Application Security Testing

Indian Computer Emergency Response Team (CERT-In)

OWASP Top 10

- Place to start for learning about application security risks.
- Periodically updated
- What is OWASP?
  - Open Web Application Security Project, a non-profit worldwide charitable organization focused on improving the security of application software.

- Good place to get started with application Security.
- Developers have to learn from the mistakes of other organisations.
- Executive have to think about management of risk that application or software create in enterprise.

- Approach application security as a people, process and technology problem(app sec requires improvement in all these area)
- Use owasp and other resources to establish strong foundation of training, standards and tools to make secure coding possible.
- Organization should integrate security into their development, verification and maintenance process

Owasp Top 10 is about top 10 Risks not about common weaknesses.

Its not an application security program.

Aim of top 10 is to educate developers , designers, architects,managers and organizations about the consequences of important web application security weakness.

A1. Injection Flaws

Definition

- Injection flaws occur when un-trusted data is sent to an interpreter as part of a command or query. The attacker's hostile data can trick the interpreter into executing unintended commands or accessing unauthorized data.

- Various flavors of injection flaws: SQL, OS, LDAP to name a few.

A1. Injection Flaws

Impact (Severe)

- Data loss or corruption
- Lack of accountability
- Denial of access
- In certain cases could lead to complete takeover of host

A1. Injection Flaws

Prevention

- Do not trust data from clients, validate all input.
- Use parameterized APIs whenever possible, e.g. SQL prepared statements
- If parameterized API not available, use escaping routines before sending data to the interpreter/shell.

A2. Cross-Site Scripting (XSS)

Definition

- XSS flaws occur whenever an application takes un-trusted data and sends it to a web browser without proper validation and escaping. XSS allows attackers to execute scripts in the victim's browser which can hijack user sessions, deface web sites, or redirect the user to malicious sites.
- Three types – stored, reflected, and DOM based XSS.
- The most prevalent web application security flaw.

A2. Cross-Site Scripting (XSS)

Bad name given to a dangerous security issue

Attack targets the user of the system rather than the system itself.

Outside client-side languages executing within the users web environment with the same level of privilege as the hosted site.

A2. Cross-Site Scripting (XSS)

Impact (Moderate)

- Attacker can execute scripts in a victim's browser, which can open the door to:
  - Hijacking the user's session
  - Defacing the web site
  - Insertion of hostile content
  - Redirecting the user to another site
  - Attempting to install malware on the user's machine

A2. Cross-Site Scripting (XSS)

Prevention

- Escape/encode all data that is written to a web page.
  - **<script>alert('got you');</script>** (raw html)
  - &lt;script&gt;alert('got you')&lt;/script&gt; (encoded html)

- Do not trust data from clients, validate all input.

A3. Broken Authentication and Session Management

Definition

- Application functions related to authentication and session management are often not implemented correctly, allowing attackers to compromise passwords, keys, session tokens, or exploit other implementation flaws to assume other users' identities.

A3. Broken Authentication and Session Management

Impact (Severe)

- Such flaws may allow some or even all accounts to be attacked.
- Once successful, the attacker can do anything the victim could do.
- Privileged accounts are frequently targeted.

A3. Broken Authentication and Session Management

Prevention

- A single set of strong authentication and session management controls. Such controls should strive to:
  - Meet the requirements defined in OWASP's Application Security Verification Standard(ASVS).
  - Have a simple interface for developers. Consider the ESAPI Authenticator and User APIs as good examples to emulate, use, or build upon.

- Strong efforts should also be made to avoid XSS flaws which can be used to steal session IDs. See A2.

A4. Insecure Direct Object References

Definition

- A direct object reference occurs when a developer exposes a reference to an internal implementation object, such as a file, directory, or database key. Without an access control check or other protection, attackers can manipulate these references to access unauthorized data.

A4. Insecure Direct Object References

Impact (Moderate)

- Such flaws can compromise all the data that can be referenced by the parameter.
- Unless the namespace is sparse, it’s easy for an attacker to access all available data of that type.

A4. Insecure Direct Object References

Prevention

- Use per-user or session indirect object references. This prevents attackers from directly targeting unauthorized resources by knowing actual keys.
- Check access. Each use of a direct object reference from an untrusted source must include an access control check to ensure the user is authorized for the requested object.

A5. Cross-Site Request Forgery (CSRF)

Definition

- A CSRF attack forces a logged-on victim's browser to send a forged HTTP request, including the victim's session cookie and any other automatically included authentication information, to a vulnerable web application.
- This allows the attacker to force the victim's browser to generate requests the vulnerable application thinks are legitimate requests from the victim.

A5. Cross-Site Request Forgery (CSRF)

Impact (Moderate)

- Attackers can cause victims to change any data the victim is allowed to change or perform many function the victim is authorized to use.

A5. Cross-Site Request Forgery (CSRF)

Prevention

- Preventing CSRF requires the inclusion of a unpredictable token in the body or URL of each HTTP request. Such tokens should at a minimum be unique per user session, but can also be unique per request.
  - The preferred option is to include the unique token in a hidden field. This causes the value to be sent in the body of the HTTP request, avoiding its inclusion in the URL, which is subject to exposure.
- OWASP's CSRF Guardcan be used to automatically include such tokens in your Java EE, .NET, or PHP application. OWASP's ESAPI includes token generators and validators that developers can use to protect their transactions.

A6. Security Misconfiguration

Definition

- Good security requires having a secure configuration defined and deployed for the application, frameworks, application server, web server, database server, and platform.
- All these settings should be defined, implemented, and maintained as many are not shipped with secure defaults. This includes keeping all software up to date, including all code libraries used by the application.

A6. Security Misconfiguration

Impact (Moderate)

- Such flaws frequently give attackers unauthorized access to some system data or functionality.
- Occasionally, such flaws result in a complete system compromise.

A6. Security Misconfiguration

Prevention

- A repeatable hardening process that makes it fast and easy to deploy another environment that is properly locked down. Development, QA, and production environments should all be configured identically. This process should be automated to minimize the effort required to setup a new secure environment.
- A process for keeping abreast of and deploying all new software updates and patches in a timely manner to each deployed environment. This needs to include all code libraries as well, which are frequently overlooked.

A7. Insecure Cryptographic Storage

Definition

- Many web applications do not properly protect sensitive data, such as credit cards, SSNs, and authentication credentials, with appropriate encryption or hashing. Attackers may steal or modify such weakly protected data to conduct identity theft, credit card fraud, or other crimes.

A7. Insecure Cryptographic Storage

Impact (Moderate)

- Failure frequently compromises all data that should have been encrypted. Typically this information includes sensitive data such as health records, credentials, personal data, credit cards, etc.

A7. Insecure Cryptographic Storage

Prevention

- Considering the threats you plan to protect this data from (e.g., insider attack, external user), make sure you encrypt all such data at rest in a manner that defends against these threats.
- Ensure offsite backups are encrypted, but the keys are managed and backed up separately.
- Ensure appropriate strong standard algorithms and strong keys are used, and key management is in place.
- Ensure passwords are hashed with a strong standard algorithm and an appropriate salt is used.
- Ensure all keys and passwords are protected from unauthorized access.

A8. Failure to Restrict URL Access

Definition

- Many web applications check URL access rights before rendering protected links and buttons. However, applications need to perform similar access control checks each time these pages are accessed, or attackers will be able to forge URLs to access these hidden pages anyway.

A8. Failure to Restrict URL Access

Example Attack Scenario

The attacker simply force browses to target URLs. Consider the following URLs which are both supposed to require authentication. Admin rights are also required for access to the **“admin_getappInfo”** page.

If the attacker is not authenticated, and access to either page is granted, then unauthorized access was allowed. If an authenticated, non-admin, user is allowed to access the **“admin_getappInfo”** page, this is a flaw, and may lead the attacker to more improperly protected admin pages. Such flaws are frequently introduced when links and buttons are simply not displayed to unauthorized users, but the application fails to protect the pages they target.

A8. Failure to Restrict URL Access

Impact (Moderate)

- Such flaws allow attackers to access unauthorized functionality.
- Administrative functions are key targets for this type of attack.

A8. Failure to Restrict URL Access

Prevention

- Select an approach for requiring proper authentication and proper authorization for each page. Frequently, such protection is provided by one or more components external to the application code. Regardless of the mechanism(s), all of the following are recommended:
  - The authentication and authorization policies be role based, to minimize the effort required to maintain these policies.
  - The policies should be highly configurable, in order to minimize any hard coded aspects of the policy.
  - The enforcement mechanism(s) should deny all access by default, requiring explicit grants to specific users and roles for access to every page
  - If the page is involved in a workflow, check to make sure the conditions are in the proper state to allow access.
-

A9. Insufficient Transport Layer Protection

Definition

- Applications frequently fail to authenticate, encrypt, and protect the confidentiality and integrity of sensitive network traffic. When they do, they sometimes support weak algorithms, use expired or invalid certificates, or do not use them correctly.

A9. Insufficient Transport Layer Protection

Impact (Moderate)

- Such flaws expose individual users' data and can lead to account theft.
- If an admin account was compromised, the entire site could be exposed.
- Poor SSL setup can also facilitate phishing and MITM attacks.

A9. Insufficient Transport Layer Protection

Prevention

- Providing proper transport layer protection can affect the site design. It's easiest to require SSL for the entire site. For performance reasons, some sites use SSL only on private pages. Others use SSL only on 'critical' pages, but this can expose session IDs and other sensitive data. At a minimum, do all of the following:
  - Require SSL for all sensitive pages. Non-SSL requests to these pages should be redirected to the SSL page.
  - Set the 'secure' flag on all sensitive cookies.
  - Configure your SSL provider to only support strong (e.g., FIPS 140-2 compliant) algorithms.
  - Ensure your certificate is valid, not expired, not revoked, and matches all domains used by the site.
  - Backend and other connections should also use SSL or other encryption technologies.

A10. Unvalidated Redirects and Forwards

Definition

- Web applications frequently redirect and forward users to other pages and websites, and use untrusted data to determine the destination pages. Without proper validation, attackers can redirect victims to phishing or malware sites, or use forwards to access unauthorized pages.
- A favorite target of phishers trying to gain the user's trust

A10. Unvalidated Redirects and Forwards

Impact

- Such redirects may attempt to install malware or trick victims into disclosing passwords or other sensitive information.
- Unsafe forwards may allow access control bypass.

A10. Unvalidated Redirects and Forwards

Prevention

- Safe use of redirects and forwards can be done in a number of ways:
  - Simply avoid using redirects and forwards, if possible.
  - If used, don't involve user parameters in calculating the destination. This can usually be done.
- If destination parameters can't be avoided, ensure that the supplied value is valid, and authorized for the user.

Thank You