OWASP Broken Access Lab Exercise

The development of this document is/was funded by three grants from the DOD Grant Number H98230-19-1-0301

1 Overview

This Labtainer exercise explores Broken access control. Broken access control happens when the application allows a user to perform unauthorized actions. There are many vulnerabilities which contribute to this risk, for instance, if the developer forgets to validate permissions

2 Lab Environment

This lab runs in the Labtainer framework, available at http://my.nps.edu/web/c3o/labtainers. That site includes links to a pre-built virtual machine that has Labtainers installed, however Labtainers can be run on any Linux host that supports Docker containers.

From your labtainer-student directory start the lab using:

```
labtainer web-brokenaccess
```

On most Linux systems, these are links that you can right click on and select "Open Link". **If you chose to edit the lab report on a different system, you are responsible for copying the completed report back to the displayed path on your Linux system before using "stoplab" to stop the lab for the last time.**

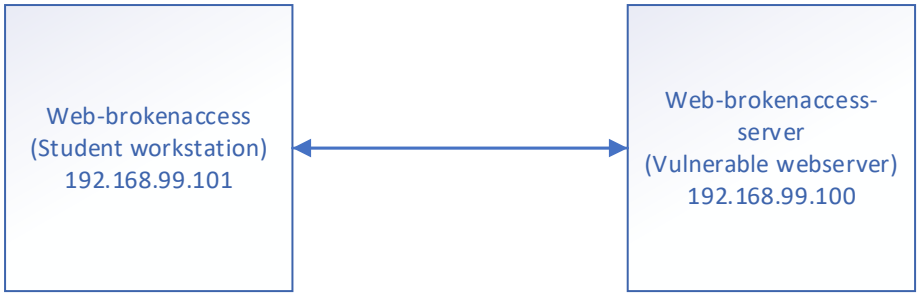
The resulting virtual terminal is connected to the student workstation; you will have OWASP ZAP and Firefox located on this workstation.

There are several accounts that are configured

Username	Password	Systems
admin@juice.org	admin123	Admin for web login
jim@juice.org	ncc-1701	web login
mc.safesearch@juice.org	Mr. N00dles	web login
Ubuntu	ubuntu	Student workstation
Ubuntu	ubuntu	vulnerable server
Ubuntu	ubuntu	Attacker workstation

3 Network Configuration

The student workstation (`web-brokenaccess`) is configured to have IP address `192.168.99.101` while the vulnerable webserver (`web-brokenaccess` -server) is `192.168.99.100;`



4 Lab Tasks

It is assumed that the student has received instruction or independent study on the basic operation of web operations.

4.1 Verify connectivity between student workstation and web server

A simple ping from the student workstation system will be sufficient.

```
ping 192.168.99.100
```

Note: to stop the ping use CTRL + C

4.2 Open Firefox and browse to the web server

At a terminal on the student workstation type:

```
firefox &
```

this will load Firefox, and type in the IP of the web server:

```
http://192.168.99.100
```

Record in Item #1 of your report why firefox might have been chosen to be the web browser used.

4.3 Open & Set up OWASP ZAP

At the terminal of the student workstation, type:

```
owasp-zap &
```

Note: if Firefox is running at the terminal and the "&" was not included then Firefox is not running in the background. Close Firefox and reopen using "Firefox &" at the terminal

OWASP ZAP Application should be open and it should be prompting the user for input.

- OWASP ZAP user input: select "yes, I want to persist this session with the name based on the current timestamp" then click start. This will open ZAP application.
- If you are prompted to "Manage Add-on" click close

4.4 Configure Firefox to use OWASP-ZAP as a Proxy

The objective of this task is to set up OWASP ZAP to be function and to allow the capture of traffic from the web-xss student workstation. Within the preference section of Firefox configure the following steps:

- In the student workstations Firefox, open "Preferences"
- In the find window type "proxy"

- In Network Proxy Setting, select “Settings”
- Select “Manual proxy configuration”
- In the HTTP Proxy section: use “127.0.0.1” and Port “8080”
- Also select “Use this proxy server for all protocols”
- Click “ok” to accept the settings

The above setting ensures that Firefox will use OWASP Zap as the proxy. Perform the following steps to ensure the Firefox is connecting and using ZAP as a proxy

- Refresh the webpage “192.168.99.100”
- A security warning stating “Your connection is not secure” will be displayed.
- This warning message must be accepted. To do that click on “Advanced”
- It will display the SSL certificate and should show a “SEC_ERROR_UNKOWN ISSUE” it is ok to use this cert, click “Add Exception”
- A confirmation window will pop up, confirm the exception by clicking the “Confirm Security Exception”

Record in Item #2 of your report why set up a proxy.

4.5 Accessing Restricted Areas

The objective of this section is to see what access is allowed and to determine even if you can access a restricted area if you will be allowed to do anything. According to OWASP WSTG-ATHZ-01, one of the first tasks when looking at a web site is to see what URLs and directories you have access to. Conducting a basic site survey using a web site crawler is the preferred method for data gathering on a web site.

Example 1 – Path Traversal

The following will allow OWASP Zap to crawl a website to determine if what paths are accessible.

- Open OWASP Zap and perform a site scan on the IP address:
  `192.168.99.100:3000`
- Under alerts you should see a section titled Path Traversal, see if you can see any paths that you would think are password protect or secured.
- Do you see any links to administrative pages?
- Please save your scan results from OWASP Zap, save it to the desktop and title it “traversal.html” if traversal is not an option, please save the entire report.

Example 2 – feedback modification

- In Firefox navigate to the following URL
  `http://192.168.99.100:3000/#/administatration`
- Once you are at the administrative page are you able to remove feedback? Remove all 1-star feedback

Record in Item #3 of your report how should a web server handle access to a administrative page?

4.6 User Creation for Admin Accounts

The objective of this task is to see if there are weakness in the browser that may allow for the creation of administrative accounts when there shouldn’t be an option. According to OWASP WSTG-ATHN-06 is the ability to look at the browser cache and inspection elements to see if it allows for PUT/GET requests and if they can access page or APIs that allow for posting those types of HTTP requests.

Example 1 – Modifying POST request to create an administrator account
Follow the steps listed below to capture and modify a POST request when creating a new user.

- With OWASP-ZAP open, create a new user. Login with the new user.
- This should take the learner to a "User Registration" page. Fill out the following information
  - Email: test@noemail.com
  - Password: Password#1
  - Repeat Password: Password#1
  - Security Question: Name of your favorite Pet?
  - Answer: Dogs
- Find the HTTP request in the History tab called :" http://192.168.99.100:3000/rest/user/whoami". Copy the Authorization field up to but not including the next field of Connection: keep-alive to clipboard.
- Under Sites, navigate to http://192.168.99.100:3000 -> user -> POST:Login. Right click, select Open/Resend with Request Editor.
- Modify the request to GET http://192.168.99.100:3000/api/Users/ Delete the Cookie field, and paste in the Authorization field, which also includes a new cookie. Change the e-mail to a new unique e-mail and hit Send. The response should do a full list of ALL user accounts. Note: You can simply do a POST rather than a GET if you do not want a list of user accounts. Doing this will add the new non-admin account to the system.
- Finally, add the following fields to the body: "role":"admin" and change back to a POST to api/Users. The user will be added, but there is no assurance that the "role" field exists or that the new user is a full administrator. If you get a non-unique user error, change the request to a new e-mail address.
- In the Firefox address bar, go to the following address to verify your new user account:

```
https://192.168.99.100:3000/rest/products/search?q=qwe
rt')) UNION SELECT id, email, password, '4', '5', '6',
'7', '8', '9' FROM Users—
```

- Login with this new account, and validate if it is an administrator by going to:

```
https://192.168.99.100:3000/#/administration
```

- If the page loads correctly, your new account has administrator access.

Record in Item #4 of your report why is the ability to capture a user and in transit modify the request so that they are granted administrative access a bad thing?

Record in Item #5 of your report what are the users registered on the site? What are there names and what is the total count?

4.7 Authentication Bypass

The objective of this task is to see if there are ways past requiring a user to login in the traditional sense. According to OWASP WSTG-ATHN-04, a common method of testing is known as Black-Box Testing using either an Injection or direct page request method for bypassing the authentication schema.

Example 1 – bypass user authentication using double dash

- By simply appending '-- (e.g., admin@juice.org'--) at the end of any user, this allows for a login with any password.
- Try logging in with jim@juice.org'-- and use anything for the password.
- What other users are susceptible to this this type of authentication bypass? User the user list below and see which users will allow this exploit to function.

Example 2 – bypass user email by using a name

- After completing example 1, and seeing the damage having two dashes can cause. Let's see if we can log in with a user name instead of an email address.
- At the log in page type:

```
('%jim%');--
```

- Select any password, did the login work?
- Does any user listed below allow for the login?

User list	Susceptible to which type of attacks
admin@juice.org	
jim@juice.org	
bender@juice.org	
bjoern.kimminich@googlemail.com	
ciso@juice.org	
Support@juice.org	
mc.safesearch@juice.org	

Record in Item #6 of your report fill out the chat and save it to your report.

Record in Item #7 of your report what are the main difference between user bypass and password cracking?

4.8 Account Impersonation

The objective of this section is to see if you can post feedback while log in to one account to a different account. According to OWASP WSTG-ATHZ-02, a common testing method is to ensure that each session is only accessible to the owner of that session and that tasks performed by the session owner is only directly effective that user. This section reviews if it is possible to access functions and resources that should be accessible to a user that holds a different role or privilege.

Example 1 – Forged feedback

- In Firefox, navigate to Customer feedback, if necessary, sign in as jim to view this page.
- In Firefox under comments, leave a comment. Take note of the Author. Leave a Ratting and enter the CAPTCHA result, don't click submit yet submit.
- In OWASP ZAP ensure that you have breakpoints set up and is turned on.
- In Firefox click submit

- In OWASP ZAP you will see that the submit was intercepted. In the body of them message you should see a line that looks like the following:

  ```
  {"UserID:1,"captchaId":1,"captcha":"30","comment":
  "comment entered (***.org)","rating":4}
  ```

  NOTE: the captcha might be different.
  - How to read the above line:
    - UserID:1 – this is the user ID who is posting the comment
    - captchaId:1 – this is the captcha question
    - captcha:30 – this is the captcha answer
    - comment: "comment entered (***.org)" – this is the comment being left
    - "rating":4 – this is the rating that was left
- Modify the User ID to number 1, and allow the message to go through.
- Check feedback and see at the following link and see if the user was the one you logged in as or if it was Jim or Admin who posted the feedback. URL is:

  http://192.168.99.100:3000/#/administatration

Record in Item #8 of your report how can forging feedback by a different user be prevented?

4.9 Account Log in and Password Bypass

The most prevalent and most easily administered authentication mechanism is a static password. The password represents the keys to the kingdom but is often subverted by users in the name of usability. In each of the recent high-profile hacks that have revealed user credentials. According to OWASP WSTG-ATHN-07, the ability to determine the resistance of the application against brute force password guessing using available password dictionaries by evaluating the length, complexity, reuse and aging requirements of passwords. Being able to take a hash and run it through a rainbow table or even a simple google search.

Example 1 – User login

- In section 4.5 the website crawler found several URLs one of them being the authentication details portion. This type of exploit doesn't always work, if the hash value is not present then move on to the next step. On the administration page lets right click the administration page and click "inspect element" you should see a network tab and you should see calls to the following URL

  ```
  https://192.168.99.100:3000/rest/user/authentication-
  details/
  ```
- The responses to the call should look like as follows:

  ```
  [...]{"id":1,"email":"admin@juice-
  sh.op","password":"0192023a7bbd73250516f069df18b500"[.
  ..].
  ```
- Notice the password is hashed. Let's Google the hash to see if we can find the password associated with the hash value.
- Can you find the password passes for any other users? Can those hashes be cracked?

Record in Item #9 of your report are all hashes created equally? Are there one set of hashes that are harder to break?

Record in Item #10 of your report what is salting when it comes to password hashing?

4.10 Exploiting OAuth 2.0

Even if the primary authentication mechanisms do not include any vulnerabilities, it may be that vulnerabilities exist in alternative legitimate authentication user channels for the same user accounts. Tests should be undertaken to identify alternative channels and, subject to test scoping, identify vulnerabilities. The alternative user interaction channels could be utilized to circumvent the primary channel or expose information that can then be used to assist an attack against the primary channel. Some of these channels may themselves be separate web applications using different host names or paths. According to OWASP WSTG-ATHN-10, a common method for attacking is looking at the how parallel websites may be used to authenticate against. in this task the web portal is authenticating against Gmail, also we are looking how default passwords are created using OAuth.

Example 1 – Google OAuth

- In this section we are looking to see if we can authenticate against an external service such as Google OAuth. Login with your own personal Gmail account and see what happens. OK, sent to Google, asked to log in, then prompted for permission for the OWASP Juice Shop. Complete the login, go back to the Juice Shop
- In Firefox right click and select "inspect element" under the network tab, look for a POST request to /api/Users
- A response like the one below should be visible

  {"email":"TEST@gmail.com","password":"bW9jLmxpYW1nQDIwb3JlbW9yLm5ldmV0cw=="}
- The password is a base64 encoded data string. Google a decoder and see what the decode text string is.
- The text string is "moc.liamg@tset" That is the email address backwards. The account creation process is setting default passwords as the Base64-encoded email address turned around backwards.
- Log in using bjoern.kimminich@gmail.com and his Base64-encoded password (his username backwards, encoded). You will need to write his email backwards and then encode it using Base 64.
- Did the log in work?

Record in Item #11 of your report what is OAuth and should third parties be allowed to authenticate users?

4.11 Generate Report

In OWASP ZAP once the tester has found the different vulnerabilities, in the ZAP application under Report on the top and save a HTML report. Save the report to the home directory of the web-inject workstation, call the file "`report_zap`"

Record in Item #12 of your report any interesting finding of the report. Find three areas of interest and explain why they were important and how they may have an impact on the security of this website.

4.12 Review Journal output

This task allows the user to see output that the server would be generating when certain attacks or exploits have run against them. The learner will review journal output on the server.

- On the shell of the web server type the following command:
  - Sudo journalctl -u juice-shop
- Review the output and space bar until the end, record anything of note and if there are

any possible exploits that had ran. Support your claim with output from the journal. Record this at the end of your report under question 13.

5 Stop the Labtainer

When the lab is completed, or you'd like to stop working for a while, run

```
stoplab web-brokenaccess
```

from the host Labtainer working directory. You can always restart the Labtainer to continue your work. When the Labtainer is stopped, a zip file is created and copied to a location displayed by the stoplab command. When the lab is completed, send that zip file to the instructor.

References