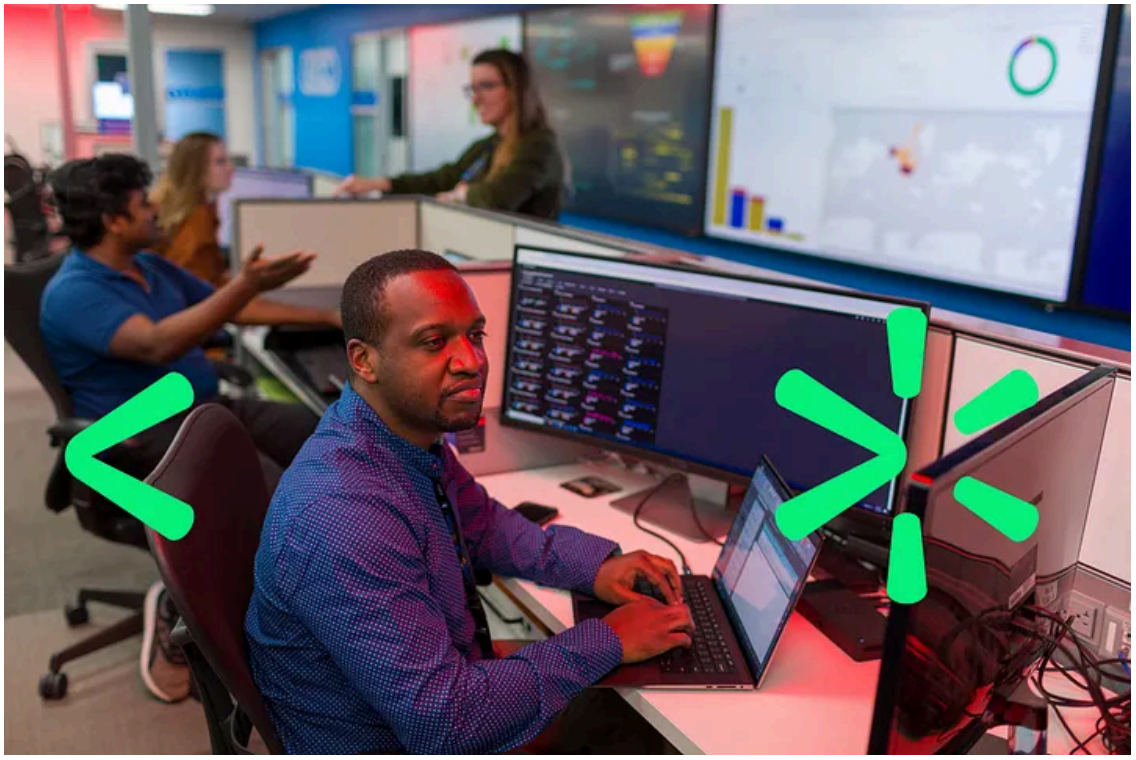
Diavol resurfaces

By Jason Reaves

Published: 2022-09-30 · Archived: 2026-04-05 15:33:28 UTC

By: Jason Reaves and Jonathan McCay

Press enter or click to view image in full size



We previously walked through the Diavol ransomware variants file encryption[1] which has been linked to the TrickBot group[2]. After the recent breakup[3,4], Diavol all but seemed to have disappeared. Curiously, we began to notice an uptick in samples submitted to VirusTotal. While investigating the more recent samples, we were able to determine that it uses a mix of RSA encryption and XOR encoding for files. In some instances, file recovery is still possible.

The following samples were identified on VirusTotal:

```
SHA256: aac969e36686f8f8517c111d30f8fb3b527988ebd31b3b762aec8d46e860eb9d
Creation Time 2022-09-05 20:01:56 UTC
First Submission 2022-09-09 21:06:06 UTC
Last Submission 2022-09-13 15:50:00 UTC
```

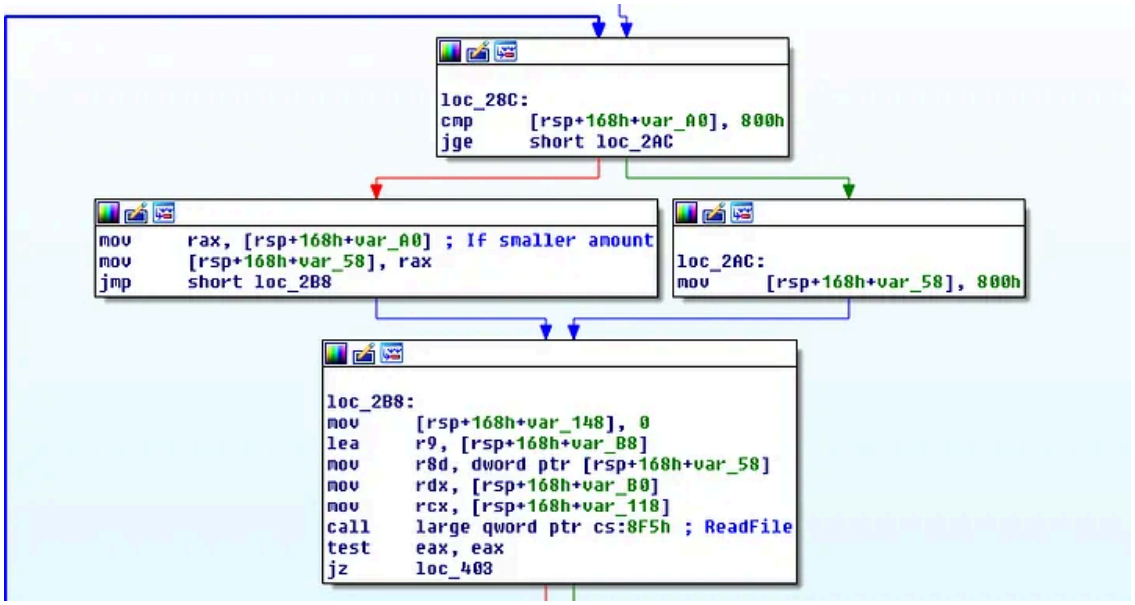
```
Last Analysis 2022-09-13 15:50:00 UTC

SHA256: fb5ee29b98446d34520bf04a82996eefec3b5692710c5631458da63ef7e44fe4
Creation Time 2022-09-05 20:04:30 UTC
First Submission 2022-09-11 20:30:20 UTC
Last Submission 2022-09-11 20:30:20 UTC
Last Analysis 2022-09-11 20:30:20 UTC

SHA256: 708806f5e2e8bfa3d1e911e391ff2ccf1edcac05cc1df80439b8b867253423df
Creation Time 2022-08-25 16:12:58 UTC
First Submission 2022-08-29 19:49:08 UTC
Last Submission 2022-09-03 15:40:44 UTC
Last Analysis 2022-09-03 15:40:44 UTC
```

The samples are now 64 bit but function similarly. For the purposes of this report we will be going through the 7088 sample above. For the purposes of this report, we will be going through the 7088 sample above.

```
group=test
file_ext=.bully
note_filename=WARNING.txt
```

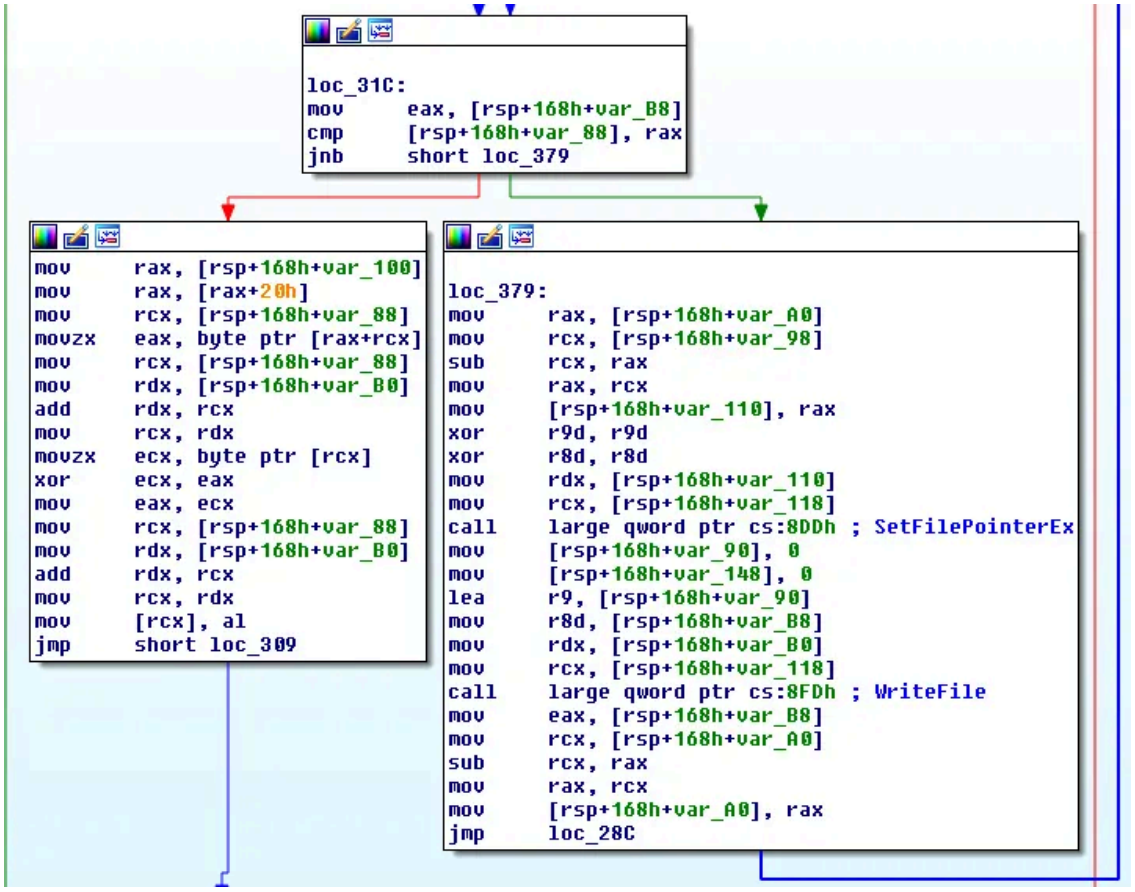
File encryption still involves the use of a 2048 byte XOR key which is randomly generated in the GENBOTID piece of the main bot. The key is then stored in the main bot and reused later in the file encryption code. Then a loop will sit reading chunks of 2048 bytes unless the amount of data to be encoded is less than 2048:

The first part of the file encryption is the aforementioned usage of the 2048 byte XOR key. For most files, the amount of bytes that will be XOR encoded is based on the overall file size divided by 10. Then a loop will sit reading chunks of 2048 bytes unless the amount of data to be encoded is less than 2048:



A similar XOR loop has been implemented, which can be seen in the previous version of Diavol[1]. The loop will handle XOR encoding the chunk of data that was read before writing it back to the file:

Press enter or click to view image in full size

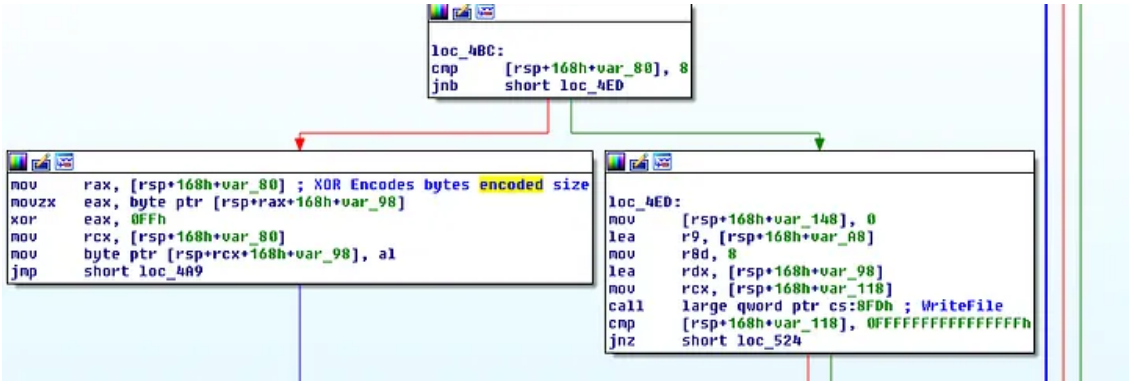


After XOR encoding the file, the RSA encrypted XOR key is written to the end of the file followed by the number of encoded bytes:

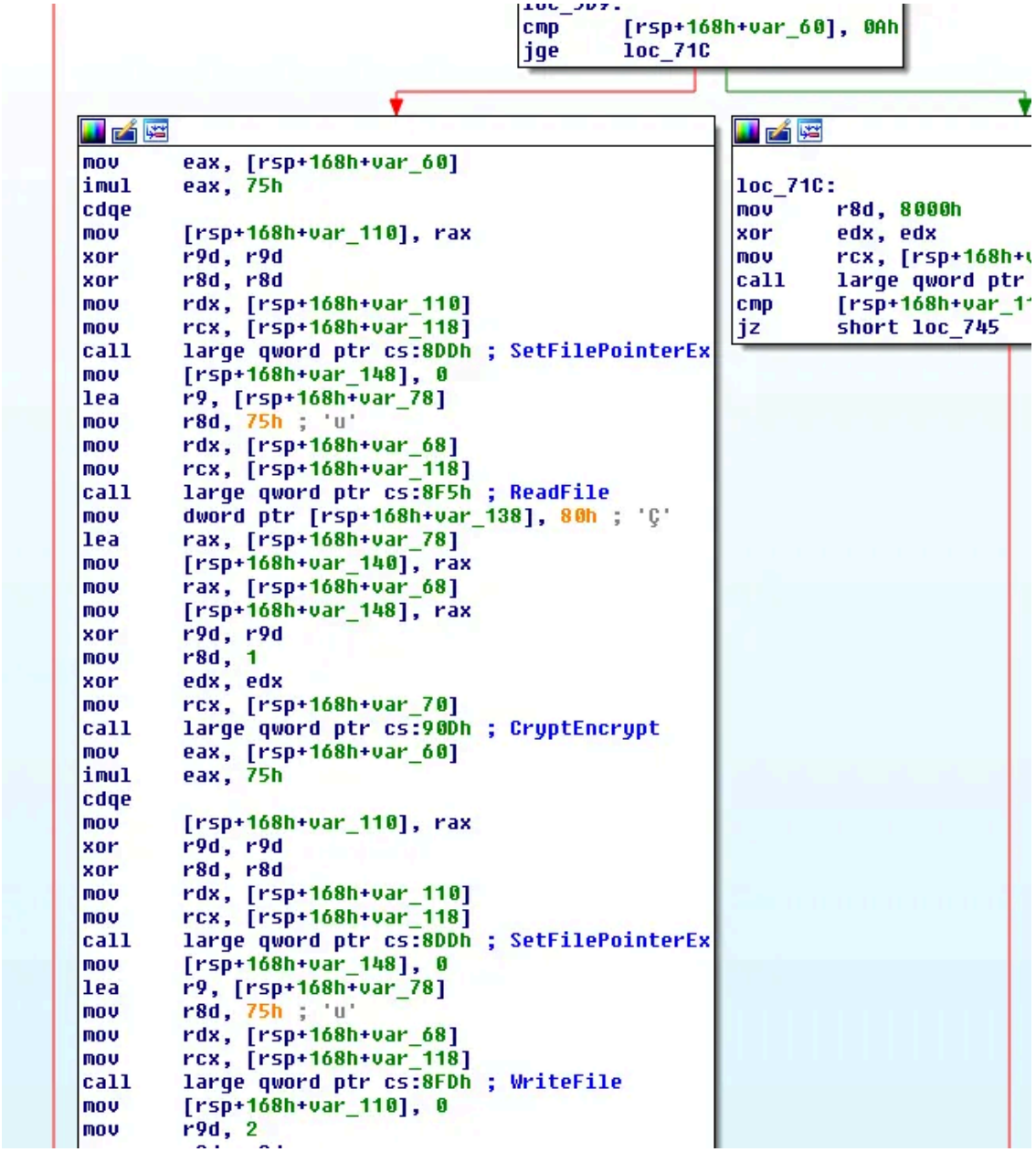
```
mov     [rsp+168h+var_148], 0
lea     r9, [rsp+168h+var_A8]
mov     r8d, 900h
mov     rax, [rsp+168h+var_100]
mov     rdx, [rax+28h]  ; Encrypted XOR key
mov     rcx, [rsp+168h+var_118]
call    large qword ptr cs:8FDh ; WriteFile
mov     [rsp+168h+var_148], 0
lea     r9, [rsp+168h+var_A8]
mov     r8d, 8
lea     rdx, [rsp+168h+var_98] ; encoded bytes
mov     rcx, [rsp+168h+var_118]
call    large qword ptr cs:8FDh ; WriteFile
mov     [rsp+168h+var_80], 0
jmp     short loc_4BC
```

Next the bot single XOR encodes the number of encoded bytes and writes that to the end of the file:

Press enter or click to view image in full size



After XOR encoding and writing the appropriate data to the end of the file, the bot goes back to the beginning of the file and begins reading in chunks of 0x75 bytes. It will RSA encrypt them and the encrypted bytes are then written back to the file but without the padding bytes. In this way, 0x75 * 10 or 1170 bytes at the beginning of the file will be RSA encrypted after getting XOR encoded.



A quick test can be performed to validate our findings, using a file of NULLs and a large MSI file. First, we validate the end data that was added to the file, which should be 110+0x900+16 bytes from the end:

```
>>> data = open('test_data.txt.bully', 'rb').read()
>>> 110+0x900+16
2430
>>> end = data[-2430:]
```

Skipping over the RSA encrypted XOR key should show the two 8 byte values with the second being XOR encoded with 0xFF

```
>>> end[0x900:]
'\x88\x13\x00\x00\x00\x00\x00w\xec\xff\xff\xff\xff\xff\xffk\xa8\x0f/6o\x12\x08\xd6\xbe\xaaw\xf1\
>>> 0x1388
```

```
5000
>>> end[0x900+8:]
'w\xec\xff\xff\xff\xff\xff\xffk\xa8\x0f/6o\x12\x08\xd6\xbe\xaaw\xf1\x1b0\x1f\x10\x12\x9b\x12\xcc?\xf
>>> l = bytearray('w\xec')
>>> l[0] ^= 0xff
>>> l[1] ^= 0xff
>>> l
bytearray(b'\x88\x13')
```

Since the file is NULLs, the clear XOR key should be the first 2048 bytes after we skip over the 1170 RSA encrypted bytes at the beginning:

```
>>> key = bytearray(data[1170:1170+2048])
```

We can test this against another file, in this case an MSI:

```
>>> data2 = open('powerpointmui.msi.bully', 'rb').read()
>>> test_block = bytearray(data2[1170:])
>>>
>>> for i in range(len(test_block)):
...   test_block[i] ^= key[i%2048]
...
>>> test_block[:10000]
bytearray(b'\xa4A(H\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\
snip
....
02\x00\x00\x12\x00\x00\x000\x02\x00\x00\x02\x00\x00\x00\xe4\x04\x00\x00\x1e\x00\x00\x00\x16\x00\x00\
...snip...
```

So it is possible to recover most of each file trivially, after you recover the XOR key. The next step is to just rebuild the first 1170 bytes.

IOCs

Endpoint:

```
WARNING.txt
warning.txt
.bully4eb5bea255c0308b296f5aa259f68626
88b41ba2d6b7cca40118de9007cf64a0e5cc9710
aac969e36686f8f8517c111d30f8fb3b527988ebd31b3b762aec8d46e860eb9dcba851aab28c4b52fb9f0c655d2c0c0e
9697acfa83a31c2925b72f627c9be51346cf5dd0
fb5ee29b98446d34520bf04a82996eefec3b5692710c5631458da63ef7e44fe4332c1a9146276bc9abc1161e13efabde
```

```
6c366cd3b4a54f8e9f7ed6016aac9e7509b06102
708806f5e2e8bfa3d1e911e391ff2ccf1edcac05cc1df80439b8b867253423df
```

Ransom Note:

```
You've been hacked. All your corporate network servers and workstations are encrypted.Your company i
```

Network:

```
hxxps://7ypnbv3snejqmgce4kbewwvym4cm5j6lkzf2hra2hyhtsvwjaxwipkyd[.]onion
173.232[.]146[.]118
```

References

1: https://medium.com/walmartglobaltech/diavol-the-enigma-of-ransomware-1fd78ffda648

2: https://www.bleepingcomputer.com/news/security/fbi-links-diavol-ransomware-to-the-trickbot-cybercrime-group/

3: https://www.advintel.io/post/the-trickbot-saga-s-finale-has-aired-but-a-spinoff-is-already-in-the-works

4: https://www.bleepingcomputer.com/news/security/conti-ransomware-shuts-down-operation-rebrands-into-smaller-units/
