**SAD Lab Exp-7**

**Shashwat Tripathi**

**D20A 64**

**Aim:** To study and implement sql injection vulnerability allowing login page to bypass

**Theory:**

**What is sql injection?**

SQL injection is a type of cyber attack that targets databases through vulnerable web applications. The core idea is to manipulate SQL queries by injecting malicious input, which can lead to unauthorized access, data retrieval, or even data modification.

**How It Works:**

1. **Vulnerable Input Fields**: SQL injection typically exploits input fields in web forms, URLs, or HTTP headers. If the application fails to properly sanitize or validate user input, an attacker can insert SQL code.
2. **SQL Queries**: When a user submits data (e.g., a username or password), the application constructs an SQL query based on that input. If the input is not properly handled, an attacker can modify the SQL query's structure.
3. **Execution**: The database executes the altered query, which could allow the attacker to bypass authentication, extract sensitive data (like user credentials), or even modify or delete data.

### **How SQL Injection Attacks Are Performed:**

### **Identifying Vulnerabilities**: Attackers search for web applications that accept user input and communicate with a database. Common targets include login forms, search bars, and other input fields.

1. **Manipulating Input**: Attackers craft specific input that alters the intended SQL query. They exploit the application's failure to properly handle or sanitize this input.
2. **Executing Malicious Queries**: If the application is vulnerable, the database executes the manipulated query, allowing attackers to gain unauthorized access or retrieve sensitive data.
3. **Data Exfiltration**: Attackers extract information by leveraging the modified queries to access confidential data, or they may exploit error messages to gather insights about the database structure.
4. **Performing Unauthorized Actions**: Once they gain access, attackers may modify or delete data, perform administrative actions, or even escalate their privileges within the database.

### **Why SQL Injection Attacks Are Performed:**

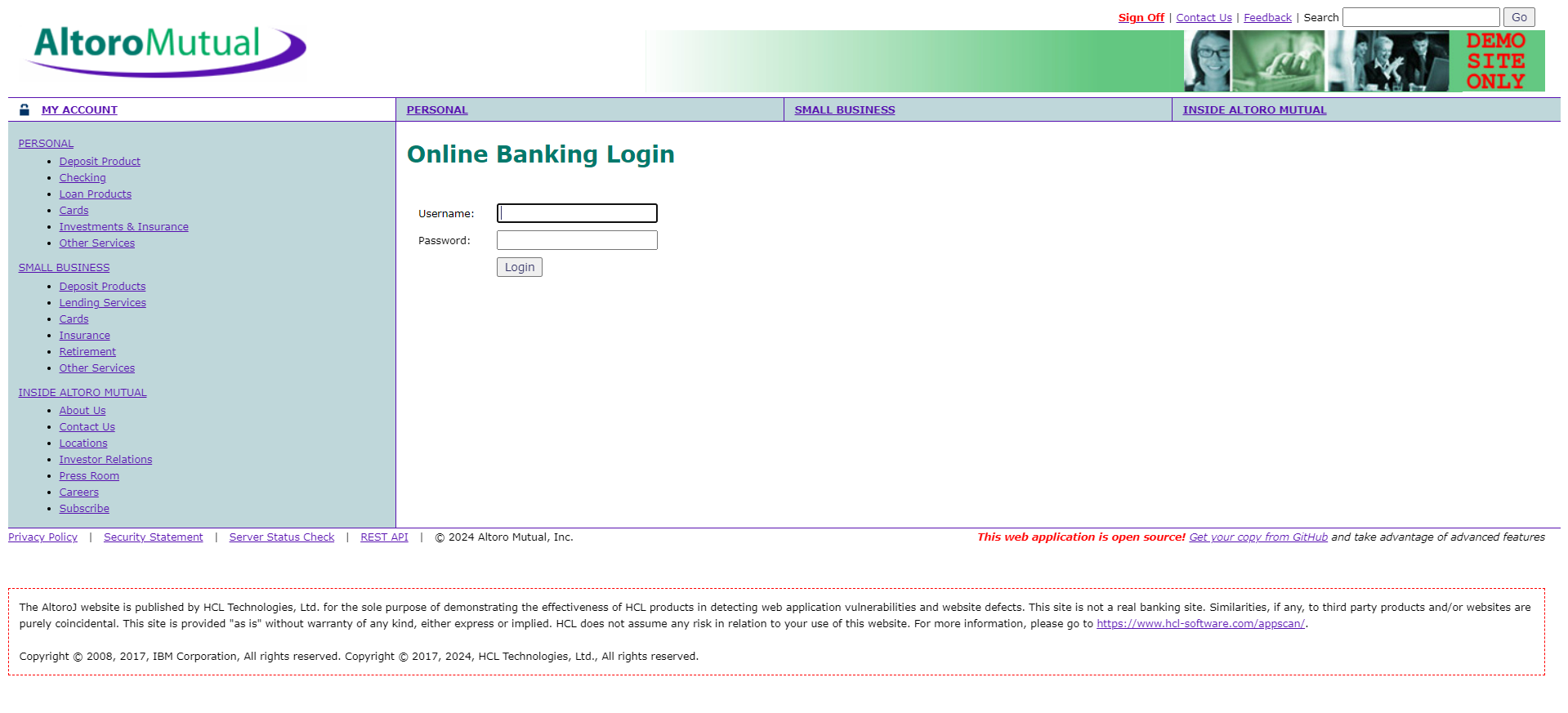
1. **Data Theft**: The primary motivation is often to steal sensitive information, such as personal data, financial records, or trade secrets.
2. **Financial Gain**: Stolen data can be sold on the dark web or used for fraudulent activities, providing a direct financial benefit to attackers.
3. **Disruption**: Some attackers aim to disrupt services by corrupting or deleting data, which can lead to operational downtime and financial losses for the organization.
4. **Reputation Damage**: Compromising an application can harm an organization’s reputation, leading to loss of customer trust and negative publicity.
5. **Political or Social Motives**: Hacktivists may target organizations for political reasons, seeking to expose information or cause embarrassment as a form of protest.
6. **Testing Security**: Ethical hackers may use SQL injection as part of security assessments, identifying vulnerabilities to help organizations improve their defenses (though this is done with permission).

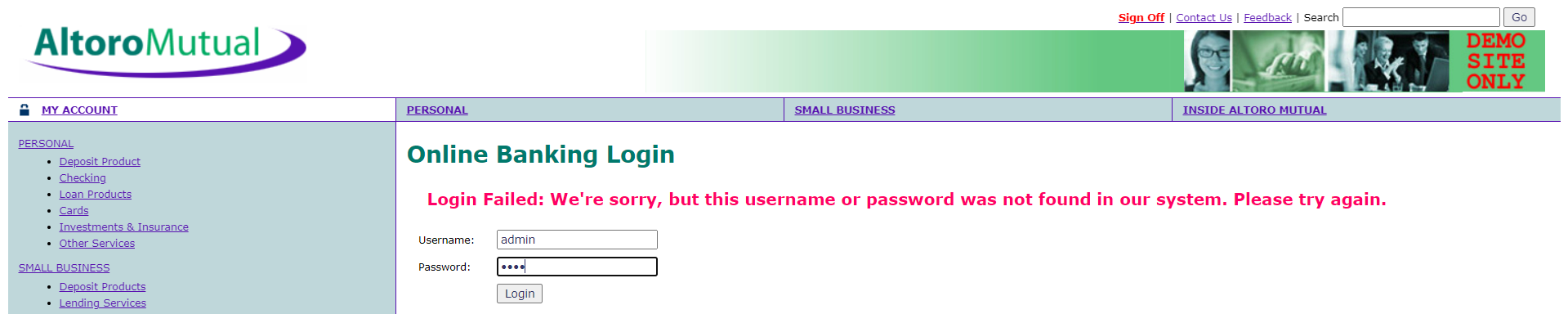
**How to prevent sql injection?**

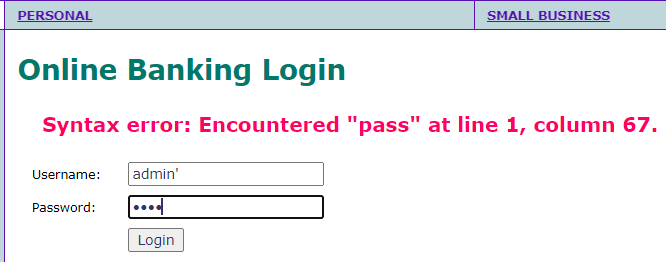
1. **Input Validation**: Always validate and sanitize user inputs to ensure they conform to expected formats.
2. **Parameterized Queries**: Use prepared statements or parameterized queries, which separate SQL code from data.
3. **Stored Procedures**: Use stored procedures, which can reduce risk if properly implemented.
4. **Web Application Firewalls (WAFs)**: Employ WAFs to filter and monitor HTTP requests for malicious input.
5. **Regular Security Testing**: Conduct penetration testing and security assessments to identify and fix vulnerabilities.

**Implementation of SQL injection:**

<https://demo.testfire.net/login.jsp> is a site that is left vulnerable to SQL injection attacks for educational purposes



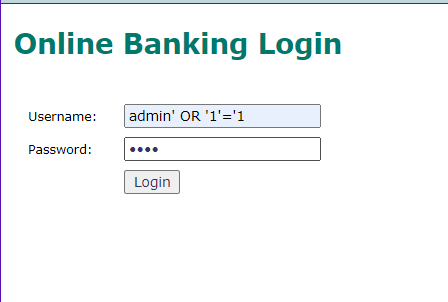


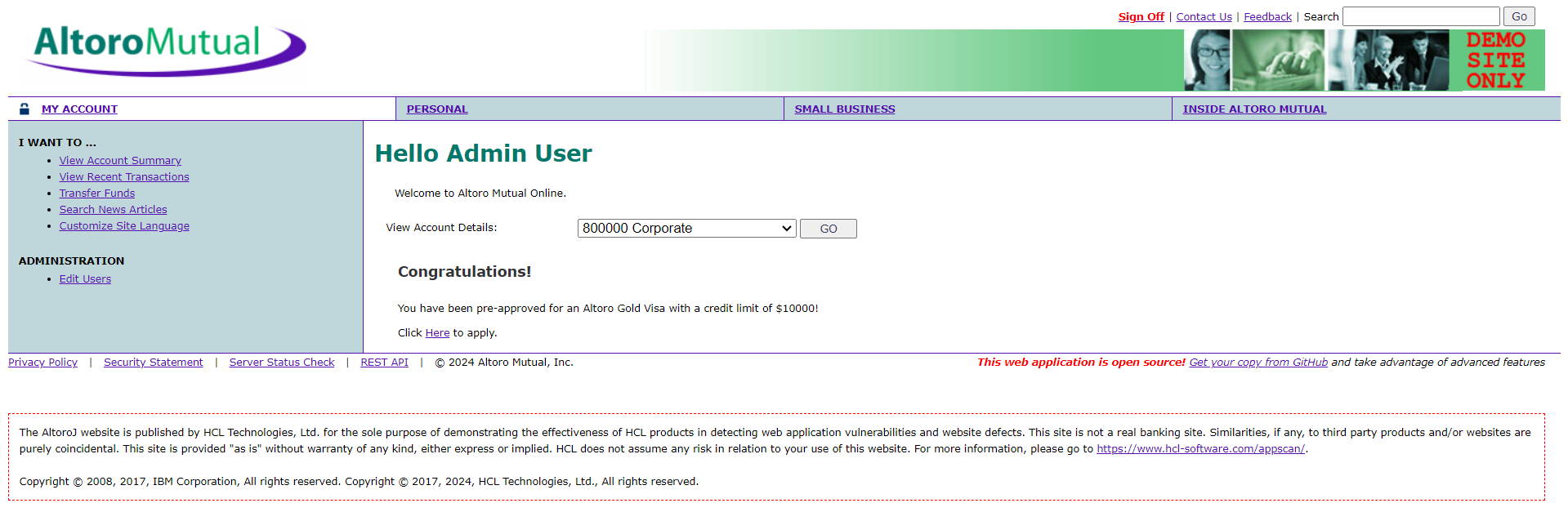


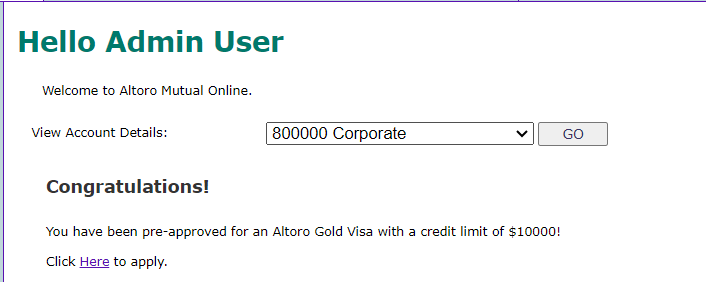
If we try to add a ‘ in the username field, we can see a syntax error being logged

This shows that the site is vulnerable to SQL injection attacks

Now we inject a malicious SQL code that always evaluates to true







Since the ‘1’ = ‘1’ condition is always true, we successfully bypass the authentication mechanism using SQL injection

### **Conclusion:** In this experiment, we successfully demonstrated how SQL injection vulnerabilities can be exploited to bypass authentication on a login page, highlighting the critical risks associated with inadequate input validation. These findings reinforce the necessity for implementing robust security practices, such as parameterized queries and thorough sanitization, to protect web applications from such attacks.