
SBD Laboratory - Solutions

Thomas Gingele

2023-10-09

Task 1

```
1 POST /WebGoat/HttpProxies/intercept-request HTTP/1.1
2 Host: localhost:8080
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101
  Firefox/115.0
4 Accept: */ *
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Content-Type: application/x-www-form-urlencoded; charset=UTF-8
8 X-Requested-With: XMLHttpRequest
9 Content-Length: 15
10 Origin: http://localhost:8080
11 Connection: close
12 Referer: http://localhost:8080/WebGoat/start.mvc
13 Cookie: JSESSIONID=e3dHiM5wF8CB2DJW6Sb_K1NAYbCAcl3W8PONY_oD;
  WEBWOLFSESSION=YR6BTRUQH_89HzCbp9q68HiWVQGdYCgWyDWVW7UL
14 Sec-Fetch-Dest: empty
15 Sec-Fetch-Mode: cors
16 Sec-Fetch-Site: same-origin
17
18 changeMe=haxx0r
```

Aufgabe 2

- **Host** -> This is the websites host. In this case `localhost` as the application is hosted locally.
- **User-Agent** -> Contains some basic information about the program that is being used to interact with the website. This may help with displaying the website correctly.
- **Accept** -> This is the media type that the browser will accept in the response.
- **Accept-Language** -> This is the language/s that the browser will accept in the response.
- **Accept-Encoding** -> These are the different types of encoding that the browser will accept in the response.
- **Content-Type** -> Describes what format is used to send data to the server. Only used in POST/PUT requests.
- **Origin** -> Enables Cross-Origin Resource Sharing, allowing a client to access otherwise restricted resources from a different domain then the resource is hosted on.
- **Connection** -> Decides whether the connection with the web server should be held open or be closed.
- **Referer** -> The URL at which the client was located when sending the request.
- **Cookie** -> Contains one or more cookie/s previously set by a **Set-Cookie** header by the server. Cookies have multiple purposes from serving as credentials to preventing basic brute force attacks and more.

- **Sec-Fetch-Dest** -> Can be used to let the server know what the response will be used for. This can help with formatting the response for the expected use case on the server.
- **Sec-Fetch-Mode** -> This header is used to distinguish between different uses for the response, for example whether it is for a user navigating a website or to load an image and so on.
- **Sec-Fetch-Site** -> Through this header, a client can tell the server whether a request is coming from the site itself, from a different site or from a completely user-generated request.

Task 3

WebGoat is using **Javascript** and **Java 17/Maven**.

Exposed Javascript frameworks:

- Backbone.js 1.4.0
- RequireJS 2.3.6

Exposed Javascript Libraries:

- jQuery 3.5.1
- jQuery UI 1.10.4
- Underscore.js version unknown

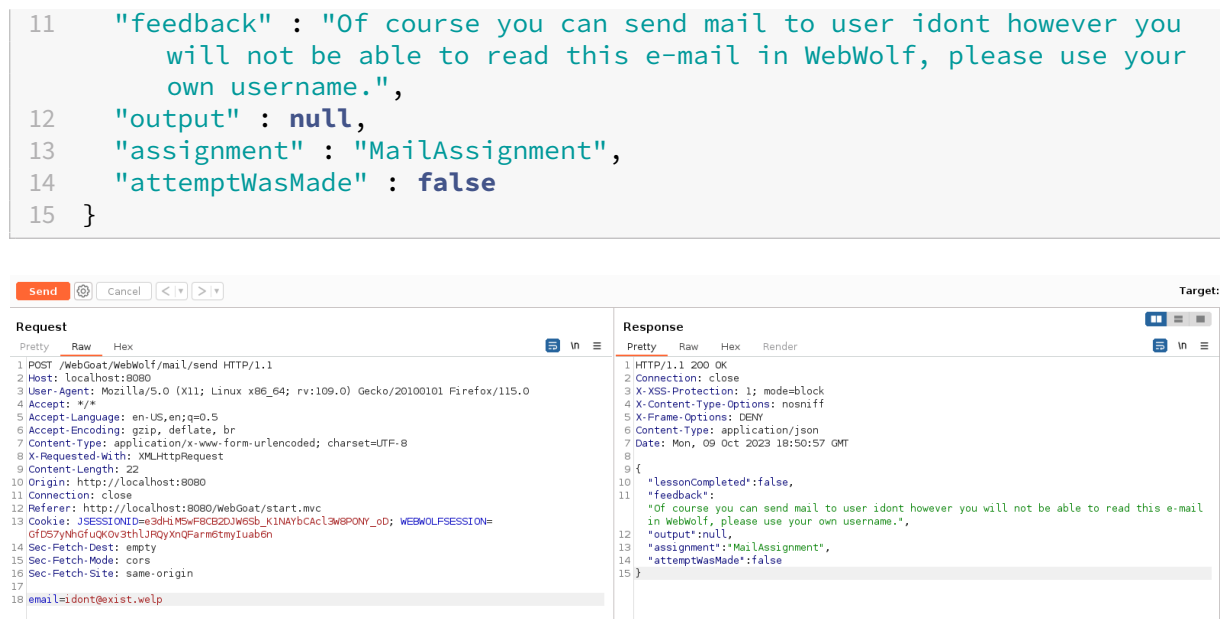
Task 4

The name of the input field is **changeMe**.

Task 5

When intercepting the request that was earlier used to send a mail to WebWolf and changing the target address to **idont@exist.welp**, the response looks like this:

```
1 HTTP/1.1 200 OK
2 Connection: close
3 X-XSS-Protection: 1; mode=block
4 X-Content-Type-Options: nosniff
5 X-Frame-Options: DENY
6 Content-Type: application/json
7 Date: Mon, 09 Oct 2023 18:50:57 GMT
8
9 {
10   "lessonCompleted" : false,
```

**Figure 1:** HTTP Proxies Task 7

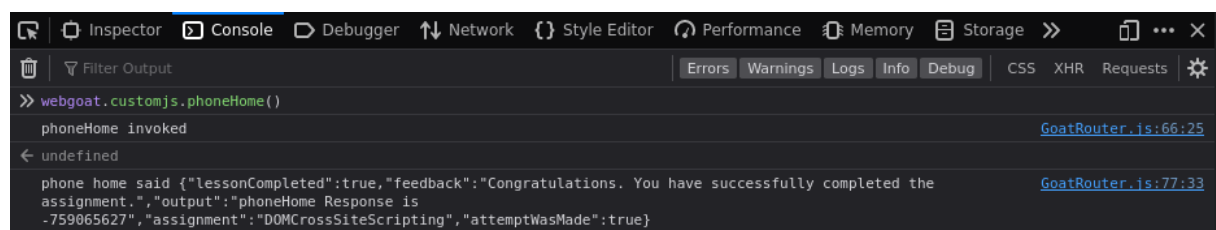
Task 6

Since I feel quite comfortable with the developer tools already, I will simply present the solution to the WebGoat tasks here.

Developer Tools Section 4

Use the console in the dev tools and call the javascript function `webgoat.customjs.phoneHome()`.

The answer for this question is randomly generated each time the function is called. Simply open your browser's developer console and run it to receive the result.

**Figure 2:** Developer Tools Task 4

Developer Tools Section 6

In this assignment you need to find a specific HTTP request and read a randomized number from it.

Open the “Network” tab in your browsers developer tools and take a look at the request body of the request. Note, that this answer is also randomly generated each time the request is send.

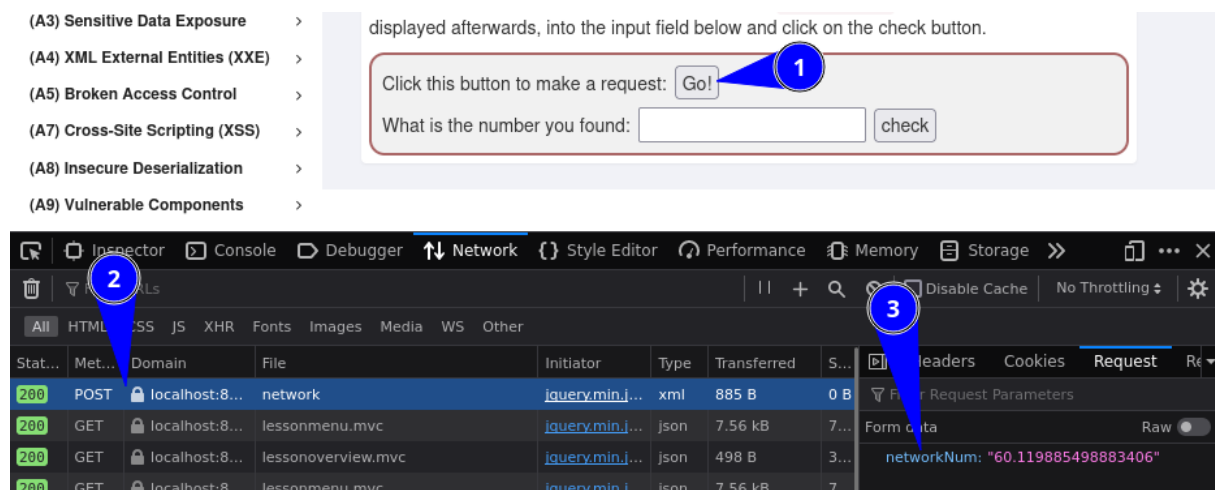


Figure 3: Developer Tools Task 6

Task 7

I did not understand this task. (TODO)

Task 8

1. How could an intruder harm the security goal of confidentiality?

Solution 3: By stealing a database where names and emails are stored and uploading it to a website.

2. How could an intruder harm the security goal of integrity?

Solution 1: By changing the names and emails of one or more users stored in a database.

3. How could an intruder harm the security goal of availability?

Solution 4: By launching a denial of service attack on the servers.

4. What happens if at least one of the CIA security goals is harmed?

Solution 2: The system's security is compromised even if only one goal is harmed.

Task 9

Base64 works by taking any binary input and splitting it into a 6-bit character representation. Usually, this is used when it is necessary to transfer binary files like images via text, but it has also been adapted for encoding of large junks of text data, like cookies or some headers in HTTP/S requests.

Task 10

Base64 has one flaw: If the original size of the binary data is not a multiple of three, there may be some empty bytes. This is solved by appending enough empty bytes to the end of the input to pad the data to a 3-byte-multiple. Since empty bytes cannot be natively encoded with Base64, they are represented through = signs at the end of the encoded data.

Task 11

HTTP Basic Authentication uses Base64 encoding to transfer a clients credentials through a request header called `Authorization`. This header contains the credentials in the following format:

```
1 <username>:<password>
```

This is then encoded and prepended with the string “Basic” to let the server know what type of authorization is used. A full header would look like this:

```
1 Authorization: Basic eW91d2lzaHRoaXN3YXNteXBhc3N3b3JkOmJ1dG5vcGUK
```

Task 12

When storing passwords using XOR, the key is repeated until it fits the length of the input. This lengthened key is then XORed with this input to create the cipher.

XOR is a very simple encoding that simply compares both strings bit by bit and outputs 1 if the bits are *not* equal and 0 if they are.

Input Bit	Key Bit	Output Bit
0	0	0
0	1	1
1	0	1

Input Bit	Key Bit	Output Bit
1	1	0

Task 13