A new Equation Editor exploit goes commercial, as maldoc attacks using it spike

Weaponized RTF documents adopt CVE-2018-0798, another Equation Editor vulnerability

Written by Gabor Szappanos

JULY 18, 2019

SOPHOSLABS UNCUT THREAT RESEARCH .DOC CVE-2018-0798 EQUATION EDITOR EXPLOIT MALDOC MICROSOFT OFFICE MICROSOFT WORD PDF RTF

There is a distinct point of maturation in the life cycle of an Office exploit: the point where it becomes generally available for the crimeware groups. Before that point the exploit has affected only a few selected victims of targeted attacks. From that point on it becomes widespread and threatens the general user population.

CVE-2018-0798, yet another Equation Editor vulnerability, reached this point of maturation around the end of June 2019.

How the exploit works

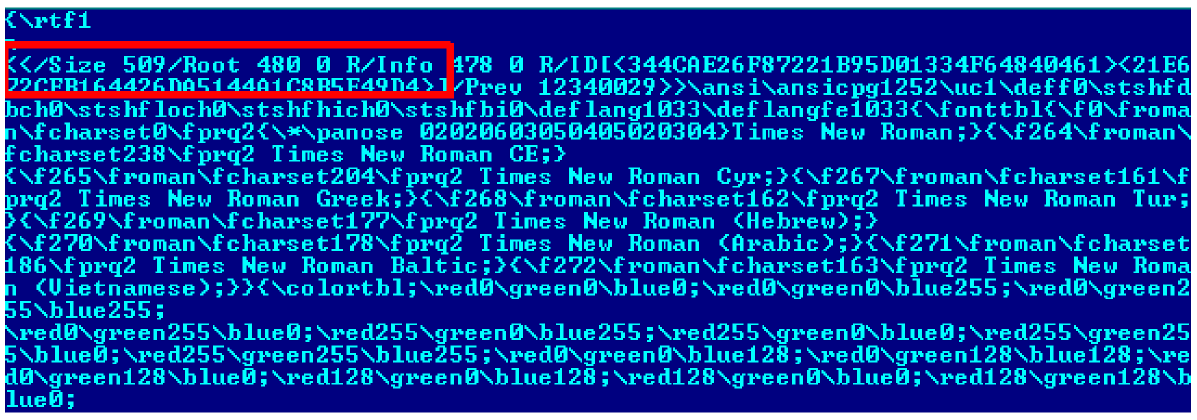
This exploit was used in targeted attacks we observed over the past several months, staying mostly under the radar. A [detailed description of the vulnerability]() and its usage in targeted attacks, published earlier this month, explains the fundamentals.

At some point, someone appears to have integrated the exploit implementation that was used in the targeted attacks into a malicious Office document *builder* tool, with only minor changes. This sets the stage for much broader adoption of the attack technique.

The similarities are unmistakable: For example, the exploit trigger is exactly the same, and the shellcode (responsible for decrypting and executing the payload) is nearly identical to the shellcode used in the targeted attacks. Although CVE-2018-0798 is a relatively simple buffer overflow vulnerability, there are a few factors that make it difficult to tweak the exploit trigger itself, hence the virtually identical implementation.

The attack uses Equation Editor to put the malicious buffer through several transformations, and during processing the memory layout will be rather different from the file layout, which complicates analysis (and tweaking). Moreover, the attacker only controls the lower word of the return address on the stack. These complications make it less likely that the crimeware implementations will significantly change the exploit trigger.

The weaponized RTF files that are part of this campaign have a curious characteristic: some junk content near the header — content that fits a pattern normally seen in PDF files. It seems like the presence of this PDF header data content might be an attempt to confuse document filetype parsers in AV scanners.



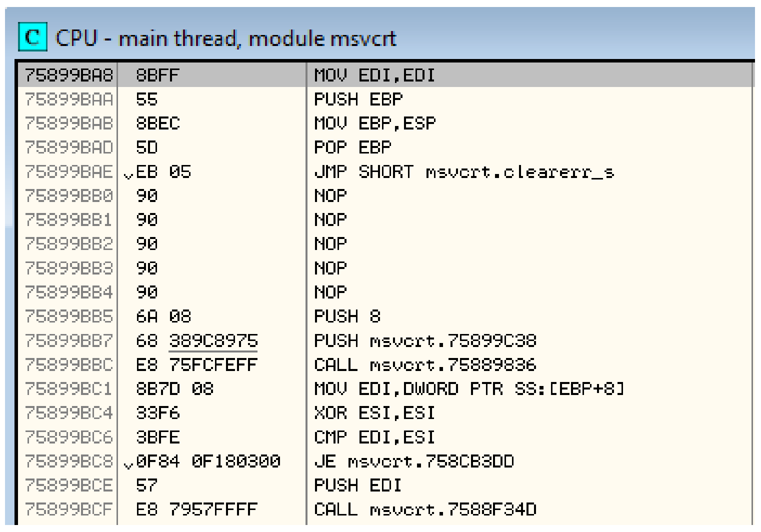
```
{\rtf1

<</Size 509/Root 480 0 R/Info 478 0 R/ID[<344CAE26F87221B95D01334F64840461><21E6
22CFB164426DA514401C8B5F49D4>]/Prev 12340029>>\ansi\ansicpg1252\uc1\deff0\stshfd
bch0\stshfloch0\stshfhich0\stshfbi0\deflang1033\deflangfe1033{\fonttbl{\f0\froma
n\fcharset0\fprq2{\*\panose 02020603050405020304}Times New Roman;}{\f264\froman\
fcharset238\fprq2 Times New Roman CE;}
{\f265\froman\fcharset204\fprq2 Times New Roman Cyr;}{\f267\froman\fcharset161\f
prq2 Times New Roman Greek;}{\f268\froman\fcharset162\fprq2 Times New Roman Tur;
}{\f269\froman\fcharset177\fprq2 Times New Roman (Hebrew);}
{\f270\froman\fcharset178\fprq2 Times New Roman (Arabic);}{\f271\froman\fcharset
186\fprq2 Times New Roman Baltic;}{\f272\froman\fcharset163\fprq2 Times New Roma
n (Vietnamese);}}{\colortbl;\red0\green0\blue0;\red0\green0\blue255;\red0\green2
55\blue255;
\red0\green255\blue0;\red255\green0\blue255;\red255\green0\blue0;\red255\green25
5\blue0;\red255\green255\blue255;\red0\green0\blue128;\red0\green128\blue128;\re
d0\green128\blue0;\red128\green0\blue128;\red128\green0\blue0;\red128\green128\b
lue0;
```

Malicous RTF file with PDF fragments

An interesting feature in the shellcode is that the Windows API functions are not called from the shellcode itself. Instead, the clearerr function from the msvcrt.dll is patched with a short code that invokes the desired API function.

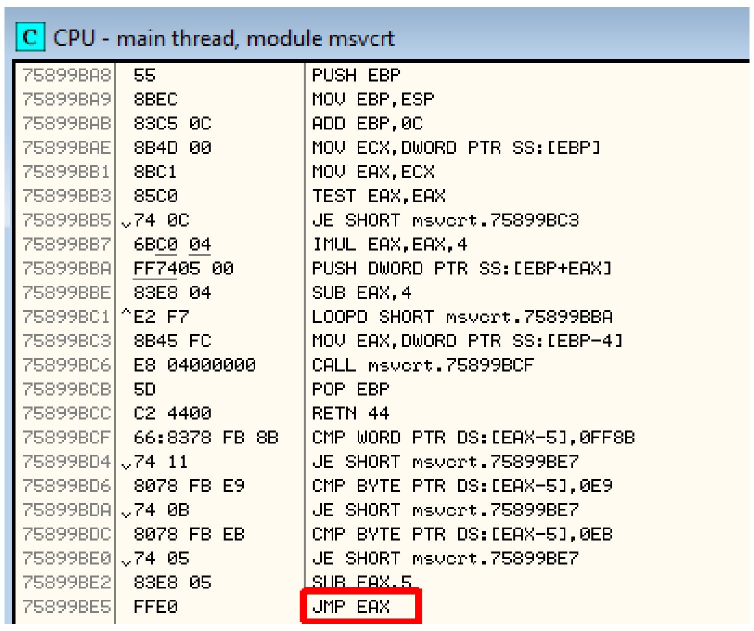
The original code of clearerr looks like this:



```
75899BA8   8BFF           MOV EDI,EDI
75899BAA   55             PUSH EBP
75899BAB   8BEC           MOV EBP,ESP
75899BAD   5D             POP EBP
75899BAE  ˅EB 05          JMP SHORT msvcrt.clearerr_s
75899BB0   90             NOP
75899BB1   90             NOP
75899BB2   90             NOP
75899BB3   90             NOP
75899BB4   90             NOP
75899BB5   6A 08          PUSH 8
75899BB7   68 389C8975    PUSH msvcrt.75899C38
75899BBC   E8 75FCFEFF    CALL msvcrt.75889836
75899BC1   8B7D 08        MOV EDI,DWORD PTR SS:[EBP+8]
75899BC4   33F6           XOR ESI,ESI
75899BC6   3BFE           CMP EDI,ESI
75899BC8  ˅0F84 0F180300  JE msvcrt.758CB3DD
75899BCE   57             PUSH EDI
75899BCF   E8 7957FFFF    CALL msvcrt.7588F34D
```

clearerr before the patch

This is replaced with a code that first checks if the API function was already hooked by a security software (the presence of 0xE9 and 0xEB indicate that the original entry of the API handler was patched, redirected to a monitoring function). If patching is detected, the shellcode will skip the first 5 bytes, the redirector, and jumps directly into the body of the function, done by the JMP EAX instruction.


CPU - main thread, module msvcrt

```
75899BA8  55              PUSH EBP
75899BA9  8BEC            MOV EBP,ESP
75899BAB  83C5 0C         ADD EBP,0C
75899BAE  8B4D 00         MOV ECX,DWORD PTR SS:[EBP]
75899BB1  8BC1            MOV EAX,ECX
75899BB3  85C0            TEST EAX,EAX
75899BB5 ⌄74 0C           JE SHORT msvcrt.75899BC3
75899BB7  6BC0 04         IMUL EAX,EAX,4
75899BBA  FF7405 00       PUSH DWORD PTR SS:[EBP+EAX]
75899BBE  83E8 04         SUB EAX,4
75899BC1 ^E2 F7           LOOPD SHORT msvcrt.75899BBA
75899BC3  8B45 FC         MOV EAX,DWORD PTR SS:[EBP-4]
75899BC6  E8 04000000     CALL msvcrt.75899BCF
75899BCB  5D              POP EBP
75899BCC  C2 4400         RETN 44
75899BCF  66:8378 FB 8B   CMP WORD PTR DS:[EAX-5],0FF8B
75899BD4 ⌄74 11           JE SHORT msvcrt.75899BE7
75899BD6  8078 FB E9      CMP BYTE PTR DS:[EAX-5],0E9
75899BDA ⌄74 0B           JE SHORT msvcrt.75899BE7
75899BDC  8078 FB EB      CMP BYTE PTR DS:[EAX-5],0EB
75899BE0 ⌄74 05           JE SHORT msvcrt.75899BE7
75899BE2  83E8 05         SUB EAX,5
75899BE5  FFE0            JMP EAX
```

clearerr after the patch

The result of this trick is that all of the Windows API calls that the shellcode uses will appear to be initiated from msvcrt.dll, which makes it a much less suspicious activity for behavior monitoring software.

The only significant change in the crimeware cases is the extracted payload. The APT samples were self-contained, the final payload was stored in the RTF file. On the other hand, in the crimeware samples the payload is a very simple downloader trojan, that fetches the final payload from a hardcoded URL.

In both cases the RTF file contains an embedded payload. There were a few different implementations in the APT cases. In the most simple of them, the payload is embedded into the RTF file without any encryption, as illustrated in the following picture where the MZ marker of the embedded executable is clearly visible in the RTF content.

Embedded executable payload's characteristic MZ header is clearly visible in RTF

In the typical cases a one-byte XOR algorithm is used where the encryption key changes for each data byte. This makes it more difficult to recognize the embedded content which at first looks like some random data.

Embedded encrypted content in RTF

The crimeware builder chose a bogus (yet more convenient) implementation of the latter, more details in the next section.

The downloader component

The CVE-2018-0798 exploit triggers the shellcode that decrypts the payload (one-byte XOR algorithm, the key is **0xFC**) and executes it. More precisely, the encryption algorithm is supposed to be a running key byte-wise XOR algorithm, but the APT sample that was used as a template for the crimeware builder used a bogus implementation of this algorithm.

Instead of key changing for each file position, the key changes only for the first 4 bytes, for the rest of the file it will be 0xFC. The following picture shows large blocks of 0xFC bytes in the embedded payload, which clearly reveals the encryption key.

The encrypted downloader with the URL in it

It should not be a coincidence that the crimeware builder chose this implementation. It is much easier to patch only the hardcoded encoded URL in the RTF file with fixed key, than to recode the embedded EXE and rebuild the RTF file for every sample.

The program downloads the hardcoded URL, saves it to the %APPDATA% folder, then executes it.

The critical Windows API function names are slightly obfuscated, the first character of the name is changed.

The same downloader executable is present in all recent crimeware sample; Only the hardcoded download URL changes. This is a good choice for the easy development of the builder.

What the malicious spam messages look like

We've observed several email distribution campaigns using these weaponized Office documents. The identified payload of the infection campaigns were all the "usual suspect" crimeware family payloads: Fareit, Lokibot, Formbook, or AzoruLT. The most widely distributed payload is Fareit.

We have observed that the spam campaigns use a wide selection of social engineering tropes.

Cargo arrival notice themed messages:

Fedex tracking was particularly popular:

And of course the order confirmation spam:

Lokibot was distributed using a “payment advice”-themed message:

And in messages that purported to contain a product inquiry:

The “give us your best price quote” email was used to distribute Formbook:

The vulnerability affects all Equation Editor versions (even the ones that were patched for CVE-2017-11882).

The patch for the CVE-2018-0802 exploit permanently “fixes” the vulnerability by eliminating the Equation Editor altogether.

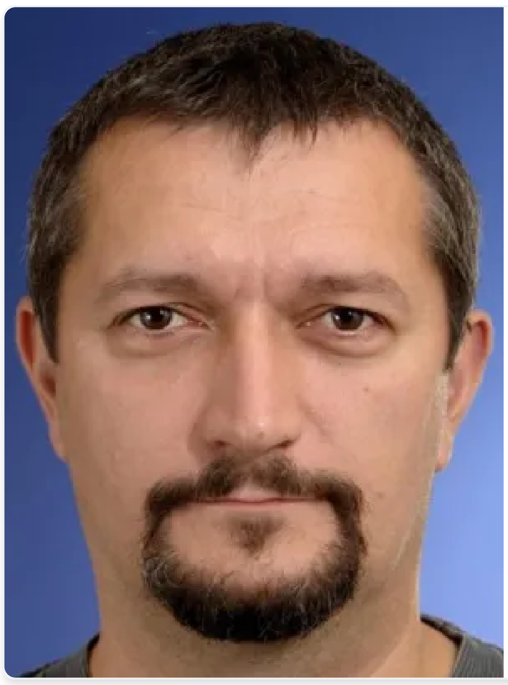
Since the end of June, we’ve started to observe an increase in the use of this vulnerability in phishing campaigns. Since that time, we’ve observed about 200 new malicious RTF documents using this exploit. These documents are very similar to each other. This, and the large number of samples indicated that a builder tool that generates the weaponized .doc files probably is in circulation. We can expect the use of this exploit to rise, at least for the near future.

We have the following detections that provide generic coverage for the known samples:

Exp/20180798-A
Troj/RtfExp-FB
Troj/RtfExp-EY

Indicators of Compromise

File hashes for the samples analyzed in this report are on the SophosLabs Github.



About the Author

Gabor Szappanos

Gabor graduated from the Eotvos Lorand University of Budapest with a degree in physics. His first job was in the Computer and Automation Research Institute, developing diagnostic software and hardware for nuclear power plants. He started antivirus work in 1995, and began developing freeware antivirus solutions in his spare time. Gabor joined VirusBuster in 2001 where he was responsible for taking care of macro virus and script malware and became head of the virus lab in 2002. In 2008 he became a member of the Board of Directors in AMTSO (Anti Malware Testing Standards Organization) and, in 2012, joined Sophos as a Principal Malware Researcher.

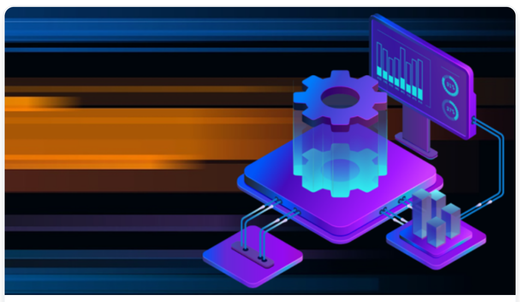
Read Similar Articles

MAY 24, 2021

What to expect when you've been hit with Avaddon...

MAY 19, 2021

What's New in Sophos EDR 4.0



MAY 19, 2021

Sophos XDR: Driven by data

Subscribe to get the latest updates in your inbox.

name@email.com

Which categories are you interested in?

- ☐ Products and Services
- ☐ Threat Research
- ☐ Security Operations
- ☐ AI Research
- ☐ #SophosLife

Subscribe

Change Region

Terms Privacy Legal