WEBSITE SECURITY RESEARCH

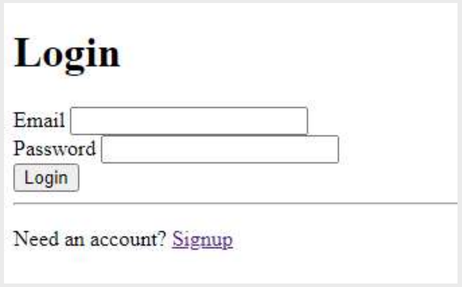
OSU Online Computer Science Capstone Project

https://github.com/PatrickDougan/Website-Security-Research-Project

Patick Dougan
Reed Hardin
Isfandyar Rabbani
Michael Volz

DESCRIPTION

- Website security is an increasingly important concern among both developers and users of online services. Today, online applications span services that include online banking, retail establishments, government entities, educational institutions, and even network infrastructure.
- Each of these can contain sensitive data, private information, or (potentially) government secrets. Ensuring that a user's data is secure and will not be compromised has become paramount for any organization operating online. The importance of web security is increasing rapidly, and the consequences of failure can be enormous.
- Some industry experts estimate that anywhere from 30,000 to 50,000 websites are compromised by hackers each and every day, with that number growing continuously.
- Our efforts focus on learning and understanding specific vulnerabilities and threads to web site application servers.
- Our "user-focused" deliverables include a Github repository with documentation and detailed descriptions and mitigation strategies for each investigated attack.
- The ultimate goal here is for users and a developers to be more informed and thoughtful about what *could* go wrong, and how to avoid those pitfalls.

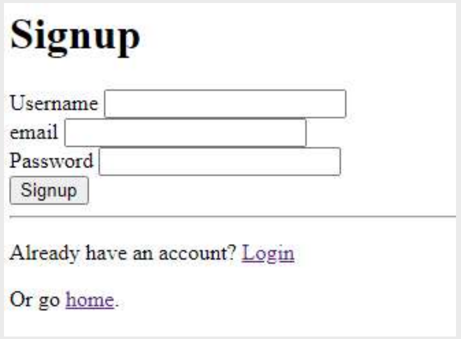


Login

Email	
Password	

Login

Need an account? Signup



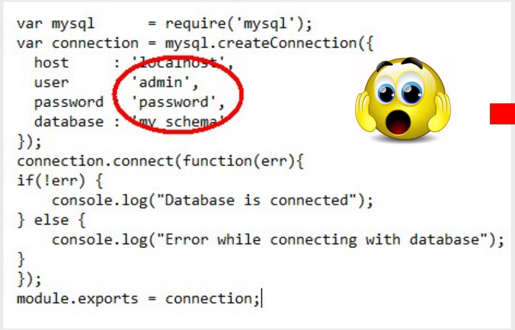
Signup

Username	
email	
Password	

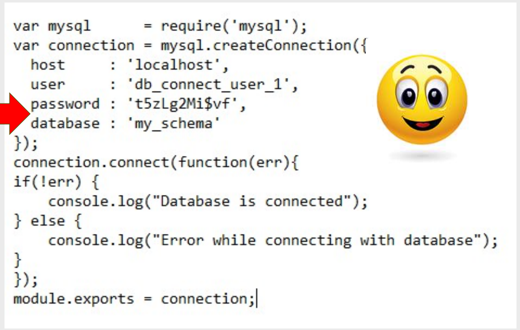
Signup

Already have an account? Login

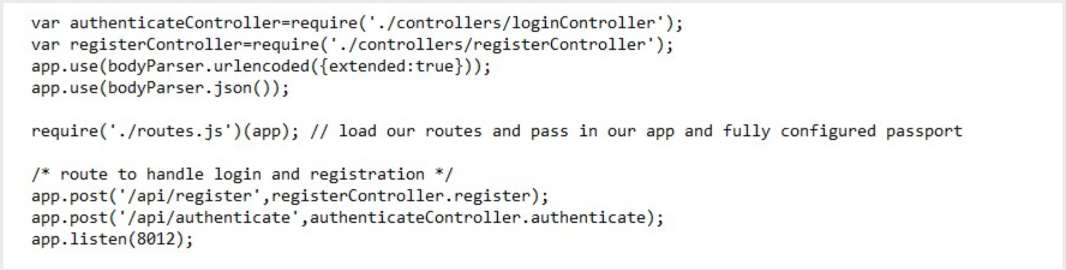
Or go home.



```
var mysql       = require('mysql');
var connection = mysql.createConnection({
  host     : 'localhost',
  user     : 'admin',
  password : 'password',
  database : 'my_schema'
});
connection.connect(function(err){
if(!err) {
    console.log("Database is connected");
} else {
    console.log("Error while connecting with database");
}
});
module.exports = connection;
```



```
var mysql       = require('mysql');
var connection = mysql.createConnection({
  host     : 'localhost',
  user     : 'db_connect_user_1',
  password : 't5zLg2Mi$vf',
  database : 'my_schema'
});
connection.connect(function(err){
if(!err) {
    console.log("Database is connected");
} else {
    console.log("Error while connecting with database");
}
});
module.exports = connection;
```



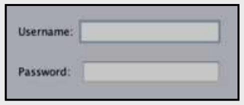
```
var authenticateController=require('./controllers/loginController');
var registerController=require('./controllers/registerController');
app.use(bodyParser.urlencoded({extended:true}));
app.use(bodyParser.json());

require('./routes.js')(app); // load our routes and pass in our app and fully configured passport

/* route to handle login and registration */
app.post('/api/register',registerController.register);
app.post('/api/authenticate',authenticateController.authenticate);
app.listen(8012);
```

ATTACKS

SQL Injection
Broken Authentication
Sensitive Data Exposure
Broken Access Control
Security Misconfiguration
Cross-Site Scripting
Components with Known Vulnerabilities

STRETCH GOAL

Password Cracking



EACH ATTACK IS DIFFERENT – EACH ATTACK IS DANGEROUS!

- **SQL Injection** is a technique of attacking data-driven applications; using malicious SQL commands, attack is usually an effort to either extract data that should not be public, or to alter / add malicious data.
- **Broken Authentication** attacks are attempts to compromise a key, a session token, or possibly passwords in order to assume (or "hijack") the identity of a valid system user
- **Sensitive Data** exposure occurs with information that is intended to be kept private (or secret) is unintentionally exposed either publicly, or to users who should not have access to the data in question.
- **Broken Access Control** can allow attackers to access systems and data that should otherwise be protected and not accessible.
- **Security Misconfiguration** occurs when an application is not kept up-to-date, or when a default account unintentionally remains active, or when credentials are easily guessed.
- **Cross-Site Scripting** is when an attacker uses an application to send untrusted data to a compute device, which then executes the data as a script.
- **Components with Known Vulnerabilities** can undermine a higher-level application's security by creating an opening for an attack within the "foundation" of an application.
- **Password Cracking** is an effort to create a long list of known words or other insecure passwords, and then encode them into a parallel list with a regularly-used encryption algorithm. If an encoded string matches, then the clear-text password can be retrieved.