How to Hack Your Web Application

Agenda

- Introduction
- „Hack Me If You Can“
- OWASP Top 10
- Break
- Hacking a Vulnerable Web Application
- Fixing the Web Application
- Ensuring Secure Software
- Next Steps

Who Am I?

- Reading passwords through JavaScript when 16 years old
- Security focus during studies
- Theses about Android Security & Security Dashboards
- Co-Founder and former Head of Product of Crashtest Security (acquired by Veracode)
- Worked with and contributed to several OSS security tools
- Infrastructure Team Lead & Information Security Officer ottonova

Janosch Braukmann

Team Lead & ISO at

ottonova

Disclaimer

Hacking is like Sex…

- Exciting
- Needs some Stamina
- Few people talk about it openly
- There are many variations
- Better if you are not alone

Bust most importantly:

- **You need consent!**

Hack Me If You Can

Hack Me If You Can

https://janosch-braukmann.de:8000/

OWASP Top 10 (2021)

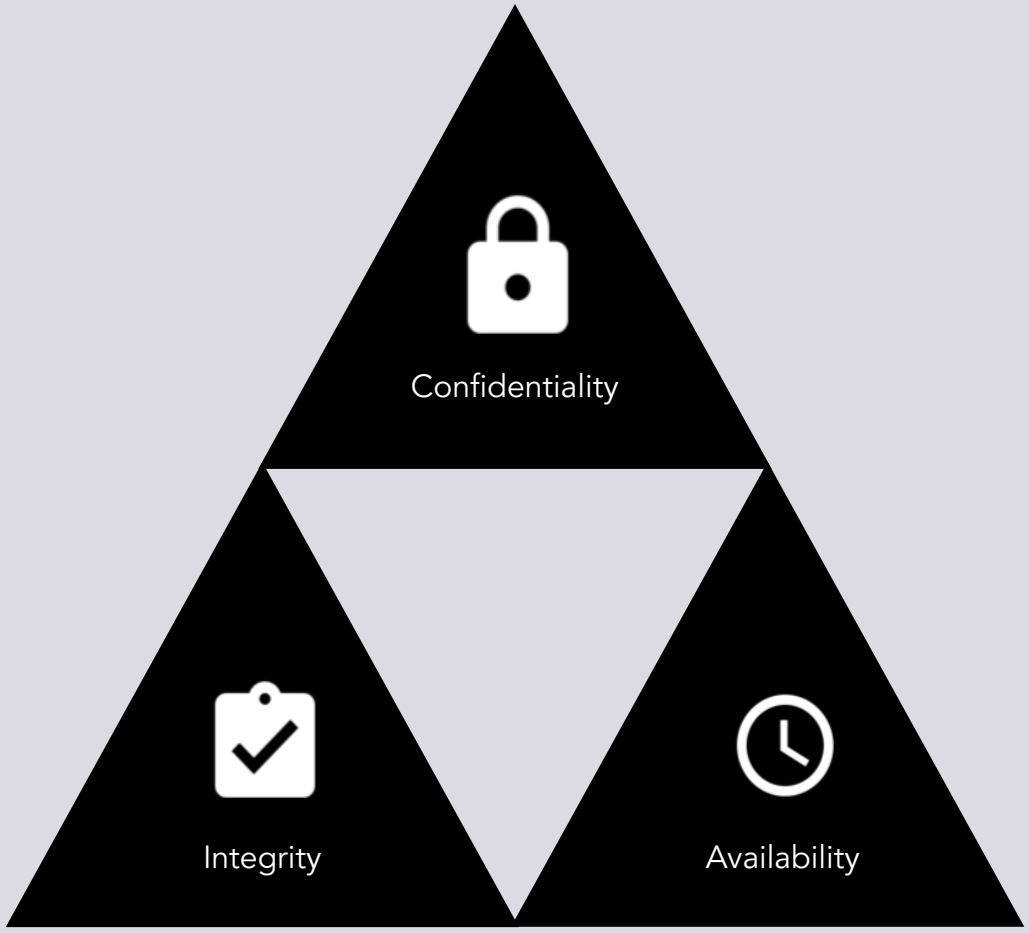
OWASP Top 10 (2021)

“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.”

OWASP Top 10 Website

- The latest release is from September 2021
- The next release is expected in 2025

IT Security Objectives



A01:2021 – Broken Access Control

Description

Access control enforces policies such that users cannot act outside of their intended permissions. Failures typically lead to **unauthorized information disclosure, modification, or destruction** of all data **or performing a business function** outside the user's limits.

Example

A user can **modify an URL parameter** to get access to information that belongs to a different account:
https://example.com/app/accountInfo?acct=notmyacct

A02:2021 – Cryptographic Failures

Description

Failures related to cryptography such as **weak hashes, re-used keys** or the complete **lack of encryption**, often lead to the exposure of sensitive data.

Example

Personal Experience: **Session ID generation used current timestamp as seed**. Therefore, the session ID was not random and could be brute forced in 2-3 minutes if you knew the approximate time a user logged in.

A03:2021 – Injection

Description

Injection vulnerabilities happen when user-supplied data is **not validated, filtered or sanitized** by the application so that source code can be provided by a user and is run in the context of the application.

Example

Live Demo: **SQL Injection, Code Execution & XSS**

A04:2021 – Insecure Design

Description

Insecure design is a broad category representing different weaknesses, expressed as "**missing or ineffective control design**". Insecure design is not the source for all other Top 10 risk categories. There is a difference between insecure design and insecure implementation.

Example

The **Luca App Keyrings** for people without Smartphone allowed checking in by presenting the QR-code to readers at different locations. A feature allowed to see the history of the visited places if you have the QR-code (i.e., your own history). An attacker at a check-in station will see the QR-code by design and can get access to the **victims' location history**.

A05:2021 – Security Misconfiguration

Description

The application might be vulnerable due to security misconfiguration if appropriate **security hardening is missing** or **improperly configured**, **unnecessary features** or **default accounts** are enabled, **stack traces** are enabled, or **debug functions** are still active.

Example

The US Department of Defense had the X-XSS-Protection header set to “DENY” which is not valid. It should have been “1; mode=block” instead. The “DENY” value is a correct setting for the X-Frame-Options header instead. This increased the attack surface for XSS attacks.

A06:2021 – Vulnerable & Outdated Components

Description

An application is likely to be vulnerable if there exists a vulnerability in any used software. This becomes more likely for **outdated or unsupported software**. This affects the OS, web/application server, database management system (DBMS), applications, APIs and all components, runtime environments, and libraries.

Example

Heartbleed is a bug in the **OpenSSL implementation** of the TLS heartbeat extension that could be exploited to leak memory content containing certificate secrets.

A07:2021 – Identification & Authentication Failures

Description

Confirmation of the user's identity, authentication, and session management is critical to protect against authentication-related attacks. There are authentication weaknesses if **default, weak or well-known passwords** are used, the passwords are **brute-forceable**, credentials can be leaked, passwords are **not hashed**, or session IDs are handled insecurely.

Example

Personal Experience: Got access to complete user information including **unencrypted passwords** while working on a **database** migration for a social network

A08:2021 – Software & Data Integrity Failures

Description

Software and data integrity failures relate to code and infrastructure that does not protect against integrity violations. An example of this is where an application relies upon plugins, libraries, or modules from **untrusted sources**, repositories, and content delivery networks (CDNs) without sufficient integrity verification.

Example

In June 2024, an attacker took over the CDN distribution of the **JavaScript package Polyfill** and **redirected** website visitors to **sports betting websites**.

A09:2021 – Security Logging & Monitoring Failures

Description

This category is to detect, escalate, and respond to **active breaches**. Without logging and monitoring, breaches cannot be detected.

Example

In November 2018, **Marriott** detected that one of their subsidiaries has been attacked. This attack started even before Marriot acquired this company and went **unnoticed for more than four years**.

A10:2021 – Server Side Request Forgery (SSRF)

Description

SSRF flaws occur whenever a web application is **fetching a remote resource without validating** the user-supplied URL. It allows an attacker to coerce the application to send a crafted request to an unexpected destination, even when protected by a firewall, VPN, or another type of network access control list (ACL).

Example

Personal Experience: Grab metrics data by accessing **internal endpoints** providing data to be scraped by **Prometheus**.

Vulnerable Web Application

Hack Me If You Can

http://janosch-braukmann.de:8080/

I consent, but for the duration of this workshop only!

Hack Me If You Can

SQL Injection

' OR '1' = '1

XSS

<script>alert(“You got Hacked”);</script>

Hack Me If You Can

https://github.com/Phylu/vulnerable-click-game

Let’s ramp up the game by checking out the source code!

Hack Me If You Can

Do you find the exploited vulnerabilities in the code?

Do you find further vulnerabilities to exploit?

Ensuring Secure Software

Ensuring Secure Software

Malicious User Story

A Malicious User Story in the context of software development is a user story written from the perspective of an attacker. It outlines the attacker's goal and motivation in exploiting a system vulnerability. This helps developers understand potential security threats and build appropriate countermeasures.

It typically follows the format:

"As a [malicious user role], I want to [perform a malicious action] so that [achieve a malicious goal]."

Ensuring Secure Software

Malicious User Story

As a malicious player, I want to manipulate my score in the click game so that I can achieve a high ranking on the leaderboard without actually earning it, deceiving other players and gaining an unfair advantage.

“Flipped” Acceptance Criteria

- The game server must calculate the score. It must not be possible to submit arbitrary scores.
- The game server must implement rate limiting to prevent excessive submissions from a single user.
- Any discrepancies between client submitted actions and server calculated possible actions must prevent the score from being updated.

Ensuring Secure Software

Malicious User Story

Propose additional Malicious User Stories for my Game

“Flipped” Acceptance Criteria

What acceptance criteria should I use to ensure the malicious user is unsuccesful?

Next Steps

Next Steps

Step 1: In your day-to-day work

- Figure out your crown jewels
- Protect your crown jewels
- If unsure start with the OWASP Top 10

Step 2: Grow your security network

- Connect with your security interested peers
- Feel free to start with me:
    - https://www.linkedin.com/in/janoschbraukmann
    - https://github.com/Phylu

Step 3: Seriously get into security

- Reach out to me for a Security or DevOps engineer position