**AIM**: To perform Bypass Authentication on <https://demo.testfire.net>

**Introduction**:

**AUTHENTICATION BYPASS** - This type of SQL Injection tries to gain access to a database by inserting SQL Queries within the input fields of a login application, so that the security mechanism is bypassed.

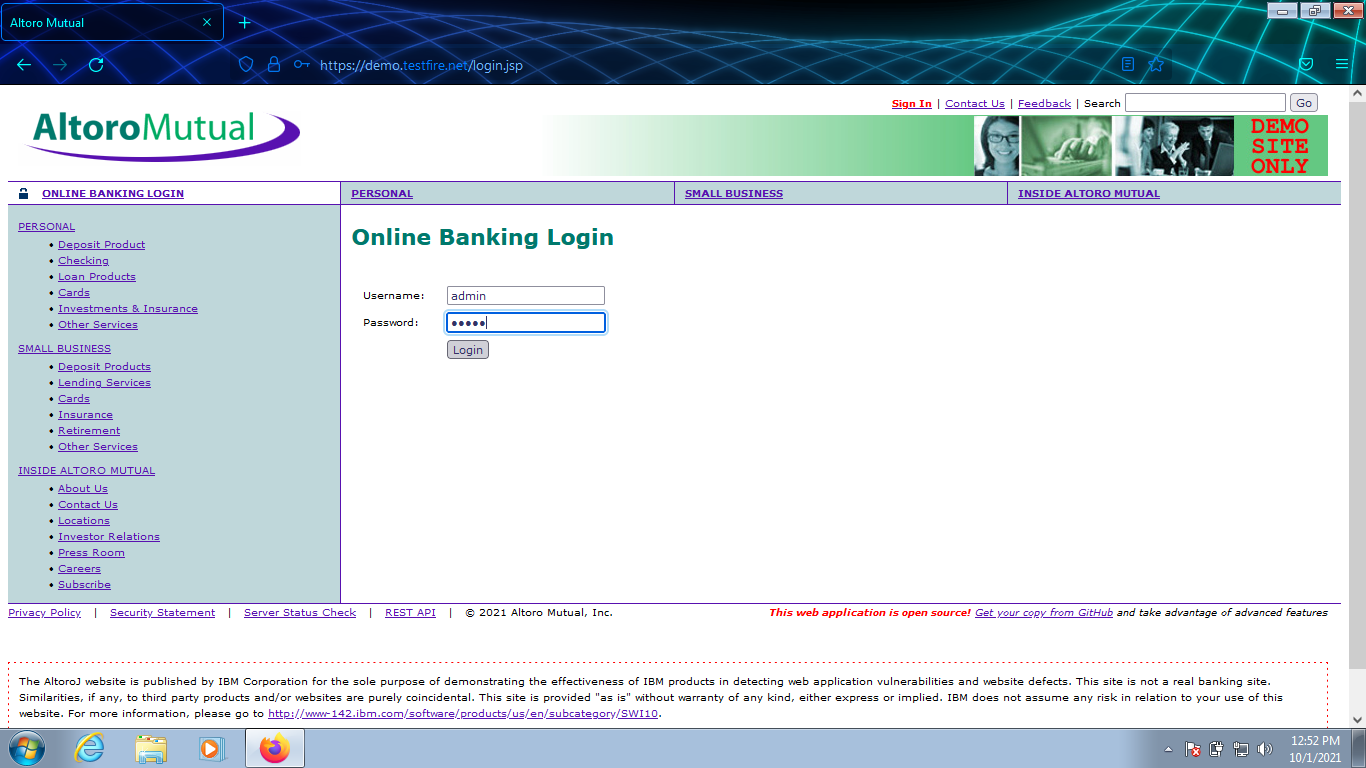
**Procedure**:

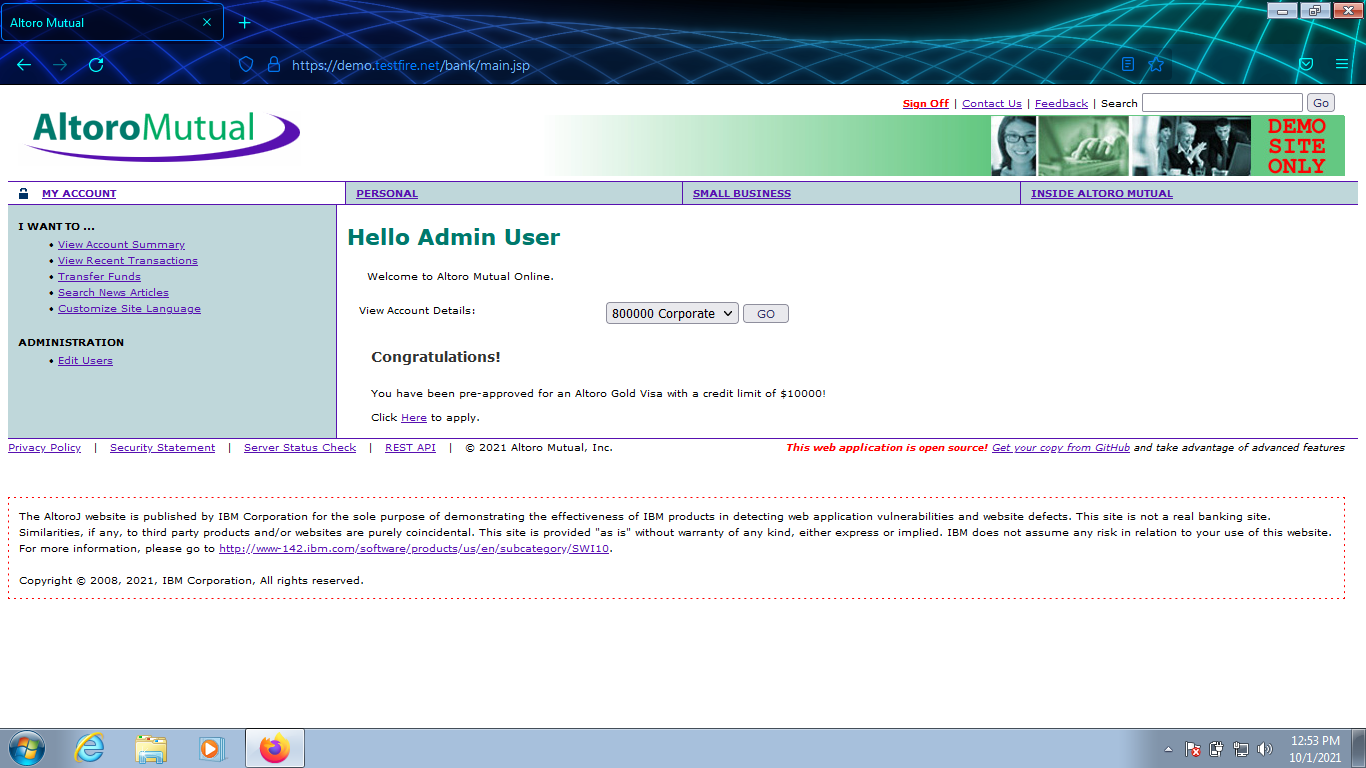
1. Let's take as victim example this demo banking account login page:  
     
   <http://demo.testfire.net/bank/login.aspx>



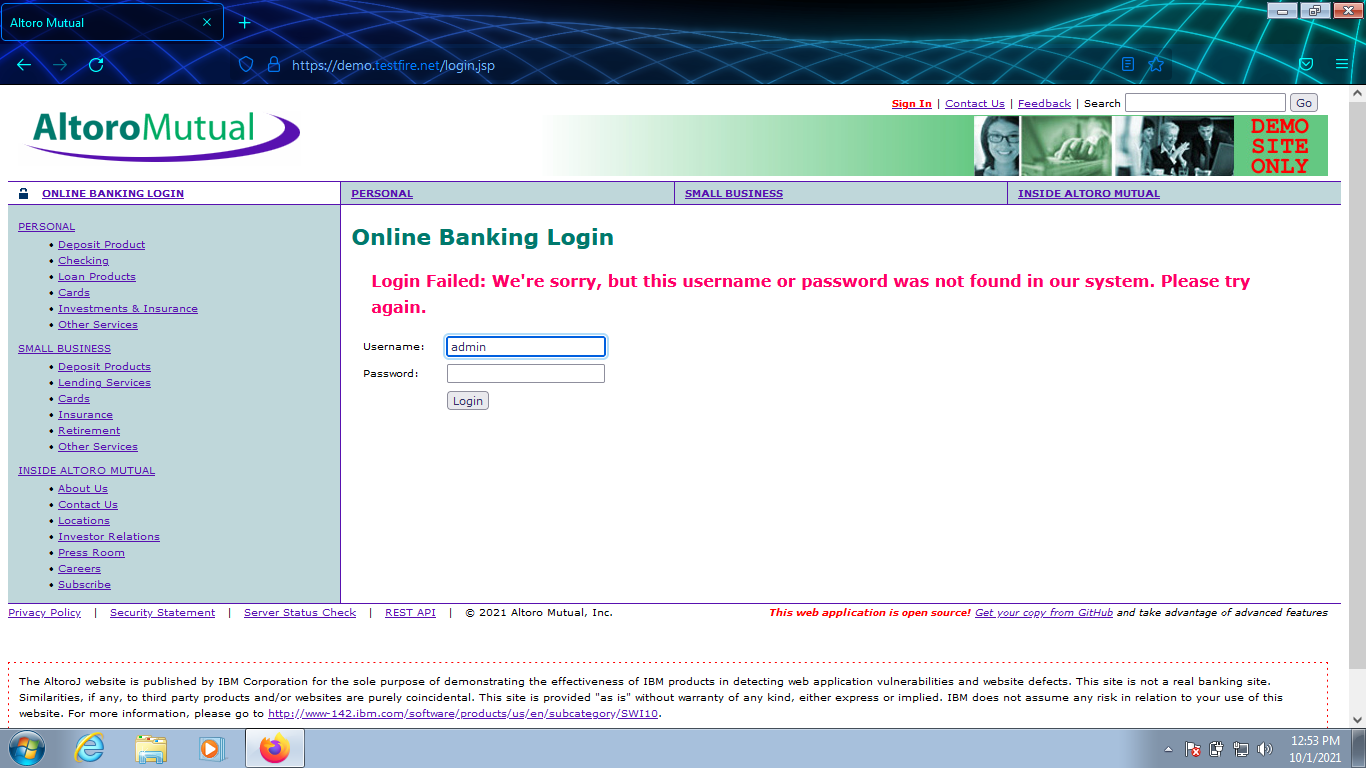
1. Let's start by examining an usual login SQL query:  
     
   *SELECT account FROM USERS WHERE username = 'admin' AND password = 'admin'*

1. The boolean statement *username = 'admin' AND password = 'admin'* is only TRUE when both boolean operators are TRUE (1 AND 1 = 1).
2. In this way, entering the correct credentials admin/admin for both the username and password fields the access is correct:

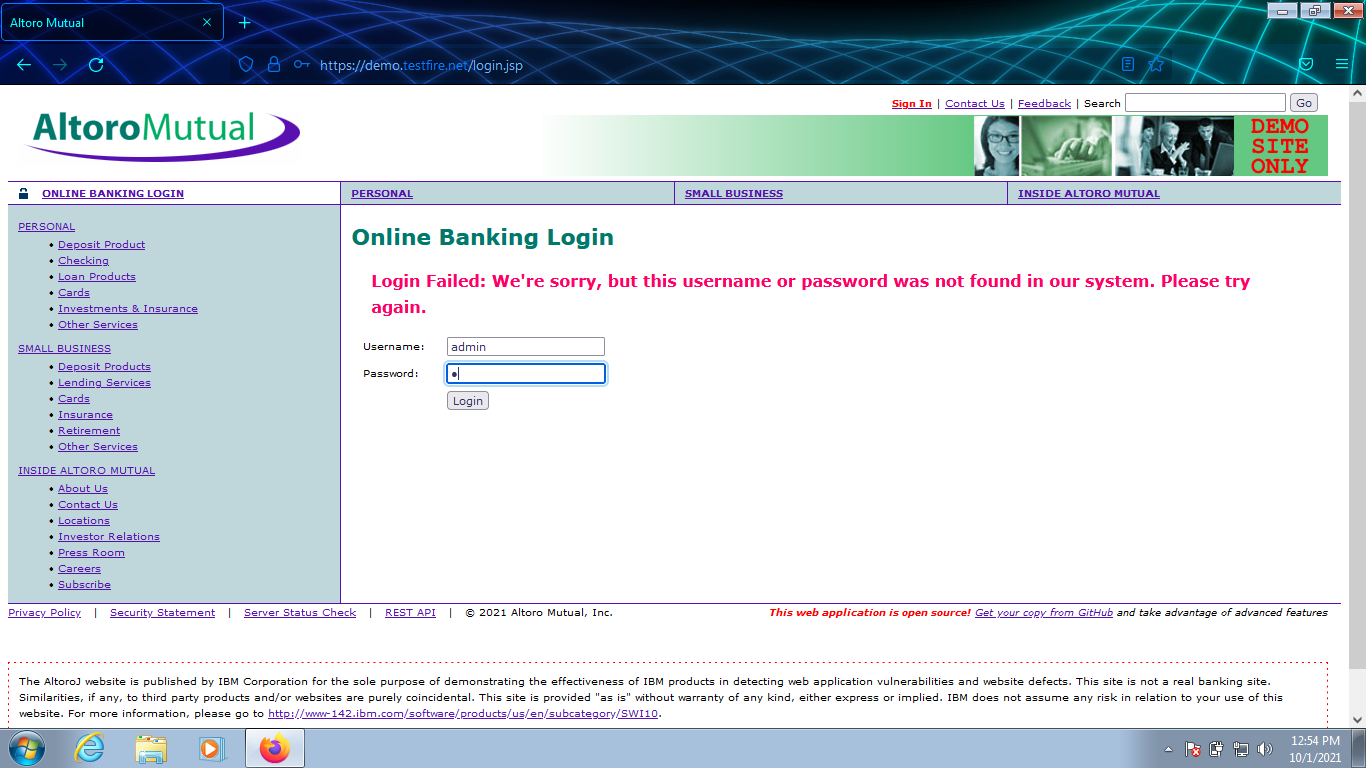


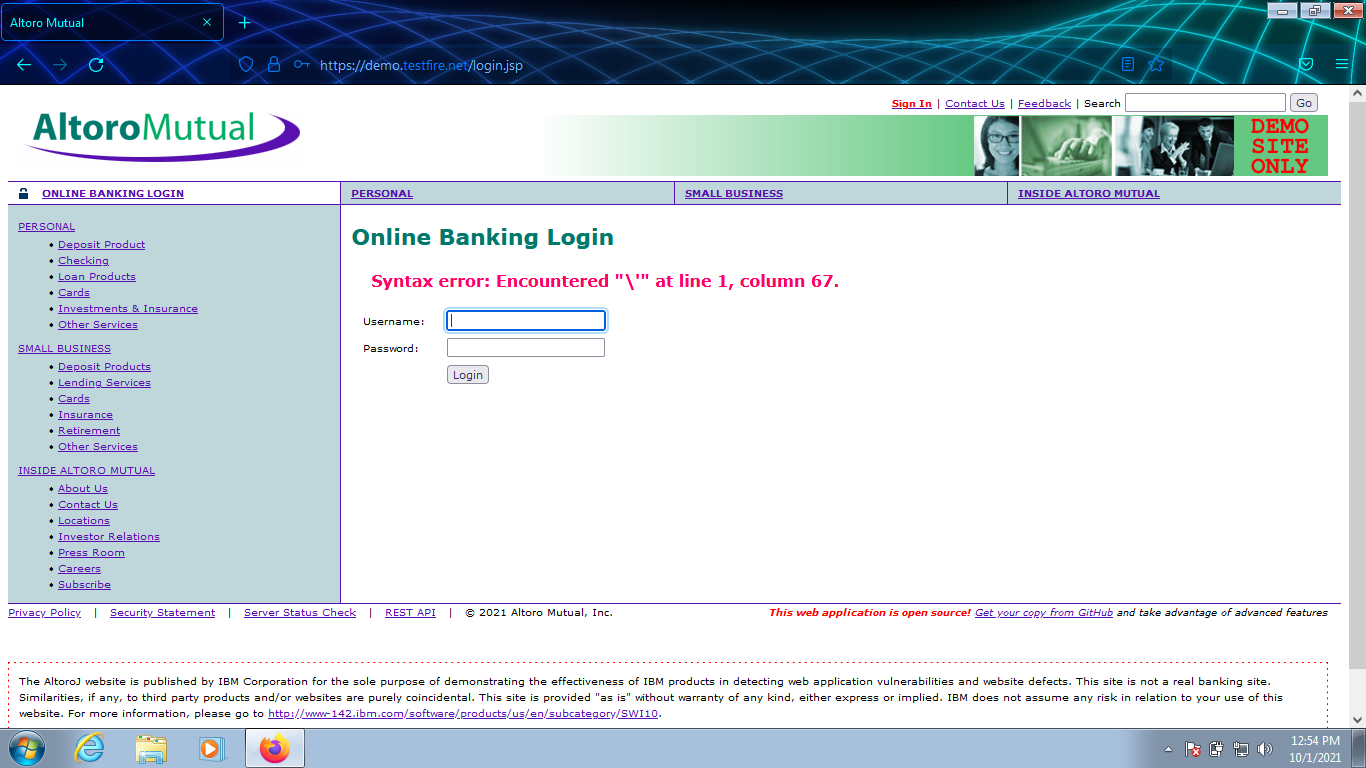


1. However, if one the operators is FALSE (password = '12345') the whole statement falls to FALSE (1 AND 0 = 0):  
     
   *SELECT account FROM USERS WHERE username = 'admin' AND password = '12345'*
2. So, entering incorrect credentials like admin/12345 the login process fails:

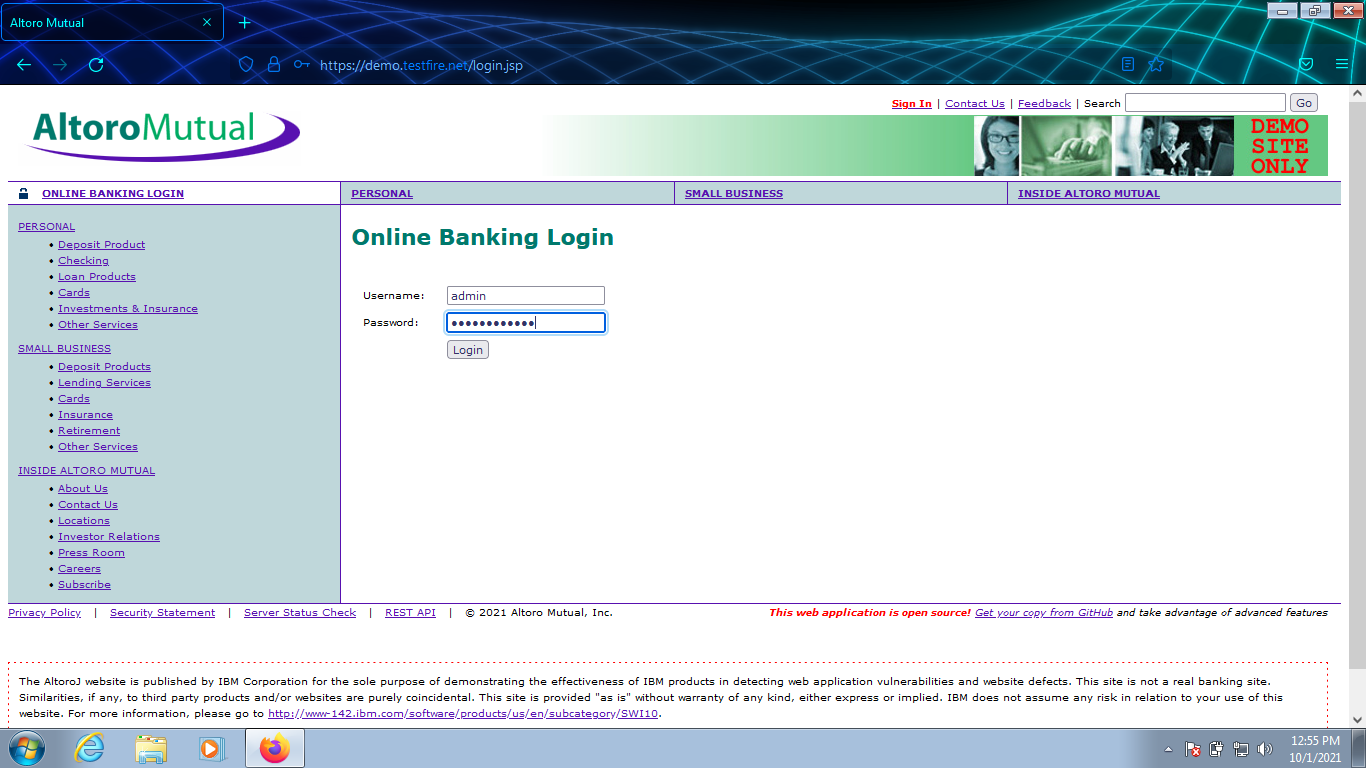


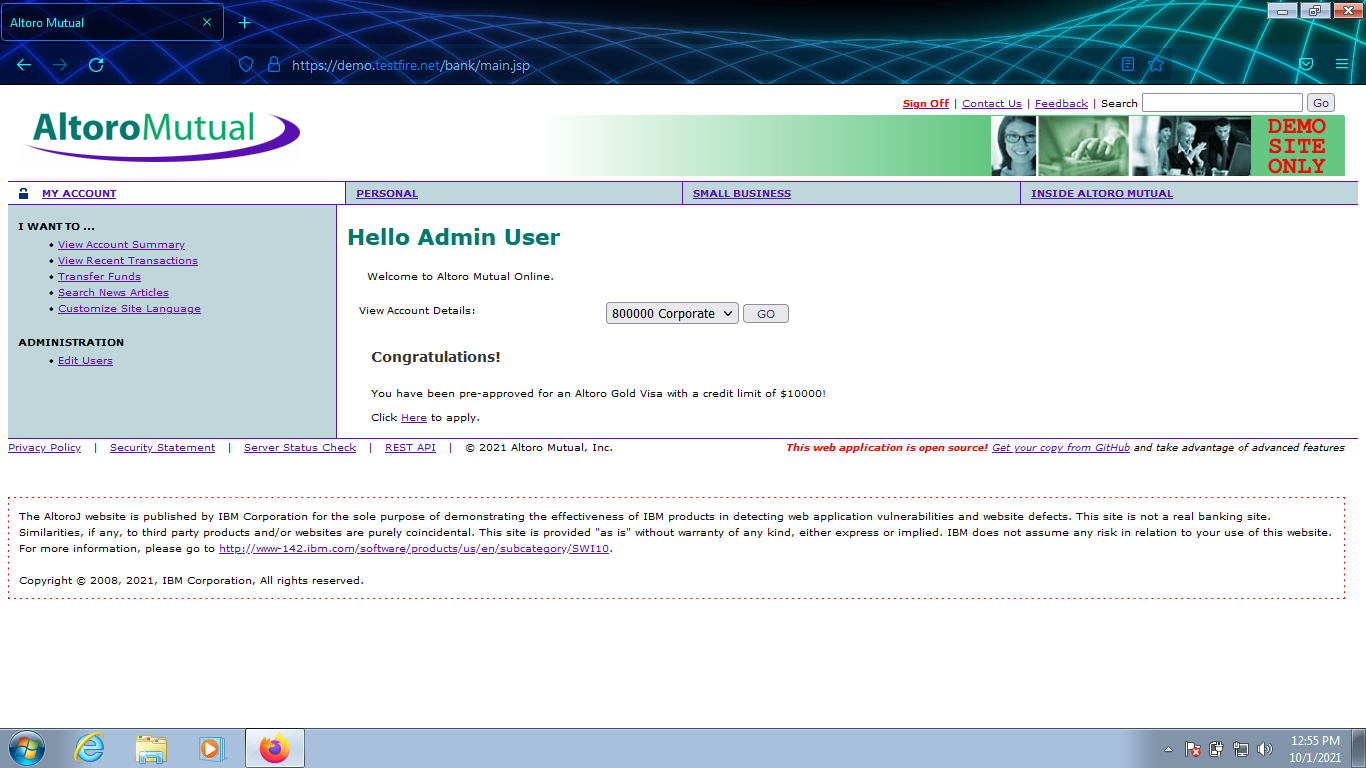
1. By the way, entering a simple quotation mark character (') is a good way to discover if the application is prone to SQL Injection, like it is the case:



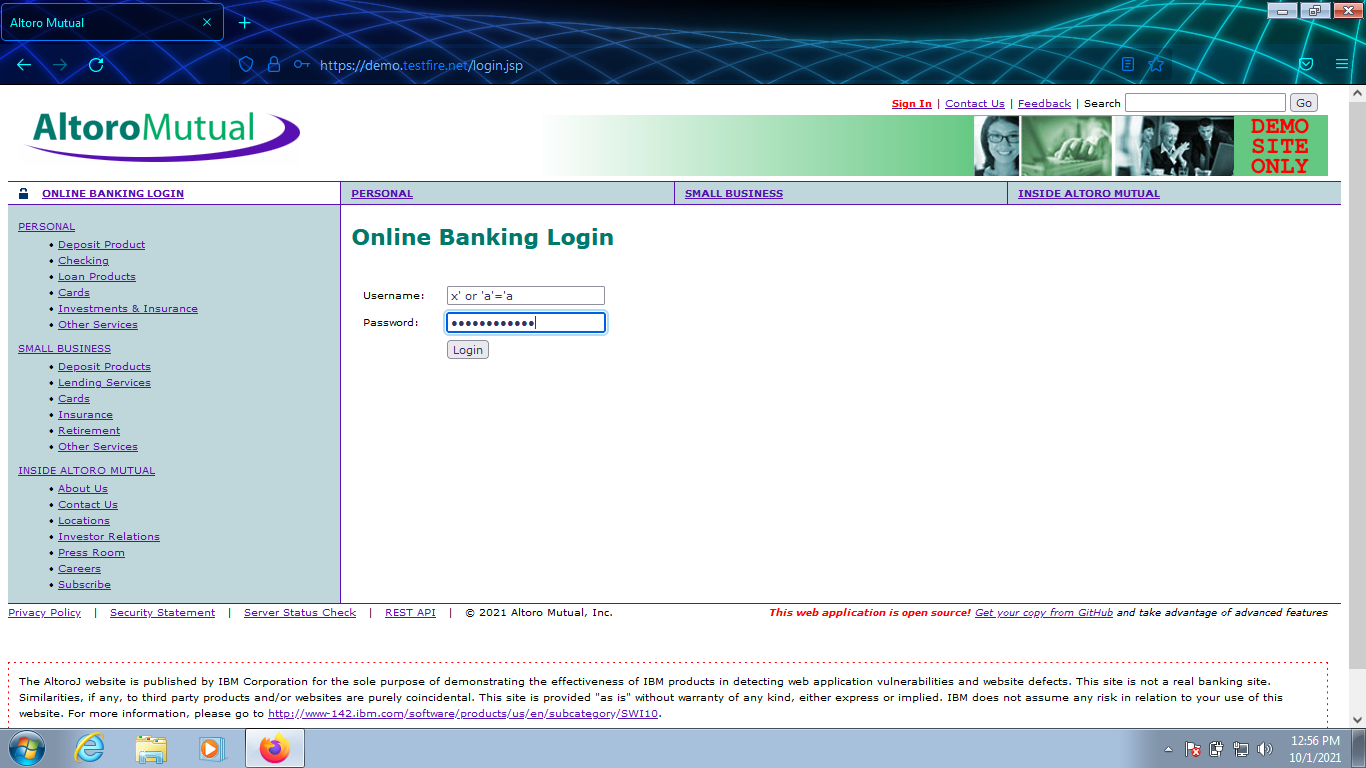


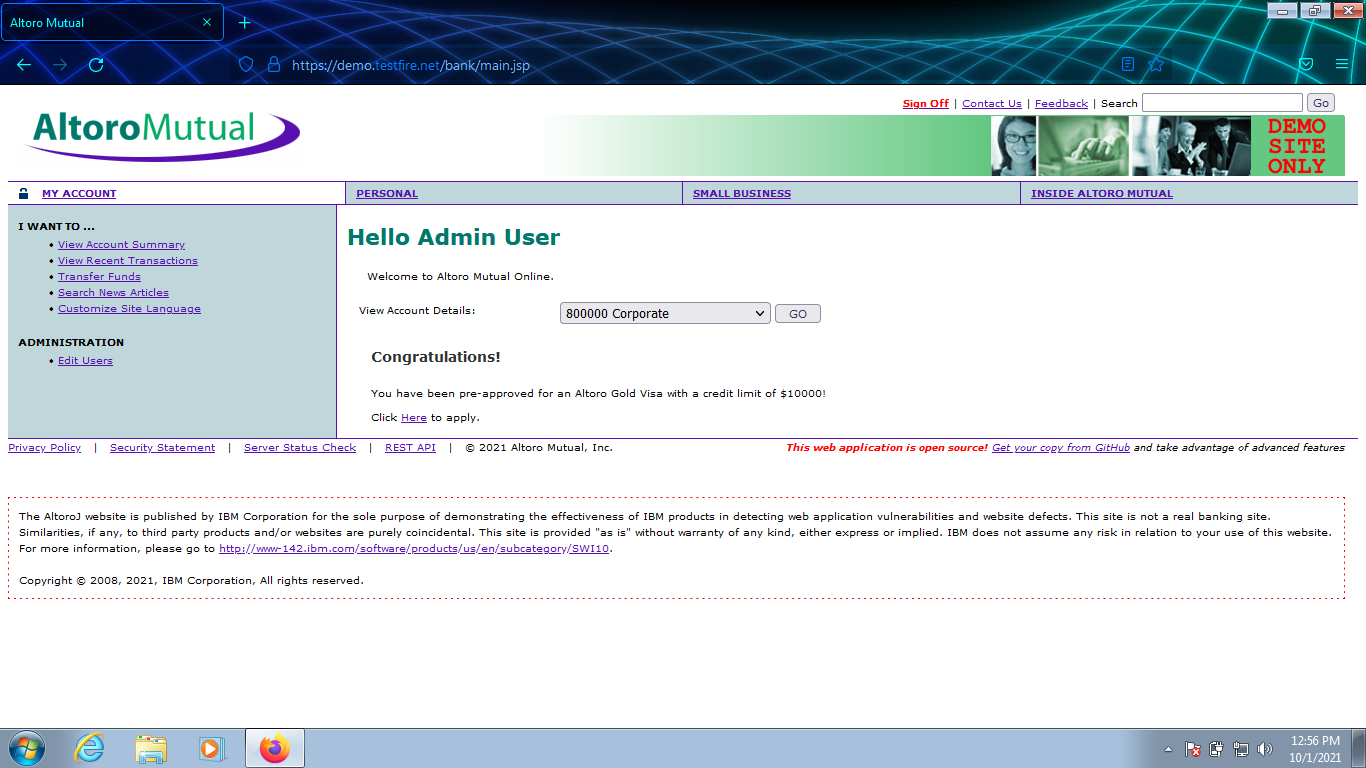
1. Taking advantage of the SQL query boolean structure, we can forge the credentials so that the whole statement becomes TRUE. For instance:  
     
   *SELECT account FROM USERS WHERE username = 'admin' AND password = 'x' or 'a'='a'*
2. Let's notice that *'x' or 'a'='a'* is always TRUE (x OR 1 = 1), so the whole statement again would be TRUE (1 AND 1 = 1)
3. Checking that a crafted password like x' or 'a'='a gives access to the database:





1. Even more, using x' or 'a'='a both for username and password (x' or 'a'='a / x' or 'a'='a) also does the trick of giving access to the database:





1. In this final case what we actually have is this:  
     
   *SELECT account FROM USERS WHERE username =  'x' or 'a'='a' AND password = 'x' or 'a'='a'*
2. The SQL query always falls to TRUE, because both AND operators are TRUE (1 AND 1 = 1). In other words:  
    *SELECT account FROM USERS WHERE TRUE AND TRUE*

Preventive steps to avoid Bypass Authentication:

1. Regulate session length: The web application must be able to end web sessions after a period of inactivity that depends on the type of requirements of the user. A secure banking portal, for example, must automatically log out the user after a few minutes to avoid  any  risks of hijacked session IDs
2. Improve session management: The web application must be able to issue a new Session ID after every successful authentication. These  IDs must be invalidated as soon as a session ends in order to prevent any misuse. Web URLs must be secure and must not include the Session ID in any form.

* Multi-factor Authentication (MFA): Among the  OWASP top 10 broken authentication, the first tips is to implement Multi-factor Authentication to prevent attacks. MFA requires an additional credential to verify the user’s identity. An example of MFA would be a One-Time Password (OTP) mailed or messaged to the user that allows for verification.
* Disallow weak passwords: Users must be required to set passwords of a specific length containing special characters, letters as well as numbers to prevent credential theft. Therefore, those passwords that do not meet the required complexity and length must be automatically rejected.
* Breached password protection: Employ a breached password protection mechanism that locks the accounts of users whose passwords have been compromised until they verify and change the password to a new one. This will ensure that if passwords are stolen, the organization is notified.
* Strict credential recovery process: The process to recover credentials must be strict, involving multiple verification checks to ensure that such recovery options are not misused by attackers.
* Secure password storage: Passwords must be encrypted, hashed, and salted as it helps slow down brute-force attacks or other attempts to infiltrate password databases.
* Employ brute-force protection: Applications should set a maximum limit for user-login attempts from a specific IP address, to prevent brute-force and credential stuffing attacks. Any user exceeding this limit must be disallowed from making any further attempts.

In addition to the above steps, it also becomes necessary to ensure that users are adequately trained and educated on the potential risks of broken authentication through phishing attacks or weak passwords. Organizations must employ strong Cybersecurity measures in line with the constantly evolving global standards and must ensure that they avoid broken authentication by all means possible.

Conclusion:

In addition to the above steps, it also becomes necessary to corroborate that users are adequately trained and educated about the potential risks of broken authentication through phishing attacks or weak passwords. Organizations must employ strong Cybersecurity measures in line with the continually evolving global standards. They and must ensure the prevention of broken authentication by all means possible. In today’s day and age, Cybersecurity is a chief concern. Protection and security against broken authentication attacks form a large part of this concern.