What is OWASP?

The Open Web Application Security Project (OWASP) is a not-for-profit worldwide charitable organization focused on improving the security of application software.

http://www.owasp.org

What the OWASP Top Ten Web Application Security Risks are NOT:

- Network vulnerabilities
- Firewall vulnerabilities
- IDS/IDP
- Application Containers such as Tomcat
- Other back end systems
- Virus Scanning

Today's Objective

We ARE talking about things that an application developer must take into account as he/she is writing the web application software.

Not Much Change...

A1 – Injection
A2 – Injection Flaws
A2 – Cross Site Scripting (XSS)
A7 – Broken Auth. & Sens. Mgt.
A4 – Insecure Direct Obj. Ref.
A4 – Insecure Direct Obj. Ref.
A5 – Cross Site Req Forgery (CSRF)
A5 – Cross Site Req Forgery (CSRF)
N/A – Security Misconfiguration
A8 – Insecure Cryptographic Storage
A7 – Insecure Cryptographic Storage
A10 – Failure to Restrict URL Access
A8 – Failure to Restrict URL Access
A9 – Insecure Communications
A9 – Insufficient Transport Layer Prot.
N/A – (Dropped in 2010)
A3 – Malicious File Execution
A6 – Information Leakage
A10 – Unvalidated Redirects & Forwards
A10 – Unvalidated Redirects & Forwards
A6 – Information Leakage

A1 – Injection

Injection flaws, such as SQL, OS, and LDAP injection, occur when untrusted 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.
### Preventing Injection

Keep input data separate from commands and queries

- Completely avoid using input as part of query
- Carefully escape special characters using the specific escape syntax for that interpreter
- Use ‘whitelist’ input validation with appropriate canonicalization

---

### A2 – Cross Site Scripting (XSS)

XSS flaws occur whenever an application takes untrusted data and sends it to a web browser without proper validation and escaping. XSS allows attackers to execute script in victim’s browser which can hijack user sessions, deface web sites, or redirect the user to malicious sites.

---

### Preventing Cross Site Scripting (XSS)

Keep untrusted data separate from active browser content

- The preferred option is to properly escape all untrusted data based on the HTML context (body, attribute, JavaScript, CSS, or URL) that the data will be placed into.
- Use ‘whitelist’ input validation with appropriate canonicalization

---

### A3 – Broken Authentication and Session Management

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

---

### Preventing Broken Authentication and Session Management

- Create a single set of strong authentication and session management controls
- These controls should have a simple interface for developers
- Be careful with too many ‘conveniences’.
- Be thorough!

---

### A4 – Insecure Direct Object References

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.
Preventing Insecure Direct Object References

- Use indirect object references
- Do an access control check for each direct object access.

A5 – Cross Site Request Forgery (CSRF)

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 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.

Cross Site Request Forgery Example

Suppose an Internet user has been doing his or her banking. Then the user gets lured to a bad site. That site contains the following:

```html
<img src="http://mybank.com/withdraw?account=boom&amount=10000&for=alice">
```

Preventing Cross Site Request Forgery (CSRF)

- The preferred option is to include a unique token in a hidden field.
- The unique token can also be included in the URL itself, or a URL parameter. But this option is risky.

A6 – Security Misconfiguration

Security depends on having a secure configuration defined for the application, framework, web server, application server, and platform. All these settings should be defined, implemented, and maintained as many are not shipped with secure defaults.

Preventing Security Misconfiguration

- Implement a repeatable hardening process that makes it fast and easy to deploy another environment that is properly locked down
- Utilize a process for keeping abreast of and deploying all new software updates and patches
- Good network design.
A7 – InsecureCryptographic Storage

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

Preventing Insecure Cryptographic Storage

• Determine sensitive data that should be protected
• Make sure you encrypt such data at rest
• Ensure offsite backups are encrypted, but the keys are managed and backed up separately
• Ensure appropriate strong algorithms and strong keys are used
• 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

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

Properly Restricting URL Access

• Verify access for each page that is available on the system
• Select an approach for requiring proper authentication and proper authorization for each page
• Use a ‘deny all’ access default, with access granted as needed per page per user (Role based access control)

A9 – Insufficient Transport Layer Protection

Applications frequently fail to encrypt network traffic when it is necessary to protect sensitive communications. When they do, they sometimes support weak algorithms, use expired or invalid certificates, or do not use them correctly.

Preventing Insufficient Transport Layer Protection

• Require SSL for all selected 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/TLS provider to only support strong (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/TLS or other encryption technologies
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.

Preventing Unvalidated Redirects and Forwards

- Simply avoid using redirects and forwards
- If used, don’t involve user parameters in calculating the destination.
- If destination parameters can’t be avoided, ensure that the supplied value is valid, and authorized for the user.

Detailed Cheat Sheets (Part 1)

- A1 Injection  
- A2 Cross Site Scripting (XSS)  
  [http://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet](http://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet)
- A3 Broken Authentication and Session Management  

Detailed Cheat Sheets (Part 2)

- A5 CSRF  
- A8 Failure to Restrict URL Access  
- A9 Insufficient Transport Layer Protection  

Thank You.

Sponsored by: