DMA Locker: New Ransomware, But No Reason To Panic

blog.malwarebytes.com/threat-analysis/2016/02/dma-locker-a-new-ransomware-but-no-reason-to-panic/

hasherezade February 3, 2016

DMA Locker is another ransomware that appeared at the beginning of this year. For now it has been observed to be active only on a small scale (source) – but we just want to warn you that it exists.

[UPDATE] READ ABOUT THE LATEST VERSION OF DMA LOCKER: 4.0

UPDATE [4 Feb 2016]: I apologize to everyone misguided by my rush conclusions about the crypto. After further analysis and consultation with other analysts (special thanks to @fwosar and @maciekkotowicz) I confirmed that in reality it is AES in ECB mode. Low entropy was just caused by the fact, that it encrypts separately 16 byte chunks, that are small enough to give this effect. Authors of the malware told many lies in their ransom note, but this one was true, just my mistake. The only way to recover the key is to find the original sample with key included. My goal is always to provide best quality analysis – this time I failed, but I tried to fix it as soon as possible and not let the false information spreading.

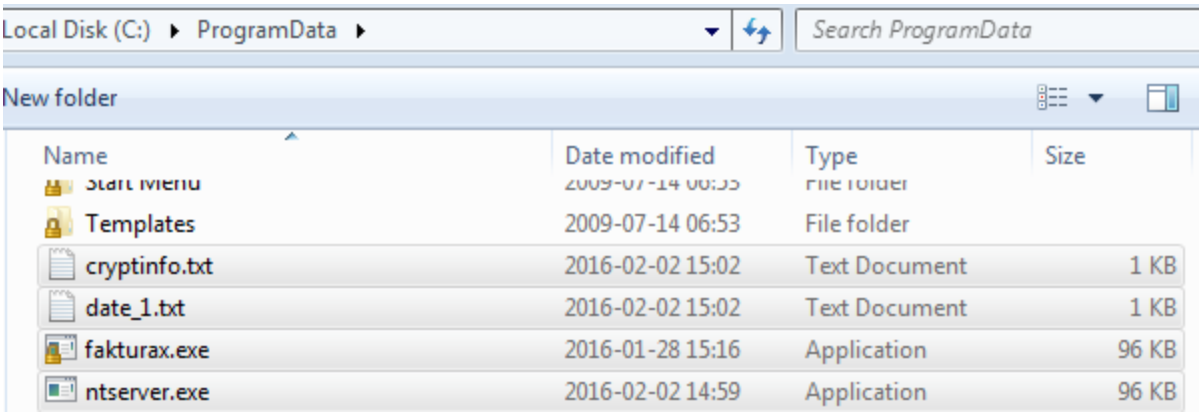
Analyzed samples

- d35344b1f48764ba083e51438121e6a9 – Polish version type 2 (from Jan 2016) <- main focus of this analysis
- 4190df2af81ece296c465e245fc0caea – English version type 2 (from Jan 2016)
- 6fbd3cdcafd6695c384a1119873786aa – Polish version type 1 (from Dec 2015)

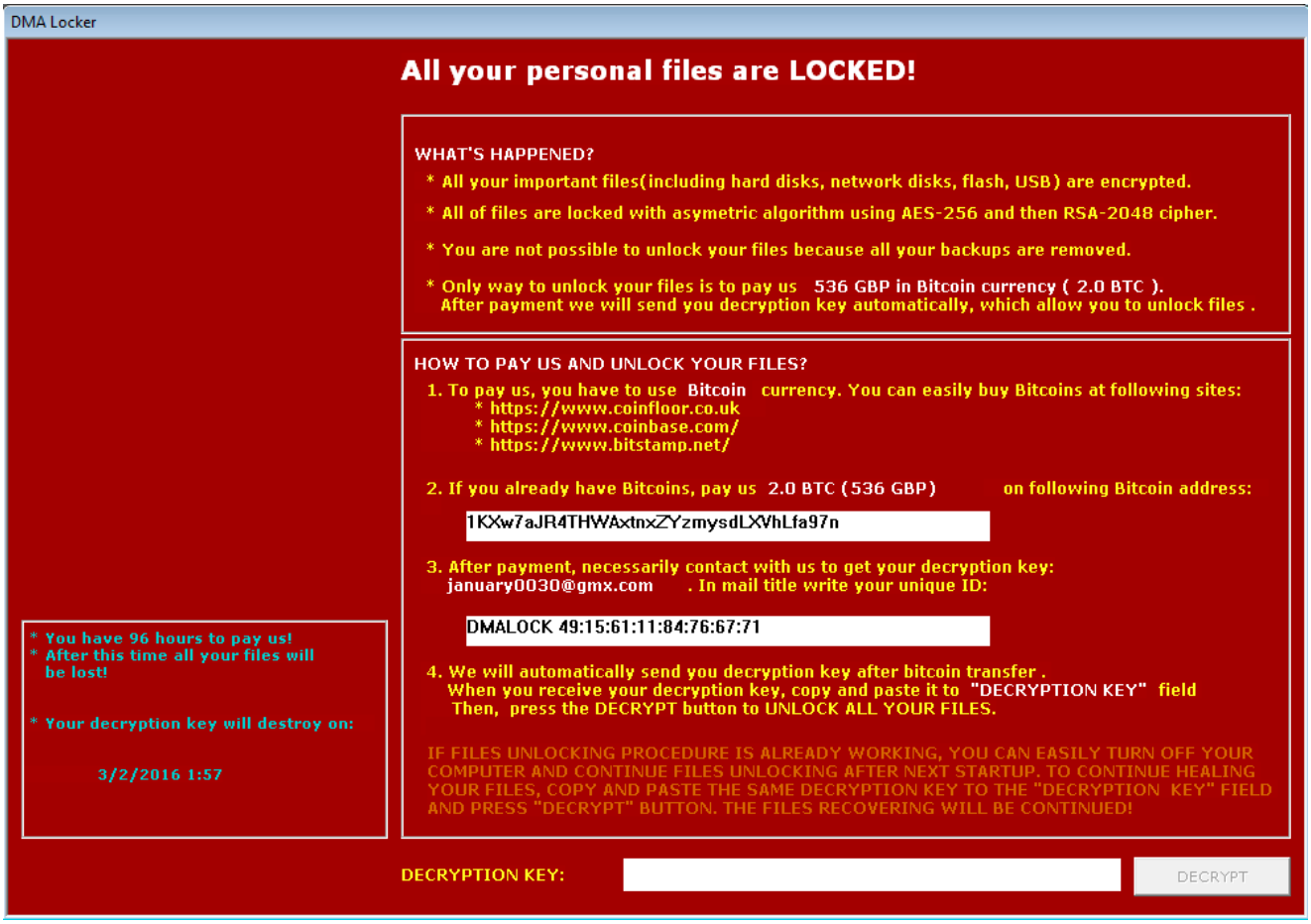
// Special thanks to malware hunters: @PhysicalDrive0 , @JAMESWT_MHT and @siri_urz for their respective help in collecting the samples!

Behavioral analysis

When deployed, the ransomware moves itself into **C:\ProgramData** (or **C:\Documents and Settings\All Users\Dokumenty\)**, renamed to **fakturax.exe** and drops another, modified copy: **ntserver.exe**. File **faktura.exe** is removed after execution. Depending on its version, it may also drop some other files in the same location.



Symptoms of this ransomware can be recognized by a red window popping up on the screen. So far, it has been observed in two language versions – Polish or English. An example of the English is below:

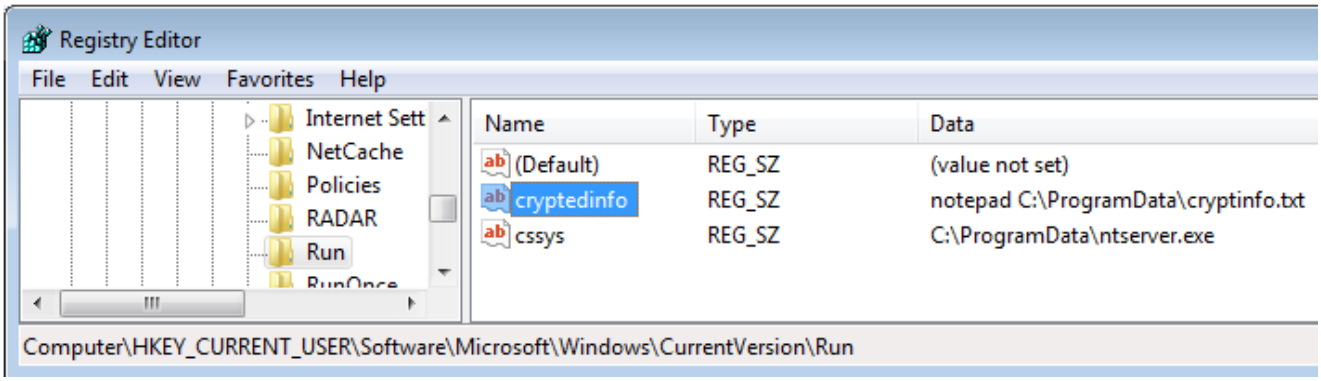


Earlier version comes with a bit different GUI (also Polish or English variant):

In contrast to other ransomware that are offering a separate decrypter, **DMA Locker** comes with a decrypting feature built-in. It is available from the GUI with ransom note. If the user enters a key (32 characters long) in the text field and clicks the button, the program switches to the decryption mode (using supplied key):

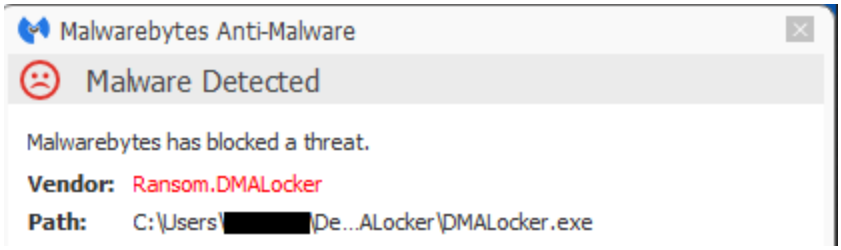
The program is not very stable and may crash during encryption. An older version has been observed to sometimes crash after finishing encryption – but before displaying any info about what happened, which may be very confusing for the victim. What makes things worse is the fact that it does not change file extensions. So, in such a case the only visible symptom will be that the attacked person cannot open some of his/her files.

Newer versions also add keys to the autorun. One is to deploy a dropped copy of the program, and the other to display a ransom note in TXT format (via notepad). However, the copy of the program (DMALOCK 41:55:16:13:51:76:67:99*ntserver.exe*) – is not always dropped successfully and then only the TXT note may be displayed.



Detection

It is detected by Malwarebytes Anti-Malware as **Ransom.DMALocker**:

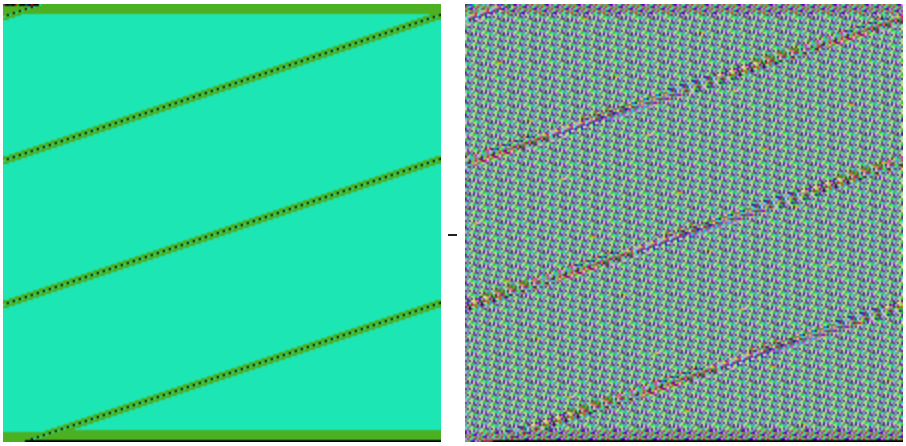


Experiment

In the ransom note, the authors mention that the data is encrypted by **AES and RSA**. Let's look at the files.

After the first look at encrypted content we can see repetitive patterns and entropy is relatively low.

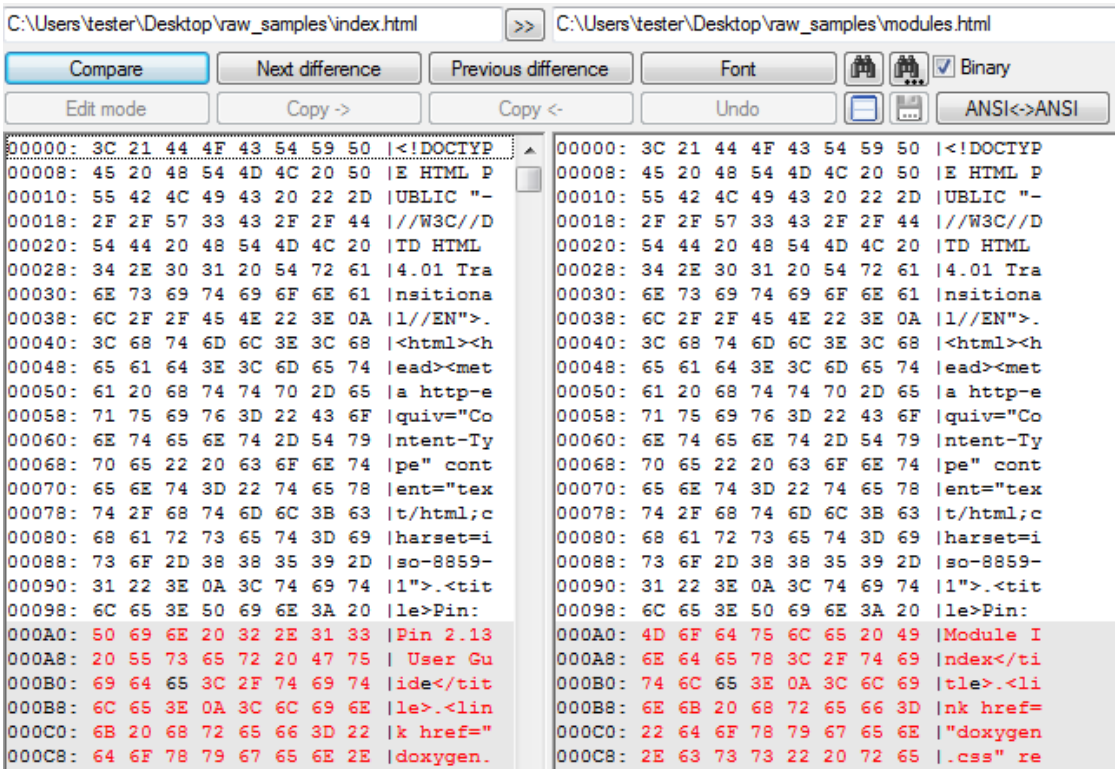
Left – raw bytes of original BMP, right – the same BMP encrypted by DMA Locker:



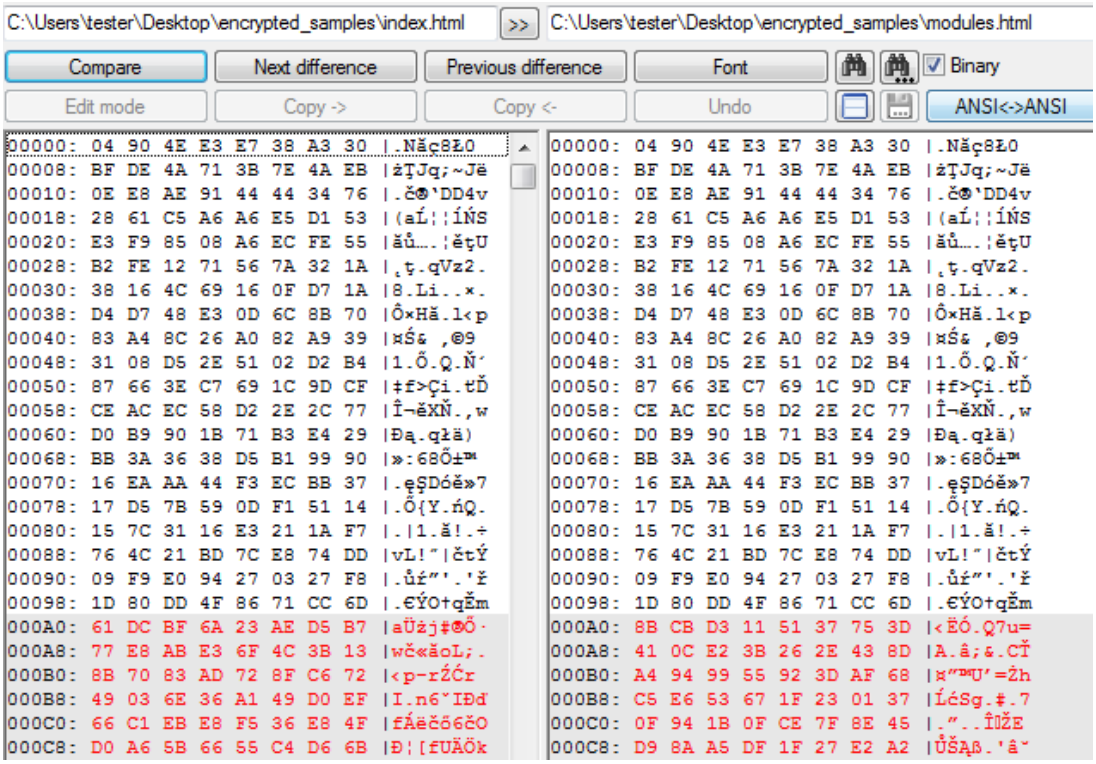
Let's compare some more files and see how they changed after being encrypted by DMA Locker.

Example 1 – HTML files:

comparison of original files:

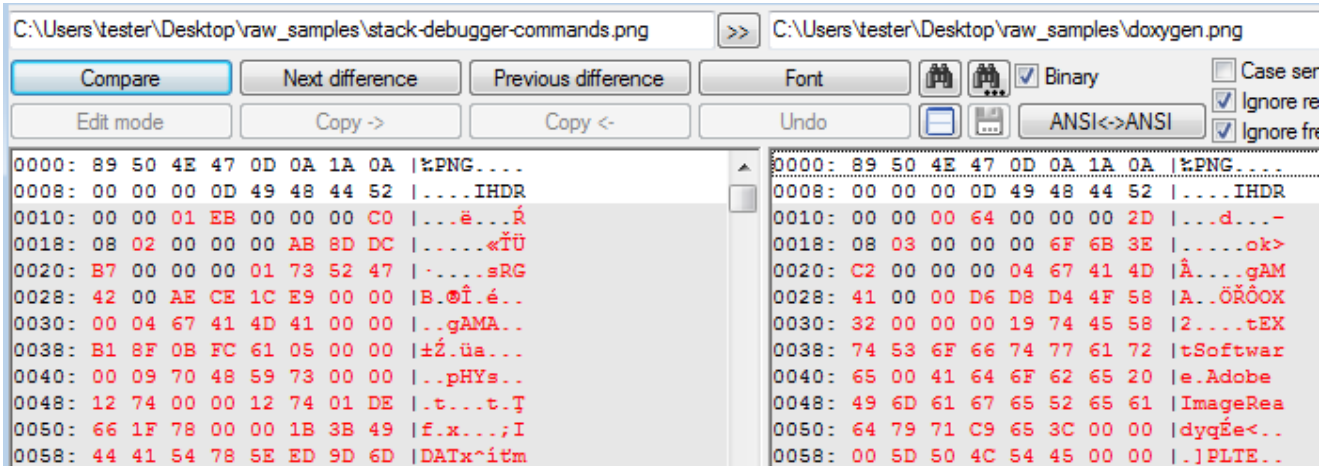


comparison of the same files encrypted:

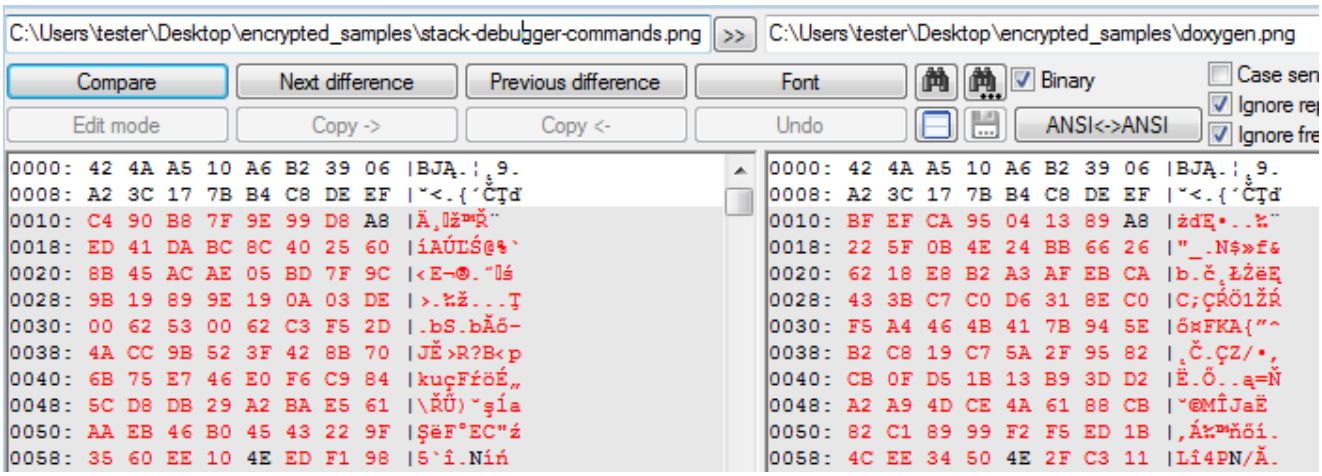


Example 2 – PNG files:

comparison of original files:



comparison of the same files encrypted:



As we can see, when the beginnings of original files are identical, the beginnings of encrypted outputs also are. But it seems that encryption is done in some chunks – possibly 8 or 16 bytes at once. Look at the comparison of PNG files – from 0x10 they have been encrypted differently – although they both have zeros at positions 0x10, 0x11…

Inside

This ransomware is distributed without any packing and no defense against analysis has been observed. All the used strings and called API functions are in plain text. In fact, the malware even “helps” the analyst by providing a lot of debug strings describing all it’s activities (original + translation):

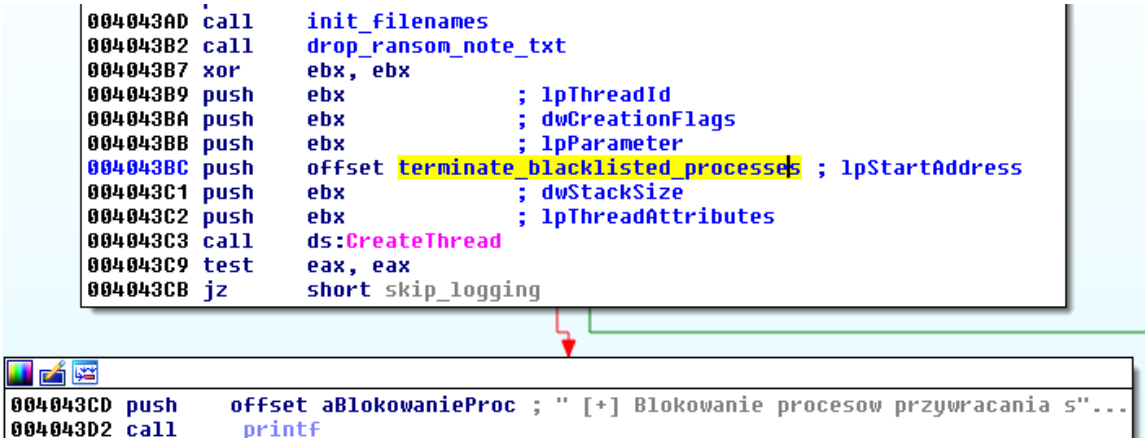
```
[+] Plik jest aktualnie zaszyfrowany, pomijanie.. //The file is already encrypted,
skipping..
[*] Rozmiar pliku = %I64d bajtow.. //File size = %I64d bytes..
[+] Rozpoczeto szyfrowanie pliku: %s //Started encrypting the file: %s
[+] Zakonczono szyfrowanie pliku: %s //Finished encrypting the file: %s
[+] Rozpoczeto zapisywanie z pamieci do pliku: %s //Started dumping from memory to a
file: %s
[+] Zakonczono zapisywanie z pamieci do pliku: %s //Finished dumping from memory to a
file: %s
[*] Plik jest aktualnie odszyfrowany, pomijanie.. //The file is already decrypted,
skipping..
[+] Rozpoczeto deszyfrowanie pliku: %s //Started decrypting file: %s
[+] Zakonczono deszyfrowanie pliku: %s //Finished decrypting file: %s
Alokacja, error: %d //Allocation error: %d
DMA Locker
Otwieranie pliku: %d //Opening file: %d
```

Thanks to the logs, finding important part of the code is trivial!

At the beginning of the execution a new thread is deployed – whose role is to check for the presence of following processes:

- rstrui.exe
- ShadowExplorer.exe
- sesvc.exe
- cbengine.exe

If any of them is detected, malware tries to terminate it. Just after deploying this thread malware logs (in Polish):
"*[+] Blocking processes of system recovery*"



Instead of a list of attacked extensions, this malware contains two blacklists. One for directories:

```
004025A3 sub     esp, 2Ch
004025A6 push    esi
004025A7 mov     [ebp+var_2C], offset aWindows ; "\\Windows\\"
004025AE mov     [ebp+var_28], offset aWindows_0 ; "\\WINDOWS\\"
004025B5 mov     [ebp+var_24], offset aProgramFiles ; "\\Program Files\\"
004025BC mov     [ebp+var_20], offset aProgramFilesX8 ; "\\Program Files (x86)\\"
004025C3 mov     [ebp+var_1C], offset aGames ; "Games"
004025CA mov     [ebp+var_18], offset aTemp ; "\\Temp"
004025D1 mov     [ebp+var_14], offset aSamplePictures ; "\\Sample Pictures"
004025D8 mov     [ebp+var_10], offset aSampleMusic ; "\\Sample Music"
004025DF mov     [ebp+var_C], offset aCache ; "\\cache"
004025E6 mov     [ebp+var_8], offset aCache_0 ; "\\Cache"
004025ED xor     esi, esi
```

and another for file extensions:

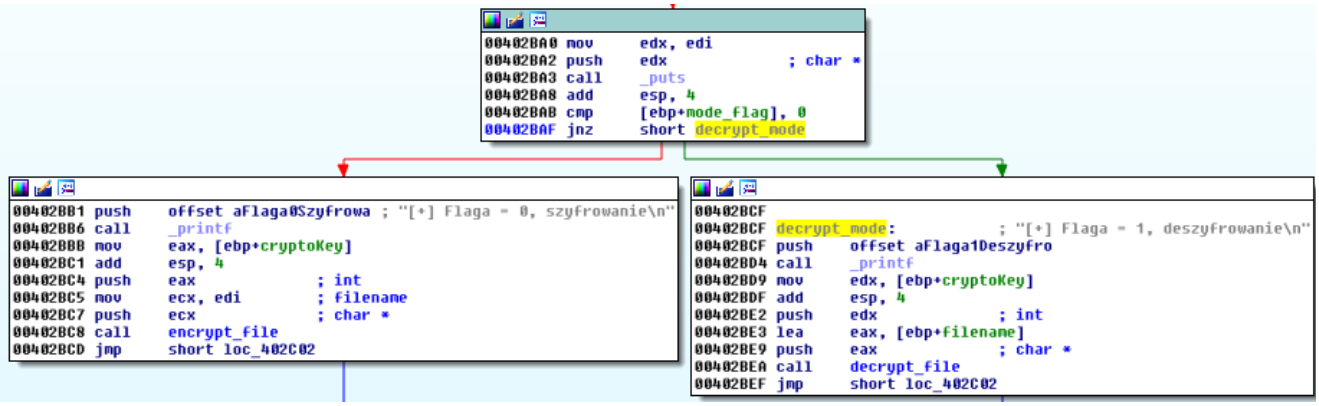
```
00402627 mov     [ebp+var_30], offset a_exe ; ".exe"
0040262E mov     [ebp+var_2C], offset a_msi ; ".msi"
00402635 mov     [ebp+var_28], offset a_dll ; ".dll"
0040263C mov     [ebp+var_24], offset a_pif ; ".pif"
00402643 mov     [ebp+var_20], offset a_scr ; ".scr"
0040264A mov     [ebp+var_1C], offset a_sys ; ".sys"
00402651 mov     [ebp+var_18], offset a_msp ; ".msp"
00402658 mov     [ebp+var_14], offset a_com ; ".com"
0040265F mov     [ebp+var_10], offset a_lnk ; ".lnk"
00402666 mov     [ebp+var_C], offset a_hta ; ".hta"
0040266D mov     [ebp+var_8], offset a_cpl ; ".cpl"
00402674 mov     [ebp+var_4], offset a_msc ; ".msc"
```

Files that contain in their path blacklisted substrings are skipped.

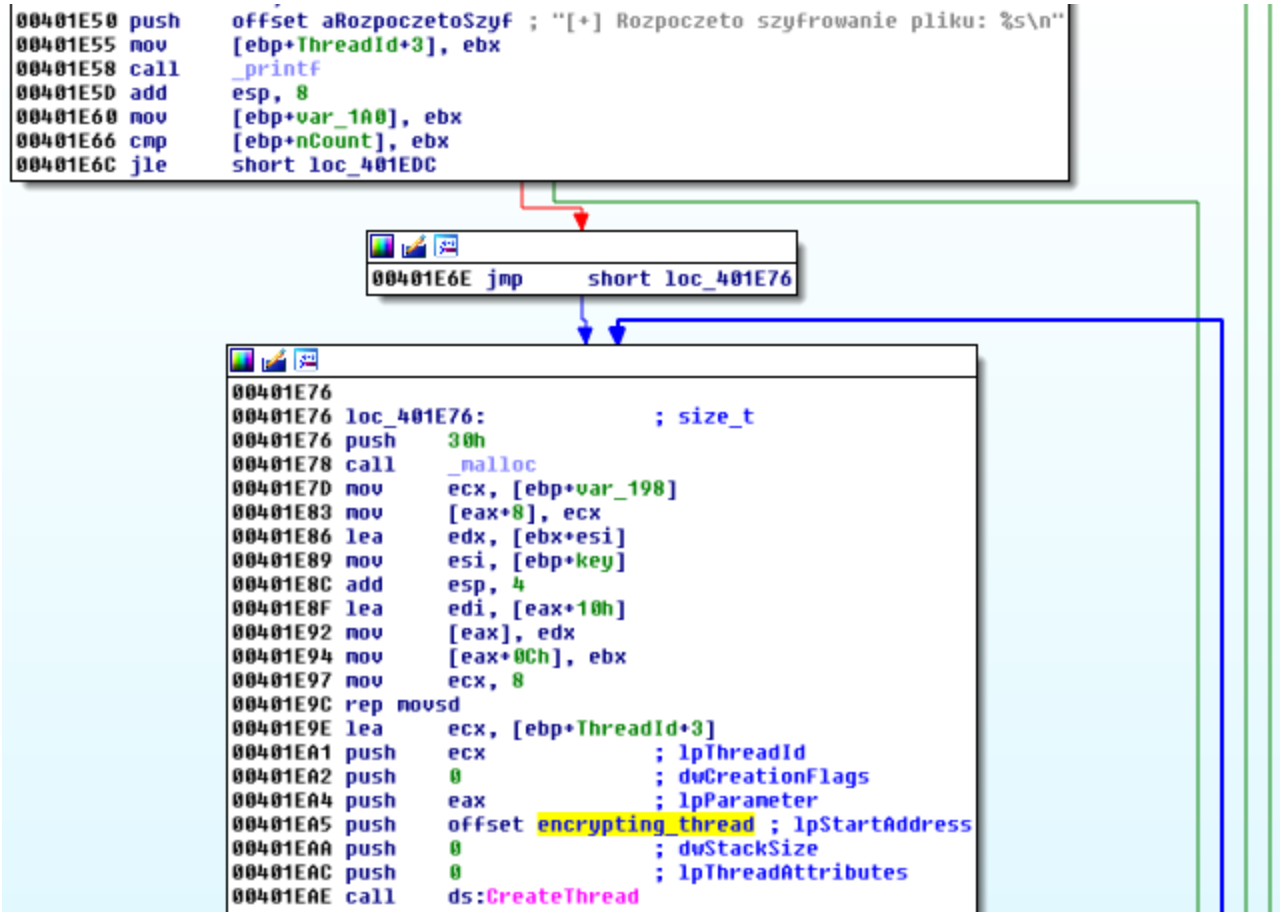
Malware enumerates all the files – browsing first logical drives, after that network resources – trying to encrypt each and every file (except the blacklisted)

```
00404640 add     esp, 0Ch
00404643 lea     edx, [esp+90h+key_buf]
00404647 push    ebx
00404648 push    edx
00404649 call    enc_dec_logical_drives
0040464E add     esp, 8
00404651 push    ebx             ; int
00404652 lea     eax, [esp+94h+key_buf]
00404656 push    eax             ; int
00404657 push    ebx             ; lpNetResource
00404658 call    enc_dec_net_resources
0040465D mov     ebx, ds:Sleep
00404663 mov     esi, 0Ah
00404668 jmp     short loc_404670
```

A single flag decides whether the malware is in encryption or decryption mode:



Encryption (as well as decryption) is deployed in a new thread:



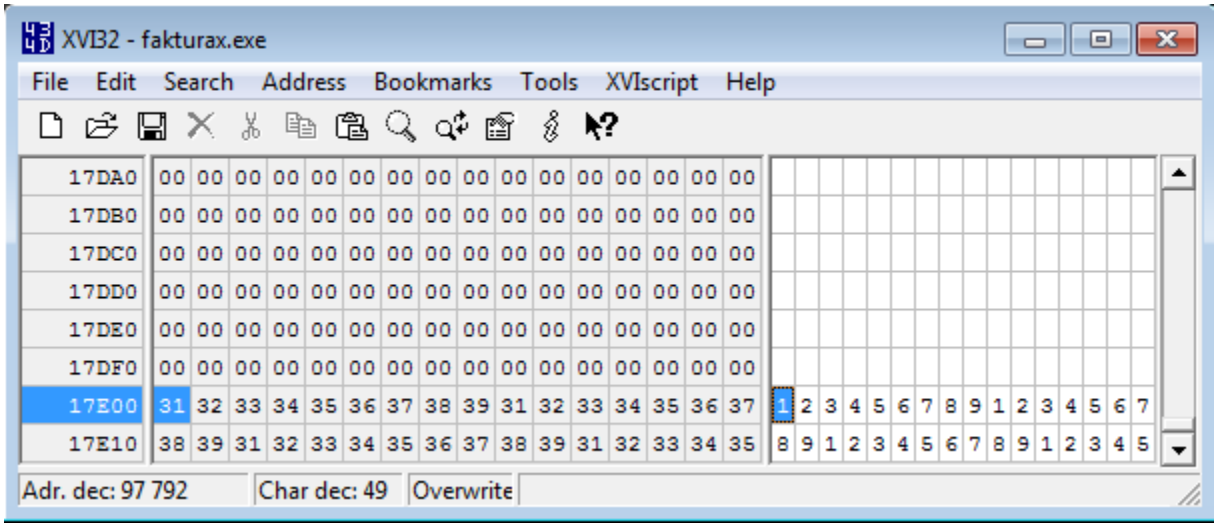
Encryption key

The encryption key is 32 byte long. In newer version of the malware it is hard-coded at the end of the original file, and then read. However, there is a twist.

During execution, two copies of the original file are dropped: **fakturax.exe** and **ntserver.exe** – but only **fakturax.exe** contains the key – **ntserver.exe** have it cleaned. After reading the key, **fakturax.exe** is removed and the key is lost along with it. That's why, we can easily

recover the key if, by any means, we managed to persist the original copy of the malware sample (it is not a problem if we know the source of infection, i.e in case if the malware arrived as an e-mail attachment).

In the examined variant of the malware (referred as the type 2, i.e 4190df2af81ece296c465e245fc0caea) – it was enough to find the key at the end of the original sample (***WARNING: this is not the original key of this sample. It has been used just to present how it works and where the real key can be found. Before trying to recover files, make sure that you made their backup, just in case if in some other editions the algorithm would be different.***)



and enter it to the text field:

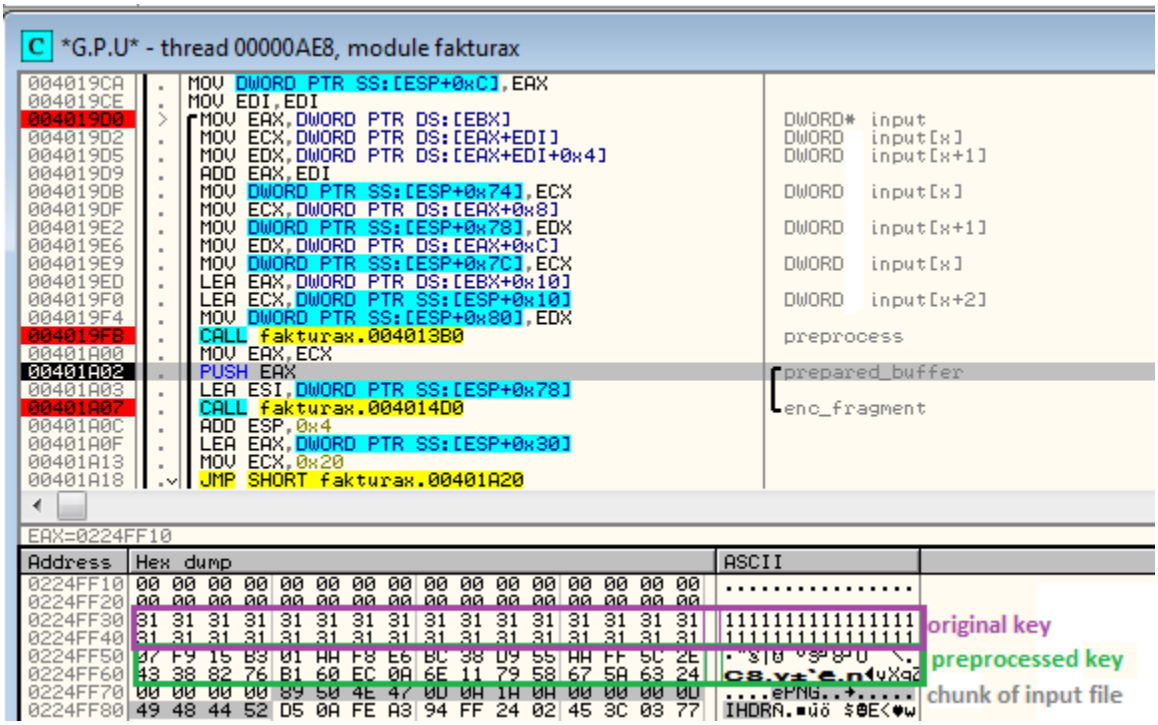
in order to get all the files back.

Encryption algorithm

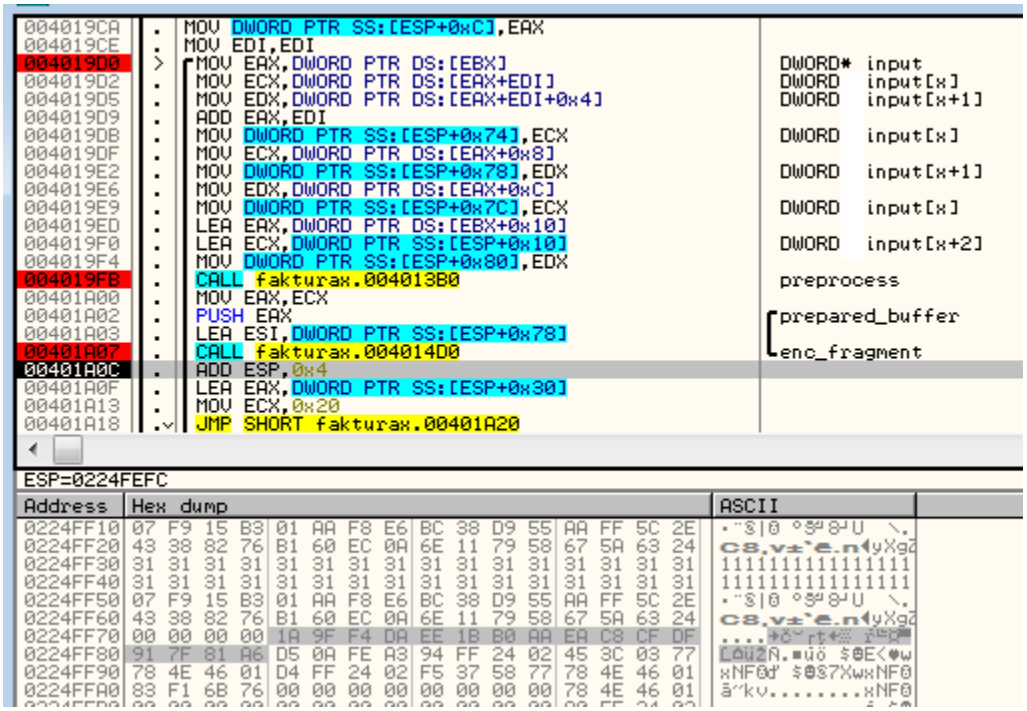
Authors claimed that they used AES and RSA. How it looks from the side of code?

File is encrypted chunk by chunk – single unit have 16 bytes (4 DWORDs). The key is 32 bytes long, and is preprocessed before the encryption. Both elements – the preprocessed key and a chunk of the input file – are copied to a buffer, that is supplied to the encrypting procedure.

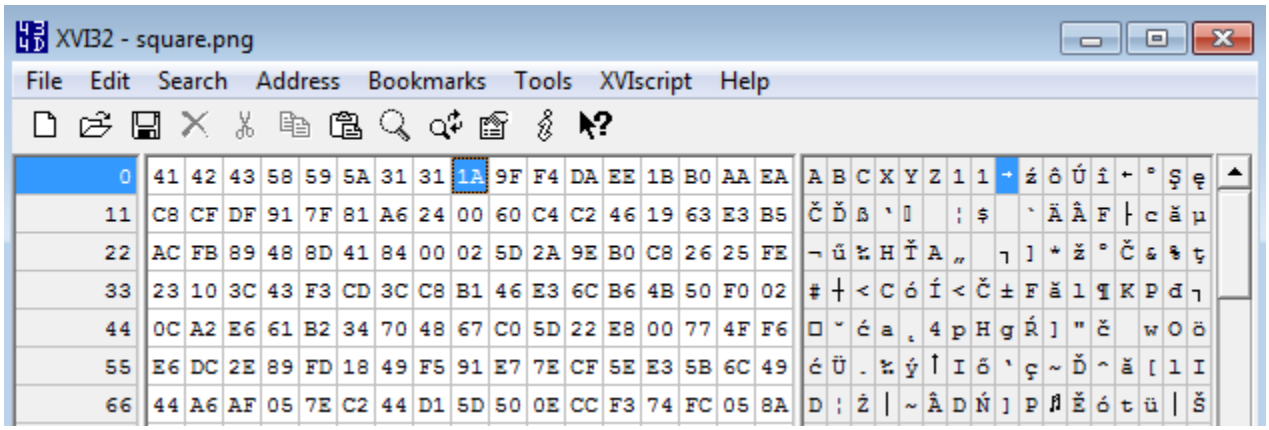
Below – a sample file: **square.png** processed by the encrypting function. Used key: “11111…”. (The copied chunk of the file has been selected on the picture)



after encryption (output marked gray):



output is then copied back to the original buffer, containing the full file. Every encrypted file has a content prefixed by “ABCXYZ11” – a magic value, used by the ransomware to recognize encrypted files (it has been introduced in the newer version). Below, we can see the sample file after being dumped on the disk.



16 byte long chunks of file are encrypted by AES in ECB mode.

Conclusion

First of all, not all what malware authors tell is true. In this case the key was neither RSA encrypted, nor randomly generated – just stored in the original file.

Second – immediately removing the malware is not always the best solution – sometimes we may need it to recover the data.

If you encountered a ransomware, it is better to try to gather information about it before taking any steps. In case you cannot find any information, the best way is to make a topic on the forum of your favorite vendor or contact some known analyst. We are in a constant search of samples of new threats, trying to describe and solve the problems.

And remember: only some families are really nasty. Other, like i.e LeChiffre have implementation flaws allowing to recover files.

Appendix

https://forum.4programmers.net/Hardware_Software/264028-dma_locker_-_zaszyfrowane_pliki – a thread on a Polish forum, created by a user infected by DMA Locker