INTRO TO BUG HUNTING

Introduction

Have you ever thought about a career in bug hunting?

Not hunting for little critters that roam outside your home or inside but hunting for software bugs, errors in the code, found in an application or service. Hackers that search, find, and report or eliminate bugs within an application or service are called bug hunters. The process of searching, identifying, and eliminating bugs in a system is bug hunting.

As an ethical hacker, one of your tasks is to search for and eliminate bugs in a system. Applications or services that have bugs are prone to vulnerabilities and since malicious actors can use vulnerabilities to gain access to a system, ethical hackers must act quickly to remove the bugs once identified.

In this lesson, you will learn and practice bug hunting. You will learn about the hacking process, ethical hacking tactics, and, lastly, how to be an effective bug hunter and ethical hacker. That said, let's begin!

The Hacking Process

Before we go hunting, let's learn about the hacking process. The hacking process is a combination of ethical hacking tactics for organizational defense. Having a list of tactics to ensure organizational defenses is a helpful resource for ethical hackers and bug hunters. Ethical hackers can use these tactics to ensure the security of a system and identify potential areas of vulnerabilities and software bugs.

The hacking process consists of the following:

- Footprinting
- Scanning
- Enumeration
- System Hacking
- Escalation of Privilege
- Planting Backdoors
- Covering Tracks

---

1. *Footprinting*, also known as the "Reconnaissance" phase, is passive information gathering of targets before active attack activities.
2. *Scanning* is an initial active/passive inspecting technique to gather technical information on target systems.
3. *Enumeration* is the consolidation and gathering of more detailed information on target systems and networks.
4. *System Hacking* is the planning and execution of attacks conducted based on the information gathered in Footprinting, Scanning, and Enumeration.
5. *Escalation of Privilege* is when an attacker is successful and can gain access to the systems/networks of the organization.
6. *Planting Backdoors* is leaving an entry point to a compromised system for easy access to further attack activities.
7. *Covering Tracks* is the act of removing or destroying signs of intrusion and activities performed on a system.

Hunting for Sensitive Data Exposure

In this exercise, you will search for bugs and vulnerabilities in a newly created bank application. The bank application is called *FakeBank Financial*. They recently hired an engineering team to create an application that will make banking much easier.

To ensure that the application is safe and secure, they reached out to you to perform an analysis of the application. In other words, they permitted you to **hack** the site to identify any bugs and vulnerabilities.

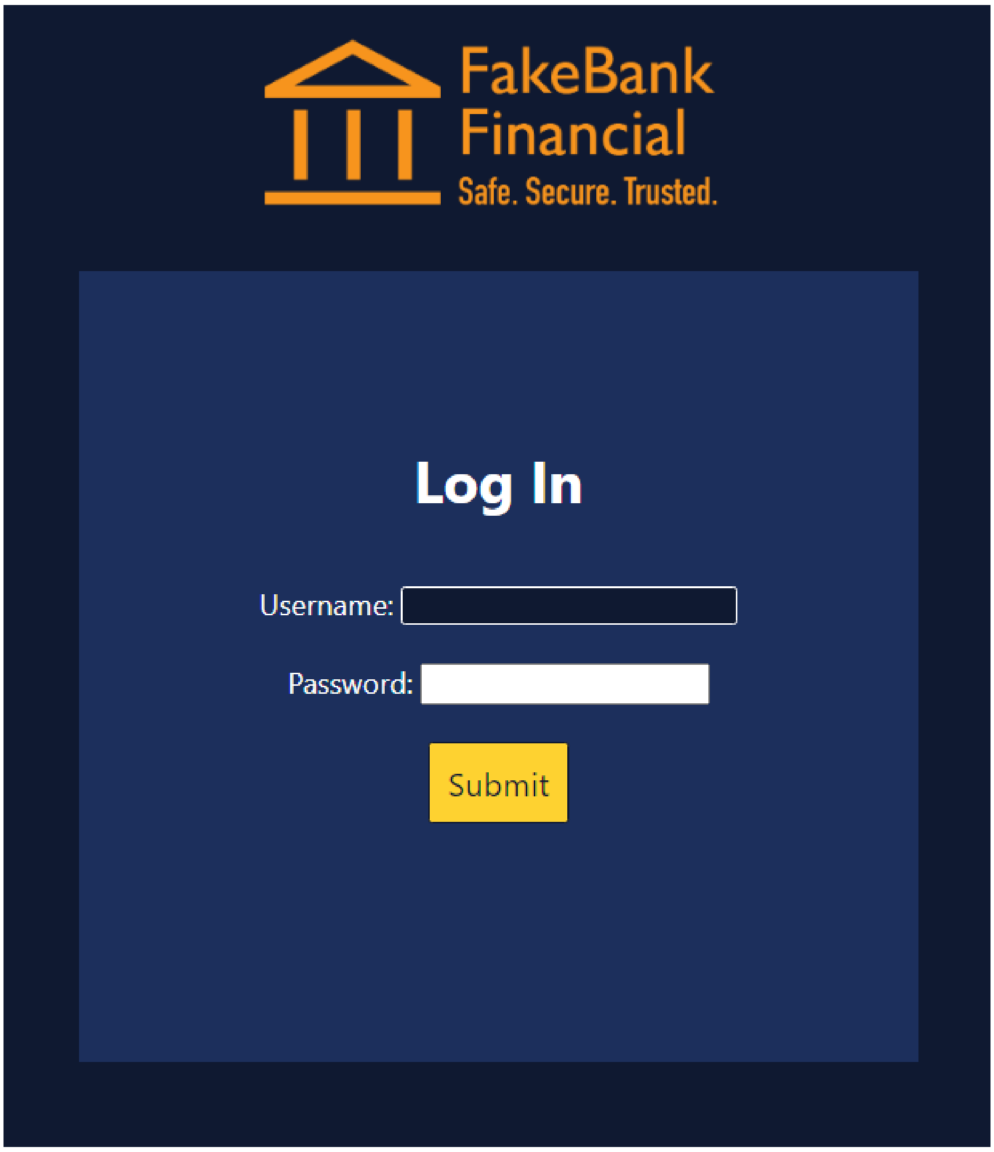
In this task, we will focus on identifying any bugs and vulnerabilities on the login page of FakeBank Financial.

index.html

```
<!DOCTYPE html>

<html>
<head>
  <title></title>
```

```html
  <link rel="stylesheet" href="styles.css">
  <script src="config.js"></script>
  <script src="appletFramework.js"></script>
</head>
<body>
  <div class="centered">
    <img src="fakebank.png" alt="FakeBank Financial logo" class="logo">
    <div id="mainContainer">
      <h1>Log In</h1>
      <p id="errorMessage" style="display: none"></p>
      <br/>
      <form id="formElem">
        <!--
        Username = admin
        Password = @dmin#1234
      -->
      <div>
        <label for="username">Username:</label>
        <input name="username" id="username" type="text" />
      <br>
    <br>
    <label for="password">Password:</label>
    <input name="password" id="password"/>
  <br>
</div>
<button type="submit">Submit</button>
</form>
<p id="injectionOutput"></p>
</div>
</div>
<script src="script_home.js"></script>
</body>
</html>
```



1.
At first glance, the application seems okay.

Let's test the *Username* and *Password* input fields to ensure that the information that returns to us is appropriate.

Input the username and password below into the appropriate fields.

Username: `random@fakefinancial.com`

Password: `random`

Once done, click the Run to continue to the next checkpoint.
Hint

Input the username: `random@fakefinancial.com`

Input the password: `random`

2.
Uh oh! While testing if the input fields return the appropriate messages, we found a bug!

The `password` field does not hide the user's password. Let's try to fix this bug by inspecting the page's code.

We can inspect the page's code by viewing the `index.html` file within the code editor.

Let's navigate to the password field, which looks like the code below:

```
<label for="password">Password:</label>
<input name="password" id="password"/>
```

To fix this bug, we can add a `type="password"` within the `<input/>` tag. The result should be the below:

```
<label for="password">Password:</label>
<input name="password" id="password" type="password"/>
```

Once you have made the change to the code, click the `Run` to save and run the new code and test out the result by using the same credentials.
Hint
Replace the following code:

```
<label for="password">Password:</label>
<input name="password" id="password"/>
```

with the new code:

```
<label for="password">Password:</label>
<input name="password" id="password" type="password"/>
```

3.
Hmm, while navigating to the password section in the `index.html` file, did you notice some vulnerable information above the username section?

It seems like it's the username and password for the admin account. Let's input those credentials into the username and password field to see if they work.

Hint
Input the username: `admin`

Input the password: `@dmin#1234`

4.
Wow, we got in!

This is a major vulnerability in the application. We were able to log in as someone else, but, most importantly, we were able to log in as admin and view personal information such as:

- email address
- phone number
- social security number
- checking balance
- saving balance

Let’s go ahead and remove the credentials from the `index.html` file.

Let’s also go back to the login page. Place the following URL in the URL tab:

`http://localhost:8000`

Once done, click the Run to continue to the next checkpoint.
Hint
Remove the following information from the `index.html` file:

```
        <!--
        Username = admin
        Password = @dmin#1234
      -->
```

5.
Let’s do one last vulnerability test, a SQL injection.

A *SQL injection* is a common vulnerability affecting applications that use SQL as their database language. A hacker can use their knowledge of the SQL language to cleverly construct text inputs that modify the backend SQL query to their liking. They can force the application to output private data or respond in ways that provide intel. To specify, we will be doing a boolean-based injection that involves SQL statements that can confirm TRUE/FALSE questions about the database.

To perform a SQL injection, we will insert the following statement in the password input field:

```
'1' OR '1' = '1'
```

Once inserted, click the `Submit` button in the application and check the page to identify if any private data has been exposed.

NOTE: You might need to zoom out to view the entire page.

Once done, click the `Run` to continue to the next checkpoint.
Hint
Input `'1' OR '1' = '1'` for the password. If the application requires a username, try the username we discovered in the previous task.

6.
We found another vulnerability! This website is prone to SQL injection. To resolve this vulnerability, there are two main things we can do; sanitization and parameterized queries.

Sanitization removes dangerous characters from user input, and *parameterized queries* are queries with placeholders used as parameters during the code execution.

Nonetheless, for the sake of this lesson, we'll pause here.

That said, great job! You've hunted, identified, and removed bugs and vulnerabilities for FakeBank Financial login page. You were able to gain a high-level overview of bug hunting and practice it on a web application.

Though we are pausing here, this does not mean that FakeBank Financial is completely free of bugs and vulnerabilities. There may be plenty of other hidden bugs and vulnerabilities within the application. Explore the application and its code to identify any other missed issues.

If you are satisfied with all that you found, click `Run` and then `Next` to continue with the lesson.
Hint
Select `Next` to continue with the checkpoint.

Challenge: Hunting for Web Vulnerability

Here's a challenge! You're free to skip this exercise if you choose. To skip, just select Next to continue.

However, if you choose to accept this challenge, your task is to identify and note as many vulnerabilities as you can find within the website. Once you identify and note all the vulnerabilities on the site, compare what you've noted with the `site_vulnerabilities.txt` file.

Hint: Try `jake@newcomsolutions.com`

Index.html

```
<!DOCTYPE html>
<html>
<head>
  <title></title>
  <link rel="stylesheet" href="styles.css">
  <script src="config.js"></script>
  <script src="appletFramework.js"></script>
</head>
<body>
  <div class="centered">
    <img src="newcom.png" alt="NewCom Financial logo" class="logo">
    <div id="mainContainer">
      <h1>Log In</h1>
      <br/>
      <form id="formElem">
      <div>
        <label for="username">Email:</label>
        <input name="username" id="username" type="text" />
        <p id="usernameError" class="errorMessage" style="display: none"></p>
      <br>
    <br>
    <label for="password">Password:</label>
    <input name="password" id="password" type="password" />
    <p id="passwordError" class="errorMessage" style="display: none"></p>
  <br>
</div>
<button type="submit">Submit</button>
```

```
</form>
</div>
</div>
<p id="injectionOutput"></p>
<script src="script_home.js"></script>
</body>
</html>
```

site_vulnerabilities.txt

1. Password field does not hide user's password

2. When you place `jake@newcomsolutions.com` in the email field and a wrong password, the error message inform us that there is a user with the email `jake@newcomsolutions.com`

3. The site is prone to SQL injection. Inserting the SQL code '1' OR '1' = '1' within the password field outputted all the account information.

Conclusion

Throughout this lesson, you've learned about ethical hacking and bug hunting, the hacking process, and practiced searching, identifying, and eliminating bugs and vulnerabilities within an application.

As mentioned earlier, as an ethical hacker, one of your tasks is to search for, and possibly eliminate, any vulnerabilities in a system. Applications or services that have bugs or vulnerabilities will give access to malicious actors to do some serious damage to the system. That said, ethical hackers must act quickly and adapt.