

VulnWebApp (VWA) Security Report

Code Revision: 1.0.0.0
Company: Acme Inc.
Report: VWA20231129
Author: [Hsin-Wen Chang]
Date: [2023/08/30]

VWA Security Report

VWA20233008 - Cross-Site Scripting (XSS) - critical

VWA20233008 - Broken Authentication - High

VWA20233008 - Cryptographic Failures - HIGH

VWA20233010 - Injection - HIGH

VWA20233011 - FALSEPOSITIVE - FALSEPOSITIVE

VWA20233012 - Security Misconfiguration - HIGH

VWA20233003 - Injection - Medium

VWA20233001 - SQLi - Critical

VWA20233005 - Broken Access Control - High

VWA20233006 - Sensitive Data Exposure - High

VWA20233007 - Insecure Deserialization - High

VWA20233009 - Security Misconfiguration - Low

VWA20233002 - FALSEPOSITIVE - LOW

VWA20233002 - Vulnerable and Outdated Components - HIGH

VWA Security Report

VWA20233008 – Cross-Site Scripting (XSS) – **Critical**

Vulnerability Exploited: Cross-Site Scripting (XSS)

Severity: [Critical, High, Medium, Low, Info]

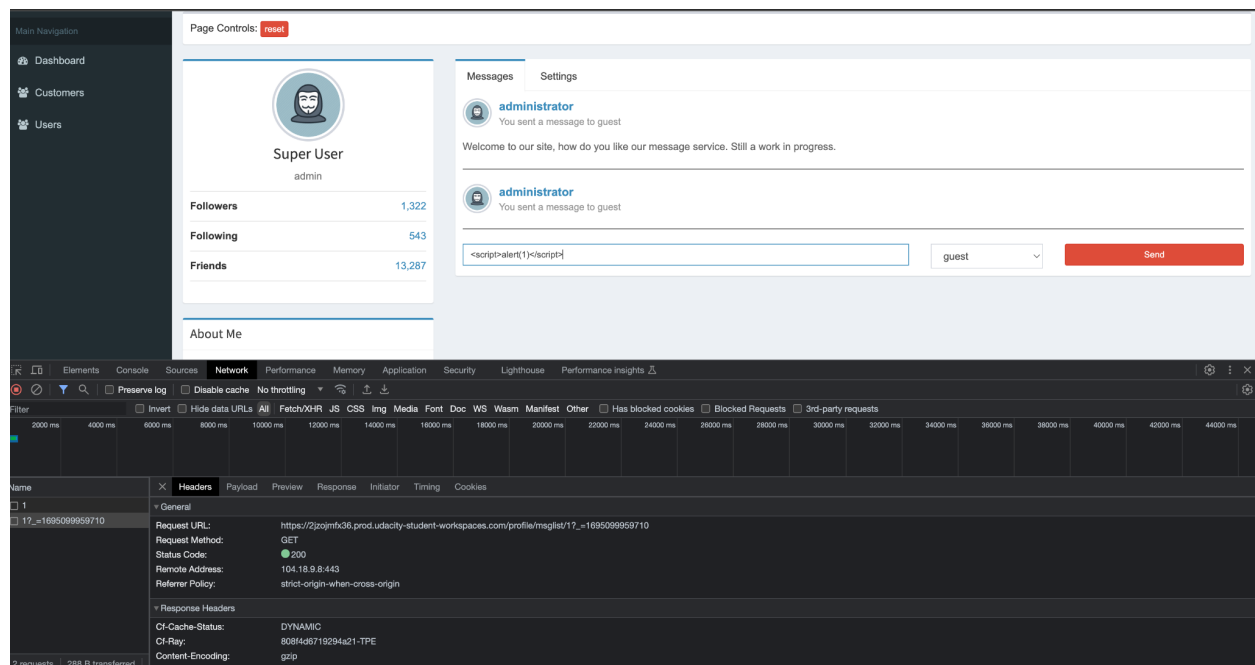
System: VWA Web Application

Vulnerability Explanation:

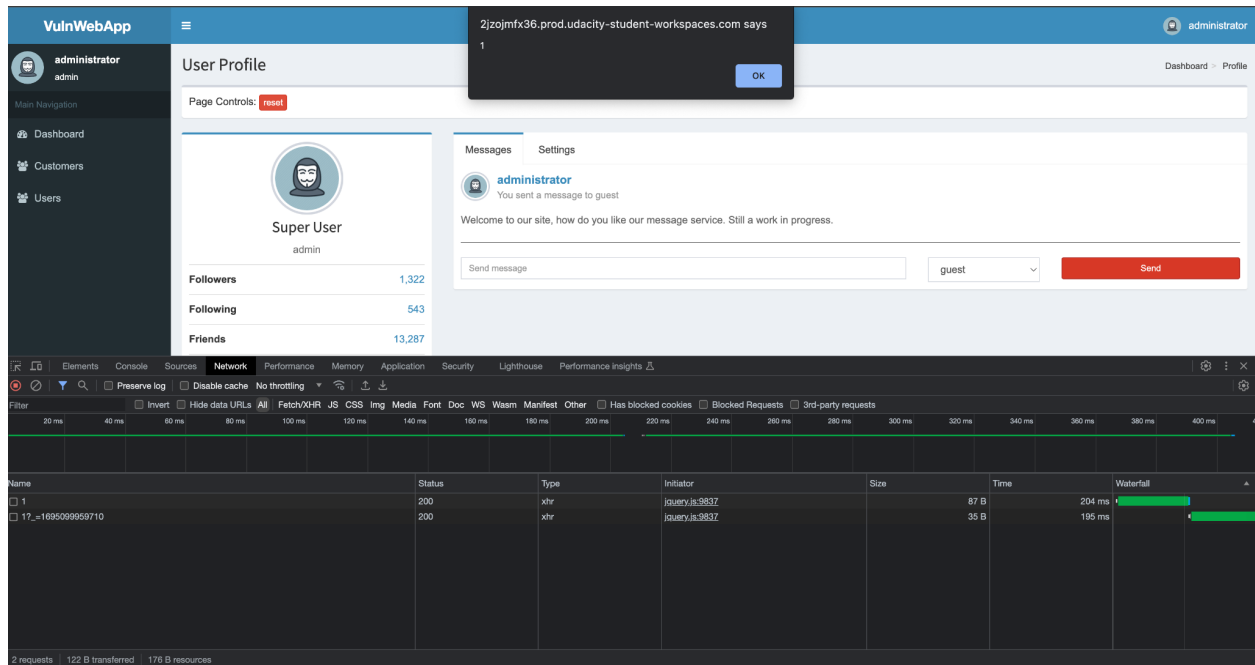
In the XSS Area we have an internal Chat service, this system is not doing any sanitizing and allowing script tags with java-script code and is executable on the client side.

Vulnerability Walk-thru:

1. Go to the XSS Section
2. Next I check to see if Direct Chat 1 was exploitable to XSS
3. Using the following XSS I was able to inject a javascript alert in Direct Chat 1 "<script>alert(1)</script>".



VWA Security Report



Recommendations:

Sanitizing all inputs is the best way to safeguard your application from XSS.

VWA Security Report

VWA20233009 – Broken Authentication – **High**

Vulnerability Exploited: Broken Authentication

Severity: [Critical, High, Medium, Low, Info]

System: VWA Web Application

Vulnerability Explanation:

This occurs when an attacker attacks an online application using common/harvest usernames as well as passwords in the hopes of discovering an account that is using a combination from the list. As more applications require supplementary information, such as multi-factor authentication (MFA), this strategy is gradually becoming outdated.

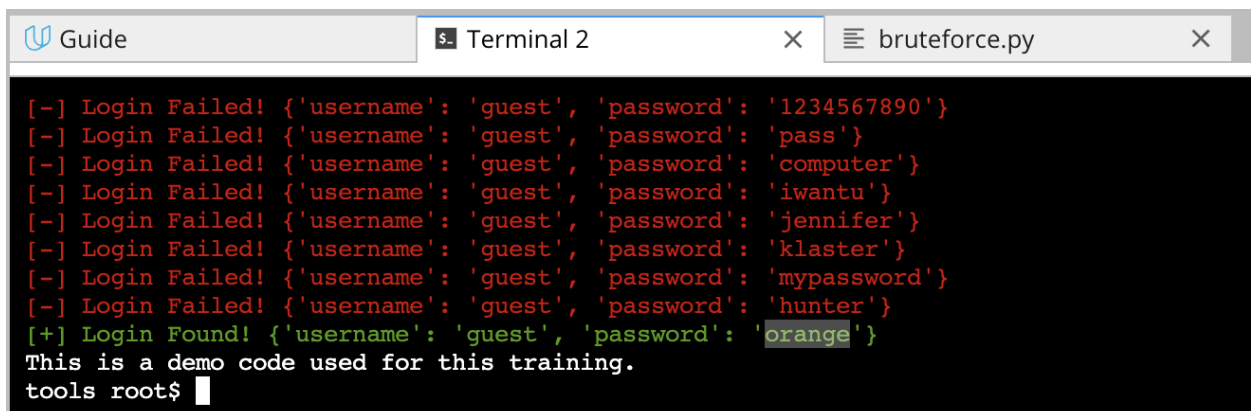
Vulnerability Walk-thru:

1. Use the bruteforce.py tool for brute force attacks

```
python bruteforce.py -U test-username.txt -P test-password.txt -d  
username=^USR^:password=^PWD^ -m POST -f "Login Failed"
```

<http://localhost:3000/login>

2. You will retrieve username and password to log in to the WebApp



```
[+] Login Found! {'username': 'guest', 'password': 'orange'}  
This is a demo code used for this training.  
tools root$
```

VWA Security Report

Recommendations:

Multi-Factor Authentication (MFA) - One of the greatest ways to protect accounts is to use multi-factor authentication (MFA). Even if a user's credentials are stolen or made public online due to an exploit or data breach, the user account is still secure because the attacker will not be able to log in without the MFA.

VWA Security Report

VWA20230830 - Cryptographic Failures - **Medium**

Vulnerability Exploited: Cryptographic Failures

Severity:[Critical, High, Medium, Low, Info]

System: VWA Web Application

Vulnerability Explanation:

Using deprecated cryptographic functions MD5 in password and is not salted or peppered

Vulnerability Walk-thru:

Recommendations:

MD5 should never be used in passwords, we need to migrate to bcrypt or another hash that is slow hashing.

https://cheatsheetseries.owasp.org/cheatsheets/Password_Storage_Cheat_Sheet.html

VWA Security Report

VWA20233010 - Injection - High

Vulnerability Exploited: Injection

Severity: [Critical, High, Medium, Low, Info]

System: VWA Web Application

Vulnerability Explanation:

By using deprecated cryptographic functions MD5 in password will expose to an attacker's perspective, the optimal place to inject malicious content is in an area that is displayed to either many users or particularly interesting users. Interesting users typically have elevated privileges in the application or interact with sensitive data that is valuable to the attacker. The attacker can engage in a number of malicious actions after injecting the malicious script. Private data, including cookies that may contain session information, could be sent from the victim's computer to the attacker by the attacker. When the victim has administrator rights to control a website, the attacker may make malicious requests to that website on the victim's behalf, which might be extremely risky for that website. To compromise the victim's account on a trustworthy website, an attacker may use phishing attacks to imitate reputable websites and deceive the victim into entering their password. Finally, the script might take advantage of a flaw in the victim's web browser to potentially take control of their computer, a technique known as "drive-by hacking."

Vulnerability Walk-thru:

Recommendations:

https://bandit.readthedocs.io/en/1.7.0/plugins/b703_django_mark_safe.html

VWA Security Report

VWA20233011 - FALSEPOSITIVE - FALSEPOSITIVE

Vulnerability Exploited: FALSEPOSITIVEComponents

Severity: [Critical, High, Medium, Low, Info]

System: VWA Web Application

Vulnerability Explanation:

This problem is low severity. Just a warning for those who might not be aware of the potential security concerns surrounding the library.

Vulnerability Walk-thru:

Recommendations:

By excluding the warning with # nsec B404 on the relevant line, or by altering scan behavior in your Bandit settings, you can disable the alert and have [B602:](#)

[subprocess_popen_with_shell_equals_true](#) and [B603: subprocess_without_shell_equals_true](#) both turned on, which are where actual security issues could happen.

https://docs.openstack.org/bandit/1.4.0/blacklists/blacklist_imports.html#b404-import-subprocess

VWA Security Report

VWA20233012 - Security Misconfiguration - High

Vulnerability Exploited: Security Misconfiguration

Severity: [Critical, High, Medium, Low, Info]

System: VWA Web Application

Vulnerability Explanation:

The product employs XML documents and permits the definition of their structure using a Document Type Definition (DTD), however it does not adequately regulate the quantity of recursive entity definitions.

Vulnerability Walk-thru:

Recommendations:

The majority of this is based on Christian Heimes' defusedxml work, which may be found at: <https://pypi.python.org/pypi/defusedxml/#defusedxml-sax>

It is well known that using different XML methods to parse untrusted XML input leaves one open to XML assaults. The defusedxml equivalents of methods should be used in their place.

VWA Security Report

VWA20233003 - Injection - Medium

Vulnerability Exploited: Injection

Severity: [Critical, High, Medium, Low, Info]

System: VWA Web Application

Vulnerability Explanation:

Possible SQL injection vector through string-based query construction.

Vulnerability Walk-thru:

SQL query insertion or "injection" via input data provided to an application. It is a typical assault method.

Recommendations:

Additionally, this plugin test will look to see if the identified string is used with the execute or executemany Python DBAPI standard methods. If so, there is a reported MEDIUM problem.

VWA Security Report

VWA20233004 - **SQLi** - **Critical**

Vulnerability Exploited: **SQLi**

Severity: [**Critical**, High, Medium, Low, Info]

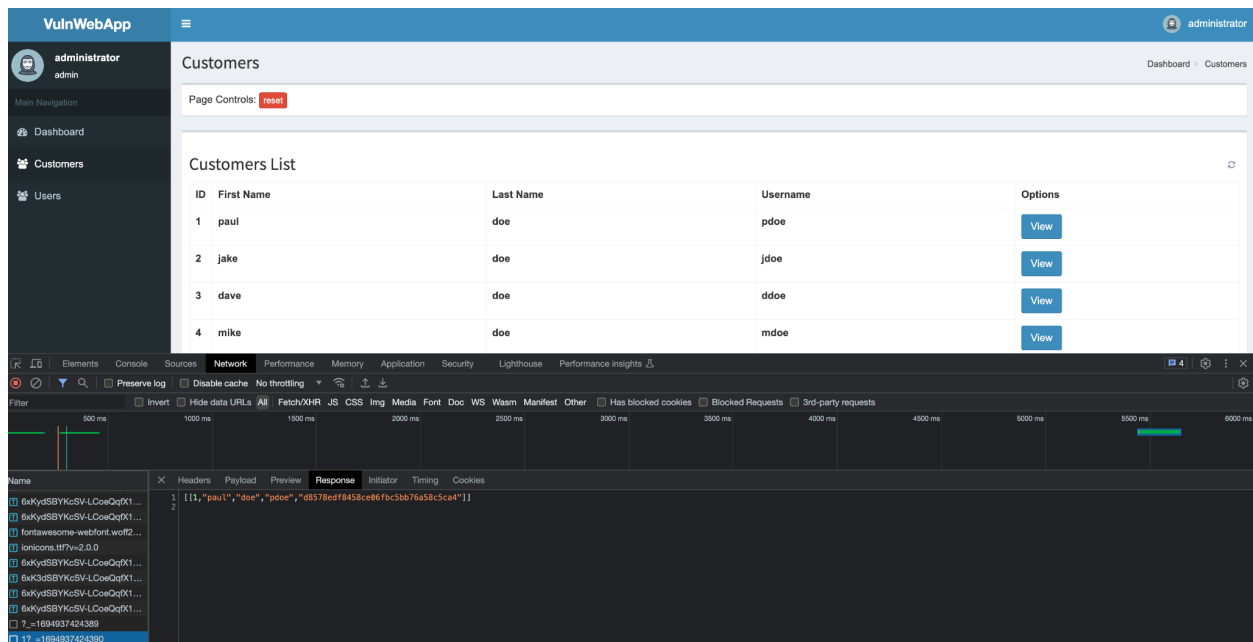
System: VWA Web Application

Vulnerability Explanation:

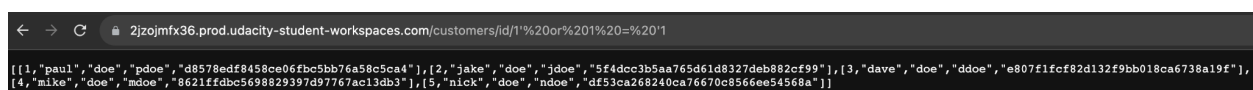
It's crucial to keep in mind that for SQL Injection to be effective, an attacker must first escape the current SQL query. This will cause the database to execute malicious code.

Vulnerability Walk-thru:

1. After we manipulated the Cookie gain admin rights.
2. Heading to Customers page hit view button.
3. Hit xhr type file go to Resonse tab then open new window.



4. We can SQLi customers page by insert ' or 1 = '1 at the end of /customers/id/1 to retrieve full customers information.



VWA Security Report

Recommendations:

- Use Parameterized Queries - Due to the fact that all variables are constrained to the data type, this technique prevents SQL Injection the best. By doing this, harmful code won't be able to escape the SQL code and won't be able to run.
- Sanitize all inputs - You should always sanitize all inputs before using them, in my opinion. This will stop dangerous code from running in your code's other sections as well as in SQL queries.
- For a full list of prevention methods please visit [OWASP Cheatsheet on SQLi Prevention](#).

VWA20233005 – Broken Access Control – **High**

VWA Security Report

Vulnerability Exploited: Broken Access Control

Severity: [Critical, High, Medium, Low, Info]

System: VWA Web Application

Vulnerability Explanation:

We can see that one of the two request objects contains an async call to the server to retrieve the user's current information. By altering the initial request, we can now see that the API used to retrieve the user information is simply returning the data rather than properly validating authentication to see if it is permitted for our user.

Vulnerability Walk-thru:

5. After we manipulated the Cookie gain admin rights.
6. Heading to Customers page hit view button.
7. Hit xhr type file go to Resonse tab then open new window.

The screenshot displays the VWA Web Application interface. The top navigation bar shows the user is logged in as 'administrator'. The main content area is titled 'Customers' and contains a 'Customers List' table. The table has columns for ID, First Name, Last Name, Username, and Options. The data rows are:

ID	First Name	Last Name	Username	Options
1	paul	doe	pdoe	View
2	jake	doe	jdoe	View
3	dave	doe	ddoe	View
4	mike	doe	mdoe	View
5	nick	doe	ndoe	View

Below the table, the browser's developer tools are open, showing the 'Network' tab. A request to 'https://2zqjmhx36.prod.udacity-student-workspaces.com/customers/id/2?_id=1694917616099' is selected. The 'Headers' tab is active, showing the following details:

- Request URL: https://2zqjmhx36.prod.udacity-student-workspaces.com/customers/id/2?_id=1694917616099
- Request Method: GET
- Status Code: 200
- Remote Address: 104.18.9.8:443
- Referer Policy: strict-origin-when-cross-origin
- Response Headers: CF-Cache-Status: DYNAMIC, CF-Ray: 807de864be464a91-TPE, Content-Encoding: gzip
- Content-Security-Policy: frame-ancestors 'self' https://*.udacity.com https://*.udacity-student-workspaces.com

8. You can see by modifying customers/id/2 we can direct retrieve the user information.

VWA Security Report

← → ↻ 2jz0jmf36.prod.udacity-student-workspaces.com/customers/id/1?_=1694917616099

```
[[1,"paul","doe","pdoe","d8578edf8458ce06fbc5bb76a58c5ca4"]]
```

← → ↻ 2jz0jmf36.prod.udacity-student-workspaces.com/customers/id/3?_=1694917616099

```
[[3,"dave","doe","ddoe","e807f1fcf82d132f9bb018ca6738a19f"]]
```

Same as the Users page:

The screenshot shows the VulnWebApp interface. The left sidebar has a 'Users' link. The main content area is titled 'Users' and contains a 'Users List' table. The table has columns for ID, First Name, Last Name, Username, and Options. It lists two users: 'Super' (User, administrator) and 'John' (Doe, guest). Below the table, there is a copyright notice and version information. At the bottom, a network request is visible in the browser's developer tools, showing a response with the same JSON data as the previous screenshots.

ID	First Name	Last Name	Username	Options
1	Super	User	administrator	View
2	John	Doe	guest	View

Copyright © 2020 Test Company LLC. All rights reserved. Ver. 1.0.0.0

← → ↻ 2jz0jmf36.prod.udacity-student-workspaces.com/users/id/1?_=1694937218644

```
[[1,"Super","User","administrator"]]
```

← → ↻ 2jz0jmf36.prod.udacity-student-workspaces.com/users/id/2?_=1694937218644

```
[[2,"John","Doe","guest"]]
```

Recommendations:

- Rate Limit Data - You can prevent users from scraping all of the data from your web application by rate-limiting their access to the data on the site.

VWA Security Report

- Revalidate all secure pages - Ensure that all secure pages and endpoints have undergone adequate testing to confirm that access control is operating as expected.
- Deny Access for Non-Public Pages by Default- We should have general rules that automatically refuse access to non-public pages and demand validation in order to access these pages.
- Log Access Failures - You should keep track of not only all access errors, but also IP addresses and users who have a large number of them.

VWA Security Report

VWA20233006 - Sensitive Data Exposure - **High**

Vulnerability Exploited: Sensitive Data Exposure

Severity: [Critical, High, Medium, Low, Info]

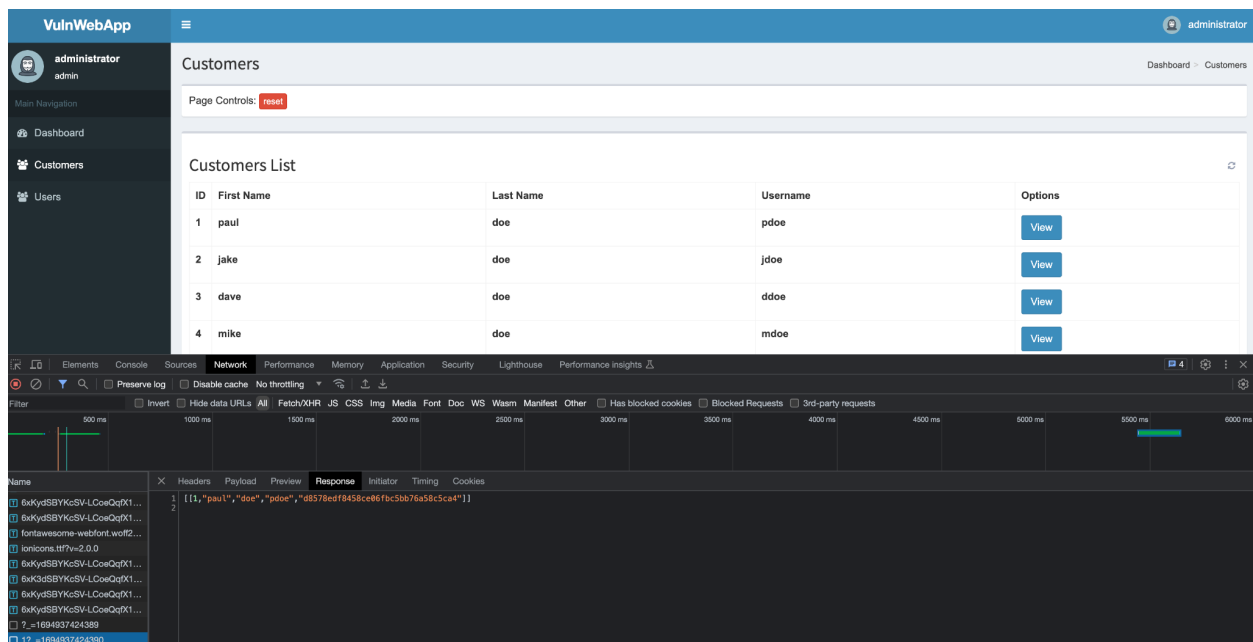
System: VWA Web Application

Vulnerability Explanation:

To find out the hash type is MD5, we click on Hash ID. The hash is then pasted once more along with the hash type after selecting Hash Cracking. Since we now know the hash type for the second password, we can proceed directly to the hash cracking to find the password.

Vulnerability Walk-thru:

1. After we manipulated the Cookie gain admin rights.
2. Heading to Customers page hit view button.
3. Hit xhr type file go to Resonse tab then the passwords will be exposed. This is considered a security breach

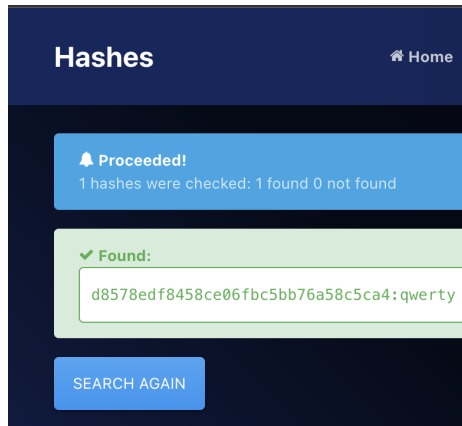


4. Copy paste the hash use hashid.py to find out the hash type is MD5

VWA Security Report

```
tools root$ python hashid.py 5f4dcc3b5aa765d61d8327deb882cf99
Analyzing '5f4dcc3b5aa765d61d8327deb882cf99'
[+] MD5
This is a demo version of the hashid.py for this training, for the full version please visit https://github.com/psypanda/hashID
```

5. Use online hashcracker: <https://hashes.com/en/decrypt/hash> to crack the password.



Recommendations :

- Check code using Sensitive Data - Must ensure every piece of code using sensitive data is secure and operating as intended.
- Strong Hashing - Use strong adaptive, salted hashing algorithms with a delay factor when creating passwords.
- Limit storing sensitive data to only what is needed - By storing little more sensitive data than is necessary, can lower the risk of web application faces from storing sensitive data..
- Encrypt all data in transit and at rest - By using HTTPS and a strong cipher, you should make sure that all data is secured both in transit and at rest.

VWA Security Report

VWA20233007 – Insecure Deserialization – High

Vulnerability Exploited: Insecure Deserialization

Severity: [Critical, High, Medium, Low, Info]

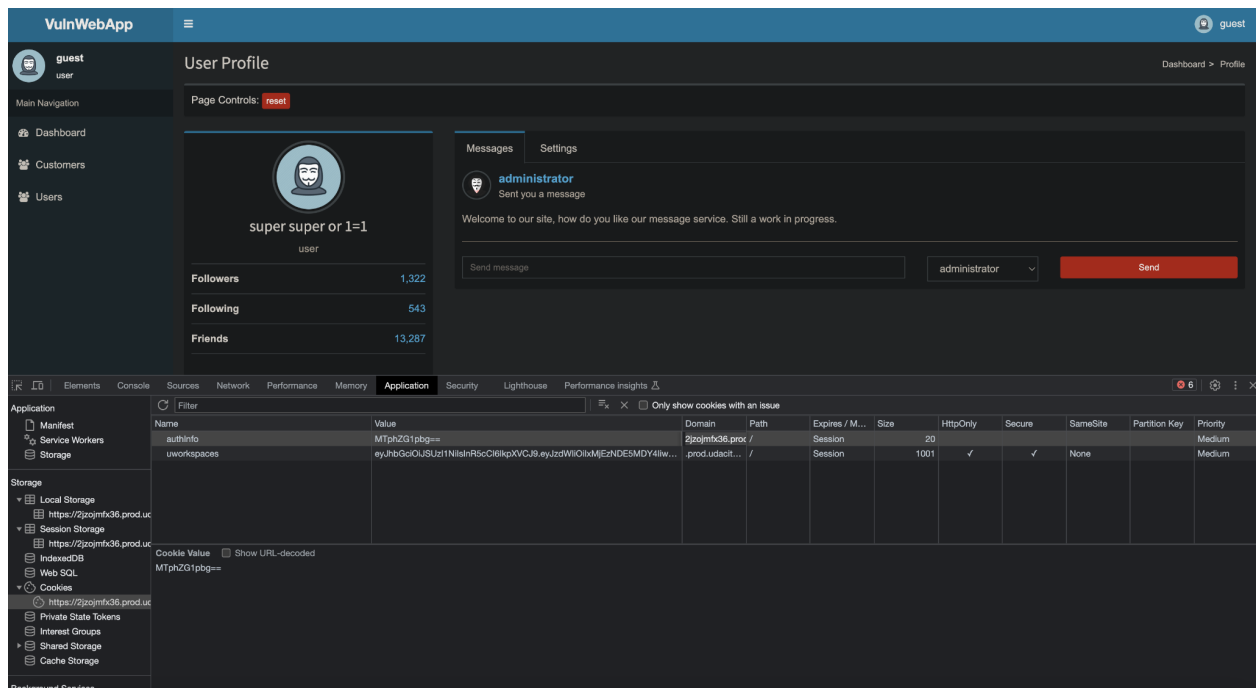
System: VWA Web Application

Vulnerability Explanation:

Modifying cookie to have admin access in the web app.

Vulnerability Walk-thru:

1 .Log in as user and copy the cookie



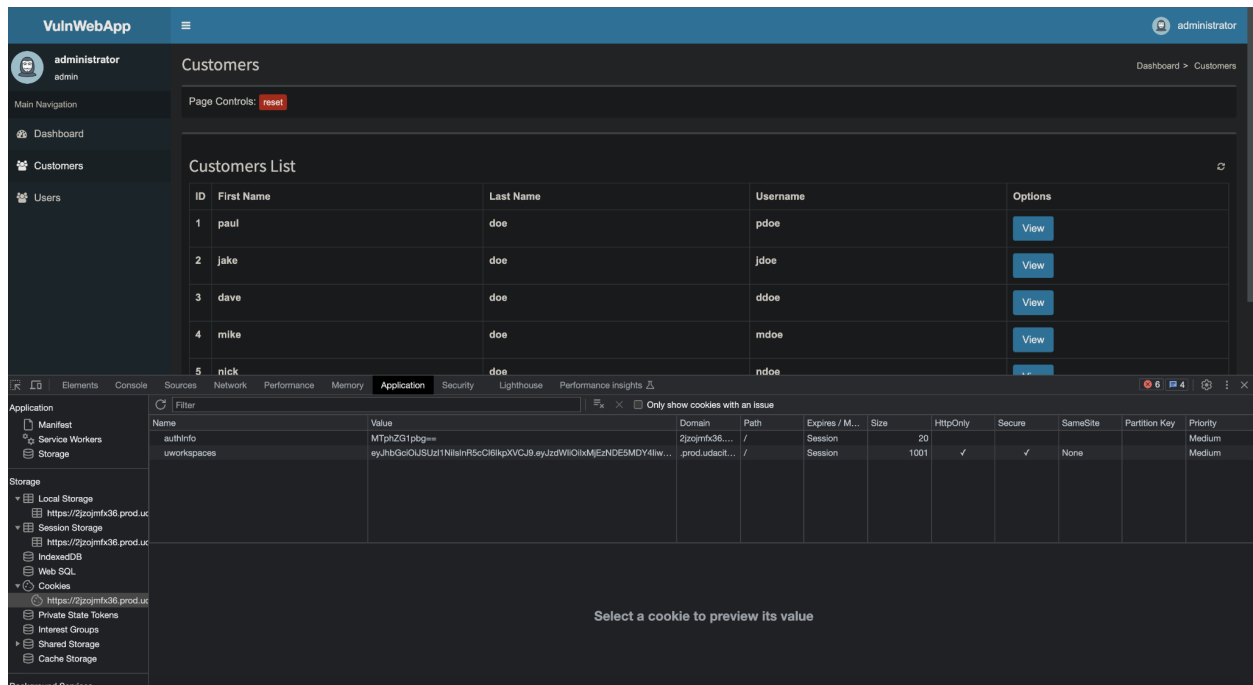
2. Decode the cookie with base64 & encode cookie

```
tools root$ python performbase64.py -d Mjplc2Vy
2:user
tools root$
```

VWA Security Report

```
tools root$ python performbase64.py 2:user
Mjp1c2Vy
tools root$ python performbase64.py 1:admin
MTphZG1pbG==
```

3. Edit cookie & refresh browser gain administrator role



Recommendations:

- Integrity Checks - A digital signature that may be used afterward to confirm that the data has not been changed can be created by using a hashing function.
- Strict Data Type - This can assist in safeguarding your data from expected data types by employing stringent data types.

VWA Security Report

VWA20233009 - Security Misconfiguration - **Critical**

Vulnerability Exploited: Security Misconfiguration

Severity: [Critical, High, Medium, Low, Info]

System: VWA Web Application

Vulnerability Explanation:

Hard-coded passwords frequently result in substantial authentication failures that are challenging for system administrators to identify. The administrator might be compelled to completely disable the product because once discovered, it can be challenging to fix. The back-end credentials could simply be hard-coded into the front-end product by the coder. The password might potentially be extracted by any user of that program. Given how easy it is to extract a password from a binary, client-side system with hard-coded passwords are considerably more dangerous.

Vulnerability Walk-thru:

Recommendations:

We should never use default passwords.

https://bandit.readthedocs.io/en/1.7.0/plugins/b106_hardcoded_password_funcarg.html

<https://cwe.mitre.org/data/definitions/259.html>

VWA Security Report

VWA20233001 - FALSEPOSITIVE - Low

Vulnerability Exploited: FALSEPOSITIVE

Severity: [Critical, High, Medium, Low, Info]

System: VWA Web Application

Vulnerability Explanation:

This is a false positive, the code is generate a random resetcode.

Vulnerability Walk-thru:

Recommendations: This is a false positive, the code is generate a random resetcode.

VWA Security Report

VWA20233002 - Vulnerable and Outdated Components -
High

Vulnerability Exploited: Vulnerable and Outdated
Components

Severity: [Critical, High, Medium, Low, Info]

System: VWA Web Application

Vulnerability Explanation:

This plugin test is part of a family of tests built to check for process spawning and warn appropriately.

Specifically, this test looks for the spawning of a subprocess without the use of a command shell.

Vulnerability Walk-thru:

Recommendations:

This type of subprocess invocation is not vulnerable to shell injection attacks, but care should still be taken to ensure validity of input.

https://docs.openstack.org/bandit/1.4.0/plugins/subprocess_without_shell_equals_true.html