2 - 4 October, 2024 / Dublin, Ireland

THE MASK HAS BEEN UNMASKED AGAIN

Georgy Kucherin & Marc Rivero López

Kaspersky, Russia & Spain

georgy.kucherin@gmail.com
mriverolopez@gmail.com

ABSTRACT

The Mask (also known as Careto) is an advanced threat actor that has been operating since at least 2007. In the past, it was observed to conduct cyber espionage campaigns that mainly targeted high-profile organizations. This actor's attacks have always been remarkable from the technical perspective, as they commonly involve use of zero-day exploits, bootkits, and modular backdoors for different operating systems.

Over the last decade, The Mask has been doing its best to avoid getting caught by researchers: since 2014 there has not been any information about the group. Nevertheless, in our recent research we have managed to uncover a number of new campaigns by this threat actor – with the latest dated up to early 2024. In our paper we provide details about these campaigns, focusing on how The Mask has been achieving initial access, lateral movement, malware execution and data exfiltration.

Specifically, we first describe how The Mask leveraged the MDaemon email server of one of the target organizations to gain an initial foothold inside it. We then detail how this threat actor used a previously unknown bug in a security solution to covertly spread malicious implants across machines. Afterwards, we discuss capabilities of the delivered implants, as well as the stealth measures implemented inside them.

The Mask has always conducted cyber attacks with extreme caution. Despite this, members of this threat group have managed to make small but fatal mistakes during their recent operations. In the paper, we describe these errors, specifying how they helped us not only detect the discussed malicious activities, but also perform attribution of the discovered campaigns.

At the end of the paper we present a comparison between The Mask's historical and more recent attacks to demonstrate how the group's operations have evolved over the years.

INTRODUCTION

The Mask is an APT group that has been observed performing highly sophisticated attacks that have mainly been targeting government organizations, diplomatic entities, energy companies and research institutions. It has been known to use zero-day exploits, as well as unique, professionally coded implants for *Windows*, *macOS* and *Linux*.

Attacks by this APT threat actor have been observed since 2007, up until 2013 [1]. Notably, no news about this threat group has been released since that time. However, we have discovered two recent cluster of infections, which we attribute to The Mask with medium to high confidence:

- An infection at an organization in Latin America, carried out in 2019.
- A successful attack against the same organization in Latin America, executed in 2022. Notably, the malware used in these attacks, which we dubbed FakeHMP, was also observed deployed on the machine of an unidentified individual or organization as recently as January 2024.

In the following sections of this paper we provide information about these infections, first discussing the recent ones that occurred in 2022 and 2024, and then the historical ones, carried out in 2019.

MDAEMON SERVER INFECTION

While there is no data trace on how the machine in 2024 came to be infected, we did manage to find out how the organization located in Latin America was compromised back in 2022. Specifically, we have established that attackers managed to compromise the organization's email server, which was running the MDaemon email software. While we have not identified the exact method used to hack the server, we have established that the compromised server was used for maintaining persistence within the organization's network. As for the persistence method used, it is rather unique. It involves using MDaemon's webmail component called WorldClient:

Figure 1: WorldClient webmail authentication panel.

WorldClient allows loading extensions (which are similar to IIS extensions), which can handle HTTP requests coming from clients. These extensions are registered in the file `C:\MDaemon\WorldClient\WorldClient.ini`. In order to register an extension, the relative URL controlled by the extension must be specified (in the parameter `CgiBase`), as well as the path to the extension DLL (in the parameter `CgiFile`). The extension DLLs are loaded by the WorldClient.exe process.

In order to establish persistence, the APT group has built its own malicious WorldClient extension and configured it by adding the entries for `CgiBase6` (containing the relative URL `/WorldClient/mailbox`) and `CgiFile6` (containing the path to the extension DLL, `C:\MDaemon\WorldClient\HTML\MDMBoxSrch.dll`). By doing this, adversaries became able to interact with the malicious extension by making HTTP requests to the URL `https://<webmail server domain name>/WorldClient/mailbox`. It is also important to mention that, as the email server had an external IP address, any computer connected to the internet could be used to interact with the malicious extension.

```
[WebServer]
BindAddress=
CgiBase1=/WorldClient.dll
CgiBase10=/WorldClientAPI
CgiBase11=/Mddp
CgiBase3=/Microsoft-Server-ActiveSync
CgiBase5=/AutoDiscover/AutoDiscover.xml
CgiBase6=/WorldClient/mailbox
CgiBase7=/webdav
CgiBase8=/.well-known/caldav
CgiBase9=/.well-known/carddav
CgiFile1=C:\MDaemon\WorldClient\HTML\WorldClient.dll
CgiFile10=C:\MDaemon\WorldClient\HTML\WorldClient.dll
CgiFile11=C:\MDaemon\ISAPI\MDDP\MDDP.dll
CgiFile3=C:\MDaemon\WorldClient\HTML\MDAirSync.dll
CgiFile5=C:\MDaemon\WorldClient\HTML\MDAutoDiscover.dll
CgiFile6=C:\MDaemon\WorldClient\HTML\MDMBoxSrch.dll
CgiFile7=C:\MDaemon\WebDAV\MDWebDAV.dll
CgiFile8=C:\MDaemon\WebDAV\MDWebDav.dll
CgiFile9=C:\MDaemon\WebDAV\MDWebDav.dll
```

Figure 2: WorldClient webmail authentication panel (entries corresponding to the malicious extension are underlined).

It should be noted that the described persistence method cannot be detected with common auto-start application detection tools such as Autoruns.

INSTALLED MALICIOUS EXTENSION

The malicious WorldClient extension discovered on the organization's email server exhibits three exports

Export name	Export function activity
`GetExtensionVersion`	Places the string `Mailbox Search Module for MDaemon Messaging Server, Version 21.0.0` inside the `HSE_VERSION_INFO` structure passed in an argument.
`TerminateExtension`	Auxiliary function, performs memory cleanup.
`HttpExtensionProc`	Processes the HTTP requests that are handled by the extension.

The extension that processes POST requests sent to the URL `https://<webmail server domain name>/WorldClient/mailbox` can serve two functions:

- Execute an attacker-supplied command. The ID of the command is specified in the `QueryID` URL parameter, while the command parameters are specified in the POST request body. In order to execute a command, the extension loads the library located at `C:\MDaemon\App\MDUserQuery.dll` and calls its `runMBoxSrch` export, passing it the URL parameters and the request body.
- Update the `MDUserQuery.dll` library (described below) that is used for processing commands. The DLL body is provided in the POST request body, while the DLL body size is specified in the `CalStart` URL parameter.

Additionally, the request's Reload URL parameter is set to 'Yes'.

THE COMMAND PROCESSING LIBRARY

As mentioned above, the MDUserQuery.dll library has one export called `runMBoxSrch`, which processes commands supplied by the attacker. It supports the following commands:

Command ID	Command description
01	Lists DLL modules loaded into the WorldClient.exe process, running processes and installed services.
02	Starts a process specified in the command arguments, redirecting its output to a pipe, waits for the process to finish and sends its output to the attacker.
10	Writes a file on disk and timestomps it.
11	Drops an executable specified in the command body to the path `C:\MDaemon\WorldClient\Temp\C3OB7RCITKJBI\MDSync.exe` and then launches it, redirecting its output to a pipe. After the process finishes, the command sends its output to the attacker and removes the executable file.
12	Sends contents of a specified file.
13	Removes a specified file.
14	Performs timestomping on a specified file.
15	Lists contents of a specified directory.
16	Overwrite file contents without changing the file's timestamps.

Having installed the MDaemon backdoor, attackers used it to perform reconnaissance of the infected organization network and perform spreading to machines of interest inside it.

SPREADING VIA SCHEDULED TASKS

In the infection case occurring in 2022, attackers were observed to perform spreading to other machines by using scheduled tasks. Over the course of doing that, attackers copied the following files to remote machines:

File name	Path on remote machine	Description
`hmpalert.sys`	`C:\Windows\System32\drivers\hmpalert.sys`	Legitimate driver of the *HitmanPro Alert* software.
`hmpalert.dll`	`C:\Windows\System32\hmpalert.dll`	A malicious DLL named exactly after a legitimate component of the *HitmanPro Alert* software, containing an implant we dubbed FakeHMP.
`~DFAE01202C5F0DBA42.cmd`	`C:\Windows\Temp\~DFAE01202C5F0DBA42.cmd`	Malicious .bat file.
`Tpm-HASCertRetr.xml`	`c:\Windows\Temp\TpmHASCertRetr.xml`	Malicious .xml file, containing a description of a scheduled task configured to launch the file `~DFAE01202C5F0DBA42.cmd`.

Following that, attackers executed the following sequence of commands that:

- Create a scheduled task with the description specified in the `Tpm-HASCertRetr.xml` file:

```
schtasks /create /S \\<remote victim machine IP address> /U <remote machine username>
/P <remote machine user password> /tn "Microsoft\Windows\WindowsColorSystem\pm-
HASCertRetr" /XML "C:Windows\Temp\Tpm-HASCertRetr.xml");
```

- Launch the created scheduled task to execute the `C:\Windows\Temp\~DFAE01202C5F0DBA42.cmd` script on the remote machine:

```
schtasks /run /S \\<remote victim machine IP address> /U <remote machine
username> /P <remote machine user password> /tn "Microsoft\Windows\
WindowsColorSystem\pm-HASCertRetr");
```

- Remove the created scheduled task after the execution of the .bat file is finished:

```
schtasks /delete /S \\<remote victim machine IP address> /U <remote
machine username> /P <remote machine user password> /tn "Microsoft\
Windows\WindowsColorSystem\pm-HASCertRetr" /F);
```

In turn, the .bat file that is launched by the scheduled task contains the following lines:

```
reg add HKLM\SYSTEM\CurrentControlSet\Services\hmpalert
reg add HKLM\SYSTEM\CurrentControlSet\Services\hmpalert\Instances
reg add HKLM\SYSTEM\CurrentControlSet\Services\hmpalert\Instances /v DefaultInstance /t REG_SZ /d CryptoGuard
reg add HKLM\SYSTEM\CurrentControlSet\Services\hmpalert\Instances\CryptoGuard
reg add HKLM\SYSTEM\CurrentControlSet\Services\hmpalert\Instances\CryptoGuard /v Altitude /t REG_SZ /d 345800
reg add HKLM\SYSTEM\CurrentControlSet\Services\hmpalert\Instances\CryptoGuard /v Flags /t REG_DWORD /d 00000000
sc create hmpalert binPath= c:\windows\system32\drivers\hmpalert.sys type= kernel start= system
```

These commands add several entries to the registry key `HKLM\SYSTEM\CurrentControlSet\Services\hmpalert`, install the hmpalert.sys driver and configure it to be loaded on machine startup. As described below, this driver is abused to load the FakeHMP implant.

SPREADING VIA GOOGLE UPDATER

In the 2024 infection case, we observed attackers configuring the FakeHMP implant in an alternative way. In order to

achieve that, they placed a malicious DLL named goopdate.dll into the folder `C:\Program Files (x86)\Google\Update`. This folder contains the legitimate *Google Updater* executable configured to run regularly on the machine. By placing the malicious DLL inside this folder, attackers made it sideload on each regular launch of *Google Updater*. This technique is not novel and has previously been observed used by other threat actors, such as MuddyWater [2].

As for the installer itself, it performs the same actions as the previously described .bat file: it adds *HitmanPro*-related entries to the registry and creates the hmpalert service to launch the driver on startup.

THE FAKEHMP IMPLANT

This implant was found on machines in both cases from 2022 and 2024, where it gets loaded on startup through `hmpalert.sys` (SHA256 hash: `9733eb1802daf774cfe576179b1096b5861593d702c6c0226b75410b13dab6ae`), a legitimate driver of the *HitmanPro Alert* security solution. One of its responsibilities is to load *HitmanPro*'s DLL located at `C:\Windows\System32\hmpalert.dll` into every running process. However, the DLL-loading implementation is flawed: the driver does not verify the DLL file's legitimacy before loading. The threat actor decided to abuse this bug to leverage the legitimate security solution's driver for its malicious persistence. By placing a malicious DLL at `C:\Windows\System32\hmpalert.dll` and installing the hmpalert.sys driver, the attackers made the legitimate driver load the malicious DLL into every running process.

With this DLL loaded into every process, it performs malicious operations from three processes only. To decide whether it should work this way from a given process, this DLL computes the SHA256 hash of its command line on startup, comparing it with the three following values:

SHA256 hash	Command line corresponding to the hash
`050bddfe5597632d7b9ec4ff1c49e8cd860ec25d077bee31f2ba3a0394b447f8`	`winlogon.exe`
`096eae3dc85a866fccd68496d897af673af6d2455ef6a8d09d5f4ac2d84d9bd1`	`"dwm.exe"`
`a3930f657677999ef3b567bd40d527f2463fc29931eac43e134739f98a90b595`	`svchost.exe -k NetworkService -p -s CryptSvc`

If the command line hash does not match any of the three hashes above, the DLL unloads itself. Otherwise, it:

- Decrypts its resource (type: `RCDATA`, ID: 300) with AES and decompresses it, thus obtaining a second-stage DLL
- Reflectively loads the decrypted DLL and invokes its entry point function
- Unloads itself.

As a result of these actions, FakeHMP gets loaded into three processes: winlogon.exe, dwm.exe and svchost.exe. It is important to note that traces of this implant cannot be spotted by examining the list of loaded modules, e.g. with tools like *Process Explorer*. Additionally, the hmpalert.dll library cannot be spotted in the list of loaded modules, since it gets unloaded immediately after launching the second-stage payload.

The instances of FakeHMP placed inside the three processes work simultaneously. They communicate with each other via a named pipe called `\\.\pipe\9952763031`, where the number may vary between samples.

The payload inside the winlogon.exe process is responsible for reading commands from the pipe and executing them. It supports two types of commands:

- Read a file from disk (the filename is specified in the command) and send its contents to the pipe.
- Reflectively load a DLL (its body is specified in the command) and invoke its entry point function.

The payload hosted in dwm.exe logs keystrokes and takes screenshots. The keylogging method involves using the DirectX library. Logged keystrokes are sent to the pipe. Screenshots are taken using the GDI API (in JPEG format) and also sent to the pipe.

The data sent to the pipe from `winlogon.exe` and `dwm.exe` is processed by `svchost.exe`, which compresses data with the `RtlCompressBuffer` function and then encrypts it with AES. The corresponding encryption key is embedded into the payload. It then uploads the encrypted data to the *OneDrive* cloud using a client ID and a refresh token embedded into the implant.

Apart from this implant, we have observed additional components that complement the functionality of FakeHMP. Their descriptions are provided in further sections of this paper.

MICROPHONE RECORDER

On one of the machines infected with The Mask implants in 2022, we discovered a malicious DLL placed at `C:\Windows\ninput.dll`. This DLL is loaded by the explorer.exe process with each startup. A legitimate counterpart bearing the same name is located at `C:\Windows\System32\ninput.dll`, representing the Microsoft Pen and Touch Input component. However, if another DLL is present at the same path, the explorer.exe process loads it together with the original ninput.dll. As a result of that, a malicious library gets loaded into explorer.exe.

This malicious DLL's purpose is to record audio via the victim's microphone. On startup, it reflectively reloads itself and then hides the tray icon that is displayed whenever the microphone is active:

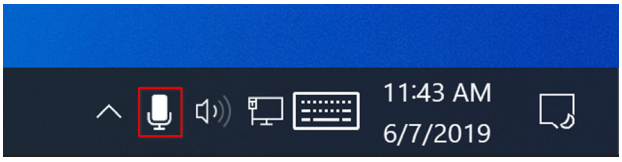

Figure 3: Microphone icon that is hidden.

The malicious library implements two methods of hiding the microphone icon:

- On *Windows 10*, it retrieves the address of the `Shell_NotifyIconW` function (exported by `shell32.dll`) and places an EAT hook on this function. The hook checks whether the `Shell_NotifyIconW` function has been called for the microphone icon and returns 1 in this case. Otherwise, it calls the original `Shell_NotifyIconW` function.
- On *Windows 11*, it patches contents of the `OnMicCapabilityUsageChanged` function (implemented inside `C:\Windows\SystemApps\MicrosoftWindows.Client.CBS_cw5n1h2txyewy\ExplorerExtensions.dll`) to the following instructions, which make the function always return 0:

  1. `mov eax, 0`
  2. `ret`

  However, unlike the `Shell_NotifyIconW` function, the `OnMicCapabilityUsageChanged` function is not exported from the ExplorerExtensions.dll library. In order to find the address of this function, the malicious library performs the following steps:

  - Downloads the PDB file that corresponds to the `ExplorerExtensions.dll` library from *Microsoft*'s public symbol server (`http://msdl.microsoft.com/download/symbols/`)
  - Parses the symbols inside the downloaded PDB file, looking for the address of the `OnMicCapabilityUsageChanged` function.

Having hidden the microphone icon, the malicious library starts recording the microphone input using *Windows Multimedia* APIs [3]. The recordings are converted to MP3 (using the LAME library) and uploaded to *Dropbox* using an application key, secret and refresh token embedded into the library binary.

FILE STEALER

Another additional component deployed in 2022 is a file stealer. Interestingly, the PDB path of this binary, `D:\GIT\Baymax\x64\Release\Baymax.pdb`, mentions Baymax, a superhero character from *Marvel* comic books. This executable accepts eight command-line arguments:

1. Remote computer address
2. Remote computer username
3. Username used for logon
4. Password used for logon
5. Domain name
6. Path to a directory on the local machine
7. *OneDrive* client ID
8. *OneDrive* refresh token.

This executable embeds an XOR encrypted list of files that need to be exfiltrated from a remote machine (its address is specified in the first argument). It uses the username, password and domain name specified in arguments 3-5 to access the remote machine. The exfiltrated files are stored on the filesystem in a directory specified in argument 6 (in case arguments 7 and 8 are empty strings) or uploaded to the *OneDrive* cloud storage (the client ID and refresh token required for uploading are provided in arguments 7 and 8).

The operators were observed to be interested in the following files:

- Cookies, form history and login data for *Edge*, *Chrome*, *Firefox* and *Opera*
- Cookies from *Threema*, *WeChat* and *WhatsApp* messengers
- Organization's confidential documents.

THE 2019 INFECTION OVERVIEW

The attack carried out in 2019 was completely different from the one described above. It involved using two frameworks – dubbed Careto2 and Goreto. The compromise of the infected machine started with deployment of Careto2. The following files were deployed to the victim machine over the compromise process:

- Framework loader (placed at `%appdata%\Media Center Programs\cversions.2.db`)
- Framework installer (named `~dfae01202c5f0dba42.cmd`)
- Auxiliary registry file (placed at `%temp%\values.reg`)

To install the Careto2 framework, the threat actor launched the `~dfae01202c5f0dba42.cmd` file from `taskeng.exe`, a process used for launching scheduled tasks. Thus, it is possible that this batch file was started by a remotely created scheduled task, just like in the 2022 infection case. This .bat file sets up the Careto2 framework by importing the `%temp%\values.reg` registry file. By doing that, it writes the following values to the registry:

- The Careto2 framework configuration, to the UserState value of the `HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer` key
- The path to the loader component of the framework (`%appdata%\Media Center Programs\cversions.2.db`), to the default value of the `HKCU\CLSID\{603d3801-bd81-11d0-a3a5-00c04fd706ec}\InProcServer` key.

By writing the latter registry value, the .bat file configures the Careto2 framework to persist on the machine via the COM hijacking technique. The hijacked CLSID represents an obsolete component of *Windows* called `SharedTaskScheduler`, which is invoked by the operating system on startup.

CARETO2 LOADER AND PLUGINS

Once launched via COM hijacking, the loader starts operating inside the `explorer.exe` process. Upon startup, it verifies that it is not running inside processes belonging to *Comodo Internet Security* (`cis.exe` or `CisTray.exe`) and reloads itself, registering in the list of loaded DLL modules as comcat.dll. It then sets up a mechanism allowing the malware to propagate into all processes started by explorer.exe by hooking the `NtCreateUserProcess` function located inside the ntdll.dll library.

Afterwards, the loader decrypts and launches plugins that perform the majority of malicious activities of the framework.

These plugins are stored in a virtual file system (VFS), located in the file `%appdata%\Media Center Programs\C_12058.NLS`. This file system is encrypted twice with Sosemanuk: the first encryption key is hard coded into the loader module, while the second one is generated from the system drive volume serial number and current user SID. The decrypted VFS consists of multiple plugin entries. The following data is provided for each plugin:

- Plugin ID, which is a 32-bit number.
- List of processes in which the plugin should be loaded. Each entry in this list is a DJB2 hash of a process name.
- List of plugin IDs that this plugin is dependent on. All plugins discovered during the research had no dependencies.
- 32-bit and 64-bit plugin bodies.

Plugin bodies are DLL files with stripped out MZ/PE headers – custom executable file headers are used instead of them. During the plugin launch process, the loader module transforms these custom headers into PE headers.

We have additionally discovered that the plugin IDs stored in this VFS are likely to be DJB2 hashes of DLL names. In the following table describing plugins, we provide possible DLL names that could have been assigned by developers for each plugin. However, these names may be wrong, as the DJB2 hash is prone to collisions.

Plugin DLL name hash	Likely DLL name	Plugin description
38568efd	ConfigMgr.dll	Provides an RPC server used by other plugins to access and modify Careto2's configuration parameters.
8f7629f1	ConfigMgrProxy.dll	Provides an RPC client that facilitates access to the previously described RPC server by exposing interfaces allowing to get, set and remove configuration values.
b6df77b6	Storage.dll	Provides an RPC server used for storing stolen files. It places files pending upload to the C2 server in a virtual file system, located in a Sosemanuk-encrypted file stored under %TMP%\~MW47YB.tmp.
d19c7b1a	StorageProxy.dll	Provides an RPC client that facilitates access to the RPC server. Its interface allows files stored inside the virtual file system to be retrieved and modified.
05962fac	Hook.dll	Provides an interface for hooking PE file imports.
8d82f0fa	KeybFilter.dll	Provides keylogging capability implemented by placing hooks of GetMessage, PeekMessage and DefWindowProc API functions using the interface of the Hook.dll plugin.
1c9f9885	Kodak.dll	Captures screenshots, writing them as files named <time>_6013.jpg to the storage VFS, using the RPC client plugin d19c7b1a (StorageProxy.dll).
5ca54969	FileFilter.dll	Monitors file modifications in specified folders. It is able to log information about these modifications and steal edited files.
82b79b83	Comm.dll	Provides an interface used for uploading files into a *OneDrive* cloud storage. To do that, it uses credentials stored with the configuration, interacting with the cloud storage using the *OneDrive* API.
861842db	DllData.dll	This plugin works only from browser processes. It periodically retrieves files from the storage VFS and uploads them to an attacker-controlled *OneDrive* storage using interfaces provided by d19c7b1a (StorageProxy.dll) and 82b79b83 (Comm.dll) plugins.

THE GORETO TOOLSET

During our analysis, we found out that machines infected with the Careto2 spyware were additionally compromised with another malicious toolset, coded in Golang. Thus, we named this toolset Goreto. It includes a set of implants that are independent from each other, namely a backdoor, a keylogger and a screenshot taker. These three implants are designed to work as a shared-process service named Svc Microsoft Service. However, instead of running implants as a service, the attackers launched them via rundll32.exe (e.g. the backdoor was launched through the rundll32.exe "%windir%\addins\hlpsvc.dll",ServiceWorkerThread" command).

This backdoor periodically (every 80-85 minutes) connects to a *Google Drive* storage with the help of a refresh token. It downloads a configuration file and a command file from this storage. The configuration file carries the name <victim ID>_cfg.bin, where the victim ID is the Base64-encoded MachineGuid value of the HKLM\SOFTWARE\Microsoft\Cryptography registry key. This file is additionally encrypted with AES-CFB, with the key 4e-<string>-rg being used. The decrypted configuration itself contains:

- The minimum and maximum allowed wait periods between consecutive C2 server interactions
- Configuration of modules launched before execution of commands received from the C2 server.

The following configuration parameters are available for these modules:

- Execution context – either run as the user owning the explorer.exe process, or as SYSTEM
- Module type – .exe or .dll (if the module type is .dll, the backdoor uses the rundll32.exe utility to launch it)
- Module source – one of the following options is used:
  - Download, decrypt, drop module file to disk and run downloaded module if it previously did not exist on disk
  - Run module previously downloaded from disk

- Download, decrypt, drop module file to disk and run downloaded module even if it previously existed on disk (this action is used to update already downloaded modules).

The command file downloaded along with the previously described configuration file is also encrypted with AES-CFB. It specifies actions that need to be executed by the backdoor. The following commands are supported:

Command name	Description
`downloadandexec`	Downloads a file from *Google Drive*, decrypts it, drops it to disk and executes it
`Downloadfile`	Downloads a file from *Google Drive*, decrypts it and drops it to disk
`Uploadfile`	Reads a specified file from disk, encrypts it and uploads it to *Google Drive*
`Exec`	Executes a specified shell command

As for the keylogger, it uses the `go-hook` open-source library that 'provides low level keyboard and mouse hook for Windows'. It collects keystrokes, then encrypts and sends them to the *Google Drive* storage as a file named `<victim ID><random string>.dat`. The encryption key, as well as *Google Drive* credentials, are the same as in the backdoor module.

Finally, the discovered screenshot module periodically (with a random interval of 5-10 minutes) takes screenshots of all active displays, which are encrypted and sent to *Google Drive* in a file named `<victim ID><random string>.png`.

Again, the encryption key and credentials are the same as in the backdoor.

ATTRIBUTION

Despite the fact that The Mask is a fairly sophisticated attacker, we have been able to spot several operational security mistakes in the group's recent operations. In particular, the file names used in 2019 attacks resembled those used by The Mask more than 10 years ago:

2007-2013 attack file names	2019 attack file names
`~df01ac74d8be15ee01.tmp`	`~dfae01202c5f0dba42.cmd`
`c_27803.nls`	`c_12058.nls`

Furthermore, the names of plugins used in the 2007-2013 and 2019 attacks are remarkably similar:

2007-2013 attack module names	2019 attack module names
`FileFlt`	`FileFilter`
`Storage`	`Storage`
`Config`	`ConfigMgr`

In addition, the campaigns conducted in the 2007-2013 and 2019 attacks overlap in terms of TTPs, mainly:

- COM hijacking is used for persistence
- The deployed malware attempts to conceal itself as a system module
- The deployed malware has a modular architecture
- The `CreateProcess` API function is used to propagate malware across running processes
- Virtual file systems are used to store additional modules, as well as data to be exfiltrated to the C2 server
- Two different frameworks are used during the attack (Careto and SGH in the 2007-2013 campaign, Careto2 and Goreto in the 2019 case).

Based on these facts, it becomes possible to attribute the activities observed in 2019 to The Mask threat actor with medium to high confidence.

As for the campaigns observed in 2022 and 2024 that both involve use of the FakeHMP implant, we also attribute them to The Mask with the same level of confidence, based on the following information:

- The organization infected in 2022 is the same one that was previously infected with both Careto2 (in 2019) and Careto (in 2007-2013)
- In both 2019 and 2022 cases, the same unique file name was used to deploy implants to infected machines: `~df01ac74d8be15ee01.tmp`
- Cloud storages are used for data exfiltration in both 2019 and 2022/2024 cases
- Deployed implants are propagated across all running processes (by hooking the `CreateProcess` API function in 2019 and by using the *HitmanPro Alert* driver in 2022 and 2024).

CONCLUSION

Over the years, the Careto APT has been developing malware bearing a remarkably high level of complexity. As such, the newly discovered implants used by this APT in 2019, 2022 and 2024 are, just like 10 years ago, complex multimodular frameworks, and the tactics and techniques exercised during their deployment are unique and sophisticated. An example of such a technique is persistence through MDaemon and *HitmanPro Alert*, and its presence indicates that The Mask implants are developed with professionalism.

While the complexity of the recently discovered attacks remained the same as 10 years ago, there has been a significant change in the architecture of the newly discovered malware. Normally, multimodular malware consists of a central orchestrator module, as well as auxiliary plugins launched by the orchestrator. However, the FakeHMP implant developed by The Mask does not include an orchestrator module. In fact, each malicious component is started via a different persistence technique, and each launched module works completely independently of the others. This complicates the process of responding to a cybersecurity incident involving The Mask because disarming all malicious modules installed on the system can become a challenging task – to some degree similar to finding a needle in a haystack.

REFERENCES

[1] Kaspersky. The Careto/Mask APT: Frequently Asked Questions. 10February 2014. https://securelist.com/thecaretomask-apt-frequently-asked-questions/58254/.

[2] United States Cyber Command. Iranian intel cyber suite of malware uses open source tools. 12 January 2022. https://www.cybercom.mil/Media/News/Article/2897570/iranian-intel-cyber-suite-of-malware-uses-open-sourcetools/.

[3] Microsoft. Windows Multimedia. https://learn.microsoft.com/en-us/windows/win32/api/_multimedia/.