Finding Name: Exploitable Cross-Origin Resource Sharing (CORS) Configuration

Name	Team	Role	Project	Quality Assurance	Is this a re-tested Finding?
Fahad Alshamri	Ontrack	Pentester	AppAttack	**Nicholas Krcevinac**	NO

Was this Finding Successful?
Yes

Finding Description

The OnTrack web application located at http://172.18.0.1:4200 is configured to respond with Access-Control-Allow-Origin: *, which allows cross-origin requests from any domain. This insecure CORS policy can be exploited by a remote attacker to retrieve API responses using JavaScript from a malicious page hosted on another domain.

Risk Rating

Impact: Significant
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

An attacker could host a malicious site that silently performs unauthorized API requests to the OnTrack server on behalf of a logged-in user or retrieve sensitive API responses if endpoints are exposed. This could result in data leaks, unauthorized access, or session manipulation if tokens are mismanaged.

Affected Assets

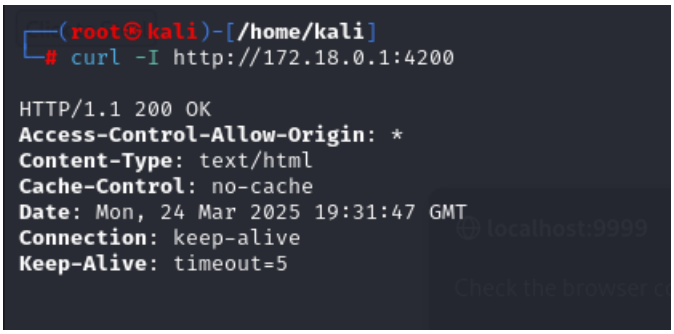
- OnTrack Web Frontend (http://172.18.0.1:4200)
- OnTrack API endpoints (/api/users)

Evidence

CORS Proof of Concept (POC)

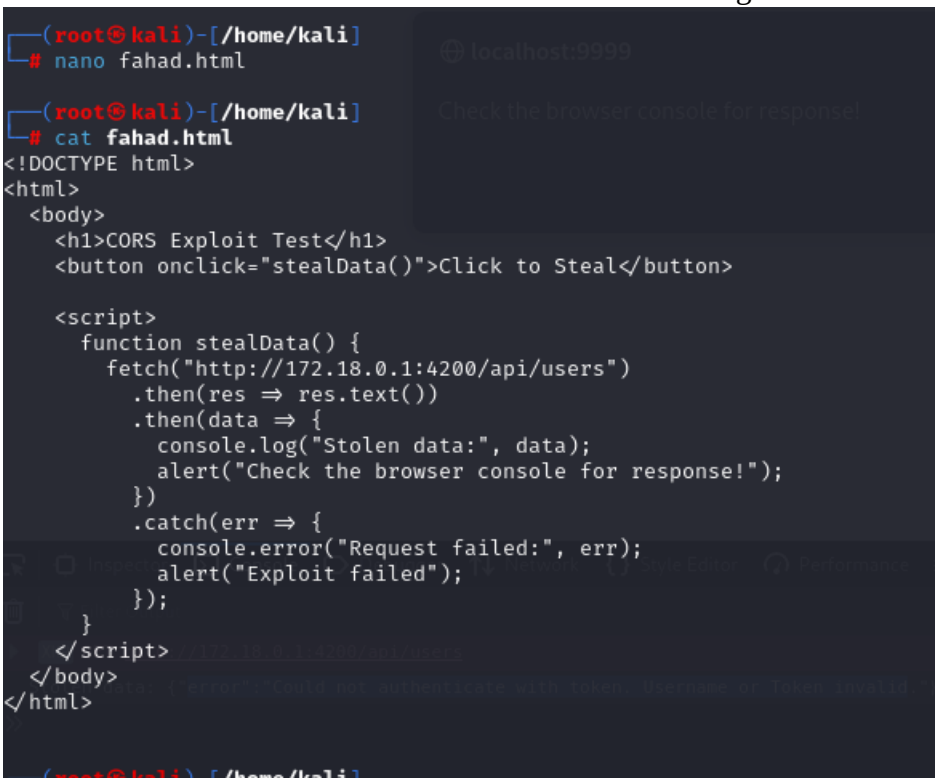
Step 1: Confirm CORS Policy

To determine whether CORS was improperly configured, a simple `curl` request was issued to the main application URL using the `-I` flag to fetch headers. The response included the header `Access-Control-Allow-Origin: *`, which indicated the server was allowing requests from any domain without restrictions.



Step 2: Create Malicious HTML Page

A malicious HTML file (`fahad.html`) was created locally. The file contained a script using the JavaScript Fetch API to send a GET request to the endpoint `/api/users` of the OnTrack web server. This simulated an attack from an unauthorized origin.



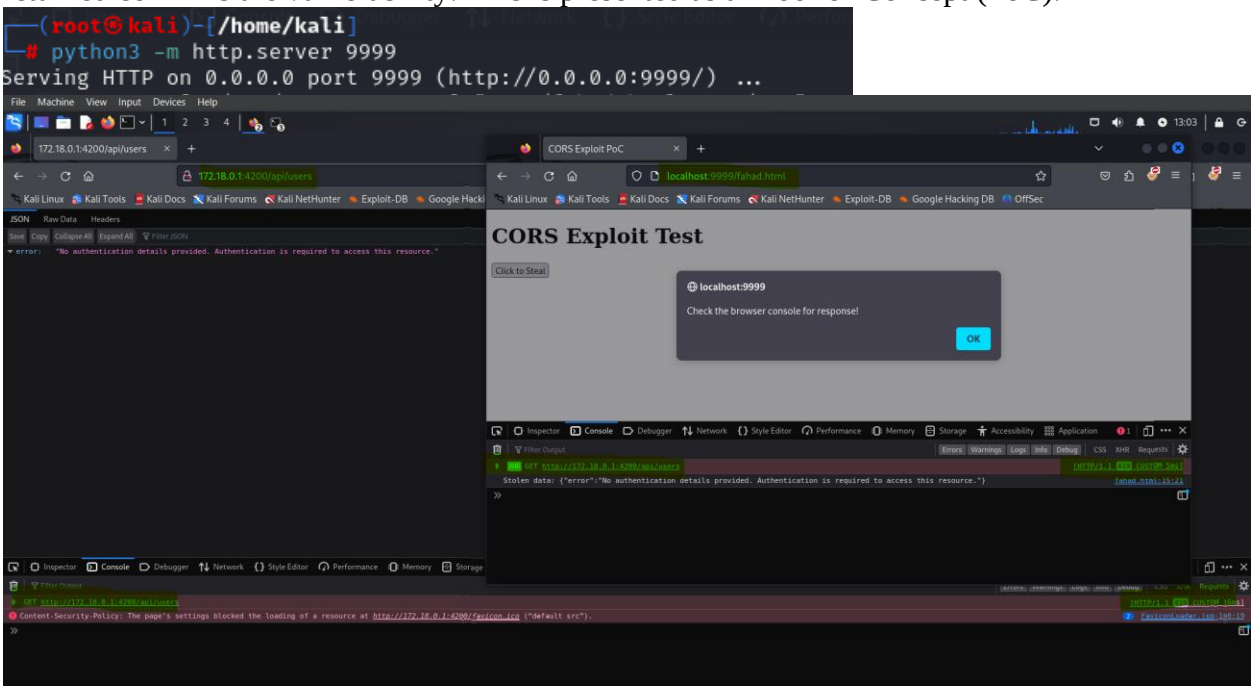
```
┌──(root㉿kali)-[/home/kali]
└─# nano fahad.html

┌──(root㉿kali)-[/home/kali]
└─# cat fahad.html
<!DOCTYPE html>
<html>
  <body>
    <h1>CORS Exploit Test</h1>
    <button onclick="stealData()">Click to Steal</button>

    <script>
      function stealData() {
        fetch("http://172.18.0.1:4200/api/users")
          .then(res => res.text())
          .then(data => {
            console.log("Stolen data:", data);
            alert("Check the browser console for response!");
          })
          .catch(err => {
            console.error("Request failed:", err);
            alert("Exploit failed");
          });
      }
    </script>
  </body>
</html>
```

Step 3: Host and Load the File & Trigger the Exploit

The fahad.html file was accessed from a different origin (localhost:9999), simulating a malicious domain. When the button was clicked, the browser issued a request to http://172.18.0.1:4200/api/users. The OnTrack server responded with a 419 error and JSON response indicating “No authentication details provided,” confirming that the server processed the cross-origin request. While no sensitive data was retrieved, the fact that a response was returned confirms the vulnerability. This is presented as a Proof of Concept (PoC).



Explanation:

This demonstration confirms a Cross-Origin Resource Sharing (CORS) misconfiguration in the OnTrack web application. The crafted fahad.html page, served from http://localhost:9999, successfully initiated a cross-origin GET request to http://172.18.0.1:4200/api/users. Although the server responded to the request, the response returned an authentication error due to the absence of a valid token. While no sensitive data was exfiltrated, the fact that the server accepted and responded to a cross-origin request from an untrusted origin indicates a CORS misconfiguration. This proves the vulnerability exists and should be labeled as a Proof of Concept (PoC).

In a real-world scenario, if an attacker can obtain or guess a valid session token (e.g., through XSS, phishing, or a malicious extension), they could abuse the misconfigured CORS policy to access sensitive API endpoints from a malicious domain — effectively bypassing the same-origin policy.

Remediation Advice

To mitigate this vulnerability, the server's Cross-Origin Resource Sharing (CORS) policy must

be restricted. Specifically, the Access-Control-Allow-Origin header should not be set to * in production environments. Instead, it should explicitly list trusted domains, such as https://yourdomain.com, which are permitted to interact with the API. Additionally, if credentials (like cookies or tokens) are used in requests, Access-Control-Allow-Credentials should be set to true, and Access-Control-Allow-Origin must not be a wildcard. Implementing proper CORS rules ensures that only authorized frontends can interact with backend services, preventing malicious third-party origins from making unauthorized API calls.

References

https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS
https://cwe.mitre.org/data/definitions/942.html
https://blog.securelayer7.net/owasp-top-10-security-misconfiguration-5-cors-vulnerability-patch/

Contact Details

Name: Fahad Alshamri.
Email: falshamri@deakin.edu.au

Pentest Leader Feedback.

Nicholas Krcevinac:

- **Inconsistent Address Reference**: There is a contradiction between the documentation and the visual evidence. In your write-up, you state that the malicious HTML file sends requests to http://172.18.0.1:4200, yet in the screenshot provided, the exploit is executed from http://localhost:9999/fahad.html. This mismatch can confuse the OnTrack Team. Please update the screenshot or the written steps to match your actual testing environment and ensure consistency across your evidence and description.
- **Text Color Formatting**: Avoid using blue text for headers or body text in formal documentation. It affects readability and professionalism. Stick to default formatting or use black text for a cleaner, consistent, and professional appearance. I would recommend to look in the "4.Ready for Final Report" for what we are looking for in documentation and follow the AppAttack Findings Checklist.docx.

  Also, if you do this is future reports there is a high chance that we will fail you straight away.

- **Proof of Concept Clarity**: While your script executes correctly and reaches the intended endpoint, the response returned an **authentication error**. This means sensitive data was not exfiltrated, but the server did respond — indicating a **CORS misconfiguration**. Given this, it would be more accurate to label your finding as **proof of concept (PoC)**. Please make this clear in the description to avoid overrepresenting the exploit's impact and in your evidence.
- **Additional Suggestions**:
  - You could enhance your finding by checking if any endpoints do return sensitive data when a valid token is present.
  - Include a clearer conclusion section or recommendation summary for better readability.