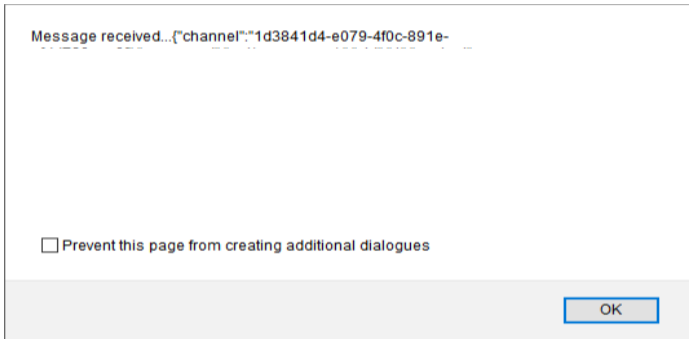


```
<script>
   function WebSocketTest() {
      var ws = new WebSocket("ws://website.com");
      ws.onopen = function() {
         ws.send("param1=example&param2=example");
      };

      ws.onmessage = function (evt) {
      var received_msg = evt.data;
         alert("Message received:" + received_msg);
      };

   }
</script>
```

The result of the code above returned personal information about the user. The fix for the company was to block any outside connections to the websocket server and in turn the issue was fixed. Another approach to fix this issue could have been to introduce CSRF/session handling on the requests.

- **Signing up using @company.com email**

When claiming ownership of a page I noticed that when creating an account if I set my email to zseano@company.com then I was whitelisted for identification and could simply bypass it. No email verification was done and I could become admin on as many pages as I wanted. Simple yet big impact! You can read the full writeup here: https://medium.com/@zseano/how-signing-up-for-an-account-with-an-company-com-email-can-have-unexpected-results-7f1b700976f5

Another example of creating impact with bugs like this is from researcher @securinti and his support ticket trick detailed here: https://medium.com/intigriti/how-i-hacked-hundreds-of-companies-through-their-helpdesk-b7680ddc2d4c

- **“false” is the new “true”**

Again extremely simple but I noticed when claiming ownership of a page each user could have a different role, Admin & Moderator. Only the admin had access to modify another user’s details and this was defined by a variable, “canEdit”:”false”. Wanting to test if this was working as intended I tried to modify the admin’s details and to my