**Regular Expression Denial of Service (ReDOS)**

The unpatched version of our application hosted in localhost:7000 is vulnerable to ReDOS attacks since the part of our code that checks the validity of the username and password uses Regular Expressions, as seen from the image below:

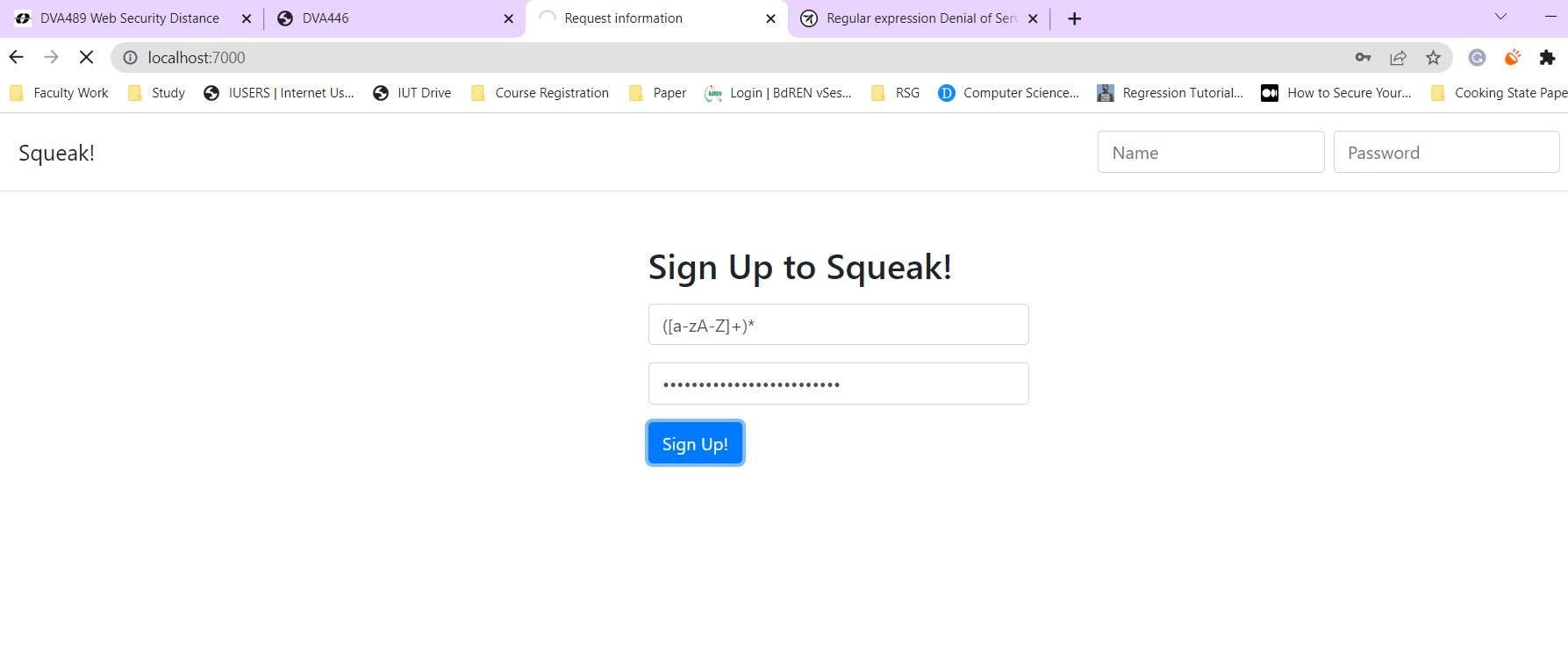
  if (signuppassword.match(new RegExp(signupusername)) != null) {

    console.log(signuppassword.match(new RegExp(signupusername)));

    flag = true;

  }

Upon entering an “evil Regex” like “([a-zA-Z]+)\*” in the username and a password like “aaaaaaaaaaaaaaaaaaaaaaaa!”, our server seems to become non-responsive.



**Protecting the application**

The patched version of our application is hosted in localhost:8000 and this version is secure from ReDOS attacks. In order to protect our application, we implement our own input validator which checks whether the input contains special characters like “^”, “\*”, “+”, etc.

const checkChar = (text,c) => {

  for (i=0; i<text.length; i++){

    if(text[i] == c){

      return true

    }

    return false

  }

}

const checkRegex = (text)=>{

  if (checkChar(text,'^') || checkChar(text,'+') || checkChar(text,'$') || checkChar(text,'\*') || checkChar(text,'[') || checkChar(text,']')){

    return false

  }

  return true

}

Then, when signing up a new user inside our postSignUp method, we check the validity of the username and password as follows:

if(checkRegex(signupusername) && checkRegex(signuppassword)){

    if (signuppassword.search(signupusername) != -1) {

      flag = true;

    }

}

else{

    flag = true

}