
Finders Keepers (KPot Stealers)

whois lilly.chalupowski

Table: *who.is results*

Name	Lilly Chalupowski
Status	Employed
Creation Date	1986
Expiry	A Long Time from Now (Hopefully)
Registrant Name	GoSecure
Administrative Contact	Travis Barlow
Job	TITAN Malware Research Lead

Agenda

What will we cover?

- Disclaimer
- Reverse Engineering
- Tools of the Trade
- Injection Techniques
- Workshop

GoSecure

disclaimer_0.log

The tools and techniques covered in this presentation can be dangerous and are being shown for educational purposes.

It is a violation of Federal laws to attempt gaining unauthorized access to information, assets or systems belonging to others, or to exceed authorization on systems for which you have not been granted.

Only use these tools with/on systems you own or have written permission from the owner. I (the speaker) do not assume any responsibility and shall not be held liable for any illegal use of these tools.

Disclaimer

Don't be a Fool

disclaimer_1.log

I (the speaker) do not assume any responsibility and shall not be held liable for anyone who infects their machine with the malware supplied for this workshop.

If you need help on preventing the infection of your host machine please raise your hand during the workshop for assistance before you run anything.

The malware used in this workshop can steal your data, shutdown nuclear power plants, encrypt your files and more.

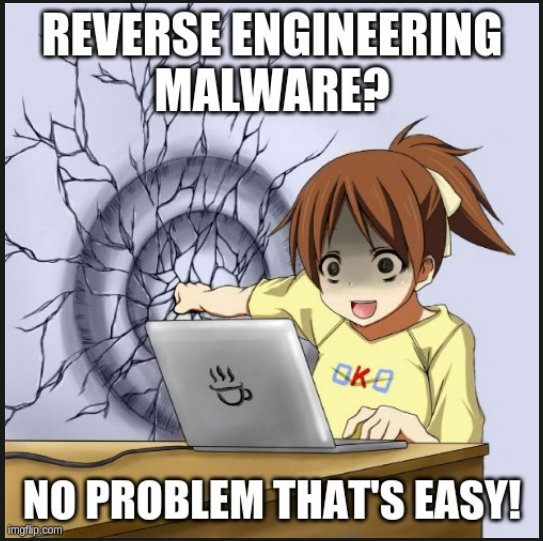
Inspiration

John F. Kennedy

GoSecure

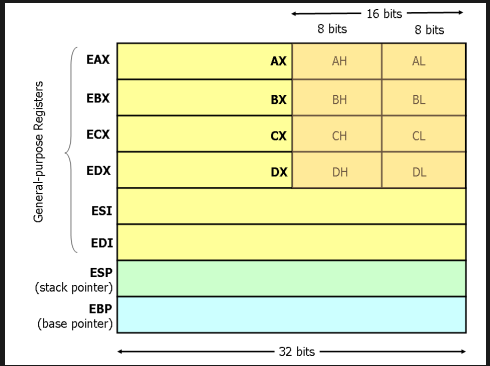
john_f_kennedy.log

We choose to reverse engineer! We choose to reverse engineer... We choose to reverse engineer and do the other things, not because they are easy, but because they are hard; because that goal will serve to organize and measure the best of our energies and skills, because that challenge is one that we are willing to accept, one we are unwilling to postpone, and one we intend to win, and the others, too. - John F. Kennedy


REVERSE ENGINEERING MALWARE?
NO PROBLEM THAT'S EASY!
imgflip.com

Registers

- EAX - Return Value of Functions
- EBX - Base Index (for use with arrays)
- ECX - Counter in Loops
- EDI - Destination Memory Operations
- ESI - Source Memory Operations
- ESP - Stack Pointer
- EBP - Base Frame Pointer

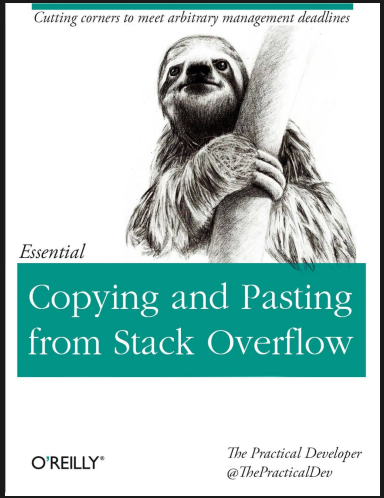


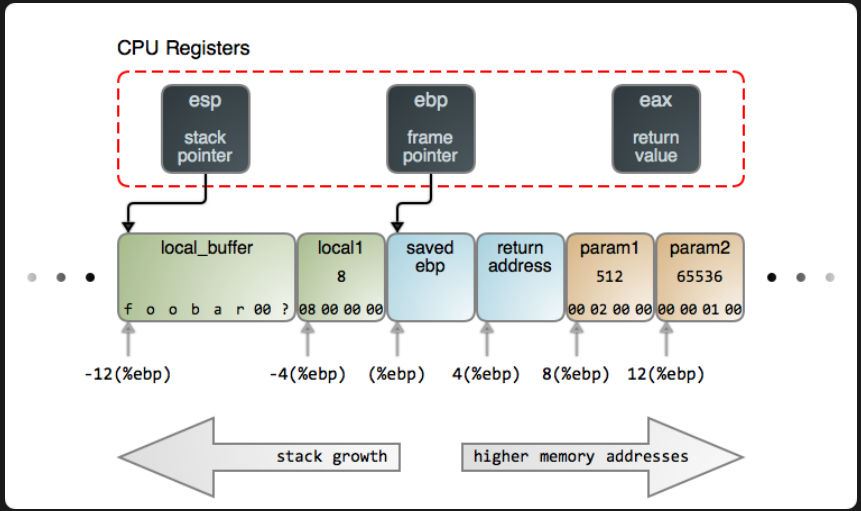
Did You Know: In computer architecture, a processor register is a quickly accessible location available to a computer's central processing unit (CPU).

Registers


We Rent Tools
Professional Tools For Professional Results
Tool Rental
WE RENT
Windows XP

- Last-In First-Out
- Downward Growth
- Function Local Variables
- ESP
- Increment / Decrement = 4
  - Double-Word Aligned



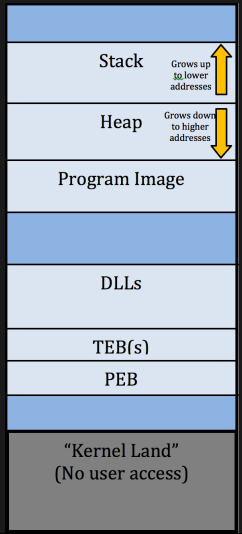

CPU Registers
esp
stack pointer
ebp
frame pointer
eax
return value
local_buffer
f o o b a r 00 ?
local1
8
08 00 00 00
saved ebp
return address
param1
512
00 02 00 00
param2
65536
00 00 01 00
-12(%ebp)
-4(%ebp)
(%ebp)
4(%ebp)
8(%ebp)
12(%ebp)
stack growth
higher memory addresses

- Conditionals
  - CMP
  - TEST
  - JMP
  - JNE
  - JNZ
- EFLAGS
  - ZF / Zero Flag
  - SF / Sign Flag
  - CF / Carry Flag
  - OF/Overflow Flag

- CDECL
  - Arguments Right-to-Left
  - Return Values in EAX
  - Caller Function Cleans the Stack
- STDCALL
  - Used in Windows Win32API
  - Arguments Right-to-Left
  - Return Values in EAX
  - The Callee function cleans the stack, unlike CDECL
  - Does not support variable arguments
- FASTCALL
  - Uses registers as arguments
  - Useful for shellcode

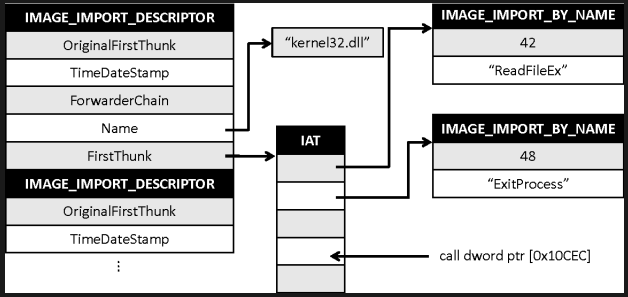
Windows Memory Structure

- Stack - Grows up to lower addresses
- Heap - Grows down to higher addresses
- Program Image
- TEB - Thread Environment Block
  - GetLastError()
  - GetVersion()
  - Pointer to the PEB
- PEB - Process Environment Block
  - Image Name
  - Global Context
  - Startup Parameters
  - Image Base Address
  - IAT (Import Address Table)



IAT and IDT

- Identical to the IDT (Import Directory Table)
- Binding - The process of where functions are mapped to their virtual addresses overwriting the IAT
- Often the IDT and IAT must be rebuilt when packing and unpacking malware



Assembly

- Common Instructions
  - MOV
  - LEA
  - XOR
  - PUSH
  - POP

Assembly CDECL (Linux)

reverse_engineering: 0x09

GoSecure

cdecl.c

```
__cdecl int add_cdecl(int a, int b){
  return a + b;
}
int x = add_cdecl(2, 3);
```

Assembly CDECL (Linux)

GoSecure

cdecl.asm

```
_add_cdecl:
  push ebp
  mov ebp, esp
  mov eax, [ebp + 8]   ; get 3 from the stack
  mov edx, [ebp + 12]  ; get 2 from the stack
  add eax, edx         ; add values to eax
  pop ebp
  ret
_start:
  push 3               ; second argument
  push 2               ; first argument
  call _add_cdecl
  add esp, 8
```

Assembly STDCALL (Windows)

reverse_engineering: 0x0b

GoSecure

stdcall.c

```
__stdcall int add_stdcall(int a, int b){
  return a + b;
}
int x = add_stdcall(2, 3);
```

Assembly STDCALL (Windows)

stdcall.asm

```
_add_stdcall:
  push ebp
  mov ebp, esp
  mov eax, [ebp + 8]  ; set eax to 3
  mov edx, [ebp + 12] ; set edx to 2
  add eax, edx
  pop ebp
  ret 8               ; how many bytes to pop
_start:               ; main function
  push 3              ; second argument
  push 2              ; first argument
  call _add_stdcall
```

Assembly FASTCALL

reverse_engineering: 0x0d

GoSecure

cdecl.c

```
__fastcall int add_fastcall(int a, int b){
  return a + b;
}
int x = add_fastcall(2, 3);
```

Assembly FASTCALL

reverse_engineering: 0x0e

GoSecure

fastcall.asm

```
_add_fastcall:
  push ebp
  mov ebp, esp
  add eax, edx          ; add and save result in eax
  pop ebp
  ret
_start:
  mov eax, 2            ; first argument
  mov edx, 3            ; second argument
  call _add_fastcall
```

Guess the Calling Convention

reverse_engineering: 0x0f

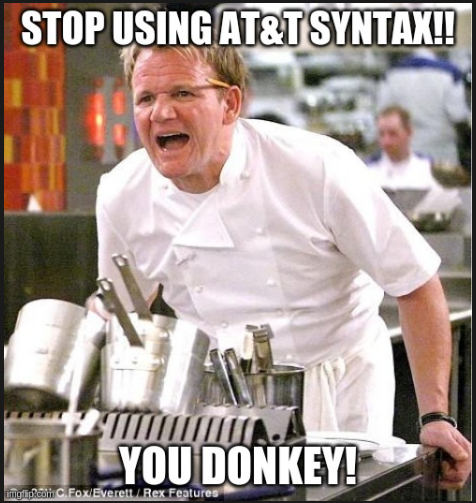
GoSecure

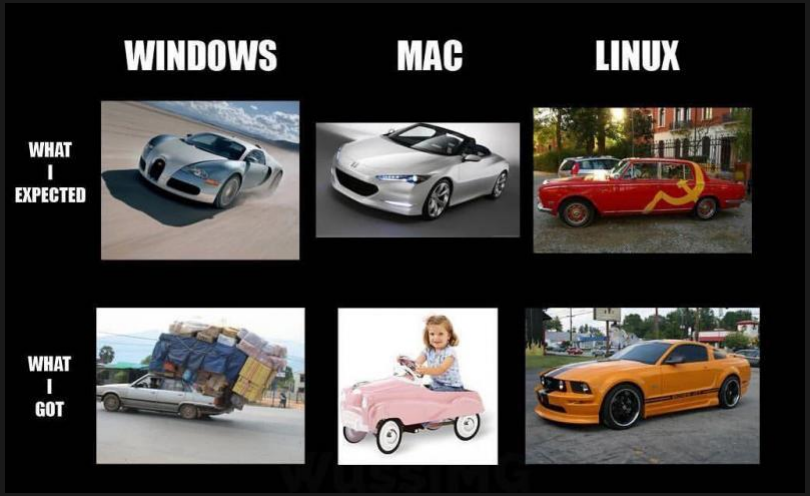
hello.asm

```
section     .text                   ; the code section
global      _start                  ; tell linker entrypoint
_start:
  mov     edx,len                   ; message length
  mov     ecx,msg                   ; message to write
  mov     ebx,1                     ; file descriptor stdout
  mov     eax,4                     ; syscall number for write
  int     0x80                      ; linux x86 interrupt
  mov     eax,1                     ; syscall number for exit
  int     0x80                      ; linux x86 interrupt
section     .data                   ; the data section
  msg     db  'Hello, world!',0x0   ; null terminated string
  len     equ \$ - msg              ; message length
```

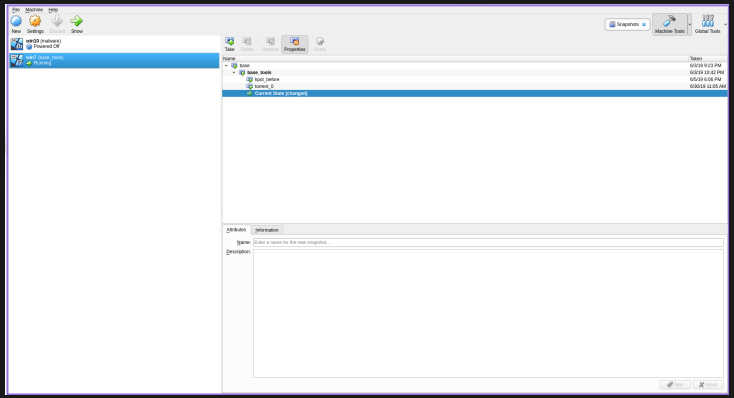
terminal

```
malware@work ~$ nasm -f elf32 -o hello.o hello.asm
malware@work ~$ ld -m elf_i386 -o hello hello.o
malware@work ~$ ./hello
Hello, World!
malware@work ~$
```

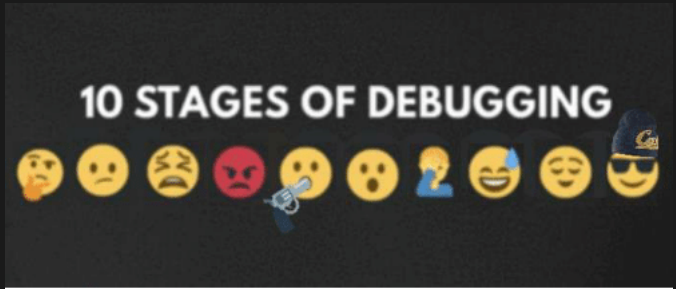

STOP USING AT&T SYNTAX!!
YOU DONKEY!
© Fox/Everett / Rex Features


WINDOWS
MAC
LINUX
WHAT I EXPECTED
WHAT I GOT

- Snapshots
- Security Layer
- Multiple Systems

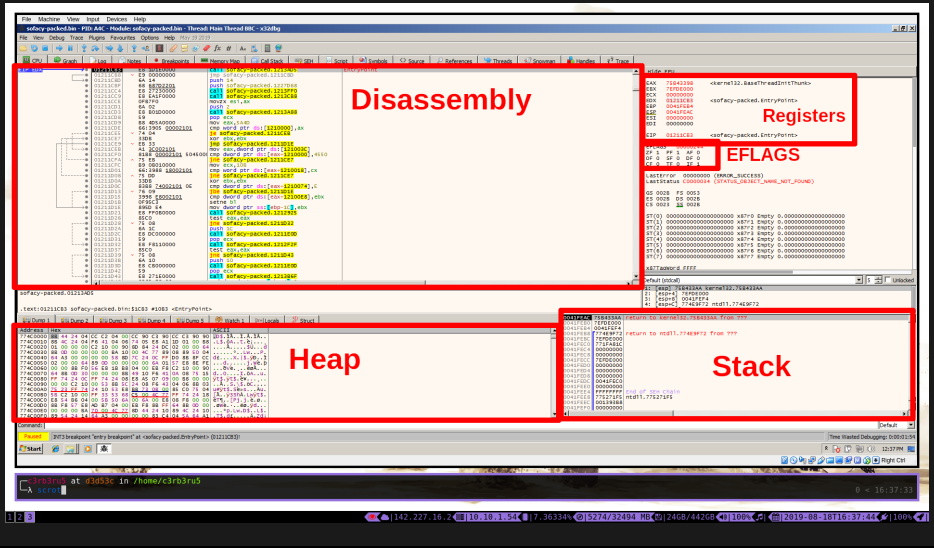


- Resolving APIs
- Dumping Memory
- Modify Control Flow
- Identify Key Behaviors



x64dbg

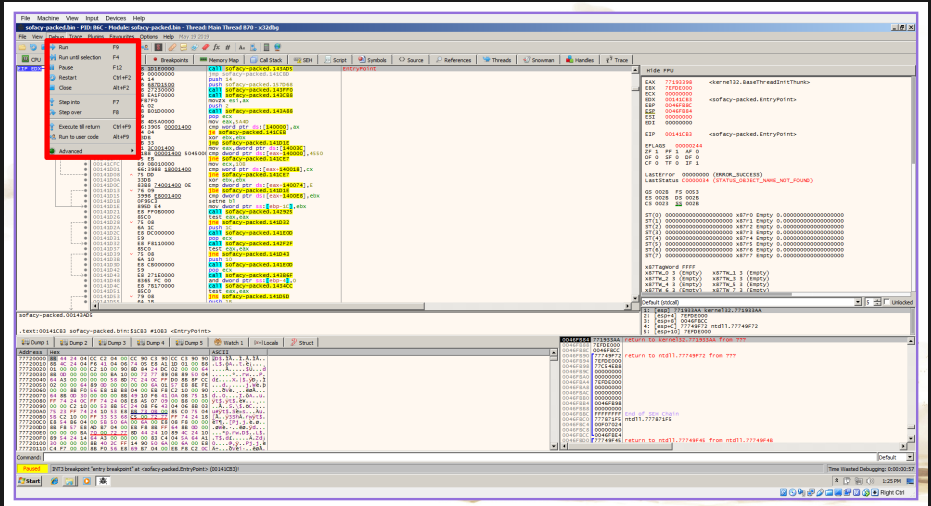
tools_of_the_trade: 0x02



x64dbg

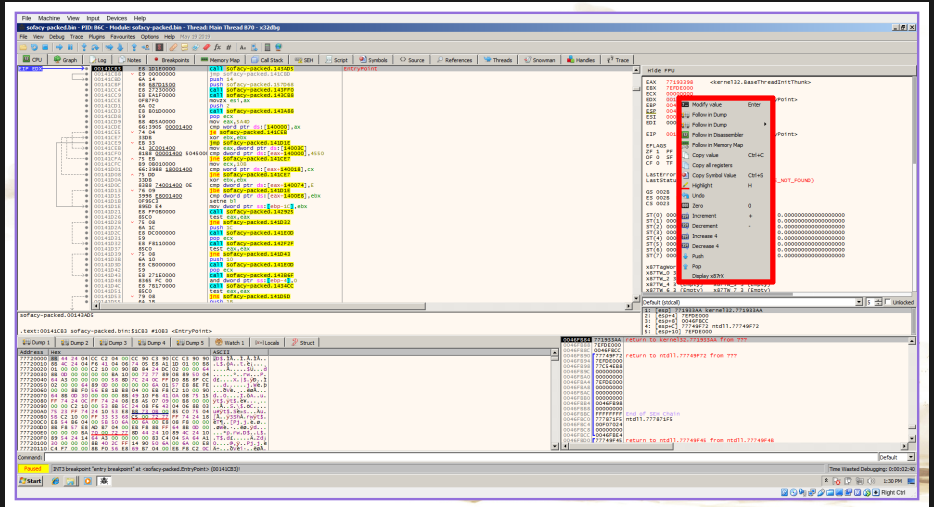
tools_of_the_trade: 0x03

GoSecure

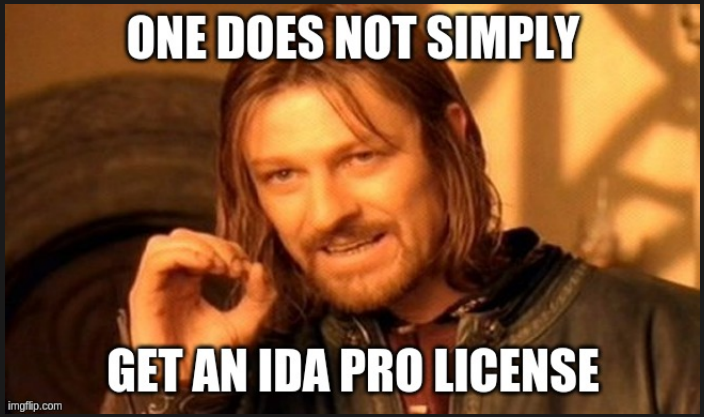


x64dbg

tools_of_the_trade: 0x04

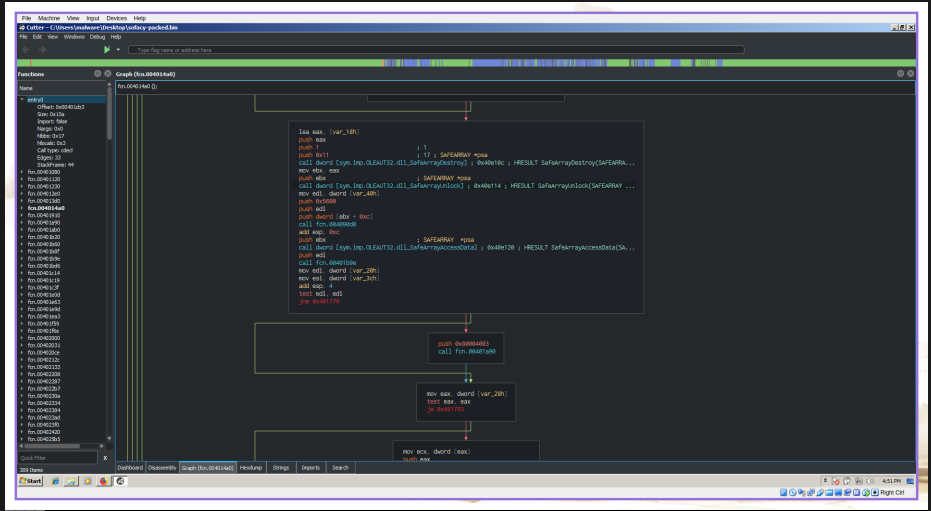


- Markup Reverse Engineered Code
- Control Flow Navigation
- Pseudo Code



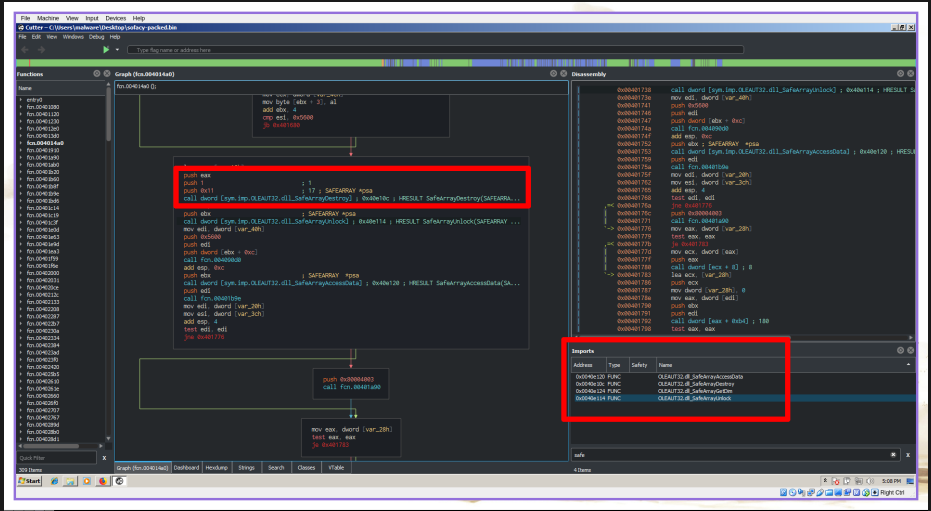
Cutter

tools_of_the_trade: 0x06



Cutter

tools_of_the_trade: 0x07



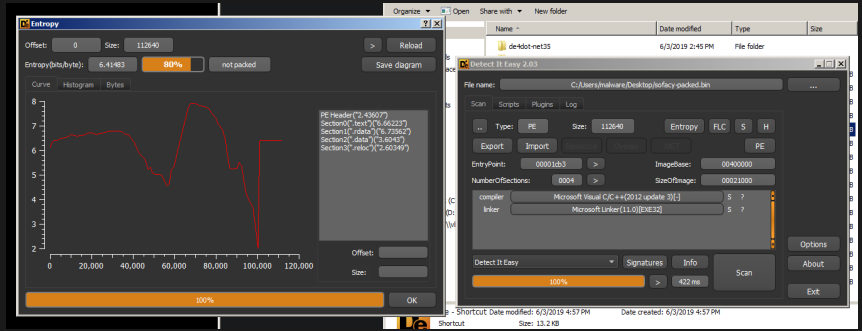
Radare2

tools_of_the_trade: 0x08

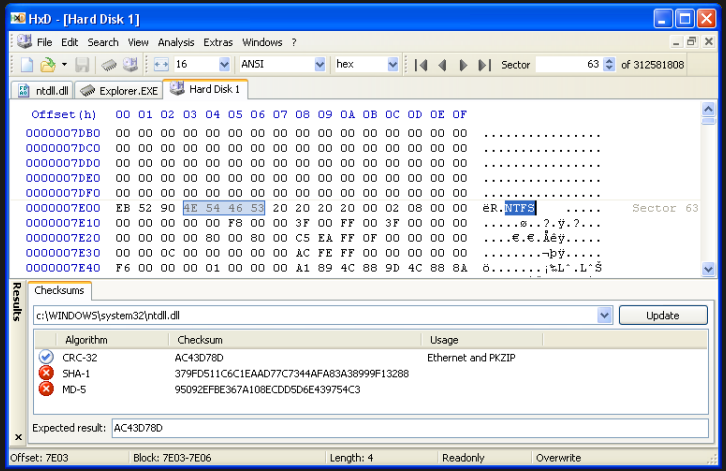
Detect it Easy

tools_of_the_trade: 0x09

- Type
- Packer
- Linker
- Entropy



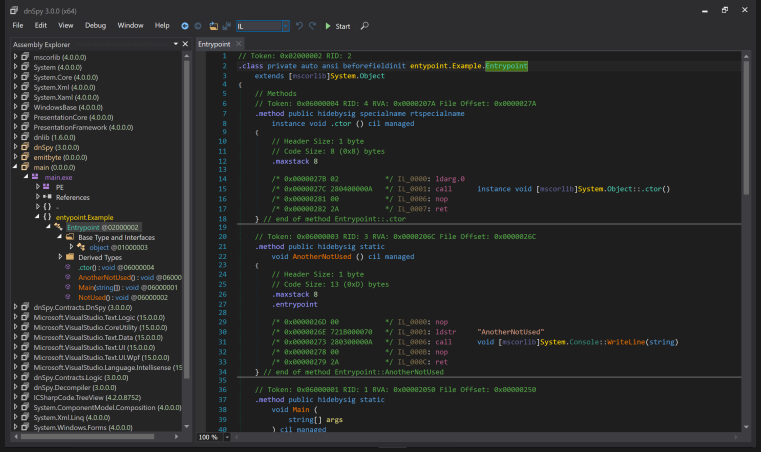
- Modify Dumps
- Read Memory
- Determine File Type



DnSpy

tools_of_the_trade: 0x0b

- Code View
- Debugging
- Unpacking



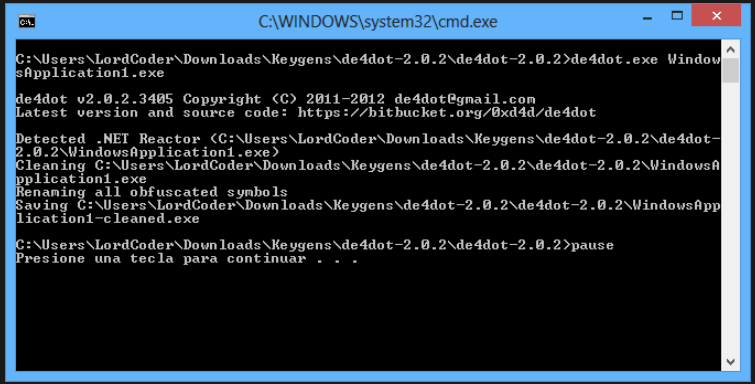
terminal

```
malware@work ~$ file sample.bin
sample.bin: PE32 executable (GUI) Intel 80386, for MS Windows
malware@work ~$ exiftool sample.bin > metadata.log
malware@work ~$ hexdump -C -n 128 sample.bin | less
malware@work ~$ VBoxManage list vms
"win10" {53014b4f-4c94-49b0-9036-818b84a192c9}
"win7" {942cde2e-6a84-4edc-b98a-d7326b4662ee}
malware@work ~$ VBoxManage startvm win7
malware@work ~$
```

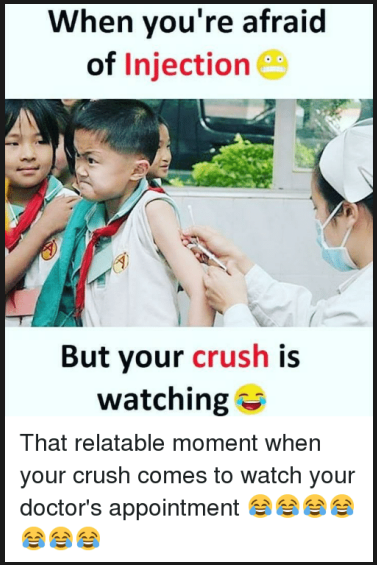
de4dot

tools_of_the_trade: 0xd

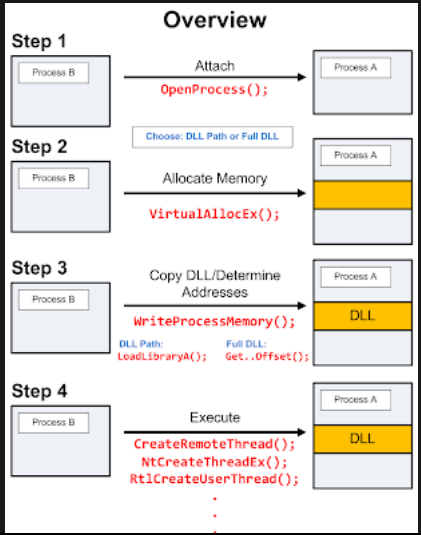
- Automated
- Deobfuscation
- Unpacking



- Unpacking
- Process Hollowing Extraction
- Dump Processes


When you're afraid of Injection 😬
But your crush is watching 😂
That relatable moment when your crush comes to watch your doctor's appointment 😂😂😂😂😂😂😂

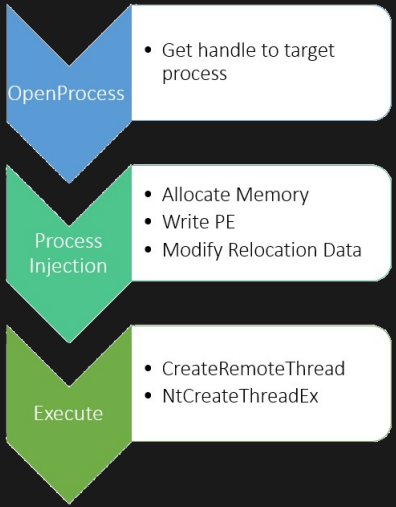
- Get Handle to Target Process
- Allocate Memory
- Write Memory
- Execute by use of Remote Thread



PE (Portable Executable) Injection

injection_techniques: 0x01

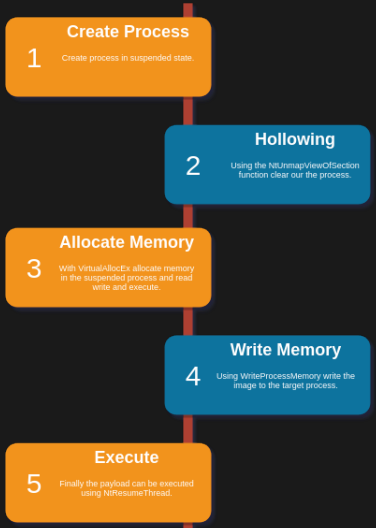
- Obtain Handle to Target Process
- Inject Image to Target Process
- Modify Base Address
- Modify Relocation Data
- Execute your Payload



Process Hollowing

injection_techniques: 0x02

- Create Suspended Process
- Hollow Process with NtUnmapViewOfSection
- Allocate Memory in Process
- Write Memory to Process
- Resume Thread / Process

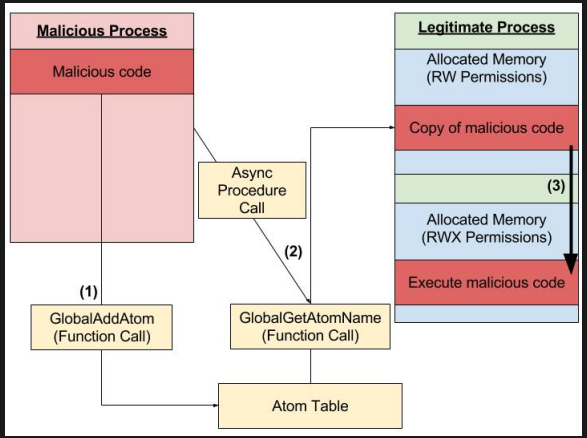


Atomic bomb test in Italy, 1957, colorized

Atom Bombing

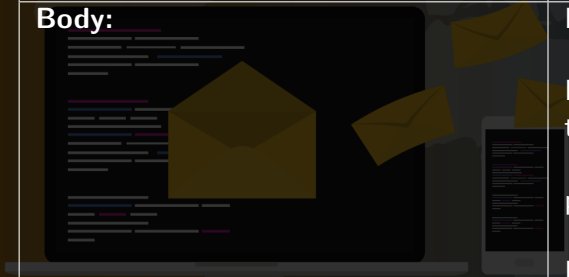
injection_techniques: 0x04

- Open Target Process
- Get Handle to Alertable Thread
- Find Code Cave
- Shellcode to Call ZwAllocateVirtualMemory and memcpy
- Call GlobalAddAtom
- Suspend Target Thread
- NtQueueApcThread
- Resume Target Thread



- 5466c52191ddd1564b4680060dc329cb

MalSpam Email	
From:	<sidney.m[at]carmellaw[.]com>
Subject:	Wire confirmation
Attached:	img-000310519000.img
Body:	Hello, Kindly find attached the confirmation for your reference please revert back ASAP Regards Sidney morris



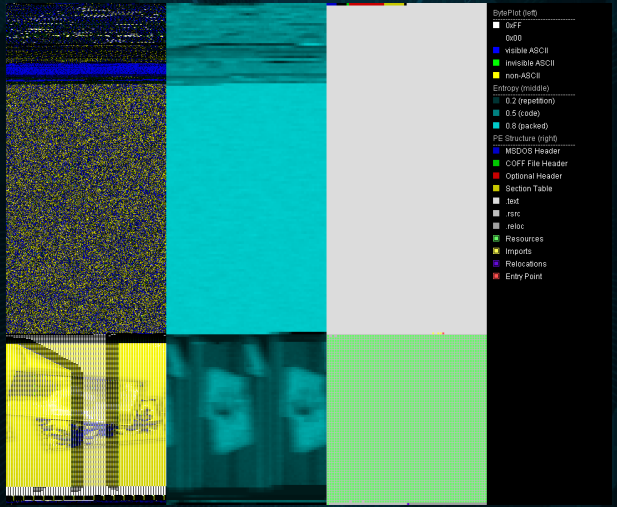
- Inbox Detection and Response (IDR) Team
- Lawyers
- Real Estate Agents
- Accountants
- Tax Firms

terminal

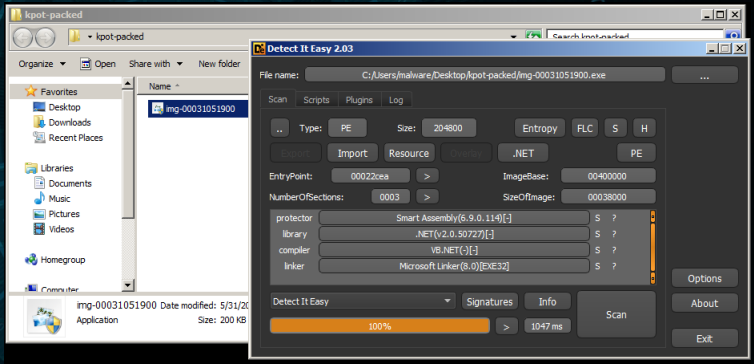
```
malware@localhost ~$ file img-000310519000.img
UDF filesystem data (version 1.5) NEW_FOLDER_2_
malware@localhost ~$ 7z e img-000310519000.img
malware@localhost ~$ file img-00031051900.exe
PE32 executable (GUI) Intel 80386 Mono/.Net assembly, for MS
Windows
malware@localhost ~$
```

Kpot

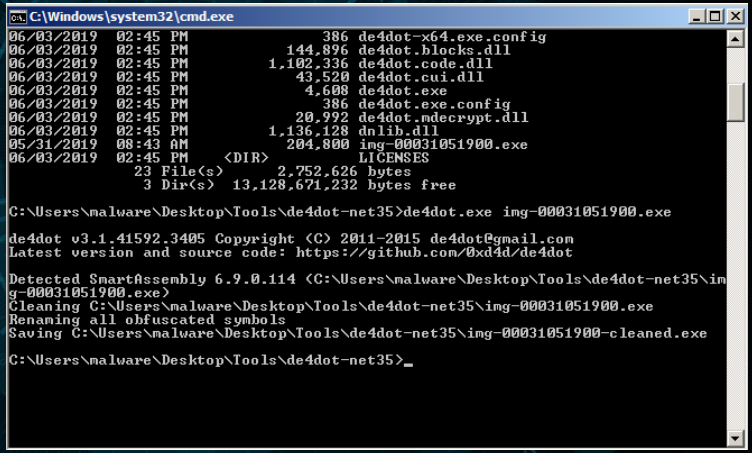
Operation Lawyer Loot - Portex Analyzer Entropy

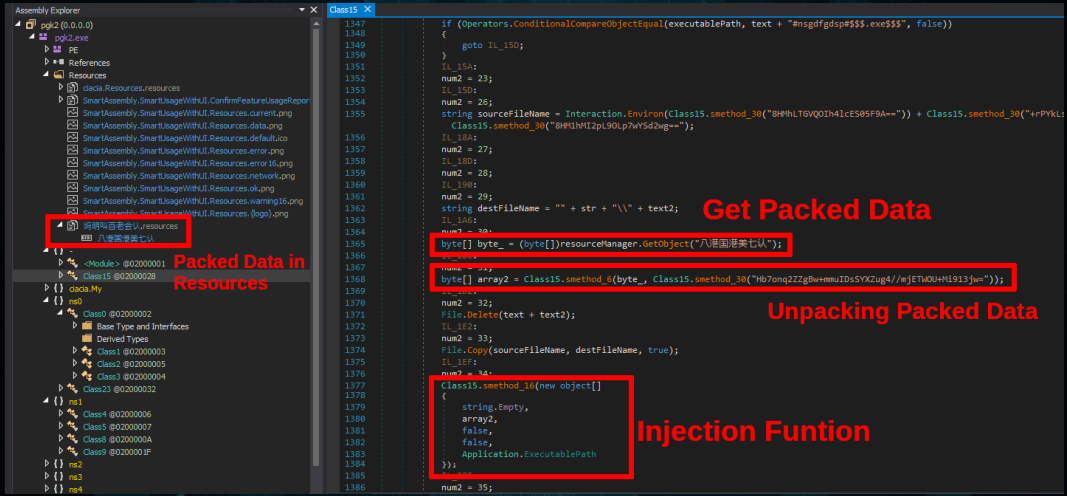


- Packed with Smart Assembly
- Let's now try de4dot



- de4dot detected smart assembly
- Let's have a look in DnSpy!




Assembly Explorer
Class15
Packed Data in Resources
Get Packed Data
Unpacking Packed Data
Injection Funtion

- Uses Rijndael for string obfuscation
- We can use C# in Visual Studio!

```
static string smethod_30(string string_0)
{
    string s = "美明八零会家美";
    RijndaelManaged rijndaelManaged = new RijndaelManaged();
    MD5CryptoServiceProvider md5CryptoServiceProvider = new MD5CryptoServiceProvider();
    string result;
    try
    {
        byte[] array = new byte[32];
        byte[] sourceArray = md5CryptoServiceProvider.ComputeHash(Encoding.ASCII.GetBytes(s));
        Array.Copy(sourceArray, 0, array, 0, 10);
        Array.Copy(sourceArray, 0, array, 15, 10);
        rijndaelManaged.Key = array;
        rijndaelManaged.Mode = CipherMode.ECB;
        ICryptoTransform cryptoTransform = rijndaelManaged.CreateDecryptor();
        byte[] array2 = Convert.FromBase64String(string_0);
        string @string = Encoding.ASCII.GetString(cryptoTransform.TransformFinalBlock(array2, 0, array2.Length));
        result = @string;
    }
    catch (Exception ex)
    {
    }
    return result;
}
```

```
class Program
{
    1 reference
    static string decode(string string_0)
    {
        string s = "美明八零会家美";
        RijndaelManaged rijndaelManaged = new RijndaelManaged();
        MD5CryptoServiceProvider md5CryptoServiceProvider = new MD5CryptoServiceProvider();
        string result;
        byte[] array = new byte[32];
            byte[] sourceArray = md5CryptoServiceProvider.ComputeHash(Encoding.ASCII.GetBytes(s));
            Array.Copy(sourceArray, 0, array, 0, 10);
            Array.Copy(sourceArray, 0, array, 15, 10);
            rijndaelManaged.Key = array;
            rijndaelManaged.Mode = CipherMode.ECB;
            ICryptoTransform cryptoTransform = rijndaelManaged.CreateDecryptor();
            byte[] array2 = Convert.FromBase64String(string_0);
            string @string = Encoding.ASCII.GetString(cryptoTransform.TransformFinalBlock(array2, 0, array2.Length));
            result = @string;
            return result;
    }
    0 references
    static void Main(string[] args)
    {
        Console.WriteLine(decode(args[0]));
    }
}
```

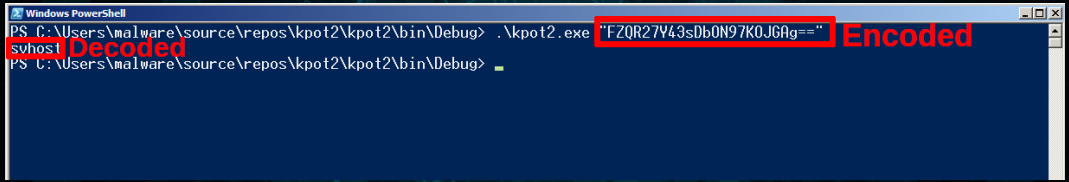


Figure: Unpacking KPot - String Deobfuscation

```
static bool smethod_27(object[] object_0)
{
    Class5.Class6 @class = new Class5.Class6();
    Class8.Delegate1 @delegate = Class5.smethod_0<Class8.Delegate1>(Class15.smethod_36("uzbZJOXM1ti/o9VzPqHexQ==", Class15.smethod_36("kVdWt2m955ig9LvIKqBJ7Q==");
    Class8.Delegate2 delegate2 = Class5.smethod_0<Class8.Delegate2>(Class15.smethod_36("uzbZJOXM1ti/o9VzPqHexQ==", Class15.smethod_36("MHCx1wPRkCQPwhR5g7XQekaqAMuhh60oEKoaAbNvuIk=");
    Class8.Delegate3 delegate3 = Class5.smethod_0<Class8.Delegate3>(Class15.smethod_36("uzbZJOXM1ti/o9VzPqHexQ==", Class15.smethod_36("QCVByD5Kb4dymngcmPozmijexOC14Oik/RQZbeXPCDo=");
    Class8.Delegate4 delegate4 = Class5.smethod_0<Class8.Delegate4>(Class15.smethod_36("uzbZJOXM1ti/o9VzPqHexQ==", Class15.smethod_36("FjiFXnN7CBcQa33vgPA3BSSYktkkXqZt0GLjZ3iitb8=");
    Class8.Delegate5 delegate5 = Class5.smethod_0<Class8.Delegate5>(Class15.smethod_36("eDAECoHuT4guqKSvOGp4yg==", Class15.smethod_36("e4WJnVJ/R9vGiA+MBU/59/rUjih2+FgywIB80+3YkVw=");
    Class8.Delegate6 delegate6 = Class5.smethod_0<Class8.Delegate6>(Class15.smethod_36("eDAECoHuT4guqKSvOGp4yg==", Class15.smethod_36("oKL4yaE9EBm/y9S+kqvCkaq4w3M21Bt7/g7gAFxXVD4=");
    Class8.Delegate7 delegate7 = Class5.smethod_0<Class8.Delegate7>(Class15.smethod_36("uzbZJOXM1ti/o9VzPqHexQ==", Class15.smethod_36("BGwD0JvCcYuc0GAsN3P9tw==");
    Class8.Delegate8 delegate8 = Class5.smethod_0<Class8.Delegate8>(Class15.smethod_36("eDAECoHuT4guqKSvOGp4yg==", Class15.smethod_36("iy1ljydF6nv9OhRn++t0LQ==");
    string text = (string)object_0[0];
    byte[] array = (byte[])object_0[1];
    bool flag = (bool)object_0[2];
    bool flag2 = (bool)object_0[3];
    string text2 = (string)object_0[4];
    int num = 0;
    string text3 = string.Format("\"{0}\"", text2);
    Class8.Struct1 @struct = default(Class8.Struct1);
    @class.struct0_0 = default(Class8.Struct0);
    @struct.uint_0 = Convert.ToUInt32(Marshal.SizeOf(typeof(Class8.Struct1)));
    bool result;
    try
    {
        Class5.Class6.Class7 class2 = new Class5.Class6.Class7();
        class2.class6_0 = @class;
        if (!string.IsNullOrEmpty(text))
```

Figure: Unpacking KPot - API Function Name Encryption

```
static string smethod_36(string string_0)
{
    string password = "fdfdfrtertert";
    string s = "fdfdfrtertert";
    string s2 = "@1B2c3D4sfg5F6g7H8";
    byte[] bytes = Encoding.ASCII.GetBytes(s2);
    byte[] bytes2 = Encoding.ASCII.GetBytes(s);
    byte[] array = Convert.FromBase64String(string_0);
    Rfc2898DeriveBytes rfc2898DeriveBytes = new Rfc2898DeriveBytes(password, bytes2, 2);
    byte[] bytes3 = rfc2898DeriveBytes.GetBytes(32);
    ICryptoTransform transform = new RijndaelManaged
    {
        Mode = CipherMode.CBC
    }.CreateDecryptor(bytes3, bytes);
    MemoryStream memoryStream = new MemoryStream(array);
    CryptoStream cryptoStream = new CryptoStream(memoryStream, transform, CryptoStreamMode.Read);
    byte[] array2 = new byte[checked(array.Length - 1 + 1)];
    int count = cryptoStream.Read(array2, 0, array2.Length);
    memoryStream.Close();
    cryptoStream.Close();
    return Encoding.UTF8.GetString(array2, 0, count);
}
```

Figure: Unpacking KPot - Decrypt Injection API Function Names

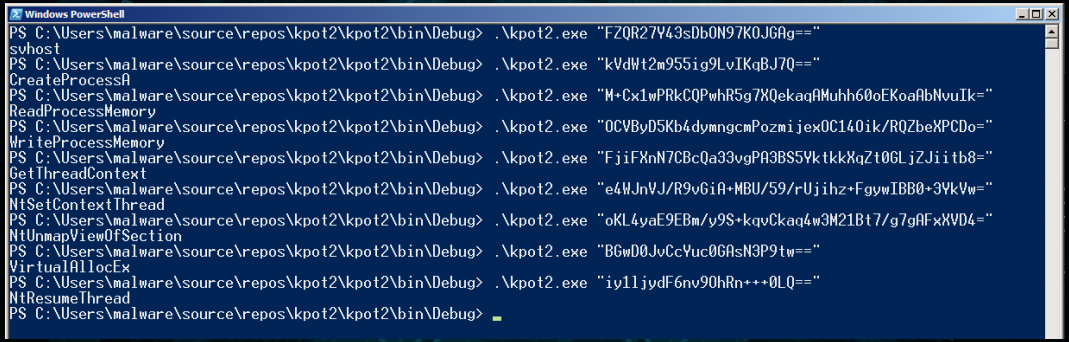
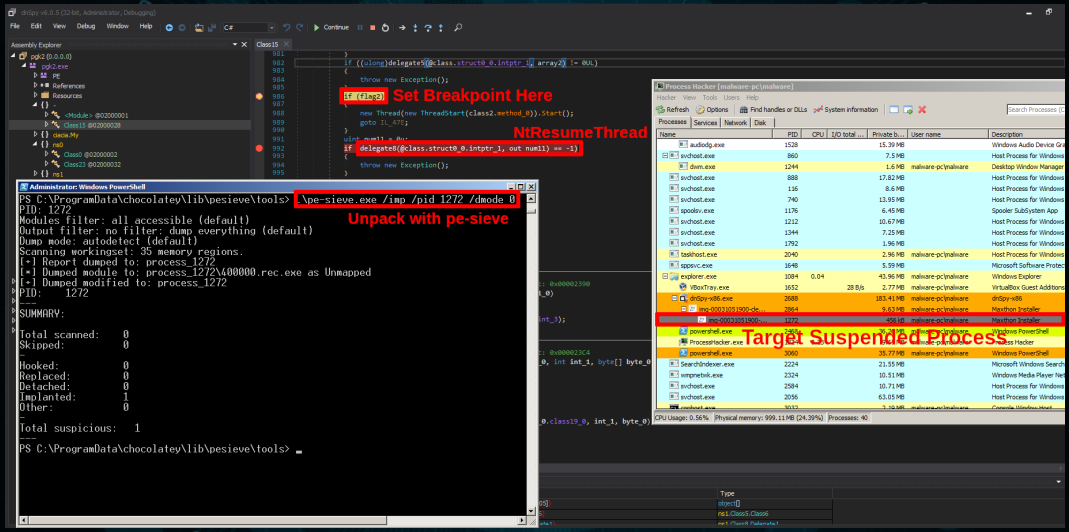
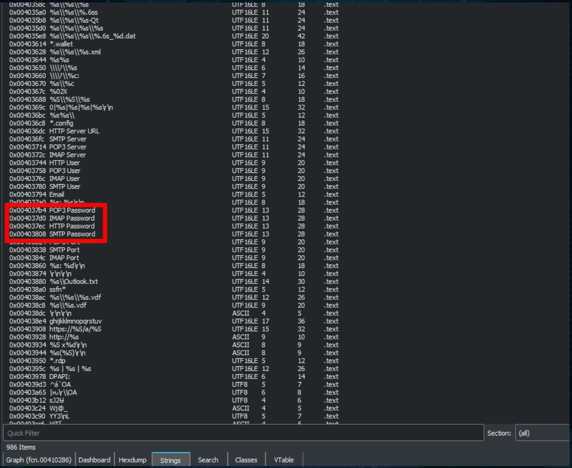


Figure: Unpacking KPot - Process Hollowing Functions

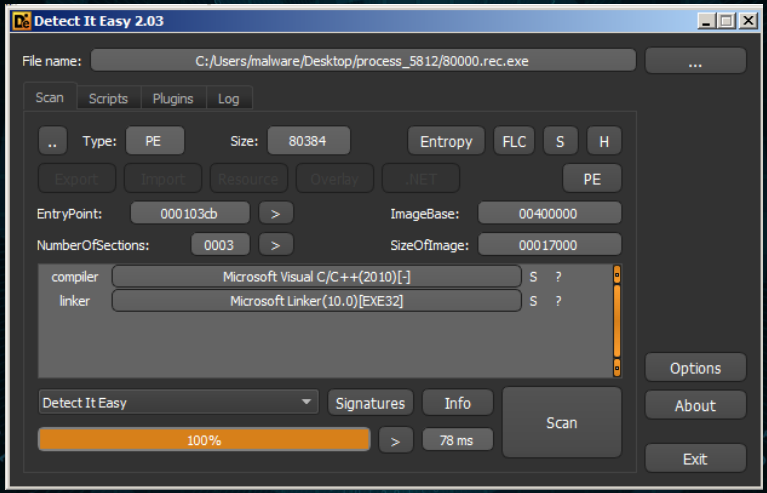

Set Breakpoint Here
NtResumeThread
Unpack with pe-sieve
Target Suspended Process
Administrator: Windows PowerShell
PS C:\ProgramData\chocolatey\lib\pesieve\tools> .\pe-sieve.exe /imp /pid 1272 /dmode 0
PID: 1272
Modules filter: all accessible (default)
Output filter: no filter: dump everything (default)
Dump mode: autodetect (default)
Scanning workingset: 35 memory regions.
[+] Report dumped to: process_1272
[*] Dumped module to: process_1272\400000.rec.exe as Unmapped
[+] Dumped modified to: process_1272
PID: 1272
SUMMARY:
Total scanned: 0
Skipped: 0
Hooked: 0
Replaced: 0
Detached: 0
Implanted: 1
Other: 0
Total suspicious: 1
PS C:\ProgramData\chocolatey\lib\pesieve\tools>

GoSecure

Operation Lawyer Loot - Stealer Strings



0x0040358c	%s\\%s\\%s	UTF16LE	8	18	.text
0x004035a0	%s\\%s\\%.6ss	UTF16LE	11	24	.text
0x004035b8	%s\\%s\\%s-Qt	UTF16LE	11	24	.text
0x004035d0	%s\\%s\\%s\\%s	UTF16LE	11	24	.text
0x004035e8	%s\\%s\\%s\\%.6s_%d.dat	UTF16LE	20	42	.text
0x00403614	*.wallet	UTF16LE	8	18	.text
0x00403628	%s\\%s\\%s.xml	UTF16LE	12	26	.text
0x00403644	%s%s	UTF16LE	4	10	.text
0x00403650	\\\\?\\%s	UTF16LE	6	14	.text
0x00403660	\\\\?\\%c:	UTF16LE	7	16	.text
0x00403670	%s\\%c	UTF16LE	5	12	.text
0x0040367c	%02X	UTF16LE	4	10	.text
0x00403688	%S\\%S\\%s	UTF16LE	8	18	.text
0x0040369c	0|%s|%s|%s|%s\r\n	UTF16LE	15	32	.text
0x004036bc	%s%s\\	UTF16LE	5	12	.text
0x004036c8	*.config	UTF16LE	8	18	.text
0x004036dc	HTTP Server URL	UTF16LE	15	32	.text
0x004036fc	SMTP Server	UTF16LE	11	24	.text
0x00403714	POP3 Server	UTF16LE	11	24	.text
0x0040372c	IMAP Server	UTF16LE	11	24	.text
0x00403744	HTTP User	UTF16LE	9	20	.text
0x00403758	POP3 User	UTF16LE	9	20	.text
0x0040376c	IMAP User	UTF16LE	9	20	.text
0x00403780	SMTP User	UTF16LE	9	20	.text
0x00403794	Email	UTF16LE	5	12	.text
0x004037a0	%s: %s\r\n	UTF16LE	8	18	.text
0x004037b4	POP3 Password	UTF16LE	13	28	.text
0x004037d0	IMAP Password	UTF16LE	13	28	.text
0x004037ec	HTTP Password	UTF16LE	13	28	.text
0x00403808	SMTP Password	UTF16LE	13	28	.text
[illegible]	[illegible]	UTF16LE	9	20	.text
0x00403838	SMTP Port	UTF16LE	9	20	.text
0x0040384c	IMAP Port	UTF16LE	9	20	.text
0x00403860	%s: %d\r\n	UTF16LE	8	18	.text
0x00403874	\r\n\r\n	UTF16LE	4	10	.text
0x00403880	%s\\Outlook.txt	UTF16LE	14	30	.text
0x004038a0	ssfn*	UTF16LE	5	12	.text
0x004038ac	%s\\%s\\%s.vdf	UTF16LE	12	26	.text
0x004038c8	%s\\%s.vdf	UTF16LE	9	20	.text
0x004038dc	\r\n\r\n	ASCII	4	5	.text
0x004038e4	ghijkklmnopqrstuv	UTF16LE	17	36	.text
0x00403908	https://%S/a/%S	UTF16LE	15	32	.text
0x00403928	http://%s	ASCII	9	10	.text
0x00403934	%S x%d\r\n	ASCII	8	9	.text
0x00403944	%s(%S)\r\n	ASCII	8	9	.text
0x00403950	*.rdp	UTF16LE	5	12	.text
0x0040395c	%s | %s | %s	UTF16LE	12	26	.text
0x00403978	DPAPI:	UTF16LE	6	14	.text
0x004039d3	^á`OA	UTF8	5	7	.text
0x00403a65	|ю\r\\OA	UTF8	6	8	.text
0x00403b12	sJ2Ʉ	UTF8	4	6	.text
0x00403c24	Wj@_	ASCII	4	5	.text
0x00403c90	YY3\nL	UTF8	5	7	.text


Detect It Easy 2.03
File name:
C:/Users/malware/Desktop/process_5812/80000.rec.exe
...
Scan
Scripts
Plugins
Log
..
Type:
PE
Size:
80384
Entropy
FLC
S
H
Export
Import
Resource
Overlay
.NET
PE
EntryPoint:
000103cb
>
ImageBase:
00400000
NumberOfSections:
0003
>
SizeOfImage:
00017000
compiler
Microsoft Visual C/C++(2010)[-]
S
?
linker
Microsoft Linker(10.0)[EXE32]
S
?
Options
About
Detect It Easy
Signatures
Info
Scan
100%
>
78 ms
Exit


AND AFTER ALL THAT WE ARE
NOT DONE YET.....
imgflip.com

References

- https://en.wikibooks.org/wiki/X86_Disassembly/Calling_Conventions
- https://github.com/m0n0ph1/Process-Hollowing
- http://blog.sevagas.com/?PE-injection-explained
- https://en.wikipedia.org/wiki/DLL_injection