1. Broken Authentication - Insecure Login Forms

Affected URLs

http://192.168.56.104/bWAPP/ba_insecure_login_1.php

Description

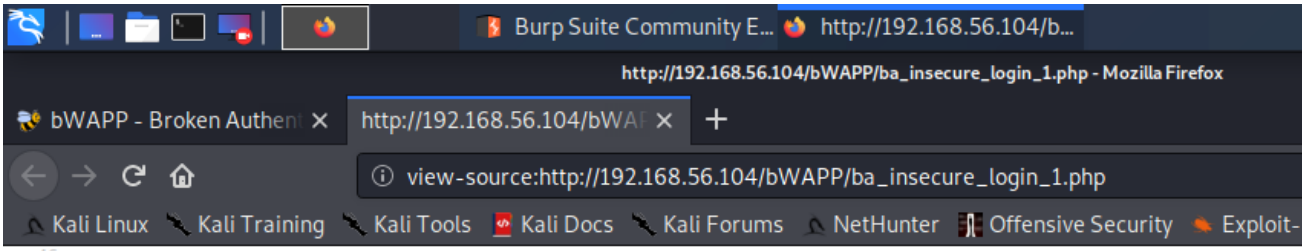
Developer has stored credentials 'login' and 'password' as obfuscated text, using white font on white background, and storing them within the source code.

Impact

Attacker can either select the obfuscated text (in front of login & password) or see the source code of the webpage to detect the credentials. He may then misuse them to login to a legitimate account.

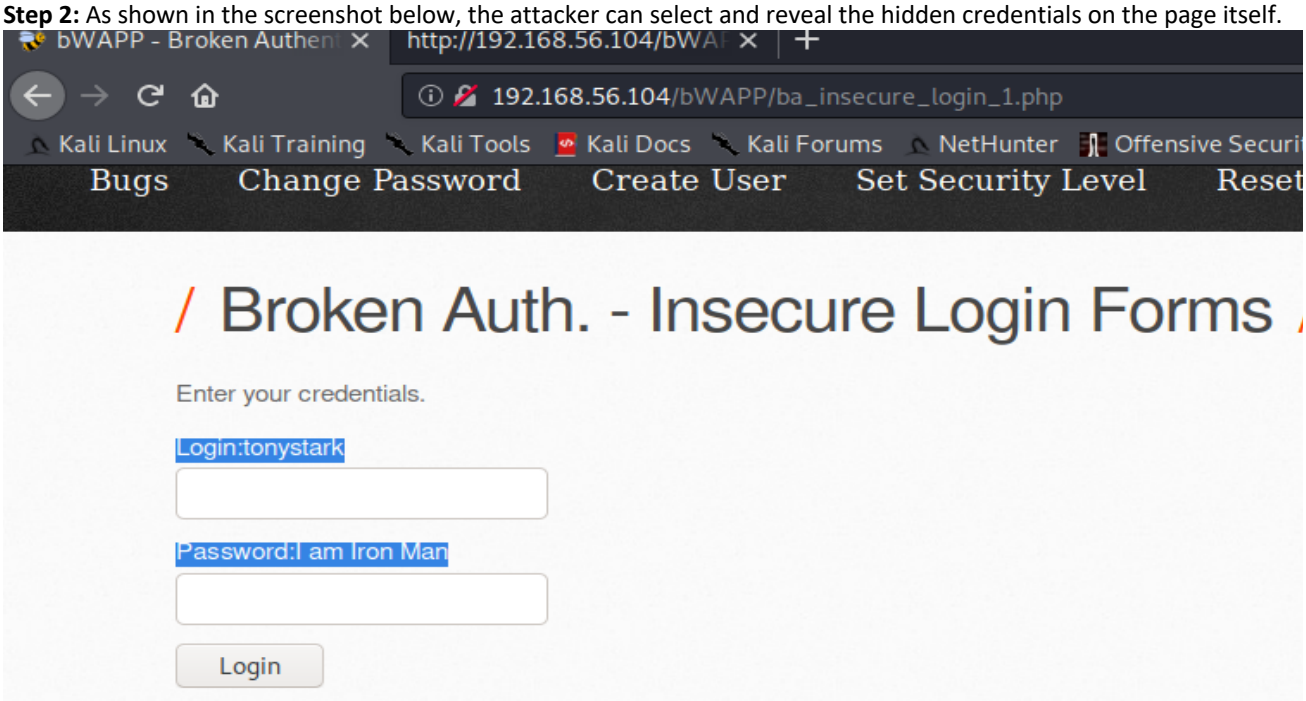
Proof Of Concept

Step 1: As shown in the screenshot below, the attacker can view the page source to reveal the user credentials.




```
46
47      </table>
48
49  </div>
50
51  <div id="main">
52
53      <h1>Broken Auth. - Insecure Login Forms</h1>
54
55      <p>Enter your credentials.</p>
56
57      <form action="/bWAPP/ba_insecure_login_1.php" method="POST">
58
59          <p><label for="login">Login:</label><font color="white">tonystark</font><br />
60          <input type="text" id="login" name="login" size="20" /></p>
61
62          <p><label for="password">Password:</label><font color="white">I am Iron Man</font><br />
63          <input type="password" id="password" name="password" size="20" /></p>
64
65          <button type="submit" name="form" value="submit">Login</button>
```

Step 2: As shown in the screenshot below, the attacker can select and reveal the hidden credentials on the page itself.



Workaround/ Solution

We recommend to:

- never store credentials in clear/obfuscated on the webpage/source code.
- use robust encryption to store passwords in the DB.
- use https encryption from login page onwards to prevent SSL strip attack as well as MITM attacks.

2. Broken Authentication - Logout Management

Affected URLs

http://192.168.56.104/bWAPP/ba_logout.php

Description

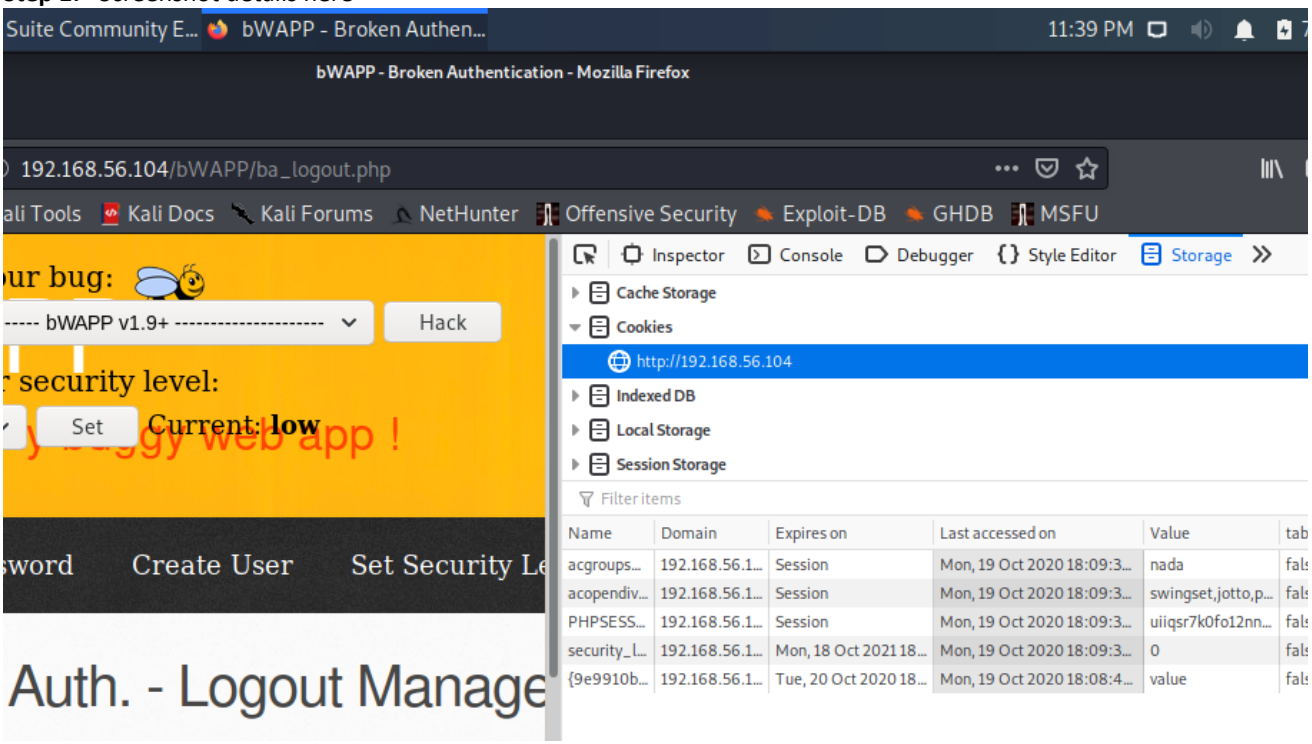
Session ID does not expire or get invalidated after logout. There is no idle timeout built into the webpage.

Impact

Attacker can access the user's account without requiring user's credentials by clicking the back-button of the same window or by copying the URL of the logout page and pasting it in a new browser window.

Proof Of Concept

Step 1: <Screenshot details here>



Step 2: <Screenshot details here>
<add screenshot here>

Workaround/ Solution

We recommend to:

- Set the cookie/session ID to expire when the user clicks the 'Logout' button.
- Set the expiration time of the session ID to a previous date or to "session".
- Provide manual session expiration, enabling the user to actively close the session once they have finished using the webpage.

3. Broken Authentication - Password Attacks

Affected URLs

http://192.168.56.104/bWAPP/ba_pwd_attacks_1.php

Description

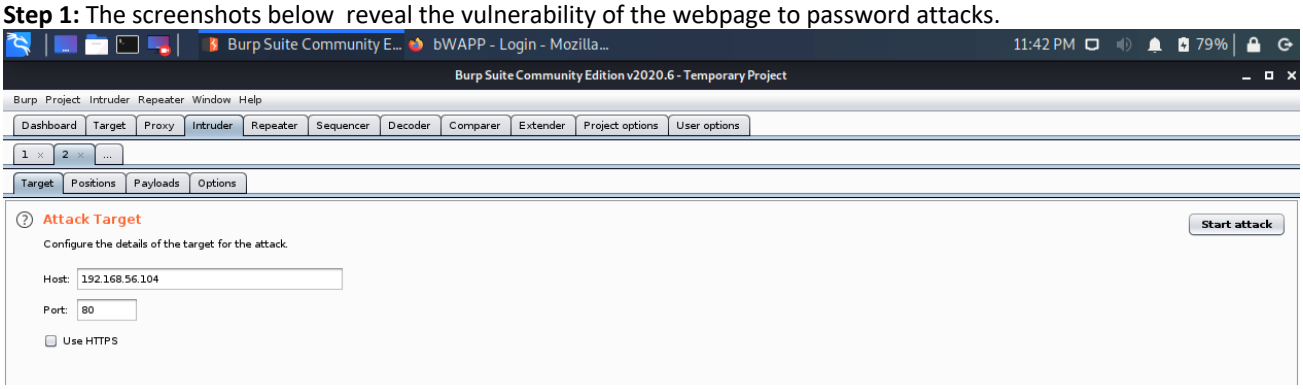
Webpage is vulnerable to password attack using brute force (sniper, pitchfork methods in Burp Suite).

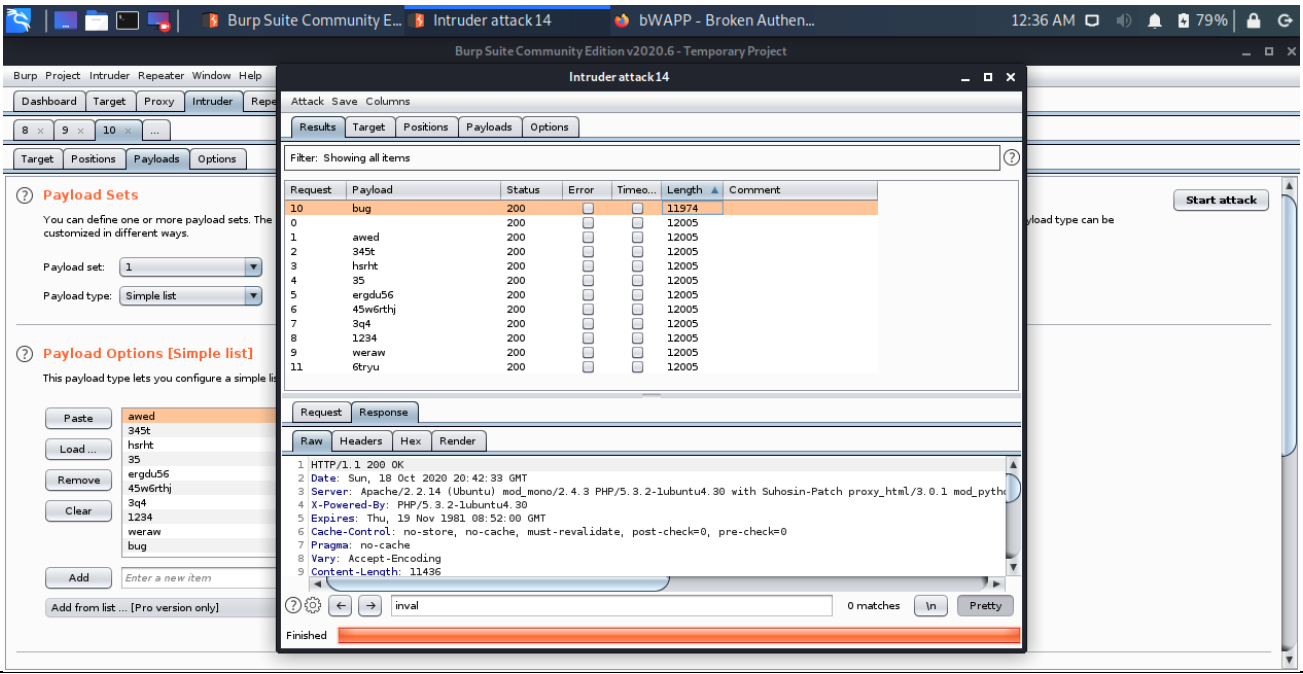
Impact

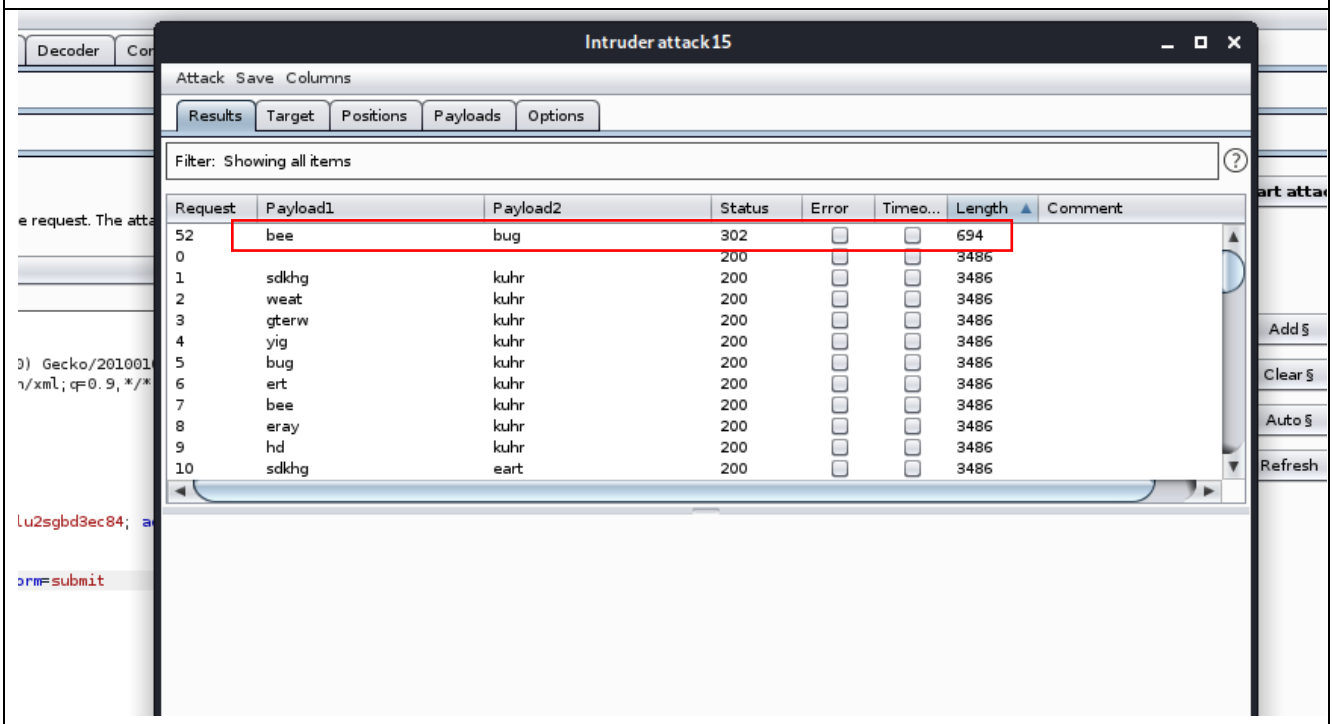
The attacker can use intercepting proxy software to bruteforce and gain credentials of the user to extract user info.

Proof Of Concept

Step 1: The screenshots below reveal the vulnerability of the webpage to password attacks.







Workaround/ Solution

We recommend to

- Use Captcha/image-recognition to prevent repeated automated attacks.
- Use rate-limiting after 3 unsuccessful attempts on sensitive websites like bank/financial sites.
- Use multi-factor authentication.
- Use IP-tracking for additional authentication (through email/SMS) during suspicious login attempts.

4. SQL Injection (Search/GET)

Affected URLs

http://192.168.56.102/bWAPP/sqli_1.php

Description

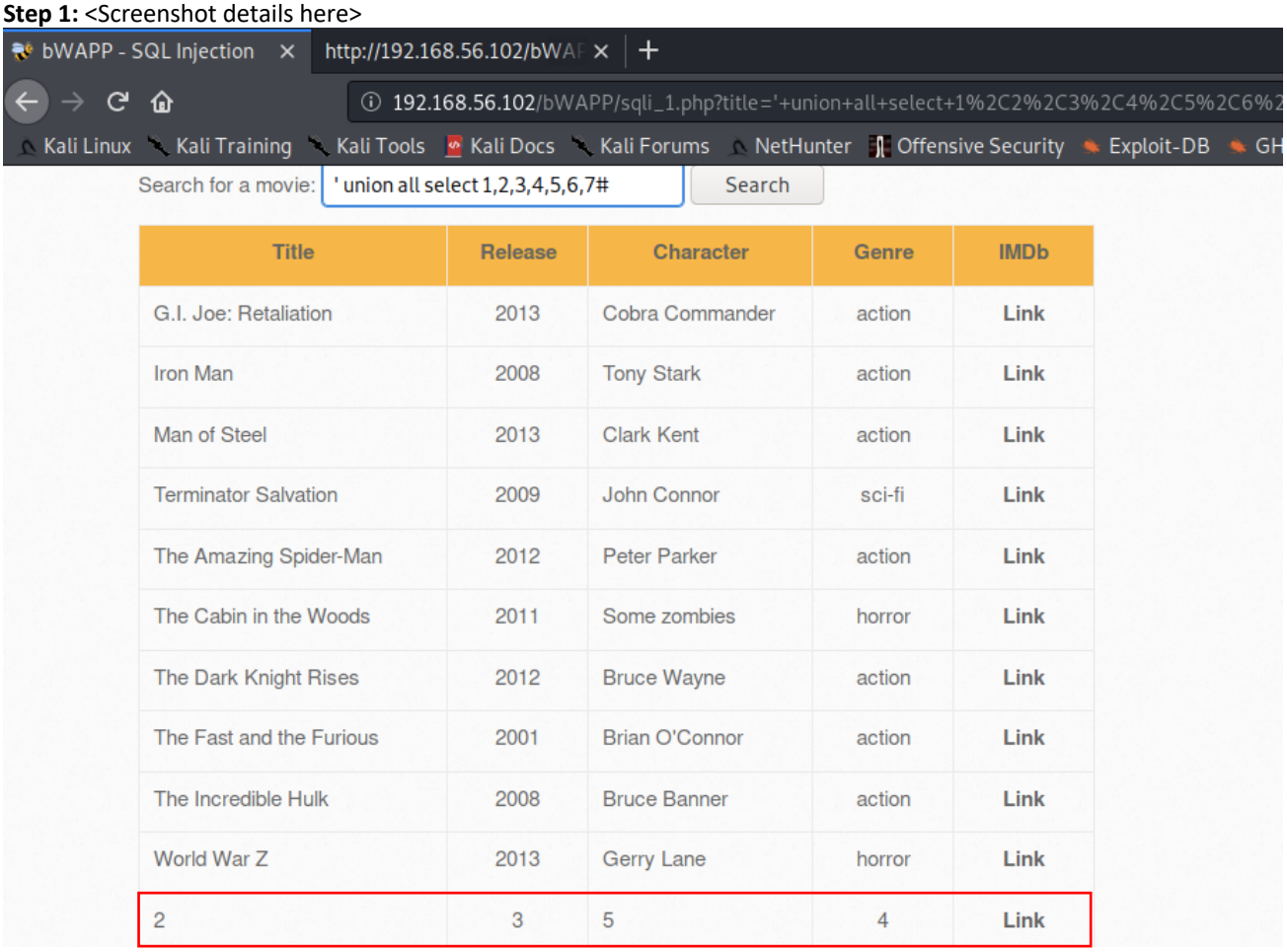
SQL injection attack consists of insertion or “injection” of a SQL query via the input data from the client to the application.

Impact

SQL injection attacks allow attackers to spoof identity, tamper with existing data, cause repudiation issues such as voiding transactions or changing balances, allow the complete disclosure of all data on the system, destroy the data or make it otherwise unavailable, and become administrators of the database server.

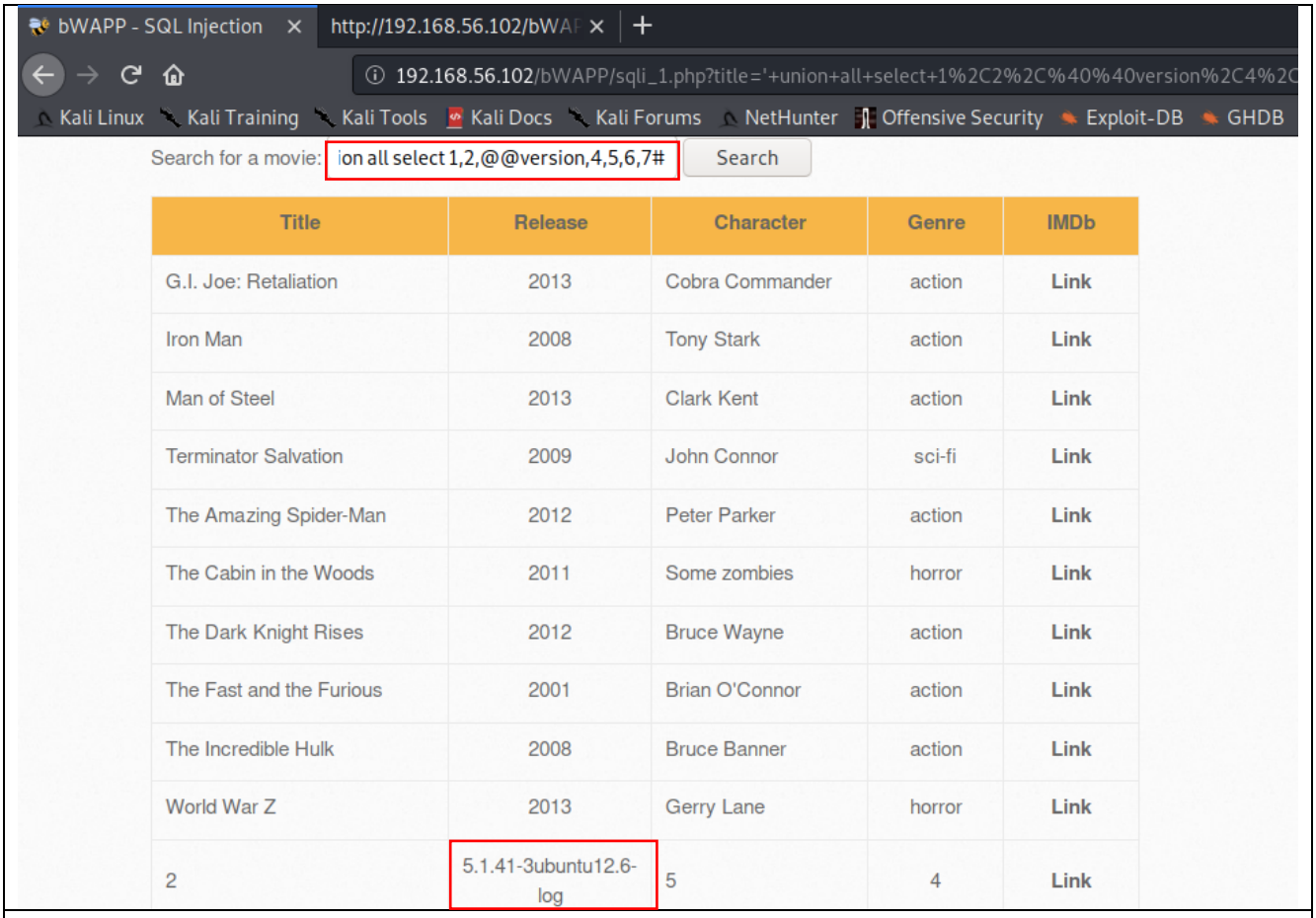
Proof Of Concept

Step 1: <Screenshot details here>



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
2	3	5	4	Link

Step 2: <Screenshot details here>



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
2	5.1.41-3ubuntu12.6-log	5	4	Link

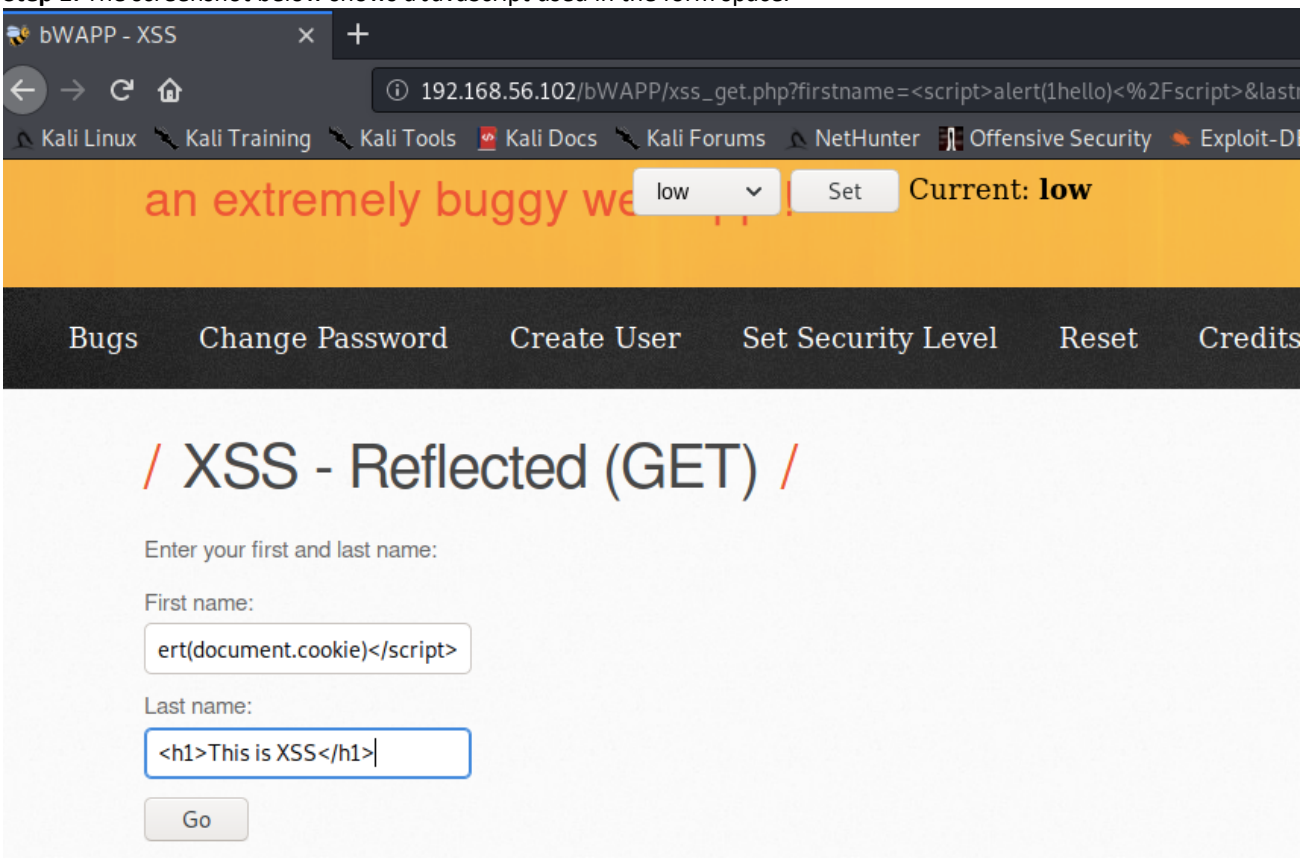
Workaround/ Solution

We recommend to:

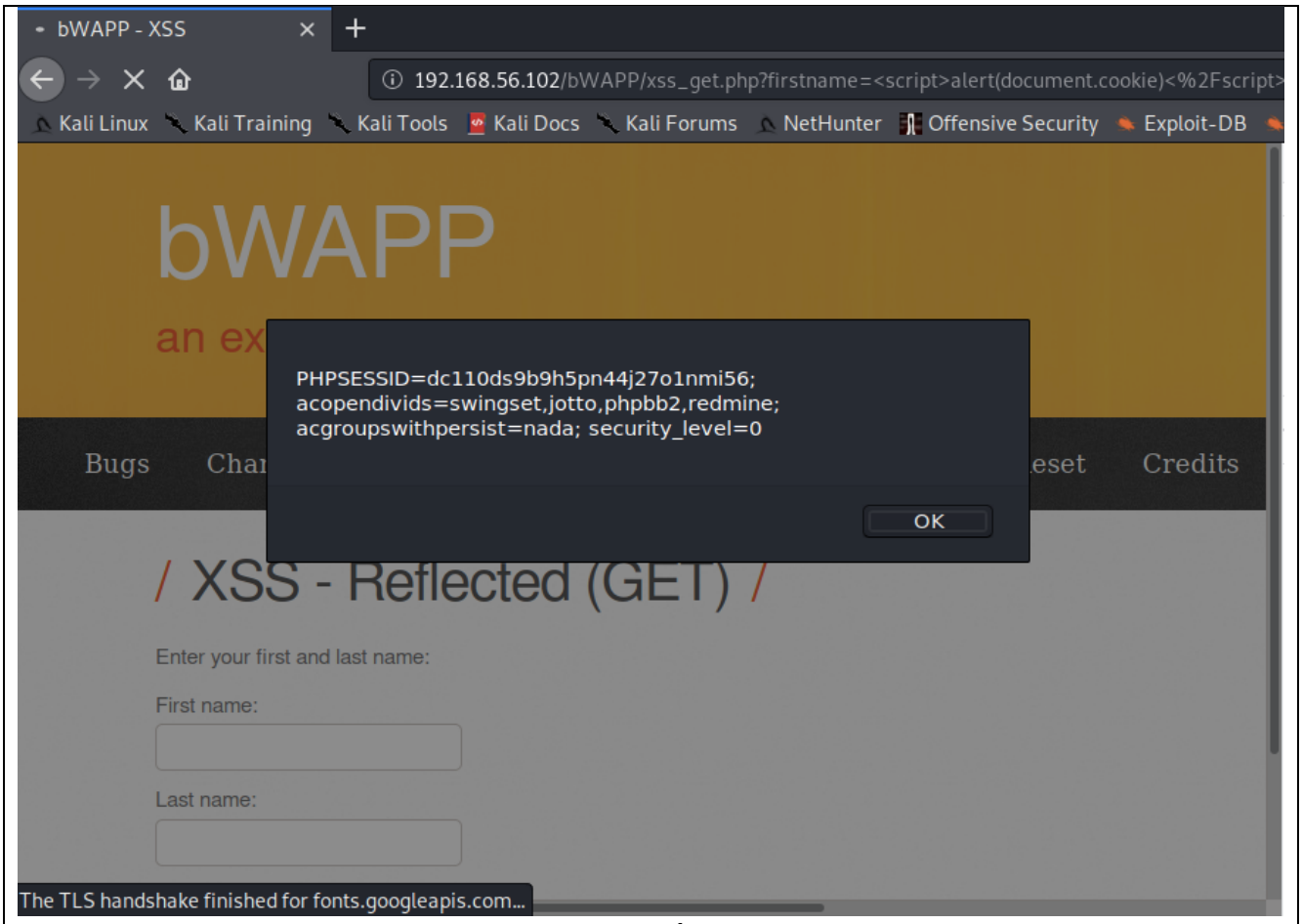
- Escape characters in input to SQL queries.
- Use stored procedures.
- Run output validation (error messages).

5. Cross Site Scripting – Reflected (GET)

Affected URLs
http://192.168.56.102/bWAPP/xss_get.php
Description
Occurs when a malicious script is reflected off of a web application to the victim's browser. The script is activated through a link, which sends a request to a website with a vulnerability that enables execution of malicious scripts.
Impact
Attacker can use crafted URLs with built-in scripts to cause the user to reveal session tokens. This can help the attacker hijack user's session, or steal cookies/session IDs, and send them to 3rd party domain (like pastebin)
Proof Of Concept
Step 1: The screenshot below shows a JavaScript used in the form space.



Step 2: The screenshot below shows the PHPSESSID revealed as a response to the attacker's malicious JavaScript.



Workaround/ Solution

We recommend to:

- Conduct strict input validation.
- Encode data on output.
- Use appropriate response headers.
- Use Content Security Policy.

6. Cross Site Scripting – Stored (Blog)

Affected URLs

http://192.168.56.102/bWAPP/xss_stored_1.php

Description

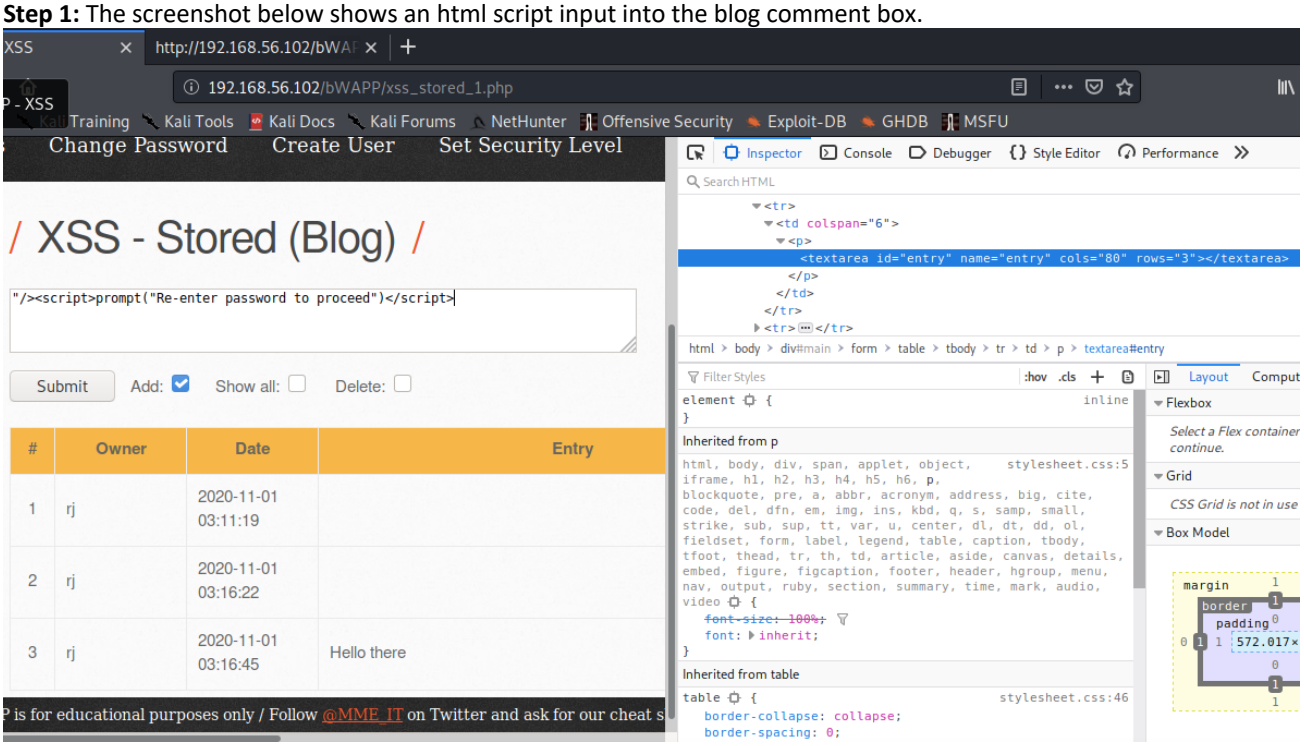
Stored/persistent XSS occurs when an application receives data from an untrusted source and includes that data within its later HTTP responses in an unsafe way.

Impact

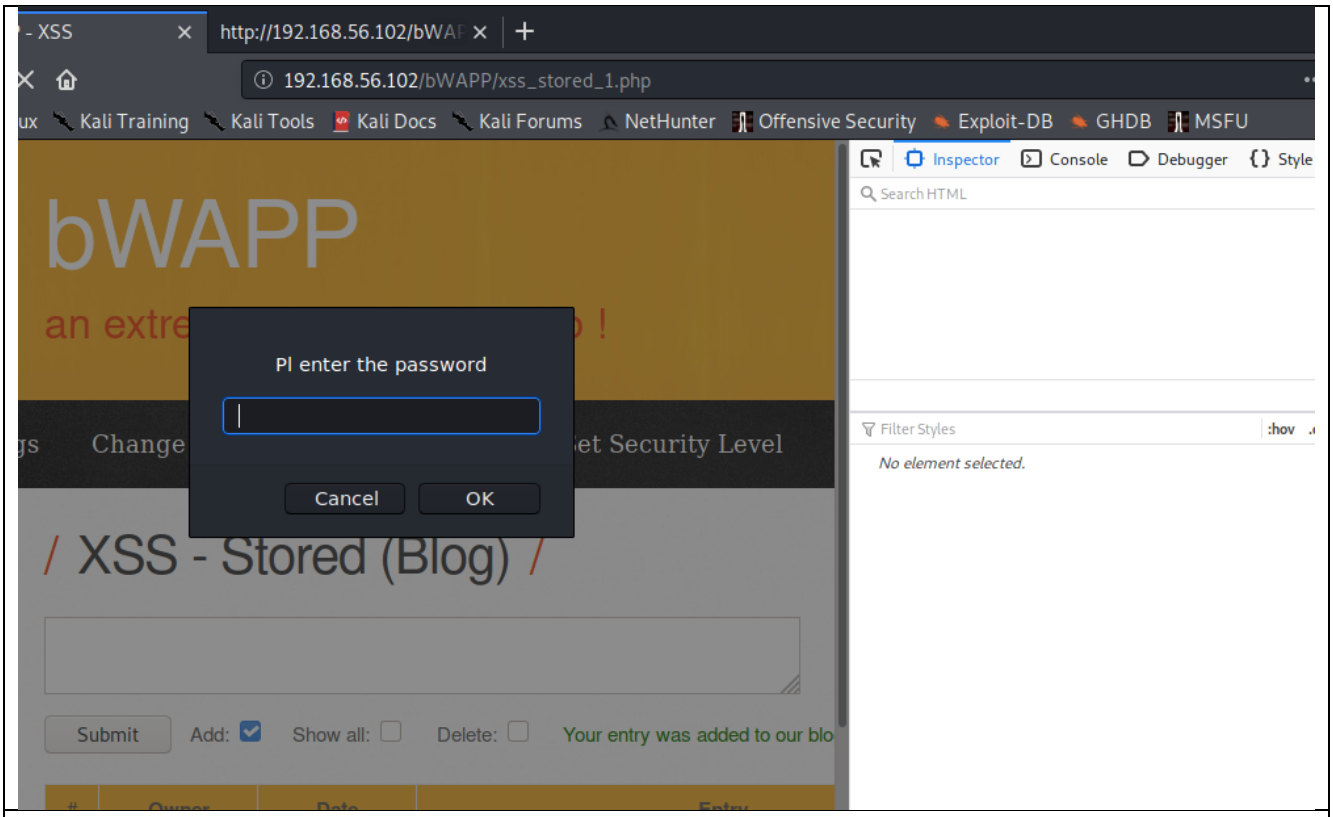
The attacker can use the self-contained nature of stored XSS to engage users who are not logged in at the time of the attack, and take user-equivalent control of the application. He can also initiate malicious interactions with other users.

Proof Of Concept

Step 1: The screenshot below shows an html script input into the blog comment box.



Step 2: As per the screenshot below, the user may input his password which may be redirected to the attacker for his misuse.



Workaround/ Solution

We recommend to:

- Conduct strict input validation of user inputs.
- Ensure that user input is taken as string, not as html tags/script.
- Use whitelisting on both server- and client-side.
- Convert input to html entity.

7. Cross Site Request Forgery (Change Password)

Affected URLs

http://192.168.56.102/bWAPP/csrf_1.php

Description

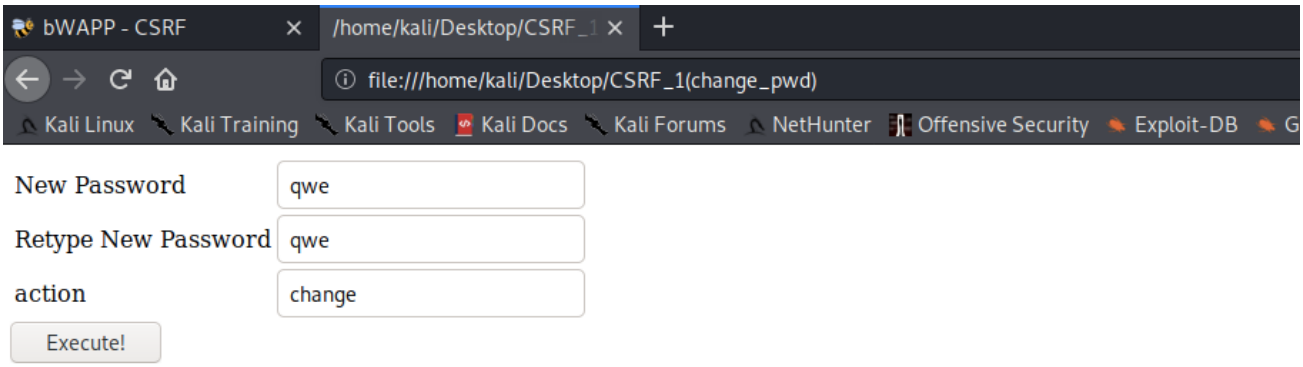
Using social engineering, a crafted URL can be sent to the user by the attacker (placed on his website in form of tiny URL). The URL once clicked will cause inadvertent change of user's credentials.

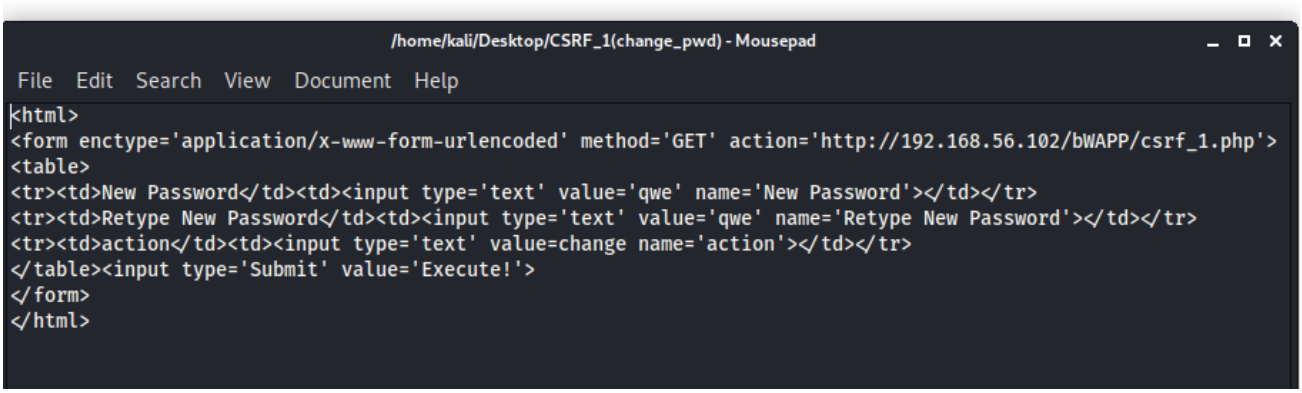
Impact

Using social engineering techniques, the attacker can change user's credentials to his choice and take control of his account.

Proof Of Concept

Step 1: The screenshots below indicate how a simple html code executed unintentionally by the user, allowed the attacker to change user password.





Step 2:

Burp Project Intruder Repeater Window Help

Dashboard | Target | Proxy | Intruder | Repeater | Sequencer | Decoder | Comparer | Extender | Project options | User options

Intercept | HTTP history | WebSockets history | Options

Filter: Hiding CSS, image and general binary content

#	Host	Method	URL	Params	Edited	Status	Length	MIME ty...	Extension	Title	Comr
59	http://192.168.56.102	GET	/bWAPP/csrf_1.php?New+Passw...	✓		200	11860	HTML	php	bWAPP - CSRF	
58	http://192.168.56.102	GET	/bWAPP/csrf_1.php?password_ne...	✓		200	11917	HTML	php	bWAPP - CSRF	
57	http://192.168.56.102	GET	/bWAPP/csrf_1.php			200	11860	HTML	php	bWAPP - CSRF	
56	http://192.168.56.102	POST	/bWAPP/portal.php	✓		302	559	HTML	php		
55	http://192.168.56.102	GET	/bWAPP/portal.php			200	20261	HTML	php	bWAPP - Portal	
54	http://192.168.56.102	POST	/bWAPP/login.php	✓		302	694	HTML	php		
53	http://192.168.56.102	GET	/bWAPP/login.php			200	3419	HTML	php	bWAPP - Login	
52	http://192.168.56.102	POST	/bWAPP/csrf_2.php	✓		302	558	HTML	php		
51	http://192.168.56.102	GET	/bWAPP/csrf_2.php?account=123...	✓		200	11849	HTML	php	bWAPP - CSRF	
41	http://192.168.56.102	GET	/bWAPP/js/html5.js			304	359	script	js		
39	http://192.168.56.102	GET	/bWAPP/csrf_2.php?account=123...	✓		200	11850	HTML	php	bWAPP - CSRF	
37	https://www.google.com	GET	/search?client=firefox-b-e&q=C...	✓							

Request | Response

Raw | Params | Headers | Hex

```
1  GET /bWAPP/csrf_1.php?New+Password=qwe&Retype+New+Password=qwe&action=change HTTP/1.1
2  Host: 192.168.56.102
3  User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0
4  Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
5  Accept-Language: en-US,en;q=0.5
6  Accept-Encoding: gzip, deflate
7  Connection: close
8  Cookie: PHPSESSID=dc110ds9b9h5pn44j27o1nmi56; acopendivids=swingset,jotto,phpbb2,redmine; acgroupswithpersist=nada; security_level=0
9  Upgrade-Insecure-Requests: 1
10
11
```

Workaround/ Solution

We recommed to:

- Ask for old password as confirmation
- Use MFA or OTP
- Use one-time-use anti-CSRF_TOKEN
- Use CAPTCHA
- Use POST method instead of GET method

8. Cross-Site Request Forgery (Transfer Amount)

Affected URLs

http://192.168.56.102/bWAPP/csrf_2.php

Description

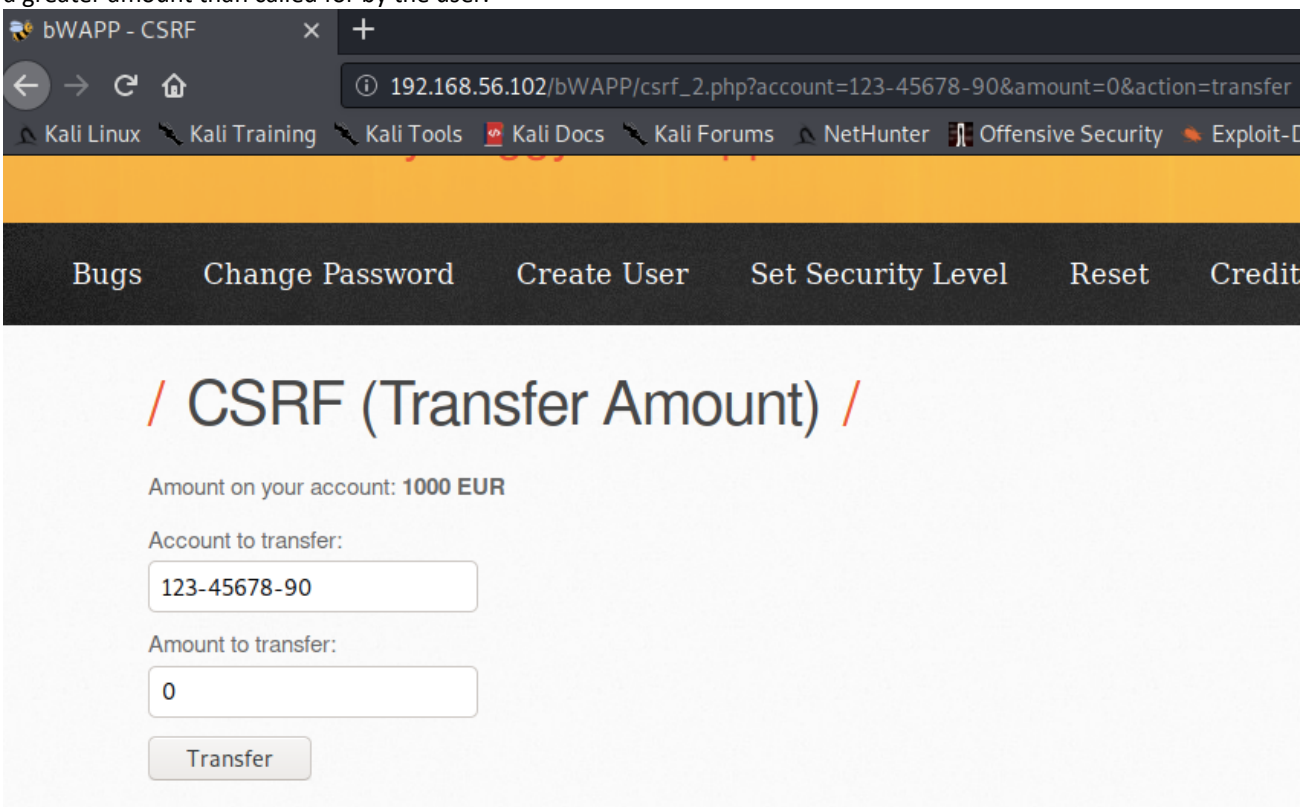
Using social engineering, a crafted URL can be sent to the user by the attacker (placed on his website in form of tiny URL). The URL once clicked will cause unintended actions to be executed.

Impact

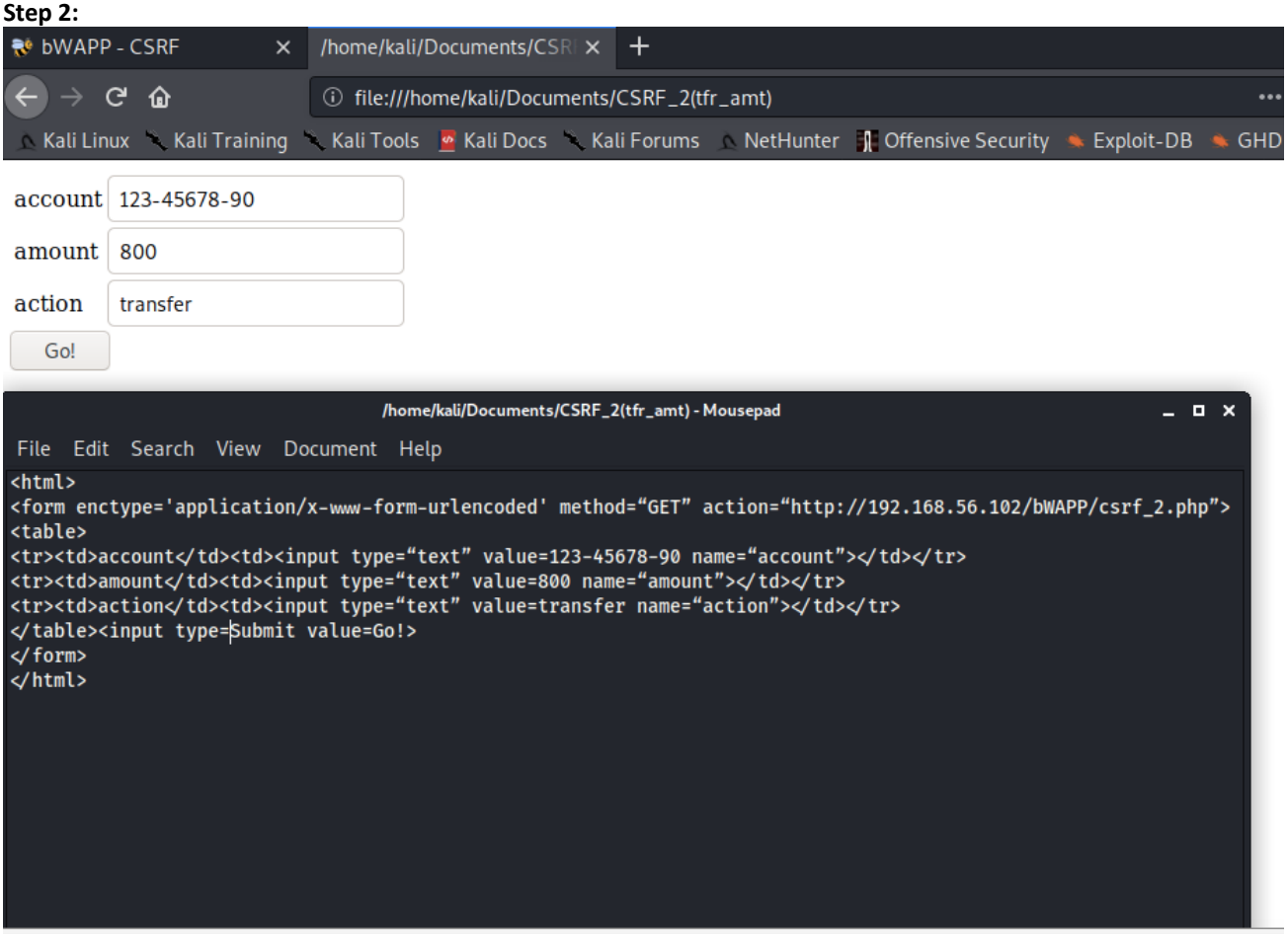
The attacker can cause the user to inadvertently run malicious scripts towards unintended actions like purchasing something, or changing the user credentials.

Proof Of Concept

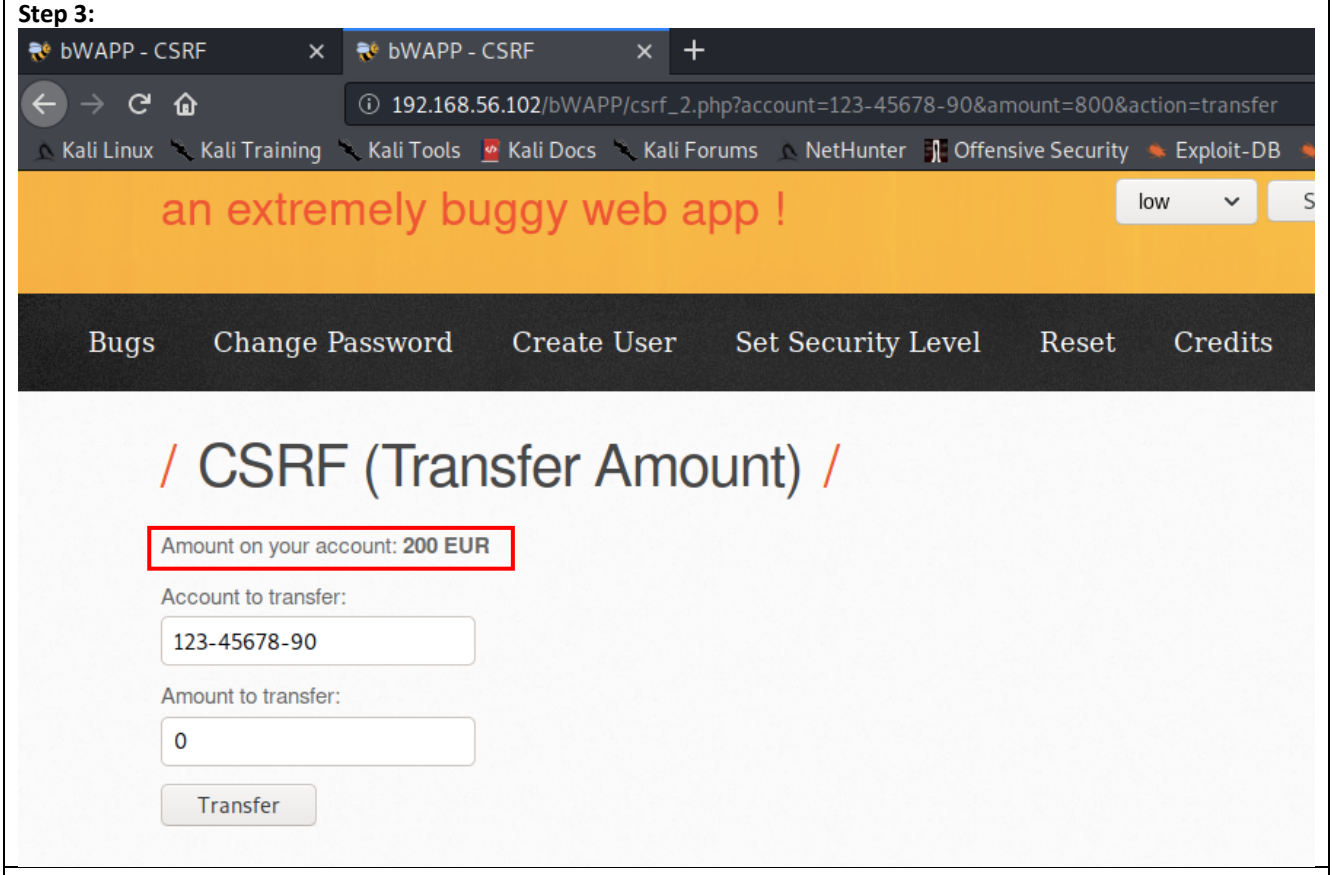
Step 1: The screenshots below indicate how a simple html code executed unintentionally by the user was able to tranfer a greater amount than called for by the user.



Step 2:


bWAPP - CSRF
/home/kali/Documents/CSRF
file:///home/kali/Documents/CSRF_2(tfr_amt)
Kali Linux
Kali Training
Kali Tools
Kali Docs
Kali Forums
NetHunter
Offensive Security
Exploit-DB
GHD
account
123-45678-90
amount
800
action
transfer
Go!
/home/kali/Documents/CSRF_2(tfr_amt) - Mousepad
File
Edit
Search
View
Document
Help
<html>
<form enctype='application/x-www-form-urlencoded' method="GET" action="http://192.168.56.102/bWAPP/csrf_2.php">
<table>
<tr><td>account</td><td><input type="text" value=123-45678-90 name="account"></td></tr>
<tr><td>amount</td><td><input type="text" value=800 name="amount"></td></tr>
<tr><td>action</td><td><input type="text" value=transfer name="action"></td></tr>
</table><input type=submit value=Go!>
</form>
</html>

Step 3:



Workaround/ Solution

We recommed to:

- Use MFA or OTP before final transfer
- Use one-time-use anti-CSRF_TOKEN
- Use CAPTCHA
- Use POST method instead of GET method

9. SQL Injection (Login Form)

Affected URLs

http://192.168.56.102/bWAPP/sqli_3.php

Description

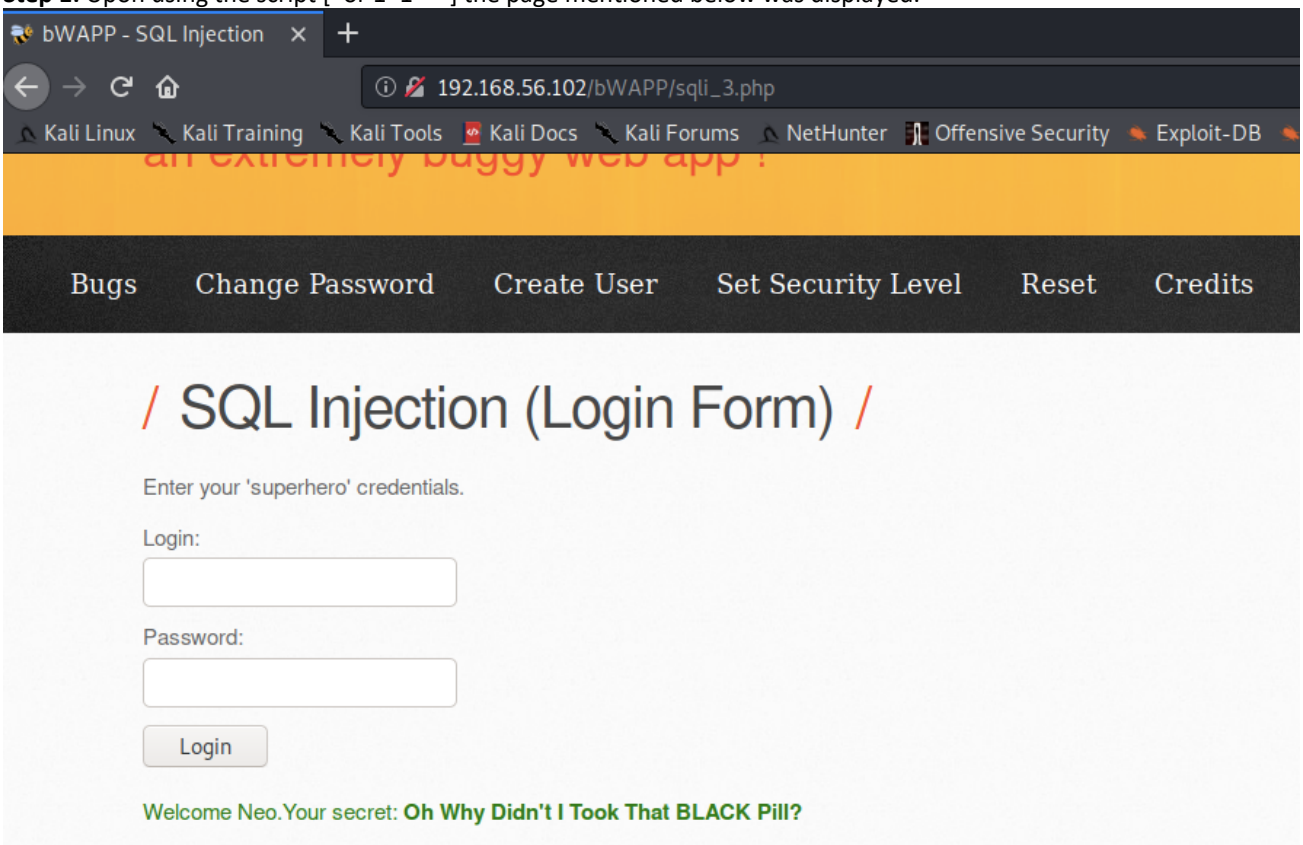
SQL injection attack consists of insertion or “injection” of a SQL query via the input data from the client to the application.

Impact

This attack enables the attacker to read and modify data from the database, execute administration operations on the database

Proof Of Concept

Step 1: Upon using the script [‘ or 1=1 -- -] the page mentioned below was displayed.



Step 2: <Screenshot details here>
<add screenshot here>

Workaround/ Solution

We recommend to:

- Program fields to filter inputs correctly.
- Install defense-in-depth countermeasures, like low-privilege connections to the database server.
- Use prepared statements and stored procedures.
- Whitelist input validation.
- Escape all user-supplied input.