GIF/Javascript Polyglots: Abusing GIFs, tags, and MIME types for evil

polyglot , malware

---

fraq #1 January 25, 2018, 12:05am

*Note: This is a repost from my **personal blog***

GIF/Javascript Polyglots: Abusing GIFs, tags, and MIME types for evil

6 minute read

The backstory

Recently I saw a feature on a product I work on where we allowed hotlinking to arbitrary gifs without pulling them in, mangling, and then saving for our own use. Right away I thought, “Well this isn’t wise” and set off to find ways to abuse it. The easiest and most obvious was to link to an image and then swap it out for a less savoury one later. Kid stuff, right? Let’s do some real damage. Spoiler alert: I made a really cool thing but didn’t get to weaponize it the way I wanted to.

After proving that I could swap out images with ease from a server I controlled, I started looking for ways to embed executable into the image, which is how I stumbled upon this idea of polyglots. In this context, a polyglot is something that is valid code in two or more languages. For our use case, we want a gif/javascript polyglot.

Getting down to business

As always, research starts with Google and IRC. I had a feeling that something like this had been done before, so I set about finding papers, posts, or anything really that pointed me to what I was looking for. In short order, I found a couple of good resources that explained the attack and how to craft a malicious gif.

The basic idea is to use ASM and manually assemble a GIF, filling in the required header and fields. By setting the width to a specific value of 10799, when the gif is rendered as a script, this is interpreted as a /* in ASCII, which is the opening of a Javascript comment block. When interpreted as an img, the browser simply renders the image that wide. The content and fields of the gif are contained between the opening and closing comment blocks of the JS portions, and the comment is closed at the end of the gif. The JS to execute is tacked onto the end of the GIF and is executed when interpreted as a script. I have included a link to the original gist at the bottom of this post, but here is an example of one such gif:

```
; a hand-made GIF containing valid JavaScript code
; abusing header to start a JavaScript comment

; inspired by Saumil Shah's Deadly Pixels presentation

; Ange Albertini, BSD Licence 2013

; yamal gifjs.asm -o img.gif

WIDTH equ 10799 ; equivalent to 2f2a, which is '/*' in ASCII, thus start

HEIGTH equ 100 ; just to make it easier to spot

db 'GIF89a'
    dw WIDTH, HEIGTH

db 0 ; GCT
    db -1 ;  background color
    db 0 ; default aspect ratio
    ;db 0fch, 0feh, 0fch
    ;times COLORS db 0, 0, 0

; no need of Graphic Control Extension
 ; db 21h, 0f9h
```

As you can see, we close the comment block at the end and add our own Javascript. When interpreted as a script, the parser skips over all the GIF-related stuff and just worries about the JS at the end.

Compiling

As a product of my own stupidity and failure to follow directions, I mistakenly assumed for a bit that the recommended compiler, yasm, was Windows-only. After far too long fighting YASM and C++ runtime trying to get the stupid thing to work, I noticed that I could just pull the source and compile it on my local machine. Sweet! After that, compiling this was a breeze:

```
$ yasm ./gifjs.asm -o img.gif
```

Execution and Exploitation

Unfortunately, here’s where things get a little sad. In order to actually get this to work, I had to do it under some contrived conditions that, while not impossible, are unlikely (in my opinion).

1. You have to have the gif interpreted with tags as opposed to tags
2. You have to send a misleading MIME type

These two conditions mean that it’s unlikely you will find something in the wild

which you can abuse. You're better off using a server you already control and setting up explotation conditions favorable to yourself.

To get this to run, I wrote the following tiny bit of HTML:

```
This is a test
<img src="img.gif">
<script src="img.gif"></script>
```

As you can see, it just throws up a bit of test, displays the malicious GIF as an image, and then again as a script. If you pop open your browser and go to the file in your local filesystem (say, `/tmp/test.html` for instance), the gif will pop the alert box for you. Fun, right?

Now try uploading it to an image hosting site such as Imgur and sourcing it from there. You'll notice something interesting happens. Or rather, doesn't happen. If you try to run the above HTML but using a direct link to a .gif on Imgur instead, you'll notice that your browser's console likely displays an error saying something to this effect:

```
Refused to execute script from 'https://i.imgur.com/IXGn93f.gif' because
```

Detour: what the heck is a MIME type?

MIME types aren't really anything more complex than a label that gets stuck onto some data to tell the receiving end what type of data it is. This lets the client know how it can handle the data. This is just a label and is built on trust. What do we do with trust? We abuse it. Back to your regularly scheduled programming.

This brings me back to point #2: You need to send a misleading MIME type to convince the browser to execute your file. We already know that Imgur won't allow us to do that, so how do we do it? In my case, I used a simple bit of python.

```
import SimpleHTTPServer
import SocketServer

PORT = 8000

class Handler(SimpleHTTPServer.SimpleHTTPRequestHandler):
    pass

Handler.extensions_map['.gif'] = 'application/octet-stream'

httpd = SocketServer.TCPServer(("", PORT), Handler)

print "serving at port", PORT
httpd.serve_forever()
```

This uses SimpleHTTPServer, which is already in Python's standard libraries, to

serve the contents of the local directory. By default, SimpleHTTPServer will try and give things appropriate MIME types based on extensions, so we add a smal change to tell it to interpret .gif extensions as application/octect-stream, which browsers will execute. If I named that html file as index.html, I can now hit **http://127.0.0.1:8000/index.html** and get our malicious gif served back with a MIME type that it is okay with executing. The result? We run the JS compiled into the GIF.

Conclusion

This is not a new or novel attack. This is also not something I feel is widely exploitable, but it is fairly sneaky and exposes a few areas of trust that we can abuse.

- Browsers do little in the way of actual heuristics when trying to determine file type. At best, they will look at the extension and magic byte to try and determine if the file is what it claims to be.
- This is a valid GIF and a valid bit of JS, so heuristics would have to be more sophisticated to catch something like this.
- Browsers trust MIME types perhaps a bit too much.
- This would be easy to exploit and hard to detect using a site you control, since users wouldn't be able to see the JS that is being executed.
- Then again, they might see that something.js was being executed but might not be able to GET the file to see what is in it.
- Obfuscation level: 3/10.

It's interesting. It's fun. It's simple.

References

https://ajinabraham.com/blog/bypassing-content-security-policy-with-a-jsgif-polyglot

https://gist.githubusercontent.com/ajinabraham/f2a057fb1930f94886a3/raw/62b8e455ad62c42222de9059cd0d20c1a79e2cdb/gifjs.asm

http://blog.portswigger.net/2016/12/bypassing-csp-using-polyglot-jpegs.html

17 Likes

---

ricksanchez #2 January 25, 2018, 8:20am

When I first heard about polyglot files I was totally blown away honestly.
After skimming through PoC||GTFO this feeling came somewhat back, since they managed to pack everything in a 'benign looking pdf'.
Cool stuff **@fraq** !

4 Likes

---

fraq #3 January 25, 2018, 1:50pm

It's a neat attack and I'm looking forward to playing with to find the best way to use it.

2 Likes

---

pry0cc (Leader & Offsec Engineer & Forum Daddy) #4 January 25, 2018, 6:30pm

This is a cool concept.

What about PHP? That seems like there is a lot more surface area for attack on Server side languages, than on client side.

1 Like

---

dostoevsky (Dostoevsky) #5 January 26, 2018, 3:19am

@pry0cc - I agree it has applications when processed by a webapp, but just to clarify my understanding, PHP wouldn't be a viable vector in this instance as the image would need to be processed with PHP in order for it to gain execution in that context. In this case, the browser is actually executing the code as JavaScript, not the web application itself.

Very neat indeed, @fraq I think I need to read it a few more times, but I've been meaning to get a grasp on this very technique for a while. Thanks!

1 Like

---

pry0cc (Leader & Offsec Engineer & Forum Daddy) #6 January 26, 2018, 11:16am

This is my proposal for exploitation:

A badly configured image sharing service.

Upload a legitimate image, with polyglot, with PHP mime type/magic byte.

Request that image. Apache/web server will interpret the file as a PHP file, and execute the PHP code inside the file.

2 Likes

---

fraq #7 January 26, 2018, 3:51pm

@dostoevsky is right about the client vs. server side languages. This attack is tricking the browser into interpreting something as a script which it should not, so PHP/Perl/Python/whatever would be a different animal altogether,

2 Likes

AHMED (Ahmed) #8 January 26, 2018, 10:09pm

hello ,

i was really happy to see something new here, its wonderful.
if i may reply on “Deadly Pixels presentation”

still its a wonderful tech, to execute, how ever it was made originally in 1998 , the usage was is how to hide msgs throw each pixels end.

i hope i could learn new things, and excuse my language or any problems in the text
🙂

---

fraq #9 January 27, 2018, 6:27pm

Hi Ahmed,

Glad to have you as part of our community. As I outlined in the article, which I’m sure you read:

1. I did acknowledge that this attack was not new or novel. In fact, I used those exact words.
2. I provided plenty of references to give you further reading on the topic.
3. As you’re probably aware, browsers change over time. If you *did* read those references, you’d notice that none of them made mention of the fact that MIME types have to be abused in order to execute this.
4. Despite its age, it’s still new information to many people here.
5. Very few (if any) of the references provide any sort of example on exploitation vectors or attempt to speculate on use cases.

I hope you find the article and the community useful and continue your infosec journey along with the rest of us.

1 Like

AHMED (Ahmed) #10 January 27, 2018, 9:46pm

hi fraq ,

i didnt mean that its old, the pixel attack was old, but shah added an excutable rather than use charsets into last pixel.

as for the GIF/JS or mime, it is new to me.

i would love to know more of what you have, you are promising to me. an bored of same attacks people explained.

best regards

---

fraq #11 January 28, 2018, 3:03am

Previous iterations of this attack have involved embedding things like JARs as well (the **GIFAR** attack). Most of these have been patched over several times.

Execute malware by opening steganographic image

---

fraq closed #12 February 24, 2018, 12:05am

This topic was automatically closed after 30 days. New replies are no longer allowed.