Diavol the Enigma of Ransomware

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

Jason Reaves March 10, 2022

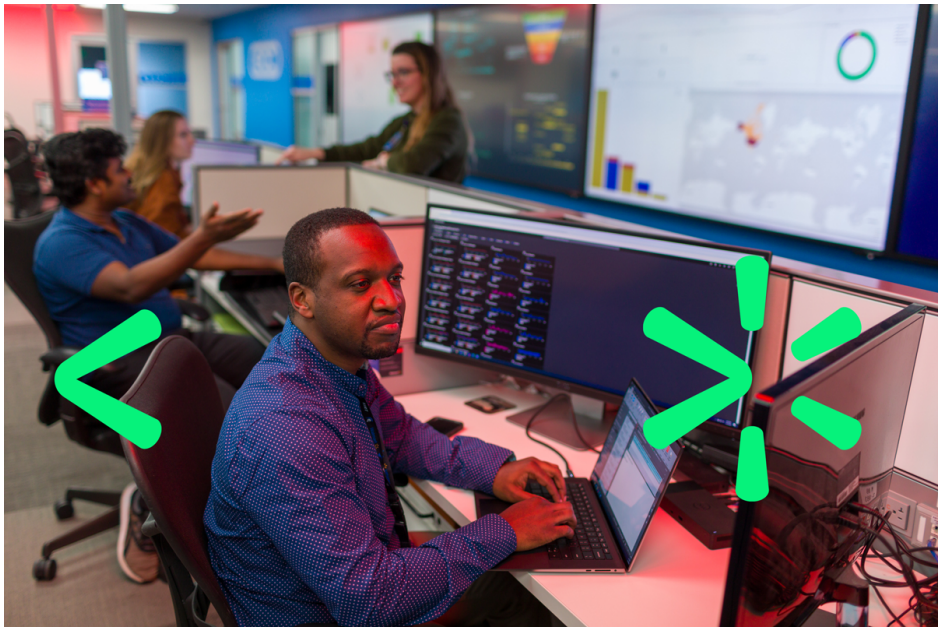
Jason Reaves

Mar 10

.

4 min read

By: Jason Reaves and Joshua Platt



Diavol ransomware was first publicly reported by Fortinet in July 2021 [1]. The posting included a technical analysis of the file that was allegedly dropped from a previous engagement in June 2021. According to the blog, the Diavol variant was found along side a Conti (v3) sample, which had also been spread during the same attack. In a follow-up article by IBM-Xforce, the researchers concluded a stronger link existed between the development of Diavol and the operators behind Trickbot malware.

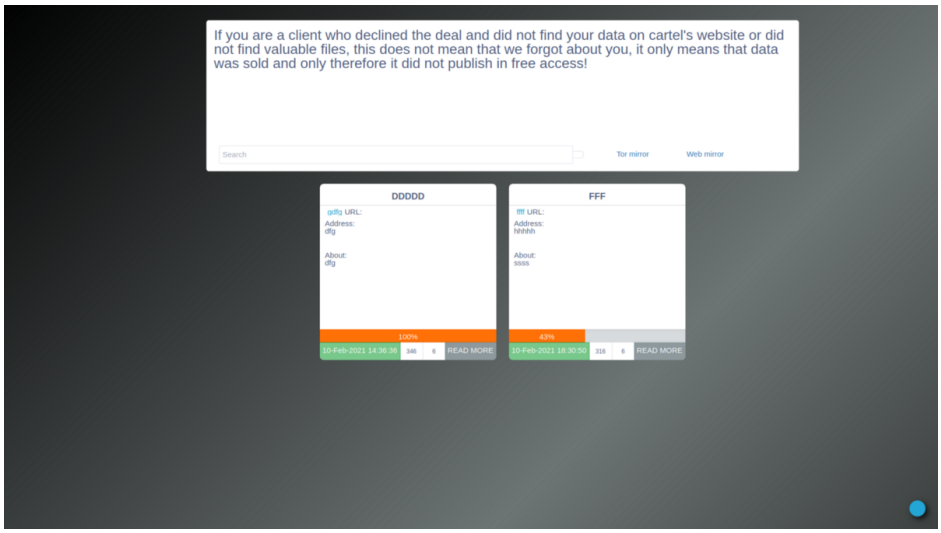
While multiple samples have been found in the wild, they appear to contain development artifacts. It was clear the locker was utilized but there was no mention of a leak site and nothing had been identified publicly. After analyzing the binary, we spotted some interesting infrastructure and began to investigate. The domain name enigma-hq[.]net stood out and was associated with '195.123.221[.]248'. According to passive DNS records an update had occurred and enigma-hq[.]net was changed to diavol-news[.]net:

Passive DNS Replication

Date resolved	Detections	Resolver	Domain
2021-02-27	2 / 90	VirusTotal	diavol-news.net
2021-01-30	0 / 90	VirusTotal	enigma-hq.net

Credit:
The HTML revealed a TOR mirror along with the web mirror pointing to 'diavol-news[.]net':

```
<input type="text" id="blogpostsearch-search" class="search" name="BlogPostSearch[search]" placeholder="Search"><p class="help-
block help-block-error"></p><button type="submit"><i class="fa fa-search"></i></button>    </form></div>
</div><div class="col-md-4 search_fix">                                    <div class="row">                                    <div
class="col-md-4">                                    <a class="pull-left" href="https://xhtnringfhbflc6d.onion" ref="noopener
noreferrer">Tor mirror</a>                                    </div>                                    <div class="col-md-4">
<a class="pull-center" href="https://diavol-news.net" ref="noopener noreferrer">Web mirror</a>
</div>                                                                    </div>                                    </div>
</div>           </div>
```



Diavol Test Leak Site

Technical Overview

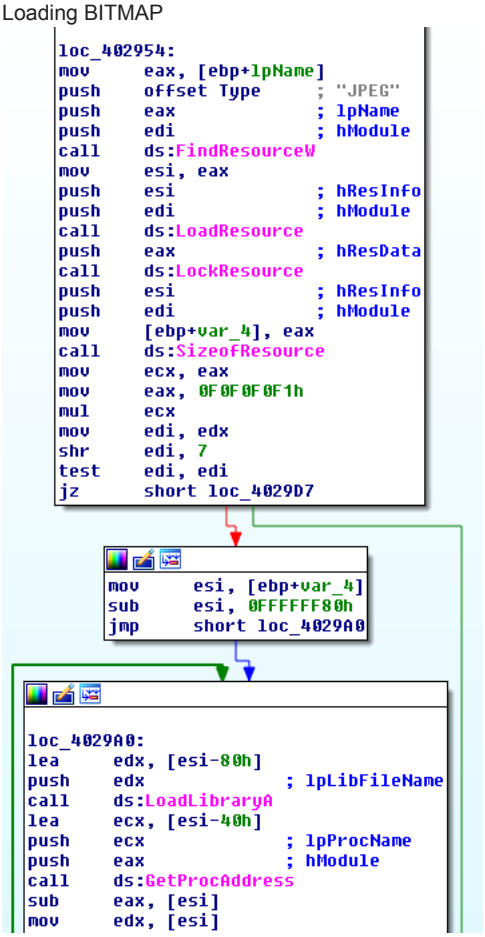
Diavol comes with an interesting assortment of code blocks onboard to accomplish various tasks:

- JPEG ENCDEFILE 0419
- JPEG ENMDSKS 0419
- JPEG FINDFILES 0419
- JPEG FROMNET 0419
- JPEG GENBOTID 0419
- JPEG KILLPR 0419
- JPEG REGISTER 0419
- JPEG RSAINIT 0419
- JPEG SERVPROC 0419
- JPEG SMB 0419
- JPEG SMBFAST 0419
- JPEG VSSMOD 0419
- TEXT TEXT2003_64 0419
- TEXT TEXT64 0419
- BITMAP ENCDEFILE 0419
- BITMAP ENMDSKS 0419
- BITMAP FINDFILES 0419
- BITMAP FROMNET 0419
- BITMAP GENBOTID 0419
- BITMAP KILLPR 0419
- BITMAP REGISTER 0419
- BITMAP RSAINIT 0419
- BITMAP SERVPROC 0419
- BITMAP SMB 0419
- BITMAP SMBFAST 0419
- BITMAP VSSMOD 0419
- MANIFEST 0001 0409

The BITMAP objects contain the code while the JPEG objects contain the imports that need to be resolved.

```
push    edi             ; hInstance
mov     [ebp+lpBits], eax
mov     [ebp+var_3C], ecx
call    ds:LoadBitmapW
push    0               ; hdc
mov     ebx, eax
call    ds:CreateCompatibleDC
mov     esi, eax
push    ebx             ; h
push    esi             ; hdc
call    ds:SelectObject
lea     edx, [ebp+pv]
push    edx             ; pv
push    18h             ; c
push    ebx             ; h
call    ds:GetObjectW
mov     eax, [ebp+lpBits]
push    eax
mov     eax, [ebp+var_3C]
lea     ecx, [ebp+pv]
mov     edx, edi
call    FixImportsFromJPEG_Resource_402980
add     esp, 4
push    0               ; hdc
mov     [ebp+lpBits], eax
call    ds:CreateCompatibleDC
mov     ecx, [ebp+cy]
mov     edx, [ebp+var_54]
push    ecx             ; cy
push    edx             ; cx
```

Loading BITMAP



Fixing imports from JPEG

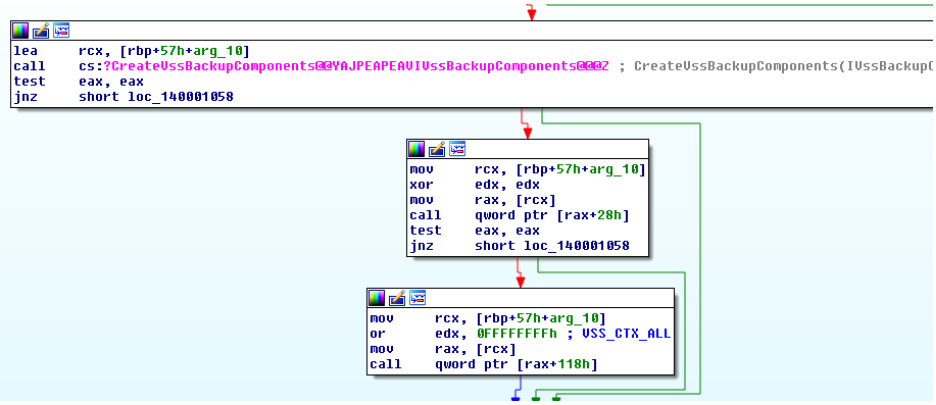
The objects were previously detailed in the Fortinet blog but here is an overview from our own analysis of a recent sample:

Code section	Purpose
ENCDEFILE	Encode a file and store the encoded symmetric key
ENMDSKS	Enumerates all local disks and drives, if it finds share then uses WNetGetConnection to retrieve the name of the network resource
FINDFILES	Given a path and a list of expressions that are blacklisted to check it will enumerate to find all files
FROMNET	Handles making HTTP requests and retrieves the response data
GENBOTID	Generates the bot ID and the random symmetric key
KILLPR	Enumerates running processes with 'CreateToolhelp32Snapshot' and kills processes from a list
REGISTER	Also sends an HTTP request but doesn't retrieve the response data
RSAINIT	Loads the onboard RSA public key and then encrypts the randomly generated file key with the public key
SERVPROC	Stop a service
SMB	Find accessible share folders
SMBFAST	The same as SMB but doesn't enumerate every entry from GetIpNetTable
VSSMOD	Used to load and execute the onboard VSS deleting module from resources

There is two interesting pieces that we discovered from our analysis, one is that because of the way VSSMOD works you can plug and play various ways to wipe shadow copies and the other is the way file encryption works.

Shadow copies

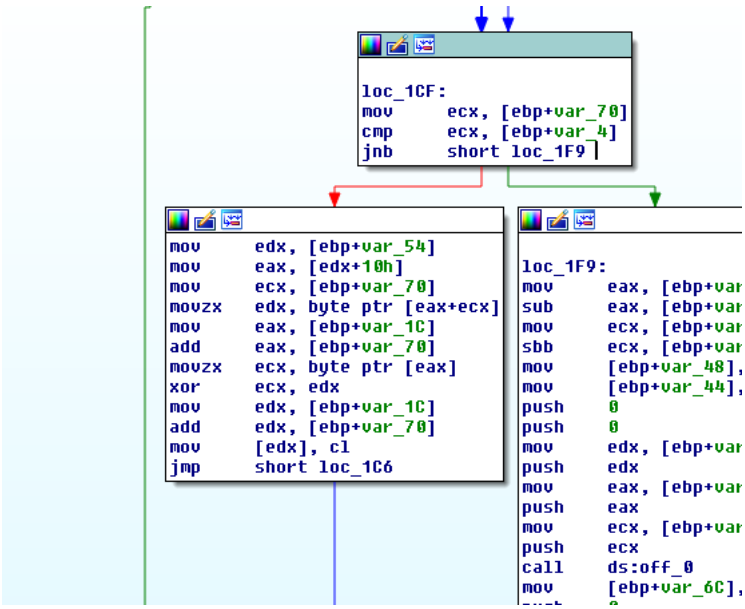
For one of the samples we analyzed the shadow copies were wiped using WinAPI which doesn't appear to be used very often by ransomware:



After calling CreateVssBackupComponents you can use the IVssBackupComponents class [5] which can then be leveraged to delete snapshots.

Encryption

File encryption in Diavol is interesting, it has a routine for decoding the onboard RSA public key and importing it before encrypting the key that will be used to encrypt the files. The file encryption key is 2048 bytes long and is randomly generated however the encryption is simply XORing the files in chunks of 2048:



Since the file encryption key is being used across multiple files and is simply a XOR operation we can abuse known plaintext vulnerabilities to recover files.

> A = ClearText
> B = EncryptedFile1
> C = EncryptedFile2
>
> key = A[:2048] ^ B[:2048]
> DecodedFileChunk = key[:2048] ^ C[:2048]

We can test this using files from a sandbox run [6] along with a random MSI file which has a semi static first chunk of bytes, it won't be a clean decrypt by any means but would prove out our hypothesis:

> >>> clear= open('a4ce1d7dfc5ab1fdee8cd0eb97d19c88a04deb8fe6b7b58413a9e2c93eb4a79d.msi', 'rb').read()
> >>> b = bytearray(open('powerpointmui.msi.lock64', 'rb').read())
> >>> c = open('sharepointdesignermui.msi.lock64', 'rb').read()
> >>> key= bytearray(a[:2048])
> >>> for i in range(len(test)):
> … test[i] ^= b[i]
> …
> >>> temp = bytearray(c)
> >>> for i in range(len(temp)):
> … temp[i] ^= test[i%len(test)]
> >>> temp[:5000]
> bytearray(b'\xd0\xcf\x11\xe0\xa1\xb1\x1a\xe1\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00>\x00\x03\x00\xfe\xff\t\x00\x
> \x00\x00\x00.\x00\x00\x00/\x00\x00\x000\x00\x00\x001\x00\x00\x002\x00\x00\x003\x00\x00\x004\x00\x00\x005\x00\x00\x006\x00\x00\x007\x
> <..snip..>
> %\x00\x00\x00&\x00\x00\x00\'\x00\x00\x00(\x00\x00\x00)\x80\x00\x00*\x00\x00\x00+\x00\x00\x00,\x80\x00\x00-
> \x00\x00\x00.\x00\x01\x00/\x80\x00\x000\x00\x00\x00=\x00\x02\x01\x12\x00\x00\x007\x80\x00\x00\xcb\x7f\xff\xff5\x00\x00\x006\x00\x00\x00
> 5\xc5ID)>\xb2D\xd1\xb7\xff\xff\xfe\xff\xff\xff\xfe\xff\xff\xffG\x00\x00\x00H\x00\x00\x00I\x00\x00\x00J\x00\x00\x00K\x00\x00\x00L\x00\x00\x00M
> \x15\xc5ID\t;\xe4E@H\x00\x00e\x00\x00\x00f\x00\x00\x00g\x00\x00\x00h\x00\x00\x00i\x00\x00\x00j\x00\x00\x00k\x00\x00\x00l\x00\x00\x00n
> \x00\x10\x84\xff\xff\x91\xfb\x00\x00\x12\x05\x00\x00o\x05\xff\xff\x14\xfa\x00\x00\x95\x06\x00\x00\xe9\x06\x00\x00\x97\x07\x00\x00\xe7\x07\
>
> <..snip..>
>
> \xff7\xffh\xff\xb4\xff\xff\xff\xb7\xff\x8b\xff\xb6\xff\xff\xff\xb1\xffh\xff)\xff\xff\xff\xb3\xff\x8b\xff\xc6\xff\x8b\xff\xad\xff\xff\xff\xac\xff\xff\xff\xaf\xff\xff\xff

It appears to have worked and since the file encryption key is generated on a per infection basis we simply only need to abuse this technique to recover 2048 bytes once in order to then recover all the files on the system.

References

1: https://www.fortinet.com/blog/threat-research/diavol-new-ransomware-used-by-wizard-spider

2: https://heimdalsecurity.com/blog/is-diavol-ransomware-connected-to-wizard-spider/

3: https://securityintelligence.com/posts/analysis-of-diavol-ransomware-link-trickbot-gang/

4: https://labs.vipre.com/conti-ransomware-part-2-utilizing-server-message-block-smb-to-share-infection/

5: https://docs.microsoft.com/en-us/windows/win32/api/vsbackup/nl-vsbackup-ivssbackupcomponents

6: https://app.any.run/tasks/27db4430-59e5-48c5-8191-f3491f13b3c4#

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