Security in knowledge

Detecting the One Percent: Advanced Targeted Malware Detection

Tomer Teller

Check Point Software Technologies

RSA CONFERENCE 2013

Session ID: SP02-T19

Session Classification: Intermediate

Antivirus 20th+ Anniversary


Check Point
SOFTWARE TECHNOLOGIES LTD.

The Halting Problem

$$h(i,x) = \begin{cases} 1 \text{ if program } i \text{ halts on input } x, \\ 0 \text{ otherwise.} \end{cases}$$

The Malware Problem

$$m(i) = \begin{cases} 1 \text{ if program } i \text{ is malicious} \\ 0 \text{ otherwise.} \end{cases}$$

1%

The Constraints

TIME

Cannot analyze program forever

- Slow down loops
- Sleep
- Time-consuming operations (Encryption/Packing)

SPACE

Cannot maintain unlimited states

- "Run out the clock"
  OpenProcess → VirtualAllocEx →
  WriteProcessMemory → LOOP → .. →
  CreateRemoteThread

Exploiting the Constraints

Advanced malware exploits these **constraints**

Thwart **static** analysis --> SPACE

Thwart **dynamic** analysis --> TIME + SPACE

More Depressing News

- Elevation of privilege to kernel mode
  - Bypassing security products
- Stolen certificate authorities
  - Breaking the trust
- Automatic static analysis is hard!
  - Packing / obfuscation / encryption
- Manual static analysis
  - Unpacking / time consuming / not scalable
- Dynamic analysis
  - The malware problem!

Relax!

Current Detection Methods (partial)

Pattern Based

- MD5 / SHA1 / SHA256
- Fuzzy hashing
- Pattern-based
- PCRE/ Regex
- Proprietary language
- Malware classifiers (J48, J48 Graft, PART)

Static Analysis

- Anti-VM
- Anti-debugging
- Anti-disassembly
- Obfuscation
- Reverse engineering

**Rodrigo Rubira Branco BH12

Dynamic Analysis

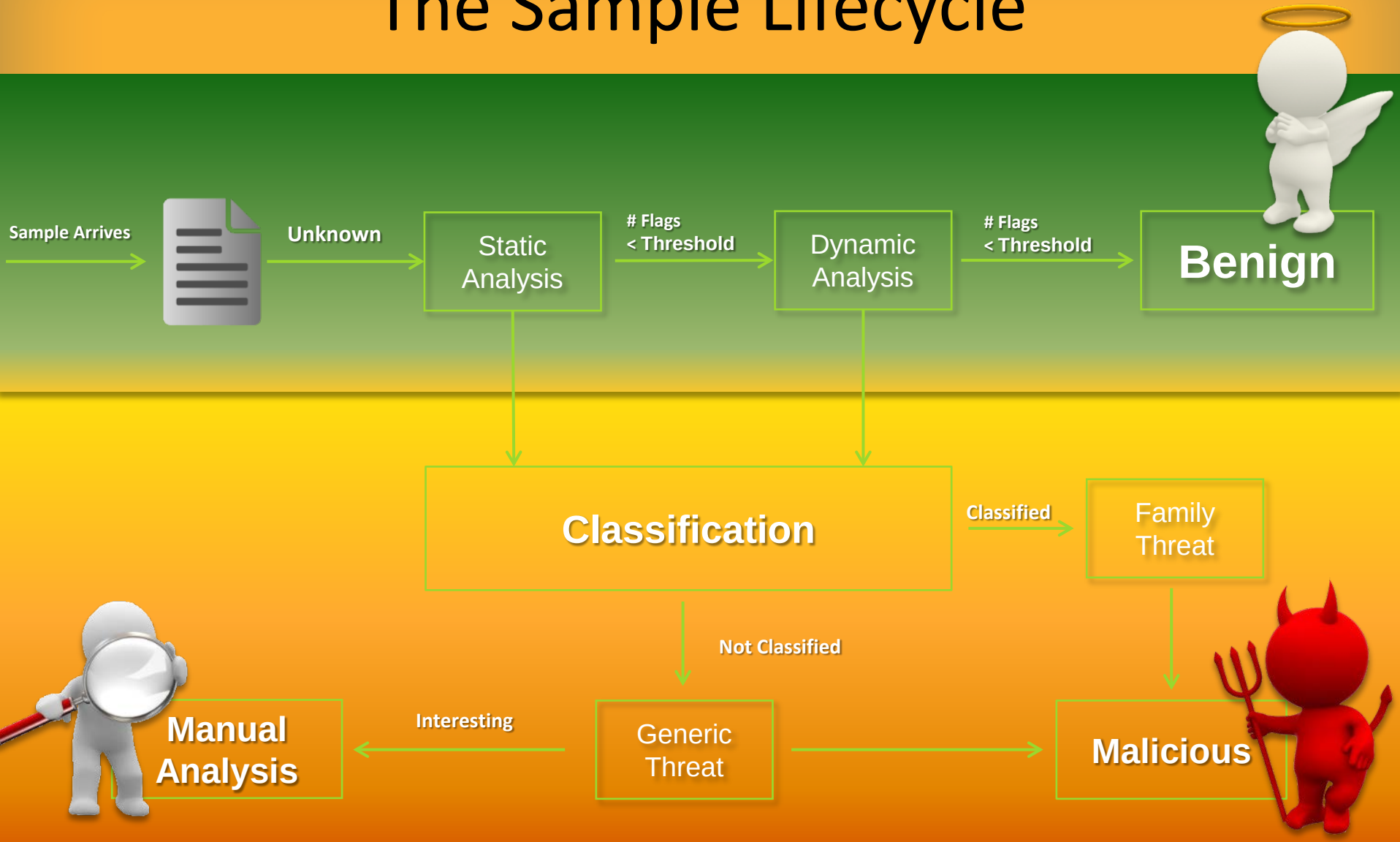
- API call trace analysis
- Network activities
- Registry modifications
- Process creation/injections
- File activities

What you see is what you get!

Hybrid Approach

- Semantic-aware detectors
  - Extract dynamic trace
  - Transform into IR
  - Compare to pre-defined templates
- Memory dump analysis (packers)

The Sample Lifecycle



Bypassing Detection Methods

Pattern Based

- Build variants (e.g. Zeus)
- Append garbage
- Encoding
- “Stay compliant”

Static Analysis

- Packing
- Obfuscation
- Encryption
- Anti-reversing techniques

Dynamic Analysis

- Detect analysis*
- Detect emulation*
- Detect security product*
- Beat the clock (AV sandbox)
- “Split the maliciousness”

*Could be detected during static analysis

Hybrid Approach

- Avoid using the same executable template
- Metasploit AV-evasion
  - Reuse “trusted templates”
  - PowerShell
  - In-memory exploits

MYTH #1

Malware executes immediately


NO!

MYTH #2

Malware is usually small


NO!

Malware Detection Based on File Size

	File Size limit										
	1	2	3	4	5	6	7	8	9	10	no limit
exploit	99.83%	99.95%	99.97%	99.97%	99.98%	99.98%	99.99%	100.00%	100.00%	100.00%	100.00%
im-worm	98.83%	99.71%	99.90%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%
mass-mailer	99.62%	99.87%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%
mobile	99.44%	99.78%	99.88%	99.90%	99.93%	99.95%	99.97%	99.98%	99.99%	99.99%	100.00%
macro virus	99.63%	99.82%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%
phish	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%
scripts	98.25%	99.64%	99.88%	99.92%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%
spyware	95.08%	97.97%	98.88%	99.47%	99.76%	99.83%	99.89%	99.91%	99.94%	99.95%	100.00%
trojan	97.02%	99.24%	99.62%	99.80%	99.88%	99.93%	99.95%	99.97%	99.98%	99.98%	100.00%
virus	98.27%	99.37%	99.63%	99.80%	99.89%	99.92%	99.95%	99.97%	99.98%	99.99%	100.00%
worm	99.02%	99.65%	99.74%	99.86%	99.89%	99.92%	99.94%	99.94%	99.95%	99.96%	100.00%

*Size in MB

Ref: http://www.fortiguard.com/sites/default/files/DetectingMalwareThreats.pdf

Malware Bypassing Detections

Stuxnet

Static Analysis	Dynamic Analysis
4 x Zero-Days	**Obfuscated Entry Point** (Needs a special Loader)
2 x Stolen Certificates (Break the trust)	**Multiple Files** (lesser maliciousness entropy)
Unknown DLL loading technique	**Execution depends on host**

Flame

Static Analysis	Dynamic Analysis
20 MB of Code!	**Does not execute immediately**
Breakthrough in cryptography (Break the trust)	**Multiple Files** (lesser maliciousness entropy)
Legitimate Libraries (LUA)	**Obfuscated Entry Point** (Needs a special Loader)

Malware Thwarting Analysis

Mac OSX

C&C

Exploit Java Vulnerability

Sen e)

Infected with FlashFake

UUID

Encryption Key

Encrypt Malware and Obfuscate

Executable will only run on the original host

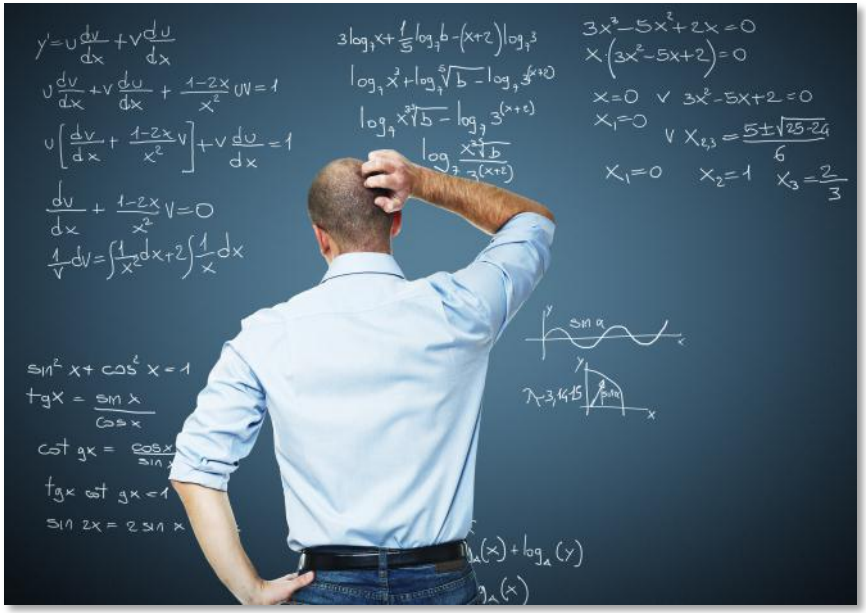
Problem

Detection is good but not great

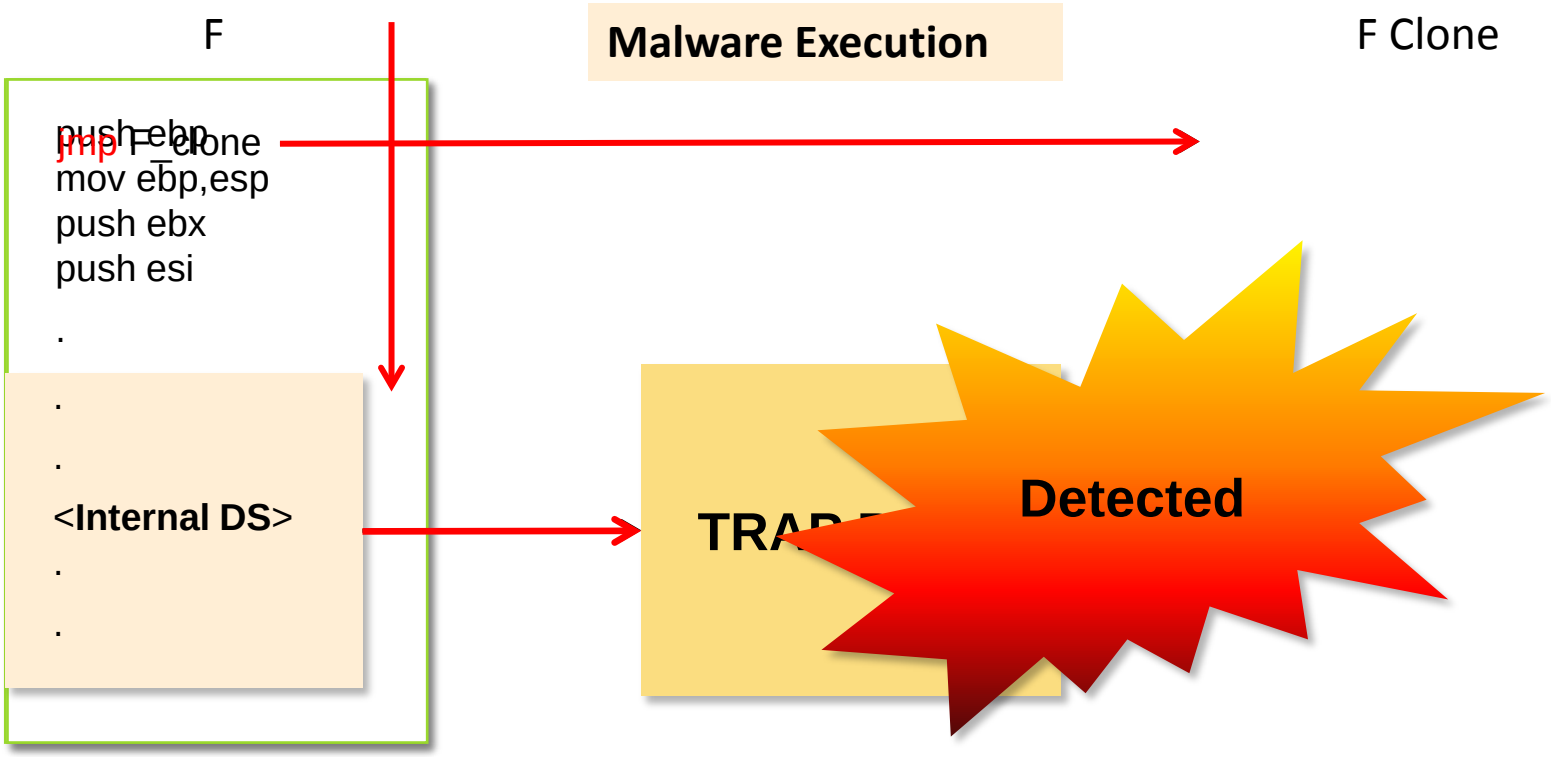
Data-Structure Modifications

Problem:
Malware modifies internal data-structures to avoid detection

Solution:
Subvert the malware!
Modify the data-structure before the malware does



Detecting Internal DS Modifications



PsSetCreateProcessNotifyRoutine

```
PAGE:0055677E ; __stdcall PsSetCreateProcessNotifyRoutine(x, x)
PAGE:0055677E                 public _PsSetCreateProcessNotifyRoutine@8
PAGE:0055677E _PsSetCreatePr
PAGE:0055677E
PAGE:0055677E arg_0
PAGE:0055677E arg_4           =               0Ch
PAGE:0055677E
PAGE:0055677E                 mov     edi, edi
PAGE:00556780                 push    ebp
PAGE:00556781                 mov     ebp, esp
PAGE:00556783                 push    ebx
PAGE:00556784                 xor     ebx, ebx
PAGE:00556786                 cmp     [ebp+arg_4], bl
PAGE:00556789                 push    esi
PAGE:0055678A                 push    edi
PAGE:0055678B                 jz      short loc_5567F2
PAGE:0055678D                 mov     edi, offset _PspCreateProcessNotifyRoutine
PAGE:00556792
PAGE:00556792 loc_556792:                             ; CODE XREF: PsSetCreateProcessNotifyRoutine(x,
```

Linear search for a signature

Internal DS

Example::PsSetCreateProcessNotifyRoutine

Original

```
PAGE:0055677E ; __stdcall PsSetCreateProcessNotifyRoutine(x, x)
PAGE:0055677E                 public _PsSetCreateProcessNotifyRoutine@8
PAGE:0055677E _PsSetCreateProcessNotifyRoutine@8 proc near
PAGE:0055677E
PAGE:0055677E arg_0           = dword ptr  8
PAGE:0055677E arg_4           = byte ptr  0Ch
PAGE:0055677E
PAGE:0055677E                 jmp     short _PsSetCreateProcessNotifyRoutineClone
PAGE:00556780                 mov     ebp, esp
PAGE:00556781                 push    ebx
PAGE:00556783                 xor     ebx, ebx
PAGE:00556784                 cmp     [ebp+arg_4], bl
PAGE:00556786                 push    esi
PAGE:00556789                 push    edi
PAGE:0055678A                 jz      short loc_5567F2
PAGE:0055678B                 mov     edi, offset _PspCreateProcessNotifyRoutine
PAGE:0055678D
PAGE:00556792 loc_556792:                             ; CODE XREF: PsSetCreateProcessNotifyRoutine(x,x)+46↓j
```

Clone

```
PAGE:0055677E ; __stdcall PsSetCreateProcessNotifyRoutineClone(x, x)
PAGE:0055677E                 public _PsSetCreateProcessNotifyRoutineClon
PAGE:0055677E _PsSetCreateProcessNotifyRoutineClone@8
PAGE:0055677E
PAGE:0055677E arg_0           = dword ptr  8
PAGE:0055677E arg_4           = byte ptr  0Ch
PAGE:0055677E
PAGE:0055677E                 mov     edi, edi
PAGE:00556780                 push    ebp
PAGE:00556781                 mov     ebp, esp
PAGE:00556783                 push    ebx
PAGE:00556784                 xor     ebx, ebx
PAGE:00556786                 cmp     [ebp+arg_4], bl
PAGE:00556789                 push    esi
PAGE:0055678A                 push    edi
PAGE:0055678B                 jz      short loc_5567F2
PAGE:0055678D                 mov     edi, offset _PspCreateProcessNotifyRoutine
PAGE:00556792
PAGE:00556792 loc_556792:
```

PAGE Guard

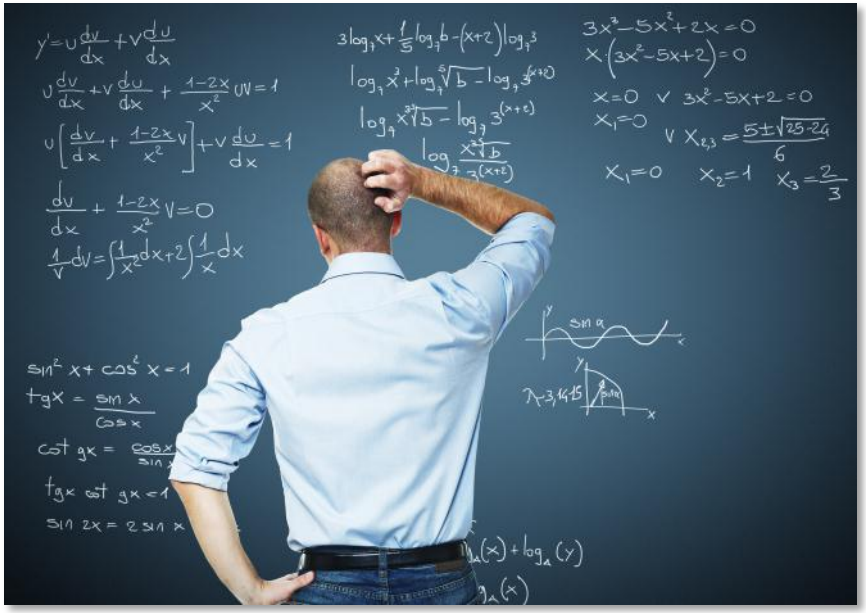
Technique Usage

- Detect function hooking tempering
  - Hook a function and monitor the hook
  - Protect the monitor routine
- Detection of linear memory scanning
  - Staged attacks
  - Egg hunt
- Detection of internal data-structure manipulation
  - Basic DKOM Detection
- Place calls to Page Guard in strategic places
  - Detect Heap Spraying (“canary” value)

Process Enumeration

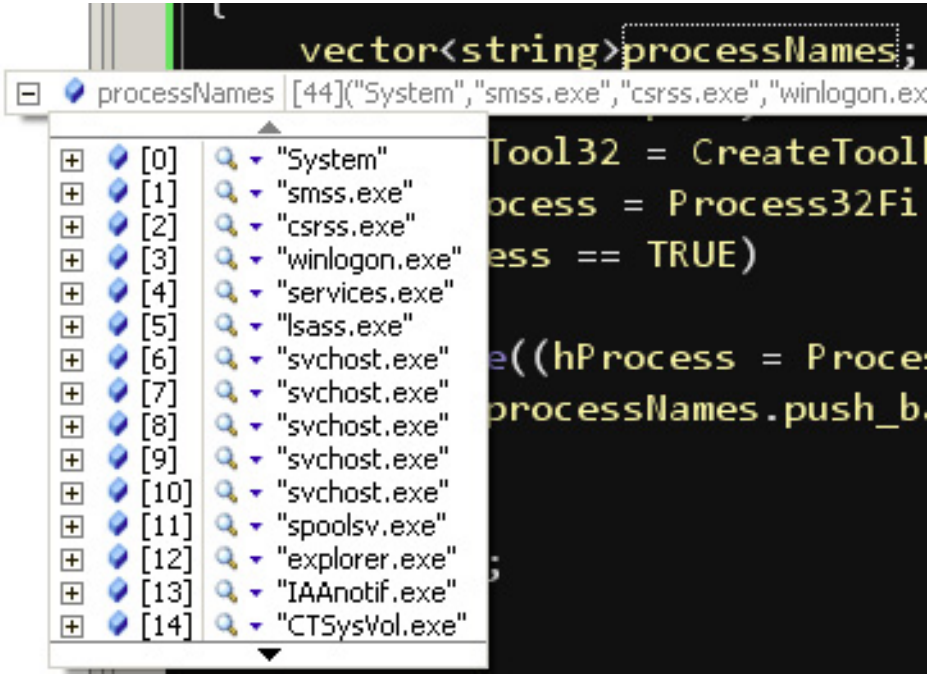
Problem:
Malware checks for the existence of a security product process

Solution:
Process enumeration using weight based mechanism taint analysis



Detecting "weird" Process Enumerations

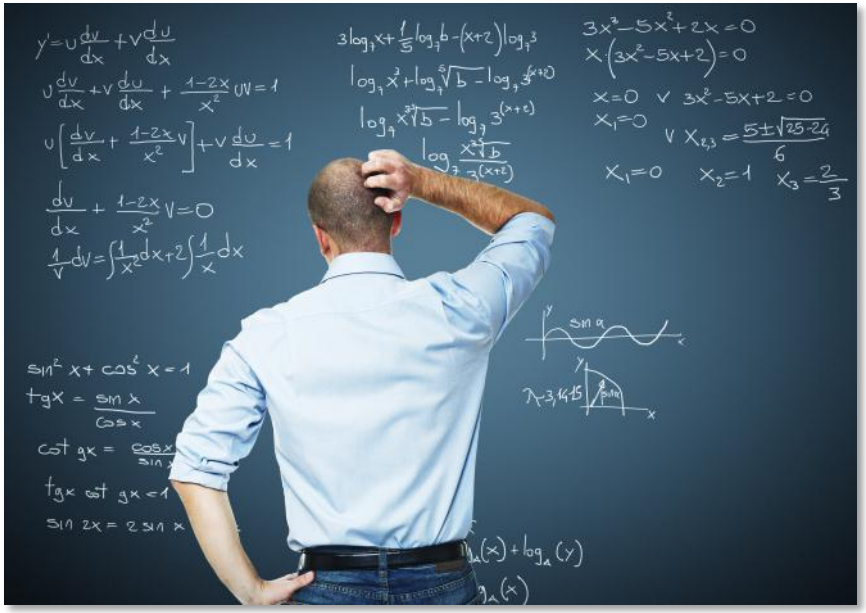
- Monitor EPROCESS structure access
- Track **process name** usage (taint analysis)
- Score the process based on "weird" usage
  - HASH
  - Encryption
  - Encoding
  - Etc.



Obfuscation!

Problem:
Malware uses obfuscation to hide malicious code during Drive-by-download attacks

Solution:
Hook the browser at strategic places and inspect the de-obfuscated buffers



Tapping The Browser

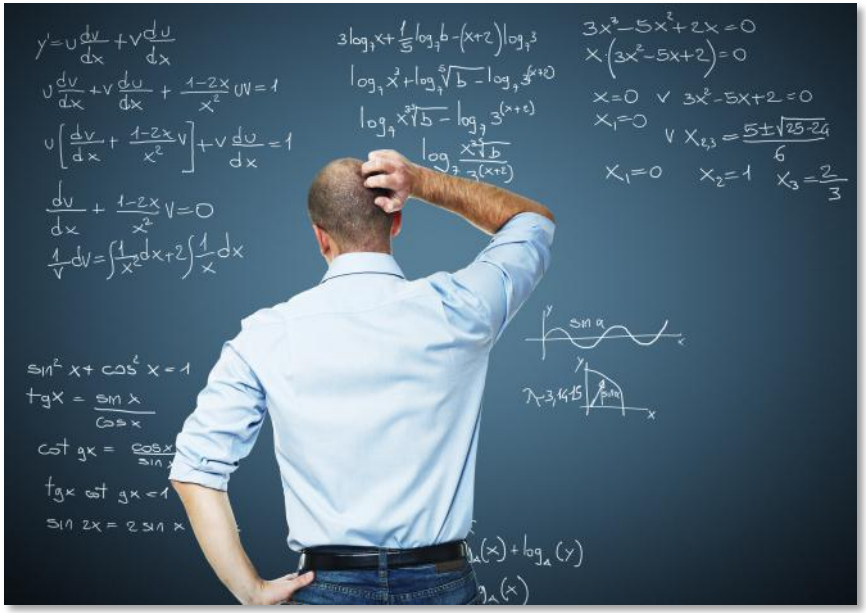
- Obfuscation is a problem!
  - Network devices are blind
- Possible solution on the network side
  - Analyze data entropy to detect possible obfuscation
  - Google uses obfuscation -> massive FP
- Better solution on the end point
  - Hook the browsers (IE/Chrome/Firefox) at strategic places
    - Eval, Document.write, innerhtml, etc'
  - Let the browser do the "heavy lifting"
  - Communicate the information back to the network devices

DEMO: Browser Tapper

Anti-VM OUT, Anti-Analysis IN

Problem:
Malware drop anti-VM technique and focus on anti-Analysis techniques

Solution:
Subvert the analysis machine with a Rootkit before executing the malware



Subvert the Analysis Machine

- Malware usually cannot detect Rootkit!
- Install a rootkit on the analysis machine
  - Hide files/processes/drivers
  - Hide open ports
  - Hide registry values
- Malware is not aware that it is being subverted
- Results in higher detection rate of advanced malware

DEMO: Tool-B-Gone

- Easy-to-use rootkit generator
- Choose the process/files/ports/registry values you wish to hide
- Generates a customize rootkit
- Install rootkit
- Benefit!

Future Directions

- Detecting internal threats using ML
  - Most network behavior analysis tools fail to deliver
    - Bad feature sets that results in massive FP
  - Feature set focus on user **behavioral** profile and **not** malware
    - Data entropy / Working hours / Keyboard typing speed'
  - Based on the protoleak project (RSA 12')
    - Profile-based decision tree per node
  - Focus on data exfiltration and behavior deviations

- Malware Interaction
  - Click/Move Mouse
  - Open Applications

How to Apply

- Force malware mistakes, don't wait for them to strike
- Raise attackers cost by innovating mitigations
- Download & try the tools
- Help fighting the 1% and suggest improvements

Questions

Thank You

@djteller

Security in knowledge