International JavaScript Conference

OWASP TOP TEN 2017

THE TEN MOST CRITICAL WEB APPLICATION SECURITY RISKS

Christian Wenz
info@christianwenz.de
@chwenz

Agenda

1. INJECTION
2. BROKEN AUTHENTICATION
3. CROSS-SITE SCRIPTING (XSS)
4. BROKEN ACCESS CONTROL
5. SECURITY MISCONFIGURATION
6. SENSITIVE DATA EXPOSURE
7. INSUFFICIENT ATTACK PROTECTION
8. CROSS-SITE REQUEST FORGERY (CSRF)
9. USING COMPONENTS WITH KNOWN VULNERABILITIES
10. UNDERPROTECTED APIS

Agenda

1. INJECTION
2. BROKEN AUTHENTICATION
3. SENSITIVE DATA EXPOSURE
4. XML EXTERNAL ENTITIES (XEE)
5. BROKEN ACCESS CONTROL
6. SECURITY MISCONFIGURATION
7. CROSS-SITE SCRIPTING
8. INSECURE DESERIALIZATION
9. USING COMPONENTS WITH KNOWN VULNERABILITIES
10. INSUFFICIENT LOGGING AND MONITORING

#1: Injection

Different kinds of injections

- SQL Injection
- LDAP Injection
- XPath Injection
- Regex Injection

We are actually talking about SQL Injection

OR Mappers help. Parametrized queries/prepared statements, too. If you must, database-specific escaping APIs.

#2: Broken Authentication

HTTP is a stateless protocol

Session management is, essentially, a hack

Different attack vectors

- Session hijacking
- Session fixation
- Other issues: insufficient session timeout, unencrypted passwords, ...

#3: Sensitive Data Exposure

Use encryption throughout

HTTPS only (if possible, enforce it)

- HTTP Strict Transport Security
- „secure" cookie flag

#4: XML External Entities (XEE)

XML processors evaluating an external entity

Is XML still a thing? ;-)

PHP: libxml_disable_entity_loader(true);

Interesting writeup of a PoC: https://sensepost.com/blog/2014/revisting-xxe-and-abusing-protocols/

#5: Broken Access Control

2013 list: Insecure Direct Object References & Missing Function Level Access Control

Access control to data

Access control to functions

#6: Security Misconfiguration

I´m not an admin!

In the days of DevOps, maybe I am

Harden server/OS

Do not send detailled error messages to the client

Use browser security headers

#7: Cross-Site Scripting (XSS)

One of the oldest attacks around

Injecting content (usually JavaScript) into server output

Might bring friends: if there is XSS, it´s very hard to defend against CSRF

Protecting Against XSS

Escape output (< > " ' &)

Use browser protection (X-XSS-Protection)

Use Content Security Policy

#8: Insecure Deserialization

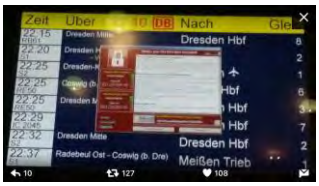
PHP: calls to unserialize() re-create classes used in the payload

Magic methods might be executed

- __construct()
- __wakeup()
- __destruct()
- ... and some more

Rule #1: Validate input

#9: Using Components with Known Vulnerabilities



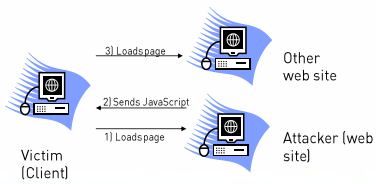
https://twitter.com/jacobhusted69/status/863151760118083584/photo/1

#10: Insufficient Logging & Monitoring

Do log!

Do look at the logs!

#11: Cross-Site Request Forgery (CSRF)



Thank you!

- @chwenz