




# You've Been Hacked

An (Interactive) Course on Web Security

Paul Duplys

 @duplys

 duplys

 [linkedin.com/in/paulduplys/](https://www.linkedin.com/in/paulduplys/)



man slides

(Interactive) course on web security based on Carsten Eiler's book "You've Been Hacked".

Who is the audience? How can I use the book? How can I explore the app?

whoami

short intro/bio.

# 0x0: Preliminaries

# Finding Vulnerabilities in Web Applications

In a nutshell, **to find vulnerabilities in your web application**, ...

1. ... test various values for parameters used by the web application and see what happens (conceptually similar to fuzzing)
2. ... check web application code for bugs that may lead to security vulnerabilities (typically missing checks of input values or missing countermeasures against certain types of attacks)

The [Open Web Application Security Project \(OWASP\)](#) maintains a list of Top 10 vulnerabilities in web applications.

### 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, healthcare, and PII. Attackers may steal or modify such weakly protected data to conduct credit card fraud, identity theft, or other crimes. Sensitive data may be compromised without extra protection, such as encryption at rest or in transit, and requires special precautions when exchanged with the browser.
4. **XML External Entities (XXE).** Many older or poorly configured XML processors evaluate external entity references within XML documents. External entities can be used to disclose internal files using the file URI handler, internal file shares, internal port scanning, remote code execution, and denial of service attacks.
5. **Broken Access Control.** Restrictions on what authenticated users are allowed to do are often not properly enforced. Attackers can exploit these flaws to access unauthorized functionality and/or data, such as access other users' accounts, view sensitive files, modify other users' data, change access rights, etc.

# Conventions

- ▶ Alice, Bob: legitimate users
- ▶ Eve: malicious user, attacker
- ▶ Server: web server running a web application
- ▶ Client: web browser (or computer running the web browser)



# GitHub Repo

- ▶ <https://github.com/duplys/youve-been-hacked>
- ▶ Dockerfile & setup instructions
- ▶ Write-ups
- ▶ Code

# Building Docker Image

- ▶ Running the demo web application in a Docker container is the easiest way to get started
- ▶ Docker dir contains `Dockerfile` for the vulnerable web application
- ▶ Build the container using `make build` or `docker-compose`

# Starting Docker Containers

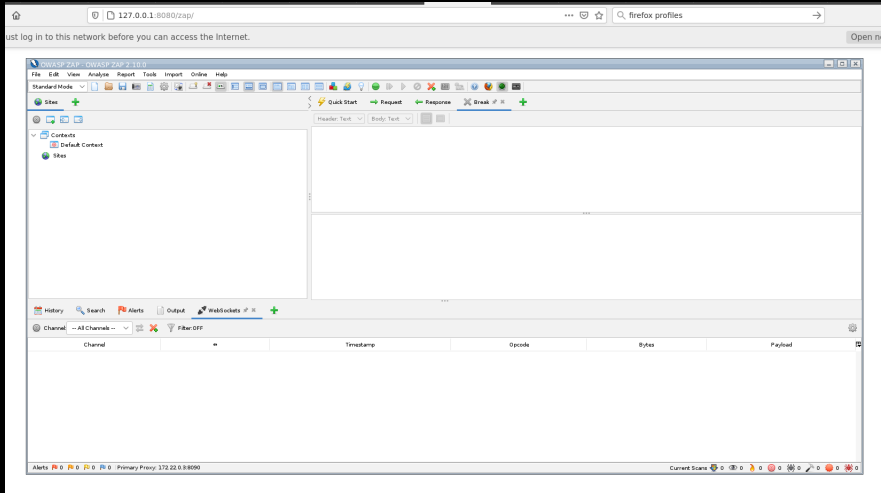
- ▶ You'll need an empty directory `tmp` in the `Docker` dir
- ▶ In `Docker` dir, run `$ docker-compose up`

# Setting up ZAP

You need to activate ZAProxy (ZAP), configure your browser to proxy via ZAP and import the public ZAP Root certificate (see <https://www.zaproxy.org/docs/docker/webswing/> for details). Do the following steps:

- ▶ Go to `http://127.0.0.1:8080/zap/`
- ▶ You'll see how the ZAP Web UI starts
- ▶ Start a ZAP session, choose "don't want to persist"
- ▶ Select "Update All" in the "Manage Add-ons" window

# Setting up ZAP



# Setting up ZAP

- ▶ Go to "Tools" → "Options" → "Dynamic SSL Certificates" → "Save"
- ▶ Save ZAP certificate on your host and import it into your browser.
- ▶ Read off the ZAP's IP address and port number at the bottom of ZAP's window
- ▶ Configure your web browser to use that IP/port as proxy

# Accessing the Vulnerable Web App through ZAP

- ▶ Look up the vulnerable app container IP address (for docker-compose)
- ▶ Run `docker container inspect docker_vulnapp_1` and look for `IPAddress:` under `Networks:`
- ▶ If the container's IP address on your machine is `x.y.z.w`, you can access the vulnerable web app under `http://x.y.z.w/daten/kapitel1.html`

## Cleaning Up

- ▶ Run `$ docker-compose down`



