5 A Flash PDF Polyglot

by Alex Inführ

5.1 PDF and SWF Reunited

I had the idea of creating a nice little file, one which is both a valid PDF and a valid Flash file. Such a polyglot can cause a lot of trouble, because they can smuggle active content like Flash in a harmless file type, PDF.[4] The PDF format is a really good container format, because the Adobe PDF parser is not very strict. The PDF header "%PDF-" does not have to be at offset `0`; the parser will search the first `1017` bytes for the header. Recently, however, Adobe decided to stop supporting PDF files that start either with CWS or FWS at offset 0. Both are possible headers for a Flash file. This should make it harder to create such polyglots.

5.2 Main File Structure

Unlike PDF, Flash files always need their header at offset 0. It is not possible to insert any data before it. To fulfill this requirement, we need to find a way to bypass Adobe's prohibition of Flash headers. The next step requires the PDF header to be embedded in the first 1,017 bytes without destroying the Flash file. If we meet all these requirements, we will be able to append the rest of the PDF data at the end of the file.

5.3 Bypassing the Header Restriction

The bypass was rather simple, all you have to do is open the SWF file format specification to page 27.

The specification mentions three possible headers: "`FWS`," "`CWS`" and "`ZWS`". The FWS is used for uncompressed Flash files, CWS for ZLIB compressed files and ZWS for LZMA compressed files. Maybe you've guessed it already, but Adobe forgot to block the ZWS header. For now the file structure looks like this:

```
>>> structure[0:3]
ZWS
>>> structure[4:]
[...Flash data...][...PDF data...]
```

Let's move on to the PDF header.

5.4 The Missing PDF Header

The last thing missing is the PDF header. Let's look in the Flash specification for a place. In the header the length of the uncompressed Flash file is stored at offset `0x04`, requiring four bytes. It seems to be useless, as no Flash parser seems to use this field! This means we can overwrite it with the PDF header, but we are missing one byte. The SWF specification defines at offset `0x03` the Flash version. Combined with the following four-byte length field, we have a perfect place for the PDF header! Our header structure looks like this.

```
>>> structure[0:3]
ZWS
>>> structure[3:8]
%PDF-
>>> structure[8:]
[...Flash data...][...PDF data...]
```

This is all it requires, but there is more!

[4] As harmless as PDF can be, at least!

5.5 The Madness

For unknown reasons the Flash file needs to be bigger than a certain size. I hard coded this size in my script. If the Flash file is too small, the created polyglot won't be rendered by the Adobe PDF reader, which makes no sense. I tested the PDF/Flash polyglot across a number of different browsers, and the results are very interesting. Please test it with your own systems.

- Windows 8 32 Bit:
  - IE 11: PDF parsed, Flash not parsed
  - Chrome: PDF parsed, Flash not parsed
  - Firefox: PDF not parsed, Flash parsed
  - Adobe Reader 11.0.07: PDF parsed
- Windows 7 64 Bit:
  - IE 11: PDF parsed, Flash not parsed
  - Chrome: PDF parsed, Flash parsed
  - Firefox: PDF not parsed, Flash parsed
  - Opera: PDF parsed, Flash parsed
  - Adobe Reader 11.0.07: PDF parsed
- Windows 7 Enterprise 32 Bit:
  - IE 11: PDF parsed, Flash parsed
  - Chrome: PDF parsed, Flash not parsed
  - Firefox: PDF not parsed, Flash parsed
  - Adobe Reader 11.0.07: PDF parsed

As you can see, IE and Chrome are not consistent between different operating systems, which seems really odd. But I have one little trick left!

5.6 Chrome Flash Player Crash!

While playing with the values of the Flash header I came across a crash in the 64 bit version of Chrome's Flash Player. At offset `0x0f` and `0x10` a part of the dictionary size is stored. This is used in the LZMA compression algorithm. Changing these to a high value like `0xBEEF` will trigger a crash. Extending this crash to an exploit, or determining that it isn't exploitable, is left as an exercise for the reader.

```
>>> structure[0x0f:0x11]
? (0xbeef)
```