A close up of a dragon's face

Description automatically generated

**Finding Name: XSS attack due to absence of X-Content-Type-Options in Econet.**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Name** | **Team** | **Role** | **Project** | **Quality Assurance** | **Is this a re-tested Finding?** |
| Dev Patel | PT | Team Member | EchoNet |  |  |
| Phuoc Hung Huynh | PT | Team Member | EchoNet |  |  |

|  |
| --- |
| **Was this Finding Successful?** |
| Yes |

**Finding Description**

During a recent security assessment of the EchoNet web application, I utilized Nikto, which uncovered significant security vulnerabilities in the HTTP response headers. These headers are essential for ensuring the integrity and security of user sessions and data. Notably, the application's responses were missing two critical headers: (X-Frame-Options and X-Content-Type-Options)

The missing headers lead us to a stored Cross-Site Scripting (XSS) vulnerability that was discovered in the application, specifically affecting the [Conservation request change]. This vulnerability occurs when user input is not properly sanitized and is subsequently rendered on the admin side without adequate validation or encoding. This allows an attacker to inject malicious JavaScript code into the application, which is then executed in the context of an authenticated admin user's browser session

**Risk Rating**  
Impact: Major  
Likelihood: High

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Impact values** | | | | |
| **Very Minor** | **Minor** | **Significant** | **Major** | **Severe** |
| Risk that holds little to no impact. Will not cause damage and regular activity can continue. | Risk that holds minor form of impact, but not significant enough to be of threat. Can cause some damage but not enough to impede regular activity. | Risk that holds enough impact to be somewhat of a threat. Will cause damage that can impede regular activity but will be able to run normally. | Risk that holds major impact to be of threat. Will cause damage that will impede regular activity and will not be able to run normally. | Risk that holds severe impact and is a threat. Will cause critical damage that can cease activity to be run. |



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Likelihood** | | | | |
| **Rare** | **Unlikely** | **Moderate** | **High** | **Certain** |
| Event may occur and/or if it did, it happens in specific circumstances. | Event could occur occasionally and/or could happen (at some point) | Event may occur and/or happens. | Event occurs at times and/or probably happens a lot. | Event is occurring now and/or happens frequently. |

**Business Impact**

The absence of X-Frame-Options and X-Content-Type-Options headers leaves the application vulnerable to several types of attacks.

First, it exposes the application to Clickjacking attacks. Without the X-Frame-Options header, malicious websites can embed vulnerable pages from your application in a frame, deceiving users into clicking on elements they didn't intend to click. This can result in unauthorized actions being executed on behalf of the user.

Additionally, the application is susceptible to MIME type sniffing attacks. If the X-Content-Type-Options header is not set to nosniff, browsers might misinterpret the content type of a document, differing from what the server specified. Attackers can exploit this to serve malicious executable code disguised as benign file types, potentially leading to cross-site scripting (XSS) attacks or other types of code injection vulnerabilities.

Because of lack of proper encoding of output data/values, XSS attack is possible. This can lead to various client-side attacks including attempts like :-

* Hijack the user’s session and perform actions on their behalf
* Steal the user’s credentials
* Hijacking the user’s browser or delivering browser-based exploits
* Obtain sensitive information stored in the user’s account or in their browser

**Affected Assets**

* Application server
* User Data

**Evidence**

First, we run the nikto tool with command. The -h option in Nikto is used to specify the target host or IP address we want to scan. This option is crucial as it tells Nikto which web server to assess for vulnerabilities.

A screenshot of a computer

Description automatically generated

*Figure 1: Running nikto to scan for vulnerabilities*

After accessing on Echo website, we run developer tool in browser to show Missing X-Content-Type-Options header:

**A screenshot of a computer

Description automatically generated**

*Figure 2: Missing X-Content-Type-Options header*

As we can see in Headers, the X-Content-Type-Options header is not listed, it indicates that the server is not setting this header. Without it, browsers that perform MIME type sniffing may incorrectly interpret the content type of files, which can lead to security vulnerabilities as previously described.

For another way, we can run curl command to show that the X-Content-Type-Options header is missing.

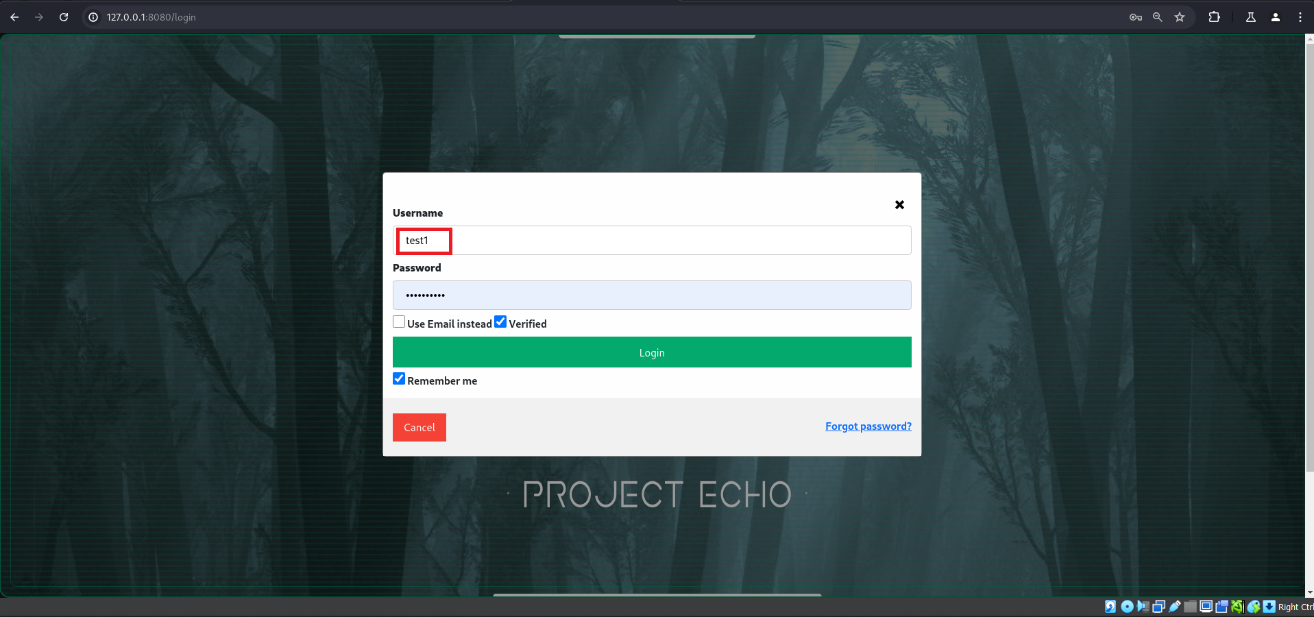
A computer screen shot of a computer

Description automatically generated

*Figure 3: X-Content-Type-Options header is missing when running curl*

In the next stage, will perform XSS attack to prove that this missing can be vulnerable to the EchoNet.

Now we will login to portal with user (test1).



*Figure 4: login page*

Now navigate to /map and select the any animal and open the summary section and scroll down and click on request Edit button.

A map of the ocean

Description automatically generated

*Figure 5: map page with all animals and microphone location*

After clicking the Edit button new pop-up window will open, after that we will fill details in input field selecting Near-threatened with adding Deakin URL in source section and before click on Submit button need to setup up the Burp-suite to intercept the request and modify that you can see in next step.

A screenshot of a computer screen

Description automatically generated

*Figure 6: Edit description request window*

Capturing the request in Burp Suite.

A screenshot of a computer

Description automatically generated

*Figure 7: Burp-suite with captured request*

Now modify that request and add the XSS payload to perform the Attack.

A screenshot of a computer

Description automatically generated

*Figure 8: Burp-suite with modified request*

Got the response with the success message

A map of the ocean

Description automatically generated

*Figure 9: Success message for the request send*

Now we need to login to Admin user (rocky-34).

A screenshot of a computer

Description automatically generated

*Figure 10: Login page*

After login to the admin user, we will redirect to dashboard then will navigate to user Activity page having list of the requests where we will find the injected code (Stored XSS).

A screenshot of a computer

Description automatically generated

*Figure 11: User Activity page with list of requests*

Now as we move the mouse curser near the text test3 the payload execute and will pop-up the window with message (AppAttack XSS).

A screenshot of a computer

Description automatically generated

*Figure 12: Stored XSS with pop-up window*

**Remediation Advice**

We have to add X-Content-Type-Options: nosniff to instruct the browser to follow the MIME types declared in the Content-Type headers.

The easiest way to add the X-Content-Type-Options header (along with other essential security headers) is by using the helmet middleware.

We install helmet by running command: npm install helmet. Then, we integrate Helmet into Application by importing const helmet = require("helmet");

Examine every input that users provide to make sure it doesn't contain any potentially harmful characters that could alter how a user's browser displays the information on your website. Make sure characters are output-escaped and apply strict input validation.

Request blocking: Depending on the likelihood that a user-submitted input has a dangerous payload, you can choose to allow or block it. You might deny the request, for instance, if the input contains <script>, which indicates that it probably contains a payload that allows cross-site scripting.

The application should have proper output encoding to avoid XSS injection attacks. The output encoding should be used consistently across all outputs to the client like converting (< to &lt) and (> to &gt) etc.

**References**

MDN Web Docs. (2019). *X-Content-Type-Options*. [online] Available at: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Content-Type-Options>.

‌Khaled, A. (2024). *Helmet: Protecting Your Web Applications from Common Vulnerabilities. [online] Medium*. Available at: <https://medium.com/@amirakhaled2027/helmet-protecting-your-web-applications-from-common-vulnerabilities-34b7f2667e26#:~:text=Helmet%20is%20a%20powerful%20middleware>

[Accessed 7 Aug. 2024].

XSS Prevention methods at: <https://brightsec.com/blog/stored-xss/>

‌

**Contact Details**

Dev Patel - S222622553@deakin.edu.au

Phuoc Hung Huynh (Harris Huynh) - S222470938@deakin.edu.au

**Pentest Leader Feedback.**

The lead will provide feedback to enact on