Introduction to the OWASP Top Ten

Kirk Jackson
RedShield
kirk@pageofwords.com
http://hack-ed.com
@kirkj

OWASP NZ
https://www.meetup.com/OWASP-Wellington/
www.owasp.org.nz
@owaspnz

Recordings:
https://goo.gl/a2VSG2
What is OWASP?

Open Web Application Security Project (OWASP) is a nonprofit foundation that works to improve the security of software.

- A website: owasp.org
- A bunch of cool tools: Zed Attack Proxy, Juice Shop, Proactive Controls, Software Assurance Maturity Model (SAMM), Application Security Verification Standard (ASVS)
- A global community of like-minded people, meetups and conferences
Who is the OWASP Foundation?

The Open Web Application Security Project (OWASP) is a nonprofit foundation that works to improve the security of software. Through community-led open source software projects, hundreds of local chapters worldwide, tens of thousands of members, and leading educational and training conferences, the OWASP Foundation is the source for developers and technologists to secure the web.

- Tools and Resources
- Community and Networking
- Education & Training

For nearly two decades corporations, foundations, developers, and volunteers have supported the OWASP Foundation and its work. Donate, Join, or become a Corporate Member today.
The OWASP Top 10 is a standard awareness document for developers and web application security. It represents a broad consensus about the most critical security risks to web applications.

Globally recognized by developers as the first step towards more secure coding.

Companies should adopt this document and start the process of ensuring that their web applications minimize these risks. Using the OWASP Top 10 is perhaps the most effective first step towards changing the software development culture within your organization into one that produces more secure code.

**Top 10 Web Application Security Risks**

1. **Injection.** Injection flaws, such as SQL, NoSQL, 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 data without proper authorization.

2. **Broken Authentication.** Application functions related to authentication and session management are often implemented incorrectly, allowing attackers to compromise passwords, keys, or session tokens, or to exploit other implementation flaws to assume other users’ identities temporarily or permanently.

3. **Sensitive Data Exposure.** Many web applications and APIs do not properly protect sensitive data, such as financial, health, or other private information.

4. **XML External Entities (XXE).** XML processors commonly load external data references from XML documents. These references can be used to cause XML Injection attacks and also lead to information exposure, data loss and denial of service attacks.

5. **Broken Access Control.** Security misconfigurations often allow attackers to manipulate application access controls, such as session hijacking, ticket spooﬁng, and more.

6. **Security Misconfiguration.** A wide variety of software and cloud misconﬁgurations make implementations vulnerable to cyber-attacks, direct access to the website, injection of malicious code, and other attacks.

7. **Cross-Site Scripting (XSS).** XSS vulnerabilities occur whenever an application includes untrusted data in a new web page without proper validation or escaping, or updates an existing web page with user-supplied data using a browser API that can create HTML or JavaScript.

8. **Insecure Deserialization.** Malicious actors can exploit deserialization vulnerabilities by injecting their own object graphs into web applications and APIs.

9. **Using Components with Known Vulnerabilities.** Using components in applications with known open source vulnerabilities, such as Apache Struts, Burp Suite, and Java XML marshalling libraries.

10. **Security Flaw via HTTP.** Insecure transport is a primary cause of many high-impact security issues in web applications.

These risks are not meant to be an exhaustive list. The OWASP Top Ten Risks are a living list and will continue to evolve and improve as new threats and vulnerabilities are discovered.
OWASP Top Ten

Globally recognized by developers as the first step towards more secure coding.

The most critical security risks to web applications.

Updated every 2-3 years from 2003 to 2017 (2020 is in progress)
Securing the user

Web Server

Site A
DOM + JS

Site B

Web Browser

sitea.com

GET /

Site A

X
Y
OWASP Top Ten 2017

A1 Injection
A2 Broken Authentication
A3 Sensitive Data Exposure
A4 XML External Entities (XXE)
A5 Broken Access Control
A6 Security Misconfiguration
A7 Cross-Site Scripting (XSS)
A8 Insecure Deserialization
A9 Using Components with Known Vulnerabilities
A10 Insufficient Logging & Monitoring
A1 Injection

Sending hostile data to an interpreter (e.g. SQL, LDAP, command line)
A1 Injection

Sending hostile data to an interpreter (e.g. SQL, LDAP, command line)

String query = "SELECT * FROM accounts WHERE custID='' + request.getParameter("id") + '';

id = '' ; drop table accounts -- "

SQL statements combine code and data
SQLi Demo
A1 Injection

Prevention:

SQL statements combine *code* and *data*

=> Separate code and data

- Parameterise your queries
- Validate which data can be entered
- Escape special characters
A2 Broken Authentication

A diagram showing a web browser communicating with a web server, where sitea.com sends a GET request to fetch content from Site A. The web server returns content that includes DOM + JS, which the browser interprets.
A2  Broken Authentication

- Weak session management
- Credential stuffing
- Brute force
- Forgotten password
- No multi-factor authentication
- Sessions don’t expire
A2  Broken Authentication

Prevention:

- Use good authentication libraries
- Use MFA
- Enforce strong passwords
- Detect and prevent brute force or stuffing attacks
A3 Sensitive Data Exposure

Web Server

Site A
DOM + JS
Site B

Web Browser

GET /

sitea.com

Site A

DOM + JS
Site B

Web Server
A3 Sensitive Data Exposure

- Clear-text data transfer
- Unencrypted storage
- Weak crypto or keys
- Certificates not validated
- Exposing PII or Credit Cards
Data Exposure Demo
A3 Sensitive Data Exposure

Prevention:

● Don’t store data unless you need to!
● Encrypt at rest and in transit
● Use strong crypto
XML External Entities (XXE)
A4 XML External Entities (XXE)

The application accepts XML, and assumes it is safe

```xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE foo [
<!ELEMENT foo ANY >
<!ENTITY xxe SYSTEM "file:///etc/passwd" >]]>
<foo>&xxe;</foo>

Can allow accessing sensitive resources, command execution, recon, or cause denial of service.
XXE Demo
A4 XML External Entities (XXE)

Prevention:

- Avoid XML
- Use modern libraries, and configure them well!
- Validate XML
A5  Broken Access Control

Web Browser
- Site A
- DOM + JS
- Site B

Web Server
- Site A

sitea.com
GET /

DOM

+ JS
A5  Broken Access Control

- Access hidden pages
  http://site.com/admin/user-management
- Elevate to an administrative account
- View other people’s data
  http://site.com/user?id=7
- Modifying cookies or JWT tokens
A5 Broken Access Control

Prevention:

- Use proven code or libraries
- Deny access by default
- Log failures and alert
- Rate limit access to resources
A6 Security Misconfiguration

Web Browser

- Site A
  - DOM + JS
- Site B

Web Server

- Site A

Site A

Site B

Sitea.com

GET /
A6  Security Misconfiguration

- Security features not configured properly
- Unnecessary features enabled
- Default accounts not removed
- Error messages expose sensitive information
Security Misconfiguration

Prevention:

- Have a repeatable build process or “gold master”
- Disable all unused services
- Use tools to review settings
A7 Cross-Site Scripting (XSS)
A7 Cross-Site Scripting (XSS)

HTML mixes content, presentation and code into one string (HTML+CSS+JS)

If an attacker can alter the DOM, they can do anything that the user can do.

XSS can be found using automated tools.
XSS Demo
A7 Cross-Site Scripting (XSS)

Prevention:

● Encode all user-supplied data to render it safe
  Kirk `<script>` => Kirk `<script>`
● Use appropriate encoding for the context
● Use templating frameworks that assemble HTML safely
● Use Content Security Policy
Insecure Deserialization

Web Server

Site A
DOM + JS
Site B

Web Browser

GET /

sitea.com

X

Y
Programming languages allow you to turn a tree of objects into a string that can be sent to the browser.

If you deserialise untrusted data, you may allow objects to be created, or code to be executed.
Deserialisation Demo
Insecure Deserialization

Prevention:

- Avoid serialising and deserialising objects
- Use signatures to detect tampering
- Configure your library safely
- Check out the OWASP Deserialisation Cheat Sheet
A9 Using Components with Known Vulnerabilities
A9 Using Components with Known Vulnerabilities

Modern applications contain a lot of third-party code.

It’s hard to keep it all up to date.

Attackers can enumerate the libraries you use, and develop exploits.
A9 Using Components with Known Vulnerabilities

Prevention:

- Reduce dependencies
- Patch management
- Scan for out-of-date components
- Budget for ongoing maintenance for all software projects
A10  Insufficient Logging & Monitoring

Site A
DOM + JS
Site B

Web Server
SITE

Web Browser

GET /

SIEM
A10 Insufficient Logging & Monitoring

You can’t react to attacks that you don’t know about.

Logs are important for:

- Detecting incidents
- Understanding what happened
- Proving who did something
OWASP Top Ten 2017

A1  Injection
A2  Broken Authentication
A3  Sensitive Data Exposure
A4  XML External Entities (XXE)
A5  Broken Access Control
A6  Security Misconfiguration
A7  Cross-Site Scripting (XSS)
A8  Insecure Deserialization
A9  Using Components with Known Vulnerabilities
A10 Insufficient Logging & Monitoring
Next Steps
Next Steps

- Attend OWASP events
- Search for OWASP Top Ten category names and your framework
  E.g. “C# XSS protection”
- Watch youtube or Pluralsight videos
- Use the terms when discussing bugs with colleagues
- Keep track of which issues affect you the most
- Go beyond the Top Ten
Introduction to the OWASP Top Ten

Kirk Jackson
RedShield
kirk@pageofwords.com
http://hack-ed.com
@kirkj

OWASP NZ
https://www.meetup.com/OWASP-Wellington/
www.owasp.org.nz
@owaspnz

Recordings:
https://goo.gl/a2VSG2