8 This OS is also a PDF

by Ange Albertini

A careful reader may have noticed that a bootable OS image was hidden in the last issue of PoC || GTFO, as one of the files in its dual PDF/ZIP structure (if you haven't, download and extract it now!). This time, though, let's hide it in plain sight. You will find by running 'qemu-system-i386 -fda pocorgtfo02.pdf' that the PDF file you are reading is also a bootable disk image.

8.1 Requirements

To combine two file types, we first need to list the requirements of each format and then produce a single file that meets both sets of requirements with no conflicts.

What makes a bootable disk image? An X86 machine begins booting by copying the first 512 byte sector, the Master Boot Record, into RAM and executing it. The requirements for a functional MBR are simple:

- 16 bit x86 code starts at offset `00`.
- It will be executing at the `0000:7c00` address in RAM.
- It must be 512 bytes long, ending with the signature `55, AA`
- Labels and primary partition tables are optional, but can go within this sector.
- It must contain code that finds and loads into RAM the code for the next boot stage (such as an OS loader).

PDF files are a mixture of text and binary fragments, which are parsed from the start of the file and delimited by words and newlines. The requirements for a valid PDF are also simple and surprisingly flexible:

- It is initially parsed as text.
- The signature "%%PDF-" must be present within the first 1024 bytes. It can be present there twice or more.
- Comment lines begin with '%', which is `25` in hex.
- Binary characters other than CRLF are acceptable in a comment.
- "Multi-line" binary objects or simply larger objects can also be stored in object streams, which are declared like this:

```
<obj number> <revision> obj
<<>>
stream
<stream content>
endstream
endobj
```

8.2 Strategy

In most cases, we can freely prepend anything at the start of the file as long as the above requirements are fulfilled. Luckily, the % comment character is `0x25`, which encodes nicely as an x86 **and** instruction. Thus, the head of the file can be `25FFFF: and ax, 0xffff`, which also starts a PDF comment. We can then add a jump into the next part of the code, which will be stored in a dummy object stream below, and then finish our first line. Adding a PDF signature will prevent any potential problem in case the stream object is too long: it can then contain anything, of any length, as long as it doesn't contain the 'endstream' keyword.

```
; this will encode as '%\xff\xff\xeb\x21', a comment line
and ax, -1
jmp start

%PDF-1.5

999 0 obj
<<>>
stream

code:
...

; put the 55AA signature at the end of the 512 block
times 200h - 2 - ($ - $$) db 0cch
    db 55h, 0aah

endstream
endobj
```

8.3 An Unexpected Challenge

This was almost too easy, but there is a caveat to keep in mind. I'll mention it here to save you the headache when reproducing these results.

This new challenge emerged as I was testing the bootable PDF files with different PDF readers. Since we pre-pend our MBR without altering the contents of the original document, the original's cross-reference table XREF is no longer in sync with the actual file offsets. Technically, this makes the XREF tables corrupted.

Corrupted XREFs are so common that they are usually transparently recovered by all PDF readers, even picky ones such as PDF.JS. However, your pdflatex **may** generate a document based on the optimized PDF 1.5 specification, where the XREF is stored not in cleartext as in PDF 1.4, but rather as a separate, compressed object. This configuration choice is made for the user by the TeX distribution, so even a freshly updated pdflatex install may generate PDF 1.4 documents.

Even when compressed, corrupted XREFs are recovered by some readers, such as GS and Sumatra. Unfortunately, Foxit, Adobe, Firefox, Chrome, and Poppler-based readers—such as Evince and Okular—would reject such a document. Although rejecting corrupted documents out of hand is the best strategy, even Pastor Laphroaig would be pretty pissed if folks couldn't read his epistles because of this.

A simple and elegant workaround that achieves 100% reader compatibility with our MBR PDF is to make sure that, even if your pdflatex distribution generates a 1.5 format document, it doesn't compress the XREF. This is easily done by adding the following command to your LaTeX source.

```
\pdfobjcompresslevel=0
```

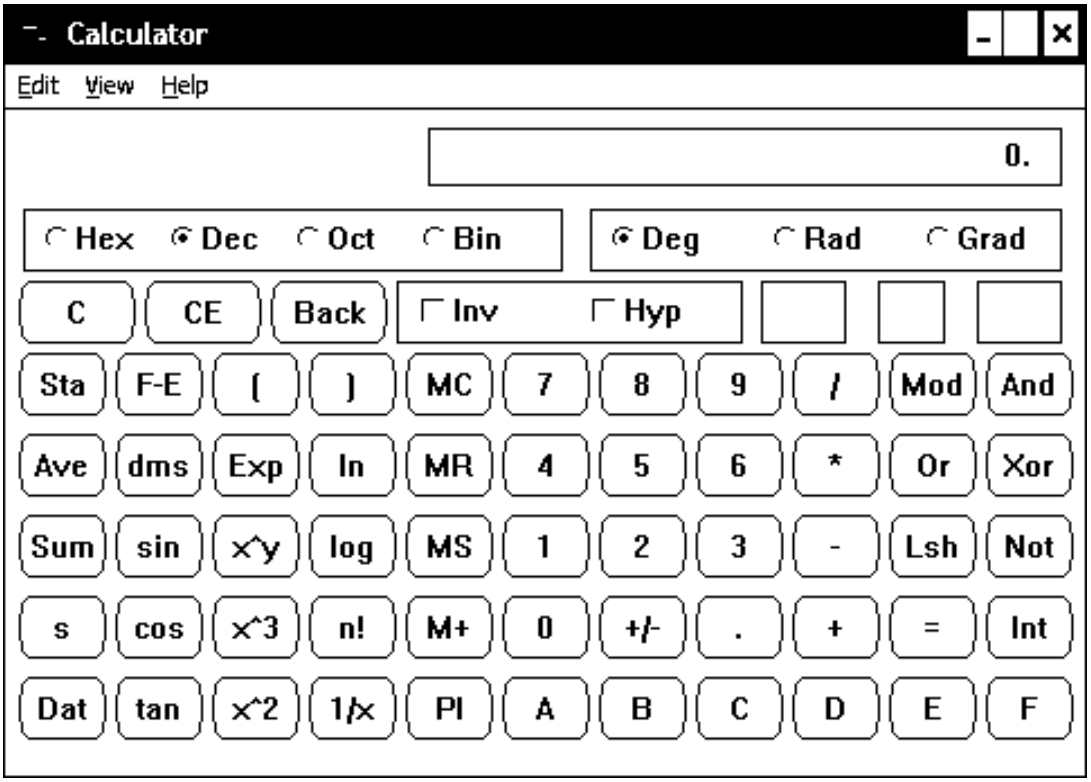
This command will cause pdflatex to store non-objects uncompressed while still taking advantage of other 1.5 features such as reducing document bloat. I should add that, although the fix looks trivial, finding the real cause and the most elegant solution was a challenge.

— — — —

Enjoy booting this PDF, and be sure to share copies—both electronic and paper—so that your neighbors can enjoy it as well!

```
00000000  25 ff ff e9 fc 00 0a 25  50 44 46 2d 31 2e 35 0a  |%......%PDF-1.5.|
00000010  39 39 39 39 20 30 20 6f  62 6a 0a 3c 3c 3e 3e 0a  |9999 0 obj.<<>>.|
00000020  73 74 72 65 61 6d 0a 0a  50 6f 43 20 6f 72 20 47  |stream..PoC or G|
00000030  54 46 4f 20 49 73 73 75  65 20 30 78 30 32 0a 0d  |TFO Issue 0x02..|
00000040  62 79 20 52 74 2e 20 52  76 64 2e 20 50 61 73 74  |by Rt. Rvd. Past|
00000050  6f 72 20 4d 61 6e 75 6c  20 4c 61 70 68 72 6f 61  |or Manul Laphroa|
00000060  69 67 20 61 6e 64 20 46  72 69 65 6e 64 73 0a 0a  |ig and Friends..|
00000070  0d 00 59 6f 75 20 68 61  76 65 20 62 65 65 6e 20  |..You have been |
00000080  65 61 74 65 6e 20 62 79  20 61 20 67 72 75 65 2e  |eaten by a grue.|
00000090  20 20 53 6f 72 72 79 2e  0a 0d 54 72 79 20 74 68  |  Sorry...Try th|
000000a0  69 73 3a 20 71 65 6d 75  2d 73 79 73 74 65 6d 2d  |is: qemu-system-|
000000b0  69 33 38 36 20 2d 66 64  61 20 70 6f 63 6f 72 67  |i386 -fda pocorg|
000000c0  74 66 6f 30 32 2e 70 64  66 0a 0d 00 31 29 20 52  |tfo02.pdf...1) R|
000000d0  65 61 64 69 6e 67 20 6b  65 72 6e 65 6c 20 66 72  |eading kernel fr|
000000e0  6f 6d 20 64 69 73 6b 2e  0a 0d 00 32 29 20 45 78  |om disk....2) Ex|
000000f0  65 63 75 74 69 6e 67 20  6b 65 72 6e 65 6c 2e 0a  |ecuting kernel..|
00000100  0d 00 be 27 7c e8 3e 00  31 c0 8e d8 30 d2 cd 13  |...'|.>.1...0...|
00000110  0f 82 97 00 be cc 7c e8  2c 00 b8 e0 07 8e c0 31  |......|.,......1|
00000120  db b8 10 02 b5 00 b1 02  b6 00 b2 00 cd 13 72 7b  |..............r{|
00000130  b8 00 7e 89 c6 e8 38 00  be eb 7c e8 08 00 ea 00  |..~...8...|.....|
00000140  00 e0 07 e8 65 00 ac 3c  00 74 06 b4 0e cd 10 eb  |....e..<.t......|
00000150  f5 c3 89 c3 c1 e8 0c e8  39 00 89 d8 c1 e8 08 e8  |........9.......|
00000160  31 00 89 d8 c1 e8 04 e8  29 00 89 d8 e8 24 00 c3  |1.......)....$..|
00000170  31 c9 ad e8 dc ff e8 2c  00 83 c1 02 81 f9 00 02  |1......,........|
00000180  75 f0 c3 30 31 32 33 34  35 36 37 38 39 41 42 43  |u..0123456789ABC|
00000190  44 45 46 50 56 83 e0 0f  05 83 7d 89 c6 ac b4 0e  |DEFPV.....}.....|
000001a0  cd 10 5e 58 c3 b8 20 0e  cd 10 c3 be 72 7c e8 95  |..^X.. .....r|..|
000001b0  ff eb fe ea 00 00 ff ff  cc cc cc cc cc cc cc cc  |................|
000001c0  cc cc cc cc cc cc cc cc  cc cc cc cc cc cc cc cc  |................|
000001d0  cc cc cc cc cc cc cc cc  cc cc cc cc cc cc cc cc  |................|
000001e0  cc cc cc cc cc cc cc cc  cc cc cc cc cc cc cc cc  |................|
000001f0  cc cc cc cc cc cc cc cc  cc cc cc cc cc cc 55 aa  |..............U.|
```

Hey kids! Can you color the bytes of this MBR to indicate what's going on?



CALC.EXE||GTFO