

# MOP Vulnerability scan

By

Usman Tariq

S217034263

[S217034263@deakin.edu.au](mailto:S217034263@deakin.edu.au)

## Vulnerability:

A vulnerability is a flaw in an IT system that can be exploited by an attacker to launch a successful attack. They can occur because of defects, features, or user error, and attackers will seek to exploit any of them, typically combining one or more to achieve their ultimate purpose.

## Target:

<https://react-test-6najyje5cq-uc.a.run.app/>

## Tools used:

Nikto, Nessus and BURP-suite.

## Using Nikto

```
(kali@kali)~$ nikto -h https://react-test-6najyje5cq-uc.a.run.app/ -C all
- Nikto v2.5.0

+ Multiple IPs found: 216.239.34.53, 216.239.38.53, 216.239.32.53, 216.239.36.53, 2001:4860:4802:32::35, 2001:4860:4802:36::35, 2001:4860:4802:34::35, 2001:4860:4802:38::35
+ Target IP: 216.239.34.53
+ Target Hostname: react-test-6najyje5cq-uc.a.run.app
+ Target Port: 443

+ SSL Info: Subject: /CN=*.a.run.app
            Ciphers: TLS_AES_256_GCM_SHA384
            Issuer: /C=US/O=Google Trust Services LLC/CN=GTS CA 1C3
+ Start Time: 2024-03-29 08:38:46 (GMT-4)

+ Server: Google Frontend
+ /: The anti-clickjacking X-Frame-Options header is not present. See: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Frame-Options
+ /: Uncommon header 'x-cloud-trace-context' found, with contents: 67fe9eac8444d5e2f48da13e968329a6.
+ /: The site uses TLS and the Strict-Transport-Security HTTP header is not defined. See: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Strict-Transport-Security
+ /: An alt-svc header was found which is advertising HTTP/3. The endpoint is: ':443'. Nikto cannot test HTTP/3 over QUIC. See: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/alt-svc
+ /: The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a different fashion to the MIME type. See: https://www.netsparker.com/web-vulnerability-scanner/vulnerabilities/mis-sing-content-type-header/
+ Server is using a wildcard certificate: *.a.run.app. See: https://en.wikipedia.org/wiki/Wildcard_certificate
```

When we used 'Nikto' to scan vulnerabilities. We found the following vulnerabilities

### Missing X-Frame-Options Header:

The absence of the X-Frame-Options header enables for clickjacking assaults, in which malicious material overlays the website and deceives visitors into interacting inadvertently.

### **Missing Strict Transport Security (HSTS) Header:**

Without HSTS, the application is subject to SSL-stripping attacks, which undermine the security of the HTTPS connection.

### **Missing X-Content-Type-Options header:**

The absence of an X-Content-Type-Options header makes the application vulnerable to MIME-sniffing attacks, which could result in the execution of malicious code.

### **Use of Wildcard Certificates:**

While not intrinsically dangerous, wildcard certificates can increase the attack surface, permitting impersonation or other security dangers across several subdomains if not properly controlled. Addressing these concerns is critical for risk mitigation, as their absence increases vulnerability to clickjacking, SSL-stripping, MIME sniffing, and other security threats. Proper security headers and setups are strongly advised.

## **Using OWASP ZAP**

After running an Active scan in 'ZAP' we found that there are few vulnerabilities.

## **Alerts**

**Risk=Medium, Confidence=High (1)**

<https://react-test-6najtje5cq-uc.a.run.app> (1)

**Content Security Policy (CSP) Header Not Set (1)**

▼ GET <https://react-test-6najtje5cq-uc.a.run.app/>

|                          |   |
|--------------------------|---|
| <b>Alert tags</b>        | <ul style="list-style-type: none"><li>▪ <a href="#">OWASP_2021_A05</a></li><li>▪ <a href="#">OWASP_2017_A06</a></li></ul>   |
| <b>Alert description</b> | <p>Content Security Policy (CSP) is an added layer of security that helps to detect and mitigate certain types of attacks, including Cross Site Scripting (XSS) and data injection attacks. These attacks are used for everything from data theft to site defacement or distribution of malware. CSP provides a set of standard HTTP headers that allow website owners to declare approved sources of content that browsers should be allowed to load on that page — covered types are JavaScript, CSS, HTML frames, fonts, images and embeddable objects such as Java applets, ActiveX, audio and video files.</p> |

### **Vulnerability:**

Content protection Policy (CSP) is an additional layer of protection that aids in the detection and mitigation of specific sorts of attacks, such as cross-site scripting (XSS) and data injection. These assaults

are used for a variety of purposes, including data theft, website defacement, and malware dissemination. CSP provides a set of standard HTTP headers that allow website owners to declare approved sources of content for browsers to load on that page, including JavaScript, CSS, HTML frames, fonts, images, and embeddable objects like Java applets, ActiveX, audio, and video files.[1]

### **Solution:**

Make sure that your web server, application server, load balancer, and so on are configured to set the Content-Security-Policy header.[1]

<https://fonts.googleapis.com> (1)

**Cross-Domain Misconfiguration (1)**

▼ GET <https://fonts.googleapis.com/css2?family=Montserrat:wght@100;400;900&family=Poppins&family=Reem+Kufi+Fun:wght@600&family=Ubuntu&display=swap>

|                          |  |
|--------------------------|--|
| <b>Alert tags</b>        | <ul style="list-style-type: none"><li>▪ <a href="#">OWASP_2021_A01</a></li><li>▪ <a href="#">OWASP_2017_A05</a></li></ul>  |
| <b>Alert description</b> | Web browser data loading may be possible, due to a Cross Origin Resource Sharing (CORS) misconfiguration on the web server   |
| <b>Other info</b>        | The CORS misconfiguration on the web server permits cross-domain read requests from arbitrary third party domains, using unauthenticated APIs on this domain. Web browser implementations do not permit arbitrary third parties to read the response from authenticated APIs, however. This reduces the risk somewhat. This misconfiguration could be used by an attacker to access data that is available in an unauthenticated manner, but which uses some other form of security, such as IP address white-listing. |

### **Vulnerability:**

A misconfiguration of Cross Origin Resource Sharing (CORS) on the web server may allow web browser data loading. The web server's CORS misconfiguration allows cross-domain read requests from arbitrary third-party domains, which use unauthenticated APIs on this domain. However, web browser implementations prevent arbitrary third parties from reading responses from authenticated APIs. This lowers the danger slightly. An attacker could leverage this misconfiguration to gain access to data that is available unauthenticated but protected by another form of security, such as IP address whitelisting.[1]

### **Solution:**

Ensure that sensitive material is not available unauthenticated (for example, by whitelisting IP addresses). range the "Access-Control-Allow-Origin" HTTP header to a more restrictive range of domains, or delete all CORS headers entirely, to allow the web browser to enforce the Same Origin Policy (SOP) more strictly.[1]

<https://react-test-6najok5c5-uc.a.run.app> (1)

### Missing Anti-clickjacking Header (1)

► GET <https://react-test-6najok5c5-uc.a.run.app/>

#### **Vulnerability:**

Clickjacking is an attack which manipulates users into thinking that they are clicking on something but in reality they clicked on something else. This can lead to unknowingly downloading malware.[1]

#### **Solution:**

The X-Frame-Options and Content-Security-Policy HTTP headers are supported by most modern web browsers. Make sure that one of them is selected on every webpage that your site or app returns. You should use SAMEORIGIN if you anticipate that the page will only ever be framed by pages on your server (for example, if it is a member of a FRAMESET); if not, you should use DENY. As an alternative, think about putting Content Security Policy's "frame-ancestors" command into practice.[1]

**Risk=Low, Confidence=High (1)**

<https://react-test-6najok5c5-uc.a.run.app> (1)

### Strict-Transport-Security Header Not Set (1)

► GET <https://react-test-6najok5c5-uc.a.run.app/>

#### **Vulnerability:**

A web server that complies with HTTP Strict Transport Security (HSTS) proclaims that compliant user agents, like a web browser, must only communicate with it through secure HTTPS connections (HTTP layered over TLS/SSL). RFC 6797 specifies the HSTS protocol, which is an IETF standards track protocol.[1]

#### **Solution:**

Ensure that Strict-Transport-Security is enforced by your load balancer, web server, application server, etc.[1]

**Risk=Low, Confidence=Medium (1)**

<https://react-test-6najok5c9-uc.a.run.app> (1)

**X-Content-Type-Options Header Missing (1)**

► GET <https://react-test-6najok5c9-uc.a.run.app/>

**Vulnerability:**



There was no setting for "no sniff" in the Anti-MIME-Sniffing header X-Content-Type-Options. Due to this, the response body may be interpreted and displayed as a content type different from the declared content type in older versions of Internet Explorer and Chrome when MIME-sniffing is performed on it. Instead of using MIME-sniffing, the defined content type (if one is set) will be used by both early 2014 versions of Firefox and older versions.[1]


**Solution:**

Make sure the application or web server sets the X-Content-Type-Options header to 'no sniff' for every web page and that the Content-Type header is set appropriately. Make every effort to guarantee that the end user is using a current, standards-compliant web browser that either doesn't do MIME-sniffing at all or can be instructed by the web application/web server to do so.[1]

## **Using Burp Suite**

I was able to perform clickjack attack on this webapp using 'Clickbandit' function which is available in Burp suite.

Burp Clickbandit

 **Burp Clickbandit**

Burp Clickbandit is a tool for generating clickjacking attacks. When you have found a web page that may be vulnerable to clickjacking, you can use Burp Clickbandit to create an attack, and confirm that the vulnerability can be successfully exploited.

Burp Clickbandit runs in your browser using JavaScript. It works on all modern browsers except for Microsoft IE and Edge. To run Burp Clickbandit, use the following steps:

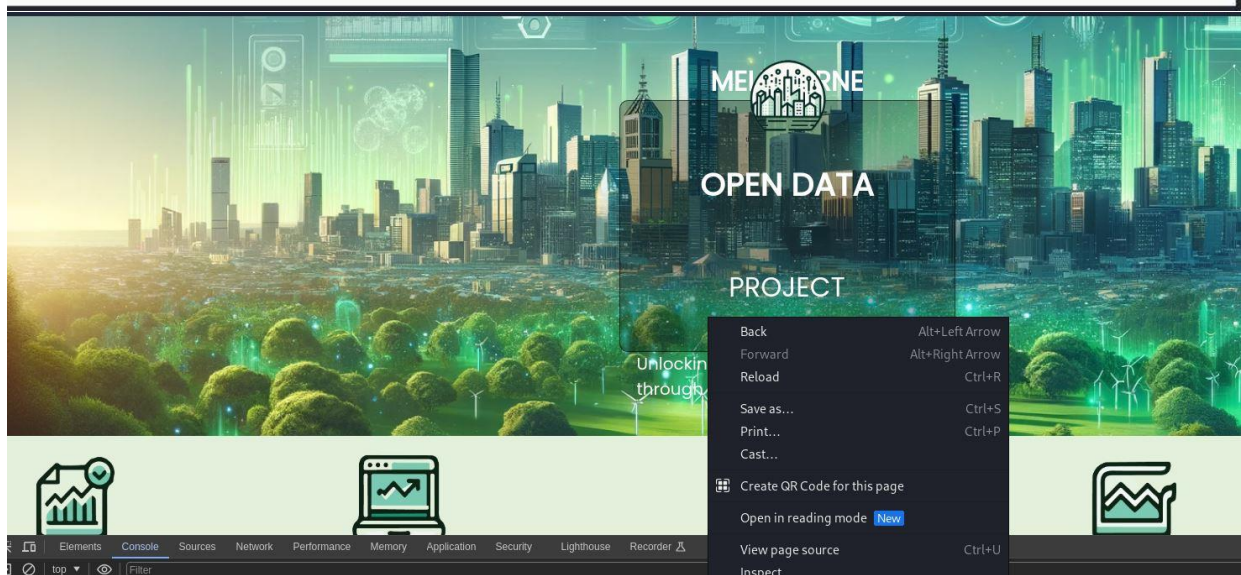
1. Click the "Copy Clickbandit to clipboard" button below. This will copy the Clickbandit script to your clipboard.
2. In your browser, visit the web page that you want to test, in the usual way.
3. In your browser, open the web developer console. This might also be called "developer tools" or "JavaScript console".
4. Paste the Clickbandit script into the web developer console, and press enter.

See the documentation for more details on using Burp Clickbandit.

*Note: Exercise caution when running Burp Clickbandit on untrusted websites. Malicious JavaScript from the target site can subvert the HTML output that is generated by Burp Clickbandit.*

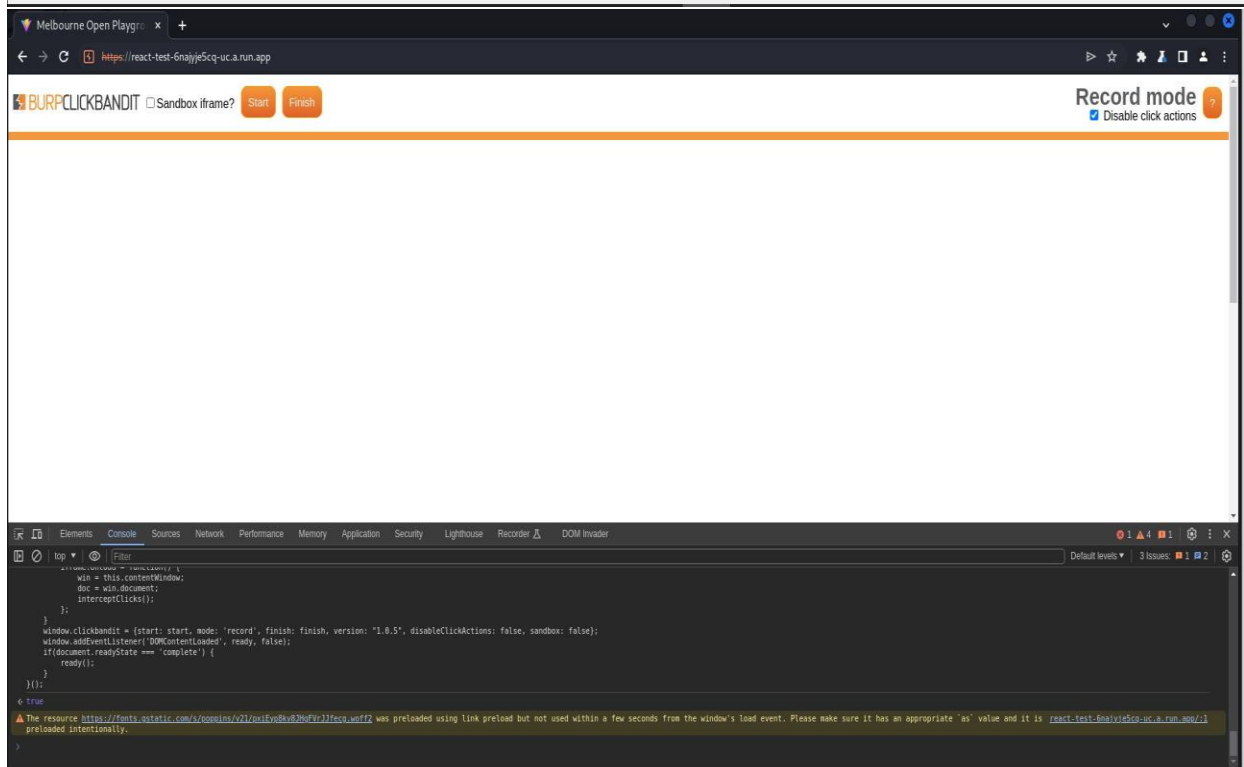
Copy Clickbandit to clipboard

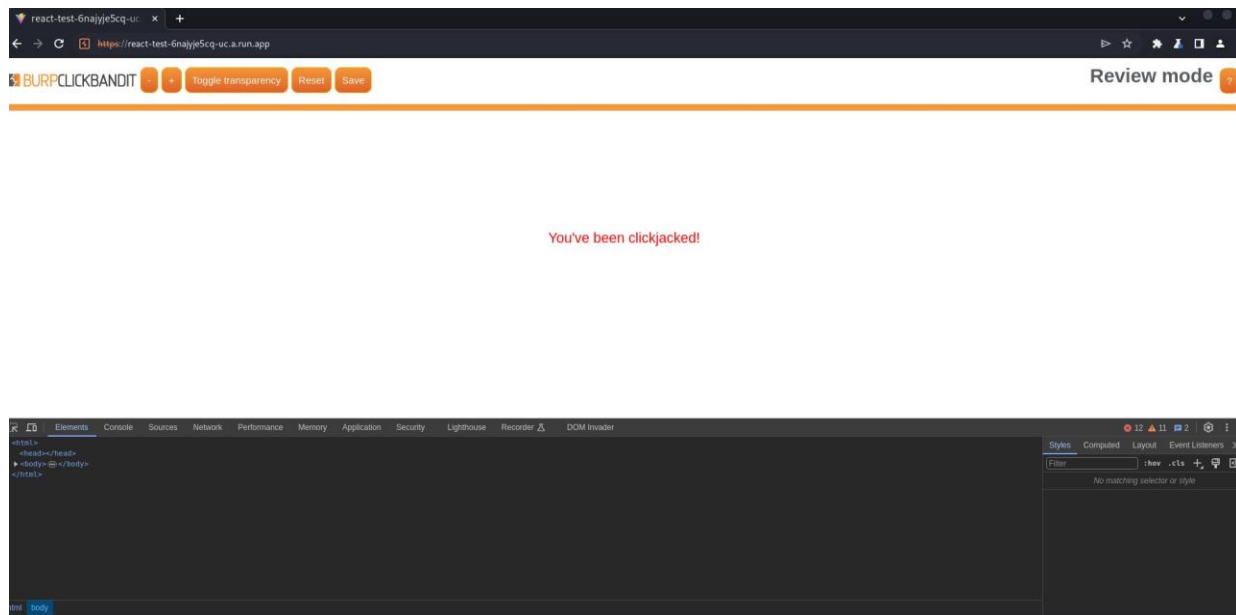
Close





```
Elements Console Sources Network Performance Memory Application Security Lighthouse Recorder DOM Invader
top [Filter]
A preload for 'https://fonts.gstatic.com/s/poppins/v21/pxiEyp8kv8JHgFVr3Jfecg.woff2' is found, but is not used because the request credentials mode does not match. Consider taking a look
GET https://react-test-6najiye5cq-uc-a.run.app/src/assets/react.svg 404 (Not Found)
The resource https://fonts.gstatic.com/s/poppins/v21/pxiEyp8kv8JHgFVr3Jfecg.woff2 was preloaded using link preload but not used within a few seconds from the window's load event. Please
preload intentionally.
> /* Copyright PortSwigger Ltd. All rights reserved. Usage is subject to the Burp Suite license terms. See https://portswigger.net for more details. */
!function(){
  var initialZoomFactor = '1.0', win, doc, width, height, clicks = [];
  function addClickTrap(element, minusY) {
    var clickTrap = doc.createElement('div'), cords = findPos(element);
    clickTrap.style.backgroundColor = 'none';
    clickTrap.style.border = 'none';
    clickTrap.style.position = 'absolute';
    clickTrap.style.left = cords[0] + 'px';
    clickTrap.style.top = cords[1] + 'px';
    clickTrap.style.width = element.offsetWidth + 'px';
    clickTrap.style.height = element.offsetHeight + 'px';
    if(element.zIndex > element.zIndex === '0') {
      clickTrap.style.zIndex = +element.zIndex+1;
    }
    clickTrap.style.opacity = '0.5';
    clickTrap.style.cursor = 'pointer';
    clickTrap.clickTrap = 1;
    clickTrap.addEventListener('click', function(e) {
      generatePoc({x:e.pageX, y: minusY?e.pageY-minusY : e.pageY});
      e.preventDefault();
      e.stopPropagation();
      return false;
    }, true);
    doc.body.appendChild(clickTrap);
  }
  function addMessage(msg) {
    var message = document.createElement('div');
    message.style.width = '100%';
    message.style.height = '20px';
    message.style.backgroundColor = 'ffff5bf';
    message.style.border = '1px solid #ff9900';
    message.style.padding = '5px';
    message.style.position = 'fixed';
    message.style.bottom = '0';
    message.style.left = '0';
    message.style.zIndex = '100000';
    message.style.textAlign = 'center';
    message.style.fontFamily = 'Arial';
    message.style.color = '#000';
    message.appendChild(document.createTextNode(msg));
    document.body.appendChild(message);
  }
}
```





### **Solution:**

The X-Frame-Options and Content-Security-Policy HTTP headers are supported by most modern web browsers. Make sure that one of them is selected on every webpage that your site or app returns. You should use SAMEORIGIN if you anticipate that the page will only ever be framed by pages on your server (for example, if it is a member of a FRAMESET); if not, you should use DENY. As an alternative, think about putting Content Security Policy's "frame-ancestors" command into practice.[1]

### **Reference:**

1. *The zap homepage*. ZAP. (n.d.). <https://www.zaproxy.org/>