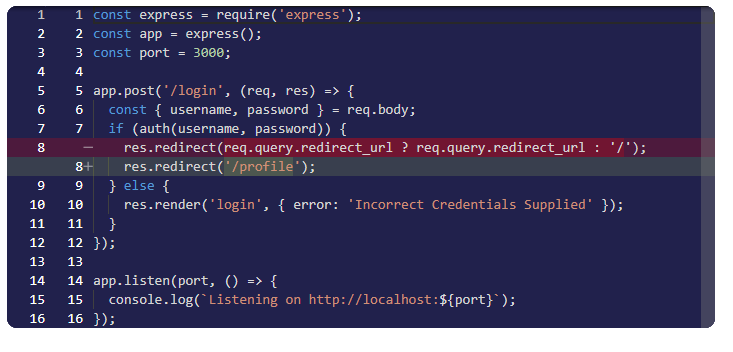
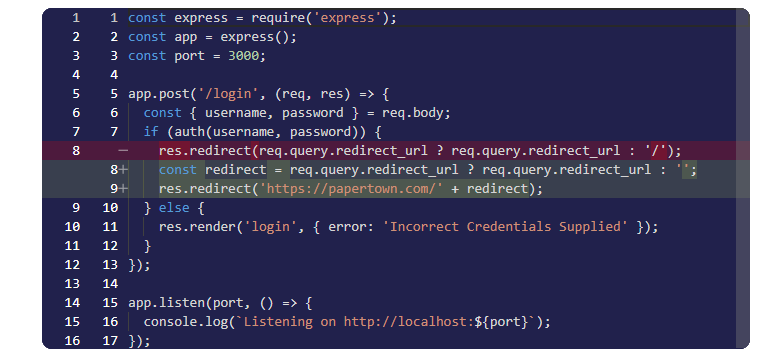
**Open Redirect** – redirection which can be controlled by client-side app. Hackers abuse it to redirect to their phishing pages.

Potential vulnerability:

  
Solution to fix with allow list:

  
Solution to fix with a fixed domain:



Code injection - injecting malicious code into an application through a user input field

eval function below is used as a concatenation (eval is dangerous)



Simple solution: const uppercaseName = name.toUpperCase()

Because eval is dynamically taking values, it is dangerous

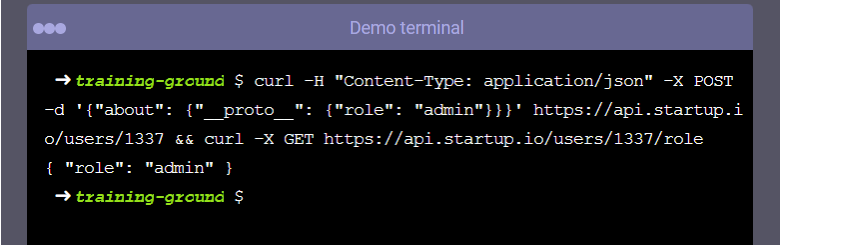
curl https://bigcorp.com/customerOnboarding?name=";process.version//

The code above returns node js version

Cross-site scripting XSS - very similar to code injection. But in code injection we were using query parms. Here we more use HTML elements. Example: `<p>${chatMessage}</p>` To **chatMessage** we can inject something like <scipt> console.log(document.cookie) </scipt>

All browsers use **same-origin policy** it means that cookies and session storage is same for 1 domain and not shared with other domains. Using XSS we can inject a script and get cookie or session data of other users in that particular domain.

Prototype Pollution - attacker can control default properties of an Object, using \_\_proto\_\_



In the following example we were able to upgrade our user role using \_\_proto\_\_

**Built-methods** How they work? In Java and JavaScript for example there is a default method toString() for strings. However, its execution works differently for each of programming languages.  
In Java it works through inheritance it looks to its current Class and tries to find toString() methods, if not found it goes up to its parent class and so on. In JavaScript it works similarly but with a property called \_\_proto\_\_ It could be like \_\_proto\_\_.\_\_proto\_\_.\_\_proto\_\_ until method is found.



