Owasp Vulnerabilities Writeup

- A01 Broken Access Control

  In this particular type of vulnerability we see that there is a weak infrastructural dissolution within the current setup of the website i.e basically variated roles and types of users are created such as admin ,viewer, editor etc.

  If implemented properly the admin will be able to have access to the whole page starting from the login to the nitty gritty that is the data stored in their DB whereas the viewer will only be able to have the surface level view of the complete structure

  Only a substructure of the complete code is visible to the user with only some certain access granted to the user.

  With the exception of public resources, deny by default.

  Implement access control mechanisms only once and reuse them throughout the application, minimizing the use of Cross-Origin Resource Sharing (CORS).

  Instead of allowing the user to create, read, update, or delete any record, model access controls should enforce record ownership.

  Domain models should enforce unique application business limit requirements.

  Disable web server directory listing and ensure that no file metadata (e.g git) or backup files exist within web roots.

  Log access control failures and notify administrators as needed (e.g repeated failures).

  Limit API and controller access to reduce the impact of automated attack tooling.

- A02 Cryptographic Failures

  Lets understand this better with an example, passwords, credit card numbers, health records, personal information, and trade secrets, especially if that information is subject to data protection laws.

  They are initialization vectors ignored, reused, or generated insecurely enough for how cryptography works used to Identify what data is sensitive according to privacy laws,

regulatory requirements, or business needs. Make sure the current strong default algorithms, protocols, and keys are in place. Use good key management.

Encrypt all data in transit using secure protocols such as Forward Secrecy (FS) ciphers, server prioritization of ciphers, and TLS with secure parameters.Enforce encryption using directives such as HTTP Strict Transport Security (HSTS).Do not transfer sensitive data using legacy protocols such as FTP or SMTP.

In many modes, this means using a CSPRNG (cryptographically secure pseudo-random number generator). For modes that require a nonce, the initialization vector (IV) does not require a CSPRNG.

Do not use legacy cryptographic functions and padding schemes such as MD5, SHA1, PKCS number 1 v1. The application uses automatic database encryption to encrypt credit card numbers in the database.

However, this data is automatically decrypted upon retrieval, so a SQL injection bug could retrieve the credit card number in plain text.The website does not use or enforce TLS on all pages or supports weak encryption.The password database stores everyone's passwords using unsalted or simple hashes.

- A03 Injection

Energetic questions or non-parameterized calls without context-aware getting away are utilized straightforwardly within the mediator.

Threatening information is utilized inside object-relational mapping (ORM) look parameters to extricate extra, delicate records.

How to Anticipate Anticipating infusion requires keeping information partitioned from commands and questions:

The favored alternative is to utilize a secure API, which maintains a strategic distance from utilizing the mediator completely, gives a parameterized interface, or relocates to Question Social Mapping Apparatuses (ORMs).

Indeed when parameterized, put away methods can still present SQL infusion in the event that:

1) PL/SQL
2) T-SQL

concatenates inquiries and information or executes threatening information with EXECUTE Prompt or exec().

- A04 Insecure Design

  An uncertain plan cannot be settled by a culminate usage as by definition, required security controls were never made to guard against particular assaults.

  One of the components that contribute to an uncertain plan is the need of commerce chance profiling characteristic within the program or framework being created, and in this way the disappointment to decide what level of security plan is required.

  Danger modeling ought to be coordinated into refinement sessions (or comparative exercises); explore for changes in data streams and get to control or other security controls.

  Within the client story improvement, decide the right stream and disappointment states, guarantee they are well caught on and concurred upon by capable and affected parties.

  Analyze suspicions and conditions for anticipated and disappointment streams, guarantee they are still exact and alluring.

- A05 Security Misconfiguration

  An uncertain plan cannot be settled by a culminate usage as by definition, required security controls were never made to guard against particular assaults.

  One of the components that contribute to an uncertain plan is the need of commerce chance profiling characteristic within the program or framework being created, and in this way the disappointment to decide what level of security plan is required.

  Danger modeling ought to be coordinated into refinement sessions (or comparative exercises); explore for changes in data streams and get to control or other security controls.

  Within the client story improvement, decide the right stream and disappointment states, guarantee they are well caught on and concurred upon by capable and affected parties.

  Analyze suspicions and conditions for anticipated and disappointment streams, guarantee they are still exact and alluring.

- A06 Vulnerable and Outdated Components

  In case you do not know the versions of all components you utilize. This incorporates components you specifically utilize as well as settled conditions.

On the off chance that the program is helpless, unsupported, or out of date. This incorporates the OS, web/application server, database administration framework (DBMS), applications, APIs and all components, runtime situations, and libraries.

On the off chance that you do not filter for vulnerabilities regularly and subscribe to security bulletins related to the components you utilize.

In case you are not settling or updating the basic stage, systems, and conditions in a risk-based, opportune design. This commonly happens in situations when fixing may be a month to month or quarterly errand beneath alter control, leaving organizations open to days or months of superfluous presentation to settle vulnerabilities.

On the off chance that computer program engineers don't test the compatibility of upgraded, updated, or fixed libraries.In case you are not secure the components' arrangements.

- A07 Identification and Authentication Failures

  Affirmation of the user's character, verification, and session administration is basic to ensure against authentication-related assaults. There may be confirmation shortcomings in case the application:

  Grants mechanized assaults such as credential stuffing, where the aggressor contains a list of substantial usernames and passwords.Licenses default, frail, or well-known passwords, such as "Pass123" or "admin/admin".

  Employments frail or incapable credential recuperation and forgotten-password forms, such as "knowledge-based answers," which cannot be made secure.

  Employments plain content, scrambled, or pitifully hashed passwords information stores lost or ineffective multi-factor verification:

  1) Uncovered session identifier within the URL.
  2) Reuse session identifier after effective login.

  Does not accurately negate Session IDs. Client sessions or confirmation tokens which are basically SSO tokens aren't legitimately nullified amid logout or a period of inertia.

- A08 Software and Data Integrity Failures

  Computer program and information keenness disappointments relate to code and framework that does not secure against keenness infringement. An illustration of this is often where an application depends upon plugins, libraries, or modules from untrusted sources, stores, and substance conveyance systems. An uncertain CI/CD pipeline can present the potential for unauthorized get to, malevolent code, or framework

compromise. In conclusion, numerous applications presently incorporate auto-update usefulness, where upgrades are downloaded without adequate judgment confirmation and connected to the already trusted application. Aggressors seem to possibly transfer their claim overhauls to be disseminated and run on all establishments. Another illustration is where objects or information are encoded or serialized into a structure that an assailant can see and adjust is helpless to unreliable deserialization.

Utilize computerized marks or comparative instruments to confirm the program or information is from the anticipated source and has not been changed.

Guarantee libraries and conditions, such as npm or Maven, are consuming trusted stores. On the off chance that you've got the next chance profile, consider facilitating an inside known-good store that's confirmed.

Guarantee that a computer program supply chain security instrument, such as OWASP Reliance Check or OWASP CycloneDX, is utilized to confirm that components don't contain known vulnerabilities

Ensure that there's a review process for code and arrangement changes to play down the chance that noxious code or setup can be presented into your program pipeline.

Guarantee that your CI/CD pipeline has appropriate isolation, arrangement, and get to control to guarantee the keenness of the code streaming through the construct and send forms.

Guarantee that unsigned or decoded serialized information isn't sent to untrusted clients without a few frame of keenness check or computerized signature to identify altering or replay of the serialized information

- A09 Security Logging and Monitoring Failures

  Computer program and information keenness disappointments relate to code and framework that does not secure against keenness infringement. An illustration of this is often where an application depends upon plugins, libraries, or modules from untrusted sources, stores, and substance conveyance systems (CDNs). An uncertain CI/CD pipeline can present the potential for unauthorized get to, malevolent code, or framework compromise. In conclusion, numerous applications presently incorporate auto-update usefulness, where upgrades are downloaded without adequate judgment confirmation and connected to the already trusted application. Aggressors seem to possibly transfer their claim overhauls to be disseminated and run on all establishments. Another illustration is where objects or information are encoded or serialized into a structure that an assailant can see and adjust is helpless to unreliable deserialization.

  Utilize computerized marks or comparative instruments to confirm the program or information is from the anticipated source and has not been changed.

Guarantee libraries and conditions, such as npm or Maven, are consuming trusted stores. On the off chance that you've got the next chance profile, consider facilitating an inside known-good store that's confirmed.

Guarantee that a computer program supply chain security instrument, such as OWASP Reliance Check or OWASP CycloneDX, is utilized to confirm that components don't contain known vulnerabilities

Ensure that there's a review process for code and arrangement changes to play down the chance that noxious code or setup can be presented into your program pipeline.

Guarantee that your CI/CD pipeline has appropriate isolation, arrangement, and get to control to guarantee the keenness of the code streaming through the construct and send forms.

Guarantee that unsigned or decoded serialized information isn't sent to untrusted clients without a few frame of keenness check or computerized signature to identify altering or replay of the serialized information

- A10 Server Side Request Forgery (SSRF)

SSRF blemishes happen at whatever point a web application is getting an inaccessible asset without approving the user-supplied URL. It permits an aggressor to coerce the application to send a made task to an unforeseen goal, indeed when ensured by a firewall, VPN, or another sort of organized get to control list (ACL).

As cutting edge web applications give end-users with helpful highlights, bringing a URL gets to be a common situation. As a result, the frequency of SSRF is expanding. Moreover, the seriousness of SSRF is getting to be higher due to cloud administrations and the complexity of structures.

Designers can anticipate SSRF by actualizing a few or all the taking after defense in profundity controls:

Fragment farther assets are useful in partitioned systems to diminish the effect of SSRF Uphold “deny by default” firewall arrangements or organize get to control rules to square all but fundamental intranet activity.

~ Set up an possession and a lifecycle for firewall rules based on applications.

~ Log all acknowledged and blocked arrange streams on firewalls

Sanitize and approve all client-supplied input information,Uphold the URL pattern, harbor, and goal with a positive permit list Don't send crude reactions to clientsImpair HTTP redirections

Be mindful of the URL consistency to maintain a strategic distance from assaults such as DNS rebinding and “time of check, time of use” (TOCTOU) race conditions

Don't moderate SSRF by means of the utilization of a deny list or normal expression. Assailants have payload records, apparatuses, and abilities to bypass deny records.

Do not send other security important administrations on front frameworks. Control neighborhood activity on these frameworks

For frontends with devoted and sensible client bunches utilize arrange encryption on autonomous frameworks to consider exceptionally tall assurance needs.