E-Commerce Web Application

Security Assessment

.

[Rajat Vij]

[12/11/2015]

Table of Contents

[Executive Summary 1](#_Toc437888652)

[Scope 3](#_Toc437888653)

[Methodology 3](#_Toc437888654)

[Findings 4](#_Toc437888655)

[Cross Site Scripting 4](#_Toc437888656)

[Cross Site Request Forgery 8](#_Toc437888657)

[Session Vulnerabilities 11](#_Toc437888658)

[SQL Injection 15](#_Toc437888659)

[Logic Flaws 17](#_Toc437888660)

[Other’s 21](#_Toc437888661)

[Recommendations 27](#_Toc437888662)

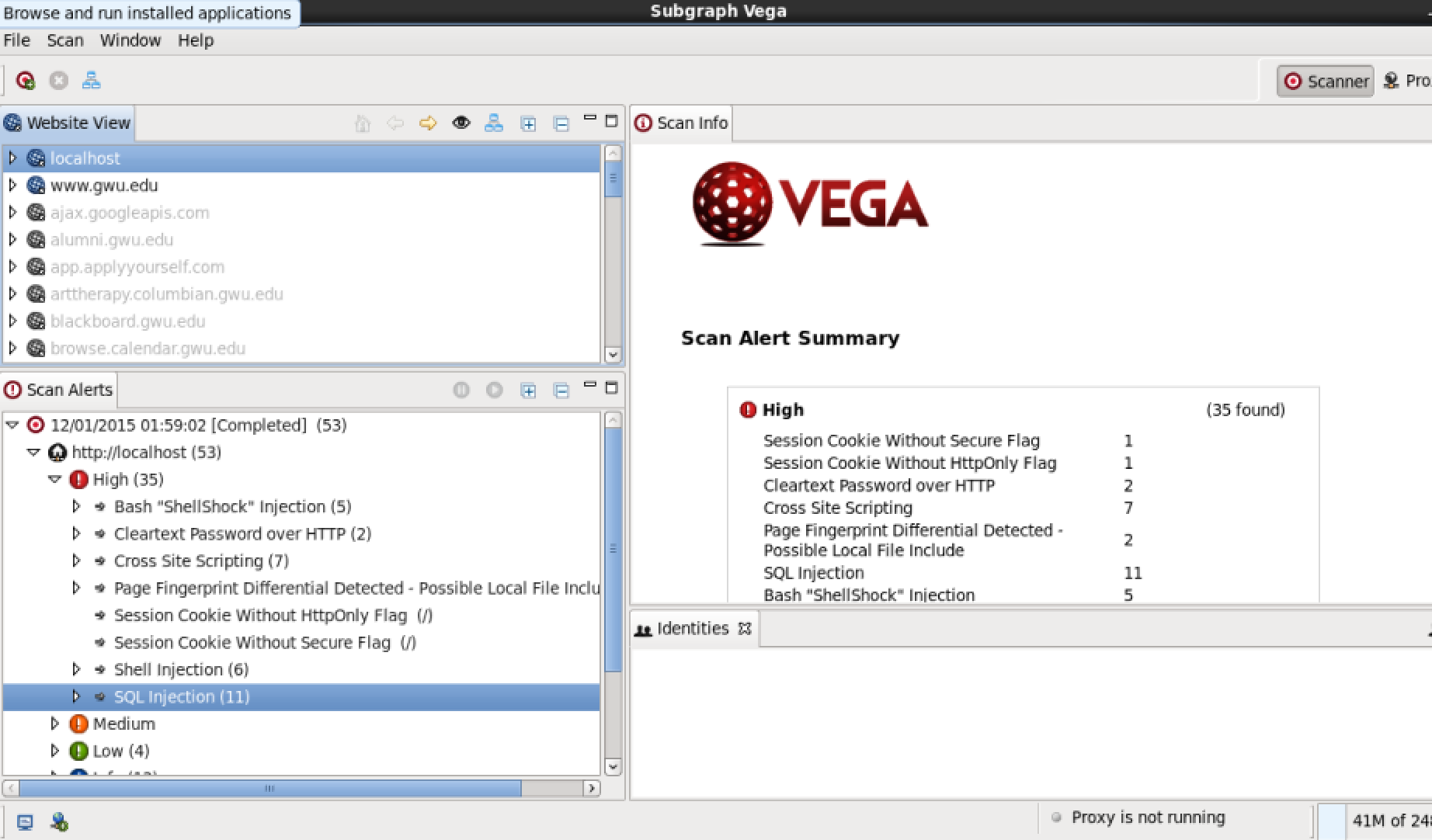
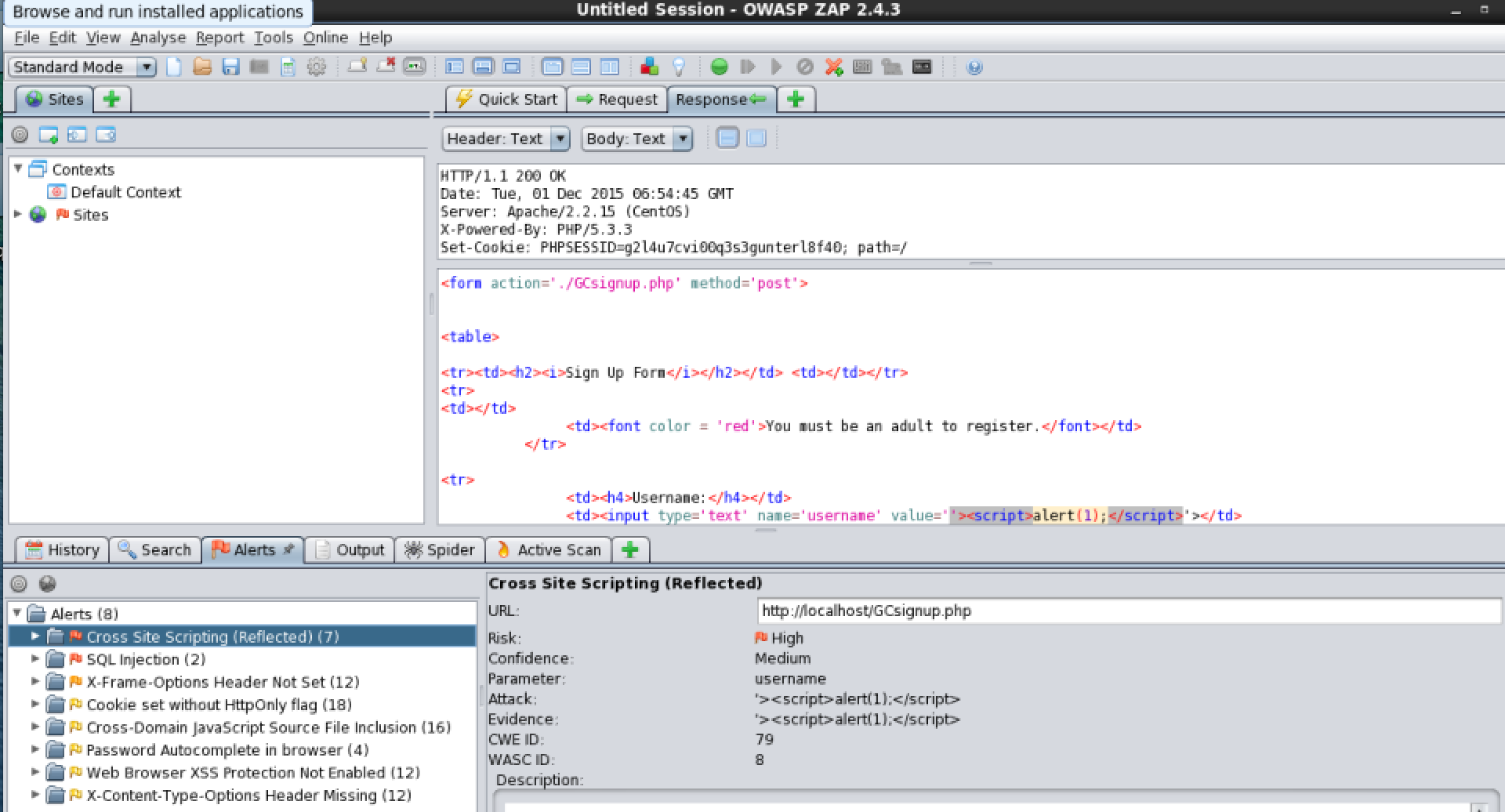
[Challenges 27](#_Toc437888663)

# Executive Summary

GameChest Inc. is a website that concentrates on e commerce of video games. The application has client side and admin side where users can register as client and purchase video games.

I tested the site by identifying common vulnerabilities for web application. I referred owasp.org for vulnerabilities and used following vulnerabilities:

1. SQL Injection
2. Broken Authentication
3. Session Management
4. Cross Site Scripting (XSS)
5. Application Logic Flaw
6. Cross Site Request Forgery
7. Invalidated Redirects and Forwards
8. Brute Force
9. Encryption
10. MITM

After identifying vulnerabilities, I used scanner (ZAP and Vega) to help me identify on which pages I can find a particular vulnerability. And then I did manual testing by using Burp Suite to attack the application.

**Findings**:

**Logic Flaws**

I found few logic flaws in the web application such as bypassing username check on login screen and we are able to enter anything an all the fields on checkout page except Credit Card, where even though the application checks for validity of credit card, it still lets us place an order with an invalid credit card or save the credit card on Profile page.

**XSS**

I found stored XSS on Comments field in Item Details page and Add and Edit Item pages in Manage Inventory.

I found reflected XSS on Signup page.

**CSRF**

I am able to forge a request to place an order with whatever items are that are present in shopping cart and if the customer information is saved in profile for a given account with its session information. So I am able to forward the request from any page that has XSS vulnerability to place an order.

**SQL Injection**

I am able to bypass username check on login page and am able to get every user on search user page.

**Session Vulnerability**

I am able to get session value stored in document.cookie from any page with XSS vulnerability.

**Others**

I am able to brute force login process as I just need to check for a valid password due to logic flaw present in login page. And since they are converting the password to md5 hash which in itself is not secure at server side instead of client side, I can see the password sent in clear text upon form submission.

**Summary**:

I found that the web application is moderately secure as even with my limited knowledge as pen tester I was able to find common vulnerabilities for web application and was able to get some result out of it. But even after knowing that the application is vulnerable at a particular I couldn’t exactly do what I aimed to. E.g., Dropping database or table’s or extracting table’s or table schema information. Getting username and password information or breaking the signup and login authentication process.

# Scope

This is a simple e commerce application with following functionalities:

1. Basic registration and login/logout authentication.
2. Inventory management
3. Inventory Browsing, item details and Customer Reviews
4. View and save user profile
5. Shopping cart
6. Checkout and save checkout info
7. Order Processing
8. Sales Information
9. Hot list for credit cards
10. Purchase Alert and events and High Purchase amount alert
11. Luhn Mod Check for credit cards.

From an attacker’s perspective I focused on following functionalities:

1. Basic Registration and login/logout authentication: I focused on this functionality with an aim to either sniffing information during registration and login or gain access to user credentials of already added customers.
2. Customer Reviews: Since there is a form to submit customer reviews we can use XSS vulnerability here to manipulate user comments.
3. View and Save user profile: To sniff profile information or change it according to what I want.
4. Saved Checkout Info: To sniff saved checkout info.
5. Luhn Mod Check for credit cards: To bypass the credit card validity check.

# Methodology

I used Web Application Vulnerabilities Scanners like ZAP and Vega to identify on which screen a vulnerability exists.

I used SQLMAP for automated SQL Injection attacks.

I used Burp Suite to initiate and automate my attacks and brute force their login functionality

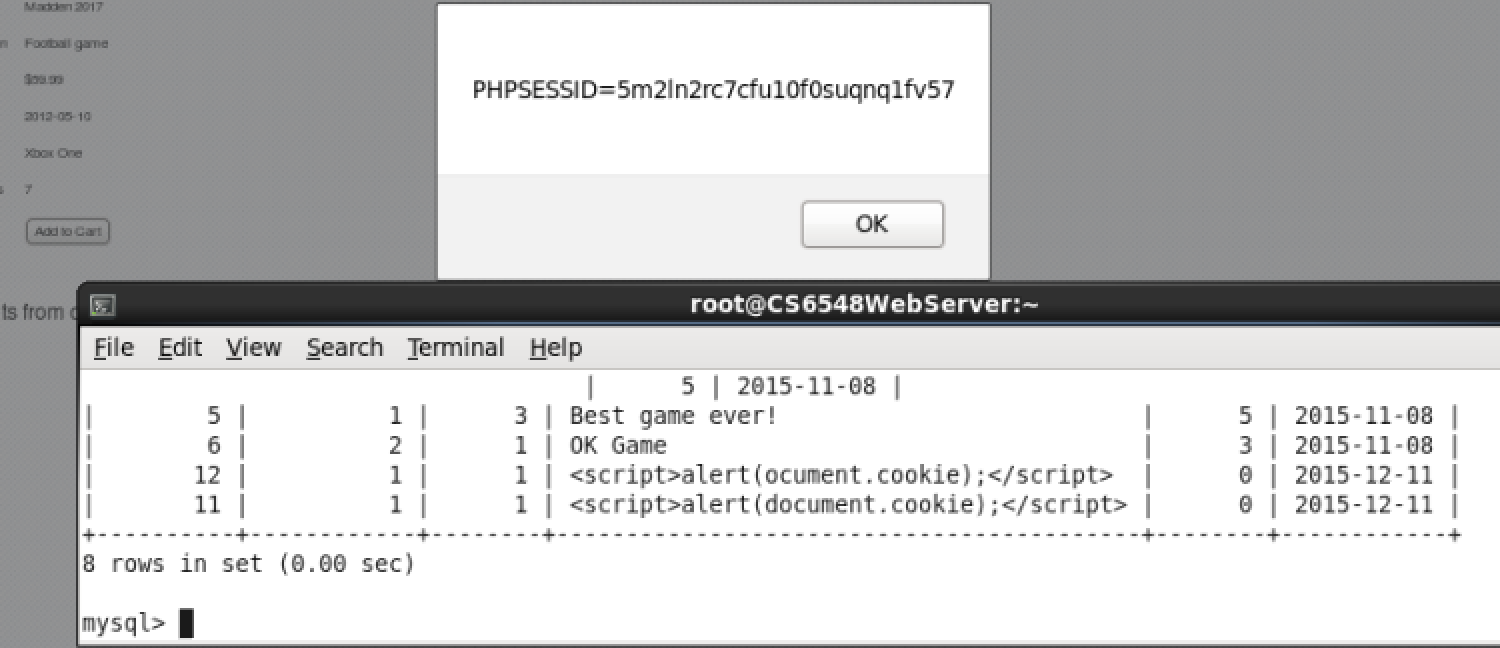
I shortlisted the vulnerabilities to study for web application, then used scanner tools to know where those vulnerabilities are. They I tried burp suit and sqlmap to automate my pentest. After that I did a code review and did manual test according the the flaws I found in their code.

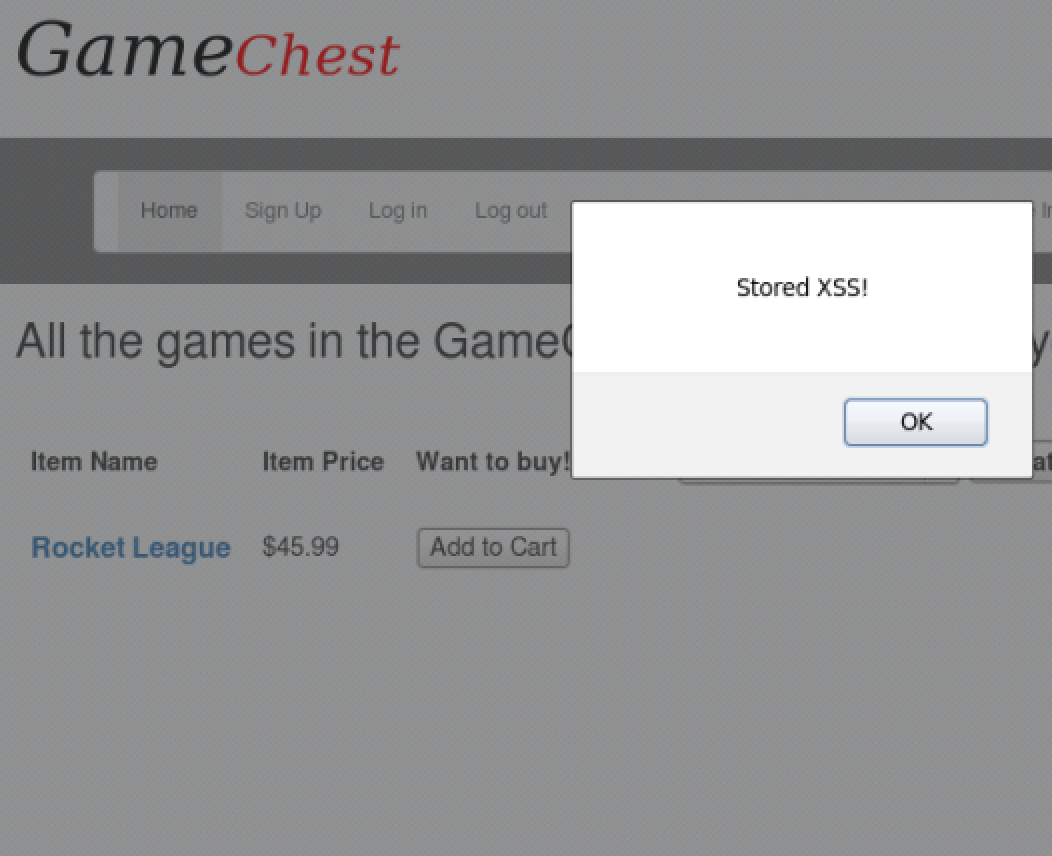
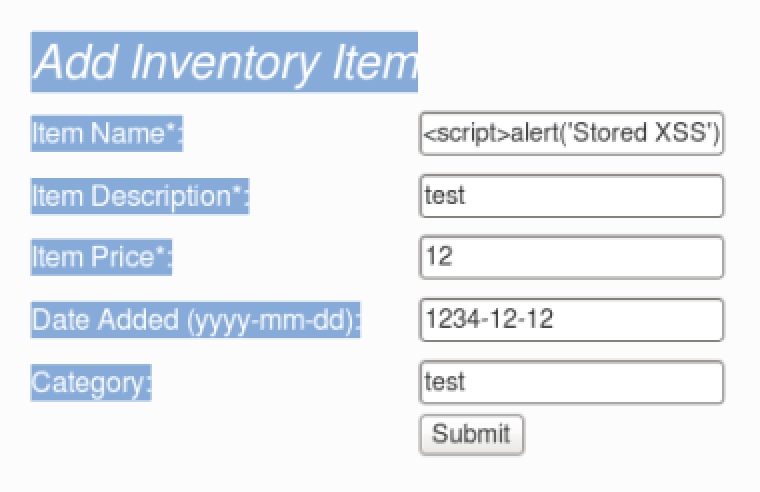
# Findings

## Cross Site Scripting

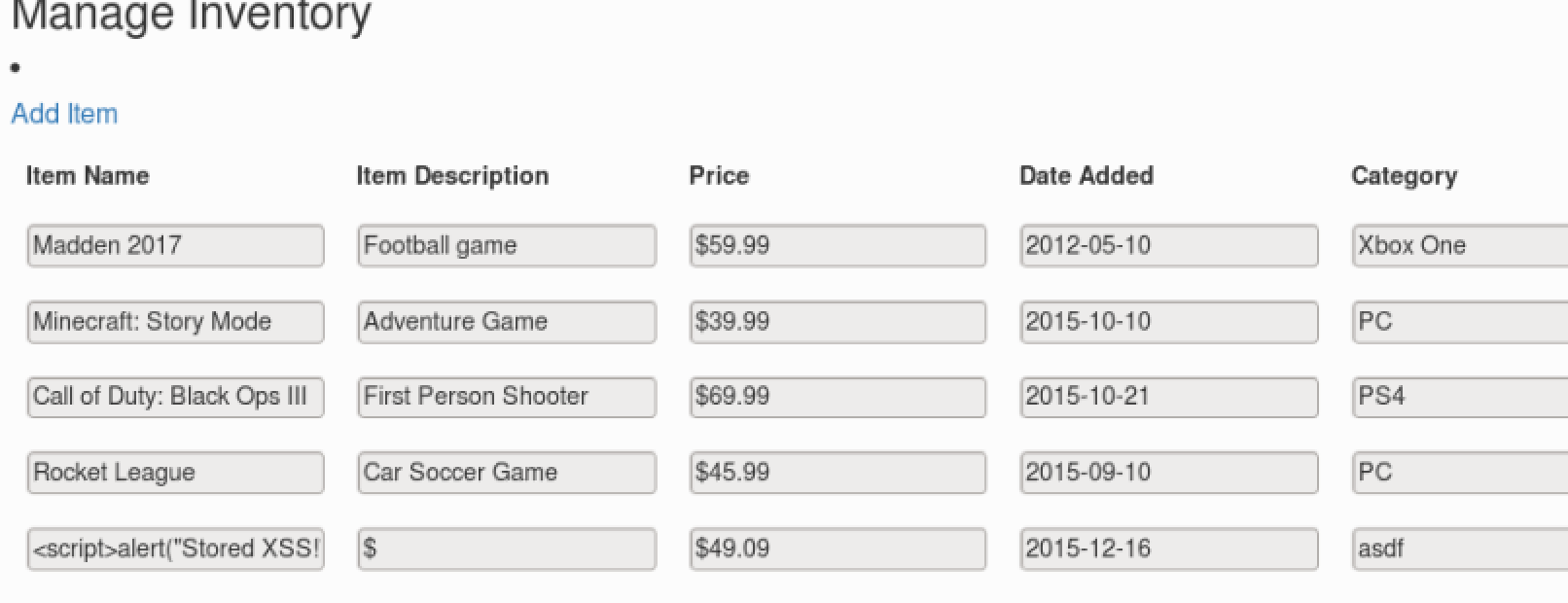
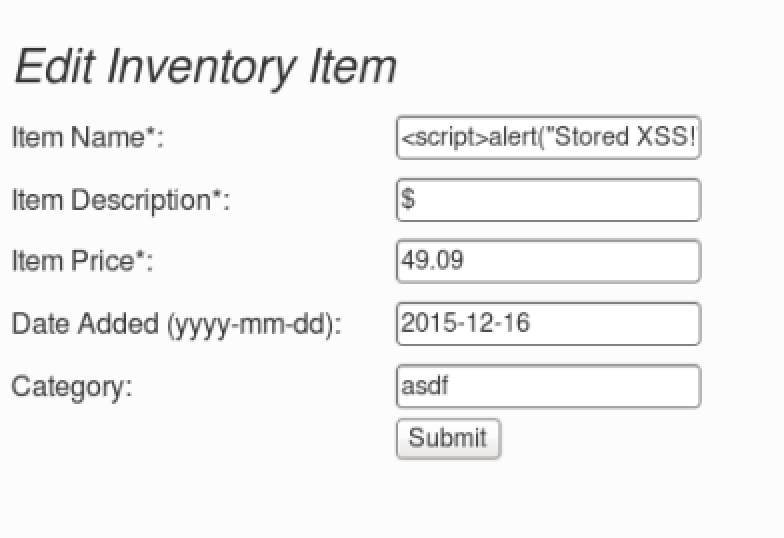
**Stored XSS**

Item Details page – Comments Field: If we put <script>alert(document.cookie);</script> in comments and save the comment. It will save the script in database and it will run it every time we access the item detail page.



Stored XSS in Add Inventory Item and Edit Inventory Item similar to Comments field in Item Detail page.: 

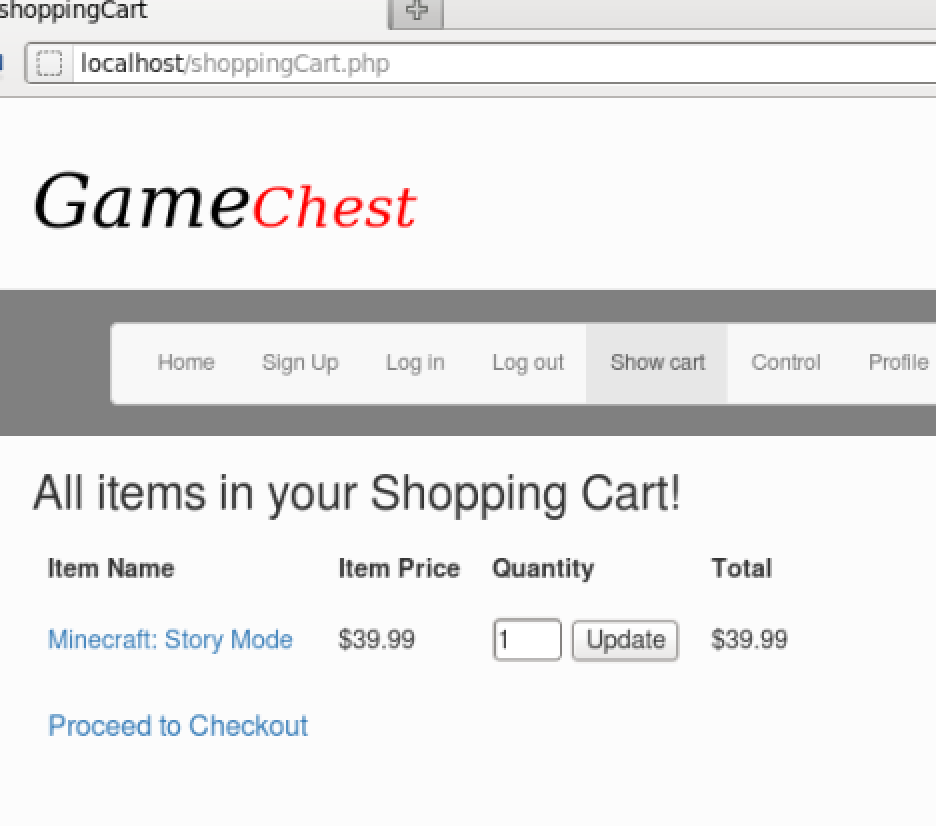
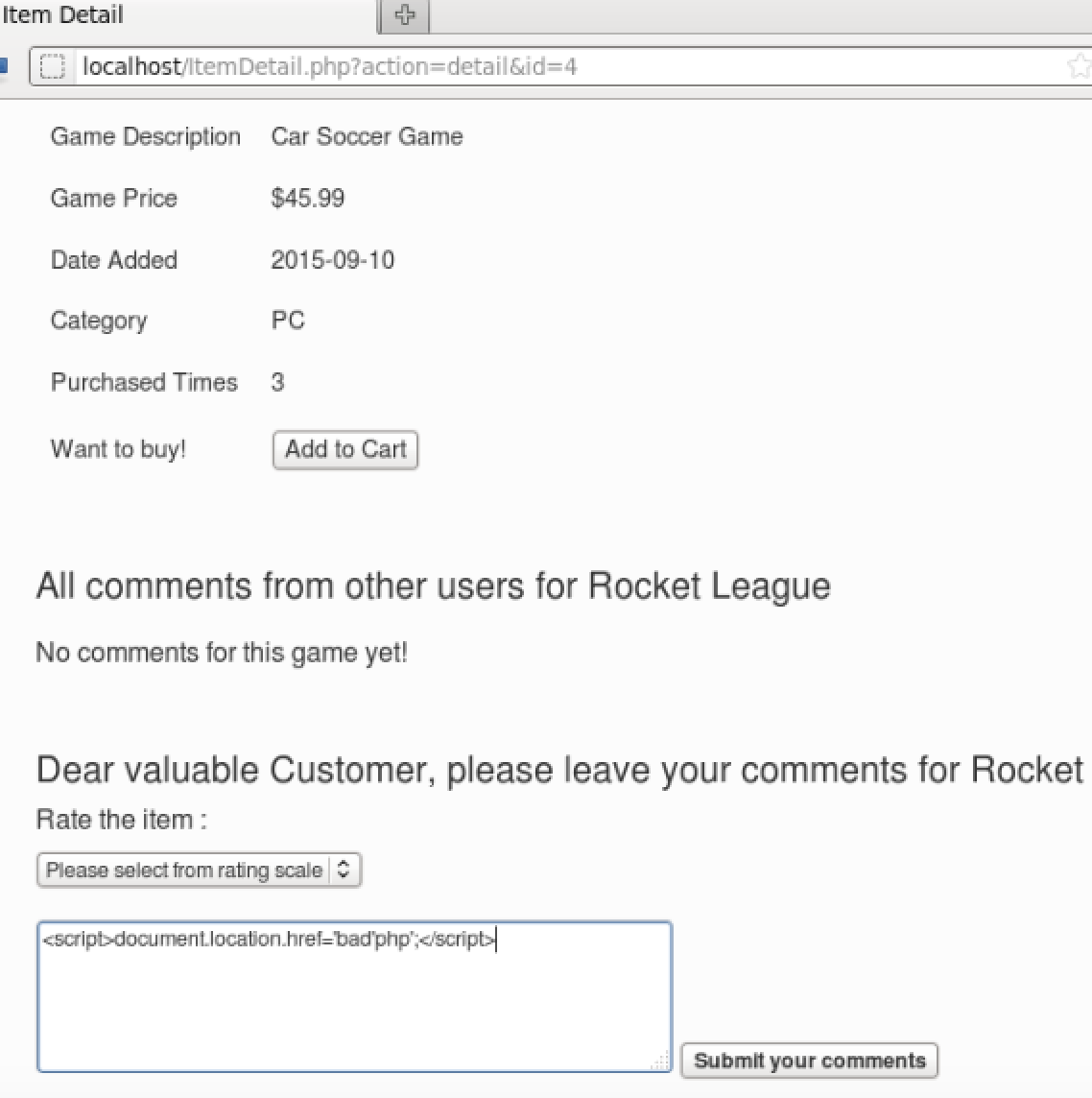


Stored XSS on Edit Inventory Page similar to add inventory page.

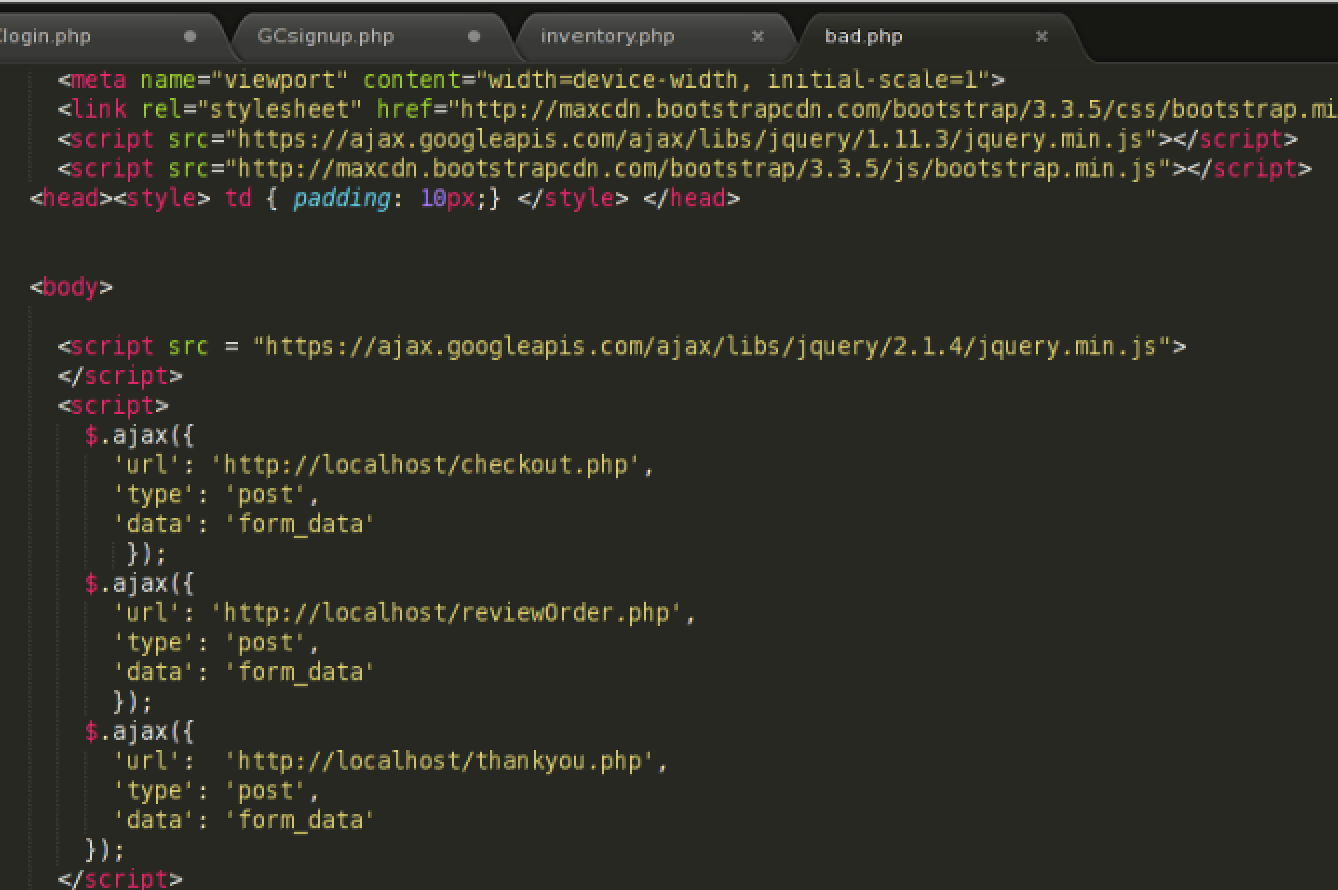
We have a reflected XSS which I have shown in Session Vulnerability.

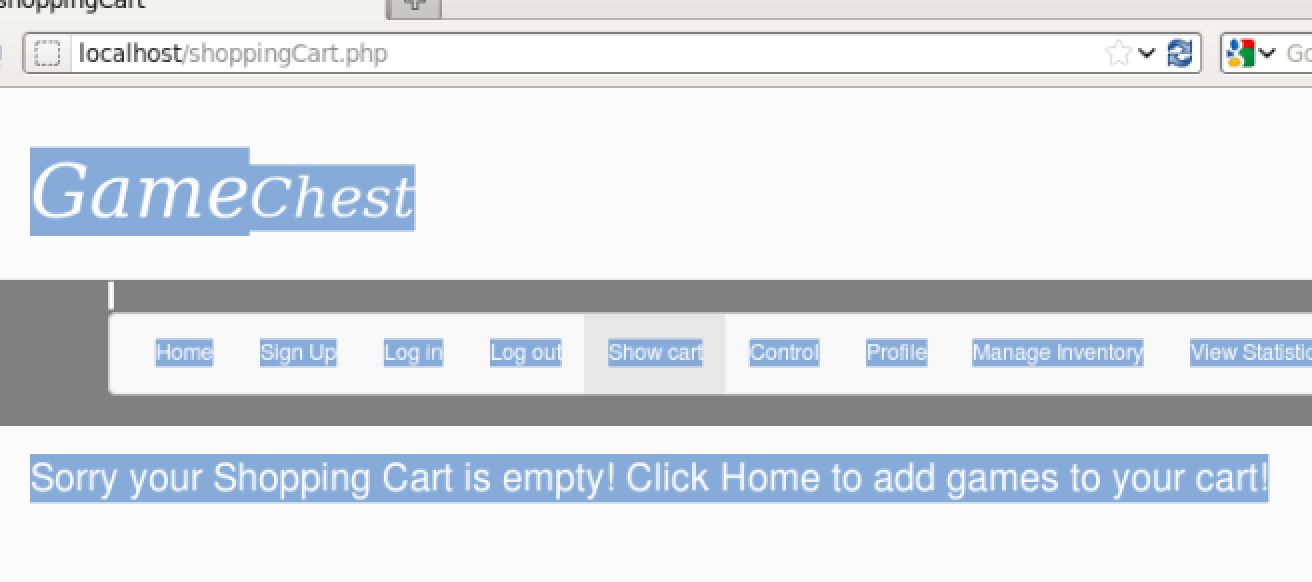
## Cross Site Request Forgery

For this we need a user session id with some item’s added in the shopping cart. We can use an XSS vulnerability to direct user to bad.php which we created which in the background sends forged request to the server to order whatever is in the shopping cart with the saved customer profile details.

Items present in shopping cart.Forwarding the page to bad.php

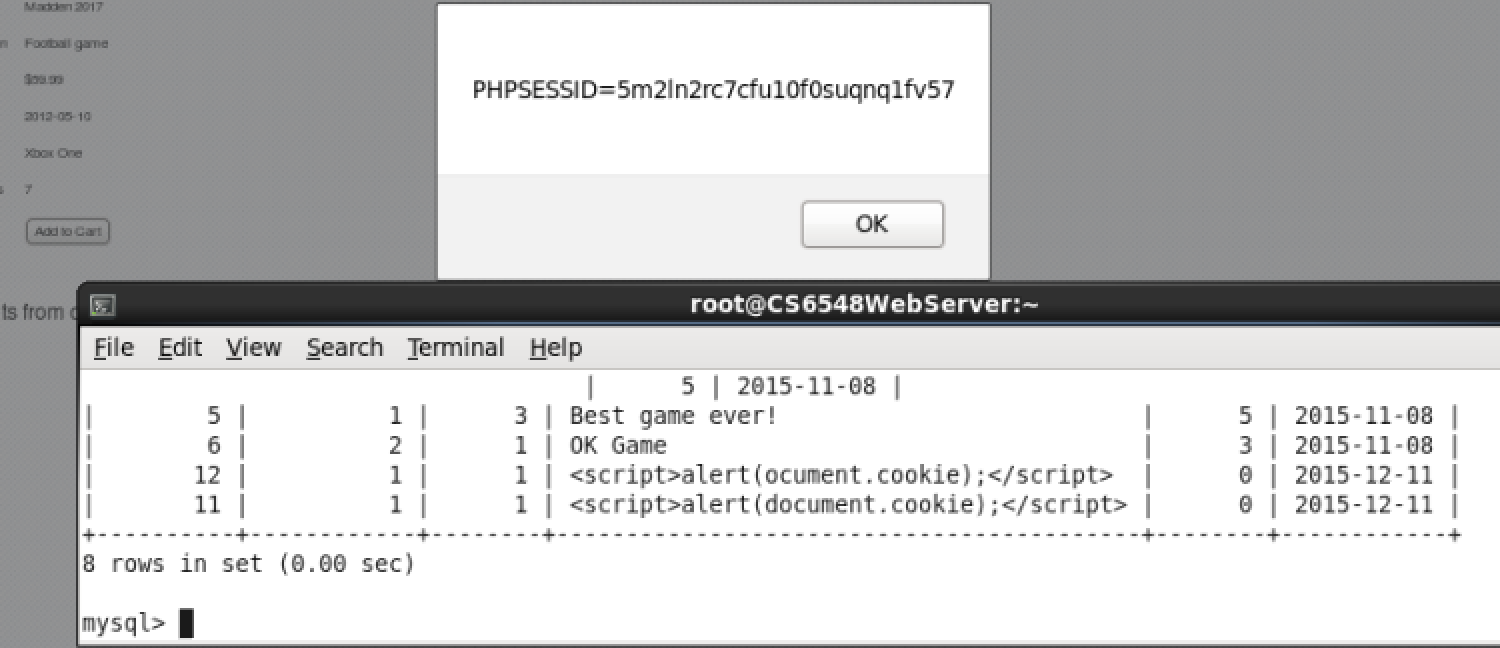
Bad.php is made up by the home page’s page source and I included Ajax to post requests via JQuery. At this moment I am only able to forward one request with my limited knowledge of JQuery but theoretically forwarding multiple request is also possible. So I edited my bad.php 3 times to place an order and refreshed bad.php 3 times to place the order.

[NOTE: This code won’t work at once as I have not managed to used form\_data, If we use similar code without any form\_data, which takes already stored value and submit the forms 3 times, we are able to do this attack]. 

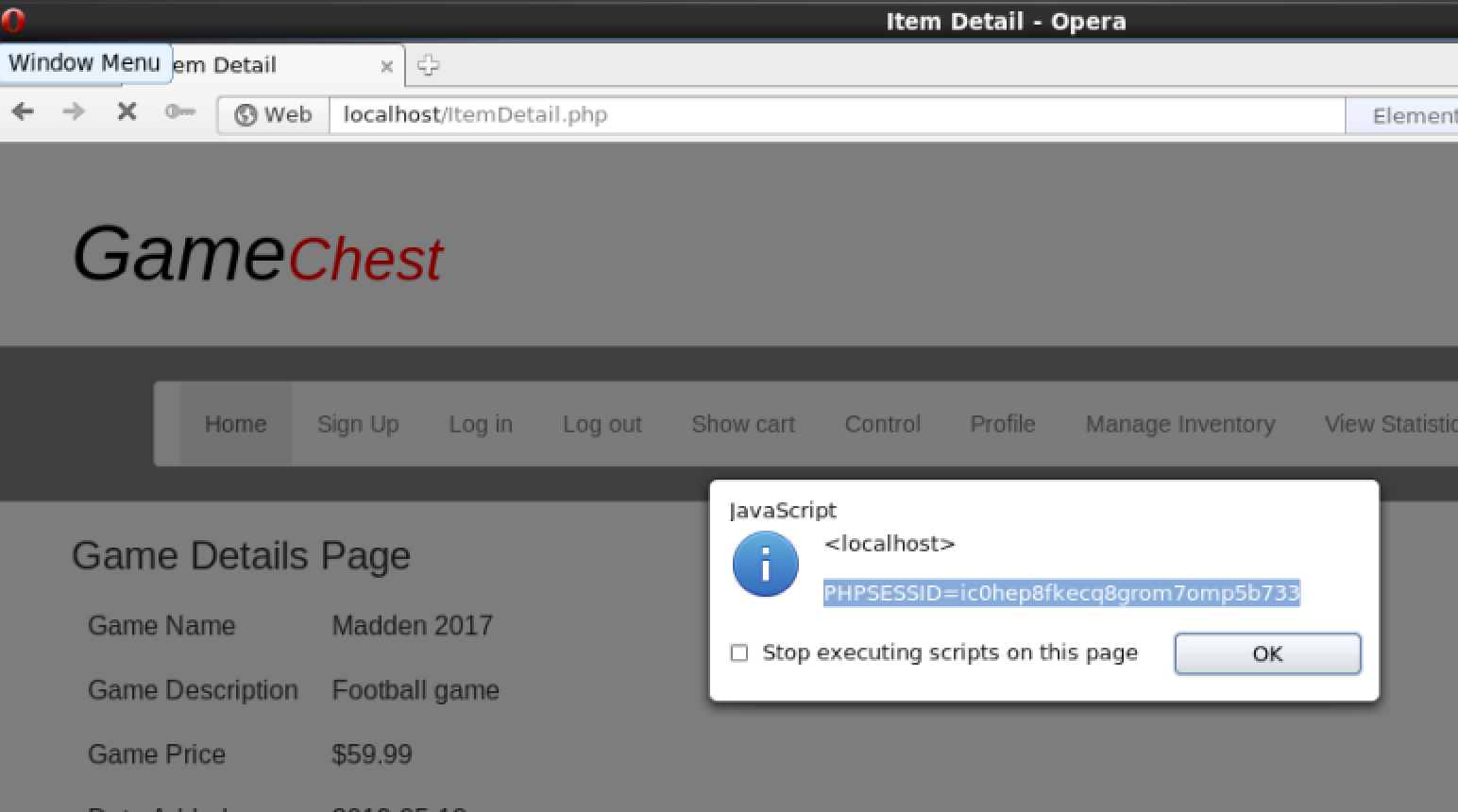
Order is placed as shopping cart is empty.

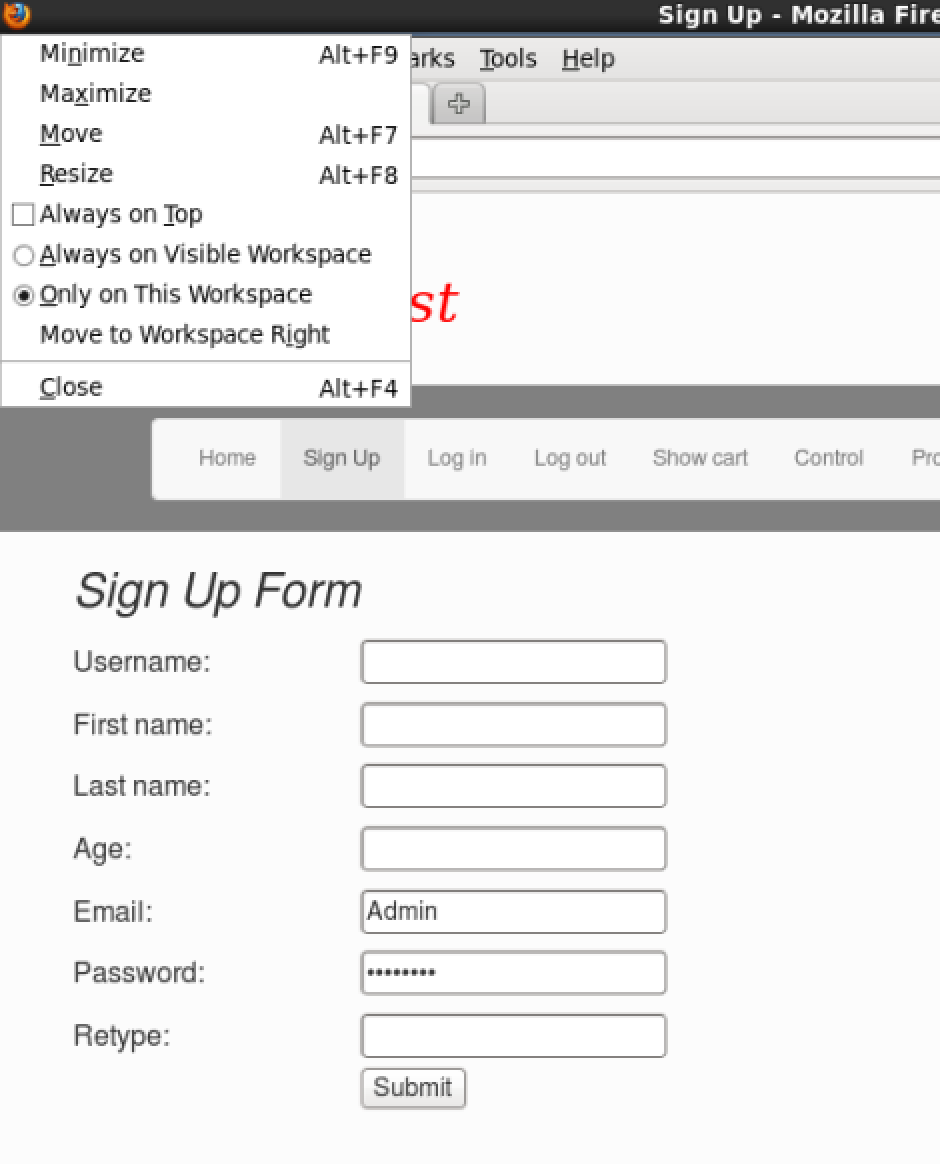
## Session Vulnerabilities

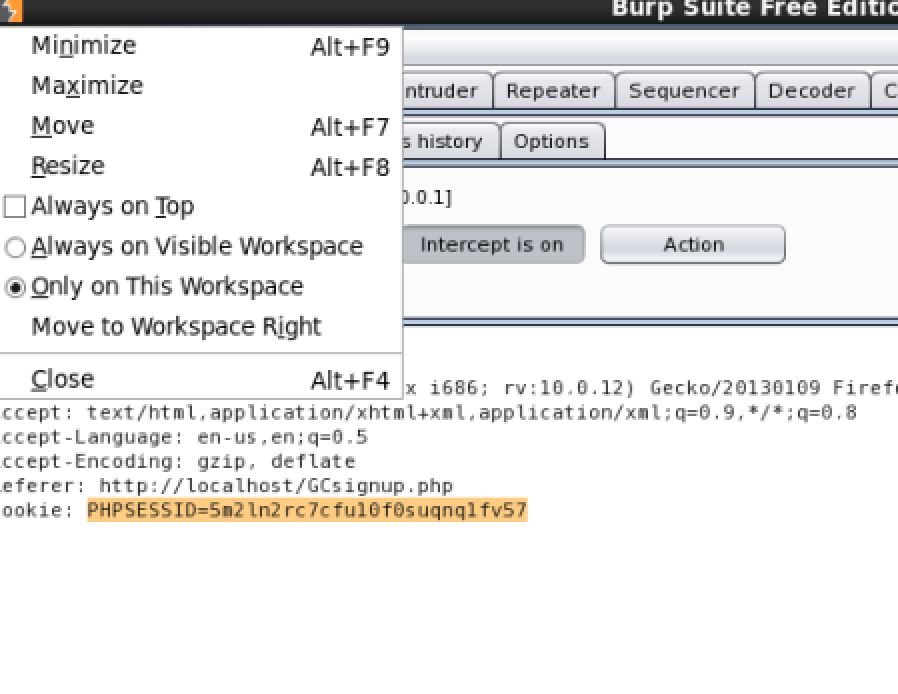
We can get php session value using document.cookie from and XSS vulnerable screen.

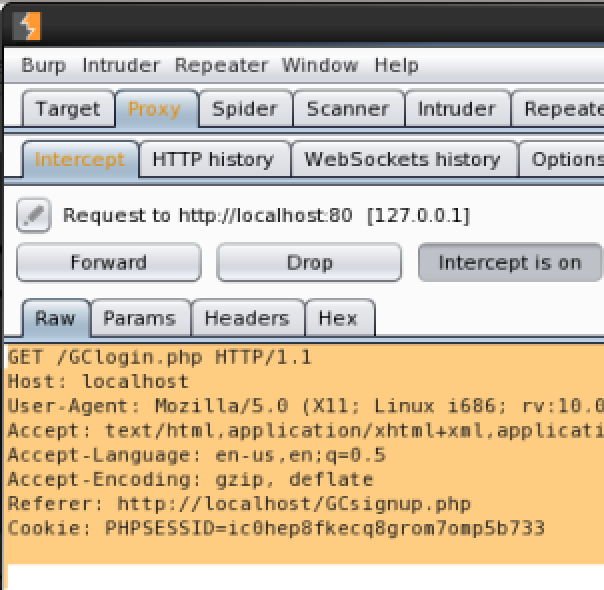


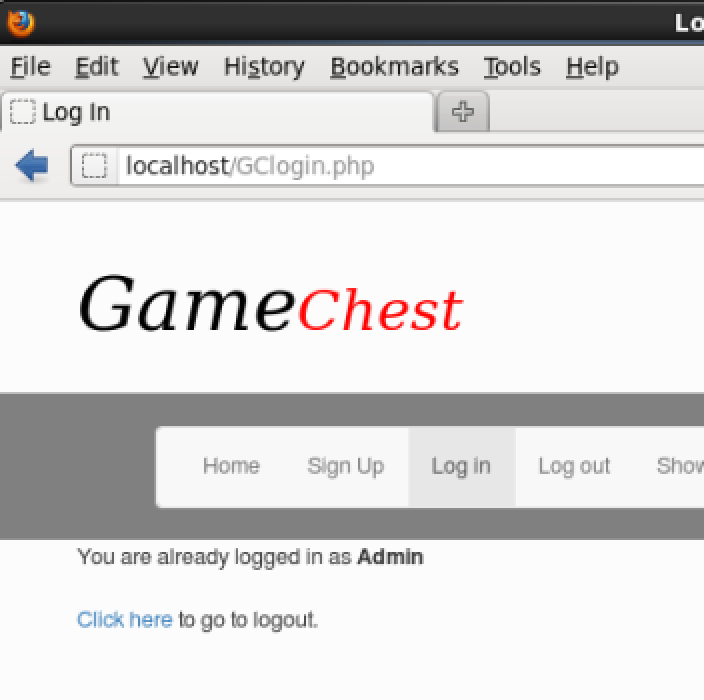
Apart from this we can use this session value to access the account without login on a different machine. I have tested this with different browser(opera and Firefox), then I changed PSPSESSID for opera to be equal to the PHPSESSID in Firefox to access the logged in account in Firefox or vice versa:

Session ID in opera with admin account:

Firefox without any login: 

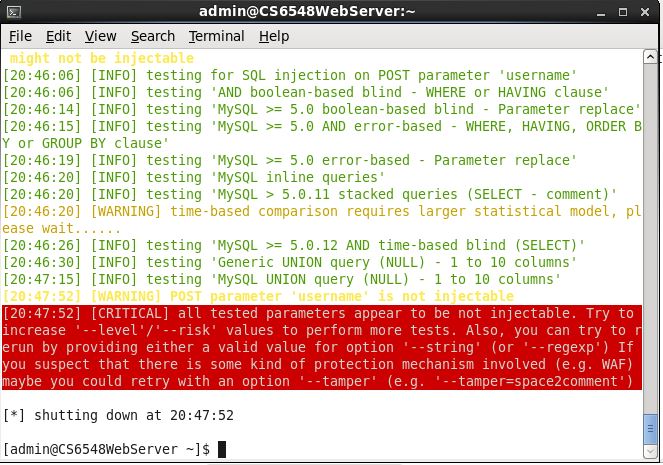
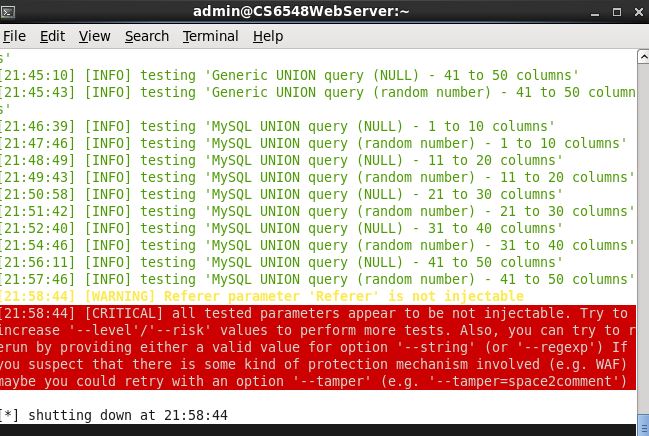
Burp Suite for Firefox: 

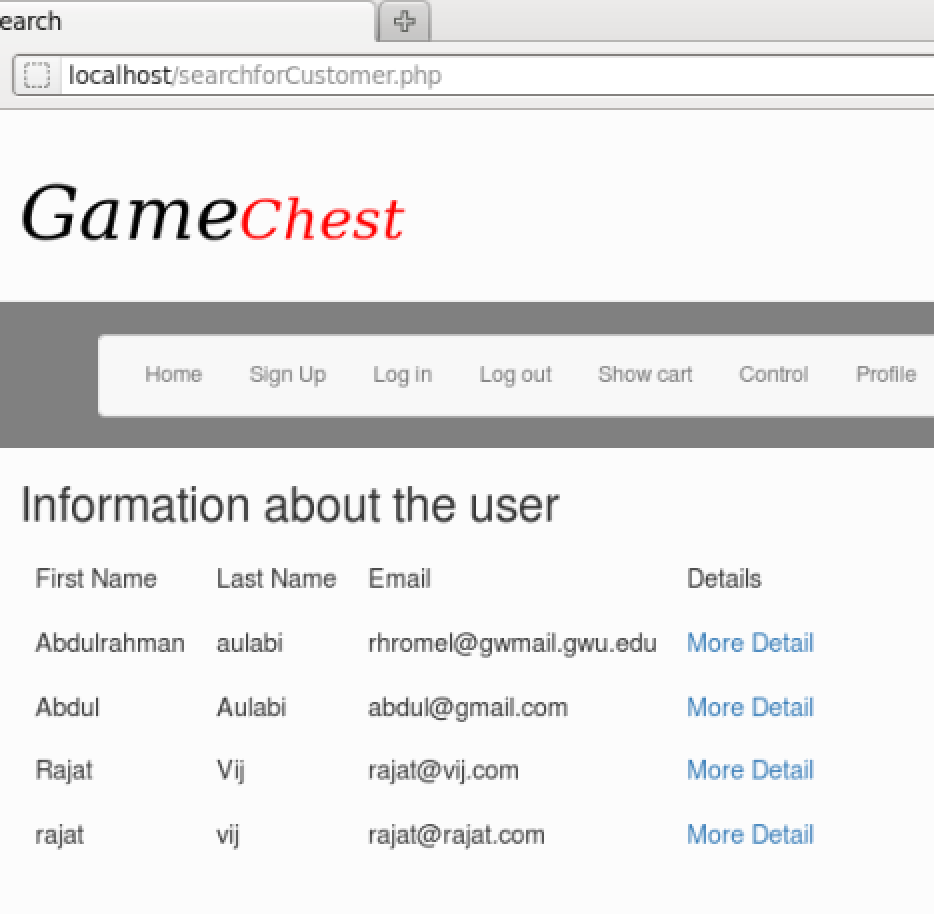
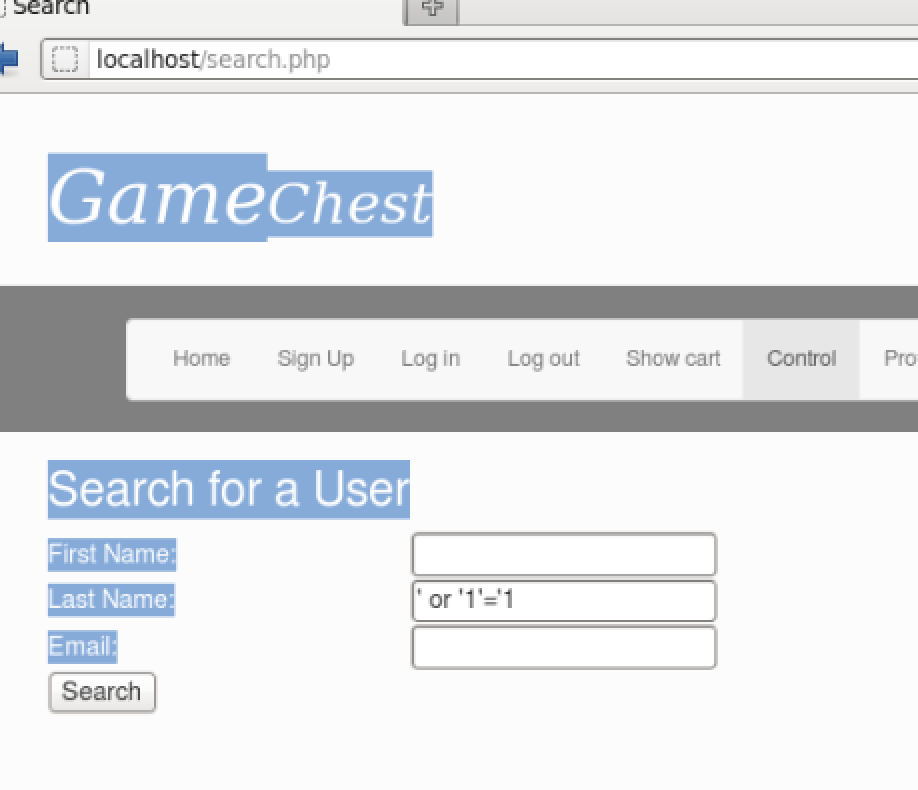
Change the Cookie value: 

We are logged in as Admin in Firefox: 

## SQL Injection

I have tried to use SQL Map to automate the sql injection which didn’t give much result. As we mentioned in our presentation:

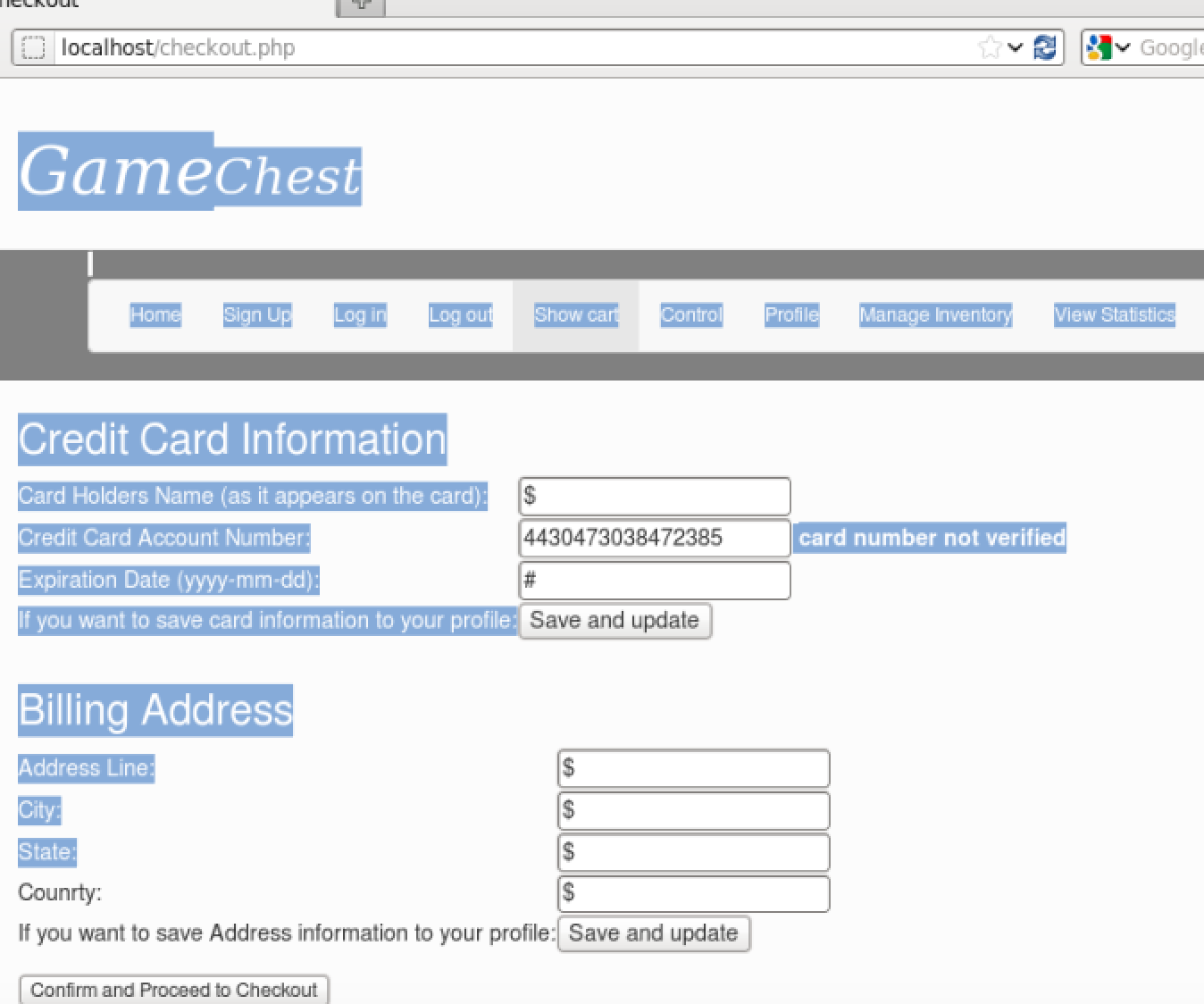


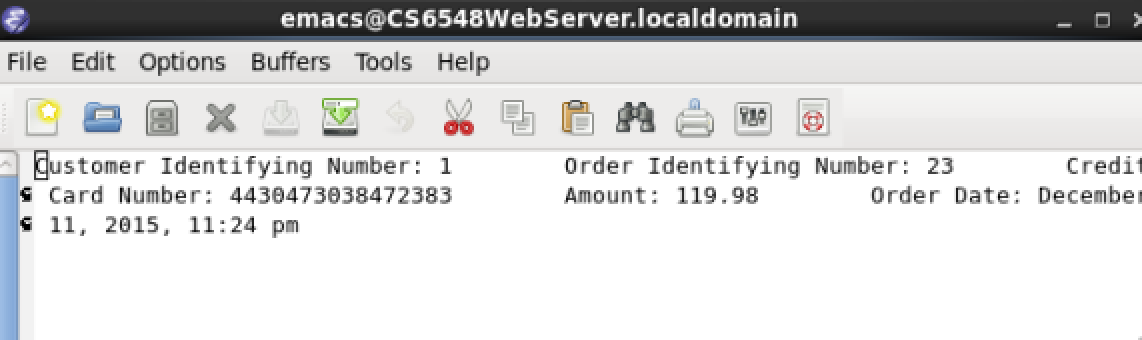
On Search user screen we can get all results from the database. 

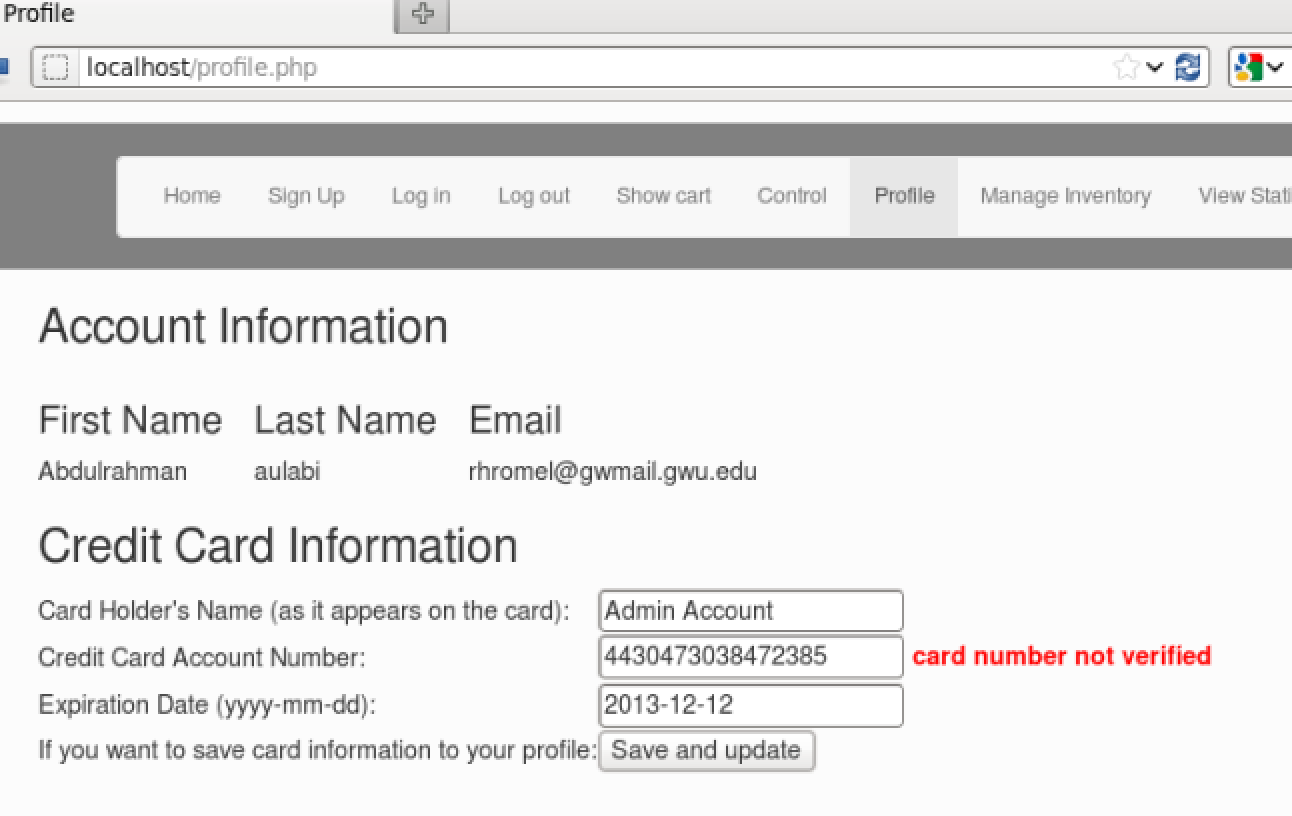
## Logic Flaws

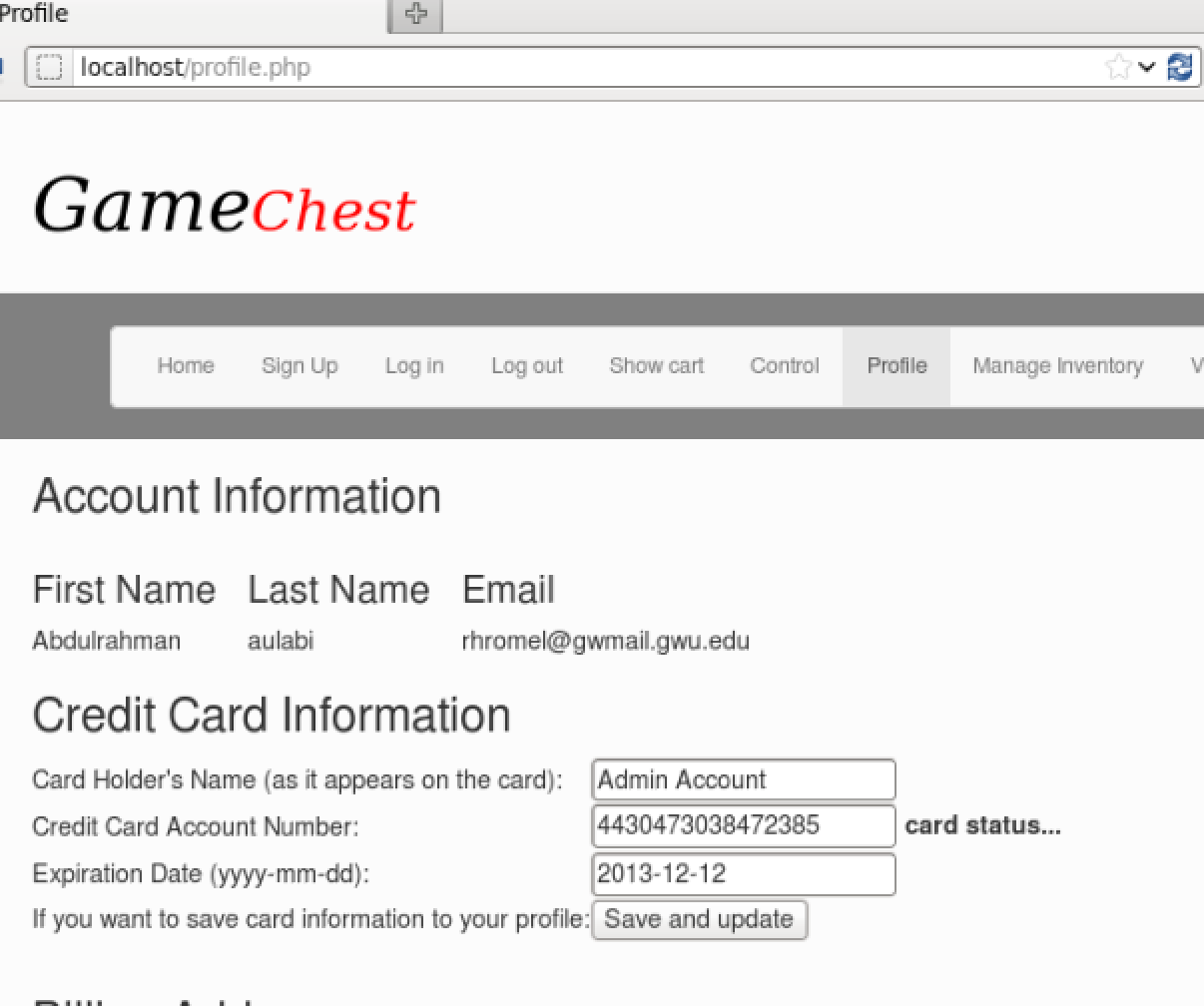
[Describe your tests for logic flaws; include screenshots, especially if you found any instances of logic flaws. Explain how any found logic flaws could be used by an attacker. ]

On Checkout and Profile page the web application is only checking for Credit card and even when credit card is invalid, I am able to checkout or save the profile with invalid credit card.

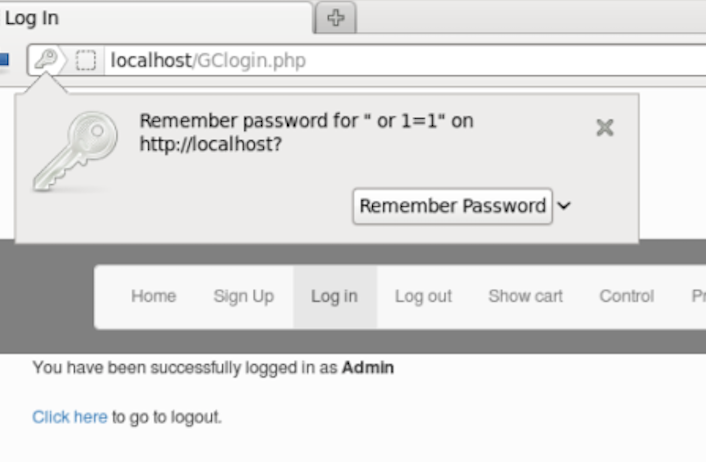








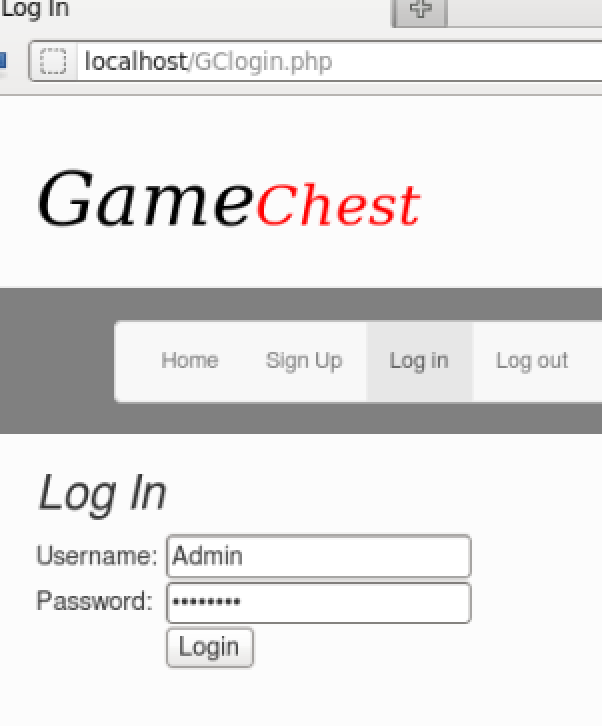
On Login Page we can Ignore Username check:

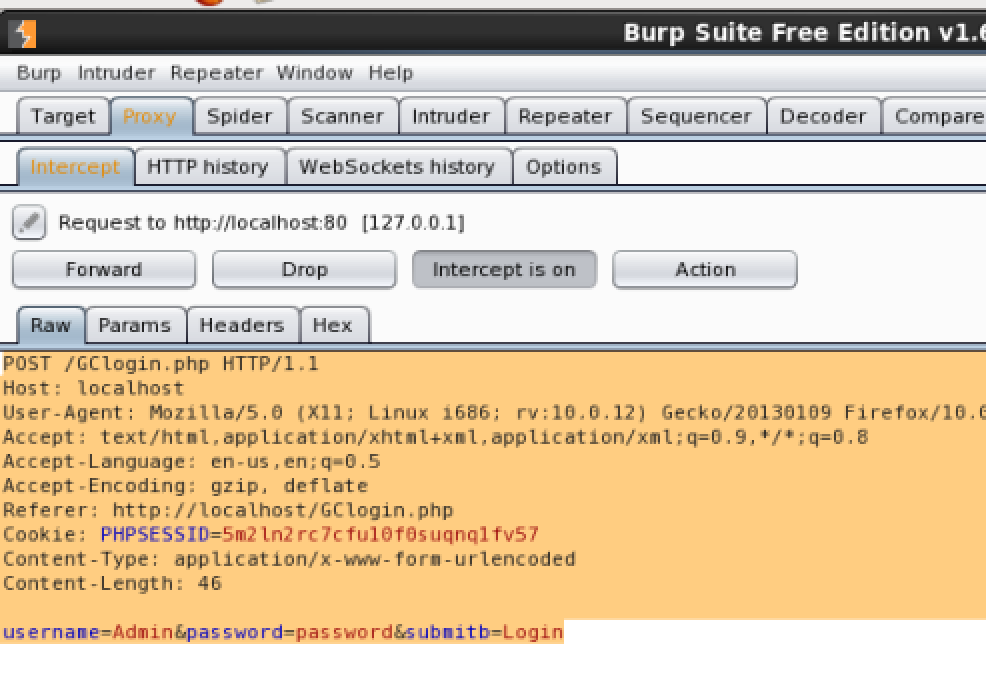


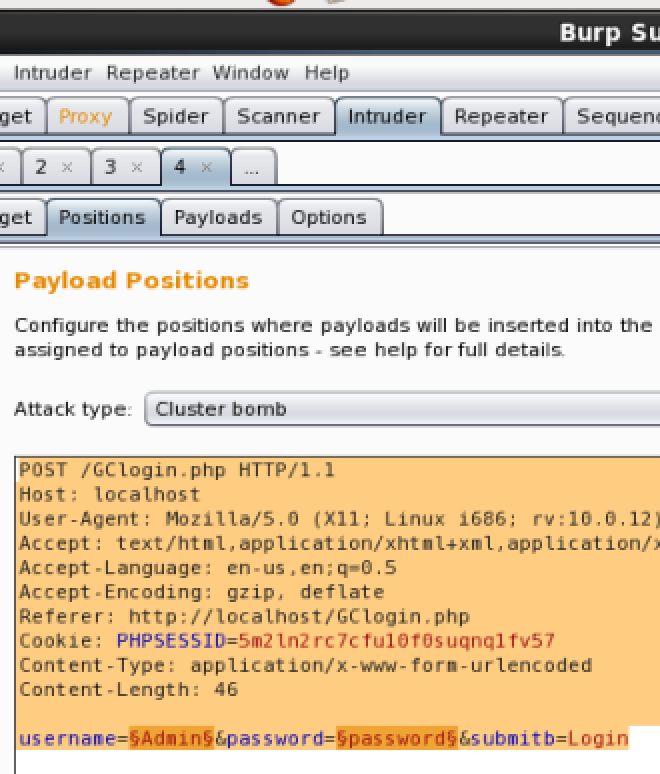
## Other’s

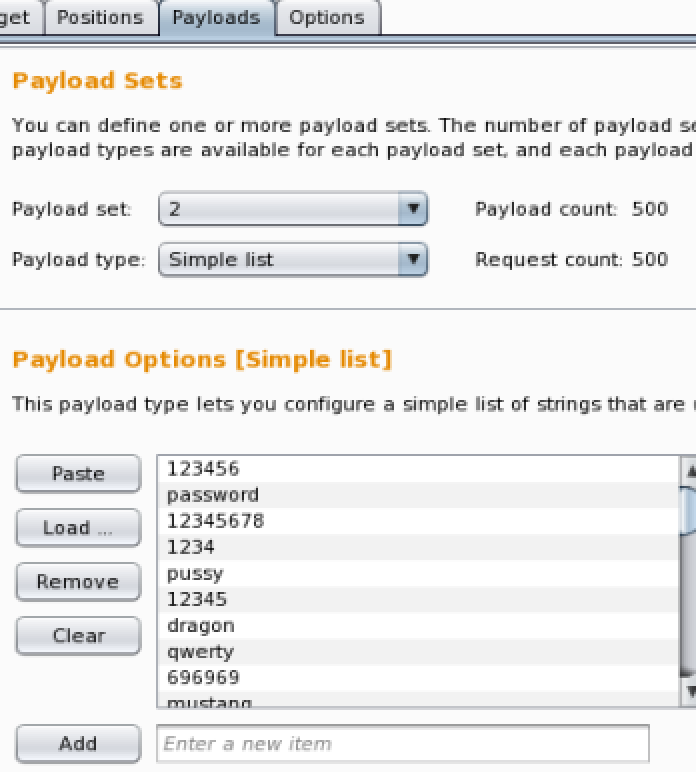
Form Submission of Password is in clear text so anyone doing MITM attack can know username and password as the md5 hashing of password is done at server side.

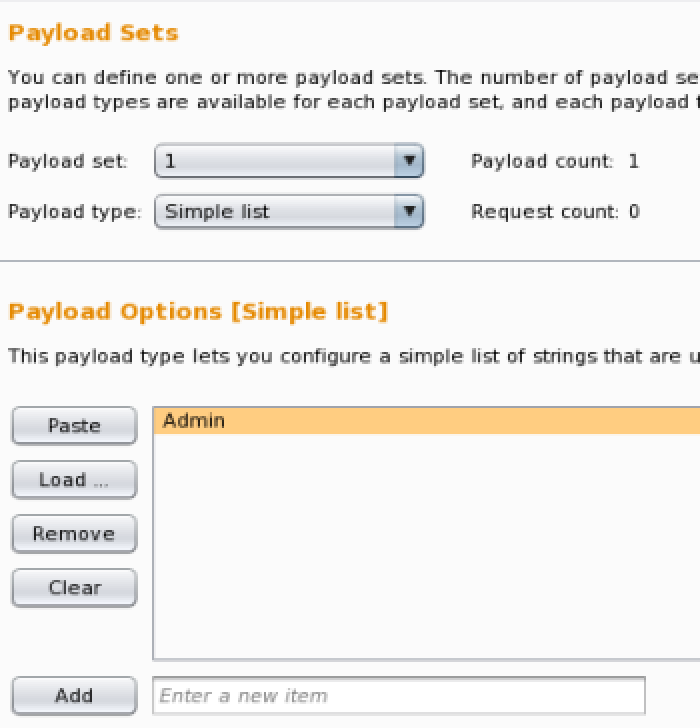
Brute Force: Due to logic flaw in Login screen and the application not checking invalid login attempts I can use brute force to login the application with a list of common passwords using Burp suite’s intrude mode.

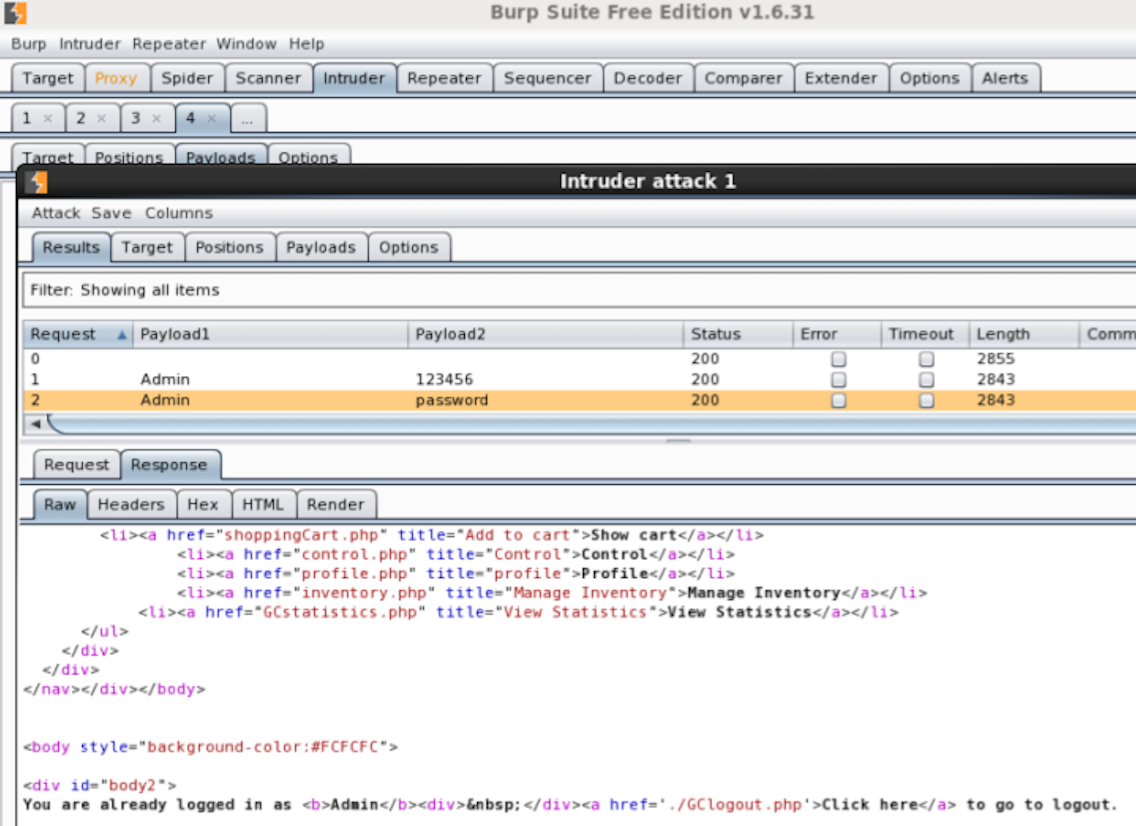




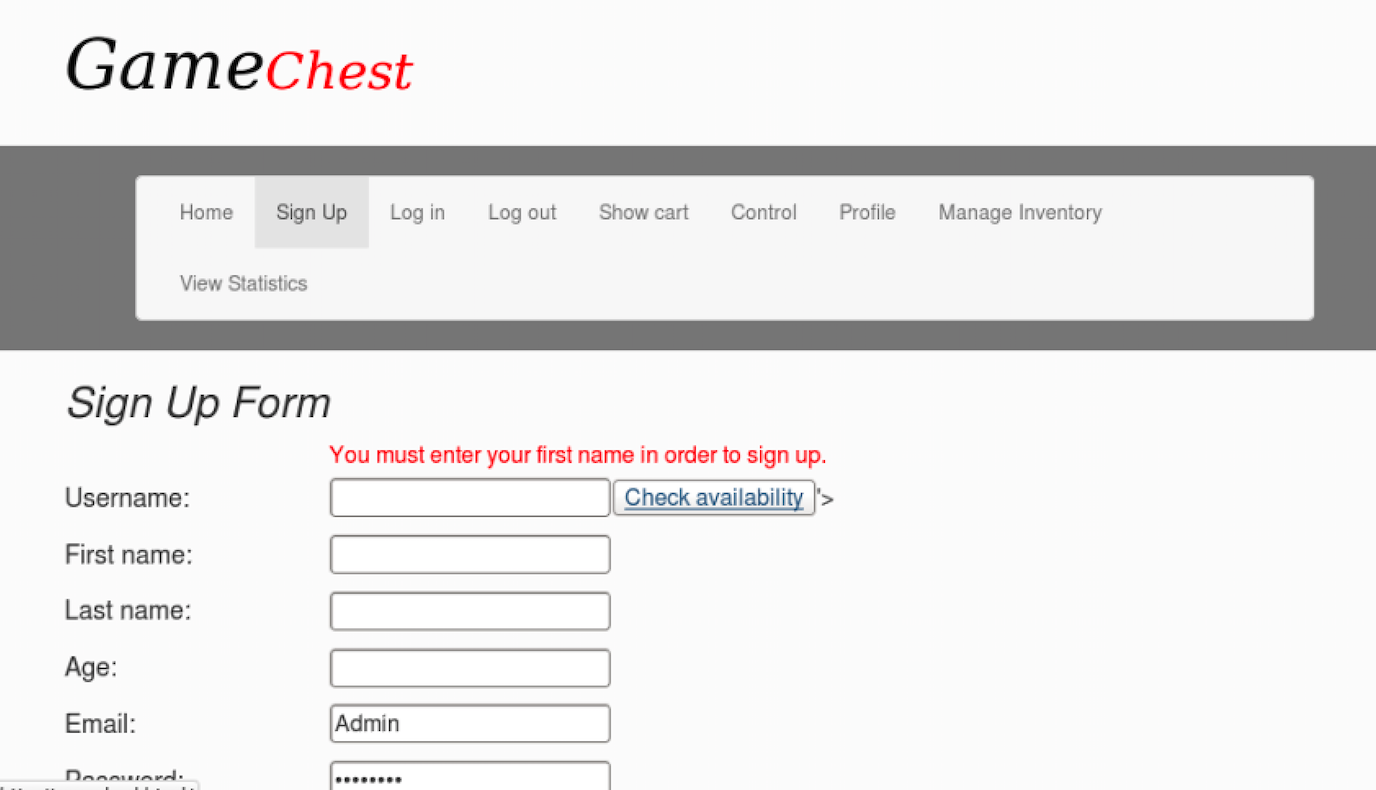
Here instead of Admin I can use ‘ or ‘1’=’1 to bypass username check.

Giving list of common passwords.





Clickjacking: Here we can add a button which forwards user to malicious page.



# Recommendations

For CSRF, we can use PHP CSRF Guard framework to prevent CSRF attacks. For XSS we can check input at the server side for script tags.

There are a few logical and validation flaw which can be corrected with minimal effort.

For SQL Injection Attacks we can use prepared SQL statements instead of normal SQL statements.

To avoid brute forcing, we can add a functionality to block a user after a limited number or invalid login attempts.

# Challenges

Doing a code review was confusing as their forms were in php variables understanding which was kind confusing.

Finding correct syntax to use XSS and SQL Injection vulnerabilities found was challenging.

I only have a limited knowledge of JQuery so I couldn’t perform CSRF attacks which are possible such as sending forged formed data in our forged requests. I could only use saved profile information even after knowing that It is possible to send forged form data using JQuery to perform better CSRF attacks.