42 minutes to secure your code….

JUG Summer Camp
18 Septembre 2015
LaRochelle - France

OWASP
The Open Web Application Security Project

Sébastien Gioria
sebastien.gioria@owasp.org
OWASP France Leader &
Evangelist

Agenda

OWASP
The Open Web Application Security Project

- OWASP ?
- Secure Coding ?
- 10 simply principles to secure code
- Controlling code security
- ~~Q&A~~ Beer ☺

http://www.google.fr/#q=sebastien gioria

OWASP
The Open Web Application Security Project

- Innovation and Technology @Advens && Application Security Expert
- OWASP France Leader & Founder & Evangelist,
- OWASP ISO Project & OWASP SonarQube Project Leader
- Application Security group leader for the CLUSIF
- Legal expert for Cour of Appeal of Poitiers
- Proud father of youngs kids trying to hack my digital life.

Twitter :@SPoint/@OWASP_France/@AppSecAcademy

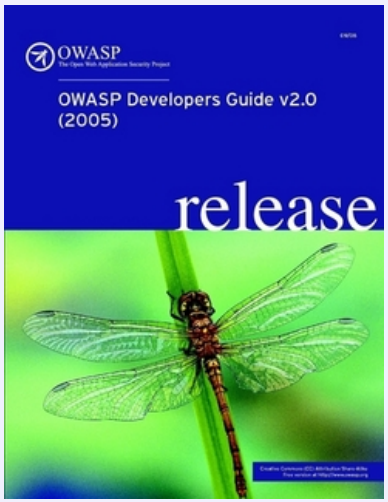
OWASP

The Open Web Application Security Project

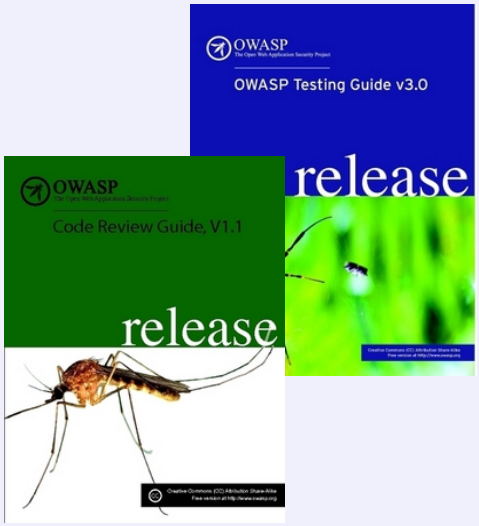
Learn

Contract

Design



Code



Testing

Maturity

OWASP publications !

OWASP
The Open Web Application Security Project

- Publications :
  - Top10 Application Security Risk ; bestseller
  - Testing Guide ; second bestseller
  - OWASP Cheat Sheets !!!
  - Application Security Verification Standard ; not the best well known document
  - OpenSAMM : improve your application security
  - OWASP Secure Contract Annex
  - OWASP Top10 for … (mobile, cloud, privacy, …)
- Tools / API
  - OWASP Zed Attack Proxy ; replace WebScarab with a lot of new functionalities
  - OWASP ESAPI : API for securing your Software
  - OWASP AppSensor ; a IDS/IPS in the heart of your software
  - OWASP Cornucoppia ; application security play with cards
  - OWASP Snake and ladder : play Top10

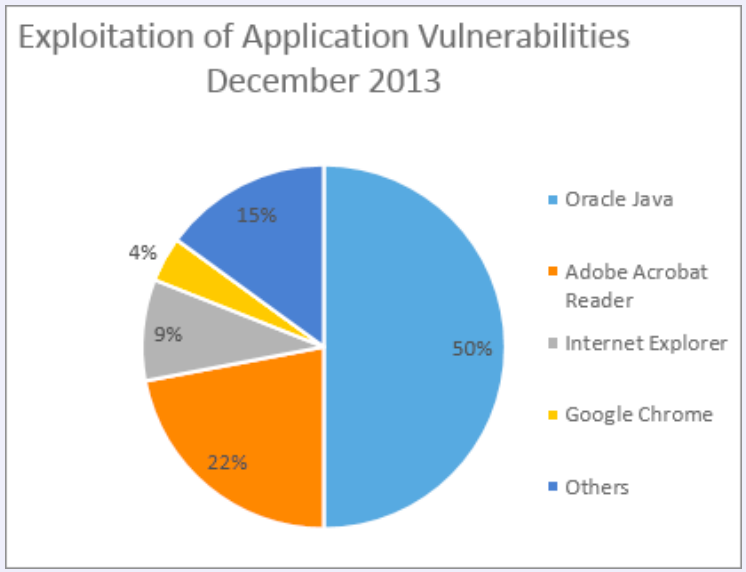
and many more....

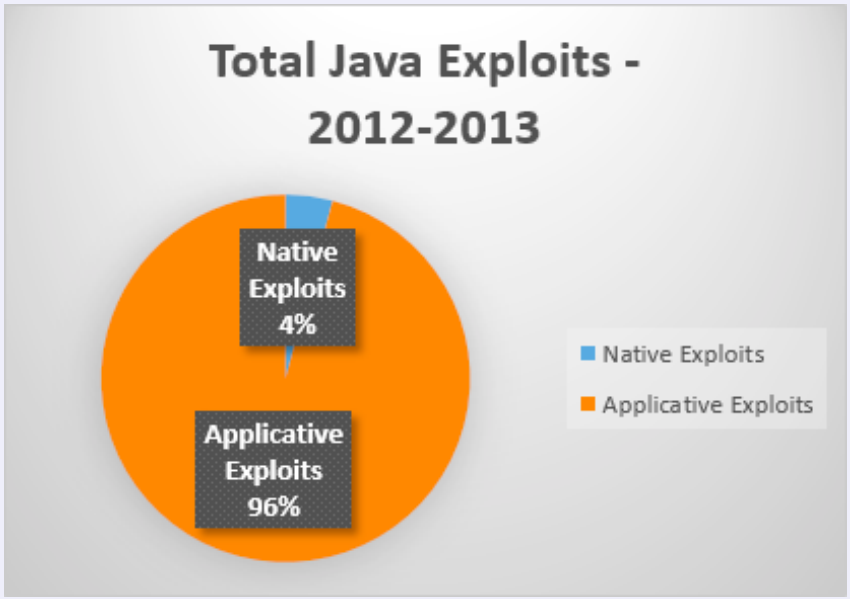
Secure Coding ?

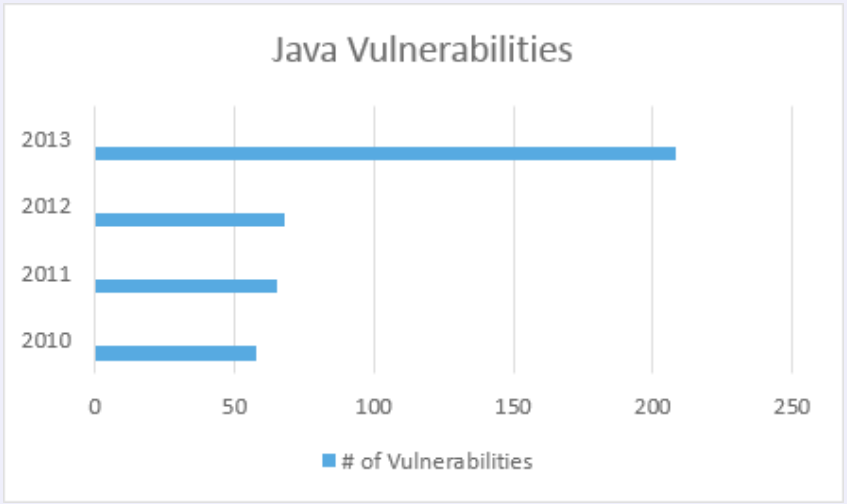
Yes, this is you !

Some Stats....

OWASP
The Open Web Application Security Project



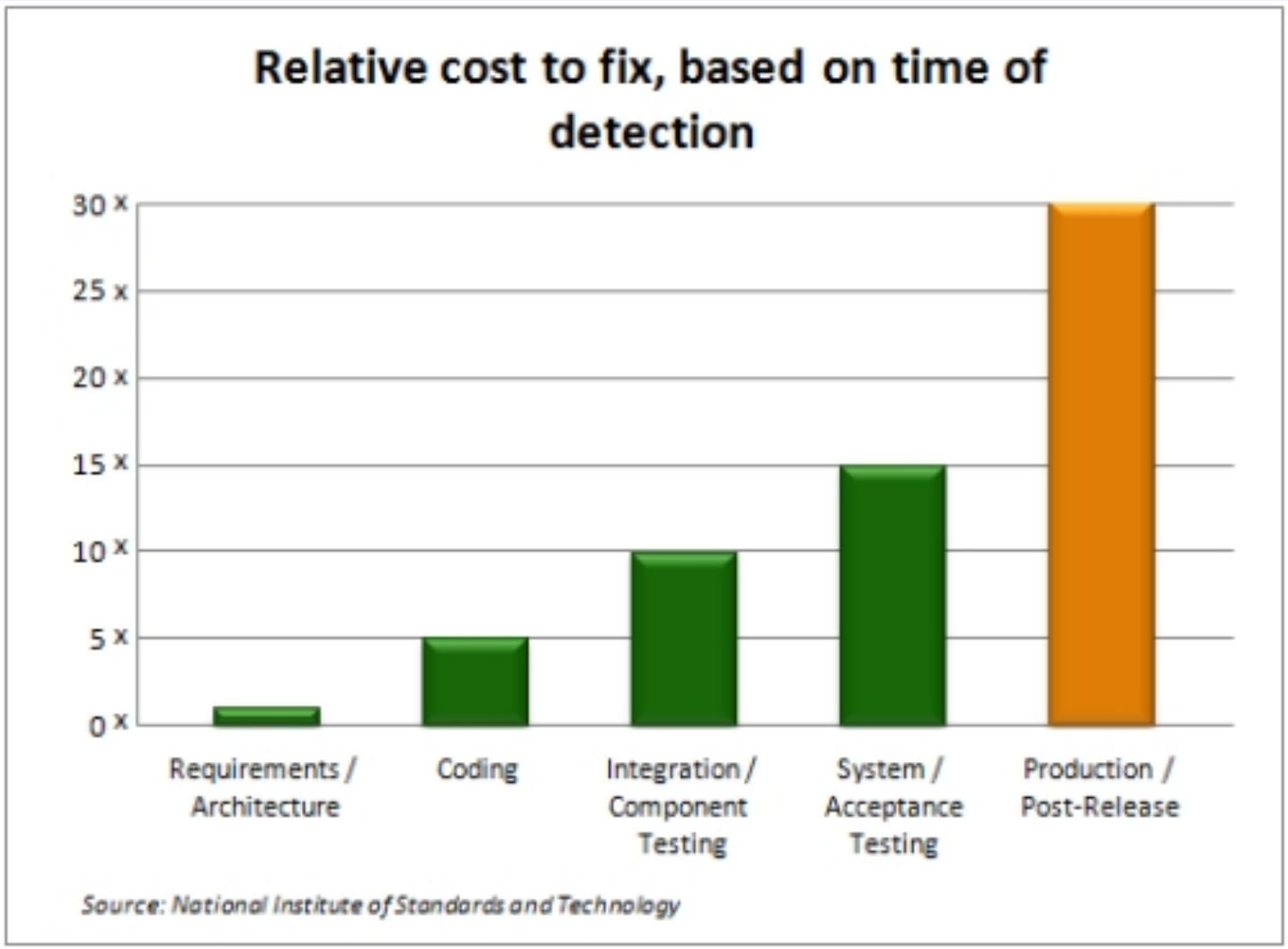




Hackers are clever

Money, Money, Money

OWASP
The Open Web Application Security Project

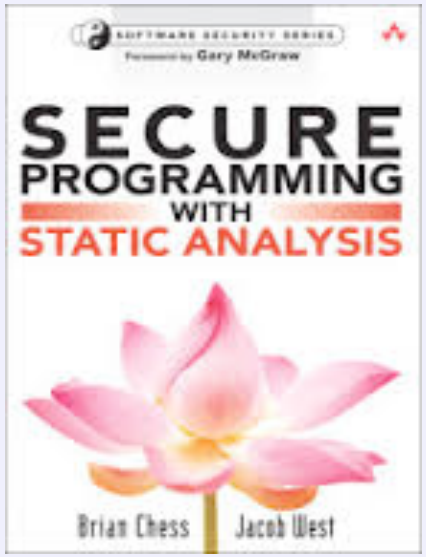


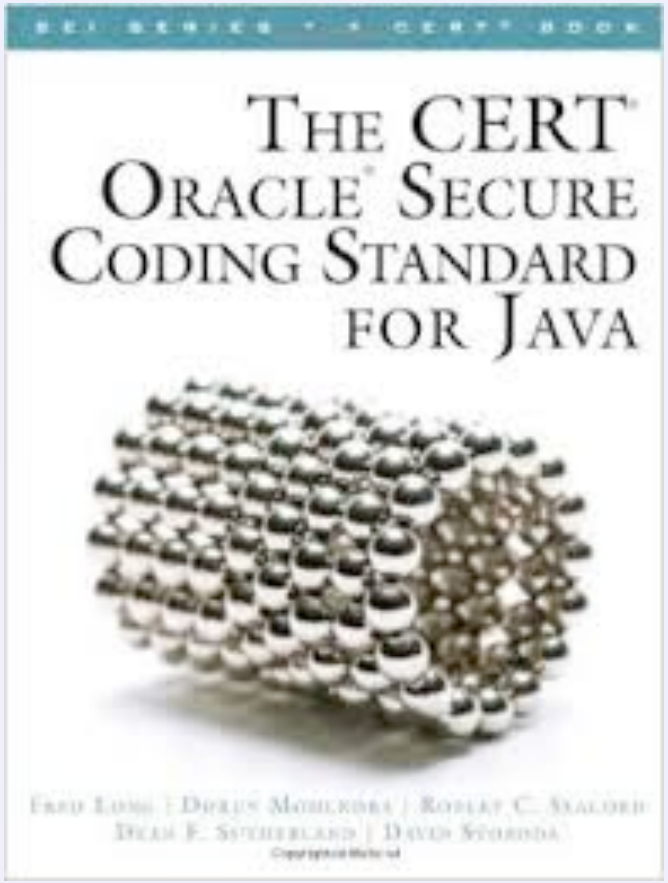
Be accurate

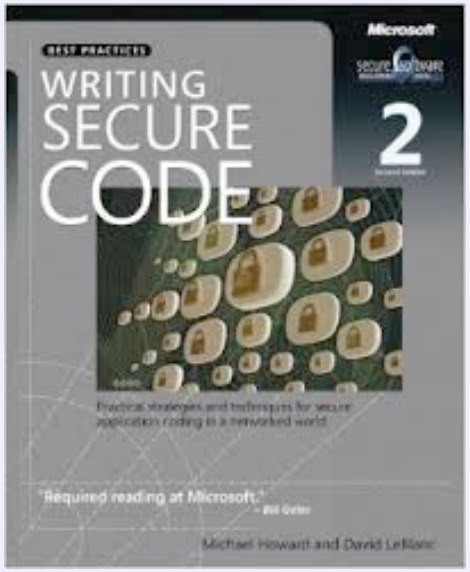
OWASP
The Open Web Application Security Project

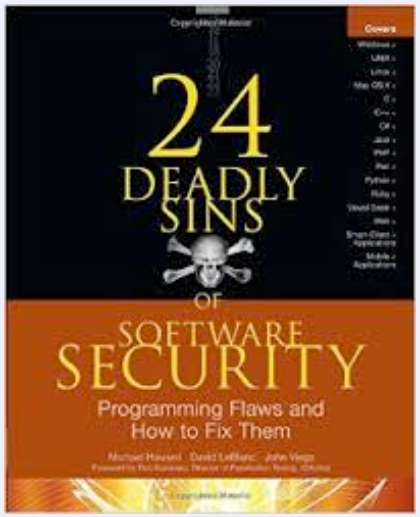
OWASP
The Open Web Application Security Project

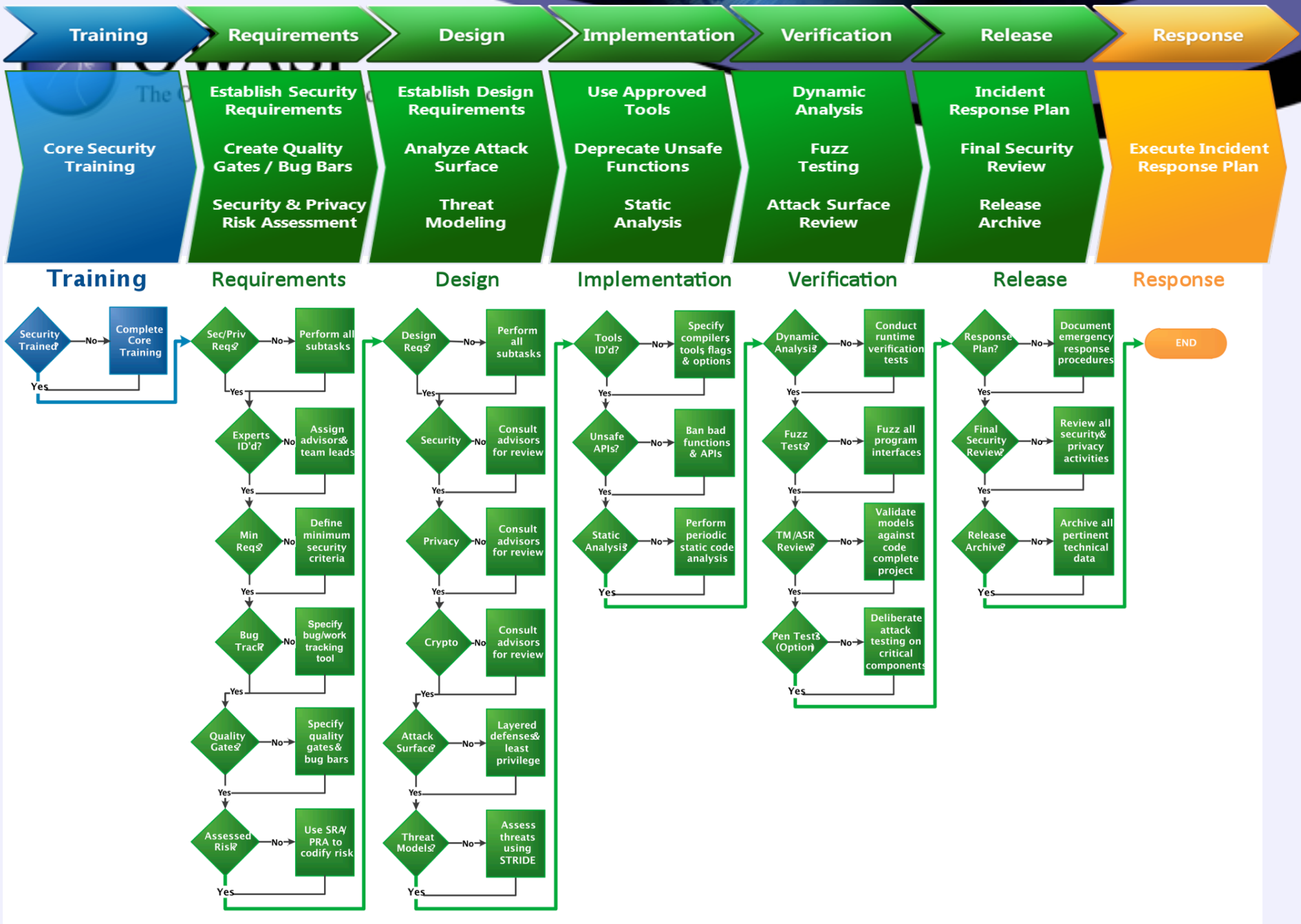

OWASP
The Open Web Application Security Project


Foreword by Gary McGraw
SECURE
PROGRAMMING
WITH
STATIC ANALYSIS
Brian Chess
Jacob West


THE CERT
ORACLE SECURE
CODING STANDARD
FOR JAVA


Microsoft
WRITING
SECURE
CODE
2
"Required reading at Microsoft."
Michael Howard and David LeBlanc


24
DEADLY
SINS
OF
SOFTWARE
SECURITY
Programming Flaws and
How to Fix Them


Training
Requirements
Design
Implementation
Verification
Release
Response
Core Security Training
Establish Security Requirements
Create Quality Gates / Bug Bars
Security & Privacy Risk Assessment
Establish Design Requirements
Analyze Attack Surface
Threat Modeling
Use Approved Tools
Deprecate Unsafe Functions
Static Analysis
Dynamic Analysis
Fuzz Testing
Attack Surface Review
Incident Response Plan
Final Security Review
Release Archive
Execute Incident Response Plan
Security Trained?
No
Complete Core Training
Yes
Sec/Priv Reqs?
Perform all subtasks
Experts ID'd?
Assign advisors& team leads
Min Reqs?
Define minimum security criteria
Bug Track?
Specify bug/work tracking tool
Quality Gates?
Specify quality gates& bug bars
Assessed Risk?
Use SRA/ PRA to codify risk
Design Reqs?
Perform all subtasks
Security
Consult advisors for review
Privacy
Consult advisors for review
Crypto
Consult advisors for review
Attack Surface?
Layered defenses& least privilege
Threat Models?
Assess threats using STRIDE
Tools ID'd?
Specify compilers, tools, flags & options
Unsafe APIs?
Ban bad functions & APIs
Static Analysis?
Perform periodic static code analysis
Dynamic Analysis?
Conduct runtime verification tests
Fuzz Tests?
Fuzz all program interfaces
TM/ASR Review?
Validate models against code complete project
Pen Test& (Optional)
Deliberate attack testing on critical components
Response Plan?
Document emergency response procedures
Final Security Review?
Review all security& privacy activities
Release Archive?
Archive all pertinent technical data
END


OWASP
The Open Web Application Security Project
KEEP
CALM
AND
SECURE
YOUR APP


OWASP
The Open Web Application Security Project
HI, THIS IS YOUR SON'S SCHOOL. WE'RE HAVING SOME COMPUTER TROUBLE.
OH, DEAR – DID HE BREAK SOMETHING?
IN A WAY–
DID YOU REALLY NAME YOUR SON Robert'); DROP TABLE Students;-- ?
OH. YES. LITTLE BOBBY TABLES, WE CALL HIM.
WELL, WE'VE LOST THIS YEAR'S STUDENT RECORDS. I HOPE YOU'RE HAPPY.
AND I HOPE YOU'VE LEARNED TO SANITIZE YOUR DATABASE INPUTS.


```
@Inject
EntityManager  em;

@Transactional
public User updateEmail(String username, String email) {
        TypedQuery<User>  q = em.createQuery(
                String.format("update  Users set email '%s' where username = '%s'", email, username),
                User.class);
        return q.getSingleResult();
}
```

Parametrize ! don't Jeopardize !!!

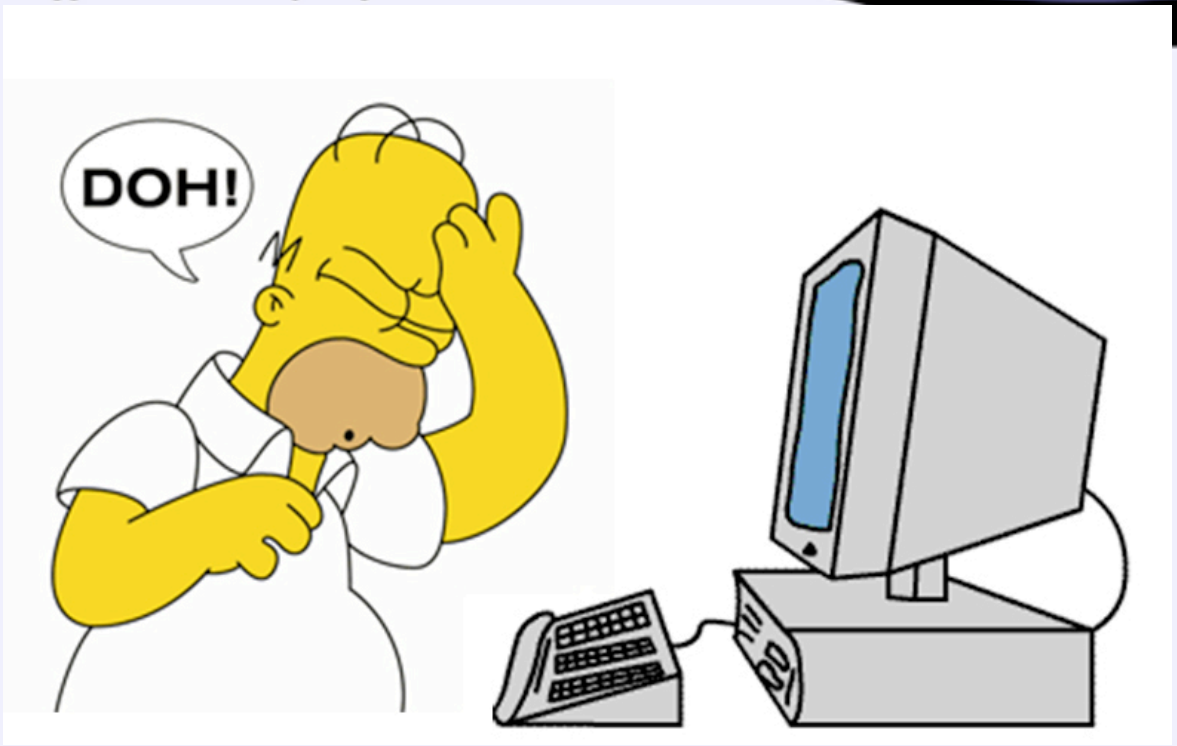
OWASP
The Open Web Application Security Project

```
String eMail= request.getParameter("email") ;
String userName= request.getParameter("username");

//SQL
PreparedStatement pstmt = con.prepareStatement("update  Users set email = ? where
username =  ?);
pstmt.setString(1,  eMail);
pstmt.setString(2,  userName);

//JPA
Query safeQuery  = entityManager.createQuery(
                        "update Users u SET u.email = ?1 WHERE u.username = ?2");
safeQuery.setParameter(1,  eMail) ;
safeQuery.setParameter(2,  userName);
safeQuery.executeUpdate();
```

Moral !



NEVER re-implements security mechanisms that are approuved in a core library !!

This is the same thing for crypto and other mechanisms that are not security....

Spot the vuln(s) !

OWASP
The Open Web Application Security Project

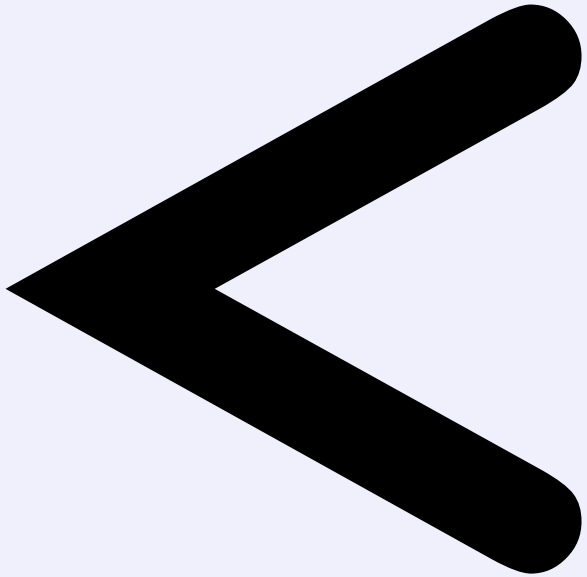
```
public final class InsertEmployeeAction extends Action {
   public ActionForward execute(ActionMapping mapping, ActionForm form,
                        HttpServletRequest request, HttpServletResponse response) throws Exception{

      Obj1 service = new Obj1 ();
      ObjForm objForm = (ObjForm) form;
      InfoADT adt = new InfoADT ();
      BeanUtils.copyProperties(adt, objForm);
      String searchQuery = objForm.getqueryString();
      String payload = objForm.getPayLoad();

      try {
         service.doWork(adt); // do something with the data
         ActionMessages messages = new ActionMessages();
         ActionMessage message = new ActionMessage("success", adt.getName() );
         messages.add( ActionMessages.GLOBAL_MESSAGE, message );
         saveMessages( request, messages );
         request.setAttribute("Record", adt);
         return (mapping.findForward("success"));
      }
      catch( DatabaseException de )
      {
         ActionErrors errors = new ActionErrors();
         ActionError error = new ActionError("error.employee.databaseException" + "Payload: "+payload);
         errors.add( ActionErrors.GLOBAL_ERROR, error );
         saveErrors( request, errors );
         return (mapping.findForward("error: "+ searchQuery));
      }
   }
}
```

XSS risks

- Session Hijacking
- Site Defacement
- Network Scanning
- Undermining CSRF Defenses
- Site Redirection/Phishing
- Load of Remotely Hosted Scripts
- Data Theft
- Keystroke Logging
- Attackers using XSS more frequently




OWASP
The Open Web Application Security Project
<

Encoding Output

OWASP
The Open Web Application Security Project

Safe ways to represent dangerous characters in a web page

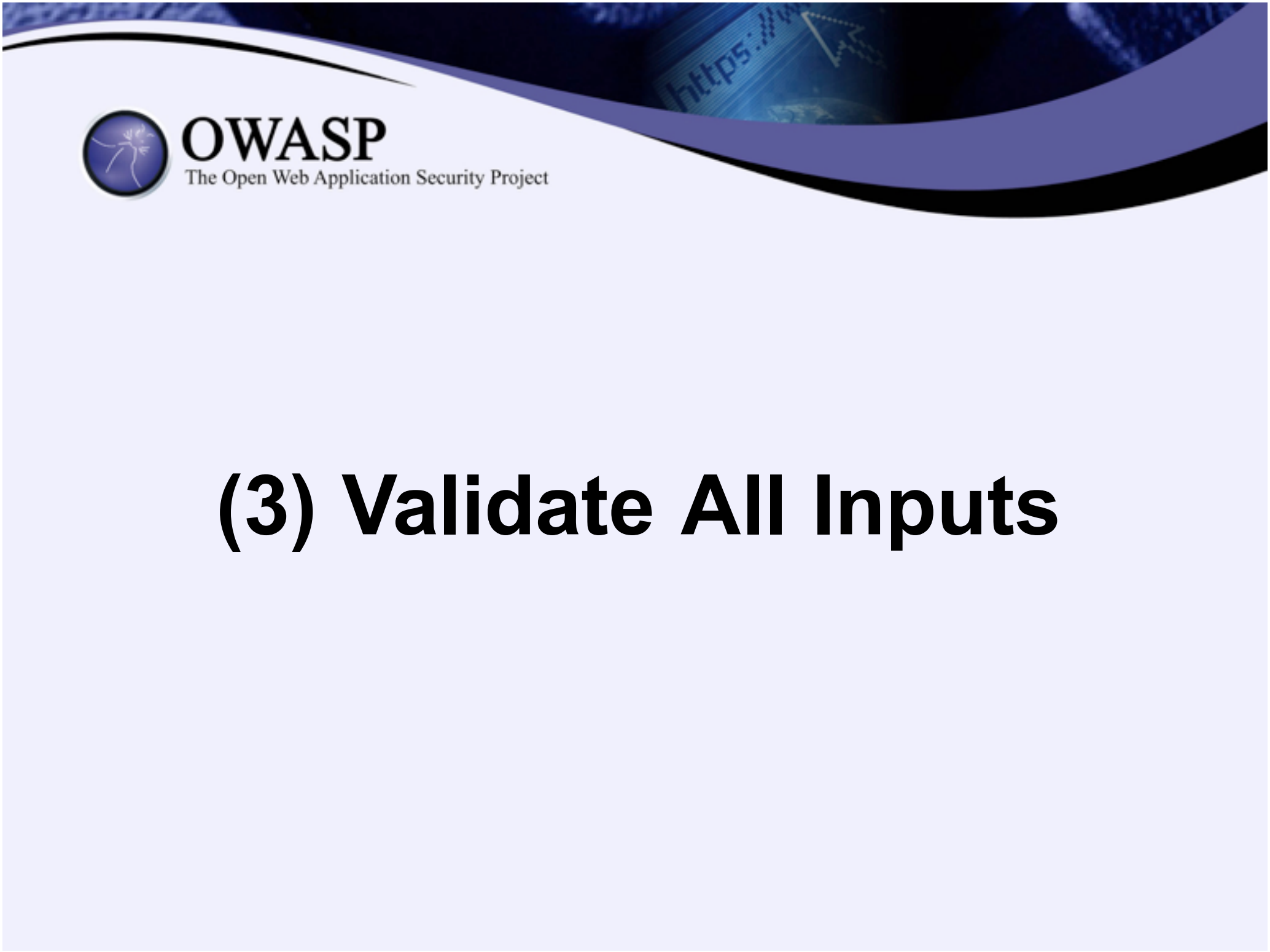
Characters	Decimal	Hexadecimal	HTML Character Set	Unicode
" (double quotation marks)	&#34;	&#x22;	&quot;	\u0022
' (single quotation mark)	&#39;	&#x27;	&apos;	\u0027
& (ampersand)	&#38;	&#x26;	&amp;	\u0026
< (less than)	&#60;	&#x3C;	&lt;	\u003c
> (greater than)	&#62;	&#x3E;	&gt;	\u003e

OWASP
The Open Web Application Security Project

OWASP Java Encoder Project

https://www.owasp.org/index.php/OWASP_Java_Encoder_Project

- No third party libraries or configuration necessary
- This code was designed for high-availability/high-performance encoding functionality
- Simple drop-in encoding functionality
- Redesigned for performance
- **More complete API (uri and uri component encoding, etc) in some regards.**
- Java 1.5+



(3) Validate All Inputs

This example displays all plugins and buttons that comes with the TinyMCE package.

Styles | Heading 1 | Font Family | Font Size

Welcome to the TinyMCE editor demo!

Feel free to try out the different features that are provided, please note that the MCImageManager and MCFileManager specific functionality is part of our commercial offering. The demo is to show the integration.

We really recommend Firefox as the primary browser for the best editing experience, but of course, TinyMCE is compatible with all major browsers.

Got questions or need help?

If you have questions or need help, feel free to visit
not miss out on the documentation, its a great resou

Path: h1 » img

SUBMIT

Source output from post

Element	HTML
content	Welcome to the TinyMCE editor demo! Feel free to try out the different features that are provided, please note that the MCImageManager and MCFileManager specific functionality is part of our commercial offering. The demo is to show the integration. We really recommend Firefox as the primary browser for the best editing experience, but of course, TinyMCE is compatible with all major browsers. Got questions or need help? If you have questions or need help, feel free to visit our community forum! We also offer Enterprise support solutions. Also do not miss out on the documentation, its a great resource wiki for understanding how TinyMCE works and integrates. Found a bug? If you think you have found a bug, you can use the Tracker to report bugs to the developers. And here is a simple table for you to play with.

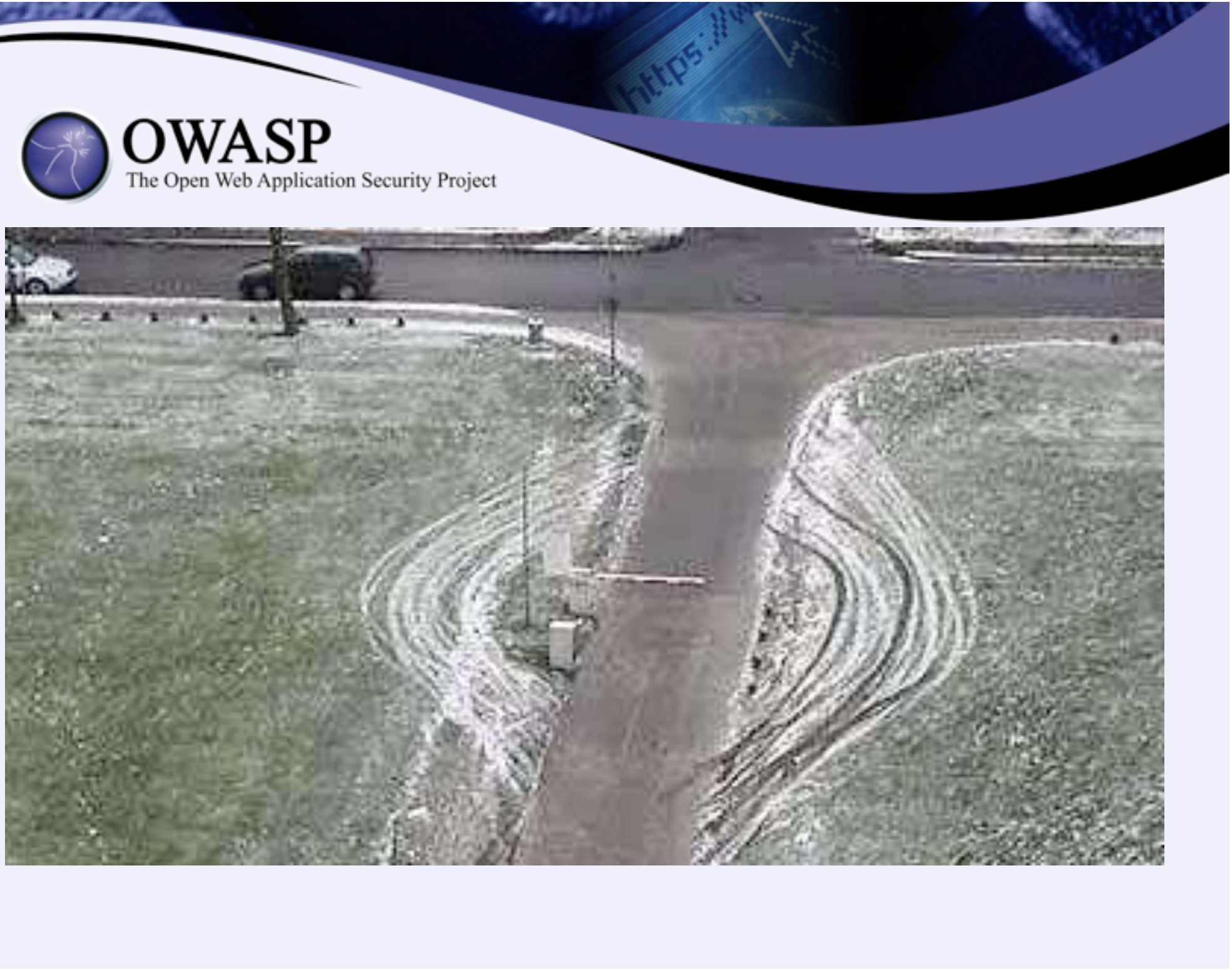
OWASP HTML Sanitizer Project

https://www.owasp.org/index.php/OWASP_Java_HTML_Sanitizer_Project

- HTML Sanitizer written in Java which lets you include HTML authored by third-parties in your web application while protecting against XSS.
- This code was written with security best practices in mind, has an extensive test suite, and has undergone adversarial security review https://code.google.com/p/owasp-java-html-sanitizer/wiki/AttackReviewGroundRules.
- It allows for simple programmatic POSITIVE policy configuration. No XML config.
- Actively maintained by Mike Samuel from Google's AppSec team!
- This is code from the Caja project that was donated by Google. It is rather high performance and low memory utilization.

OWASP
The Open Web Application Security Project

- **Upload Verification**
  - Filename and Size validation + antivirus
- **Upload Storage**
  - Use only trusted filenames + separate domain
- **Beware of "special" files**
  - "crossdomain.xml" or "clientaccesspolicy.xml".
- **Image Upload Verification**
  - Enforce proper image size limits
  - Use image rewriting libraries
  - Set the extension of the stored image to be a valid image extension
  - Ensure the detected content type of the image is safe
- **Generic Upload Verification**
  - Ensure decompressed size of file < maximum size
  - Ensure that an uploaded archive matches the type expected (zip, rar)
  - Ensure structured uploads such as an add-on follow proper standard


OWASP
The Open Web Application Security Project

What is Access Control?

OWASP
The Open Web Application Security Project

- Authorization is the process where a system determines if a specific user has access to a resource
- **Permission**: Represents app behavior only
- **Entitlement**: What a user is actually allowed to do
- **Principle/User**: Who/what you are entitling
- **Implicit Role**: Named permission, user associated
  - if (user.isRole("Manager"));
- **Explicit Role**: Named permission, resource associated
  - if (user.isAuthorized("report:view:3324");

Access Control Anti-Patterns

- Hard-coded role checks in application code
- Lack of centralized access control logic
- Untrusted data driving access control decisions
- Access control that is “open by default”
- Lack of addressing horizontal access control in a standardized way (if at all)
- Access control logic that needs to be manually added to every endpoint in code
- Access Control that is “sticky” per session
- Access Control that requires per-user policy

Access Controls Impact

OWASP
The Open Web Application Security Project

- Loss of accountability
  - Attackers maliciously execute actions as other users
  - Attackers maliciously execute higher level actions
- Disclosure of confidential data
  - Compromising admin-level accounts often results in access to user's confidential data
- Data tampering
  - Privilege levels do not distinguish users who can only view data and users permitted to modify data

HardCode Auth Rule

OWASP
The Open Web Application Security Project

```
void editProfile(User u, EditUser eu) {
 if (u.isManager()) {
   editUser(eu)
 }
}
```

Policy evolution....

OWASP
The Open Web Application Security Project

```
void editProfile(User u, EditUser eu) {
        if ((user.isManager() ||
                user.isAdministrator() ||
                user.isEditor()) &&
                user.id() != 1132))
        {
        // do stuff
        }
}
```

Hard-Coded Roles

OWASP
The Open Web Application Security Project

- Makes “proving” the policy of an application difficult for audit or Q/A purposes
- Any time access control policy needs to change, new code need to be pushed
- RBAC(http://en.wikipedia.org/wiki/Role-based_access_control) is often not granular enough
- Fragile, easy to make mistakes

Never Depend on Untrusted Data

OWASP
The Open Web Application Security Project

- Never trust request data for access control decisions
- Never make access control decisions in JavaScript
- Never make authorization decisions ***based solely on:***
  - hidden fields
  - *cookie values*
  - *form parameters*
  - *URL parameters*
  - *anything else from the request*
- Never depend on the order of values sent from the client

Best Practice: Centralized AuthZ

OWASP
The Open Web Application Security Project

- Define a centralized access controller
  - ACLService.isAuthorized(PERMISSION_CONSTANT)
  - ACLService.assertAuthorized(PERMISSION_CONSTANT)
- Access control decisions go through these simple API's
- Centralized logic to drive policy behavior and persistence
- May contain data-driven access control policy information

Should be like this...

OWASP
The Open Web Application Security Project

```
int userId= request.getInt(”userID"); //Best to get it from sessionID....

if (validateUser(userId) ) {
        if ( currentUser.isPermitted(  ”module:function:" + userId) ) {
                log.info(userId + “are permitted to access .");
                doStuff(userId);
        } else {
                log.info(userId + “ is notallowed to access!");
                throw new  AuthorizationException (userId);
        }
}
```

Establish Authentication and Identity Controls

OWASP
The Open Web Application Security Project

1) **Do not limit the type of characters or length of user password within reason**
    - Be wary of systems that allow unlimited password sizes (Django DOS Sept 2013)
- **2) Use a cryptographically strong credential-specific salt**
- **3) Impose difficult verification**
    - **on the attacker and defender :** use PBKDF2 or script
    - **on *only* the attacker : use** HMAC-SHA-256

2) Use a cryptographically strong credential-specific salt

- protect([**salt**] + [**password**]);
- Use a 32char or 64char salt (actual size dependent on protection function);
- Do not depend on hiding, splitting, or otherwise obscuring the salt

3a) Impose difficult verification on the attacker and defender

- PBKDF2([salt] + [password], c=140,000);
- Use **PBKDF2** when FIPS certification or enterprise support on many platforms is required
- Use **Scrypt** where resisting any/all hardware accelerated attacks is necessary but enterprise support and scale is not. (bcrypt is also a reasonable choice)

3b) Impose difficult verification on *only* the attacker

- HMAC-SHA-256([private key], [salt] + [password])
- Protect this key as any private key using best practices
- Store the key outside the credential store
- Build the password-to-hash conversion as a separate webservice (cryptograpic isolation).


OWASP
The Open Web Application Security Project
Password1!

Changing Password

OWASP
The Open Web Application Security Project

Require 2 identity questions

- Last name, account number, email, DOB
- Enforce lockout policy

Ask one or more good security questions

- https://www.owasp.org/index.php/Choosing_and_Using_Security_Questions_Cheat_Sheet

Send the user a randomly generated token via out-of-band

- app, SMS or token

Verify code in same web session

- Enforce lockout policy

Change password

- Enforce password policy

SecurePasswordStorage

OWASP
The Open Web Application Security Project

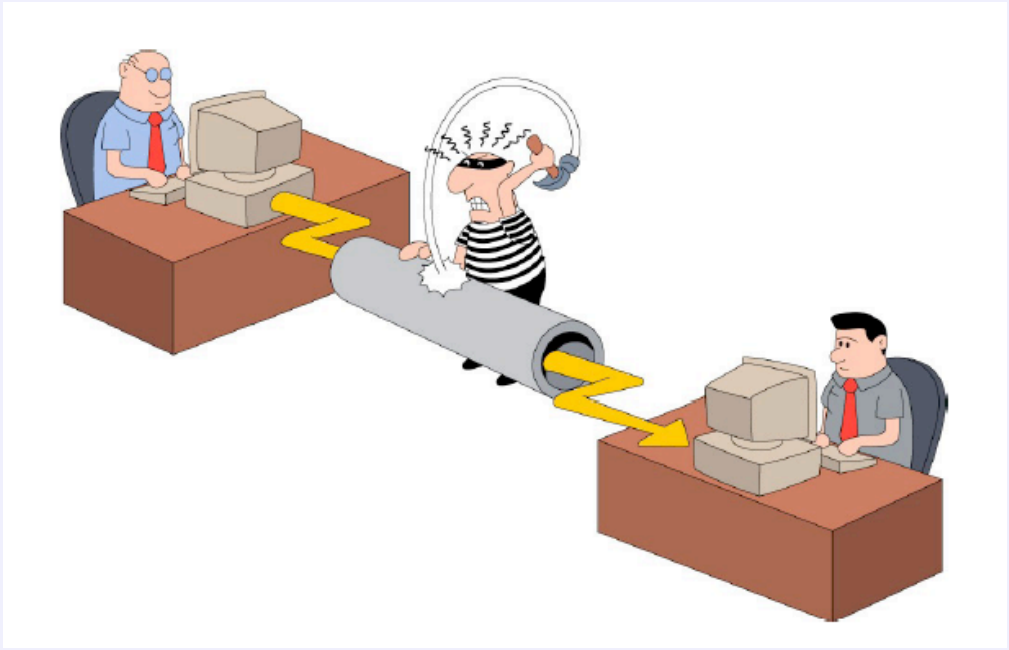
```
//import java.security and java.crypto and java.util needed
// Based on recommendation from NIST SP800-132.pdf
public class PasswordEncryptionService {
  public boolean authenticate(String attemptedPassword, byte[] encryptedPassword, byte[] salt)
        throws NoSuchAlgorithmException, InvalidKeySpecException {
       byte[] encryptedAttemptedPassword = getEncryptedPassword(attemptedPassword, salt);
      return Arrays.equals(encryptedPassword, encryptedAttemptedPassword);
  }
  public byte[] getEncryptedPassword(String password, byte[] salt)
        throws NoSuchAlgorithmException, InvalidKeySpecException {
    String algorithm = "PBKDF2WithHmacSHA1";  // SHA1 recommende by nist
    int derivedKeyLength = 160;
    int iterations = 20000; // minimum 1000 its from NIST
    KeySpec spec = new PBEKeySpec(password.toCharArray(), salt, iterations, derivedKeyLength);
    SecretKeyFactory f = SecretKeyFactory.getInstance(algorithm);
    return f.generateSecret(spec).getEncoded();
  }
  public byte[] generateSalt() throws NoSuchAlgorithmException {
   SecureRandom random = SecureRandom.getInstance("SHA1PRNG");
    byte[] salt = new byte[8];
    random.nextBytes(salt);
    return salt;
  }
```

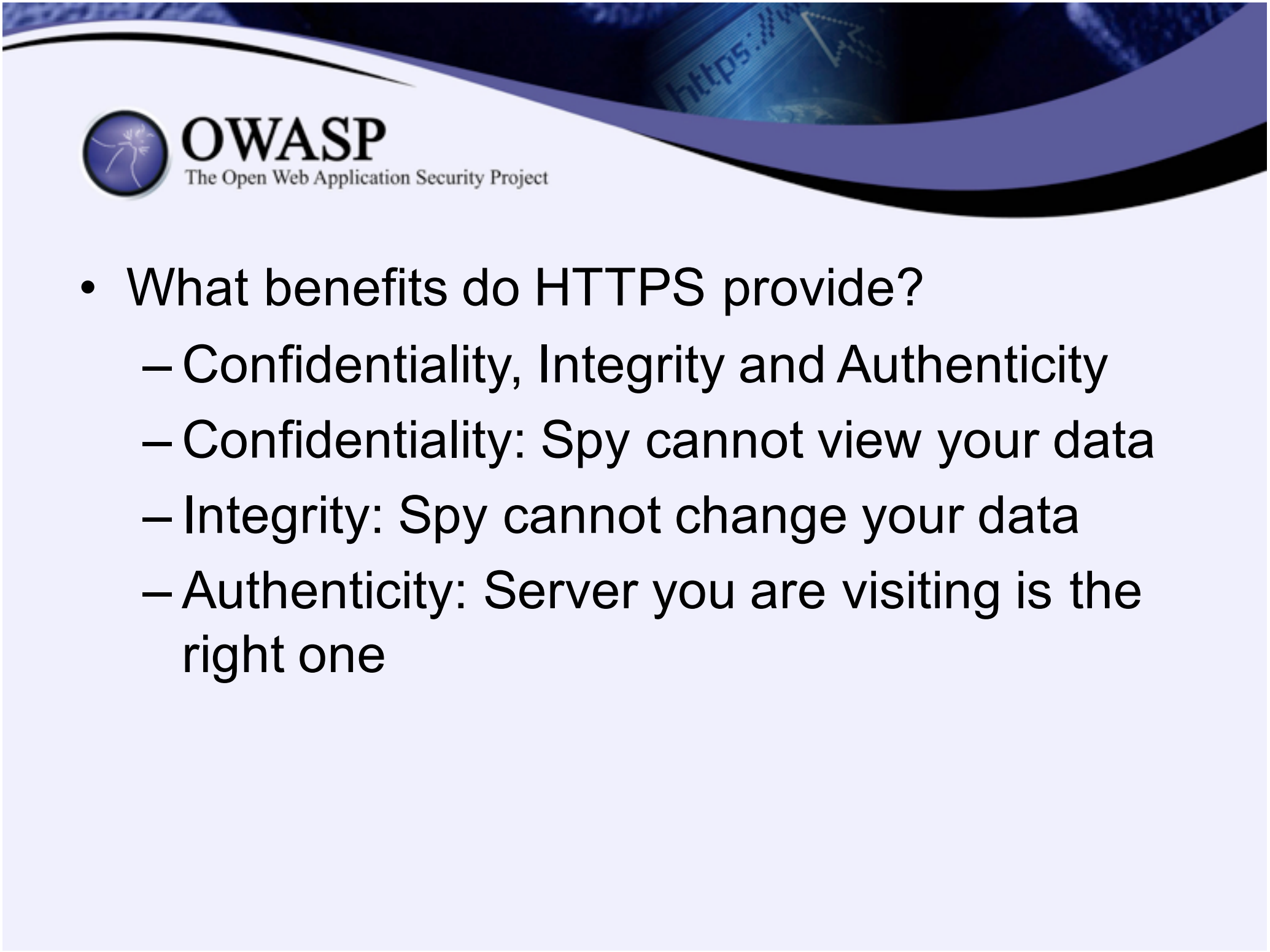
OWASP
The Open Web Application Security Project

- Authentication Cheat Sheet
  - https://www.owasp.org/index.php/Authentication_Cheat_Sheet
- Password Storage Cheat Sheet
  - https://www.owasp.org/index.php/Password_Storage_Cheat_Sheet
- Forgot Password Cheat Sheet
  - https://www.owasp.org/index.php/Forgot_Password_Cheat_Sheet
- Session Management Cheat Sheet
  - https://www.owasp.org/index.php/Session_Management_Cheat_Sheet
- ASVS AuthN and Session Requirements
  - https://www.owasp.org/index.php/OWASP_Application_Security_Verification_Standard_Project
- Choose Security Questions
  - https://www.owasp.org/index.php/Choosing_and_Using_Security_Questions_Cheat_Sheet

OWASP

The Open Web Application Security Project





- What benefits do HTTPS provide?
  - Confidentiality, Integrity and Authenticity
  - Confidentiality: Spy cannot view your data
  - Integrity: Spy cannot change your data
  - Authenticity: Server you are visiting is the right one

Encryption in Transit (HTTPS/TLS)

- HTTPS configuration best practices
  - https://www.owasp.org/index.php/Transport_Layer_Protection_Cheat_Sheet
  - https://www.ssllabs.com/projects/best-practices/

- Certificate Pinning
  - https://www.owasp.org/index.php/Pinning_Cheat_Sheet
- HSTS (Strict Transport Security)
  - http://www.youtube.com/watch?v=zEV3HOuM_Vw
  - *Strict-Transport-Security: max-age=31536000*
- Forward Secrecy
  - https://whispersystems.org/blog/asynchronous-security/
- Certificate Creation Transparency
  - http://certificate-transparency.org
- Browser Certificate Pruning
  - Etsy/Zane Lackey

HSTS – Strict Transport Security

- HSTS (Strict Transport Security)
  - http://www.youtube.com/watch?v=zEV3HOuM_Vw
  - *Strict-Transport-Security: max-age=31536000;* ***includeSubdomains***
- Forces browser to only make HTTPS connection to server
- Must be initially delivered over a HTTPS connection

Intrusion Detection

Logs and errors

OWASP
The Open Web Application Security Project

Risks ?

- Error messages can disclose information valuable to an attacker
- Failure can lead to an unhandled state, which can lead to denial of service – Unhandled failures can lead to malicious behavior being unnoticed

Fail Securely !

OWASP
The Open Web Application Security Project

- Not Just catching exception
- Not Just logging exception/errors

Handle all failures securely and return the system to a proper state

Design Patterns

OWASP
The Open Web Application Security Project

- Don't assume that an error condition won't occur
- It's what the attackers want you to assume
- Errors are like accidents, you don't expect them, but they happen
- Any code that can throw an exception should be in a Try Block
- Handle all possible exceptions
- Use Finally Blocks: leaving files open or exceptions defined after use creates resource leaks and possible system failure
- Short specific Try Blocks give you more control over the error state

App Layer Intrusion Detection

- Great detection points to start with
  - Input validation failure server side when client side validation exists
  - Input validation failure server side on non-user editable parameters such as hidden fields, checkboxes, radio buttons or select lists
  - Forced browsing to common attack entry points
  - Honeypot URL (e.g. a fake path listed in robots.txt like e.g. /admin/secretlogin.jsp)

App Layer Intrusion Detection

- Others
  - Blatant SQLi or XSS injection attacks
  - Workflow sequence abuse (e.g. multi-part form in wrong order)
  - Custom business logic (e.g. basket vs catalogue price mismatch)
  - Further Study:
    - **“libinjection: from SQLi to XSS” – Nick Galbreath**
    - **“Attack Driven Defense” – Zane Lackey**

OWASP AppSensor (Java)

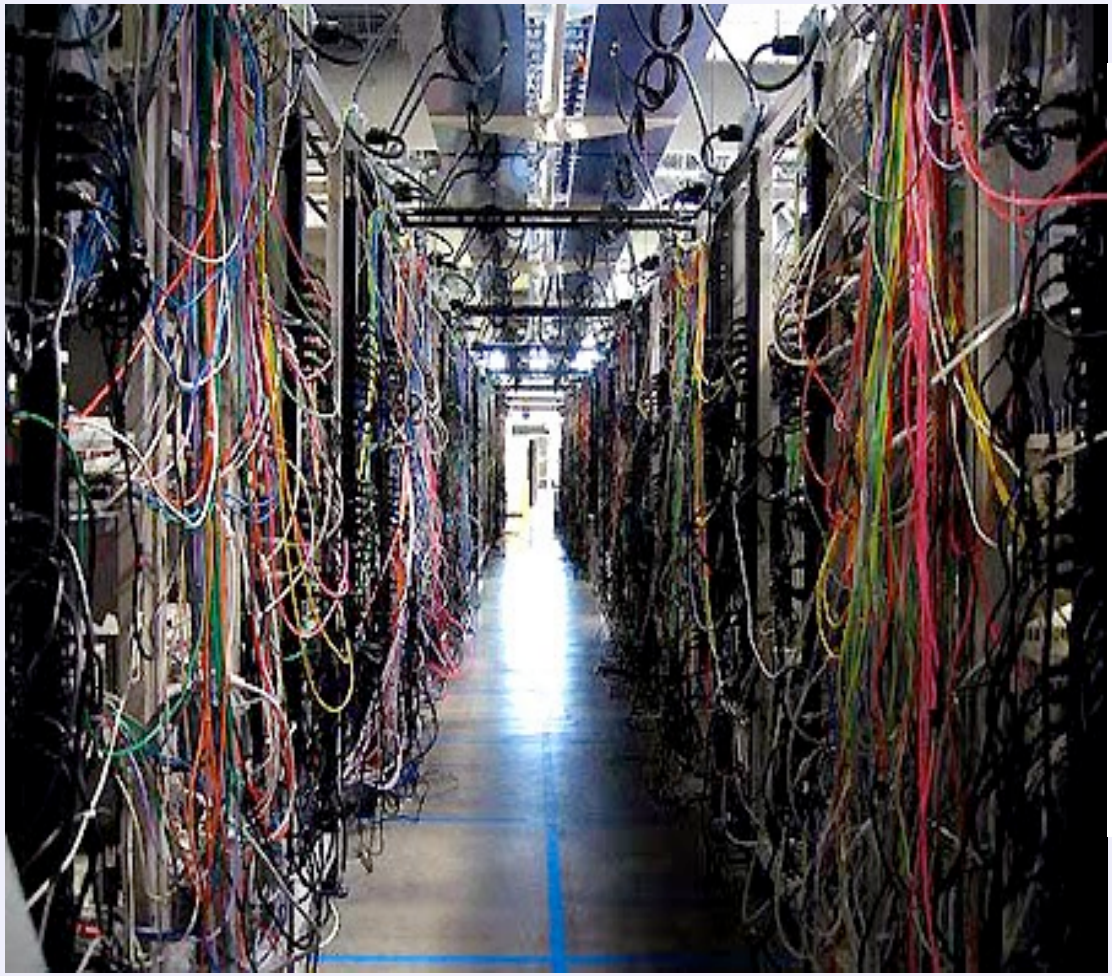
- Project and mailing list https://www.owasp.org/index.php/OWASP_AppSensor_Project
- Four-page briefing, Crosstalk, Journal of Defense Software Engineering
- http://www.crosstalkonline.org/storage/issue-archives/2011/201109/201109-Watson.pdf

Security Requirements

OWASP ASVS

https://www.owasp.org/index.php/Category:OWASP_Application_Security_Verification_Standard_Project

Bad Design



```
10 Input A
15 Input B
20 B = A + 10
30 IF B > 12 GOTO 60
40 C = B / 3
50 IF C < 24 GOTO 10
60 Write C
70 IF Write Failed GOTO 15
80 Input D
```

Security Architecture and Design

Strategic effort

- Business, technical and security stakeholders
- Functional and non-functional security properties
- Different flavors/efforts based on SDL/culture

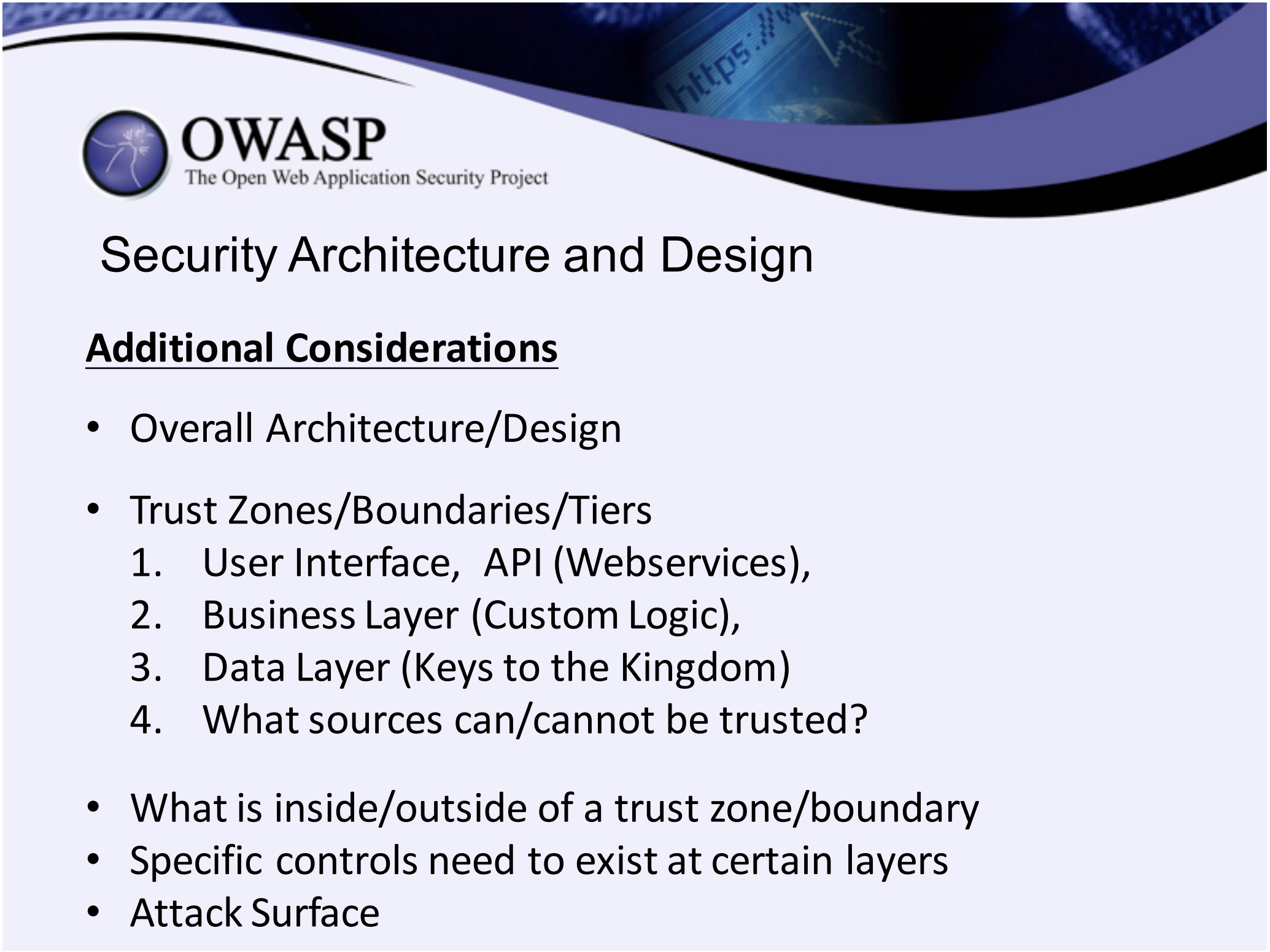
Example: state

- Should you use the request?
- Should you use a web session?
- Should you use the database?

These decisions have dramatic security implications

Trusting Input

- Treating all client side data as untrusted is important, and can be tied back to trust zones/boundaries in design/architecture.
- Ideally we want to consider all tiers to be untrusted and build controls at all layers, but this is not practical or even possible for some very large systems.

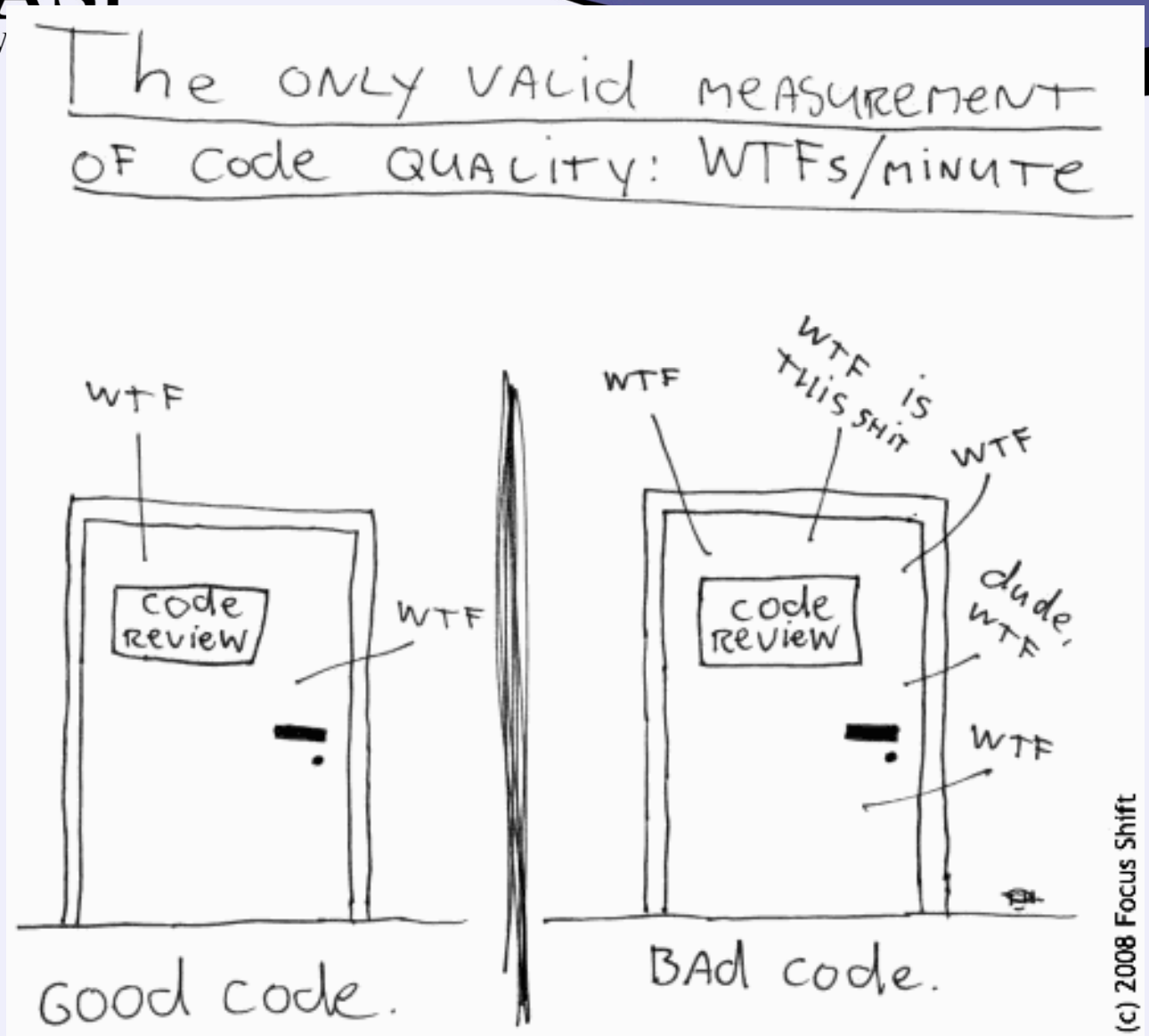


Security Architecture and Design

Additional Considerations

- Overall Architecture/Design
- Trust Zones/Boundaries/Tiers
  1. User Interface, API (Webservices),
  2. Business Layer (Custom Logic),
  3. Data Layer (Keys to the Kingdom)
  4. What sources can/cannot be trusted?
- What is inside/outside of a trust zone/boundary
- Specific controls need to exist at certain layers
- Attack Surface

Controlling code in one picture



Pentest or code review for CISO

OWASP
The Open Web Application Security Project

Top10 Web	Pentest	Code Review
A1-Injection	++	+++
A2-Broken Authentication and Session Management	++	+
A3-Cross-Site Scripting (XSS)	+++	+++
A4-Insecure Direct Object References	+	+++
A5-Security Misconfiguration	+	++
A6-Sensitive Data Exposure	++	+
A7-Missing Function Level Access Control	+	+
A8-Cross-Site Request Forgery (CSRF)	++	+
A9-Using Components with Known Vulnerabilities		+++
A10-Unvalidated Redirects and Forwards	+	+

Pentesting and code review for a developer

10 reasons to use ....

OWASP
The Open Web Application Security Project

- Written in Java ☺
- Integrated with CI Tools (Jenkins/Hudson/Bamboo)
- Modular (plugins by languages)
- Developer tools
- Dashboard
- OWASP project ☺
- Open-Source (and commercial too)

OWASP
The Open Web Application Security Project

- https://www.owasp.org/index.php/OWASP_SonarQube_Project

Of Course !

License

OWASP
The Open Web Application Security Project

- @SPoint

- sebastien.gioria@owasp.org


OWASP
The Open Web Application Security Project
LAST BUT NOT LEAST....

Lack of knowledge...

OWASP
The Open Web Application Security Project

```
String myString = "insecure";

myString.replace("i","1")
// do Stuff
```

Good Try...but...catch ☺

OWASP
The Open Web Application Security Project

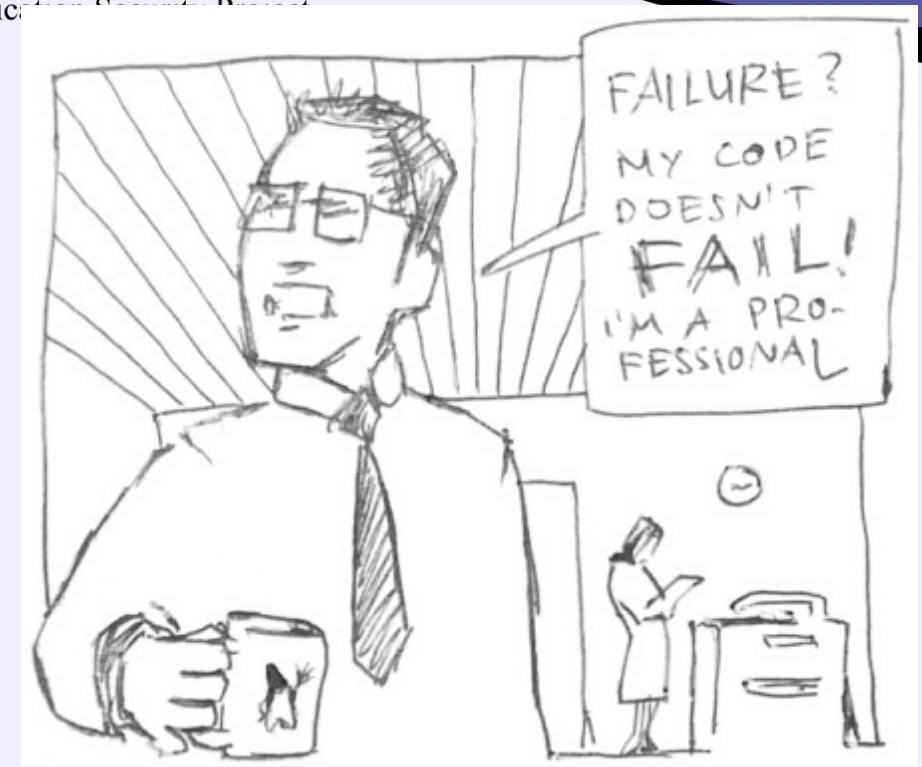
```
public class BadCatch {
  public void mymethodErr1() {
     try {
        //do Stuff
     } catch (SecurityException se){
        System.err.println (se);
     }
  }
}
```

Nice comments


```
// Pattern to match the string
Pattern myPattern = "\bword1\\W+(?:\\w+/W+){1,6}?word2\b";
String updated = MYCONSTANT1.replaceAll(mypattern, "$2");

// i iterate from 0 to 10
for (int i=0; i <10 ; i++) {
   // do stuff
   myMethod(updated);
}
```

Unit testing



- In computer programming, **unit testing** is a software testing method by which individual units of source code, sets of one or more computer program modules together with associated control data, usage procedures, and operating procedures, are tested to determine whether they are fit for use..
  (c) Wikipedia