DearCry ransomware attacks exploit Exchange server vulnerabilities

news.sophos.com/en-us/2021/03/15/dearcry-ransomware-attacks-exploit-exchange-server-vulnerabilities/

Mark Loman March 15, 2021

A recently-patched set of vulnerabilities in on-premises versions of Microsoft Exchange Server has been actively exploited for over two months. The exploit, initially attributed to a Chinese state-sponsored actor, has now been adopted for a range of cybercrime activities—the latest being a ransomware called DearCry. Sophos recently detected and stopped a DearCry attack using the exploit, and obtained samples for analysis.

The DearCry ransomware appears to be created by a beginner—it is unsophisticated, and does little to hide itself from detection. Its most notable feature is that the encryption header that DearCry adds to the attacked files looks similar to the header used by the notorious WannaCry ransomware, which seems more than a coincidence.

The exploit

The bugs leveraged in the exploit—CVE-2021-26855 and CVE-2021-27065—were uncovered and reported to Microsoft in December by researchers at DEVCORE. On New Year's Day, the DEVCORE researchers chained the bugs together and created a workable pre-authentication remote code execution exploit they dubbed "ProxyLogon," as it exploits bugs in Exchange's proxy architecture and logon mechanism .

DEVCORE contacted the Microsoft Security Response Center through its MSRC portal on January 5, 2021, with a 120-day public disclosure deadline. On February 18, MSRC confirmed the bug would be fixed in the March 9 “Patch Tuesday” release. But Microsoft was prompted to make an out-of-band patch release on March 3 after the memory analysis and incident response company Volexity reported in-the-wild exploitation.

According to Volexity, in-the-wild exploitation appears to have started as early as January 6, 2021—the day after the proof-of-concept was submitted to Microsoft. Volexity observed the attacker writing webshells (ASPX files) to disk and conducting further operations to dump credentials, add user accounts, steal copies of the Active Directory database, and move laterally to other systems and environments. Microsoft Threat Intelligence Center (MSTIC) attributed this campaign with high confidence to HAFNIUM, a group assessed to be state-sponsored and operating out of China, based on observed victimology, tactics and procedures.

After investigation, DEVCORE confirmed that the in-the-wild exploit observed by Volexity was the same one DEVCORE submitted to Microsoft. The exploited path in the exploit attacks is similar (/ecp/<single char>.js) and the webshell password is “orange” (hardcoded by DEVCORE exploit developer Orange Tsai @orange_8361).

Sophos first detected and blocked a DearCry attack on a customer’s network in Austria on March 13. A few days earlier, on March 11, the same Exchange server was hit with a webshell, which was also blocked.

DearCry

The anti-ransomware team within SophosLabs evaluated two samples of DearCry for this analysis. In both cases, the binaries were unsigned, and showed no evidence of version control or other professional development practices. The binaries had no defense against anti-virus signatures—they were not packed or obfuscated, so all ransomware text strings are in plain sight for detection by analysts and signature-based malware protection. The absence of these characteristics leads us to believe that the ransomware author is a beginner, or that this is an early prototype.

Both samples contained an identical PDB reference to the machine and source file used to compile the malware binary:

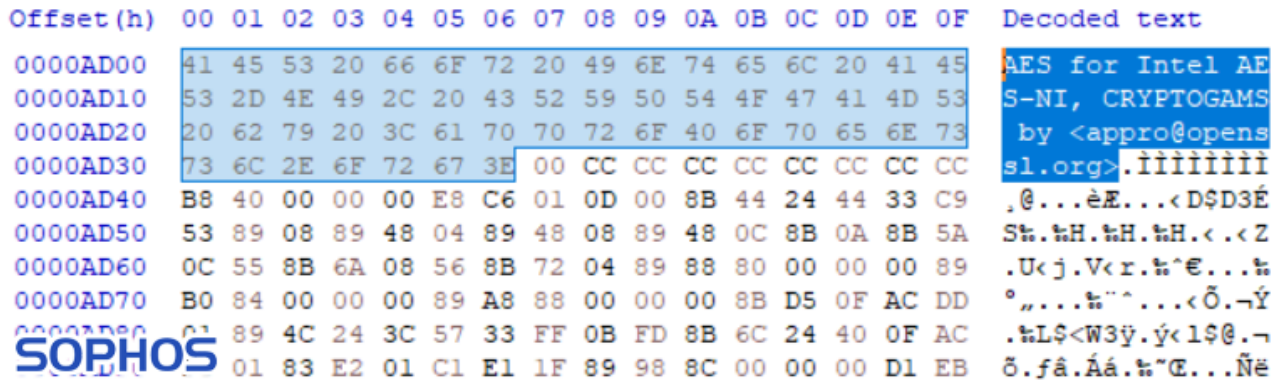
```
C:\Users\john\Documents\Visual Studio 2008\Projects\EncryptFile -
svcV2\Release\EncryptFile.exe.pdb
```

Each binary appeared to be created specially to be delivered to the victim. DearCry uses a fully standalone encryption method, with the public key embedded within the ransomware binary, so it does not have to contact a C2 server in order to begin encrypting files. The two samples we studied were sent to different victims, and had different unique identifiers in their ransom notes, and used different keys.

Crypto combo

Before encrypting a file, DearCry first creates a new file with a filename based on the name of the document it attacks, but adds a .CRYPT file extension. Once created, DearCry starts reading contents of the original file and writes it back, encrypted, into the .CRYPT file.

There are two different encryption methods used by DearCry. Files are encrypted using the AES-256 symmetric encryption algorithm, using an OpenSSL library embedded in the ransomware.



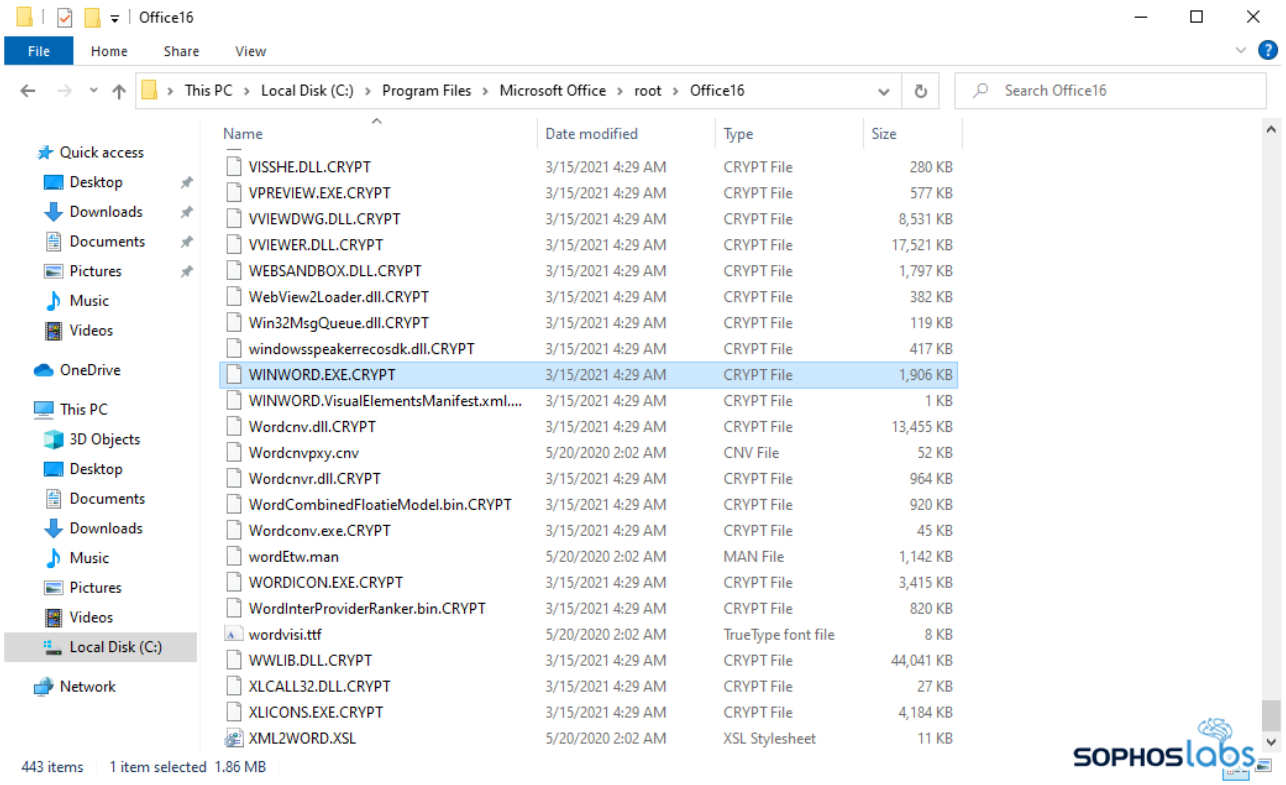
OpenSSL code embedded in the DearCry binary

But the AES key itself is encrypted by the attacker with an RSA public-key algorithm. A public key for decrypting the AES key is embedded in the code, but the private key is retained by the attacker. Using the RSA encryption of the AES key allows the actor to deploy the ransomware without needing a command and control server to deploy the key. And the DearCry actor can create a ransomware binary for each victim, with a unique, victim-specific public encryption key. However, based on our findings, the actor has attempted to deliver the same binary to multiple victims.

Interestingly, the list of file-types the ransomware targets can differ per victim. For example, popular image files (like JPG), CAD drawings, programs (EXE) and dynamic link libraries (DLL) were not targeted in sample 1, which seems earlier distributed according to the compiler timestamp. Most other ransomware families exclude programs and DLLs from their list of encryption targets, because encrypting the wrong files can result in the computer becoming unbootable, which makes it much harder for the victim to read the ransom note.

Sample 1 E044D9F2D0F1260C3F4A543A1E67F33FCAC265BE114A1B135FD575B860D2B8C6	Sample 2 2B9838DA7EDB0DECD32B086E47A31E8F5733B5981AD8247A2F9508E232589BFF
Compiler timestamp: 0x6045C431 Monday, 8 March 2021 07:29:05 GMT+01:00	Compiler timestamp: 0x60472D07 Tuesday, 9 March 2021 09:08:39 GMT+01:00
.TIF .TIFF .PDF .XLS .XLSX .XLTM .PS .PPS .PPT .PPTX .DOC .DOCX .LOG .MSG .RTF .TEX .TXT .CAD .WPS .EML .INI .CSS .HTM .HTML .XHTML .JS .JSP .PHP .KEYCHAIN .PEM .SQL .APK .APP .BAT .CGI .ASPX .CER .CFM .C .CPP .GO .CONFIG.CSV .DAT .ISO .PST .PGD .7Z .RAR .ZIP .ZIPX .TAR .PDB .BIN .DB .MDB .MDF .BAK .LOG .EDB .STM .DBF .ORA	.TIF .TIFF .PDF .XLS .XLSX .XLTM .PS .PPS .PPT .PPTX .DOC .DOCX .LOG .MSG .RTF .TEX .TXT .CAD .WPS .EML .INI .CSS .HTM .HTML .XHTML .JS .JSP .PHP .KEYCHAIN .PEM .SQL .APK .APP .BAT .CGI .ASPX .CER .CFM .C .CPP .GO .CONFIG .PL .PY .DWG .XML .JPG .BMP .PNG .EXE .DLL .CAD .AVI .H.CSV .DAT .ISO .PST .PGD .7Z .RAR .ZIP .ZIPX .TAR .PDB .BIN .DB .MDB .MDF .BAK .LOG .EDB .STM .DBF .ORA .GPG .EDB .MFS *The highlighted items differ from the other sample.
-----BEGIN RSA PUBLIC KEY----- MIIBCAKCAQEA5+mVBe75OvCzCW4oZHl7vqPwV2O4kgzgfp9odcL9LZc8Gy2+NJPD wrHbttKI3z4Yt3G04lX7bEp1RZjxUYfzX8qvaPC2EBduOjSN1WMSbJJrINs1Izkq XRrggJhSbp881Jr6NmpE6pns0Vfv//Hk1idHhxsXg6QKtfXlzAnRbgA1WepSDJq5 H08WGFBZrgUVM0zBYI3JJH3b9jIRMVQMJUQ57w3jZpOnpFXSZoUy1YD7Y3Cu+n/Q 6cEft6t29/FQgacXmeA2ajb7ssSbSntBpTpoyGc/kKoaihYPrHtNRhkMcZQayy5a XTgYtEjhzJAC+esXiTYqklWMXJS1EmUpoQIBAw== -----END RSA PUBLIC KEY-----	-----BEGIN RSA PUBLIC KEY----- MIIBCAKCAQEAyLBClz9hsFGRf9fk3z0zmY2rz2J1qqGfV48DSjPV4lcwnhCi4/5+ C6UsAhk/dI4/5HwbfZBAiMySXNB3DxVB2hOrjDjIeVAkFjQgZ19B+KQFWkSo1ube VdHjwdv74evE/ur9Lv9HM+89iZdzEpVPO+AjOTtsQgFNtmVecC2vmw9m60dgyR/1 CJQSg6Moblo2NVF50AK3cIG2/lVh82ebgedXsbVJpjVMc03aTPWV4sNWjTO3o+aX 6Z+VGVLjuvcpfLDZb3tYppkqZzAHfrCt7lV0qO47FV8sFCltuoNiNGKiP084KI7b 3XEJepbSJB3UW4o4C4zHFrqmdyOoUlnqcQIBAw== -----END RSA PUBLIC KEY-----

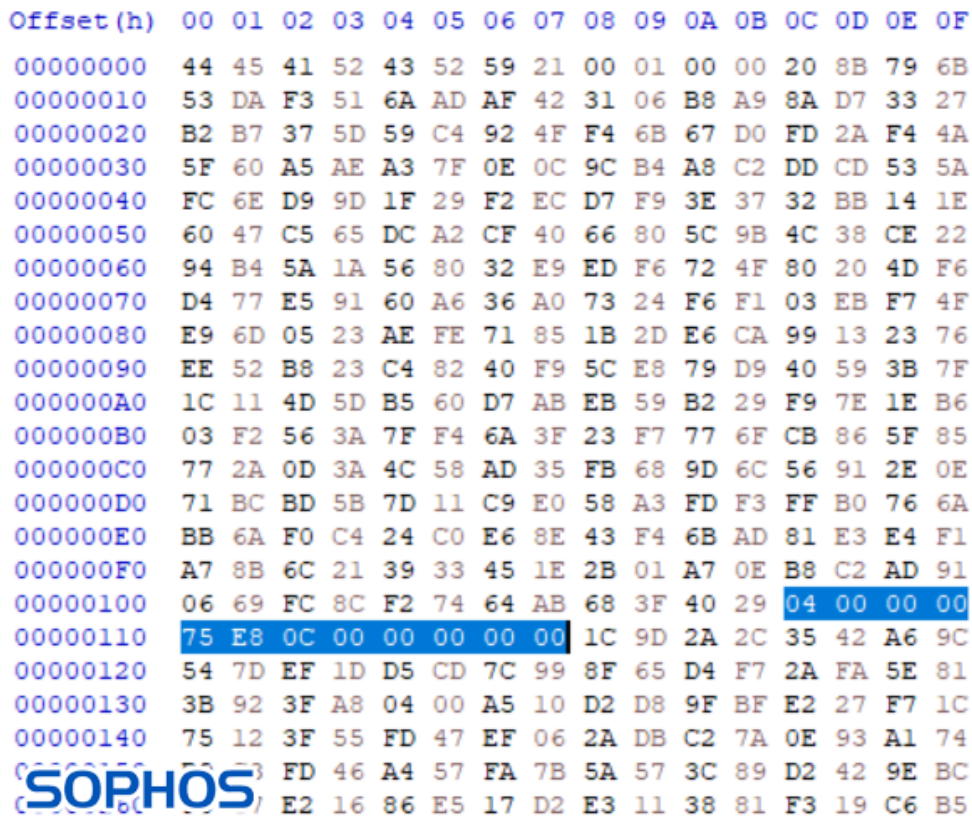
Looking at these file types, DearCry also targets ASPX files. This would mean the attacker could encrypt an Exchange webshell that allowed hands-on-keyboard from remote. This leads us to believe that the ransomware may not be deployed via a webshell, or that the attacker has no interest in keeping webshell-access. The later variant (sample 2) also targets installed software, i.e., EXE and DLL files, effectively making the machine useless:



DearCry ransomware encrypting applications and DLLs, preventing them from running.

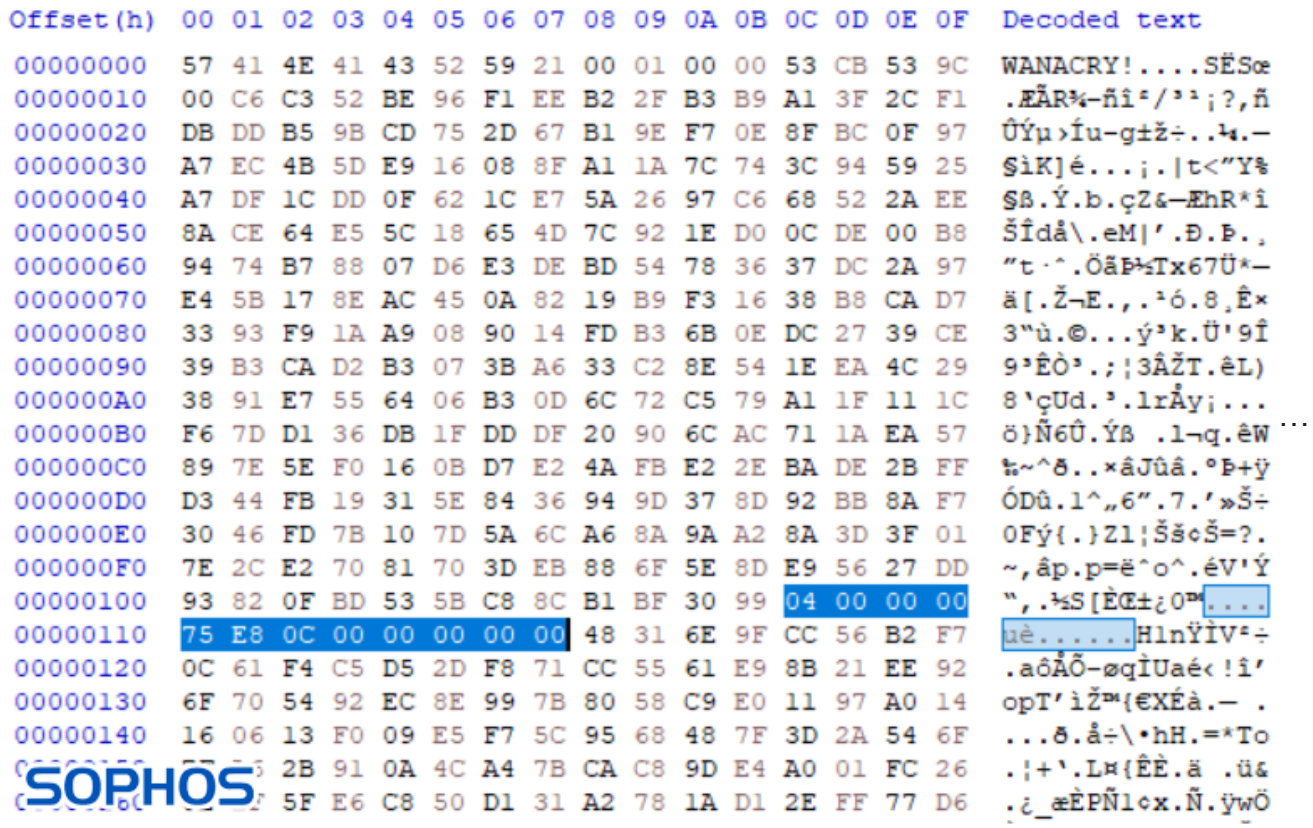
DearCry and WannaCry

Intriguingly, the encryption header that DearCry adds to the attacked files looks similar to the header used by the notorious WannaCry – a ransomware worm that shook the cyber-realm in May 2017.



Offset (h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Decoded text
00000000	44	45	41	52	43	52	59	21	00	01	00	00	20	8B	79	6B	DEARCRY!.... ‹yk
00000010	53	DA	F3	51	6A	AD	AF	42	31	06	B8	A9	8A	D7	33	27	SÚóQj.¯B1.¸©Š×3'
00000020	B2	B7	37	5D	59	C4	92	4F	F4	6B	67	D0	FD	2A	F4	4A	²·7]YÄ'OôkgÐý*ôJ
00000030	5F	60	A5	AE	A3	7F	0E	0C	9C	B4	A8	C2	DD	CD	53	5A	_`¥®£...œ´¨ÂÝÍSZ
00000040	FC	6E	D9	9D	1F	29	F2	EC	D7	F9	3E	37	32	BB	14	1E	ünÙ..)òì×ù>72»..
00000050	60	47	C5	65	DC	A2	CF	40	66	80	5C	9B	4C	38	CE	22	`GÅeÜ¢Ï@f€\›L8Î"
00000060	94	B4	5A	1A	56	80	32	E9	ED	F6	72	4F	80	20	4D	F6	"´Z.V€2éíörO€ Mö
00000070	D4	77	E5	91	60	A6	36	A0	73	24	F6	F1	03	EB	F7	4F	Ôwå'`¦6 s$öñ.ë÷O
00000080	E9	6D	05	23	AE	FE	71	85	1B	2D	E6	CA	99	13	23	76	ém.#®þq….-æÊ™.#v
00000090	EE	52	B8	23	C4	82	40	F9	5C	E8	79	D9	40	59	3B	7F	îR¸#Ä‚@ù\èyÙ@Y;.
000000A0	1C	11	4D	5D	B5	60	D7	AB	EB	59	B2	29	F9	7E	1E	B6	..M]µ`×«ëY²)ù~.¶
000000B0	03	F2	56	3A	7F	F4	6A	3F	23	F7	77	6F	CB	86	5F	85	.òV:.ôj?#÷woË†_…
000000C0	77	2A	0D	3A	4C	58	AD	35	FB	68	9D	6C	56	91	2E	0E	w*.:LX.5ûh.lV'..
000000D0	71	BC	BD	5B	7D	11	C9	E0	58	A3	FD	F3	FF	B0	76	6A	q¼½[}.ÉàX£ýóÿ°vj
000000E0	BB	6A	F0	C4	24	C0	E6	8E	43	F4	6B	AD	81	E3	E4	F1	»jðÄ$ÀæŽCôk.ãäñ
000000F0	A7	8B	6C	21	39	33	45	1E	2B	01	A7	0E	B8	C2	AD	91	§‹l!93E.+.§.¸Â.'
00000100	06	69	FC	8C	F2	74	64	AB	68	3F	40	29	04	00	00	00	.iüŒòtd«h?@)....
00000110	75	E8	0C	00	00	00	00	00	1C	9D	2A	2C	35	42	A6	9C	uè.....¦œ
00000120	54	7D	EF	1D	D5	CD	7C	99	8F	65	D4	F7	2A	FA	5E	81	T}ï.ÕÍ|™.eÔ÷*ú^.
00000130	3B	92	3F	A8	04	00	A5	10	D2	D8	9F	BF	E2	27	F7	1C	;'?¨..¥.ÒØŸ¿â'÷.
00000140	75	12	3F	55	FD	47	EF	06	2A	DB	C2	7A	0E	93	A1	74	u.?UýGï.*ÛÂz.“¡t
[illegible]	[illegible]	[illegible]	[illegible]	FD	46	A4	57	FA	7B	5A	57	3C	89	D2	42	9E	BC	²ÃýF¤Wú{ZW<‰ÒBž¼
[illegible]	[illegible]	[illegible]	[illegible]	E2	16	86	E5	17	D2	E3	11	38	81	F3	19	C6	B5	.Çâ.†å.Òã.8.ó.Æµ

Compare the encryption header and file type and size codes added by DearCry ransomware to a file…



Offset (h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Decoded text
00000000	57	41	4E	41	43	52	59	21	00	01	00	00	53	CB	53	9C	WANACRY!....SËSœ
00000010	00	C6	C3	52	BE	96	F1	EE	B2	2F	B3	B9	A1	3F	2C	F1	.ÆÃR¾–ñî²/³¹¡?,ñ
00000020	DB	DD	B5	9B	CD	75	2D	67	B1	9E	F7	0E	8F	BC	0F	97	ÛÝµ›Íu-g±ž÷..¼.—
00000030	A7	EC	4B	5D	E9	16	08	8F	A1	1A	7C	74	3C	94	59	25	§ìK]é...¡.|t<”Y%
00000040	A7	DF	1C	DD	0F	62	1C	E7	5A	26	97	C6	68	52	2A	EE	§ß.Ý.b.çZ&—ÆhR*î
00000050	8A	CE	64	E5	5C	18	65	4D	7C	92	1E	D0	0C	DE	00	B8	ŠÎdå\.eM|'.Ð.Þ.¸
00000060	94	74	B7	88	07	D6	E3	DE	BD	54	78	36	37	DC	2A	97	"t·ˆ.ÖãÞ½Tx67Ü*—
00000070	E4	5B	17	8E	AC	45	0A	82	19	B9	F3	16	38	B8	CA	D7	ä[.Ž¬E.‚.¹ó.8¸Ê×
00000080	33	93	F9	1A	A9	08	90	14	FD	B3	6B	0E	DC	27	39	CE	3“ù.©...ý³k.Ü'9Î
00000090	39	B3	CA	D2	B3	07	3B	A6	33	C2	8E	54	1E	EA	4C	29	9³ÊÒ³.;¦3ÂŽT.êL)
000000A0	38	91	E7	55	64	06	B3	0D	6C	72	C5	79	A1	1F	11	1C	8'çUd.³.lrÅy¡...
000000B0	F6	7D	D1	36	DB	1F	DD	DF	20	90	6C	AC	71	1A	EA	57	ö}Ñ6Û.Ýß .l¬q.êW
000000C0	89	7E	5E	F0	16	0B	D7	E2	4A	FB	E2	2E	BA	DE	2B	FF	‰~^ð..×âJûâ.ºÞ+ÿ
000000D0	D3	44	FB	19	31	5E	84	36	94	9D	37	8D	92	BB	8A	F7	ÓDû.1^„6”.7.'»Š÷
000000E0	30	46	FD	7B	10	7D	5A	6C	A6	8A	9A	A2	8A	3D	3F	01	0Fý{.}Zl¦ŠšcŠ=?.
000000F0	7E	2C	E2	70	81	70	3D	EB	88	6F	5E	8D	E9	56	27	DD	~,âp.p=ë^o^.éV'Ý
00000100	93	82	0F	BD	53	5B	C8	8C	B1	BF	30	99	04	00	00	00	“‚.½S[ÈŒ±¿0™....
00000110	75	E8	0C	00	00	00	00	00	48	31	6E	9F	CC	56	B2	F7	uè.....H1nŸÌV²÷
00000120	0C	61	F4	C5	D5	2D	F8	71	CC	55	61	E9	8B	21	EE	92	.aôÅÕ-øqÌUaé‹!î'
00000130	6F	70	54	92	EC	8E	99	7B	80	58	C9	E0	11	97	A0	14	opT'ìŽ™{€XÉà.— .
00000140	16	06	13	F0	09	E5	F7	5C	95	68	48	7F	3D	2A	54	6F	...ð.å÷\•hH.=*To
[illegible]	[illegible]	[illegible]	[illegible]	2B	91	0A	4C	A4	7B	CA	C8	9D	E4	A0	01	FC	26	.¦+'.L¤{ÊÈ.ä .ü&
[illegible]	[illegible]	[illegible]	[illegible]	5F	E6	C8	50	D1	31	A2	78	1A	D1	2E	FF	77	D6	.¿_æÈPÑ1¢x.Ñ.ÿwÖ

to a version of the same file, encrypted by WannaCry ransomware

The above two images compare an encrypted file Desert.jpg, a sample picture from a default Windows 7 machine. Both DearCry and WannaCry add their name at the beginning of the file. Furthermore, in WannaCry the highlighted parts represent file type and the size of the original unencrypted file. File type at offset 0x010C and file size at offset 0x0110.

```
61    }
62                        /* Make filename for encrypted file by adding ".CRYPT" */
63    lstrcpynA(&EncryptedFileName_local_524,OrgFileName_2_00,(int)pcVar2);
64    lstrcatA(&EncryptedFileName_local_524,".CRYPT");
65                        /* Open original file for read and open encrypted file for write */
66    OrgFileHandle_local_5c0 = _fopen(OrgFileName_2_00,"rb+");
67    if ((OrgFileHandle_local_5c0 != (FILE *)0x0) &&
68       (EncFileHandle_pFVar3 = _fopen(&EncryptedFileName_local_524,"wb"),
69       EncFileHandle_pFVar3 != (FILE *)0x0)) {
70      local_558 = 0;
71      _memset(local_557,0,0x30);
72      FUN_00406ff0(&local_558,0x30);
73                        /* Write "DEARCRY!" to encrypted file */
74      _fwrite("DEARCRY!",1,8,EncFileHandle_pFVar3);
75      iVar3 = FUN_004070e0(param_3);
76      Header_pvVar4 = _malloc(iVar3 + 1U);
77      _memset(Header_pvVar4,0,iVar3 + 1U);
78                        /* Construct the header, unknown what data the header contains */
79      HeaderSize_sStack1456 = FUN_00407100(0x30,&local_558,Header_pvVar4,param_3,1);
80      if ((int)HeaderSize_sStack1456 < 0) {
81        iVar3 = FUN_004da3e6();
82        FUN_004067b0(iVar3 + 0x20);
83        _fclose(EncFileHandle_pFVar3);
84  LAB_004015a3:
85        _fclose(OrgFileHandle_local_5c0);
86      }
87      else {
88                        /* Write Header size [4 bytes], we know the header is 0x100 bytes */
89        _fwrite(&HeaderSize_sStack1456,1,4,EncFileHandle_pFVar3);
90                        /* Write Header */
91        _fwrite(Header_pvVar4,1,HeaderSize_sStack1456,EncFileHandle_pFVar3);
92        uVar8 = HeaderSize_sStack1456;
93        iVar3 = (int)HeaderSize_sStack1456 >> 0x1f;
94        bVar9 = 0xfffffff3 < HeaderSize_sStack1456;
95        uVar7 = HeaderSize_sStack1456 + 0xc;
96        _free(Header_pvVar4);
97        uStack1432 = 4;
98                        /* Write '4' in 4 bytes => 04 00 00 00 */
99        _fwrite(&uStack1432,1,4,EncFileHandle_pFVar3);
100       __stat64(local_5a8,auStack1428);
101                       /* Write File Size, comes from previous call to __stat64() */
102       _fwrite(auStack1404,1,8,EncFileHandle_pFVar3);
103                       /* Here after the encryption takes place */
104       uVar8 = uVar8 + 0x18;
105       iVar3 = iVar3 + (uint)bVar9 + (uint)(0xfffffff3 < uVar7);
106       Header_pvVar4 = (void *)FUN_00405990();
107       pvStack1452 = Header_pvVar4;
108       uVar4 = FUN_00402f00(0,0,0,1);
109       OpenSSL_FUN_00406270(Header_pvVar4,uVar4);
110       OpenSSL_FUN_00406270(Header_pvVar4,0,0,&local_558,auStack1336,1);
111       _fread(local_5a4,1,0x100000,OrgFileHandle_local_5c0);
112       Header_pvVar4 = local_5b4;
```


Code in DearCry writing the header in format similar to WannaCry (with comments added for clarification).

The hybrid approach

From an anti-ransomware perspective, looking at DearCry's file system behaviors reveals more interesting details. The following table illustrates its behavior for each file DearCry attacks (these can be observed using Process Monitor):

Step	Operation	Purpose
1	CreateFile (Generic Read)	Open original document for read only.
2	ReadFile	Read original document from offset 0, length 4,096 bytes (Check if document was already encrypted)
3	CloseFile	Close original document.
4	CreateFile (Generic Read/Write)	Open original file for reading and writing.
5	CreateFile (Generic Write)	Create a new file with .CRYPT extension, opened for writing.
6	ReadFile	Read original document
7	WriteFile	Write 4,096 bytes in encrypted file, at offset 0 bytes. (Write header in encrypted document)
8	WriteFile	Write encrypted document in encrypted file.
9	CloseFile	Close encrypted file.
10	WriteFile	Overwrite original document to prevent recovery via undelete.
11	CloseFile	Close original document.
12	CreateFile (Read Attributes)	Open original document.
13	SetDispositionInformationFile	Delete: True.
14	CloseFile	Close original document, committing delete.

SOPHOSlabs

File system behavior of DearCry. Step 8 = Copy. Step 10 = In-Place.

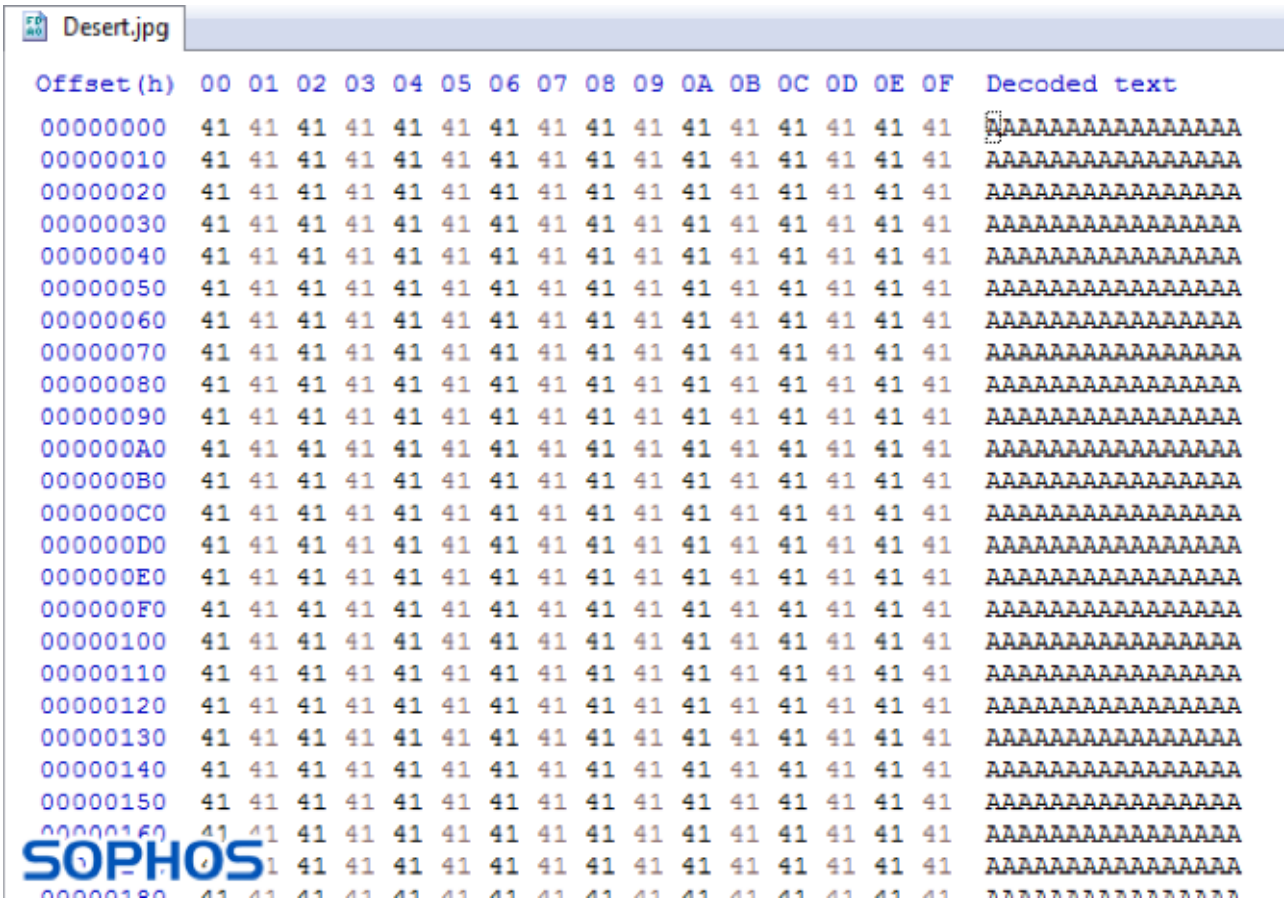
From this behavior, DearCry is what we'd normally call a Copy ransomware. It creates encrypted copies of the attacked files and deletes the originals. This causes the encrypted files to be stored on different logical sectors, normally allowing victims to recover maybe some data – depending on whether Windows reuses the freed logical sectors.

Compared to DearCry, more notorious human-operated ransomware like Ryuk, REvil, BitPaymer, Maze and Clop, are In-Place ransomware, where the attack immediately causes the encrypted file to be stored on logically the same sectors as the original document, making recovery via undelete tools impossible.

But DearCry has an added trick to make recovery impossible. Before deleting the original document and after closing the encrypted copy, it also overwrites the original document. This means DearCry has a hybrid encryption approach: it performs both a Copy and In-Place encryption attack.

```
139       _fclose(EncFileHandle_pFVar3);
140       EncFileHandle_pFVar3 = OrgFileHandle_local_5c0;
141                     /* Rewind file pointer for original file to begin of the file */
142       __fseeki64(OrgFileHandle_local_5c0,0,0);
143       iVar3 = local_5b8;
144       uStack1476 = local_5bc;
145       if ((-1 < local_5b8) && ((0 < local_5b8 || (0x100000 < local_5bc)))) {
146         uStack1476 = 0x100000;
147       }
148       uVar8 = 0;
149       iVar5 = 0;
150       if ((-1 < local_5b8) && ((0 < local_5b8 || (local_5bc != 0)))) {
151         do {
152           if ((int)uStack1476 < 0) {
153             uStack1476 = 0x100000;
154           }
155                     /* Overwrite the original file in blocks of 0x100000 (1MB).
156                        Note: The block data OverWiteData_DAT_00539ef0 is filled 0x41 by function  */
157           sVar6 = _fwrite(&OverWiteData_DAT_00539ef0,1,uStack1476,EncFileHandle_pFVar3);
158           bVar9 = CARRY4(uVar8,sVar6);
159           uVar8 = uVar8 + sVar6;
160           iVar5 = iVar5 + ((int)sVar6 >> 0x1f) + (uint)bVar9;
161           uStack1476 = local_5bc - uVar8;
162           iStack1440 = (iVar3 - iVar5) - (uint)(local_5bc < uVar8);
163           if ((iStack1440 >= 0) &&
164              ((iStack1440 != 0 &&
165                (SBORROW4(iVar3,iVar5) != SBORROW4(iVar3 - iVar5,(uint)(local_5bc < uVar8))) ==
166                iStack1440 < 0 || (0x100000 < uStack1476)))) {
167             uStack1476 = 0x100000;
168           }
169           __fseeki64(EncFileHandle_pFVar3,CONCAT44(iVar5,uVar8),0);
170         } while ((iVar5 < iVar3) || ((iVar5 <= iVar3 && (uVar8 < local_5bc))));
171       }
172       _fclose(EncFileHandle_pFVar3);
173       DeleteFile_004da686(local_5a8);
174 SOPHOS
175 }
```

DearCry overwriting the original document after making its encrypted copy.



Original Desert.jpg overwritten by DearCry

This approach is not common. It is so uncommon, the only ransomware that comes to mind that also employed a hybrid approach was WannaCry. It also first created an encrypted copy and then overwrote the original file to prevent recovery:

Step	Operation	Purpose
1	CreateFile (Generic Read)	Open original document for reading only.
2	ReadFile	Read original document from offset 0, length 8 bytes (Check if document was already encrypted)
3	CloseFile	Close original document.
4	CreateFile (Generic Read)	Open original file for reading only.
5	CreateFile (Generic Write)	Create a new file with `.WNCRYT` extension, opened for writing.
6	WriteFile	Write header in encrypted file.
7	ReadFile	Read original document.
8	WriteFile	Write encrypted document in encrypted file.
9	CloseFile	Close original document.
10	SetRenameInformationFile	Change file extension of encrypted file to `.WNCRY`
11	CloseFile	Close encrypted file.
12	CreateFile (Generic Write)	Open original file for writing only.
13	WriteFile	Overwrite original document to prevent recovery via undelete.
14	CloseFile	Close original document.
15	OpenFile (Read Attributes)	Open encrypted file.
16	SetRenameInformationFile	Rename (and move) original file to `%temp%\<num>.WNCRYT`. ReplaceIfExists: True.
17	CloseFile	Close encrypted file.
#	SetDispositionInformationFile	Once all documents on the disk are encrypted, a separate application `TASKDL.EXE` is run to delete `%temp%\*.WNCRYT` (i.e., all original files).

SOPHOSlabs

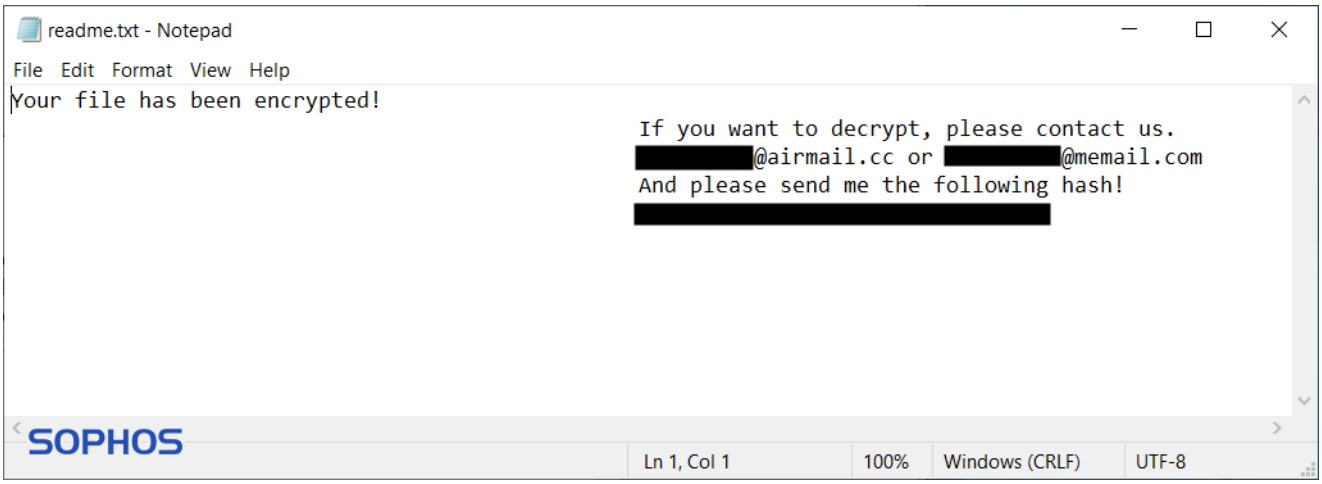
File system behavior of WannaCry. Step 8 = Copy. Step 13 = In-Place.
Details about file system behaviors of notorious ransomware, including WannaCry, are documented in our How Ransomware Attacks whitepaper.

These attributes don't directly link DearCry to WannaCry's creator. DearCry's code, approach and abilities differ significantly from WannaCry: it does not use a command-and-control (C2) server, has an embedded RSA encryption key, shows no user interface with a timer and – most importantly – does not spread itself to other machines on the network. And the DearCry ransomware binary itself does not delete volume shadow copies.

Ransom notes

To inform victims about what happened and who they must contact, a file 'readme.txt' is dropped in every folder containing the word 'desktop' and in the root folder of the system disk. This ransom note contains two e-mail addresses and a hash. This hash is an identifier,

so the attacker knows what decryption key is associated with the specific attack.



DearCry ransom note.

Detections

Sophos customers may see DearCry detected as **Troj/Ransom-GFE**. In some circumstances, the endpoint protection tools on affected Exchange servers also detected a web shell dropped a couple of days before the ransomware; Those components will be reported as **Troj/WShell-A**.

Specific indicators for the samples described herein have been published to the SophosLabs Github.

Acknowledgments

SophosLabs would like to acknowledge the contributions of Alex Vermaning and Fraser Howard to this report.