OWASP Secure Coding Practices Quick Reference Guide

Justin Clarke
justin.clarke@owasp.org

OWASP
AppSec Asia Pacific
13 April 2012

The OWASP Foundation
http://www.owasp.org

Some Background

- Goal: Build a secure coding kick-start tool, to help development teams quickly understand secure coding
- Originally developed for use inside The Boeing Company
- July 2010, Boeing assigned copyright to OWASP
- August 2010, project goes live on owasp.org
- November 2010, SCP v2 goes live (current stable version)

Project Structure / Localizations

- English – Keith Turpin (Project leader)
  - Korean
  - Portuguese
  - Brazilian Portuguese
  - Spanish

- https://www.owasp.org/index.php/OWASP_Secure_Coding_Practices_-_Quick_Reference_Guide

Guide Overview

- Technology agnostic coding practices
- What to do, not how to do it
- Compact, but comprehensive checklist format
- Focuses on secure coding requirements, rather then on vulnerabilities and exploits
- Includes a cross referenced glossary to get developers and security folks talking the same language

Sections of the Guide

- The bulk of the document is in the checklists, but it contains all of the following:
  - Table of contents
  - Introduction
  - Software Security Principles Overview
  - Secure Coding Practices Checklist
  - Links to useful resources
  - Glossary of important terminology

Checklist Sections - *Only 9 pages long*

- Input Validation
- Output Encoding
- Authentication and Password Management
- Session Management
- Access Control
- Cryptographic Practices
- Error Handling and Logging
- Data Protection
- Communication Security
- System Configuration
- Database Security
- File Management
- Memory Management
- General Coding Practices

Checklist Practices

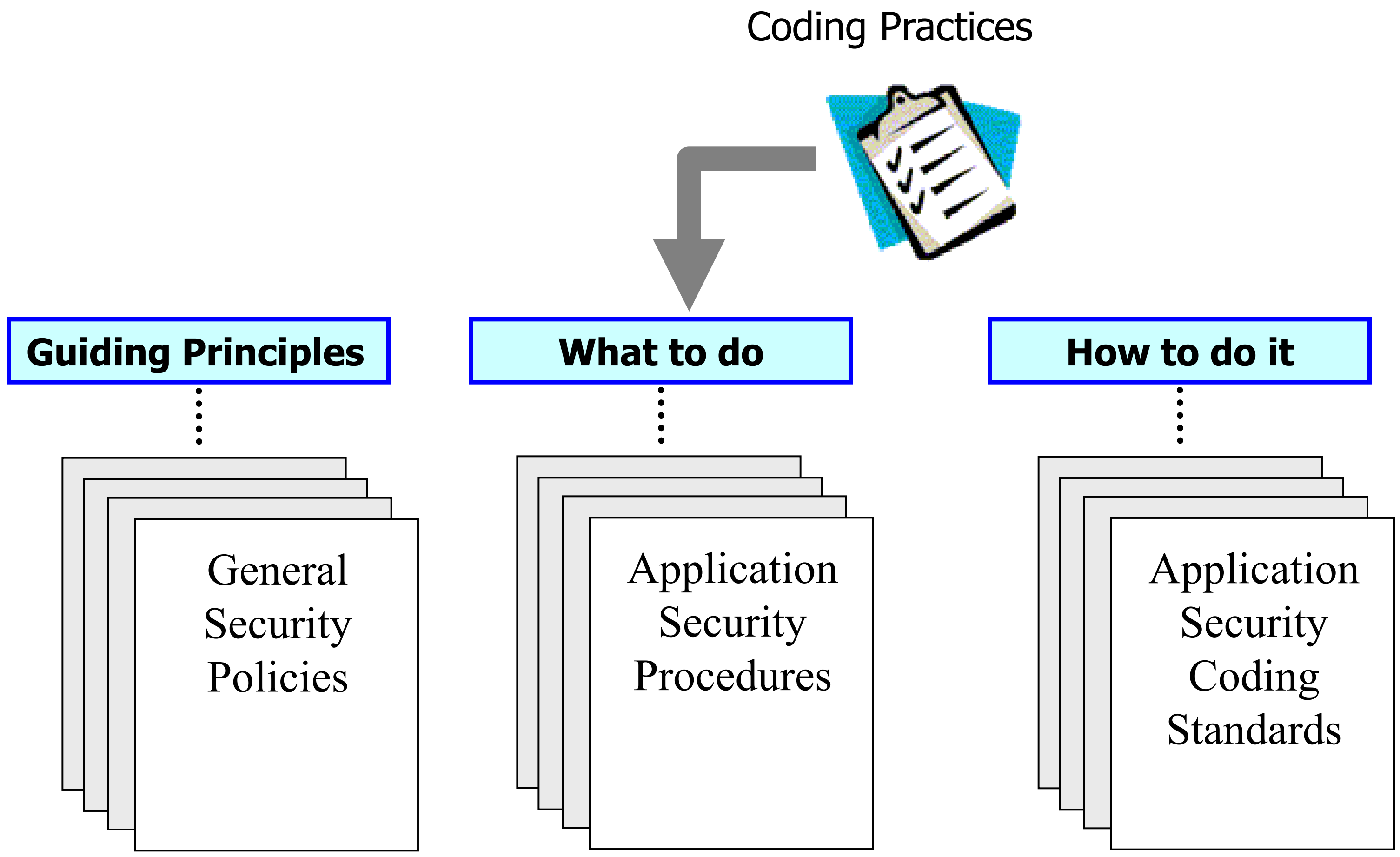
- Short and to the point
- Straight forward "do this" or "don't do that"
- Does not attempt to rank the practices
- Some practices are conditional recommendations that depend on the criticality of the system or information
- The security implications of not following any of the practices that apply to the application, should be clearly understood

Extract - Database Security

- Use strongly typed *parameterized queries*
- Utilize input validation and output encoding and be sure to address meta characters. If these fail, do not run the database command
- Ensure that variables are strongly typed
- The application should use the lowest possible level of privilege when accessing the database
- Use secure credentials for database access
- Do not provide connection strings or credentials directly to the client. If this is unavoidable, encrypted them
- Use stored procedures to abstract data access
- Close the connection as soon as possible
- Remove or change all default database administrative passwords. Utilize strong passwords/phrases or implement multi-factor authentication
- Turn off all unnecessary database functionality (e.g., unnecessary stored procedures or services, utility packages, install only the minimum set of features and options required (surface area reduction))

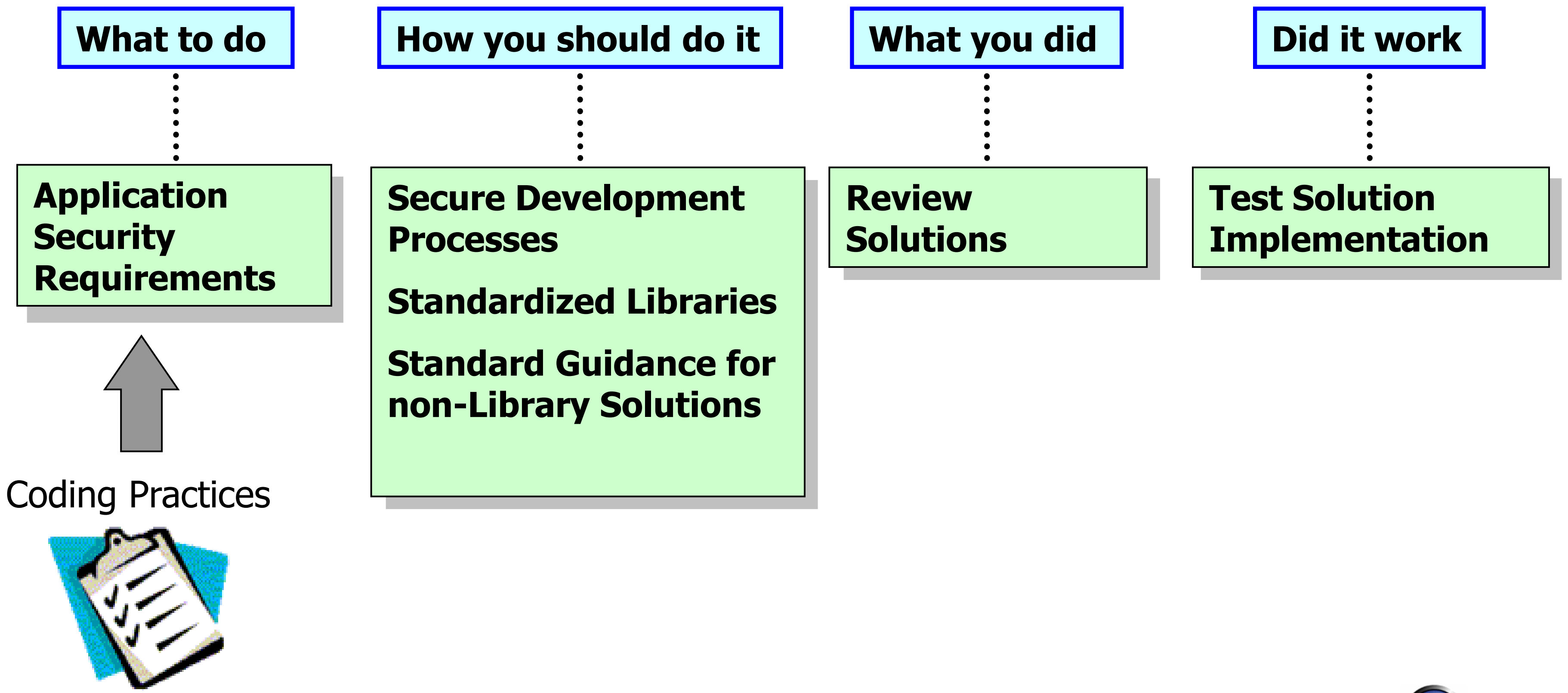
Using the guide

- Scenario #1: Developing Guidance Documents


Coding Practices
Guiding Principles
What to do
How to do it
General Security Policies
Application Security Procedures
Application Security Coding Standards

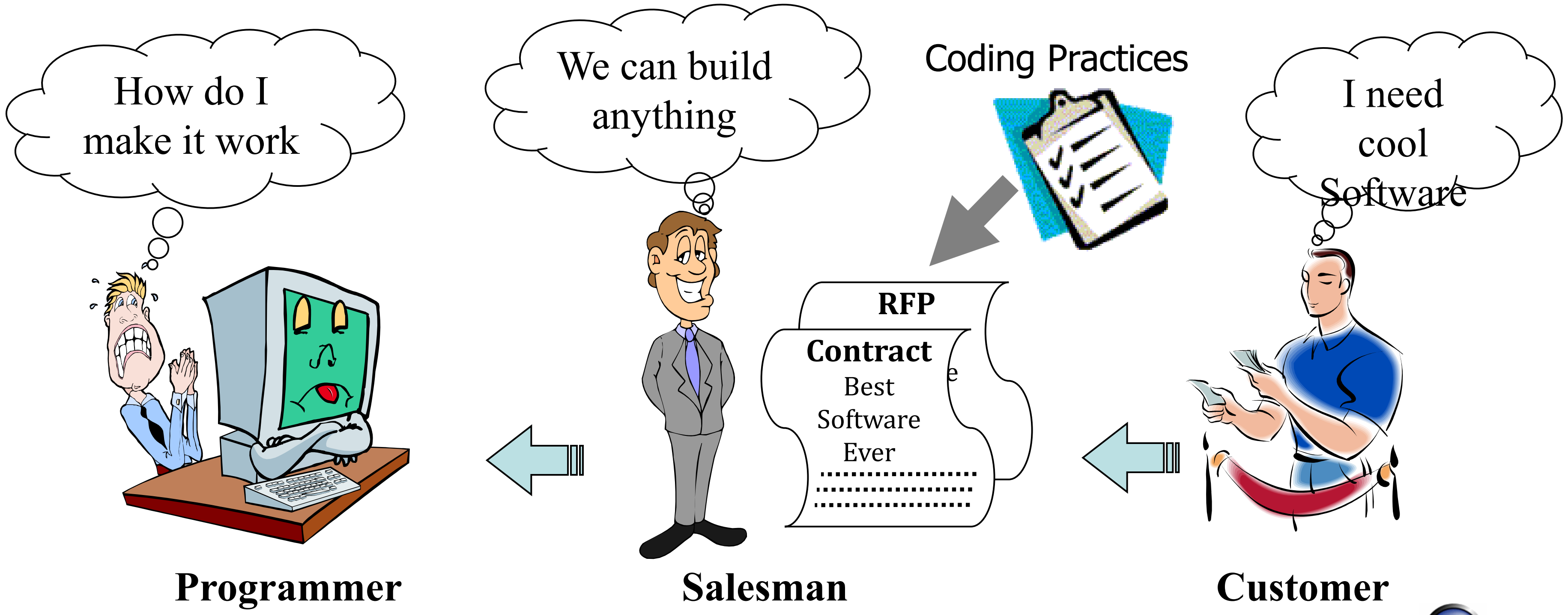
Using the guide *continued*

- Scenario #2: Support Secure Development Lifecycle


What to do
How you should do it
What you did
Did it work
Application Security Requirements
Secure Development Processes
Standardized Libraries
Standard Guidance for non-Library Solutions
Review Solutions
Test Solution Implementation
Coding Practices

Using the guide *continued*

- Scenario #3: Contracted Development
  - Identify security requirements to be added to outsourced software development projects.
  - Include them in the RFP and Contract



Summary

- Makes it easier for development teams to quickly understand secure coding practices
- Assists with defining requirements and adding them to policies and contracts
- Provides a context and vocabulary for interactions with security staff
- Serves as an easy desk reference

A Secure Development Framework

Guidance on implementing a secure software development framework is beyond the scope of the Quick reference Guide, however the following OWASP projects can help:

- Implement a secure software development lifecycle
  - OWASP CLASP Project
  - OpenSAMM
- Establish secure coding standards
  - OWASP Development Guide Project
- Build a re-usable object library
  - OWASP Enterprise Security API (ESAPI) Project
- Verify the effectiveness of security controls
  - OWASP Application Security Verification Standard (ASVS) Project)
- Establish secure outsourced development practices including defining security requirements and verification methodologies in both the RFP and contract
  - OWASP Legal Project

Questions