Look Into Locky Ransomware

blog.malwarebytes.com/threat-analysis/2016/03/look-into-locky/

hasherezade March 1, 2016

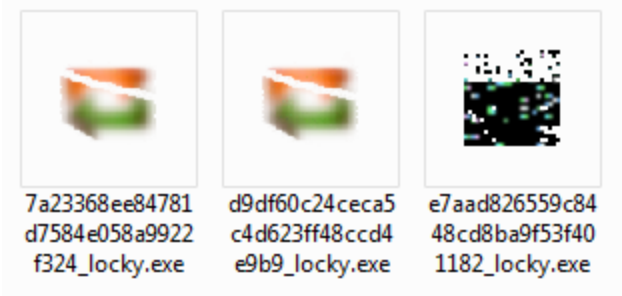
Locky is a new ransomware that has been released (most probably) by the Dridex gang (source). Not surprisingly, it is well prepared, which means that the threat actor behind it has invested sufficient resources for it, including its mature infrastructure. Let's take a look.

Analyzed samples

- 7a23368ee84781d7584e058a9922f324
  payload: 74dde1905eff75cf3328832988a785de <- main focus of this analysis
- d9df60c24ceca5c4d623ff48ccd4e9b9
- e7aad826559c8448cd8ba9f53f401182

Behavioral analysis

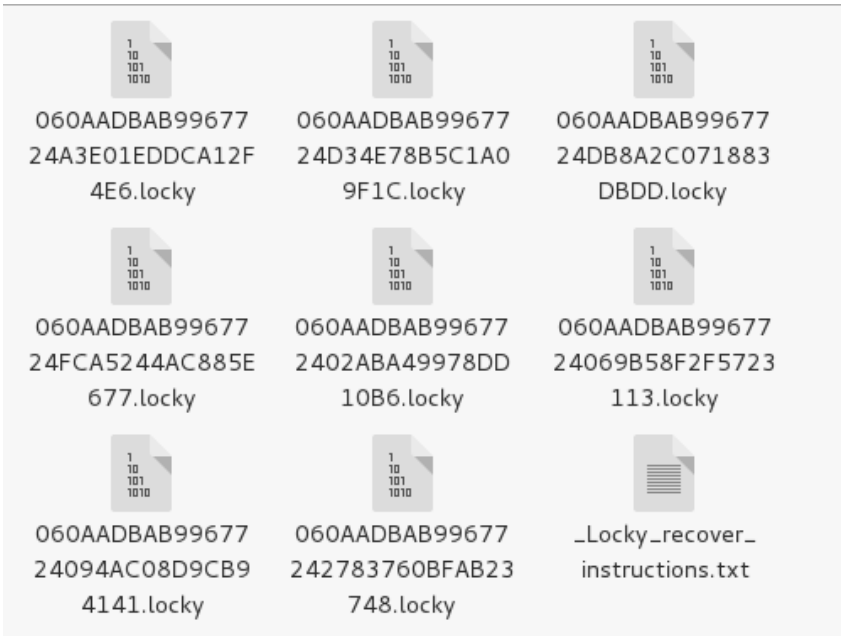
Locky is usually delivered via downloader in MS Office document (i.e. DOC) or JavaScript – e-mail attachment in a phishing campaign. The payload is a 32-bit Windows executable, containing the malicious core packed in a crypter/dropper (they are various, with various icons).



After being deployed it disappears and runs its dropped copy (renamed to **svchost.exe**) from the **%TEMP%** folder.

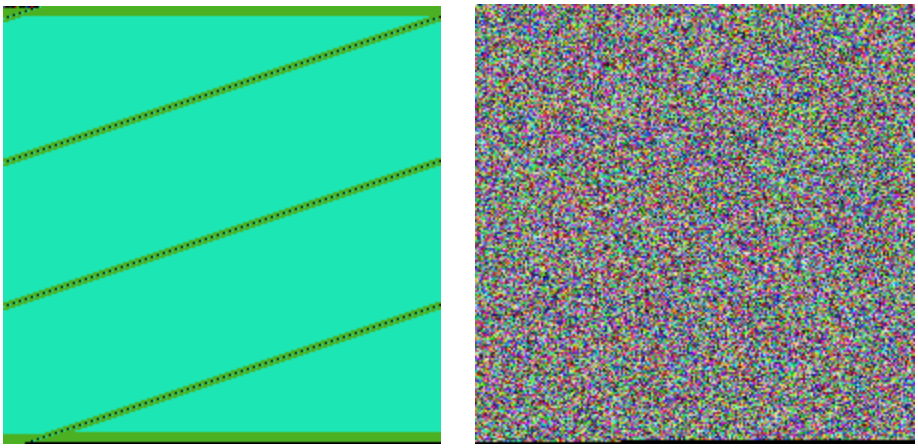
Encryption process

Files that have been encrypted are fully renamed. The beginning of the name (first 16 characters) is the unique ID of the victim. Then comes the ID of the file and the extension **.locky** that is typical for this ransomware.

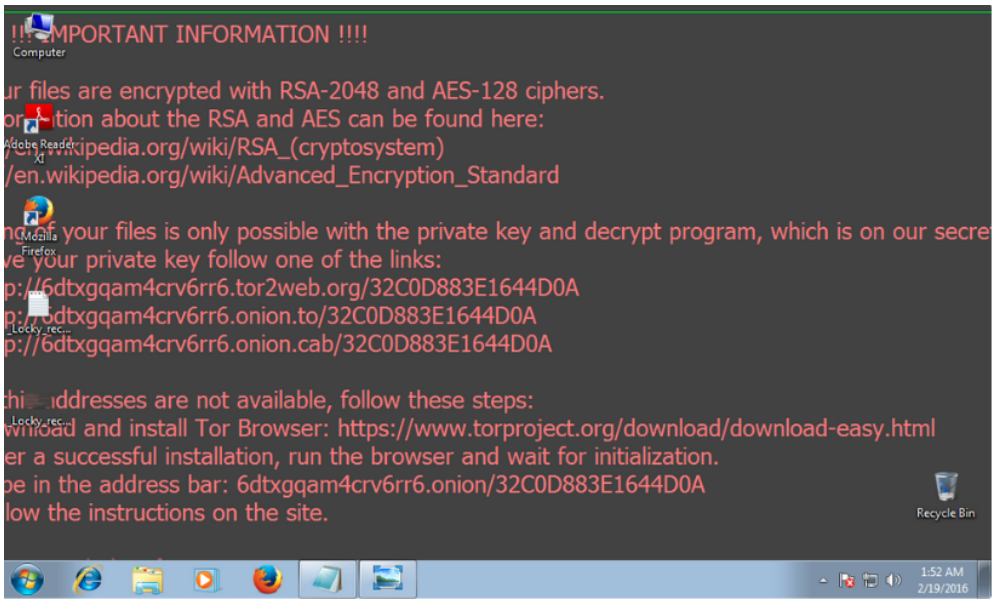


The encrypted content has a high level of entropy and no patterns are visible.

Below: visualization of raw bytes of **square.bmp**. Left: unencrypted, right: encrypted.



After executing, Locky displays the ransom note in text and bitmap forms, setting the latter as the affected user's wallpaper.

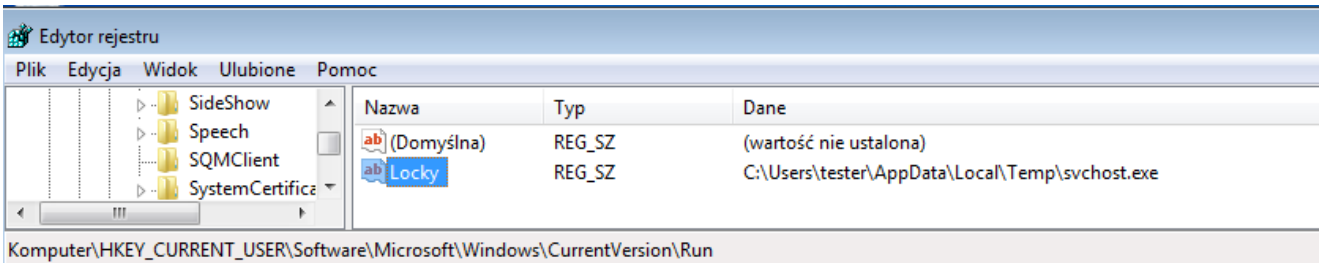


Text is localized to the language detected in the system. Translation looks professional enough (not from the auto translator), which may indicate that the threat actors target multiple countries – and prepared about this particular detail well. See sample translations (Polish, Spanish) here.

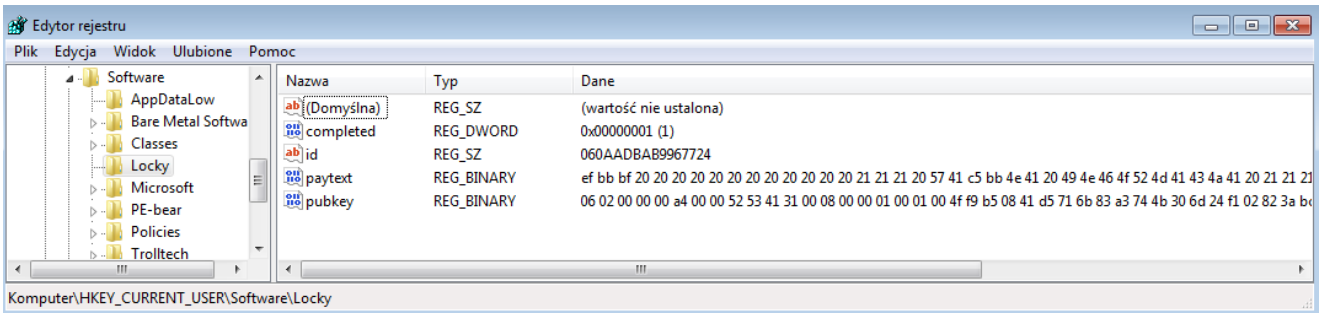
Registry keys

Looking at the registry we can find that a few elements have been added.

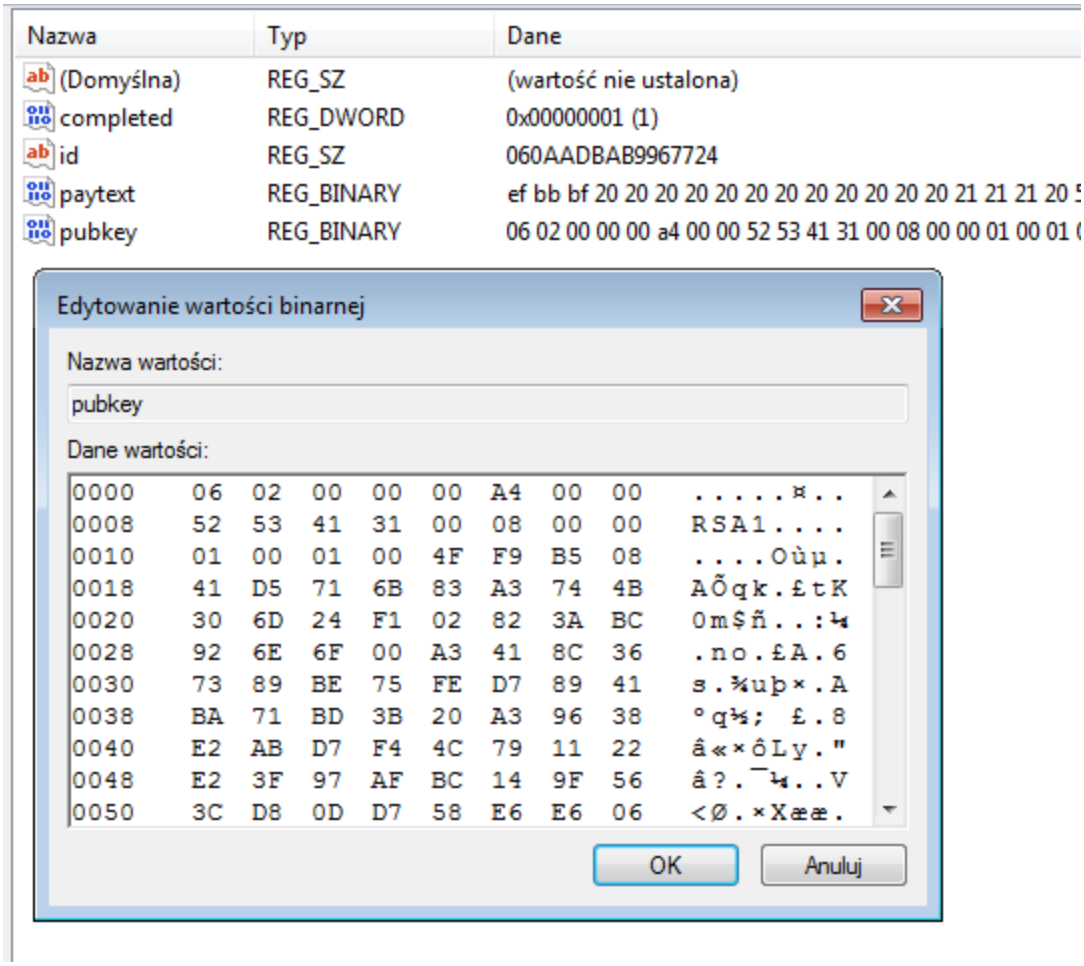
Key in autorun, to start the malware automatically after the system restart:



Data specific to the victim – individual ID, public RSA key and text of the ransom note to be displayed:

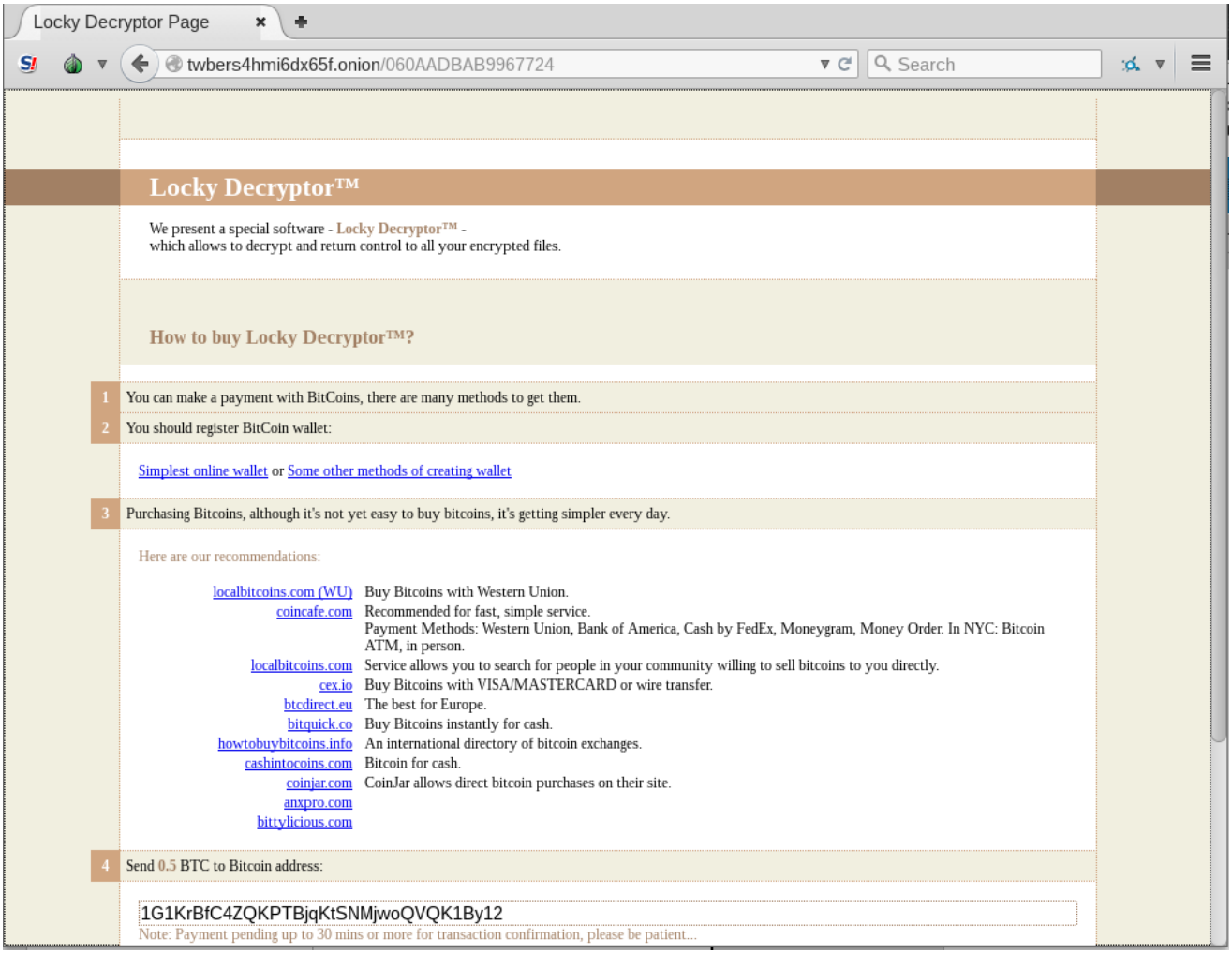


Public key stored in the registry:



Website for the victim

Each Locky victim has a Web page that can be accessed via Tor. These pages contain further instructions to the victim and support for managing payments.



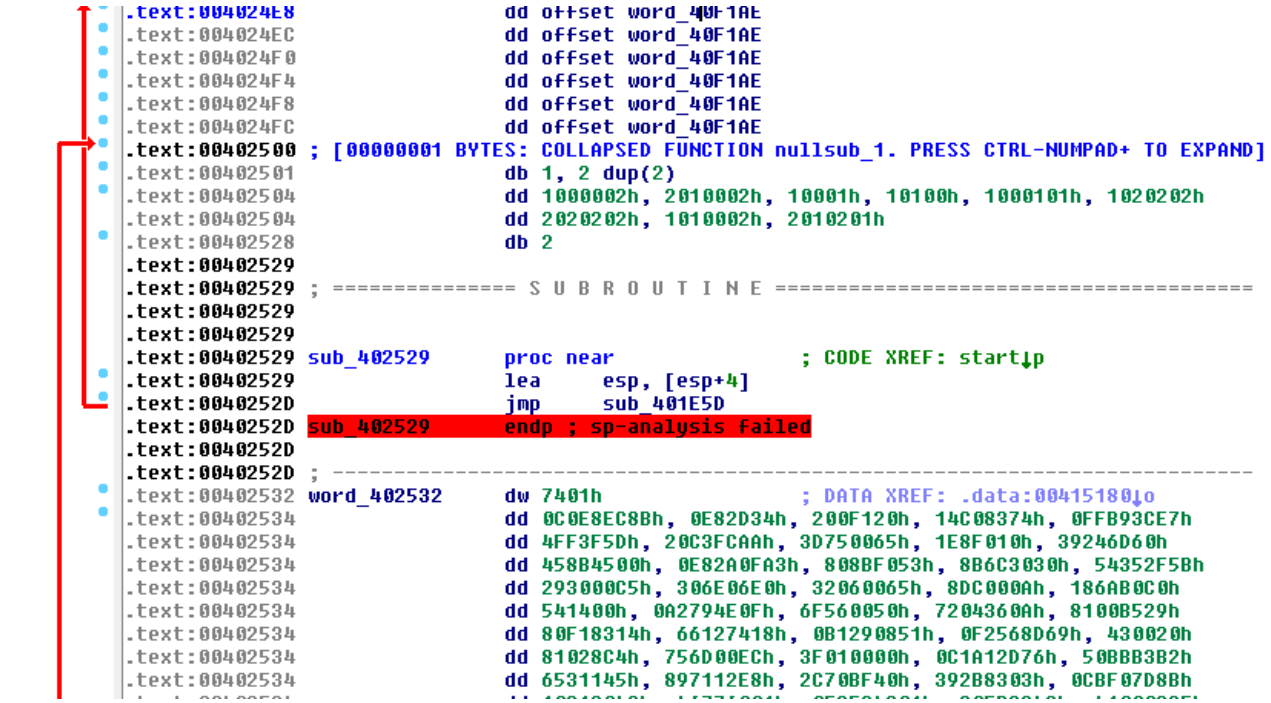
Network communication

Locky communicates with the CnC, but it is difficult to analyze it via simple sniffing tools because full communication is encrypted:

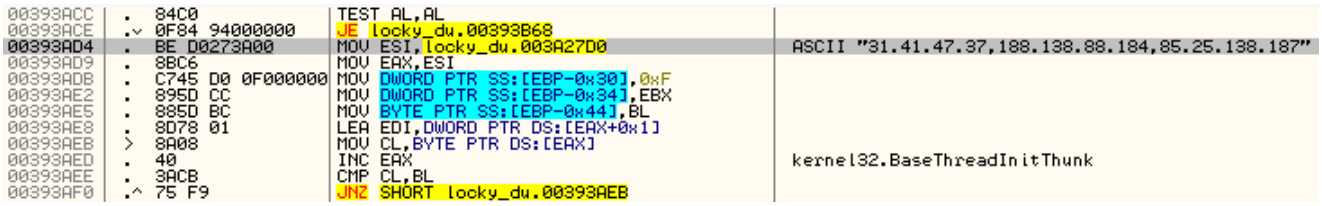
More about the protocol can be learned by reading the code…

Inside

Every sample of Locky comes packed in some crypter, so the code is unreadable at first.



However, the core itself is not that obfuscated. After unpacking the outer layer of its defense, we can see valid strings and function calls. They give some explanation to the unreadable network capture. The RSA key as well as the ransom note are fetched from the server by a HTTP based protocol. The current sample comes with a list of 3 IP addresses.



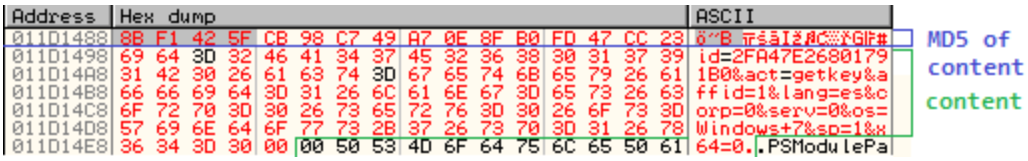
- 31.41.47.37
- 188.138.88.184
- 85.25.138.187

Additionally it makes use of DGA – Domain Generation Algorithm (more described here).

Communication protocol

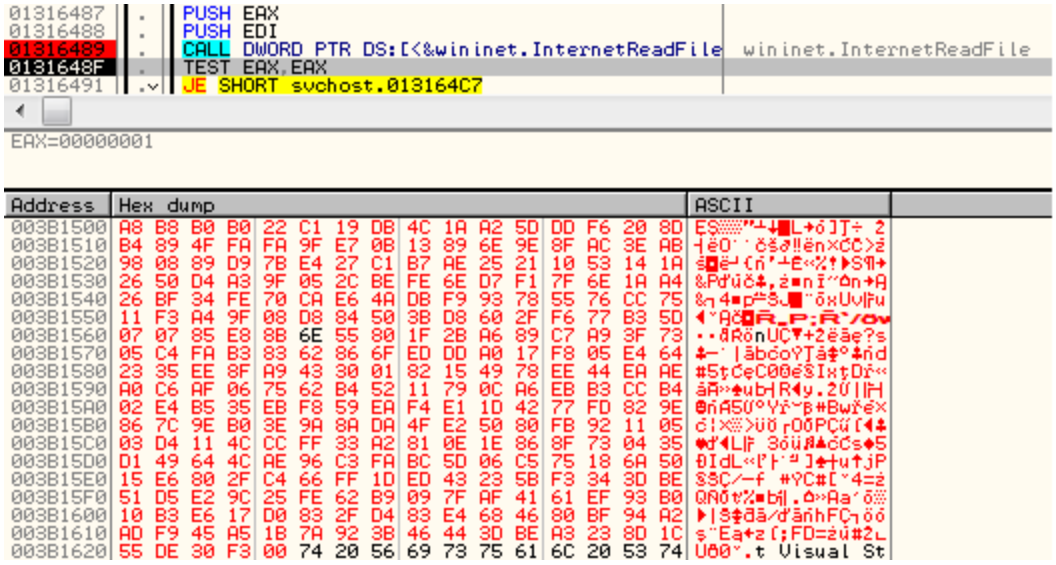
Locky's communication protocol is pretty simple: it consists of a POST request with parameters in a typical *key=value* format. However, as mentioned before, they are not sent by an open text, but wrapped and encrypted. First, the request is prepared and it's parameters are filled. Then its MD5 is calculated. Both elements are concatenated and encrypted together.

Example of wrapped request (before encryption):

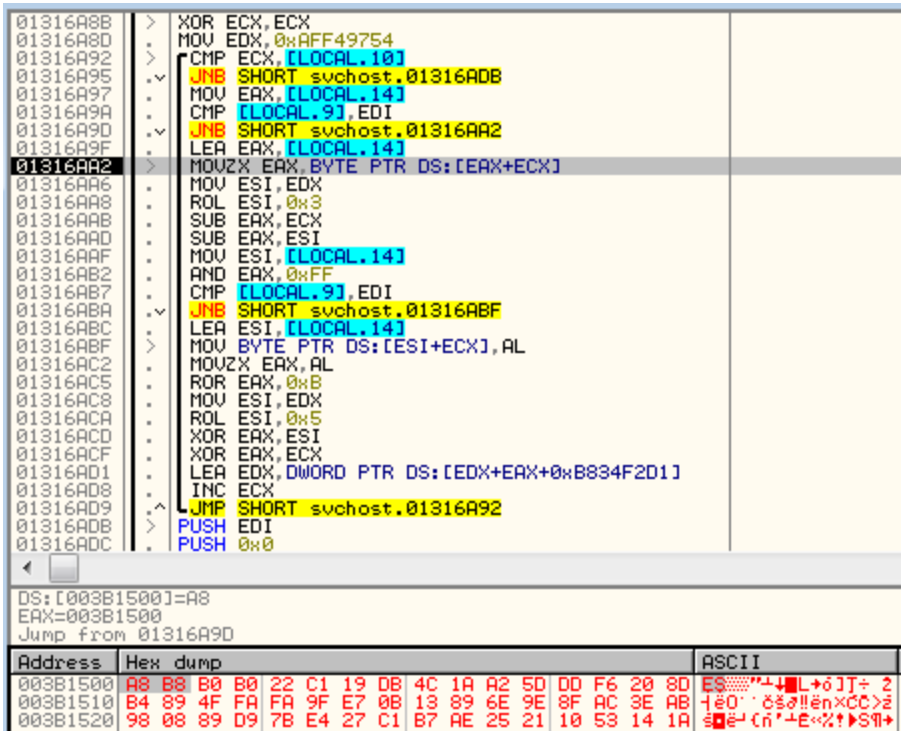


Similarly, when the response comes, first it gets decrypted, then its MD5 is validated – and if it passed the validation then it is parsed.

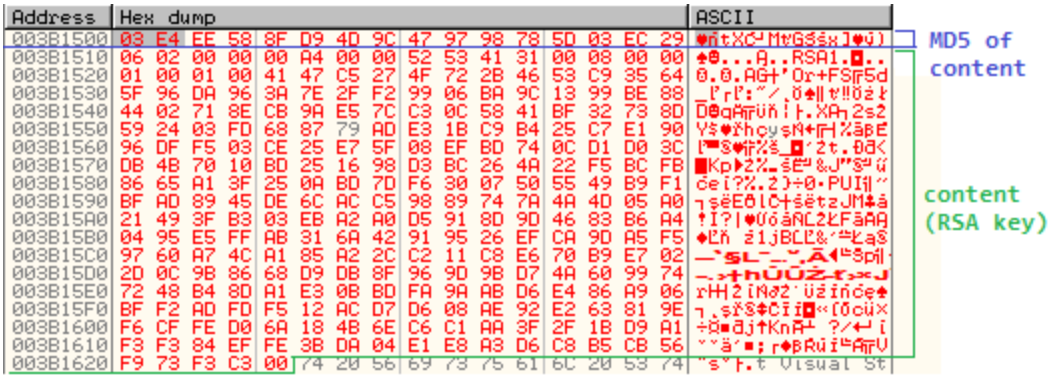
Example of received response (encrypted):



Decrypting:



Decrypted response turns out to be an RSA key prompted by its hash:



Locky uses 3 commands (identified by the key ***act***):

- getkey
- gettext
- stats

We have explained the actions in further detail below.

[getkey] Initial registration and fetching the RSA key:

```
id=[16]&act=getkey&affid=1&lang=[2:lang]&corp=[0-1]&serv=[0-1]&os=[Windows name]&sp=[num]&x64=[0-1]
```

Unique user **ID** is 16 byte long hexadecimal string, created locally (pseudocode):

```
win_dir = GetWindowsDirectory
mount_point_name = GetVolumeNameForVolumeMountPoint(win_dir)
GUID = get_GUID(mount_point_name)
md5sum = MD5(GUID)
id = md5sum.uppercase().substr(0,16)
```

After that follows:

Language: obtained by functions: GetLocaleInfo , GetUserDefaultUILanguage. System info – fetched by GetVersionEx and GetSystemMetrics(**SM_SERVERR2**) and translated to the built in lists. IsWow64Process is used to identify if the system is 64bit.

[gettext] Fetching the ransom text:

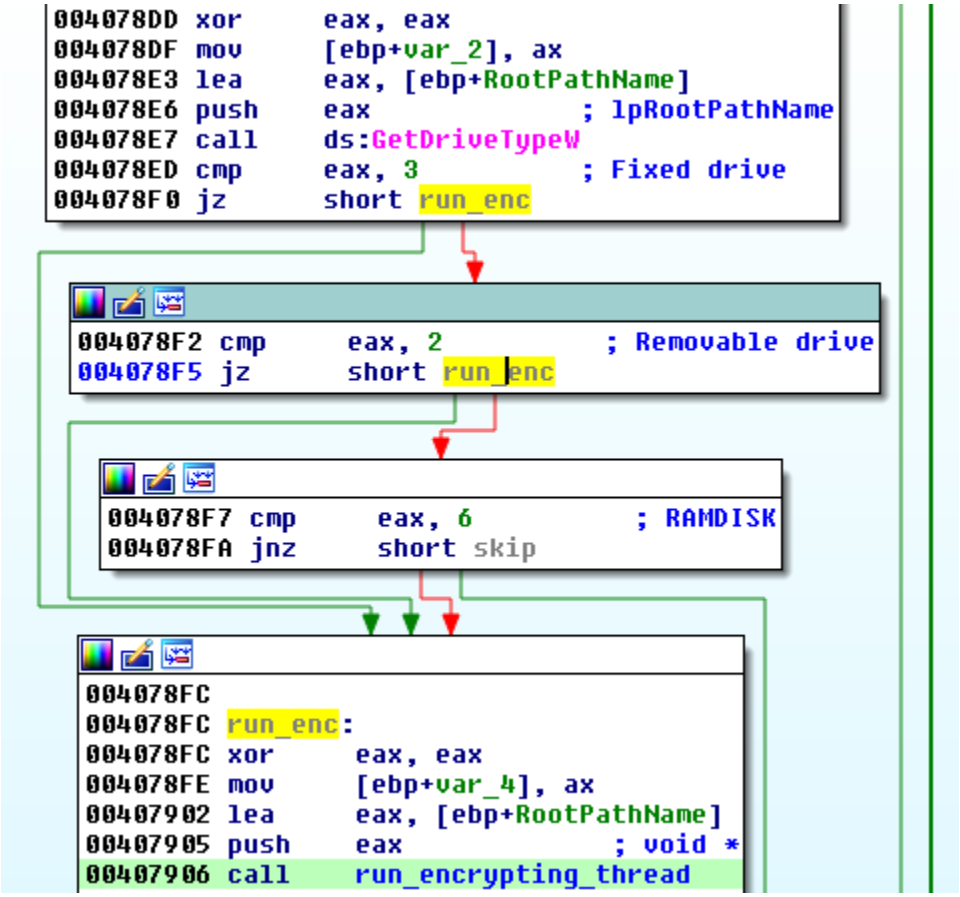
```
id=[16]&act=gettext&lang=[2:lang]
```

[stats] Sending statistics about encrypted files:

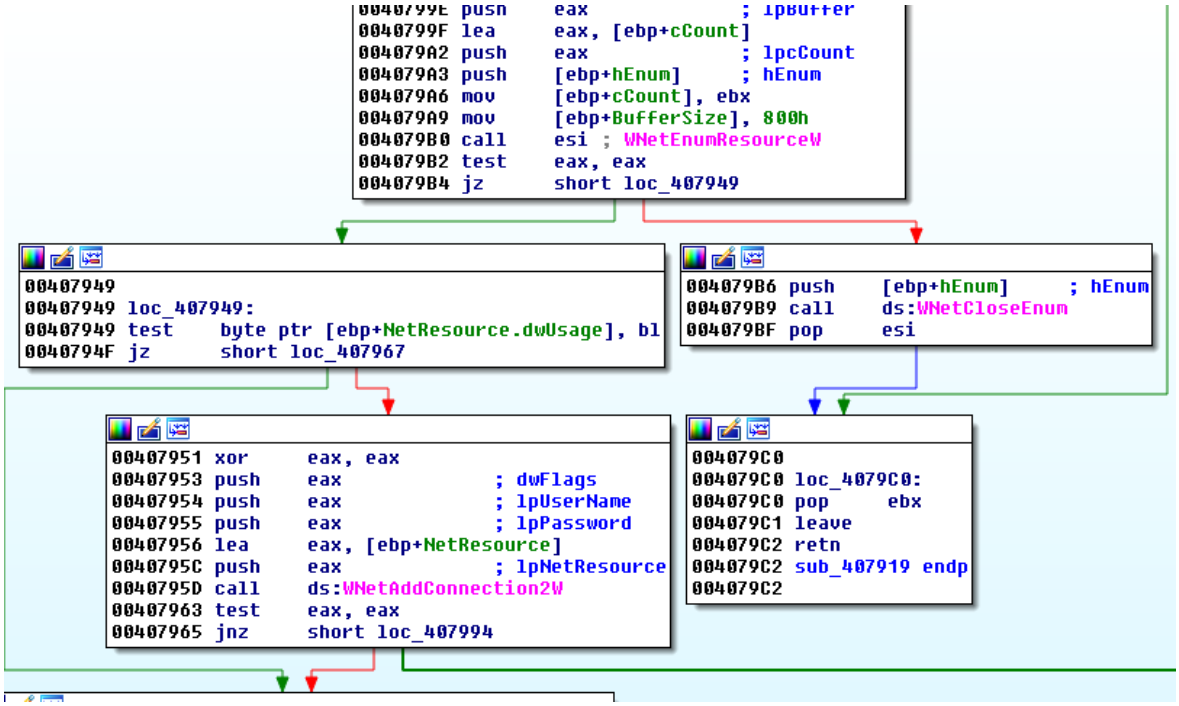
```
id=[16]&act=stats&path=[root_path]&encrypted=[num]&failed=[num]&length=[num]
```

What is attacked?

Locky attacks 3 types of local drives: fixed, removable and ramdisks…



…as well as network resources. Network shares are mapped using `WNetAddConnection2`

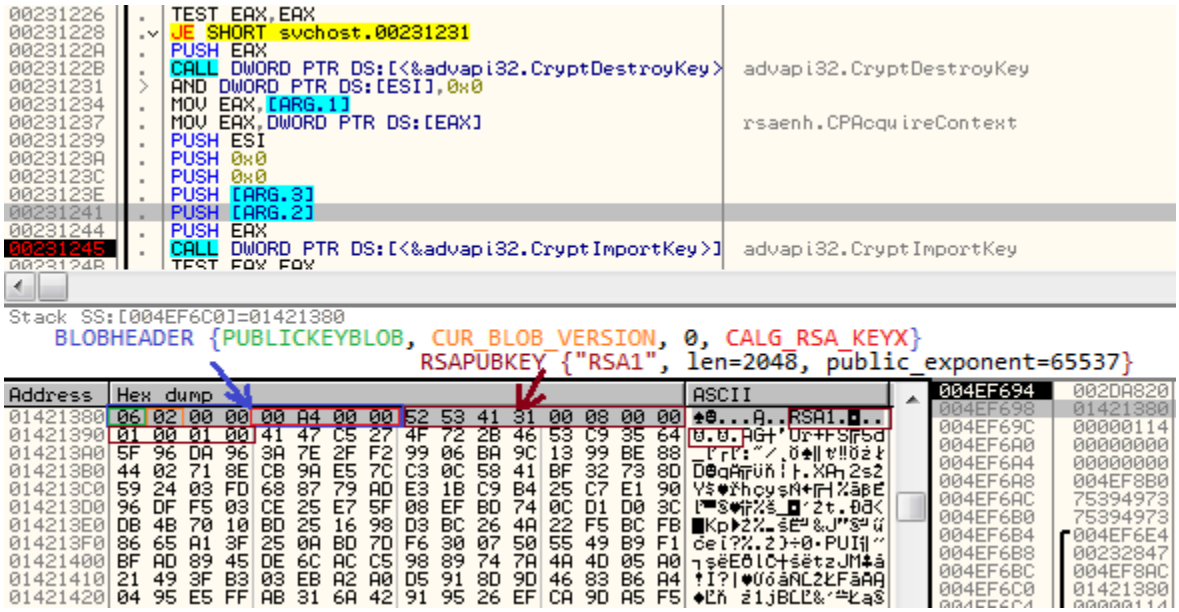


For every drive a new encrypting thread is started.

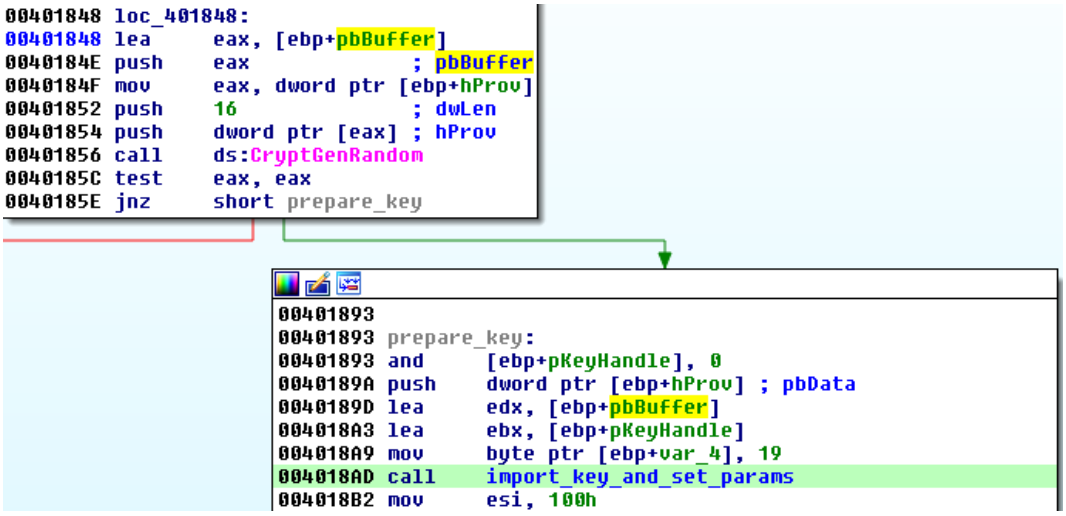
How does the encryption work?

In the ransom note attackers claimed that Locky uses both RSA and AES algorithms. Looking at the code we can confirm this. Cryptography is implemented using Windows Crypto API and really uses the mentioned algorithms.

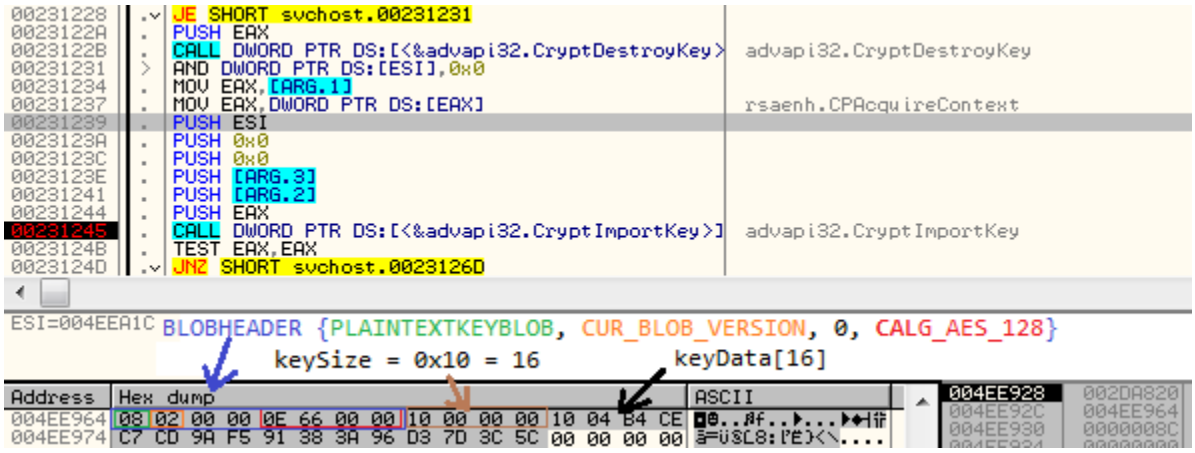
First, RSA key (2048 bit) is fetched from the server and imported:



The RSA key is used to encrypt AES keys, which are randomly generated for each file.



Below – importing a random AES key (128 bit long):



Processing of the files starts by enumerating them and storing in a list. Then the encryption proceeds by this list.

```
00402C7E
00402C7E ; DWORD __stdcall encrypting_thread(LPVOID root_path)
00402C7E encrypting_thread proc near
00402C7E
00402C7E var_20= byte ptr -20h
00402C7E var_10= dword ptr -10h
00402C7E var_C= dword ptr -0Ch
00402C7E var_4= dword ptr -4
00402C7E root_path= dword ptr  8
00402C7E
00402C7E mov     eax, offset loc_40EE53
00402C83 call    __EH_prolog
00402C88 sub     esp, 1Ch
00402C8B and     [ebp+var_4], 0
00402C8F push    ebx
00402C90 push    esi
00402C91 push    edi
00402C92 mov     [ebp+var_10], esp
00402C95 push    [ebp+root_path]
00402C98 lea     esi, [ebp+var_20]
00402C9B call    enumerate_files
00402CA0 pop     ecx
00402CA1 mov     eax, esi
00402CA3 push    eax
00402CA4 push    [ebp+root_path]
00402CA7 mov     byte ptr [ebp+var_4], 1
00402CAB call    encrypt_and_drop_note
00402CB0 pop     ecx
```

Every thread collects statistics about the encrypted files (i.e summary of how many files has been encrypted in a particular path):

```
00402A82 mov     edi, eax
00402A84 lea     eax, [ebp+var_C8]
00402A8A push    offset aId_0    ; "id="
00402A8F push    eax
00402A90 mov     ebx, offset dword_416C78
00402A95 mov     byte ptr [ebp+var_4], 0Bh
00402A99 call    sub_404430
00402A9E push    offset aActStatsPath ; "&act=stats&path="
00402AA3 push    eax
00402AA4 lea     eax, [ebp+var_1C4]
00402AAA mov     byte ptr [ebp+var_4], 0Ch
00402AAE call    sub_4044AA
00402AB3 mov     ecx, eax
00402AB5 mov     eax, edi
00402AB7 lea     edi, [ebp+var_154]
00402ABD mov     byte ptr [ebp+var_4], 0Dh
00402AC1 call    sub_4044E1
00402AC6 push    offset aEncrypted ; "&encrypted="
00402ACB push    eax
00402ACC lea     eax, [ebp+var_1FC]
00402AD2 mov     byte ptr [ebp+var_4], 0Eh
00402AD6 call    sub_4044AA
00402ADB add     esp, 40h
00402ADE mov     ecx, eax
00402AE0 mov     eax, [ebp+arg_4]
00402AE3 lea     edi, [ebp+var_11C]
00402AE9 mov     byte ptr [ebp+var_4], 0Fh
00402AED call    sub_4044E1
00402AF2 push    offset aFailed  ; "&failed="
00402AF7 push    eax
00402AF8 lea     eax, [ebp+var_AC]
00402AFE mov     byte ptr [ebp+var_4], 10h
00402B02 call    sub_4044AA
00402B07 mov     ecx, eax
00402B09 mov     eax, [ebp+var_14]
00402B0C lea     edi, [ebp+var_E4]
00402B12 mov     byte ptr [ebp+var_4], 11h
00402B16 call    sub_4044E1
00402B1B push    offset aLength  ; "&length="
00402B20 mov     byte ptr [ebp+var_4], 12h
00402B24 push    eax
```

Statistics are encrypted and sent to the C&C.

Ransom note

As mentioned before, ransom note in a language detected language by GetUserDefaultUILanguage is downloaded from the server.

Most ransomware drops ransom notes in HTML form, and then opens it in a Web browser. Locky does something more interesting: it renders and sets a bitmap as wallpaper.

```
004037FB loc_4037FB:              ; lpFileName
004037FB push    eax
004037FC mov     ecx, offset lpData
00403801 call    create_and_write_file ; drop .txt note
00403806 pop     ecx
```

```
00403807
00403807 loc_403807:
00403807 lea     eax, [ebp+pvParam]
0040380A call    find_file
0040380F test    al, al
00403811 jnz     short loc_403869
```

```
00403813 lea     eax, [ebp+var_A4]
00403819 push    eax
0040381A call    set_codepage
0040381F push    eax
00403820 lea     eax, [ebp+var_28]
00403823 push    eax
00403824 mov     byte ptr [ebp+var_4], 3
00403828 call    render_ransom_bitmap
0040382D add     esp, 0Ch
```

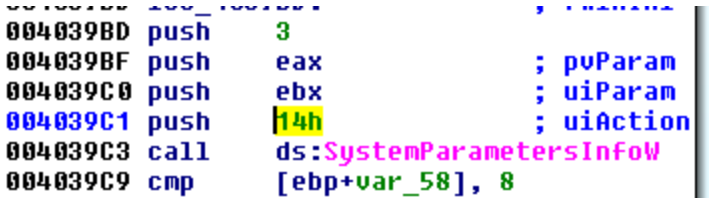
Bitmap rendering:

```
001A34E2  .   PUSH 0x404040                          ┌Color = RGB(64.,64.,64.)
001A34E7  .   MOV [LOCAL.37],EBX                     │
001A34ED  .   MOV [LOCAL.36],EBX                     │
001A34F3  .   MOV [LOCAL.34],EAX                     │kernel32.BaseThreadInitThunk
001A34F9  .   CALL EDI                               └CreateSolidBrush
001A34FB  .   MOV ESI,EAX                             kernel32.BaseThreadInitThunk
001A34FD  .   MOV [LOCAL.8],ESI
001A3500  .   CMP ESI,EBX
001A3502  .v  JNZ SHORT locky_un.001A3509
001A3504  .   CALL locky_un.001A4CF8
001A3509  >   PUSH ESI                               ┌hBrush = NULL
001A350A  .   LEA EAX,[LOCAL.37]                     │
001A3510  .   PUSH EAX                               │pRect = 75823C33 (-1957298293.,264865260.,34437.,141950720.)
001A3511  .   MOV BYTE PTR SS:[EBP-0x4],0x4          │
001A3515  .   PUSH [LOCAL.11]                        │hDC = NULL
001A3518  .   CALL DWORD PTR DS:[<&user32.FillRect>] └FillRect
001A351E  .   TEST EAX,EAX                            kernel32.BaseThreadInitThunk
001A3520  .v  JNZ SHORT locky_un.001A353F
001A3522  >   CALL DWORD PTR DS:[<&kernel32.GetLastErr ┌GetLastError
001A3528  .   MOV [LOCAL.4],EAX                       kernel32.BaseThreadInitThunk
001A352B  .   MOV [LOCAL.5],locky_un.001B2218
001A3532  .   PUSH locky_un.001B3C64
001A3537  .   LEA EAX,[LOCAL.5]
001A353A  .^  JMP locky_un.001A340A
001A353F  >   MOV BYTE PTR SS:[EBP-0x4],0x3
001A3543  .   CMP ESI,EBX
001A3545  .v  JE SHORT locky_un.001A354E
001A3547  .   PUSH ESI                               ┌hObject = NULL
001A3548  .   CALL DWORD PTR DS:[<&gdi32.DeleteObject> └DeleteObject
001A354E  >   PUSH 0x8080FF                          ┌Color = RGB(255.,128.,128.)
001A3553  .   PUSH [LOCAL.11]                        │hDC = NULL
001A3556  .   CALL DWORD PTR DS:[<&gdi32.SetTextColor> └SetTextColor
001A355C  .   CMP EAX,-0x1
001A355F  .^  JE SHORT locky_un.001A3522
001A3561  .   PUSH 0x1                               ┌BkMode = TRANSPARENT
001A3563  .   PUSH [LOCAL.11]                        │hDC = NULL
001A3566  .   CALL DWORD PTR DS:[<&gdi32.SetBkMode>] └SetBkMode
001A356C  .   CMP EAX,EBX
```

Wallpaper settings are edited by registry keys:

```
00403990 lea     eax, [ebp+phkResult]
00403993 push    eax
00403994 lea     ecx, [ebp+var_28]
00403997 mov     edx, offset aTilewallpaper ; "TileWallpaper"
0040399C mov     byte ptr [ebp+var_4], 8
004039A0 call    set_reg_value
004039A5 push    1
004039A7 xor     edi, edi
```

After successful rendering and saving the bitmap, it sets it as a wallpaper using SystemParamsInfo (action 0x14 = SPI_SETDESKWALLPAPER)



Conclusion

Locky struck in February but it has already gained popularity. Due to the fact that it is a wide spread attack, carried by the same entities that distribute Dridex, it easily triggered interest of many researchers. Upon closer inspection, however, we can say that it is not that different from common ransomware. It looks solidly written and well prepared, but it doesn't show too much novelty so far.

Appendix