

multiple different queries. If your condition reaches an `UPDATE` or `DELETE` statement, for example, it can result in an accidental loss of data.



APPRENTICE

SQL injection vulnerability in WHERE clause allowing retrieval of hidden data →

✓ Solved

Subverting application logic

Imagine an application that lets users log in with a username and password. If a user submits the username `wiener` and the password `bluecheese`, the application checks the credentials by performing the following SQL query:

```
SELECT * FROM users WHERE username = 'wiener' AND password = 'bluecheese'
```

If the query returns the details of a user, then the login is successful. Otherwise, it is rejected.

In this case, an attacker can log in as any user without the need for a password. They can do this using the SQL comment sequence `--` to remove the password check from the `WHERE` clause of the query. For example, submitting the username `administrator>--` and a blank password results in the following query:

```
SELECT * FROM users WHERE username = 'administrator'--' AND password = ''
```

This query returns the user whose `username` is `administrator` and successfully logs the attacker in as that user.



APPRENTICE

SQL injection vulnerability allowing login bypass →

✓ Solved

Retrieving data from other database tables

In cases where the application responds with the results of a SQL query, an attacker can use a SQL



SQL injection tehtävät:

SQL injection vulnerability in WHERE clause allowing retrieval of hidden data:

Tehtävä oli mielenkiintoinen koska siinä suoraan syöttämällä URL linkin jatkaksi `+OR+1=1`—koodin sai tietokannassa kyselyn `SELECT * FROM products WHERE category = 'Gifts' AND released = 1` aikaiseksi. Tämä paljasti käyttäjälle piilotettua sisältöä palvelimelta. Olin kuin SQL injektioita tehdään periaatteessa ja kuinka niitä voi tehdä myös URL kenttään syöttämällä. Olin joskus aiemmin törmänyt credentials kenttään syötettyihin arvoihin SQL injektioissa.

Tehtävä oli haastava ilman apua, koska minulla ei ollut aiempaa kokemusta SQL injektioiden tekemisestä. Käytin ohjetta avuksi, mutta en tarvinnut ohjevideota avuksi.

SQL injection vulnerability allowing login bypass:

Tehtävässä opin Kuinka tietyn komennon syöttämällä login credentialseihin on mahdollista saada tietokannassa kysely, joka ohittaa salasanan varmistuksen. Kun tehtävässä syötti `'''` merkin niin se saa aikaiseksi selaimella SQL kyselyn. Kun antoi usernameksi "administrator" niin sai admin

oikeudet, joilla voi esim. poistaa käyttäjiä. Kun laittoi "--" kyselyn loppuun niin se ohittaa salasanan tarkistuksen. Opin siis tehtävässä paljon tuon haavoittuvuuden moniulotteisuudesta ja logiikasta.

Tehtävä sujui toisena SQL injektiona ihan hyvin ja turvautuin vain ohjeen apuun. Minun ei tarvinnut katsoa apuvideoita.

not visible on the rendered page.

- **Response times:** If most of the requests were handled with a similar response time, any that deviate from this suggest that something different was happening behind the scenes. This is another indication that the guessed username might be correct. For example, a website might only check whether the password is correct if the username is valid. This extra step might cause a slight increase in the response time. This may be subtle, but an attacker can make this delay more obvious by entering an excessively long password that the website takes noticeably longer to handle.

LAB	APPRENTICE	Username enumeration via different responses →	Solved
LAB	PRACTITIONER	Username enumeration via subtly different responses →	Solved
LAB	PRACTITIONER	Username enumeration via response timing →	Not solved

Flawed brute-force protection

It is highly likely that a brute-force attack will involve many failed guesses before the attacker successfully compromises an account. Logically, brute-force protection revolves around trying to make it as tricky as possible to automate the process and slow down the rate at which an attacker can attempt logins. The two

Passwd based login tehtävät:

Username enumeration via different responses:

Opin käytämään paljon kurssin työkaluja kuten Burpia. Tämä tehtävä oli ensimmäinen "hakkerointi" minulle, joten se oli hyvin mielenkiintoista. Terhi brute force on minulle hyvin tuttu, mutta nyt oli hauskaa päästä käytämään ihan työkalua sitä varten. Opin tehtävässä tosiaan Burp käyttöä, proxyn käyttöä, http pyyntöjen tarkastelua Burpin avulla vs. devtools ja sniper attackin toteuttamisen Burpin avulla. Myös periaate ensin username ja sitten salasana selkeni tehtävässä hyvin eri pyyntöjen koodien takia. Tehtävässä tärkeintä oli huomata kuinka oikean usernamen sattuessa kohdalle, virheilmoitus muuttui merkittävästi.

Tehtävässä haastavinta oli siihen pystyminen täysin itse, vaan seurasin opettajan luentotallennetta, kun tein kyseisen tehtävän alusta loppuun.

Username enumeration via subtly different responses

Opin taas lisää burpin käyttöä ja brute force tekniikkaa. Tehtävässä oli hienoa huomata, kuinka oikean usernamen sattuessa virheilmoituksessa oli vain hyvin pieni virhe "Invalid username or password." vs.

"Invalid username or password", jossa pisteen puuttuminen paljasti usernamen olevan oikea. Pyynnöissä ei ollut muuten eroa niiden numeroinnissa.

Tehtävä oli haastava tehdä, koska minulla ei ole aiempaa kokemusta penetraatiotestauksesta. Jouduin turvautumaan apuvideoihin tehtävää tehdessäni. Pidin kovasti toisesta hakkeroinnistani.

Access control vulnerabilities

△ LAB	APPRENTICE Unprotected admin functionality →	Solved
△ LAB	APPRENTICE Unprotected admin functionality with unpredictable URL →	Not solved
△ LAB	APPRENTICE User role controlled by request parameter →	Not solved
△ LAB	APPRENTICE User role can be modified in user profile →	Solved

Access control vulnerabilities tehtävät:

User role can be modified in user profile: Opin kuinka käyttäjän roolia voi muuttaa, kun on lähetetty post sähköpostiosoitteen tallentamisesta. Samanlaisella, mutta uudella post pyynnöllä kykeni muuttamaan Wiener:peter käyttäjän roleid 1 to roleid 2 =adminiksi ja kykenin poistamaan Carloksen.

Haastavinta oli keksiä miten se email pyynnön avulla saa ujutettua myös roolin muutoksen. Tarvitsin ohjevideon apua. Myös jouduin käynnistämään kaikki prosessini ja tietokoneeni uudestaan, koska en kyennyt kirjautumaan enää wiener:peter tunnuksilla, vaan hakkasin niissä päätäni seinään pitkän aikaa.

Unprotected admin functionality:

Opin että, kirjoittamalla domain kenttään tiettyjä parametereja, kuten admin-panel, on mahdollista päästää kirjautumatta muuttamaan käyttäjätietoja palvelusta. Myös robots.txt tekstitiedoston sisältöä pääsi tarkastelemaan kirjautumatta palveluun. Sieltä näki, että polku admin paneeliin loppuu:

User-agent: *

Disallow: /administrator-panel.

Tehtävä ei ollut kovin haastava vaan onnistuin tekemään sen ilman apuvideota. Tehtävässä ei tarvinnut tarkastella ollenkaan koodia.

Jatkan PortSwigger tehtävien tekemistä myöhemmin korottaakseni arvosanaani. 😊