Basicauth.docx. (pdf)
Author: Elhadji Amadou Touré '25
CS 338: Computer Security; Fall 2024; Carleton College
Created 09/24/2024
Last Modified 09/24/2024

Sequence of Events in Basic Access Authentication

Utilizing Burpsuite, there are a couple of things going on here:

1. Initial Request to Access Protected Resource (Without Credentials)
   - When we first try to access a resource protected by Basic Authentication, such as "/basicauth/" on "cs338.jeffondich.com", the browser sends an HTTP "GET" request without providing any credentials.

```
GET /basicauth/ HTTP/1.1
Host: cs338.jeffondich.com
Accept-Language: en-US
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/127.0.6533.89
Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v
=b3;q=0.7
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
```

   - **Server Response**: The server responds with an HTTP "{401 Unauthorized}" status, which means the requested resource is protected, and we need to provide credentials.
   - The response also contains a "WWW-Authenticate" header that indicates the type of authentication required and a description of the realm (a string that identifies the protected area).

```
HTTP/1.1 401 Unauthorized
Server: nginx/1.18.0 (Ubuntu)
Date: Wed, 25 Sep 2024 03:56:31 GMT
Content-Type: text/html
Content-Length: 590
Connection: keep-alive
WWW-Authenticate: Basic realm="Protected Area"

<html>
  <head>
    <title>
      401 Authorization Required
    </title>
  </head>
  <body>
    <center>
      <h1>
        401 Authorization Required
      </h1>
    </center>
    <hr>
    <center>
      nginx/1.18.0 (Ubuntu)
    </center>
  </body>
</html>
<!-- a padding to disable MSIE and Chrome friendly error page -->
<!-- a padding to disable MSIE and Chrome friendly error page -->
<!-- a padding to disable MSIE and Chrome friendly error page -->
<!-- a padding to disable MSIE and Chrome friendly error page -->
<!-- a padding to disable MSIE and Chrome friendly error page -->
<!-- a padding to disable MSIE and Chrome friendly error page -->
```

   At this point, our browser prompts us to enter a username and password.

2. **Providing Credentials via the Authorization Header**:
- After we provide the credentials (in this case, username “cs338” and password “password”), the browser resends the HTTP “GET” request, but this time it includes the “Authorization” header, which contains our credentials encoded in Base64 format.
- The “Authorization” header is pretty important in Basic Authentication, as it carries the user's credentials in a specific format: “Basic <Base64-encoded-credentials>”.

- **Explanation of the Authorization Header**:
The `Authorization` header specifies the credentials using the "Basic" authentication scheme. Y3MzMzg6cGFzc3dvcmQ=` is the Base64-encoded version of “cs338:password”. Base64 is a simple encoding method that converts the username and password into a format that can be transmitted in HTTP headers. However, Base64 encoding is not a form of encryption; it can easily be decoded. Without HTTPS, these credentials can be intercepted and decoded by attackers (allegedly, I believe).

```
1  GET /basicauth/ HTTP/1.1
2  Host: cs338.jeffondich.com
3  Cache-Control: max-age=0
4  Authorization: Basic Y3MzMzg6cGFzc3dvcmQ=
5  Accept-Language: en-US
6  Upgrade-Insecure-Requests: 1
7  User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
   AppleWebKit/537.36 (KHTML, like Gecko) Chrome/127.0.6533.89
    Safari/537.36
8  Accept:
   text/html,application/xhtml+xml,application/xml;q=0.9,image
   /avif,image/webp,image/apng,*/*;q=0.8,application/signed-ex
   change;v=b3;q=0.7
9  Accept-Encoding: gzip, deflate, br
10 Connection: keep-alive
```

3. **Server Authenticates the User**:
- The server decodes the Base64 string from the “Authorization” header to retrieve the username (“cs338”) and password (“password”).
- The server then checks these credentials against its internal database. If the credentials are correct, the server authorizes the user and grants access to the “protected resource”.

```
HTTP/1.1 200 OK
Server: nginx/1.18.0 (Ubuntu)
Date: Wed, 25 Sep 2024 03:57:49 GMT
Content-Type: text/html
Connection: keep-alive
Content-Length: 509

<html>
  <head>
    <title>
      Index of /basicauth/
    </title>
  </head>
  <body>
    <h1>
      Index of /basicauth/
    </h1>
    <hr>
    <pre>
      <a href="../">
        ../
      </a>
      <a href="amateurs.txt">
        amateurs.txt
      </a>                                    04-Apr-2022 14:10                  75
      <a href="armed-guards.txt">
        armed-guards.txt
      </a>                                   04-Apr-2022 14:10                 161
      <a href="dancing.txt">
        dancing.txt
      </a>                                    04-Apr-2022 14:10                 227
    </pre>
    <hr>
```

- In this case, the server responds with an HTTP “200 OK” status and provides the requested content, such as a directory listing or a specific file (e.g., “armed-guards.txt”).

4. **Subsequent Requests**:
   - After successfully authenticating, the browser automatically includes the same “Authorization” header in all subsequent requests to the protected directory or its contents.
   - I believe this is why the following requests to resources like “armed-guards.txt” and “dancing.txt” also include the “Authorization” header without needing the user to re-enter credentials.

Request

```
GET /basicauth/armed-guards.txt HTTP/1.1
Host: cs338.jeffondich.com
Authorization: Basic Y3MzMzg6cGFzc3dvcmQ=
Accept-Language: en-US
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/127.0.6533.89
 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image
/avif,image/webp,image/apng,*/*;q=0.8,application/signed-ex
change;v=b3;q=0.7
Referer: http://cs338.jeffondich.com/basicauth/
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
```

Response

```
HTTP/1.1 200 OK
Server: nginx/1.18.0 (Ubuntu)
Date: Wed, 25 Sep 2024 04:00:25 GMT
Content-Type: text/plain
Content-Length: 161
Last-Modified: Mon, 04 Apr 2022 14:10:51 GMT
Connection: keep-alive
ETag: "624afc6b-a1"
Accept-Ranges: bytes

"The only truly secure system is one that is powered off, cast in a block of
concrete and sealed in a lead-lined room with armed guards."

    -- Gene Spafford
```

This process is effectively repeated for all the other documents accessed, with different content lengths, types, dates, etc., and the text / HTML of those documents printed.

That said, here is a general summary of the events:

1. **Initial Request (No Credentials):**
   - The browser requests a protected resource, and the server responds with a “401 Unauthorized” status, asking for credentials.

2. **Credentials Provided via “Authorization” Header**:
   - The user provides credentials, which the browser sends in the “Authorization” header (Base64-encoded).

3. **Server Authenticates and Authorizes**:
   - The server decodes the credentials, verifies them, and grants access if they are correct, responding with a “200 OK” status.

4. **Subsequent Requests**:
   - For any further requests to the protected directory, the browser automatically includes the “Authorization” header, making sure we have seamless access without repeatedly prompting for credentials.

Concerning “Authorization” Header:

Basic Authentication is effectively a pretty simple method for protecting resources. It actually relies on the “Authorization” header to transmit credentials, and the server responds with the appropriate HTTP status depending on the validity of those credentials.

The header follows the format `Authorization: Basic <Base64-encoded-credentials>. Then, the Base64 Encoding is used for transmitting the credentials in a more standardized format, but it is not secure on its own. Finally, Basic Authentication is only secure when used over HTTPS, as it prevents attackers from intercepting and decoding the credentials.