UTC CHALLENGE POC DOC

The web application with the given URL: <http://www.testfire.net/default.aspx> has the following vulnerabilities.

* Authentication Bypass
* SQL Injection
* Blind SQL
* XSS

The above stated vulnerabilities are exploited using some tools and scanners, along with some SQL statements which can modify and retrieve the webpages. Each of the vulnerabilities and how they are exploited is stated below along with the screenshots of the images, and scripts.

1. Authentication Bypass: The given problem statement has two usernames and passwords given namely

* Username: jsmith, password: demo1234
* Username: admin, password: admin – tried directly from common passwords.

I have tried simple SQL Injection attacks, through which the authentication can be broken. The SQL injection can also be covered in this, although there are many.

In the login page, for the authentication details, the following strings are introduced and gained access to the website main page.

USERNAME FIELD: **$username = 1' or '1' = '1**

PASSWORD FIELD: **$password = 1' or '1' = '1**

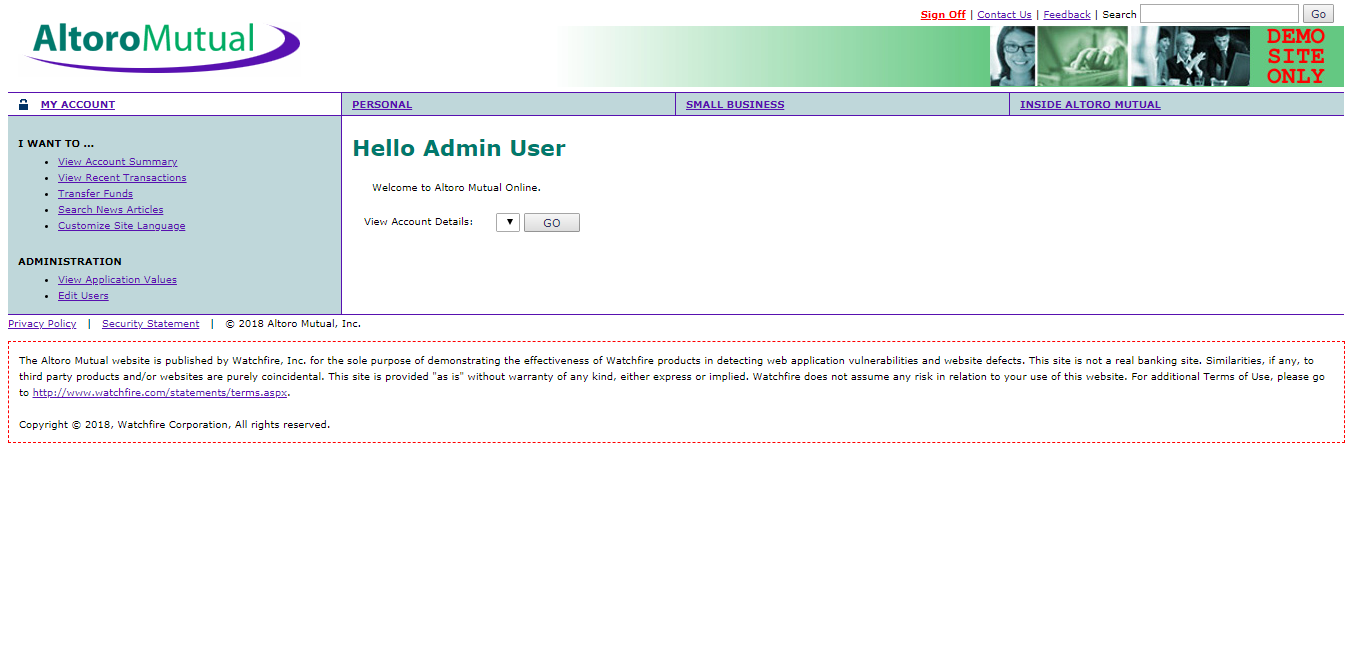
The above strings will break the SQL query to the following:

**SELECT \* FROM Users WHERE Username='1' OR '1' = '1' AND Password='1' OR '1' = '1'**

Which is always correct, (since 1=1 obviously) and hence we got access. Screenshot attached. I got the admin access of the webpage. Maybe admin is first record in the database.

**Possible root cause:** this is because of theunprepared sql statement to verify and check for the user records. The sql query on the backend need to perform certain validations and condition checks so as to overcome the SQL Injection attacks to bypass authentication.

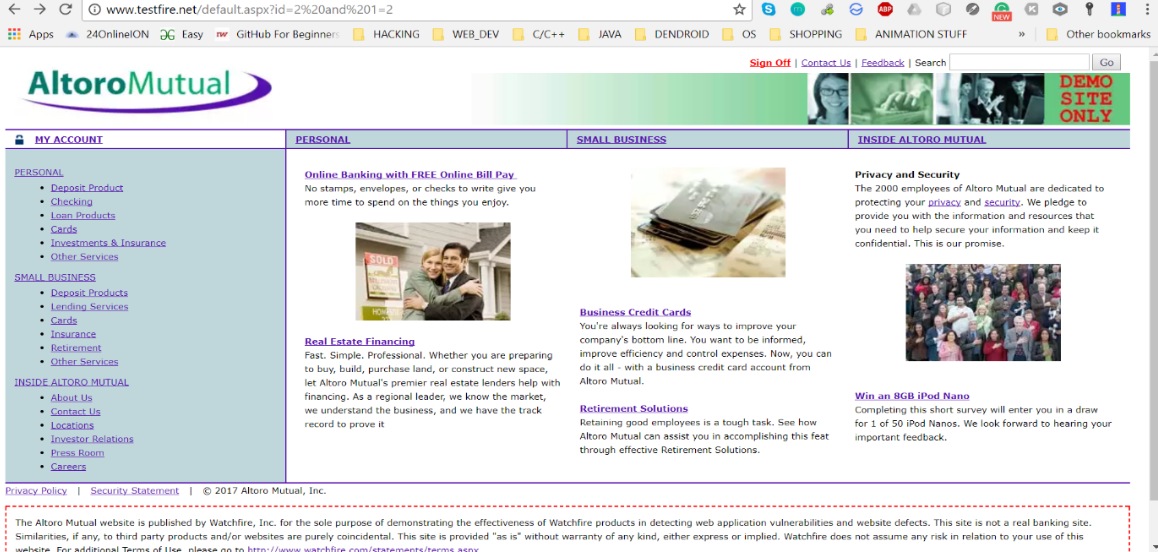
**Recommendation:** Use prepared statements with validations to handle all kinds of string manipulation attacks, also use certain coding standards, while authenticating. Put additional authentication verifications (email /otp). Never store passwords in plain text. Using hash code is a good way of encrypting passwords. Also perform validations on both server and client side.



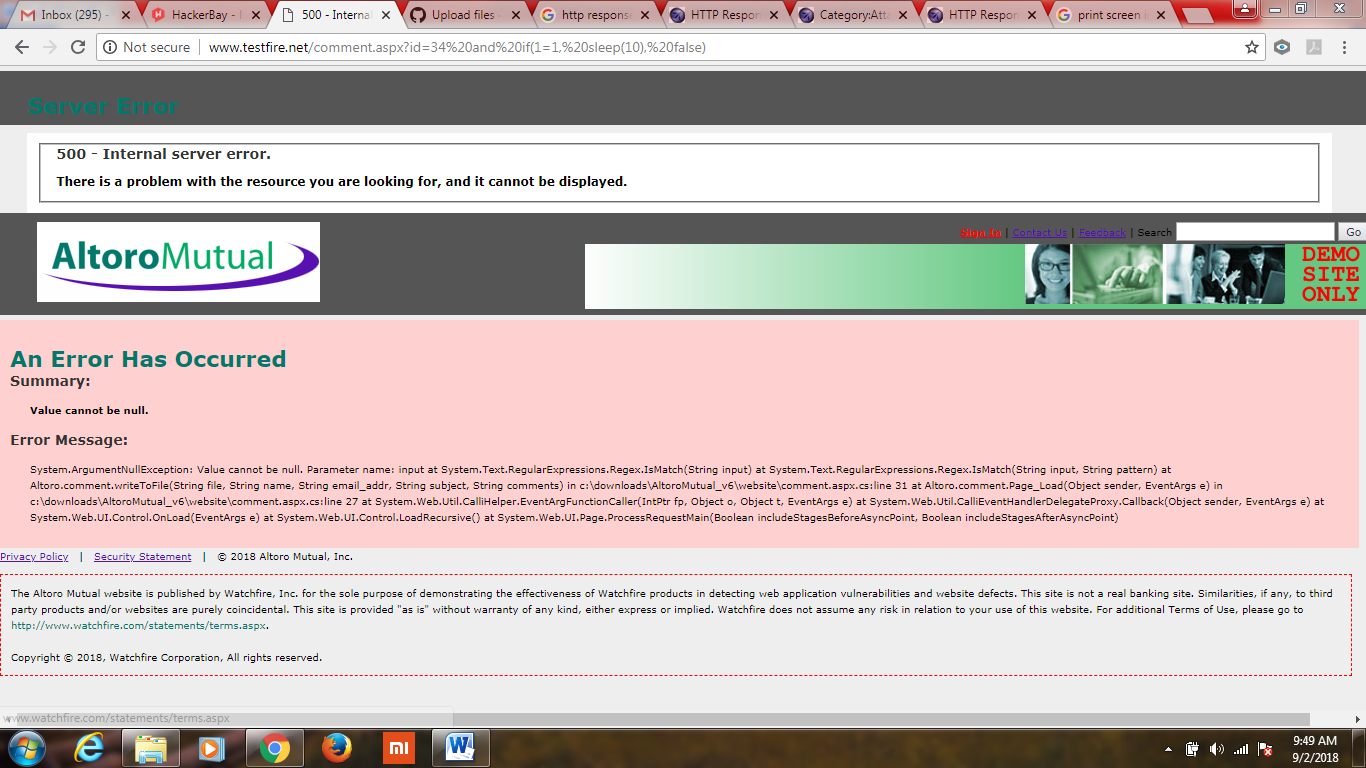
1. Blind SQL: This is modifying the contents of the URL to retrieve the necessary data accordingly, which means there are no proper input checks.

**http://www.testfire.net/comment.aspx?id=2%20and%201=1**

The above statement is working fine and displaying the contents of the page



<http://www.testfire.net/comment.aspx?id=34%20and%20if(1=1,%20sleep(10),%20false)> **this makes the webpage sleep for some time, also breaking the background code.**

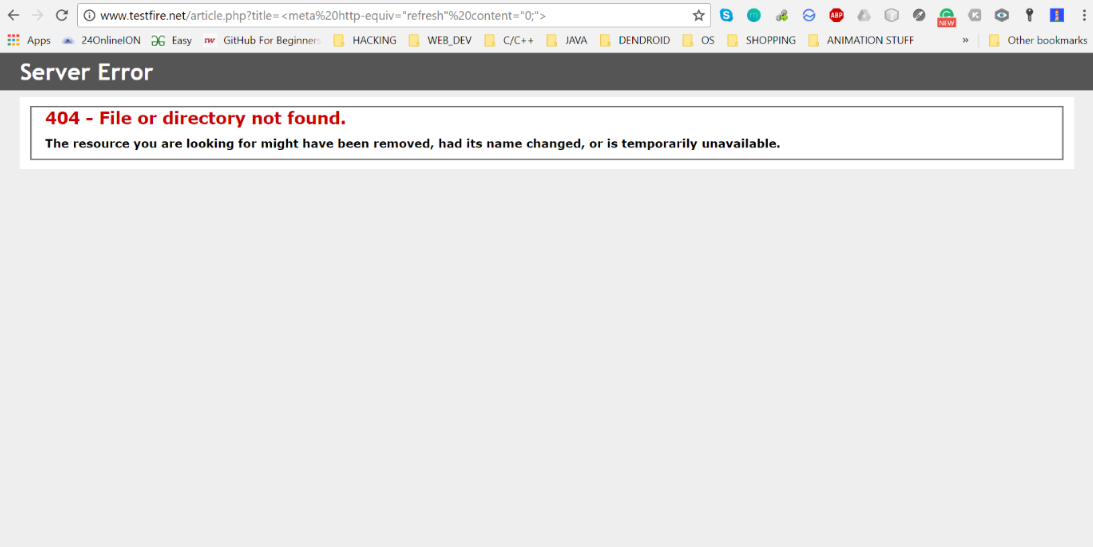
****

1. XSS Vulnerability:

After scanning using a few XSS Scanners Namely Burp suite and XSS online scanner, the web application is proved to have XSS vulnerabilities.

The Vulnerabilities are mainly detected in input field in the feedback form, which is sending the sensitive parameters through the query string, that can be modified to run any kind of script, performing a **DOS attack** etc. to name a few.

DOS attack which makes the content unavailable.



The source code in the feedback form has some serious vulnerabilities which can be used to send unlimited requests to the web server making it crash, again summing up to DOS. Refer the screenshot in the next page, observe the value of name field is set to empty space and all the other input controls aren’t even validating for necessary checks, whatever input is given it blindly accepts it.

Some more screenshots are attached showing the various XSS vulnerabilities in the next page, these are found through multiple scanners.

