

Week 7 task 7.2D**Question 1**

1. What does SQL stand for? How is it different compared with a DBMS?
 - As the name implies, SQL, often referred to as Standard Query Language, is a query language, database management systems, or DBMSs, are used to administer databases. SQL is helpful for storing, modifying, and obtaining data from databases.
2. In your own words, define what is an SQL injection attack and what vulnerabilities allow an SQL injection attack to occur?
 - An example of a cyberattack is a SQL injection attack, which modifies application queries submitted to a database by using software vulnerabilities in those applications. By interfering in this way, an attacker could be able to bypass authentication requirements, access, alter, or remove undesired data, or even seize administrative control of the database. In order to alter the SQL query that is submitted to the database, a SQL injection attack entails inserting malicious SQL code into a field or changing the URL. To mislead the database into allowing unauthorized access to the application's contents, for example, an attacker could insert SQL commands into a poorly designed login form.

Deficiencies That Permit SQL Injection Attacks

- Inadequate Output Encoding: SQL commands may be interpreted instead of being regarded as data if output is not encoded correctly. Attackers may use carelessness to insert malicious SQL statement.
- Inadequate access Controls: An SQL injection's impact might be made worse by improperly configured database rights. An SQL injection could potentially have far more of an impact if the application's database user account has a lot of privileges.
- Disclosure of Error Messages: In-depth error messages may divulge sensitive information such as database schemas, which attackers may utilize to home their SQL injection attempts.
- Absence of Input validation: User inputs may be maliciously entered into SQL queries by applications that do not properly validate user inputs. This gives hackers the ability to introduce SQL code and change the intended behavior of the query.
- Vulnerabilities may arise when SQL queries are created dynamically by directly utilizing user input. Attackers can readily utilize dynamic SQL to change the query structure when it concatenates strings without parameterization.
- Outdated Code and Subpar Development methods Applications with badly designed or antiquated code frequently don't follow current security procedures. Because these systems rely on outdated libraries or have inadequate security safeguards, they may be especially susceptible to SQL injection.

3. What are some of the recent attacks that have been initiated by SQL injection? How were they conducted?
 - The online resume service platform ResumeLooters.com was the victim of the ResumeLooters SQL Injection attack, which was first made public in June of 2024. SQL Injection (SQLi) is a widely used vulnerability in which an attacker can manipulate the queries a program submits to its database. The intruders discovered that a few of the ResumeLooters website's input fields were open to SQL Injection attacks. Sending carefully constructed input through forms, URL parameters, or other data entry points that are directly utilized in SQL queries is usually required for this. In this instance, the user login form and the job search capability were discovered to be vulnerable.
The attackers manipulated input that contained SQL code in order to take advantage of the weakness. Using a tautology-based attack, for instance, is a popular strategy where the input may be something like "'OR'1='1'." It circumvents authentication when used to a query such as "SELECT * FROM users WHERE username = '\$input'" since it modifies the logic to always return true.
 - The attackers circumvented authentication procedures and obtained unauthorized access to the database by using SQL Injection. They might be able to obtain private data, including administrator credentials, resumes, and user information. Attackers can extract enormous amounts of data with this degree of access, which is frequently referred to as a "dump" of the database.
The attackers stole information from the database when they gained access. This entails executing extra SQL queries to obtain particular tables and columns that hold confidential data. As an illustration, "SELECT * FROM users;"
These kind of searches would provide all user data kept in the database, including submitted resumes, login credentials, and personal data.
 - Attackers usually take precautions to reduce their visibility. They may tidy logs or make use of sophisticated SQL strategies to avoid having their harmful queries discovered. They might have used anonymous channels and disguised their searches in this instance to prevent being traced. The personal and professional information of thousands of people was exposed as a result of the ResumeLooters SQL Injection attack. The organization promptly addressed the vulnerability by introducing prepared statements and parameterized queries that are impervious to SQL injection attacks. They also carried out a thorough security examination in order to find and address any further potential weaknesses. It is important to secure web applications against common vulnerabilities, as demonstrated by the ResumeLooters SQL Injection attack. Organizations can guard against these types of attacks by following secure coding techniques and routinely upgrading security protocols.

4. Can a firewall prevent an SQL Injection attack? Briefly discuss and support your answer.

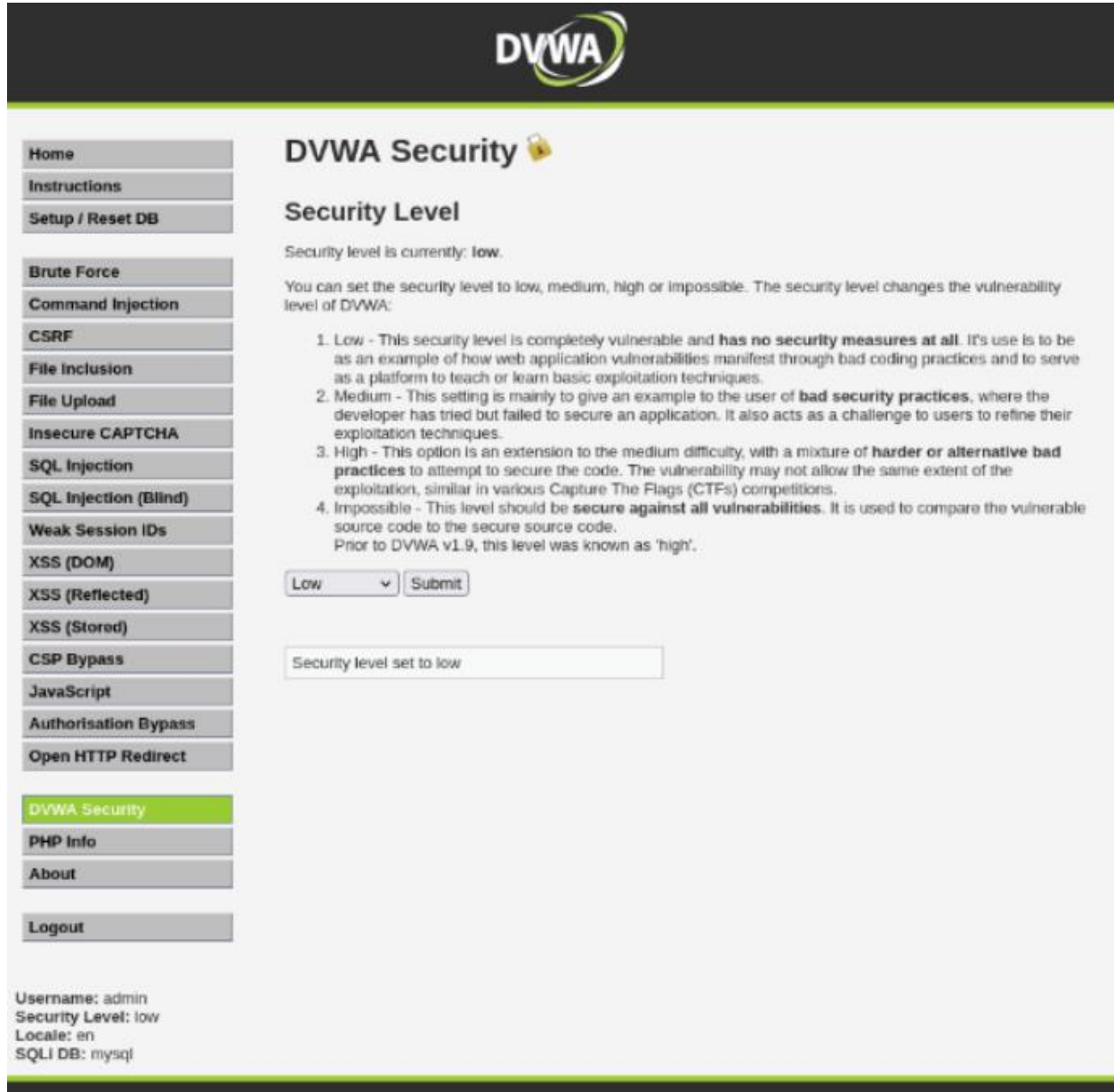
- While it can support a defense-in-depth approach, a firewall cannot completely prevent SQL Injection assaults.
The main purpose of firewalls, especially classic network firewalls, is to regulate traffic across trusted and untrusted networks by using pre-established security rules. They can filter traffic by IP addresses, port numbers, and protocols and function at the network and transport layers (OSI layers 3 and 4). They do not, however, check the payload of HTTP requests to look for SQL Injection attempts, which take place at the OSI layer 7 application layer.
- An online Application Firewall (WAF) keeps an eye on and filters HTTP traffic going to and from online applications, offering more pertinent protection against SQL Injection. WAFs examine the payload of HTTP requests for questionable inputs and behaviors, allowing them to identify and prevent common SQL Injection patterns and anomalies in online traffic. For example, a WAF may recognize and stop queries that contain SQL metacharacters typically used in injection attacks, such as '--', ';', or 'DROP'. WAFs are not infallible, despite the fact that they can reduce some dangers. Attackers might create complex payloads that evade WAF filters or take advantage of vulnerabilities that are not yet patched. For complete security, therefore, depending only on a firewall (network or WAF) is inadequate.

For SQL Injection to be effectively prevented, a layered security strategy that consists of;

- ✓ Practice Secure Coding: To lessen vulnerabilities in the application code, use secure coding guidelines.
- ✓ Check that all user inputs have been verified and cleaned up before the database processes them. By doing this, harmful inputs may not be interpreted as SQL instructions.
- ✓ Frequent Security Testing: To find and fix possible flaws, perform frequent penetration tests, vulnerability assessments, and code reviews.
- ✓ Queries with parameters: To reduce the danger of injection, employ parameterized queries and prepared statements to keep SQL code and user input separate.

Question 2

1. Go to the SQL Injection tab on DVWA and show a successful SQL injection attack. Include screenshots or link to a screencast confirming that you have successfully conducted an SQL injection attack.
- Turn down the DVWA's security level to



The screenshot shows the DVWA (Damn Vulnerable Web Application) interface. The top navigation bar includes links for Home, Instructions, and Setup / Reset DB. The main content area is titled "DVWA Security" and displays the "Security Level" section. The current security level is "Low". A list of four security levels is provided: 1. Low (completely vulnerable), 2. Medium (bad security practices), 3. High (harder or alternative bad practices), and 4. Impossible (secure against all vulnerabilities). A dropdown menu is set to "Low" and a "Submit" button is visible. Below the dropdown, a message states "Security level set to low". The left sidebar contains a list of vulnerability categories, with "SQL Injection" highlighted. At the bottom, the user's session information is displayed: Username: admin, Security Level: low, Locale: en, and SQLI DB: mysql.

DVWA Security

Security Level

Security level is currently: **low**.

You can set the security level to low, medium, high or impossible. The security level changes the vulnerability level of DVWA:

1. **Low** - This security level is completely vulnerable and **has no security measures at all**. It's use is to be as an example of how web application vulnerabilities manifest through bad coding practices and to serve as a platform to teach or learn basic exploitation techniques.
2. **Medium** - This setting is mainly to give an example to the user of **bad security practices**, where the developer has tried but failed to secure an application. It also acts as a challenge to users to refine their exploitation techniques.
3. **High** - This option is an extension to the medium difficulty, with a mixture of **harder or alternative bad practices** to attempt to secure the code. The vulnerability may not allow the same extent of the exploitation, similar in various Capture The Flags (CTFs) competitions.
4. **Impossible** - This level should be **secure against all vulnerabilities**. It is used to compare the vulnerable source code to the secure source code.
Prior to DVWA v1.9, this level was known as 'high'.

Low Submit

Security level set to low

DVWA Security

PHP Info

About

Logout

Username: admin
Security Level: low
Locale: en
SQLI DB: mysql

- Navigate to the SQLi tab within DVWA.

The screenshot displays the DVWA web application interface. At the top, the DVWA logo is visible. On the left side, there is a vertical menu with various security challenges. The 'SQL Injection' tab is currently selected and highlighted in green. The main content area is titled 'Vulnerability: SQL Injection'. Below the title, there is a form with a 'User ID:' label, an input field, and a 'Submit' button. Underneath the form, a section titled 'More Information' lists four links to external resources about SQL injection. At the bottom left, system information is displayed, including the username 'admin', security level 'impossible', locale 'en', and SQLi DB 'mysql'. At the bottom right, there are buttons for 'View Source' and 'View Help'.

DVWA

Home
Instructions
Setup / Reset DB

Brute Force
Command Injection
CSRF
File Inclusion
File Upload
Insecure CAPTCHA
SQL Injection
SQL Injection (Blind)
Weak Session IDs
XSS (DOM)
XSS (Reflected)
XSS (Stored)
CSP Bypass
JavaScript
Authorisation Bypass
Open HTTP Redirect

DVWA Security
PHP Info
About

Logout

Vulnerability: SQL Injection

User ID:

More Information

- https://en.wikipedia.org/wiki/SQL_injection
- <https://www.netsparker.com/blog/web-security/sql-injection-cheat-sheet/>
- https://owasp.org/www-community/attacks/SQL_injection
- <https://bobby-tables.com/>

Username: admin
Security Level: impossible
Locale: en
SQLi DB: mysql

- Because its condition is set to true, the traditional SQLi statement `1' or '1' = 1"` always returns a result or group of values.

Vulnerability: SQL Injection

User ID:

More Information

- https://en.wikipedia.org/wiki/SQL_injection
- <https://www.netsparker.com/blog/web-security/sql-injection-cheat-sheet/>
- https://owasp.org/www-community/attacks/SQL_injection
- <https://bobby-tables.com/>

- Output

Vulnerability: SQL Injection

User ID:

ID: 1' or '1' = '1
First name: admin
Surname: admin

ID: 1' or '1' = '1
First name: Gordon
Surname: Brown

ID: 1' or '1' = '1
First name: Hack
Surname: Me

ID: 1' or '1' = '1
First name: Pablo
Surname: Picasso

ID: 1' or '1' = '1
First name: Bob
Surname: Smith

Question 3

1. What does CSRF stand for? How does this attack work?
 - One kind of security flaw in online application is called Cross-Site Request Forgery, or CSRF. When a user is authenticated into a web application, an attacker can device them into executing undesired actions.
 - How the CSRF Attack Works:
 - ✓ Execution of the action, the crafted request appears genuine and is executed using the victim's credentials when it is delivered to the target application since the victim's browser still contains the authentication credentials for the target application. Taken action, without cheaking to see if the user truly intended to do the action, the web application takes action since it thinks the request is from the verified user. User Authentication, when a victim logs in and has an active session with a target online application, they are considered authenticated within the application. Usually session cookies or tokens are used for this authentication. The act of creating a malicious website or a specially crafted URL with the intention of carrying action on the target application, such as modifying user settings, making a purchase, or starting a financial transfer, is known as malicious request creation. Victim, when the victim clicks on the specially designed URL, they unintentionally engage with the malicious website. Social media, email links, and fraudulent advertisement can all cause this

2. How can a CSRF attack be prevented?

- Same site Cookies: By limiting the way cookies are delivered along with cross-site requests, the SameSite cookie property helps prevent CSRF attacks. The SameSite attribute has three possible values They are Lax , Strict
 - ✓ Lax, with a few exceptions, like a accessing the website from an external link, cookies are not delivered during cross-site request.
 - ✓ Strict, Cookies don't respond to requests from third-party website; instead, they are exclusively sent in first-party contexts.
- By limiting the transmission of cookies in conjunction with cross-site requests, the SameSite attribute can be set to 'Lax' or 'Strict' lowering the possibility of cross-site request forgery.

3. Go to the CSRF tab on DVWA. Show a successful CSRF attack. Include screenshots or link to a screencast confirming that you have successfully conducted a CSRF attack.

- Go to the DVWA's CSRF section now.

Vulnerability: Cross Site Request Forgery (CSRF)

Change your admin password:

New password:

Confirm new password:

- You can verify that the current admin password is correct by opening the test credentials.

Test Credentials

Vulnerabilities/CSRF

Valid password for 'admin'

Username

Password

Vulnerability: Cross Site Request Forgery (CSRF)

Change your admin password:

New password:
●●●●●●

Confirm new password:

Password Changed.

4. What is a browser Cookie (or HTTP cookie)? What is it used for?
- While a user is on a website, a web browser stores a small bit of information on their device called a browser cookie. User's preference and settings are stored in cookies, and user data is tracked for advertising and analytics purposes.

Question 4

1. Go to the XSS reflected tab on DVWA. Type “<script>alert (document.cookie) </script>” in the textbox and click Submit. What happens? What information are you shown?

Vulnerability: Reflected Cross Site Scripting (XSS)

What's your name?

- A warning message appears. This causes the echo command to be called out because the input is under the name variable. Any malicious code can be used by the hacker, as demonstrated in this case where a Java Script code loads an alert box revealing the contents of the cookie. The user's session ID is then shown in the alert box, which the hacker may utilize for evil purposes.

2. What is the difference between CSRF and XSS

- CSRF

✓ Prevention

Counter CSRF Coins.

The attribute of SameSite Cookies.

Send in two Cookies at once.

✓ Attack Process

Using the session cookies set up by the user, the user's browser sends a request to the trusted website that looks authentic.

A malicious email, scripts, URL is created by the attacker.

After authenticating, the user engages with the harmful material.

- ✓ CSRF takes advantage of a web application's faith in the user's browser. A cross-site request forgery (CSRF) attack deceives the user's browser into sending unsolicited requests to an application while the user is authenticated and signed in. For instance, a CSRF attack might cause an unapproved money transfer if a signed into a financial website.

- XSS

- ✓ Prevention

Coding of Output.

Cookies that are HTTP only.

Third, the Content Security Policy.

Verification of Input.

- ✓ Attack Process

When a user submits a form or uses a URL, a web application reflects the malicious scripts onto their browser.

This type of attack operates by altering the DOM of the webpages, bypassing the server.

Users are served the malicious script that is kept on the server.

- ✓ Through XSS, attackers can insert harmful code onto other user's websites. These scripts have the ability to reroute users, change the DOM, and steal data while operating within the user's browser.