Web application security

Week2

Timo Lehosvuo, M3426
TTV18S1

Harjoitustyö
Web application security, Heikki Salo, Joni Ahonen
1.3.2021
Tekniikan ala

1. Reading Report

Why is finding SQL injection vulnerabilities fun:

- Finding SQL injections is fun when you get bug bounty rewards.
- Finding SQL injection vulnerabilities is fun since it can help you to get a job since you have something concrete to show to your might be future boss.
- Finding SQL injections make you a better coder, hacker, and overall a better IT expert
- “Breaking” stuff and finding weaknesses can be fun

Blind SQLi:

- Blind SQLi is an SQL injection where you can’t get the query’s direct output.

2. Issue Report (Login Jim)

2.1 SQL injection makes it possible to login without proper password

Description: The code does not sanitize input fields properly and thus making SQL injection possible.

Steps to produce:

- Go to juice shop front page
- Search for “Green smoothie” product and get the email address from the review
- On the top right corner click account -> login
- Type Jims email to the email section and add symbols “‘--" to it
- Type whatever password you want and click login
- You should now be logged in as Jim

Mitigation:

- Modify to the login.js code so that it checks the user input fields.
- See: https://cheatsheetseries.owasp.org/cheat-sheets/SQL_Injection_Prevention_Cheat_Sheet.html

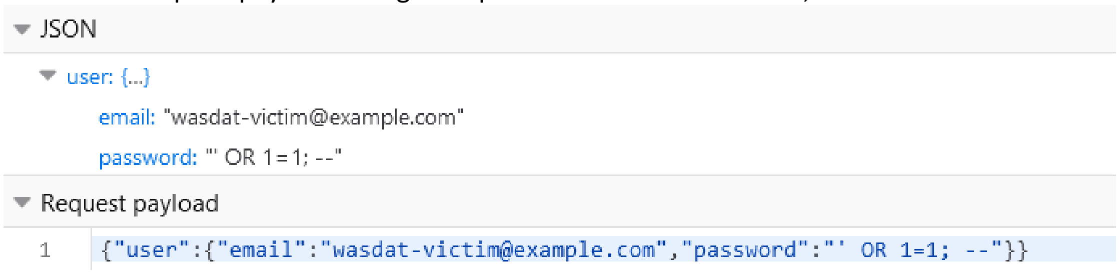
3. Issue Report (Wasdat)

3.1 SQL injection to password field isn’t sanitized

Description: Modifying the HTTP POST message allows the user to inject the pass-word field with SQL command. The backend thinks that every password is hashed but it does not check it.

Steps to produce:

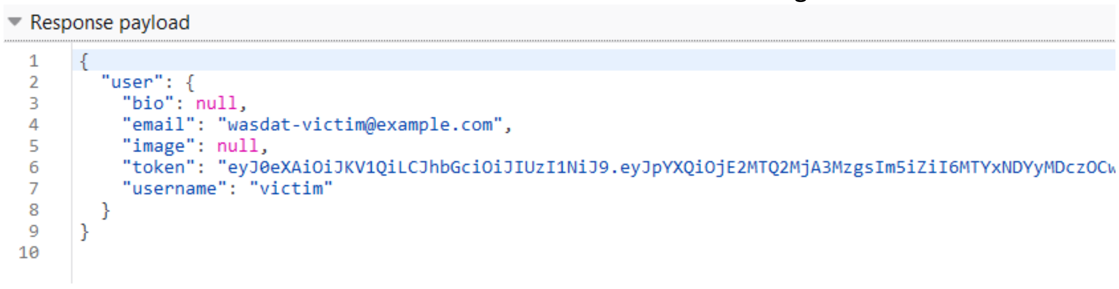
- Register a new user to wasdat.
- Open the “Inspect element” menu and go to the network tab.
- Login using the users email account but leave the password field empty.
- Search for your login attempt and right click it and click “Edit and Resend”.
- On the request payload change the password field to “’ OR 1=1; --”.



```
email: "wasdat-victim@example.com"
password: "' OR 1=1; --"
```

```
{"user":{"email":"wasdat-victim@example.com","password":"' OR 1=1; --"}}
```

- You should get a POST method with a 200 status code.
- You can find the username and the JWT token from the message



```
{
  "user": {
    "bio": null,
    "email": "wasdat-victim@example.com",
    "image": null,
    "token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpYXQiOjE2MTQ2MjA3MzgsIm5iZiI6MTYxNDYyMDczOCw
    "username": "victim"
  }
}
```

Mitigation:

- Modify the frontend to not sanitize the inputs and instead make the backend do it since frontend can be “skipped” but backend can’t.