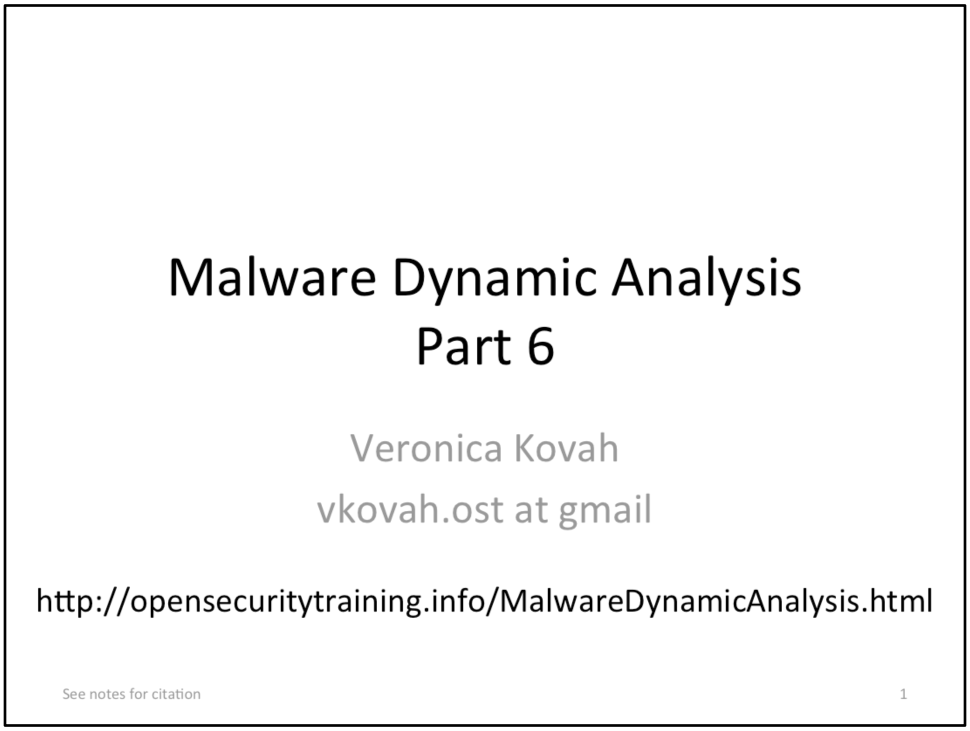


Malware Dynamic Analysis Part 6

Veronica Kovah

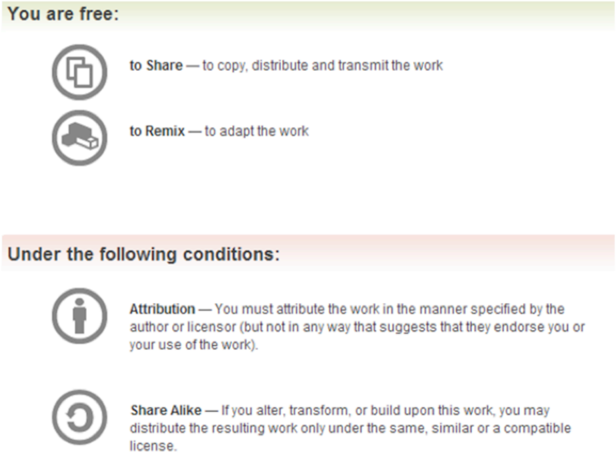
vkovah.ost at gmail

http://opensecuritytraining.info/MalwareDynamicAnalysis.html

See notes for citation

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

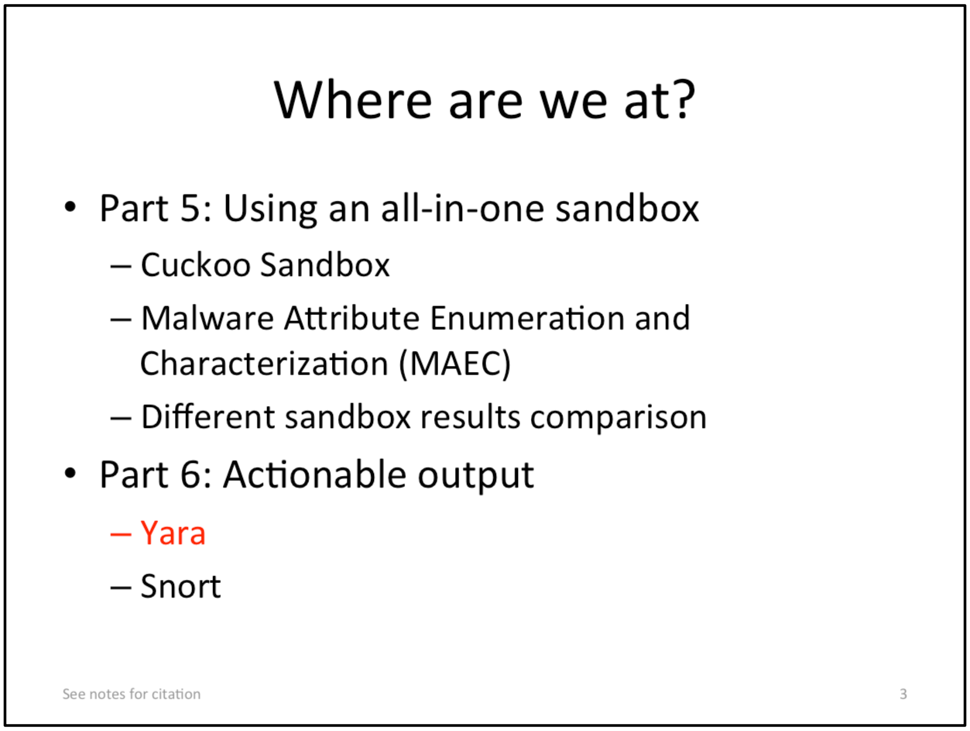
to Remix — to adapt the work

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).

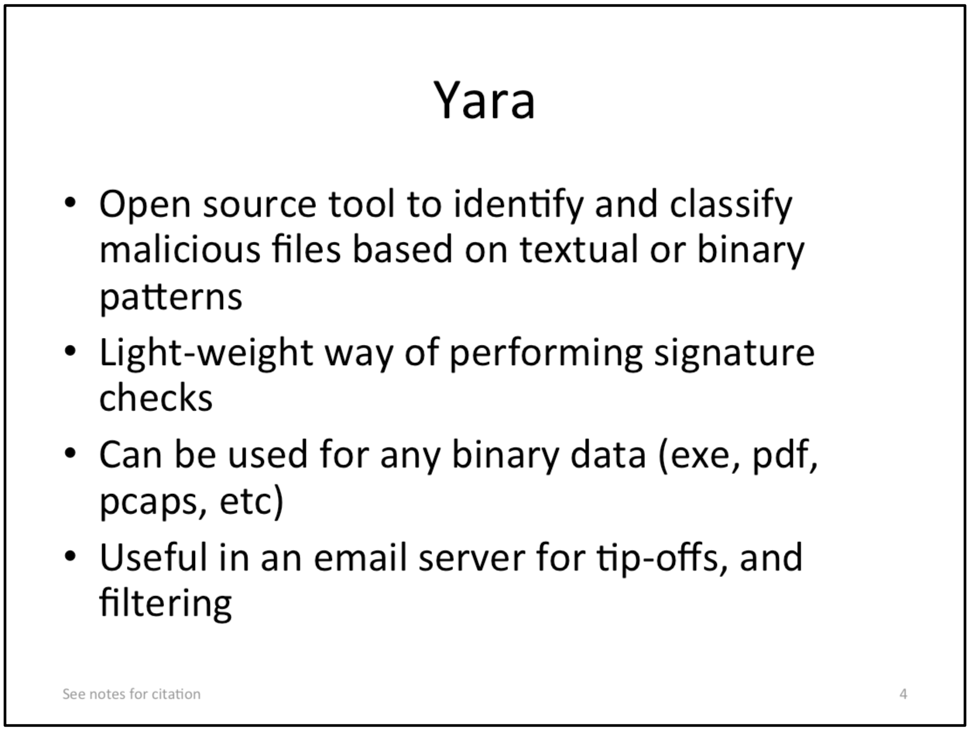
Share Alike — If you alter, transform, or build upon this work, you may distribute the resulting work only under the same, similar or a compatible license.

See notes for citation



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

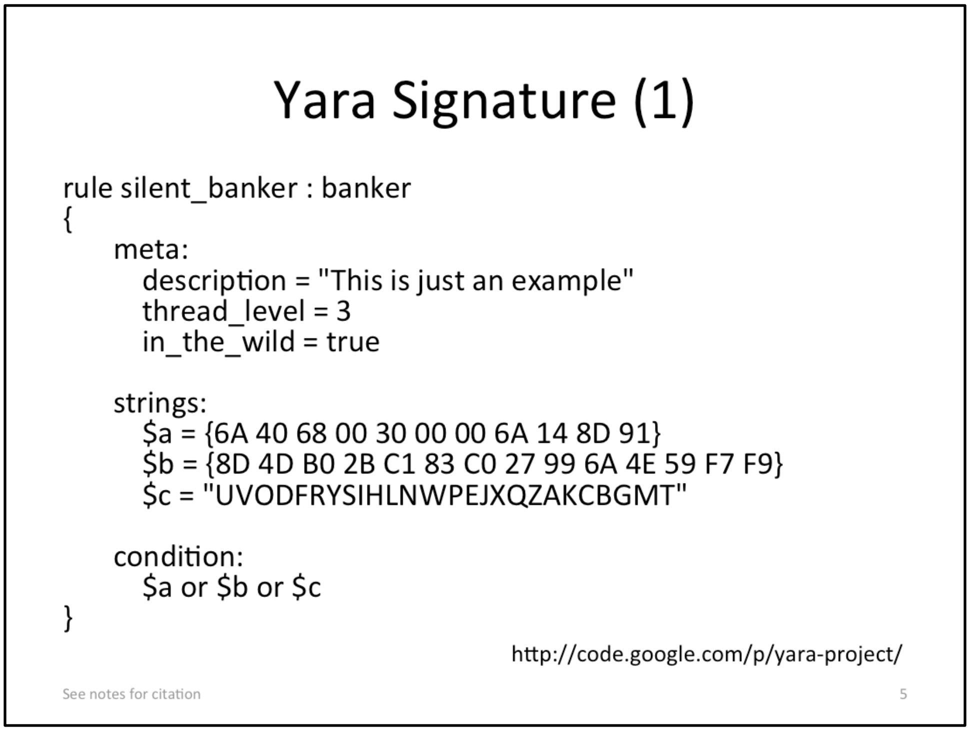
- Open source tool to identify and classify malicious files based on textual or binary patterns
- Light-weight way of performing signature checks
- Can be used for any binary data (exe, pdf, pcaps, etc)
- Useful in an email server for tip-offs, and filtering

See notes for citation

4

[References]

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



Yara Signature (1)

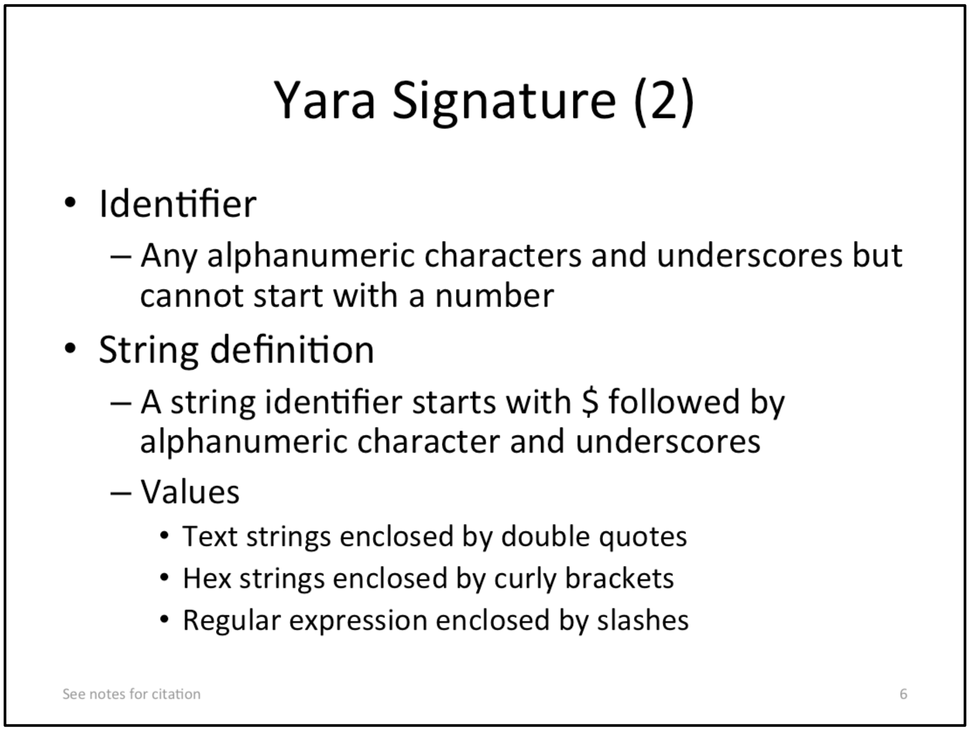
```
rule silent_banker : banker
{
      meta:
         description = "This is just an example"
         thread_level = 3
         in_the_wild = true

      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/

See notes for citation



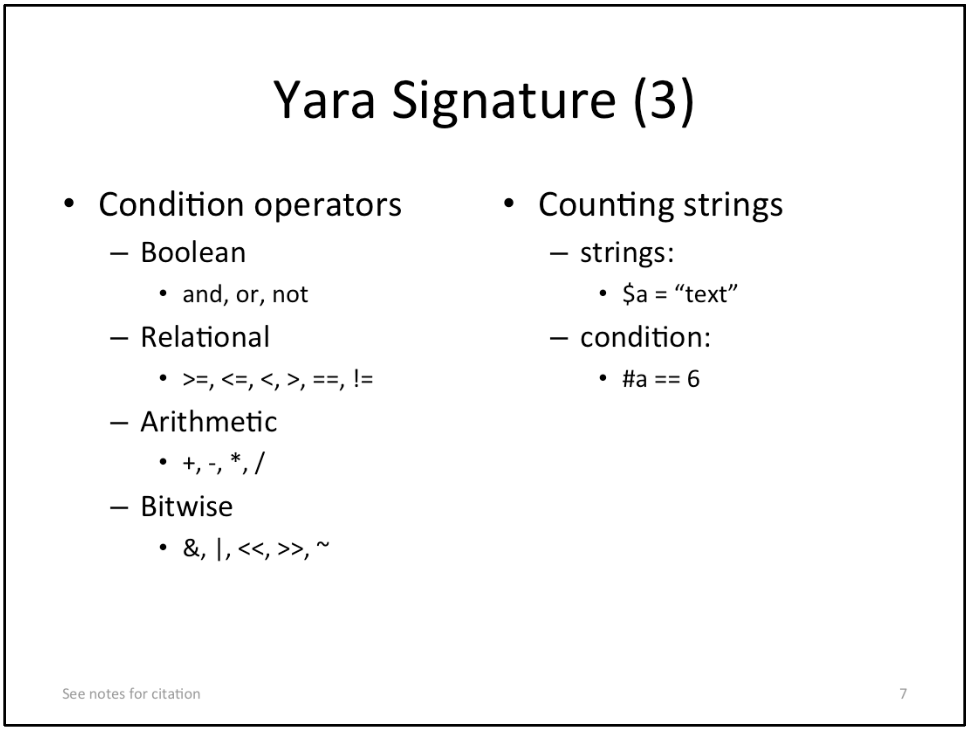
Yara Signature (2)

- 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

See notes for citation

[References]

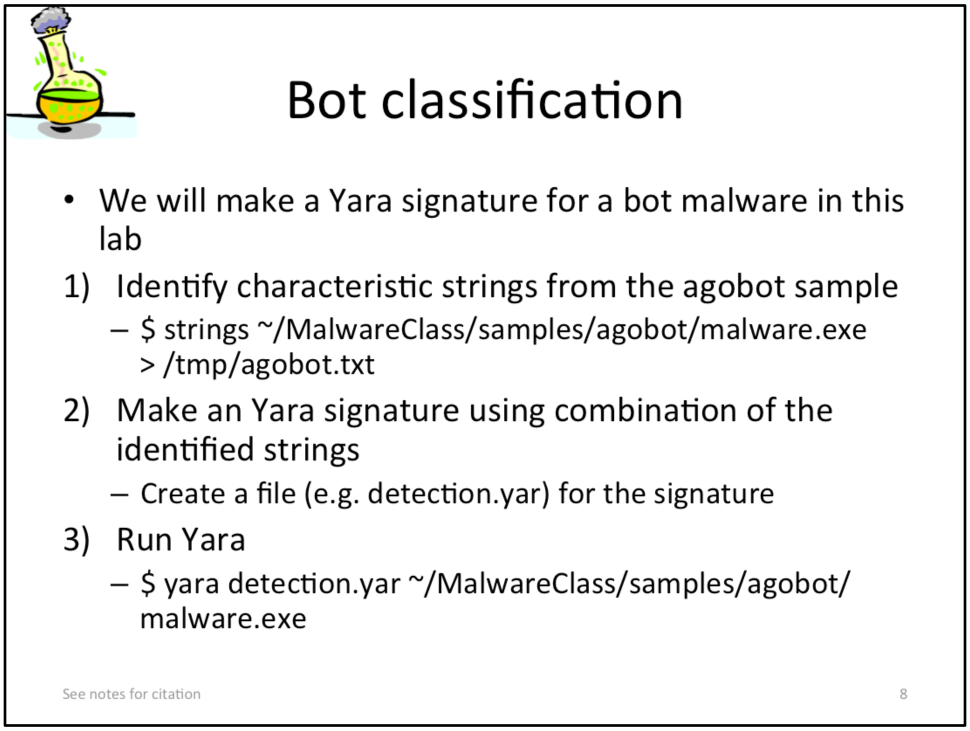
- Víctor Manuel Álvarez, YARA User's Manual 1.6, http://code.google.com/p/yara-project/downloads/detail?name=YARA%20User%27s%20Manual%201.6.pdf



Yara Signature (3)

- Condition operators
  - Boolean
    - and, or, not
  - Relational
    - >=, <=, <, >, ==, !=
  - Arithmetic
    - +, -, *, /
  - Bitwise
    - &, |, <<, >>, ~
- Counting strings
  - strings:
    - $a = “text”
  - condition:
    - #a == 6

See notes for citation

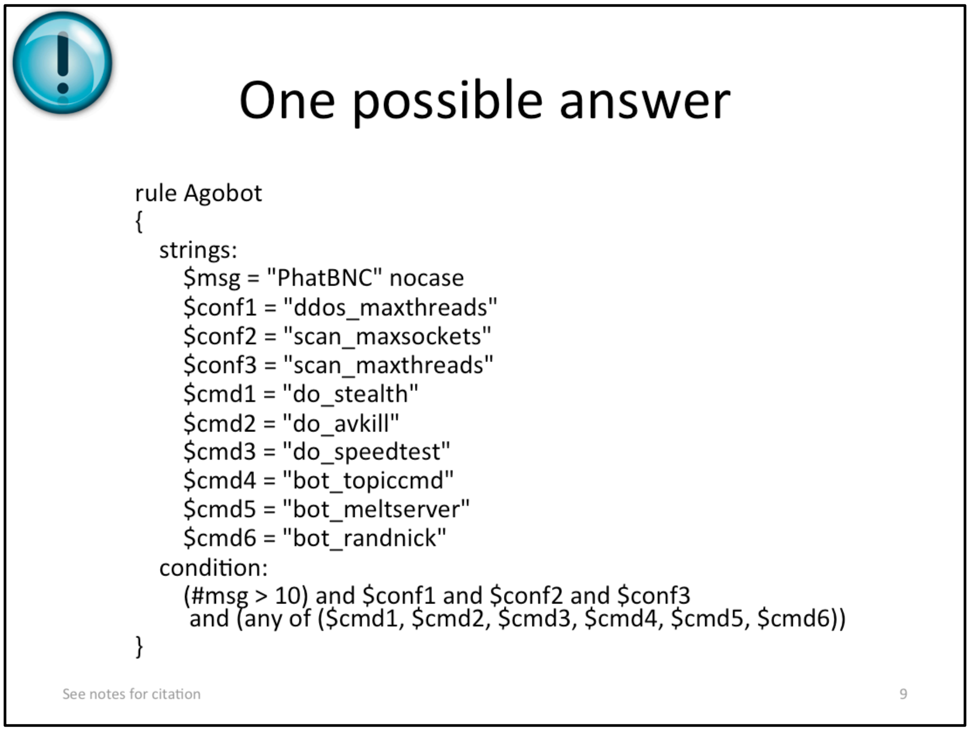


Bot classification

- 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

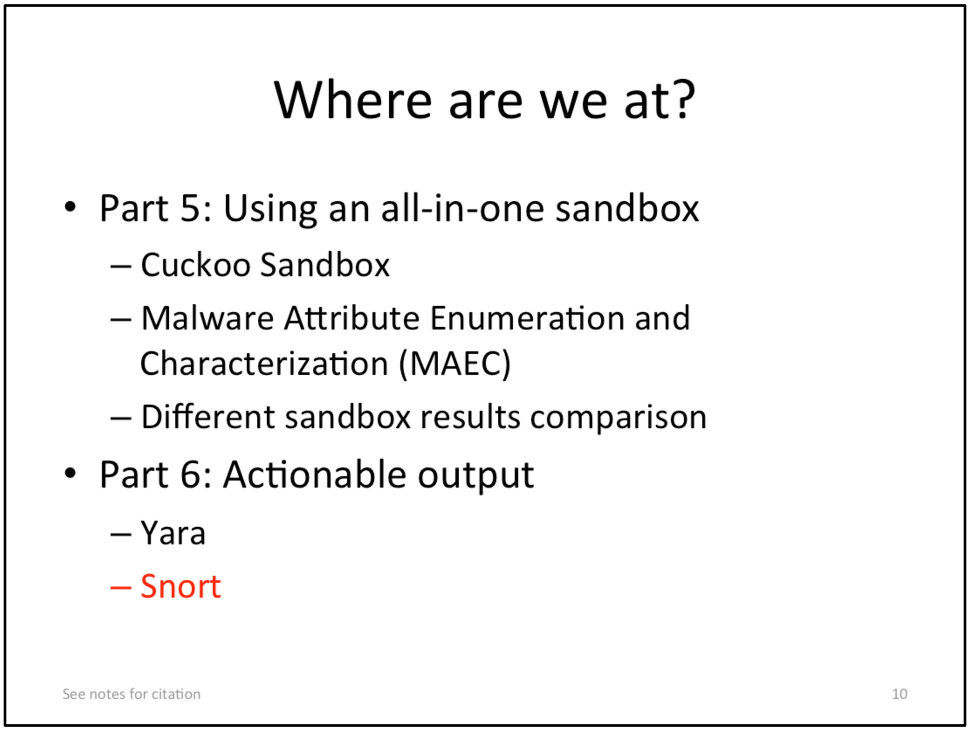
See notes for citation



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))
}
```

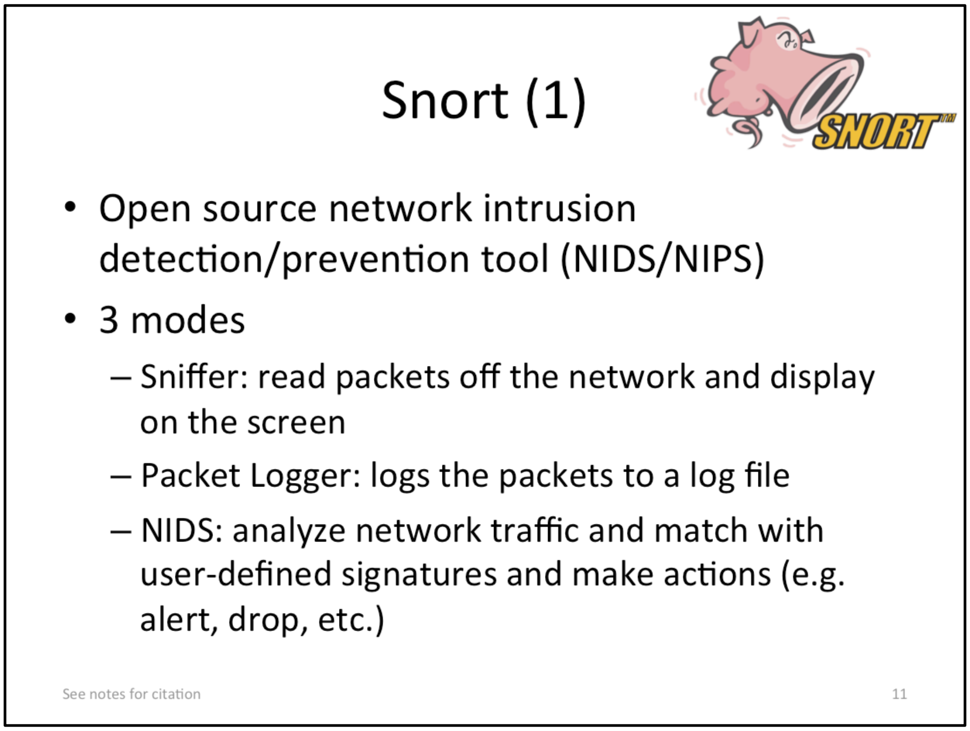
See notes for citation



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

See notes for citation



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.)

See notes for citation

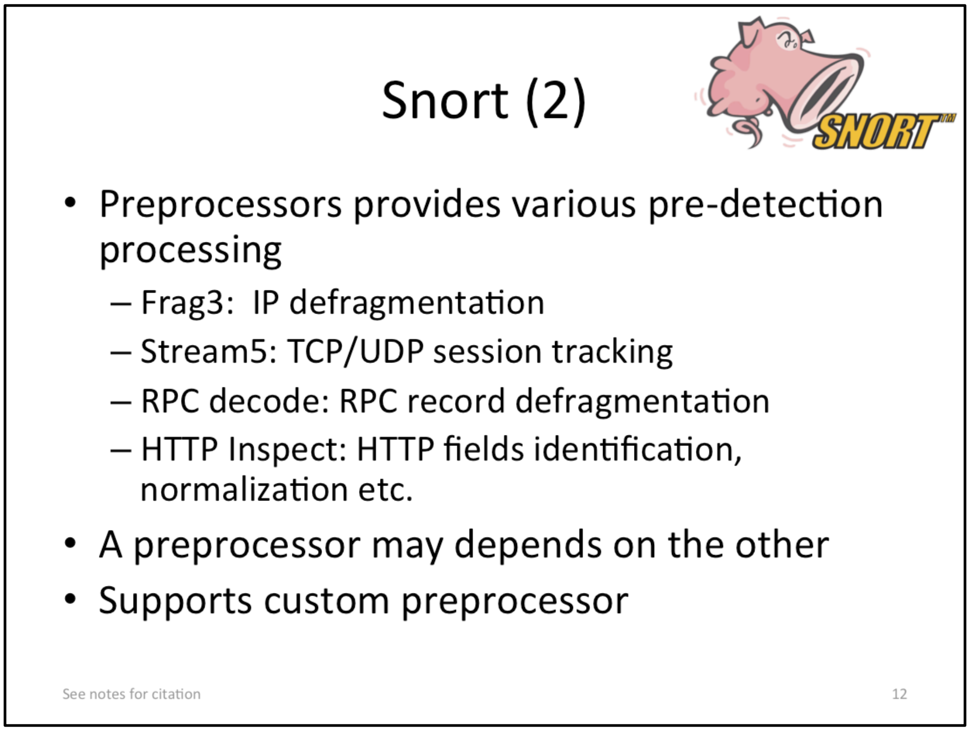
11

[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]

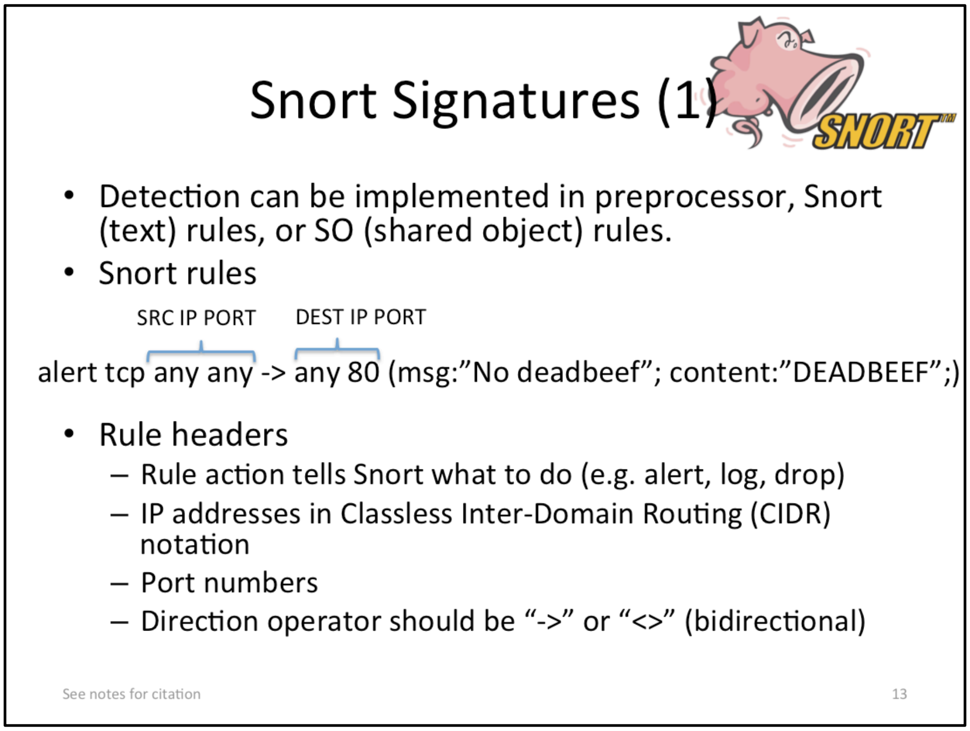
- http://4.bp.blogspot.com/_2IvFH57W8Hc/TPfpzDtwQwI/AAAAAAAAAFk/YFngxr8jLgI/s1600/snort_large.gif



Snort (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

See notes for citation



Snort Signatures (1)

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

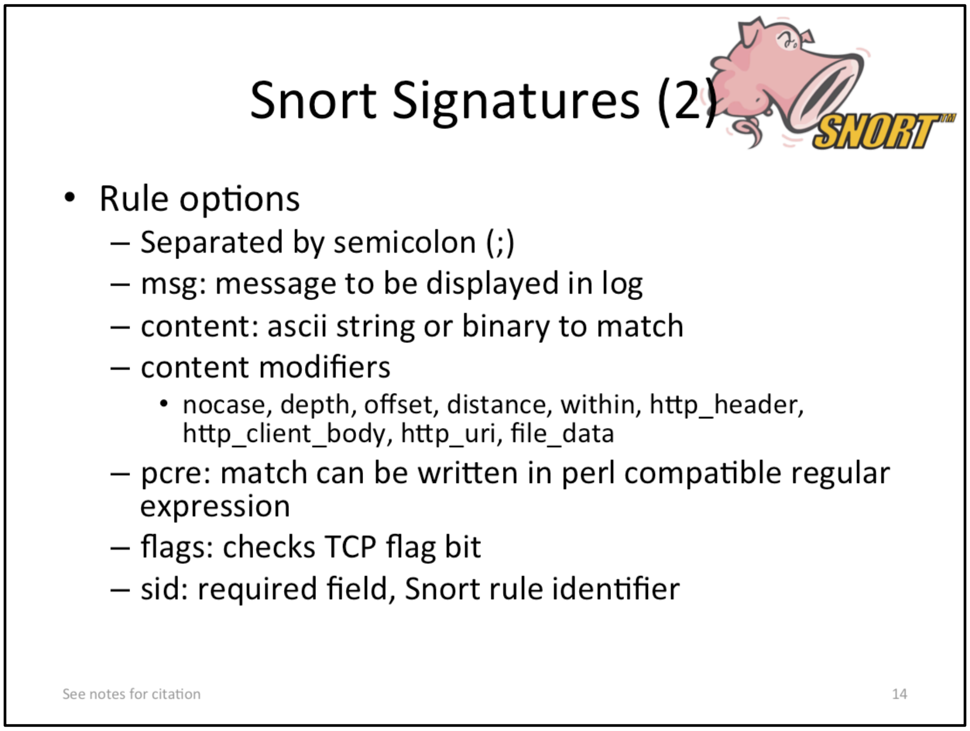
```
          SRC IP PORT   DEST IP PORT
alert tcp any any -> any 80 (msg:”No deadbeef”; content:”DEADBEEF”;)
```

- 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)

See notes for citation

[References]

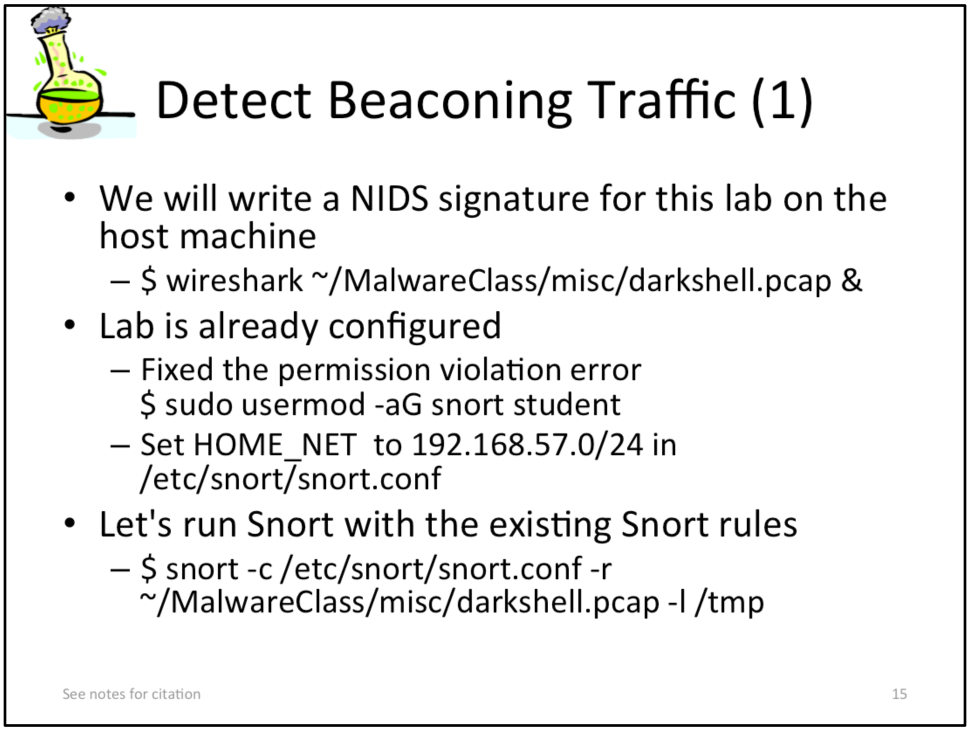
- Pre-Compile SO Rules: Supported Platforms, https://www.snort.org/snort-rules/shared-object-rule



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

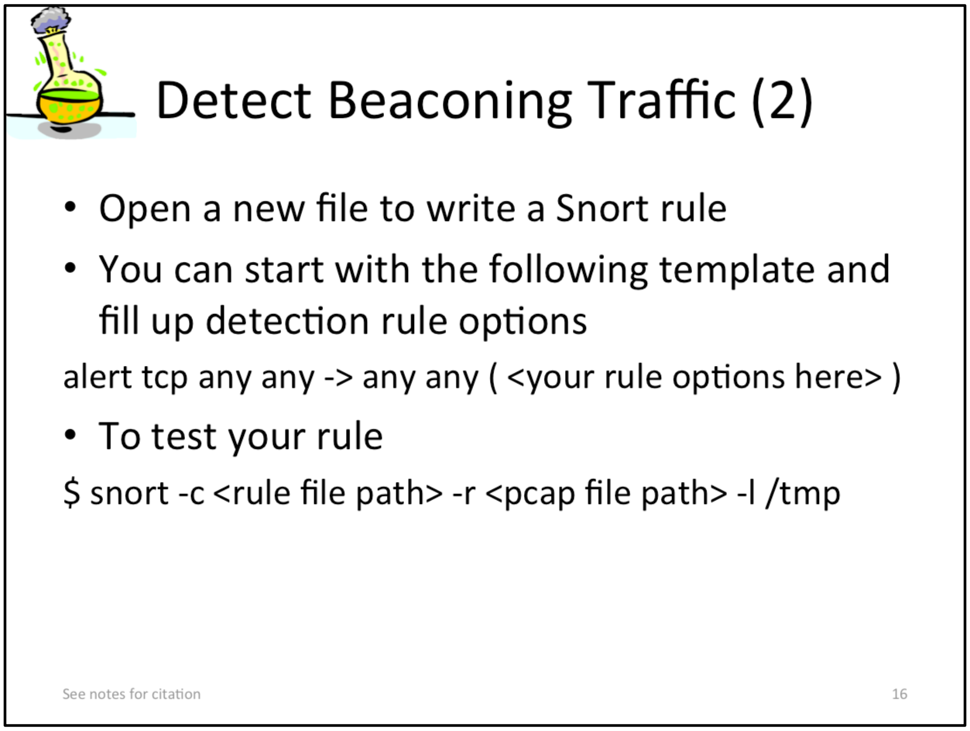
See notes for citation



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)

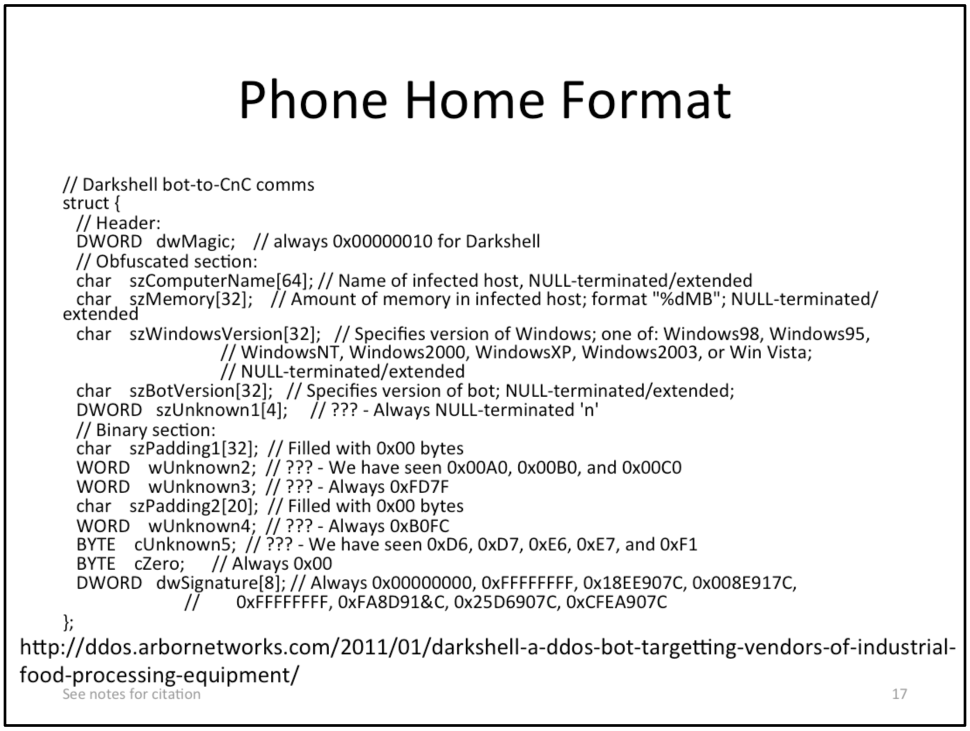
- Open a new file to write a Snort rule
- You can start with the following template and fill up detection rule options

```
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

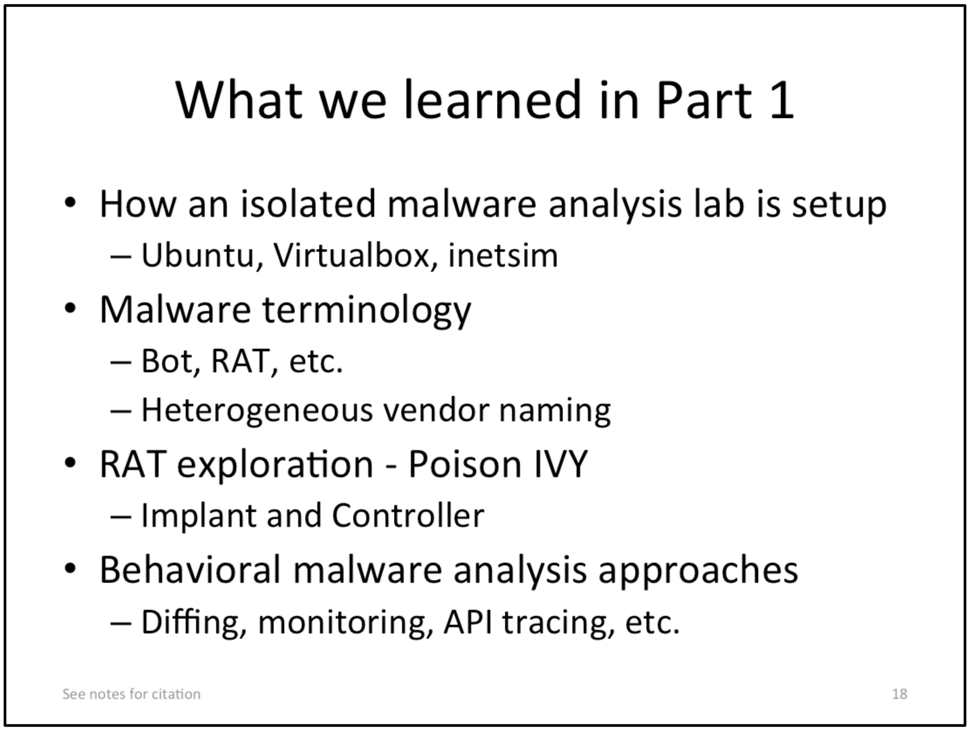


Phone Home Format

```
// 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/

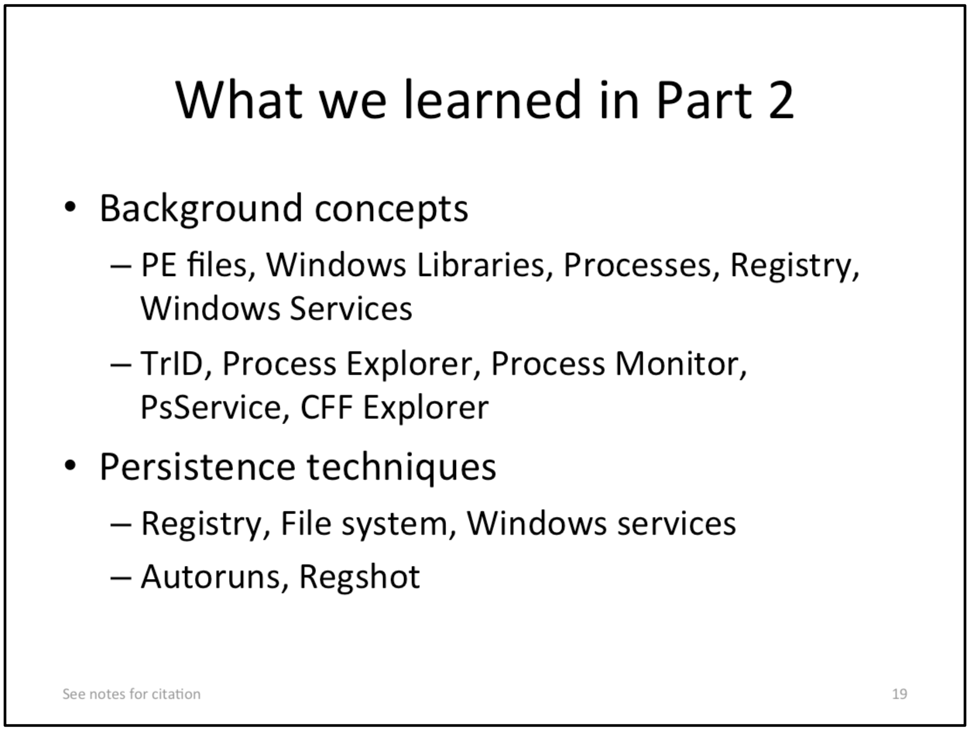
See notes for citation



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.

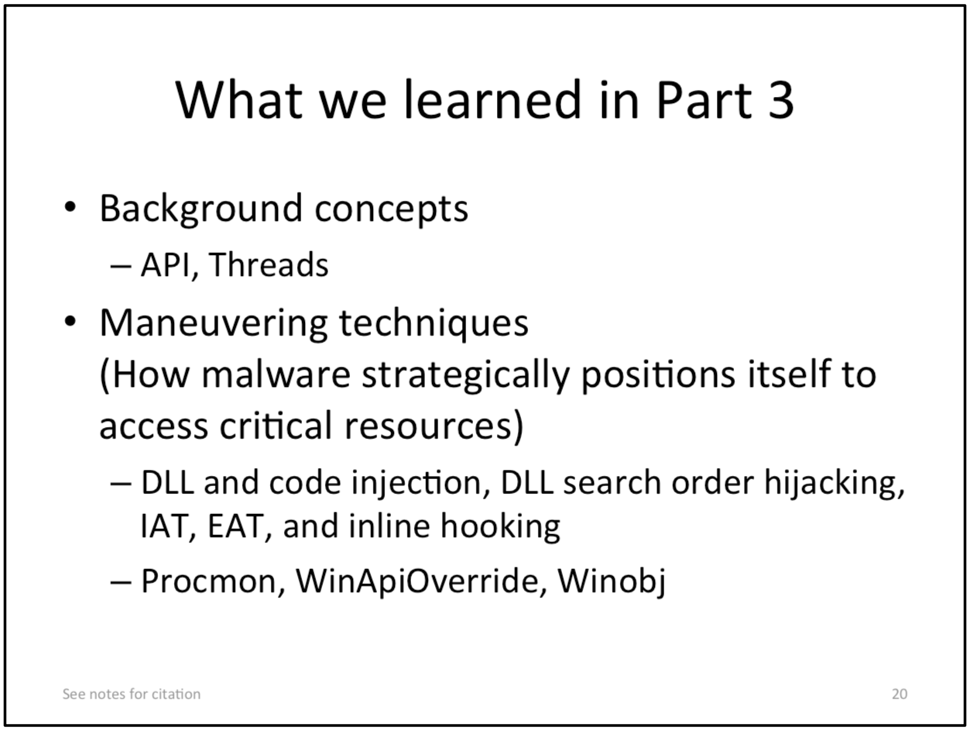
See notes for citation



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

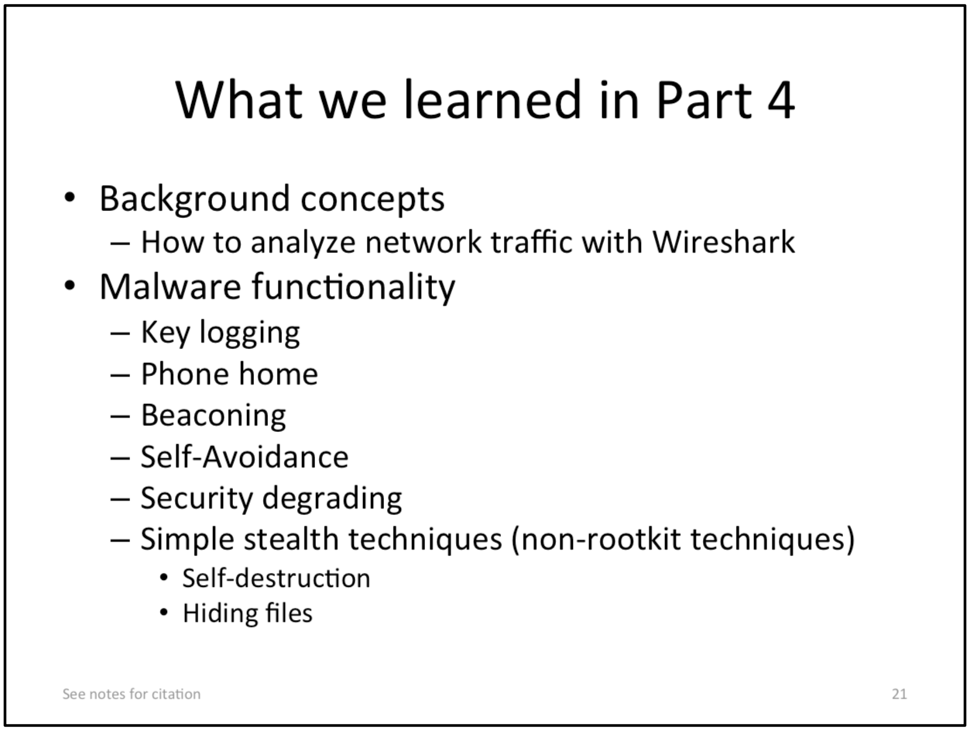
See notes for citation



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

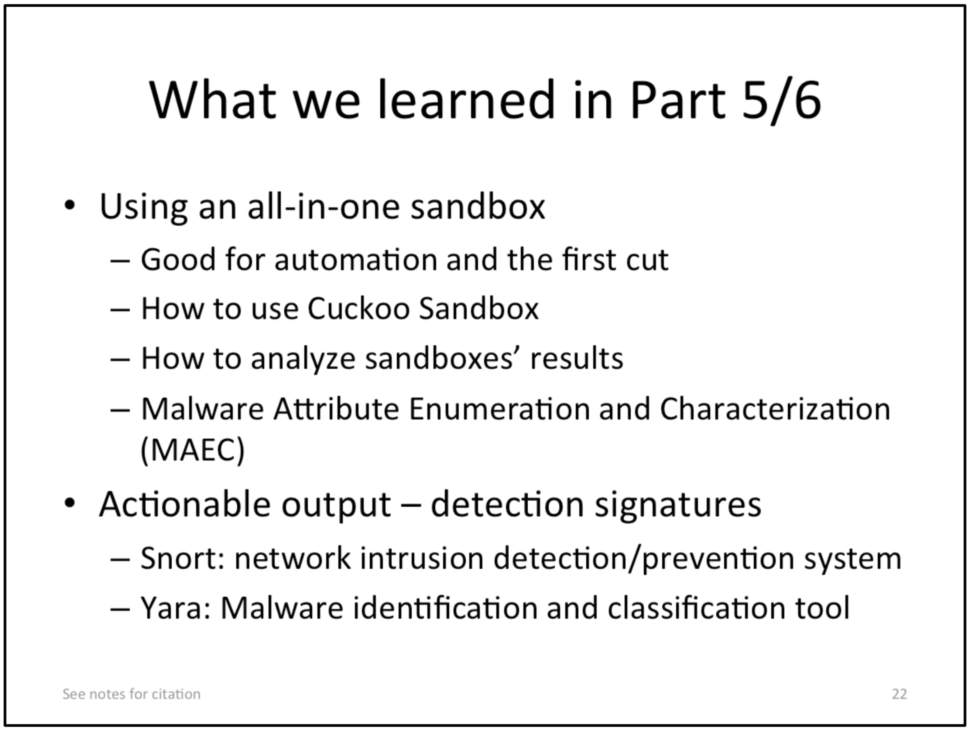
See notes for citation



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

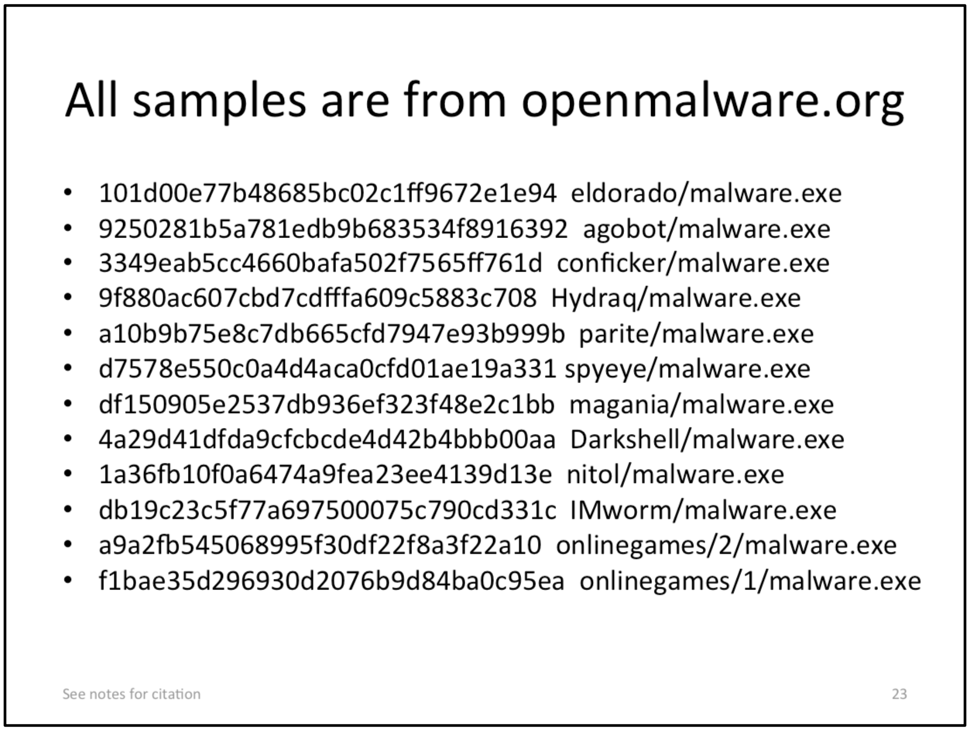
See notes for citation



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

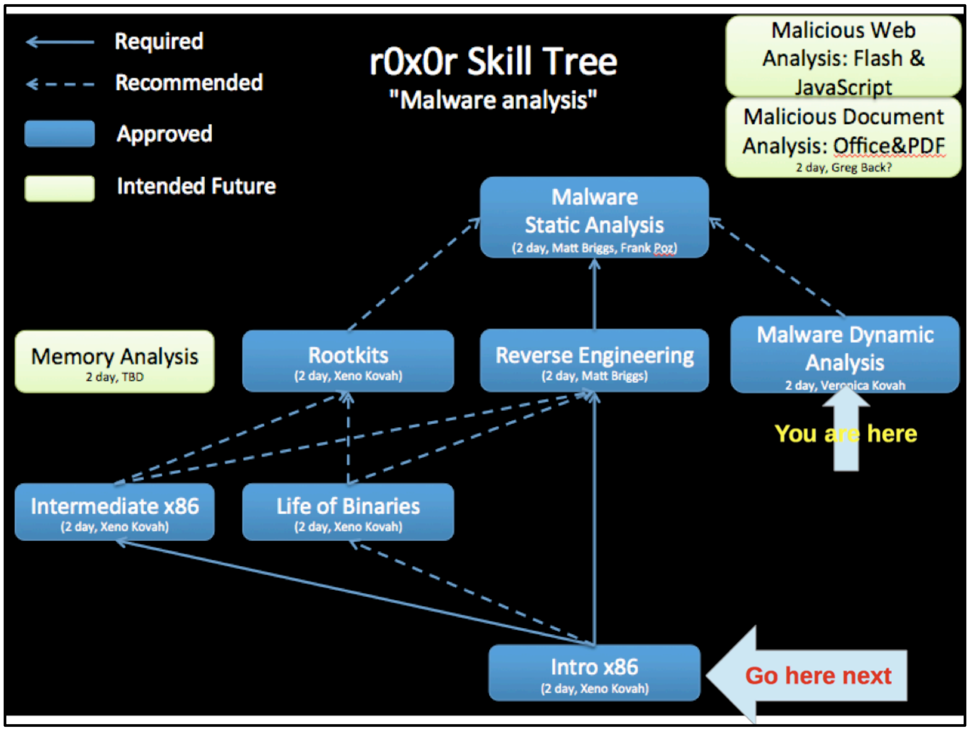
See notes for citation



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



r0x0r Skill Tree

"Malware analysis"

← Required

← - - - Recommended

🟦 Approved

🟨 Intended Future

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

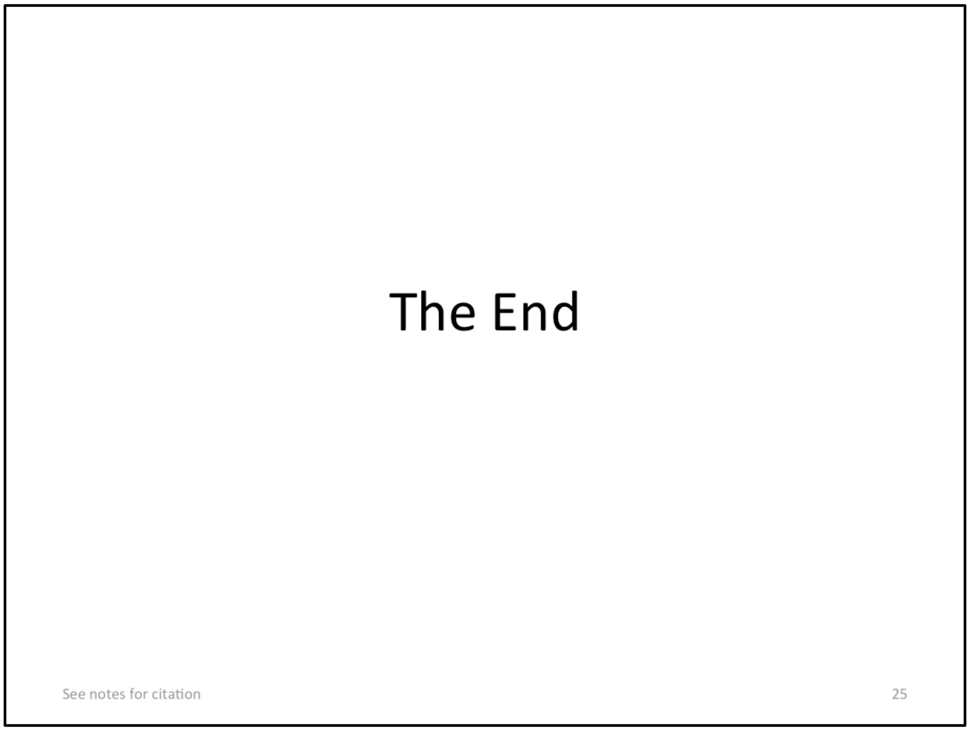
You are here

Intermediate x86
(2 day, Xeno Kovah)

Life of Binaries
(2 day, Xeno Kovah)

Intro x86
(2 day, Xeno Kovah)

Go here next



The End

See notes for citation