RTF Abuse:
Exploitation, Evasion and Counter Measures

Devon Greene
Member of Ixia’s Application and Threat Intelligence (ATI) Team

Focus on Malware Analysis, Exploit Development and Product Development.

<3 CTFTime.org & Vulnhub Challenges

Opinions are my own, not Ixia’s

I DON’T ALWAYS GET FLASH BANGED, BUT WHEN I DO, IT’S BECAUSE I THREW IT.
How I Met RTF

- Working on a strike.
- Created 6 new evasion profiles
- … in Ruby (Not better than Python)
Key Points

➢ Identify malicious RTF documents
➢ Enhance detection capabilities
➢ System hardening techniques
Key Points

- Obfuscation Techniques
- Vulnerability Discovery Approaches
- Exploitation Techniques
To Understand RTF…

You Must RTFM!
<table>
<thead>
<tr>
<th>Interesting Features</th>
<th>Features You Expect</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ability to Query DBs / Flat Files</td>
<td>Embedded Fonts</td>
</tr>
<tr>
<td>Hyperlinks</td>
<td>Pictures</td>
</tr>
<tr>
<td>Object Linking and Embedding</td>
<td>Hex / Unicode Support</td>
</tr>
<tr>
<td>Document Variables</td>
<td>Much moar!</td>
</tr>
<tr>
<td>Functions and Parameters (limited)</td>
<td></td>
</tr>
</tbody>
</table>
Hyper Text Markup Language (HTML)

```html
1  <html>
2      <b>Hello World!</b>
3      <img src="kitty.png"/>
4  </html>
```

Rich Text Format (RTF)

```rtf
1  {\rtf
2      \b Hello World!
3      {\/*/pict<metatags>\<encoded_data>}
4  }
```
Let’s Play

➤ Build an RTF doc from scratch

➤ Use an RTF doc to perform a DB query

➤ Quick look at built-in functions
Exploitation

IT'S MAGIC.

I AIN'T GOTTA EXPLAIN SHIT.
\*
\*\Exploitation

Attack Paths

- N-Day Vulnerabilities (Automagic)
- Embedded Font Vulnerabilities
- Insecure Library Loading Vulns
- Packager Objects (CVE-Free)

Death From Above!
Historically Powerful

- DuQu Malware leveraged 0-Day TTF Exploit (CVE-2011-3402)
- Font engine lives in the Windows Kernel
- Downside: bloats the file quite a bit.
How It Works…

If `SafeDllSearchMode` is enabled, the search order is as follows:

1. The directory from which the application loaded.
2. The system directory. Use the `GetSystemDirectory` function to get the path of this directory.
3. The 16-bit system directory. There is no function that obtains the path of this directory, but it is searched.
4. The Windows directory. Use the `GetWindowsDirectory` function to get the path of this directory.
5. The current directory.
6. The directories that are listed in the PATH environment variable. Note that this does not include the per-application path specified by the App Paths registry key. The App Paths key is not used when computing the DLL search path.
PACKAGER EXT CHECK PROCESS

- packager.dll
- shlwapi.dll

GetPathExtension

Hard Coded Ext. Check

Registry Based Check

FTA_AlwaysUnsafeBit (0x00020000)

Policy based check

<table>
<thead>
<tr>
<th>Offset</th>
<th>Extension</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x01D98</td>
<td>.exe</td>
</tr>
<tr>
<td>0x1D9C</td>
<td>.com</td>
</tr>
<tr>
<td>0x1DA0</td>
<td>.bat</td>
</tr>
<tr>
<td>0x1DA4</td>
<td>.lnk</td>
</tr>
<tr>
<td>0x1DA8</td>
<td>.cmd</td>
</tr>
<tr>
<td>0x1DAC</td>
<td>.plf</td>
</tr>
<tr>
<td>0x1DB0</td>
<td>.scr</td>
</tr>
<tr>
<td>0x1DB4</td>
<td>.js</td>
</tr>
<tr>
<td>0x1DB8</td>
<td>.jse</td>
</tr>
<tr>
<td>0x1DBC</td>
<td>.vbs</td>
</tr>
<tr>
<td>0x1DCC</td>
<td>.vbe</td>
</tr>
<tr>
<td>0x1D08</td>
<td>.wsh</td>
</tr>
<tr>
<td>0x1D0C</td>
<td>.sct</td>
</tr>
<tr>
<td>0x1D00</td>
<td>.vb</td>
</tr>
<tr>
<td>0x1D08</td>
<td>.wsc</td>
</tr>
<tr>
<td>0x1D0C</td>
<td>.wsf</td>
</tr>
<tr>
<td>0x1D00</td>
<td>.wmz</td>
</tr>
</tbody>
</table>

Aero::Safe::Dangerous
Forging Images

- Embed file in word document
- Save as RTF
- Copy/Paste `\pict` object

Interesting Packager Quirks

- Place any file you want in a users `%temp%` directory
- Seriously… any file.
- Email Providers Don’t Care
Embedded Objects

- Few Fun Techniques
  - Take advantage of `%temp%`
  - Take advantage of local env

- Compatible with other doc types 😊
Embedded Font File

- Noted earlier, bypasses Packager Checks.
- Warning: VM gonna go BOOM!
- Note: this is a packaged font file, not an `embfont` tag.
Vulnerability Discovery

COUNTER TERRORISTS KNOW THE BOMB CODE FOR 16 YEARS

TERRORISTS STILL HAVEN'T CHANGED IT
Fuzzing

Mutation Based

I DON'T HAVE A PLAN....
I JUST DO THINGS

Generation Based

Researcher defines how the input should be formed.
Search for “MUST”

for the numeric parameter. An RTF parser must allow for up to 10 digits optionally preceded by a minus sign. If the delimiter is a space, it is discarded, that is, it’s not included in subsequent processing.

this group. All RTF readers must recognize all destinations defined in the 1987 RTF Specification. The reader may skip past the whole header tables. A property must be defined before being referenced.

braces. The \rtfN control word must follow the opening brace. is restored. The reader must keep a stack of counts seen and use the most recent one to skip the

\listidN Each list must have a unique list ID that should be randomly generated. N is a long integer. The list ID cannot be between −1 and −5.
Built a thorough data model of the RTF specification.

> Distributed fuzzing amongst 6 machines

> 1 Cycle was approximately 2,189,235 fuzzing iterations

> 500+ crashes // 6 unique
IF THE CREATOR HAS ITS FLAWS
WHAT DO THE IMITATORS HAVE?

Other Targets
>
- Open Office
- Corel Word Perfect
- Text Wrangler
- Cloud-based document services
- MS Office on other Platforms
Obfuscation Techniques

Sniper

Camouflage

FAIL
Jan.01 – Jun.30

- 725 .doc exts
- 100 .rtf exts
- < 10 .docx exts
- 300 other exts

**Generation Based**

<table>
<thead>
<tr>
<th>Extension</th>
<th>MS Word 2010</th>
<th>MS Word 2016</th>
</tr>
</thead>
<tbody>
<tr>
<td>DOC</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>DOCHTML</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>DOT</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>DOTHTML</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>WBK</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>WIZ</td>
<td>Y</td>
<td>Y</td>
</tr>
</tbody>
</table>
Magic File Tampering

- MS Word respects `{rt` as a minimum magic file header.
- MS Word pad requires `{rtf#`

Mixed Case

- Utilized anywhere `#PCDATA` is defined.
- Useful in bypassing static signatures
Encoding Contrast
- URL Encoding
  A = %41
- Double URL Encoding
  A = %2541

Unicode Escaping
- A = 0x41
- A = \u0041

Hex Escaping
- A = 0x41
- A = \'41
evasions
Bin Substitution

- Works in MS Office Only
- Does not work in MS Wordpad

Whitespace

- Simple and Effective
- Chunk up your payloads and other shady stuff 😊

\r \n \t \s

Syntax:
\bin# <ASCII>
\*\*\*evasions

Fictitious Control Words

> Detection Slayer
> Double Edged Sword
> Some AV heuristic checks will catch this.

Syntax:
{\*\HELLO WORLD\*}
evasions
By applying evasion techniques, can we throw off RTFScan.exe’s analysis capabilities?
Bypassing AVs?

By applying evasion techniques, can we make a bad guys malicious document harder to detect?
Counter Measures
Focus On

- File Extensions
  IE: .doc

- Malformed file headers
  IE: {\rtvnp

- Embedded objects
  IE: \objdata

- Unknown RTF tags
  IE: \*\HaiMom

Special Cases

- Non required params
  IE: \objclass

- Encoding Techniques
  IE: \u0041

- Mixed Cases
  IE: \objclass name
Focus On

- This… obvious… Tag…
- Generator tag
- `\*\generator MsftEdit`

Obvious is Obvious
3 Tips

> Set Office Killbit on the packager clsid

> Update Executable Extensions

> Change .rtf association back to Wordpad 😊
RTF Analysis Tools

- Didier Steven’s rtfdump
- Declage’s rtfobj
- PhishMe psparser.py
- RTFScan.exe

Fool Proof?

I changed all my passwords to "incorrect".

So whenever I forget, it will tell me "your password is incorrect."
Recap

- Update your magic file header for RTF
- Scrutinize `\*\generator tags`
- Focus on required parameters first
- Lookout for `.WIZ` and `.WBK`
- Disable Packager Objects

Punch On!

THIS IS WHAT THE OTHER TEAMS ARE GOING TO DO

WHEN THE BLUE TEAM WINS NEXT WEEK
Recap

➢ Take advantage of obfuscation techniques!
➢ Trade warning signs by using packager objects.
➢ Save as other doc-types when necessary!
➢ Fuzz the hell out of RTF!