Automatic malicious code extraction using Volatility Framework

Martin G. Korman

What's your take away?

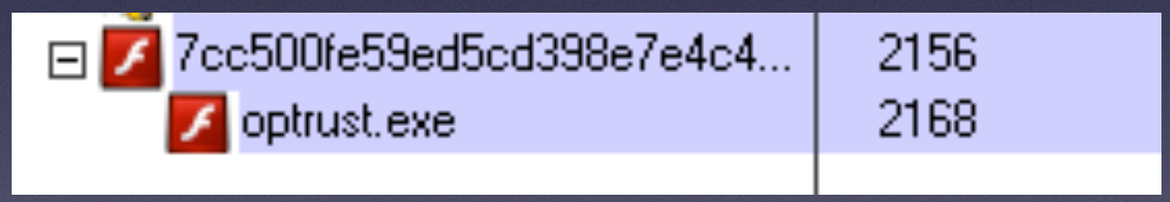
An open source tool, that extracts malicious code from memory and prepares it for deeper static analysis

http://www.volatilityfoundation.org/

Malicious Code – Where is it hiding?

New Processes

- New processes created by the Malware



Self Modifying Code

- Packers and Crypters have to unpack the malware in memory, in order to execute it.

Name	Virtual Size	Virtual Address	Raw Size	Raw Address	Reloc Address	Linenumbers	Relocations ...	Linenumber...	Characteristics
Byte[8]	Dword	Dword	Dword	Dword	Dword	Dword	Word	Word	Dword
lhkpk	0022E000	00001000	0002BA00	00000400	00000000	00000000	0000	0000	40000040
akepu	00022000	0022F000	00019000	0002BE00	00000000	00000000	0000	0000	C8BE946E
ntnvg	00012000	00251000	00001000	00044E00	00000000	00000000	0000	0000	CAB911A9
tobor	00008000	00263000	00004600	00045E00	00000000	00000000	0000	0000	C38CBA14
eopan	00004000	0026B000	00000600	0004A400	00000000	00000000	0000	0000	C4541365
xyfmd	00004000	0026F000	00000600	0004AA00	00000000	00000000	0000	0000	C54F1E3A
jyacd	00004000	00273000	00000600	0004B000	00000000	00000000	0000	0000	CCF6D1A4
vnidw	00010000	00277000	00003A00	0004B600	00000000	00000000	0000	0000	CB94C4F9

Remote Code Injection

- A malicious process writes code into the memory space of a target process and forces it to execute.

```
push    0                   ; lpNumberOfBytesWritten
push    [ebp+nSize]         ; nSize
push    [ebp+lpBuffer]      ; lpBuffer
push    esi                 ; lpBaseAddress
push    dword ptr [ecx+eax*8] ; hProcess
call    ds:WriteProcessMemory
```

```
lea     ecx, [esp+24h+var_4]
push    ecx                 ; lpThreadId
push    4                   ; dwCreationFlags
push    eax                 ; lpParameter
push    edi                 ; lpStartAddress
push    esi                 ; dwStackSize
push    esi                 ; lpThreadAttributes
push    ebx                 ; hProcess
call    ds:CreateRemoteThread
mov     edi, eax
test    edi, edi
jz      short loc_B24C7
```

VOLATILITY BOT

Hollow Process Code Injection

- A malicious process starts a new instance of a legitimate process (i.e explorer.exe, svchost.exe) in suspended mode. (0x00000004 Flag)

- Before resuming it, the executable section(s) are freed and reallocated with malicious code.

```
lea     eax, [ebp+hProcess]
push    eax             ; lpProcessInformation
lea     eax, [ebp+StartupInfo]
push    eax             ; lpStartupInfo
push    ebx             ; lpCurrentDirectory
push    ebx             ; lpEnvironment
push    1000004h        ; dwCreationFlags
push    ebx             ; bInheritHandles
push    ebx             ; lpThreadAttributes
push    ebx             ; lpProcessAttributes
lea     eax, [ebp+CommandLine]
push    eax             ; lpCommandLine
push    ebx             ; lpApplicationName
call    ds:CreateProcessW
push    [ebp+var_1C]
mov     esi, eax
call    dword ptr [edi+4]
test    esi, esi
jz      loc_B9A73
```

VOLATILITY BOT

Kernel Modules

- Usually they serve to hide malware evidence, make the malware harder to remove or obstruct the research process.
- “Advanced control and data flow hijacking techniques that leverage the lower layers of the OS architecture” [1]

1. Alex Matrosov, Eugene Rodionov, and Sergey Bratus, *Rootkits and Bootkits*, No Starch Press, 2015

API Hooks

- Hi-jacking the code flow of a legitimate windows API call, in order to make it do something else, i.e grab your POST request.

```
0:013> !chkimg wininet -d
    7532c87e-7532c882  5 bytes - WININET!InternetCloseHandle
      [ 8b ff 55 8b ec:e9 cd 8a e2 8c ]
    7532cc02-7532cc06  5 bytes - WININET!HttpQueryInfoA (+0x3f4)
      [ 8b ff 55 8b ec:e9 29 80 e2 8c ]
    7532e2a4-7532e2a8  5 bytes - WININET!InternetReadFile (+0x16a2)
      [ 8b ff 55 8b ec:e9 47 70 e2 8c ]
    7533420b-7533420f  5 bytes - WININET!InternetQueryDataAvailable (+0x5f67)
      [ 8b ff 55 8b ec:e9 10 0b e2 8c ]
    75351331-75351335  5 bytes - WININET!InternetReadFileExA (+0x1d126)
      [ 8b ff 55 8b ec:e9 7a 3e e0 8c ]
25 errors : wininet (7532c87e-75351335)
```

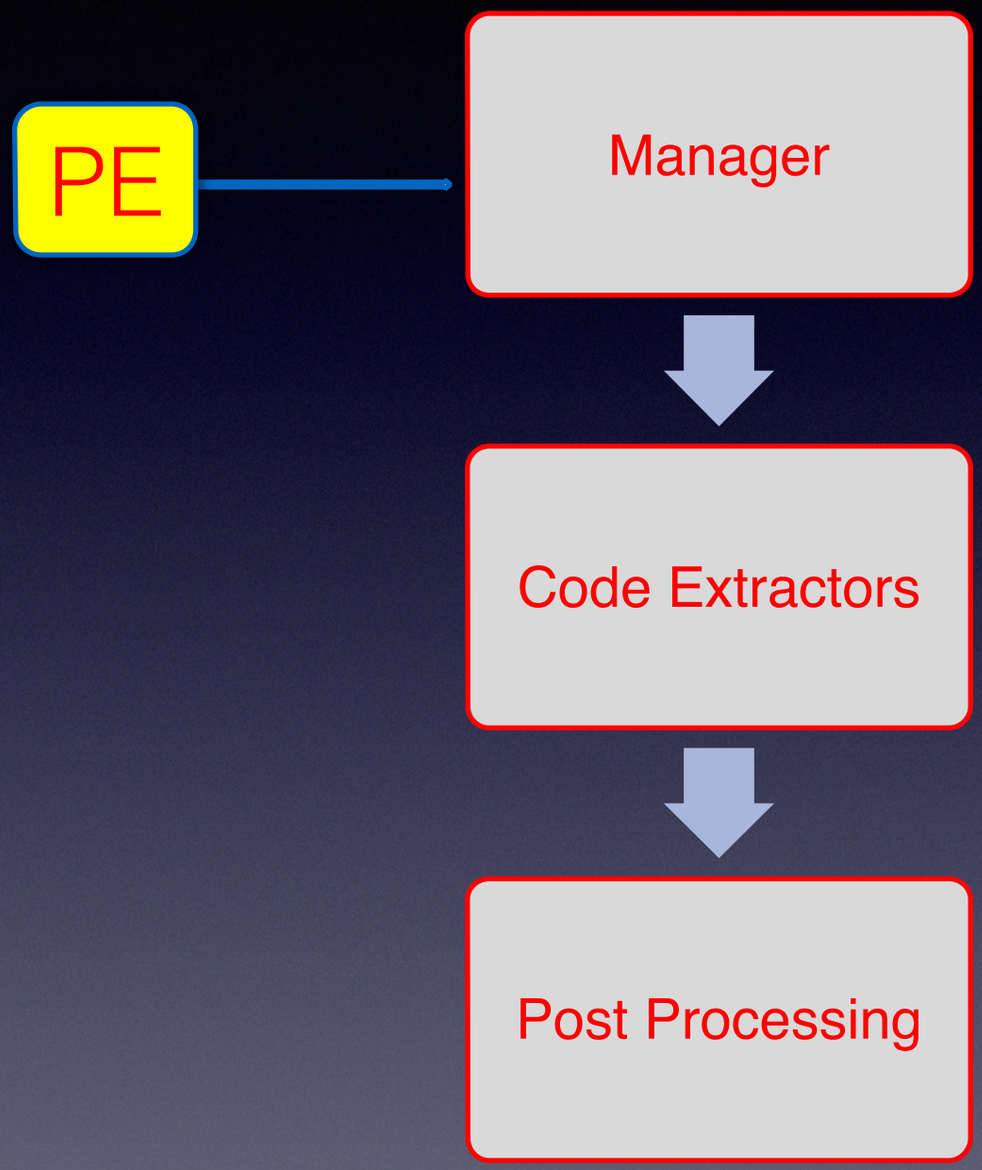
```
0:013> u 7532c87e l16
WININET!InternetCloseHandle:
7532c87e e9cd8ae28c      jmp     02155350
7532c883 83ec38          sub     esp,38h
7532c886 56              push    esi
7532c887 6a38            push    38h
7532c889 8d45c8          lea     eax,[ebp-38h]
7532c88c 6a00            push    0
7532c88e 50              push    eax
7532c88f e8706ffeff      call    WININET!memset (75313804)
7532c894 83c40c          add     esp,0Ch
7532c897 8d45c8          lea     eax,[ebp-38h]
7532c89a 50              push    eax
7532c89b e8cd86feff      call    WININET!InternetSaveThreadInfo (75314f6d)
7532c8a0 ff7508          push    dword ptr [ebp+8]
7532c8a3 e817000000      call    WININET!InternalInternetCloseHandle (7532c8bf)
```

```
0:013> u 02155350 l16
02155350 55              push    ebp
02155351 8bec            mov     ebp,esp
02155353 56              push    esi
02155354 8b7508          mov     esi,dword ptr [ebp+8]
02155357 57              push    edi
02155358 56              push    esi
02155359 ff151cc11802    call    dword ptr ds:[218C11Ch]
0215535f 8bf8            mov     edi,eax
02155361 3b356cc11802    cmp     esi,dword ptr ds:[218C16Ch]
02155367 7409            je      02155372
02155369 8bce            mov     ecx,esi
0215536b e800f8ffff      call    02154b70
02155370 8bc7            mov     eax,edi
02155372 5f              pop     edi
02155373 5e              pop     esi
02155374 5d              pop     ebp
02155375 c20400          ret     4
```

KEEP
CALM
AND
Keep
Guessing

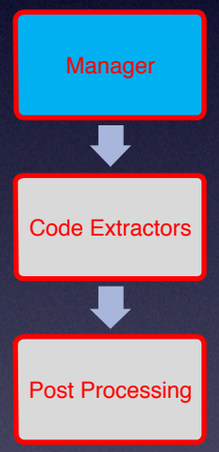
VolatilityBot

- Automated
- Modular
- Extraction of various artifacts


PE
Manager
Code Extractors
Post Processing

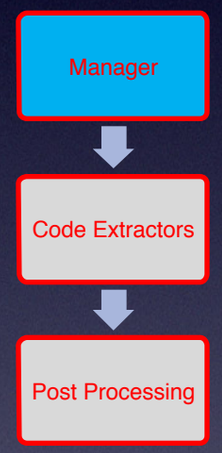
Manager

- Can theoretically manage an unlimited quantity of machines.
- Tags - Multiple tags can be defined on execution
- Dynamic Tags - Some post processing modules add tags to the sample. i.e.: Code_Injection, Hooks_API



Machines

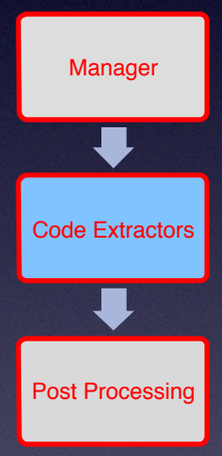
- Abstract model of a machine that has five basic functions:
  - Revert
  - Suspend
  - Start
  - Get Memory Path
  - Cleanup



VOLATILITY BOT

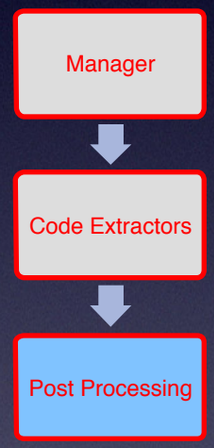
Code Extractors

- Injected Code
- Kernel Modules
- New Processes
- Entire Address Spaces
- Hooks



Post Process Modules

- YARA (And YARA Semantic Analysis)
- Strings
- Basic Static PE analysis
- Generation of IDC file with imports from memory
- Fix PE header (sections, image base…)



VOLATILITY BOT

YARA Semantic Analysis

- Dynamic YARA rules
- Detect specific behaviors

Efficacy & Results

Virus Share Malware Subset

Total Samples	3875	
Samples with at least one successful dump	3395	88%
New processes dumped	3363	86%
Injected Code extractions	992	25%
Kernel Modules dumped	119	0.03%

88% Success Rate

VOLATILITY BOT

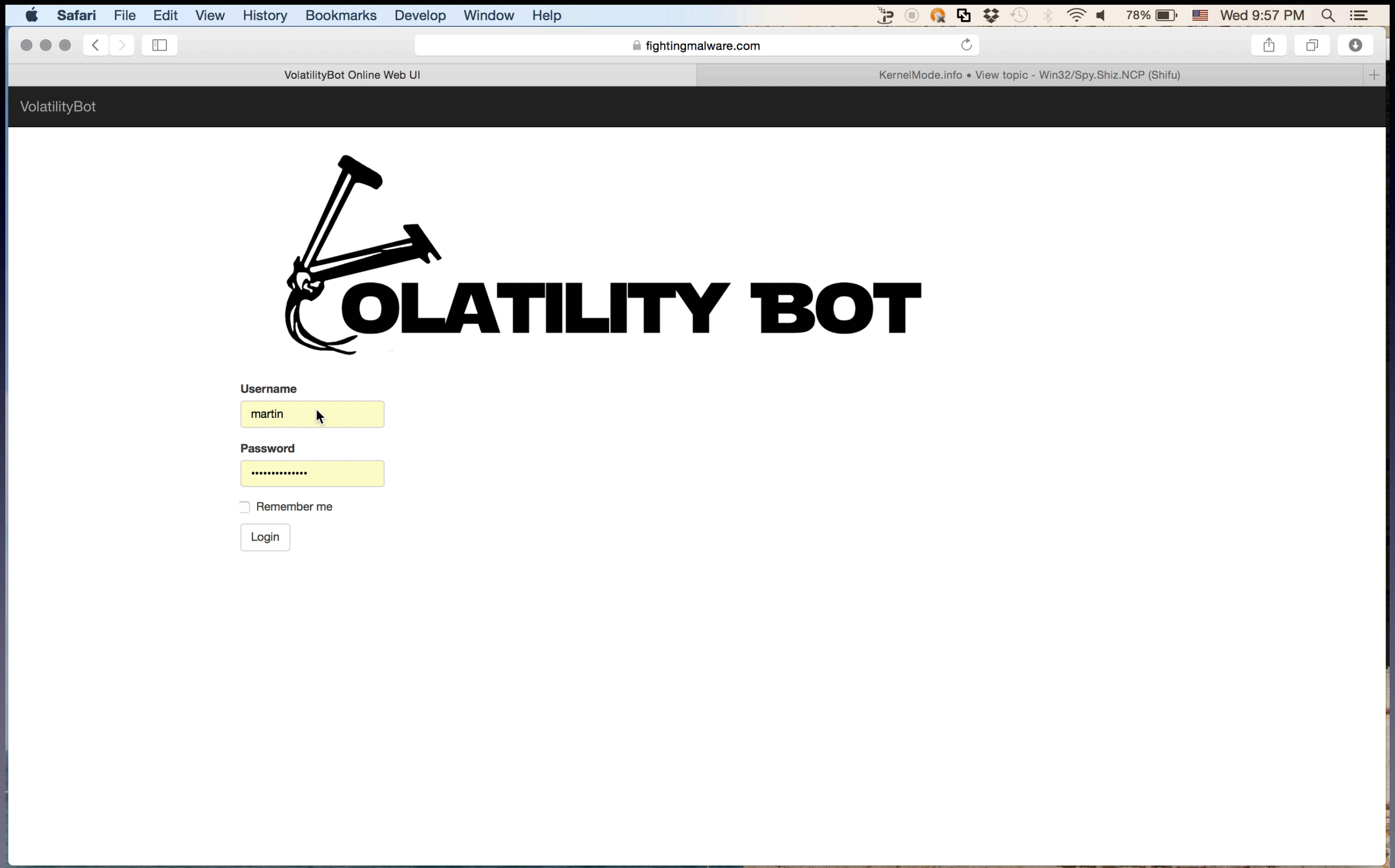
Malware Families Subset

Total Samples	68	
Samples with at least one successful dump	63	92%
Injected Code extractions	41	60%
New processes dumped	31	45%
Kernel Modules dumped	4	0.05%

92% Success Rate

VOLATILITY BOT

Demo Time!


Safari File Edit View History Bookmarks Develop Window Help
78% Wed 9:57 PM
fightingmalware.com
VolatilityBot Online Web UI
KernelMode.info • View topic - Win32/Spy.Shiz.NCP (Shifu)
VolatilityBot
VOLATILITY BOT
Username
martin
Password
Remember me
Login

What's Next?

- Automated Dumping of injected shellcode
- Extraction of malware configurations
- Additional information extraction (URLs,Mutexes)

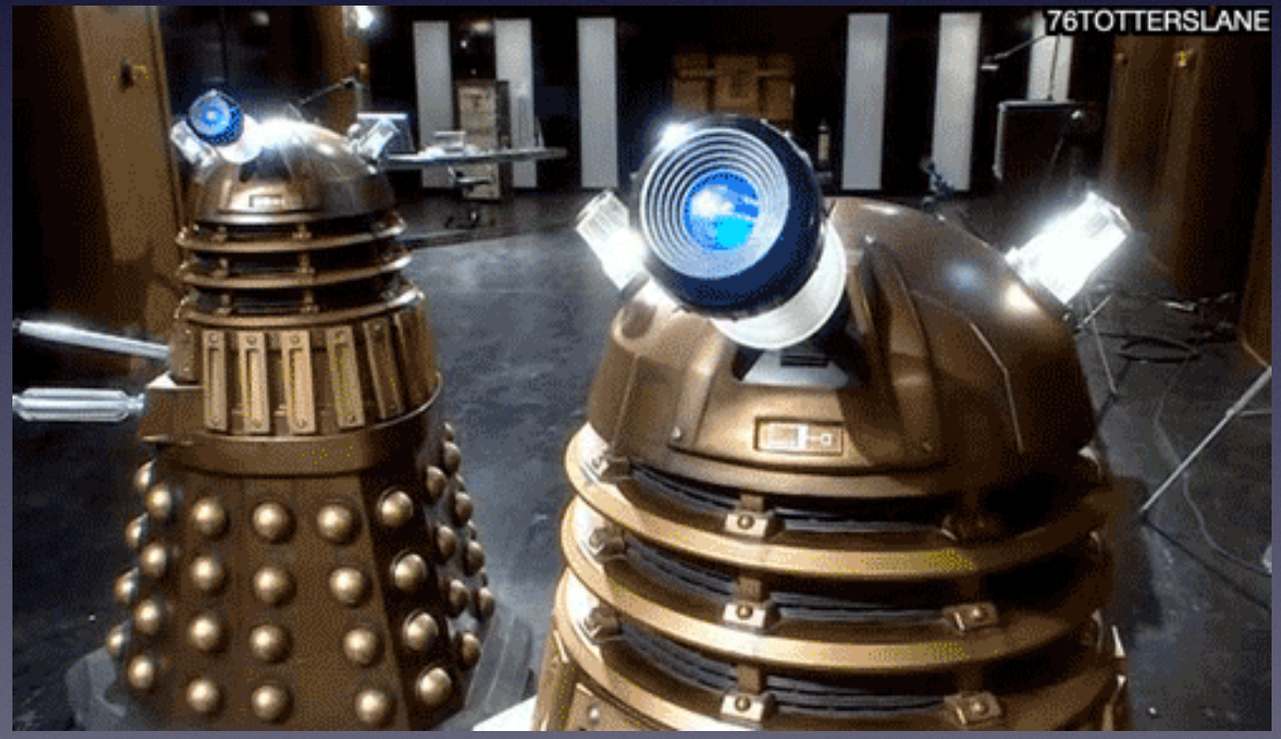
Caveats

- False positives
- Anti-Research mechanisms

Cool! Where can I get it?

- BitBucket Repository: https://bitbucket.org/martink90/volatilitybot_public/overview
- Communicate with me, via Mail in order to get the source code (kormanmartin@gmail.com)
- Use my Web-Service: https://fightingmalware.com
- @MartinKorman
- blog.fightingmalware.com

Questions?



Thanks for you time!

VOLATILITY BOT