

# Authentication

## Bruteforcing login page techniques

The following techniques can be applied if you do not know a valid username or password. Depending on what controls the site has active, you can test a bruteforce username list to find a valid one before trying a password brute force with the valid username. This exponentially decreases the time it takes to find a login opposed to trying every password with every username.

### Username enumeration via different responses

Using burp intruder sniper, we put in an invalid password and test a username list.  
In this method, only one of the requests will return a different length response, meaning that entries username is valid.

### Username enumeration via subtly different responses

This method generally relies upon human error. Assume that the length is either always the same, or all different so you can not use method 1. We could test to see the message that pops up when a failed login attempt happens. If the user/pass are both invalid, does it say "Invalid username and password.", but when only the password is invalid it displays "Invalid username and password". In the portswigger lab it is a human error of not including the period in the username valid pass invalid case. We can use grep to search results that contain the "Invalid username and password.", cases that do not contain it are a valid username.

### Username enumeration via response timing

This assumes two things:

- 1) the other 2 options are not possible
- 2) The program only checks if the password is valid if the username is valid

In this case, we can supply an extremely long password, atleast 100 characters, and then brute force our username list. Because the password is extremely long, when the program hits a valid username it will take a longer time to process the request. This allows us to check the response timing to see which usernames are valid.

In the portswigger lab it also applied a IP-brute force protection system, this is easily ignored by using the "X-Forwarded-For: \$0\$" tag and a pitchfork attack to increase it and our username by 1 per run.

#### **Flawed brute-force protection**

Assumptions:

- 1) We know a valid username/password, such as our own.
- 2) We know the username of another user, and are attempting to crack the password.

Some login forms will disable your ability to login if you have too many failed attempts in a short period of time. This example exploits a flawed creation, where this counter is reset whenever you login to a valid user.

Through this technique, we can login to our own account/another user account we have access to, attempt a few brute force passwords, log in to our own account again, and repeat.

How to do this with burp: bruteforce password file must be edited to include known password every 2 spots, and we must create a username file that contains username we are trying to crack twice, then ours repeated.

For example, if our username is wiener and user we are trying to crack is carlos

User file:

Carlos  
Carlos  
Wiener  
Carlos  
Carlos  
Wiener

Our password is peter, others are passwordlist

Password file:

123  
321  
peter  
abc  
cba  
peter

In some cases, a users account will be locked after x amount of failed login attempts.

This is different from the previous account, where it was our computer being locked out from attempting to login too many times.

We can use this as a conditional error similar to our SQLi labs. We use a bruteforce

username list and then a password list of 5-6(however many attempts a username

gets before locked) random entries, and attempt a cluster bomb using burp intruder.

Any account that responds differently in the results is a valid username.

We can then test our password list against the found username, only a few should return

different results. Some of them will be the first few responses, and one will be our actual password.

This assumes that the software is made wrong and will give you a different output

when you attempt to log in using the correct credentials vs incorrect credentials.

#### **Broken brute-force protection, multiple credentials per request**

\*I don't totally understand this one, it is an expert lab so pretty wonky.

This another version of user rate limiting, where the above examples will not work.

The lab on PortSwigger has a login panel that takes the username /password as a json.

Because of this, we are able to supply our entire password list through one single request

allowing us to only submit once, but test every password, bypassing the defenses.

This is done in burp repeater, and will result in a 302 response.

Right click on the request

and select "Show response in browser". Copy URL and load it in your browser, and you will

be logged into his account.

## Bypassing 2FA

### 2FA Simple Bypass

Sometimes 2FA is made extremely poorly, to the point where it will load your account page and store it in your sessions as soon as you've entered valid login info. Simply login, and when it asks you for your 2fa confirmation open a new tab and go to your account page.

### 2FA Bypass where it does not check user submitting verification is same user that logged in

In this example, we require:

- 1) A valid username/password that can be 2fa'd, such as our own
- 2) The username of the a user we are trying to crack

We begin by going through and logging in fully to our own account, then by inspecting how the process went through in the system we can see there is a "verify user" cookie containing our name. Due to the way this is set up, it is not confirmed that the user submitting the verification code is the same user that logged in to the first form.

We log in to our own account, then when hitting the 2nd login we can edit the cookie to be the username of the user we wish to crack. This will generate a 2fa code for the second user, which we can now attempt to brute force.

Enter an invalid 2fa code to generate the appropriate request that we may edit, and brute the code. Check results, one should be a 302/valid response, send to repeat/show in browser.

### 2FA Bypass where invalid 2fa code sends you back to original login screen

Assumptions:

- 1) We know a username and password that we have cracked from some other way
- 2) The login form redirects you to login again after 2 incorrect 2fa codes are entered
- 3) Generating a new 2fa code does not invalidate an old one

This one involved some fancy burp stuff called macros.

You have to create a macro and then add the pages get/post requests related to login in and then spawning the 2fa form. By creating this, when you enter two incorrect 2fa codes, you are sent back to the original login form(Step 1 of the macro), which then enter our login info, processes the post, send you to the get 2fa, rinse and repeat with a brute force of the 2fa code.

Once a 302 has been found, show in browser. This will take an extremely long time in burp as there are 10000 possibliites and you can onyl use 1 thread.

### **Brute-forcing a stay-logged-in cookie**

Assumptions:

- 1) There is a cookie that allows you to stay logged in
- 2) We have a valid account or can create one to view the cookie.
- 3) Cookie is made in a way that is reversible, error on developers side

In the portswigger example, when we log in to our valid account we can inspect the cookie.

It turns out to be: d2llbmVyOjUxZGMzMGRkYzQ3M2Q0M2E2MDExZTllYmJhNmNhNzcw  
You can test some common encoding/decoding techniques, this one turned out to be base64.

If you decode it, you get: wiener:51dc30ddc473d43a6011e9ebba6ca770

wiener is the username

of our valid account so we know this is the correct decoding, and gives us some info about the cookie.

What comes after the colon appears to be an MD5 hash, which we could be a hash of anything

but the safest guesses would be users password, time created or something of the sorts.

We can test this by creating some payload processing rules for our valid account to create

what we think the cookie is supposed to be, and see if it will allow us to stay logged in.

Use payload processing and add the 3 steps to construct cookie:

- 1) MD5 hash password
- 2) add prefix of username:
- 3) b64 encode
- 4) grep "Update Email" to confirm it took us to the login page.

With this information, we can now brute force the victims cookie!

Simply put in your list,

change the prefix to victims username and start attack.

## Offline password cracking

This lab requires XSS, will come back to it later.

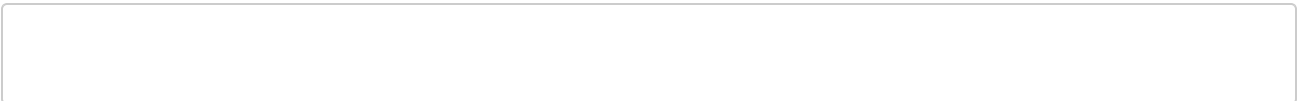
## Resetting passwords using a URL

### Password reset broken logic

Sometimes password reset URL's can be vulnerable!  
In this first example, you must enter a username and a password  
resetlink with a token will  
be emailed to the user. However, the token is not validated when  
submitted back.

This means we can load a valid password reset page through our own  
login info, and simply  
edit the value of the username field through burp to the victims user,  
allowing us to edit  
their password.

#### **Password reset poisoning via middleware**



#### **Password brute-force via password change**

Exploiting a password change form that allows us to enter an arbitrary  
username.

Requires some sort of change password form, generally from logging in  
to your own account

In this example, if the current password is entered incorrectly and the  
2 new passwords  
match, it boots you back to the login screen. However, if you enter two  
incorrect new pass's  
and the correct password, it will same "new passwords do not match". We  
can use this by  
intentionally sending requests with incorrect new passwords and a  
bruteforce list of  
old password until we find the old password of the victim.