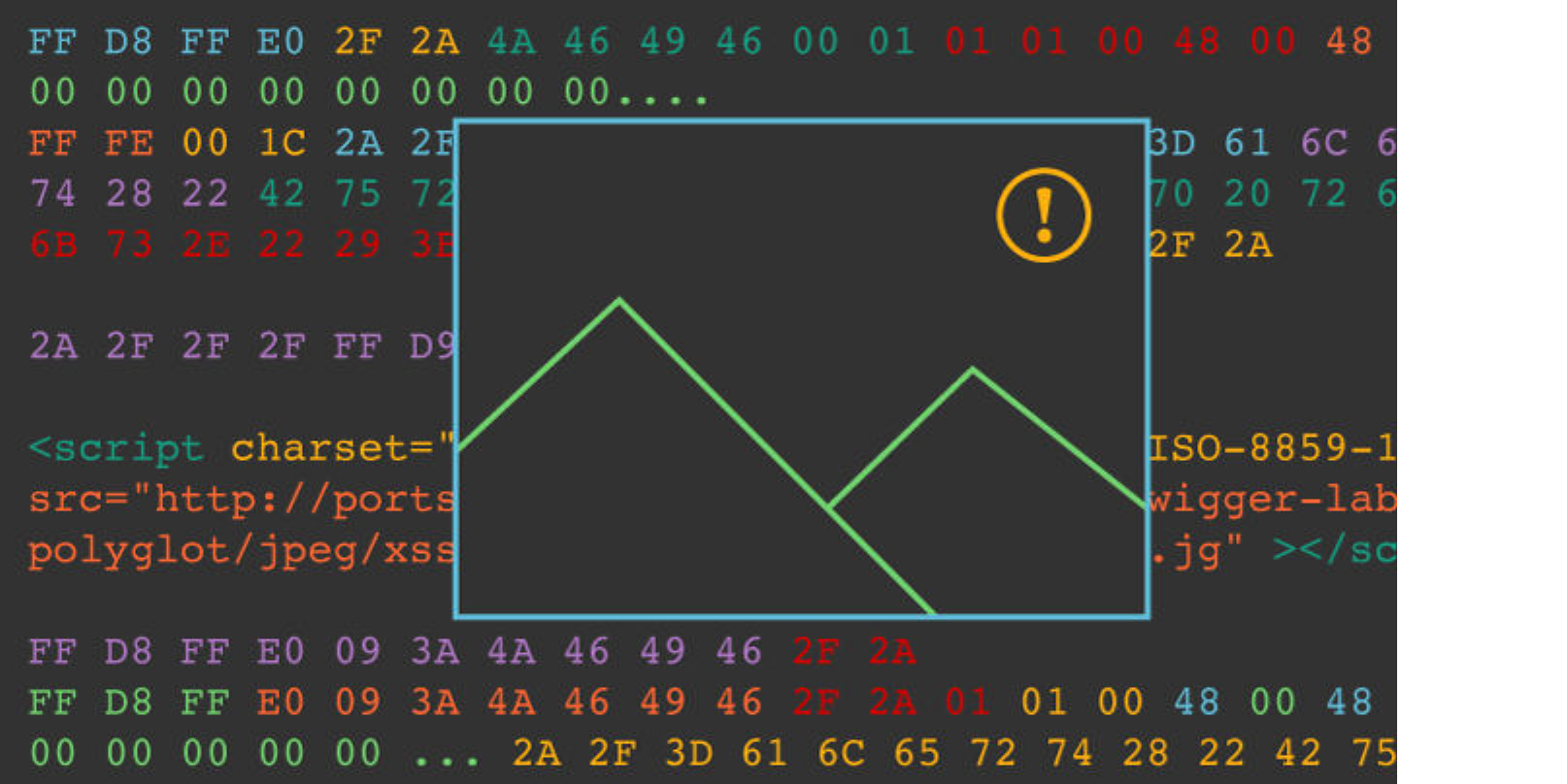
Bypassing CSP using polyglot



JPEGs

James challenged me to see if it was possible to create a polyglot JavaScript/JPEG. Doing so would allow me to bypass CSP on almost any website that hosts user- uploaded images on the same domain. I gleefully took up the challenge and begun dissecting the format. The first four bytes are a valid non-ASCII JavaScript

variable 0xFF 0xD8 0xFF 0xE0. Then the next two bytes specify the length of the JPEG header. If we make that length of the header 0x2F2A using the bytes 0x2F 0x2A as you might guess we have a non-ASCII variable followed by a multi-line JavaScript comment. We then have to pad out the JPEG header to the length of 0x2F2A with nulls. Here's what it looks like:

FF D8 FF E0 2F 2A 4A 46 49 46 00 01 01 01 00 48 00 48 00 00 00 00 00 00 00

00 00 00....

Inside a JPEG comment we can close the JavaScript comment and create an assignment for our non-ASCII JavaScript variable followed by our payload, then create another multi-line comment at the end of the JPEG comment.

FF FE 00 1C 2A 2F 3D 61 6C 65 72 74 28 22 42 75 72 70 20 72 6F 63 6B 73 2E

22 29 3B 2F 2A

0xFF 0xFE is the comment header 0x00 0x1C specifies the length of the comment then the rest is our JavaScript payload which is of course \*/=alert("Burp rocks.")/\*

Next we need to close the JavaScript comment, I edited the last four bytes of the image data before the end of image marker. Here's what the end of the file looks like:

2A 2F 2F 2F FF D9

0xFF 0xD9 is the end of image marker. Great so there is our polyglot JPEG, well not quite yet. It works great if you don't specify a charset but on Firefox when using a UTF-8 character set for the document it corrupts our polyglot when included as an script! On MDN it doesn't state that the script supports the charset attribute but it does. So to get the script to work you need to specify the ISO-8859-1 charset on the script tag and it executes fine.

It's worth noting that the polyglot JPEG works on Safari, Firefox, Edge and IE11. Chrome sensibly does not execute the image as JavaScript.

Here is the polyglot JPEG: Polyglot JPEG

The code to execute the image as JavaScript is as follows:

<script charset="ISO-8859-1" [src="http://portswigger](http://portswigger-)-

labs.net/polyglot/jpeg/xss.jpg"></script>

File size restrictions

I attempted to upload this graphic as a phpBB profile picture but it has restrictions in place. There is a 6k file size limit and maximum dimensions of 90x90. I reduced the size of the logo by cropping and thought about how I could reduce the JPEG data. In the JPEG header I use /\* which in hex is 0x2F and 0x2A, combined 0x2F2A which results in a length of 12074 which is a lot of padding and will result in a graphic far too big to fit as a profile picture. Looking at the ASCII table I tried to find a combination of characters

that would be valid JavaScript and reduce the amount of padding required in the JPEG

header whilst still being recognised as a valid JPEG file.

The smallest starting byte I could find was 0x9 (a tab character) followed by 0x3A (a colon) which results in a combined hex value of 0x093A (2362) that shaves a lot of bytes from our file and creates a valid non-ASCII JavaScript label statement, followed by a variable using the JFIF identifier. Then I place a forward slash 0x2F instead of the NULL character at the end of the JFIF identifier and an asterisk as the version number. Here's what the hex looks like:

FF D8 FF E0 09 3A 4A 46 49 46 2F 2A

Now we continue the rest of the JPEG header then pad with NULLs and inject our

JavaScript payload:

FF D8 FF E0 09 3A 4A 46 49 46 2F 2A 01 01 00 48 00 48 00 00 00 00 00 00 00

... (padding more nulls) 2A 2F 3D 61 6C 65 72 74 28 22 42 75 72 70 20 72 6F

63 6B 73 2E 22 29 3B 2F 2A

Here is the smaller graphic:

Polyglot JPEG smaller

Impact

If you allow users to upload JPEGs, these uploads are on the same domain as your app, and your CSP allows script from "self", you can bypass the CSP using a polyglot JPEG by injecting a script and pointing it to that image.

Conclusion

In conclusion if you allow JPEG uploads on your site or indeed any type of file, it's worth placing these assets on a separate domain. When validating a JPEG, you should

rewrite the JPEG header to ensure no code is sneaked in there and remove all JPEG comments. Obviously it's also essential that your CSP does not whitelist your image assets domain for script.

This post wouldn't be possible without the excellent work of Ange Albertini. I used his JPEG format graphicextensively to create the polygot JPEG. Jasvir Nagra also inspired me with his blog post about polyglot GIFs.

PoC