Ordinypt Malware Analysis

By

Valthek

The last days appear a new ransomware called "Ordinypt" that targets German users. This report talks about this malware and the reversing session to discover the truth about this malware.

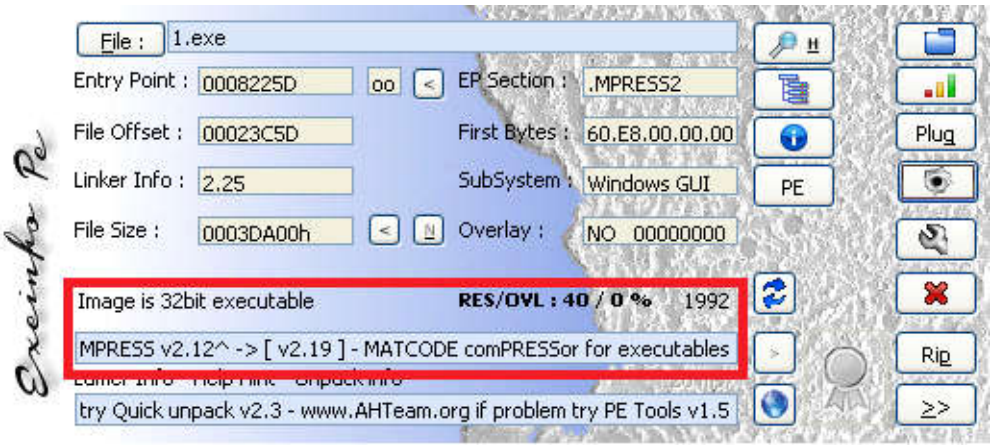
Ordinypt appears be a ransomware, but is a wiper. As you will see the malware requests money for recover the files but the files are destroyed without any chance to recover it from the malware creators.

So, lets start talking about this sample.

The sample with hash "**3ba3272977dfe0d1a0c48ef6cb9a06b2**" will be the sample used for this analysis.

I. LOOKING THE MALWARE

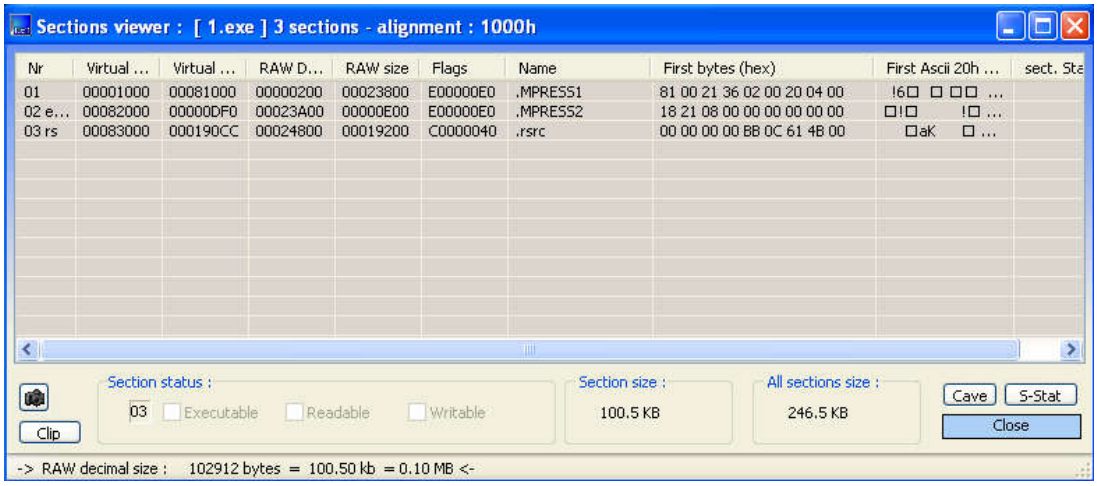
The first action is look if the malware is packed or not. I use ExeInfo but you can use PeID or another tool.



ExeInfo told us that is packed with MPRESS 2.12, so we can use some tool as the program said us or try unpack manually.

Lets unpack it manually to learn about this packer.

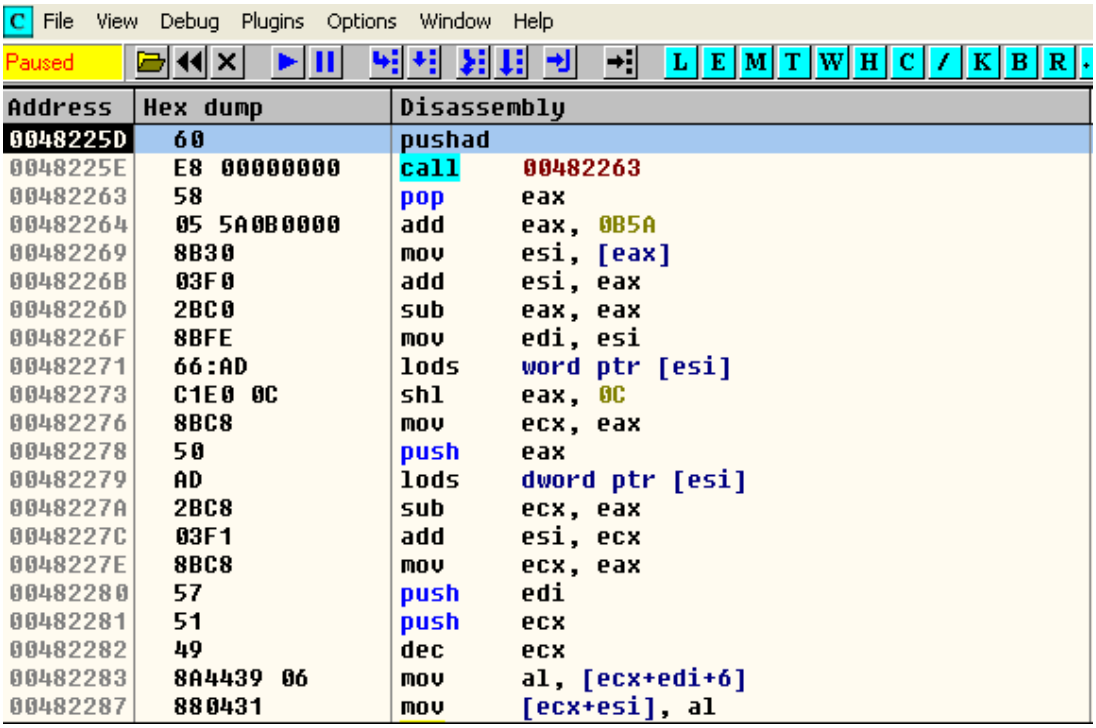
Looking the sections we can found this:



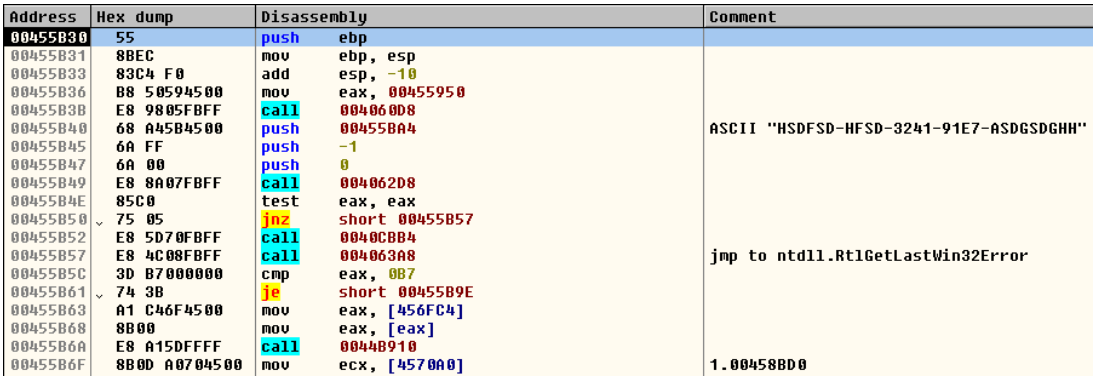
The first one section called ".MPRESS1" have a size of 142KB, the second one called ".MPRESS2" have 3.5KB and the last one 100.5KB.

The last one don't have the Executable flag, so its not have the Entry Point to the program (its have the resources of the binary). ExeInfo said us the EntryPoint is at the offset in memory "0x8225D", so, looking the sections we discover that the entry point is in the second section.

Lets open in Olly the binary.



We are in the second section, so, lets start debugging all this code. After debugging all this packer code finally arrive to the offset 0x455b30:



This point is the beginning of malware. The tricks that you can use to debug the packer without any problem and quickly are:

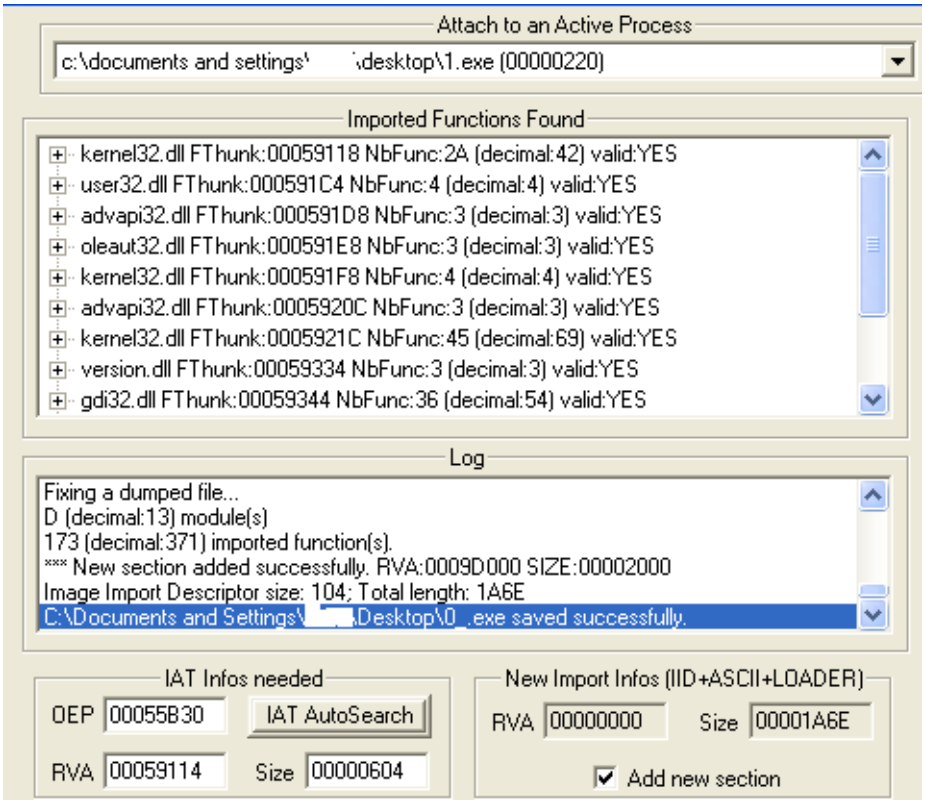
- Use hardware breakpoints to avoid loops. With this packer you can use normal breakpoints but is useful learn to use hardware instead the normal ones because some programs can use his own code to unpack, make a key, or whatever think. A normal breakpoint puts a byte for the opcode “int3” 0xCC in the memory that you want break, so this memory is altered and this

programs can detect or have a wrong operation.

- Look the jumps to keep all tracks under your control and put hardware breakpoints in the addresses that finish some loops.
- The packer use a lot of time calls to the next instruction following the call so is easy keep the track.
- The packer gets some exports from kernel32.dll as DeleteCriticalSection, etc. You can put a bp in GetModuleHandleA or GetProcAddress to help you.
- Finally the packer uses a jmp to reach the original binary unpacked in memory.

So, with the malware unpacked in memory lets dump it to debugging, in the offset 0x455B30 use the plugin of OllyDump (or another one that makes the same) but remember uncheck the rebuild imports option.

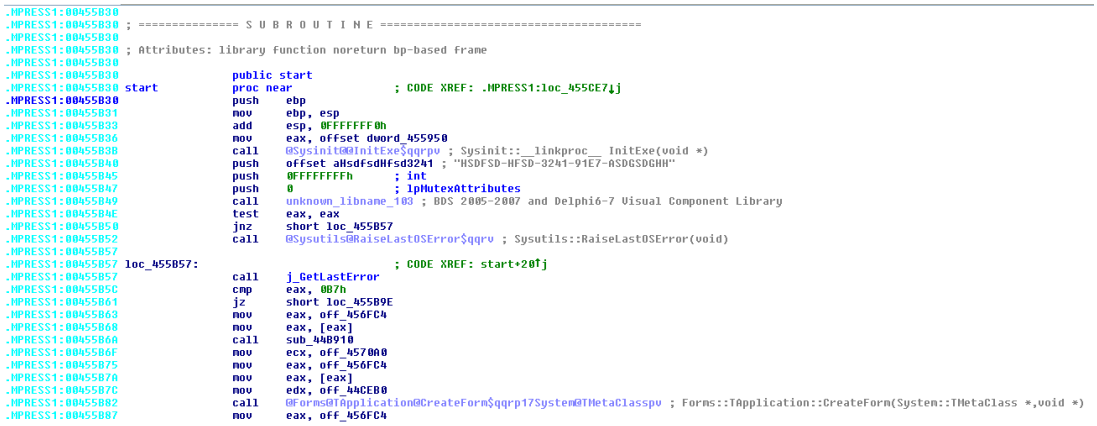
Now we have a binary with 628KB but its have a problem, the IAT is corrupted, so lets fix it with ImportREC.



Fix the dump, and finally gets the malware with a size of 632KB (or 636KB) and with the IAT fixed and ready to run without any packer.

The sections names remains as the name of the packer but don't care about it.

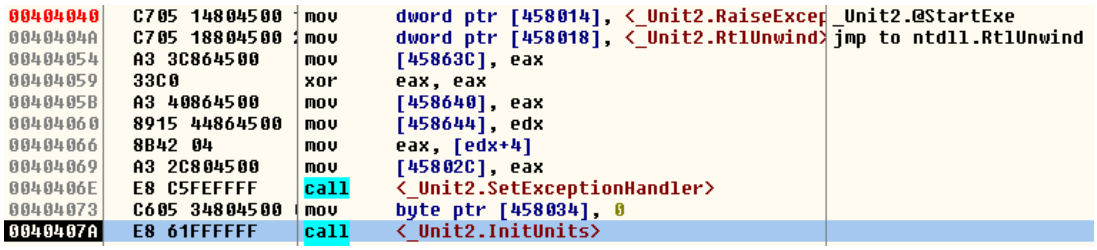
Now lets open the malware using IDA to discover that this wiper is writed in Delphi 7.



IDA can help us with Delphi but is better use "Interactive Delphi Reconstructor", so lets open too the malware in this program and wait that its analyze all the code.

When its finish select the option "Tools->Map Generator" or "Tools->IDC Generator". In my case I use a map file, and with the plugin in Olly of "Load Map" I load this map file to apply all info in the debugging session that Olly will save in the UDD file for later if I needed.

The malware payload remains in the unit1.pas file in the code, so, lets start debugging to reach the function called "_Unit2.InitUnits". In this function the code called a register in a big loop, we need wait that enters in the function called "Unit1.Initialization" that is the function that starts the malware code. The anothers calls belongs to Delphi starting his own libraries and units.



Finally we reached in the function that starts the malware in the offset "0x455194".

```
53              push    ebx                             Unit1.Initialization
56              push    esi
57              push    edi
BB D48B4500     mov     ebx, 00458BD4
BE DC8B4500     mov     esi, 00458BDC
BF E48B4500     mov     edi, 00458BE4
832D BC8C4500   sub     dword ptr [458CBC], 1
0F83 90050000   jnb     00455743
6A 00           push    0
A1 C46F4500     mov     eax, [456FC4]
8B00            mov     eax, [eax]
8B40 30         mov     eax, [eax+30]
50              push    eax
E8 7B19FBFF     call    <Windows.ShowWindow>            jmp to user32.ShowWindow
57              push    edi
8BCE            mov     ecx, esi
8BD3            mov     edx, ebx
B8 50574500     mov     eax, 00455750                   ASCII "A:"
E8 587EFFFF     call    <Unit1.sub_0044D02C>
```

The first action of the malware is check if have free space the disk of all logic units starting from “A:” to “Z:” using the API “GetDiskFreeSpaceExA”.

In the case that some unit exists its will save a in a global var the free space of this unit.

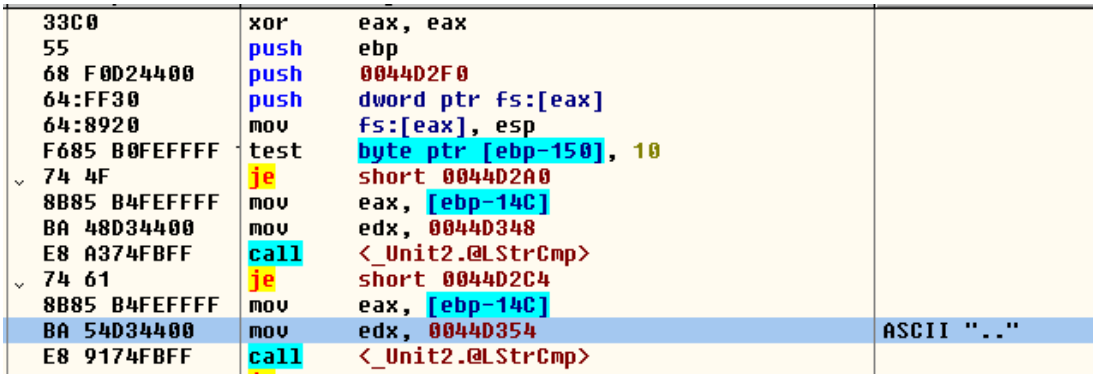
After this the malware access to the units starting from “C:\” to “P:\”.

Starting with “C:\” the malware concats the string of the unit with “*” to search for all files and directories in this unit.

```
898D A4FEFFFF   mov     [ebp-15C], ecx
8BF2            mov     esi, edx
8BD8            mov     ebx, eax
8D85 A8FEFFFF   lea     eax, [ebp-158]
8B15 78714000   mov     edx, [407178]                   0_.0040717C
E8 907AFBFF     call    <_Unit2.@InitializeRecord>
33C0            xor     eax, eax
55              push    ebp
68 26D34400     push    0044D326
64:FF30         push    dword ptr fs:[eax]
64:8920         mov     fs:[eax], esp
8D85 A4FEFFFF   lea     eax, [ebp-15C]
B9 3CD34400     mov     ecx, 0044D33C                   ASCII "\*"
8BD3            mov     edx, ebx
E8 E873FBFF     call    <_Unit2.@LStrCat3>
8B85 A4FEFFFF   mov     eax, [ebp-15C]

EAX 0012FE10
ECX 0044D33C ASCII "\*"
EDX 00455888 ASCII "C:\"
EBX 00455888 ASCII "C:\"
ESP 0012FDF4
EBP 0012FF6C
ESI 0044D364 <0_.Unit1.sub_0044D364>
EDI 00458BE4 0_.00458BE4
EIP 0044D217 0_.0044D217
C 0  ES 0023 32bit 0(FFFFFFFF)
P 1  CS 001B 32bit 0(FFFFFFFF)
A 0  SS 0023 32bit 0(FFFFFFFF)
Z 1  DS 0023 32bit 0(FFFFFFFF)
S 0  FS 003B 32bit 7FFDD000(FFF)
T 0  GS 0000 NULL
```

The malware starts looking using “FindFirst” from the SysUtils library of Delphi and FindNext. For each file or directory founded, its checks the name with “.” and “..” to avoid get the actual directory or the previous one.



Its checks if it is a directory or not, and if it is, call the same function in a recursive way to enter in all subfolders in all tree of directories of the unit.

If its found a file checks that the file not is in some folders black listed. To make this the malware uses the function “ExtractFileDir” of the SysUtils library.

The black list of the folders it is:

- windows
- Windows
- program files
- Program Files
- Program Files (x86)
- Programme
- Programme (x86)
- program files (x86)
- programme
- programme (x86)
- programdata

The malware have a bug here. Its try avoid destroy the windows directory, but only checks for two strings and not check if the windows folder have the name all in caps (WINDOWS). So, sometimes the windows folder can be affected.

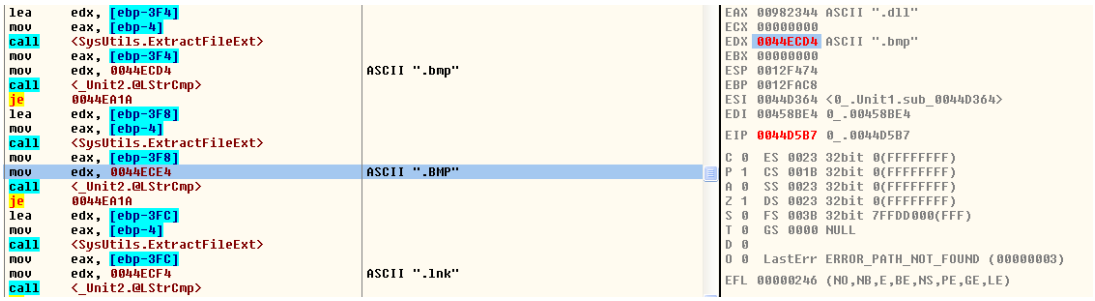
The malware checks the filename using the function “ExtractFileName” of the

SysUtils library to check that the file not is the ransom note with the german name “Wo_sind_meine_Datein.html”.

```
call    <SysUtils.ExtractFileDir>
mov     edx, [ebp-3B4]
mov     eax, 0044EB7C                   ASCII "windows"
call    <_Unit2.@LStrPos>
mov     ebx, eax
lea     edx, [ebp-3B8]
mov     eax, [ebp-4]
call    <SysUtils.ExtractFileDir>
mov     edx, [ebp-3B8]
mov     eax, 0044EB8C                   ASCII "program files"
call    <_Unit2.@LStrPos>
or      ebx, eax
lea     edx, [ebp-3BC]
mov     eax, [ebp-4]
call    <SysUtils.ExtractFileName>
mov     edx, [ebp-3BC]
mov     eax, 0044EBA4                   ASCII "Wo_sind_meine_Dateien.html"
call    <_Unit2.@LStrPos>
```

If the file remains in some of the black listed list or have the ransom note name the malware avoids this file.

If not its gets the extension of the file using the function “ExtractFileExt” of SysUtils library. After this, its checks the extension with a long list of extensions. If have one of them will destroy the data as I will explain now, if not its avoid this file.



The code quality is bad as you can see, because is a copy-paste of the same code or macro extracting each time the extension of the file and compare with the new extension to check.

The list of extensions affected for this is:

jpg	mp3	exe	ODS	m3u	asm
JPG	MP3	EXE	ppt	M3U	ASM
jpeg	mov	msi	PPTX	backup	eps
JPEG	MOV	MSI	odp	BACKUP	EPS
bmp	webm	txt	ODP	back	ai
BMP	WEBM	TXT	xlt	BACK	AI
lnk	ogg	rtf	XLT	cpp	p12
LNK	OGG	RTF	pwm	CPP	P12
gif	m4p	doc	PWM	psd	pem
GIF	M4P	DOC	html	PSD	PEM
wav	mpeg	docx	HTML	msg	ppk
WAV	MPEG	DOCX	css	MSG	PPK
avi	mpg	xls	CSS	xml	csv
AVI	MPG	XLS	asp	XML	CSV
mkv	7zip	xlsx	ASP	sqlited	json
MKV	7ZIP	XLSX	aspx	bSQL	JSON
pdf	tif	dat	ASPX	ITEDB	bat
PDF	TIF	DAT	xlm	sqlite3	BAT
zip	tiff	dbDB	XLM	SQLITE3	ico
ZIP	TIFF	sql	js	pptx	ICO
rar	mp4	SQL	JS	docm	atn
RAR	MP4	odt	tsTS	DOCM	ATN
7z	iso	ODT	java	xlsm	svg
7Z	ISO	ods	JAVA	XLSM	SVG

In case the file have a target extension the malware gets a random value to make a random file name:

```
mov     eax, 3
call    <_Unit2.@RandInt>
sub     eax, 1
jb      short 0044D136
je      short 0044D159
dec     eax
je      short 0044D17C
jmp     short 0044D19D
mov     eax, 1A
call    <_Unit2.@RandInt>
mov     edx, eax
add     edx, 61
lea     eax, [ebp-4]
call    <_Unit2.@LStrFromChar>
mov     edx, [ebp-4]
mov     eax, esi
call    <_Unit2.@LStrCat>
jmp     short 0044D19D
mov     eax, 1A
call    <_Unit2.@RandInt>
mov     edx, eax
```

Usually the name have 13 characters. After it the malware gets another random value from 0 to 0x3E80 and add to it 0x1F40 to make the new size of garbage that will have the new “crypted” file.

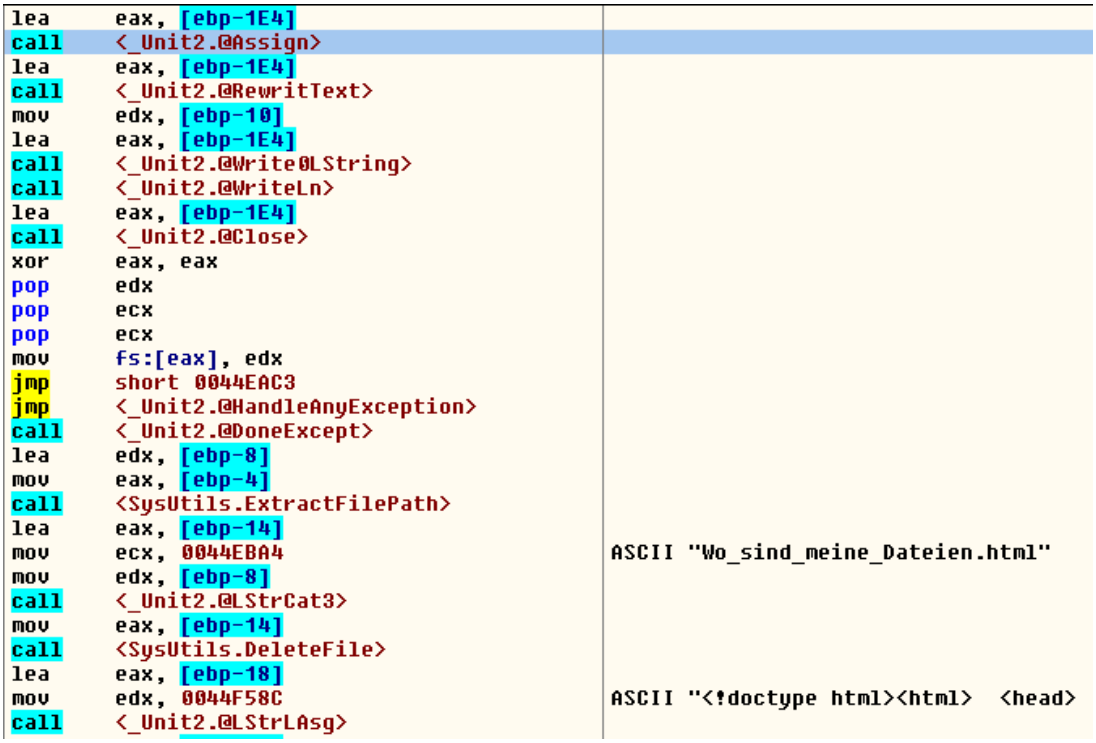
With this size will fill a buffer with random chars, later will apply this buffer in the file.

The next operation is get the file without path and delete it:

```
lea     edx, [ebp-8]
mov     eax, [ebp-4]
call    <SysUtils.ExtractFilePath>
mov     eax, [ebp-4]
call    <SysUtils.DeleteFile>
lea     eax, [ebp-63C]
mov     ecx, [ebp-C]
mov     edx, [ebp-8]
```

One important thing here is, the malware delete the file but not overwrite with garbare or null all his information, so, the file will remains in the raw hard disk untouched (later, with luck, perhaps can be recovered using some programa as Recuva).

After delete it concat the path with the new file random name, create the file and fill with the buffer with garbage maked before. Finally its close the handle to the new file, and create the ransom note in the same folder with the name “Wo_sind_meine_Datein.html”.

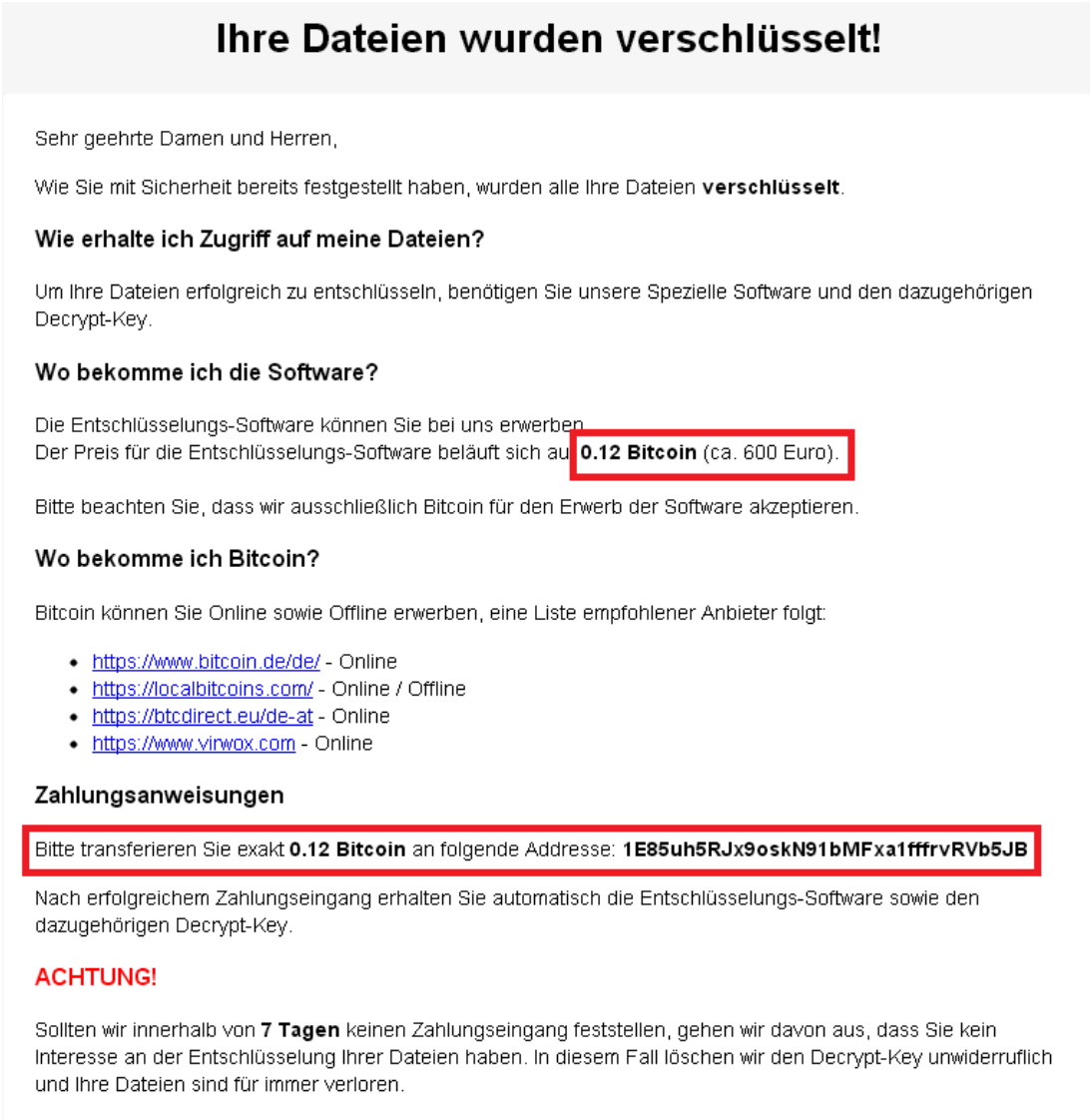


As said before the quality of the code is bad, because this ransom note is created for EACH file that destroy, before delete it, and later make again instead check if the folder have it previously or not with the ransom info.

After finish all process in all units the malware create a mutex with the name “HSDFSD-HFSD-3241-91E7-ASDGSDGHH” and checks if exists looking with GetLastError API. If exists the malware exists and if not the malware create his form (invisible form) and keeps running in the memory.

II. RECOVER FILES

The malware drops as said before a ransom note in german as this:



Ihre Dateien wurden verschlüsselt!

Sehr geehrte Damen und Herren,

Wie Sie mit Sicherheit bereits festgestellt haben, wurden alle Ihre Dateien **verschlüsselt**.

Wie erhalte ich Zugriff auf meine Dateien?

Um Ihre Dateien erfolgreich zu entschlüsseln, benötigen Sie unsere Spezielle Software und den dazugehörigen Decrypt-Key.

Wo bekomme ich die Software?

Die Entschlüsselungs-Software können Sie bei uns erwerben.
Der Preis für die Entschlüsselungs-Software beläuft sich au **0.12 Bitcoin** (ca. 600 Euro).

Bitte beachten Sie, dass wir ausschließlich Bitcoin für den Erwerb der Software akzeptieren.

Wo bekomme ich Bitcoin?

Bitcoin können Sie Online sowie Offline erwerben, eine Liste empfohlener Anbieter folgt:

- https://www.bitcoin.de/de/ - Online
- https://localbitcoins.com/ - Online / Offline
- https://btcdirect.eu/de-at - Online
- https://www.virwox.com - Online

Zahlungsanweisungen

Bitte transferieren Sie exakt **0.12 Bitcoin** an folgende Addresse: **1E85uh5RJx9oskN91bMFxa1fffrvRVb5JB**

Nach erfolgreichem Zahlungseingang erhalten Sie automatisch die Entschlüsselungs-Software sowie den dazugehörigen Decrypt-Key.

ACHTUNG!

Sollten wir innerhalb von **7 Tagen** keinen Zahlungseingang feststellen, gehen wir davon aus, dass Sie kein Interesse an der Entschlüsselung Ihrer Dateien haben. In diesem Fall löschen wir den Decrypt-Key unwiderruflich und Ihre Dateien sind für immer verloren.

But the malware destroy files, so, a victim CANT recover the files paying (anyways not is a good idea pay to the ransomware makers because, as you can see, they can lie and you cant get the files recovered). In this note they said that uses AES to crypt your files and request you about 0.12 BTC (more or less 600 EUR) for nothing!

But the malware don't destroy any Shadow Volume or Restore Point in the system, so, with luck and using some program that manage Shadows Volumes as Shadow Explorer.

https://www.bleepingcomputer.com/download/shadowexplorer/

Or following this steps:

https://www.bleepingcomputer.com/tutorials/how-to-recover-files-and-folders-using-shadow-volume-copies/

Another option is use a special program for recover files from raw disk as Recuva but perhaps its not will recover all your files.

https://www.piriform.com/recuva

III. FINAL WORDS

A stupid malware that destroy information of enterprises and innocent people and try steal money saying that is a ransomware. Bad coding style, a easy packer, only need 1 hour of my time to reverse it and writing this report.

Valthek