LAPSUS$ is dead, long live HexaLocker?

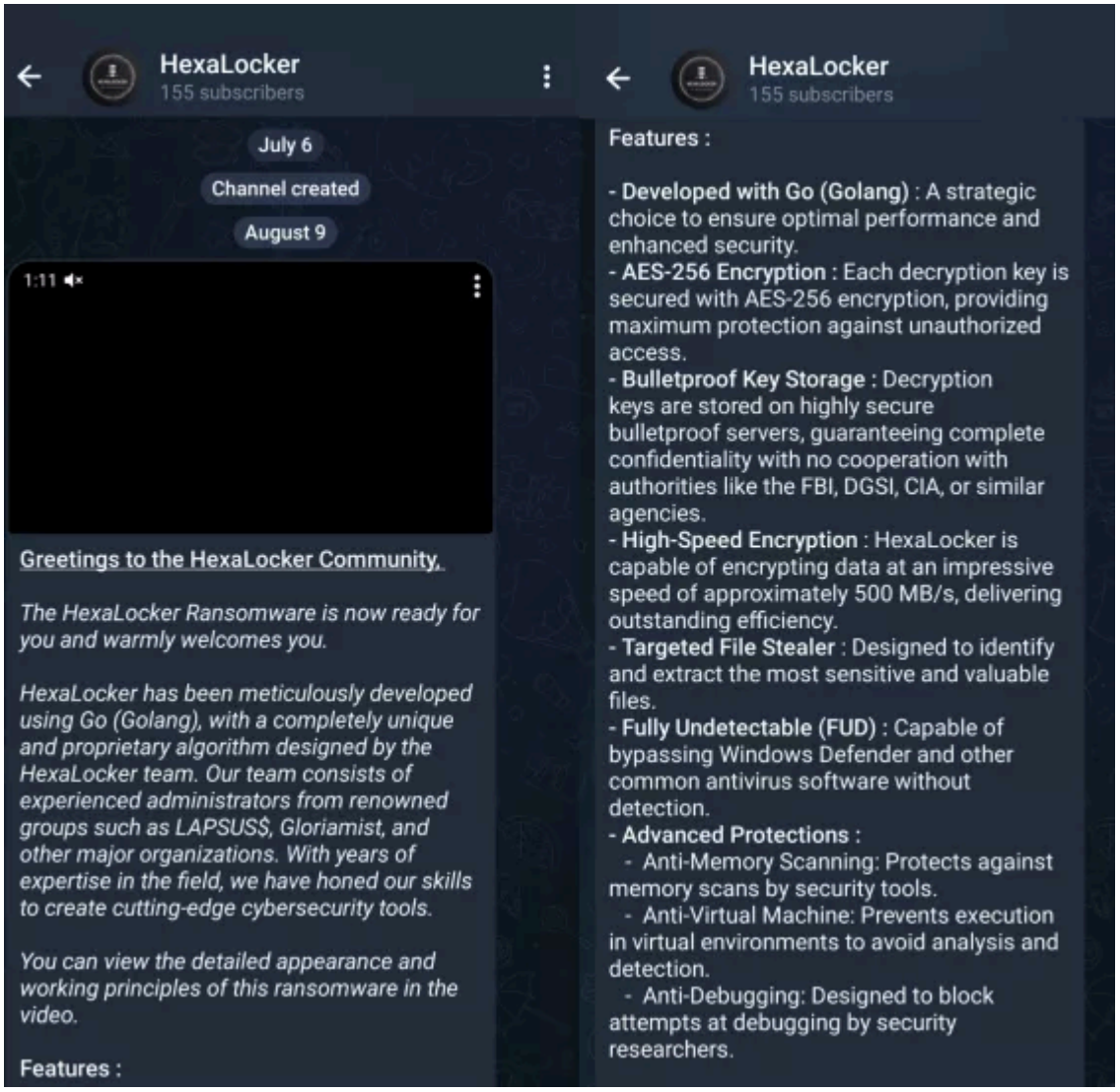
By Théo Letailleur

Archived: 2026-04-05 15:47:09 UTC

The LAPSUS$ threat group has been known since 2021 for spear phishing, data theft, and extortion against large companies (e.g., Microsoft, Nvidia, Uber). Although evidence of destruction methods was reported, there was no known use of ransomware. In June 2024, LAPSUS$ announced its closure. However, two months later, a new ransomware called HexaLocker was advertised on Telegram channels. Its "only real" admin and probable developer is ZZART3XX, one of the LAPSUS$ administrators. This article will dissect the HexaLocker ransomware sample to uncover its capabilities and help organizations that could be impacted by this new strain.

Introduction

On August 9th, 2024, the HexaLocker team advertised a new Windows ransomware on its Telegram channel. The message included a demonstration video and text promoting a Golang ransomware that implements a *proprietary algorithm*. Its features were also listed:



HexaLocker ransomware advertisement on Telegram

It seems that the HexaLocker ransomware stores decryption keys on bulletproof servers. This suggests that the decryption keys are sent to distant servers during the encryption process, but may not be protected with asymmetric cryptography (RSA, ECDH), as we would usually observe with this kind of malware. Finally, data exfiltration, FUD, and anti-analysis capabilities are promoted.

This article describes the actual capabilities of HexaLocker, based on one of its samples. A YARA rule and list of indicators are mentioned at the end of the article.

HexaLocker analysis

Since July, a few HexaLocker samples have been available on VirusTotal, showing different evolutions. The sample dissected in this article is the most recent and advanced one (for HexaLocker) in terms of features that we could find to date (August 23rd).

Basic information

HexaLocker basic information

SHA256	`87c1869871e9be8adaacb41a16c8fff691f86591416a592a77e308c4b7c041be`
File type	PE32+ executable (console) x86-64, Compiler: Go (>=1.15)
File size	7966720 bytes
Threat	Windows Golang ransomware and file stealer

The sample is not stripped, and the metadata provides the pathname of the Golang source file: `C:/Users/zzart/Desktop/MalwareDeveloppement/HexaLocker RaaS/crypter_files.go`. The username `zzart` most likely refers to ZZART3XX1, one of the LAPSUS$ administrators.

Capabilities

Here is a summary of HexaLocker's capabilities (order of execution):

1. **Anti-analysis (anti-VM, anti-debugging):** Uses an open-source Golang module called GoDefender2. We have also seen another HexaLocker sample (`be759e58413431dbe40d29ea5e399b1ebbfe75847c19a5a8f2610dab9f78ca8b`) using a different anti-VM module called chacal3.
2. **File encryption:** Recursively encrypts files in `C:\Users` with AES-256-GCM using a random password derived with Argon2ID. The ".hexalocker" extension is appended to the filename. Before encryption starts, decryption keys are AES-encrypted with a hardcoded key and sent to a remote HTTPS server via GET method parameters.
3. **Ransom:** The ransom note is created on the current user's desktop as "Readme.txt" and opened with `notepad.exe`.
4. **File stealing:** The current user's files, whose extensions correspond to specific patterns (documents, source code, databases), are compressed into a single zip file and sent to a remote HTTPS server. Since this occurs after encryption, the searched files also have the ".hexalocker" extension.

Upon execution, it opens a command prompt and outputs many debugging logs, which is not very stealthy.

Anti-analysis

Routines from GoDefender are executed early in the program in a function called `main.FortniteProtect`. The following mechanisms are used:

- **Anti-Virtualization:**
  - Detecting USB drives.
  - Checking for default virtualization usernames (e.g., Johnson, John Doe, malware, sandbox, test, etc.).
  - Checking for default virtualization video controllers (e.g., "vmware", "virtualbox").
  - Checking for KVM environment files (e.g., `baloon.sys`, `netkvm.sys`).
  - Detecting triage environments based on disk drive names (e.g., `DADY HARDDISK`, `QEMU HARDDISK`).

    - Detecting small monitors (resolutions under 800×600 are suspicious).
    - Detecting VM artifacts (VMware and VirtualBox drivers or guest tools).
    - Detecting repetitive processes (more than 60 `svchost.exe` processes are suspicious).
    - Detecting parallel virtualization environments (guest tools).
- **Anti-Debug:**
    - Detecting common hooked functions used for anti-anti-debugging (e.g., `CheckRemoteDebuggerPresent`, `GetTickCount`, etc.).
    - Checking for blacklisted window names (e.g., IDA, ILSpy, Fiddler, x32dbg, etc.).
    - Using the classic `IsDebuggerPresent` and `CheckRemoteDebuggerPresent` routines.
    - Performing an internet connection check (TCP request to `google.com:80`).
    - Checking the parent process (must be `explorer.exe` or `cmd.exe`).
    - Verifying the number of processes (must be above 50).
    - Checking uptime (must be above 20 minutes).

If you need more technical information, most of those mechanisms are detailed in the great Unprotect Project[4]. Though, since they are all packed into one calling function, the bypass is trivial with a debugger, but it might prevent execution on automatic sandboxes.

File encryption

Now, for the pièce de résistance: HexaLocker's file encryption capability.

First, a 50-byte random alphanumeric password is generated, followed by a 16-byte random salt. This password is then derived using the Argon2ID algorithm with the following parameters:

- Threads: 4
- Iterations: 3
- Memory: 65536 KiB
- Generated hash length: 32 bytes
- And the previously generated salt

Before the encryption starts, an HTTPS request is made to a distant server to send the password, the salt, and some host information for identification purposes. This allows the ransomware operator to obtain the necessary cryptographic parameters to decrypt the files. The HTTP request is sent to `https://darkslategray-baboon-853641.hostingersite[.]com/index.php`, and the information is stored in the GET parameters. The password and salt are only AES-GCM encrypted with a hardcoded key in the sample.

Below is a capture of such an HTTP request (intercepted with FakeNet tool):

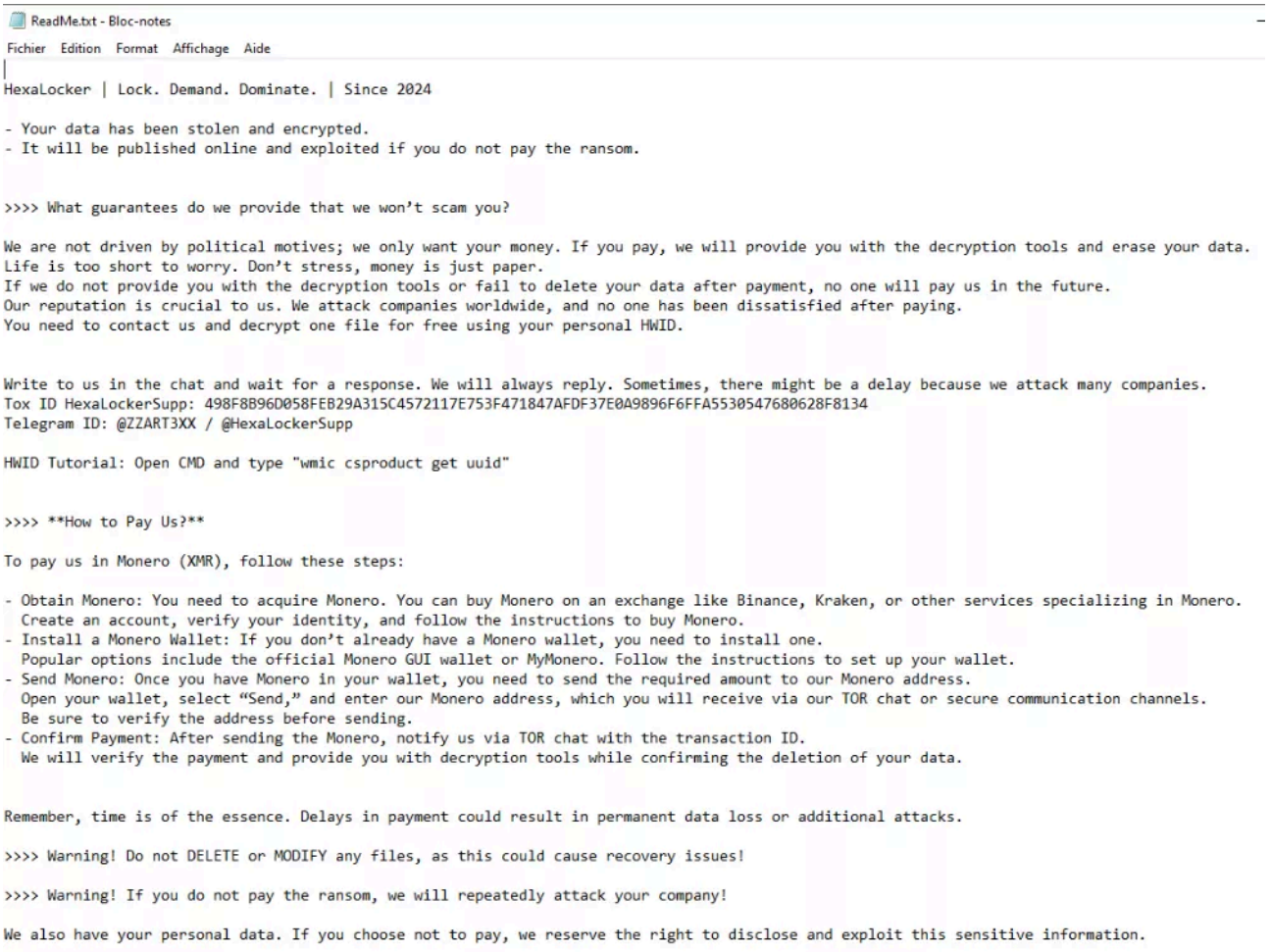
```
 08/23/24 04:30:33 AM [        DNS Server] Received A request for domain 'darkslategray-baboon-853641.hostingers
08/23/24 04:30:33 AM [   HTTPListener443]   GET /index.php?method=new&hwid=9F905EA7-EBD1-4D49-84F6-AE84E484E49F8
08/23/24 04:30:33 AM [   HTTPListener443]   Host: darkslategray-baboon-853641.hostingersite.com
08/23/24 04:30:33 AM [   HTTPListener443]   User-Agent: Go-http-client/1.1
08/23/24 04:30:33 AM [   HTTPListener443]   Accept-Encoding: gzip
08/23/24 04:30:33 AM [   HTTPListener443]
```

The values in the *password* and *sel* fields (sel stands for salt in French) are 28 bytes longer than expected. This is because they also contain the AES-GCM nonce (12 bytes) and the AES-GCM tag (16 bytes) (`nonce + ciphertext + tag`).

Once the important cryptographic values are stored on the operator's server, the encryption process begins. This sample uses `C:\Users` as the root directory and encrypts every file whose name matches a list of extensions. The extension list is quite extensive and encrypts nearly all file types. The targeted files are encrypted using AES-GCM. The 32-byte Argon2ID hash is used as the key with a 12-byte random nonce. Every encrypted file is saved on disk with the ".hexalocker" extension. The encrypted data also includes the nonce and the tag.

Since there is no asymmetric cryptography involved, if a victim recovers the ransomware sample and has TLS proxy logs, they can decrypt and recover the files. We developed a proof-of-concept (PoC) program that successfully recovers the encrypted files based on the password, salt, and hardcoded key stored in the sample. We can provide the code for this program upon request.

Finally, the ransom note is generated as a "Readme.txt" file on the current user's desktop and is automatically opened with `notepad.exe` .



```
ReadMe.txt - Bloc-notes
Fichier  Edition  Format  Affichage  Aide

HexaLocker | Lock. Demand. Dominate. | Since 2024

- Your data has been stolen and encrypted.
- It will be published online and exploited if you do not pay the ransom.


>>>> What guarantees do we provide that we won't scam you?

We are not driven by political motives; we only want your money. If you pay, we will provide you with the decryption tools and erase your data.
Life is too short to worry. Don't stress, money is just paper.
If we do not provide you with the decryption tools or fail to delete your data after payment, no one will pay us in the future.
Our reputation is crucial to us. We attack companies worldwide, and no one has been dissatisfied after paying.
You need to contact us and decrypt one file for free using your personal HWID.


Write to us in the chat and wait for a response. We will always reply. Sometimes, there might be a delay because we attack many companies.
Tox ID HexaLockerSupp: 498F8B96D058FEB29A315C4572117E753F471847AFDF37E0A9896F6FFA5530547680628F8134
Telegram ID: @ZZART3XX / @HexaLockerSupp

HWID Tutorial: Open CMD and type "wmic csproduct get uuid"


>>>> **How to Pay Us?**

To pay us in Monero (XMR), follow these steps:

- Obtain Monero: You need to acquire Monero. You can buy Monero on an exchange like Binance, Kraken, or other services specializing in Monero.
  Create an account, verify your identity, and follow the instructions to buy Monero.
- Install a Monero Wallet: If you don't already have a Monero wallet, you need to install one.
  Popular options include the official Monero GUI wallet or MyMonero. Follow the instructions to set up your wallet.
- Send Monero: Once you have Monero in your wallet, you need to send the required amount to our Monero address.
  Open your wallet, select "Send," and enter our Monero address, which you will receive via our TOR chat or secure communication channels.
  Be sure to verify the address before sending.
- Confirm Payment: After sending the Monero, notify us via TOR chat with the transaction ID.
  We will verify the payment and provide you with decryption tools while confirming the deletion of your data.


Remember, time is of the essence. Delays in payment could result in permanent data loss or additional attacks.

>>>> Warning! Do not DELETE or MODIFY any files, as this could cause recovery issues!

>>>> Warning! If you do not pay the ransom, we will repeatedly attack your company!

We also have your personal data. If you choose not to pay, we reserve the right to disclose and exploit this sensitive information.
```

File stealing

HexaLocker searches for files in the current user's directories: Desktop, Documents, Favorites, Pictures, and Videos. The search pattern involves a list of extensions with the ".hexalocker" suffix, as these files are already encrypted. Below is the list of extensions (split into categories for better readability):

- Documents: .txt, .doc, .docx, .odt, .xls, .xlsx, .ods, .ppt, .pptx, .odp, .rtf, .md, .tex, .wps, .pages,
- Data : .sql, .mdb, .accdb, .sqlite, .db, .dbf, .json, .csv, .numbers, .dif, .key, .plist, .trace, .tmp,
- Source code: .py, .java, .c, .cpp, .js, .html, .css, .ruby, .php, .swift, .go, .r, .asp, .jsp,
- Configuration files: .ini, .cfg, .log, .xml, .yaml, .yml,
- Archives: .zip, .rar, .tar, .gz, .7z, .bz2, .lz, .xz,
- Pictures: .jpg, .jpeg, .png, .gif, .bmp, .tiff,
- Videos: .mp3, .wav, .avi, .mp4, .mov

All the corresponding files are added to a new zip file named with the host UUID, located in the `%Temp%` folder. The zip file is then sent to the remote server via a single POST HTTPS request. The PHP file in the URL is different from the previous one: `https://darkslategray-baboon-853641.hostingersite[.]com/receive.php`.

Below is a capture of such an HTTP request (intercepted with FakeNet tool):

```
 08/23/24 05:20:13 AM [        DNS Server] Received A request for domain 'darkslategray-baboon-853641.hostingers
08/23/24 05:20:14 AM [   HTTPListener443]   POST /receive.php HTTP/1.1
08/23/24 05:20:14 AM [   HTTPListener443]   Host: darkslategray-baboon-853641.hostingersite.com
08/23/24 05:20:14 AM [   HTTPListener443]   User-Agent: Go-http-client/1.1
08/23/24 05:20:14 AM [   HTTPListener443]   Content-Length: 36618
08/23/24 05:20:14 AM [   HTTPListener443]   Content-Type: multipart/form-data; boundary=a9cf213cc8cf5b6fffefc664
08/23/24 05:20:14 AM [   HTTPListener443]   Accept-Encoding: gzip
08/23/24 05:20:14 AM [   HTTPListener443]
08/23/24 05:20:14 AM [   HTTPListener443]   --a9cf213cc8cf5b6fffefc664759d93533d3ed8975bb56ad6f6603dd6414d
08/23/24 05:20:14 AM [   HTTPListener443]   Content-Disposition: form-data; name="file"; filename="9F905EA7-EBD
08/23/24 05:20:14 AM [   HTTPListener443]   Content-Type: application/octet-stream
08/23/24 05:20:14 AM [   HTTPListener443]
08/23/24 05:20:14 AM [   HTTPListener443]  .C:\Users\bonjour\Desktop\loremipsum.txt.hexalocker@|-h#!.m,~bO@dGYx
:nfDЋhfUY{w7fttg0hH-}i   #^F_i(,aryNچfL  .0%#MqvAi]espPeوF.Me%<IpHOPK")$dx       X&uk˜

[... Compressed and encrypted data ...]
```

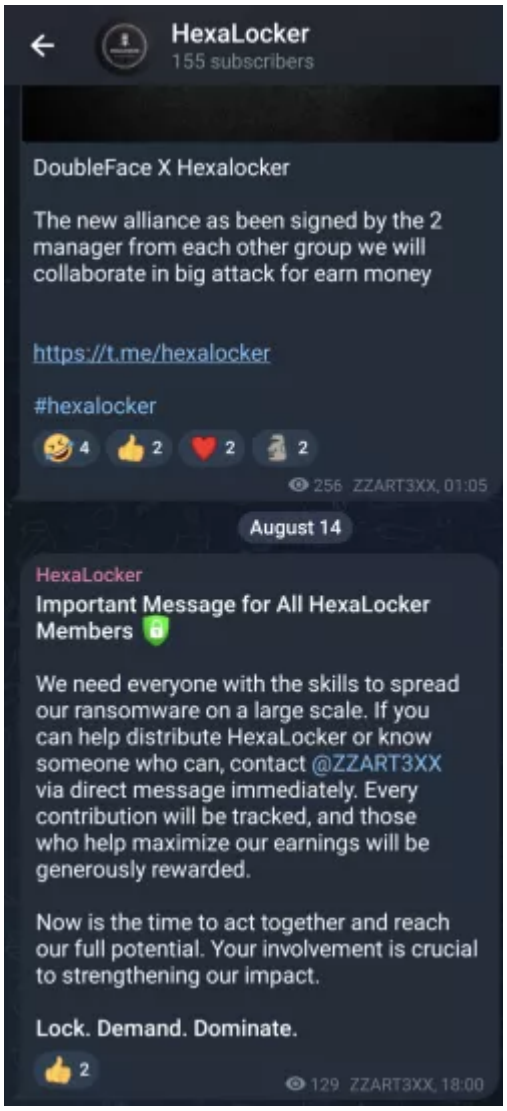
We did not stress-test the server, but the HTTP request could become quite large depending on the size and number of documents located in the victim's user profile.

Conclusion

HexaLocker is a Golang ransomware that currently operates only on Windows operating systems. It encrypts files with AES-256-GCM using a random password derived with Argon2ID. Decryption keys are then AES-encrypted with a hardcoded key and finally sent to a remote HTTPS server. No asymmetric cryptography is involved. HexaLocker also includes file-stealing capabilities. Additionally, the developer has used the open-source module GoDefender to protect the code from dynamic analysis and debugging.

There are certainly expected evolutions in HexaLocker's code, but we can already observe the main features and behaviour of this new strain, as revealed by the analysed sample. If HTTPS requests are intercepted (e.g., with a corporate proxy) the password and salt can be collected and used to decrypt affected files. To perform the decryption, the PoC program we developed can be reused, though it may need some adjustments, as we anticipate the hardcoded keys will change in future builds.

Moreover, HexaLocker's impact on organizations could be imminent, as its team has announced an alliance with a new ransomware gang called DoubleFace to "*collaborate on big attacks*". They are also seeking other partners to help them spread their ransomware on a large scale.



HexaLocker teaming up with DoubleFace

The X (formerly Twitter) handle @ZZART3XX is mentioned, describing themselves as an administrator of (formerly) LAPSUS$, GLORIAMIST, and HexaLocker.

You can find a YARA rule and a list of IOCs in this GitHub repository: https://github.com/synacktiv/hexalocker-analysis

If any organization requires assistance in doubt removal or responding to a compromise, please feel free to contact Synacktiv.
