VWA Security Report

VulnWebApp (VWA) Security Report

Code Revision: 1.0.0.0
Company: Acme Inc.
Report: VWAYYMMDD
Author: [Omar Adel Sayed]
Date: [12-3-2023]

VWA Security Report

Vulnerability Exploited: A4 - XML External Entities (XXE)

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

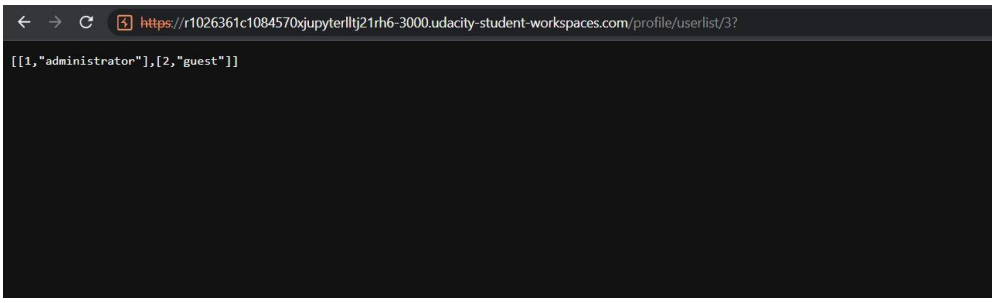
Severity : High

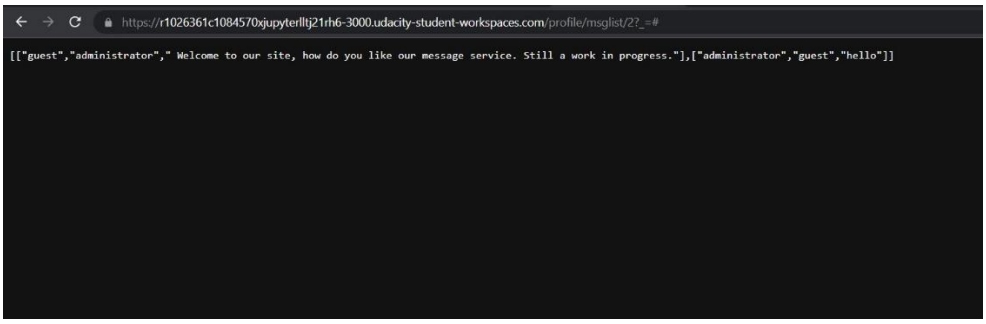
System: Flask web application

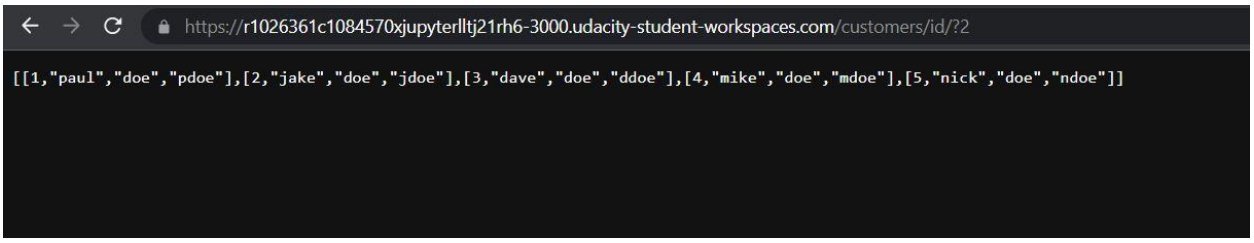
Vulnerability Explanation:

The scan result indicates that the lxml.etree.fromstring() function is being used to parse untrusted XML data, which is vulnerable to XML External Entities (XXE) attacks. An attacker can inject malicious XML content that could lead to various attacks, including reading sensitive data from the server or even executing arbitrary code on the server.

Vulnerability Walk-thru:







Recommendations:

VWA Security Report

To fix this issue, replace the lxml.etree.fromstring() function with its defusedxml equivalent function, which is designed to prevent XXE attacks, it's also recommended to implement proper input validation and sanitization to prevent any malicious XML content.

VWA Security Report

Vulnerability Exploited: A3:Sensitive Data Exposure

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

Severity : High

System:

Vulnerability Explanation:

The subprocess.Popen() call is potentially vulnerable to command injection attacks. If the "startupcmd" parameter is not properly sanitized, an attacker could inject arbitrary commands to be executed by the operating system.

Vulnerability Walk-thru:

```
root@4500d322ae8a:/home/workspace/tools# python performbas
e64.py -d Mjp1c2Vy
2:user
root@4500d322ae8a:/home/workspace/tools#
```

```
root@402c117e5880:/home/workspace# curl 'https://r1026361c1084570xjupyterlltj21rh6-3000.udacity-s
tudent-workspaces.com/reset?_=1678809801951' -H 'Cookie: userid=Mjp1c2Vy'
curl: (60) SSL certificate problem: certificate has expired
More details here: https://curl.haxx.se/docs/sslcerts.html

curl performs SSL certificate verification by default, using a "bundle"
 of Certificate Authority (CA) public keys (CA certs). If the default
 bundle file isn't adequate, you can specify an alternate file
 using the --cacert option.
If this HTTPS server uses a certificate signed by a CA represented in
 the bundle, the certificate verification probably failed due to a
 problem with the certificate (it might be expired, or the name might
 not match the domain name in the URL).
If you'd like to turn off curl's verification of the certificate, use
 the -k (or --insecure) option.
root@402c117e5880:/home/workspace# curl 'https://r1026361c1084570xjupyterlltj21rh6-3000.udacity-s
tudent-workspaces.com/reset?_=1678809801951' -H 'Cookie: userid=2'
curl: (60) SSL certificate problem: certificate has expired
More details here: https://curl.haxx.se/docs/sslcerts.html

curl performs SSL certificate verification by default, using a "bundle"
 of Certificate Authority (CA) public keys (CA certs). If the default
 bundle file isn't adequate, you can specify an alternate file
 using the --cacert option.
If this HTTPS server uses a certificate signed by a CA represented in
 the bundle, the certificate verification probably failed due to a
 problem with the certificate (it might be expired, or the name might
 not match the domain name in the URL).
If you'd like to turn off curl's verification of the certificate, use
 the -k (or --insecure) option.
```

VWA Security Report

Recommendations:

it is recommended to properly validate and sanitize any input to prevent command injection attacks.

VWA Security Report

Vulnerability Exploited: A4. XML External Entities (XXE)

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

Severity : High

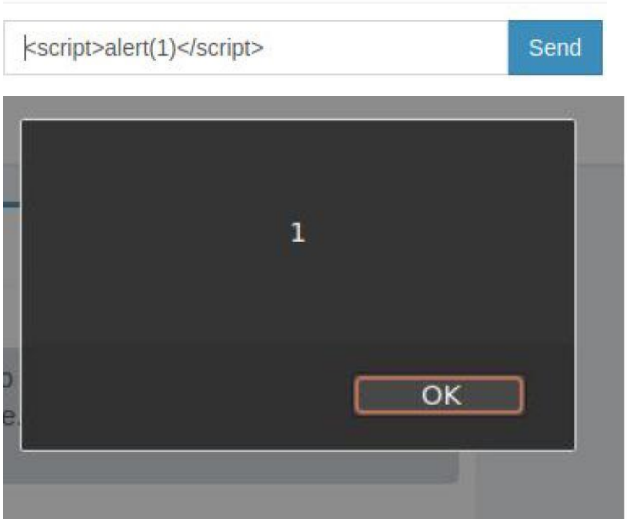
System: Python subprocess module

Vulnerability Explanation:

The issue is that the subprocess module is being used without proper caution.

Vulnerability Walk-thru:

1. **Go to the profile**
2. **In the message box**
3. **I first try a basic XSS "<script>alert(1)</script>" to see if the code is exploitable.**



VWA Security Report

Recommendations:

We should use a standard library that does field sanitizing for

XSS.

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

VWA Security Report

Vulnerability Exploited: A8.Insecure Deserialization

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

Severity : High

System: This code is part of a remote code execution that manages customer data and authentication

Vulnerability Explanation:

Insecure deserialization often leads to remote code execution.without proper caution.

Vulnerability Walk-thru:

```
from flask import current_app as app
import random
import string
import json

from . import insecured

userinfo = {
    'username': 'jdoe',
    'role': 'user'
}

@register_breadcrumb(insecured, '.', 'insecured')
def main():
    resp = make_response(render_template("insecured/index.html", siteInfo=g, ti
    resp.set_cookie('auth', json.dumps(userinfo))
    return resp
```

```
ws.py ×
insecured > views.py
@register_breadcrumb(insecured, '.', 'insecured')
def adminarea():
    try:
        auth = request.cookies.get('auth')
        userinfo = json.loads(auth)
        username = userinfo['role']
        role = userinfo['role']
    except:
        username = ""
        role = ""

    if role == 'admin':
        resp = make_response(render_template("insecured/inside.html", siteInfo=
    else:
        resp = make_response(render_template("insecured/unauth.html", siteInfo=

    return resp
```

```
ws.py ●
insecured > views.py
    try:
        auth = request.cookies.get('auth')
        userinfo = json.loads(auth)
        username = userinfo['username']
        role = userinfo['role']
        key = userinfo['key'] #md5(username + role + 'asdfkasdfkjasdfkj')
    except:
        username = ""
        role = ""

    if role == 'admin':
        resp = make_response(render_template("insecured/inside.html", siteInfo
    else:
        resp = make_response(render_template("insecured/unauth.html", siteInfo

    return resp
```

Recommendations:

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

VWA Security Report

Vulnerability Exploited: A5.Broken Access Control

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

Severity : High

System:hack(broken) access control

Vulnerability Explanation:

Broken Access Control is when the web application fails to secure the access to both an area of this site and data they consider to be sensitive.execution.without proper caution.

Vulnerability Walk-thru:

```
root@131960a6019e:/home/workspace/tools# python bruteforce.py -U test-username.
txt -P test-password.txt -f notfound https://r1026361c1084570xjupyterlltj21rh6-
3000.udacity-student-workspaces.com/login
[+] Login Found! {'username': 'root', 'password': '123456'}
This is a demo code used for this training.
root@131960a6019e:/home/workspace/tools#
```

Recommendations:

-create automation process that alert you of IP/Users that have a high level of failures.

- have general rules that auto deny access to non-public pages

VWA Security Report

Vulnerability Exploited: **A5 - Broken Authentication and Session Management**

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

Severity : Medium

System: This code is part of a larger application that manages customer data and authentication.

Vulnerability Explanation:

The code uses an insecure hashing algorithm (MD5) to store passwords. This can make it easier for attackers to obtain passwords using methods like brute-forcing or rainbow tables. This vulnerability can allow unauthorized access to customer data and other sensitive information. To fix this vulnerability, the code should use a stronger hashing algorithm, such as bcrypt or scrypt, that is more resilient to attacks. Additionally, the code should implement salting to further enhance the security of the password storage. These measures will improve the overall security of the system by making it harder for attackers to obtain customer passwords and gain unauthorized access.

Vulnerability Walk-thru:

1. Go to vulnerability tools
2. Using a python script call "bruteforce.py"
3. I ran the following cmd.

a. python bruteforce.py -U test-username.txt -P test-password.txt -f False

4. After a min of running it was a find a working username and

password combination that allowed me to login.

VWA Security Report

```
root@131960a6019e:/home/workspace/tools# python bruteforce.py -U test-username.
txt -P test-password.txt -f notfound https://r1026361c1084570xjupyterlltj21rh6-
3000.udacity-student-workspaces.com/login
[+] Login Found! {'username': 'root', 'password': '123456'}
This is a demo code used for this training.
root@131960a6019e:/home/workspace/tools#
```

```
root@4500d322ae8a:/home/workspace/tools# python bruteforce
.py -Utest-username.txt -P test-password.txt -f "Login Fai
led" http://0.0.0.0:3000/login
[+] Login Found! {'username': 'guest', 'password': 'orange
'}
This is a demo code used for this training.
root@4500d322ae8a:/home/workspace/tools#
```

Recommendations:

The code should use a stronger hashing algorithm, such as bcrypt or scrypt, to securely hash passwords. These algorithms are specifically designed for password hashing and are more resilient to brute-force attacks. Additionally, the code should implement a salted hash to further enhance the security of the password storage. Finally, the code should also consider adding other security measures, such as multi-factor authentication and rate-limiting, to mitigate the risk of brute-force attacks.

VWA Security Report

Vulnerability Exploited: A2: Broken Authentication and Session Management

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

Severity : Medium

System: PostgreSQL database

Vulnerability Explanation:

The password for the database is hardcoded in the code, which means it is visible to anyone who has access to the codebase. This is a security risk because if an attacker gains access to the codebase, they can easily obtain the password and use it to compromise the database.

Vulnerability Walk-thru:

1. Go to vulnerability tools
2. Using a python script call “bruteforce.py”
3. I ran the following cmd.
4. python bruteforce.py -U test-username.txt -P test-password.txt -f False
5. I f password found you should see it but if password is not found you should the false result with the false message you write.

```
root@131960a6019e:/home/workspace/tools# python bruteforce.py -U test-username.
txt -P test-password.txt -f notfound https://r1026361c1084570xjupyterlltj21rh6-
3000.udacity-student-workspaces.com/login
[+] Login Found! {'username': 'root', 'password': '123456'}
This is a demo code used for this training.
root@131960a6019e:/home/workspace/tools#
```

```
root@4500d322ae8a:/home/workspace/tools# python bruteforce
.py -Utest-username.txt -P test-password.txt -f "Login Fai
led" http://0.0.0.0:3000/login
[+] Login Found! {'username': 'guest', 'password': 'orange
'}
This is a demo code used for this training.
root@4500d322ae8a:/home/workspace/tools#
```

Recommendations:

VWA Security Report

Store the password in a secure location outside of the codebase, such as an environment variable or a configuration file that is not tracked in version control. Use a secure password that is not easily guessable and change it regularly to minimize the risk of a compromise. Additionally, consider implementing two-factor authentication and other authentication and session management best practices to improve overall security.

VWA Security Report

Vulnerability Exploited: A2: Broken Authentication and Session Management

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

Severity : Medium

System: PostgreSQL database

Vulnerability Explanation:

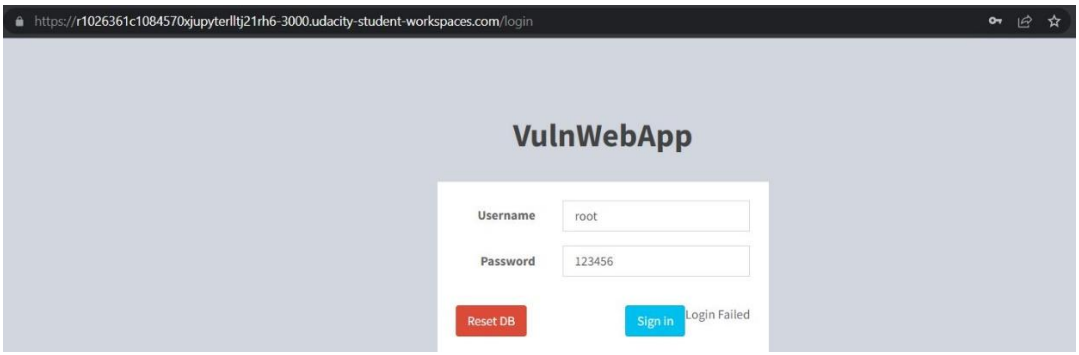
The random password generator used in the randPassword function is based on the random module in Python, which is not suitable for security/cryptographic purposes, as it uses a standard pseudo-random generator. An attacker could potentially predict the output of this function and compromise the security of the application.

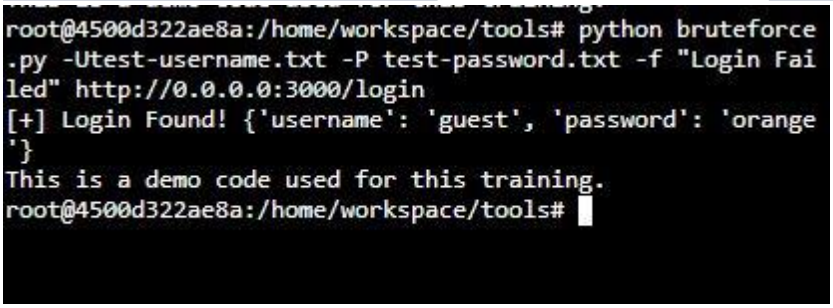
Vulnerability Walk-thru:

1. Go to vulnerability tools
2. Using a python script call “bruteforce.py”
3. I ran the following cmd.
4. python bruteforce.py -U test-username.txt -P test-password.txt -f False
5. I f password found you should see it but if password is not found you should the false result with the false message you write.
5. after that if found password and username try to access to the website

```
root@131960a6019e:/home/workspace/tools# python bruteforce.py -U test-username.
txt -P test-password.txt -f notfound https://r1026361c1084570xjupyterlltj21rh6-
3000.udacity-student-workspaces.com/login
[+] Login Found! {'username': 'root', 'password': '123456'}
This is a demo code used for this training.
root@131960a6019e:/home/workspace/tools#
```

VWA Security Report





Recommendations:

Should use a cryptographically secure random number generator, such as the secrets module in Python, to generate random passwords.

###

Replace the line result_str = ''.join(random.choice(letters) for i in range(length)) with result_str = ''.join(secrets.choice(letters) for i in range(length)) in the randPassword function.

###

VWA Security Report

Vulnerability Exploited: A9.Using Components with Known Vulnerabilities

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

Severity : Medium

System:using framework or known vulns

Vulnerability Explanation:

application or network used to host your application is using a module with known vulnerabilities that could be

Recommendations:

- Only download resources from trusted secure sources.

-monitor all resources used, to make sure they are still secure.

VWA Security Report

Vulnerability Exploited: A10.Insufficient Logging and Monitoring

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

Severity : Medium

System: web application server is setup and either the logs

Vulnerability Explanation:

web application server is setup and either the logs are not configured

Recommendations:

- automating the log parse, that can reduce the noise from events that need attention.

- Using a centralized logging system that will help reduce the amount of (time ,effort) that needed between going thruogh all servers.

VWA Security Report

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

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

Severity : Low

System: This code is part of a larger Django application that manages customer data and authentication

Vulnerability Explanation:

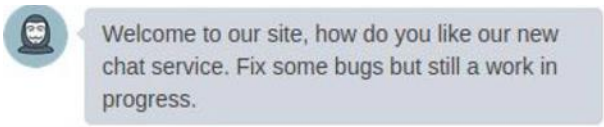
The code uses the mark_safe function to render user-supplied content in the banner field. This function tells Django not to escape the content, which can allow an attacker to inject malicious scripts or code into the page, potentially stealing sensitive user data or performing other harmful actions. This vulnerability can lead to Cross-Site Scripting (XSS) attacks.

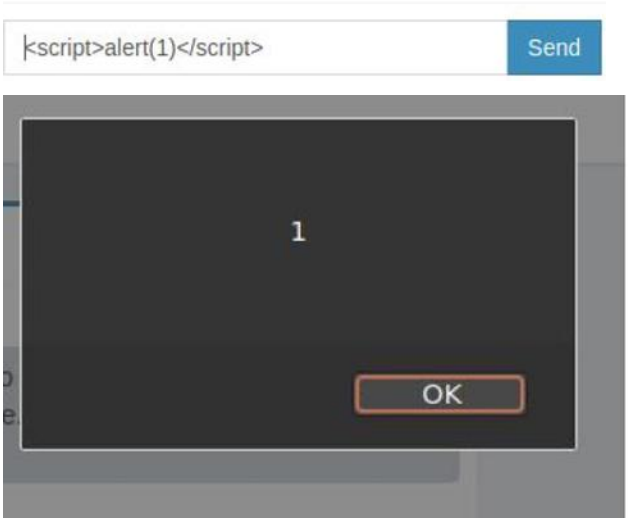
Vulnerability Walk-thru:

1. Go to the profile
2. In the message box
3. I first try a basic XSS "<script>alert(1)</script>" to see if the code is exploitable.

And there a lot of options you may try to write a fake login auth field (form) to keep user enter username and password to exploit the user page

VWA Security Report





Recommendations:

Instead of using mark_safe, the code should use Django's built-in template escaping to ensure that all user-supplied content is properly sanitized and rendered safely. This can be achieved by using Django's escape or force_escape functions to properly escape user input before rendering it. Additionally, it's important to validate all user input and only allow known safe content to be rendered.

VWA Security Report

Vulnerability Exploited: N/A

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

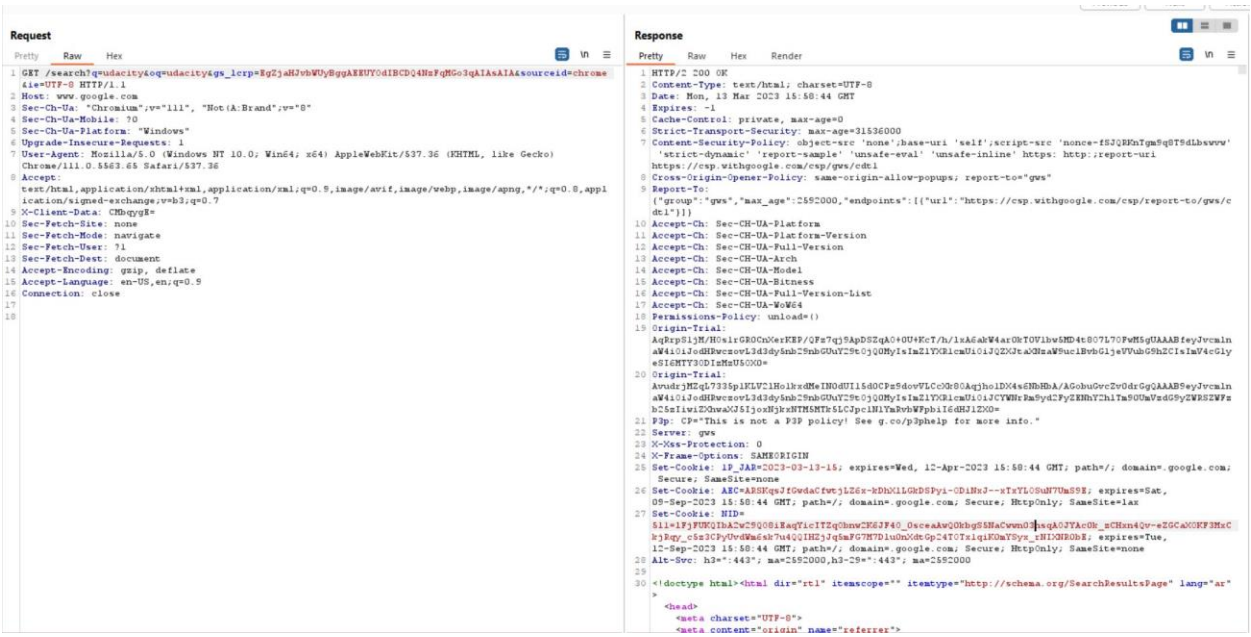
Severity : Low

System: Python subprocess module

Vulnerability Explanation:

The issue is that the subprocess module is being used without proper caution.

Vulnerability Walk-thru:



Recommendations:

The recommended approach is to avoid using subprocess whenever possible, and instead use libraries that are specifically designed for the task at hand (e.g., os or shutil for file operations). If subprocess must be used, then the input should be sanitized properly to prevent command injection attacks.