LockerGoga

Summary

Payload is executed
Payload moves itself to %TEMP% location
Payload tries to logoff the user
Payload starts spawning itself (subprocess) to encrypt files, it also creates a mutex
After encrypting files, the payload runs net command to change the user password
Payload uses **cipher** utility to overwrite deleted areas on the disk. Normally ransomware uses vssadmin to delete the shadow copy
Payload deletes itself
User can’t login as the password is already changed

Initial commands

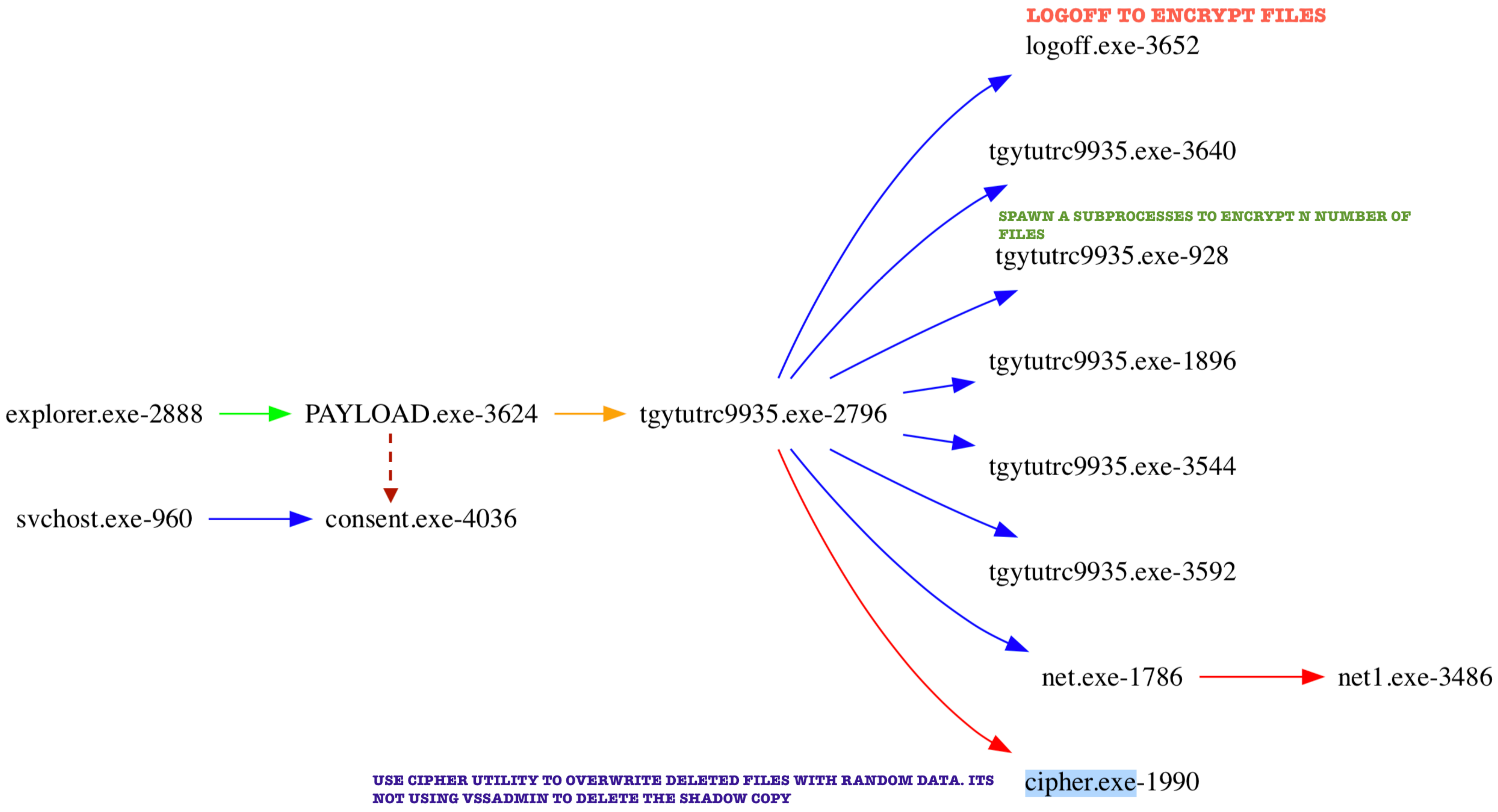
```
C:\Windows\system32\cmd.exe /c move /y C:\Users\foo\Desktop\PAYLOAD.exe
C:\Users\foo\AppData\Local\Temp\tgytutrc9935.exe  // NAMING CONVENTION: tgytutrc<RAND4DIGIT>
C:\Users\foo\AppData\Local\Temp\tgytutrc9935.exe -m
C:\Windows\system32\logoff.exe 0
C:\Users\foo\AppData\Local\Temp\tgytutrc9935.exe -i SM-tgytutrc -s
```

MUTEX

I developed a small ransomware payload in 2017 that demonstrated this technique, where the file encryption job was handled by a worker pool and parent only assigns the tasks and also the frequency for file encryption per process or thread. Click on the following link for the video

https://www.youtube.com/watch?v=whPeRFMBhzY

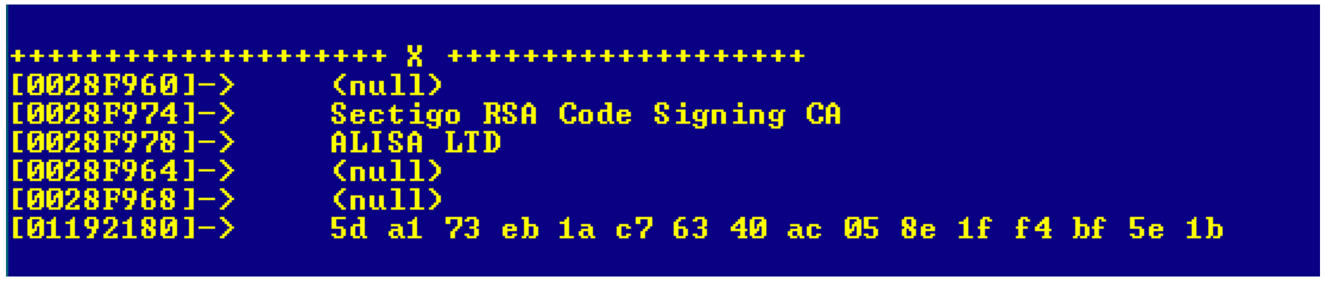
Flow for the impatient



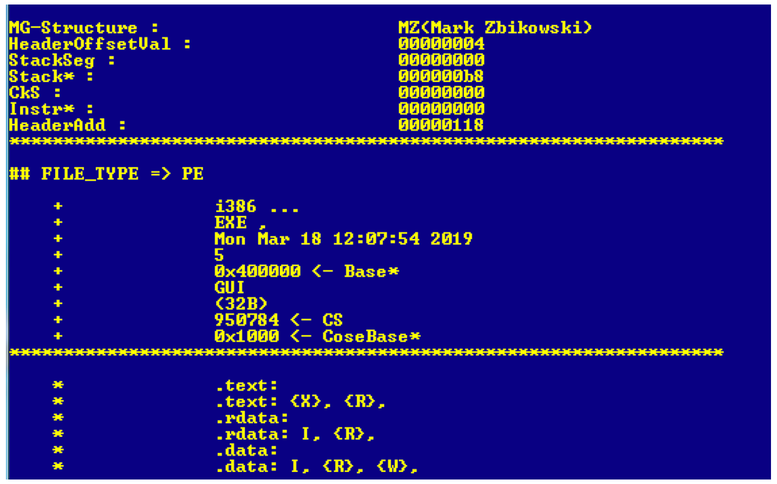
More info

Story of malware is within the payload itself, so let's begin

Back in january, hackers infected another company in France with a very similar payload. Payload used a spawned process (spawning itself with a different command line) to encrypt files. This payload is using similar technique but this time its using a signed executable, must be a stolen cert.



```
++++++++++++++++++++ X +++++++++++++++++++
[0028F960]->     (null)
[0028F974]->     Sectigo RSA Code Signing CA
[0028F978]->     ALISA LTD
[0028F964]->     (null)
[0028F968]->     (null)
[01192180]->     5d a1 73 eb 1a c7 63 40 ac 05 8e 1f f4 bf 5e 1b
```

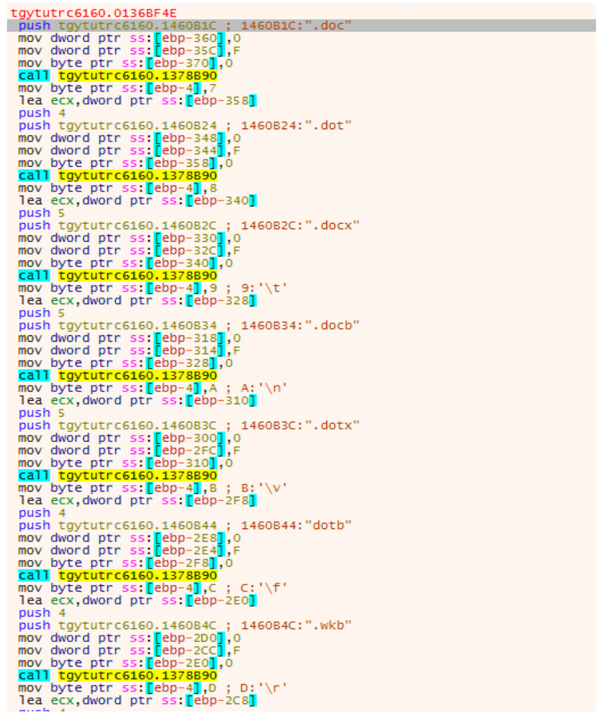


```
MG-Structure :                    MZ(Mark Zbikowski)
HeaderOffsetVal :                 00000004
StackSeg :                        00000000
Stack* :                          000000b8
CkS :                             00000000
Instr* :                          00000000
HeaderAdd :                       00000118
****************************************************************************

## FILE_TYPE => PE

    +               i386 ...
    +               EXE ,
    +               Mon Mar 18 12:07:54 2019
    +               5
    +               0x400000 <- Base*
    +               GUI
    +               (32B)
    +               950784 <- CS
    +               0x1000 <- CoseBase*
****************************************************************************

    *               .text:
    *               .text: (X), (R),
    *               .rdata:
    *               .rdata: I, (R),
    *               .data:
    *               .data: I, (R), (W),
```

I don't know much about the propagation component of the attack but one could assume that the attacker stole the AD database along with the required hives.

Payload is looking for the following file extensions

```
.doc,.dot,.docx,.docb,.dotx,dotb,.wkb,.xml,.xls,.xlsx,.xlt,.xltx,.xlsb,.xlw,.ppt,.pps,.pot
.ppsx,.pptx,.posx,.potx,.sldx,.pdf,.sql
```



```
tgytutrc6160.0136BF4E
 push tgytutrc6160.1460B1C ; 1460B1C:".doc"
 mov dword ptr ss:[ebp-360],0
 mov dword ptr ss:[ebp-35C],F
 mov byte ptr ss:[ebp-370],0
 call tgytutrc6160.1378B90
 mov byte ptr ss:[ebp-4],7
 lea ecx,dword ptr ss:[ebp-358]
 push 4
 push tgytutrc6160.1460B24 ; 1460B24:".dot"
 mov dword ptr ss:[ebp-348],0
 mov dword ptr ss:[ebp-344],F
 mov byte ptr ss:[ebp-358],0
 call tgytutrc6160.1378B90
 mov byte ptr ss:[ebp-4],8
 lea ecx,dword ptr ss:[ebp-340]
 push 5
 push tgytutrc6160.1460B2C ; 1460B2C:".docx"
 mov dword ptr ss:[ebp-330],0
 mov dword ptr ss:[ebp-32C],F
 mov byte ptr ss:[ebp-340],0
 call tgytutrc6160.1378B90
 mov byte ptr ss:[ebp-4],9 ; 9:'\t'
 lea ecx,dword ptr ss:[ebp-328]
 push 5
 push tgytutrc6160.1460B34 ; 1460B34:".docb"
 mov dword ptr ss:[ebp-318],0
 mov dword ptr ss:[ebp-314],F
 mov byte ptr ss:[ebp-328],0
 call tgytutrc6160.1378B90
 mov byte ptr ss:[ebp-4],A ; A:'\n'
 lea ecx,dword ptr ss:[ebp-310]
 push 5
 push tgytutrc6160.1460B3C ; 1460B3C:".dotx"
 mov dword ptr ss:[ebp-300],0
 mov dword ptr ss:[ebp-2FC],F
 mov byte ptr ss:[ebp-310],0
 call tgytutrc6160.1378B90
 mov byte ptr ss:[ebp-4],B ; B:'\v'
 lea ecx,dword ptr ss:[ebp-2F8]
 push 4
 push tgytutrc6160.1460B44 ; 1460B44:"dotb"
 mov dword ptr ss:[ebp-2E8],0
 mov dword ptr ss:[ebp-2E4],F
 mov byte ptr ss:[ebp-2F8],0
 call tgytutrc6160.1378B90
 mov byte ptr ss:[ebp-4],C ; C:'\f'
 lea ecx,dword ptr ss:[ebp-2E0]
 push 4
 push tgytutrc6160.1460B4C ; 1460B4C:".wkb"
 mov dword ptr ss:[ebp-2D0],0
 mov dword ptr ss:[ebp-2CC],F
 mov byte ptr ss:[ebp-2E0],0
 call tgytutrc6160.1378B90
 mov byte ptr ss:[ebp-4],D ; D:'\r'
 lea ecx,dword ptr ss:[ebp-2C8]
```

Other than those, it can encrypt executables as well. It has the ability to remove file locks and encrypt some of the running executable images as well. Matrix ransomware also used this technique by using a sysinternal tool. For more info on matrix ransomware click on the following link

http://udurrani.com/exp0/matrix_ransomware.pdf

Stage 1 tries to logoff the user

```
CreateProcessW ( "C:\Windows\system32\logoff.exe", "C:\Windows\system32\logoff.exe 0", NULL,
NULL, FALSE, 0, NULL, NULL, … )
```

Stage 1 initiates / spawns a subprocess to start encryption

```
CreateProcessA ( "C:\Users\foo\AppData\Local\Temp\tgytutrc3003.exe", "C:
\Users\foo\AppData\Local\Temp\tgytutrc3003.exe -i SM-tgytutrc -s ", NULL, NULL, FALSE, 0,
NULL, NULL, … )
```

Subprocess looks for the mutex passed to it as an argument

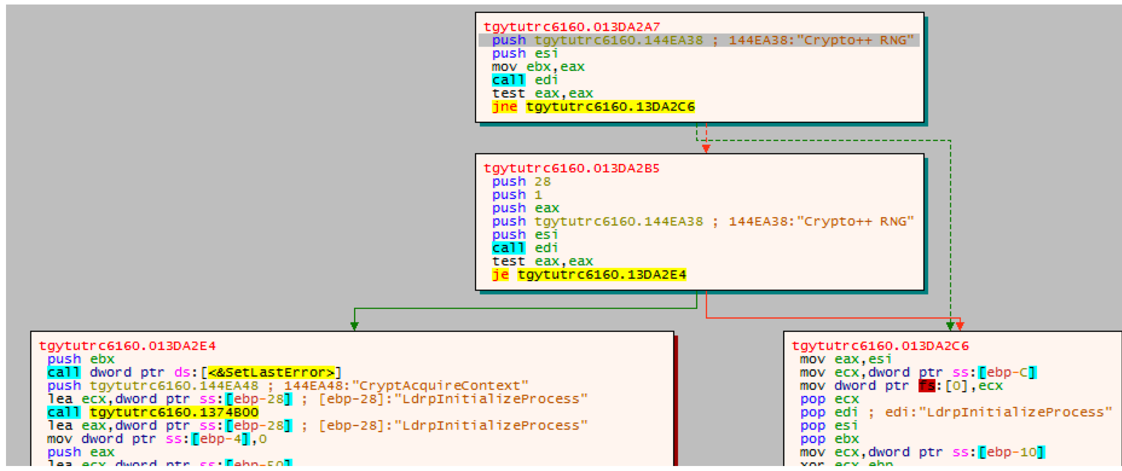
Once the subprocess is initiated, it creates a mutex (if not present in the memory). If its already there the payload will exit

```
CreateMutexA(0x0, 0x0, “MX-tgytutrc")        // Returns a HANDLE


ReleaseMutex(HANLDE)     // expects a non-zero value
```

Encryption

It uses C++ crypto++ to encrypt all files.

```
memset(key, zero-it-out, sizeof(key));
memset(iv, zero-it-out, sizeof(iv));
DataStruct( key, sizeof(key), iv);

StartEncryption [STREAM] and pass the struct and file
```



```
tgytutrc6160.013DA2A7
 push tgytutrc6160.144EA38 ; 144EA38:"Crypto++ RNG"
 push esi
 mov ebx,eax
 call edi
 test eax,eax
 jne tgytutrc6160.13DA2C6

tgytutrc6160.013DA2B5
 push 28
 push 1
 push eax
 push tgytutrc6160.144EA38 ; 144EA38:"Crypto++ RNG"
 push esi
 call edi
 test eax,eax
 je tgytutrc6160.13DA2E4

tgytutrc6160.013DA2E4
 push ebx
 call dword ptr ds:[<&SetLastError>]
 push tgytutrc6160.144EA48 ; 144EA48:"CryptAcquireContext"
 lea ecx,dword ptr ss:[ebp-28] ; [ebp-28]:"LdrpInitializeProcess"
 call tgytutrc6160.1374B00
 lea eax,dword ptr ss:[ebp-28] ; [ebp-28]:"LdrpInitializeProcess"
 mov dword ptr ss:[ebp-4],0
 push eax

tgytutrc6160.013DA2C6
 mov eax,esi
 mov ecx,dword ptr ss:[ebp-C]
 mov dword ptr fs:[0],ecx
 pop ecx
 pop edi ; edi:"LdrpInitializeProcess"
 pop esi
 pop ebx
 mov ecx,dword ptr ss:[ebp-10]
```

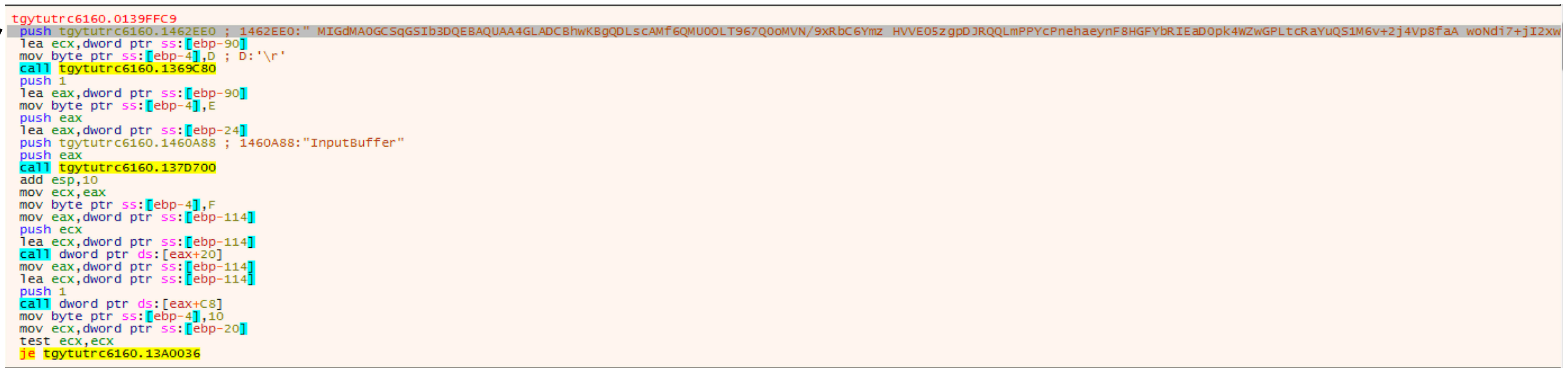
InterProcess Communication

IPC is using sharedMemory via CreateFileMapping(). It uses ***0xffffffff*** or simply ***INVALID_HANDLE_VALUE* as the 1st argument.**

```
CreateFileMappingA(0xffffffff, …); // FileMapping via paging file
```

The spawned process uses ***OpenFileMappingA***() to get info from the sharedMemory and encrypt file(s). Both processes will share the same page of physical storage. In simple words, the attacker has mapped the same file into 2 address spaces, however both these processes are using the same page of physical storage.

Subprocess uses the library called Crypto++ to encrypt files. Its stream based encryption i.e. it will act on stream of data. It uses **NtReadFile**(), acts on the data as streams and then uses **NtWriteFile**() to modify data.



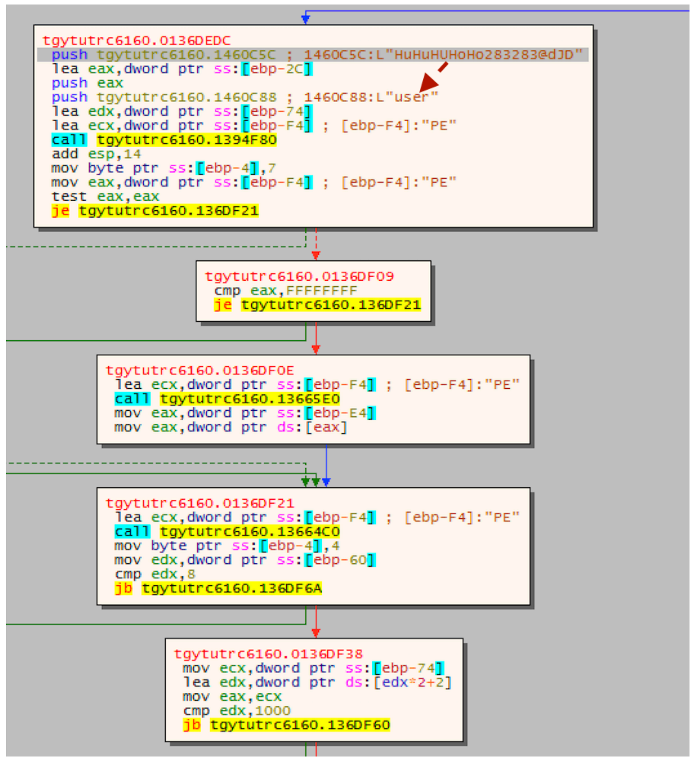
```
tgytutrc6160.0139FFC9
push tgytutrc6160.1462EE0 ; 1462EE0:" MIGdMA0GCSqGSIb3DQEBAQUAA4GLADCBhwKBgQDLscAMf6QMU0OLT967Q0oMVN/9xRbC6Ymz HVVE05zgpDJRQQLmPPYcPnehaeynF8HGFYbRIEaD0pk4WZwGPLtcRaYuQS1M6v+2j4Vp8faA woNdi7+jI2xw
lea ecx,dword ptr ss:[ebp-90]
mov byte ptr ss:[ebp-4],D ; D:'\r'
call tgytutrc6160.1369C80
push 1
lea eax,dword ptr ss:[ebp-90]
mov byte ptr ss:[ebp-4],E
push eax
lea eax,dword ptr ss:[ebp-24]
push tgytutrc6160.1460A88 ; 1460A88:"InputBuffer"
push eax
call tgytutrc6160.137D700
add esp,10
mov ecx,eax
mov byte ptr ss:[ebp-4],F
mov eax,dword ptr ss:[ebp-114]
push ecx
lea ecx,dword ptr ss:[ebp-114]
call dword ptr ds:[eax+20]
mov eax,dword ptr ss:[ebp-114]
lea ecx,dword ptr ss:[ebp-114]
push 1
call dword ptr ds:[eax+C8]
mov byte ptr ss:[ebp-4],10
mov ecx,dword ptr ss:[ebp-20]
test ecx,ecx
je tgytutrc6160.13A0036
```

```
NtOpenFile -> if_not_useful -> NtCLose

ELSE:

NtReadFile -> NtWriteFile


FUNC_("MIGdMA0GCSqGSIb3DQEBAQUAA4GLADCBhwKBgQDLscAMf6QMU0OLT967Q0oMVN/9xRbC6Ymz HVVE05zgpDJRQQLmPPYcPnehaeynF8HGFYbRIEaD0pk4WZwGPLtcRa
YuQS1M6v+2j4Vp8faA woNdi7+jI2xw0kQao29FJ8WUQDvrPqODALf8bjiOIO7f1Nc5g9vOEbWyCA1w/vbaVwIBEQ==", 0xdb, 0x0);

CHANGE EXTENSION TO (u".locked")
```

Base64 text shown above is written to the sharedMemory buffer by the parent process, while the subProcess reads the path and encrypts the file(s). In the process of encryption, parent process spawns multiple sub processes.

Once the data is encrypted, the parent process uses net command to change the user password. Password is passed as w_char

```
u"HuHuHUHoHo283283@dJD"
```



```
tgytutrc6160.0136DEDC
 push tgytutrc6160.1460C5C ; 1460C5C:L"HuHuHUHoHo283283@dJD"
 lea eax,dword ptr ss:[ebp-2C]
 push eax
 push tgytutrc6160.1460C88 ; 1460C88:L"user"
 lea edx,dword ptr ss:[ebp-74]
 lea ecx,dword ptr ss:[ebp-F4] ; [ebp-F4]:"PE"
 call tgytutrc6160.1394F80
 add esp,14
 mov byte ptr ss:[ebp-4],7
 mov eax,dword ptr ss:[ebp-F4] ; [ebp-F4]:"PE"
 test eax,eax
 je tgytutrc6160.136DF21

tgytutrc6160.0136DF09
 cmp eax,FFFFFFFF
 je tgytutrc6160.136DF21

tgytutrc6160.0136DF0E
 lea ecx,dword ptr ss:[ebp-F4] ; [ebp-F4]:"PE"
 call tgytutrc6160.13665E0
 mov eax,dword ptr ss:[ebp-E4]
 mov eax,dword ptr ds:[eax]

tgytutrc6160.0136DF21
 lea ecx,dword ptr ss:[ebp-F4] ; [ebp-F4]:"PE"
 call tgytutrc6160.13664C0
 mov byte ptr ss:[ebp-4],4
 mov edx,dword ptr ss:[ebp-60]
 cmp edx,8
 jb tgytutrc6160.136DF6A

tgytutrc6160.0136DF38
 mov ecx,dword ptr ss:[ebp-74]
 lea edx,dword ptr ds:[edx*2+2]
 mov eax,ecx
 cmp edx,1000
 jb tgytutrc6160.136DF60
```

Function is called to run the net.exe command, where first function parameter is the command name

```
FUNCTION_TO_CALL_NET_COMMAND("net.exe", 0x7);
```

Payload is able to disable network interfaces as well. First it retrieves adapter info i.e. depending on the family e.g. AF_INET, AF_INET6 etc. It allocates a buffer and calls GetAdaptorAddress. This function returns an int value. Address information is retrieved from the structure passed to the function i.e. IP_ADAPTER_ADDRESSES. This is a linked List.

```
if (GetAdaptersAddresses(0x0, 0x1c, 0x0, POINTER_TO_BUFFER, &val) == 0x0)   // 0x0 means ipv4 || 6.
```

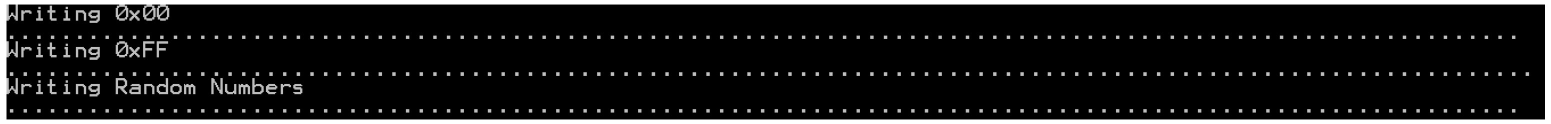
$\downarrow$

```
FUNC_1("netsh.exe", 0x9);
FUNC_2("interface", …, "interface", edi, "DISABLED", &val, …);
```

Deleted Files

Most ransomware uses VSSADMIN to delete the shadow copies. This payload calls a CIPHER utility. This utility will write random date to deleted sections.

c:\windows\system32\cipher.exe /w: C:\



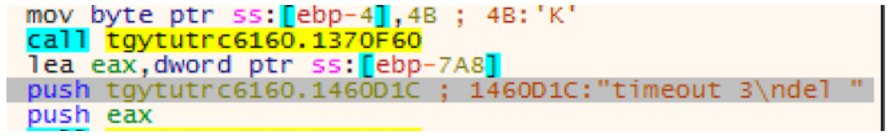
```
Writing 0x00
.....................................................................................................
Writing 0xFF
.....................................................................................................
Writing Random Numbers
.....................................................................................................
```

Let me explain this a bit. OS keeps track of all the files through a pointer. Think of it as a mark that tells the OS where the file begins and where it ends. What if you delete this pointer???
FileSystem will think there is no file at that location. NOTE: Only the pointer is removed, not the file itself.

If you want to remove the file traces, you need to overwrite that sector. If you don't try to retrieve the deleted files instantly, any random data could overwrite that location in the sector. This is what the cipher.exe (Microsoft utility) is doing for you. This could also be done by writing a tool that writes random data to the disk but it won't be 100%. The more random data you write the more chances to overwrite the file(s)

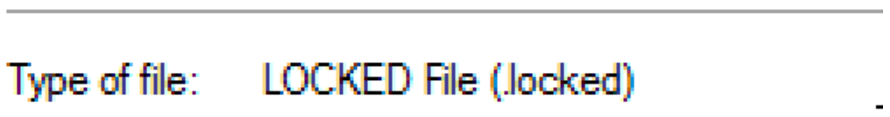
At the end, the payload runs a .cmd file thats sitting in the %TEMP% location.
payloadName.exe.cmd

This /cmd file contains few things like sleep, delete the *.cmd script etc.



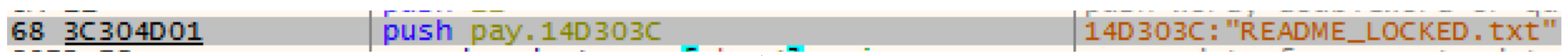
```
mov byte ptr ss:[ebp-4],4B ; 4B:'K'
call tgytutrc6160.1370F60
lea eax,dword ptr ss:[ebp-7A8]
push tgytutrc6160.1460D1C ; 1460D1C:"timeout 3\ndel "
push eax
```

All files are encrypted with a .locked extension

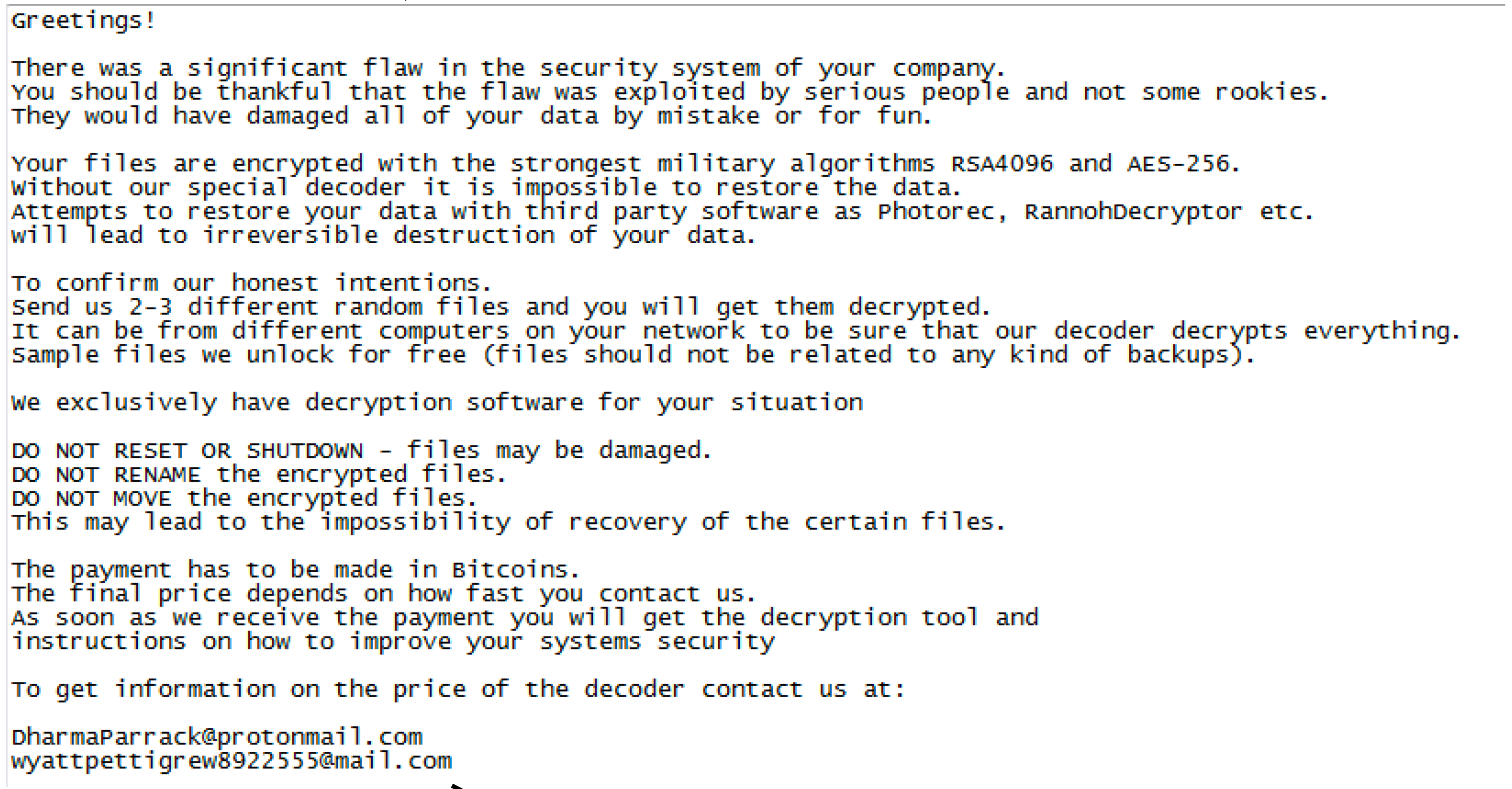


Type of file: LOCKED File (.locked)

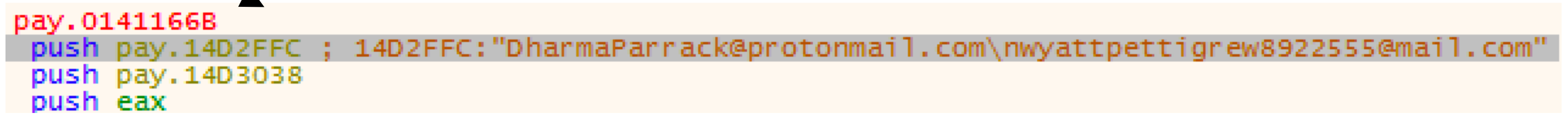
Ransom note



```
68 3C304D01       push pay.14D303C       14D303C:"README_LOCKED.txt"
```



```
Greetings!

There was a significant flaw in the security system of your company.
You should be thankful that the flaw was exploited by serious people and not some rookies.
They would have damaged all of your data by mistake or for fun.

Your files are encrypted with the strongest military algorithms RSA4096 and AES-256.
Without our special decoder it is impossible to restore the data.
Attempts to restore your data with third party software as Photorec, RannohDecryptor etc.
will lead to irreversible destruction of your data.

To confirm our honest intentions.
Send us 2-3 different random files and you will get them decrypted.
It can be from different computers on your network to be sure that our decoder decrypts everything.
Sample files we unlock for free (files should not be related to any kind of backups).

We exclusively have decryption software for your situation

DO NOT RESET OR SHUTDOWN - files may be damaged.
DO NOT RENAME the encrypted files.
DO NOT MOVE the encrypted files.
This may lead to the impossibility of recovery of the certain files.

The payment has to be made in Bitcoins.
The final price depends on how fast you contact us.
As soon as we receive the payment you will get the decryption tool and
instructions on how to improve your systems security

To get information on the price of the decoder contact us at:

DharmaParrack@protonmail.com
wyattpettigrew8922555@mail.com
```



```
pay.0141166B
  push pay.14D2FFC ; 14D2FFC:"DharmaParrack@protonmail.com\nwyattpettigrew8922555@mail.com"
  push pay.14D3038
  push eax
```

Conclusion

- ☑ *Hire smart folks*
- ☑ *Secure your AD*

STAY AWAY FROM RANSOMWARE