

GitHub - reecdeep/HiveV5_file_decryptor: Hive v5 file decryption algorithm

By reecdeep

Archived: 2026-04-10 02:34:43 UTC

HiveV5 file decryptor PoC

Introduction

The work done in the last few months has been necessary to reveal the malicious file encryption mechanism of Hive v5-5.2. The work was divided into two parts

1. [Keystream decryption](#)
2. File decryption using the decrypted keystream

I would like to thank the great [@rivitna](#) for the support, dialogue and advices of these months of work! Please take note of [rivitna's github](#) full of useful informations about Hive ransomware and more.

In this readme you will find some information about the file decryption algorithm, referring you to the PoC for a more complete picture of how it works. A keystream is an encrypted cleartext. A cleartext is a set of 0xA00000 bytes to which the first 0x2FFF00 bytes have been appended, for a total of 0xCFFF00 bytes. These bytes were created with the weak algorithm already discussed in the first part released in July 2022. Here below is a example of cleartext:

```

00000000 BC E8 4C B0 14 78 DC 40 A4 08 6C D0 98 FC 60 C4 4èL°.xÛ@#.1Ð4~ü`Ä
00000010 28 8C F0 54 B8 1C 80 E4 48 10 74 D8 3C A0 04 68 (ËÏT, .ëäH-.tØ< .h
00000020 CC 30 94 F8 5C 24 88 EC 50 B4 18 7C E0 44 A8 0C Ì0"ø\Ï^iP'.|àD".
00000030 70 D4 9C 00 64 C8 2C 90 F4 58 BC 20 84 E8 4C 14 pÔø.dÈ, .ôX4 „èL.
00000040 78 DC 40 A4 08 6C D0 34 98 FC 60 28 8C F0 54 B8 xÛ@#.1Ð4~ü` (ËÏT,
00000050 1C 80 E4 48 AC 10 74 D8 A0 04 68 CC 30 94 F8 5C .ëäH-.tØ .hÌ0"ø\
00000060 C0 24 88 50 B4 18 7C E0 44 A8 0C 70 D4 38 9C 00 ÀÏ^P'.|àD".pÔøø.
00000070 C8 2C 90 F4 58 BC 20 84 E8 4C B0 14 78 40 A4 08 È, .ôX4 „èL°.x@#.
00000080 6C D0 34 98 FC 60 C4 28 8C 54 B8 1C 80 E4 48 AC 1Ð4~ü`Ä(ËT, .ëäH-
00000090 10 74 D8 3C A0 04 CC 30 94 F8 5C C0 24 88 EC 50 .tØ< .Ì0"ø\ÀÏ^iP
000000A0 B4 7C E0 44 A8 0C 70 D4 38 9C 00 64 C8 90 F4 58 `|àD".pÔøø.dÈ.ôX
    ●●●                ●●●
00CFFE30 28 8C F0 54 B8 1C 80 E4 48 AC 10 74 D8 3C A0 04 (ËÏT, .ëäH-.tØ< .
00CFFE40 68 CC 30 94 F8 C0 24 88 EC 50 B4 18 7C E0 44 A8 hÌ0"ø\ÀÏ^iP'.|àD"
00CFFE50 0C 70 D4 38 9C 00 64 C8 2C 90 F4 58 BC 20 84 E8 .pÔøø.dÈ, .ôX4 „è
00CFFE60 4C B0 14 78 DC 40 A4 08 6C D0 34 98 60 C4 28 8C L°.xÛ@#.1Ð4~ü`Ä(Ë
00CFFE70 F0 54 B8 1C 80 E4 48 AC 10 74 D8 3C A0 04 68 CC ËT, .ëäH-.tØ< .hÌ
00CFFE80 30 94 F8 5C C0 24 88 EC 50 B4 18 7C E0 44 A8 0C 0"ø\ÀÏ^iP'.|àD".
00CFFE90 70 D4 9C 00 64 C8 2C 90 F4 58 BC 20 84 E8 4C B0 pÔøø.dÈ, .ôX4 „èL°
00CFFEA0 14 78 DC 40 A4 08 6C D0 34 98 FC 60 C4 28 8C F0 .xÛ@#.1Ð4~ü`Ä(Ë
00CFFEB0 54 B8 1C 80 E4 48 AC 10 74 3C A0 04 68 CC 30 94 T, .ëäH-.t< .hÌ0"
00CFFEC0 F8 5C C0 24 88 EC 50 B4 18 7C E0 44 A8 0C 70 D4 ø\ÀÏ^iP'.|àD".pÔ
00CFFED0 38 9C 00 64 C8 2C 90 F4 58 BC 20 84 E8 4C B0 14 øø.dÈ, .ôX4 „èL°.
00CFFEE0 78 40 A4 08 6C D0 34 98 FC 60 C4 28 8C F0 54 B8 x@#.1Ð4~ü`Ä(ËÏT,
00CFFEF0 1C 80 E4 48 AC 10 74 D8 3C A0 04 68 CC 30 94 F8 .ëäH-.tØ< .hÌ0"ø

```

The Hive sample analyzed and referred to in this document was chosen from [this list](#) created by [@rivitna](#) to which my warmest thanks go. To get an idea of the complexity of ransomware, please take a look at [this analysis](#) published by Microsoft Threat Intelligence Center (MSTIC).

File encryption algorithm

The cleartext (a decrypted keystream) is used by Hive ransomware when encrypting each file. When encrypting a file, Hive ransomware calculates two integers referring to precise positions in the cleartext (offsets) to be used to encrypt the file according to the following formula:

$$file_{encrypted}[i] = cleartext[offset_1 + c] \oplus cleartext[offset_2 + d] \oplus file_{clear}[i]$$

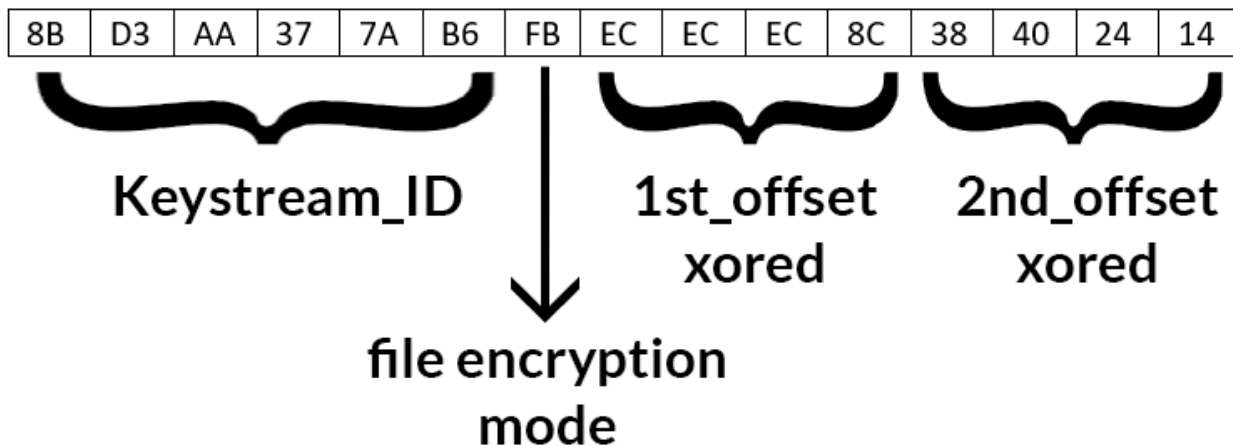
where $c = i \% 0x2FFF00$ e $d = i \% 0x2FFD00$, with i as a byte counter.

The encrypted file extension

The preliminary operations before writing a file are:

- Renaming the file using MoveFileExW and changing its extension;
- Writing the renamed file with the result of the xor operation shown above.

i9OqN3q2--zs7lw4QCQU



Also in this case the cleartext plays a fundamental role. In fact it is used for:

1. Determine the keystream ID (first 6 bytes) using a hash function
2. Encrypt the positions (offsets) used to extract bytes from the cleartext However, the first offset is encrypted using a fixed position of the cleartext and is different for each Hive 5/5.1/5.2 sample. A kind of magical value. In many Hive 5/5.1 artifacts this magic value is shown explicitly inside a memory reference, like in this case 0x98072A :

```

00423E03  80CB FB          or bl,FB
00423E06  8A8C16 2A079800  mov cl,byte ptr ds:[esi+edx+98072A]
00423E0D  304C14 2C          xor byte ptr ss:[esp+edx+2C],cl
00423E11  42             inc edx
00423E12  ^ 75 F2         jne x.423E06
00423E14  8B4C24 28          mov ecx,dword ptr ss:[esp+28]
    
```

Or this case 0x7539D:

```

00581A00  884C24 0C          mov byte ptr ss:[esp+cl],cl
00581A04  8A8C17 9D530700  mov cl,byte ptr ds:[edi+edx+7539D]
00581A0B  304C14 44          xor byte ptr ss:[esp+edx+44],cl
00581A0F  42             inc edx
00581A10  ^ 75 F2         jne a0h2u1h3d2.581A04
    
```

But in the next evidence the for loop is slightly different and has been written in such a way as not to explicit the magic value that we need to identify. This concerns an artifact belonging to Hive 5.2:

```

00AA2B65  C0E3 02          shi bl,2
00AA2B68  80CB FB          or bl,FB
00AA2B6B  8A3C02          mov bh,byte ptr ds:[edx+eax]
00AA2B6E  40             inc eax
00AA2B6F  30B0C0C 04010000  xor byte ptr ss:[esp+ecx+104],bh
00AA2B76  41             inc ecx
00AA2B77  ^ 75 F2         jne q45.AA2B6B
00AA2B79  8B4424 18          mov eax,dword ptr ss:[esp+18]
00AA2B7D  8B9424 00010000  mov edx,dword ptr ss:[esp+100]
00AA2B84  8D7C24 28          lea edi,dword ptr ss:[esp+28]
00AA2B88  898424 00010000  mov dword ptr ss:[esp+100],eax
00AA2B8F  8D05 4F8340AA  lea eax,dword ptr ds:[AA40834F]
00AA2B95  894424 28          mov dword ptr ss:[esp+28],eax
00AA2B99  8B4424 28          mov eax,dword ptr ss:[esp+28]
00AA2B9D  8B88 69FC7256  mov ecx,dword ptr ds:[eax+5672FC69]
00AA2BA3  83C1 FC          and ecx,FFFFFFFF
    
```

In this case it is possible to use the offset bruteforce function present in the released tool, using a file with a known extension and the relative decrypted keystream. Using the header of the encrypted file and the header of the unencrypted file it is possible to understand what is the offset from which the decryptor must start to decrypt the file.

The file encryption mode can have two values: 0xFB or 0xFF

- 0xFB means that the ransomware encrypted the entire file without leaving any portion of the file unencrypted.
- 0xFF means that the ransomware calculated a NCB (not encrypted block) for each file and encrypting blocks of 0x100000 bytes. For further information regarding the calculation of the size of the unencrypted blocks and the cleartext offset, please refer to the PoC code.

Usage

The program offers two options:

```
Hive ransomware V5 - file decryptor PoC
-----
1. Decrypt a file using decrypted keystream
2. Offset bruteforce
your move:
```

1. Decryption of files using the decrypted keystream. You need to enter the special offset present in the sample that encrypted the files.
2. Given a file with a known header (PDF, JPG, PNG, Office files) brute the possible value of the special offset by decrypting the first bytes and looking for a match with the known signature

References

https://github.com/rivitna/Malware/blob/main/Hive/Hive_samples.txt

Source: https://github.com/reecdeep/HiveV5_file_decryptor