

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.

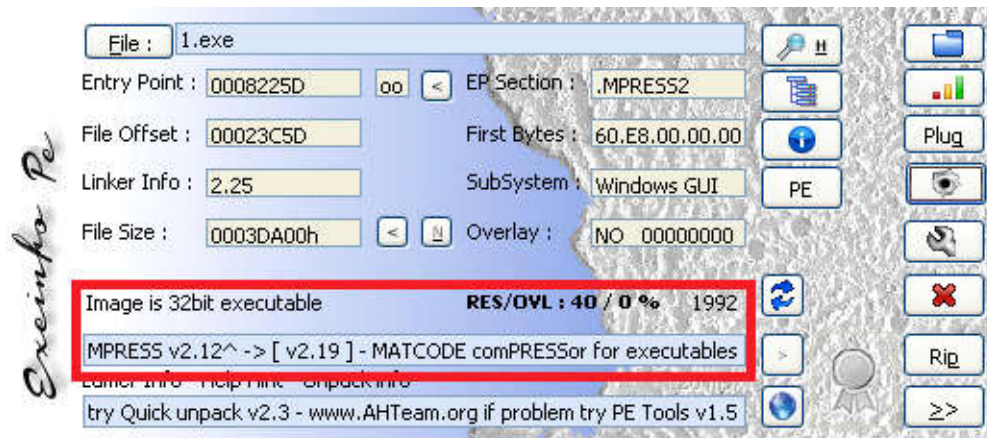
Ordinypt appears be a ransomware, but is a **wiper**. As you will see the malware requests money for recover the files but the files are destroyed without any chance to recover it from the malware creators.

So, lets start talking about this sample.

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

I. LOOKING THE MALWARE

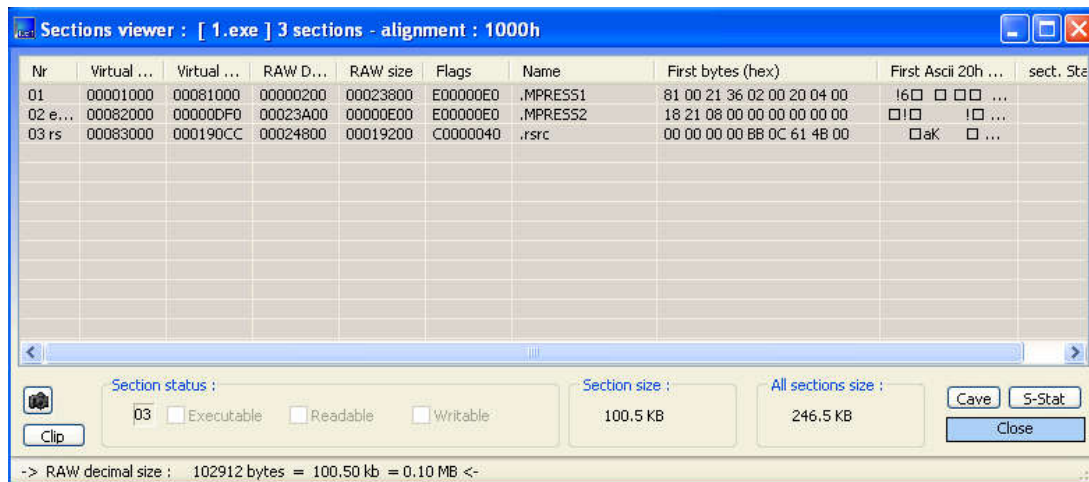
The first action is look if the malware is packed or not. I use ExeInfo but you can use PeID or another tool.



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.

Address	Hex dump	Disassembly
0048225D	60	pushad
0048225E	E8 00000000	call 00482263
00482263	58	pop eax
00482264	05 5A0B0000	add eax, 0B5A
00482269	8B30	mov esi, [eax]
0048226B	03F0	add esi, eax
0048226D	2BC0	sub eax, eax
0048226F	8BFE	mov edi, esi
00482271	66:AD	lods word ptr [esi]
00482273	C1E0 0C	shl eax, 0C
00482276	8BC8	mov ecx, eax
00482278	50	push eax
00482279	AD	lods dword ptr [esi]
0048227A	2BC8	sub ecx, eax
0048227C	03F1	add esi, ecx
0048227E	8BC8	mov ecx, eax
00482280	57	push edi
00482281	51	push ecx
00482282	49	dec ecx
00482283	8A4439 06	mov al, [ecx+edi+6]
00482287	880431	mov [ecx+esi], al

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

Address	Hex dump	Disassembly	Comment
00455B30	55	push ebp	
00455B31	8BEC	mov ebp, esp	
00455B33	83C4 F0	add esp, -10	
00455B36	B8 50594500	mov eax, 00455950	
00455B38	E8 9805FBFF	call 004060D8	
00455B40	68 A45B4500	push 00455B44	
00455B45	6A FF	push -1	
00455B47	6A 00	push 0	
00455B49	E8 8A07FBFF	call 004062D8	
00455B4E	85C0	test eax, eax	
00455B50	75 05	jnz short 00455B57	
00455B52	E8 5D70FBFF	call 0040CBB4	
00455B57	E8 4C08FBFF	call 004063A8	
00455B5C	3D B7000000	cmp eax, 0B7	jmp to ntdll.RtlGetLastWin32Error
00455B61	74 3B	je short 00455B9E	
00455B63	A1 C46F4500	mov eax, [456FC4]	
00455B68	8B00	mov eax, [eax]	
00455B6A	E8 A15DFFFF	call 0044B910	
00455B6F	8B00 A0704500	mov ecx, [4570A0]	1.004588D0

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

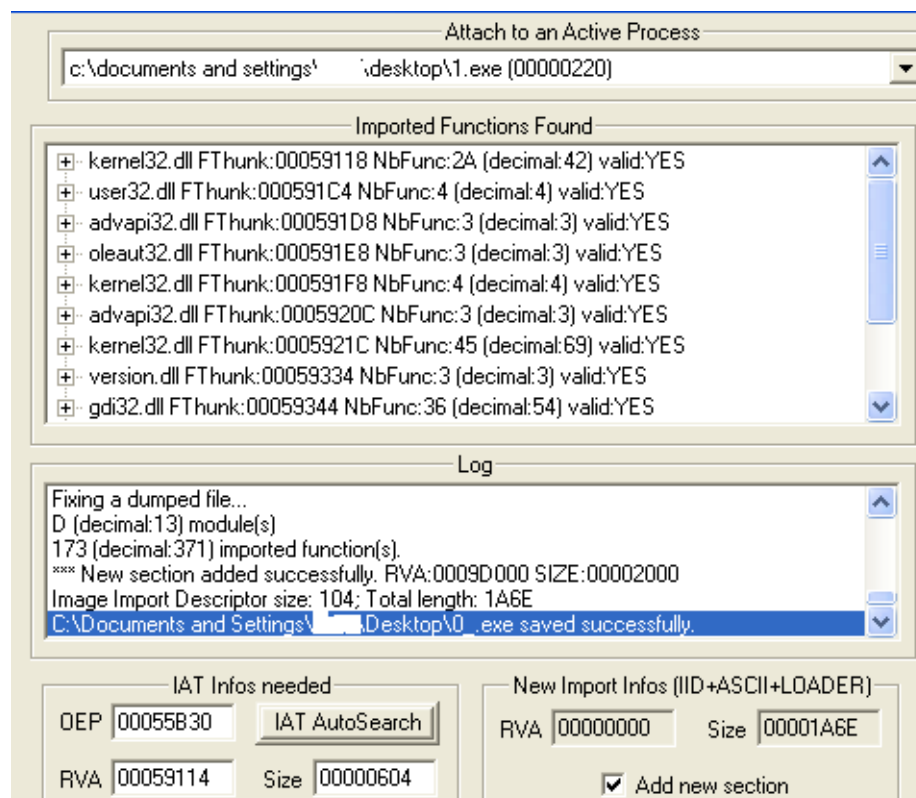
- Use hardware breakpoints to avoid loops. With this packer you can use normal breakpoints but is useful learn, to use hardware instead the normal ones because some programs can use his own code to unpack, make a key, or whatever think. A normal breakpoint puts a byte for the opcode "int3" 0xCC in the 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 written in Delphi 7.

```

.NPRESS1:00455B80 ; ===== S U B R O U T I N E =====
.NPRESS1:00455B80 ; Attributes: library function noreturn bp-based frame
.NPRESS1:00455B80 public start
.NPRESS1:00455B80 proc near ; CODE XREF: .NPRESS1:loc_455CE74j
.NPRESS1:00455B81 push ebp
.NPRESS1:00455B81 mov ebp, esp
.NPRESS1:00455B83 add esp, 0FFFFFF0h
.NPRESS1:00455B86 mov eax, offset dword_455950
.NPRESS1:00455B89 call @SysInit@InitExe$qqw ; Sysinit::_linkproc__InitExe(void *)
.NPRESS1:00455B84 push offset aHsdFsdHfsd3241 ; "HSDfSD-HfSD-3241-91E7-ASDCS0GHH"
.NPRESS1:00455B85 push 0FFFFFFFh ; int
.NPRESS1:00455B87 push 0 ; lpPutexAttributes
.NPRESS1:00455B89 call unknown_libname_103 ; BDS 2005-2007 and Delphi6-7 Visual Component Library
.NPRESS1:00455B8E test eax, eax
.NPRESS1:00455B90 jnz short loc_455B97
.NPRESS1:00455B92 call @Sysutils@RaiseLastOSError$qqw ; Sysutils::RaiseLastOSError(void)
.NPRESS1:00455B97 loc_455B97: call j_GetLastError ; CODE XREF: start+207j
.NPRESS1:00455B9C cmp eax, 007h
.NPRESS1:00455B9E jz short loc_455B9E
.NPRESS1:00455BA3 mov eax, off_456FC4
.NPRESS1:00455BA6 mov eax, [eax]
.NPRESS1:00455BA8 call sub_448910
.NPRESS1:00455BAE mov ecx, off_457000
.NPRESS1:00455BB0 mov eax, off_456FC4
.NPRESS1:00455BB2 mov eax, [eax]
.NPRESS1:00455BB4 mov edx, off_44CEB0
.NPRESS1:00455BB6 call @Forms@TApplication@CreateForm$qqqr7System@TMetaClasspw ; Forms::TApplication::CreateForm(System::TMetaClass *,void *)
.NPRESS1:00455BB8 mov eax, off_456FC4

```

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 others calls belongs to Delphi starting his own libraries and units.

00404040	C705 14804500	mov	dword ptr [458014], <_Unit2.RaiseExcept	_Unit2.@StartExe
0040404A	C705 18804500	mov	dword ptr [458018], <_Unit2.RtlUnwind	jmp to ntdll.RtlUnwind
00404054	A3 3C864500	mov	[45863C], eax	
00404059	33C0	xor	eax, eax	
0040405B	A3 40864500	mov	[458640], eax	
00404060	8915 44864500	mov	[458644], edx	
00404066	8B42 04	mov	eax, [edx+4]	
00404069	A3 2C804500	mov	[45802C], eax	
0040406E	E8 C5FEFFFF	call	<_Unit2.SetExceptionHandler>	
00404073	C605 34804500	mov	byte ptr [458034], 0	
0040407A	E8 61FFFFFF	call	<_Unit2.InitUnits>	

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.

098D A4FEFFFF	mov	[ebp-15C], ecx		EAX 0012FE10
88F2	mov	esi, edx		ECX 0044D03C ASCII "*"
88D8	mov	ebx, eax		EDX 00455888 ASCII "C:\\"
8885 08FEFFFF	lea	eax, [ebp-158]		EBX 00455888 ASCII "C:\\"
8815 78714000	mov	edx, [407178]	0_0040717C	ESP 0012FDF4
E8 907AFBFF	call	<Unit2.@InitializeRecord>		EBP 0012FF6C
33C0	xor	eax, eax		ESI 0044D0364 <0_Unit1.sub_0044D0364>
55	push	ebp		EDI 00458BE4 0_00458BE4
68 26D34400	push	0044D026		EIP 0044D027 0_0044D027
64:FF30	push	dword ptr fs:[eax]		C 0 ES 0023 32bit 0(FFFFFFFF)
64:8920	mov	fs:[eax], esp		P 1 CS 001B 32bit 0(FFFFFFFF)
8085 A4FEFFFF	lea	eax, [ebp-15C]		A 0 SS 0023 32bit 0(FFFFFFFF)
B9 3CD34400	mov	ecx, 0044D03C	ASCII "*"	Z 1 DS 0023 32bit 0(FFFFFFFF)
88D3	mov	edx, ebx		S 0 FS 003B 32bit 7FFDD000(FFF)
E8 E873FBFF	call	<Unit2.@LStrCat3>		T 0 GS 0000 NULL
8885 A4FEFFFF	mov	eax, [ebp-15C]		

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.

33C0	xor	eax, eax	
55	push	ebp	
68 F0D24400	push	0044D2F0	
64:FF30	push	dword ptr fs:[eax]	
64:8920	mov	fs:[eax], esp	
F685 B0FEFFFF	test	byte ptr [ebp-150], 10	
74 4F	je	short 0044D2A0	
8B85 B4FEFFFF	mov	eax, [ebp-14C]	
BA 48D34400	mov	edx, 0044D348	
E8 A374FBFF	call	<_Unit2.@LStrCmp>	
74 61	je	short 0044D2C4	
8B85 B4FEFFFF	mov	eax, [ebp-14C]	
BA 54D34400	mov	edx, 0044D354	ASCII ".."
E8 9174FBFF	call	<_Unit2.@LStrCmp>	

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_Dateien.html".

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

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.

<pre> lea edx, [ebp-3F4] mov eax, [ebp-4] call <SysUtils.ExtractFileExt> mov eax, [ebp-3F4] mov edx, 0044ECD4 call <_Unit2.@LStrCmp> je 0044EA1A lea edx, [ebp-3F8] mov eax, [ebp-4] call <SysUtils.ExtractFileExt> mov eax, [ebp-3F8] mov edx, 0044ECE4 call <_Unit2.@LStrCmp> je 0044EA1A lea edx, [ebp-3FC] mov eax, [ebp-4] call <SysUtils.ExtractFileExt> mov eax, [ebp-3FC] mov edx, 0044ECF4 call <_Unit2.@LStrCmp> </pre>	<pre> ASCII ".bmp" ASCII ".BMP" ASCII ".lnk" </pre>	<pre> EAX 00982344 ASCII ".dll" ECX 00000000 EDX 0044ECD4 ASCII ".bmp" EBX 00000000 ESP 0012F474 EBP 0012FAC8 ESI 00440364 <_Unit1.sub_00440364> EDI 00458BE4 0_00458BE4 EIP 0044D5B7 0_0044D5B7 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 O 0 I 0 LastErr ERROR_PATH_NOT_FOUND (00000003) EFL 00000246 (NO,NB,E,BE,NS,PE,GE,LE) </pre>
---	---	--

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 garbage 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”.

lea	eax, [ebp-1E4]	
call	<_Unit2.@Assign>	
lea	eax, [ebp-1E4]	
call	<_Unit2.@RewritText>	
mov	edx, [ebp-10]	
lea	eax, [ebp-1E4]	
call	<_Unit2.@Write0LString>	
call	<_Unit2.@WriteLn>	
lea	eax, [ebp-1E4]	
call	<_Unit2.@Close>	
xor	eax, eax	
pop	edx	
pop	ecx	
pop	ecx	
mov	fs:[eax], edx	
jmp	short 0044EAC3	
jmp	<_Unit2.@HandleAnyException>	
call	<_Unit2.@DoneExcept>	
lea	edx, [ebp-8]	
mov	eax, [ebp-4]	
call	<SysUtils.ExtractFilePath>	
lea	eax, [ebp-14]	
mov	ecx, 0044EBA4	ASCII "Wo_sind_meine_Dateien.html"
mov	edx, [ebp-8]	
call	<_Unit2.@LStrCat3>	
mov	eax, [ebp-14]	
call	<SysUtils.DeleteFile>	
lea	eax, [ebp-18]	
mov	edx, 0044F58C	ASCII "<!doctype html><html> <head>
call	<_Unit2.@LStrLAsg>	

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 auf **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.vinwox.com> - Online

Zahlungsanweisungen

Bitte transferieren Sie exakt **0.12 Bitcoin** an folgende Adresse: **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.

Valtek