org
- Microsoft took down the entire 3322.org (google “Operation b70”) but they came back online after agreeing to clean out malware users

See notes for citation

IMworm

- Submit nitol sample to Cuckoo Sandbox
  - $ cd ~/MalwareClass/tools/cuckoo/utils
  - $ python submit.py ~/MalwareClass/samples/IMworm/malware.exe

Q1. Does this drop files with randomized names?

Q2. What's the file's original name?

Q3. How does it persist?

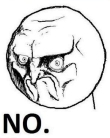
Q4. Does it have self-avoidance?

Q5. Does it do self-destruction?

Q6. Where does it try to connect to?

See notes for citation

Answers for IMworm



A1. **NO.**

A2. worm2007.exe

A3. Answer for persistence

- Copy itself to a start up directory
  - Copy to C:\Windows\lsass.exe then create a process
  - C: \Document and Settings\All Users\Start Menu\Programs\Startup\MSconfig.exe

See notes for citation

[Image Sources]

• http://i0.kym-cdn.com/entries/icons/original/000/007/423/untitle.JPG

Answers for IMworm

- Sets registry value:
  - SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\Userinit = ? (maybe, maybe not)
  - NOTE: It's not setting Run registry key

A4. No, false positive

- ShimCacheMutex is opened by side effect

A5. No apparent self-destruction

A6. Tried to get http://quicknews.info/YMWorm.exe

See notes for citation

Outline

- Part 3
  - Malware functionality
    - Keylogging, Phone home, Security degrading, Self-destruction, etc.
- Part 4
  - Using an all-in-one sandbox – Cuckoo Sandbox
  - Malware Attribute Enumeration and Characterization (MAEC)
  - **Actionable output**
    - **Detection – Snort and Yara**

See notes for citation

Yara

- Open source tool to identify and classify malicious files based on textual or binary patterns
- Light-weight way of performing signature checks
- Can be used for any binary data (exe, pdf, pcaps, etc)
- Useful in an email server for tip-offs, and filtering

See notes for citation

[References]

• yara-project, http://code.google.com/p/yara-project/

Yara Signature

```
rule silent_banker : banker
{
    meta:
        description = "This is just an example"
        thread_level = 3
        in_the_wild = true

    strings:
        $a = {6A 40 68 00 30 00 00 6A 14 8D 91}
        $b = {8D 4D B0 2B C1 83 C0 27 99 6A 4E 59 F7
F9}
        $c = "UVODFRYSIHLNWPEJXQZAKCBGMT"

    condition:
        $a or $b or $c
}
```

http://code.google.com/p/yara-project/

See notes for citation

Yara Signature

- Identifier
  - Any alphanumeric characters and underscores but cannot start with a number
- String definition
  - A string identifier starts with $ followed by alphanumeric character and underscores
  - Values
    - Text strings enclosed by double quotes
    - Hex strings enclosed by curly brackets
    - Regular expression enclosed by slashes

See notes for citation

[References]

• Víctor Manuel Álvarez, YARA User's Manual 1.6, http://code.google.com/p/yara-project/downloads/detail?name=YARA%20User%27s%20Manual%201.6.pdf

Yara Signature

- Condition
  - Boolean operators
    - and, or, not
  - Relational operators
    - >=, <=, <, >, ==, !=
  - Arithmetic and bitwise operators
    - +, -, *, /, &, |, <<, >>, ~
- Counting strings

```
strings:
        $a = “text”
condition:
        #a == 6
```

Bot classification

- We will make a yara signature for a bot malware in this lab
- Identify characteristic strings from the agobot sample
  - $ strings ~/MalwareClass/samples/agobot/malware.exe > /tmp/agobot.txt
- Make an yara signature using combination of the identified strings
  - Create a file (e.g. detection.yar) for the signature
- To run yara
  - $ yara detection.yar ~/Malware/samples/agobot/malware.exe

See notes for citation

One possible answer

```
rule Agobot
{
   strings:
      $msg = "PhatBNC" nocase
      $conf1 = "ddos_maxthreads"
      $conf2 = "scan_maxsockets"
      $conf3 = "scan_maxthreads"
      $cmd1 = "do_stealth"
      $cmd2 = "do_avkill"
      $cmd3 = "do_speedtest"
      $cmd4 = "bot_topiccmd"
      $cmd5 = "bot_meltserver"
      $cmd6 = "bot_randnick"
   condition:
      (#msg > 10) and $conf1 and $conf2 and $conf3
     and (any of ($cmd1, $cmd2, $cmd3, $cmd4, $cmd5, $cmd6))
}
```

See notes for citation

Snort

- Open source network intrusion detection/prevention tool (NIDS/NIPS)
- 3 modes
  - Sniffer: read packets off the network and display on the screen
  - Packet Logger: logs the packets to a log file
  - NIDS: analyze network traffic and match with user-defined signatures and make actions (e.g. alert, drop, etc.)

See notes for citation

[References]

•Snort, http://www.snort.org/
•Snort Users Manual 2.9.4, http://s3.amazonaws.com/snort-org/www/assets/166/snort_manual.pdf

[Image Sources]

•http://4.bp.blogspot.com/_2IvFH57W8Hc/TPfpzDtwQwI/AAAAAAAAAFk/YFngxr8jLgI/s1600/snort_large.gif

Snort

- Preprocessors provides various pre-detection processing
  - Frag3: IP defragmentation
  - Stream5: TCP/UDP session tracking
  - RPC decode: RPC record defragmentation
  - HTTP Inspect: HTTP fields identification, normalization etc.
- A preprocessor may depends on the other
- Supports custom preprocessor

See notes for citation

Snort Signatures

- Detection can be implemented in preprocessor, Snort (text) rules, or SO (shared object) rules.
- Snort rules

SRC IP PORT DEST IP PORT

alert tcp any any -> any 80 (msg:"No deadbeef"; content:"DEADBEEF";)

- Rule headers
  - Rule action tells Snort what to do (e.g. alert, log, drop)
  - IP addresses in Classless Inter-Domain Routing (CIDR) notation
  - Port numbers
  - Direction operator should be "->" or "<>" (bidirectional)

See notes for citation

[References]

• Pre-Compile SO Rules: Supported Platforms, https://www.snort.org/snort-rules/shared-object-rule

Snort Signatures

- Rule options
  - Separated by semicolon (;)
  - msg: message to be displayed in log
  - content: ascii string or binary to match
  - content modifiers
    - nocase, depth, offset, distance, within, http_header, http_client_body, http_uri, file_data
  - pcre: match can be written in perl compatible regular expression
  - flags: checks TCP flag bit

Detect Beaconing Traffic

- We will write a NDIS signature for this lab on the host machine

  $ wireshark ~/Malware/misc/darkshell.pcap &

- Lab is already configured
  - Fixed the permission violation error
    $ sudo usermod -aG snort student
  - Set HOME_NET to 192.168.57.0/24 in /etc/snort/snort.conf
- Let's run Snort with the existing Snort rules
  - $ snort -c /etc/snort/snort.conf -r ~/Malware/misc/darkshell.pcap -l /tmp

Detect Beaconing Traffic

- Open a new file to write a Snort rule
- You can start with the following template and fill up detection rule options
  - alert tcp any any -> any any (<your rule options here>)
- To test your rule
  - $ snort -c <rule file path> -r <pcap file path> -l /tmp

See notes for citation

Phone Home Format

```
// Darkshell bot-to-CnC comms
struct {
  // Header:
  DWORD   dwMagic;    // always 0x00000010 for Darkshell
  // Obfuscated section:
  char    szComputerName[64]; // Name of infected host, NULL-terminated/extended
  char    szMemory[32];    // Amount of memory in infected host; format "%dMB"; NULL-
terminated/extended
  char    szWindowsVersion[32];   // Specifies version of Windows; one of: Windows98, Windows95,
                                  // WindowsNT, Windows2000, WindowsXP, Windows2003, or Win Vista;
                                  // NULL-terminated/extended
  char    szBotVersion[32];   // Specifies version of bot; NULL-terminated/extended;
  DWORD   szUnknown1[4];     // ??? - Always NULL-terminated 'n'
  // Binary section:
  char    szPadding1[32];  // Filled with 0x00 bytes
  WORD    wUnknown2;  // ??? - We have seen 0x00A0, 0x00B0, and 0x00C0
  WORD    wUnknown3;  // ??? - Always 0xFD7F
  char    szPadding2[20];  // Filled with 0x00 bytes
  WORD    wUnknown4;  // ??? - Always 0xB0FC
  BYTE    cUnknown5;  // ??? - We have seen 0xD6, 0xD7, 0xE6, 0xE7, and 0xF1
  BYTE    cZero;      // Always 0x00
  DWORD   dwSignature[8]; // Always 0x00000000, 0xFFFFFFFF, 0x18EE907C, 0x008E917C,
                          //       0xFFFFFFFF, 0xFA8D91&C, 0x25D6907C, 0xCFEA907C
};
```

http://ddos.arbornetworks.com/2011/01/darkshell-a-ddos-bot-targetting-vendors-of-industrial-food-processing-equipment/

What We Learned in Part 1

- Background concepts & tools
  - PE files, Windows Libraries, Processes, Threads, Registry, Windows Services,
  - TrID, Process Explorer, Process Monitor, PsServices, Wireshark, CFF Explorer
- Observing an isolated malware analysis lab setup
  - Ubuntu, Virtualbox, inetsim
- Malware terminology

See notes for citation

What We Learned in Part 2

- RAT exploration - Poison IVY
  - Server and client
- Persistence techniques
  - Registry, File system
  - Autoruns, Regshot
- Maneuvering techniques
  (How malware strategically positions itself)
  - Code and DLL injection, DLL search order hijacking, IAT, EAT, and inline hooking
  - Procmon, WinApiOverride, Winobj

See notes for citation

What We Learned in Part 3

- Malware functionality
  - Key logging
  - Phone home
  - Beaconing
  - Self-Avoidance
  - Security degrading
  - Simple stealth techniques (non-rootkit techniques)
    - Self-destruction
    - Hiding files

See notes for citation

What We Learned in Part 4

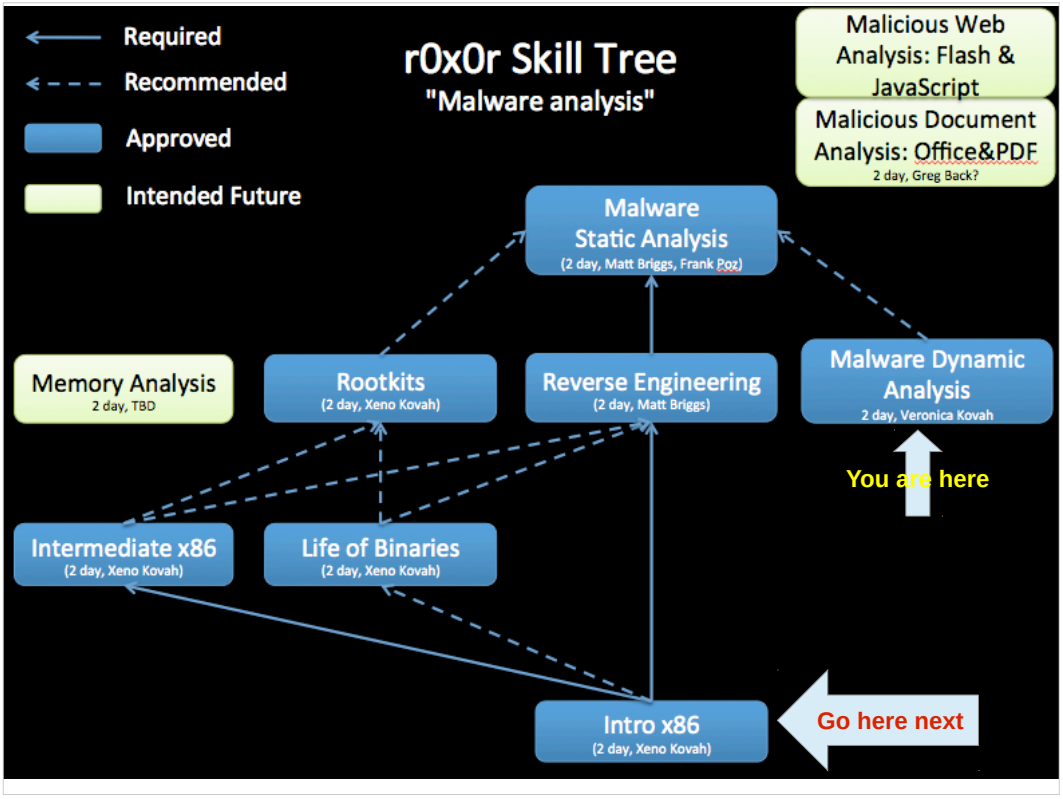
- Using an all-in-one sandbox – Cuckoo Sandbox
  - Good for automation and the first cut
- Malware Attribute Enumeration and Characterization (MAEC)
- Actionable output – detection signatures
  - Snort: network intrusion detection/prevention system
  - Yara: Malware identification and classification tool

See notes for citation

All samples are from openmalware.org

- 101d00e77b48685bc02c1ff9672e1e94 eldorado/malware.exe
- 9250281b5a781edb9b683534f8916392 agobot/malware.exe
- 3349eab5cc4660bafa502f7565ff761d conficker/malware.exe
- 9f880ac607cbd7cdfffa609c5883c708 Hydraq/malware.exe
- a10b9b75e8c7db665cfd7947e93b999b parite/malware.exe
- d7578e550c0a4d4aca0cfd01ae19a331 spyeye/malware.exe
- df150905e2537db936ef323f48e2c1bb magania/malware.exe
- 4a29d41dfda9cfcbcde4d42b4bbb00aa Darkshell/malware.exe
- 1a36fb10f0a6474a9fea23ee4139d13e nitol/malware.exe
- db19c23c5f77a697500075c790cd331c IMworm/malware.exe
- a9a2fb545068995f30df22f8a3f22a10 onlinegames/2/malware.exe
- f1bae35d296930d2076b9d84ba0c95ea onlinegames/1/malware.exe

See notes for citation


Required
Recommended
Approved
Intended Future
r0x0r Skill Tree
"Malware analysis"
Malicious Web Analysis: Flash & JavaScript
Malicious Document Analysis: Office&PDF
2 day, Greg Back?
Malware Static Analysis
(2 day, Matt Briggs, Frank Poz)
Memory Analysis
2 day, TBD
Rootkits
(2 day, Xeno Kovah)
Reverse Engineering
(2 day, Matt Briggs)
Malware Dynamic Analysis
2 day, Veronica Kovah
You are here
Intermediate x86
(2 day, Xeno Kovah)
Life of Binaries
(2 day, Xeno Kovah)
Intro x86
(2 day, Xeno Kovah)
Go here next

The End