Task 6.2D

Name of Student

Institutional Affiliation

Question 1 - SQL Injection

1. What does SQL stand for? How is it different compared with a DBMS?

The technology for building and maintaining databases is known as database management software (DBMS). It provides users with a systematic user interface for creating, retrieving, and managing data. SQL is a language that is used to access DBMS and databases. DBMS does not allow distributed properties, whereas SQL does, and DBMS does not support normalization, whereas SQL does. SQL programs can be used to explore the data in your servers. Despite the fact that SQL is a smaller part of the overall DBMS, it is not fully engulfed by it. There are some functions that are only present in SQL software and cannot be done with any other tool in a DBMS.

2. In your own words, define what is an SQL injection attack and what vulnerabilities allow an SQL injection attack to occur?

SQL Injection (SQLi) is a type of injection attack that allows malicious SQL statements to be executed. These claims manage a database server that is hidden behind a web application. SQL Injection bugs may be used by attackers to circumvent device protection steps. They will bypass authentication and authorization of a web page or web application and retrieve the entire SQL database's content. SQL Injection can also be used to add, alter, and erase records in the database.

Any website or web program that uses a SQL database, such as MySQL, Oracle, SQL Server, or others, can be vulnerable to SQL Injection. Criminals can exploit it to obtain unauthorized access to your confidential data, including consumer information, personal information, trade secrets, intellectual property, and other information. SQL injection attacks are one of the most common, widespread, and dangerous web application vulnerabilities. In their OWASP Top 10 2017 paper, the OWASP organization (Open Web Application Security Project) ranks injections as the number one vulnerability to web application security.

An attacker must first identify insecure user inputs inside the web page or web application in order to launch a SQL Injection attack. A web page or web server with a SQL Injection flaw implicitly uses such user feedback in a SQL query. The assailant has the ability to generate input information. This type of content is often referred to as a malicious payload, and it is the most important component of the attack. Following the attacker's transmission of this content, malicious SQL commands are executed in the database.

SQL is a query language intended for managing data in relational databases. It allows you to browse, change, and delete records. SQL databases are used to store all data in many web apps and websites. SQL commands can also be used to execute operating system commands in some situations. As a result, a good SQL Injection attack may have far-reaching effects.

3. What are some of the recent attacks that have been initiated by SQL injection? How were they conducted? (Suggested word count: up to 400 words)

SQL injection attacks are simple and inexpensive to carry out, and the consequences can be devastating for the victims. Without a doubt, this approach is still common among hackers. SQL injection vulnerabilities are unavoidable and quickly abused in today's applications because they are data-driven and available through the site. Furthermore, as shared database technology has grown in importance, hackers have been able to exploit SQL injection vulnerabilities, affecting programs that share the same database. SQL injection attacks enable attackers to quickly retrieve and expose confidential data, delete database information, forge identities, change transactions, and force privilege escalation to become database server administrators.

In a recent SQL injection attack on the Flaticon website, threat actors stole emails and password hashes for 8.3 million Freepik and Flaticon users. After the data breach, Freepik has used bcrypt to hash all of their user passwords and has conducted a thorough analysis of internal and external security processes under the supervision of external security experts.

Hackers were discovered deliberately attempting to exploit SQL injection authentication flaws in the Discount Rules for WooCommerce WordPress plugin. An influx of attacks from an IP address that attempted to insert a script into the WooCommerce template hook was detected. Aside from SQL injection, the Discount Rules for WooCommerce plugin contains a number of flaws, including authorization problems and unauthenticated stored Cross-Site Scripting (XSS).

A few Stanford students registered for Connection, a forum for users and their crushes. The site was discovered to be vulnerable to SQL injection, which may have exposed the data of many users. A newspaper received an email from an unidentified person containing user details from the site, as well as an enclosed spreadsheet containing the email addresses, names, and crushes of approximately 100 people. Furthermore, the individual posted screenshots and a screen-recorded video of the suspected hack.

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

Yes. A web server firewall is one of the best practices for identifying SQL injection threats (WAF). A WAF running before the web servers watches and detects trends that are threatening to enter and exit the web servers. It is, in essence, a bridge between the internet and the web application.

A WAF operates through specified web protection rules, which can be personalized. These policies remind the WAF about the vulnerabilities and traffic behaviour. Based on this information, the WAF continues to track and block malicious traffic on the requests of both GET and POST. The importance of a WAF is partly due to the easy implementation of policy changes. In no time, new regulations can be added to allow quick rules and quick emergency response.

A firewall is imposed between the web application and the Internet by deploying a WAF in front of a Web application. As a proxy server is used as an intermediary to shield the identity of a client host, WAF is an inverse proxy that defends the server from disclosure by allowing clients to transfer the WAF prior to accessing the server.

A WAF uses a series of guidelines sometimes referred to as policies. These policies are designed to protect the program against bugs by removing malicious traffic. The WAF benefit derives partly from the speed at which policy change can be applied, so that various attack vectors can be responded to faster; rate limitation can be enforced easily during a DDoS attack by changing WAF policies.

5. 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

Screenshots

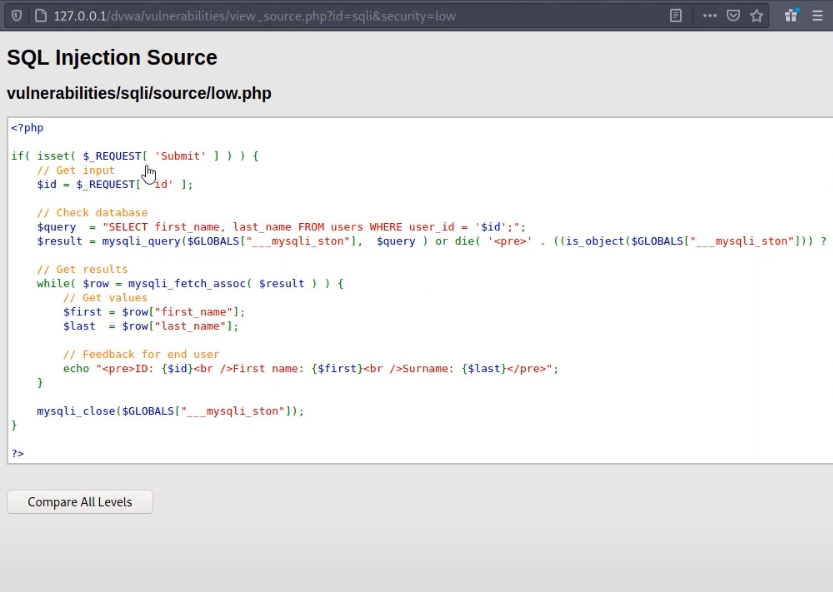


Figure 1: Source File

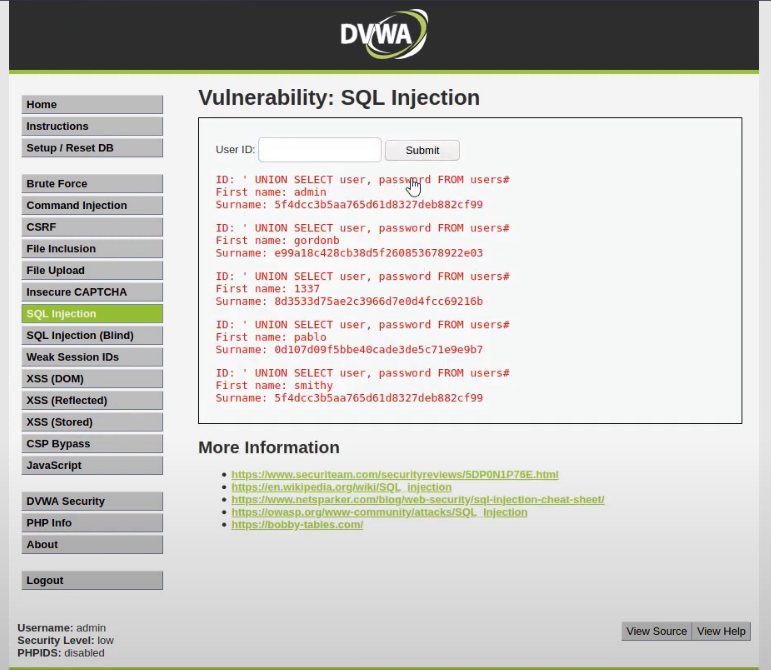


Figure 2: SQL injection using UNION SELECT

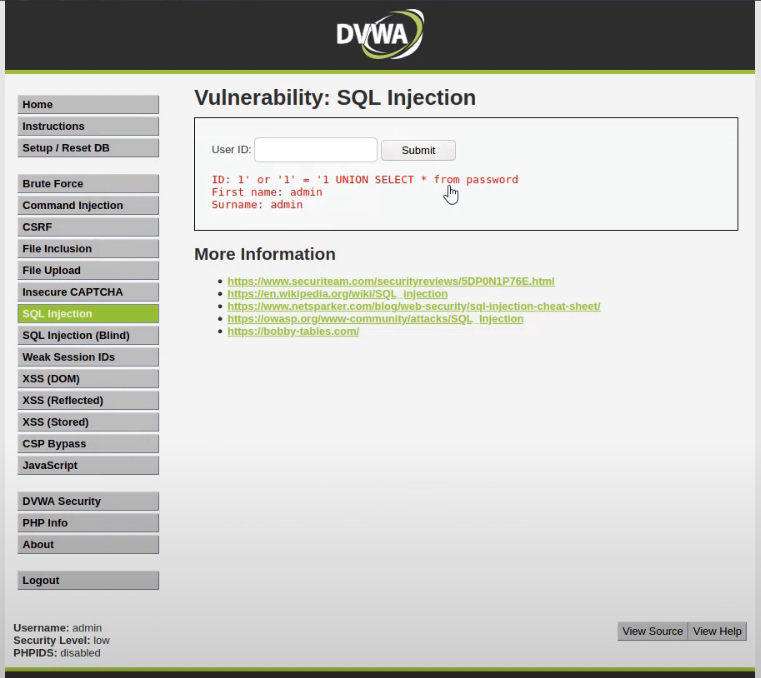


Figure 3: SL Injection Snapshot

**Question 2 - CSRF**

1. What does CSRF stand for? How does this attack work?

Cross-site Request Forgery (CSRF) is an assault on a web server where the end user is currently authenticated to perform unauthorized acts. With some social engineering support (for example sending a connection via email or a chat) a web application attacker may get users to execute the attacker's choosing behavior. If the victim is a regular user, a successful CSRF attack will cause the user to make state-changing demands, such as money transfer, e-Mail change etc. CSRF can jeopardize the whole web application because the victim is an administrative account.

The intruder forces the victim to unintendedly perform an operation in a successful CSRF assault. This could include changing the e-mail address of the accounts, changing your password, or transferring money. The perpetrator will take complete ownership of the user's account, depending on the type of the operation. The attacker may take complete control of the application's data and features if the infected user has a privileged feature inside the application.

In order to obtain data on behalf of a customer on another domain, there is no security issue if the request is not authenticated, i.e. a cookie of the session is not submitted. However, when the cookie for session of the user has been submitted, a cross-site forgery attack may be launched that abusses the relationship of trust between the browser of the victim and the web server.

CSRF attacks may have drastic implications combined with social engineering to convince users to open a malicious connection - so just how do they work? Let us begin with a summary of HTTP requests, cookies, and user sessions before discussing CSRF vulnerabilities.

2. How can a CSRF attack be prevented? [3 approaches are enough]

i) Use of Custom Request Headers

Adding CSRF tokens, a cookie with a double-submitted value, an encrypted token or other protection involving the change in your UI will also be difficult or difficult. A custom request header is an alternative security which is particularly suitable for AJAX or API endpoints. This defense relies on the SOP limit, that a custom header can only be used by JavaScript and only inside the origin. By default, JavaScript does not support cross-source requests with custom headers.

If this applies to your device, you will only be able to check the existence and value of this header on all your AJAX endpoints on your server side to shield it from CSRF attacks. This approach has the double benefit that no UI modifications are normally required and that no server-side conditions are introduced which are especially desirable for REST services. If you choose, you can still add your own custom header and value.

This strategy clearly works for AJAX calls, but <form> tags must also be protected by approaches such as tokens mentioned in this paper. CORS should also be stable to make this solution function properly (as custom headers for requests coming from other domains trigger a pre-flight CORS check).

ii) Synchronizer Token Pattern

On the server side, CSRF tokens are to be created. It can be produced for each user session or application once. The time frame for the intruder to take advantage of the stolen tokens is less secure than the session tokens. This might, though, lead to questions about usability. For instance, the "back" browser functionality is always impeded since a token that is no longer true can be found on the previous tab. This previous page would lead to a wrong, optimistic server protection case for the CSRF. After the initial token generation, the value is saved in the session per session and used until the session on any subsequent order.

When requests are made by the client, the server portion must ensure the token is correct relative to the token used in the user session. When requests are made by the client. If you did not find the token or if the given token did not fit the value of that token inside the user session, you could abort the request, end the user session and record the event in the course of the event as a possible attack by CSRF.

iii) Using Built-In or Existing CSRF Implementations

The CSRF Security Synchronizer token protections have been integrated into several architectures. It is highly advised to investigate whether a default CSRF defense can be achieved for the frame you are using before you attempt to create your personalized token scheme. For example, .NET has integrated security, which attaches a sensitive resource token to CSRF. Until using these integrated CSRF defenses that create tokens to secure vulnerable CSRF resources, you are responsible for the correct configuration (for example, key management and token management).

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.

4. What is a browser Cookie (or HTTP cookie)? What is it used for?

Cookies are tiny bits of information on your screen that are stored on websites. Cookies contain only pieces of text, no more. Text can be a user identification, session identification or some other text. Web pages may, for example, be configured - a web page can have a connection hidden from the page, which hides any feature. This settings can be saved with a cookie on your machine. The website will look at the cookie and hide it immediately when you load the page in future.

If you delete your cookies, you will not recall all of the configurations you have updated on all pages or domains. You store and handle cookies on your web browser. A list of websites that store cookies can be found and you can see the cookies in your browser settings – but it is not interesting to look at the contents of these cookies. Each browser has its own collection of cookies when you use many web browsers on your computer. Websites can only view their own cookies, for instance, we cannot check cookies from other websites while you are visiting How-To Geek. This avoids sniffing and robbing malicious websites of your login meetings.

Cookies can, however, also be used for dubious purposes. Networks that advertise and track cookies use to monitor you on the Internet. When visiting a site that uses advertisement network scripts, your tab will include a cookie. When you visit another website that uses scripts from the same network, your cookie is checked by the ads network – you know the same individual visited both websites. This tracks you around the web through the advertisement networks.

5. 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 your shown? Include screenshots or link to a screencast confirming that you have successfully conducted an XSS attack.



Figure 4: XSS Attack Source

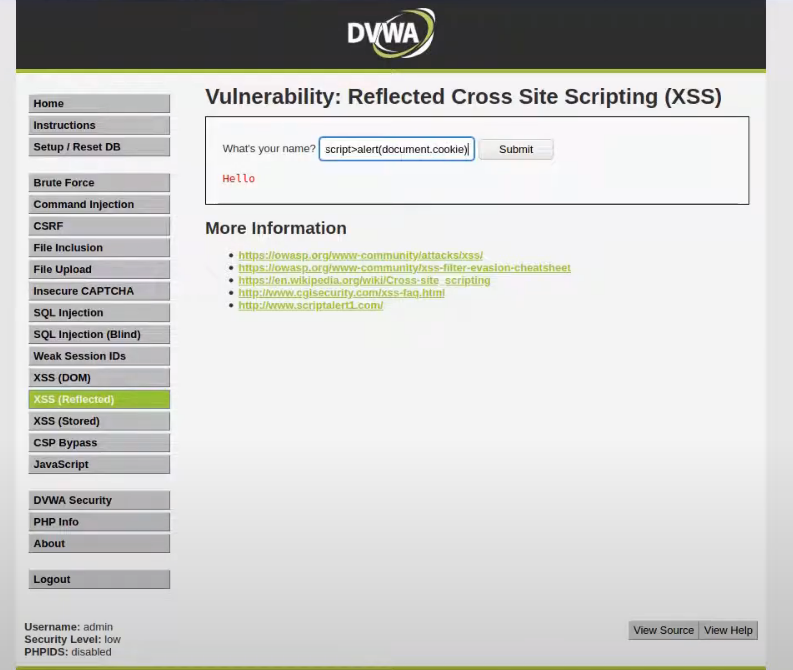


Figure 5: XSS Document Cookie

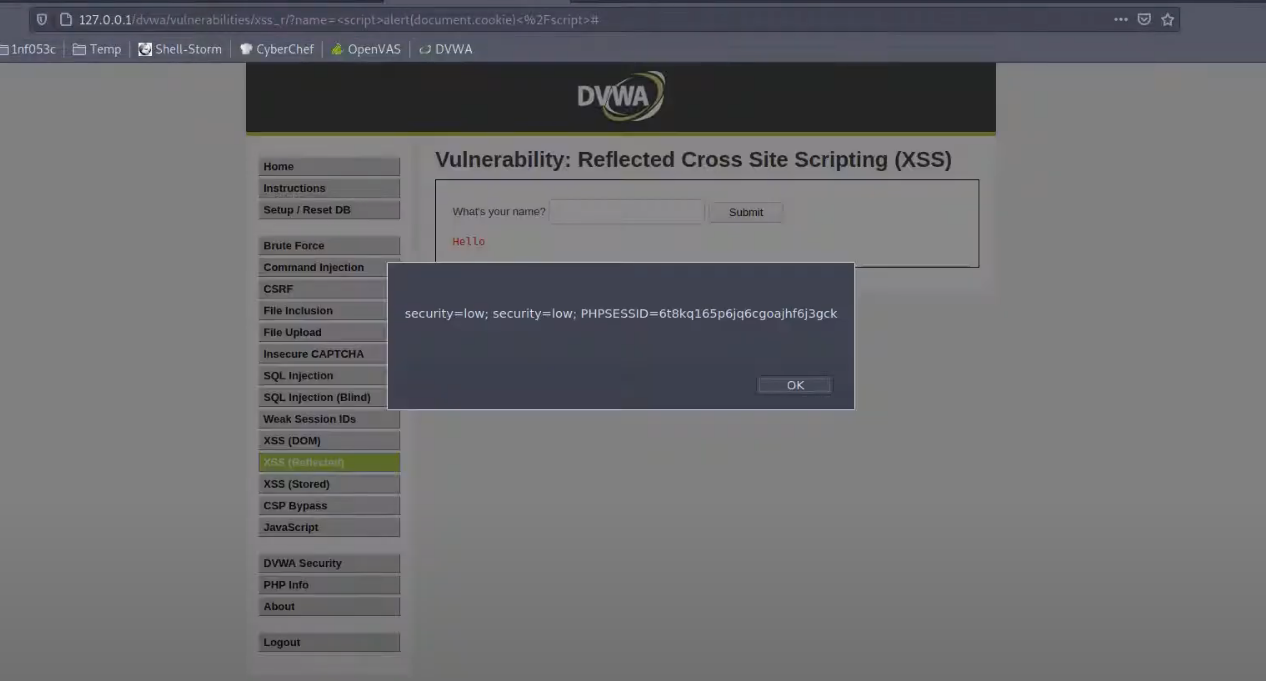


Figure 6: XSS Document Cookie Result

6. What is the difference between CSRF and XSS

XSS is a web server information security loophole that allows a cybercriminal to insert client-side scripts into user-viewed web pages. The cybercriminal lets the browser of the victim perform a script, which the attacker inserted while using a reputable website (mostly written in JavaScript). The cyber attacker will insert the JavaScript into a website the victim trusts. It does not require an authenticated session and can be taken advantage of when there is no validation or escape of feedback on the insecure website.

Cross Site Request Forgery is one of the most serious security flaws in a variety of areas, ranging from modifying user details to complete user access. The cyber criminals are using the background of the victim such as cookies to force/push you to make a request that you did not expect. The server scans the cookie you are sending with the request any time you connect with the web site, so that it knows what you are.

Both attacks commonly include on the network and require some forms of user interaction (e.g. clicking a link or visiting a website). You are targeting a user instead of the server, unlike RFI or SQLi vulnerabilities. XSS is inherently more efficient than CSRF because it normally enables random script code to be executed while CSRF is limited to a certain operation (e.g. changing the password). It should be remembered that a successful XSS attack successfully circumvents all steps against CSRF.

7. Can an Intrusion Prevention System such as Snort prevent CSRF and XSS attacks? Briefly discuss and support your answer.

The most popular online attacks of the framework layer are SQL injection and cross-site scripting (XSS). Both attacks happen if the user feedback is not filtered or checked properly to ensure it is beyond the expected limits of the program. To detect those forms of SQL injection and XSS attacks, a network-based intrusion detection tool (IDS) such as Snort can be set. In fact, Snort has a default rule collection containing signatures for these intrusions to be detected. However, an attacker can easily be bypassed, particularly if the malicious input string is converted into its hex-encoded value.

Basically, Snort uses predefined fixed signatures to follow the trend of network traffic. They will parse HTTP traffic and analyze it fundamentally. These capabilities can be used to develop custom designs to detect CSRF and XSS attacks, which are unique and pre-known. This are not enough for an easy-to-use solution, precisely because no single string match or regular expression can generally be observed for such attacks.

Snort is also unable to analyze its own HTTPS traffic. This ensures that the traffic has to be strategically positioned in the network for processing.

**Question 3 – Interview questions**

**For SQL attacks:**

How does SSL work? Suppose a proxy server (Burpsuite) is in b/w server and client so which certificate (server SSL or Burpsuite, SSL) client browser will validate?

What is SQL injection? How do you check it?

What is XSS and Difference between XSS and SQL Injection?

How to check open ports in Linux?

**For CSRF Attacks:**

What is Cross-Site Request Forgery?

In what category XXE falls?

How can you mitigate SQL injection/ XSS/ CSRF attack?