VulnWebApp (VWA) Security Report

Code Revision: 1.0.0.0
Company: Acme Inc.
Report: VWAYYMMDD
Author: [Shamima Sultana]
Date: [1/11/21]

VWA Security Report

VWA Security Report

VWA21-1-14-1 – A2:2017 - Broken Authentication - MEDIUM

Vulnerability Exploited: A2-Broken Authentication

Severity: [Medium]

System: VWA Web Application

Vulnerability Explanation:

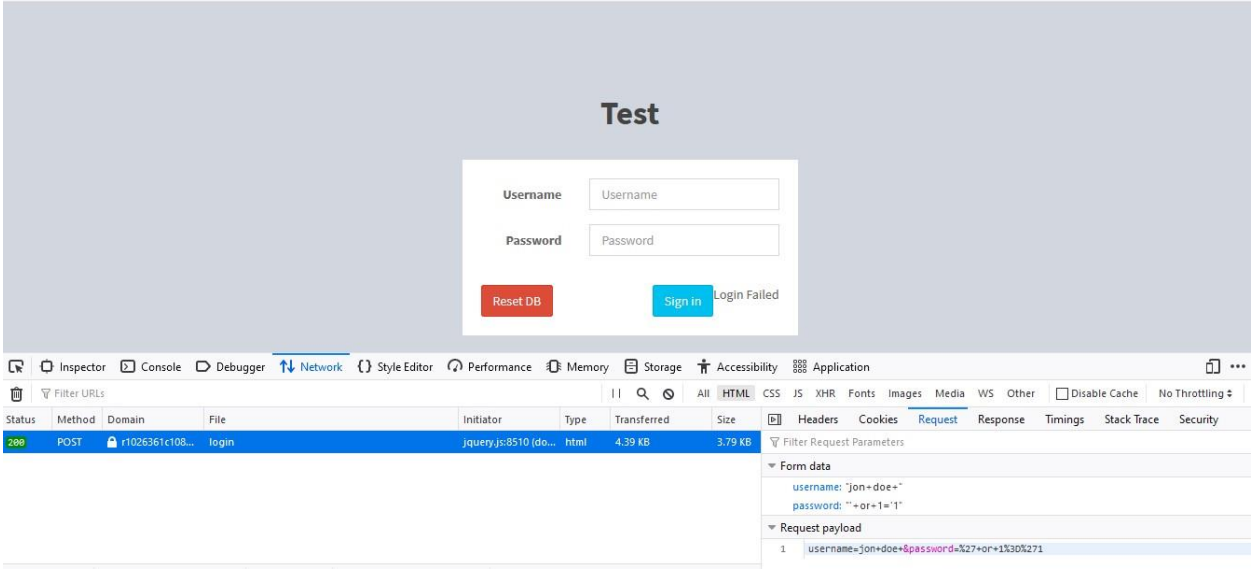
Here brute force method is used in broken authentication. By using large number of common username and password through brute-force login tools against a web application an attacker tries to find the combination from the list.

Vulnerability Walk-thru:

A detailed walk-thru on how to reproduce this vulnerability with screenshots.

1. For login go to the url: https://r1026361c1084570xjupyterlltj21rh6-3000.udacity-student-workspaces.com/login

2. Credentials for login is unknown and SQL injection is failed

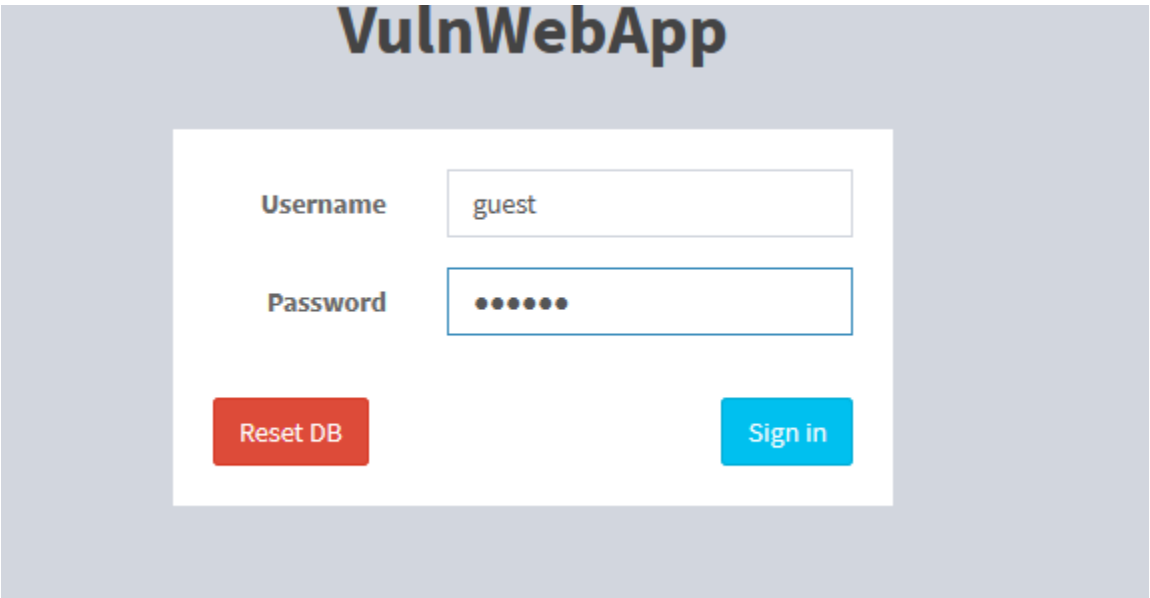


3. From tools option in workspace using bruteforce.py to get match credentials for login.

VWA Security Report

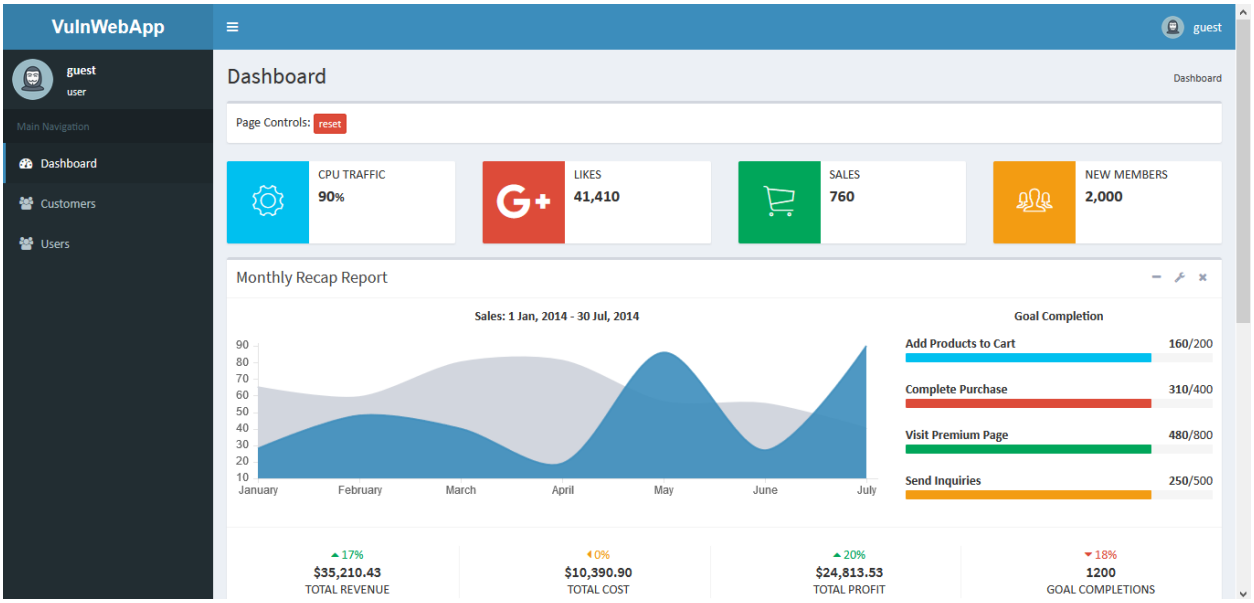
4. python bruteforce.py -U top-usernames-shortlist.txt -P top-passwords-shortlist.txt -d username=^USR^:password=^PWD^ -m 'post' -f 'Login Failed' https://r1026361c1084570xjupyterlltj21rh6-3000.udacity-student-workspaces.com/login

5.Using bruteforce.py tool got the matched credentials for login where user: guest password: orange



6.Got the access of test page.

VWA Security Report



Recommendations:

- ✓ https://cheatsheetseries.owasp.org/cheatsheets/Credential_Stuffing_Prevention_Cheat_Sheet.html
- ✓ https://github.com/danielmiessler/SecLists/tree/master/Passwords

Best Practice:

- **Multi-Factor Authentication (MFA)** – MFA prevents automated attacks like credential stuffing, brute force, and stolen credential re-use attacks. So, using MFA would be the best methods for protecting accounts, even if a user credentials are exposed to the internet either by data breaches or stolen using some type of exploit the user account is still safe because the attacker will not be able to login without the MFA.
- **Captcha** - Using a captcha is a good way to slow down or even stop attackers from using some sort of automation on your web application.

VWA Security Report

- **Weak-password check –** Attempting weak-password check by using top 1000 worst passwords against new or changed password.
- **Failed Login Attempts** - By using failed login attempts you can auto lock accounts or even block IP address that appears to be using some sort of brute forcing against a user account.

VWA Security Report

VWA21-1-14-2 – A7:2017 - Cross-Site Scripting (XSS) - MEDIUM

Vulnerability Exploited: A7-Cross-Site Scripting (XSS)

Severity: [Medium]

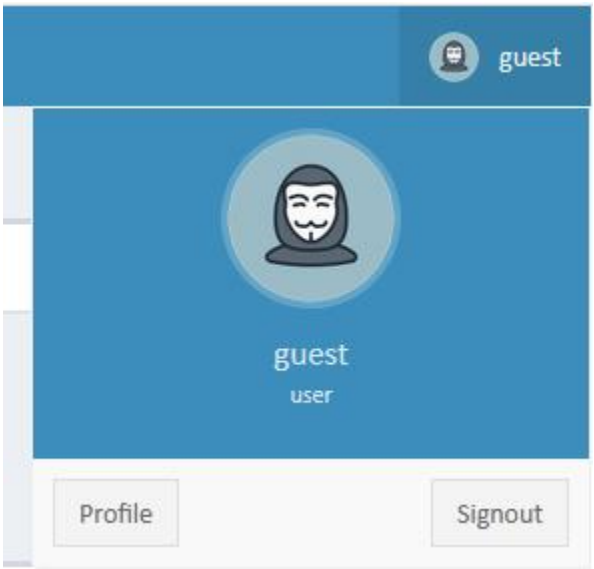
System: VWA Web Application

Vulnerability Explanation:

Cross-Site Scripting (XSS) is used to run unintended code on a victim's local machine it permits attackers to perform scripts in the victim's browser which can hijack user sessions, deface web sites, or redirect the user to malicious sites

Vulnerability Walk-thru:

1. Go to the profile page

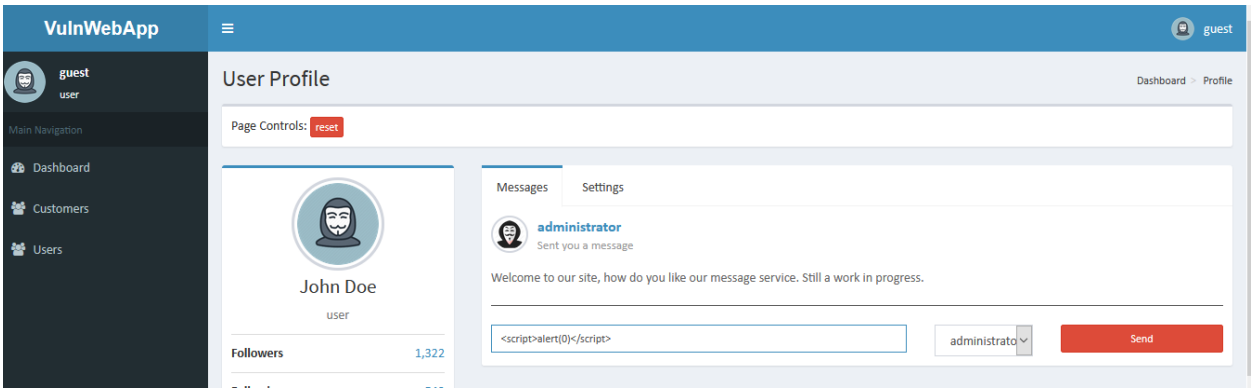


VWA Security Report

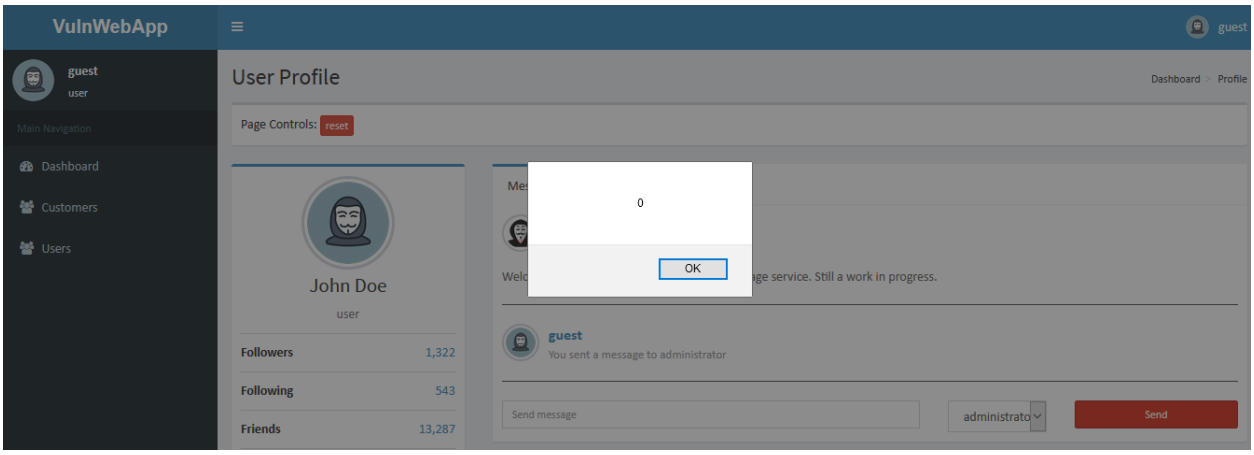
Stored XSS

In this type of attack, a payload is sent to the web application where it is saved in the DB and then when called from the database it is executed on the target local machine

2. Injecting <script>alert(0)</script> in send message option



3.Showing the executed alert in profile page

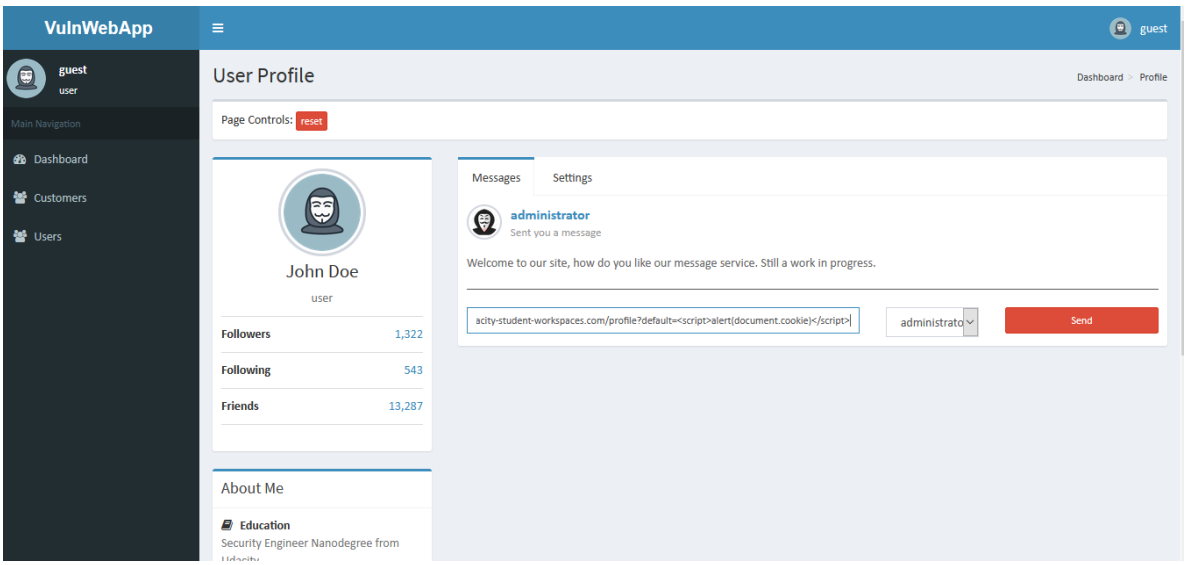


VWA Security Report

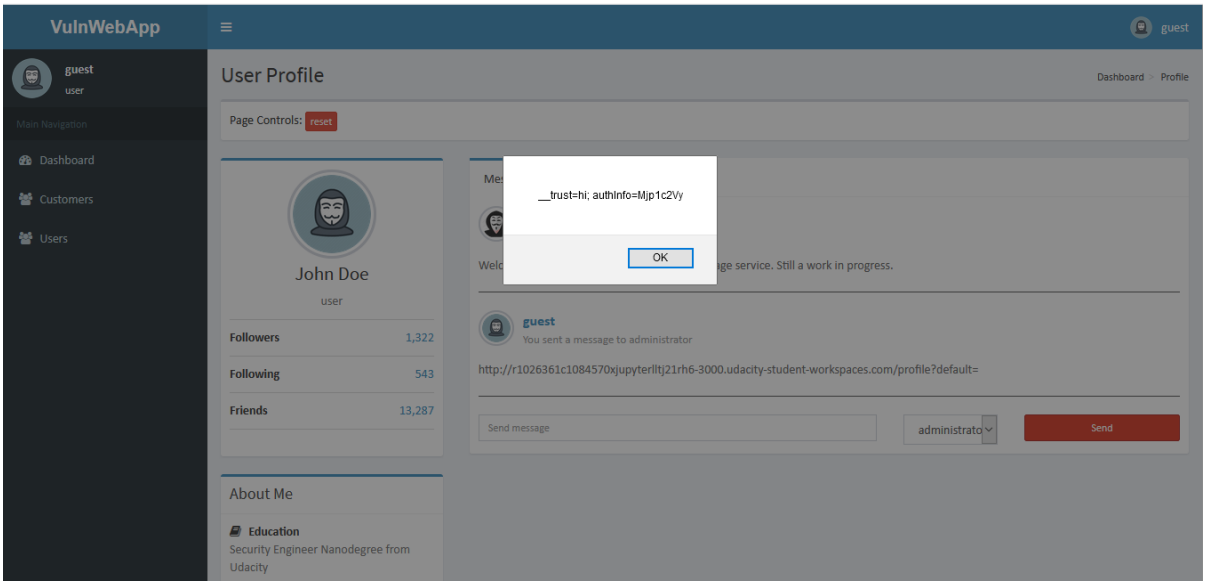
DOM XSS:

This type of attack is considered a more advanced type of XSS and is when the vulnerability appears up in the Document Object Model (DOM) rather than in the html page.

4. Injecting http://r1026361c1084570xjupyterlltj21rh6-3000.udacity-student-workspaces.com/profile?default=<script>alert(document.cookie)</script>



5. Vulnerability appearing up in document object model.



VWA Security Report

Recommendations:

- ✓ https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html
- ✓ https://cheatsheetseries.owasp.org/cheatsheets/DOM_based_XSS_Prevention_Cheat_Sheet.html

Best Practice:

- **Using Frameworks –** It automatically escape XSS by design, such as the latest Ruby on Rails, React JS. Learn the limitations of each framework's XSS protection and appropriately handle the use cases which are not covered
- **Sanitizing all inputs** – All input that is entered by the user should be precisely validated, because the user's input may find its way to the output.
- **Filtering inputs** - The idea of the filtering is to search for risky keywords in the user's input and remove them or replace them by empty strings.
- **Characters escaping** – You can use appropriate characters to change them by special codes. There exist appropriate libraries to escape the characters.

VWA Security Report

VWA21-1-14-3 – A8:2017 - Insecure Deserialization - LOW

Vulnerability Exploited: A8 - Insecure Deserialization

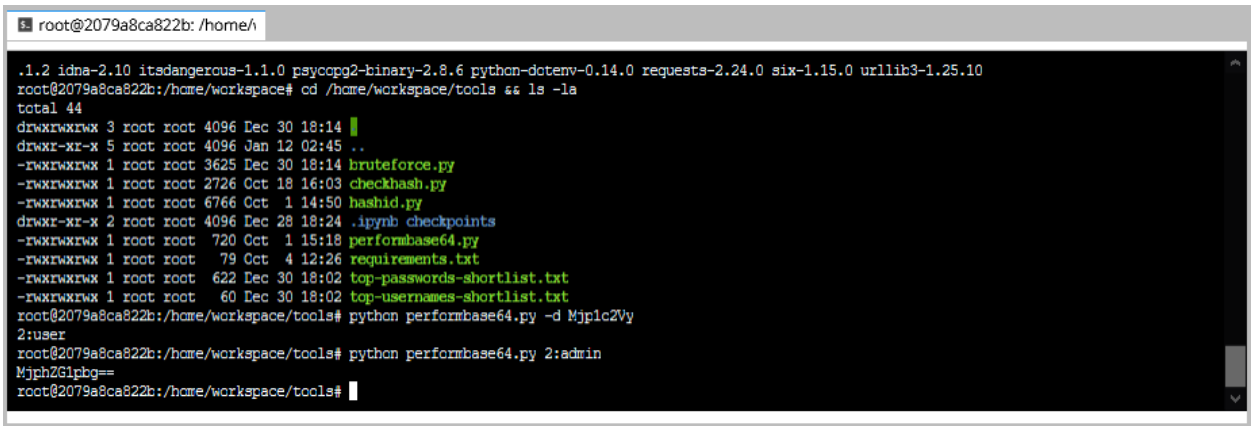
Severity: [Low]

System: VWA Web Application

Vulnerability Explanation:

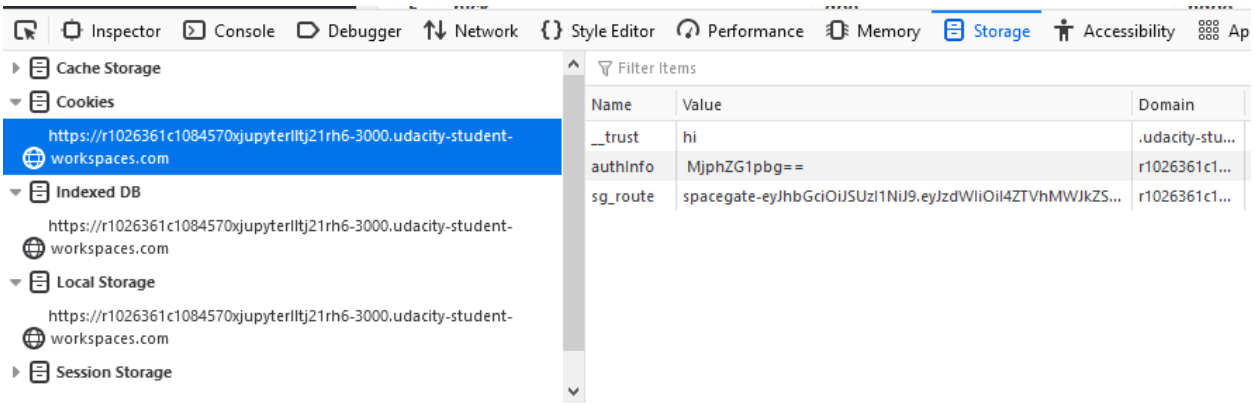
Insecure deserialization often leads to remote code execution by trusting a data source without validating. An attacker can modify their access level and view area of the site they were not intended to see.

Vulnerability Walk-thru:

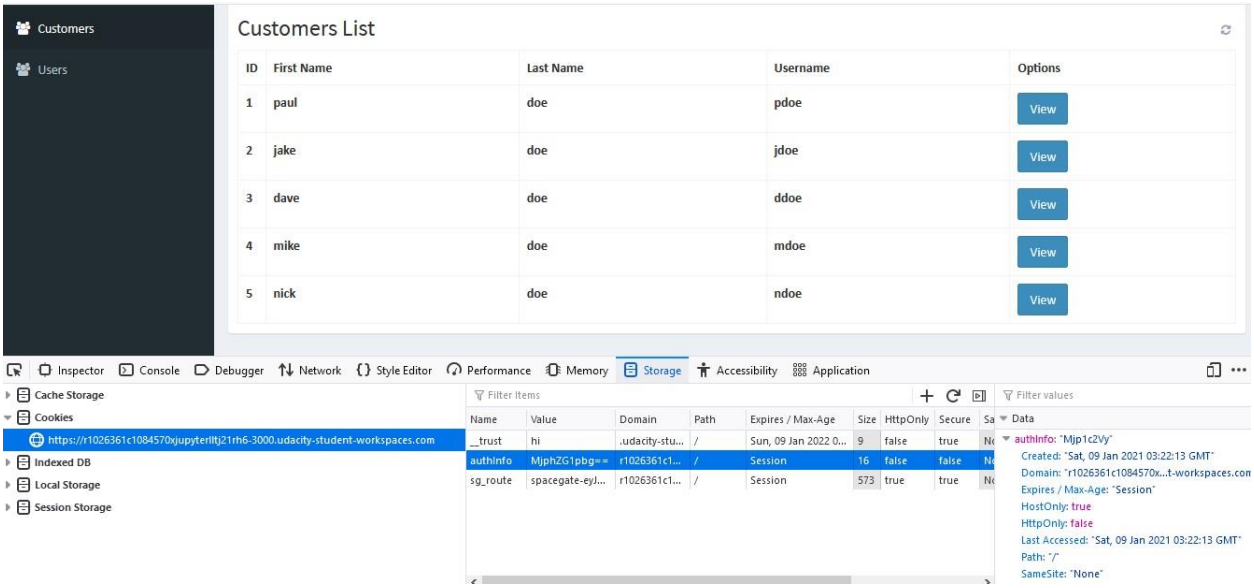


4. Placed the encoded 2:admin value in string MjphZG1pbg== in authinfo value.

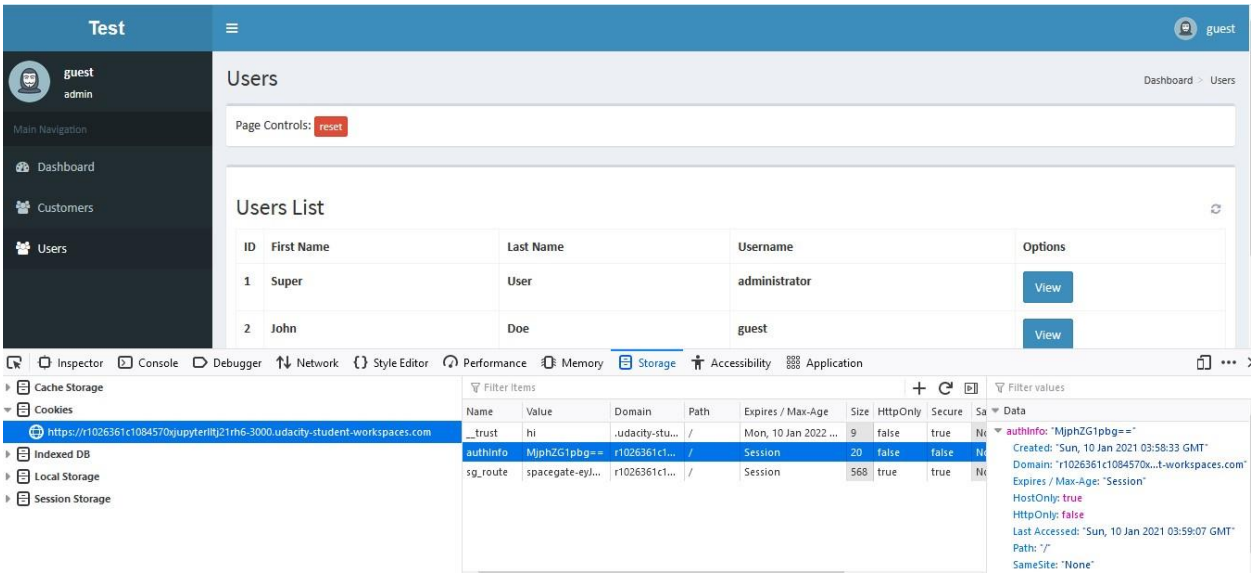
VWA Security Report



5. Got the access as admin to view the customers and users page.



VWA Security Report



Recommendations:

- ✓ https://cheatsheetseries.owasp.org/cheatsheets/Deserialization_Cheat_Sheet.html
- ✓ https://owasp.org/www-project-proactive-controls/v3/en/c5-validate-inputs

Best Practice

- **Use Serialization Mediums** - The only safe architectural pattern is not to accept serialized objects from untrusted sources or to use serialization mediums that only permit primitive data types
- **Integrity Checks** - By using a hashing function you can create a digital signature that then can be used later on to verify that the data has not been altered
- **Strict Data Type** - By using strict data types, this can help protect your data from expected data types.

VWA Security Report

VWA21-1-14-4 – A5:2017 - Broken Access Control - MEDIUM

Vulnerability Exploited: A5 - Broken Access Control (customer)

Severity: [Medium]

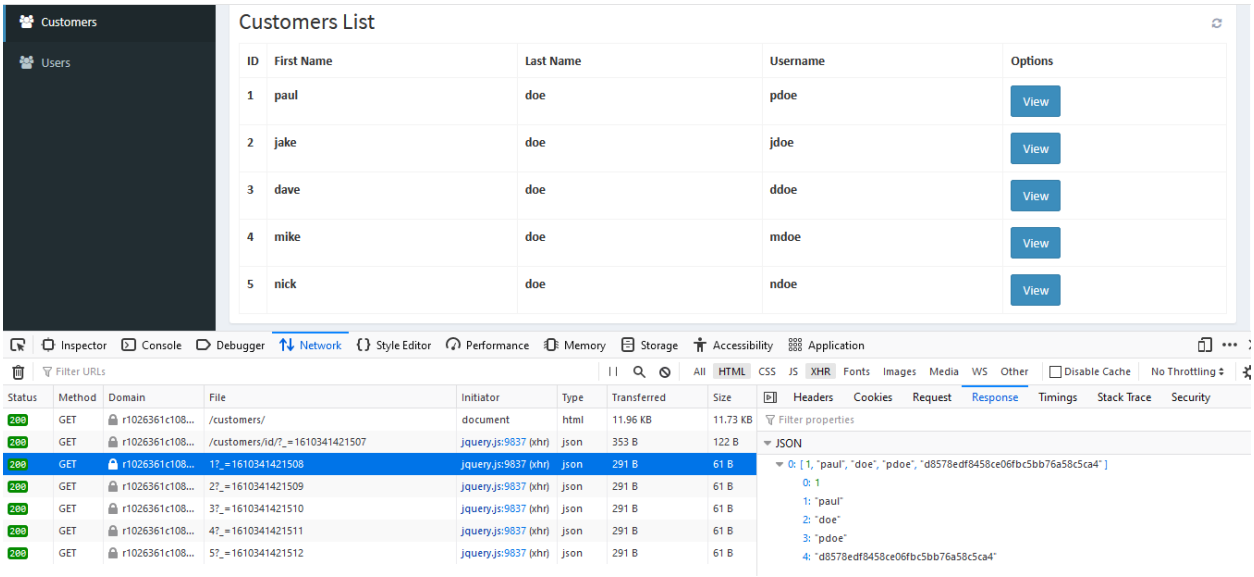
System: VWA Web Application

Vulnerability Explanation:

Broken access controls often fail to enforce restrictions on authenticated users. An attacker takes this advantage to get unauthorized access to restricted areas of the application such as accessing other users' accounts, viewing sensitive files, modifying other users' data, changing access rights, etc.

Vulnerability Walk-thru:

1. I explained in VWAYYMMDD-3 how I got the access as admin to view customers list.
2. There are 5 customers in customers list with respected id and value

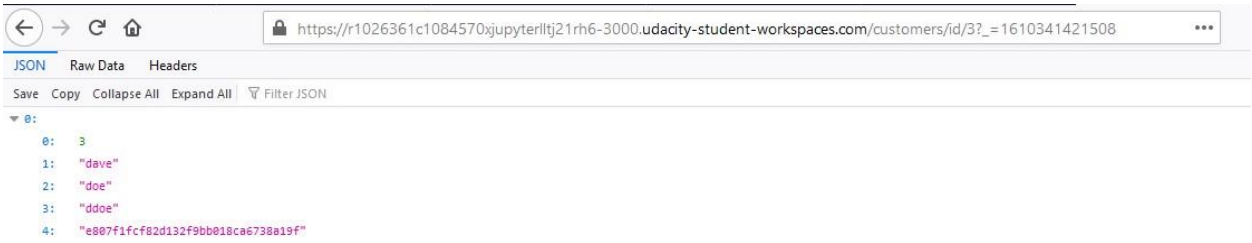


VWA Security Report

3. Information of customer where id=1

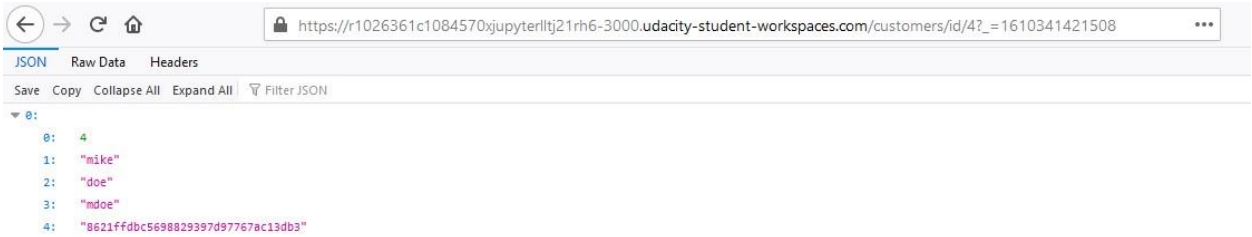
4. Information of customer where id=2 is showing by changing the id number in url instead of viewing it from customer page.

5. Information of customer where id=3 is showing by changing the id number in url instead of viewing it from customer page.



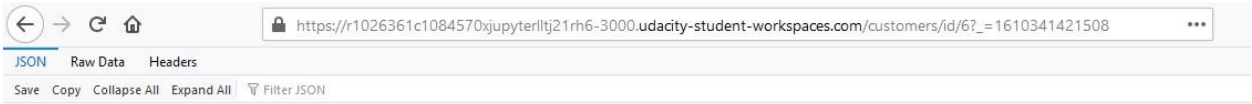
6. Information of customer where id=4 is showing by changing the id number in url instead of viewing it from customer page.

VWA Security Report



7. Information of customer where id=5 is showing by changing the id number in url instead of viewing it from customer page.

8. As there are 5 customers so there exist no information for id=6



VWA Security Report

Recommendations:

- ✓ https://cheatsheetseries.owasp.org/cheatsheets/Access_Control_Cheat_Sheet.html
- ✓ https://owasp.org/www-project-application-security-verification-standard/

Best Practice:

- **Rate Limit Data** - by rate limiting the user access to data on the site, you can slow down their ability to scrape all data from your web application.
- **Re-validate on all secure pages** - make sure you have proper testing around all secure pages and endpoints to validate access control are working as expected
- **Deny Access for non-public pages by default** - you should have general rules that auto deny access to non-public pages and require validation to access these pages
- **Log Access failures** - you should not only log all access failures, but also create automation process that alert you of IP/Users that have high level of failures.

VWA Security Report

VWA21-1-14-5 – A5:2017 - Broken Access Control - LOW

Vulnerability Exploited: A5 - Broken Access Control (user)

Severity: [Low]

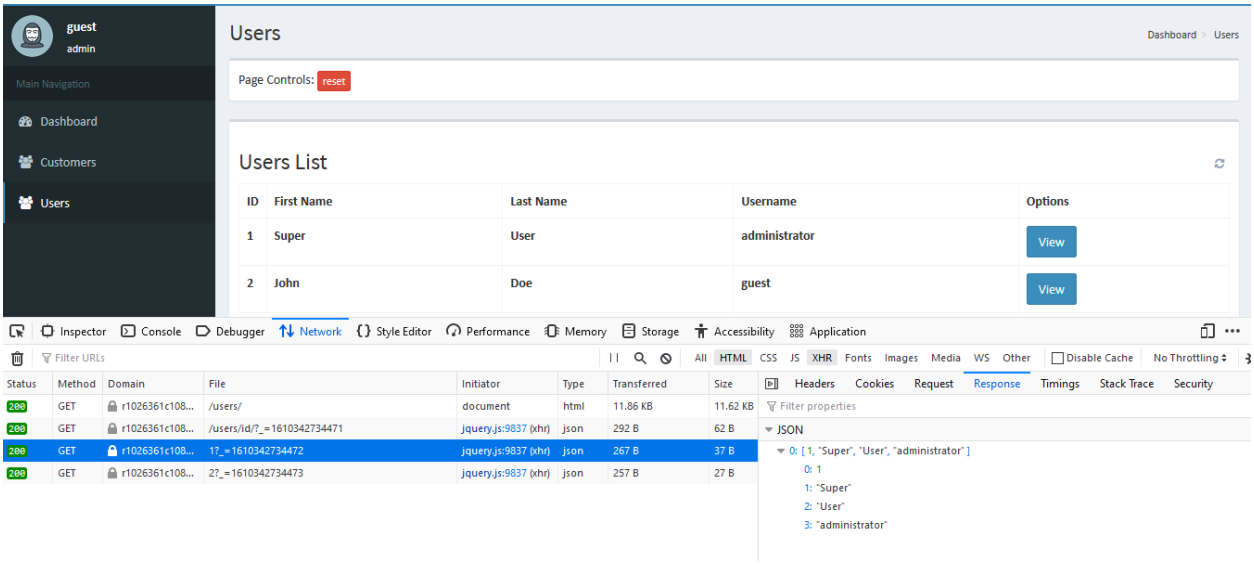
System: VWA Web Application

Vulnerability Explanation:

Broken access controls often fail to enforce restrictions on authenticated users. An attacker takes this advantage to get unauthorized access to restricted areas of the application such as accessing other users' accounts, viewing sensitive files, modifying other users' data, changing access rights, etc.

Vulnerability Walk-thru:

1. I explained in VWAYYMMDD-3 how I got the access as admin to view customers list.
2. There are 2 users in the users list with respected id and value.



VWA Security Report

3.Information of user where id=1

4. Information of user where id=2 is showing by changing the id in url instead of viewing it from user page.

5. As there are 2 users in userlist so no information is available for id=3.

VWA Security Report

Recommendations:

- ✓ https://cheatsheetseries.owasp.org/cheatsheets/Access_Control_Cheat_Sheet.html
- ✓ https://owasp.org/www-project-application-security-verification-standard/

Best Practice:

- **Rate Limit Data** - by rate limiting the user access to data on the site, you can slow down their ability to scrape all data from your web application.
- **Re-validate on all secure pages** - make sure you have proper testing around all secure pages and endpoints to validate access control are working as expected
- **Deny Access for non-public pages by default** - you should have general rules that auto deny access to non-public pages and require validation to access these pages
- **Log Access failures** - you should not only log all access failures, but also create automation process that alert you of IP/Users that have high level of failures.

VWA Security Report

VWA21-1-14-6 – A5:2017 - Broken Access Control - MEDIUM

Vulnerability Exploited: A5 - Broken Access Control (profile)

Severity: [Medium]

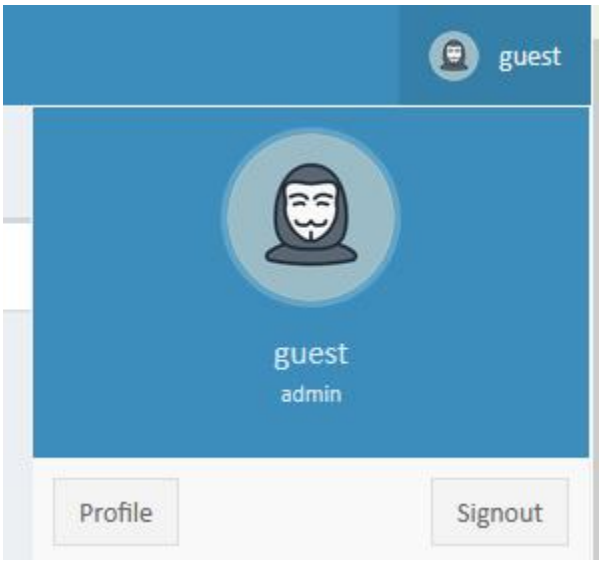
System: VWA Web Application

Vulnerability Explanation:

Broken access controls often fail to enforce restrictions on authenticated users. An attacker takes this advantage to get unauthorized access to restricted areas of the application such as accessing other users' accounts, viewing sensitive files, modifying other users' data, changing access rights, etc.

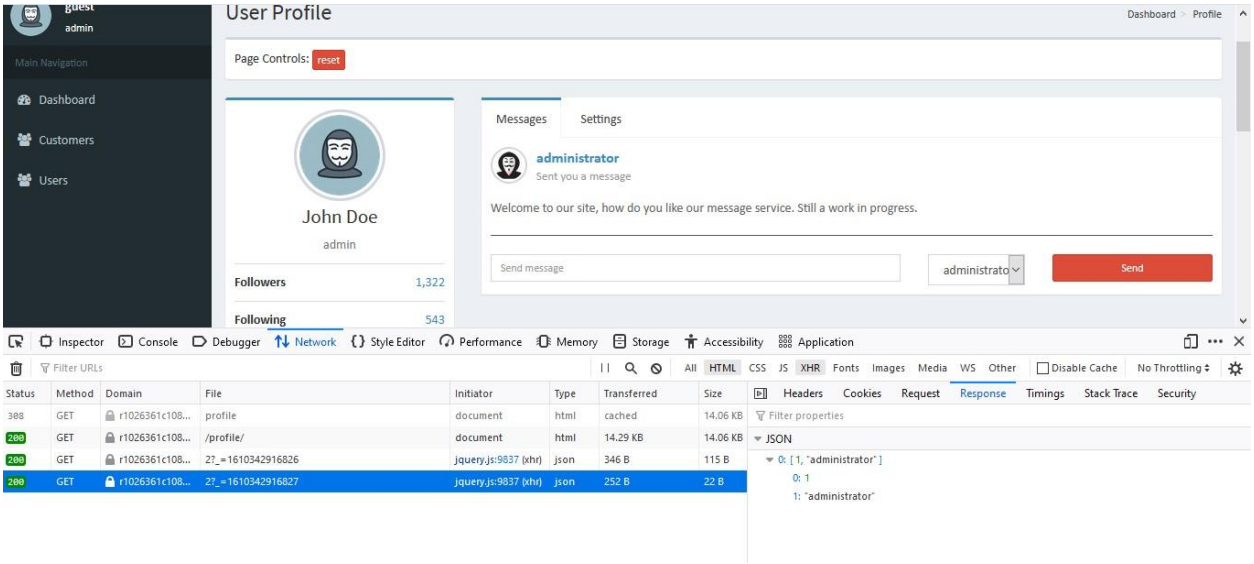
Vulnerability Walk-thru:

1. Go to profile as admin



VWA Security Report

2. Open the inspect go to network tab. There are 2 userlists in profile one is administration another is guest.

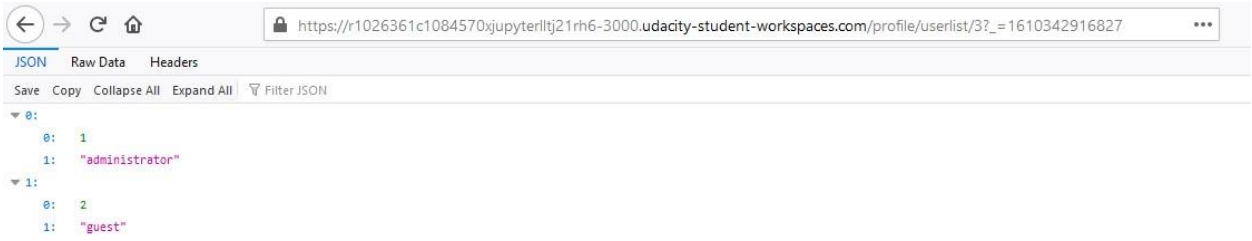


3. Open the userlist in new tab where id =2 is showing the information of administration

4. Information of userlist where id=1 is showing the information by changing the id number in url instead of viewing it from profile page.

VWA Security Report

5. As there are only two userlist where id=3 is showing both guest and administration. Whatever we use except id=1 and id=2 it shows both guest and administration information.



Recommendations:

- ✓ https://cheatsheetseries.owasp.org/cheatsheets/Access_Control_Cheat_Sheet.html
- ✓ https://owasp.org/www-project-application-security-verification-standard/

Best Practice:

- **Rate Limit Data** - by rate limiting the user access to data on the site, you can slow down their ability to scrape all data from your web application.
- **Re-validate on all secure pages** - make sure you have proper testing around all secure pages and endpoints to validate access control are working as expected
- **Deny Access for non-public pages by default** - you should have general rules that auto deny access to non-public pages and require validation to access these pages
- **Log Access failures** - you should not only log all access failures, but also create automation process that alert you of IP/Users that have high level of failures.

VWA Security Report

VWA21-1-14-7 – A6:2017 - Security Misconfiguration – HIGH

Vulnerability Exploited: A6 - Security Misconfiguration

Severity: [High]

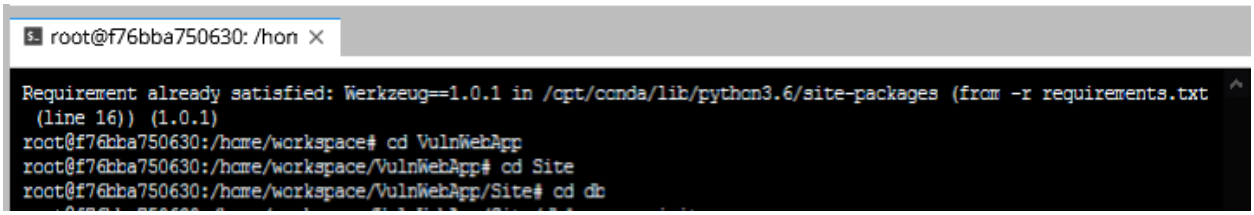
System: VWA Web Application

Vulnerability Explanation:

Security Misconfiguration is basically distinct as deteriorating to implement all the security controls for a server or web application. It is commonly a result of insecure default configurations, incomplete or ad hoc configurations, open cloud storage, misconfigured HTTP headers, and verbose error messages containing sensitive information.

Vulnerability Walk-thru:

By scanning VulnWebApp/Site/db/ more __init__.py file one security misconfiguration found regarding password issue.



VWA Security Report

Issue: Possible hardcoded password: 'weakpasswordrule'

```
root@2079a8ca822b: /hon
root@2079a8ca822b:/home/workspace/VulnWebApp/Site# cd db
root@2079a8ca822b:/home/workspace/VulnWebApp/Site/db# more __init__.py
import sqlite3
import psycopg2
import random
import string

class Conn_postgres():
    conn = None
    cursor = None

    def __init__(self):
        self.open()

    def open(self):
        self.conn = psycopg2.connect(user = "vulnwebsiteuser",
                                     password = "weakpasswordrule",
                                     host = "localhost",
                                     port = "5432",
                                     database = "vulnwebsite")
        self.cursor = self.conn.cursor()
```

Recommendations:

- ✓ https://owasp.org/www-community/vulnerabilities/Use_of_hard-coded_password

Best Practice:

- **All environment should be the same** - By keeping all of your environments the same you will be able to find any issues before they make it out to production.
- **Limit components** - You should push to limit the different components used to only what is really needed, then this will reduce your risk and amount of work to make sure everything is updated and working correctly.
- **Automate Environment creation and validation** - Using automation to create all environments will help reduce user errors and make sure all environments are built exactly the same, next you should automate the validation of the environment to help you understand the health and state of your environments.

VWA Security Report

VWA21-1-14-8 – A3:2017 - Sensitive Data Exposure – MEDIUM

Vulnerability Exploited: A3-Sensitive Data Exposure

Severity: [Medium]

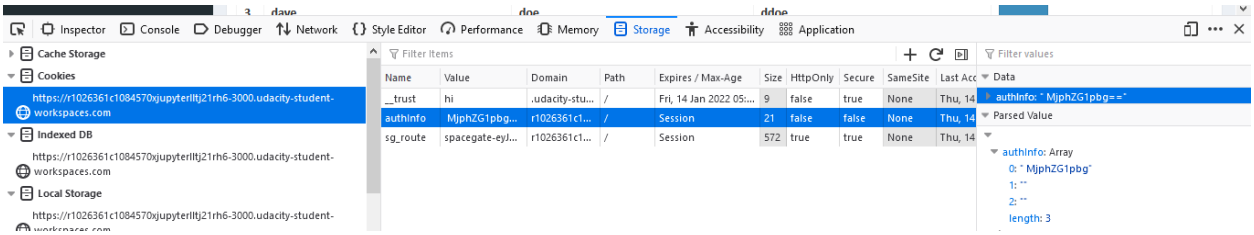
System: VWA Web Application

Vulnerability Explanation:

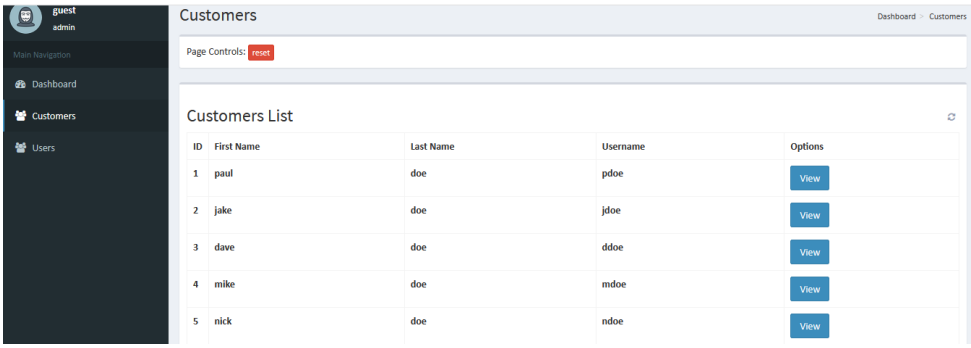
Sensitive data exposure arises when an application, company, or other entity unintentionally exposes personal data. Many web applications and APIs do not properly protect sensitive data, such as financial, healthcare, and PII. An attacker targets those data to steal or modify to conduct mischief like credit card fraud, identity theft, or other crimes.

Vulnerability Walk-thru:

1. I explained it in VWA21-1-14-3 how I got the string value of 2:admin to get access as admin.

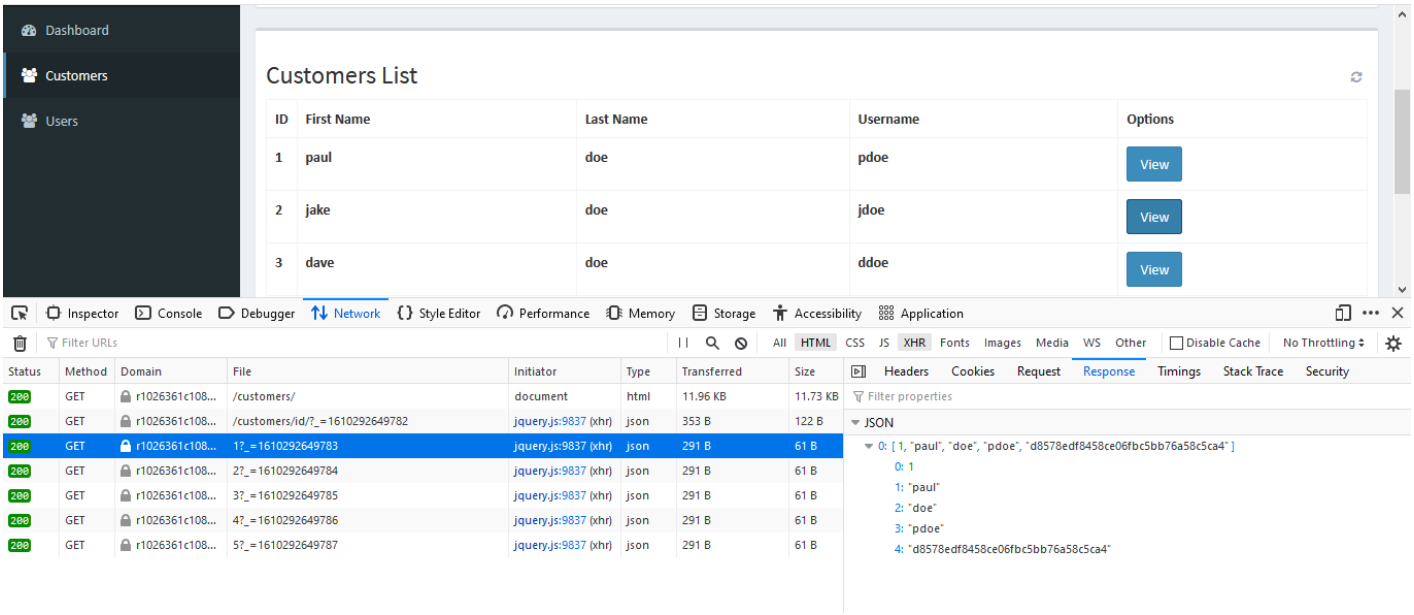


2. After changing the authinfo value refresh the customer page.

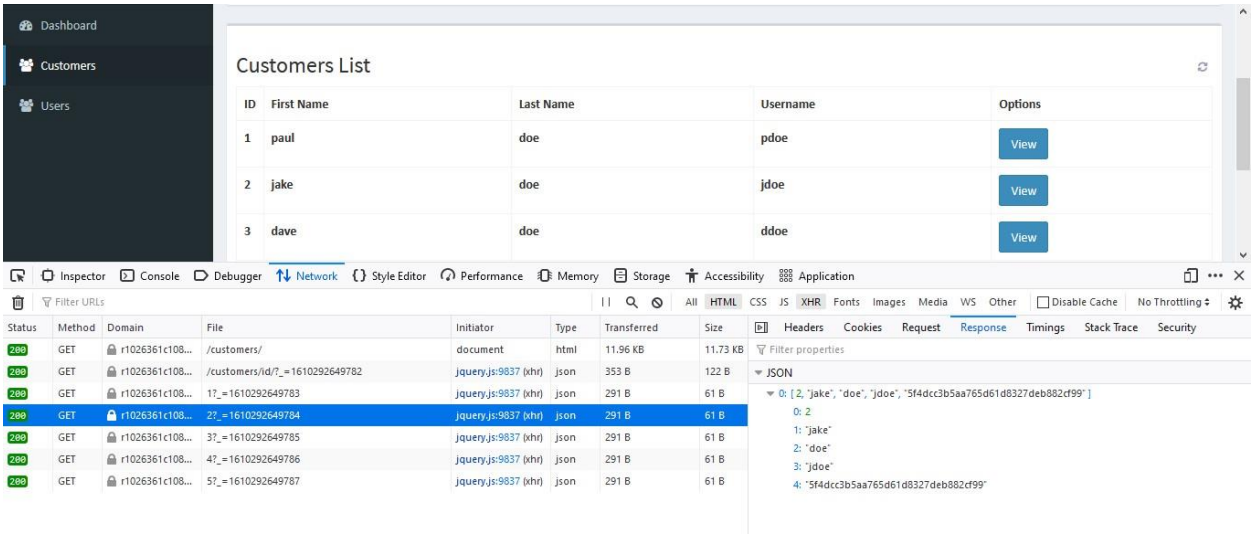


VWA Security Report

3. From customer page by viewing the information in developer tools network tab we can see the information of id=1



4.From customer page by viewing the information in developer tools network tab we can see the information of id=2



VWA Security Report

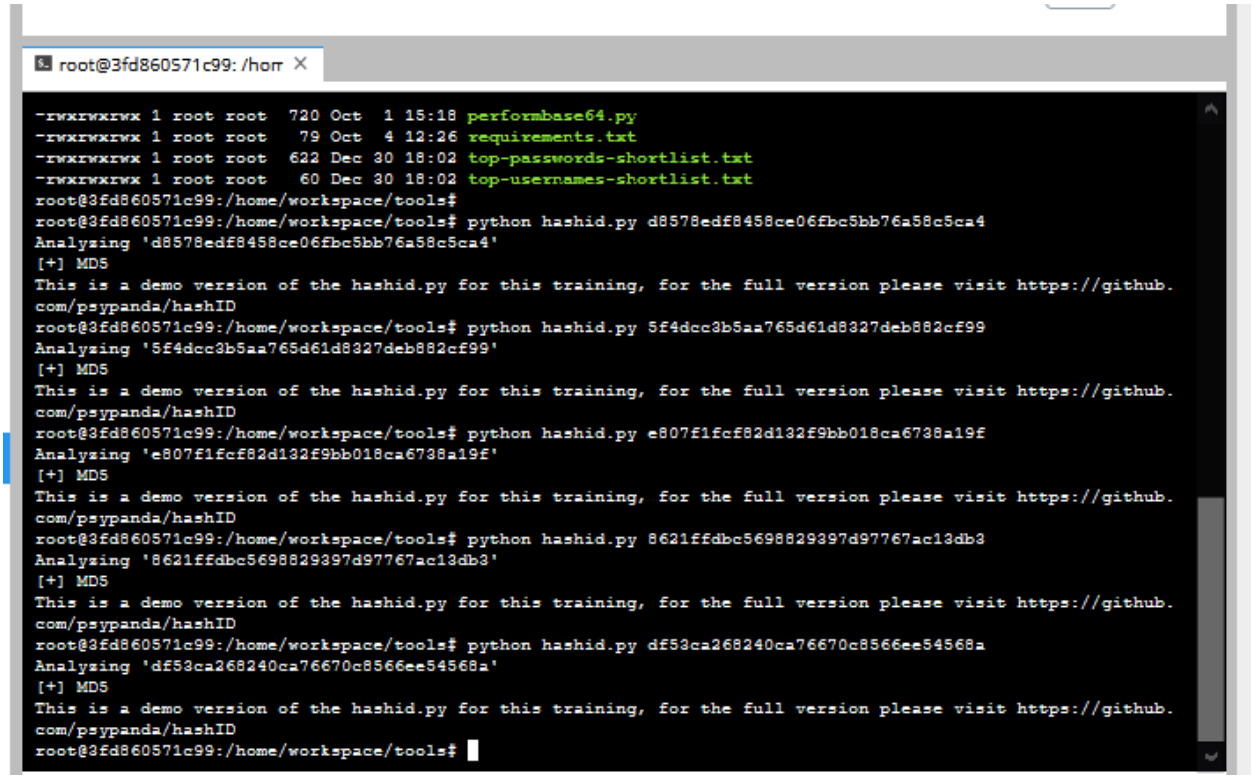
5. From customer page by viewing the information in developer tools network tab and opening in it in new tab we can see the information of id=3

6. From customer page by viewing the information in developer tools network tab and opening in it in new tab we can see the information of id=4

VWA Security Report

7.Cheking the hash type using hashid.py in workspace

```
python hashid.py d8578edf8458ce06fbc5bb76a58c5ca4
python hashid.py 5f4dcc3b5aa765d61d8327deb882cf99
python hashid.py e807f1fcf82d132f9bb018ca6738a19f
python hashid.py 8621ffdbc5698829397d97767ac13db3
python hashid.py df53ca268240ca76670c8566ee54568a
```



VWA Security Report

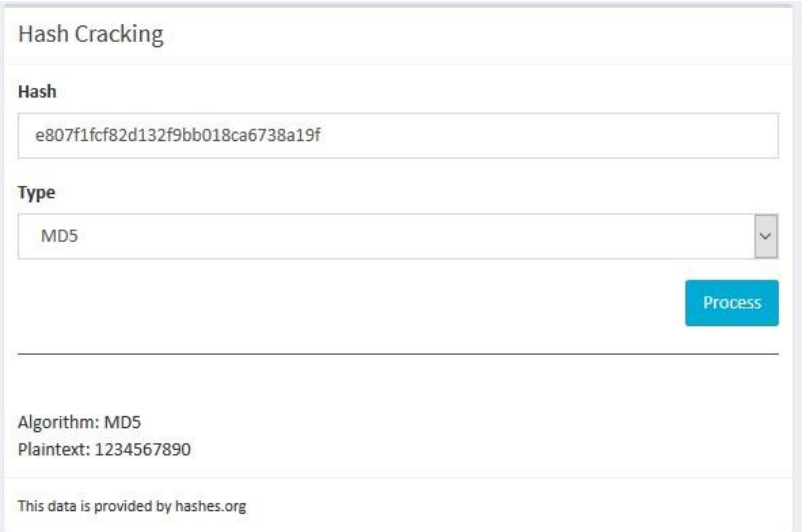
8. Using hash cracking from lesson exercise got the passwords.

1. If we give a close look, we can see this is the first 5 letter from of first row. This type of password is easy to guess need to avoid using this.

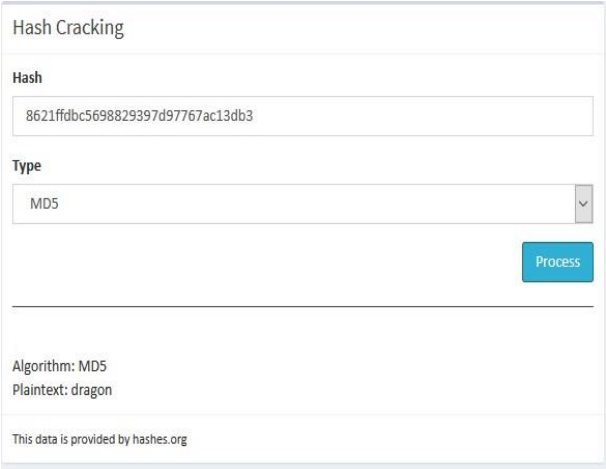
2. Commonly used password to attack.

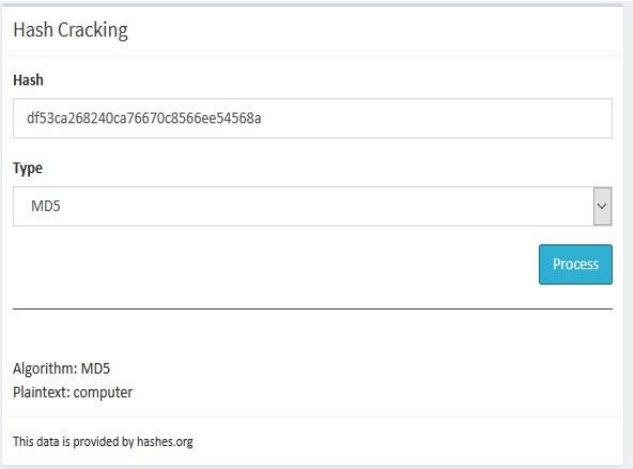
VWA Security Report

3. This password is first 10 digit from number row. Can be cracked by easy guess or trial and error.



4. Brute force may crack this password using common password.





VWA Security Report

Recommendations:

- ✓ https://cheatsheetseries.owasp.org/cheatsheets/Password_Storage_Cheat_Sheet.html
- ✓ https://cheatsheetseries.owasp.org/cheatsheets/User_Privacy_Protection_Cheat_Sheet.html

Best Practice:

- **Classify data -** Classify data processed, stored or transmitted by an application. Identify which data is sensitive according to privacy laws, regulatory requirements, or business needs. Apply controls as per the classification.
- **Encryption:** Make sure to encrypt all sensitive data at rest. Encrypt all data in transit with secure protocols such as TLS with perfect forward secrecy (PFS) ciphers, cipher prioritization by the server, and secure parameters. Enforce encryption using directives like HTTP Strict Transport Security (HSTS)
- **Strengthen Password -** Store passwords using strong adaptive and salted hashing functions with a work factor (delay factor), such as Argon2, scrypt, bcrypt or PBKDF2. Salt and Pepper will increase the strength of your user passwords and make it very difficult to crack.

VWA Security Report

VWA21-1-14-9 – A1:2017 – Injection – CRITICAL

Vulnerability Exploited: A1 – Injection (customer)

Severity: [Critical]

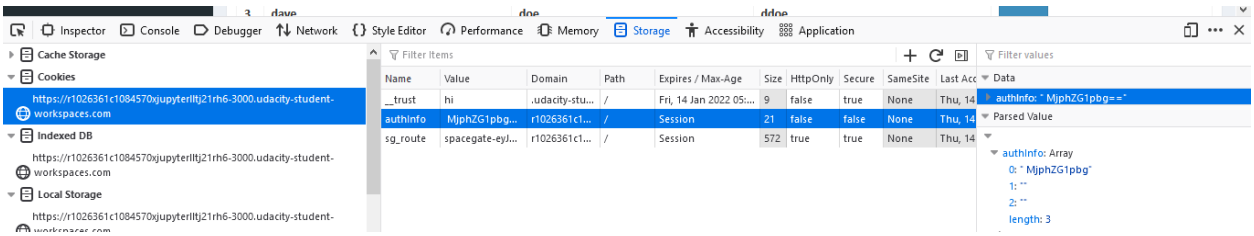
System: VWA Web Application

Vulnerability Explanation:

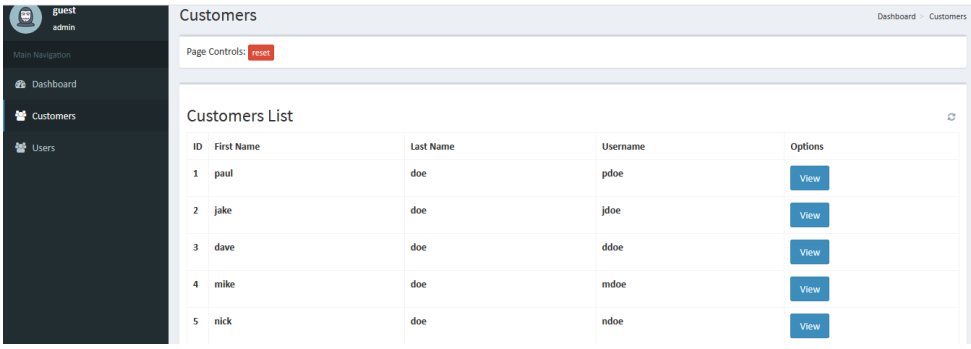
A1-injection occur when untrusted data is sent to an interpreter as part of a command or query. Injection flaws are SQL, NoSQL, OS, and LDAP. We found out SQL injection in VulWebApp. An attacker can exploit the application by using one of the most common SQL Injection commands. An attacker must first breakout of the current SQL query which will then trick the Database into executing the malicious code.

Vulnerability Walk-thru:

1. I explained it in VWA21-1-14-3 how I got the string value of 2:admin to get access as admin.

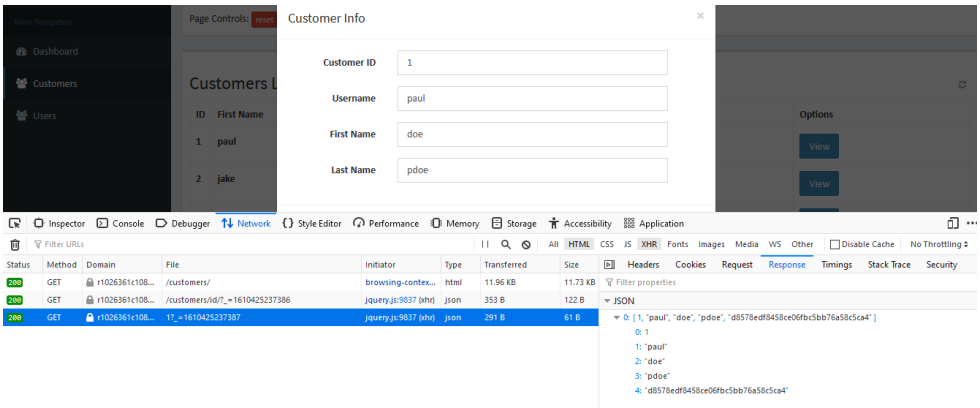


2. After changing the authinfo value refresh the customer page.

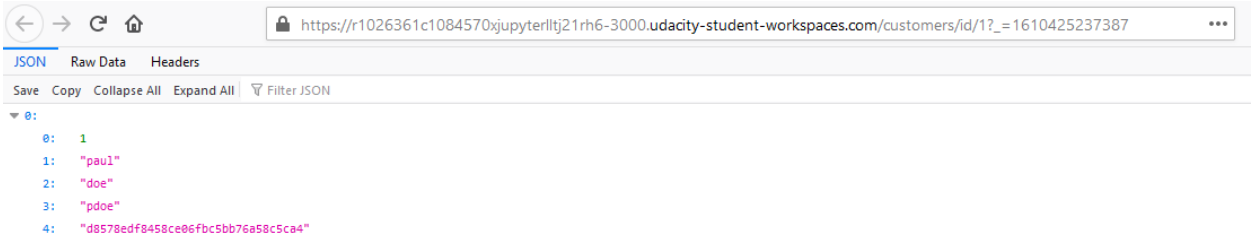


VWA Security Report

3. Open inspect go to network tab. Click view to see information of customers.

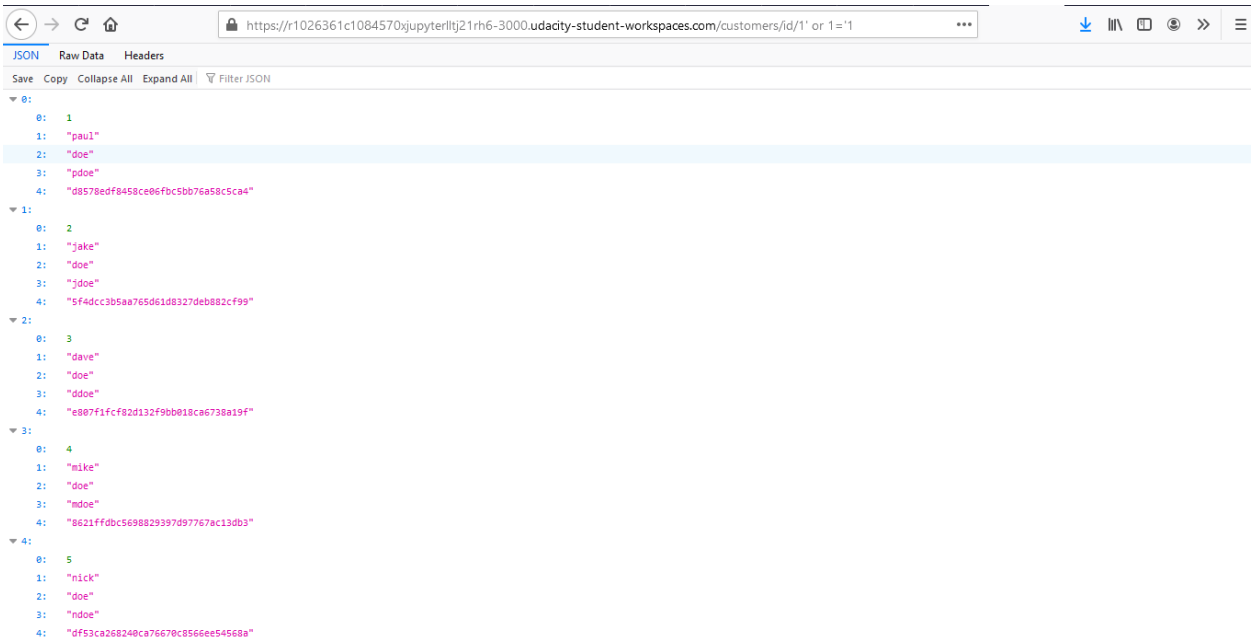


4. Open the customer id=1 information from network in new tab.



VWA Security Report

5. Using the most common SQL "' or 1='1" injection in one customer url found the information of all customer.



VWA Security Report

Recommendations:

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

Best Practice:

- **Use Parameterized Queries** - This is the best method in preventing SQL Injection, because all variables are limited to the data type which will prevent malicious code from breaking out of the SQL code and preventing the malicious code from running.
- **Sanitize all inputs** - I recommend that you always sanitize all inputs before using them, this will prevent malicious code from running not only in SQL Queries but also in other parts of your code.

VWA Security Report

VWA21-1-14-10 – A1:2017 – Injection – CRITICAL

Vulnerability Exploited: A1 – Injection Advance SQL (profile)

Severity: [Critical]

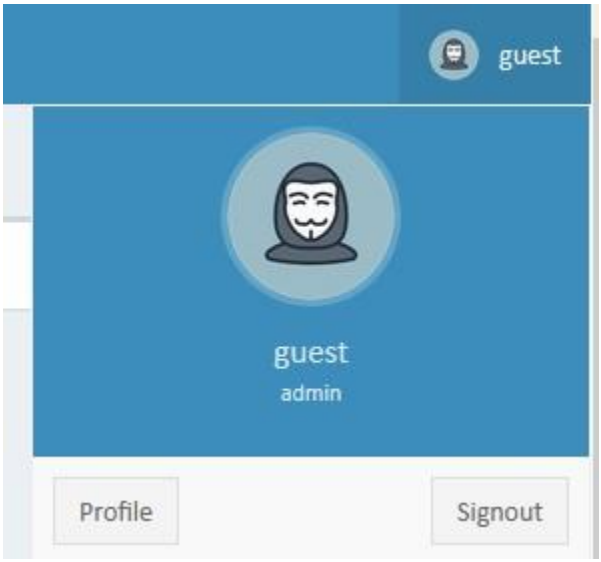
System: VWA Web Application

Vulnerability Explanation:

A1-injection occur when untrusted data is sent to an interpreter as part of a command or query. Injection flaws are SQL, NoSQL, OS, and LDAP. We found out SQL injection in VulWebApp. An attacker can exploit the application by using one of the most common SQL Injection commands. An attacker must first breakout of the current SQL query which will then trick the Database into executing the malicious code.

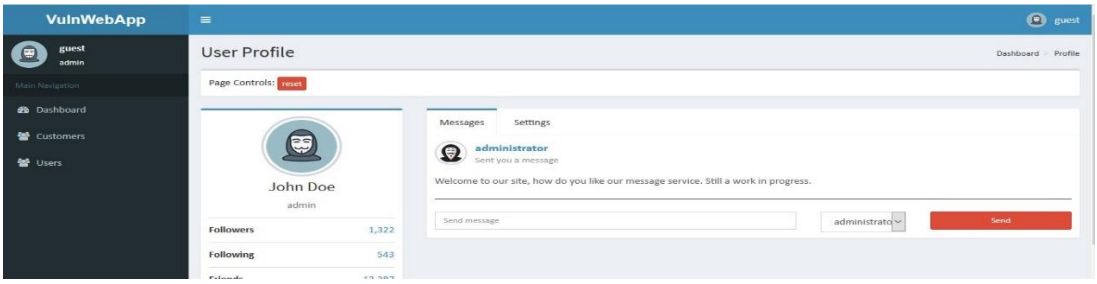
Vulnerability Walk-thru:

1. Go to the profile page as admin.

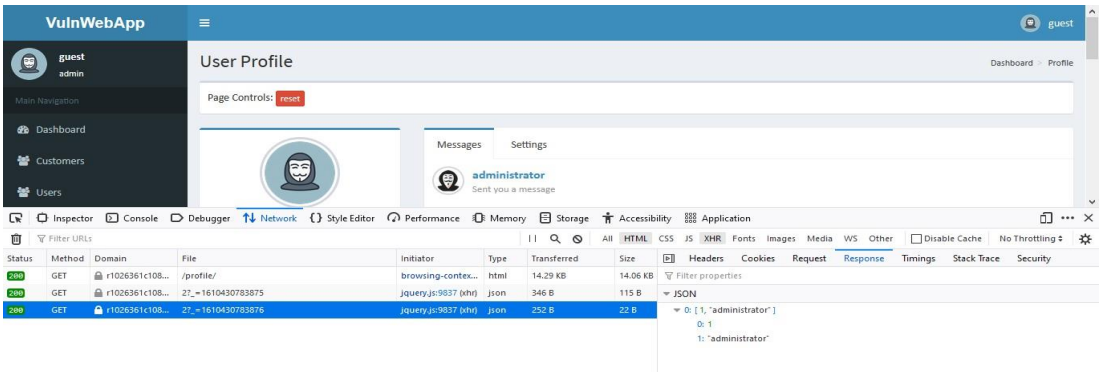


VWA Security Report

2. Profile page of admin



3. Open the inspect and reload the network tab to see userlist.



4. Open then selected inspect in new tab.

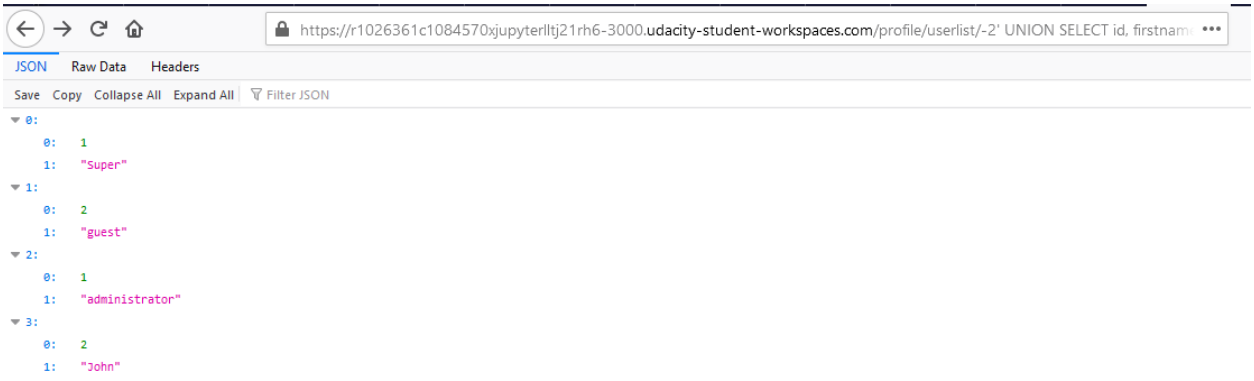
VWA Security Report

5. To see the number of columns inject the SQL injection "-2' ORDER BY 2—"

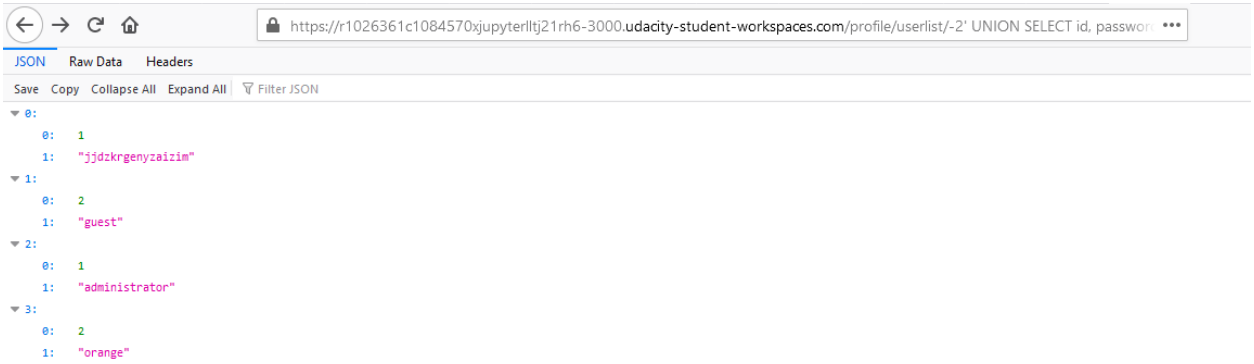
6. Again inject the same SQL increasing the value from 2 to 3 "-2' ORDER BY 3—". Here error is showing which means there is only two user value in userlist.

7. To get the details by changing parameters we will inject -2' UNION SELECT id, Firstname FROM users WHERE username!='0.Here username!='0 means need to show both users details . All details of both guest and administrator will be changed from Firstname parameter to Lastname and username parameter.

VWA Security Report



8. To get the password from the details we will put password in firstname place. inject -2' UNION SELECT id, password FROM users WHERE username!='0. For both guest and administrator, we got the password parameter.



Recommendations:

https://cheatsheetseries.owasp.org/cheatsheets/SQL Injection Prevention Cheat Sheet.html

Best Practice:

- **Use Parameterized Queries** - This is the best method in preventing SQL Injection, because all variables are limited to the data type which will prevent malicious code from

breaking out of the SQL code and preventing the malicious code from running.

- **Sanitize all inputs** - I recommend that you always sanitize all inputs before using them, this will prevent malicious code from running not only in SQL Queries but also in other parts of your code.

VWA21-1-14-11 – A1:2017 – Injection - HIGH

Vulnerability Exploited: A1 – Injection SQL (profile)

Severity: [High]

System: VWA Web Application

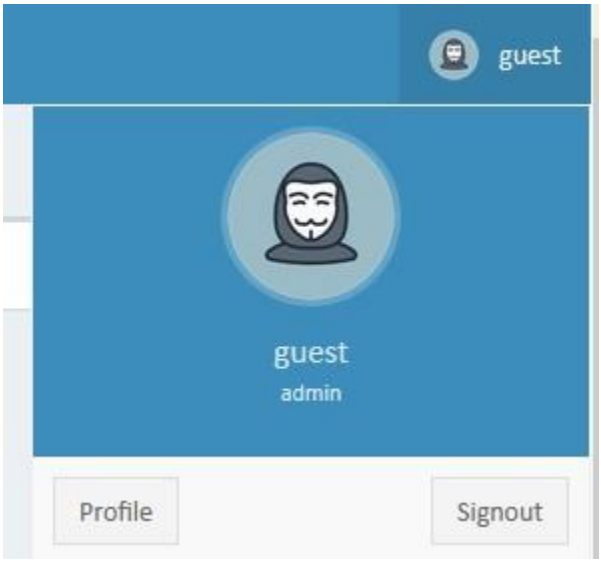
Vulnerability Explanation:

A1-injection occur when untrusted data is sent to an interpreter as part of a command or query. Injection flaws are SQL, NoSQL, OS, and LDAP. We found out SQL injection in VulWebApp. An attacker can exploit the application by using one of the most common SQL Injection commands. An attacker must first breakout of the current SQL query which will then trick the Database into executing the malicious code.

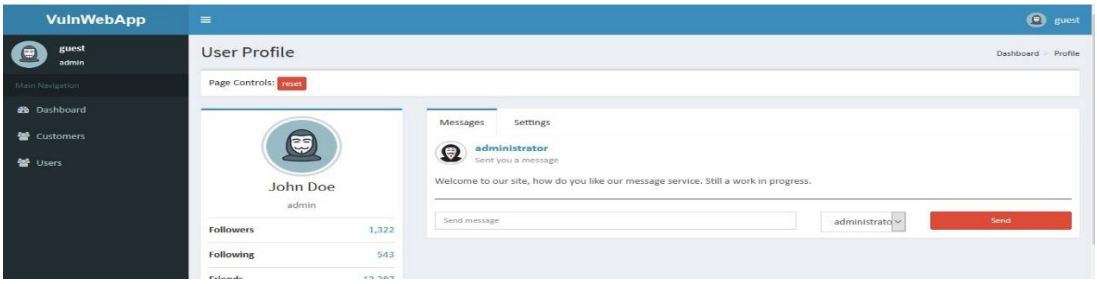
Vulnerability Walk-thru:

1. Go to the profile page as admin.

VWA Security Report

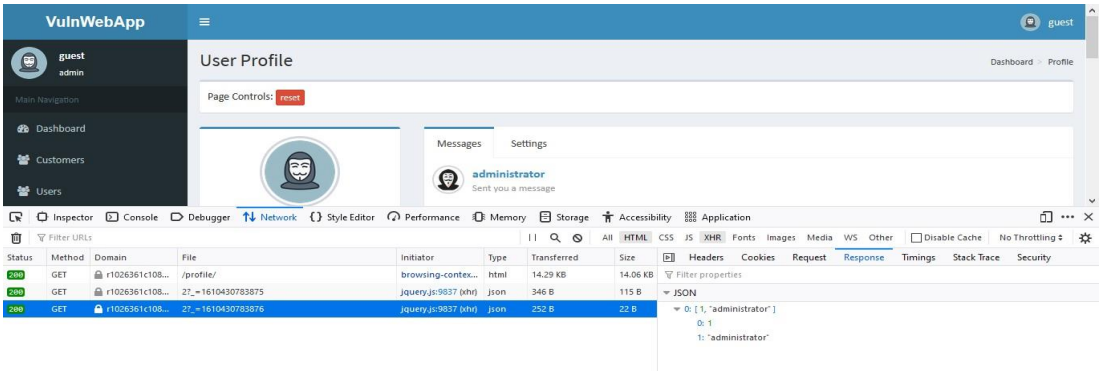


2. Profile page of admin



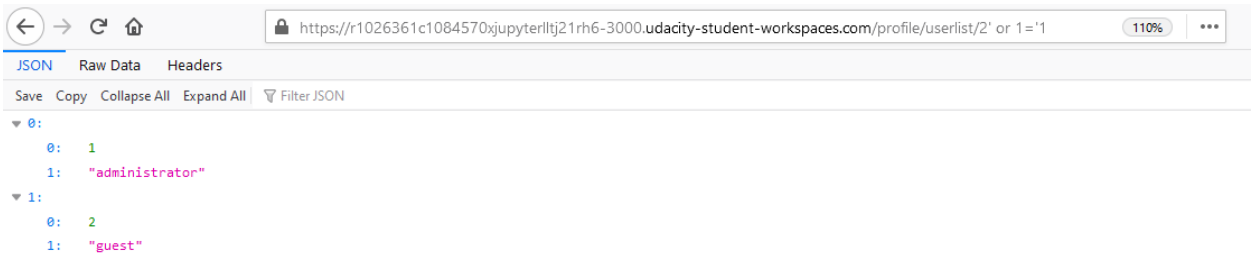
3. Open the inspect and reload the network tab to see userlist.

VWA Security Report



4. Open then selected inspect in new tab.

5. Using the most common SQL "' or 1='1" injection in profile userlist url found the information of all userlist of profile.



Recommendations:

VWA Security Report

https://cheatsheetseries.owasp.org/cheatsheets/SQL Injection Prevention Cheat Sheet.html

Best Practice:

- **Use Parameterized Queries** - This is the best method in preventing SQL Injection, because all variables are limited to the data type which will prevent malicious code from breaking out of the SQL code and preventing the malicious code from running.
- **Sanitize all inputs** - I recommend that you always sanitize all inputs before using them, this will prevent malicious code from running not only in SQL Queries but also in other parts of your code.