Malware Dynamic Analysis
Part 6

Veronica Kovah
vkovah.ost at gmail

http://opensecuritytraining.info/MalwareDynamicAnalysis.html
All materials is licensed under a Creative Commons “Share Alike” license

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

You are free:
- to Share — to copy, distribute and transmit the work
- to Remix — to adapt the work

Under the following conditions:
- Attribution — You must attribute the work in the manner specified by the author or licensor (but not in any way that suggests that they endorse you or your use of the work).
- Share Alike — If you alter, transform, or build upon this work, you may distribute the resulting work only under the same, similar or a compatible license.
Where are we at?

• Part 5: Using an all-in-one sandbox
  – Cuckoo Sandbox
  – Malware Attribute Enumeration and Characterization (MAEC)
  – Different sandbox results comparison
• Part 6: Actionable output
  – Yara
  – Snort
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

[References]
• yara-project, http://code.google.com/p/yara-project/
Yara Signature (1)

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/
Yara Signature (2)

- 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

References
Yara Signature (3)

• Condition operators
  – Boolean
    • and, or, not
  – Relational
    • >=, <=, <, >, ==, !=
  – Arithmetic
    • +, -, *, /
  – Bitwise
    • &, |, <<, >>, ~

• Counting strings
  – strings:
    • $a = “text”
  – condition:
    • #a == 6
Bot classification

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

rule Agobot
{
    strings:
        $msg = "PhatBNC" nocode
        $conf1 = "ddos_maxthreads"
        $conf2 = "scan_maxsockets"
        $conf3 = "scan_maxthreads"
        $cmd1 = "do_stealth"
        $cmd2 = "do_avkill"
        $cmd3 = "do_speedtest"
        $cmd4 = "bot_topiccmd"
        $cmd5 = "bot_meltsserver"
        $cmd6 = "bot_randnick"
    condition:
        ($msg > 10) and $conf1 and $conf2 and $conf3
        and (any of ($cmd1, $cmd2, $cmd3, $cmd4, $cmd5, $cmd6))
}
Where are we at?

- Part 5: Using an all-in-one sandbox
  - Cuckoo Sandbox
  - Malware Attribute Enumeration and Characterization (MAEC)
  - Different sandbox results comparison
- Part 6: Actionable output
  - Yara
  - Snort
Snort (1)

- 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.)

[References]

[Image Sources]
- http://4.bp.blogspot.com/_2lVFH57W8Hc/TPfpzDtwQwI/AAAAAAAAAFk/YFngxr8jLgl/s1600/snort_large.gif
Snort (2)

- 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
Snort Signatures (1)

- 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)

[References]
Snort Signatures (2)

- 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
  - sid: required field, Snort rule identifier
Detect Beaconing Traffic (1)

- We will write a NIDS signature for this lab on the host machine
  - `$ wireshark ~/MalwareClass/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
    ~/MalwareClass/misc/darkshell.pcap -l /tmp`

See notes for citation
Detect Beaconing Traffic (2)

- Open a new file to write a Snort rule
- You can start with the following template and fill up detection rule options
  ```plaintext
  alert tcp any any -> any any ( <your rule options here> )
  ```
- To test your rule
  ```bash
  $ snort -c <rule file path> -r <pcap file path> -l /tmp
  ```
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 szBotVersion[32]; // Specifies version of bot; NULL-terminated/extended;
    DWORD szUnknown[4]; // ??? - Always NULL-terminated “n”
    // Binary section:
    char szPadding[1][32]; // Filled with 0x00 bytes
    WORD wUnknown1; // ??? - We have seen 0x00A0, 0x00B0, and 0x00C0
    WORD wUnknown3; // ??? - Always 0xFD7F
    char szPadding[1][20]; // Filled with 0x00 bytes
    WORD wUnknown2; // ??? - Always 0x0B0F
    WORD wUnknown5; // ??? - We have seen 0x06, 0x07, 0x06, 0x07, and 0x01
    BYTE dZero; // Always 0x00
    DWORD dwSignature[8]; // Always 0x00000000, 0xFFFFFFF, 0x18EE9017C, 0x0008E917C,
    // 0xFFFFFFF, 0xA8D916C, 0x25D6907C, 0xC8EA907C
};

What we learned in Part 1

• How an isolated malware analysis lab is setup
  – Ubuntu, Virtualbox, inetsim
• Malware terminology
  – Bot, RAT, etc.
  – Heterogeneous vendor naming
• RAT exploration - Poison IVY
  – Implant and Controller
• Behavioral malware analysis approaches
  – Diffing, monitoring, API tracing, etc.
What we learned in Part 2

- Background concepts
  - PE files, Windows Libraries, Processes, Registry, Windows Services
  - TrID, Process Explorer, Process Monitor, PsService, CFF Explorer
- Persistence techniques
  - Registry, File system, Windows services
  - Autoruns, Regshot
What we learned in Part 3

• Background concepts
  – API, Threads

• Maneuvering techniques
  (How malware strategically positions itself to access critical resources)
  – DLL and code injection, DLL search order hijacking, IAT, EAT, and inline hooking
  – Procmon, WinApiOverride, Winobj
What we learned in Part 4

• Background concepts
  – How to analyze network traffic with Wireshark

• Malware functionality
  – Key logging
  – Phone home
  – Beaconing
  – Self-Avoidance
  – Security degrading
  – Simple stealth techniques (non-rootkit techniques)
    • Self-destruction
    • Hiding files
What we learned in Part 5/6

• Using an all-in-one sandbox
  — Good for automation and the first cut
  — How to use Cuckoo Sandbox
  — How to analyze sandboxes’ results
  — Malware Attribute Enumeration and Characterization (MAEC)

• Actionable output – detection signatures
  — Snort: network intrusion detection/prevention system
  — Yara: Malware identification and classification tool
All samples are from openmalware.org

- 101d00e77b48685bc02c1ff9672e1e94 eldorado/malware.exe
- 9250281b5a781edbb9b683534f8916392 agobot/malware.exe
- 3349e5cb4660bafa502f7565ff7d1d conficker/malware.exe
- 9f880ac607cbbfd5f609c5883c708 Hydraq/malware.exe
- a10b9b75e8c7db665cf7947e93b999b parite/malware.exe
- d7578e550c0a4d4aca0cfd01ae19a331 spyeye/malware.exe
- df150905e2537db936ef323f48e2c1bb magania/malware.exe
- 4a29d41dfda9cfbcde4d42b4bb00aa Darkshell/malware.exe
- 1a36fb10f0a6474a9f0a23ee4139d13e nitol/malware.exe
- db19c23c5f77a69750075c790cd331c IMworm/malware.exe
- a9a2f54506899f30df22f8a3f22a10 onlinegames/2/malware.exe
- fb1ae35d296930d2076b9d84ba0c95ea onlinegames/1/malware.exe
The End