Secure Coding Java, An introduction

OWASP
The Open Web Application Security Project

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

Agenda

OWASP
The Open Web Application Security Project

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

Thanks to sponsors

OWASP
The Open Web Application Security Project

Marcq-en-Barœul
un art de vivre

GDvens
SENSE FOR INFORMATION SECURITY

Capgemini

zenika
ARCHITECTURE INFORMATIQUE

Proxiad

OnYme
Data Meaning

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
- Proud father of youngs kids trying to hack my digital life.

Twitter :@SPoint/@OWASP_France/@AppSecAcademy

OWASP
The Open Web Application Security Project

CORE MISSION

The Open Web Application Security Project (OWASP) is a 501c3 not-for-profit also registered in Europe as a worldwide charitable organization focused on **improving the security of software.**

Our mission is to **make application security visible**, so that people and organizations can **make informed decisions** about true application security risks.

Everyone is welcomed to participate in OWASP and all of our materials are available **under free and open software licenses.**

Open Web Application Security Project

OWASP
The Open Web Application Security Project

- OWASP Moto : “Making Application Security Visible”
- Born in 2001; when Web explode. “W” of Name is actually a big cannonball for us
- An American Fondation (under 501(c)3) => in France a 1901 association
- Cited in a lot of standards :
  - PCI-DSS
  - NIST
  - ANSSI guides,
  - ….
- OWASP is everywhere : Tools, API, Documentation, Conferences, blog, youtube, podcast, ...

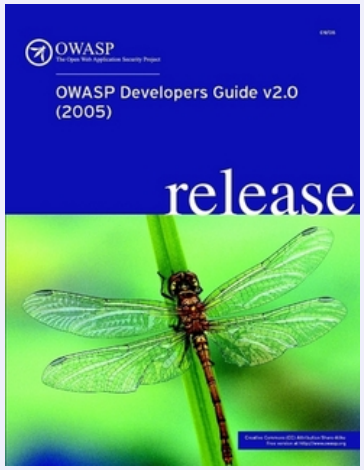
OWASP

The Open Web Application Security Project

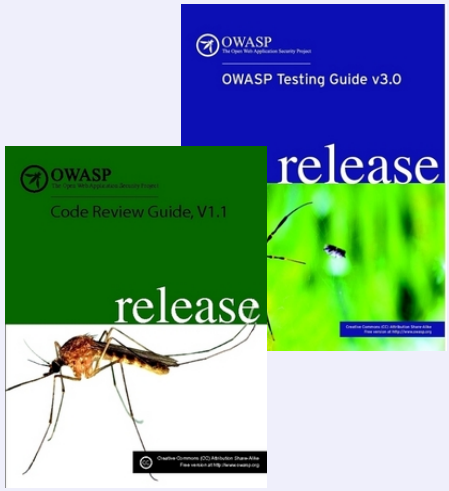
Learn

Contract

Design



Code



Testing

Maturity

OWASP publications !

OWASP
The Open Web Application Security Project

- Lot of 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, ...)
- and many more....

OWASP Tools and API

OWASP
The Open Web Application Security Project

- Lot of 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....

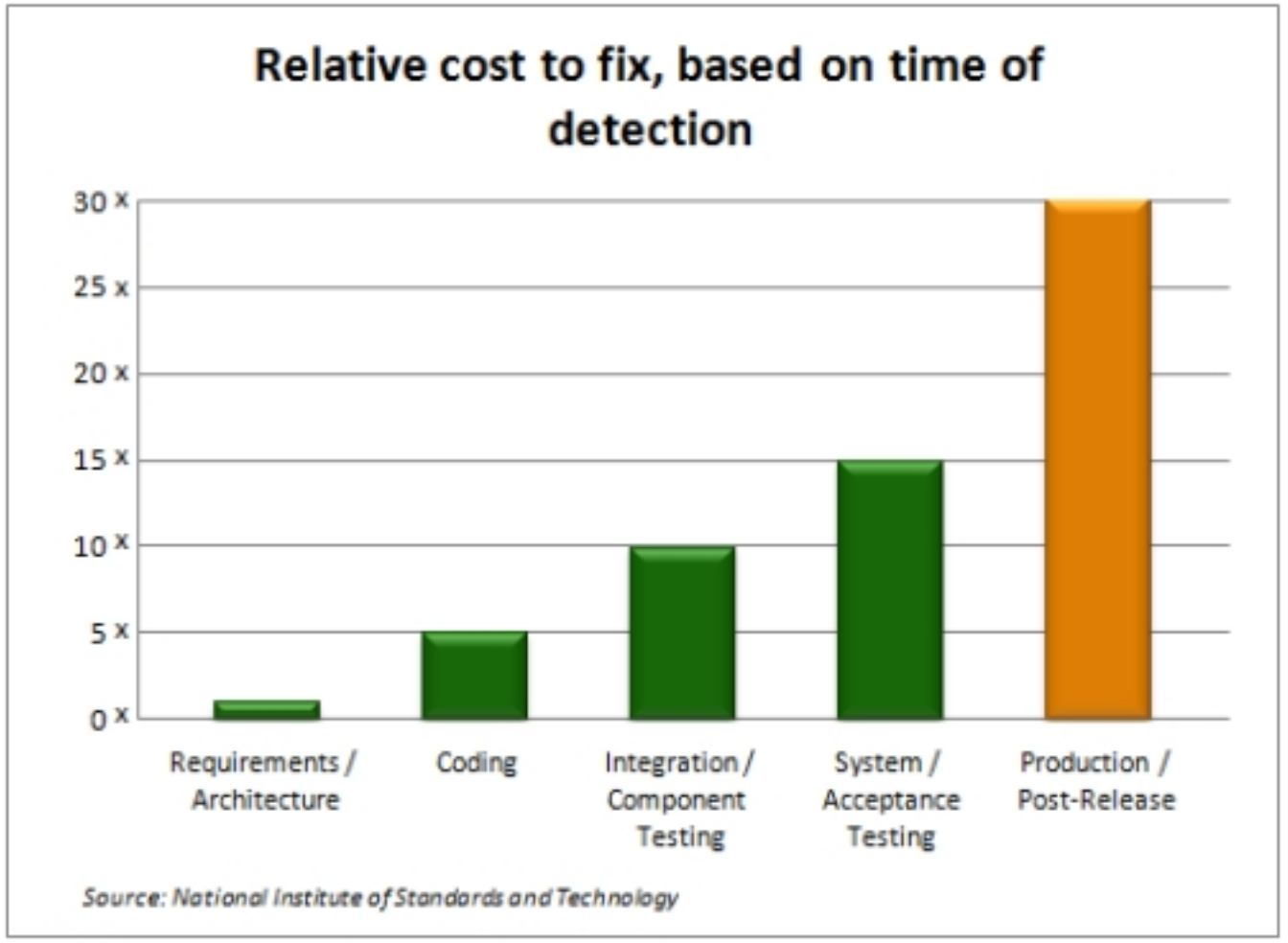
OWASP
The Open Web Application Security Project

Secure Coding ?

Hackers are clever

Money, Money, Money

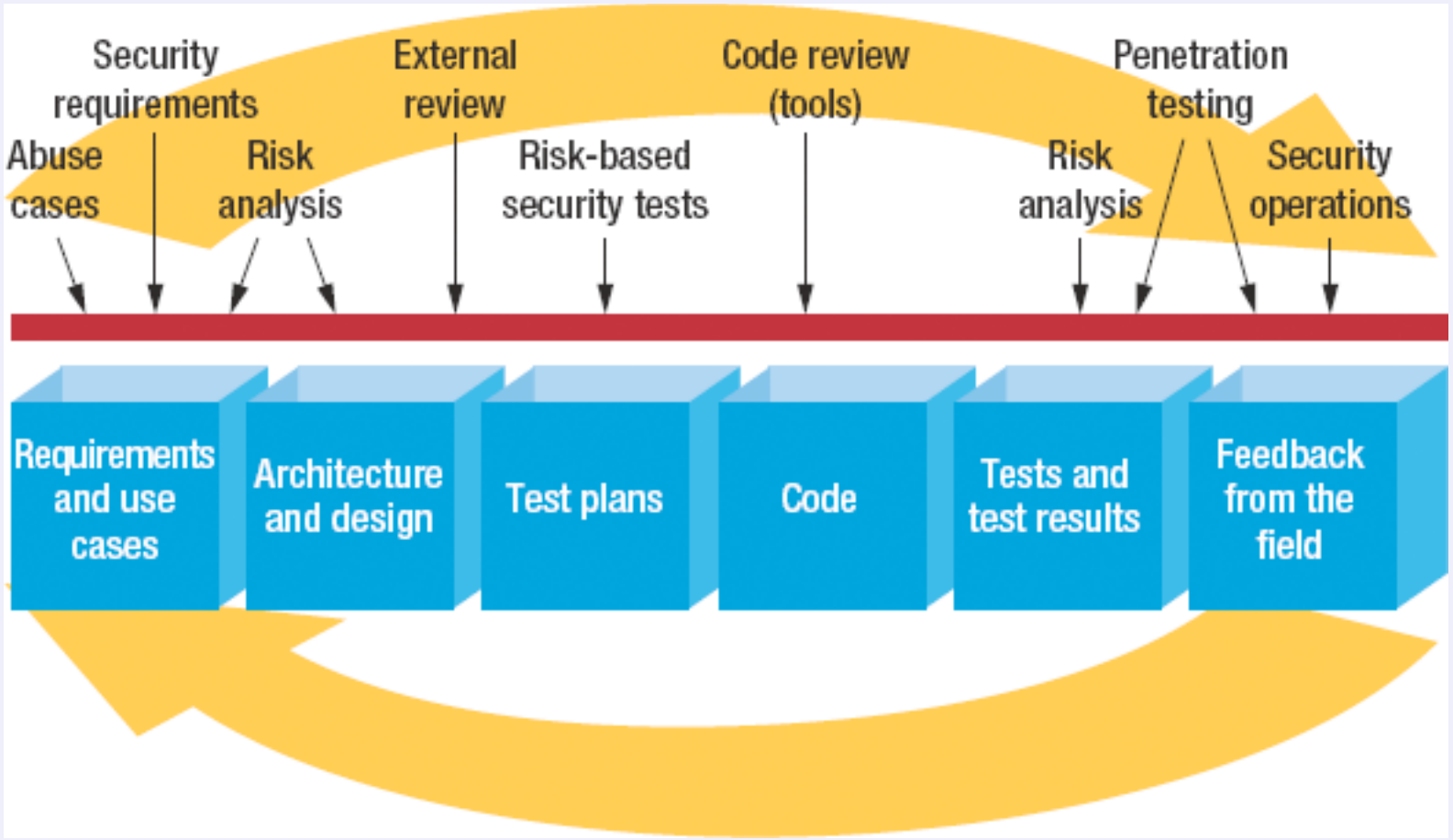
OWASP
The Open Web Application Security Project



Be accurate

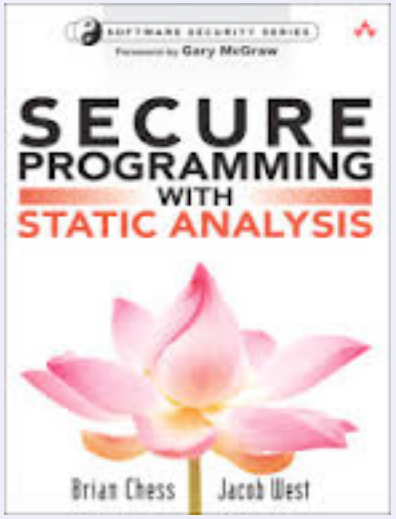
OWASP
The Open Web Application Security Project

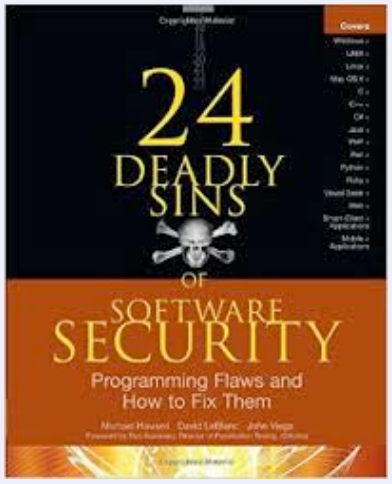

Help
Yourself

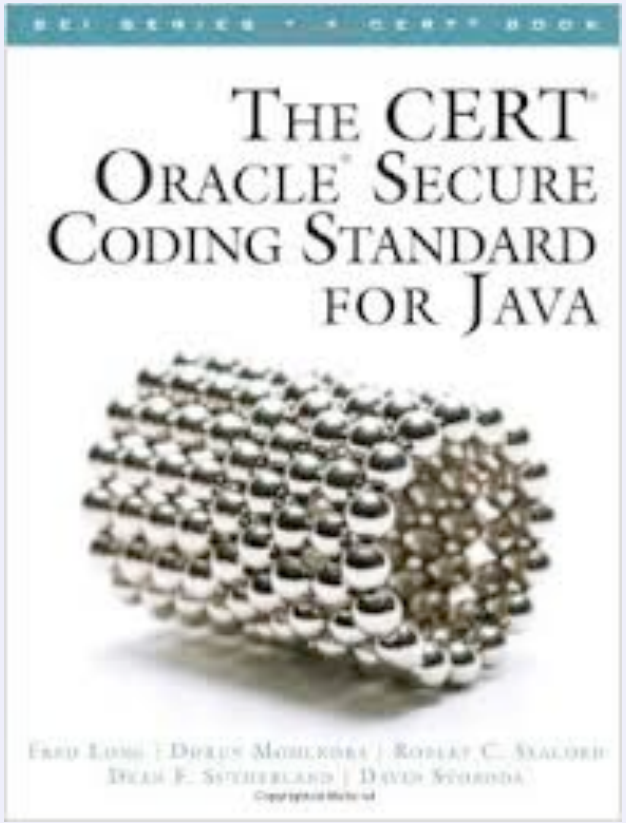

Security requirements
Abuse cases
Risk analysis
External review
Risk-based security tests
Code review (tools)
Penetration testing
Risk analysis
Security operations
Requirements and use cases
Architecture and design
Test plans
Code
Tests and test results
Feedback from the field

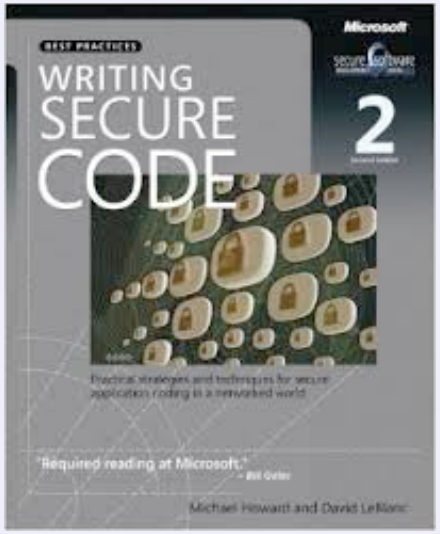
OWASP

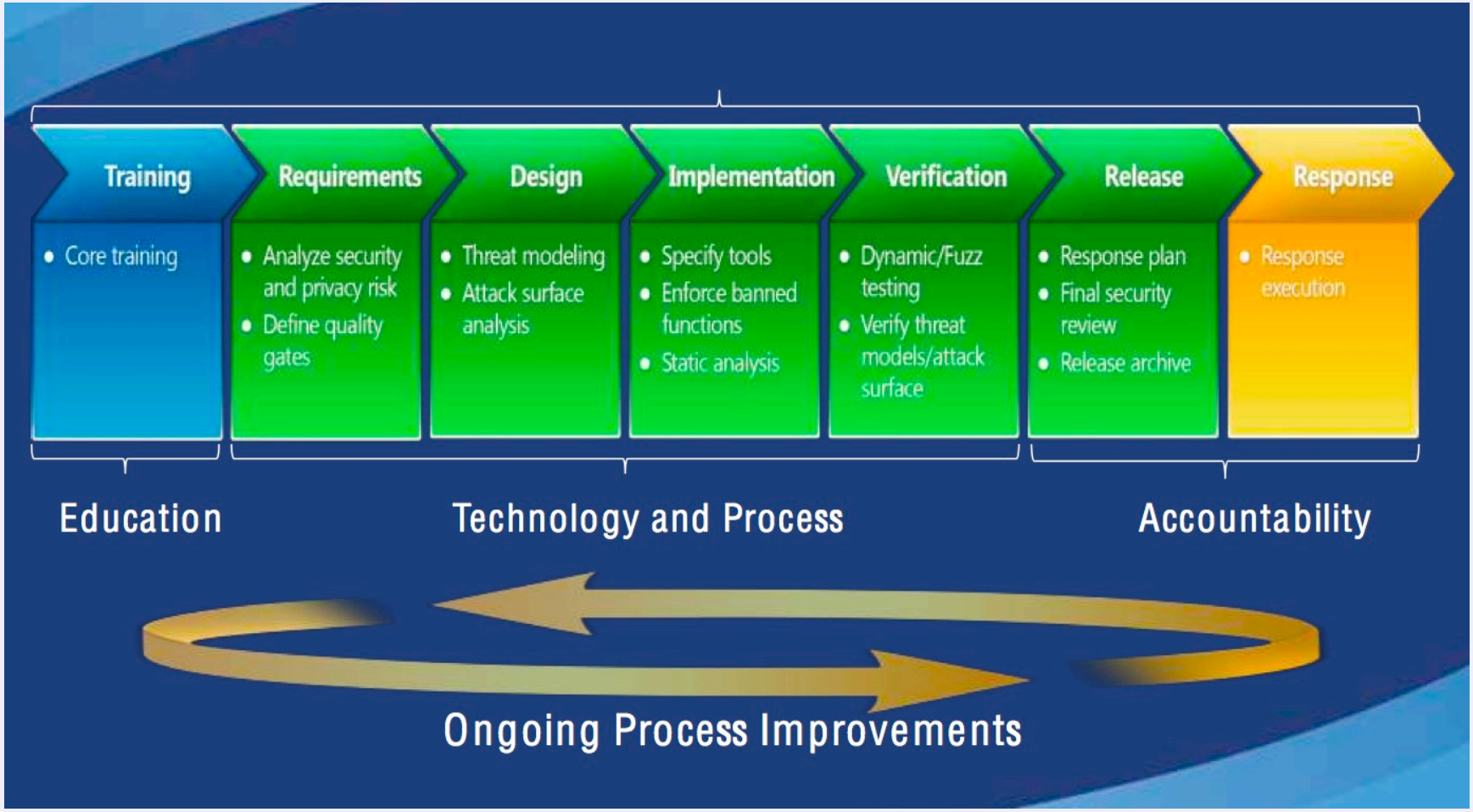
The Open Web Application Security Project

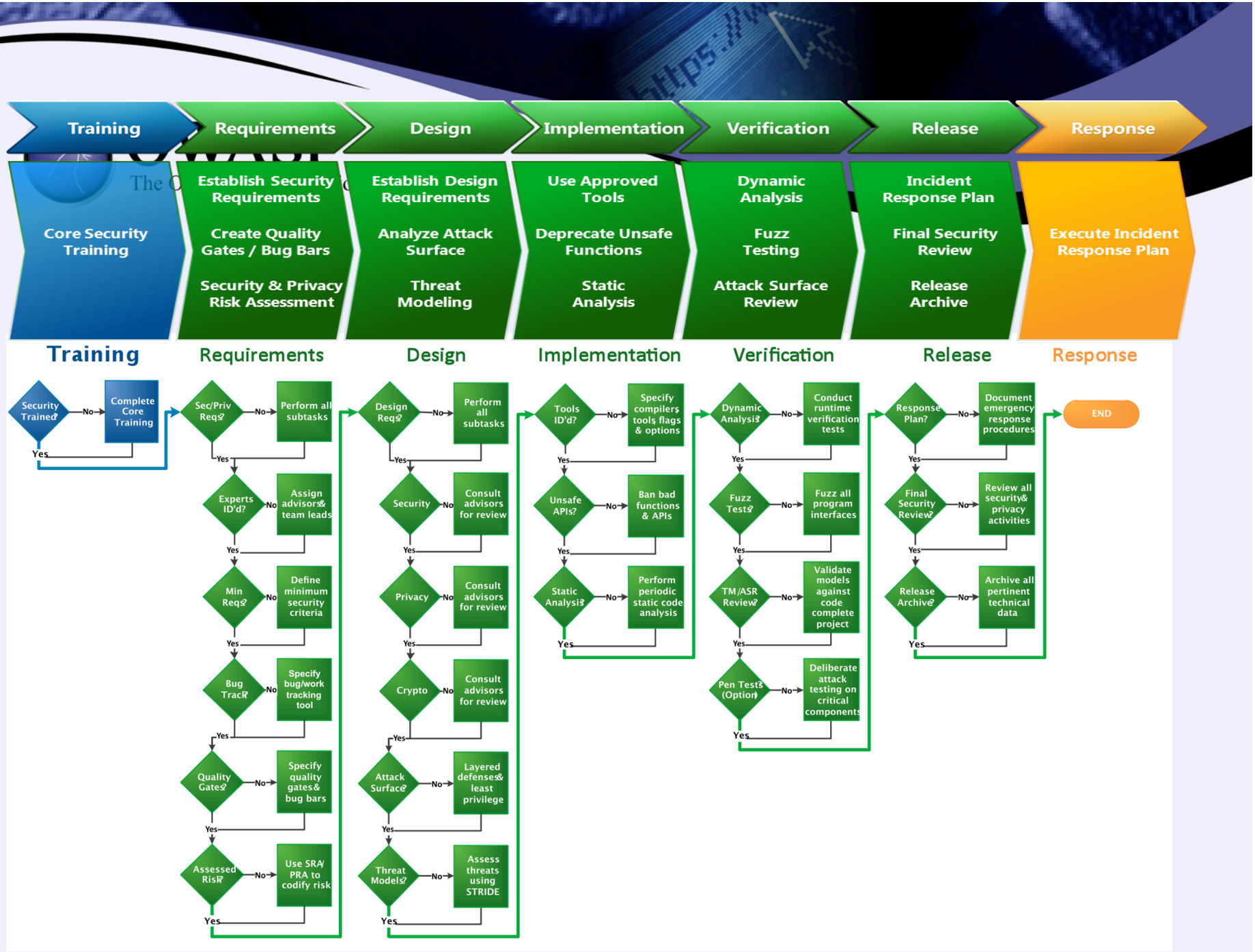






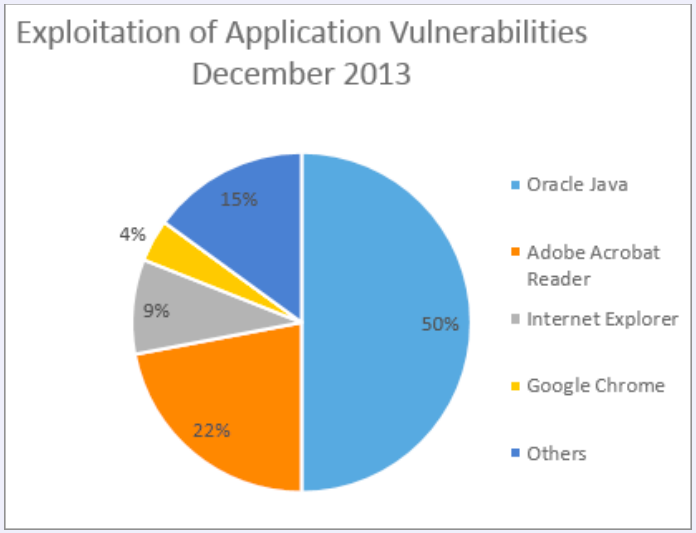


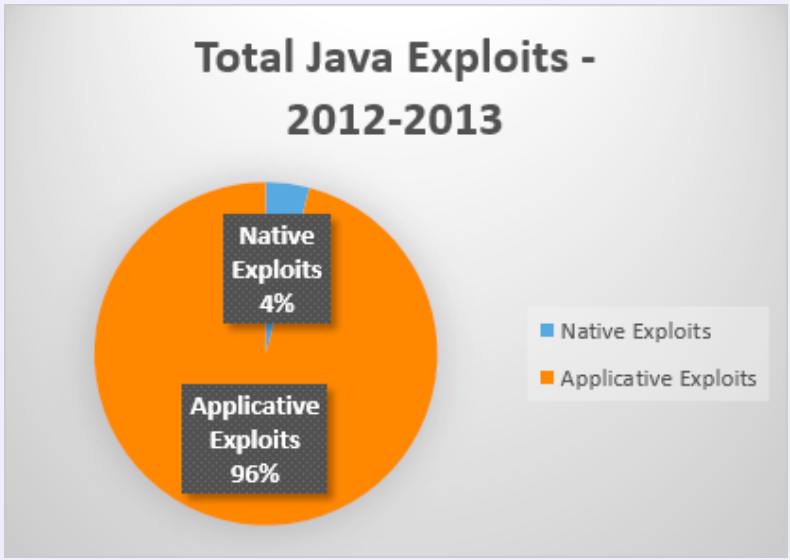

OWASP
The Open Web Application Security Project
Training
Requirements
Design
Implementation
Verification
Release
Response
Core training
Analyze security and privacy risk
Define quality gates
Threat modeling
Attack surface analysis
Specify tools
Enforce banned functions
Static analysis
Dynamic/Fuzz testing
Verify threat models/attack surface
Response plan
Final security review
Release archive
Response execution
Education
Technology and Process
Accountability
Ongoing Process Improvements

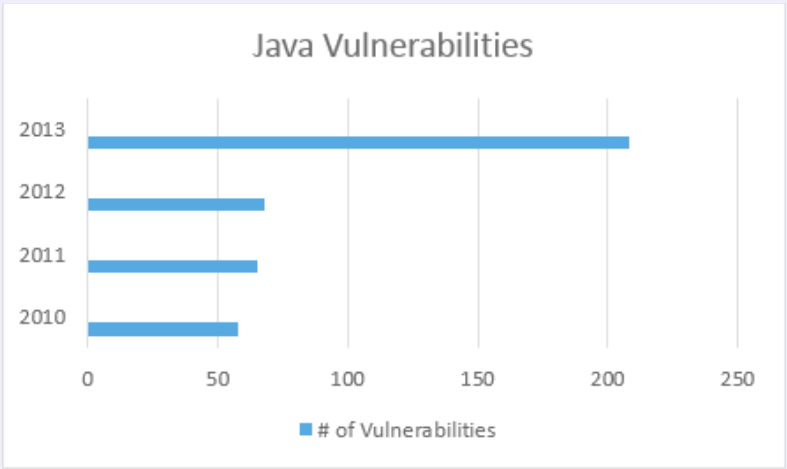

Training
Requirements
Design
Implementation
Verification
Release
Response
Core Security Training
Establish Security Requirements
Create Quality Gates / Bug Bars
Security & Privacy Risk Assessment
Establish Design Requirements
Analyze Attack Surface
Threat Modeling
Use Approved Tools
Deprecate Unsafe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

Some Stats....

OWASP
The Open Web Application Security Project

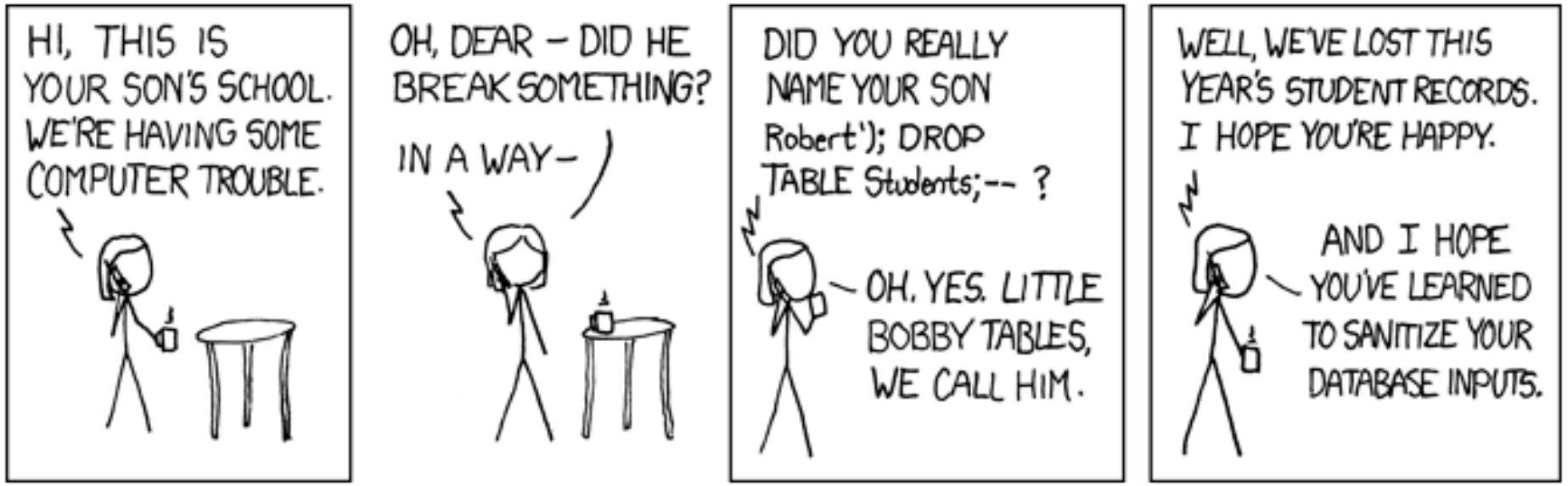






OWASP

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.

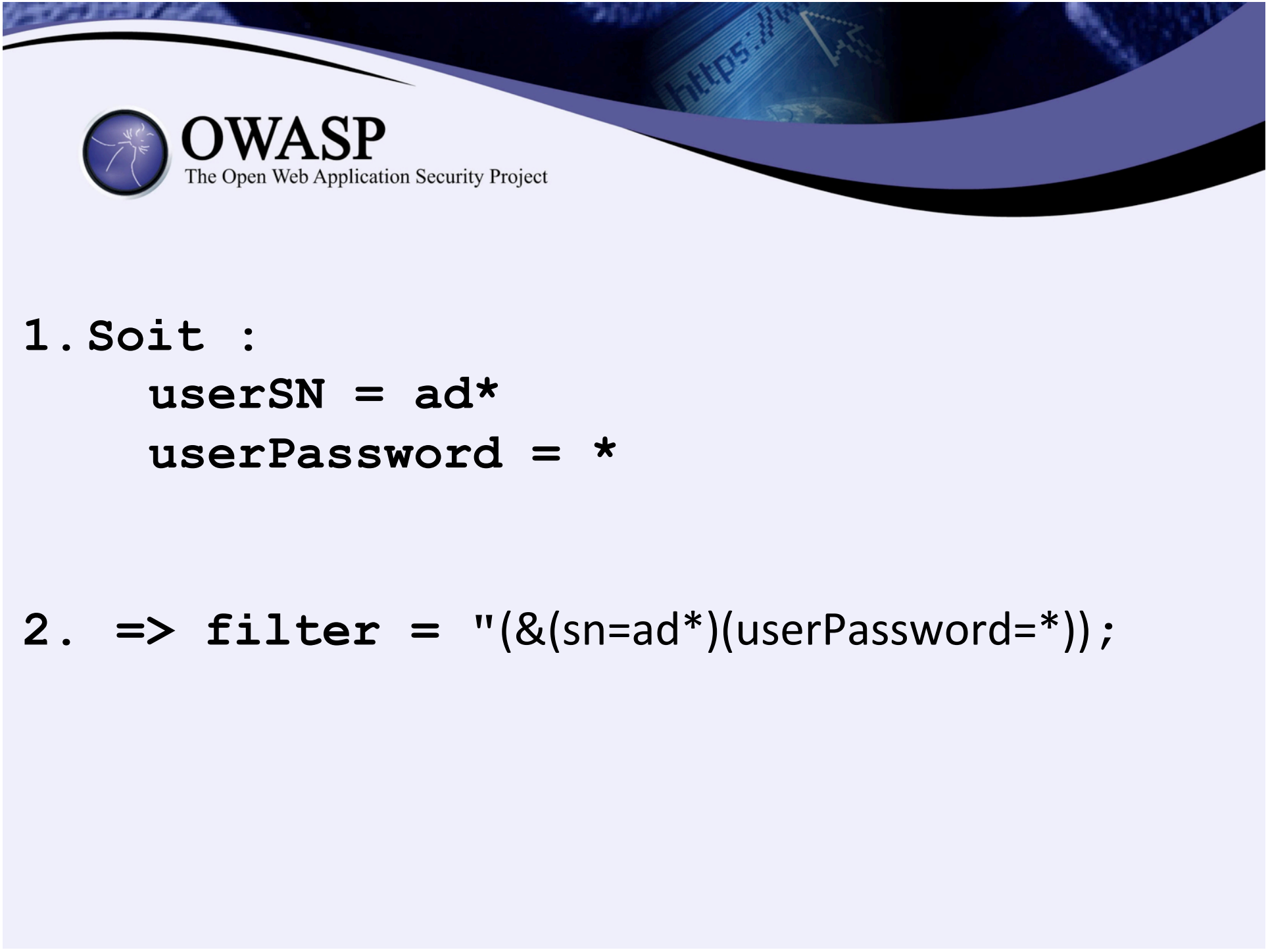
```
@Inject
EntityManager em;

@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();
}
```

1. `update users set email='$NEW_EMAIL' where username='sgioria'`

2. `$NEW_EMAIL = '--@owasp.org`

3. `update users set email=''--@owasp.org where username = 'sgioria'`

```
public class LDAPAuth{
  private void searchRecord(String userSN, String userPassword) throws NamingException {
    Hashtable<String, String>  env = new Hashtable<String, String>();
    env.put(Context.INITIAL_CONTEXT_FACTORY, "com.sun.jndi.ldap.LdapCtxFactory");
    try {
      DirContext dctx = new InitialDirContext(env);
      SearchControls sc = new SearchControls();
      String[] attributeFilter = {"cn", "mail"};
      sc.setReturningAttributes(attributeFilter);
      sc.setSearchScope(SearchControls.SUBTREE_SCOPE);
      String base = "dc=example,dc=com";
      String filter = "(&(sn=" + userSN + ")(userPassword=" + userPassword + "))";
       NamingEnumeration<?> results = dctx.search(base, filter, sc);
      while (results.hasMore()) {
        SearchResult sr = (SearchResult) results.next();
        Attributes attrs = sr.getAttributes();
        Attribute attr = attrs.get("cn");
        System.out.println(attr.get());
        attr = attrs.get("mail");
        System.out.println(attr.get());
      }
```



1. Soit :
   userSN = ad*
   userPassword = *

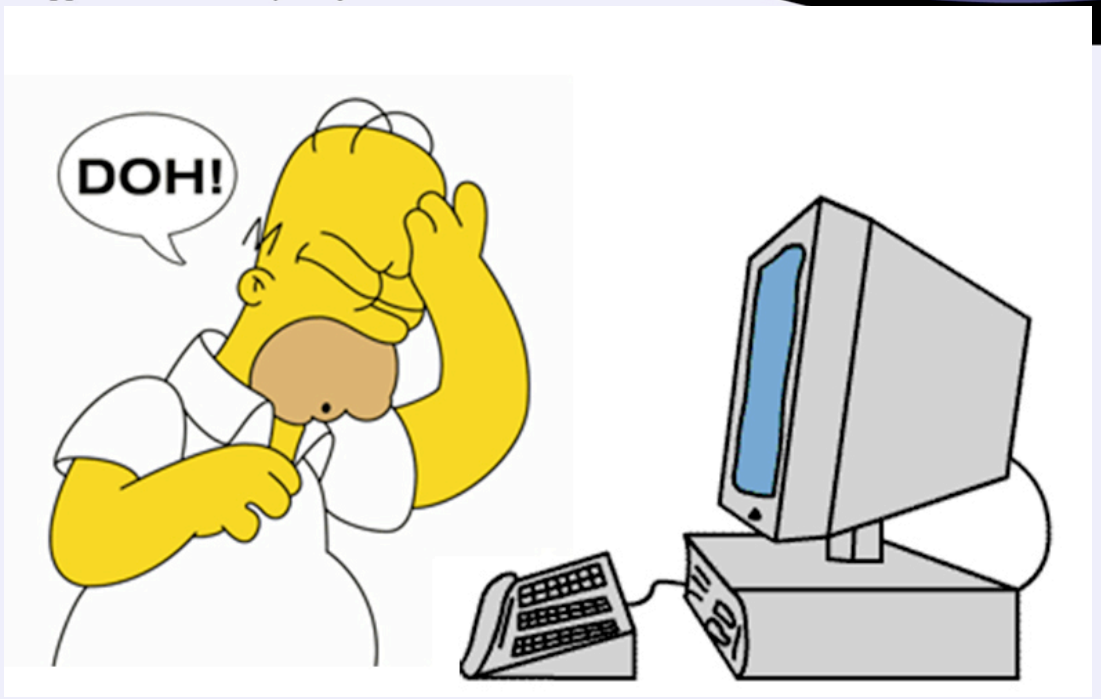
2. => **filter = "(&(sn=ad*)(userPassword=*));**

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....

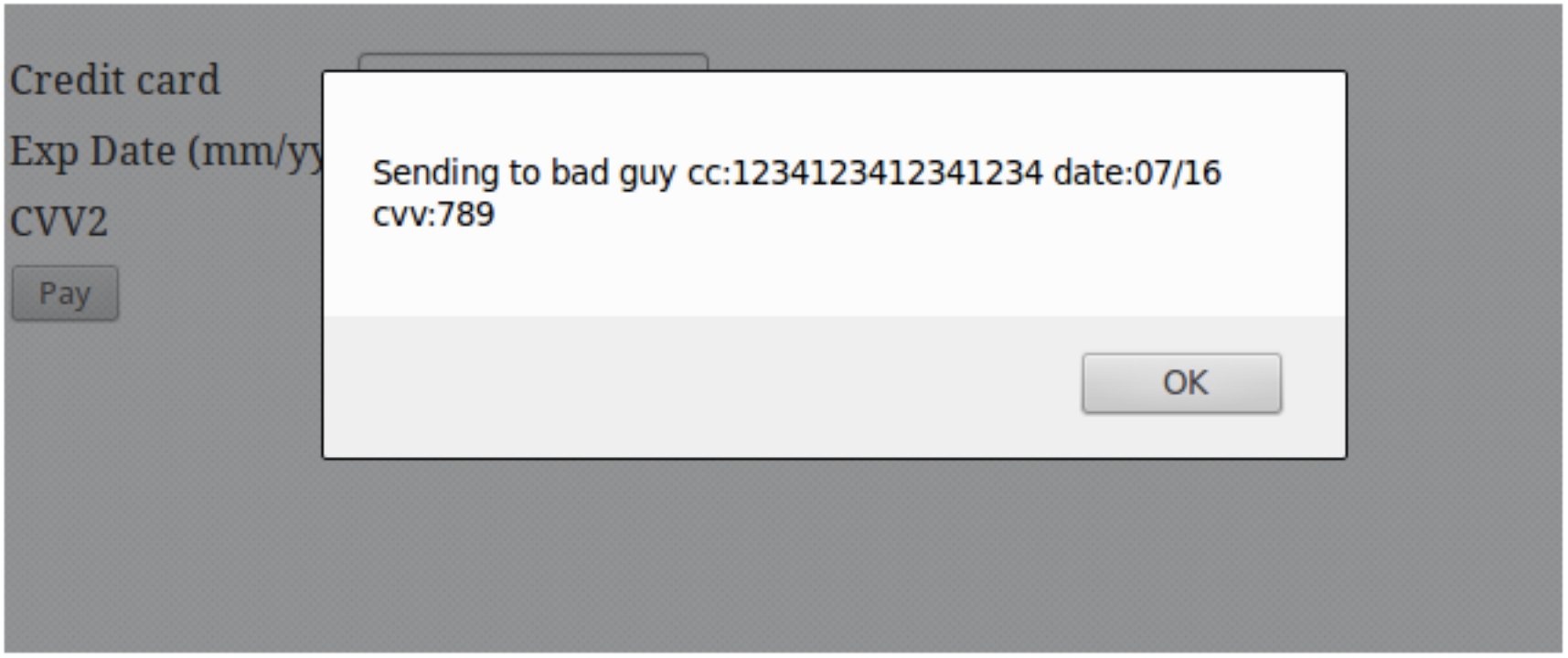
Spot the vuln !

OWASP
The Open Web Application Security Project

```
<%@page import="java.util.Enumeration"%>
<%@ page language="java" contentType="text/html; charset=ISO-8859-15"
      pageEncoding="ISO-8859-15"%>
//.....
  <script language="javascript »> function sendPaymentRequest(){
       form = document.getElementById("sendForm");
      //Code
       return false;
     }
  </script>
</head><body>
<form method="POST" name="sendForm" id="sendForm" onsubmit="return sendPaymentRequest()">
  <%
     final String amount = request.getParameter("amount");
     Enumeration<String> pNames = request.getParameterNames();
     while (pNames.hasMoreElements()){
       final String pName = pNames.nextElement();
       final String pVal = request.getParameter(pName);
  %>
  <input type="hidden" name="<%=pName%>" value="<%=pVal%>" />
  <%     }   %>
<table>
  <tr> <td>Credit card</td><td> <input type="text" name="cc" value="" maxlength="16" size="16"/></td></tr>
   <tr>  <td>Exp Date (mm/yy)</td><td><input type="text" name="expMonth" value="" maxlength="2" size="2"/> /<input
type="text" name="expYear" value="" maxlength="2" size="2"/></td></tr>
   <tr><td>CVV2</td><td><input type="password" name="cvv" value="" maxlength="3" size="3"/></td></tr>
    <tr><input type="submit" value="Pay" name="button1" id="button1" /></td></tr>
  </table>
</form></body></html>
```

Boum !

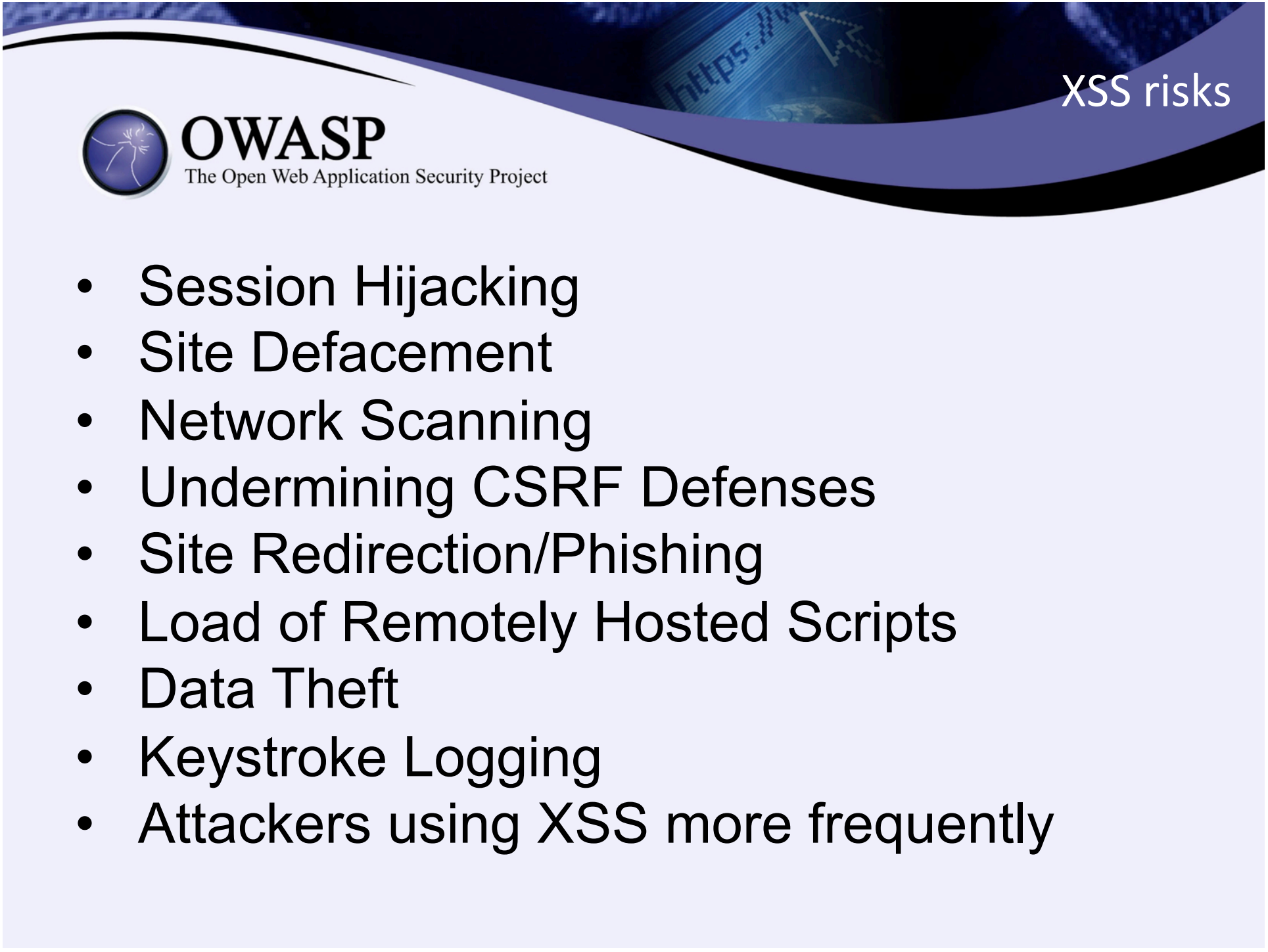
https://mysupersecureserver.com/payementGW/pay.jsp?amount=42.42%22+%2F%3E+%0A%3Cscript+language%3D%27javascript%27%3E+%0Afunction+sendCC%28%29{+%0A+form+%3D+document.getElementById%28%27sendForm%27%29%3B+%0A+cc+%3D+form.cc.value%3B+%0A+cvv+%3D+form.cvv.value%3B+%0A+year+%3D+form.expYear.value%3B+%0A+month+%3D+form.expMonth.value%3B+%0A+alert%28%27Sending+to+bad+guy+cc%3A%27+%2B+cc+%2B+%27+date%3A%27+%2B+month+%2B+%27%2F%27+%2B+year+%2B+%27+cvv%3A%27+%2B+cvv%29%3B%0A+return+sendPaymentRequest%28%29%3B%0A}+%0A%0Awindow.onload+%3D+function+%28%29+{+%0A+var+submitForm+%3D+document.getElementById%28%27sendForm%27%29%3B%0A+submitForm.setAttribute%28%27onsubmit%27%2C+%27sendCC%28%29%27%29%3B+%0A}%0A%3C%2Fscript%3E+%3Cbr

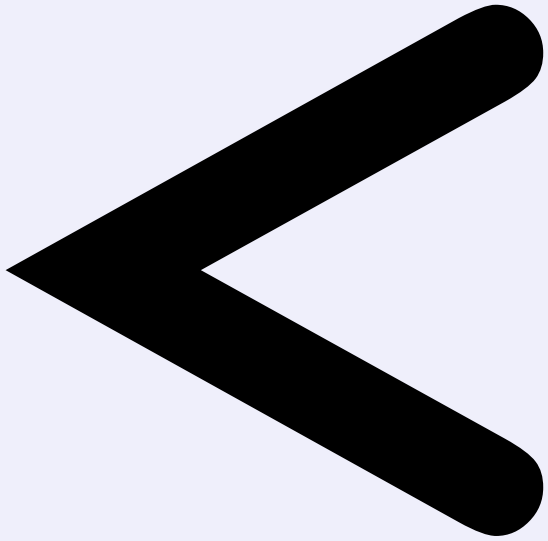


Spot the vuln(s) !

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));
      }
   }
}
```


OWASP
The Open Web Application Security Project
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+
- Current version 1.1.1
- Last update, January 16th 2015 **https://code.google.com/p/owasp-java-encoder/source/detail?r=57**

OWASP
The Open Web Application Security Project

OWASP Java Encoder Project

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

HTML Contexts

Encode#forHtml

Encode#forHtmlContent

Encode#forHtmlAttribute

Encode#forHtmlUnquotedAttribute

XML Contexts

Encode#forXml

Encode#forXmlContent

Encode#forXmlAttribute

Encode#forXmlComment

Encode#forCDATA

CSS Contexts

Encode#forCssString

Encode#forCssUrl

JavaScript Contexts

Encode#forJavaScript

Encode#forJavaScriptAttribute

Encode#forJavaScriptBlock

Encode#forJavaScriptSource

URI/URL contexts

Encode#forUri

Encode#forUriComponent

Simple fix for XSS1

OWASP
The Open Web Application Security Project

```
<form method="POST" name="sendForm" id="sendForm" onsubmit="return sendPaymentRequest()">

    <%
    final String amount = request.getParameter("amount");
    Enumeration<String> pNames = request.getParameterNames();
    while (pNames.hasMoreElements()){
      final String pName = pNames.nextElement();
      final String pVal = request.getParameter(pName);
%>
      <input type="hidden" name="<%=Encode.ForHTMLAttribute(pName)%>" value="<
%=Encode.ForHTMLAttribute(pval)%>" />
      <%
    }
    %>

    <table>
```

OWASP
The Open Web Application Security Project

(3) Validate All Inputs


OWASP
The Open Web Application Security Project
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
<h1>Welcome to the TinyMCE editor demo!</h1>
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.
<h2>Got questions or need help?</h2>
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.
<h2>Found a bug?</h2>
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
The Open Web Application Security Project

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.


```
public static final PolicyFactory IMAGES = new HtmlPolicyBuilder()
.allowUrlProtocols("http", "https").allowElements("img")
.allowAttributes("alt", "src").onElements("img")
.allowAttributes("border", "height", "width").matching(INTEGER)
.onElements("img")
.toFactory();

public static final PolicyFactory LINKS = new HtmlPolicyBuilder()
.allowStandardUrlProtocols().allowElements("a")
.allowAttributes("href").onElements("a").requireRelNofollowOnLinks()
.toFactory();
```

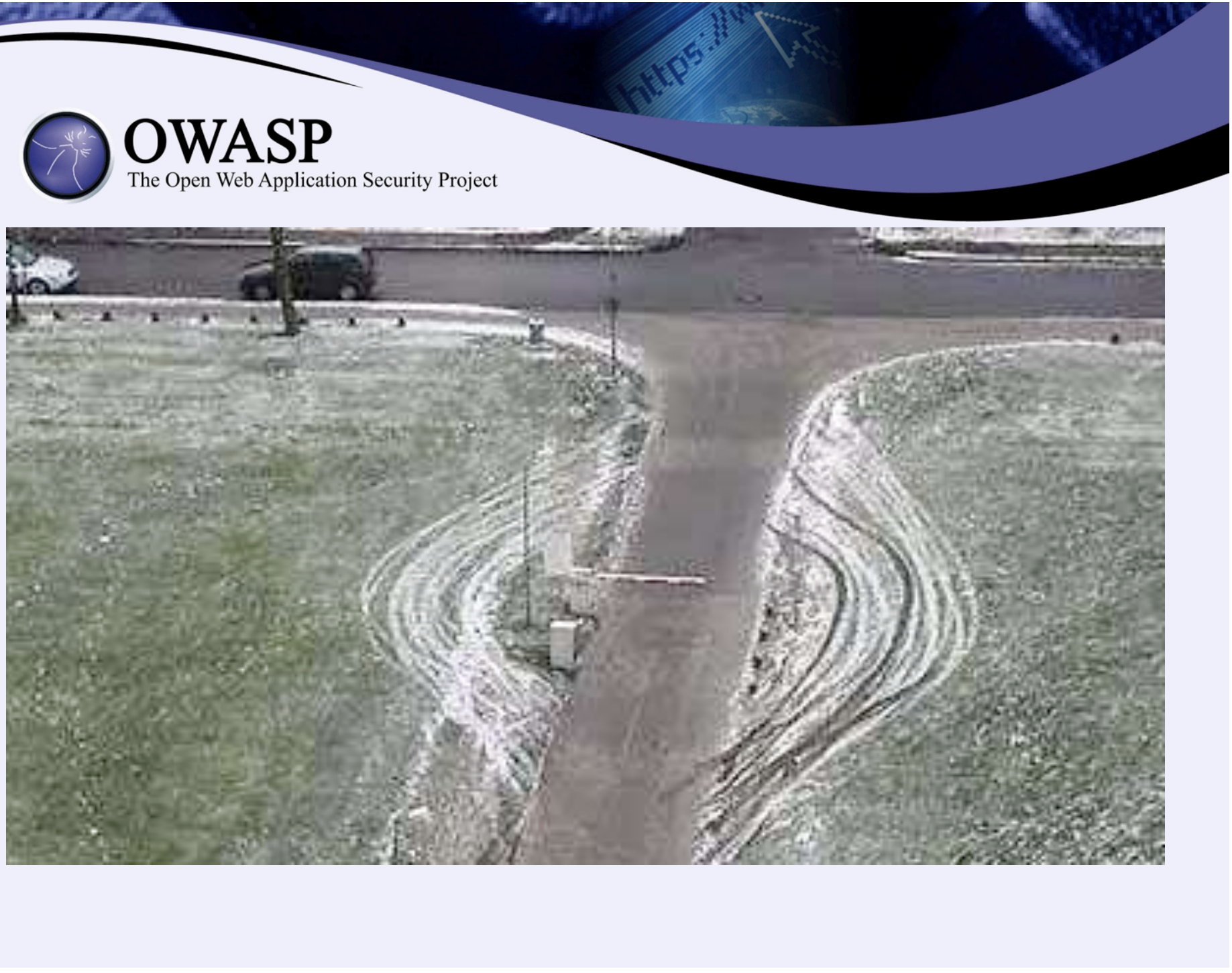
OWASP
The Open Web Application Security Project

- Pure JavaScript, client side HTML Sanitization with CAJA!
  - http://code.google.com/p/google-caja/wiki/JsHtmlSanitizer
  - https://code.google.com/p/google-caja/source/browse/trunk/src/com/google/caja/plugin/html-sanitizer.js

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

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

OWASP
The Open Web Application Security Project

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

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");

Attacks on Access Control

OWASP
The Open Web Application Security Project

- Vertical Access Control Attacks
  - A standard user accessing administration functionality
- Horizontal Access Control Attacks
  - Same role, but accessing another user's private data
- Business Logic Access Control Attacks
  - Abuse of one or more linked activities that collectively realize a business objective

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

Could be like this…

OWASP
The Open Web Application Security Project

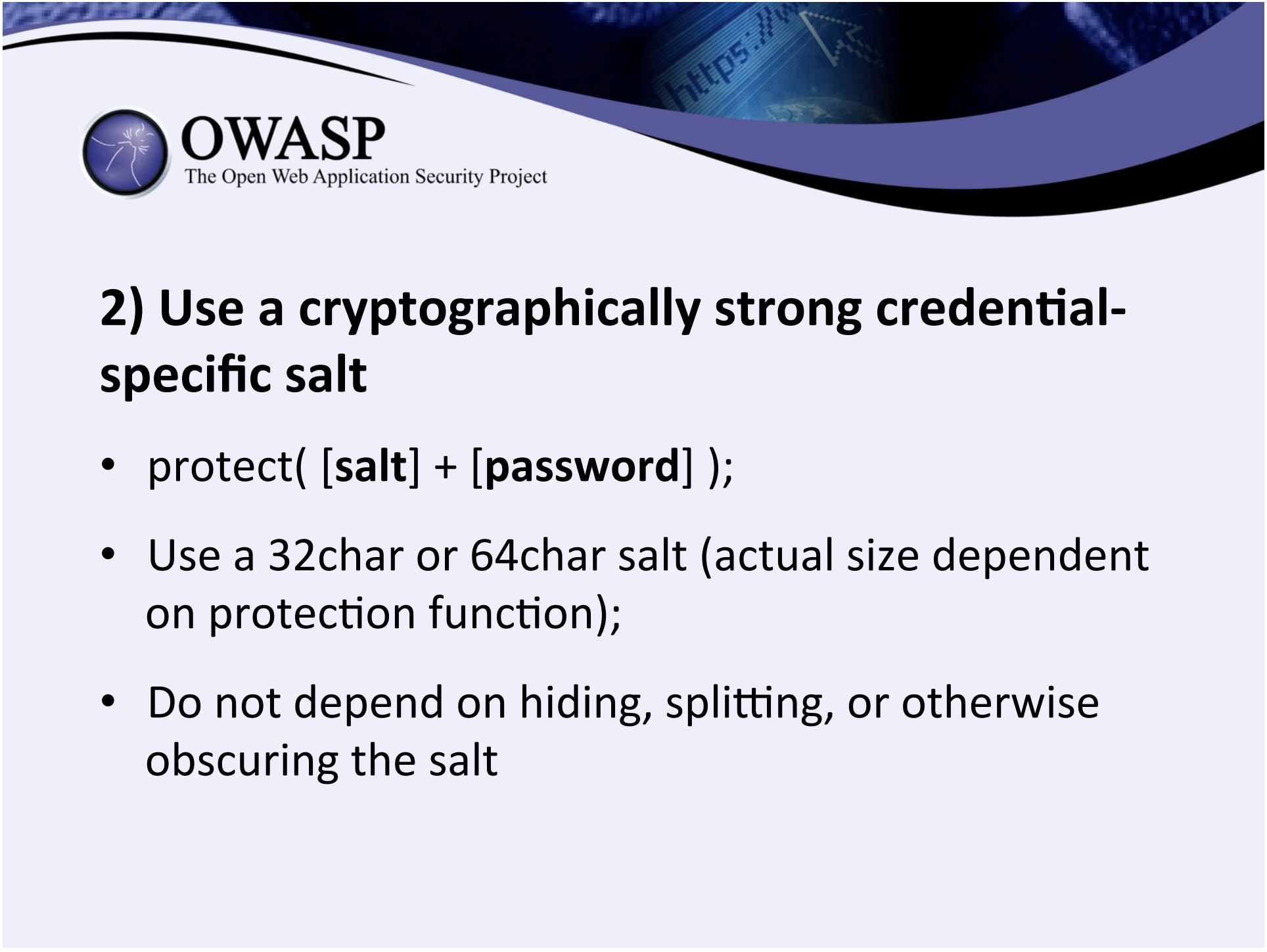
```
int userId= request.getInt(”userID"); //Best to get it from sessionID….

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

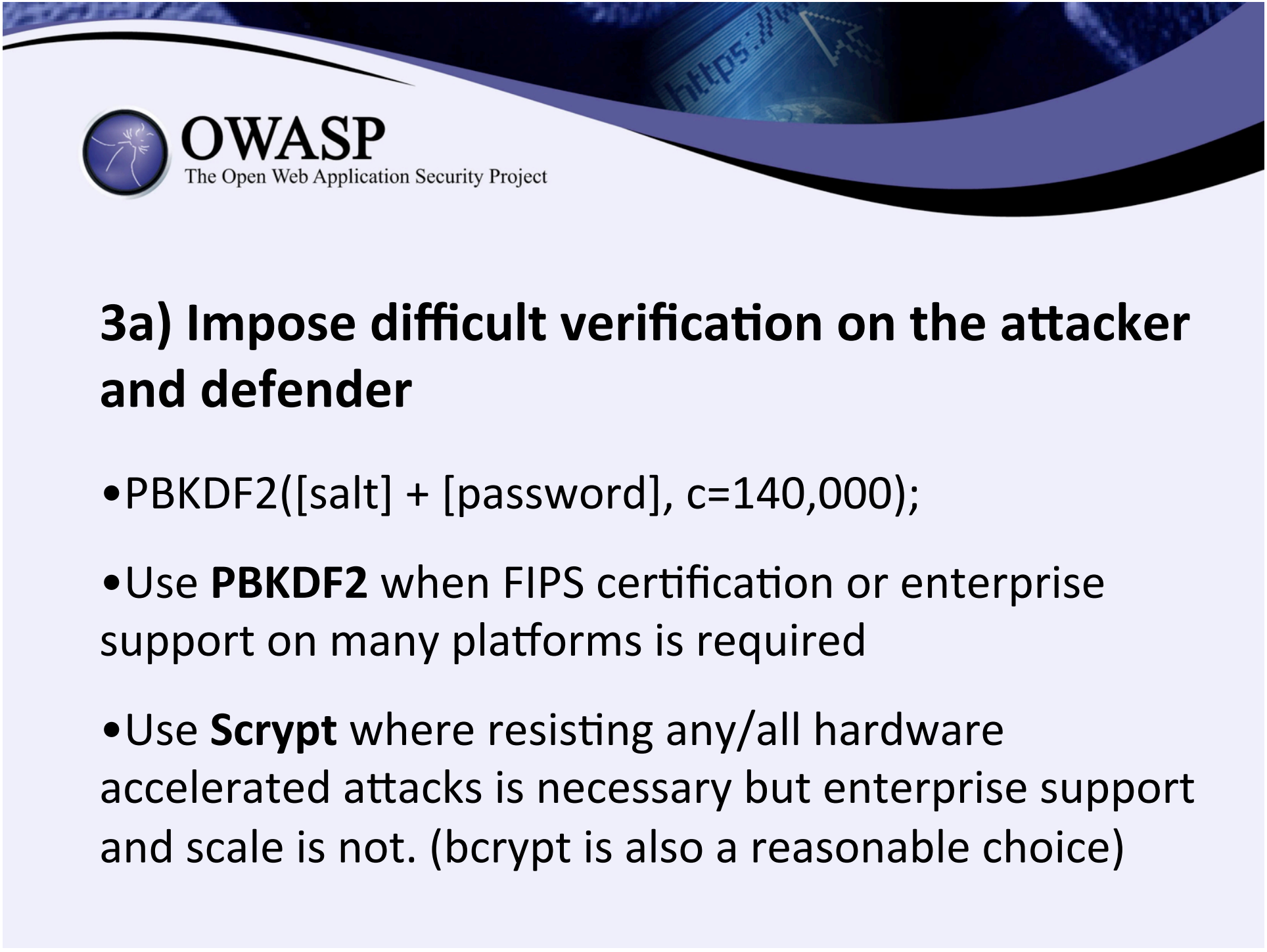
1) Do not limit the type of characters or length of user password within reason

- Limiting passwords to protect against injection is doomed to failure
- Use proper encoder and other defenses described instead
- Be wary of systems that allow unlimited password sizes (Django DOS Sept 2013)



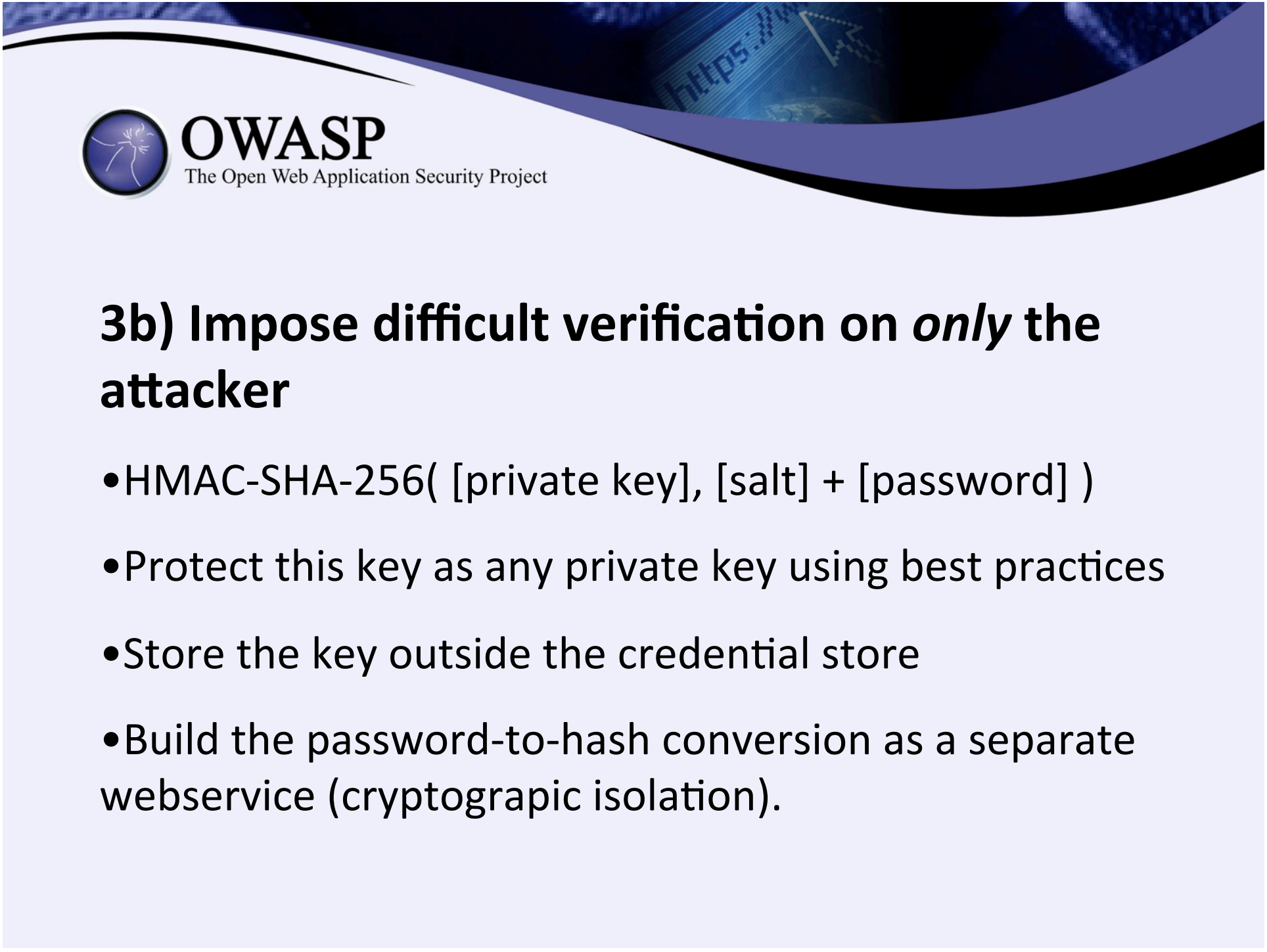
2) Use a cryptographically strong credential-specific salt

- 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

OWASP
The Open Web Application Security Project

DATA PROTECTION AND PRIVACY

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

OWASP
The Open Web Application Security Project

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

OWASP
The Open Web Application Security Project

- Current HSTS Chrome preload list http://src.chromium.org/viewvc/chrome/trunk/src/net/http/transport_security_state_static.json
- If you own a site that you would like to see included in the preloaded Chromium HSTS list, start sending the HSTS header and then contact: https://hstspreload.appspot.com/
- A site is included in the Firefox preload list if the following hold:
  - It is in the Chromium list (with force-https).
  - It sends an HSTS header.
  - The max-age sent is at least 10886400 (18 weeks).
- More info at: http://dev.chromium.org/sts

SecurePasswordStorage

OWASP
The Open Web Application Security Project

```
/*
  in the  doGet() or doPost()
 */

if(request.getScheme().equals("https"))
      {
      // HTTPs so force HSTS
      response.setHeader("Strict-Transport-Security", "max-age=3628800;
includeSubdomains");
      } else {
      // other protocol than HTTPS (HTTP ? )
      response.setStatus(301);
      // Force HTTPS
      // Avoid redirection to pishing site
      String serverName = validateServerName (request.getServerName());
      String serverUrl = validateURL(request.getRequestURL());
      String url = "https://" + serverName + serverURL;
      response.setHeader("Location", url);
      }
```

Perfect Forward Secrecy

OWASP
The Open Web Application Security Project

- If you use older SSL ciphers, every time anyone makes a SSL connection to your server, that message is encrypted with (basically) the same private server key
- Perfect forward secrecy: Peers in a conversation instead negotiate secrets through an ephemeral (temporary) key exchange
- With PFS, recording ciphertext traffic doesn't help an attacker even if the private server key is stolen!

From https://whispersystems.org/blog/asynchronous-security/

SSL/TLS Example Ciphers

OWASP
The Open Web Application Security Project

- Forward Secrecy:

```
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (0xc02f)
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 (0xc027)
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA    (0xc013)
TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (0xc030)
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384 (0xc028)
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA    (0xc014)
```

- NOT Forward Secrecy

```
TLS_RSA_WITH_AES_128_GCM_SHA256 (0x9c)
TLS_RSA_WITH_AES_128_CBC_SHA256 (0x3c)
TLS_RSA_WITH_AES_128_CBC_SHA    (0x2f)
TLS_RSA_WITH_AES_256_GCM_SHA384 (0x9d)
TLS_RSA_WITH_AES_256_CBC_SHA256 (0x3d)
```

OWASP
The Open Web Application Security Project

Solving Real World Crypto Storage Problems With Google KeyCzar

```
Crypter crypter = new Crypter("/path/to/your/keys");
String ciphertext = crypter.encrypt("Secret message");
String plaintext = crypter.decrypt(ciphertext);
```

Keyczar is an open source cryptographic toolkit for Java

Designed to make it easier and safer for developers to use cryptography in their applications.

- A simple API
- Key rotation and versioning
- Safe default algorithms, modes, and key lengths
- Automated generation of initialization vectors and ciphertext signatures
- Java – Python – C++

OWASP
The Open Web Application Security Project

Error Handling, Logging and Intrusion Detection

Risks ?

OWASP
The Open Web Application Security Project

- 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

SecurePasswordStorage

OWASP
The Open Web Application Security Project

```
public class MyAccessControl {
   public boolean accessGrant (Object ressource, Objec user) {
      boolean accessGranted = false;
      try {
         if (myAccessControl().isAuthorize(ressource, user)){
            accessGranted = true;
            //do Stuff  like logging and other opperation than may fail
            operationThatMayFailed (ressource);
         }
      } catch (Exception ex){
         logger.audit ("Access control fail") ;
         // Fail Securely
         accessGranted = false;
      }
      return accessGranted;
   }
}
```

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

OWASP
The Open Web Application Security Project

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)

OWASP
The Open Web Application Security Project

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
The Open Web Application Security Project

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

Better than catching NPE...

OWASP
The Open Web Application Security Project

```
if (s == null) {
      return false;
   }
   String names[] = s.split(" ");
   if (names.length != 2) {
      return false;
   }
   return (isCapitalized(names[0]) && isCapitalized(names[1]));
}
```

Preventing StackTrace leak

OWASP
The Open Web Application Security Project

```
<error-page>
   <exception-type>java.lang.Throwable </exception-type>
   <location>/WEB-INF/error.jsp </location>
</error-page>
```

OWASP
The Open Web Application Security Project

Security Requirements

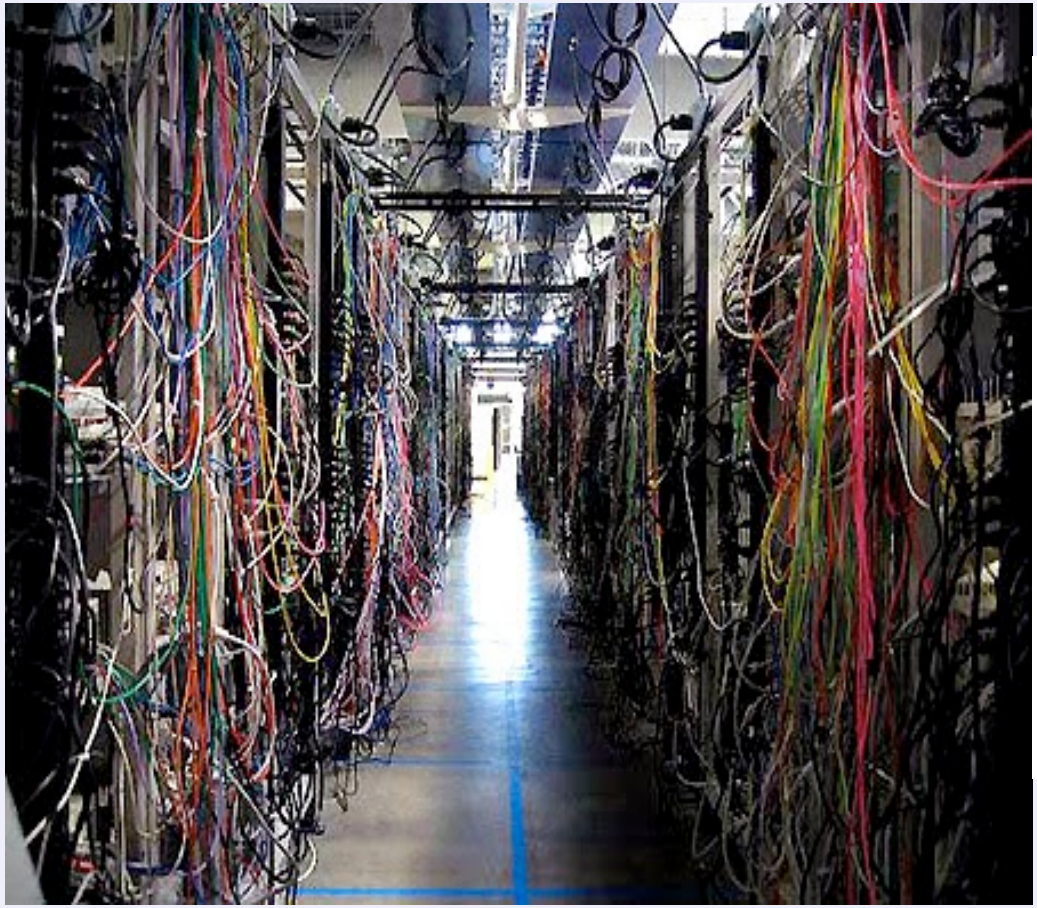
OWASP
The Open Web Application Security Project

OWASP ASVS

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

Bad Design

OWASP
The Open Web Application Security Project



```
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
```

OWASP
The Open Web Application Security Project

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

OWASP
The Open Web Application Security Project

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.

OWASP
The Open Web Application Security Project

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

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 ☺

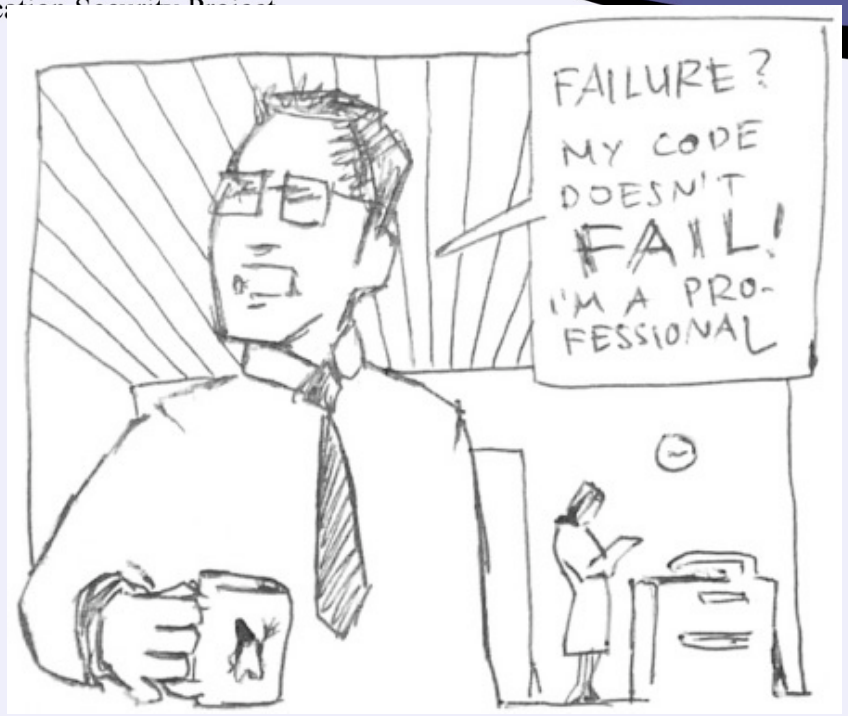
OWASP
The Open Web Application Security Project

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

Nice comments

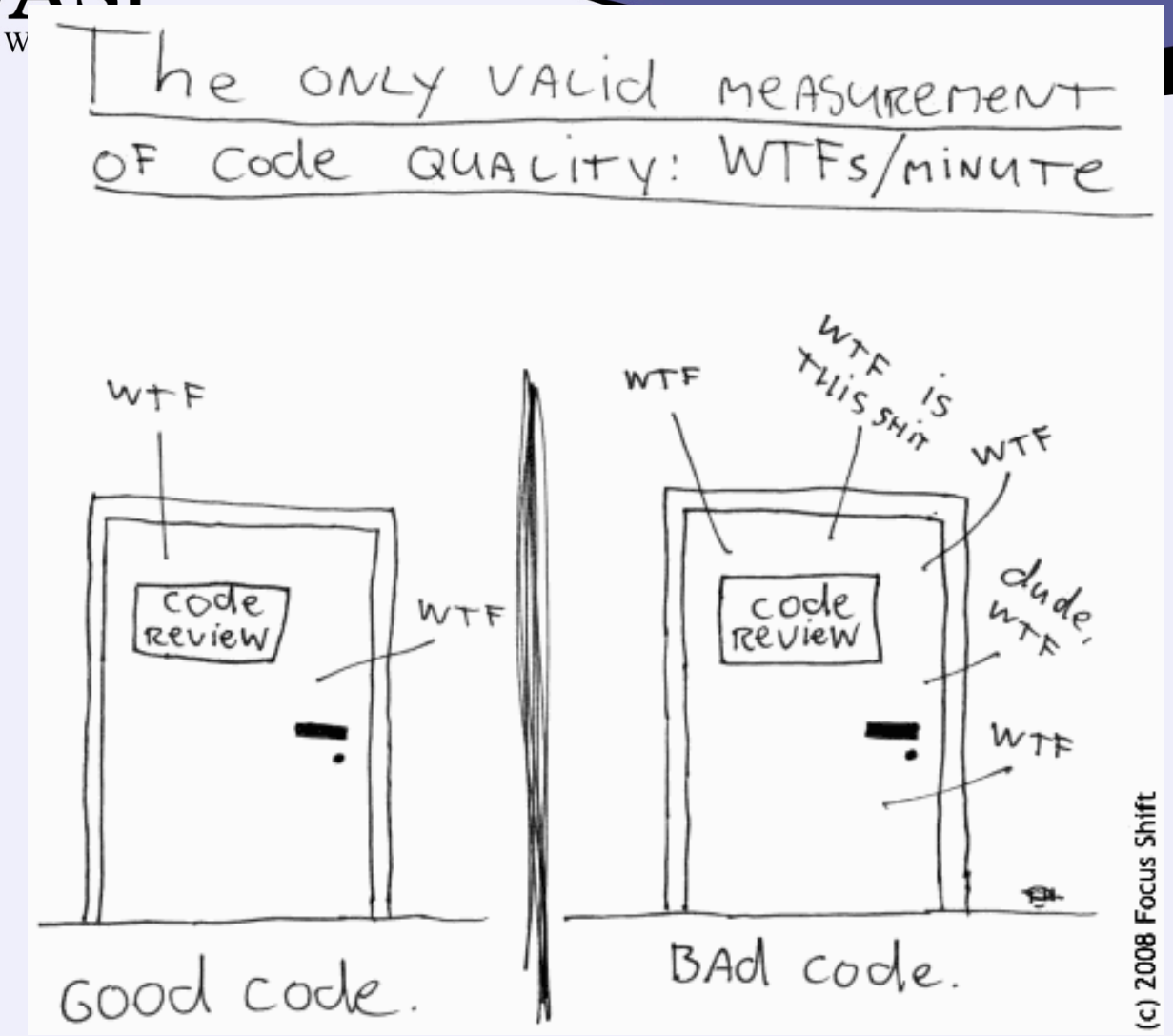

```
// Pattern to match the string
Pattern myPattern = "\bword1\\W+(?:\\w+/W+){1,6}?word2\b";
String updated = MYCONSTANT1.replaceAll(mypattern, "$2");

// 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

Controlling code in one picture



Pentest or code review for CISO

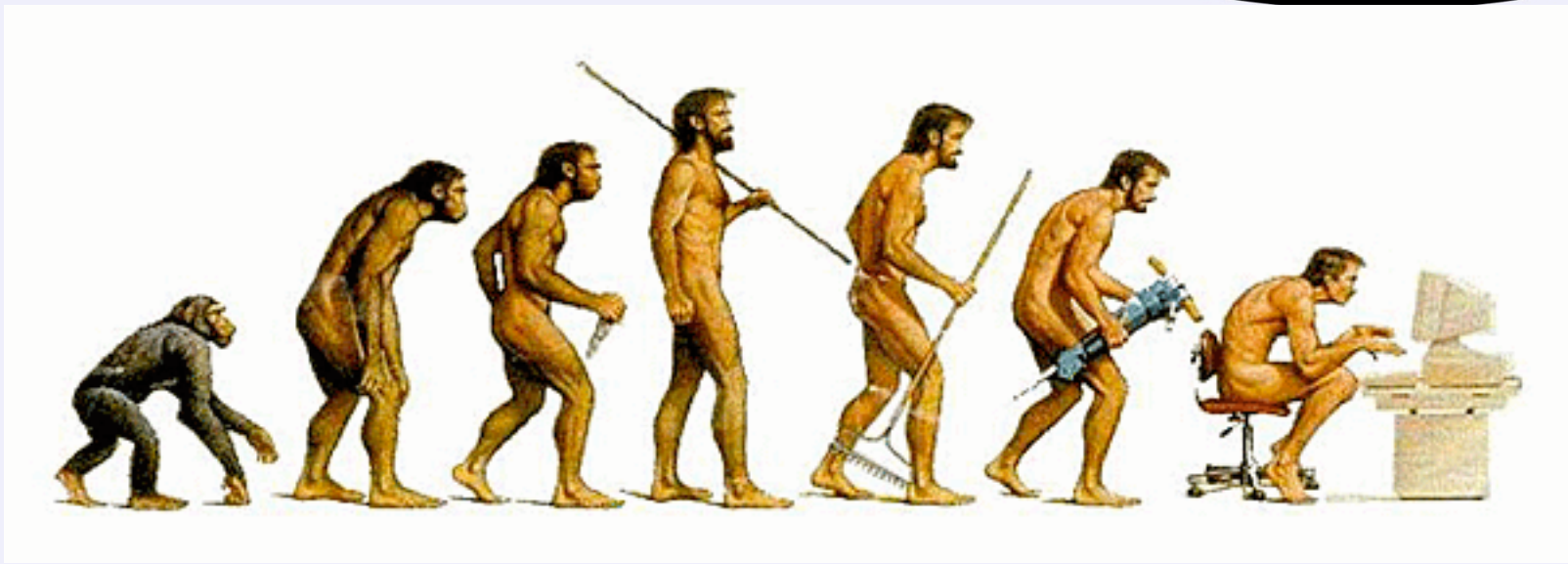
OWASP
The Open Web Application Security Project

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

Evolution of software developement

OWASP
The Open Web Application Security Project



Makefile

Git/CVS/SVN

Continuous Security

Unit testing

Continuous inspection

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

Of Course !

Démo

Thanks

OWASP
The Open Web Application Security Project

- Somes slides from Jim Manico (@manicode) && OWASP Top10 Proactive Project

License

OWASP
The Open Web Application Security Project

- @SPoint

- sebastien.gioria@owasp.org