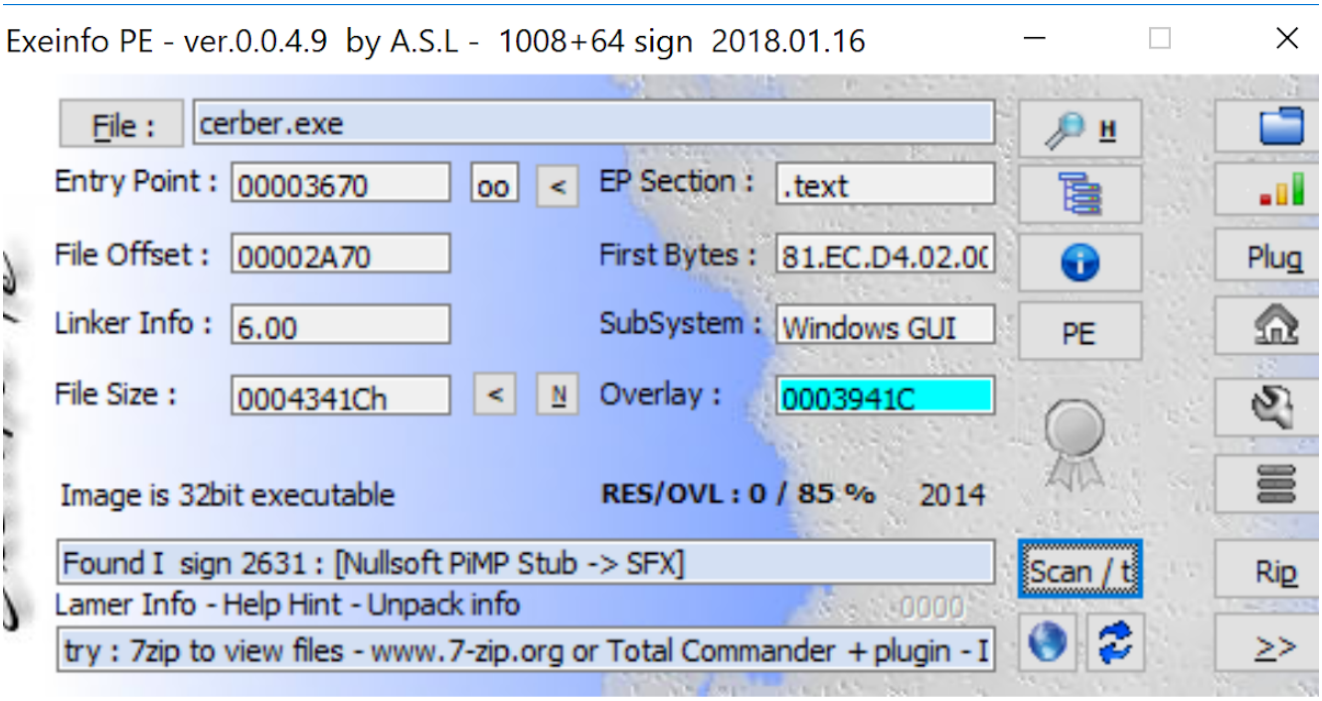
Reversing Cerber - RaaS

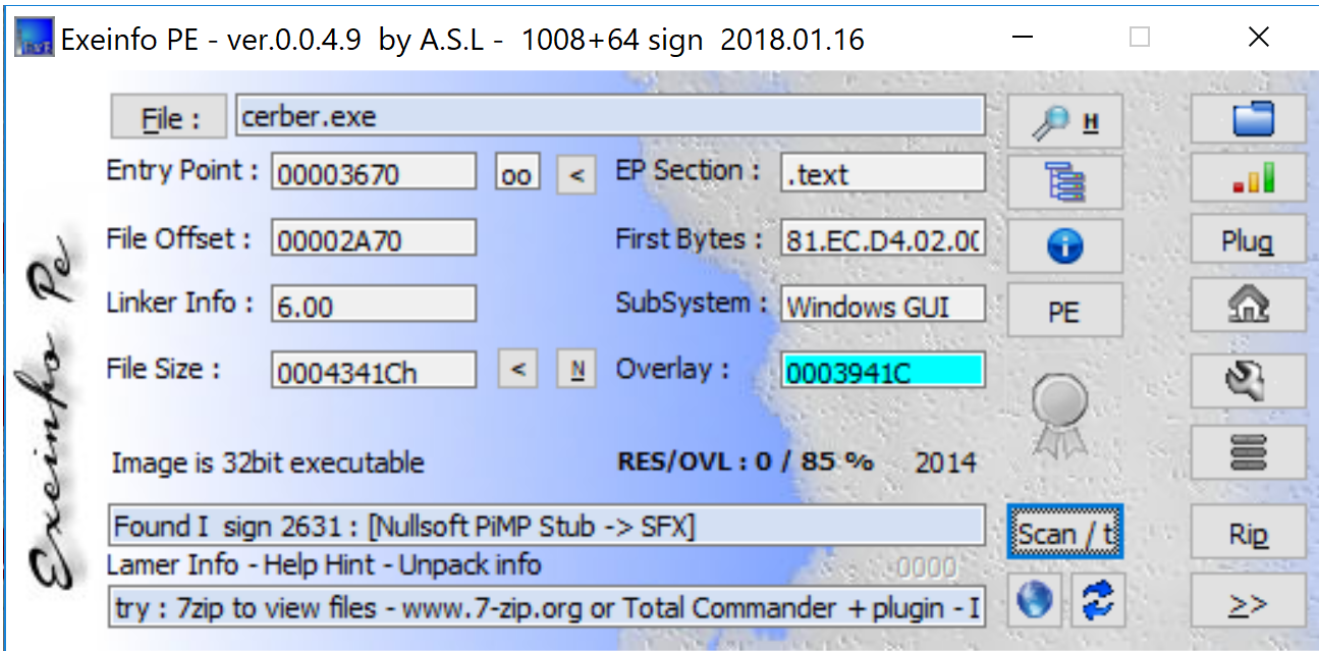
rinseandrepeatanalysis.blogspot.com/2018/08/reversing-cerber-raas.html

James Haughom

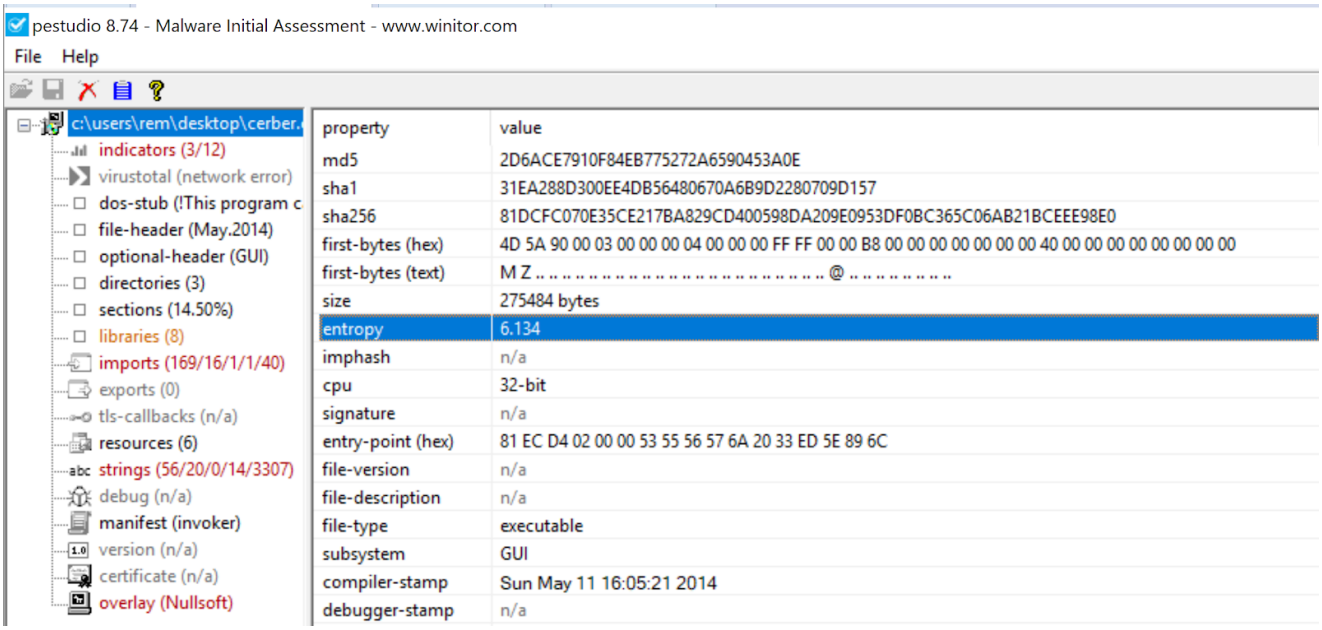


Cerber has established itself as one of the most successful ransomware families to date. Distributed as Raas (Ransomware as a Service), the malware has retained popularity with over 6 known variants.

The malware is packed with Nullsoft PiMP (plugin Mini Packager) which hides much of the malware's true functionality.



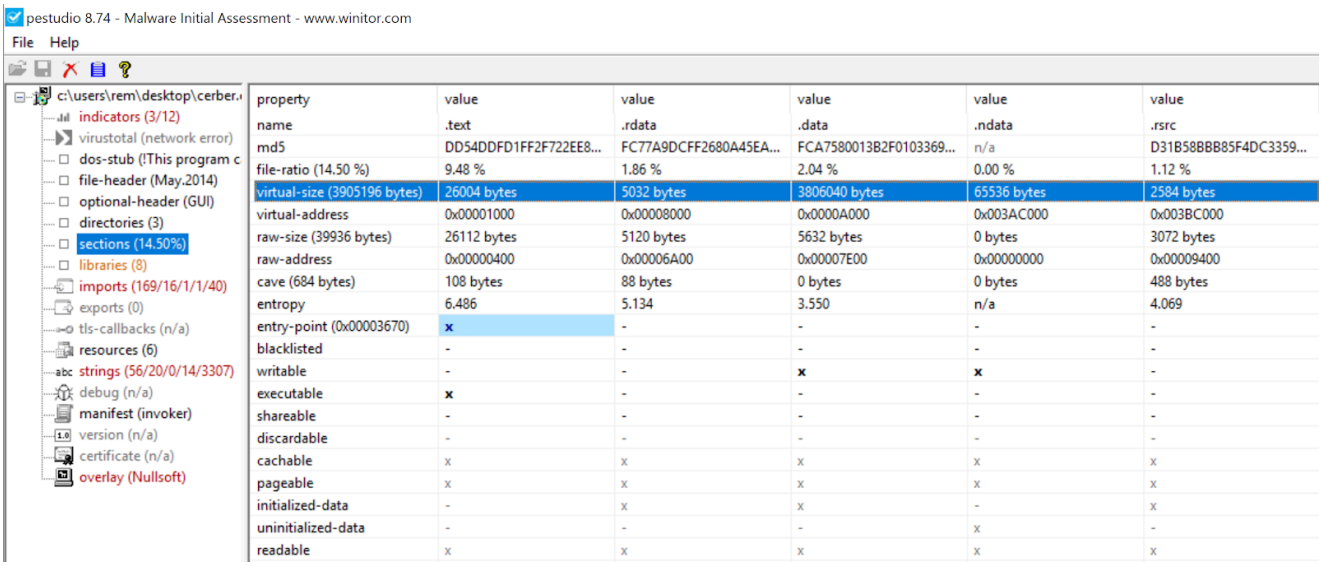
As a whole, the malware's entropy is quite high (6.1), indicating packed code.



The malware contains an anomalous PE section ".ndata" which has a virtual size much higher than its physical/raw size. This indicates that there is a good chunk of code that won't present itself until runtime, once the malware is loaded in memory.

```
Sections
================================================================================
==========
Name       VirtAddr     VirtSize     RawSize    MD5
 Entropy
--------------------------------------------------------------------------------
----------
.text      0x1000       0x6594       0x6600     dd54ddfd1ff2f722ee8321d66d57b679
 6.486495
.rdata     0x8000       0x13a8       0x1400     fc77a9dcff2680a45ea84b8557e704f2
 5.134399
.data      0xa000       0x3a1358     0x1600     fca7580013b2f0103369e99ff7b40893
 3.549948
.ndata     0x3ac000     0x10000      0x0        d41d8cd98f00b204e9800998ecf8427e
 0.000000    [SUSPICIOUS]
.rsrc      0x3bc000     0xa18        0xc00      d31b58bbb85f4dc33599a78f3ac0eddc
 4.069419
```

pestudio marks the .text/.code section as highly entropic (6.4), this is where the actual code/instructions are stored in a PE.



property	value	value	value	value	value
name	.text	.rdata	.data	.ndata	.rsrc
md5	DD54DDFD1FF2F722EE8...	FC77A9DCFF2680A45EA...	FCA7580013B2F0103369...	n/a	D31B58BBB85F4DC3359...
file-ratio (14.50 %)	9.48 %	1.86 %	2.04 %	0.00 %	1.12 %
virtual-size (3905196 bytes)	26004 bytes	5032 bytes	3806040 bytes	65536 bytes	2584 bytes
virtual-address	0x00001000	0x00008000	0x0000A000	0x003AC000	0x003BC000
raw-size (39936 bytes)	26112 bytes	5120 bytes	5632 bytes	0 bytes	3072 bytes
raw-address	0x00000400	0x00006A00	0x00007E00	0x00000000	0x00009400
cave (684 bytes)	108 bytes	88 bytes	0 bytes	0 bytes	488 bytes
entropy	6.486	5.134	3.550	n/a	4.069
entry-point (0x00003670)	x	-	-	-	-
blacklisted	-	-	-	-	-
writable	-	-	x	x	-
executable	x	-	-	-	-
shareable	-	-	-	-	-
discardable	-	-	-	-	-
cachable	x	x	x	x	x
pageable	x	x	x	x	x
initialized-data	-	x	x	-	x
uninitialized-data	-	-	-	x	-
readable	x	x	x	x	x

I conducted this analysis in a host-only virtual network with a Windows 10 VM routing its network traffic to a REMnux VM running fakedns, iNetSim, and Wireshark. The malware happily runs in the VM with security tools running, dropping several files to disk. The number of bytes written to the file 'collages.dll' is the same as the virtual size of the '.ndata' section.

Process Monitor - Sysinternals: www.sysinternals.com

File Edit Event Filter Tools Options Help

Time ...	Process Name	PID	Operation	Path	Result	Detail
11:50:...	cerber.exe	6668	WriteFile	C:\Users\REM\AppData\Local\Temp\floppy_disk.png	SUCCESS	Offset: 0, Length: 3,676, Priority: Normal
11:50:...	cerber.exe	6668	WriteFile	C:\Users\REM\AppData\Local\Temp\floppy_disk_disabled.png	SUCCESS	Offset: 0, Length: 1,416, Priority: Normal
11:50:...	cerber.exe	6668	WriteFile	C:\Users\REM\AppData\Local\Temp\flat.xsl	SUCCESS	Offset: 0, Length: 1,858, Priority: Normal
11:50:...	cerber.exe	6668	WriteFile	C:\Users\REM\AppData\Local\Temp\Diacid.cpmO	SUCCESS	Offset: 0, Length: 32,768, Priority: Normal
11:50:...	cerber.exe	6668	WriteFile	C:\Users\REM\AppData\Local\Temp\Diacid.cpmO	SUCCESS	Offset: 32,768, Length: 32,768, Priority: Normal
11:50:...	cerber.exe	6668	WriteFile	C:\Users\REM\AppData\Local\Temp\Diacid.cpmO	SUCCESS	Offset: 65,536, Length: 32,768
11:50:...	cerber.exe	6668	WriteFile	C:\Users\REM\AppData\Local\Temp\Diacid.cpmO	SUCCESS	Offset: 98,304, Length: 32,768, Priority: Normal
11:50:...	cerber.exe	6668	WriteFile	C:\Users\REM\AppData\Local\Temp\Diacid.cpmO	SUCCESS	Offset: 131,072, Length: 32,768
11:50:...	cerber.exe	6668	WriteFile	C:\Users\REM\AppData\Local\Temp\Diacid.cpmO	SUCCESS	Offset: 163,840, Length: 15,723
11:50:...	cerber.exe	6668	WriteFile	C:\Users\REM\AppData\Local\Temp\collages.dll	SUCCESS	Offset: 0, Length: 32,768, Priority: Normal
11:50:...	cerber.exe	6668	WriteFile	C:\Users\REM\AppData\Local\Temp\collages.dll	SUCCESS	Offset: 32,768, Length: 32,768, Priority: Normal
11:50:...	cerber.exe	6668	WriteFile	C:\Users\REM\AppData\Local\Temp\collages.dll	SUCCESS	Offset: 65,536, Length: 17,408
11:50:...	cerber.exe	6668	WriteFile	C:\Users\REM\AppData\Local\Temp\nsmAD93.tmp\System.dll	SUCCESS	Offset: 0, Length: 11,776, Priority: Normal

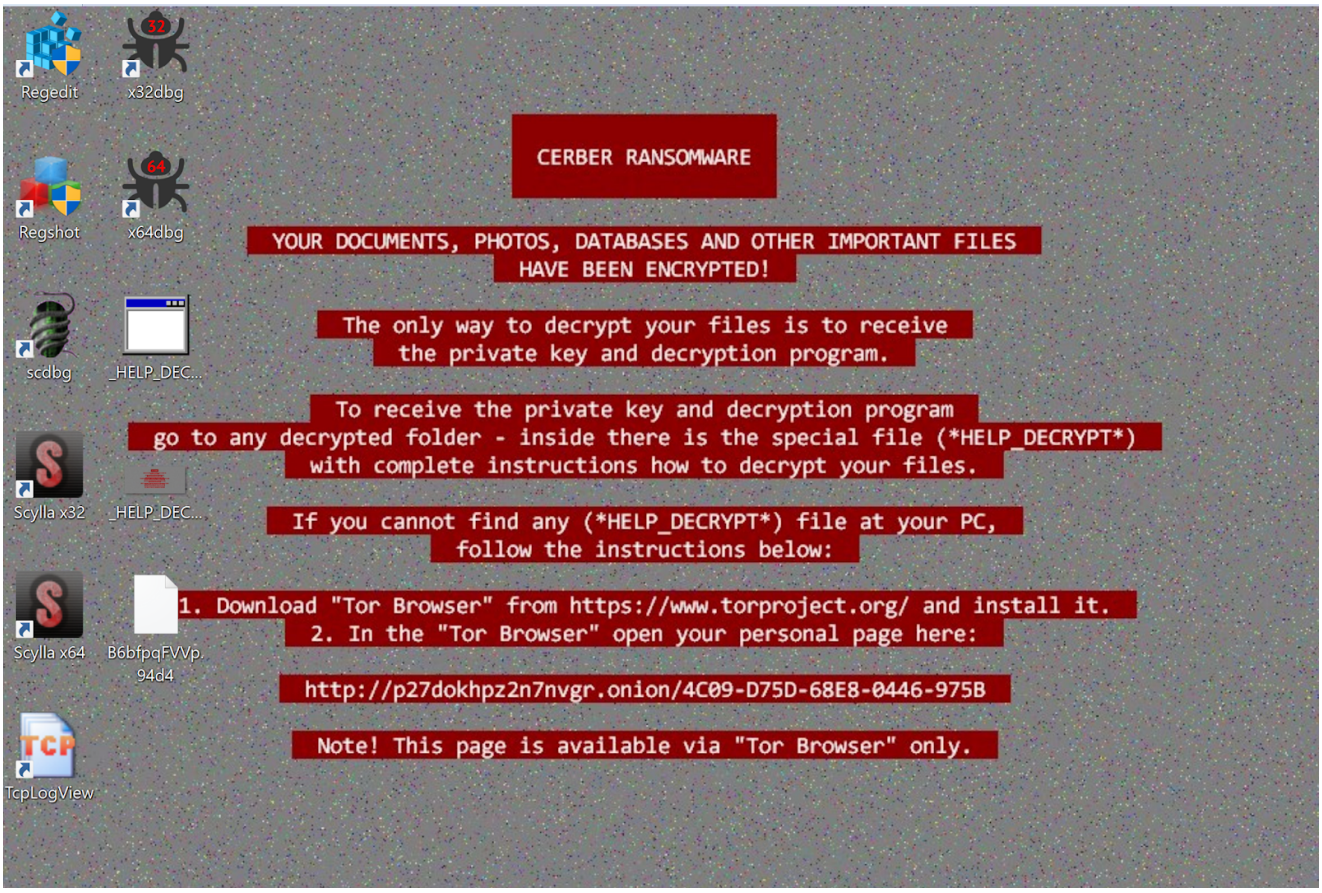
The malware then makes a few modifications to the registry. A couple of these modifications have to do with what the user is presented with (Wallpaper), the rest have to do with network activity. This malware (interestingly enough) does not establish persistence, just encrypts and exits.

Process Name	PID	Operation	Path	Result	Detail
cerber.exe	4436	RegSetValue	HKCU\Software\Classes\Local Settings\MuiCache\a\52C64B7E\LanguageList	SUCCESS	Type: REG_MULTI_SZ, Length: 20, Data: en-US, en
cerber.exe	4436	RegSetValue	HKCU\Control Panel\Desktop\Wallpaper	SUCCESS	Type: REG_SZ, Length: 88, Data: C:\Users\REM\AppData\
cerber.exe	4436	RegSetValue	HKCU\Software\Microsoft\Windows\CurrentVersion\ApplicationAssociationToasts\htafile_.hta	SUCCESS	Type: REG_DWORD, Length: 4, Data: 0
cerber.exe	4436	RegSetValue	HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\FileExts\.hta\OpenWithProgids\htafile	SUCCESS	Type: REG_NONE, Length: 0
cerber.exe	4436	RegSetValue	HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\FileExts\.hta\OpenWithProgids\htafile	SUCCESS	Type: REG_NONE, Length: 0
cerber.exe	4436	RegSetValue	HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap\ProxyBypass	SUCCESS	Type: REG_DWORD, Length: 4, Data: 1
cerber.exe	4436	RegSetValue	HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap\IntranetName	SUCCESS	Type: REG_DWORD, Length: 4, Data: 1
cerber.exe	4436	RegSetValue	HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap\UNCAsIntranet	SUCCESS	Type: REG_DWORD, Length: 4, Data: 1
cerber.exe	4436	RegSetValue	HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap\AutoDetect	SUCCESS	Type: REG_DWORD, Length: 4, Data: 0
cerber.exe	4436	RegSetValue	HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap\ProxyBypass	SUCCESS	Type: REG_DWORD, Length: 4, Data: 1
cerber.exe	4436	RegSetValue	HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap\IntranetName	SUCCESS	Type: REG_DWORD, Length: 4, Data: 1
cerber.exe	4436	RegSetValue	HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap\UNCAsIntranet	SUCCESS	Type: REG_DWORD, Length: 4, Data: 1
cerber.exe	4436	RegSetValue	HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap\AutoDetect	SUCCESS	Type: REG_DWORD, Length: 4, Data: 0
cerber.exe	4436	RegSetValue	HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\FileExts\.jpg\OpenWithProgids\jpegfile	SUCCESS	Type: REG_NONE, Length: 0

The malware spawns an instance of itself, which then opens the ransom note '_HELP_DECRYPT_N0BR8ST0_.hta'. The filename for this ransom note appears to be unique to the system, format is '_HELP_DECRYPT_[A-Z0-9]{8}_.hta' <- simple regex for the 8 digit alphanumeric string.

Process	Description	Image Path	Command
svchost.exe (6380)	Host Process f...	c:\windows\system32\svchost.exe	c:\windows\system32\svchost.exe -k localsystemnetworkrestricted -p -s WdiSystemHost
svchost.exe (5080)	Host Process f...	c:\windows\system32\svchost.exe	c:\windows\system32\svchost.exe -k localsystemnetworkrestricted -p -s PcaSvc
svchost.exe (176)	Host Process f...	c:\windows\system32\svchost.exe	c:\windows\system32\svchost.exe -k netsvcs -p -s Appinfo
consent.exe (3092)	Consent UI for ...	c:\windows\system32\consent.exe	consent.exe 176 312 000001FACFF22990
svchost.exe (7088)	Host Process f...	C:\WINDOWS\system32\svchost.exe	C:\WINDOWS\system32\svchost.exe -k appmodel -p -s tiledatamodelsvc
WmiApSrv.exe (4864)	WMI Performa...	C:\WINDOWS\system32\wbem\WmiApSrv.exe	C:\WINDOWS\system32\wbem\WmiApSrv.exe
svchost.exe (3428)	Host Process f...	C:\WINDOWS\system32\svchost.exe	C:\WINDOWS\system32\svchost.exe -k netsvcs -p -s wisvc
lsass.exe (764)	Local Security ...	C:\WINDOWS\system32\lsass.exe	C:\WINDOWS\system32\lsass.exe
fontdrvhost.exe (864)	Usermode Fon...	C:\WINDOWS\system32\fontdrvhost.exe	"fontdrvhost.exe"
csrss.exe (672)	Client Server ...	C:\WINDOWS\system32\csrss.exe	%SystemRoot%\system32\csrss.exe ObjectDirectory=\Windows SharedSection=1024,20480,768 Win
winlogon.exe (756)	Windows Logo...	C:\WINDOWS\system32\winlogon.exe	winlogon.exe
fontdrvhost.exe (872)	Usermode Fon...	C:\WINDOWS\system32\fontdrvhost.exe	"fontdrvhost.exe"
dwm.exe (848)	Desktop Wind...	C:\WINDOWS\system32\dwm.exe	"dwm.exe"
Explorer.EXE (3936)	Windows Expl...	C:\WINDOWS\Explorer.EXE	C:\WINDOWS\Explorer.EXE
vmtoolsd.exe (5824)	VMware Tools ...	C:\Program Files\VMware\VMware Tools\vmtoolsd.exe	"C:\Program Files\VMware\VMware Tools\vmtoolsd.exe" -n vmusr
Regshot-x64-ANSI.exe (70	Regshot 1.9.0 ...	C:\Program Files\Regshot\Regshot-x64-ANSI.exe	"C:\Program Files\Regshot\Regshot-x64-ANSI.exe"
Procmon.exe (5756)	Process Monitor	C:\Program Files (x86)\Process Monitor\Procmon.exe	"C:\Program Files (x86)\Process Monitor\Procmon.exe"
Procmon64.exe (3300)	Process Monitor	C:\Users\REM\AppData\Local\Temp\Procmon64.exe	"C:\Users\REM\AppData\Local\Temp\Procmon64.exe" /originalpath "C:\Program Files (x86)\Process
cerber.exe (6668)		C:\Users\REM\Desktop\cerber.exe	"C:\Users\REM\Desktop\cerber.exe"
cerber.exe (4436)		C:\Users\REM\Desktop\cerber.exe	"C:\Users\REM\Desktop\cerber.exe"
mshta.exe (6716)	Microsoft (R) ...	C:\Windows\SysWOW64\mshta.exe	"C:\Windows\SysWOW64\mshta.exe" "C:\Users\REM\Desktop_HELP_DECRYPT_N0BR8ST0_.hta".

Like most ransomware, a ransom note is dropped to the desktop, the Wallpaper is changed, and encrypted files are tagged with a weird file extension '.94d4'. Another file extension found during analysis was '.bde6' - this value appears to be randomly generated.

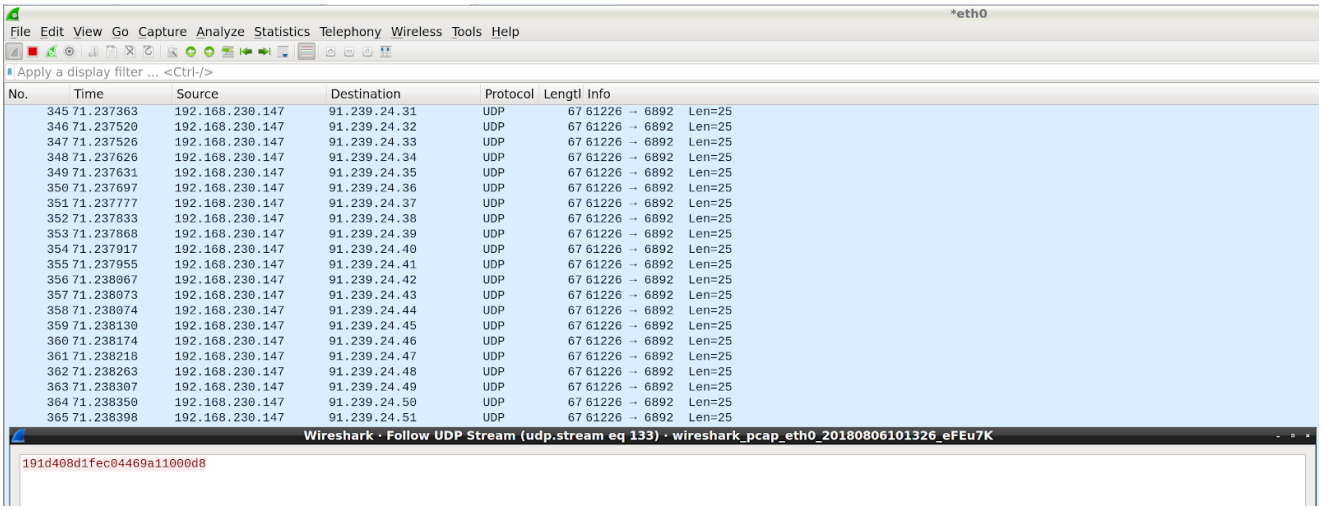


The malware contacts hundreds of hosts over port 6892.

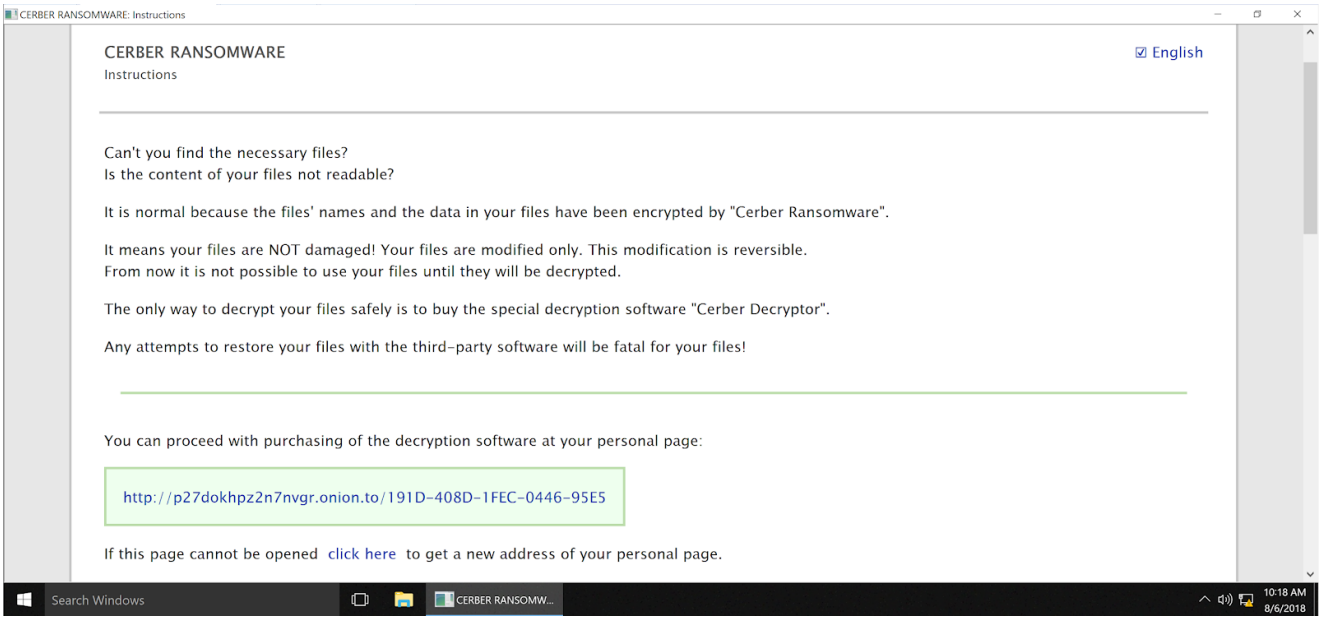
```
Respuesta: 0.12.15.97.in-addr.arpa. -> 192.168.230.146
Respuesta: 1.12.15.97.in-addr.arpa. -> 192.168.230.146
Respuesta: 2.12.15.97.in-addr.arpa. -> 192.168.230.146
Respuesta: 3.12.15.97.in-addr.arpa. -> 192.168.230.146
Respuesta: 4.12.15.97.in-addr.arpa. -> 192.168.230.146
Respuesta: 5.12.15.97.in-addr.arpa. -> 192.168.230.146
Respuesta: 6.12.15.97.in-addr.arpa. -> 192.168.230.146
Respuesta: 7.12.15.97.in-addr.arpa. -> 192.168.230.146
Respuesta: 8.12.15.97.in-addr.arpa. -> 192.168.230.146
Respuesta: 9.12.15.97.in-addr.arpa. -> 192.168.230.146
Respuesta: 10.12.15.97.in-addr.arpa. -> 192.168.230.146
Respuesta: 11.12.15.97.in-addr.arpa. -> 192.168.230.146
Respuesta: 12.12.15.97.in-addr.arpa. -> 192.168.230.146
Respuesta: 13.12.15.97.in-addr.arpa. -> 192.168.230.146
Respuesta: 14.12.15.97.in-addr.arpa. -> 192.168.230.146
Respuesta: 15.12.15.97.in-addr.arpa. -> 192.168.230.146
Respuesta: 16.12.15.97.in-addr.arpa. -> 192.168.230.146
Respuesta: 17.12.15.97.in-addr.arpa. -> 192.168.230.146
Respuesta: 18.12.15.97.in-addr.arpa. -> 192.168.230.146
Respuesta: 19.12.15.97.in-addr.arpa. -> 192.168.230.146
Respuesta: 20.12.15.97.in-addr.arpa. -> 192.168.230.146
Respuesta: 21.12.15.97.in-addr.arpa. -> 192.168.230.146
Respuesta: 22.12.15.97.in-addr.arpa. -> 192.168.230.146
Respuesta: 23.12.15.97.in-addr.arpa. -> 192.168.230.146
Respuesta: 24.12.15.97.in-addr.arpa. -> 192.168.230.146
Respuesta: 25.12.15.97.in-addr.arpa. -> 192.168.230.146
Respuesta: 26.12.15.97.in-addr.arpa. -> 192.168.230.146
Respuesta: 27.12.15.97.in-addr.arpa. -> 192.168.230.146
Respuesta: 28.12.15.97.in-addr.arpa. -> 192.168.230.146
Respuesta: 29.12.15.97.in-addr.arpa. -> 192.168.230.146
Respuesta: 30.12.15.97.in-addr.arpa. -> 192.168.230.146
Respuesta: 31.12.15.97.in-addr.arpa. -> 192.168.230.146
```

Source	Destination	Protocol	Lengtl	Info
192.168.230.146	192.168.230.147	DNS	100	Standard query response 0xd577 PTR 1.24.239.91.in-addr.arpa A 192.168.230.146
192.168.230.146	192.168.230.147	DNS	100	Standard query response 0x05ab PTR 2.24.239.91.in-addr.arpa A 192.168.230.146
192.168.230.146	192.168.230.147	DNS	100	Standard query response 0xbbdb PTR 3.24.239.91.in-addr.arpa A 192.168.230.146
192.168.230.146	192.168.230.147	DNS	100	Standard query response 0xe8de PTR 4.24.239.91.in-addr.arpa A 192.168.230.146
192.168.230.146	192.168.230.147	DNS	100	Standard query response 0xcc16 PTR 5.24.239.91.in-addr.arpa A 192.168.230.146
192.168.230.146	192.168.230.147	DNS	100	Standard query response 0x2ff2 PTR 6.24.239.91.in-addr.arpa A 192.168.230.146
192.168.230.146	192.168.230.147	DNS	100	Standard query response 0xa56d PTR 7.24.239.91.in-addr.arpa A 192.168.230.146
192.168.230.146	192.168.230.147	DNS	100	Standard query response 0xb92f PTR 8.24.239.91.in-addr.arpa A 192.168.230.146
192.168.230.146	192.168.230.147	DNS	100	Standard query response 0xdc63 PTR 9.24.239.91.in-addr.arpa A 192.168.230.146
192.168.230.146	192.168.230.147	DNS	101	Standard query response 0xa1b1 PTR 10.24.239.91.in-addr.arpa A 192.168.230.146
192.168.230.146	192.168.230.147	DNS	101	Standard query response 0xa8a9 PTR 11.24.239.91.in-addr.arpa A 192.168.230.146
192.168.230.146	192.168.230.147	DNS	101	Standard query response 0x3e18 PTR 12.24.239.91.in-addr.arpa A 192.168.230.146
192.168.230.146	192.168.230.147	DNS	101	Standard query response 0x3550 PTR 13.24.239.91.in-addr.arpa A 192.168.230.146
192.168.230.146	192.168.230.147	DNS	101	Standard query response 0x604f PTR 14.24.239.91.in-addr.arpa A 192.168.230.146
192.168.230.146	192.168.230.147	DNS	101	Standard query response 0x2773 PTR 15.24.239.91.in-addr.arpa A 192.168.230.146
192.168.230.146	192.168.230.147	DNS	101	Standard query response 0x2299 PTR 16.24.239.91.in-addr.arpa A 192.168.230.146
192.168.230.146	192.168.230.147	DNS	101	Standard query response 0x353b PTR 17.24.239.91.in-addr.arpa A 192.168.230.146
192.168.230.146	192.168.230.147	DNS	101	Standard query response 0xb3b4 PTR 18.24.239.91.in-addr.arpa A 192.168.230.146
192.168.230.146	192.168.230.147	DNS	101	Standard query response 0x7ff3 PTR 19.24.239.91.in-addr.arpa A 192.168.230.146
192.168.230.146	192.168.230.147	DNS	101	Standard query response 0x4cdb PTR 20.24.239.91.in-addr.arpa A 192.168.230.146
192.168.230.146	192.168.230.147	DNS	101	Standard query response 0xbf3a PTR 21.24.239.91.in-addr.arpa A 192.168.230.146
192.168.230.146	192.168.230.147	DNS	101	Standard query response 0xf297 PTR 22.24.239.91.in-addr.arpa A 192.168.230.146
192.168.230.146	192.168.230.147	DNS	101	Standard query response 0xfffd PTR 23.24.239.91.in-addr.arpa A 192.168.230.146
192.168.230.146	192.168.230.147	DNS	101	Standard query response 0x77b0 PTR 24.24.239.91.in-addr.arpa A 192.168.230.146
192.168.230.146	192.168.230.147	DNS	101	Standard query response 0x5435 PTR 25.24.239.91.in-addr.arpa A 192.168.230.146
192.168.230.146	192.168.230.147	DNS	101	Standard query response 0x3387 PTR 26.24.239.91.in-addr.arpa A 192.168.230.146
192.168.230.146	192.168.230.147	DNS	101	Standard query response 0x6d96 PTR 27.24.239.91.in-addr.arpa A 192.168.230.146

The same UDP packet is sent over and over.

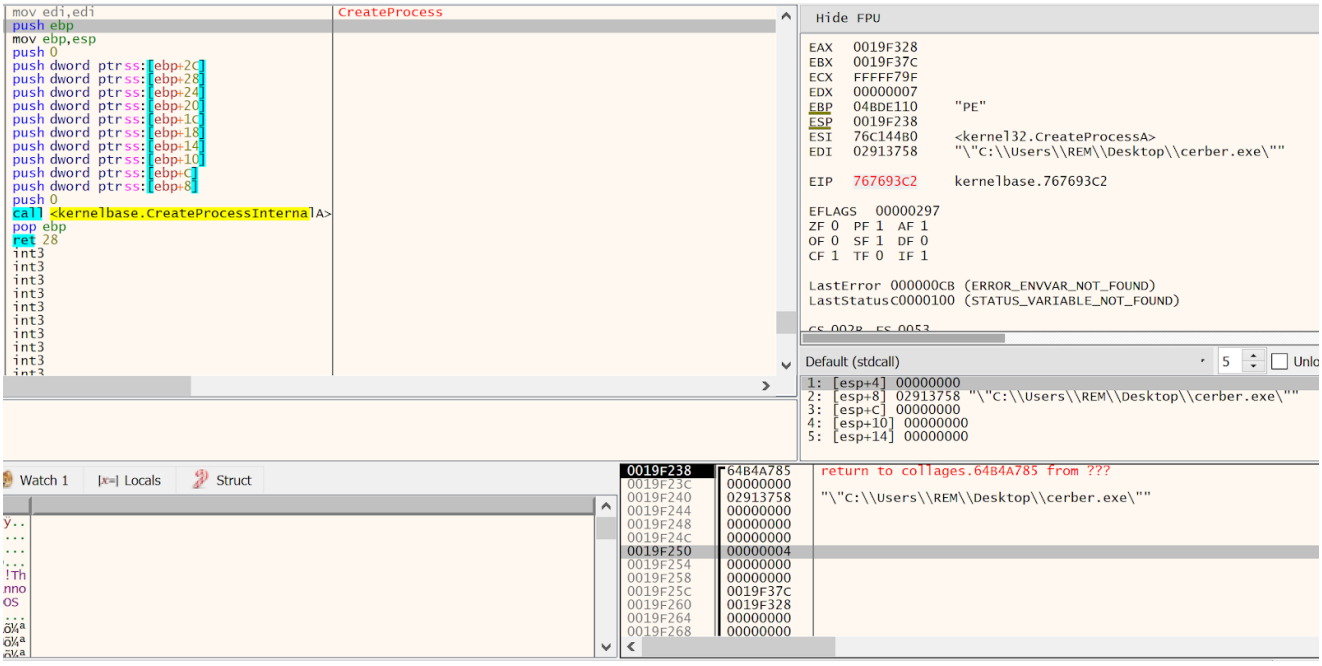


The first half of the string is the same as the unique string at the end of the provided URL in the ransom note.

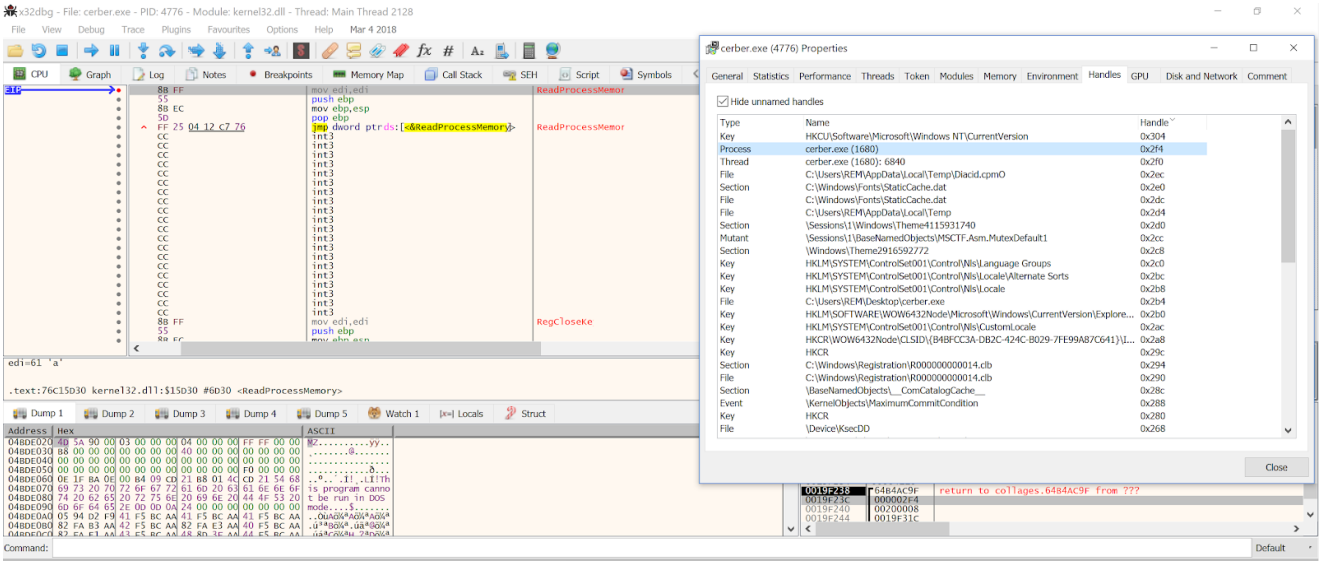


LET THE REVERSING COMMENCE!!!

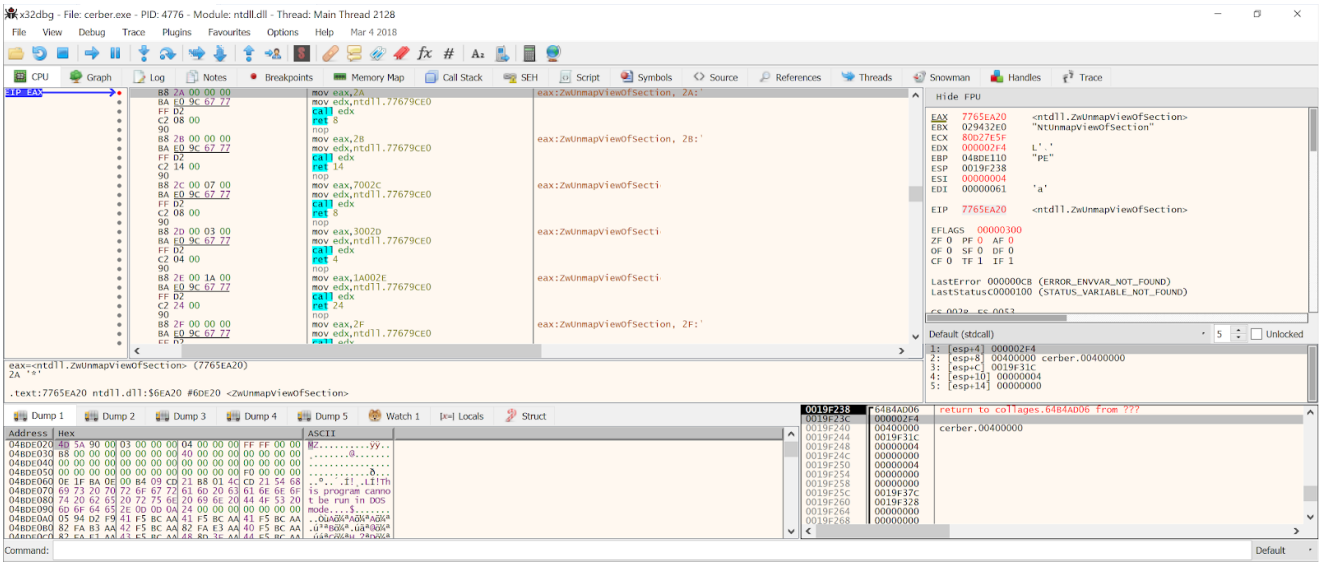
The process tree from behavioral analysis showed the original instance of cerber launching a new cerber, this turned out to be pretty interesting. Notice the sixth value pushed onto the stack for the CreateProcess API Call, the value 4 is passed for the dwCreationFlags argument. The 4 indicates that this process is created in a suspended state. Do I sense code injection?



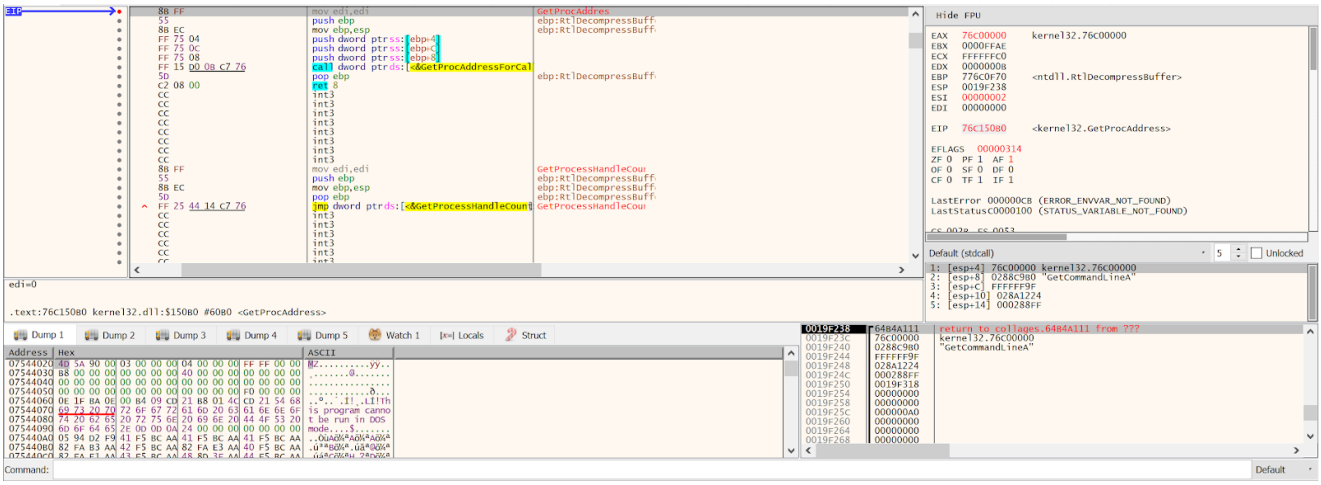
The malware then takes a fairly common path for the injection. Reads memory of the newly spawned cerber.exe, the parameter '2F4' is a handle to said cerber.



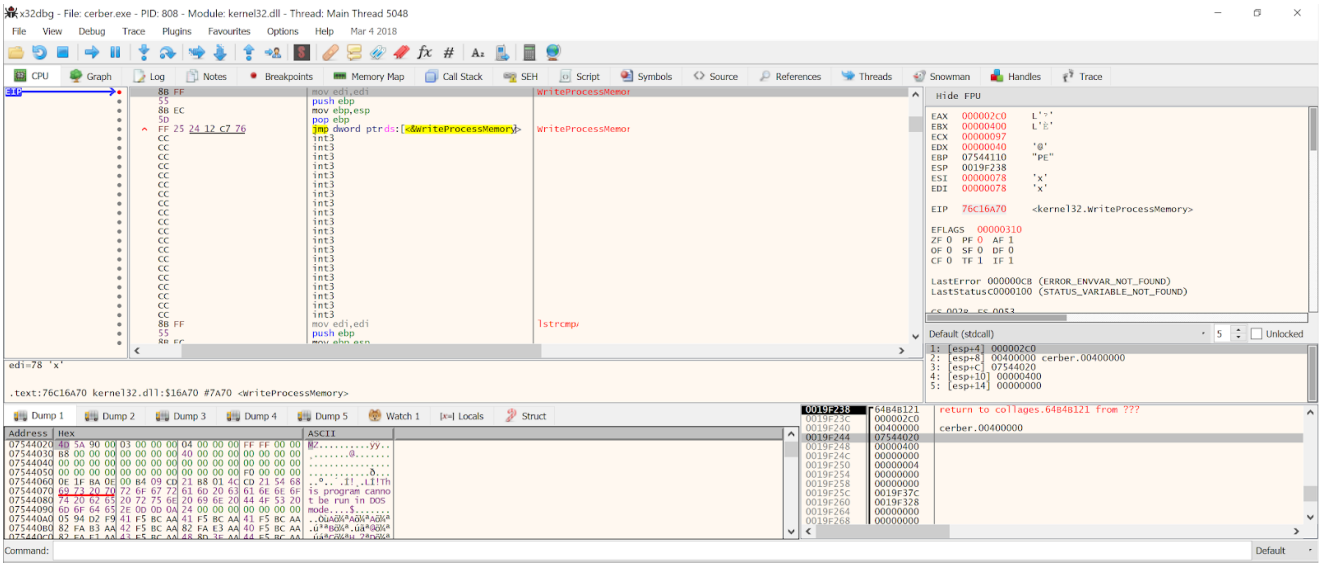
The malware then hollows out the suspended process through the WinAPI call 'UnMapViewOfSection'. The parameter '400000' is passed as the base address as to where begin hollowing/unmapping, which is the very beginning/start of the PE.



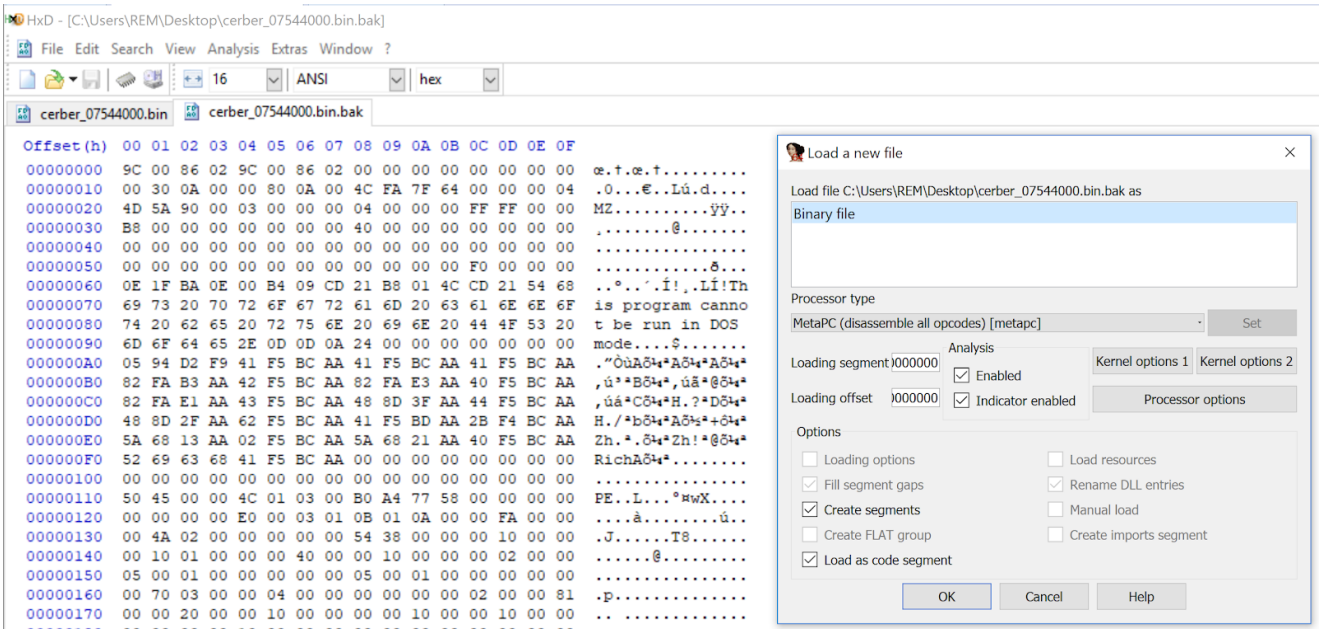
A buffer is then filled via 'RtlDecompressBuffer', which stores the contents that will be injected into the suspended process. Notice the 'MZ' header, this an executable that will be injected into the target process.



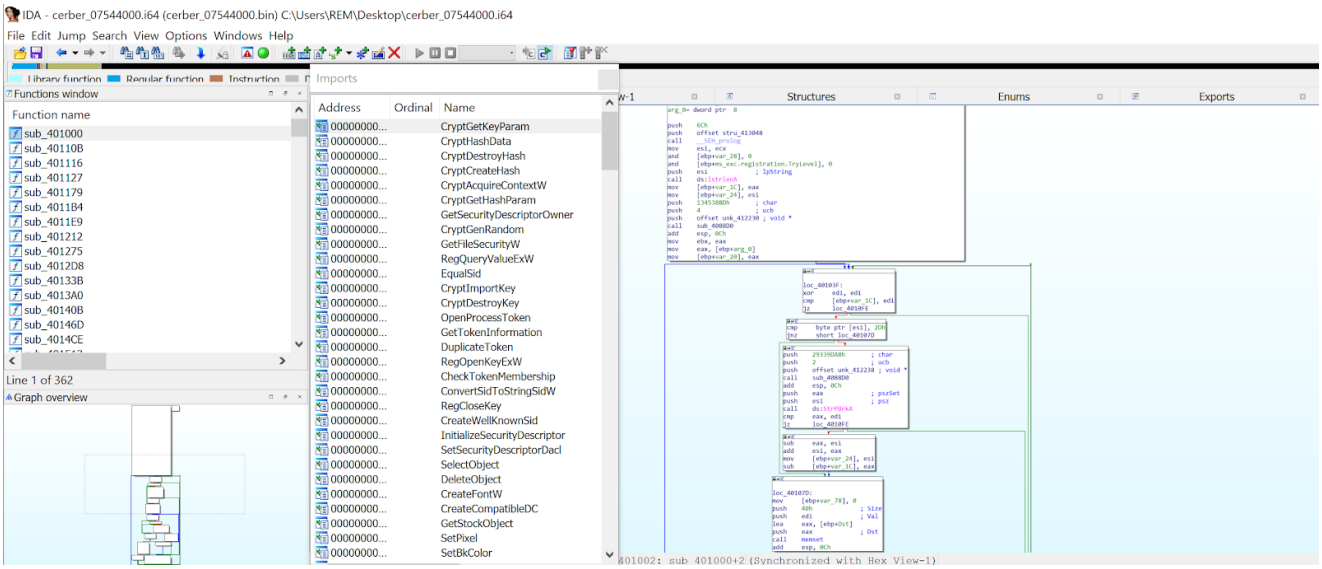
The buffer is then written to the target process via 'WriteProcessMemory'. A pointer to the executable seen in the dump window is passed as the data to be written. The base address '400000', which was the start address of the hollowing, is now passed as the base address for this executable to be written to in the hollowed out process.



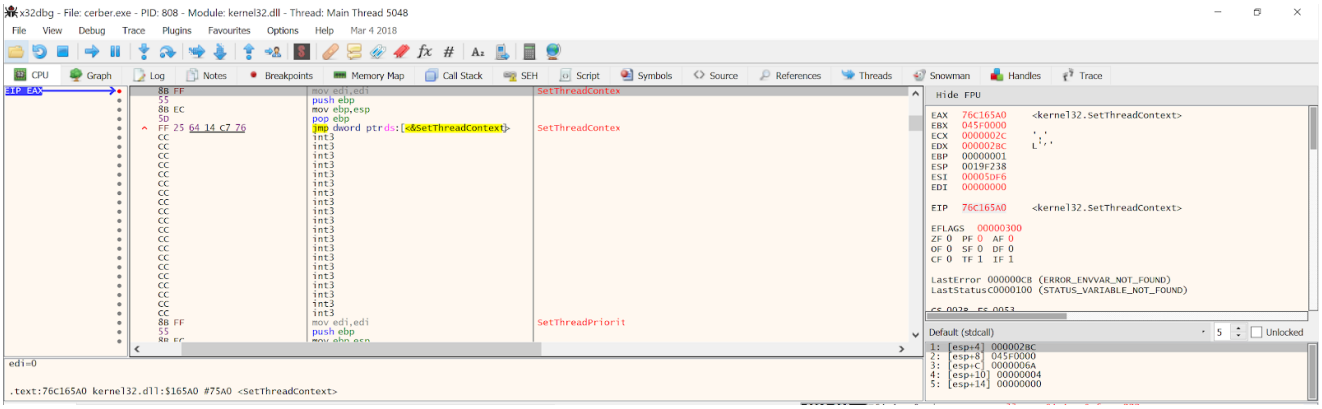
To intercept this executable, I followed the base address in the memory map and then dumped it. When attempting to load it into IDA, it is not recognized as a valid PE. Looking at the file in HxD, there are around 32 bytes of noise before the magic bytes of the executable.



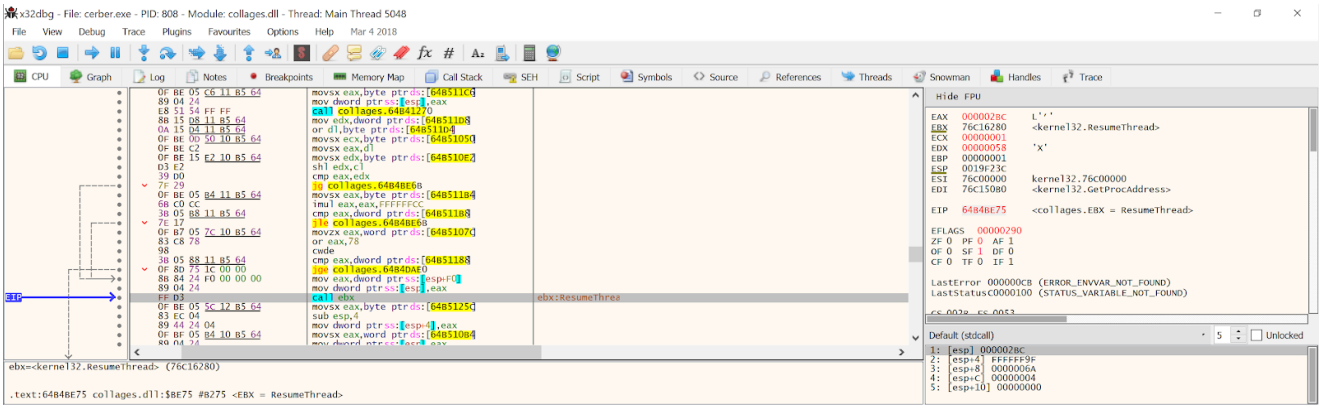
Deleting up until the 'MZ'/'4D5A' fixes the problem. This looks to be where the true payload/ransomware code lies, this is the first time we have seen crypt-related APIs. So essentially, the malware uses code injection as a way to unpack itself.



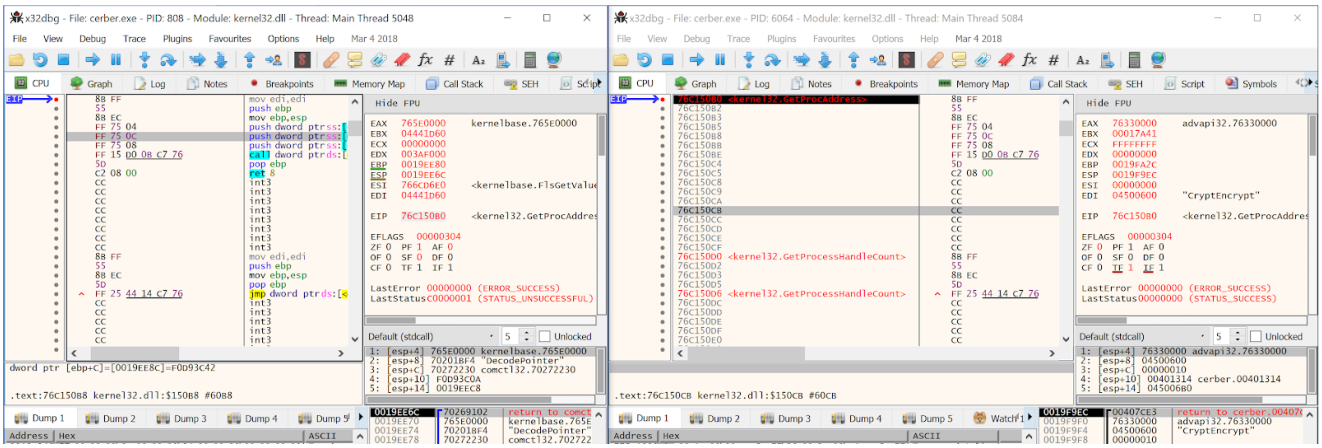
Now that the code has been injected/written, the malware must start a thread of execution to invoke the code. The context of the thread is set via 'SetThreadContext'.



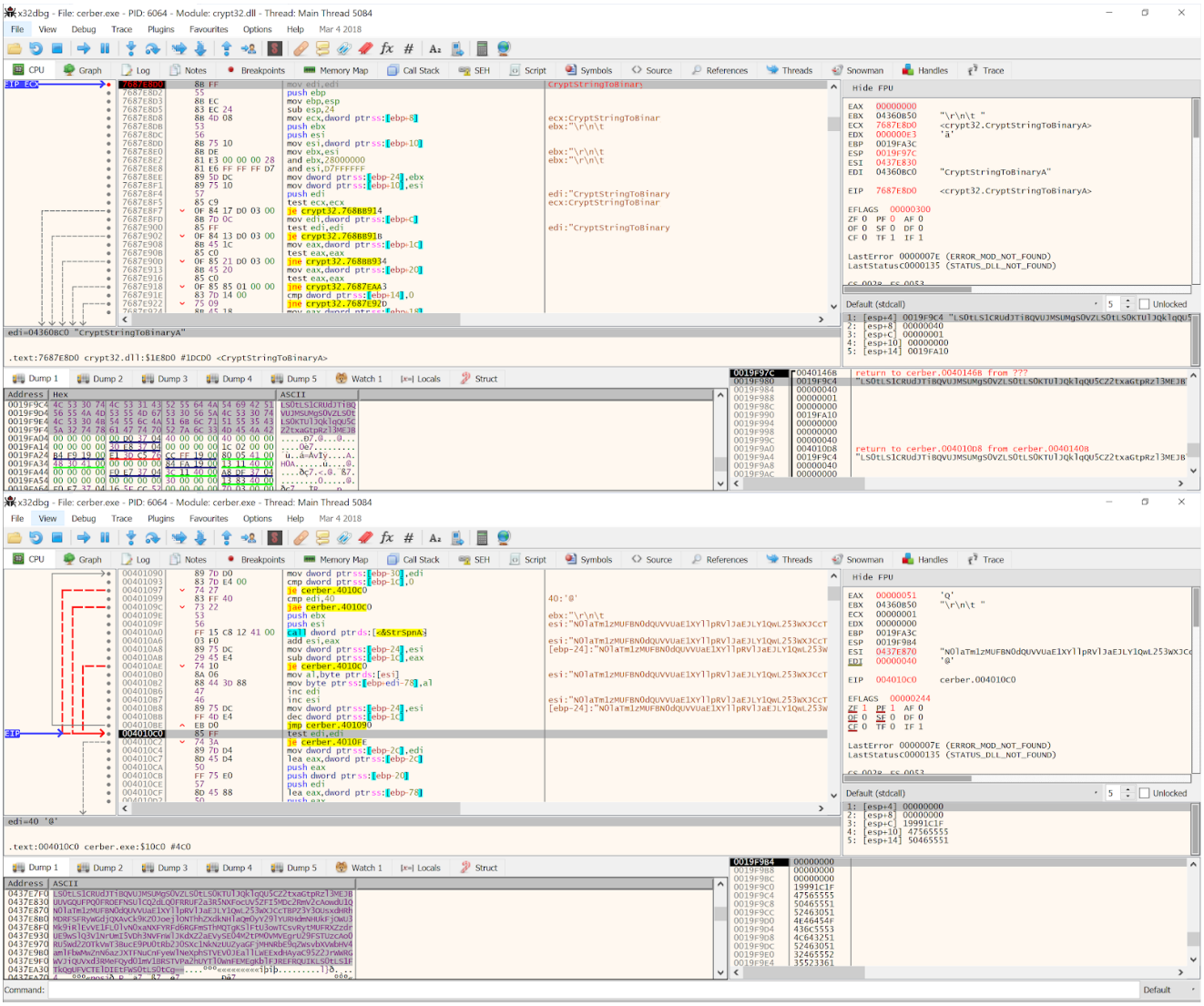
And finally, the thread is resumed and the code begins executing in the target process.



Just before taking the instruction to allow 'ResumeThread' to execute, I spawned a new instance of x64dbg and attached to the still suspended cerber.exe. I then set breakpoints on thread entry and thread start to halt execution once 'ResumeThread' is called. Next, I set break points on all crypt-related APIs, as well as GetProcAddress, so that I can identify any additional code that may be dynamically loaded. The first function to be dynamically resolved via 'GetProcAddress' is 'CryptEncrypt'.



The next interesting code block was the usage of the API 'CryptStringToBinary'. This API converts a string to an array of bytes. The data to be converted is a very long base64 string.

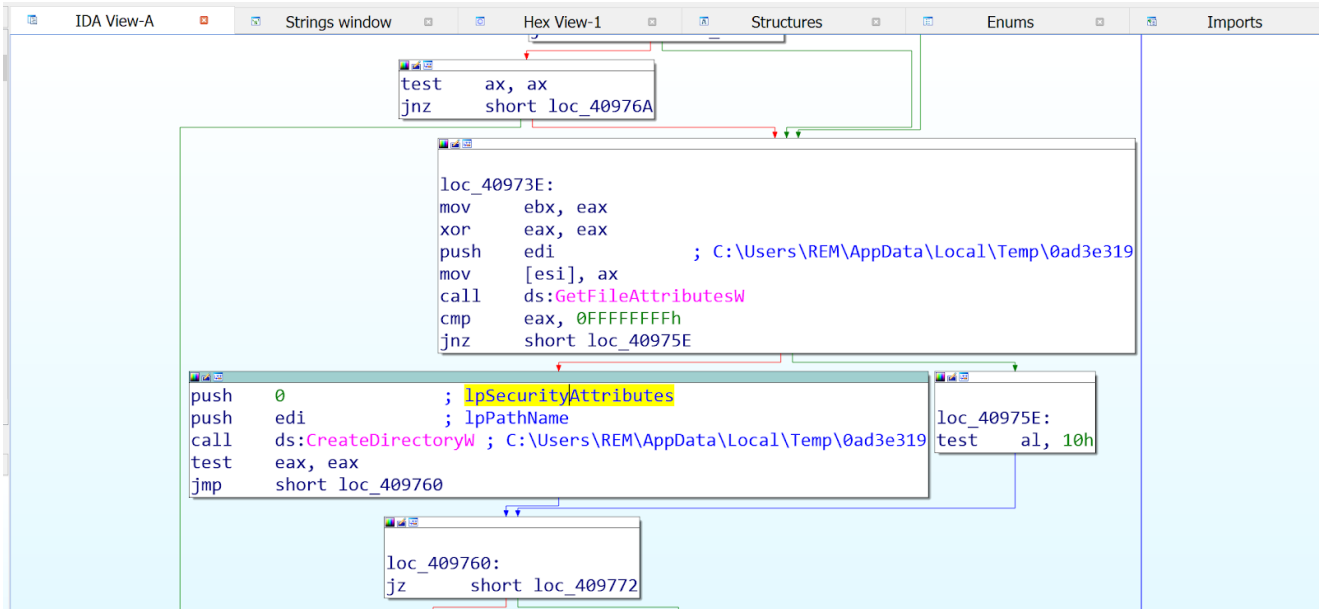


Decoded, the string looks to be a hard-coded Public Key.

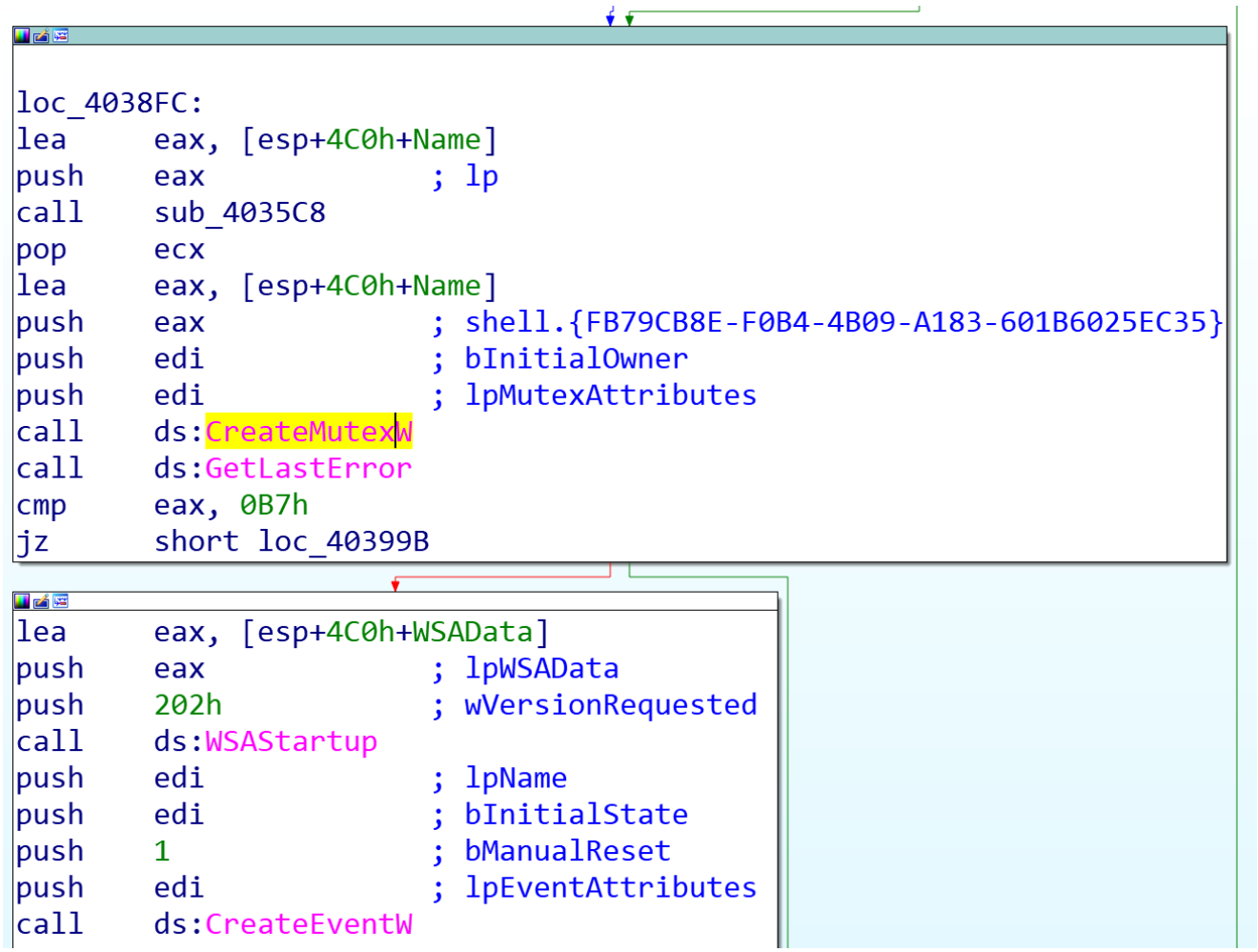
```
remnux@remnux:~/Desktop$ echo "LS0tLS1CRUdJTiBQVUJMSUMgS0VZLS0tLS0KTUlJQklqQU5C
Z2txaGtpRzl3MEJBUUVGQUFPQ0FROEFNSUlCQ2dLQ0FRRUF2a3R5NXFocUV5ZFI5MDc2RmV2cAowdU1
QN0laTm1zMUFBN0dQUVVUaE1XYllpRVlJaEJLY1QwL253WXJCcTBPZ3Y3OUsxdHRhMDRFSFRyWGdjQX
AvCk9KZ0JoejlONThhZXdkNHlaQm0yY29lYURHdmNHUkFjOWU3Mk9iRlEvVE1FL0lvN0xaNXFYRFd6R
GFmSThMQTgKSlFtU3owTCsvRytMUFRXZzdrUE9wSlQ3V1NrUmI5VDh3NVFnWlJKdXZ2aEVySE04M2tP
M0VMVEgrU29FSTUzcAo0RU5Wd2ZOTkVwT3BucE9PU0tRb2J0SXc1NkNzUUZyaGFjMHNRbE9qZWsvbXV
WbHV4amlFbWMwZnN6azJXTFNuCnFyeWlNeXphSTVEV0JEallLWEExdHAyaC95Z2JrWWRGWVJiQUVxd3
RMeFQyd01mV1BRSTVPa2hUYTl0WnFEMEgKblFJREFRQUIKLS0tLS1FTkQgUFVCTElDIEtFWS0tLS0tC
g==" | base64 --decode
-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAvkty5qhqEydR9076Fevp
0uMP7IZNms1AA7GPQUThMWbYiEYIhBKcT0/nwYrBq0Ogv79K1tta04EHTrXgcAp/
OJgBhz9N58aewd4yZBm2coeaDGvcGRAc9e72ObFQ/TME/Io7LZ5qXDWzDafI8LA8
JQmSz0L+/G+LPTWg7kPOpJT7WSkRb9T8w5QgZRJuvvhErHM83kO3ELTH+SoEI53p
4ENVwfNNEpOpnpOOSKQobtIw56CsQFrhac0sQlOjek/muVluxjiEmc0fszk2WLSn
qryiMyzaI5DWBDjYKXA1tp2h/ygbkYdFYRbAEqwtLxT2wMfWPQI5OkhTa9tZqD0H
nQIDAQAB
-----END PUBLIC KEY-----
```

The Public Key Info suggests that this is 'RSA 1.2.840.113549.1.1 - PKCS-1' encryption.

Next, the malware creates a directory in the AppData folder where it stores some housekeeping data.



Then a mutex is created. The name of the mutex is resolved dynamically ---- 'shell. {FB79CB8E-F0B4-4B09-A183-601B6025EC35}' and is created just before network activity occurs ('WSAStartup').



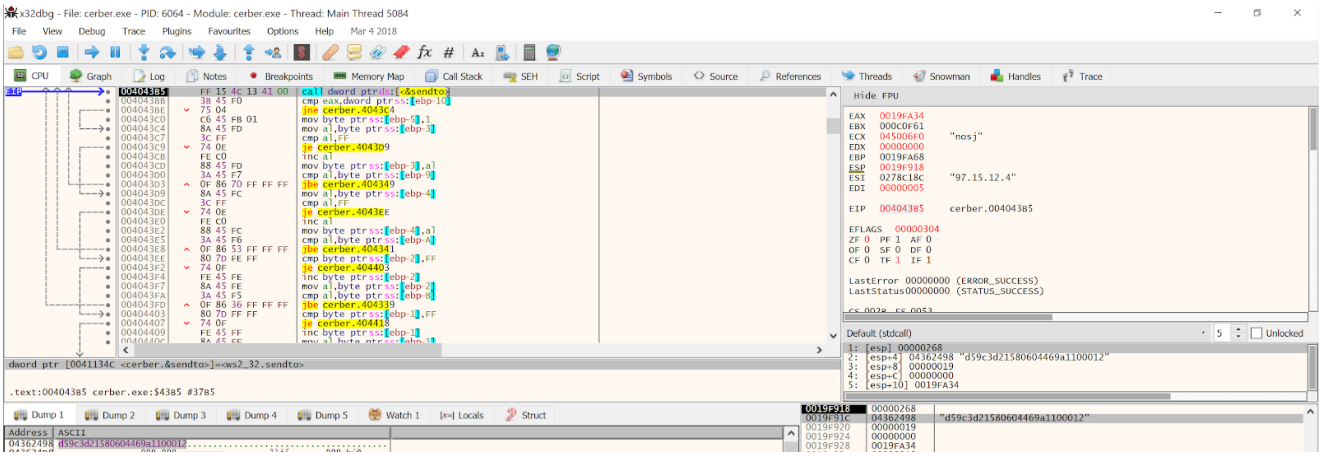
```
loc_4038FC:
lea     eax, [esp+4C0h+Name]
push    eax             ; lp
call    sub_4035C8
pop     ecx
lea     eax, [esp+4C0h+Name]
push    eax             ; shell.{FB79CB8E-F0B4-4B09-A183-601B6025EC35}
push    edi             ; bInitialOwner
push    edi             ; lpMutexAttributes
call    ds:CreateMutexW
call    ds:GetLastError
cmp     eax, 0B7h
jz      short loc_40399B
```

```
lea     eax, [esp+4C0h+WSAData]
push    eax             ; lpWSAData
push    202h            ; wVersionRequested
call    ds:WSAStartup
push    edi             ; lpName
push    edi             ; bInitialState
push    1               ; bManualReset
push    edi             ; lpEventAttributes
call    ds:CreateEventW
```

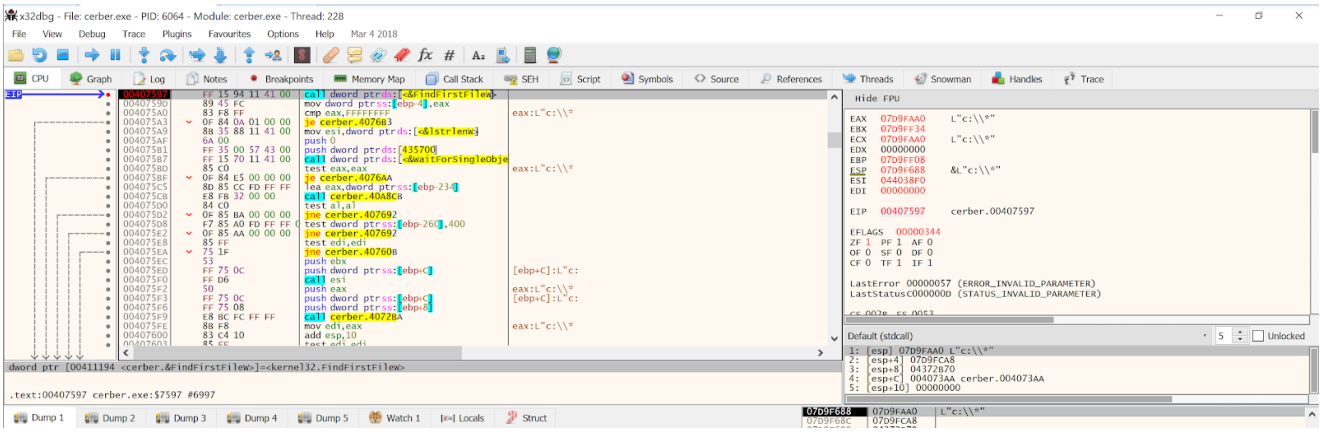
A UDP socket is created and will be used to blast that single string to hundreds of IPs.

```
.text:00404258                  mov     eax, esi
.text:0040425A                  push    ecx             ; hostshort
.text:0040425B                  mov     [ebp+to.sa_family], ax
.text:0040425F                  call    ds:htons
.text:00404265                  push    17              ; IPPROTO_UDP
.text:00404267                  push    esi             ; SOCK_DGRAM
.text:00404268                  push    esi             ; AF_INET // IPv4
.text:00404269                  mov     word ptr [ebp+to.sa_data], ax
.text:0040426D                  call    ds:socket
.text:00404273                  mov     [ebp+s], eax
.text:00404276                  cmp     eax, 0FFFFFFFFh
.text:00404279                  jz      loc_404430
```

'sendto' function is included in a loop to contact the external hosts.



Next, the encryption commences. Traversing directories via 'FindFirstFile' and 'FindNextFile'. The malware also looks for network resources to encrypt via 'WNetOpenEnum'.



Once encryption completes, the ransom note is opened via 'ShellExecute' with parameters 'Open' and '_HELP_DECRYPT_N0BR8ST0_.hta'. This ransom note is far more robust than most. Most ransom notes are a simple text file, this is an .hta that has several functions, and is apparently quite universally accommodating. The ransom note even has a button to change the language.

```
<body>
    <div class="container">
        <div class="header">
            <a id="change_language" href="#" onclick="return changeLanguage1();" title="English">&#9745; English</a>
            <h1>C&#069;&#82;BE&#82; &#82;ANSOMWA&#82;&#069;</h1>
            <small id="title">Instructions</small>
        </div>
        <div id="languages">
            <p>&#9745; Select your language</p>
            <ul>
                <li><a href="#" title="English" onclick="return showBlock('en');">English</a></li>
                <li><a href="#" title="Arabic" onclick="return showBlock('ar');">العربية</a></li>
                <li><a href="#" title="Chinese" onclick="return showBlock('zh');">中文</a></li>
                <li><a href="#" title="Dutch" onclick="return showBlock('nl');">Nederlands</a></li>
                <li><a href="#" title="French" onclick="return showBlock('fr');">Français</a></li>
                <li><a href="#" title="German" onclick="return showBlock('de');">Deutsch</a></li>
                <li><a href="#" title="Italian" onclick="return showBlock('it');">Italiano</a></li>
                <li><a href="#" title="Japanese" onclick="return showBlock('ja');">日本語</a></li>
                <li><a href="#" title="Korean" onclick="return showBlock('ko');">한국어</a></li>
                <li><a href="#" title="Polish" onclick="return showBlock('pl');">Polski</a></li>
                <li><a href="#" title="Portuguese" onclick="return showBlock('pt');">Português</a></li>
                <li><a href="#" title="Spanish" onclick="return showBlock('es');">Español</a></li>
                <li><a href="#" title="Turkish" onclick="return showBlock('tr');">Türkçe</a></li>
            </ul>
        </div>
```

Another interesting code segment is that the .hta file checks the victim's MAC address against a few MAC addresses associated with VMware and some popular network technology companies. If there is a match, the URL in the ransom note is updated to English. Interesting!

```
    showBlock(lang);
    try {
        var macAddress = "";
        var myEnum = new Enumerator(GetObject("winmgmts:{impersonationLevel=impersonate}").ExecQuery("SELECT * FROM Win32_NetworkAdapterConfiguration WHERE IPEnabled = True"));
        for (; !myEnum.atEnd() ; myEnum.moveNext()) {
            var item = myEnum.item();
            macAddress = item.MACAddress.substr(0, 8);
        }
        if ("00:50:56, 00:0C:29, 00:1C:14, 00:05:69, 0A:00:27, 00:03:FF, 00:1C:42, 00:0F:4B, 00:16:3E, 08:00:27".indexOf(macAddress) < 0) {
            setTimeout(function() {
                updateUrl("en");
            }, 100);
        }
    }
```

The most interesting part of this malware to me was how it unpacked itself through code injection and process hollowing. This malware just performs the encryption and then exits, no persistence! Most filenames are randomized to evade signature based detection, so regex will be our friends when sweeping for/detecting these artifacts. A Snort rule may be plausible due to the port (6892), but UDP can be noisy. The rule would use PCRE to match the unique long string passed over 6892.

Key Takeaways:
- Encrypts files on disk and in network shares
- Modifies registry
- Drops files on disk
- Performs code injection
- Contacts external hosts

Host-based IOCs:
cerber.exe
2d6ace7910f84eb775272a6590453a0e - md5
\AppData\Local\Temp\collages.dll
2A4BF3D01B6C84A2130C110D02C772AC - md5

\AppData\Local\Temp\floppy_disk.png
\AppData\Local\Temp\floppy_disk_disabled.png
\AppData\Local\Temp\flat.xsl
\AppData\Local\Temp\tmpCBD8.bmp
\AppData\Local\Temp\0ad3e319\4f11.tmp
\AppData\Local\Temp\0ad3e319\280c.tmp
\AppData\Local\Temp\nshBD76.tmp\System.dll
3E6BF00B3AC976122F982AE2AADB1C51 - md5
\Desktop_HELP_DECRYPT_N0BR8ST0_.hta - Ransom note

\Desktop_HELP_DECRYPT_N0BR8ST0_.jpg - Wallpaper

*.94d4 - file extension tagged onto encrypted files (randomly generated)
*.bde6 - file extension tagged onto encrypted files (randomly generated)
*.[a-z0-9]{4} - regex for file extension
shell.{FB79CB8E-F0B4-4B09-A183-601B6025EC35} - Mutex
\Sessions\1\BaseNamedObjects\SM0:6064:168:WilStaging_02 - Based named object
HKEY_CURRENT_USER\Control Panel\Desktop\WallPaper -REG_SZ -
C:\Users\REM\AppData\Local\Temp\tmpCBD8.bmp

Hard-coded Public Key:
-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAvkty5qhqEydR9076Fevp
0uMP7IZNms1AA7GPQUThMWbYiEYIhBKcT0/nwYrBq0Ogv79K1tta04EHTrXgcAp/
OJgBhz9N58aewd4yZBm2coeaDGvcGRAc9e72ObFQ/TME/Io7LZ5qXDWzDafI8LA8
JQmSz0L+/G+LPTWg7kPOpJT7WSkRb9T8w5QgZRJuvvhErHM83kO3ELTH+SoEI53p
4ENVwfNNEpOpnpOOSKQobtIw56CsQFrhac0sQlOjek/muVluxjiEmc0fszk2WLSn
qryiMyzaI5DWBDjYKXA1tp2h/ygbkYdFYRbAEqwtLxT2wMfWPQI5OkhTa9tZqD0H
nQIDAQAB
-----END PUBLIC KEY-----

Network-based IOCs:
97.15.12.xxx:6892 - UDP
91.239.24.xxx: 6892 - UDP
xxx.12.15.97: 6892 - UDP

xxx.24.239.91: 6892 - UDP