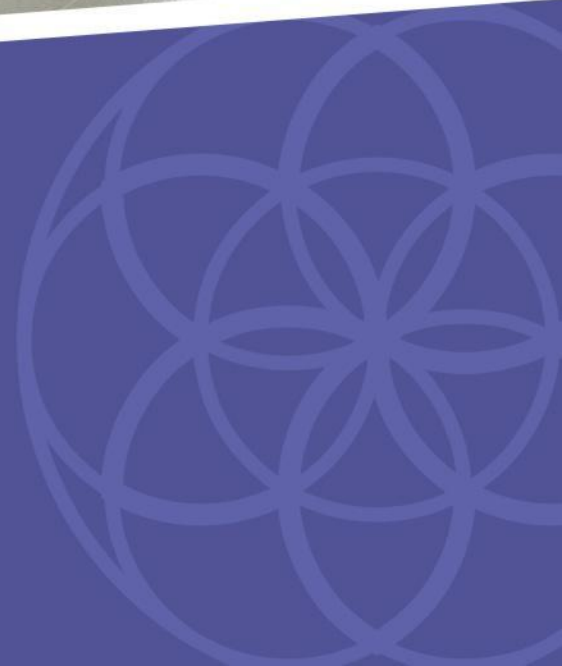




# Security Knowledge Framework



Making the web secure by design





Glenn ten Cate

Security Engineer  
@Schuberg Philis

Riccardo ten Cate

Security Researcher





# Agenda

- Why SKF
- What you will get/learn
- Stages of development
- Intro & how to
- Hands on - testing
- ASVS
- Getting involved
- Questions?



# Why S.K.F.

- Missing security at design
- Missing defensive coding approach
- Missing security guidance
- Missing security requirements
- Security is hard
- Security information should be available for everybody





# What you will get/learn

- Guide to secure programming  
Do security by design, not implementing security afterwards.
- Security awareness  
It'll inform you about threats even before you wrote a single line of code.
- Central place for security reference  
Provides information applicable for your specific needs on the spot.



# Stages of development

- Pre development stage

Here we detect threats before hand and we provide developers with secure development patterns as well as providing feedback and solutions on how to handle their threats.

- Post development stage

By means of checklists we guide developers through a process where we harden their application infrastructure and functions by providing feedback and solutions



Security Knowledge Framework

Login



Supported by OWASP

Demo:  
<https://securityknowledgeframework.org/demo.php>



# Hands on - testing

- Visit the web application

Check out this new web application developed by SloppyCode INC.  
Visit the website on <http://192.168.8.133>

- Source of SloppyCode INC available at: <https://github.com/blabla1337/skf-workshop>

- Learn the hackers mindset

In order to defend we first must think like an attacker.





# Some hacking exercises

- Cross site scripting
- Reflective file download
- Directory/path traversal
- Identifier injection



# Cross site scripting

- Attack vector:

If you can imagine it, you can do it. Under the right conditions XSS can do almost anything. For example: <http://beefproject.com/>

- Objective

Due to the variety of attacks and encodings there are a lot of different possible attack strings to execute an XSS attack. For this exercise though you will only need a simple : `<script>alert(1337);</script>`. Now find the XSS injection in the application.



# Reflective file download

- **Attack vector:**

Reflective file download occurs whenever an attacker can "forge" a download through misconfigurations in your "disposition" and "content type" headers. This attack can lead to compromising the victim's entire workstation.

- **Objective**

Try tampering with the filenames and see if you can forge your own download on the web application. Try to make your attack access a local file on your machine.



# Directory/path traversal

- **Attack vector:**

A Path Traversal attack aims to access files and directories that are stored outside the web root folder. By browsing the application, the attacker looks for absolute links to files stored on the web server. By manipulating variables that reference files with dot-dot-slash (../); sequences and its variations, it may be possible to access arbitrary files and directories stored on file system.

- **Objective**

Try to read the “secrets.txt” from the webserver by doing a path traversal attack.



# Identifier injection

- **Attack vector:**

An application uses parameters in order to process data. These parameters can also be used to assign certain roles and retrieve content corresponding with those parameters. Whenever these parameters are tampered with an attacker might gain access to other users data.

- **Objective**

Watch the applications operation and look out for identifiers like user= or id= when you are searching for identifier injection. Let's see if we can read some user data not intended for our eyes.



# OWASP ASVS

- ASVS lvl1 Opportunistic

It adequately defends against application security vulnerabilities that are easy to discover.

- ASVS lvl2 Standard

It adequately defends against prevalent application security vulnerabilities whose existence poses moderate-to-serious risk.

- ASVS lvl3 Advanced

It adequately defends against all advanced application security vulnerabilities, and also demonstrates principles of good security design.





# Getting involved?

- Website  
[secureby.desgin](https://secureby.desgin)

Together we can make it big, strong and helpful!



# Questions?

