Cerber ransomware: new, but mature

blog.malwarebytes.com/threat-analysis/2016/03/cerber-ransomware-new-but-mature/

hasherezade March 11, 2016

Ransomware authors seem to love mythological creatures. We have seen Chimera, now we will take a look at Cerber. Both are named after powerful beasts and both are prepared in a professional way. As SenseCy states (source), Cerber is sold to distributors on underground Russian forums.

This malware is often distributed via Exploit Kits (read more here).

UPDATE: Checkpoint released a decryption tool working for some cases of Cerber

Analyzed samples

- f5146a3bbe6c71e5a0ef2f04f955b1a1
- 2f7059d7b1dda3080e391d99788fff18
  payload: 9a7f87c91bf7e602055a5503e80e2313 <- main focus of this analysis

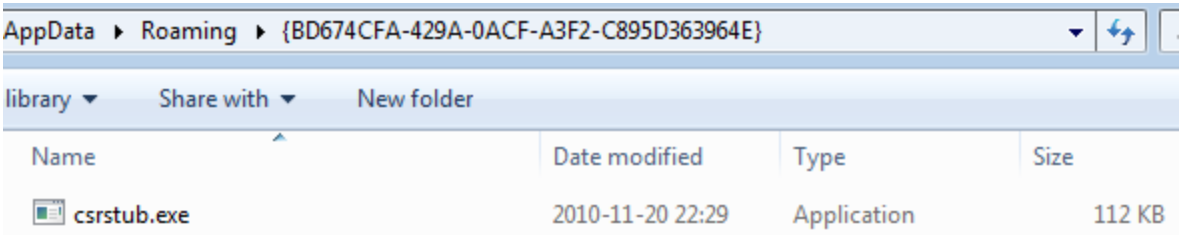
Behavioral analysis

After being deployed it disappears and runs its dropped copy (renamed to [a random word].exe from the hidden folder created in **%APPDATA%**. Name of the folder is specific to a particular sample – in the analyzed one it is: **{BD674CFA-429A-0ACF-A3F2-**

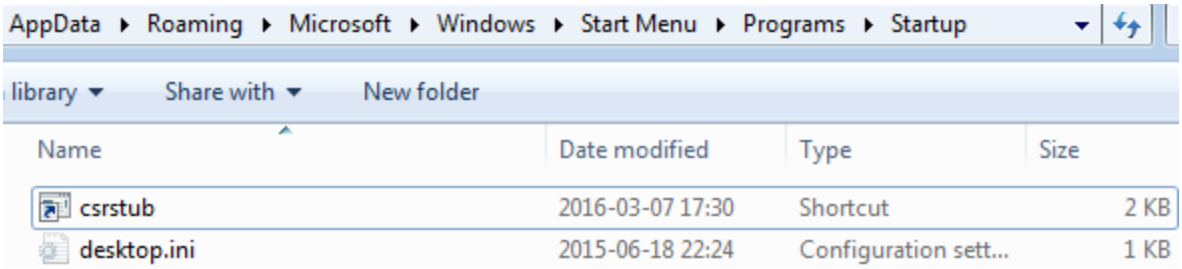
C895D363964E}.

Some observed file names: *csrstub.exe*, *dinotify.exe, ndadmin.exe, setx.exe, rasdial.exe, RelPost.exe, ntkrnlpa.exe*

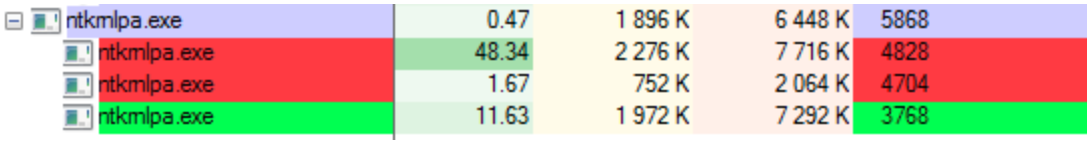
The dropped file has an edited creation timestamp.



It also creates a link to the dropped malware in: *%APPDATA%/Microsoft/Windows/Start Menu/Programs/Startup*:



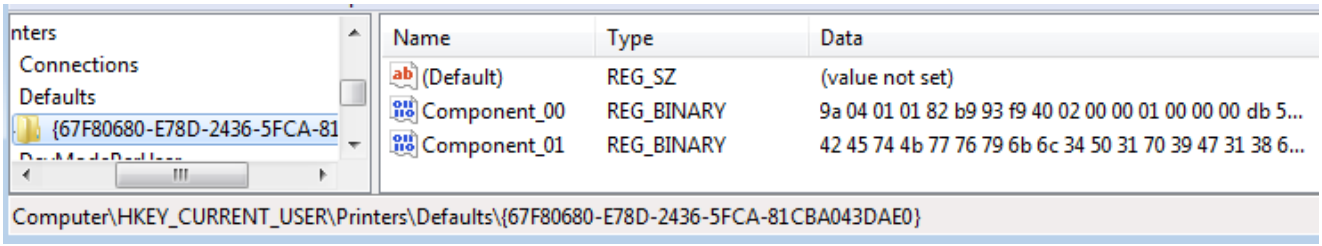
Looking via Process Explorer we can see the dropped sample deploying new instances (it is used in order to divide the work of encrypting files).



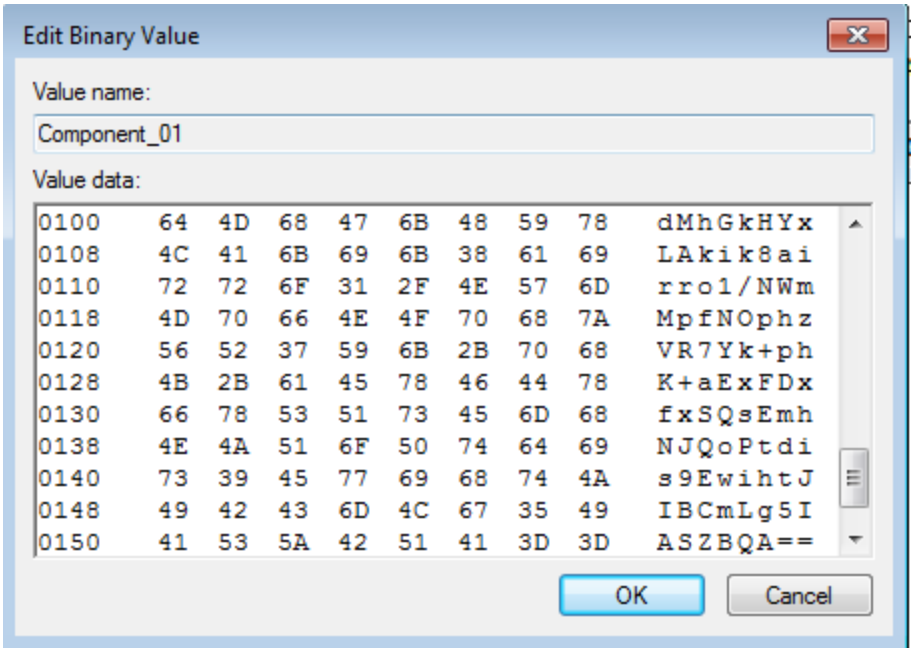
Registry keys

The malware makes changes in the Windows registry.

Two entries (Component_00, Component_01) are dropped in Printers\Defaults:



Compont_01 contains some binary data in base64:



Registry keys for the persistance are added in various places, i.e:

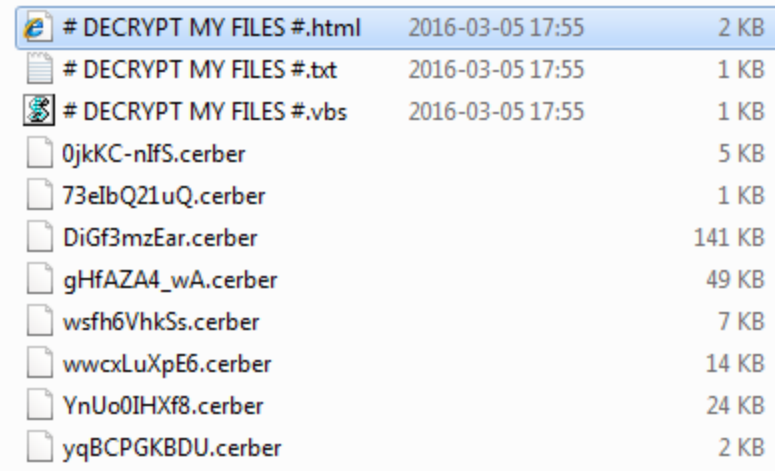
HKEY_USERS -> [current user's SID]:

- "Software\\Microsoft\\Windows\\CurrentVersion\\Run"
- "Software\\Microsoft\\Windows\\CurrentVersion\\RunOnce"
- "Software\\Microsoft\\Windows\\CurrentVersion\\Policies\\Explorer" -> "Run"
- "Software\\Microsoft\\Command Processor" -> "AutoRun"

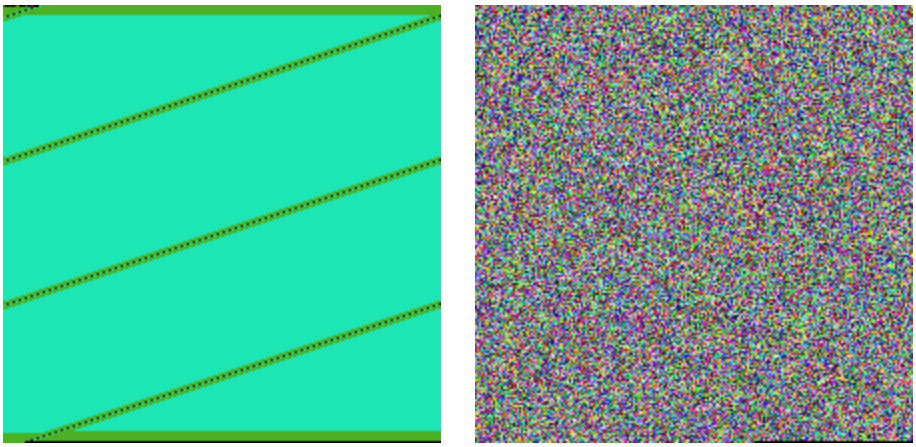
However, when the encryption finishes successfully, the dropped sample is deleted.

Encryption process

Cerber can encrypt files in offline mode – it means it doesn't need to fetch the key from the CnC server. Files that have been encrypted are fully renamed and appended with the extension typical for this ransomware: **.cerber**. Pattern of the name: **[0-9a-zA-Z_-]{10}.cerber**

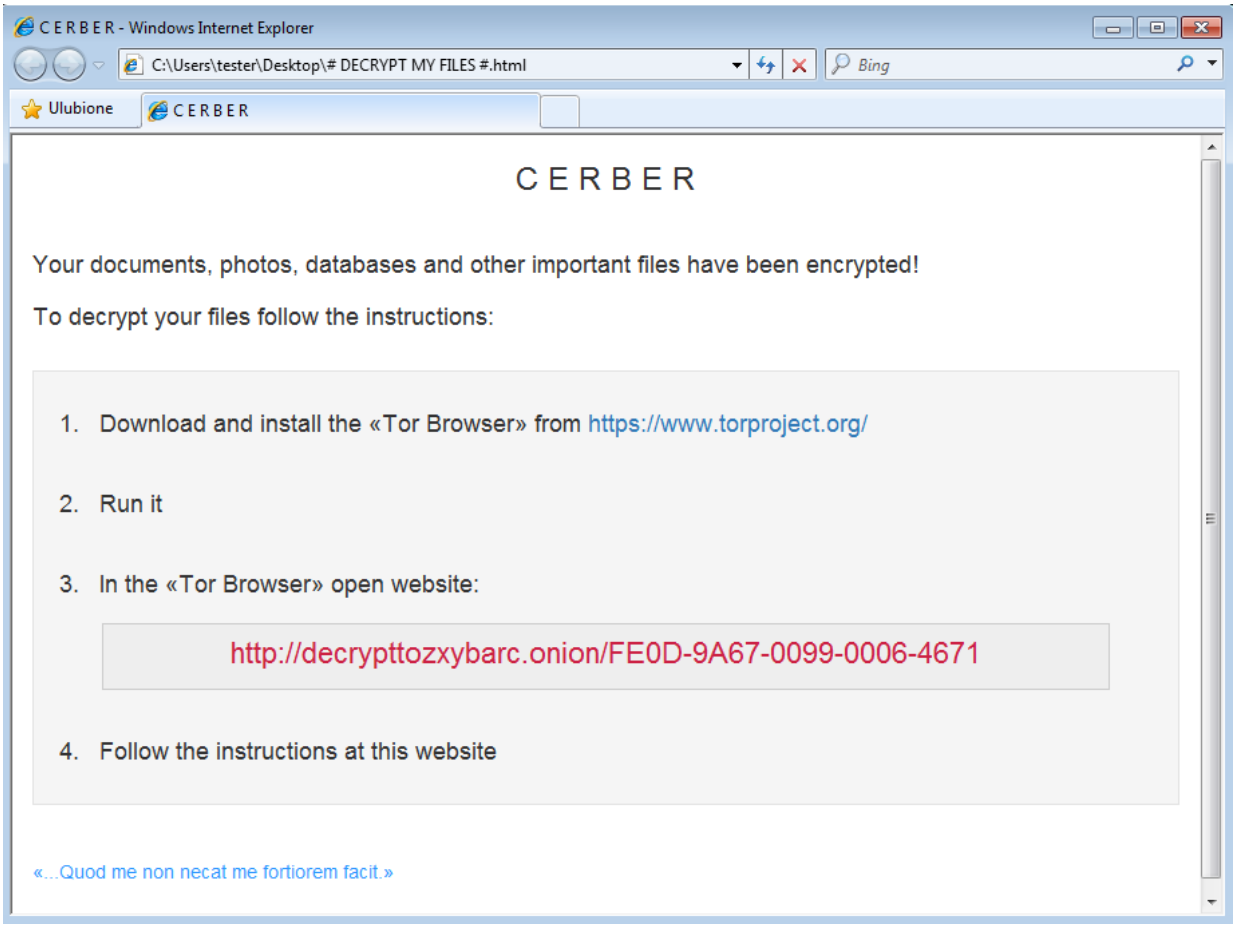


The encrypted content has a high level of entropy and no patterns are visible. Below: visualization of bytes of ***square.bmp*** : left – original, right encrypted with Cerber:



Content of the encrypted file is different on every encryption – probably keys are dynamically generated. After encryption size of the file content is increased about 384 bytes* – it may suggest, that the RSA encrypted AES key is appended to the file (*depending on the file this value may vary a bit, probably because of various padding).

After executing it displays a ransom note in two forms: HTML and TXT. The note is available only in English. Example below:




```
C E R B E R

Your documents, photos, databases and other important files have been encrypted!
```

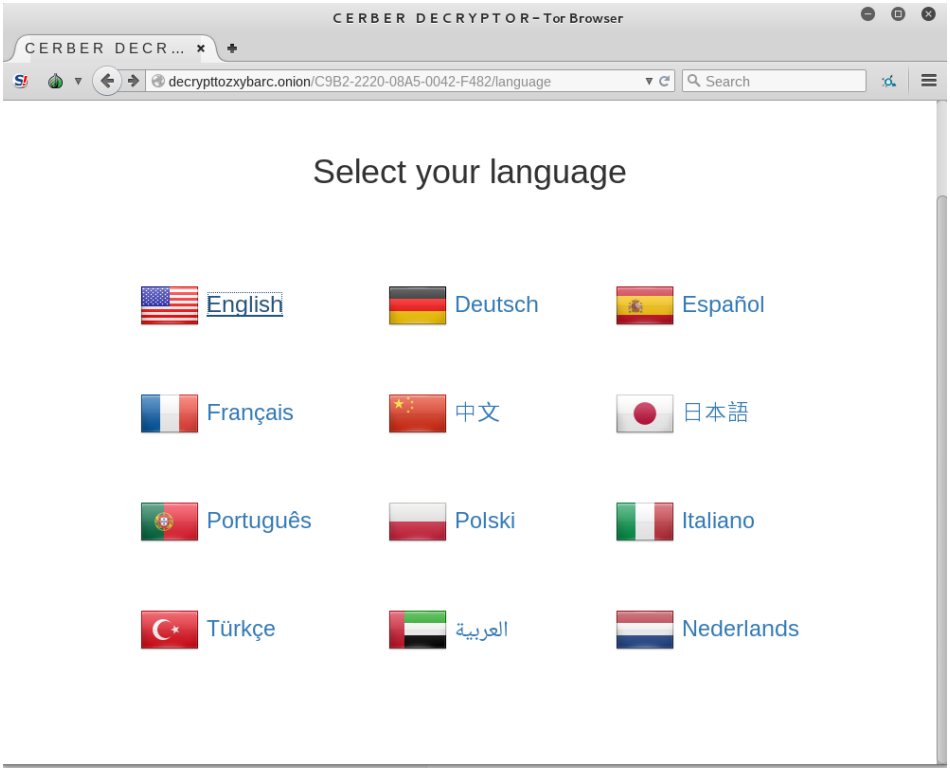
At the bottom of the ransom note attackers added a quote in Latin: «…Quod me non necat me fortiorem facit.» (“What doesn’t kill me, makes me stronger”). We can only speculate what they wanted to convey – to share their own motto, or to console the victim of the attack?

It comes also with a VB macro that is supposed to speak up the message with the help of a local text-to-speech emulator:

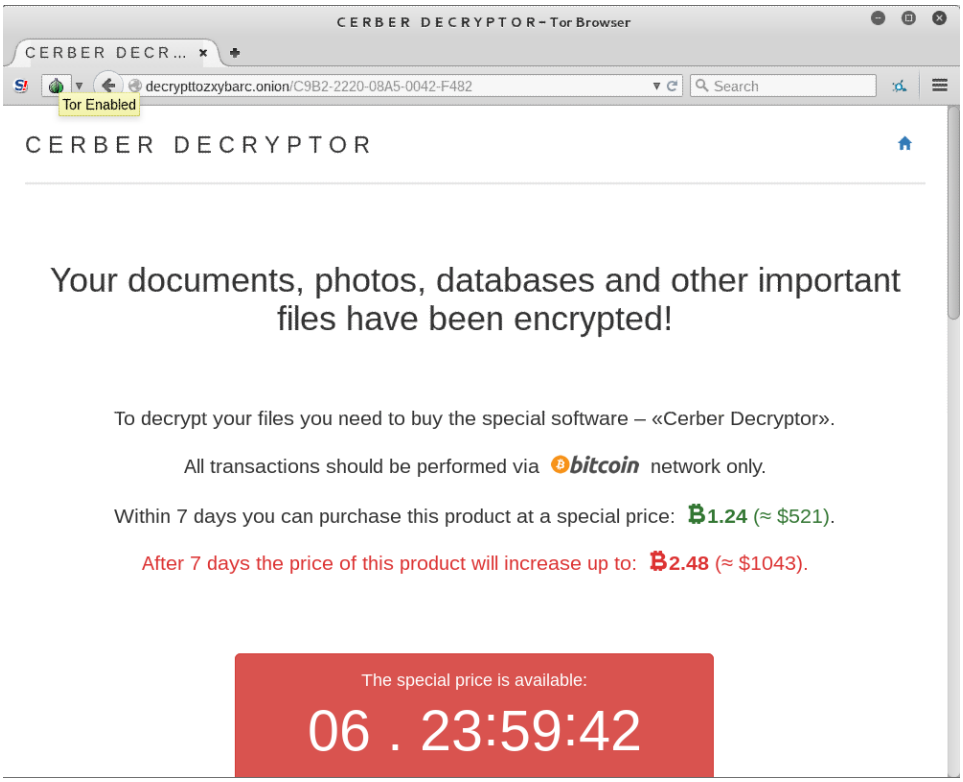
```
Set SAPI = CreateObject("SAPI.SpVoice")
SAPI.Speak "Attention! Attention! Attention!"
For i = 1 to 5
SAPI.Speak "Your documents, photos, databases and other important files have been
encrypted!"
Next
```

Website for the victim

Each victim has a Web page that can be accessed via Tor. Although the ransom note is available only in English, the Tor website can be customized to several languages:



These pages contain further instructions to the victim and support for managing payments. The time to an increase in the ransom price is counted from the first access to this website.




```
To decrypt your files you need to buy the special software - <<Cerber Decryptor>>.
```

Network communication

Cerber can manage well without CnC and accomplish its task offline. However, if given opportunity, it can communicate with CnC in order to send statistics from encryption process.

First, it fetches geolocation info (in JSON format) of the local computer by querying a genuine service: http:/ipinfo.io/json

Then, we can observe sending UDP requests to a predefined range of IP addresses:

Destination	Protocol	Length	Info
87.98.128.0	UDP	68	Source port: 33919 Destination port: 6891
87.98.128.1	UDP	68	Source port: 33919 Destination port: 6891
87.98.128.2	UDP	68	Source port: 33919 Destination port: 6891
87.98.128.3	UDP	68	Source port: 33919 Destination port: 6891
87.98.128.4	UDP	68	Source port: 33919 Destination port: 6891
87.98.128.5	UDP	68	Source port: 33919 Destination port: 6891
87.98.128.6	UDP	68	Source port: 33919 Destination port: 6891
87.98.128.7	UDP	68	Source port: 33919 Destination port: 6891
87.98.128.8	UDP	68	Source port: 33919 Destination port: 6891
87.98.128.9	UDP	68	Source port: 33919 Destination port: 6891
87.98.128.10	UDP	68	Source port: 33919 Destination port: 6891
87.98.128.11	UDP	68	Source port: 33919 Destination port: 6891
87.98.128.12	UDP	68	Source port: 33919 Destination port: 6891
87.98.128.13	UDP	68	Source port: 33919 Destination port: 6891

Inside

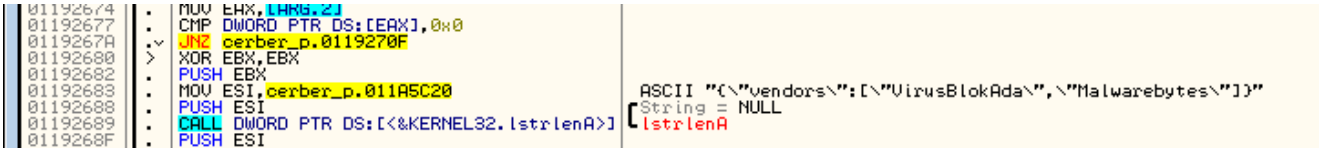
Cerber samples come packed by some crypters/FUDs, so the code is not readable at first. Even when we unpack the core (i.e. 9a7f87c91bf7e602055a5503e80e2313), only a few strings are readable. It is caused by the fact that the authors decided to encrypt the strings and decrypt them just before the usage. Example:

```
00346BED  r$  PUSH EBP
00346BEE  .   MOV EBP,ESP
00346BF0  .   PUSH ECX                          cerber_p.0034B21E
00346BF1  .   PUSH ECX                          cerber_p.0034B21E
00346BF2  .   PUSH EBX
00346BF3  .   PUSH ESI
00346BF4  .   PUSH EDI
00346BF5  .   PUSH 0x1
00346BF7  .   PUSH 0xECB03F2D
00346BFC  .   PUSH 0x6
00346BFE  .   PUSH cerber_p.003565AC
00346C03  .   CALL cerber_p.0034A12A            decrypt_string
00346C08  .   PUSH EAX                          L"CERBER"

EAX=0155B0E0, (UNICODE "CERBER")

Address   Hex dump                                        ASCII
003565AC  75 D1 A7 D1 0E E5 00 00 81 51 AB FB B9 1A DD 47  u╤з╤♫е..ü Q½ы╣→▌G
003565BC  E8 33 1D 19 BE E2 9E 2C 18 50 A5 EB 2A 6E 2F CB  Ф3↔↓Ґ╜ѓ,↑P№л*n/╦
003565CC  9F A2 61 0C 5D 41 F2 E5 72 CB 00 00 EC B2 1F 6B  ƒóa♀]A≥х r╦..м▓▼k
003565DC  74 B7 36 41 B1 31 D7 CF C7 44 E4 EF 9A 1D 04 E2  t╖6A▒1╪╧╟Dфпљ↔♦Г
003565EC  EA EB 75 CA D4 AA 51 07 4D A8 CA 66 00 00 00 00  клuЙ╘к Q•M¨╩f....
```

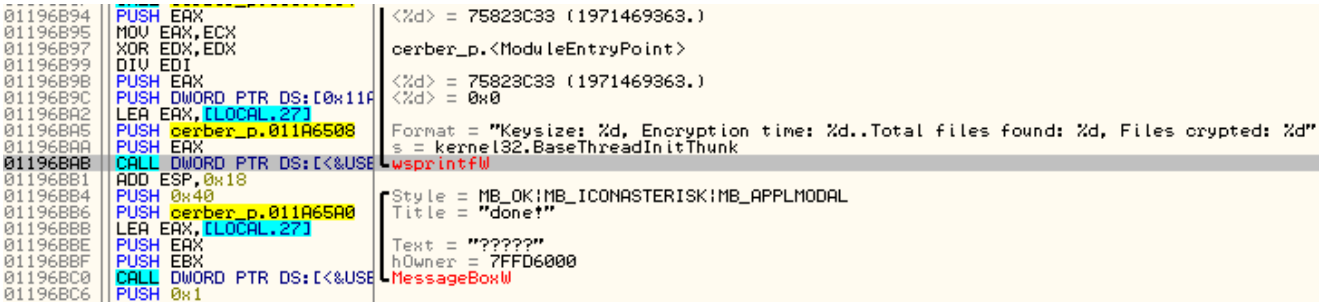
The decrypting function takes the following parameters:

```
decrypt_string(char* input_buffer, DWORD input_lenght, DWORD key, BOOL is_unicode)
```

One of the few strings that hasn't been encrypted was a check against anti-malware vendors (one of them is Malwarebytes). The list of vendors is in JSON – this format have been used extensively by Cerber.

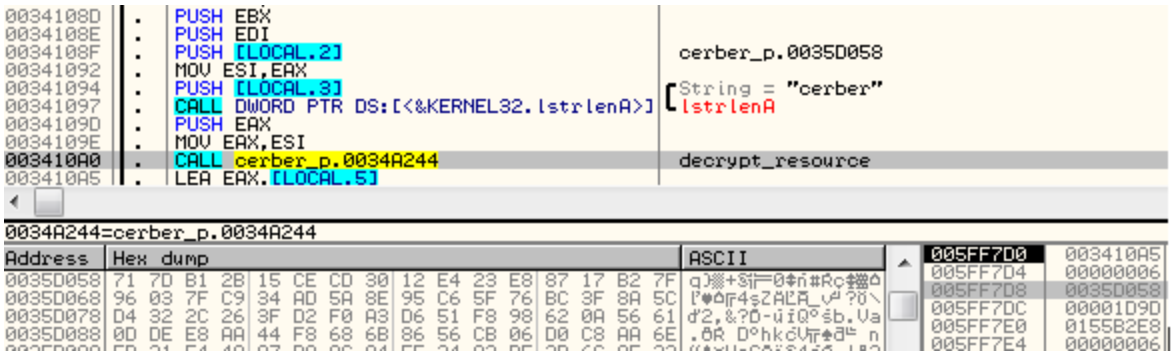


Another interesting unencrypted string was a log, showing the statistics from encryption (the feature used if the malware is deployed in the debug mode):

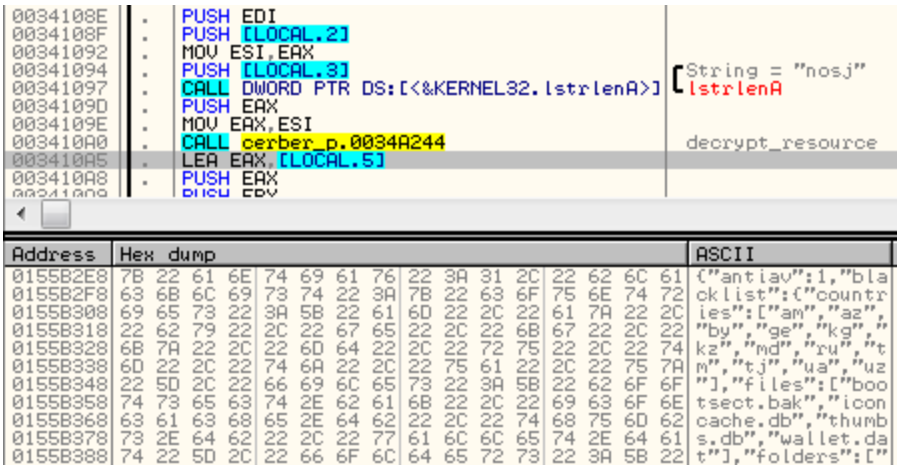


Configuration file

Cerber comes with an encrypted resource, stored as *RC Data*. It is decrypted by a dedicated function:



After decryption, it turns out to be a configuration in JSON format (you can see it full here):



Configuration is rich in options. Contains i.e:

- a blacklist used to exclude some countries, languages, file names and directories from the attack
- a list of attacked extensions
- environment checks that are enabled
- whether or not to deploy the sample in a debug mode
- encryption settings and output extension
- public RSA key in base64 (decoded).
- files with ransom note to be dropped
- list of services used to obtain geolocation
- range of IPs where to send statistics *(compare with IPs described in the section 'Network communication')*
- format of statistics to be sent

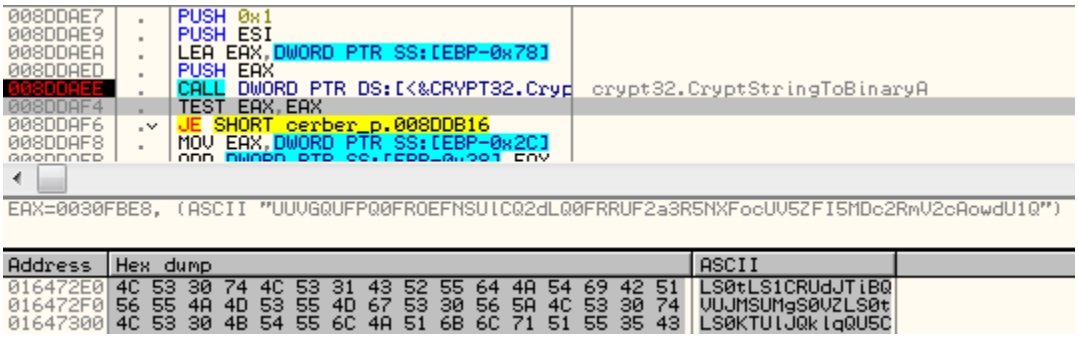
Distributors can customize many things with the help of the config file. Changing the full look-and-feel of the malware – attacked extensions, ransom note and even extension of encrypted files – can make it appear like a new product. This flexibility made me wonder if the same package is not being distributed in a different campaign – not as a Cerber, but under some other name.

The distributor of the analyzed sample decided to exclude several countries form the attack (Armenia, Azerbaijan, Belarus, Georgia, Kyrgyzstan, Kazakhstan, Moldova, Russia, Turkmenistan, Tajikistan, Ukraine, Uzbekistan). It will also spare your default Windows directories, Tor browser and Bitcoin wallet.

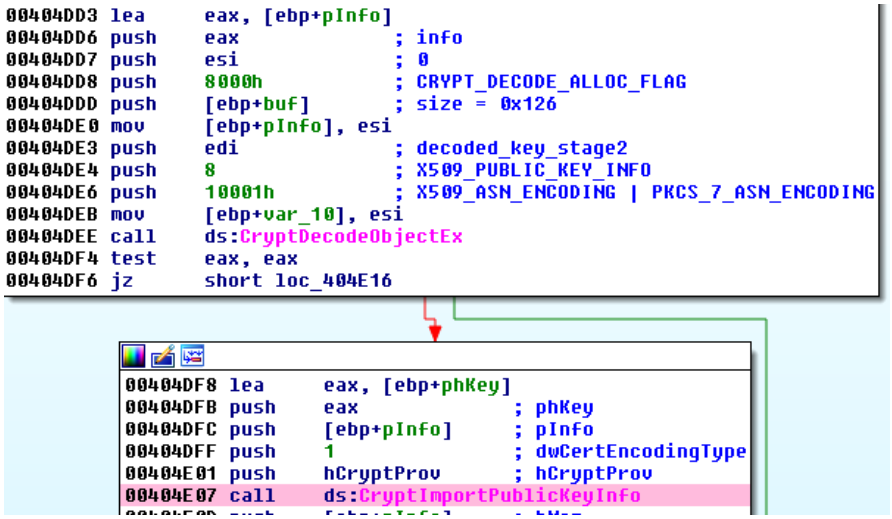
Loading the key

The sample comes with a public RSA key shipped in the configuration file (described in the previous section).

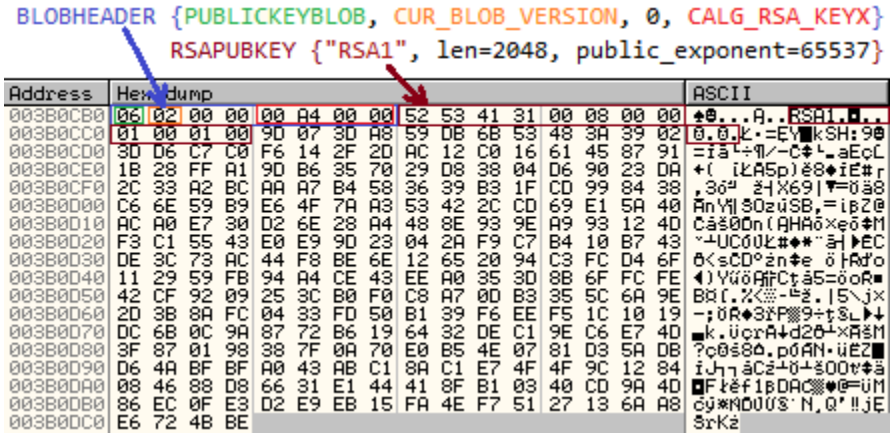
Below – decrypting public key from Base64:



Key is imported using function CryptImportPublicKeyInfo.

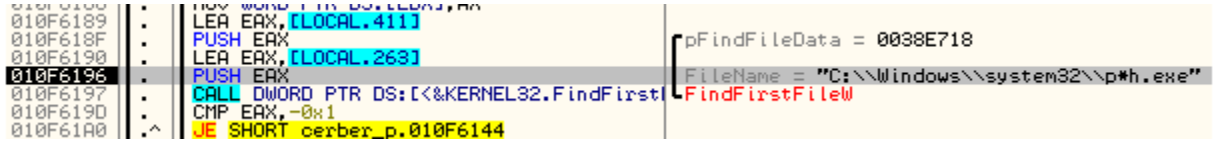


Configuration mentioned: *"rsa_key_size": 576* – but it turns out to be a 2048 bit key (BLOB size – 276 bytes)



Installation

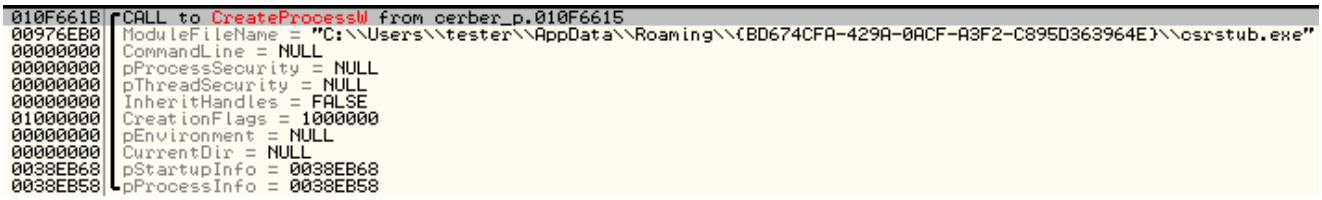
A file name of the dropped sample is created in a pretty interesting way. It is not fully random, but based on name of some file existing in the system, that is searched in the system using a random filter (format: “[random char]*[random char]. exe”, i.e “p*h.exe”):



The found file is compared with some built-in blacklist. When it pass the check, it is chosen as the new name of the dropped copy of the malware.

In order to prevent user from finding the malicious file by its creation timestamp it is changed to the timestamp of *kernel32.dll* existing on the local system.

After the successful installation, the initial malware sample terminates and deploys the dropped copy instead.



UAC Bypass

Cerber uses tricks to bypass Windows User Account Controll (UAC) and deploy itself with elevated privileges. It is achieved by the following steps:

1. Search an executable in C:\Windows\system32, that can auto elevate it’s privileges.
2. Search in it’s import table a DLL that can be hijacked
3. Copy the DLL into %TEMP% folder and patch it – add a code in a new section and patch entry point in order to redirect execution there. It will be used in order to run the cerber sample with elevated privileges. It uses: WinExec(“[cerber_path] **-eval 2524**“, SW_SHOWNORMAL)
4. Inject the code into explorer.exe – it is responsible for executing the UAC bypass. Creates a new folder in C:\Windows\system32 and copy there both files – an EXE and the patched DLL – under original names, then it deploys the EXE causing DLL to load and execute the malicious code.
5. When the UAC bypass is executed successfully, it is signalized to the original cerber sample by setting a property **cerber_uac_status** – added to a Shell_TrayWnd. Then, the original sample deletes dropped files and exits. Otherwise, it tries the same trick with different pair of EXE + DLL.

See below how it looks in action:

First, it searches an application that can be used to elevate privileges. The check is based on the fields in application manifest:

```
<autoElevate>true</autoElevate>
<requestedExecutionLevel level="requireAdministrator"/>
```

Among it's imported DLLs it searches a candidate suitable to be hijacked. This DLL is copied into *%TEMP%* folder

Then, it creates a suspended process of ***explorer.exe***, allocates memory in it's context and injects there own code. Details given below.

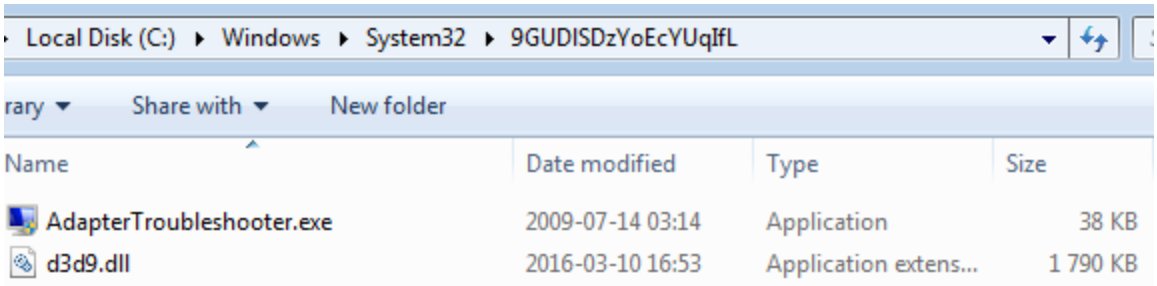
Injection into explorer is performed in several steps. First – malware is coping memory from the context of current process into the context of ***explorer.exe***. Current image of Cerber sample is replicated into a memory allocated in explorer at 0x70000. Similarly, the page containing filled data is copied at offset 0x91000 in explorer.

```
00DE4FD9  .   LEA EAX,[LOCAL.3]
00DE4FDC  .   PUSH EAX                          pBytesWritten = 0026DF38
00DE4FDD  .   PUSH ESI                          BytesToWrite = 7580C1DE (1971372510.)
00DE4FDE  .   PUSH [ARG.2]                      Buffer = cerber_p.00DE0000
00DE4FE1  .   MOV ESI,DWORD PTR DS:[<&KERNEL32.Wri kernel32.WriteProcessMemory
00DE4FE7  .   PUSH EBX                          Address = 0x70000
00DE4FE8  .   PUSH [ARG.1]                      hProcess = 00000148
00DE4FEB  .   CALL ESI                          WriteProcessMemory
00DE4FED  .   LEA EAX,[LOCAL.3]
00DE4FF0  .   PUSH EAX                          pBytesWritten = 0026DF38
00DE4FF1  .   PUSH [ARG.5]                      BytesToWrite = 10BC (4284.)
00DE4FF4  .   PUSH [ARG.4]                      Buffer = cerber_p.00DFA170
00DE4FF7  .   PUSH [LOCAL.4]                    Address = 0x91000
00DE4FFA  .   PUSH [ARG.1]                      hProcess = 00000148
00DE4FFD  .   CALL ESI                          WriteProcessMemory
```

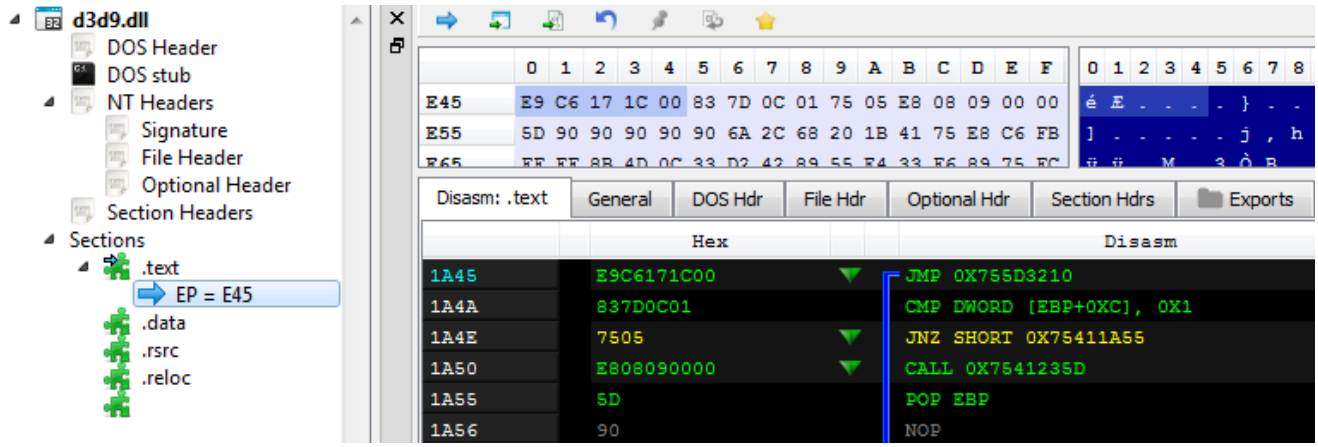
In order to run the injected code when the explorer.exe is resumed, malware performs patching of the carrier's Entry Point:

```
00DE500B  .   PUSH EAX                          pBytesWritten = 0026DF20
00DE500C  .   PUSH 0xE                          BytesToWrite = E (14.)
00DE500E  .   LEA EAX,[LOCAL.9]
00DE5011  .   PUSH EAX                          Buffer = 0026DF20
00DE5012  .   PUSH EDI                          Address = 0x630EFA
00DE5013  .   PUSH [ARG.1]                      hProcess = 00000148
00DE5016  .   MOV BYTE PTR SS:[EBP-0x24],0xB8
00DE501A  .   MOV WORD PTR SS:[EBP-0x1F],0xB850
00DE5020  .   MOV DWORD PTR SS:[EBP-0x1D],EBX
00DE5023  .   MOV WORD PTR SS:[EBP-0x19],0xD0FF
00DE5029  .   MOV BYTE PTR SS:[EBP-0x17],0xC3
00DE502D  .   CALL ESI                          WriteProcessMemory
00DE502F  .   PUSH 0x0                          RegionSize = 0x0
00DE5031  .   PUSH 0x0                          RegionBase = NULL
00DE5033  .   PUSH [ARG.1]                      hProcess = 00000148
00DE5036  .   CALL DWORD PTR DS:[<&KERNEL32.FlushInstructior FlushInstructionCache

Stack SS:[0026DF50]=00DE0000 (cerber_p.00DE0000)

Address   Hex dump     Disassembly              Comment
0026DF20  B8 00100900  MOV EAX,0x91000
0026DF25  50           PUSH EAX
0026DF26  B8 E1550700  MOV EAX,0x755E1
0026DF2B  FFD0         CALL EAX
0026DF2D  C3           RETN
```

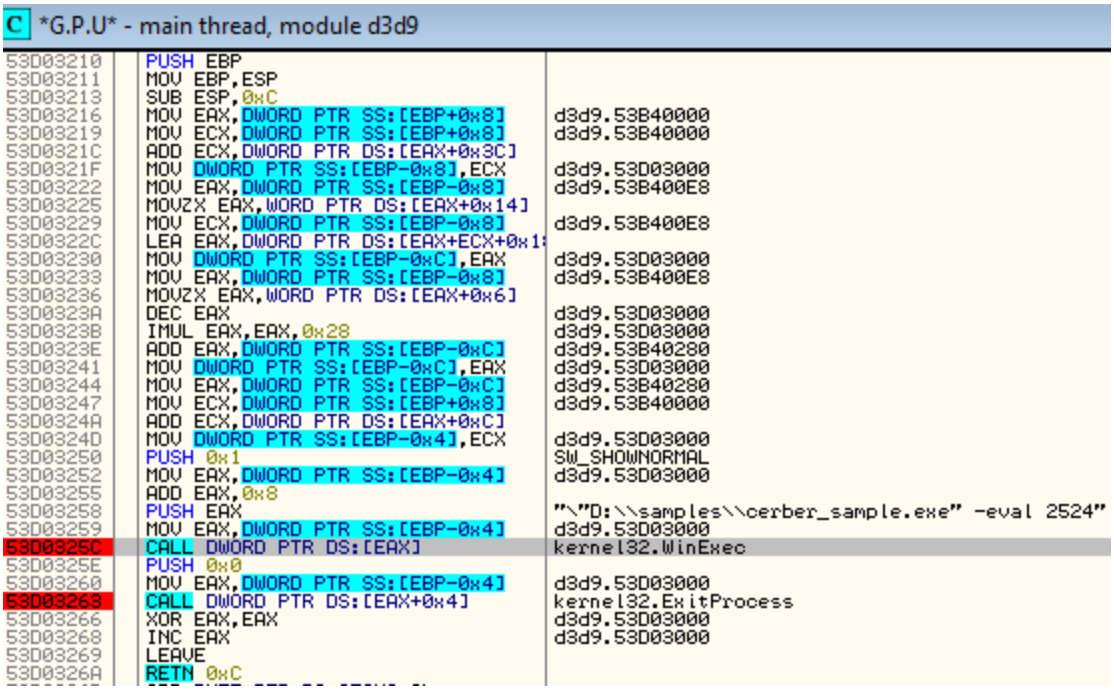
Now, Explorer's execution starts from the call to injected code. It is a function of Cerber sample – at RVA 0x55E1, called with a parameter 0x91000 – pointer to the memory page containing various dynamically loaded data, like function's handlers, paths of the files to be used, etc.
From inside this code injected to explorer, the DLL patched for UAC bypass is copied under the original name – along with the appropriate EXE. The executable is deployed (using ShellExecuteExW) and along with it, the patched DLL also runs.



The *d3d9.dll* is used in order to run the Cerber sample with elevated privileges. Entry Point of the DLL is patched with a jump to the new section.



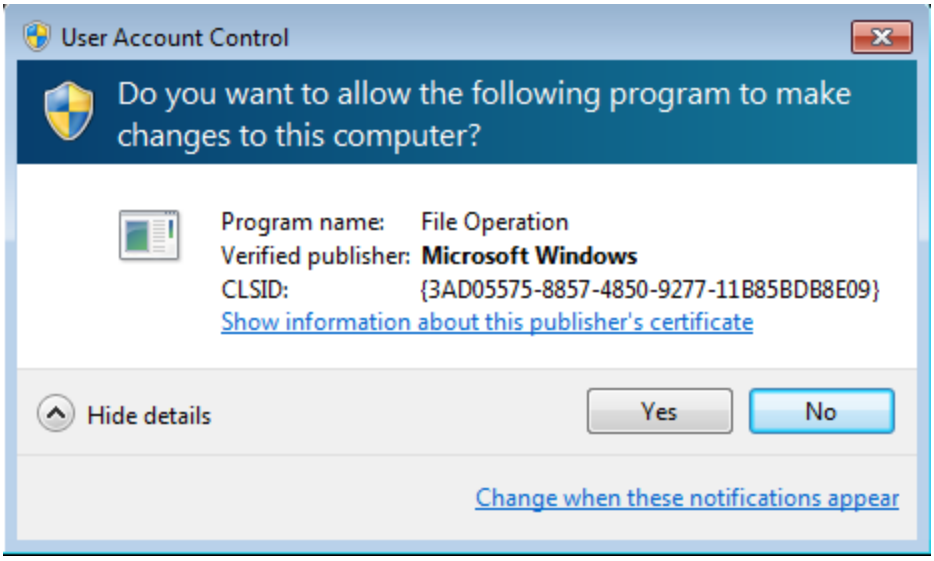
The new section contains the code that is supposed to execute the Cerber sample:



Successful UAC bypass is signalized by setting a property named "**cerber_uac_status**" in a found window of the class "*Shell_TrayWnd*". The initial Cerber sample waits for this status to change. If the timeout passed and it didn't changed it makes a new attempt of UAC bypass – using a different pair (EXE+DLL). Otherwise it cleans up the environment and terminates. Infection proceeds from inside of the elevated sample.

```
00DE570F  .   PUSH EBX                                ; Title = NULL
00DE5710  .   PUSH EAX                                ; Class = 00021000 ???
00DE5711  .   CALL DWORD PTR DS:[<&USER32.FindWindowW>] ; FindWindowW
00DE5717  .   PUSH EBX                                ; hData = NULL
00DE5718  .   MOV EBP,cerber_p.00DFB024               ; UNICODE "cerber_uac_status"
00DE571D  .   PUSH EBP                                ; Property = "cerber_uac_status"
00DE571E  .   PUSH EAX                                ; hWnd = 00021000
00DE571F  .   MOV DWORD PTR DS:[0xDFA170],EAX
00DE5724  .   CALL DWORD PTR DS:[<&USER32.SetPropW>]  ; SetPropW
00DE572A  .   MOV EDI,cerber_p.00DE55E1
00DE572F  .   PUSH EDI
00DE5730  .   CALL cerber_p.00DEDC6A
00DE5735  .   MOV ESI,EAX                             ; image_base
00DE5737  .   CALL cerber_p.00DEDCD4
00DE573C  .   MOV DWORD PTR SS:[ESP],0x10BC
00DE5743  .   PUSH cerber_p.00DFA170                  ; Arg5 = 00DFA170
00DE5748  .   SUB EDI,ESI                             ; cerber_p.00DE0000
00DE574A  .   PUSH EDI                                ; Arg4 = 000055E1
00DE574B  .   PUSH EAX                                ; Arg3 = 00021000
00DE574C  .   PUSH ESI                                ; Arg2 = 00DE0000
00DE574D  .   PUSH cerber_p.00DFB230                  ; "C:\\Windows\\explorer.exe"
00DE5752  .   CALL cerber_p.00DE5052                  ; inject_into_explorer
00DE5757  .   ADD ESP,0x18
00DE575A  .   TEST EAX,EAX
00DE575C  .v  JE SHORT cerber_p.00DE5786
00DE575E  .   XOR ESI,ESI                             ; cerber_p.00DE0000
00DE5760  >   PUSH EBP                                ; Property = "cerber_uac_status"
00DE5761  .   PUSH DWORD PTR DS:[0xDFA170]            ; hWnd = 0001004E (class='Shell_TrayWnd')
00DE5767  .   CALL DWORD PTR DS:[<&USER32.GetPropW>]  ; GetPropW
00DE576D  .   TEST EAX,EAX
00DE576F  .v  JNZ SHORT cerber_p.00DE5784             ; is_uac_bypassed?
00DE5771  .   PUSH 0x3E8                              ; Timeout = 1000. ms
00DE5776  .   CALL DWORD PTR DS:[<&KERNEL32.Sleep>]   ; Sleep
00DE577C  .   INC ESI                                 ; cerber_p.00DE0000
00DE577D  .   CMP ESI,0x3C
00DE5780  .^  JB SHORT cerber_p.00DE5760
00DE5782  .v  JMP SHORT cerber_p.00DE5786
00DE5784  >   MOV BL,0x1                              ; success
00DE5786  >   POP EDI                                 ; cerber_p.00DFB230
```

In case if UAC level is set to default (or lower), Cerber can bypass it silently. However, in case if it is set to the highest, the following alert pops up:



It keeps reappearing till the user click “Yes”:

Conclusion

Cerber is a pretty powerful ransomware written with attention to details. This analysis highlighted only some of the elements. It has rich customization options and various tricks to make analysis harder. Although this product appeared recently, for sure its authors are not new in the field of malware development. We can expect it will be gaining popularity and may carry some new tricks in the future.

Appendix