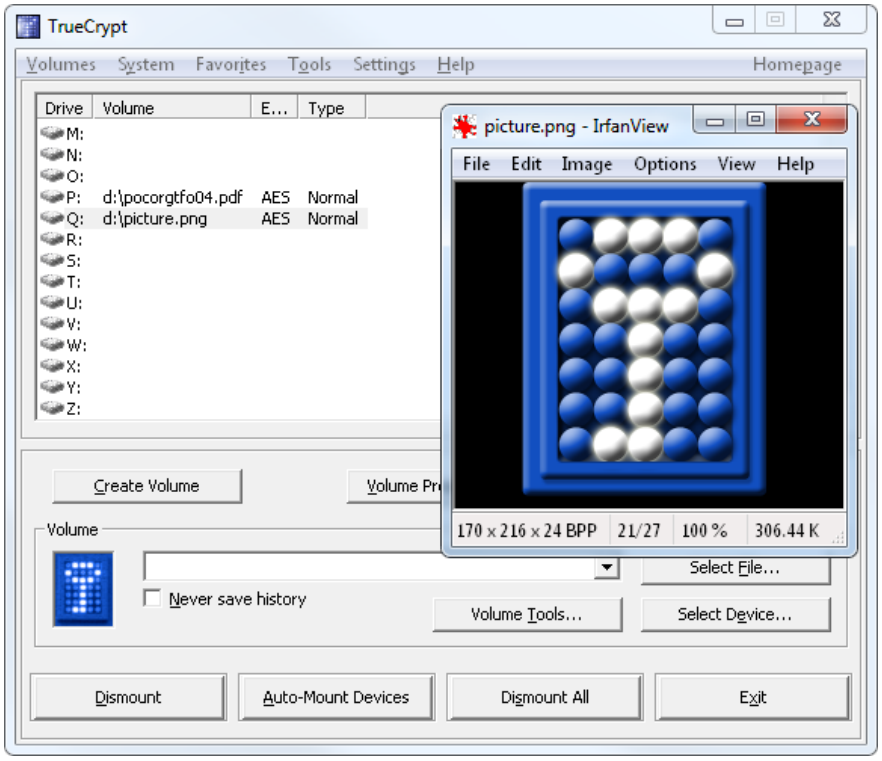
11 This Encrypted Volume is also a PDF; or, A Polyglot Trick for Bypassing TrueCrypt Volume Detection

by Ange Albertini



In this article I will show you a nifty way to make a PDF that is also a valid TrueCrypt encrypted volume. This *Truecryption* trick draws on *Angecryption* from PoC||GTFO 03:11, so if you missed it you can go back in PoC-time now or later, and enjoy even more common file format schizophrenia!

11.1 What is TrueCrypt?

If you open a TrueCrypt container in a hex editor, you'll see that, unlike many binary formats, it looks like entirely random bytes. It does in fact have a header that starts with the magic signature string `TRUE` at file offset `0x40`, but this header is stored encrypted, and thus you can't spot it offhand. To decrypt the header, one needs both the correct password and the hopefully random salt that is stored in the bytes 0-63, just before the encrypted header.

So, a TrueCrypt file starts with 64 bytes of randomness, used as salt to derive the *header key* from the password. This key is used to decrypt the header. If the result of the decryption starts with `TRUE`, then it means the password was correct, and the now decrypted header is parsed further. In particular, this header contains *volume keys*, which are, in turn, used to encrypt/decrypt the blocks and sectors of the encrypted drive.

Importantly, the salt itself is only used to decrypt the header. This is to defend against rainbow table-like precomputing attacks.

Let's start with an existing TrueCrypt volume file for which we know the password. We are not going to change its actual contents or the header's plaintext, but we are going to re-encrypt the header so that the whole becomes a valid PDF file while remaining a valid TrueCrypt volume as well.

Because the salt is supposed to be random, it can be anything we choose. In particular, it can double as any other file format's header. Using the original salt and password, we can decrypt the header. By choosing a new salt—which starts with the header of our new binary target—we derive new keys, and can thus re-encrypt the header to match our new salt.

So, our new file contains the new salt, the re-encrypted header, and the original data sectors of the TrueCrypt container. But where will the new PDF binary content go?

For merging in the new content, we are going to use the trick that the readers of *Angecryption*, PoC‖GTFO 03:11, must have guessed already. As we showed there, in many binary formats such as PDF, PNG, etc., it is possible to reserve a big chunk of space filled with dummy data right after the format's header, and have the binary format's interpreters simply skip over that chunk. This is exactly what we are going to do: all of the TrueCrypt volume data would go into the dummy chunk, followed by the new binary content.

If we want a valid binary file to be a TrueCrypt polyglot, we must fit its header and the declaration for the dummy chunk within 64 bytes, the size of the salt. For *Angecryption*, we managed with only 16 bytes to play with, so having 64 bytes almost feels like sinful and exuberant waste.

11.2 An elegant PDF integration

So far, our PDF/TrueCrypt polyglot looks like no contest. To add a bit of challenge, let's make it with standard PDF-making tools alone. We'll ask `pdflatex(1)` nicely to include the TrueCrypt volume into our polyglot.

Specifically, we'll create a dummy stream object directly inside the document, using the following `pdflatex` commands:

```
\begingroup
  \pdfcompresslevel=0\relax
    \immediate\pdfobj stream
        file {pocorgtfo/truecryption/volume}
\endgroup
```

The bytes between the start of the resulting PDF file and our object that contains the TrueCrypt container will depend on the PDF version and its corresponding structure. Luckily, the size of this PDF head-matter data is typically around `0x20`, well below `0x40`. Plenty of legroom on this polyglot flight!

So our PDF will start with its usual header, followed by this standard stream object we created to play the role of a dummy buffer for the TrueCrypt data. We now need to readjust the contents of this buffer so that the encrypted TrueCrypt header matches its salt, which contains the PDF header, and we then get a standard PDF that is also a TrueCrypt container.

11.3 Conclusion

This technique can naturally be applied to any other file format where we can fit the header and a dummy space allocation within its first 64 bytes, the size of TrueCrypt's initial salt.

Moreover, inserting your encrypted volume into a valid file—while keeping it usable—also has the benefit of putting it under the radar of typical TrueCrypt detection heuristics. These heuristics rely on encrypted TrueCrypt volumes having a round file size, uniformly high entropy, and no known header present. Our method breaks all of these heuristics, and, on top of that, leaves the original document perfectly valid and plausibly deniable.[20]

[20] *Of course, this advice is legally worth exactly what you paid for it, and likely less. No warranty intended or implied, void where prohibited by law, etc., etc., etc. Not endorsed by any lawyers real, imaginary, or played-on-TV, but may be considered "digital cyber-bullets" by some. You may be called a merchant of digital cyber-polyglot death, too—you have been warned!* –PML