

**COURSE:   
INTRODUCTION TO CYBER SECURITY (CY101)**  
**PROGRAM:   
BS CYBER SECURITY   
SEMESTER:  
4TH  
BATCH:  
2023-SPRING  
DEPARTMENT:   
SOFTWARE ENGINEERING  
  
SUBMITTED BY:  
ZAINAB ARIF  
2023s-BCYS-028  
  
  
SUBMITTED TO:**INSTRUCTOR**: MUHAMMAD WAQAR**

https://www.linkedin.com/in/zainabaarif?utm\_source=share&utm\_campaign=share\_via&utm\_content=profile&utm\_medium=android\_app

**PROJECT TITLE:**  
**“BROKEN AUTHENTICATION”**



# **WHAT IS BROKEN AUTHENTICATION**

Authentication is a critical process that ensures only authorized users can access specific systems or services. However, when authentication mechanisms are improperly implemented or inadequately secured, they can become a point of vulnerability, leading to broken authentication. This security flaw is one of the most common causes of data breaches and cyberattacks. The term "broken authentication" refers to weaknesses or flaws in the authentication mechanisms, which attackers can exploit to impersonate legitimate users or bypass security controls.

## **2. Understanding Broken Authentication**

Authentication is the process of verifying the identity of a user, system, or entity, typically through credentials such as usernames, passwords, and biometric data. In modern applications, authentication is fundamental for user privacy, data integrity, and system security. When authentication mechanisms are flawed, attackers may gain unauthorized access to sensitive information, leading to severe security breaches.

Broken authentication occurs when there are weaknesses in the authentication flow or process, which can be exploited by malicious actors to bypass security controls. These flaws might arise from a variety of factors, such as improper session management, weak password policies, or inadequate encryption of credentials.

A key aspect of broken authentication is its potential for providing attackers with unauthorized access to systems or resources, often with escalated privileges. This vulnerability can compromise not only the user's personal data but also affect entire systems, applications, and networks.

## **3. Causes of Broken Authentication**

Several factors contribute to broken authentication vulnerabilities. These include design flaws, misconfigurations, and insecure coding practices. Common causes include:

## **1. Weak Passwords**

Weak or easily guessable passwords (e.g., "123456" or "password") are one of the most common causes of broken authentication. If users choose weak passwords or if the system allows them, attackers can easily gain unauthorized access through brute-force attacks.

## **2. Insufficient Session Management**

Improper handling of user sessions can lead to unauthorized access. For example, if a session remains active after the user logs out or fails to expire after a reasonable time, attackers can hijack the session and impersonate the user.

## **3. Predictable or Insecure Credentials**

Many systems rely on hard-coded or predictable credentials (e.g., "admin "). These credentials are easy for attackers to guess and can be exploited to gain unauthorized access to systems.

## **4. Insecure Password Storage**

Storing passwords in plain text or using weak hashing algorithms makes it easier for attackers to retrieve and misuse passwords. Without proper encryption or hashing, stolen passwords can lead to significant data breaches.

**4. Common Attack Vectors**

## **1. Brute-Force Attacks**

Brute-force attacks occur when an attacker systematically attempts all possible combinations of passwords until the correct one is found. This attack is highly effective when weak passwords are used and when rate-limiting protections are not in place.

## **2. Credential Stuffing**

Credential stuffing involves using lists of previously breached usernames and passwords to attempt logins across various services. Many users reuse the same credentials across multiple platforms, making this type of attack especially effective.

## **3. Session Hijacking**

Session hijacking occurs when an attacker gains access to a user's active session, often through stealing session cookies. By doing so, the attacker can impersonate the user without needing to authenticate again.

**5. Real-World Examples of Broken Authentication**

## **1. GitHub Password Reuse Incident**

In 2014, a GitHub security incident revealed how the reuse of credentials contributed to the exposure of user data. Attackers exploited a common weakness in password reuse, compromising multiple accounts by using stolen credentials from previous data breaches.

**6. Impact of Broken Authentication**

## **1. Data Breaches**

Unauthorized access to user accounts can lead to the theft of sensitive data such as personal information, financial records, and login credentials. This information can be sold on the dark web or used for identity theft.

## **2. Financial Loss**

For businesses, a successful attack due to broken authentication can lead to direct financial losses from stolen funds, as well as indirect losses from legal liabilities, fines, and damage to reputation.

## **3. Implement Multi-Factor Authentication (MFA)**

Enabling MFA ensures that an attacker must compromise more than just a password to gain unauthorized access. MFA can involve something the user knows (password), something they have (a smartphone or hardware token), or something they are (biometric data).

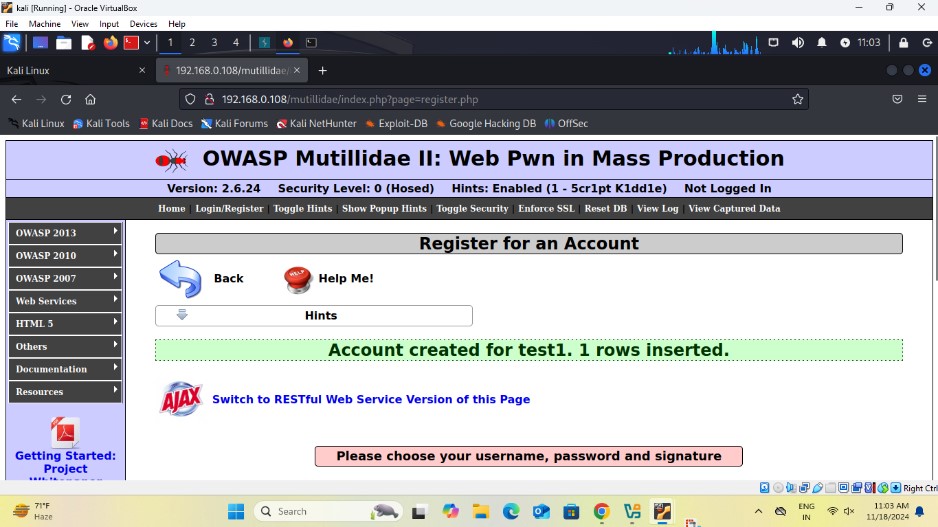
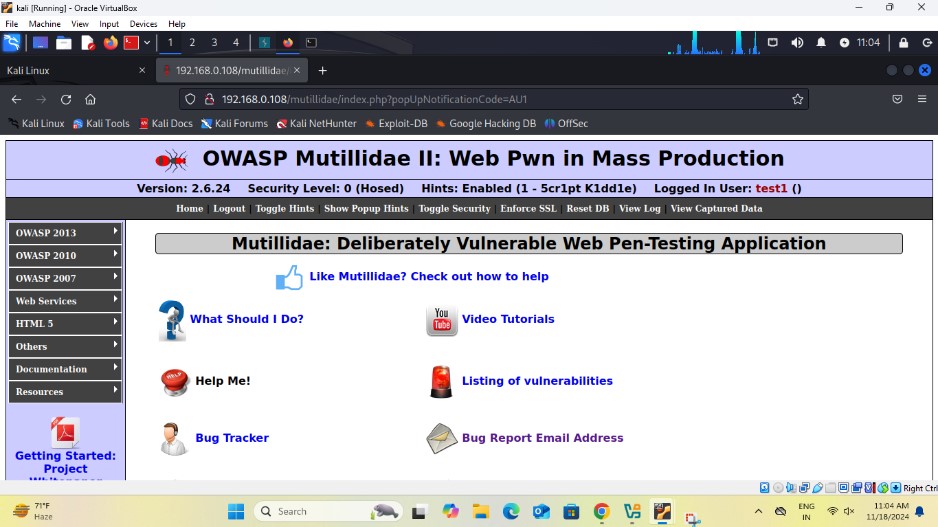
## **4. Secure Session Management**

Sessions should be securely managed to prevent session hijacking. This includes using secure, HttpOnly cookies for session tokens, implementing session timeouts, and ensuring that sessions are properly invalidated upon logout.

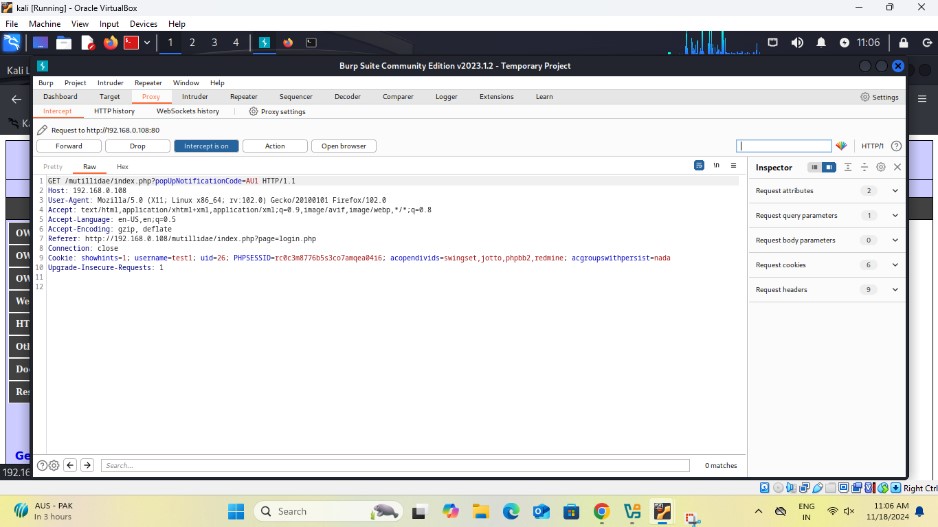
**SCENARIO BASED ATTACK PERFORMING:**

## **BROKEN AUTHENTICATION VIA COOKIES**

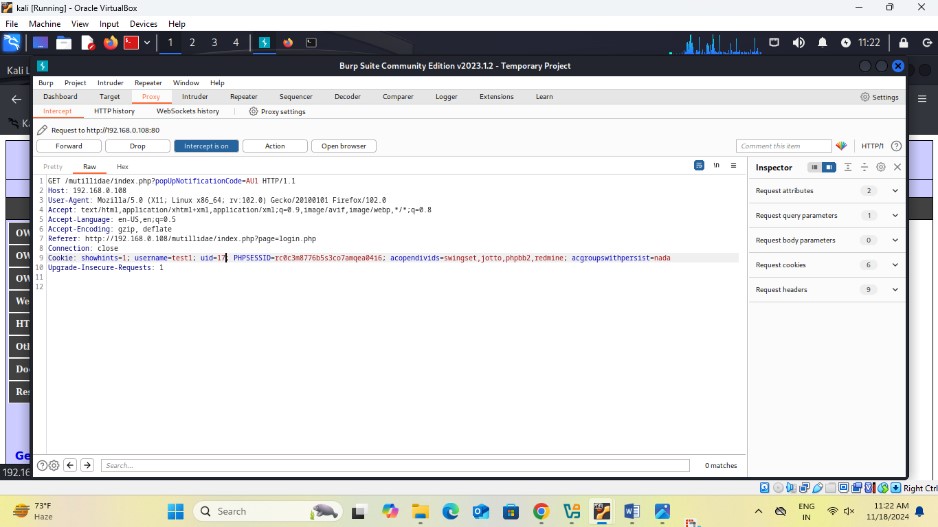
Firstly open Owasp bwa and go to mutillidae II,then OWASP 2013 then select broken authentication by using via cookies then register and create account and login as test1

Firstly intercepting the request for user test1 after intercepting it shows all data and shows most important is uid of user test1 is **26.**



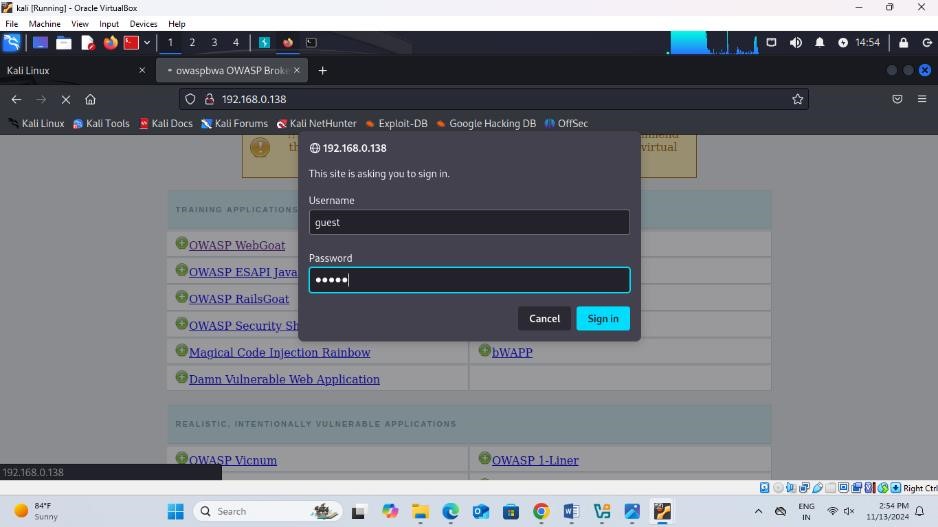
Then changing uid for test1 25 to 17 ,then using this uid account see the change in user account as rocky.



This attack shows: If cookies are not checked properly by the server, hackers can change them to access someone else's account.

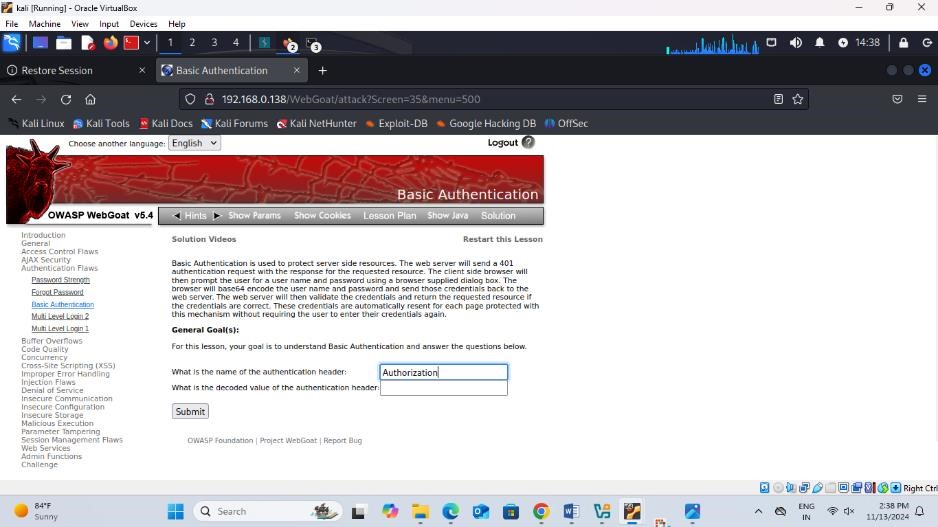
## **BROKEN AUNTHENTICATION USING BASIC AUTHENTICATION**

Simply open owasp bwa and click on webgot and start the webgot.

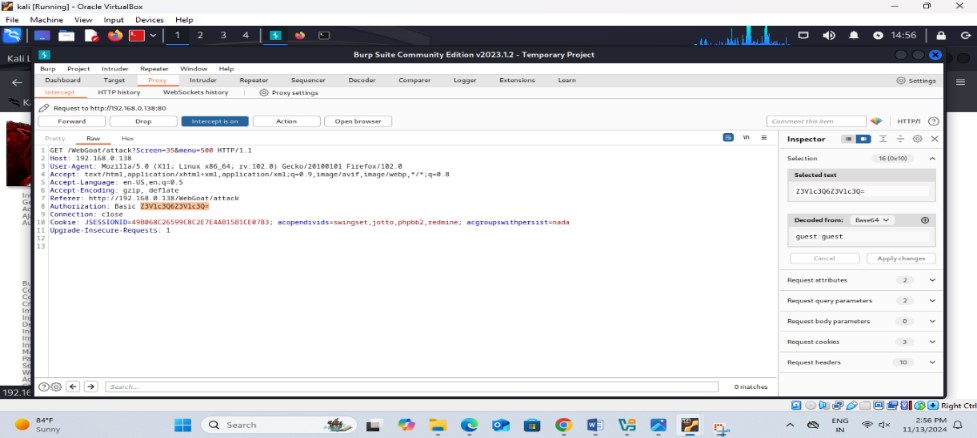


Then we go to Authentication flaws and select basic authentication and answer the question 1, In a **Basic Authentication** scheme, when a user tries to access a resource, the server will send a request for authentication (a 401 Unauthorized status). The client (browser) will then send the **Authorization** header with the base64-encoded credentials (username:password) to prove their identity.

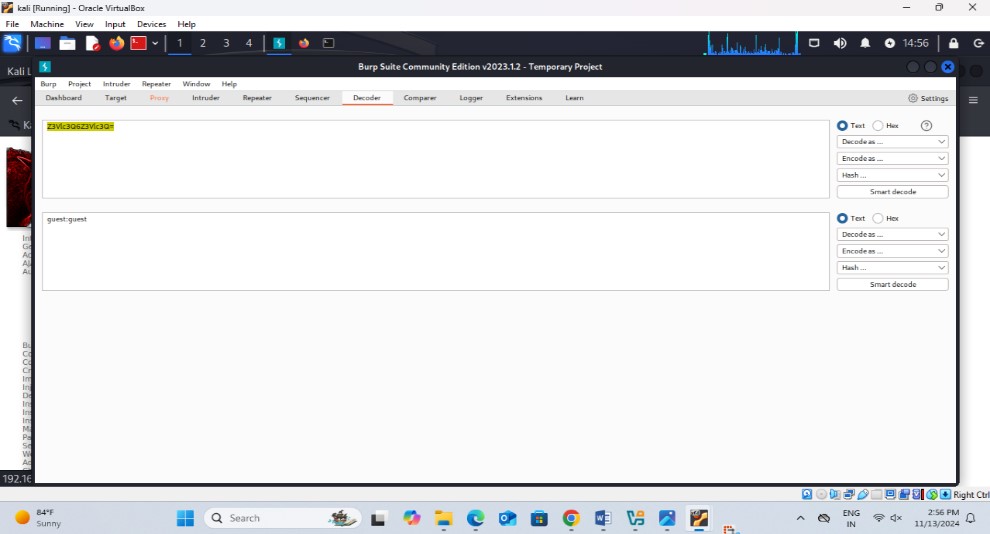
but 2nd question is little bit difficult how to solve it following steps mention



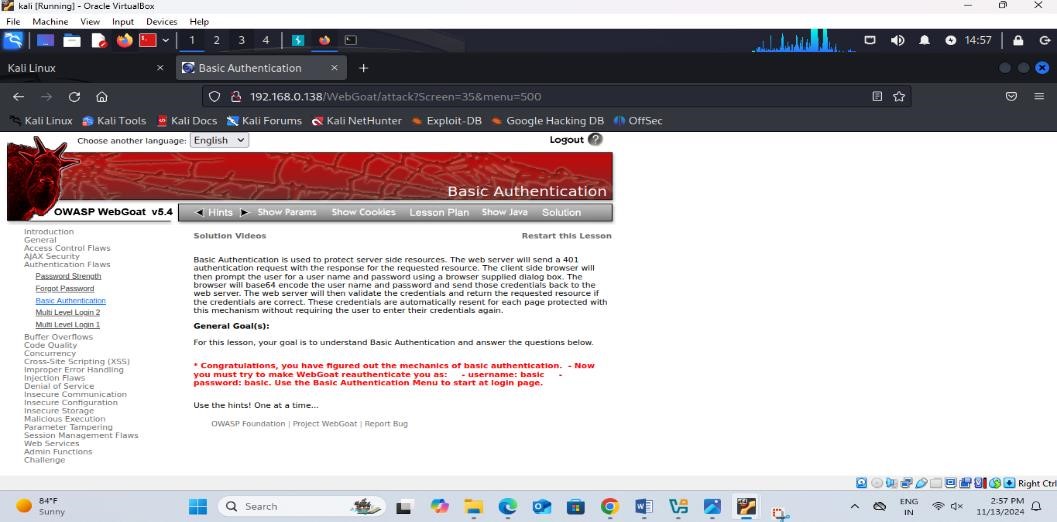
Then intercept the request copy the authorization encrypted ciphertext code format .



Encrypted format copy and and request send to decoder and paste it in decoder and decode as base 64 and we get the ans as **guest:guest.**



Paste it in the decoded value for authentication header is guest:guest and submit and get sign and congratulation



This attack exploits **Basic Authentication** by intercepting the HTTP request and decoding the **base64-encoded credentials** (e.g., guest:guest). The attacker then reuses these credentials to authenticate and gain unauthorized access. It highlights the weakness of transmitting credentials in an easily decodable format.

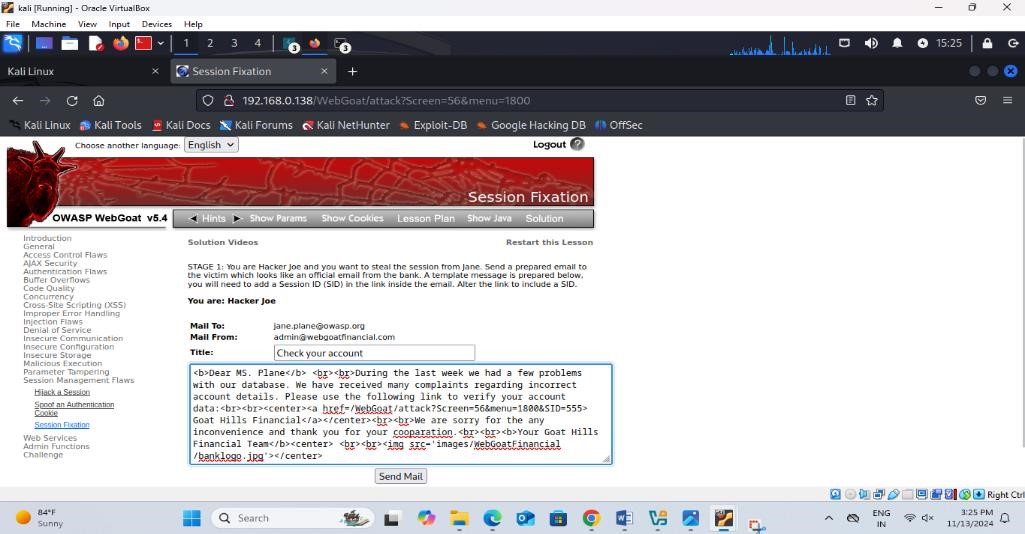
# **SESSION FIXATION**

Session fixation is a specific type of vulnerability within the broader category of **broken authentication**. This vulnerability occurs when an attacker can set or manipulate a valid session ID before a user logs in, allowing the attacker to hijack the session after the user authenticates.

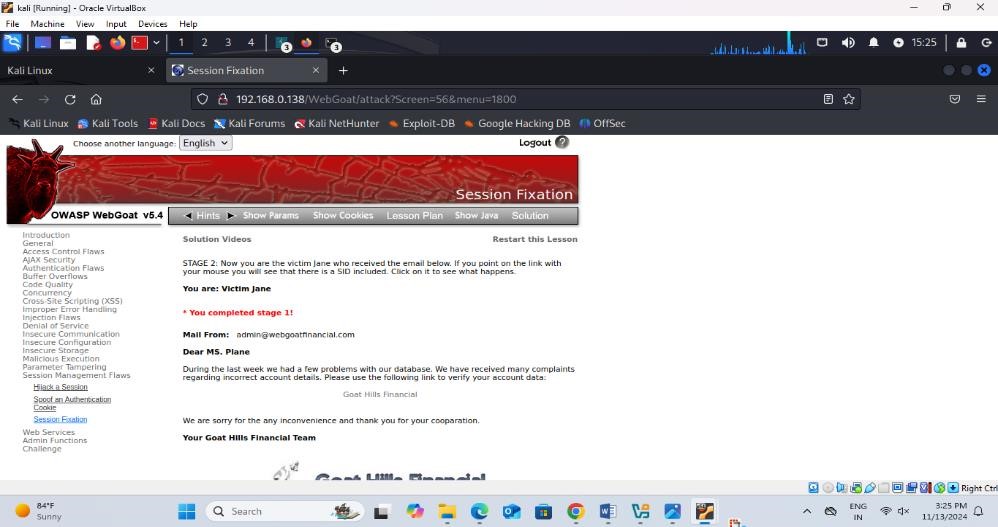
## How Session Fixation Works

1. Victim Logs In: The victim unknowingly authenticates to the application using the attacker-specified session ID.
2. Unauthorized Gains Access: Since the victim's session is now associated with the attacker's chosen session ID, the attacker can access the user's session and impersonate them.

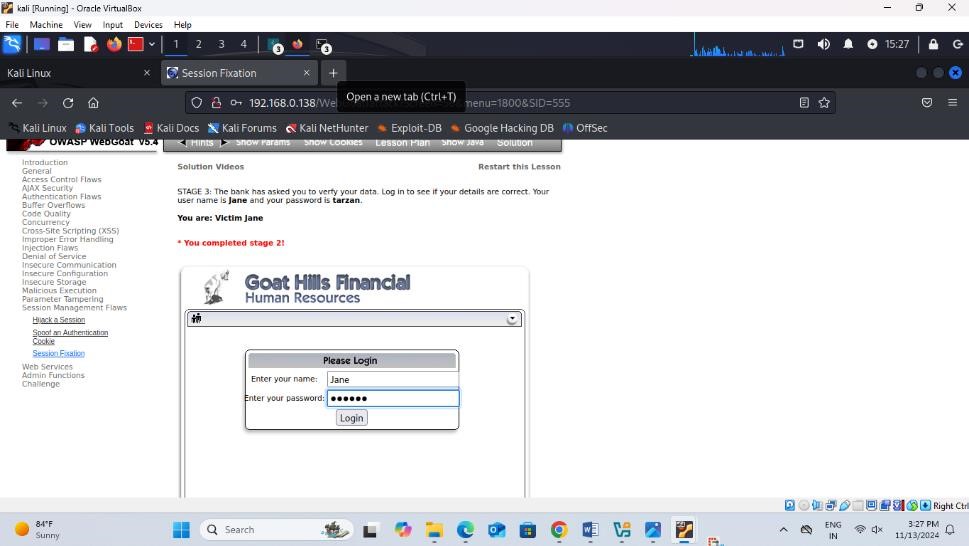
First start OWASP BWA and go to webgoat and click on session management flaws and session fixation and person change or add session id in that mail box and send to other.



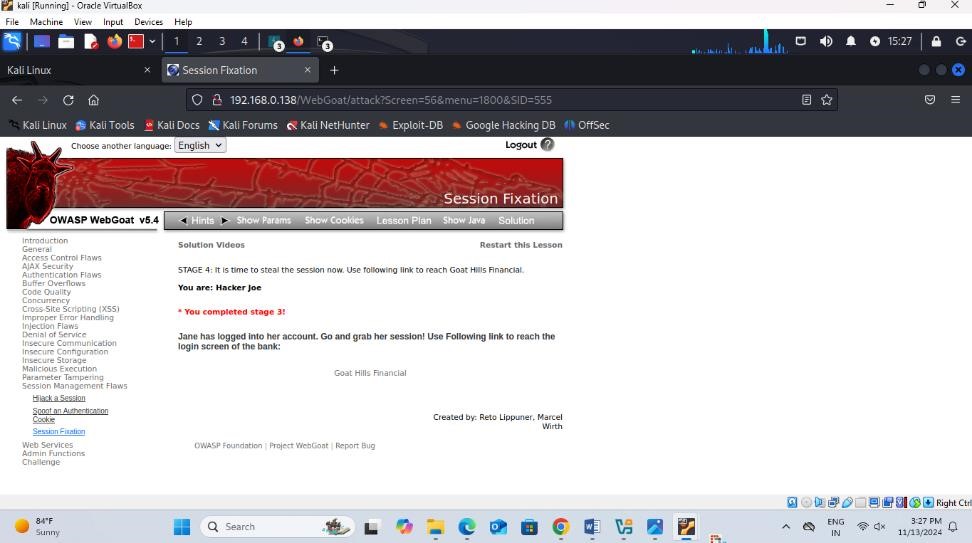
And another person receiveing the mail and click on it where the unauthorized



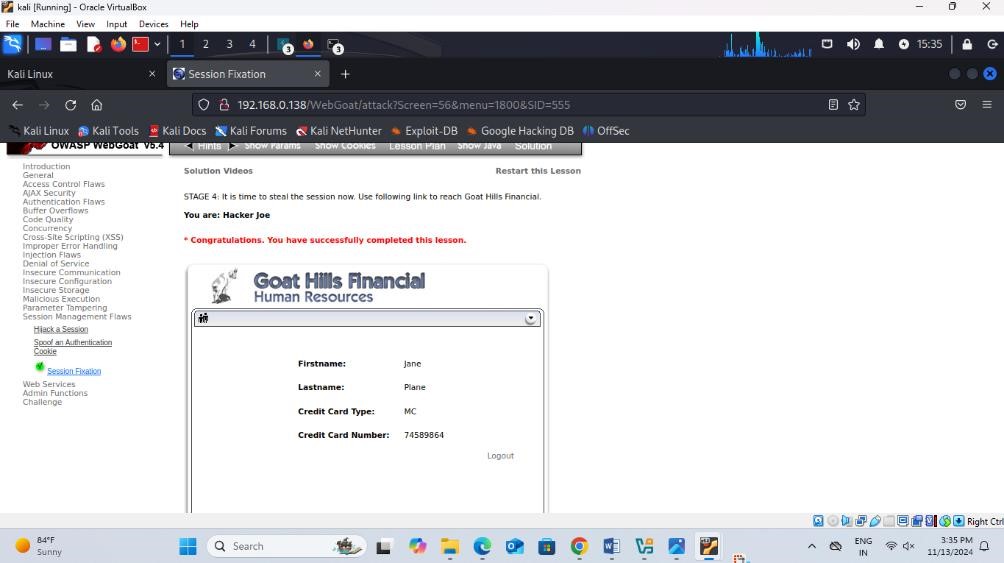
Getting username and password then login there account.



After the logging account and add session id in url SID=555.



Any one or unauthorized person access the data other person.



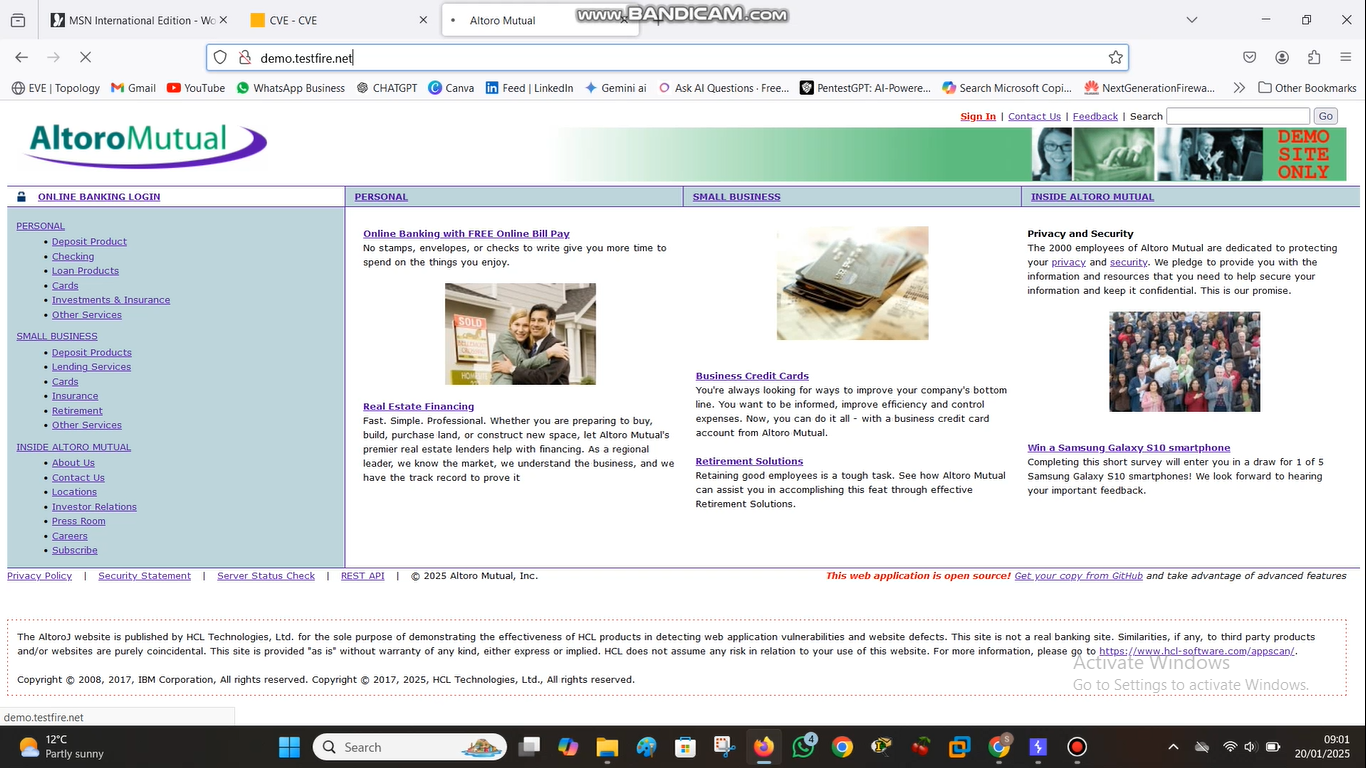
## **BROKEN AUNTHENTICATION ON ONLINE BANKING USING BURP SUITE**

### **Steps to Perform Broken Authentication Attack Using Burp Suite**

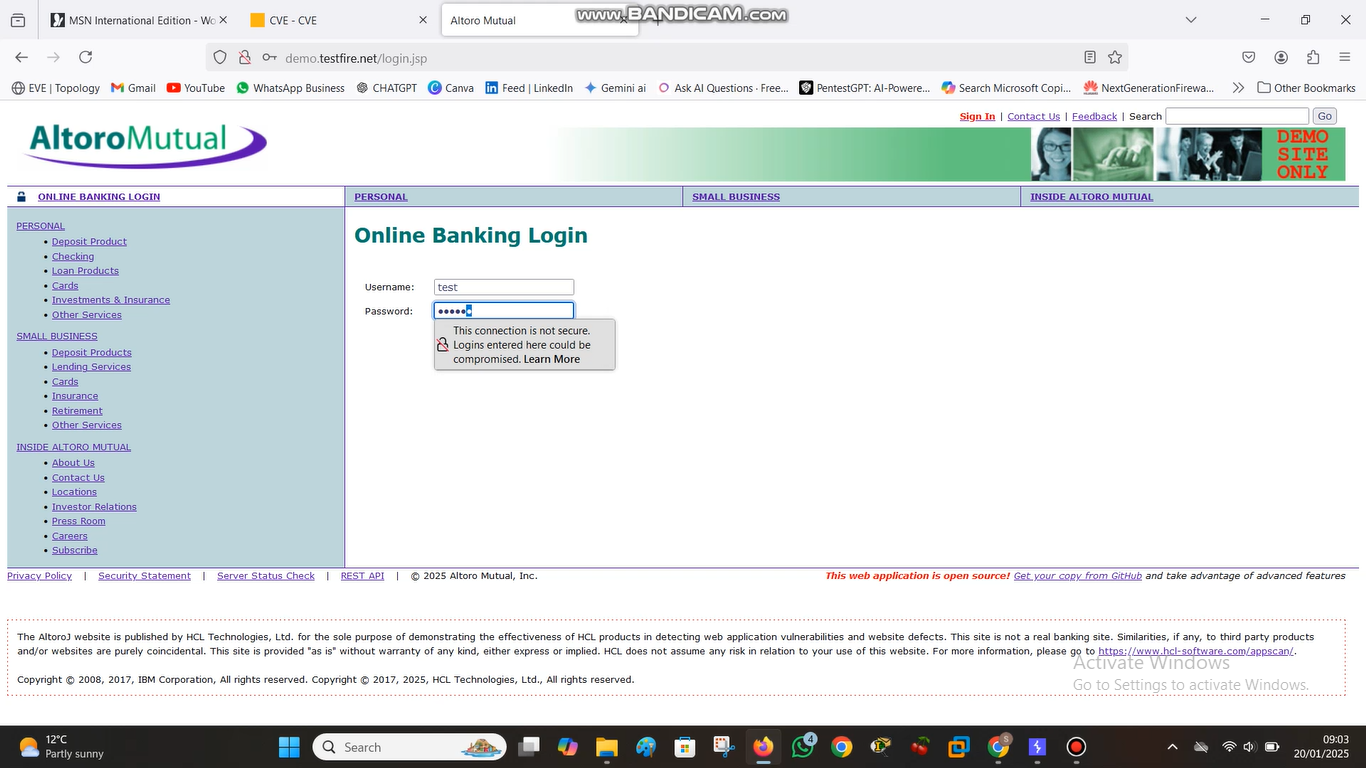
#### **Step 1: Setup Environment**

1. Open **Burp Suite (Community Edition)**.
2. Configure the **proxy settings** in your **Firefox browser** and turn on the proxy to ensure traffic is routed through Burp Suite.

#### **Step 2: Navigate to the Test Application**

1. Go to Google and search for demo.testfire.net.
2. Access the website to test for broken authentication vulnerabilities. 

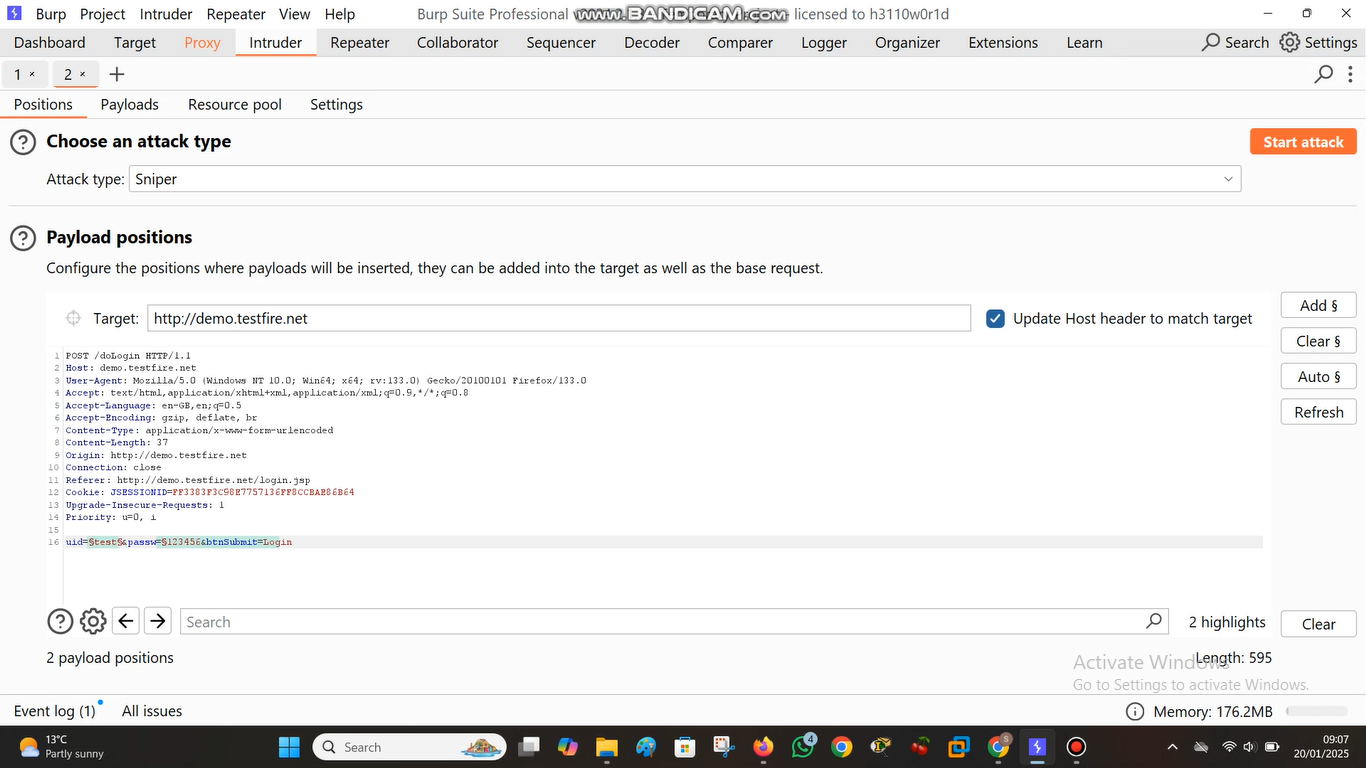
#### **Step 3: Test Burp Suite Interception**

1. Enable **Intercept** in Burp Suite (Proxy > Intercept > Intercept is on).
2. Perform a dummy login attempt by entering any random username and password.  
   
3. Confirm that Burp Suite is capturing the requests.

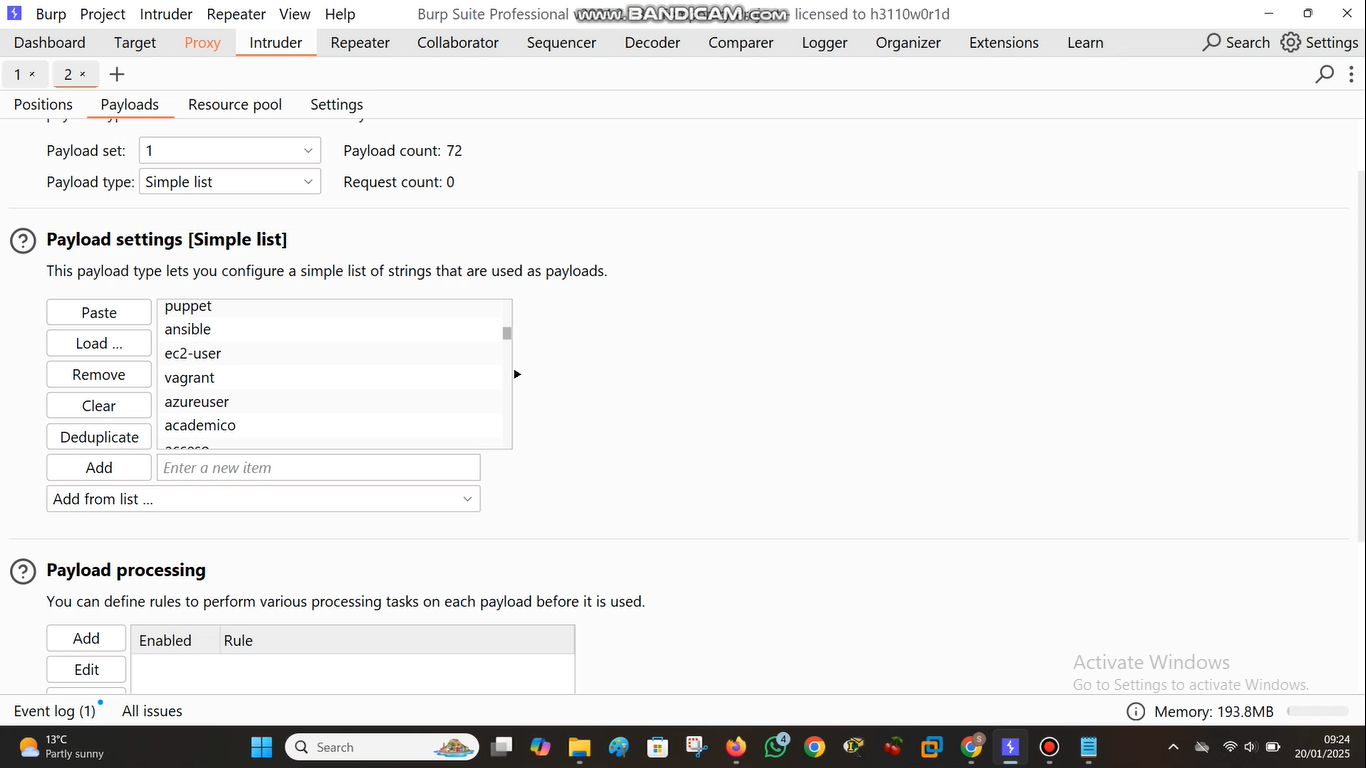
#### **Step 4: Analyze the HTTP Login Request**

1. Go to Burp Suite’s **HTTP History** tab (Proxy > HTTP History).
2. Look for the login request made to demo.testfire.net. 
3. Right-click on the login request and select **Send to Intruder**.

#### **Step 5: Configure Intruder for Brute Force Attack**

1. Go to the **Intruder** tab in Burp Suite. 
2. Highlight the **username** and **password** fields in the request and add $ markers around them to designate them as payload positions.

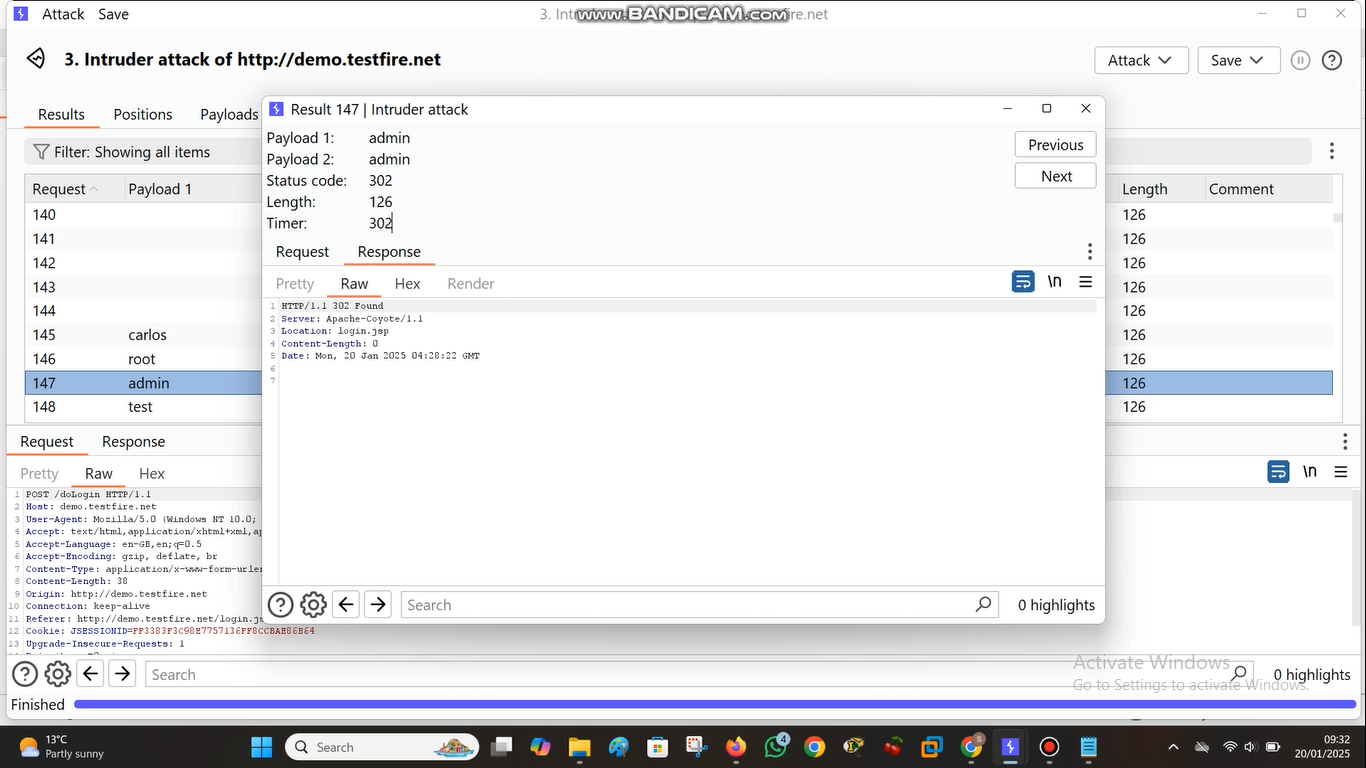
#### **Step 6: Prepare Payloads**

1. Create two payload lists—one for potential usernames and another for passwords.
   * You can create or load these lists in a text file.
2. In the Intruder tab, go to the **Payloads** section.
3. Assign the **first payload position** (username) to your list of usernames. 
   * Assign the **second payload position** (password) to your list of passwords.

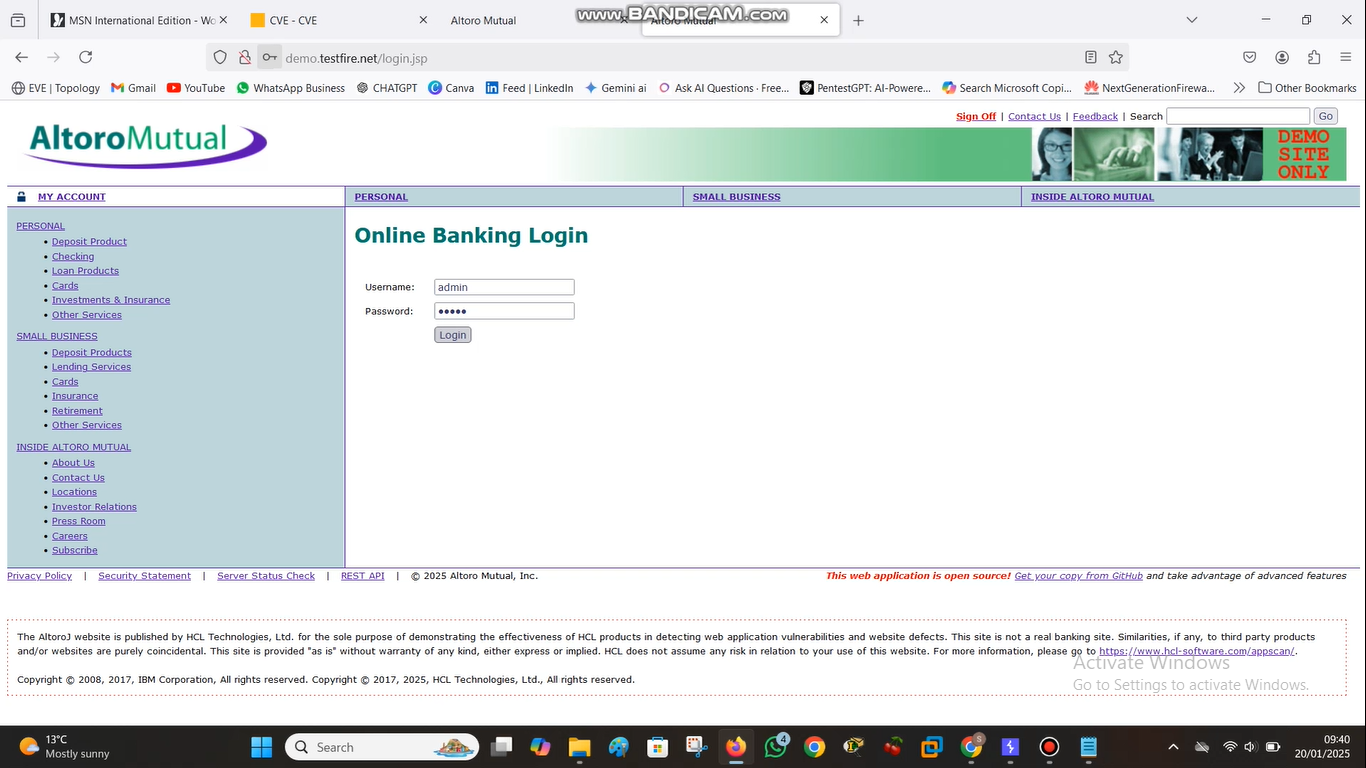
#### **Step 7: Start the Brute Force Attack**

1. Under the **Attack Type** section, select **Cluster Bomb**.
2. Click on **Start Attack** to begin the brute force attack.

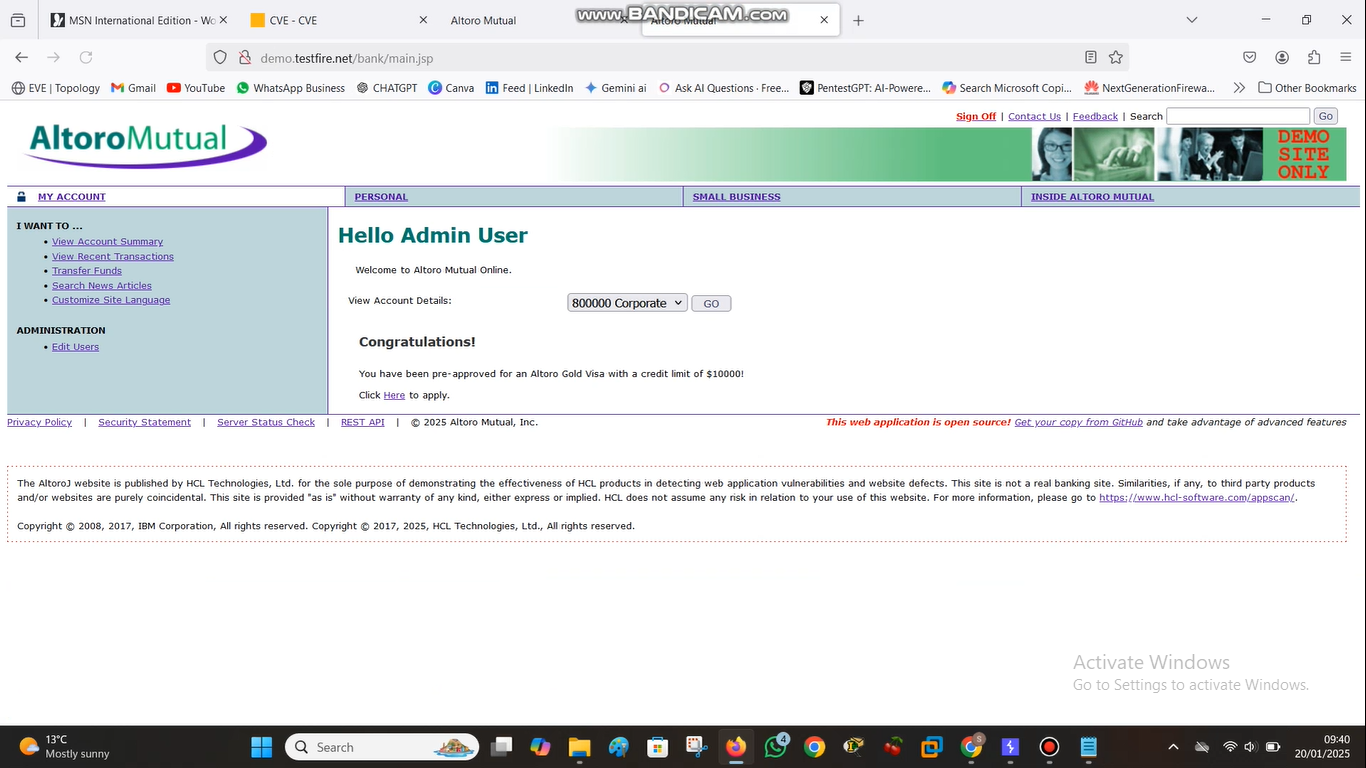
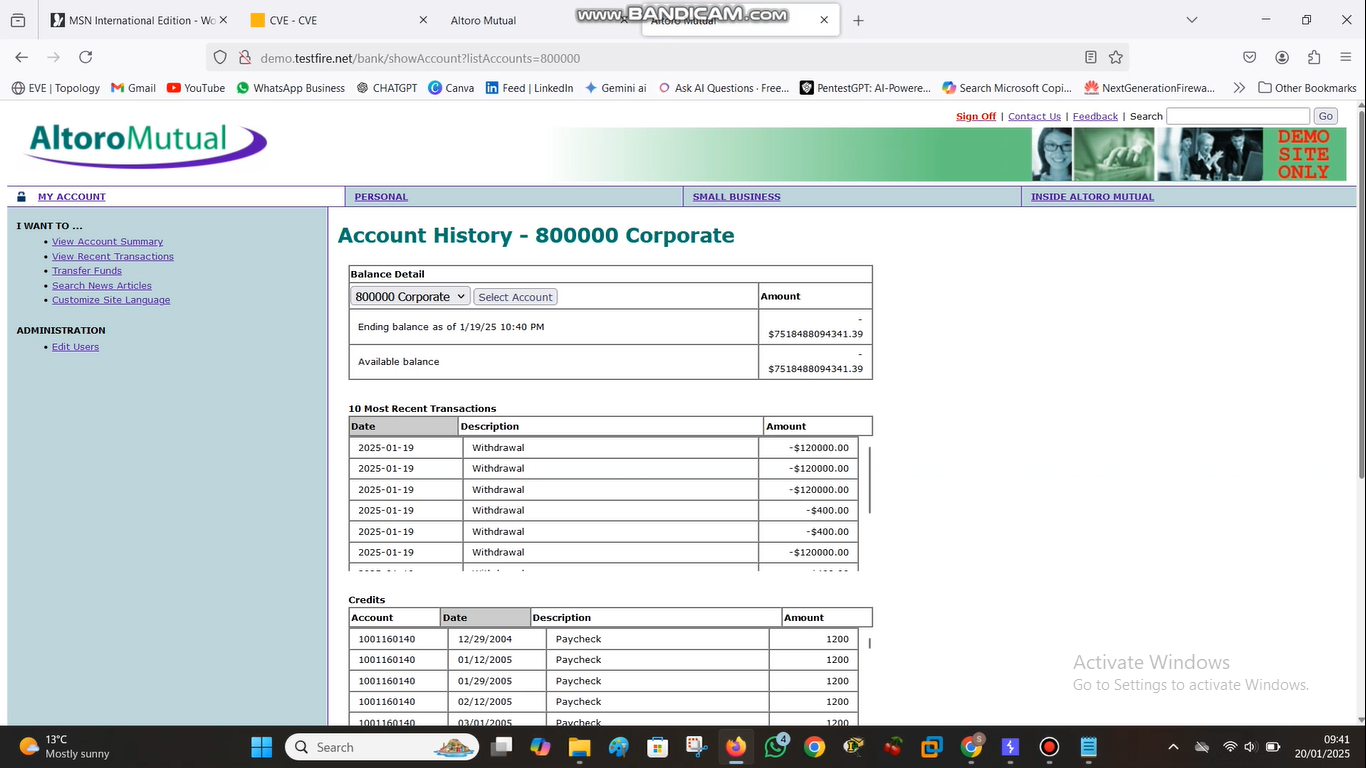
#### **Step 8: Analyze Results**

1. Once the attack completes, review the results.
2. Identify the combination of **username and password** that results in a successful login (check for status codes or success messages in the response). 

#### **Step 9: Log In with Discovered Credentials**

1. Right-click the successful request and **Send to Browser**.
2. Open the URL in your browser.
3. Enter the discovered credentials (e.g., admin for both username and password). 

#### **Step 10: Access Sensitive Data**

1. Upon successful login, view the account details. 
2. Highlight how this exposes sensitive data due to poor authentication mechanisms.  
   

## **8. Conclusion:**

Broken authentication remains one of the most significant security risks for modern applications and systems. Whether due to weak passwords, poor session management, or lack of proper authorization checks, broken authentication can expose sensitive data and lead to devastating breaches.

By understanding the causes and attack vectors associated with broken authentication, organizations can implement best practices and secure their authentication mechanisms. Enforcing strong passwords, using multi-factor authentication, and securing sessions are critical steps in protecting users and systems from malicious attacks. Through ongoing vigilance and security improvements, organizations can significantly reduce the risk posed by broken authentication vulnerabilities.

**CERTIFICATE COMPLETE WITH THIS COURSE:**   
  
