

Malware analysis with IDA/Radare2 - Multiple unpacking (Ramnit worm)

artik.blue/malware4



Introduction

Today we are following our previous post on these series unpacking a sample (Ramnit) that looks a bit more complex as it performs multiple unpacking, combining own methods with an open source packer. As this piece of malware is relatively old, I'll be using a Windows 7 box with radare2, processhacker, pebear, pestudio and upx to go with it. The endgame here will be to go on unpacking the malware as many times as necessary until we reach to the final one, then get a general idea of what it does.

About Ramnit

Ramnit is a Computer worm affecting Windows users. It was estimated that it infected 800 000 Windows PCs between September and December 2011. The Ramnit botnet was dismantled by Europol and Symantec in 2015. At its peak in 2015, Ramnit was estimated to have infected 3,200,000 PCs. Ramnit infects removable media such as USB drives and also

hides itself within the master boot record. As soon as it infects a computer, it copies itself to all attached and removable drives. It also searches and infects files with the extensions .exe, .dll, .htm and .html.

The sample we'll be working with can be downloaded [here](#) and a similar one can be downloaded from [anyrun](#)

First stage

So considering that we are malware analysts who just received a fresh sample of a potential malware, let's go step by step. Usually the first goal of the analysis will be to determine whether the malware is packed/encrypted or not, then to try to extract some configuration, go for the C2 mechanisms and basic capabilities/actions and finally we may try to perform more advanced reversing to determine its full capabilities and detailed workflow. Having done that the next action(s) to carry would include to extract signatures for yara, automate the unpacking/scan process maybe and to start with threat intelligence.

Today we are focusing mainly on the unpacking part.

So to start with the sample we'll want to get some basic info on the binary. As we already know, knowing about the entropy may help in detecting packing, a high entropy may be a clear indicator. We can easily check that with rahash:

```
PS C:\Users\labo\Desktop> rahash2.exe -a entropy '.\Zeichnungen Muster.bin'  
.\Zeichnungen Muster.bin: 0x00000000-0x001f91f6 entropy: 7.63339648  
PS C:\Users\labo\Desktop>
```

We can also check for the strings, a large number of nonsense strings along with strings related to api-calls may indicate suspicious behavior as well.

```
PS C:\Users\lab\Desktop > rafind2.exe -Z '.\Zeichnungen Muster.bin' | more
0x0000004d !This program cannot be run in DOS mode.\r\r\n$
0x000000f8 RichR
0x00000208 .text
0x0000022f `.rdata
0x00000257 @.data
0x00000280 .rsrc
0x000002a7 @.reloc
0x000002cf B.text
0x0040129d u\bWV
0x004012b5 u/;u\f
0x004012d3 u\fj V
0x00401306 9=thL
0x00401330 =thL
0x00401336 5xhL
0x00401363 =lhL
0x004013b4 =lhL
0x004013f5 5dhL
0x004013fb %hhL
0x00401408 %dhL
0x0040140f \rthL
0x00401491 \bt\\HH
0x004014c5 M\b;S\f|
0x004014f7 t\awV
0x0040156f 9^\f~4
0x0040158b 8\bsS
0x0040159c IU\f;F\f
0x004015ec t +␣i
0x004016a4 u\fj\a
0x00401759 u\fh8
0x0040176d \\SVW
0x0040178a D$(P
0x004017c8 %hhL
```

We can also check for the overall api calls/imports on the program. Calls related to network, file management, the register, virtual memory or process management may also indicate that the program has a suspiciously unusual interest in manipulating our operating system. LoadLibrary and GetprocessHandle may work in combination with hardcoded api-call strings! That is, to dynamically resolve them in memory (not usual on legitimate programs)

```
PS C:\Users\labo\Desktop > rabin2.exe -I -l -s '.\Zeichnungen Muster.bin'
[Symbols]
```

nth	paddr	vaddr	bind	type	size	lib	name
116	0x0008ebc8	0x0048f7c8	NONE	FUNC	0	WSOCK32.dll	imp.WSACleanup
23	0x0008ebcc	0x0048f7cc	NONE	FUNC	0	WSOCK32.dll	imp.socket
12	0x0008ebd0	0x0048f7d0	NONE	FUNC	0	WSOCK32.dll	imp.ioctlsocket
21	0x0008ebd4	0x0048f7d4	NONE	FUNC	0	WSOCK32.dll	imp.setsockopt
15	0x0008ebd8	0x0048f7d8	NONE	FUNC	0	WSOCK32.dll	imp.ntohs
17	0x0008ebdc	0x0048f7dc	NONE	FUNC	0	WSOCK32.dll	imp.recvfrom
10	0x0008ebe0	0x0048f7e0	NONE	FUNC	0	WSOCK32.dll	imp.inet_addr
9	0x0008ebe4	0x0048f7e4	NONE	FUNC	0	WSOCK32.dll	imp.htons
115	0x0008ebe8	0x0048f7e8	NONE	FUNC	0	WSOCK32.dll	imp.WSStartup
151	0x0008ebec	0x0048f7ec	NONE	FUNC	0	WSOCK32.dll	imp.__WSAFDIsSet
18	0x0008ebf0	0x0048f7f0	NONE	FUNC	0	WSOCK32.dll	imp.select
1	0x0008ebf4	0x0048f7f4	NONE	FUNC	0	WSOCK32.dll	imp.accept
13	0x0008ebf8	0x0048f7f8	NONE	FUNC	0	WSOCK32.dll	imp.listen
2	0x0008ebfc	0x0048f7fc	NONE	FUNC	0	WSOCK32.dll	imp.bind
3	0x0008ec00	0x0048f800	NONE	FUNC	0	WSOCK32.dll	imp.closesocket
111	0x0008ec04	0x0048f804	NONE	FUNC	0	WSOCK32.dll	imp.WSAGetLastError
16	0x0008ec08	0x0048f808	NONE	FUNC	0	WSOCK32.dll	imp.recv
20	0x0008ec0c	0x0048f80c	NONE	FUNC	0	WSOCK32.dll	imp.sendto
19	0x0008ec10	0x0048f810	NONE	FUNC	0	WSOCK32.dll	imp.send
11	0x0008ec14	0x0048f814	NONE	FUNC	0	WSOCK32.dll	imp.inet_ntoa
52	0x0008ec18	0x0048f818	NONE	FUNC	0	WSOCK32.dll	imp.gethostbyname
57	0x0008ec1c	0x0048f81c	NONE	FUNC	0	WSOCK32.dll	imp.gethostname
4	0x0008ec20	0x0048f820	NONE	FUNC	0	WSOCK32.dll	imp.connect
108	0x0008e710	0x0048f310	NONE	FUNC	0	KERNEL32.dll	imp.GetWindowsDirectoryW
109	0x0008e714	0x0048f314	NONE	FUNC	0	KERNEL32.dll	imp.GetCurrentProcessId
110	0x0008e718	0x0048f318	NONE	FUNC	0	KERNEL32.dll	imp.GetProcessIoCounters
111	0x0008e71c	0x0048f31c	NONE	FUNC	0	KERNEL32.dll	imp.CreateProcessW
112	0x0008e720	0x0048f320	NONE	FUNC	0	KERNEL32.dll	imp.GetProcessId
113	0x0008e724	0x0048f324	NONE	FUNC	0	KERNEL32.dll	imp.SetPriorityClass
114	0x0008e728	0x0048f328	NONE	FUNC	0	KERNEL32.dll	imp.LoadLibraryW
115	0x0008e72c	0x0048f32c	NONE	FUNC	0	KERNEL32.dll	imp.VirtualAlloc
116	0x0008e730	0x0048f330	NONE	FUNC	0	KERNEL32.dll	imp.IsDebuggerPresent
117	0x0008e734	0x0048f334	NONE	FUNC	0	KERNEL32.dll	imp.GetCurrentDirectoryW
118	0x0008e738	0x0048f338	NONE	FUNC	0	KERNEL32.dll	imp.lstrcpw
119	0x0008e73c	0x0048f33c	NONE	FUNC	0	KERNEL32.dll	imp.DecodePointer
120	0x0008e740	0x0048f340	NONE	FUNC	0	KERNEL32.dll	imp.GetLastError
121	0x0008e744	0x0048f344	NONE	FUNC	0	KERNEL32.dll	imp.RaiseException
122	0x0008e748	0x0048f348	NONE	FUNC	0	KERNEL32.dll	
imp.InitializeCriticalSectionAndSpinCount							
3	0x0008ebc0	0x0048f7c0	NONE	FUNC	0	WINMM.dll	imp.mciSendStringW
1 [...]							
120	0x0008eaa8	0x0048f6a8	NONE	FUNC	0	USER32.dll	imp.SetProcessWindowStation
121	0x0008eaac	0x0048f6ac	NONE	FUNC	0	USER32.dll	imp.GetProcessWindowStation
122	0x0008eab0	0x0048f6b0	NONE	FUNC	0	USER32.dll	imp.OpenWindowStationW
123	0x0008eab4	0x0048f6b4	NONE	FUNC	0	USER32.dll	imp.GetUserObjectSecurity
124	0x0008eab8	0x0048f6b8	NONE	FUNC	0	USER32.dll	imp.MessageBoxW
125	0x0008eabc	0x0048f6bc	NONE	FUNC	0	USER32.dll	imp.DefWindowProcW

126	0x0008eac0	0x0048f6c0	NONE	FUNC	0	USER32.dll	imp.SetClipboardData
127	0x0008eac4	0x0048f6c4	NONE	FUNC	0	USER32.dll	imp.EmptyClipboard
128	0x0008eac8	0x0048f6c8	NONE	FUNC	0	USER32.dll	imp.CountClipboardFormats
129	0x0008eacc	0x0048f6cc	NONE	FUNC	0	USER32.dll	imp.CloseClipboard
130	0x0008ead0	0x0048f6d0	NONE	FUNC	0	USER32.dll	imp.GetClipboardData
131	0x0008ead4	0x0048f6d4	NONE	FUNC	0	USER32.dll	imp.IsClipboardFormatAvailable
132	0x0008ead8	0x0048f6d8	NONE	FUNC	0	USER32.dll	imp.OpenClipboard
133	0x0008eadc	0x0048f6dc	NONE	FUNC	0	USER32.dll	imp.BlockInput
3	0x0008e814	0x0048f414	NONE	FUNC	0	OLEAUT32.dll	imp.SysAllocStringLen
6	0x0008e818	0x0048f418	NONE	FUNC	0	OLEAUT32.dll	imp.SysStringLen
38	0x0008e81c	0x0048f41c	NONE	FUNC	0	OLEAUT32.dll	imp.SafeArrayDestroyData
39	0x0008e820	0x0048f420	NONE	FUNC	0	OLEAUT32.dll	imp.SafeArrayRedim
24	0x0008e824	0x0048f424	NONE	FUNC	0	OLEAUT32.dll	imp.SafeArrayGetElement
23	0x0008e828	0x0048f428	NONE	FUNC	0	OLEAUT32.dll	imp.SafeArrayUnaccessData
37	0x0008e82c	0x0048f42c	NONE	FUNC	0	OLEAUT32.dll	imp.SafeArrayDestroyDescriptor
41	0x0008e830	0x0048f430	NONE	FUNC	0	OLEAUT32.dll	imp.SafeArrayCreateEx
411	0x0008e834	0x0048f434	NONE	FUNC	0	OLEAUT32.dll	imp.SafeArrayCopyData
163	0x0008e838	0x0048f438	NONE	FUNC	0	OLEAUT32.dll	imp.QueryPathOfRegTypeLib
32	0x0008e83c	0x0048f43c	NONE	FUNC	0	OLEAUT32.dll	imp.RegisterActiveObject
146	0x0008e840	0x0048f440	NONE	FUNC	0	OLEAUT32.dll	imp.VariantChangeTypeEx
12	0x0008e844	0x0048f444	NONE	FUNC	0	OLEAUT32.dll	imp.VariantTimeToDosDateTime
7	0x0008e848	0x0048f448	NONE	FUNC	0	OLEAUT32.dll	imp.VariantInit
185	0x0008e84c	0x0048f44c	NONE	FUNC	0	OLEAUT32.dll	imp.UnRegisterTypeLib
220	0x0008e850	0x0048f450	NONE	FUNC	0	OLEAUT32.dll	imp.VarDateFromI1
77	0x0008e854	0x0048f454	NONE	FUNC	0	OLEAUT32.dll	imp.VarR8FromUI1
10	0x0008e858	0x0048f458	NONE	FUNC	0	OLEAUT32.dll	imp.VariantCopyInd
9	0x0008e85c	0x0048f45c	NONE	FUNC	0	OLEAUT32.dll	imp.VariantCopy
418	0x0008e860	0x0048f460	NONE	FUNC	0	OLEAUT32.dll	imp.OleCreatePictureIndirect
164	0x0008e864	0x0048f464	NONE	FUNC	0	OLEAUT32.dll	imp.LHashValOfNameSys
442	0x0008e868	0x0048f468	NONE	FUNC	0	OLEAUT32.dll	imp.UnRegisterTypeLibForUser
443	0x0008e86c	0x0048f46c	NONE	FUNC	0	OLEAUT32.dll	
							imp.OaEnablePerUserTLibRegistration
186	0x0008e870	0x0048f470	NONE	FUNC	0	OLEAUT32.dll	imp.VarDecFix
31	0x0008e874	0x0048f474	NONE	FUNC	0	OLEAUT32.dll	imp.CreateStdDispatch
2	0x0008e878	0x0048f478	NONE	FUNC	0	OLEAUT32.dll	imp.SysReAllocString
8	0x0008e87c	0x0048f47c	NONE	FUNC	0	OLEAUT32.dll	imp.VariantClear
arch	x86						
baddr	0x400000						
binsz	2068983						
bintype	pe						
canary	true						
retguard	false						
class	PE32						
cmp.csum	0x002047e2						
compiled	Sun Apr 19 18:03:00 2020						
crypto	false						
endian	little						
havecode	true						
hdr.csum	0x002047e2						
laddr	0x0						
lang	c						
linenum	false						

```

lsyms      false
machine    i386
nx         false
os         windows
overlay    true
cc         cdecl
pic        true
relocs     false
signed     false
sanitize   false
static     false
stripped   false
subsys     Windows GUI
va         true
[Linked libraries]
wsock32.dll
version.dll
winmm.dll
comctl32.dll
mpr.dll
wininet.dll
psapi.dll
iphlpapi.dll
userenv.dll
uxtheme.dll
kernel32.dll
user32.dll
gdi32.dll
comdlg32.dll
advapi32.dll
shell32.dll
ole32.dll
oleaut32.dll

```

```

18 libraries
PS C:\Users\labo\Desktop >

```

We can also check for the program sections:

```

[0x005e4000]> is
[Sections]

```

nth	paddr	size	vaddr	vsize	perm	name
0	0x00000400	0x8e000	0x00401000	0x8e000	-r-x	.text
1	0x0008e400	0x2fe00	0x0048f000	0x30000	-r--	.rdata
2	0x000be200	0x5200	0x004bf000	0x9000	-rw-	.data
3	0x000c3400	0x113a00	0x004c8000	0x114000	-r--	.rsrc
4	0x001d6e00	0x7200	0x005dc000	0x8000	-r--	.reloc
5	0x001de000	0x1b000	0x005e4000	0x1b000	-rwx	.text_1

```

[0x005e4000]>

```

In this case it's a bit suspicious to have a .text and a .text_1 both with X permissions, one of them RWX... Also it looks like the program won't start from a typical mem addr...

We can go check for the content of the sections, to see if any of them actually holds a bufer (other binary for example, compressed stuff etc):

```

pxw 800 @ 0x005d4000
0x005d4000 0x0000e860 0x8b5d0000 0xa8ed81c5 0x2b2001a6 `.....]..... +
0x005d4010 0x01ae0f85 0x0b858920 0xb02001ae 0x40858600 .... ..@
0x005d4020 0x3c2001b0 0xbc850f01 0x83000001 0x01af3bbd .. <.....;..
0x005d4030 0x33740020 0xaf3fbd83 0x74002001 0x0b858b2a .t3..?.. t*...
0x005d4040 0x2b2001ae 0x01af3b85 0x89008b20 0x01af7885 .. +.;.. ..x..
0x005d4050 0x0b858b20 0x2b2001ae 0x01af3f85 0x89008b20 ..... +.?.. ..
0x005d4060 0x01af7c85 0x8361eb20 0x01af43bd 0x58740020 .|. .a..C.. tX
0x005d4070 0xae0b858b 0x852b2001 0x2001af43 0x858d30ff ..... +.C.. .0..
0x005d4080 0x2001aa94 0xc00bd0ff 0x85893e74 0x2001af47 ... ..t>..G..
0x005d4090 0xaf4b858d 0xff502001 0x01af47b5 0x858d5520 ..K.. P..G.. U..
0x005d40a0 0x2001aad3 0x8589d0ff 0x2001af78 0xaf58858d ... ..x.. ..X.
0x005d40b0 0xff502001 0x01af47b5 0x858d5520 0x2001aad3 . P..G.. U.....
0x005d40c0 0x8589d0ff 0x2001af7c 0xaf7cbd83 0x0f002001 ....|. ..|. ..

```

```

[0x005e4000]> pxw @ 0x00401000
0x00401000 0x4c5144b9 0xa507e800 0xa9680003 0xe80043b7 .DQL.....h..C..
0x00401010 0x00021f6c 0xb7e8c359 0x6800003a 0x0043b7b3 l...Y.....h..C.
0x00401020 0x021f5be8 0xe8c35900 0x000039e4 0x43b7b968 .[...Y...9..h..C
0x00401030 0x1f4ae800 0xc3590002 0x43b7be68 0x1f3ee800 ..J...Y.h..C..>.
0x00401040 0xc3590002 0x002c5fe8 0xb7c36800 0x2de80043 ..Y.._,.h..C..-
0x00401050 0x5900021f 0x15efe8c3 0xc8680000 0xe80043b7 ...Y.....h..C..
0x00401060 0x00021f1c 0xe851c359 0x0000e863 0x43b7cd68 ....Y.Q.c...h..C
0x00401070 0x1f0ae800 0xc3590002 0x4c6310a1 0x408b5100 .....Y...cL.Q.@
0x00401080 0x63100504 0xe850004c 0x0000615f 0x43b7e268 ...cL.P._a..h..C
0x00401090 0x1eeae800 0xc3590002 0x003196e8 0xb7e76800 .....Y...1..h..
0x004010a0 0xd9e80043 0x5900021e 0x7546e8c3 0xec680000 C.....Y..Fu..h.
0x004010b0 0xe80043b7 0x00021ec8 0xc7e8c359 0x6800014d .C.....Y...M..h
0x004010c0 0x0043b7f1 0x021eb7e8 0xb9c35900 0x004c7404 ..C.....Y...tL.
0x004010d0 0x0066f2e8 0xb8006800 0xa1e80043 0x5900021e ..f..h..C.....Y
0x004010e0 0xc0000000 0xc0000000 0xc0000000 0xc0000000 .....
0x004010f0 0x8b575653 0x5b0d6af1 0x016cbe8d 0xef830000 SVW..j.[..l.....

```

But don't expect to find a lot by just pxw'ing whats in there...

And we can also go check the entropy for each sections, so in here we are already getting some hints:

```
[0x005e4000]> iS entropy
[Sections]
```

nth	paddr	size	vaddr	vsize	perm	entropy	name
0	0x00000400	0x8e000	0x00401000	0x8e000	-r-x	6.67524835	.text
1	0x0008e400	0x2fe00	0x0048f000	0x30000	-r--	5.76322253	.rdata
2	0x000be200	0x52000	0x004bf000	0x90000	-rw-	1.19638192	.data
3	0x000c3400	0x113a00	0x004c8000	0x114000	-r--	7.98164843	.rsrc
4	0x001d6e00	0x72000	0x005dc000	0x80000	-r--	6.78395556	.reloc
5	0x001de000	0x1b000	0x005e4000	0x1b000	-rwx	7.87108038	.text_1

And we can be even more precise...

```
PS C:\Users\labo\Desktop> radare2.exe -qfnc 'p=e 30'
"C:\Users\labo\Desktop\Zeichnungen Muster.bin.exe"
0x00000000 000 00d1 |#####
0x00010d66 001 00c3 |####
0x00021acc 002 00d5 |#
0x00032832 003 00d7 |#####
0x00043598 004 00d1 |#
0x000542fe 005 00d3 |#####
0x00065064 006 00cf |#
0x00075dca 007 00d0 |#####
0x00086b30 008 00c6 |#####
0x00097896 009 00bd |#####
0x000a85fc 00a 009f |#####
0x000b9362 00b 0086 |#####
0x000ca0c8 00c 00fa |#
0x000dae2e 00d 00fe |#
0x000ebb94 00e 00fe |#
0x000fc8fa 00f 00fe |#
0x0010d660 010 00fe |#
0x0011e3c6 011 00fe |#
0x0012f12c 012 00fe |#
0x0013fe92 013 00fe |#
0x00150bf8 014 00fe |#
0x0016195e 015 00fe |#
0x001726c4 016 00fe |#
0x0018342a 017 00fe |#
0x00194190 018 00fe |#
0x001a4ef6 019 00fe |#
0x001b5c5c 01a 00fe |#
0x001c69c2 01b 00fe |#####
0x001d7728 01c 00f6 |#
0x001e848e 01d 00f6 |#####
PS C:\Users\labo\Desktop>
```

Having done this basic analysis I would be confident that the binary holds something packed into it. So the next step would involve opening it in radare2 to do the debugging:


```

radare2 -AAA
[0x005e4000]> ood
Spawned new process with pid 2904, tid = 2104
= attach 2904 2104
File dbg://C:\\Users\\labo\\Desktop\\Zeichnungen\\Muster.bin.exe reopened in read-
write mode
2904
[0x777b3820]> pd 10
;-- rip:
0x777b3820      48          dec eax
0x777b3821      83ec48      sub esp, 0x48
0x777b3824      4c          dec esp
0x777b3825      8bc9        mov ecx, ecx
0x777b3827      48          dec eax
0x777b3828      8b055af40d00 mov eax, dword [0xdf45a] ;
[0xdf45a:4]=-1
0x777b382e      48          dec eax
0x777b382f      85c0        test eax, eax
,=< 0x777b3831  740c        je 0x777b383f
| 0x777b3833    4c          dec esp
[0x777b3820]> dcu entry0
Continue until 0x00ad4000 using 1 bpsize
(2904) loading library at 0x0000000077760000 (C:\\Windows\\System32\\ntdll.dll)
ntdll.dll
(2904) loading library at 0x0000000077920000 (C:\\Windows\\SysWOW64\\ntdll.dll)
ntdll.dll
(2904) loading library at 0x0000000074410000 (C:\\Windows\\System32\\wow64.dll)
wow64.dll
(2904) loading library at 0x00000000743B0000 (C:\\Windows\\System32\\wow64win.dll)
wow64win.dll
(2904) loading library at 0x00000000743A0000 (C:\\Windows\\System32\\wow64cpu.dll)
wow64cpu.dll
[0x77806fb1]> dcu entry0
Continue until 0x00ad4000 using 1 bpsize
(2904) loading library at 0x0000000077540000 (C:\\Windows\\System32\\kernel32.dll)
kernel32.dll
(2904) unloading library at 0x0000000077540000 (C:\\Windows\\System32\\kernel32.dll)
kernel32.dll
(2904) loading library at 0x00000000765F0000 (C:\\Windows\\SysWOW64\\kernel32.dll)
kernel32.dll
(2904) unloading library at 0x00000000765F0000 (C:\\Windows\\SysWOW64\\kernel32.dll)
kernel32.dll
(2904) loading library at 0x0000000077540000 (C:\\Windows\\System32\\kernel32.dll)
kernel32.dll
(2904) unloading library at 0x0000000077540000 (C:\\Windows\\System32\\kernel32.dll)
kernel32.dll
(2904) loading library at 0x0000000077660000 (C:\\Windows\\System32\\user32.dll)
user32.dll
(2904) unloading library at 0x0000000077660000 (C:\\Windows\\System32\\user32.dll)
user32.dll
(2904) loading library at 0x00000000765F0000 (C:\\Windows\\SysWOW64\\kernel32.dll)
kernel32.dll

```

(2904) loading library at 0x0000000076800000 (C:\Windows\SysWOW64\KernelBase.dll)
KernelBase.dll
(2904) loading library at 0x0000000074390000 (C:\Windows\SysWOW64\wsock32.dll)
wsock32.dll
(2904) loading library at 0x0000000076880000 (C:\Windows\SysWOW64\ws2_32.dll)
ws2_32.dll
(2904) loading library at 0x0000000077480000 (C:\Windows\SysWOW64\msvcrt.dll)
msvcrt.dll
(2904) loading library at 0x0000000075E60000 (C:\Windows\SysWOW64\rpcrt4.dll)
rpcrt4.dll
(2904) loading library at 0x0000000075090000 (C:\Windows\SysWOW64\sspicli.dll)
sspicli.dll
(2904) loading library at 0x0000000075080000 (C:\Windows\SysWOW64\cryptbase.dll)
cryptbase.dll
(2904) loading library at 0x00000000761C0000 (C:\Windows\SysWOW64\sechost.dll)
sechost.dll
(2904) loading library at 0x0000000077530000 (C:\Windows\SysWOW64\nsi.dll) nsi.dll
(2904) loading library at 0x0000000074380000 (C:\Windows\SysWOW64\version.dll)
version.dll
(2904) loading library at 0x0000000074340000 (C:\Windows\SysWOW64\winmm.dll)
winmm.dll
(2904) loading library at 0x0000000077370000 (C:\Windows\SysWOW64\user32.dll)
user32.dll
(2904) loading library at 0x00000000761F0000 (C:\Windows\SysWOW64\gdi32.dll)
gdi32.dll
(2904) loading library at 0x0000000076410000 (C:\Windows\SysWOW64\lpk.dll) lpk.dll
(2904) loading library at 0x0000000075F60000 (C:\Windows\SysWOW64\usp10.dll)
usp10.dll
(2904) loading library at 0x0000000076700000 (C:\Windows\SysWOW64\advapi32.dll)
advapi32.dll
(2904) loading library at 0x000000006E780000
(C:\Windows\winsxs\x86_microsoft.windows.common-ctrl132.dll) comctl32.d
(2904) loading library at 0x00000000750F0000 (C:\Windows\SysWOW64\shlwapi.dll)
shlwapi.dll
(2904) loading library at 0x0000000074320000 (C:\Windows\SysWOW64\mpr.dll) mpr.dll
(2904) loading library at 0x0000000076A70000 (C:\Windows\SysWOW64\wininet.dll)
wininet.dll
(2904) loading library at 0x00000000768C0000 (C:\Windows\SysWOW64\api-ms-win-
downlevel-user32-l1-1-0.dll) api-ms-win-downlevel-user32-l1-1-0
(2904) loading library at 0x00000000764B0000 (C:\Windows\SysWOW64\api-ms-win-
downlevel-shlwapi-l1-1--0.dll) api-ms-win-downlevel-shlwapi-l1-1-1
(2904) loading library at 0x0000000075E50000 (C:\Windows\SysWOW64\api-ms-win-
downlevel-version-l1-1--0.dll) api-ms-win-downlevel-version-l1-1-1
(2904) loading library at 0x0000000077360000 (C:\Windows\SysWOW64\api-ms-win-
downlevel-normaliz-l1-1-1-0.dll) api-ms-win-downlevel-normaliz-l1-1-1-0
(2904) loading library at 0x00000000761B0000 (C:\Windows\SysWOW64\normaliz.dll)
normaliz.dll
(2904) loading library at 0x0000000076F30000 (C:\Windows\SysWOW64\iertutil.dll)
iertutil.dll
(2904) loading library at 0x0000000075F50000 (C:\Windows\SysWOW64\api-ms-win-
downlevel-advapi32-l1-1-1-0.dll) api-ms-win-downlevel-advapi32-l1-1-1-0
(2904) loading library at 0x0000000075150000 (C:\Windows\SysWOW64\userenv.dll)

```
userenv.dll
(2904) loading library at 0x0000000077470000 (C:\Windows\SysWOW64\profapi.dll)
profapi.dll
(2904) loading library at 0x00000000761E0000 (C:\Windows\SysWOW64\psapi.dll)
psapi.dll
(2904) loading library at 0x0000000074300000 (C:\Windows\SysWOW64\IPHLPAPI.DLL)
IPHLPAPI.DLL
(2904) loading library at 0x00000000742F0000 (C:\Windows\SysWOW64\winnsi.dll)
winnsi.dll
(2904) loading library at 0x0000000074210000 (C:\Windows\SysWOW64\uxtheme.dll)
uxtheme.dll
(2904) loading library at 0x0000000076280000 (C:\Windows\SysWOW64\comdlg32.dll)
comdlg32.dll
(2904) loading library at 0x0000000075200000 (C:\Windows\SysWOW64\shell32.dll)
shell32.dll
(2904) loading library at 0x0000000076020000 (C:\Windows\SysWOW64\ole32.dll)
ole32.dll
(2904) loading library at 0x0000000076310000 (C:\Windows\SysWOW64\oleaut32.dll)
oleaut32.dll
[0x779c0fc5]> dcu entry0
Continue until 0xad4000 using 1 bpsize
(2904) loading library at 0x0000000077300000 (C:\Windows\SysWOW64\imm32.dll)
imm32.dll
(2904) loading library at 0x0000000076E60000 (C:\Windows\SysWOW64\msctf.dll)
msctf.dll
hit breakpoint at: 0xad4000
[0xad4000]>
```

So we run the program until we reach the entry point (entry0) to allow it to load the required libraries.

After that, we can try to pd/pdf and see if we can easily grasp something. But in many cases such as that its either we have a lot of time and/or a huge interesting in knowing everything or we'd just set breakpoints on relevant api calls and try to detect some behaviour.

So the program starts with what looks like some decoding process:

```

[0x00ad4000]> pd 20
      ;-- section..text_1:
      ;-- map.IMAGE____.__x:
      ;-- rdx:
      ;-- rip:
/ 506: entry0 (int32_t arg_1ch);
|           ; var int32_t var_1ch_2 @ rsp+0x1c
|           ; var int32_t var_1ch @ rsp+0x60
|           ; arg int32_t arg_1ch @ rsp+0x70
|           0x00ad4000      60           pushal                    ; [05] -rwx
section size 110592 named .text_1
|           0x00ad4001      e800000000      call 0xad4006
|           ; CALL XREF from entry0 @ 0xad4001
|           0x00ad4006      5d           pop ebp
|           0x00ad4007      8bc5        mov eax, ebp
|           0x00ad4009      81eda8a60120 sub ebp, 0x2001a6a8
|           0x00ad400f      2b850fae0120 sub eax, dword [ebp + 0x2001ae0f]
|           0x00ad4015      89850bae0120 mov dword [ebp + 0x2001ae0b], eax
|           0x00ad401b      b000        mov al, 0
|           0x00ad401d      868540b00120 xchg byte [ebp + 0x2001b040], al
|           0x00ad4023      3c01        cmp al, 1                ; 1
|           ,=< 0x00ad4025      0f85bc010000 jne 0xad41e7
|           | 0x00ad402b      83bd3baf0120. cmp dword [ebp + 0x2001af3b], 0
|           ,==< 0x00ad4032      7433        je 0xad4067
|           || 0x00ad4034      83bd3faf0120. cmp dword [ebp + 0x2001af3f], 0
|           ,===< 0x00ad403b      742a        je 0xad4067
|           ||| 0x00ad403d      8b850bae0120 mov eax, dword [ebp + 0x2001ae0b]
|           ||| 0x00ad4043      2b853baf0120 sub eax, dword [ebp + 0x2001af3b]
|           ||| 0x00ad4049      8b00        mov eax, dword [eax]
|           ||| 0x00ad404b      898578af0120 mov dword [ebp + 0x2001af78], eax
|           ||| 0x00ad4051      8b850bae0120 mov eax, dword [ebp + 0x2001ae0b]
[0x00ad4000]>

```

Ok so the packed hypothesis gains traction. The next thing from this point on is to go for the breakpoint in api calls strategy. In order to do that, let's go check the libraries we have:

```

[0x005d4000]> dmi
[0x00ad4000]> dmi
0x008f0000 0x00aef000 C:\Users\labo\Desktop\Zeichnungen Muster.bin.exe
0x77760000 0x778ff000 C:\Windows\SYSTEM32\ntdll.dll
0x74410000 0x7444f000 C:\Windows\SYSTEM32\wow64.dll
0x743b0000 0x7440c000 C:\Windows\SYSTEM32\wow64win.dll
0x743a0000 0x743a8000 C:\Windows\SYSTEM32\wow64cpu.dll
0x008f0000 0x00aef000 C:\Users\labo\Desktop\Zeichnungen Muster.bin.exe
0x77920000 0x77aa0000 C:\Windows\SysWOW64\ntdll.dll
0x765f0000 0x76700000 C:\Windows\syswow64\kernel32.dll
0x76800000 0x76847000 C:\Windows\syswow64\KERNELBASE.dll
0x74390000 0x74397000 C:\Windows\SysWOW64\WSOCK32.dll
0x76880000 0x768b5000 C:\Windows\syswow64\WS2_32.dll
0x77480000 0x7752c000 C:\Windows\syswow64\msvcrt.dll
0x75e60000 0x75f50000 C:\Windows\syswow64\RPCRT4.dll
0x75090000 0x750f0000 C:\Windows\syswow64\SspiCli.dll
0x75080000 0x7508c000 C:\Windows\syswow64\CRYPTBASE.dll
0x761c0000 0x761d9000 C:\Windows\SysWOW64\sechost.dll
0x77530000 0x77536000 C:\Windows\syswow64\NSI.dll
0x74380000 0x74389000 C:\Windows\SysWOW64\VERSION.dll
0x74340000 0x74372000 C:\Windows\SysWOW64\WINMM.dll
0x77370000 0x77470000 C:\Windows\syswow64\USER32.dll
0x761f0000 0x76280000 C:\Windows\syswow64\GDI32.dll
0x76410000 0x7641a000 C:\Windows\syswow64\LPK.dll
0x75f60000 0x75ffd000 C:\Windows\syswow64\USP10.dll
0x76700000 0x767a1000 C:\Windows\syswow64\ADVAPI32.dll
0x6e780000 0x6e91e000 C:\Windows\WinSxS\x86_microsoft.windows.common-
controls_6595b64144ccf1df_6.0.2.dllbd5705d\COMCTL
0x750f0000 0x75147000 C:\Windows\syswow64\SHLWAPI.dll
0x74320000 0x74332000 C:\Windows\SysWOW64\MPR.dll
0x76a70000 0x76ae51000 C:\Windows\syswow64\WININET.dll
0x768c0000 0x768c4000 C:\Windows\syswow64\api-ms-win-downlevel-user32-l1-1-0.dll
0x764b0000 0x764b4000 C:\Windows\syswow64\api-ms-win-downlevel-shlwapi-l1-1-0.dll
0x75e50000 0x75e54000 C:\Windows\syswow64\api-ms-win-downlevel-version-l1-1-0.dll
0x77360000 0x77363000 C:\Windows\syswow64\api-ms-win-downlevel-normaliz-l1-1-0.dll
0x761b0000 0x761b3000 C:\Windows\syswow64\normaliz.DLL
0x76f30000 0x77166000 C:\Windows\syswow64\iertutil.dll
0x75f50000 0x75f55000 C:\Windows\syswow64\api-ms-win-downlevel-advapi32-l1-1-0.dll
0x75150000 0x75167000 C:\Windows\syswow64\USERENV.dll
0x77470000 0x7747b000 C:\Windows\syswow64\profapi.dll
0x761e0000 0x761e5000 C:\Windows\syswow64\PSAPI.DLL
0x74300000 0x7431c000 C:\Windows\SysWOW64\IPHLPAPI.DLL
0x742f0000 0x742f7000 C:\Windows\SysWOW64\WINNSI.DLL
0x74210000 0x74290000 C:\Windows\SysWOW64\UxTheme.dll
0x76280000 0x762fb000 C:\Windows\syswow64\COMDLG32.dll
0x75200000 0x75e4c000 C:\Windows\syswow64\SHELL32.dll
0x76020000 0x7617d000 C:\Windows\syswow64\ole32.dll
0x76310000 0x763a1000 C:\Windows\syswow64\OLEAUT32.dll
0x77300000 0x77360000 C:\Windows\SysWOW64\IMM32.DLL
0x76e60000 0x76f2d000 C:\Windows\syswow64\MSCTF.dll
[0x00ad4000]>

```

And then identify some interesting file / memory management / process injection related api calls in kernel32 and set those breakpoints there:

```
[0x00ad4000]> dmi kernel32 VirtualAlloc  
[Symbols]
```

nth	paddr	vaddr	bind	type	size	lib	name
1264	0x00011826	0x76601826	GLOBAL	FUNC	0	KERNEL32.dll	VirtualAlloc
4	0x00010908	0x76600908	NONE	FUNC	0	API-MS-Win-Core-Memory-L1-1-0.dll	imp.VirtualAlloc

```
[0x00ad4000]> dmi kernel32 VirtualProtect  
[Symbols]
```

nth	paddr	vaddr	bind	type	size	lib	name
1270	0x000143be	0x766043be	GLOBAL	FUNC	0	KERNEL32.dll	VirtualProtect
8	0x00010918	0x76600918	NONE	FUNC	0	API-MS-Win-Core-Memory-L1-1-0.dll	imp.VirtualProtect

```
[0x00ad4000]> dmi kernel32 WriteFile  
[Symbols]
```

nth	paddr	vaddr	bind	type	size	lib	name
1324	0x00011282	0x76601282	GLOBAL	FUNC	0	KERNEL32.dll	WriteFile
9	0x000109e4	0x766009e4	NONE	FUNC	0	API-MS-Win-Core-File-L1-1-0.dll	imp.WriteFile

```
[0x00ad4000]> dmi kernel32 CreateProcessInternalW  
[Symbols]
```

nth	paddr	vaddr	bind	type	size	lib	name
170	0x00023c23	0x76613c23	GLOBAL	FUNC	0	KERNEL32.dll	CreateProcessInternalW

```
[0x00ad4000]> dmi kernel32 CreateProcessInternalA  
[Symbols]
```

nth	paddr	vaddr	bind	type	size	lib	name
169	0x0002a507	0x7661a507	GLOBAL	FUNC	0	KERNEL32.dll	CreateProcessInternalA

```
[  
[0x00ad4000]>  
[0x00ad4000]> dmi kernel32 IsDebuggerPresent  
[Symbols]
```

nth	paddr	vaddr	bind	type	size	lib	name
770	0x0001494d	0x7660494d	GLOBAL	FUNC	0	KERNEL32.dll	IsDebuggerPresent
4	0x00010d94	0x76600d94	NONE	FUNC	0	API-MS-Win-Core-Debug-L1-1-0.dll	imp.IsDebuggerPresent

Basically we'll want to see if the program writes any content on disk and/or on memory and if it does we'll also want to know if the malware tries to run that in some way.

NtResumeThread/NtResumeProcess are useful for that same reason as well:

```
0x00ad4000]> dmi ntdll NtResumeThread
[Symbols]
```

```
nth paddr      vaddr      bind  type size lib      name
-----
480 0x000691c0 0x777c9dc0 GLOBAL FUNC 0      ntdll.dll NtResumeThread
```

Here's a list of some breakpoints that I would set, based on the functions listed:

```
[0x00ad4000]> db 0x7660494d
[0x00ad4000]> db
0x76601826 - 0x76601827 1 --x sw break enabled valid cmd="" cond="" name="0x76601826"
module=""
0x766043be - 0x766043bf 1 --x sw break enabled valid cmd="" cond="" name="0x766043be"
module=""
0x76601282 - 0x76601283 1 --x sw break enabled valid cmd="" cond="" name="0x76601282"
module=""
0x76613c23 - 0x76613c24 1 --x sw break enabled valid cmd="" cond="" name="0x76613c23"
module=""
0x7661a507 - 0x7661a508 1 --x sw break enabled valid cmd="" cond="" name="0x7661a507"
module=""
0x777c9dc0 - 0x777c9dc1 1 --x sw break enabled valid cmd="" cond="" name="0x777c9dc0"
module=""
0x7660494d - 0x7660494e 1 --x sw break enabled valid cmd="" cond="" name="0x7660494d"
module=""
[0x00ad4000]>
```

Having set those, let's do dc and see if we hit any:

We hit the first breakpoint, and it corresponds to WriteFile. Let's inspect the stack:

```
[0x762c35b0]> pxr @rsp
0x015ff9bc 0x005d46ed .F]. @ rsp IMAGE .text_1 R W X 'mov ebx, dword [ebp + 8]'
'IMAGE '
0x015ff9c0 0x00000210 .... 528 rdx
0x015ff9c4 0x005d499f .I]. IMAGE .text_1 R W X 'dec ebp' 'IMAGE '
0x015ff9c8 0x0001a600 ....
0x015ff9cc 0x015ff9d8 .._. PRIVATE rax R W 0xca373ff5
0x015ff9d0 ..[ null bytes ].. 00000000
0x015ff9d4 0x00000210 .... 528 rdx
0x015ff9d8 0xca373ff5 .??. @ rax
0x015ff9dc 0xe05b995e ^.[. @ rbp rbx
0x015ff9e0 0x005d41c6 .A]. IMAGE .text_1 entry0 R W X 'cmp eax, 1' 'IMAGE '
0x015ff9e4 0xe05b995e ^.[. rbx
0x015ff9ec 0x005d499f .I]. IMAGE .text_1 R W X 'dec ebp' 'IMAGE '
```

Ok si it's defefinetely writing something into the stack, what it is?

```
[0x76601282]> pxw 400 @ 0x00ad499f
0x00ad499f 0x00905a4d 0x00000003 0x00000004 0x0000ffff MZ.....
0x00ad49af 0x000000b8 0x00000000 0x00000040 0x00000000 .....@.....
0x00ad49bf 0x00000000 0x00000000 0x00000000 0x00000000 .....
0x00ad49cf 0x00000000 0x00000000 0x00000000 0x00000088 .....
0x00ad49df 0xc5f5da09 0x25ea24c8 0x7e5c05f0 0xfa1056b5 .....$.%..\~.V..
0x00ad49ef 0x8427934a 0x1bc0eb0a 0xcb8422dc 0xb1054f0c J.'....."....0..
0x00ad49ff 0x04feeb0e 0x2b7bbfa2 0x7338965b 0xbe932367 .....{+[.8sg#..
0x00ad4a0f 0x6a7d9a28 0x17c1b915 0x10bcf4d0 0x0ce64e2f (.}j...../N..
0x00ad4a1f 0x00000000 0x00000000 0x00004550 0x0003014c .....PE..L...
0x00ad4a2f 0x356b0f6b 0x00000000 0x00000000 0x010f00e0 k.k5.....
0x00ad4a3f 0x0407010b 0x00013000 0x00008000 0x0003a000 .....0.....
0x00ad4a4f 0x0004d240 0x0003b000 0x0004e000 0x00400000 @.....@.
0x00ad4a5f 0x00001000 0x00000200 0x00000005 0x00020007 .....
0x00ad4a6f 0x00000004 0x00000000 0x0005d000 0x00001000 .....
0x00ad4a7f 0x00000000 0x00000002 0x00100000 0x00001000 .....
0x00ad4a8f 0x00100000 0x00001000 0x00000000 0x00000010 .....
0x00ad4a9f 0x00000000 0x00000000 0x00055874 0x000000e4 .....tX.....
0x00ad4aaf 0x0004e000 0x0000f000 0x00000000 0x00000000 .....
0x00ad4abf 0x00000000 0x00000000 0x00000000 0x00000000 .....
0x00ad4acf 0x00000000 0x00000000 0x00000000 0x00000000 .....
0x00ad4adf 0x00000000 0x00000000 0x00000000 0x00000000 .....
0x00ad4aef 0x00000000 0x00000000 0x00000000 0x00000000 .....
0x00ad4aff 0x00000000 0x00000000 0x00000000 0x00000000 .....
0x00ad4b0f 0x00000000 0x00000000 0x00000000 0x00000000 .....
0x00ad4b1f 0x30585055 0x00000000 0x0003a000 0x00001000 UPX0.....
```

Nice it is writting something that looks like a UPX packed binary(so vintage)! Let's move on, let's see if it loads it/runs it at some point:

And we hit dc and... we get to a CreateProcess* call!


```

hit breakpoint at: 0x76613c23
[0x76613c23]> pxr @ rsp
0x0064f68c 0x7661a62a *.av @ rsp IMAGE kernel32.dll | .text R X | .text'rd [ebp -
0x1c], eax' 'IMAGE kernel32.dll
0x0064f690 ..[ null bytes ].. 00000000
0x0064f698 0x006e77d8 .wn. PRIVATE R W 0x3a0043
0x0064f69c ..[ null bytes ].. 00000000
0x0064f6b4 0x0064f6d0 ..d. PRIVATE rax,rcx R W 0x48
0x0064f6b8 0x0064f7d0 ..d. PRIVATE R W 0x0
0x0064f6bc ..[ null bytes ].. 00000000
0x0064f6c0 0x55a1e22d -..U
0x0064f6c4 0x0064f7e0 ..d. PRIVATE R W 0x0
0x0064f6c8 0x014d4795 .GM. IMAGE Zeichnungen Muster.bin.exe | .text .text_1 R W
Xster.bin.exe | .text' r.binmgr.exe

```

```

[0x76613c23]> pxw @ 0x006e77d8
0x006e77d8 0x003a0043 0x0055005c 0x00650073 0x00730072 C:.\.U.s.e.r.s.
0x006e77e8 0x006c005c 0x00620061 0x005c006f 0x00650044 \.l.a.b.o.\.D.e.
0x006e77f8 0x006b0073 0x006f0074 0x005c0070 0x0065005a s.k.t.o.p.\.Z.e.
0x006e7808 0x00630069 0x006e0068 0x006e0075 0x00650067 i.c.h.n.u.n.g.e.
0x006e7818 0x0020006e 0x0075004d 0x00740073 0x00720065 n. .M.u.s.t.e.r.
0x006e7828 0x0062002e 0x006e0069 0x0067006d 0x002e0072 ..b.i.n.m.g.r...
0x006e7838 0x00780065 0x00000065 0xabababab 0xabababab e.x.e.....
0x006e7848 0x00000000 0x00000000 0x71fd6008 0x0000de77 .....`.qw...
0x006e7858 0x006e9820 0x006e00c4 0x77fe600d 0x1800de65 .n...n...`.we...
0x006e7868 0x00000000 0x00c70138 0x006e7ee0 0x006e35c8 ....8....~n..5n.
0x006e7878 0x00000000 0x00000000 0x00000000 0xbaad0000 .....
0x006e7888 0xabababab 0xabababab 0x00000000 0x00000000 .....
0x006e7898 0x34fe604e 0x1800de60 0xffffde210 0xffffde210 N`.4`.....
0x006e78a8 0x00000000 0x00c707e0 0x00000000 0x00000000 .....

```

And its creating a process on our newly created executable 8)

So at this point, if we proceed hitting dc we'll see a call to NtResumeProcess (as the created process will spawn in a suspended state), so the newly created binary will start its executing while the original one will go one, getting inside a loop checking for processes starting the second program if it stops somehow (more or less).

As we are interested in the unpacking part, at this point we can either attach to the newly created process with (radare2 -d process id) or just open the generated binary for analysis/debug. I'll go for the second option.

Unpacking the second binary



So as we detected, the second binary is UPX packed. We can either unpack it manually by checking for VirtualAlloc and stuff like that or we can just simply do `upx -d` (upx is included in the FLARE-VM or can be downloaded from the [site](#))

```
PS C:\Users\labo\Desktop> upx -d '.\Zeichnungen Muster.binmgr.exe'
                    Ultimate Packer for eXecutables
                    Copyright (C) 1996 - 2020
UPX 3.96w          Markus Oberhumer, Laszlo Molnar & John Reiser   Jan 23rd 2020

      File size      Ratio      Format      Name
-----
159232 <- 108032 67.85%  win32/pe  Zeichnungen Muster.binmgr.exe
```

```
Unpacked 1 file.
PS C:\Users\labo\Desktop>
```

After we unpack the program, its entropy descends to something that looks more normal:

```
PS C:\Users\labo\Desktop> rahash2.exe -a entropy '.\Zeichnungen Muster.binmgr.exe'
.\Zeichnungen Muster.binmgr.exe: 0x00000000-0x00026dff entropy: 4.46272451
PS C:\Users\labo\Desktop>
```

Also, we are not seeing weird stuff on its sections:

```
[0x0040101a]> iS
[Sections]

nth paddr          size vaddr          vsize perm name
-----
0  0x00000400    0x1200 0x00401000    0x2000 -r-x .text
1  0x00001600    0x1cc00 0x00403000    0x3f000 -rw- .data
2  0x0001e200     0x200 0x00442000    0x1000 -r-- .rdata
3  0x0001e400    0x8a00 0x00443000    0x9000 -r-- .rsrc
```

But still, let's go on and debug:

```
[0x0040101a]> ood
Spawned new process with pid 2316, tid = 1172
= attach 2316 1172
File dbg://C:\\Users\\labo\\Desktop\\Zeichnungen\\Muster.binmgr.exe reopened in
read-write mode
2316
[0x777b3820]> dcu entry0
Continue until 0x0040101a using 1 bpsize
(2316) loading library at 0x0000000077760000 (C:\\Windows\\System32\\ntdll.dll)
ntdll.dll
(2316) loading library at 0x0000000077920000 (C:\\Windows\\SysWOW64\\ntdll.dll)
ntdll.dll
(2316) loading library at 0x00000000743E0000 (C:\\Windows\\System32\\wow64.dll)
wow64.dll
(2316) loading library at 0x0000000074380000 (C:\\Windows\\System32\\wow64win.dll)
wow64win.dll
(2316) loading library at 0x0000000074450000 (C:\\Windows\\System32\\wow64cpu.dll)
wow64cpu.dll
[0x77806fb1]> dcu entry0
Continue until 0x0040101a using 1 bpsize
(2316) loading library at 0x0000000077540000 (C:\\Windows\\System32\\kernel32.dll)
kernel32.dll
(2316) unloading library at 0x0000000077540000 (C:\\Windows\\System32\\kernel32.dll)
kernel32.dll
(2316) loading library at 0x00000000765F0000 (C:\\Windows\\SysWOW64\\kernel32.dll)
kernel32.dll
(2316) unloading library at 0x00000000765F0000 (C:\\Windows\\SysWOW64\\kernel32.dll)
kernel32.dll
(2316) loading library at 0x0000000077540000 (C:\\Windows\\System32\\kernel32.dll)
kernel32.dll
(2316) unloading library at 0x0000000077540000 (C:\\Windows\\System32\\kernel32.dll)
kernel32.dll
(2316) loading library at 0x0000000077660000 (C:\\Windows\\System32\\user32.dll)
user32.dll
(2316) unloading library at 0x0000000077660000 (C:\\Windows\\System32\\user32.dll)
user32.dll
(2316) loading library at 0x00000000765F0000 (C:\\Windows\\SysWOW64\\kernel32.dll)
kernel32.dll
(2316) loading library at 0x0000000076800000 (C:\\Windows\\SysWOW64\\KernelBase.dll)
KernelBase.dll
(2316) loading library at 0x0000000077370000 (C:\\Windows\\SysWOW64\\user32.dll)
user32.dll
(2316) loading library at 0x00000000761F0000 (C:\\Windows\\SysWOW64\\gdi32.dll)
gdi32.dll
(2316) loading library at 0x0000000076410000 (C:\\Windows\\SysWOW64\\lpk.dll) lpk.dll
(2316) loading library at 0x0000000075F60000 (C:\\Windows\\SysWOW64\\usp10.dll)
usp10.dll
(2316) loading library at 0x0000000077480000 (C:\\Windows\\SysWOW64\\msvcrt.dll)
msvcrt.dll
(2316) loading library at 0x0000000076700000 (C:\\Windows\\SysWOW64\\advapi32.dll)
advapi32.dll
(2316) loading library at 0x00000000761C0000 (C:\\Windows\\SysWOW64\\sechost.dll)
```

```
sechost.dll
(2316) loading library at 0x0000000075E60000 (C:\Windows\SysWOW64\rpcrt4.dll)
rpcrt4.dll
(2316) loading library at 0x0000000075090000 (C:\Windows\SysWOW64\sspicli.dll)
sspicli.dll
(2316) loading library at 0x0000000075080000 (C:\Windows\SysWOW64\cryptbase.dll)
cryptbase.dll
[0x779c0fc5]> dcu entry0
Continue until 0x0040101a using 1 bpsize
(2316) loading library at 0x0000000077300000 (C:\Windows\SysWOW64\imm32.dll)
imm32.dll
(2316) loading library at 0x0000000076E60000 (C:\Windows\SysWOW64\msctf.dll)
msctf.dll
hit breakpoint at: 0x40101a
[0x0040101a]>
```

In here we follow the same process, I'll look for memory management and potential process injection / infection techniques:

hit breakpoint at: 0x40101a

[0x0040101a]> dmi kernel32 VirtualAlloc

[Symbols]

nth	paddr	vaddr	bind	type	size	lib	name
-----	-------	-------	------	------	------	-----	------

1264	0x00011826	0x76601826	GLOBAL	FUNC	0	KERNEL32.dll	
------	------------	------------	--------	------	---	--------------	--

VirtualAlloc

4	0x00010908	0x76600908	NONE	FUNC	0	API-MS-Win-Core-Memory-L1-1-0.dll	
---	------------	------------	------	------	---	-----------------------------------	--

imp.VirtualAlloc

[0x0040101a]> dmi kernel32 VirtualProtect

[Symbols]

nth	paddr	vaddr	bind	type	size	lib	name
-----	-------	-------	------	------	------	-----	------

1270	0x000143be	0x766043be	GLOBAL	FUNC	0	KERNEL32.dll	
------	------------	------------	--------	------	---	--------------	--

VirtualProtect

8	0x00010918	0x76600918	NONE	FUNC	0	API-MS-Win-Core-Memory-L1-1-0.dll	
---	------------	------------	------	------	---	-----------------------------------	--

imp.VirtualProtect

[0x0040101a]> dmi kernel32 WriteFile

[Symbols]

nth	paddr	vaddr	bind	type	size	lib	name
-----	-------	-------	------	------	------	-----	------

1324	0x00011282	0x76601282	GLOBAL	FUNC	0	KERNEL32.dll	WriteFile
------	------------	------------	--------	------	---	--------------	-----------

9	0x000109e4	0x766009e4	NONE	FUNC	0	API-MS-Win-Core-File-L1-1-0.dll	
---	------------	------------	------	------	---	---------------------------------	--

imp.WriteFile

nth	paddr	vaddr	bind	type	size	lib	name
-----	-------	-------	------	------	------	-----	------

1333	0x0002da28	0x7661da28	GLOBAL	FUNC	0	KERNEL32.dll	
------	------------	------------	--------	------	---	--------------	--

WriteProcessMemory

9	0x0001091c	0x7660091c	NONE	FUNC	0	API-MS-Win-Core-Memory-L1-1-0.dll	
---	------------	------------	------	------	---	-----------------------------------	--

imp.WriteProcessMemory

[0x0040101a]> dmi kernel32 IsDebuggerPresent

[Symbols]

nth	paddr	vaddr	bind	type	size	lib	name
-----	-------	-------	------	------	------	-----	------

770	0x0001494d	0x7660494d	GLOBAL	FUNC	0	KERNEL32.dll	
-----	------------	------------	--------	------	---	--------------	--

IsDebuggerPresent

4	0x00010d94	0x76600d94	NONE	FUNC	0	API-MS-Win-Core-Debug-L1-1-0.dll	
---	------------	------------	------	------	---	----------------------------------	--

imp.IsDebuggerPresent

[0x0040101a]> dmi kernel32 CreateProcessInternalA

[Symbols]

nth	paddr	vaddr	bind	type	size	lib	name
-----	-------	-------	------	------	------	-----	------

169	0x0002a507	0x7661a507	GLOBAL	FUNC	0	KERNEL32.dll	CreateProcessInternalA
-----	------------	------------	--------	------	---	--------------	------------------------

[0x0040101a]> dmi kernel32 CreateProcessInternalW

[Symbols]

```

nth paddr      vaddr      bind    type size lib          name
-----
170 0x00023c23 0x76613c23 GLOBAL FUNC 0     KERNEL32.dll CreateProcessInternalW

```

And we hit dc, the program calls VirtualAlloc. When the program calls VirtualAlloc and the call returns RAX/EAX will contain a pointer to the newly allocated space. We will keep track of those spaces to see if something gets written into them as the execution goes on:

```

[0x0040101a]> dc
hit breakpoint at: 0x76601826
[0x76601826]> dcr
hit breakpoint at: 0x7680f13d
[0x7680f13e]> dr eax
0x00220000
[0x7680f13e]>

```

After the first call, some content gets written into the first buffer, mm not very clear:

```

[0x7680f13e]> dc
hit breakpoint at: 0x76601826
[0x76601826]> pxw @ 0x00220000
0x00220000 0x81ec8b55 0xffff448c4 0xe3e853ff 0x8b000006 U....H...S.....
0x00220010 0x4589fc45 0x18a164c8 0x8b000000 0x408b3040 E..E.d.....@0.@
0x00220020 0x4c858954 0x6afffff6 0x04408d04 0x48858d50 T..L...j...@.P..H
0x00220030 0x50ffffff4 0xbec1c368 0x080de836 0xa1680000 ...Ph...6.....h.
0x00220040 0x50753cf4 0x00085fe8 0x8bd0ff00 0xffff44885 .<uP._.....H..
0x00220050 0x75c00bff 0x4c858b0b 0x8bffffff6 0x03eb3840 ...u...L...@8..
0x00220060 0x8b04408b 0x200d0440 0x3d002000 0x0077007c .@..@.. .|=|.w.
0x00220070 0xc95b0374 0xa485c6c3 0x41ffffffd 0xfda585c6 t.[.....A....
0x00220080 0xc644ffff 0xffffda685 0x85c656ff 0xfffffda7 ..D.....V.....
0x00220090 0xa885c641 0x50ffffffd 0xfda985c6 0xc649ffff A.....P.....I.
0x002200a0 0xffffdaa85 0x85c633ff 0xfffffdab 0xac85c632 .....3.....2...
0x002200b0 0x2effffffd 0xfdad85c6 0xc644ffff 0xfffdae85 .....D.....
0x002200c0 0x85c64cff 0xfffffdaf 0xb085c64c 0x00ffffffd .L.....L.....
0x002200d0 0xfda4858d 0xe850ffff 0x0000083c 0xf65485c6 .....P.<.....T.
0x002200e0 0xc653ffff 0xffff65585 0x85c64fff 0xfffff656 ..S..U...O..V...
0x002200f0 0x5785c646 0x54ffffff6 0xf65885c6 0xc657ffff F..W...T..X...W.

```

We run the second call till return, then check again:

```

[0x76601826]> dcr
hit breakpoint at: 0x7680f13d
[0x7680f13e]> dr eax
0x00230000

```

And we do it another time:

```
[0x7680f13e]> dc
hit breakpoint at: 0x76601826
[0x76601826]> dcr
hit breakpoint at: 0x7680f13d
[0x7680f13e]> dr eax
0x00240000
```

Then look at this!

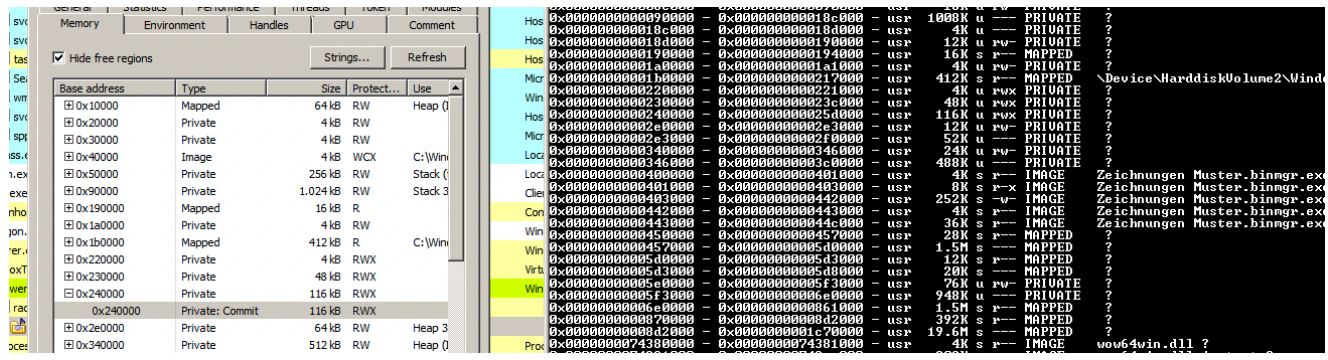
```
[0x7680f13e]> dr eax
0x00240000
[0x7680f13e]> dc
hit breakpoint at: 0x766043be
[0x766043be]> pxw 600 @ 0x00240000
0x00240000 0x00905a4d 0x00000003 0x00000004 0x0000ffff MZ.....
0x00240010 0x000000b8 0x00000000 0x00000040 0x00000000 .....@.....
0x00240020 0x00000000 0x00000000 0x00000000 0x00000000 .....
0x00240030 0x00000000 0x00000000 0x00000000 0x000000c8 .....
0x00240040 0x0eba1f0e 0xcd09b400 0x4c01b821 0x685421cd .....!..L.!Th
0x00240050 0x70207369 0x72676f72 0x63206d61 0x6f6e6e61 is program canno
0x00240060 0x65622074 0x6e757220 0x206e6920 0x20534f44 t be run in DOS
0x00240070 0x65646f6d 0x0a0d0d2e 0x00000024 0x00000000 mode....$.
0x00240080 0x55d9e2b7 0x06b783f3 0x06b783f3 0x06b783f3 ...U.....
0x00240090 0x06a49c7d 0x06b783c1 0x06a5a30f 0x06b783f2 }.....
0x002400a0 0x06b18534 0x06b783f2 0x68636952 0x06b783f3 4.....Rich....
0x002400b0 0x00000000 0x00000000 0x00000000 0x00000000 .....
0x002400c0 0x00000000 0x00000000 0x00004550 0x0003014c .....PE..L..
0x002400d0 0x4cdbdf6c 0x00000000 0x00000000 0x010f00e0 l..L.....
0x002400e0 0x0c05010b 0x0000b000 0x00001000 0x00015000 .....P..
0x002400f0 0x000209e0 0x00016000 0x00021000 0x00400000 .....`.....@.
0x00240100 0x00001000 0x00000200 0x00000004 0x00000000 .....
0x00240110 0x00000004 0x00000000 0x00022000 0x00001000 .....
0x00240120 0x00000000 0x00000002 0x00100000 0x00001000 .....
0x00240130 0x00100000 0x00001000 0x00000000 0x00000010 .....
0x00240140 0x00000000 0x00000000 0x000216b8 0x00000114 .....
0x00240150 0x00021000 0x000006b8 0x00000000 0x00000000 .....
0x00240160 0x00000000 0x00000000 0x00000000 0x00000000 .....
0x00240170 0x00000000 0x00000000 0x00000000 0x00000000 .....
0x00240180 0x00000000 0x00000000 0x00000000 0x00000000 .....
0x00240190 0x00000000 0x00000000 0x00000000 0x00000000 .....
0x002401a0 0x00000000 0x00000000 0x00000000 0x00000000 .....
0x002401b0 0x00000000 0x00000000 0x00000000 0x00000000 .....
0x002401c0 0x30585055 0x00000000 0x00015000 0x00001000 UPX0....P.....
0x002401d0 0x00000000 0x00000400 0x00000000 0x00000000 .....
0x002401e0 0x00000000 0xe0000080 0x31585055 0x00000000 .....UPX1....
0x002401f0 0x0000b000 0x00016000 0x0000ac00 0x00000400 .....`.....
0x00240200 0x00000000 0x00000000 0x00000000 0xe0000040 .....@...
0x00240210 0x7273722e 0x00000063 0x00001000 0x00021000 .rsrc.....
0x00240220 0x00000800 0x0000b000 0x00000000 0x00000000 .....
0x00240230 0x00000000 0xc0000040 0x00000000 0x00000000 .....@.....
0x00240240 0x00000000 0x00000000 0x00000000 0x00000000 .....
```

Another UPX packed binary!!

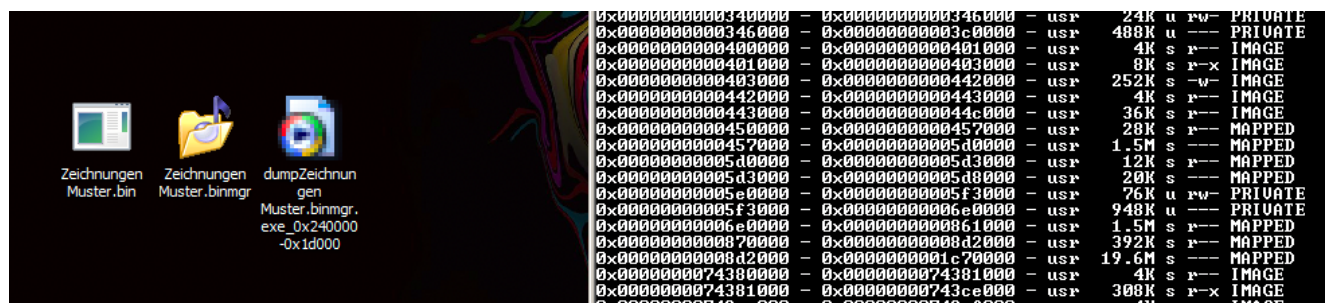
At this point, if we follow the execution of the program we'll see calls on VirtualProtect, related to changing RWX/RW permissions on the program itself, some content being written/decoded and a call to the binary via some decoded shellcode

```
[0x766043be]> pxx @ rsp
0x0018f3a0 0x002209db  ..". @ rsp PRIVATE  R W X 'leave' 'PRIVATE'
0x0018f3a4 0x00400000  ..@. IMAGE Zeichnungen Muster.binmgr.exe ascii ('
0x0018f3a8 0x0004c000  ....
0x0018f3ac 0x00000040  @... 64 ascii ('@')
0x0018f3b0 0x0018f3b4  .... PRIVATE  R W 0x1e1f5b
0x0018f3b4 0x001e1f5b  [... MAPPED
\Device\HarddiskVolume2\Windows\System32\locale.nls rsi R 0x6e006500
```

So if we want to move forward, we can just go dump the memory space of the UPX binary we detected. We can do it with process hacker or with radare itself + cat as we did on the previous post.



Unpacking the final binary



So we repeat the unpacking process with the final binary:


```
PS C:\Users\labo\Desktop> upx -d '.\dumpZeichnungen Muster.binmgr.exe_0x240000-0x1d000.exe'
```

Ultimate Packer for eXecutables

Copyright (C) 1996 - 2020

UPX 3.96w Markus Oberhumer, Laszlo Molnar & John Reiser Jan 23rd 2020

File size	Ratio	Format	Name
-----	-----	-----	-----
189952 <- 118784	62.53%	win32/pe	dumpZeichnungen

Muster.binmgr.exe_0x240000-0x1d000.exe

Unpacked 1 file.

```
PS C:\Users\labo\Desktop>
```

And we will see that this time everything looks clearer, we can even get a general idea of what the malware does:

```
PS C:\Users\labo\Desktop> radare2.exe '.\dumpZeichnungen Muster.binmgr.exe_0x240000-0x1d000.exe'
```

```
-- Mess with the best, Die like the rest
```

```
[0x00402c23]> aaa
```

```
[Warning: set your favourite calling convention in `e anal.cc=?`
```

```
[x] Analyze all flags starting with sym. and entry0 (aa)
```

```
[x] Analyze function calls (aac)
```

```
[x] Analyze len bytes of instructions for references (aar)
```

```
[x] Finding and parsing C++ vtables (avrr)
```

```
[x] Type matching analysis for all functions (aافت)
```

```
[x] Propagate noreturn information (aanr)
```

```
[x] Use -AA or aaaa to perform additional experimental analysis.
```

```
[0x00402c23]> afl
```

```
0x00402c23      6 245      entry0
0x0040181b     17 396      fcn.0040181b
0x00402d60      1  6      sub.KERNEL32.DLL_GetModuleHandleA
0x00402d66      1  6      sub.KERNEL32.DLL_GetProcAddress
0x00401000      1 24      fcn.00401000
0x004016d6     12 325      fcn.004016d6
0x00401611      6 106      fcn.00401611
0x00402d48      1  6      sub.KERNEL32.DLL_GetCurrentProcess
0x00402e02      1  6      sub.advapi32.dll_OpenProcessToken
0x00402dfc      1  6      sub.advapi32.dll_LookupPrivilegeValueA
0x00402df6      1  6      sub.advapi32.dll_AdjustTokenPrivileges
0x00402d18      1  6      fcn.00402d18
0x00402a32      1 92      fcn.00402a32
0x00402786      1 14      fcn.00402786
0x00402d4e      1  6      sub.KERNEL32.DLL_GetCurrentProcessId
0x004026d8     13 174      fcn.004026d8
0x0040289a      4 61      fcn.0040289a
0x004027c2      5 216      fcn.004027c2
0x00402dd2      1  6      sub.KERNEL32.DLL_VirtualProtect
0x004027a2      3 32      fcn.004027a2
0x00401dbf      1 25      fcn.00401dbf
0x00401c61     62 350      fcn.00401c61
0x00402dc0      1  6      sub.KERNEL32.DLL_VirtualAlloc
0x00401018      3 37      fcn.00401018
0x00402794      1 14      fcn.00402794
0x00401223      1 93      fcn.00401223
0x00402d24      1  6      sub.KERNEL32.DLL_CreateProcessA
0x00402a8e      3 55      fcn.00402a8e
0x004028d7      5 122      fcn.004028d7
0x00402dcc      1  6      sub.KERNEL32.DLL_VirtualFree
0x004014cc      4 290      fcn.004014cc
0x00402d72      1  6      sub.KERNEL32.DLL_GetWindowsDirectoryA
0x00401052      5 57      fcn.00401052
0x00402d6c      1  6      sub.KERNEL32.DLL_GetVolumeInformationA
0x004013e0      1 58      fcn.004013e0
0x00402df0      1  6      sub.user32.dll_wsprintfA
0x004015ee      3 35      fcn.004015ee
0x00402d8a      1  6      sub.KERNEL32.DLL_OpenMutexA
0x00402ac5      5 169      fcn.00402ac5
```

0x00402da2	1 6		sub.KERNEL32.DLL_Sleep
0x00402bd6	6 77		fcn.00402bd6
0x00402d30	1 6		sub.KERNEL32.DLL_ExitProcess
0x0040108b	5 55		fcn.0040108b
0x00402d78	1 6		sub.KERNEL32.DLL_GlobalAlloc
0x00402d42	1 6		sub.KERNEL32.DLL_FindFirstFileA
0x00402d3c	1 6		sub.KERNEL32.DLL_FindClose
0x00401280	4 20		fcn.00401280
0x00401294	19 167		fcn.00401294
0x0040133b	6 95		fcn.0040133b
0x004013be	1 34		fcn.004013be
0x0040139a	1 36		fcn.0040139a
0x00402e0e	1 6		sub.advapi32.dll_RegOpenKeyA
0x00402e14	1 6		sub.advapi32.dll_RegQueryValueExA
0x0040103d	1 21		fcn.0040103d
0x00402e08	1 6		sub.advapi32.dll_RegCloseKey
0x00402dea	1 6		sub.KERNEL32.DLL_lstrlenA
0x00402de4	1 6		sub.KERNEL32.DLL_lstrcpyA
0x0040167b	8 91		fcn.0040167b
0x004011df	4 68		fcn.004011df
0x00402d36	1 6		sub.KERNEL32.DLL_ExpandEnvironmentStringsA
0x004019d5	1 28		fcn.004019d5
0x00401a3e	4 43		fcn.00401a3e
0x00401a1e	4 32		fcn.00401a1e
0x00401a69	4 63		fcn.00401a69
0x00401aa8	4 72		fcn.00401aa8
0x00401af0	4 66		fcn.00401af0
0x00402dc6	1 6		sub.KERNEL32.DLL_VirtualAllocEx
0x00402dde	1 6		sub.KERNEL32.DLL_WriteProcessMemory
0x0040255c	15 380		fcn.0040255c
0x00401dd8	12 84		fcn.00401dd8
0x00401e2c	3 61		fcn.00401e2c
0x00402d5a	1 6		sub.KERNEL32.DLL_GetModuleFileNameA
0x004019a7	1 28		fcn.004019a7
0x00402d96	1 6		sub.KERNEL32.DLL_ReadProcessMemory
0x00402168	4 98		fcn.00402168
0x00401b32	4 72		fcn.00401b32
0x00401b7a	21 231	-> 229	fcn.00401b7a
0x00402d54	1 6		sub.KERNEL32.DLL_GetCurrentThreadId
0x00402d2a	1 6		sub.KERNEL32.DLL_CreateToolhelp32Snapshot
0x00402dae	1 6		sub.KERNEL32.DLL_Thread32First
0x00402d90	1 6		sub.KERNEL32.DLL_OpenThread
0x00402da8	1 6		sub.KERNEL32.DLL_SuspendThread
0x00402d9c	1 6		sub.KERNEL32.DLL_ResumeThread
0x00402db4	1 6		sub.KERNEL32.DLL_Thread32Next
0x004021ca	7 128		fcn.004021ca
0x00401f0e	21 602		fcn.00401f0e
0x00402dd8	1 6		sub.KERNEL32.DLL_VirtualProtectEx
0x00402d1e	1 6		sub.KERNEL32.DLL_CreateFileMappingA
0x00402d7e	1 6		sub.KERNEL32.DLL_MapViewOfFile
0x00402d84	1 6		sub.KERNEL32.DLL_OpenFileMappingA

```
0x00402dba    1 6          sub.KERNEL32.DLL_UnmapViewOfFile  
[0x00402c23]>
```

For what I see here, it looks like it performs some kind of process injection. It may also try to gain some persistence or information on the machine by the register and maybe resolve some more calls related to its actual behavior by GetModuleHandle, LoadLibrary etc.

```

[0x00402c23]> pdf
      ;-- eip:
/ 245: entry0 ();
|      0x00402c23      e8f3ebffff      call fcn.0040181b
|      0x00402c28      83f801          cmp eax, 1          ; 1
|      ,=< 0x00402c2b    0f85e0000000    jne 0x402d11
|      | 0x00402c31      6800040000      push 0x400          ; 1024
|      | 0x00402c36      6870da4100      push 0x41da70
|      | 0x00402c3b      e8c0e3ffff      call fcn.00401000
|      | 0x00402c40      6800040000      push 0x400          ; 1024
|      | 0x00402c45      6870da4100      push 0x41da70
|      | 0x00402c4a      e887eaffffff      call fcn.004016d6
|      | 0x00402c4f      83f801          cmp eax, 1          ; 1
|      ,==< 0x00402c52    0f85b9000000    jne 0x402d11
|      || 0x00402c58      6821404000      push 0x404021      ; '!@@' ;
"SeDebugPrivilege"
|      || 0x00402c5d      e8afe9ffff      call fcn.00401611
|      || 0x00402c62      6800100100      push 0x11000
|      || 0x00402c67      6870424000      push 0x404270      ; 'pB@' ;
"MZ\x90"
|      || 0x00402c6c      e8c1fdffff      call fcn.00402a32
|      || 0x00402c71      6a01          push 1          ; 1
|      || 0x00402c73      6870da4100      push 0x41da70
|      || 0x00402c78      e8a6e5ffff      call fcn.00401223
|      || 0x00402c7d      e80cfeffff      call fcn.00402a8e
|      || 0x00402c82      b800880000      mov eax, 0x8800
|      || 0x00402c87      83f801          cmp eax, 1          ; 1
|      ,===< 0x00402c8a    0f8681000000    jbe 0x402d11
|      ||| 0x00402c90      6871de4100      push 0x41de71
|      ||| 0x00402c95      6a00          push 0
|      ||| 0x00402c97      68ea0d0000      push 0xdea      ; 3562
|      ||| 0x00402c9c      e82be8ffff      call fcn.004014cc
|      ||| 0x00402ca1      68b1de4100      push 0x41deb1
|      ||| 0x00402ca6      6a00          push 0
|      ||| 0x00402ca8      686b050000      push 0x56b      ; 1387
|      ||| 0x00402cad      e81ae8ffff      call fcn.004014cc
|      ||| 0x00402cb2      6871de4100      push 0x41de71
|      ||| 0x00402cb7      e832e9ffff      call fcn.004015ee
|      ||| 0x00402cbc      0bc0          or eax, eax
|      ,====< 0x00402cbe    7551          jne 0x402d11
|      |||| 0x00402cc0      6860424000      push 0x404260      ; '`B@'
|      |||| 0x00402cc5      6a00          push 0
|      |||| 0x00402cc7      6800880000      push 0x8800
|      |||| 0x00402ccc      6870524100      push 0x415270      ; 'pRA' ;
"MZ\x90"
|      |||| 0x00402cd1      68b1de4100      push 0x41deb1
|      |||| 0x00402cd6      e8eafdffff      call fcn.00402ac5
|      |||| 0x00402cdb      6800880000      push 0x8800
|      |||| 0x00402ce0      6870524100      push 0x415270      ; 'pRA' ;
"MZ\x90"
|      |||| 0x00402ce5      e848fdffff      call fcn.00402a32
|      |||| 0x00402cea      6a01          push 1          ; 1

```

```

|      | 0x00402cec      6870da4100      push 0x41da70
|      | 0x00402cf1      e82de5ffff      call fcn.00401223
|      | 0x00402cf6      e893fdffff      call fcn.00402a8e
|      | 0x00402cfb      6888130000      push 0x1388
|      | 0x00402d00      e89d000000      call sub.KERNEL32.DLL_Sleep
|      | 0x00402d05      6a01            push 1                ; 1
|      | 0x00402d07      6860424000      push 0x404260        ; '\`B@'
|      | 0x00402d0c      e8c5fefeffff      call fcn.00402bd6
|      | ; CODE XREFS from entry0 @ 0x402c2b, 0x402c52, 0x402c8a, 0x402cbe
|      | ````-> 0x00402d11      6a00            push 0
\      | 0x00402d13      e818000000      call sub.KERNEL32.DLL_ExitProcess
[0x00402c23]>

```

If we look at the imports, we see that same thing, we can also guess an infection / replication on the machine or network.

[0x00402c23]> ii

[Imports]

nth	vaddr	bind	type	lib	name
1	0x0040301c	NONE	FUNC	KERNEL32.DLL	FindFirstFileA
2	0x00403020	NONE	FUNC	KERNEL32.DLL	GetCurrentProcess
3	0x00403024	NONE	FUNC	KERNEL32.DLL	GetCurrentProcessId
4	0x00403028	NONE	FUNC	KERNEL32.DLL	GetCurrentThreadId
5	0x0040302c	NONE	FUNC	KERNEL32.DLL	GetModuleFileNameA
6	0x00403030	NONE	FUNC	KERNEL32.DLL	GetModuleHandleA
7	0x00403034	NONE	FUNC	KERNEL32.DLL	GetProcAddress
8	0x00403038	NONE	FUNC	KERNEL32.DLL	GetVolumeInformationA
9	0x0040303c	NONE	FUNC	KERNEL32.DLL	GetWindowsDirectoryA
10	0x00403040	NONE	FUNC	KERNEL32.DLL	GlobalAlloc
11	0x00403044	NONE	FUNC	KERNEL32.DLL	MapViewOfFile
12	0x00403048	NONE	FUNC	KERNEL32.DLL	OpenFileMappingA
13	0x0040304c	NONE	FUNC	KERNEL32.DLL	OpenMutexA
14	0x00403050	NONE	FUNC	KERNEL32.DLL	OpenThread
15	0x00403054	NONE	FUNC	KERNEL32.DLL	FindClose
16	0x00403058	NONE	FUNC	KERNEL32.DLL	ResumeThread
17	0x0040305c	NONE	FUNC	KERNEL32.DLL	Sleep
18	0x00403060	NONE	FUNC	KERNEL32.DLL	SuspendThread
19	0x00403064	NONE	FUNC	KERNEL32.DLL	Thread32First
20	0x00403068	NONE	FUNC	KERNEL32.DLL	Thread32Next
21	0x0040306c	NONE	FUNC	KERNEL32.DLL	UnmapViewOfFile
22	0x00403070	NONE	FUNC	KERNEL32.DLL	VirtualAlloc
23	0x00403074	NONE	FUNC	KERNEL32.DLL	VirtualAllocEx
24	0x00403078	NONE	FUNC	KERNEL32.DLL	VirtualFree
25	0x0040307c	NONE	FUNC	KERNEL32.DLL	VirtualProtect
26	0x00403080	NONE	FUNC	KERNEL32.DLL	VirtualProtectEx
27	0x00403084	NONE	FUNC	KERNEL32.DLL	WriteProcessMemory
28	0x00403088	NONE	FUNC	KERNEL32.DLL	lstrcpyA
29	0x0040308c	NONE	FUNC	KERNEL32.DLL	lstrlenA
30	0x00403090	NONE	FUNC	KERNEL32.DLL	ExpandEnvironmentStringsA
31	0x00403094	NONE	FUNC	KERNEL32.DLL	ExitProcess
32	0x00403098	NONE	FUNC	KERNEL32.DLL	CreateToolhelp32Snapshot
33	0x0040309c	NONE	FUNC	KERNEL32.DLL	CreateProcessA
34	0x004030a0	NONE	FUNC	KERNEL32.DLL	CreateFileMappingA
35	0x004030a4	NONE	FUNC	KERNEL32.DLL	ReadProcessMemory
36	0x004030a8	NONE	FUNC	KERNEL32.DLL	CloseHandle
1	0x00403000	NONE	FUNC	advapi32.dll	OpenProcessToken
2	0x00403004	NONE	FUNC	advapi32.dll	LookupPrivilegeValueA
3	0x00403008	NONE	FUNC	advapi32.dll	AdjustTokenPrivileges
4	0x0040300c	NONE	FUNC	advapi32.dll	RegOpenKeyA
5	0x00403010	NONE	FUNC	advapi32.dll	RegQueryValueExA
6	0x00403014	NONE	FUNC	advapi32.dll	RegCloseKey
1	0x004030b0	NONE	FUNC	user32.dll	wsprintfA

[0x00402c23]>

izz

```

204 0x00002a21 0x00404021 16 17 .data ascii SeDebugPrivilege
205 0x00002a32 0x00404032 45 46 .data ascii %ProgramFiles%\Internet
Explorer\iexplore.exe
206 0x00002a60 0x00404060 64 65 .data ascii
SOFTWARE\Microsoft\Windows\CurrentVersion\App Paths\IEXPLORE.EXE
207 0x00002aa1 0x004040a1 23 24 .data ascii http\shell\open\command

```

And the strings located inside point to interesting places.

Conclusions

So at this time I think we can confidently say that we unpacked the malware, now we have three let's say "modules" we can independently analyse, to see if they unpack anything else, to check their functionalities and see if they interact and how.

The report on anyrun related to this malware, offers an interesting process diagram, that correctly corresponds to our findings!



Tackling the problem on Windows 10

If you tried to replicate this analysis but decided to use Windows 10 instead of Windows 7 you'll probably come up with some errors leading to not being able to unpack the last stage or actually execute the program.

So, to sum it up. This malware is not modern (it uses upx!!) it was developed and spread in a time where Win7 was the latest, so it is prepared for it. An important detail related to this, we see at least in the second binary is that at some point it allocates space, and write some random data that looks like shellcode into it. Well it is actually shellcode, and the execution flow of the program gets transferred there at some point...

And if we dump it and manually reverse it, we'll see plenty of references to "hardcoded" stuff, some calls that are not even present in Windows10, so that's the problem.

[0x00000000]> pd 120

```
0x00000000 55 push rbp
0x00000001 8bec mov ebp, esp
0x00000003 81c448f4ffff add esp, 0xfffff448
0x00000009 53 push rbx
0x0000000a e8e3060000 call 0x6f2
0x0000000f 8b45fc mov eax, dword [rbp - 4]
0x00000012 8945c8 mov dword [rbp - 0x38], eax
0x00000015 64a118000000. movabs eax, dword fs:[0x8b30408b00000018]
0x0000001f 4054 push rsp
0x00000021 89854cf6ffff mov dword [rbp - 0x9b4], eax
0x00000027 6a04 push 4
0x00000029 8d4004 lea eax, [rax + 4]
0x0000002c 50 push rax
0x0000002d 8d8548f4ffff lea eax, [rbp - 0xbb8]
0x00000033 50 push rax
0x00000034 68c3c1be36 push 0x36bec1c3
0x00000039 e80d080000 call 0x84b
0x0000003e 68a1f43c75 push 0x753cf4a1
0x00000043 50 push rax
0x00000044 e85f080000 call 0x8a8
0x00000049 ffd0 call rax
0x0000004b 8b8548f4ffff mov eax, dword [rbp - 0xbb8]
0x00000051 0bc0 or eax, eax
,=< 0x00000053 750b jne 0x60
| 0x00000055 8b854cf6ffff mov eax, dword [rbp - 0x9b4]
| 0x0000005b 8b4038 mov eax, dword [rax + 0x38]
,==< 0x0000005e eb03 jmp 0x63
|`-> 0x00000060 8b4004 mov eax, dword [rax + 4]
`--> 0x00000063 8b4004 mov eax, dword [rax + 4]
0x00000066 0d20002000 or eax, 0x200020
0x0000006b 3d7c007700 cmp eax, 0x77007c
,=< 0x00000070 7403 je 0x75
| 0x00000072 5b pop rbx
| 0x00000073 c9 leave
| 0x00000074 c3 ret
`-> 0x00000075 c685a4fdffff. mov byte [rbp - 0x25c], 0x41 ; 'A'
0x0000007c c685a5fdffff. mov byte [rbp - 0x25b], 0x44 ; 'D'
0x00000083 c685a6fdffff. mov byte [rbp - 0x25a], 0x56 ; 'V'
0x0000008a c685a7fdffff. mov byte [rbp - 0x259], 0x41 ; 'A'
0x00000091 c685a8fdffff. mov byte [rbp - 0x258], 0x50 ; 'P'
0x00000098 c685a9fdffff. mov byte [rbp - 0x257], 0x49 ; 'I'
0x0000009f c685aafdffff. mov byte [rbp - 0x256], 0x33 ; '3'
0x000000a6 c685abfdffff. mov byte [rbp - 0x255], 0x32 ; '2'
0x000000ad c685acfdffff. mov byte [rbp - 0x254], 0x2e ; '.'
0x000000b4 c685adfdffff. mov byte [rbp - 0x253], 0x44 ; 'D'
0x000000bb c685aefdffff. mov byte [rbp - 0x252], 0x4c ; 'L'
0x000000c2 c685affdffff. mov byte [rbp - 0x251], 0x4c ; 'L'
0x000000c9 c685b0fdffff. mov byte [rbp - 0x250], 0
0x000000d0 8d85a4fdffff lea eax, [rbp - 0x25c]
0x000000d6 50 push rax
0x000000d7 e83c080000 call 0x918
```

```

0x000000dc    c68554f6ffff.  mov byte [rbp - 0x9ac], 0x53 ; 'S'
0x000000e3    c68555f6ffff.  mov byte [rbp - 0x9ab], 0x4f ; 'O'
0x000000ea    c68556f6ffff.  mov byte [rbp - 0x9aa], 0x46 ; 'F'
0x000000f1    c68557f6ffff.  mov byte [rbp - 0x9a9], 0x54 ; 'T'
0x000000f8    c68558f6ffff.  mov byte [rbp - 0x9a8], 0x57 ; 'W'
0x000000ff    c68559f6ffff.  mov byte [rbp - 0x9a7], 0x41 ; 'A'
0x00000106    c6855af6ffff.  mov byte [rbp - 0x9a6], 0x52 ; 'R'
0x0000010d    c6855bf6ffff.  mov byte [rbp - 0x9a5], 0x45 ; 'E'
0x00000114    c6855cf6ffff.  mov byte [rbp - 0x9a4], 0x5c ; '\\\
0x0000011b    c6855df6ffff.  mov byte [rbp - 0x9a3], 0x4d ; 'M'
0x00000122    c6855ef6ffff.  mov byte [rbp - 0x9a2], 0x69 ; 'i'
0x00000129    c6855ff6ffff.  mov byte [rbp - 0x9a1], 0x63 ; 'c'
0x00000130    c68560f6ffff.  mov byte [rbp - 0x9a0], 0x72 ; 'r'
0x00000137    c68561f6ffff.  mov byte [rbp - 0x99f], 0x6f ; 'o'
0x0000013e    c68562f6ffff.  mov byte [rbp - 0x99e], 0x73 ; 's'
0x00000145    c68563f6ffff.  mov byte [rbp - 0x99d], 0x6f ; 'o'
0x0000014c    c68564f6ffff.  mov byte [rbp - 0x99c], 0x66 ; 'f'
0x00000153    c68565f6ffff.  mov byte [rbp - 0x99b], 0x74 ; 't'
0x0000015a    c68566f6ffff.  mov byte [rbp - 0x99a], 0x5c ; '\\\
0x00000161    c68567f6ffff.  mov byte [rbp - 0x999], 0x57 ; 'W'
0x00000168    c68568f6ffff.  mov byte [rbp - 0x998], 0x69 ; 'i'
0x0000016f    c68569f6ffff.  mov byte [rbp - 0x997], 0x6e ; 'n'
0x00000176    c6856af6ffff.  mov byte [rbp - 0x996], 0x64 ; 'd'
0x0000017d    c6856bf6ffff.  mov byte [rbp - 0x995], 0x6f ; 'o'
0x00000184    c6856cf6ffff.  mov byte [rbp - 0x994], 0x77 ; 'w'
0x0000018b    c6856df6ffff.  mov byte [rbp - 0x993], 0x73 ; 's'
0x00000192    c6856ef6ffff.  mov byte [rbp - 0x992], 0x20 ;
"T\x89\x85L\xfb\xff\xffj\x04\x8d@\x04P\x8d\x85H\xfb\xff\xffPh\xc3\u007e6\xe8\r\b"
0x00000199    c6856ff6ffff.  mov byte [rbp - 0x991], 0x4e ; 'N'
0x000001a0    c68570f6ffff.  mov byte [rbp - 0x990], 0x54 ; 'T'
0x000001a7    c68571f6ffff.  mov byte [rbp - 0x98f], 0x5c ; '\\\
0x000001ae    c68572f6ffff.  mov byte [rbp - 0x98e], 0x43 ; 'C'
0x000001b5    c68573f6ffff.  mov byte [rbp - 0x98d], 0x75 ; 'u'
0x000001bc    c68574f6ffff.  mov byte [rbp - 0x98c], 0x72 ; 'r'
0x000001c3    c68575f6ffff.  mov byte [rbp - 0x98b], 0x72 ; 'r'
0x000001ca    c68576f6ffff.  mov byte [rbp - 0x98a], 0x65 ; 'e'
0x000001d1    c68577f6ffff.  mov byte [rbp - 0x989], 0x6e ; 'n'
0x000001d8    c68578f6ffff.  mov byte [rbp - 0x988], 0x74 ; 't'
0x000001df    c68579f6ffff.  mov byte [rbp - 0x987], 0x56 ; 'V'
0x000001e6    c6857af6ffff.  mov byte [rbp - 0x986], 0x65 ; 'e'
0x000001ed    c6857bf6ffff.  mov byte [rbp - 0x985], 0x72 ; 'r'
0x000001f4    c6857cf6ffff.  mov byte [rbp - 0x984], 0x73 ; 's'
0x000001fb    c6857df6ffff.  mov byte [rbp - 0x983], 0x69 ; 'i'
0x00000202    c6857ef6ffff.  mov byte [rbp - 0x982], 0x6f ; 'o'
0x00000209    c6857ff6ffff.  mov byte [rbp - 0x981], 0x6e ; 'n'
0x00000210    c68580f6ffff.  mov byte [rbp - 0x980], 0x5c ; '\\\
0x00000217    c68581f6ffff.  mov byte [rbp - 0x97f], 0x57 ; 'W'
0x0000021e    c68582f6ffff.  mov byte [rbp - 0x97e], 0x69 ; 'i'
0x00000225    c68583f6ffff.  mov byte [rbp - 0x97d], 0x6e ; 'n'
0x0000022c    c68584f6ffff.  mov byte [rbp - 0x97c], 0x6c ; 'l'
0x00000233    c68585f6ffff.  mov byte [rbp - 0x97b], 0x6f ; 'o'
0x0000023a    c68586f6ffff.  mov byte [rbp - 0x97a], 0x67 ; 'g'

```

```
0x00000241      c68587f6ffff.  mov byte [rbp - 0x979], 0x6f ; 'o'
0x00000248      c68588f6ffff.  mov byte [rbp - 0x978], 0x6e ; 'n'
0x0000024f      c68589f6ffff.  mov byte [rbp - 0x977], 0x5c ; '\\
0x00000256      c6858af6ffff.  mov byte [rbp - 0x976], 0
0x0000025d      8d8550f6ffff  lea eax, [rbp - 0x9b0]
0x00000263      50             push rax
0x00000264      8d8554f6ffff  lea eax, [rbp - 0x9ac]
0x0000026a      50             push rax
```

So my advice in here would be to try to adjust the OS version to the Malware as much as possible, to avoid losing precious time in the analysis.