*Software security is simply a collection of methods used to protect computer programs and the sensitive information handled by them against malicious attacks. It covers a wide range of functions to safeguard software and its correlated data on privacy, accuracy, and accessibility respectively.*

***What is Software Security?***

*Software Security is aimed at finding and reducing security risks. These risks can be different and include external threats in the form of cyber attacks or internal weak points due not only to coding mistakes but also inadequate design or other defects that may potentially exist in a particular piece of software. Essentially, software security is a shield from many threats that if not addressed may cause data leaks, loss of money, or users’ lack of trust in the company.*

***What are the threats to Software?***

*Threats can be broadly categorized into two main types –*

1. *External Threats - External threat is the term used for referring to the likes of hackers, the criminals operating on the internet and also the state’s sponsored entities. This may allow them to use weak points in software in order to steal confidential information and even break into systems.*
   1. *Malware: Malware such as viruses, worms and ransomware may enter through vulnerable software.*
   2. *Distributed Denial of Service (DDoS) Attacks: In essence, these attacks are characterized by flooding of a system or a network with traffic and making it inaccessible for users who need to make genuine requests. DDoS attacks, on the other hand, are more difficult to block. This is because certain DDoS attacks originate from hundreds or thousands of systems at the same time—meaning they also make multiple thousands or millions of requests to your web application simultaneously—leading to system strain and a serious slowdown of your web application. Employing rate-limiting in your web application can prevent these types of attacks.*
   3. *Phishing: Therefore, attackers use deceitful ways of making people reveal their confidential data like login credentials and other finances.*
   4. *Data Breaches: One may lose vital data like personal information or financial transactions, which can then be used by unwanted individuals.*
2. *Internal Threats - These internal threats result from people within one organization, whether inadvertently or purposely. They may include:*
   1. *Insider Threats: Such privileged people such as employees or others who have access to the software may use it against the organization and steal data.*
   2. *Human Error: Unintentional employee behaviours including, misconfiguration and accident data leaks are among the main risks.*

*Importance of Software Security –*

* *Data Protection: Since most software are based on such confidential data, failure of securing software results into data breach, identity theft and monetary losses.*
* *Business Continuity: Security incidents can put operations on hold, resulting in revenue losses and a negative impact on the image of an organization.*
* *Regulatory Compliance: There are many countries across the globe with rigid data protection laws, failure to comply may lead to legal liabilities and loss of reputation.*
* *User Trust: Data handling must conform to user-expectations, being highly confidential and careful. Customers can lose faith in a company and refuse to buy its products.*
* *Intellectual Property Protection: The software constitutes crucial intellectual property. It is important to guard it from those that may use it without permission and lose money.*

***Front-end security best practices*** *– Overlooking front-end security can leave your web applications vulnerable to a wide range of threats, including cross-site scripting (XSS) attacks, cross-site request forgery (CSRF) attacks, and other security vulnerabilities. The front-end developer, safeguard your web applications from malicious scripts and potential security risks.*

*Front-end security primarily deals with protecting the client side of web applications, including the user interface and any JavaScript code executed in the user’s browser.*

*Common security threats in front-end development –*

1. ***Cross-Site Scripting (XSS) attacks****: XSS attacks occur when malicious code is injected into a web application and executed within a user’s browser. This can lead to the theft of sensitive data and other malicious activities. XSS attacks are drawn to a lack of sanitization in a web application's input and output, which can lead to a variety of attacks. For example, checkout buttons can be replaced with buttons redirecting users to fake banking pages, legitimate download buttons can be replaced with buttons resulting in malware downloads, and more. With XSS attacks, an attacker can inject JavaScript libraries, which then execute on the client side—logging the user's IP address, geolocation and other personal details.*
2. ***Cross-Site Request Forgery (CSRF) attacks****: CSRF attacks involve tricking users into performing actions they didn’t intend to take. Attackers exploit the trust that a website has in a user’s browser to execute unauthorized actions. For example, a user is logged into his banking application and browsing the internet at the same time. The user then comes across a "Download" button which he clicks on, and instead of downloading anything for the user, that fateful click transfers funds from the user's bank account to the attacker. CSRF attacks can be prevented by using a token value similar to an md5sum or sha256sum of random characters, which is generated on every page load and passed to a form via HTTP headers, upon the submission of any form. If the header token value is missing or if there is a token mismatch, the action is not performed, and the user remains safe.*
3. ***Injection attacks****: Injection attacks, such as SQL injection, involve inserting malicious code into input fields, which can then be executed on the server side, potentially compromising sensitive information.*
4. ***Broken access control****: Broken access control occurs when users can access unauthorized areas or perform actions they shouldn’t. It’s a critical security flaw that threatens data confidentiality and application integrity.*
5. ***Security risks in external scripts****: Third-party libraries and external resources in your web application can introduce security vulnerabilities if not properly vetted.*

***Front-end security best practices***

1. *Input validation and sanitization - One of the fundamental steps in front-end security is proper input validation and sanitization. User input should not be trusted under any circumstances. validation and sanitization are two distinct processes. Validation acts as a filter for the user input, making sure it meets a set of rules decided by the developer. On the other hand, sanitization consists of removing any unwanted characters that could be potentially harmful. They help you protect common and harmful security vulnerabilities such as XSS, local file inclusion (LFI), or SQL injection, maintaining the integrity of sensitive data and preventing any unexpected behavior of the web application.*

*There are many ways to implement validation and sanitization in your code. At a very basic level, you could use HTML5 attributes. But, if you want to take it one step further, you could use JavaScript to perform complex validation or remove certain characters from the input string. If you use any framework, it is also advised to look into any packages that may help you with input handling. If your code uses the input to perform database queries, you should also use prepared statements, as they protect against SQL injection-type attacks. To ensure best practices, it is recommended to perform input validation on both the front-end and back-end of the web application.*

1. *Content Security Policy - XSS attacks take advantage of the browser being unable to differentiate between legitimate code and malicious code. So instead of blindly letting the browser execute any code a page requests, we are going to filter it based on its source. CSP allows developers to specify which sources of scripts, styles, and other resources are considered trusted within a web application. Using a CSP in front-end security is a proactive measure to mitigate XSS and other code injection attacks.*

*To use this utility in your application you have to define the Content-Security-Policy HTTP header, with which you can create a whitelist of trusted sources that can execute code.*

*Ex - Content-Security-Policy: script-src ‘self’*

*CSP restricts the execution of inline scripts and unauthorized external resources, reducing the attack surface for potential security threats.*

1. *Avoid inline scripts - Protecting against inline scripts is vital for front-end security. These scripts pose a significant risk, as they can execute arbitrary code. The best practice is to separate JavaScript from HTML, using external scripts, and implement Content Security Policies (CSPs) to define trusted sources.*
2. *Secure HTTP requests - When making HTTP requests, ensure they are secure by using HTTPS. HTTPS uses TLS to encrypt HTTP traffic, improving safety and security. So, why is using HTTPS so important? Using HTTPS is more secure for both the user and the web server, as the encryption goes both ways: from the server to the end user and vice versa. This way, none of the information transmitted during the connection is in plain text, preventing attacks such as man-in-the-middle. Secondly, an SSL certificate authenticates the website, meaning that a trustworthy third party verifies that the web server is who it claims to be, protecting the user against different threats like website spoofing.*
3. *Penetration testing - Web applications are critical systems as they are directly exposed to the outside world. Given this criticality, you should be worried about how secure your network is. The practice of penetration testing proves to be a valuable asset in detecting vulnerabilities before they can be exploited by malicious actors. By thoroughly evaluating the security measures in place, potential weaknesses can be identified and addressed, reducing the risk of a successful cyber attack. During a vulnerability assessment, some crucial types of attacks are being tested for:*

* *Injection attacks*
* *Broken access control*
* *Improper error handling*
* *Broken authentication*
* *XSS attacks*

*Sometimes, the people who create your applications may make mistakes. To ensure that your team’s work is error-free, you can ask an external partner to conduct a penetration test. This test helps to identify and fix any weak spots that could be exploited by hackers.*

1. *Dependency management -Managing software dependencies can be a challenging task, as it involves dealing with external libraries that perform specific functions, and may vary in size and complexity. Using outdated dependencies could potentially harm the security of your web application since certain libraries may have known vulnerabilities.*

* *Keep track of the dependencies you utilize and the versions of those dependencies.*
* *Review and update your dependencies on a regular basis. Check that you are utilizing the most stable versions of your dependencies.*
* *Map out your dependencies using a consumable, shared resource such as a software bill of materials (SBOM).*

*Backend Security Approaches –*

1. *Validate input - The first step to secure a back-end application is to validate the input that comes from the user or other sources. Validation means checking that the input is in the expected format, type, and range, and that it does not contain any malicious code or commands. You can use built-in functions, libraries, or frameworks to perform validation, depending on the back-end language and framework you are using.*
2. *Sanitize output - The second step to secure a back-end application is to sanitize the output that goes to the user or other systems. Sanitization means removing or escaping any characters or symbols that could be interpreted as code or commands by the browser or the database. You can use built-in functions, libraries, or frameworks to sanitize output, depending on the back-end language and framework you are using.*
3. *Encrypt data - The third step to secure a back-end application is to encrypt the data that is stored or transmitted. Encryption means converting the data into a form that can only be read by authorized parties who have the key or the password. This prevents unauthorized access, modification, or theft of the data.*
4. *Implement authentication - The fourth step to secure a back-end application is to implement authentication. Authentication means verifying the identity of the user or the system that is accessing the back-end application. This prevents unauthorized or impersonated access to the back-end application.*
5. *Implement authentication - The fourth step to secure a back-end application is to implement authentication. Authentication means verifying the identity of the user or the system that is accessing the back-end application. This prevents unauthorized or impersonated access to the back-end application.*
6. *Monitor and update - The sixth and final step to secure a back-end application is to monitor and update. Monitoring means keeping track of the performance, activity, and errors of the back-end application. Updating means applying patches, fixes, and upgrades to the back-end application. This prevents performance issues, security breaches, and vulnerabilities in the back-end application.*

*Database Attacks and Security*

*Attacks –*

1. *SQL/NoSQL Injection Attacks - A database-specific threat involves the use of arbitrary non-SQL and SQL attack strings into database queries. Typically, these are queries created as an extension of web application forms, or received via HTTP requests. Any database system is vulnerable to these attacks, if developers do not adhere to secure coding practices, and if the organization does not carry out regular vulnerability testing.*
2. *Buffer Overflow Attacks - Buffer overflow takes place when a process tries to write a large amount of data to a fixed-length block of memory, more than it is permitted to hold. Attackers might use the excess data, kept in adjacent memory addresses, as the starting point from which to launch attacks.*
3. *An Evolving IT Environment - The evolving IT environment is making databases more susceptible to threats. Here are trends that can lead to new types of attacks on databases.*

* *Growing data volumes.*
* *Distributed infrastructure*
* *Increasingly tight regulatory requirements*
* *Cybersecurity skills shortage*

*How To Secure Database Server – Securing a database server, also known as “hardening”, is a process that includes physical security, network security, and secure operating system configuration.*

* *Ensure Physical Database Security - Refrain from sharing a server for web applications and database applications, if your database contains sensitive data. If you do rely on a web hosting service to manage your database, ensure that it has a strong security track record. If you manage your database in an on-premise data centre, Ensure you have physical security measures, including locks, cameras, and security personnel in your physical facility. Any access to physical servers only granted to authorized individuals.*
* *Lock Down Accounts and Privileges - It is critical to ensure that every privileged account on a database server is configured with a strong, unique password. If accounts are not needed, they should be expired and locked. For the remaining accounts, access has to be limited to the absolute minimum required. Each account should only have access to the tables and operations (for example, SELECT or INSERT) required by the user.*
* *Regularly Patch Database servers - Ensure that patches remain current. Effective database patch management is a crucial security practice. A timely deployment of up-to-date versions of database service packs, critical security hotfixes, and cumulative updates will improve the stability of database performance.*
* *Disable Public Network Access - Organizations store their applications in databases. In most real-world scenarios, the end-user doesn’t require direct access to the database. Thus, you should block all public network access to database servers.*
* *Encryption and Backups - Encrypting your data makes it unreadable to both attackers and employees. Without an encryption key, they cannot access it, this provides a last line of defense against unwelcome intrusions. Encrypt all-important application files, data files, and backups so that unauthorized users cannot read your critical data.*