

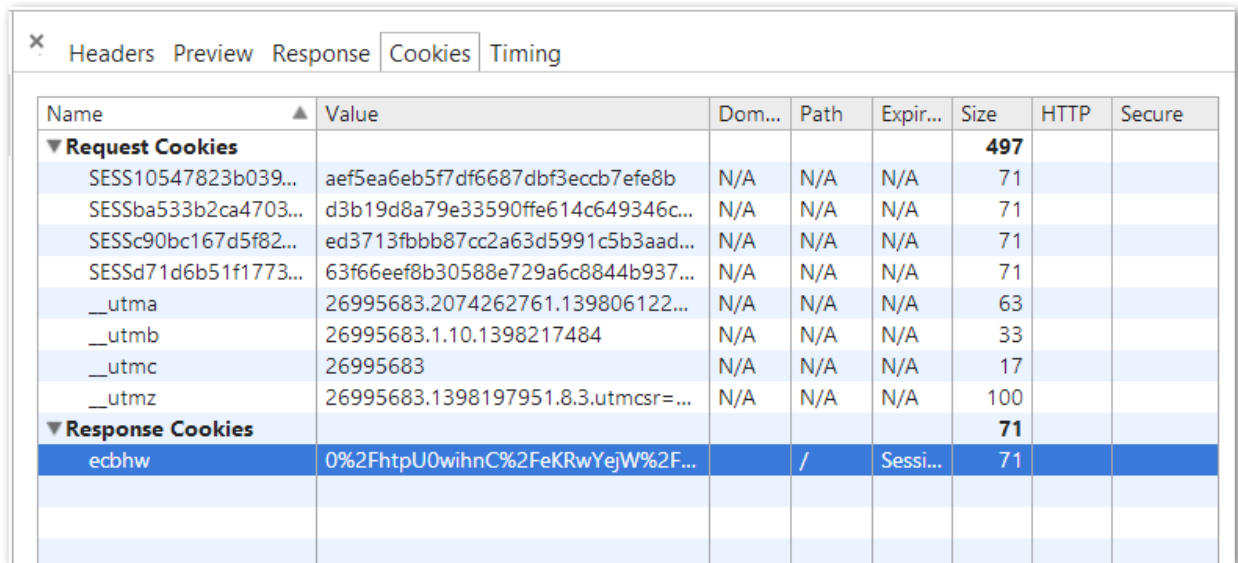
ECE 478 Network Security: Homework #3

Disclaimer

This submission reflects my own understanding of the homework and solutions. All of the ideas are my own, unless I explicitly acknowledge otherwise.

1. What is the name of the cookie that is set by this page?

Reading your source code it looks like the cookie is named ecbhwh. The source code where you are making a new cookie it looks like you set `-name => "ecbhwh",.`



| Name | Value | Dom... | Path | Expir... | Size | HTTP | Secure |
|----------------------|------------------------------------|--------|------|----------|------|------|--------|
| ▼ Request Cookies | | | | | | 497 | |
| SESS10547823b039... | aef5ea6eb5f7df6687dbf3eccb7efe8b | N/A | N/A | N/A | 71 | | |
| SESSba533b2ca4703... | d3b19d8a79e33590ffe614c649346c... | N/A | N/A | N/A | 71 | | |
| SESSc90bc167d5f82... | ed3713fbbb87cc2a63d5991c5b3aad... | N/A | N/A | N/A | 71 | | |
| SESSd71d6b51f1773... | 63f66eef8b30588e729a6c8844b937... | N/A | N/A | N/A | 71 | | |
| __utma | 26995683.2074262761.139806122... | N/A | N/A | N/A | 63 | | |
| __utmb | 26995683.1.10.1398217484 | N/A | N/A | N/A | 33 | | |
| __utmc | 26995683 | N/A | N/A | N/A | 17 | | |
| __utmz | 26995683.1398197951.8.3.utmcsr=... | N/A | N/A | N/A | 100 | | |
| ▼ Response Cookies | | | | | | 71 | |
| ecbhwh | 0%2FhttpU0wihnC%2FeKRwYejW%2F... | | / | Sessi... | 71 | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |

2. Authentication tokens are made by encrypting (then base-64 encoding) some raw data. Looking at the source code of the script, what must a token for the admin user contain as its raw data?

The \$data that is being encrypted then passed into the base 64 encoding is created with this code.

```
## scalar(time) returns # of seconds since 1/1/1970
my $data = sprintf "ok|%s|%s|",
    scalar(time), param("accountname");
```

This code will look like this for the admin account.

```
## scalar(time) returns # of seconds since 1/1/1970
my $data = sprintf "ok|%s|%s|",
    scalar(time), param("$ADMIN_ACCOUNT");
```

The raw data will be “**ok|1398220350|admin**” . The center number 1398220350 is the time since the epoch in seconds since the professor created the admin account. This is that raw data that is sent into the encrypt then encode_base64. When the username, timestamp, and ok need to be recovered we decrypt the long token cookie value, then send it through the decode_base64, and finally we separate the data on the | character.

That is what this code accomplishes.

```
my $dat = decrypt(decode_base64($cookie->value));  
my ($ok, $timestamp, $username) = split /\|/, $dat;
```

Then this code basically sees if the cookie is an actual cookie created with the encode_base64 and encrypt. It does this by checking if the first part of the decrypted data is equal to “ok” which would not be true if ok was not encrypted into the cookie in the first place. The rest of the code just print out the username and time when the user created their account out to the user.

```
if ($ok eq "ok") {  
    $AUTHORIZED = $username;  
    my $t = localtime($timestamp);  
    msg "Welcome back, $username (member since $t)";  
} else {  
    msg "Invalid authentication cookie detected";  
}
```

3. The token data is encrypted with a block cipher in ECB mode. If you weren't told that it used ECB, and you didn't have access to the source code, how could you verify that indeed used ECB?

We learned in class multiple times why Adobe was stupid in using ECB mode. ECB mode leaks data. Any data that is the same decrypted data before encrypting becomes the same encrypted data. For instance if I encrypted “Password” two different times both times I would end up with the same encrypted base 64 output. The way that I can test if a site used the ECB mode is to create a username two time like “Password”, “Password” and compare to see if anything has changed. I will delete my cookies in between these two tries. If some part of the cookie is still the same using the same two passwords then the website is using ECB mode.

ECB mode also has a block size. Looking over the notes only if the whole block is the same will the output also be the same. So it is dependent on block size.

I created a password account 4 different times on the hw3 website. As you can see the decoded last parts are the exact same with the password account name. This proves that the website for homework number 3 is using the ECB mode.

yQ6N+ZNIX14GwN7kG/gWA=

Password

nMftaVp15FBBRIjlpJAB1QyQ6N+ZNIX14GwN7kG/gWA=

mSdM+3VMTaJFJxL0VaXGcgyQ6N+ZNIX14GwN7kG/gWA=

gUZCKCN7BUALICkrxOcGgwyQ6N+ZNIX14GwN7kG/gWA=

ERojNsKInAJkYxdds8WRmQyQ6N+ZNIX14GwN7kG/gWA=

This is the encoded cookie, I have listed everything as decoded. Since that is how Chrome shows it.

ecbhw=ERojNsKInAJkYxdds8WRmQyQ6N%2BZNIX14GwN7kG%2FgWA%3D%0A

“Eddie” account name cookies.

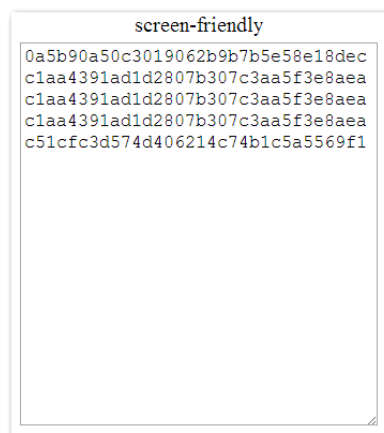
zC4k1Sl5hn9ga50ZReOIRYnaq6c445VkGyvKppsduEI=

U0MD0S3gvKhnhlm4zFXe4naq6c445VkGyvKppsduEI=

3uyiCmII6E4/IjLI73Mwg4naq6c445VkGyvKppsduEI=

4. How can you determine the block-length of the block cipher, just from the observable behavior of the website? What is the block-length used in this page? (your answer should be a number with units!)

After having class on Wednesday I see now that the block length can be determined by taking the base64 output changing it into Hex then looking at the output. Basically to see the block length I created an account with repeated bbbbbbbbbbbbbbbbbbbb. Looking at the base64 output converted to hex I see that it is repeated every 32 hex numbers. This is equivalent of a block size of 16 bytes long, or 128 bit block length. This block length was determined by the cookie that the website creates from base64 encoding and encryption. I don't see this directly on the site, I have to look in the chrome developer tools.



I then tested this theory of a 16 byte length block size by increasing the number of bbb's until another identical block was created, then removed bb's until the block was removed from the output. This proved that the block cipher is using 128bits. I went from 50 b's down to 34.

5. What parts of the token raw-data can you (as the attacker) control?

The raw data looks like "ok|timestampnumber|accountname".

When creating the account we can control the account name and the timestamp based on when we login. As an attacker we should be able to create a cookie that uses the knowledge that a certain number of characters at the end of the cookie are dedicated to the account name, then use these characters to encode and base64 and account name named \$ADMIN_ACCOUNT.

6. You will need to get an admin token, which means you need to get encrypted blocks that encode the raw data you described in problem 2. What should you submit to the form in order to get these encrypted blocks? Why is it necessary/helpful to know the block-length for this step?

Since I know that the block length is 16 bytes\128bits I can create an account named something along the lines of adminadminadminadmin to see what the repeated plain text will be shown.

From problem 2 I know that the encoded data will look like this.

" admin"

14 bytes for the time since the Epoch 1970, the two |, and the ok. 5 more bytes for the admin making a total of 19 bytes.

So if I use an account name **adminadminadminadm** the bolded is admin

Cookie for admin.

dtWuh4SvqqRLoqPhFkmVoNd0u2qIbHUsko8Rf9tMues=

LHbuc80uGMgGQUIMGTPnFtd0u2qIbHUsko8Rf9tMues=

Since ok|1398400121| takes the 14 bytes and the block length is 16 bytes. I can simply use an account name like aaok|1398400121|admin. This will make the first 16 byte block of the cookie look like

“ok|1398400121|aa” first 16 bytes

“ok|1398400121|ad” second 16 bytes

“min00000000000000” last 16 byte block

I then simply take the cookie convert it from base64 to hex take out the first 16 bytes, then convert it back into base64. It is now ready to be used as a cookie for the admin. I only need to URL encode the cookie now, and insert it with JavaScript.

7. In the previous step, you would have to obtain some kind of cookie token. Describe in full detail how you can use this cookie to obtain an admin token.

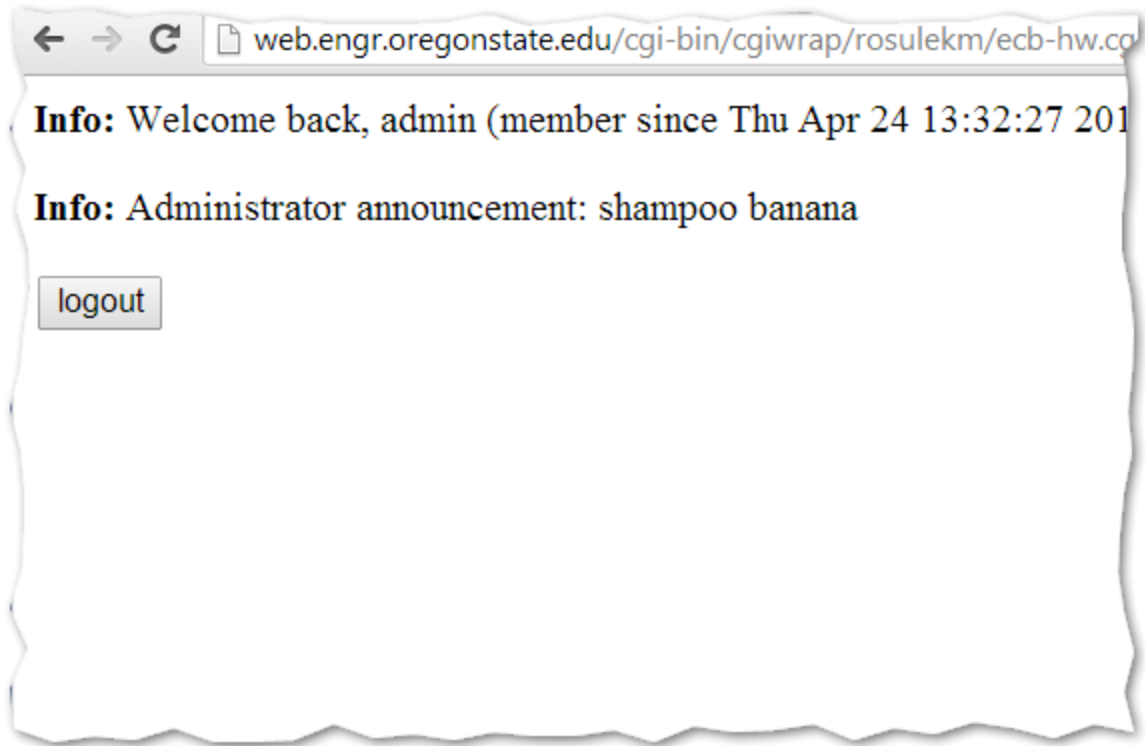
The full detail like said in the end of problem 6 is to take the cookie that has the admin cookie token I need convert it to hex remove the first 32 hex values. Re-encode these last two 32 hex values to base64 and now I have a cookie I can use to be seen as the admin by the server. Lastly it just needs to be converted to URL encoded cookie when inserted into the webpage with JavaScript.

8. Find out how to use JavaScript to set a cookie for a page. Then manually set the cookie for this page to one of your own choosing, by using your browser’s developer tools. For me, in Chrome, it’s the “console” tab of the developer tools, where I can type in JavaScript that is executed in the context of the page.

I went to the Chrome console and used the cookie I had created in problem 6.

```
Document.cookie = “ecbhw=” + encodeURIComponent(“LHbuc80uGMgGQUIMGTPnFtd0u2qIbHUsko8Rf9tMues=”)
```

Using this cookie from problem six I am able to get the unguessable message “shampoo banana”.



9. What is the secret admin message?

I accidently answered problem 9 in problem 8. Look there for the answers.