Microsoft Office OLE2Link vulnerability samples - a quick triage

Ahmed Zaki Rich Warren David Cannings

April 2017

Contents

1 Handling information

This document was produced by the NCC Group Cyber Defence Operations team. The content of this document should be considered proprietary information. NCC Group has released this report publicly and gives permission to copy it at TLP WHITE. Please see the US CERT website for full details of the traffic light marking system.

2 Introduction

On April 7th 2017 Haifei Li published on the McAfee blog[1] about a "Critical Office Zero-Day" in the wild. Few details were given and no hashes were available, which made it interesting to find samples and conduct an initial analysis. A further blog by FireEye[2] titled "Acknowledgement of Attacks Leveraging Microsoft Zero-Day" provided additional useful information.

During testing we were able to generate a number of proof-of-concept (PoC) documents both with and without a prompt to the user. It is likely the vulnerability will be documented in full detail over the coming days. Therefore we instead discuss a number of ways to detect and analyse these documents using freely available tools. This information may be useful to any incident responder or blue team looking to defend an organisation.

At the time of writing there is no assigned CVE or credit from Microsoft. It seems likely that Ryan Hanson of Optiv was responsible for some of the initial disclosure[3] to Microsoft. This document will be updated as further information emerges.

3 Finding samples

3.1 Initial hints

The McAfee and FireEye blogs provide few details. However, the following hints are sufficient to find initial samples:

- The vulnerability has been exploited using RTF documents with `.doc` extensions.
- An `OLE2Link` object is used.
- The document will connect to a remote server and download a Microsoft HTML Application (`.hta`) file.

The McAfee blog provides some screenshots which confirm the `Content-Type` returned by the server is `application/hta`.

3.2 Likely candidates

OLE objects can be embedded in RTF documents by class identifier or class name. Mappings for class identifier to name are stored in the registry under `HKEY_CLASSES_ROOT\CLSID\` (with appropriate duplicates in `Wow6432Node` for 64-bit systems).

When searching for exploit documents it is useful to compile a list of likely class identifiers. In this case there are no matches for `OLE2Link`. However, a search for `OLElink` finds the object `StdOleLink` with the class identifier `00000300-0000-0000-C000-000000000046`.

With this information it was possible to start a retrohunt on VirusTotal. The Yara rule used is provided in section 6.2. The search returned 11 hits, discussed in further detail below.

3.3 Other search tips

VirusTotal intelligence subscribers can search for files tagged `ole-link` or `ole-autolink` to find related samples.

[1] https://securingtomorrow.mcafee.com/mcafee-labs/critical-office-zero-day-attacks-detected-wild/
[2] https://www.fireeye.com/blog/threat-research/2017/04/acknowledgement_ofa.html
[3] https://twitter.com/ryHanson/status/851338798369722369

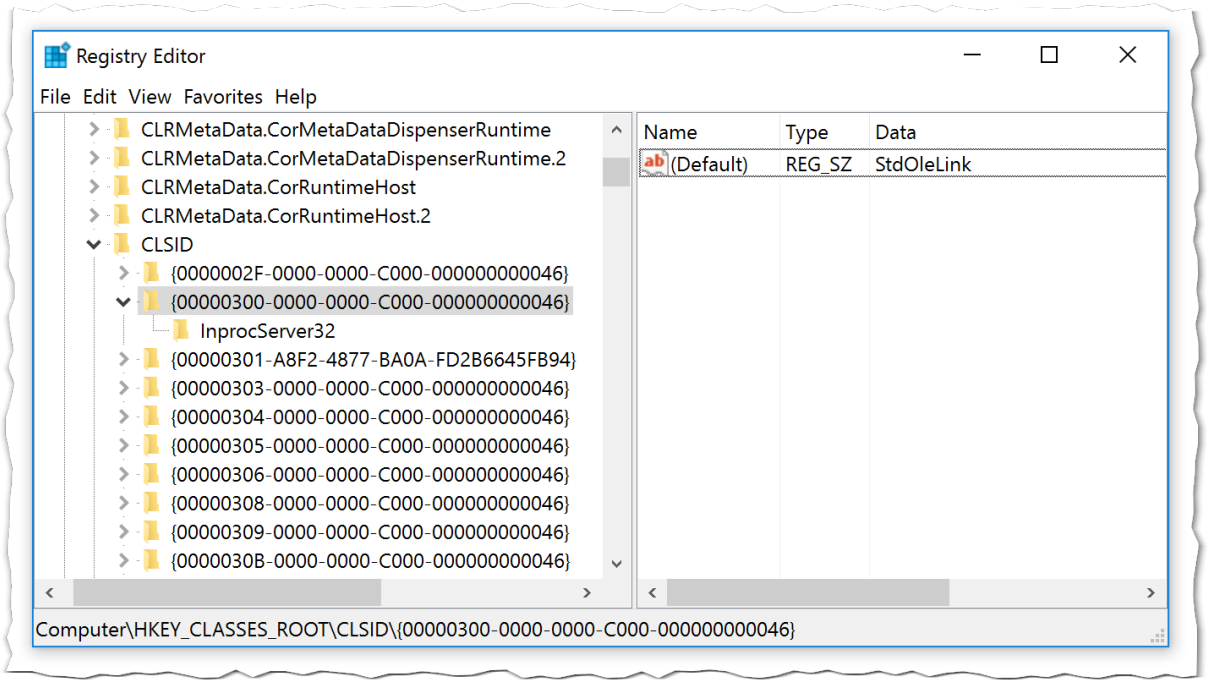


Figure 1: StdOleLink registry entry

4 Samples

4.1 Timeline

The earliest known times obtained by NCC Group are shown below. However, note that document metadata can be trivially modified. It is interesting that a sample using the `OLE2Link` class identifier was first seen in-the-wild in January 2016.

Time (timezones unverified)	Notes
2015-10-23 15:28	Creation time of `5af7..` from RTF metadata
2016-01-13 07:06:06	VirusTotal - First seen in-the-wild for `5af7..`
2016-10-11 11:12:17	VirusTotal - First submission for `5af7..`
2016-11-23 22:11:07	VirusTotal - First seen in-the-wild for `14e4..` (potentially created by Optiv)
2016-11-27 22:42	Creation time of multiple group 3 docs from RTF metadata
2016-12-27 11:20	Creation time of `e933..` from RTF metadata (potentially created by Optiv)
2016-12-29 18:20:25	VirusTotal - First seen in-the-wild for `e933..`

4.2 Group 1 - early sample

4.2.1 5af7fe6b74cf91450961cdb7fc31919e4cb6e401b693d99d2f4956697c5cb8ad

This document was named `pi.doc` and `devenum-noclick.rtf` and was first submitted to VirusTotal on 13th January 2016. The creation time from RTF metadata was the 23rd October 2015. This file was uploaded to the Hybrid Analysis sandbox[4] on January 18th 2017.

Although the `OLE2Link` class identifier and name are present the document does not contain an obvious URL and crashed Word during testing.

4.3 Group 2 - likely Optiv

Based on RTF metadata the documents in this group appears to have been created by a red team associated with Optiv. The earliest document was first submitted to VirusTotal on 23rd November 2016 and the most recent on 16th February 2017.

4.3.1 14e4d9269304d5e92f300adfcc5cc4f65ead9b3898a3efbeac7e321ef3ca3b40

This document was potentially named after a client and contains only the following text:

```
Error: 0X00001324 - There was a problem loading this file. (Server Unresponsive)
Please try again later.
```

Subsequent stages are downloaded from `http://107[.]170[.]240[.]244/download/omgrtf.doc`, which was not active at the time of writing.

4.3.2 a7fa6e64286134448b369e4241798907eb9afd01d4024d51bc3a2790c453dd15

This document was named `resume.doc` and is a CV for “Robin Chase”. This appears to have come from the LiveCareer[5] website. According to the CV Robin has “Superior abilities in using MS Office suite applications” :)

This document also fetches a subsequent stage from `http://107[.]170[.]240[.]244/download/omgrtf.doc`.

[4] https://www.hybrid-analysis.com/sample/5af7fe6b74cf91450961cdb7fc31919e4cb6e401b693d99d2f4956697c5cb8ad
[5] http://www.livecareer.co.uk/templates/cv-samples/cv-objectives/account-executive-cv-objectives/

4.3.3 e9339747b31f576e6d4049696a4f4bd7053bcd29dafb0a7f2e55b8aab1539b67

This document was named `Malicious.rtf` and contains a screenshot of a website relating to a US energy supplier.

Subsequent stages would be downloaded from `http://d218w8g44zaxak[.]cloudfront[.]net/Doc1.jpg`, which was not active at the time of writing.

4.4 Group 3

This group contains a number of apparently related documents which download a subsequent file named `template.doc`. It appears likely these samples are the ones referred to in the McAfee blog post.

Sample sashes and file names are shown below. It is likely some of these were created by analysts or sandbox environments. SHA256 sums of these documents are:

- `13d0d0b67c8e881e858ae8cbece32ee464775b33a9ffcec6bff4dd3085dbb575`
- `3c0a93d05b3d0a9564df63ed6178d54d467263ad6e3a76a9083a43a7e4a9cca5`
- `b3b3cac20d93f097b20731511a3adec923f5e806e1987c5713d840e335e55b66`
- `b9b92307d9fffff9f63c76541c9f2b7447731a289d34b58d762d4e28cb571fbd`
- `d3cba5dcdd6eca4ab2507c2fc1f1f524205d15fd06230163beac3154785c4055`
- `b9147ca1380a5e4adcb835c256a9b05dfe44a3ff3d5950bc1822ce8961a191a1`
- `4453739d7b524d17e4542c8ecfce65d1104b442b1be734ae665ad6d2215662fd`

Filenames used include `hire_form.doc`, `document.doc`, `testThis.txt`, `!!!URGENT!!!!READ!!!.doc`, `PDP.doc`, `~WRD0000.tmp` and `!!!!URGENT!!!!READ!!!.rtf`. A number of these are available from online sandboxes.

The following URLs are used by these documents for the subsequent stage, which contains an embedded script. Most of these were still available at the time of writing.

- `http://212[.]86[.]115[.]71/template.doc`
- `http://46[.]102[.]152[.]129/template.doc`
- `http://95[.]141[.]38[.]110/mo/dnr/tmp/template.doc`
- `http://95[.]46[.]99[.]199/template.doc`

5 Analysing a sample

For the purposes of this document we have chosen the sample with SHA256 beginning `3c0a..` to analyse in more detail.

5.1 Identifying the file

The filetype can be determined very quickly using standard command line tools. This matches information provided by McAfee, despite being named `document.doc` this is actually an RTF file.

```
% file 3c0a93d05b3d0a9564df63ed6178d54d467263ad6e3a76a9083a43a7e4a9cca5
3c0a93d05b3d0a9564df63ed6178d54d467263ad6e3a76a9083a43a7e4a9cca5:
  Rich Text Format data, version 1, unknown character set
```

This can easily be confirmed by inspecting the first few bytes of the file. Many malicious RTFs begin with only `{\rt` (presumably for file scanning avoidance) however this sample uses the standards compliant `{\rtf1` string:

```
00000000: 7b5c 7274 6631 5c61 6465 666c 616e 6731  {\rtf1\adeflang1
00000010: 3032 355c 616e 7369 5c61 6e73 6963 7067  025\ansi\ansicpg
```

5.2 Finding the embedded object

To confirm this is a sample of the malicious document it is necessary to understand a little about the RTF file structure. Embedded OLE objects are found in a `{\object` container, as shown below:

```
00003130: 3820 7b5c 6f62 6a65 6374 5c6f 626a 6175  8 {\object\objau
00003140: 746c 696e 6b5c 7273 6c74 7069 6374 0d0a  tlink\rsltpict..
00003150: 5c6f 626a 7734 325c 6f62 6a68 3532 7b5c  \objw42\objh52{\
00003160: 2a5c 6f62 6a63 6c61 7373 204f 6666 6963  *\objclass Offic
00003170: 6544 4f43 7d7b 5c2a 5c6f 626a 6461 7461  eDOC}{\*\objdata
00003180: 2030 3130 3530 3030 3030 3230 3030 3030   010500000200000
00003190: 3030 3930 3030 3030 3034 6634 6334 3533  0090000004f4c453
000031a0: 3234 6336 3936 6536 6230 3030 3030 3030  24c696e6b0000000
000031b0: 3030 3030 3030 3030 3030 3030 3030 6130  00000000000000a0
000031c0: 3030 300d 0a64 3063 6631 3165 3061 3162  000..d0cf11e0a1b
```

Neither this sample or related samples include the `\oleclsid` in the RTF file. This is different from files in group 1 and 2 which do include the correct `\oleclsid`. However, the class name is `4f4c45324c696e6b` which is the hex encoded version of `OLE2Link`.

The data beginning `d0cf11e` is a sure sign that a Composite Document File (CDF) has been embedded in the RTF. CDF is used as a container for a number of file types but is probably best known as the “binary” Office document format, e.g. `.doc`.

5.3 Automatically extracting the OLE object

This manual analysis can be automated easily using the excellent rtfobj[6] tool from Decalage. Running with the filename as an argument will display a list of embedded OLE objects:

```
% rtfobj 3c0a93d05b3d0a9564df63ed6178d54d467263ad6e3a76a9083a43a7e4a9cca5
rtfobj 0.50 - http://decalage.info/python/oletools
THIS IS WORK IN PROGRESS - Check updates regularly!
Please report any issue at https://github.com/decalage2/oletools/issues
```

[6] https://github.com/decalage2/oletools/wiki/rtfobj

```
================================================================================
File: '3c0a..' - size: 37518 bytes
---+----------+------------------------------+------------------------------
id |index     |OLE Object                    |OLE Package
---+----------+------------------------------+------------------------------
0  |00003180h |format_id: 2                  |Not an OLE Package
   |          |class name: 'OLE2Link'        |
   |          |data size: 2560               |
---+----------+------------------------------+------------------------------
```

This object can be extracted using rtfobj, which automatically converts the hexadecimal representation to raw bytes.

```
% rtfobj -s 0 3c0a93d05b3d0a9564df63ed6178d54d467263ad6e3a76a9083a43a7e4a9cca5

.. snip ..

Saving file embedded in OLE object #0:
  format_id  = 2
  class name = 'OLE2Link'
  data size  = 2560
  saving to file 3c0a.._object_00003180.bin
```

5.4 Analysing the embedded document

The file type of the embedded document can also be confirmed:

```
% file 3c0a.._object_00003180.bin
3c0a.._object_00003180.bin: Composite Document File V2 Document,
                            Cannot read section info
```

Using OffVis[7] the CDF structure can be inspected. This shows the first directory entry contains an `OLE2Link` object.

The raw representation of the class identifier is at offset 0x450 in the file, which matches the GUID found earlier (`00000300-0000-0000-C000-000000000046`).

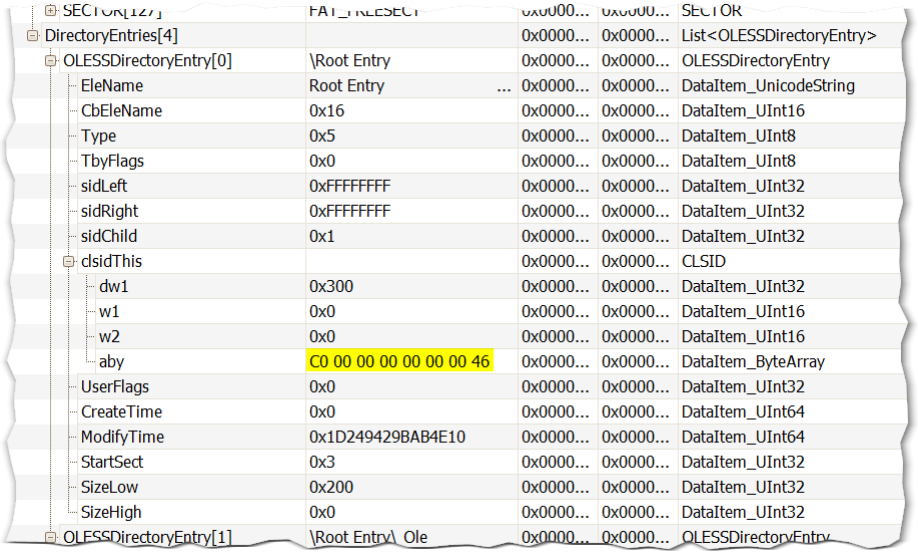
```
00000450: 0003 0000 0000 0000 c000 0000 0000 0046  ...............F
```

Note the byte representation of the GUID is a different order to how it is displayed, this is important when writing signatures.

Data for the second OLE directory entry relates to the `OLE2Link` object and contains a URL:

```
00000800: 0100 0002 0900 0000 0100 0000 0000 0000  ................
00000810: 0000 0000 0000 0000 5c01 0000 e0c9 ea79  ........\......y
00000820: f9ba ce11 8c82 00aa 004b a90b 4401 0000  .........K..D...
00000830: 6800 7400 7400 7000 3a00 2f00 2f00 3200  h.t.t.p.:././.2.
00000840: 3100 3200 2e00 3800 3600 2e00 3100 3100  1.2...8.6...1.1.
00000850: 3500 2e00 3700 3100 2f00 7400 6500 6d00  5...7.1./.t.e.m.
00000860: 7000 6c00 6100 7400 6500 2e00 6400 6f00  p.l.a.t.e...d.o.
00000870: 6300 0000 0000 0000 0000 0000 0000 0000  c...............
```

[7] http://go.microsoft.com/fwlink/?LinkId=158791



SECTOR[127]	FAT_FREESECT	0x0000...	0x0000...	SECTOR
DirectoryEntries[4]		0x0000...	0x0000...	List<OLESSDirectoryEntry>
OLESSDirectoryEntry[0]	\Root Entry	0x0000...	0x0000...	OLESSDirectoryEntry
EleName	Root Entry ...	0x0000...	0x0000...	DataItem_UnicodeString
CbEleName	0x16	0x0000...	0x0000...	DataItem_UInt16
Type	0x5	0x0000...	0x0000...	DataItem_UInt8
TbyFlags	0x0	0x0000...	0x0000...	DataItem_UInt8
sidLeft	0xFFFFFFFF	0x0000...	0x0000...	DataItem_UInt32
sidRight	0xFFFFFFFF	0x0000...	0x0000...	DataItem_UInt32
sidChild	0x1	0x0000...	0x0000...	DataItem_UInt32
clsidThis		0x0000...	0x0000...	CLSID
dw1	0x300	0x0000...	0x0000...	DataItem_UInt32
w1	0x0	0x0000...	0x0000...	DataItem_UInt16
w2	0x0	0x0000...	0x0000...	DataItem_UInt16
aby	C0 00 00 00 00 00 00 46	0x0000...	0x0000...	DataItem_ByteArray
UserFlags	0x0	0x0000...	0x0000...	DataItem_UInt32
CreateTime	0x0	0x0000...	0x0000...	DataItem_UInt64
ModifyTime	0x1D249429BAB4E10	0x0000...	0x0000...	DataItem_UInt64
StartSect	0x3	0x0000...	0x0000...	DataItem_UInt32
SizeLow	0x200	0x0000...	0x0000...	DataItem_UInt32
SizeHigh	0x0	0x0000...	0x0000...	DataItem_UInt32
OLESSDirectoryEntry[1]	\Root Entry\ Ole	0x0000...	0x0000...	OLESSDirectoryEntry

Figure 2: OffVis output

5.5 Downloaded file

Analysis of the fetched URL confirms that it is served with a `Content-Type` of `application/hta` as described by both McAfee and FireEye.

```
HTTP request sent, awaiting response...
  HTTP/1.1 200 OK
  Date: Mon, 10 Apr 2017 15:24:32 GMT
  Server: Apache/2.4.6 (CentOS)
  Last-Modified: Mon, 10 Apr 2017 08:46:24 GMT
  ETag: "6b4c-54ccc02975e2f"
  Accept-Ranges: bytes
  Content-Length: 27468
  Keep-Alive: timeout=5, max=100
  Connection: Keep-Alive
  Content-Type: application/hta
Length: 27468 (27K) [application/hta]
```

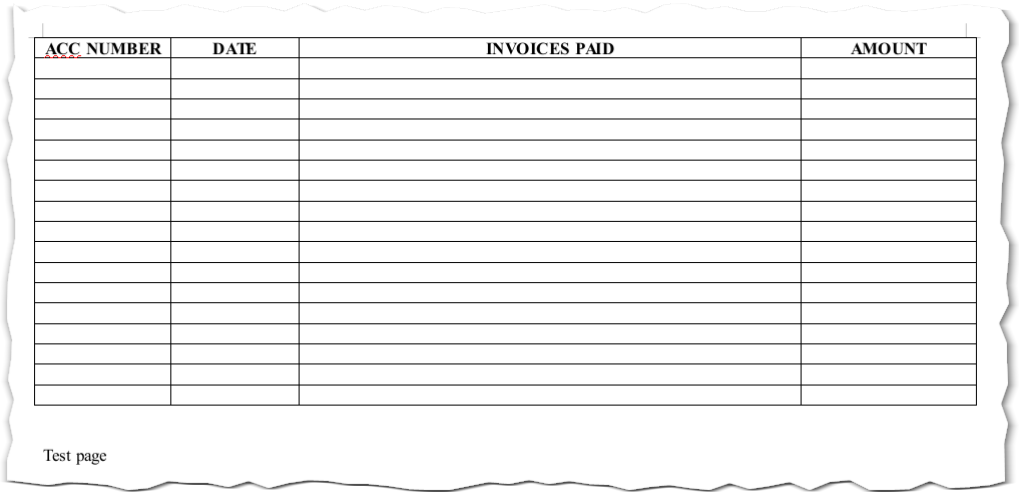
As described by McAfee the file `template.doc` contains a script which uses Powershell to:

- Move the current window to coordinates -2000,-2000.
- Kill `winword.exe`, to close the initial exploit document.
- Download an executable file, in this sample `http://212[.]86[.]115[.]71/sage50.exe` (saved to a start menu run key, named `winword.exe`).
- Download a decoy document, in this sample `http://212[.]86[.]115[.]71/Transactions.doc`.
- Remove all entries from the `\Software\Microsoft\Office\16.0\Word\Resiliency` registry key, to prevent any attempts at file recovery.
- Show the decoy document.

The document `template.doc` was apparently authored by `XMEN` and contains the single word `XMEN`.

5.6 Decoy document

The decoy document `Transactions.doc` is launched and displayed to the user. In this case it contains an empty list of account number and invoices paid.



ACC NUMBER	DATE	INVOICES PAID	AMOUNT

Test page

Figure 3: Decoy document contents

6 Detection & mitigation

This section contains a number of rules which can be used during hunting or incident response to find potential samples.

Note that trivial obfuscation could be applied to many samples to evade the static rules below, we do not suggest they are sufficient to detect every possible exploit document.

6.1 Host based blocking

Several mitigations using RTF file blocking have been published [8][9]. However, these do not completely block all vectors and it is still possible to exploit this issue without using RTF.

A more robust solution is to use Microsoft's AppLocker tool[10] or configure an ActiveX kill bit[11].

6.1.1 AppLocker

AppLocker can be used to block `mshta.exe` from executing. When this is configured correctly the exploit will be halted when it tries to execute the `.hta` file, regardless of which vector or filetype is used.

Please note this will completely block the use of `.hta` files, potentially including critical parts of Windows Update. Therefore we do not currently recommend this without significant further testing.

6.1.2 ActiveX kill bits

The following registry entry will disable the `StdOleLink` (or `OLE2Link`) object in Office.

```
Windows Registry Editor Version 5.00

[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Office\Common\COM Compatibility\
{00000300-0000-0000-C000-000000000046}]
"Compatibility Flags"=dword:00000400
```

[8] https://twitter.com/ryHanson/status/851160174060384256
[9] https://twitter.com/TheMuffinManFPS/status/851592276748808192
[10] https://technet.microsoft.com/en-gb/itpro/windows/keep-secure/applocker-overview
[11] https://technet.microsoft.com/en-us/security/dn535768.aspx

During our testing this was sufficient to block exploitation, however we encourage thorough testing before this is deployed across an organisation to ensure it has no unintended side effects.

6.2 Yara rules

```
rule exploit_ole_stdolelink {
  meta:
    author = "David Cannings"
    description = "StdOleLink, potential 0day in April 2017"

  strings:
    // Parsers will open files without the full 'rtf'
    $header_rtf = "{\\rt" nocase
    $header_office = { D0 CF 11 E0 }
    $header_xml = "<?xml version=" nocase wide ascii

    // Marks of embedded data (reduce FPs)
    // RTF format
    $embedded_object   = "\\object" nocase
    $embedded_objdata  = "\\objdata" nocase
    $embedded_ocx      = "\\objocx" nocase
    $embedded_objclass = "\\objclass" nocase
    $embedded_oleclass = "\\oleclsid" nocase

    // XML Office documents
    $embedded_axocx      = "<ax:ocx"  nocase wide ascii
    $embedded_axclassid  = "ax:classid"  nocase wide ascii

    // OLE format
    $embedded_root_entry = "Root Entry" wide
    $embedded_comp_obj   = "Comp Obj" wide
    $embedded_obj_info   = "Obj Info" wide
    $embedded_ole10      = "Ole10Native" wide

    $data0 = "00000300-0000-0000-C000-000000000046" nocase wide ascii
    $data1 = { 0003000000000000C000000000000046 }
    $data2 = "OLE2Link" nocase wide ascii
    $data3 = "4f4c45324c696e6b" nocase wide ascii
    $data4 = "StdOleLink" nocase wide ascii
    $data5 = "5374644f6c654c696e6b" nocase wide ascii

  condition:
    // Mandatory header plus sign of embedding, then any of the others
    for any of ($header*) : ( @ == 0 ) and 1 of ($embedded*)
        and (1 of ($data*))
}
```

6.3 Network

Unusual activity can be identified by searching for `application/hta` in the `Content-Type` of HTTP responses. These files are rarely used and could typically be blocked, as they contain active scripting content.

Any suspicious responses can be triaged by inspecting the response data for RTF (or potentially CDF) content. Note that in the sample analysed a number of newline characters were inserted before the RTF header, presumably to avoid basic network detection.

The following Suricata rule will detect potential RTF content being returned with a content type of `application/hta`.

```
alert http $EXTERNAL_NET any -> $HOME_NET any (msg:"Possible Office 0-day, RTF
content with HTA header"; flow:established,from_server;
content:"Content-Type|3a 20|application/hta|0d 0a|"; http_header; file_data;
content:"|7b 5c 72 74|"; within: 128; classtype:trojan-activity; sid:1; rev:1;)
```

Emerging Threats rules 2024192 and 2024193 will also detect related activity.

7 Changes

Version	Changes
0.1	First public release. CVE number and acknowledgements are not available at this time.
0.2	Minor corrections to typos.
0.3	Added AppLocker information and Suricata rule.

8 Contact details

To contact the authors with questions, suggestions or corrections please use the email address david.cannings@nccgroup.trust (GPG key `0x06211f5797f3b650`).

For all other queries about NCC Group please email response@nccgroup.trust who will direct your query appropriately.