Hive Ransomware: Actively Targeting Hospitals

netskope.com/blog/hive-ransomware-actively-targeting-hospitals

Gustavo Palazolo

September 10, 2021

Summary

Most ransomware groups operating in the RaaS (Ransomware-as-a-Service) model have an internal code of ethics that includes avoiding breaching some specific sectors, such as hospitals or critical infrastructure, thus avoiding great harm to society and consequently drawing less attention from law enforcement. For example, the BlackMatter ransomware states they are not willing to attack hospitals, critical infrastructure, defense industry, non-profit companies, and oil and gas industry targets, having learned from the mistakes of other groups, such as DarkSide, who shut down its operations after the Colonial Pipeline attack.

However, this code of ethics is not always adopted by attackers, as is the case with Hive, a new family of ransomware discovered in June 2021. On August 15, 2021, Hive ransomware was responsible for an attack against the Memorial Health System, a non-profit integrated health system with three hospitals in Ohio and West Virginia (Marietta Memorial Hospital, Selby General Hospital, and Sistersville General Hospital), causing radiology exams and surgical cases to be canceled. According to the FBI, the group uses phishing emails with malicious attachments to gain access into networks, allowing the attackers to move laterally over the network to steal data and infect more machines.

HiveLeaks

In addition to encrypting files, Hive also steals sensitive data from networks, threatening to publish everything in their HiveLeak website, hosted on the deep web, which is a common practice among ransomware working in this double extortion scheme.

There are two websites maintained by the group, the first one is protected by username and password, accessible only by the victims who obtain the credentials in the ransom note.

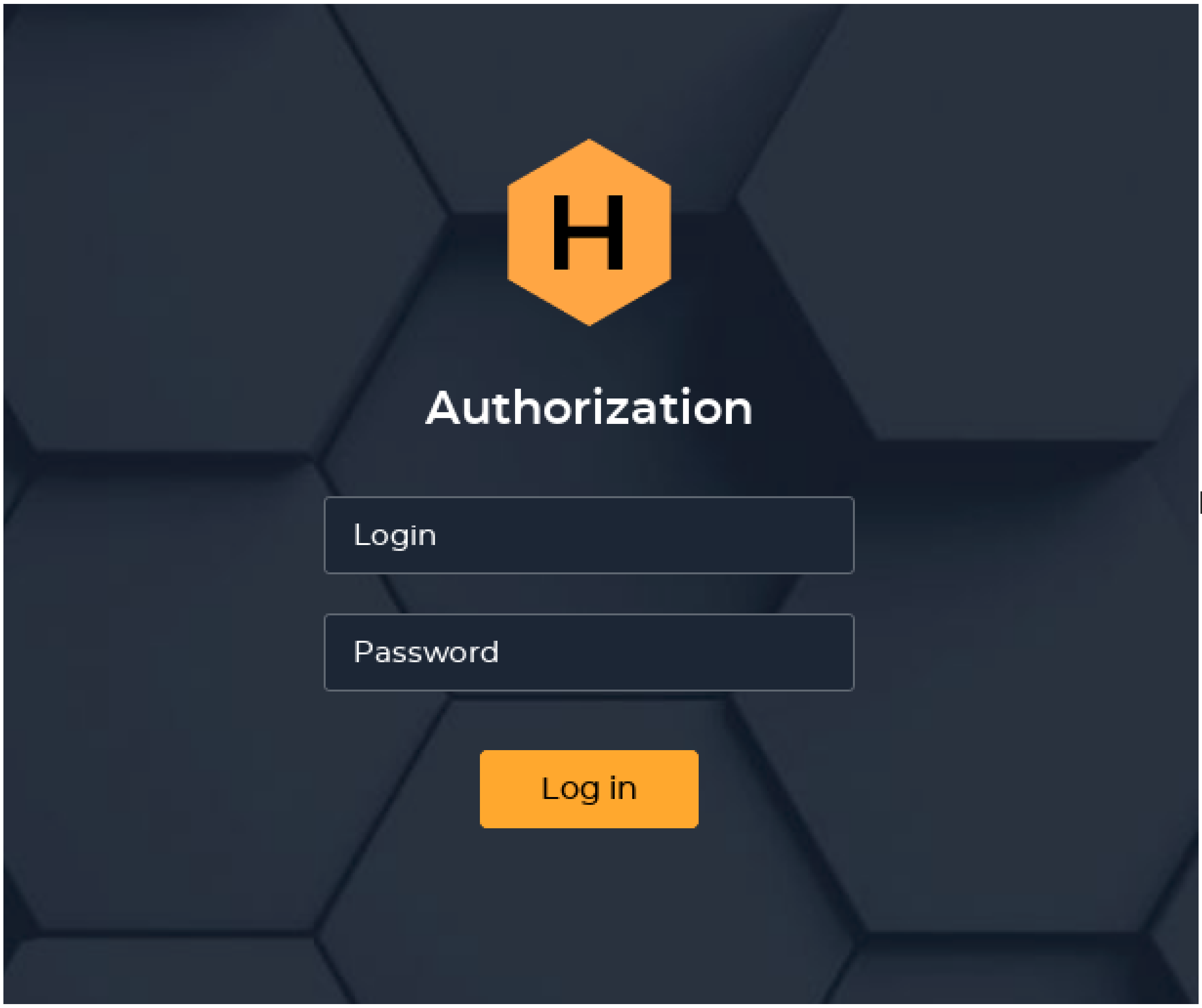


Figure 01. Hive ransomware private website

Once authenticated, the victim can see:

1. The name of the infected organization;
2. A live chat, where the victim can interact with the attackers;
3. A file upload system, where the victim can send files to the attackers;
4. A link to Hive's decryption software, if the ransom is paid by the victims.

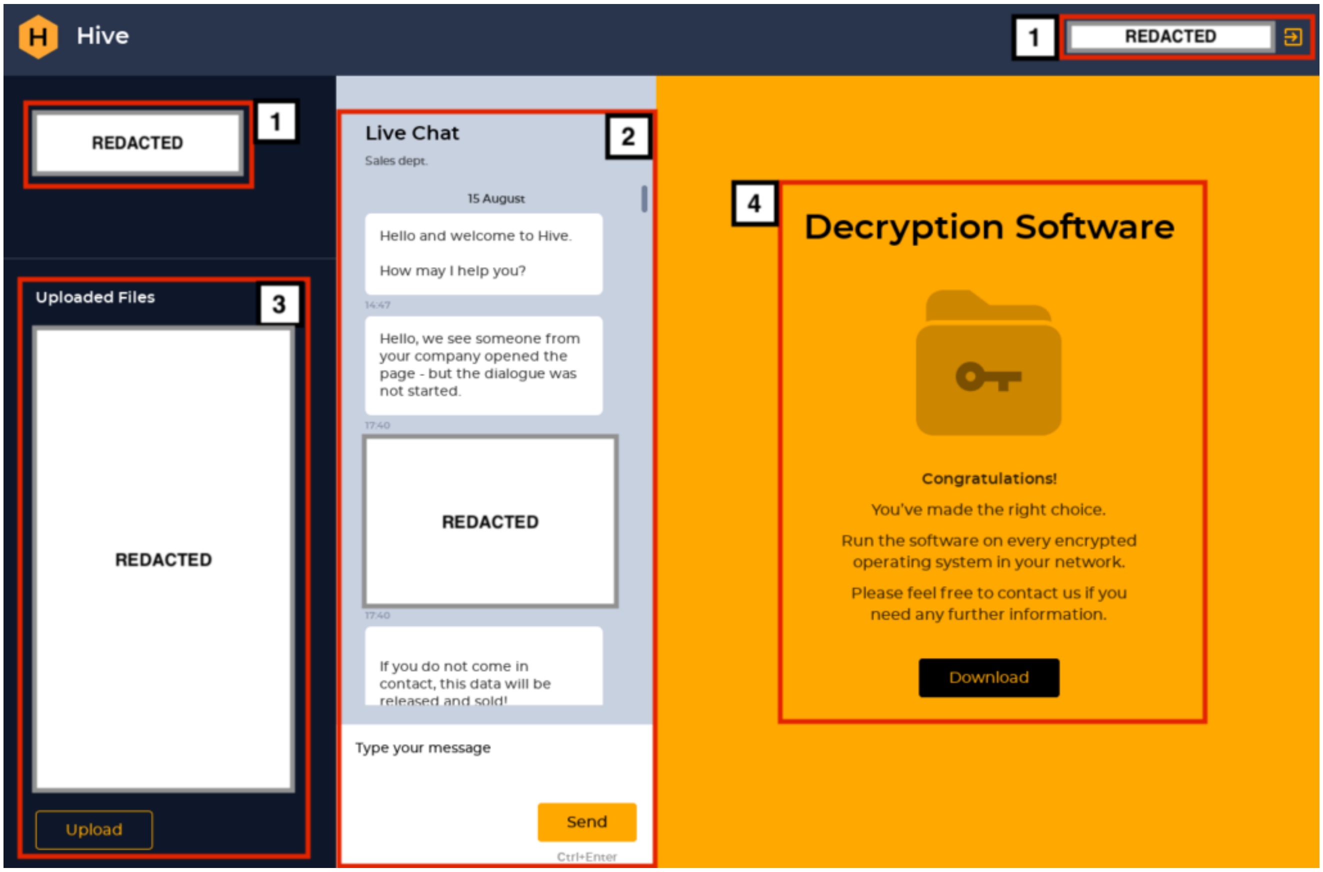


Figure 02. Victim's private website by Hive ransomware

The second website, "HiveLeaks," is where the attackers publish data about their targets and is publicly accessible.

Figure 03. "HiveLeaks" logo.

For each target, you can see the name, a small description, the website, the revenue, and the number of employees at the company. Also, you can see two dates, when the files were encrypted and when the attack was made public. Curiously enough, there are also two social media buttons where you can share this information.

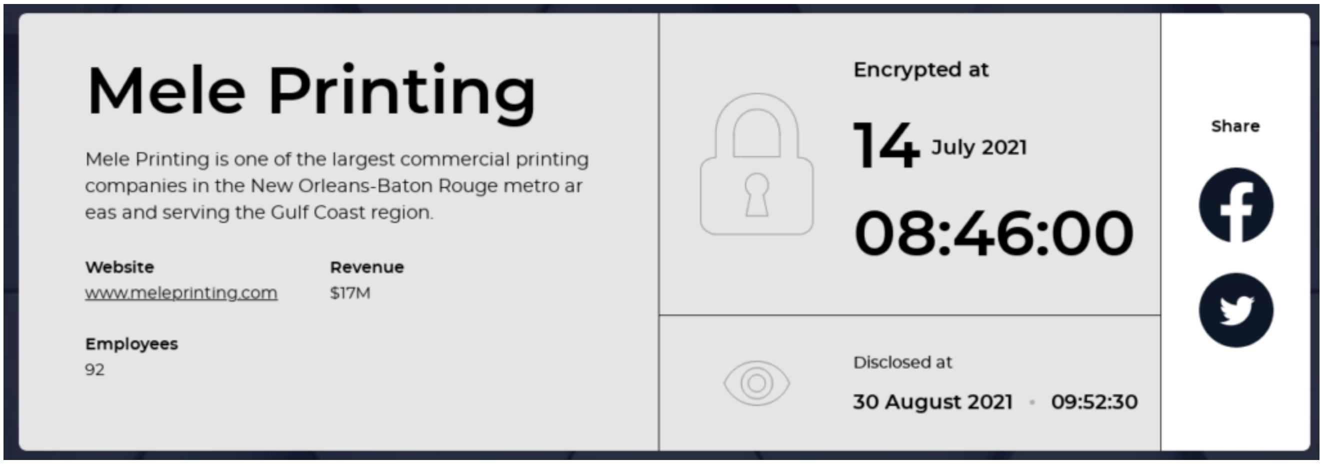


Figure 04. Information about the infected company on the “HiveLeaks” website.

If any data is published by the attackers, you will also find a link where the files can be downloaded. Hive uses common file-sharing services for this purpose, such as PrivatLab, AnonFiles, MEGA, UFile, SendSpace, and Exploit.in, as shown in Figure 05.

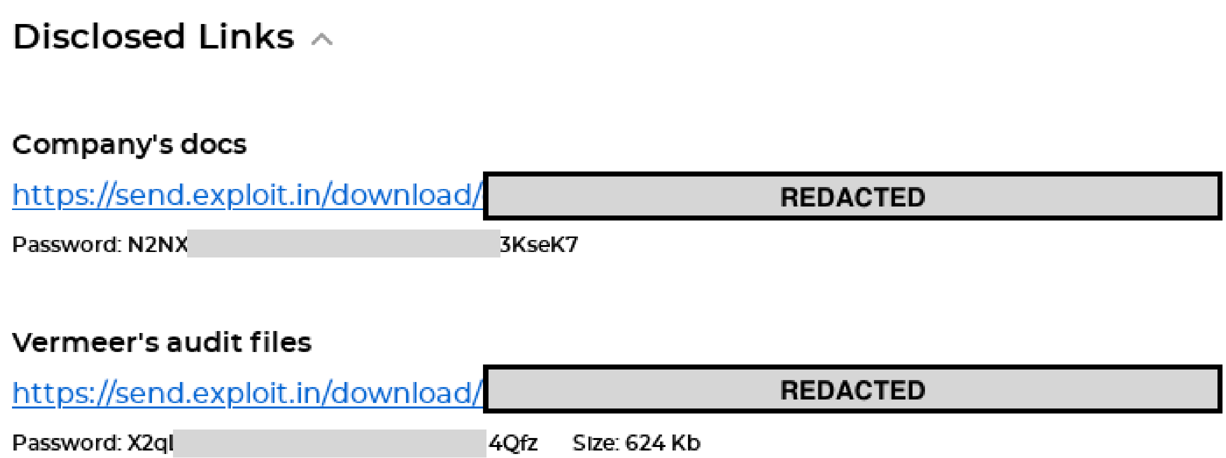


Figure 05. Links to download stolen data by Hive.

Memorial Health System Attack

The Hive ransomware infected the Memorial Health System (MHS) on August 15, 2021. The attackers claim to have stolen patient data including names, social security numbers, dates of birth, addresses and phone numbers, and medical histories for 200,000 patients, and an additional 1.2 TB of other data.

MHS tried to appeal to the attackers to provide the decrypter for free but ultimately ended up paying 1.8M, divided equally into two Bitcoin wallets. The attackers moved the Bitcoins to another wallet just a few minutes after the transaction was made by MHS.

Aside from the decryptor, the attackers also promise a security report, a file tree describing all stolen data, and the logs proving that they had erased everything from their servers.

Analysis

The ransomware was written in Go, an open-source programming language that allows cross-compilation, meaning that the same source code can be compiled to different OS, such as Linux, Windows, and macOS.

Although we have only seen Windows versions in the wild at this point, we have strong indications that the group is able to infect other systems such as Linux, as well as the Hypervisor ESXi, as we will demonstrate later in the analysis.

We have analyzed two different samples, being 32 and 64-bit Windows versions of the malware. Both of them are packed with UPX, which is an open-source executable packer.

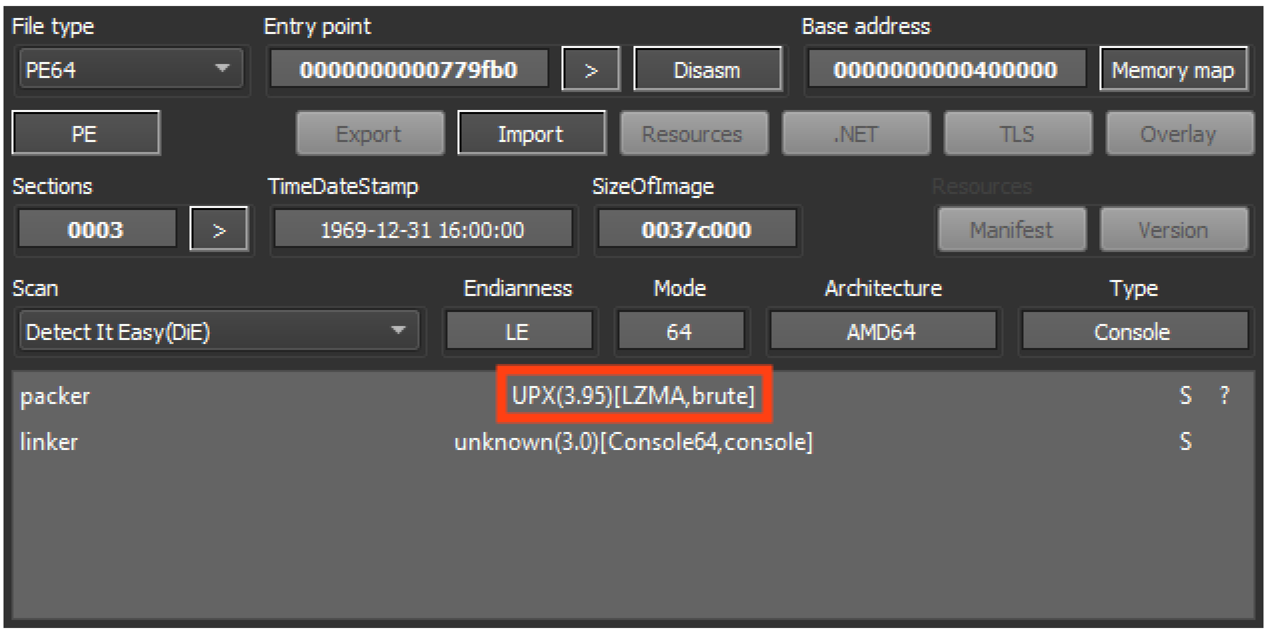


Figure 06. Main Hive ransomware payload, packed with UPX.

The first thing we noticed is that both samples we analyzed had a command line interface (CLI), accepting parameters and also showing log messages throughout the malware execution.

The **64-bit** sample accepts two parameters:

- **kill:** Kill processes specified as value (case insensitive regex)
- **stop:** Stop services specified as value (case insensitive regex)

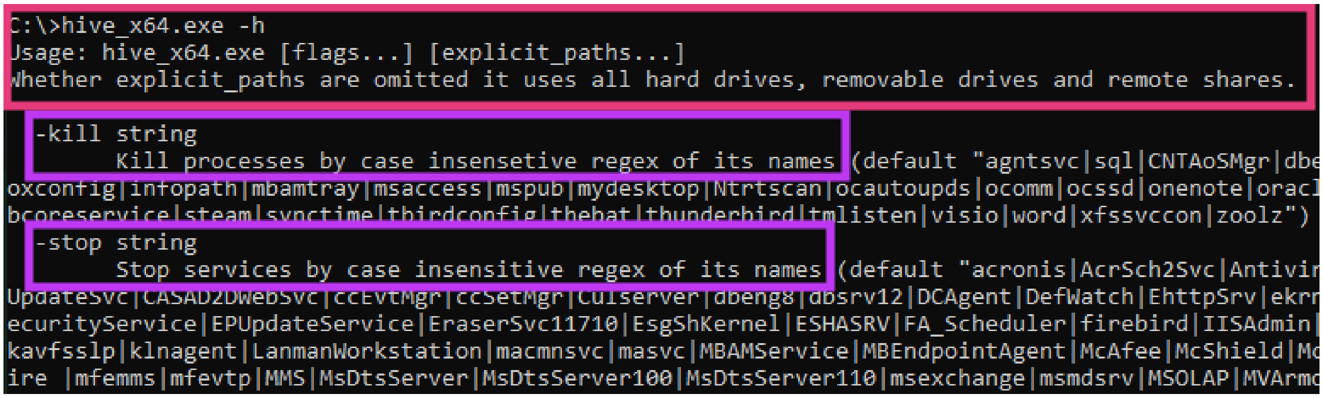


Figure 07. Parameters accepted by the 64-bit sample of Hive.

On the other hand, the **32-bit** sample offers three more options:

- **kill:** Kill processes specified as value (case insensitive regex)
- **no-clean:** Do not clean disk space (described later in this analysis)
- **skip:** Files that the attacker doesn’t want to encrypt (case insensitive regex)
- **skip-before:** Skips files created before the specified date.

stop: Stop services specified as value (case insensitive regex)

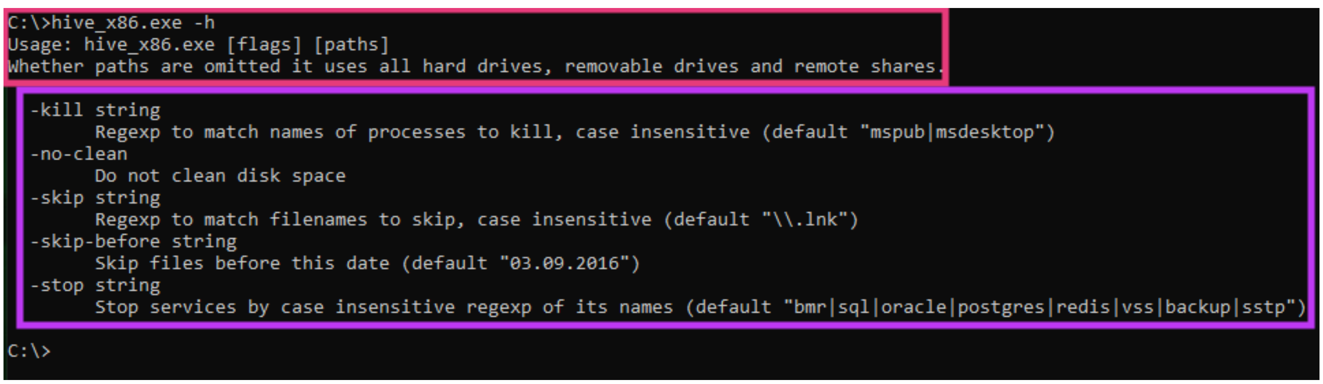


Figure 08. Parameters accepted by the 32-bit sample of Hive.

Aside from the parameters above, the attacker can also specify the path containing the files that need to be encrypted. If this path isn’t specified, the ransomware will list all the files in the machine, skipping the ones specified in the **“-skip”** and **“-skip-before”** parameters.

For analysis purposes, we have created a folder named **“C:\to_encrypt”**, containing three different pictures. Once executed, the ransomware starts printing out log messages throughout the whole encryption process.

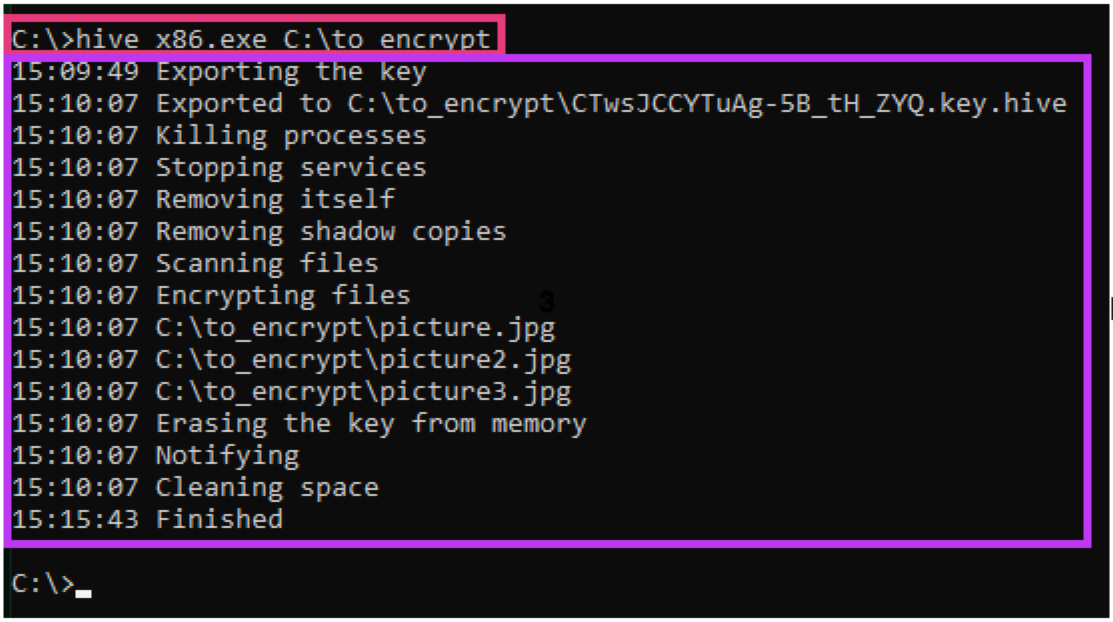


Figure 09. 32-bit Hive ransomware execution.

The log messages show pretty much everything the malware is doing, however, let's take a look at each one of the aspects being printed out.

Analyzing this 32-bit sample closely, we can see some of the function names parsed by the disassembler, from a package the attackers named as "google.com", perhaps as an attempt to deceive the analyst.

```
Function name
google_com_config_pubkeys_RSAPublicKeys
google_com_encryptor_NewApp
google_com_encryptor__ptr_App__BasicPaths
google_com_encryptor__ptr_App__CleanSpace
google_com_encryptor__ptr_App__EncryptFile
google_com_encryptor__ptr_App__EncryptFiles
google_com_encryptor__ptr_App__EraseKey
google_com_encryptor__ptr_App__ExportKey
google_com_encryptor__ptr_App__KillProcess
google_com_encryptor__ptr_App__KillProcesses
google_com_encryptor__ptr_App__Notify
google_com_encryptor__ptr_App__Notify_func1
google_com_encryptor__ptr_App__Notify_func1_1
google_com_encryptor__ptr_App__PreNotify
google_com_encryptor__ptr_App__PreNotify_func1
google_com_encryptor__ptr_App__PrepareToEncrypt
google_com_encryptor__ptr_App__RemoveItself
google_com_encryptor__ptr_App__RemoveShadowCopies
google_com_encryptor__ptr_App__Run
google_com_encryptor__ptr_App__ScanFiles
google_com_encryptor__ptr_App__ScanFiles_func1
google_com_encryptor__ptr_App__ScanFiles_func1_1
google_com_encryptor__ptr_App__StopServices
google_com_encryptor__ptr_App__encryptFilesGroup
google_com_encryptor_cleanSpaceGroup
```

Figure 10. 32-bit Hive function names.

First, the malware calls a function **encryptor**.**NewApp().**

```
mov     [esp+0F0h+var_BC], ecx
mov     [esp+0F0h+var_B8], edx
mov     [esp+0F0h+var_B4], eax
call    google_com_encryptor_NewApp
mov     eax, [esp+0F0h+var_B0]
mov     [esp+0F0h+var_80], eax
```

Figure 11. “NewApp” Hive function.

Simply put, this function initializes some important data used by the ransomware, such as the primary key.

```
sub     esp, 14Ch
call    google_com_keys_NewPrimaryKey
mov     eax, [esp+14Ch+var_14C]
mov     [esp+14Ch+var_A4], eax
mov     ecx, [esp+14Ch+var_144]
mov     [esp+14Ch+var_100], ecx
mov     edx, [esp+14Ch+var_148]
mov     [esp+14Ch+var_104], edx
lea     edi, [esp+14Ch+var_A0]
xor     eax, eax
call    loc_45AA0E
lea     ecx, a00010203040506+0C8h ; "Your network has been breached and all "...
mov     [esp+14Ch+var_A0], ecx
mov     [esp+14Ch+var_9C], 109h
mov     ecx, off_61B570 ; "http://hivecust6vhekztbqgdnkks64ucehqac"...
mov     ebp, dword_61B574
mov     [esp+14Ch+var_98], ecx
mov     [esp+14Ch+var_94], ebp
lea     ecx, aLogin      ; "\r\n      Login: "
```

Figure 12. “NewApp” function flow.

The function **keys.NewPrimaryKey()** generates a 10 MB random key used in the encryption process.

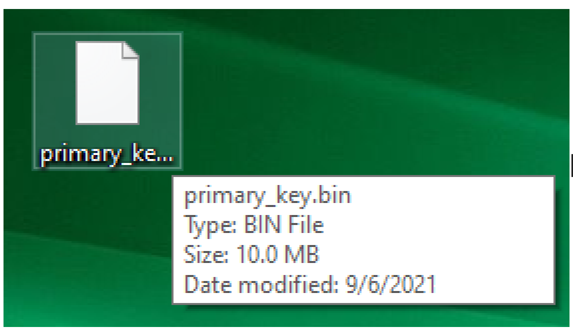


Figure 13. 10 MB key generated by Hive.

Once the key is generated, the ransom note and a batch script are loaded into memory, which will be eventually saved to the disk during the process.

After this setup is completed, the ransomware calls a function named **App.Run()**, which starts the flow we saw in the log messages.

```
loc_508E78:
mov     eax, [esp+0F0h+var_80]
mov     [esp+0F0h+var_F0], eax
call    google_com_encryptor___ptr_App__Run
mov     eax, [esp+0F0h+var_EC]
mov     ecx, [esp+0F0h+var_EC]
mov     edx, [esp+0F0h+var_E8]
test    eax, eax
jz      short loc_508EDE
```

Figure 14. Hive “Run” function.

The first function called inside **App.Run()** is **App.ExportKey()**.

```
sub     esp, 1Ch
mov     eax, [esp+1Ch+arg_0]
mov     [esp+1Ch+var_1C], eax
call    google_com_encryptor___ptr_App__ExportKey
mov     eax, [esp+1Ch+var_18]
mov     ecx, [esp+1Ch+var_18]
mov     edx, [esp+1Ch+var_14]
test    eax, eax
jnz     loc_505F28
```

Figure 15. “ExportKey” function.

This function is responsible for encrypting the 10 MB key generated by **keys.NewPrimaryKey().**

```
call    log_Println
call    google_com_config_pubkeys_RSAPublicKeys
mov     eax, [esp+84h+arg_0]
mov     ecx, [eax]
mov     edx, [eax+4]
mov     ebx, [eax+8]
mov     ebp, [esp+84h+var_84]
mov     esi, [esp+84h+var_80]
mov     edi, [esp+84h+var_7C]
mov     [esp+84h+var_84], ecx
mov     [esp+84h+var_80], edx
mov     [esp+84h+var_7C], ebx
mov     [esp+84h+var_78], ebp
mov     [esp+84h+var_74], esi
mov     [esp+84h+var_70], edi
call    google_com_keys_PrimaryKey_Export
mov     eax, [esp+84h+var_60]
```

```
dd offset ptr_RSA_Keys
dd 100
```

Figure 16. Main flow of “ExportKey” function.

Hive contains 100 public RSA keys embedded in the binary, which are used to encrypt the key generated previously. They are all parsed through the function **ParsePKCS1PublicKey** from the pkcs1.go library.

Disassembly	Hits
call <hive_x86.ParsePKCS1PublicKey>	100

Figure 17. Hive ransomware loading public RSA keys.

The malware then encrypts the data using the **EncryptOAEP** function from the rsa.go library.

```
mov     [esp+0B4h+var_90], 0
mov     [esp+0B4h+var_8C], 0
call    crypto_rsa_EncryptOAEP
```

Figure 18. Hive encrypting the key using RSA.

The encrypted key is then saved into a file that ends with ".key.hive" extension (or "key.<random>" for the 64-bit version). This is the file that is eventually loaded by the decryptor to retrieve the encryption key used in the process.

CTwsJCCYTuAg-5B_tH_ZYQ.key.hive

Figure 19. Key file saved by Hive during the process.

After creating the encrypted key, the malware calls two functions named **App.KillProcesses()** and **App.StopServices()**.

```
mov     [esp+1Ch+var_1C], eax
call    google_com_encryptor___ptr_App__KillProcesses
mov     eax, [esp+1Ch+arg_0]
mov     [esp+1Ch+var_1C], eax
call    google_com_encryptor___ptr_App__StopServices
mov     eax, [esp+1Ch+arg_0]
```

Figure 20. Hive functions to kill processes and stop services.

The name of these functions are self-explanatory, and the full list of default values for stopped processes and services can be found in our GitHub repository.

Next, Hive executes the functions **App.RemoveItself()** and **App.RemoveShadowCopies()**.

```
mov     [esp+1Ch+var_1C], eax
call    google_com_encryptor___ptr_App__RemoveItself
mov     eax, [esp+1Ch+arg_0]
mov     [esp+1Ch+var_1C], eax
call    google_com_encryptor___ptr_App__RemoveShadowCopies
mov     eax, [esp+1Ch+arg_0]
```

Figure 21. Next two functions executed by the "Run".

The first one is responsible for creating a batch script that was loaded into memory by the function **encryptor**.**NewApp().** The purpose of this script is to delete the ransomware payload once this process is done.

hive.bat

```
:Repeat
timeout 1 || sleep 1
del "C:\hive_x86.exe"
if exist "C:\hive_x86.exe" goto Repeat
del "hive.bat"
```

Figure 22. Batch script created by Hive to delete the payload from disk

The second function creates another batch script in disk that is responsible for deleting Windows Shadow Copies, to prevent any file restoration.

```
vssadmin.exe delete shadows /all /quiet
del shadow.bat
```

Figure 23. “shadow.bat” script created by the 32-bit Hive.

Here, we have a big difference between the two samples we have analyzed. Instead of creating a batch script, the 64-bit version we found uses several commands to delete not only the Windows Shadow Copies, but also to stop services, including Windows Defender.

```
schtasks.exe /Change /TN "Microsoft\Windows\ExploitGuard\ExploitGuard MDM policy Refresh" /Disable
schtasks.exe /Change /TN "Microsoft\Windows\Windows Defender\Windows Defender Cache Maintenance" /Disable
schtasks.exe /Change /TN "Microsoft\Windows\Windows Defender\Windows Defender Cleanup" /Disable
schtasks.exe /Change /TN "Microsoft\Windows\Windows Defender\Windows Defender Scheduled Scan" /Disable
schtasks.exe /Change /TN "Microsoft\Windows\Windows Defender\Windows Defender Verification" /Disable
vssadmin.exe delete shadows /all /quiet
```

Figure 24. Commands executed by the 64-bit Hive sample we analyzed.

The full list of commands executed by the 64-bit version can be found in our GitHub repository.

Next in the flow, we have two important functions:

```
mov     [esp+1Ch+var_1C], eax
call    google_com_encryptor___ptr_App__ScanFiles
mov     eax, [esp+1Ch+arg_0]
mov     [esp+1Ch+var_1C], eax
call    google_com_encryptor___ptr_App__EncryptFiles
```

Figure 30. “ScanFiles” and “EncryptFiles” functions of Hive.

App.ScanFiles() is responsible for fetching all the files that will be encrypted by the ransomware. Also, this function creates the ransom note in disk, which was already loaded in memory previously.

App.EncryptFiles() does exactly what the name describes. Within that function, the code is calling another two, respectively **encryptFilesGroup()** and **EncryptFile()**, loading the contents of the targeted file in memory, encrypting the data with what seems to be a custom algorithm created by Hive developers. Then, the encrypted file is written into disk, using the extension “.hive”.

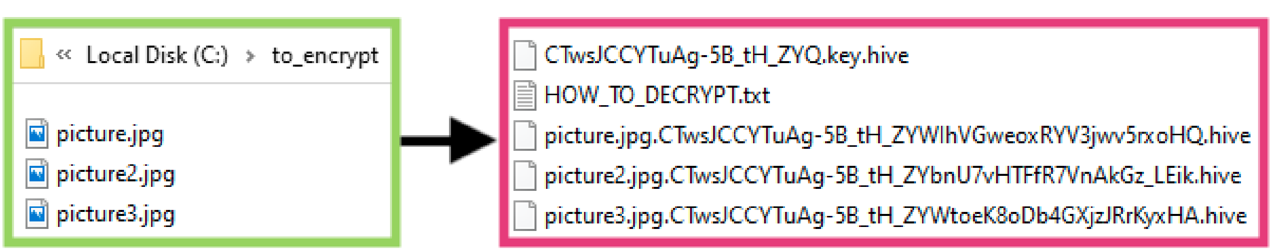


Figure 26. Files encrypted by Hive ransomware.

Following the file encryption, we have another two functions executed by **App.Run()**.

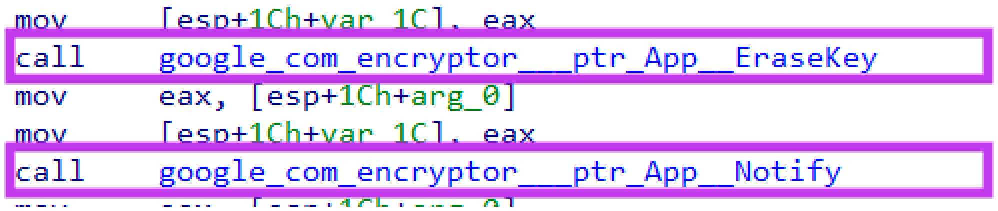


Figure 27. “EraseKey” and “Notify” functions.

The function **App.EraseKey()** accesses the memory location where the 10 MB primary key was stored by Hive and replaces all its bytes with random data.

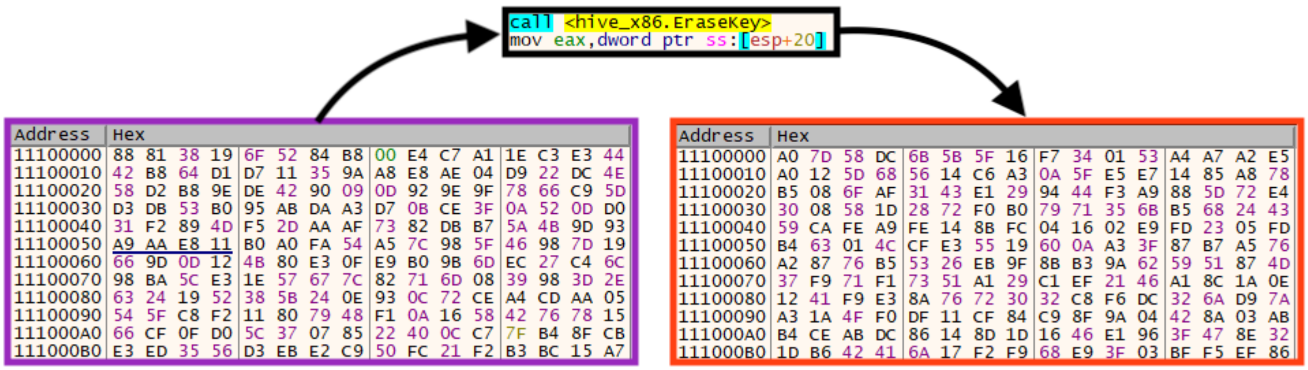


Figure 28. Before and after the “EraseKey” function

App.Notify() creates the ransom note in disk, which is redundant since this file is also created by the function **App.ScanFiles()**.

Last but not least, we have a curious function executed by the ransomware if the flag **“-no-clean”** wasn’t specified, named **App.CleanSpace()**.

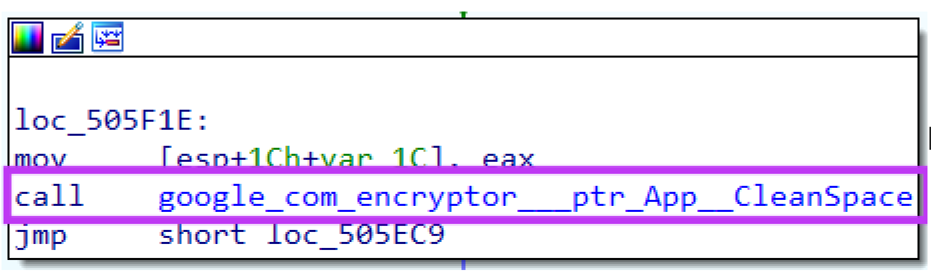


Figure 29. “CleanSpace” function.

Simply put, if executed, this code creates several files with 1GB+ each until the disk is full. Then, these newly created files are deleted.

temp1.swap.hive	9/3/2021 3:10 PM	HIVE File	1,126,400 KB
temp2.swap.hive	9/3/2021 3:10 PM	HIVE File	1,126,400 KB
temp3.swap.hive	9/3/2021 3:10 PM	HIVE File	1,126,400 KB
temp4.swap.hive	9/3/2021 3:10 PM	HIVE File	1,126,400 KB
temp5.swap.hive	9/3/2021 3:10 PM	HIVE File	1,126,400 KB
temp6.swap.hive	9/3/2021 3:11 PM	HIVE File	1,126,400 KB
temp7.swap.hive	9/3/2021 3:11 PM	HIVE File	1,126,400 KB
temp8.swap.hive	9/3/2021 3:11 PM	HIVE File	1,126,400 KB
temp9.swap.hive	9/3/2021 3:11 PM	HIVE File	1,126,400 KB

Figure 30. Files created by the “CleanSpace” function.

Since Hive deletes files that have been encrypted, this process is likely performed to overwrite any bytes on disk that could potentially be restored to their original state, creating new files to replace deleted ones.

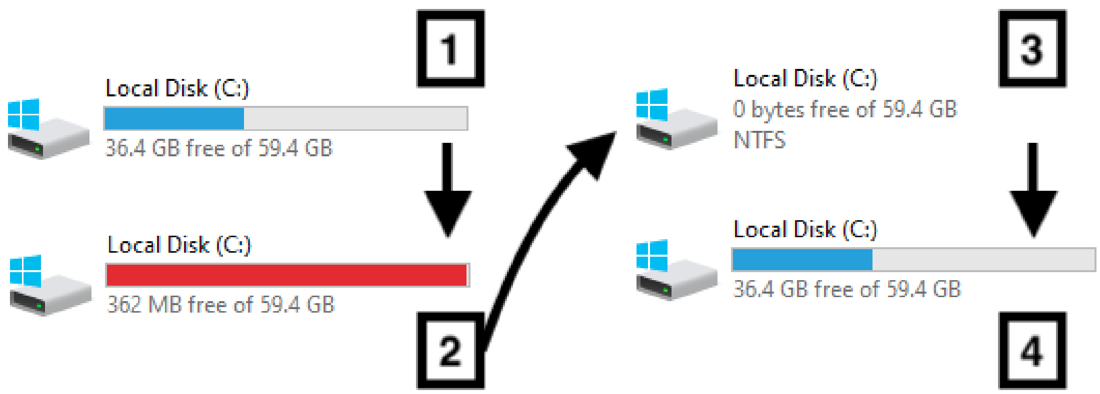
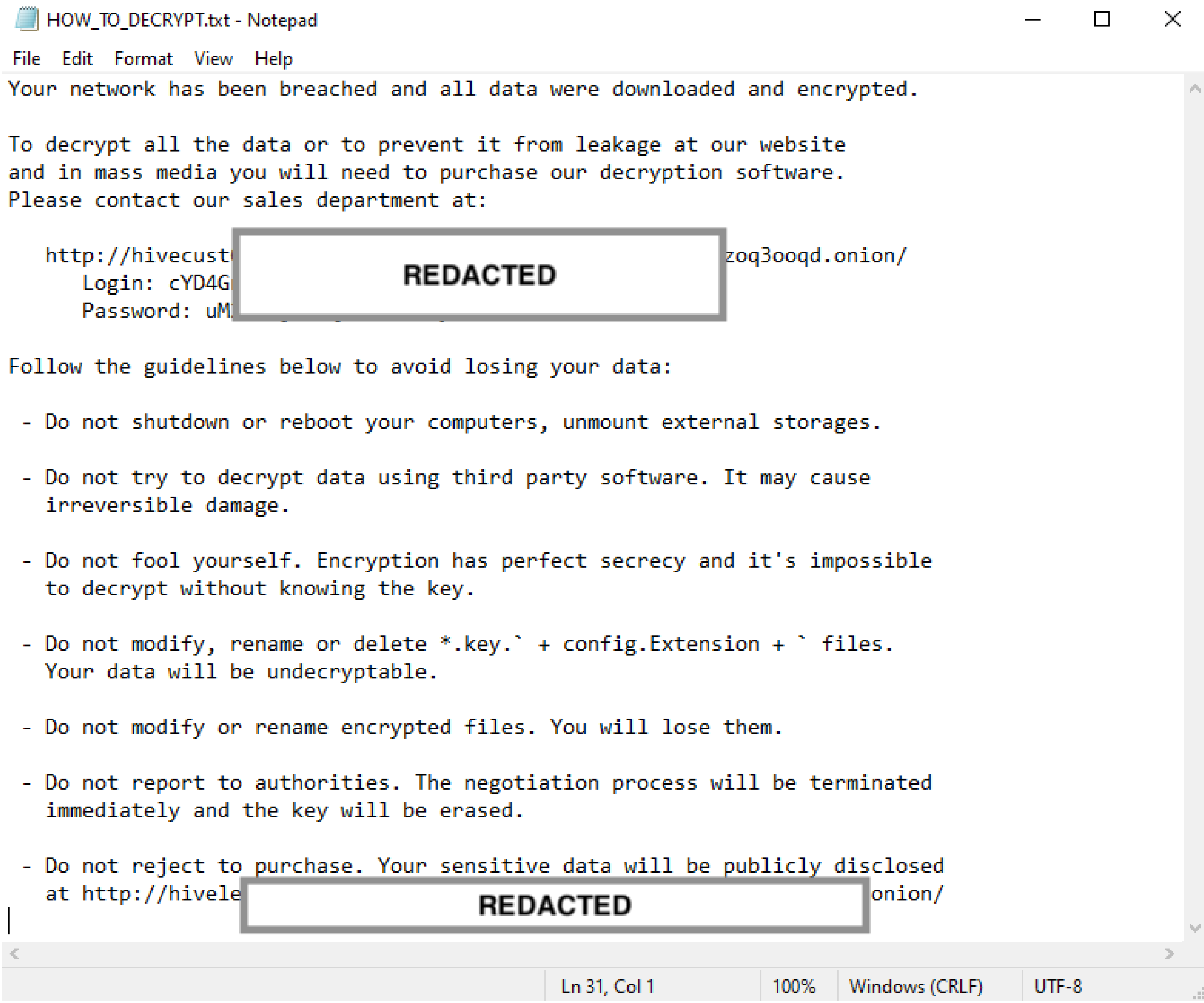


Figure 31. Disk space while the “CleanSpace” function is being executed.

Different from other ransomware families, Hive doesn’t change the user background, the only message available to the victim is the ransom note.



HOW_TO_DECRYPT.txt - Notepad

File Edit Format View Help

```
Your network has been breached and all data were downloaded and encrypted.

To decrypt all the data or to prevent it from leakage at our website
and in mass media you will need to purchase our decryption software.
Please contact our sales department at:

   http://hivecust REDACTED zoq3ooqd.onion/
      Login: cYD4G
      Password: uM

Follow the guidelines below to avoid losing your data:

 - Do not shutdown or reboot your computers, unmount external storages.

 - Do not try to decrypt data using third party software. It may cause
   irreversible damage.

 - Do not fool yourself. Encryption has perfect secrecy and it's impossible
   to decrypt without knowing the key.

 - Do not modify, rename or delete *.key.` + config.Extension + ` files.
   Your data will be undecryptable.

 - Do not modify or rename encrypted files. You will lose them.

 - Do not report to authorities. The negotiation process will be terminated
   immediately and the key will be erased.

 - Do not reject to purchase. Your sensitive data will be publicly disclosed
   at http://hivele REDACTED onion/
```

Ln 31, Col 1 | 100% | Windows (CRLF) | UTF-8

Figure 32. Hive ransom note.

According to the note, if the user deletes the file that has the ".key" extension, the data will be undecryptable, which leads us to the next part of this blog.

Decryptor

Hive provides decryptors for ESXi, Linux, and Windows (32 / 64-bit).

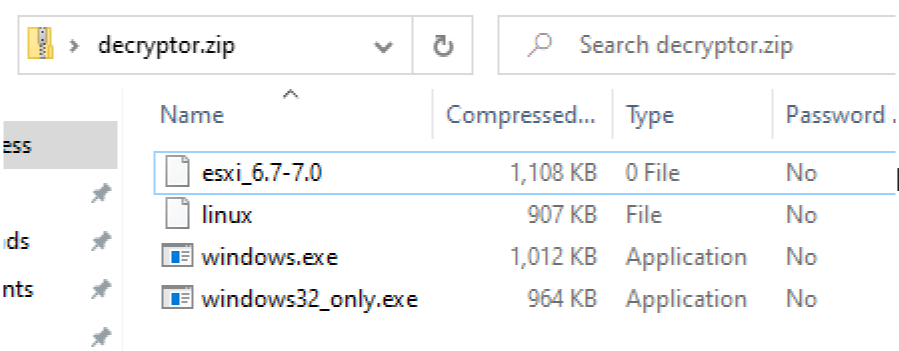


Figure 33. Hive ransomware decryptors for MHS.

Although we only found Windows versions of Hive in the wild, this is a strong indication that they have payloads for other systems, aligning with the fact that the whole code was built in Go language, which is multi-platform.

When it comes to the decryption process, the file first loads the encrypted key from disk, which is why the ransom note states that you can't delete this file.

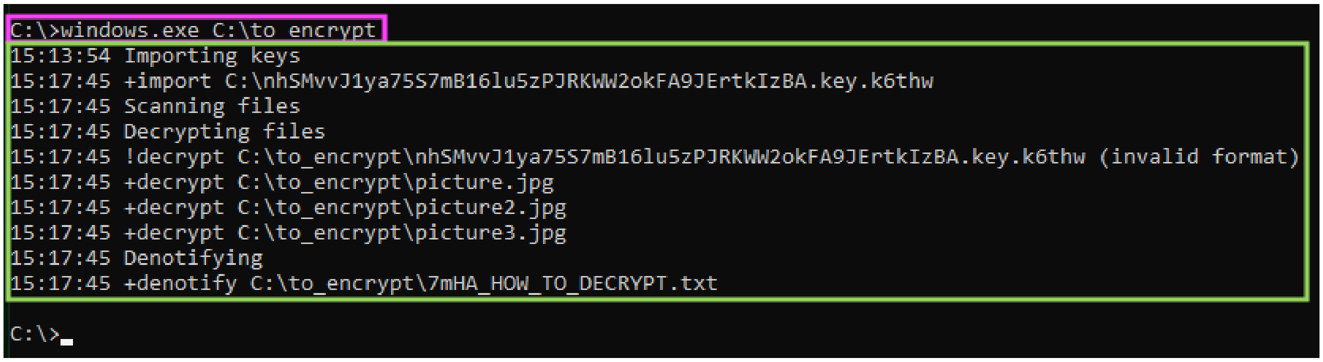


Figure 34. Hive decryption process

Once the key is loaded and decrypted, Hive scans all directories searching for encrypted files, and then proceeds with the decryption process.

Conclusion

Hive is yet another ransomware group that is likely operating in the RaaS model. However, the process used to encrypt the files is quite unusual.

Usually, the encryption process implemented by ransomware in the wild is to generate a unique symmetric key for each file, that is eventually encrypted and stored along with the encrypted data, so it can be recovered later. Instead, Hive creates a unique key that is eventually encrypted and written into disk, making the decryption process irreversible if this file is deleted by accident. Furthermore, this ransomware contains functionalities that make the execution slow, such as "wiping" the disk until it's full to avoid file restoration.

Regardless of these points, we consider Hive a dangerous threat, as it's already causing damage to people and organizations, combined with the fact that the threat is multi-platform.

Protection

Netskope Threat Labs is actively monitoring this campaign and has ensured coverage for all known threat indicators and payloads.

- **Netskope Threat Protection**
  - Gen:Variant.Ransom.Hive.2
  - Trojan.GenericKD.37237769

- **Netskope Advanced Threat Protection** provides proactive coverage against this threat.
  - `Gen.Malware.Detect.By.StHeur` indicates a sample that was detected using static analysis
  - `Gen.Malware.Detect.By.Sandbox` indicates a sample that was detected by our cloud sandbox

IOCs

SHA256

hive_x86	1e21c8e27a97de1796ca47a9613477cf7aec335a783469c5ca3a09d4f07db0ff
hive_x64	321d0c4f1bbb44c53cd02186107a18b7a44c840a9a5f0a78bdac06868136b72c

A full list of IOCs is available in our Git repo.

About the author

Gustavo Palazolo is an expert in malware analysis, reverse engineering and security research, working many years in projects related to electronic fraud protection. He is currently working on the Netskope Research Team, discovering and analyzing new malware threats.

Gustavo Palazolo is an expert in malware analysis, reverse engineering and security research, working many years in projects related to electronic fraud protection. He is currently working on the Netskope Research Team, discovering and analyzing new malware threats.

Read Gustavo Palazolo's full Bio >
More Articles by Gustavo Palazolo >
Read full Bio >
More articles >