OWASP

Open Web Application Security Project

OWASP

This is an **online community** devoted to web application security

They create freely available articles, methodologies, tools, documentation and technology in this field

OWASP

www.owasp.org

This is the website devoted to OWASP which you can use to access its resources

OWASP

www.owasp.org

OWASP is **not for profit** and does not make recommendations for commercial products and services

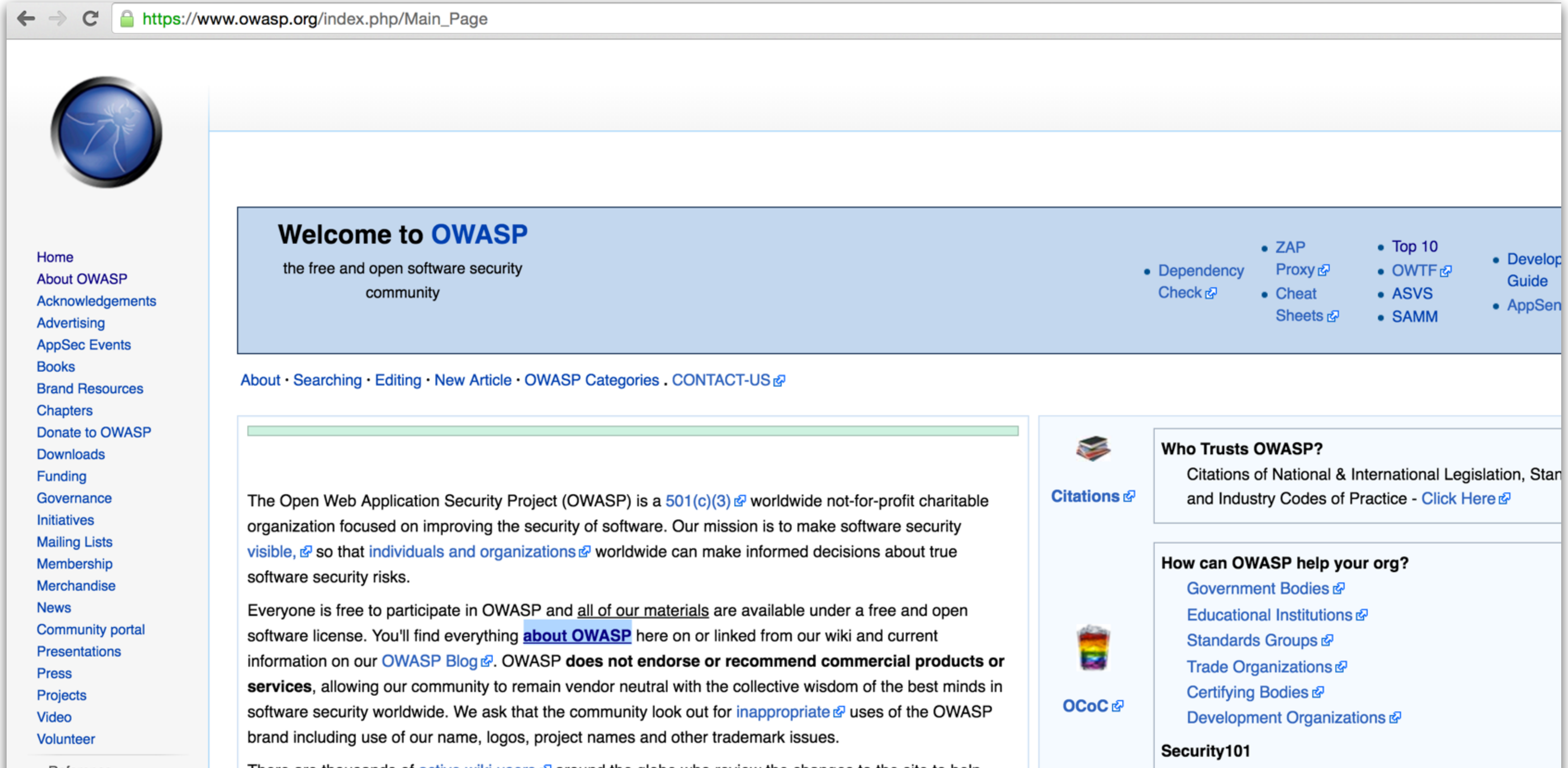
OWASP

www.owasp.org

They are a treasure trove of resources - a lot of the material in this course is using their documentation and examples

OWASP

www.owasp.org



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

Home
About OWASP
Acknowledgements
Advertising
AppSec Events
Books
Brand Resources
Chapters
Donate to OWASP
Downloads
Funding
Governance
Initiatives
Mailing Lists
Membership
Merchandise
News
Community portal
Presentations
Press
Projects
Video
Volunteer

Welcome to OWASP

the free and open software security community

- Dependency Check
- ZAP Proxy
- Cheat Sheets
- Top 10
- OWTF
- ASVS
- SAMM
- Develop Guide
- AppSen

About · Searching · Editing · New Article · OWASP Categories . CONTACT-US

The Open Web Application Security Project (OWASP) is a 501(c)(3) worldwide not-for-profit charitable organization focused on improving the security of software. Our mission is to make software security visible, so that individuals and organizations worldwide can make informed decisions about true software security risks.

Everyone is free to participate in OWASP and all of our materials are available under a free and open software license. You'll find everything about OWASP here on or linked from our wiki and current information on our OWASP Blog. OWASP **does not endorse or recommend commercial products or services**, allowing our community to remain vendor neutral with the collective wisdom of the best minds in software security worldwide. We ask that the community look out for inappropriate uses of the OWASP brand including use of our name, logos, project names and other trademark issues.

Citations

Who Trusts OWASP?

Citations of National & International Legislation, Stan and Industry Codes of Practice - Click Here

OCoC

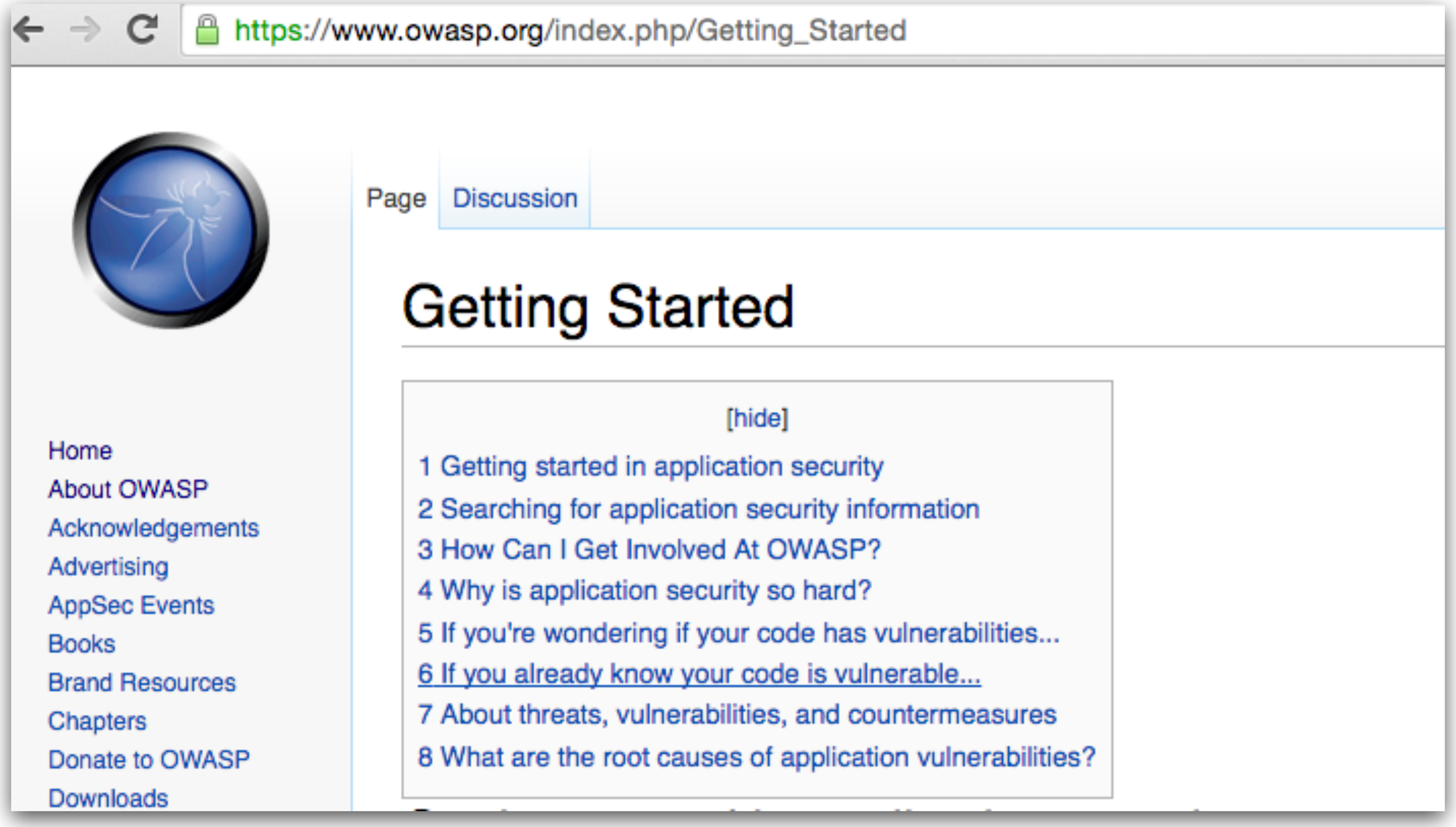
How can OWASP help your org?

Government Bodies
Educational Institutions
Standards Groups
Trade Organizations
Certifying Bodies
Development Organizations

Security101

OWASP

They have a getting started guide which points you to a whole bunch of reading



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

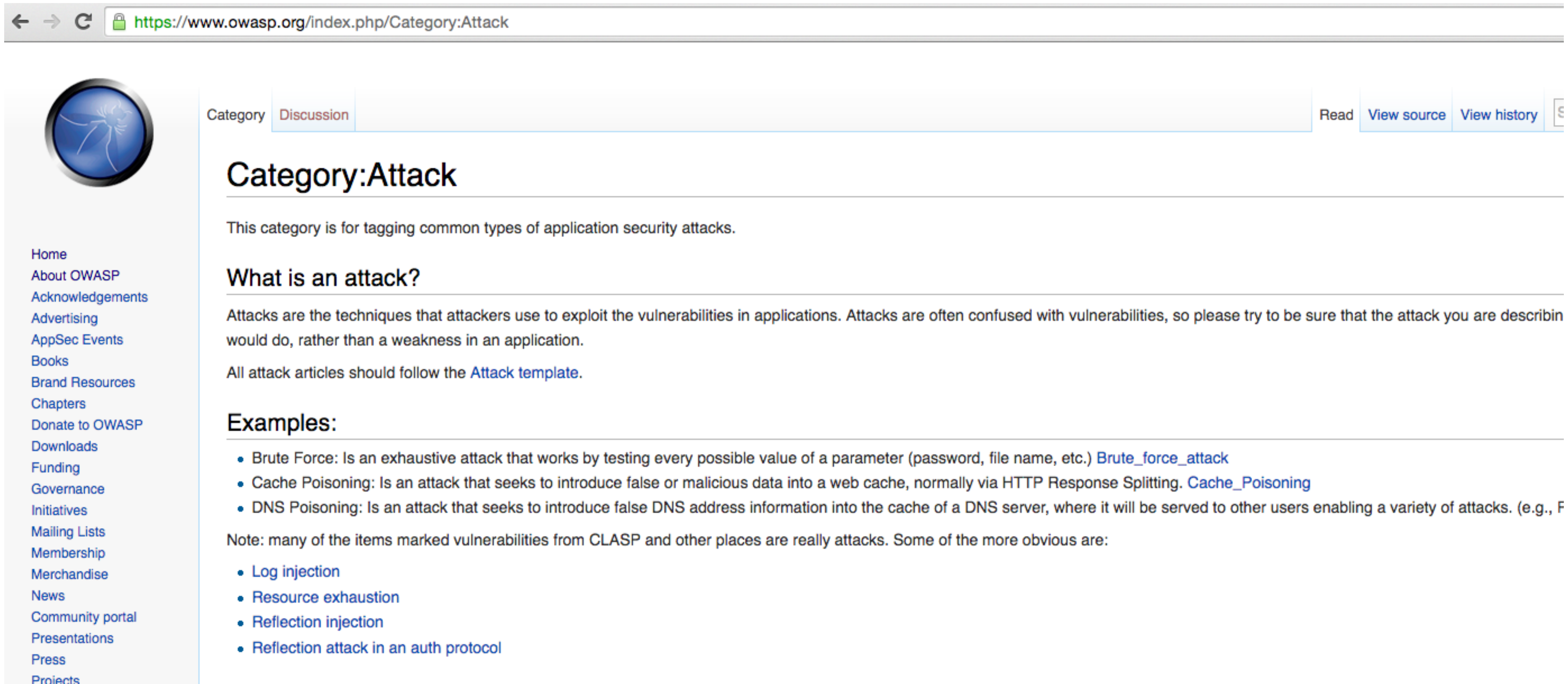
Getting Started

[hide]

1 Getting started in application security
2 Searching for application security information
3 How Can I Get Involved At OWASP?
4 Why is application security so hard?
5 If you're wondering if your code has vulnerabilities...
6 If you already know your code is vulnerable...
7 About threats, vulnerabilities, and countermeasures
8 What are the root causes of application vulnerabilities?

OWASP

Some interesting pages are the ones which have a listing of all possible attacks



Home
About OWASP
Acknowledgements
Advertising
AppSec Events
Books
Brand Resources
Chapters
Donate to OWASP
Downloads
Funding
Governance
Initiatives
Mailing Lists
Membership
Merchandise
News
Community portal
Presentations
Press
Projects

Category:Attack

This category is for tagging common types of application security attacks.

What is an attack?

Attacks are the techniques that attackers use to exploit the vulnerabilities in applications. Attacks are often confused with vulnerabilities, so please try to be sure that the attack you are describin would do, rather than a weakness in an application.

All attack articles should follow the Attack template.

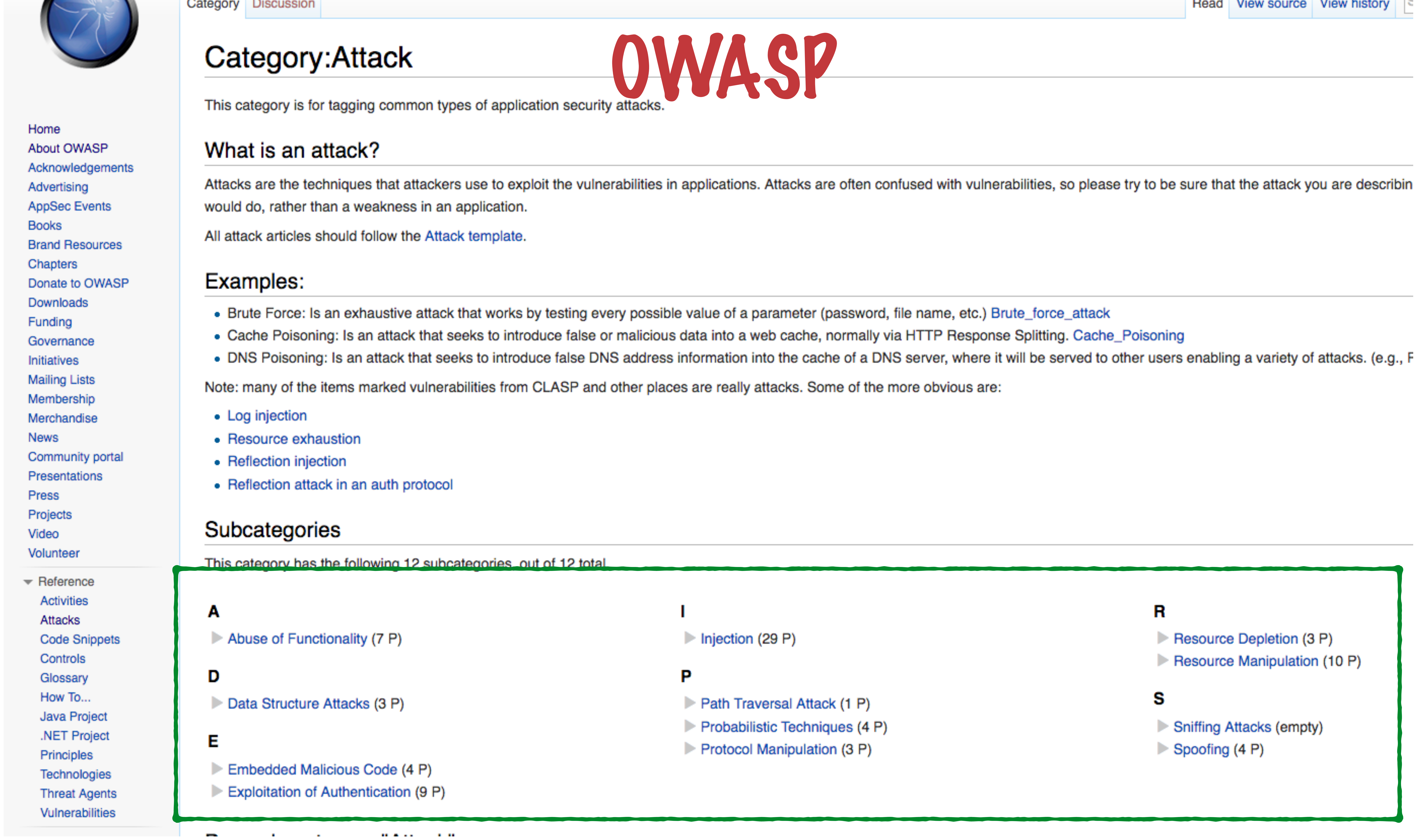
Examples:

- Brute Force: Is an exhaustive attack that works by testing every possible value of a parameter (password, file name, etc.) Brute_force_attack
- Cache Poisoning: Is an attack that seeks to introduce false or malicious data into a web cache, normally via HTTP Response Splitting. Cache_Poisoning
- DNS Poisoning: Is an attack that seeks to introduce false DNS address information into the cache of a DNS server, where it will be served to other users enabling a variety of attacks. (e.g., F

Note: many of the items marked vulnerabilities from CLASP and other places are really attacks. Some of the more obvious are:

- Log injection
- Resource exhaustion
- Reflection injection
- Reflection attack in an auth protocol

OWASP



Category:Attack

This category is for tagging common types of application security attacks.

What is an attack?

Attacks are the techniques that attackers use to exploit the vulnerabilities in applications. Attacks are often confused with vulnerabilities, so please try to be sure that the attack you are describin would do, rather than a weakness in an application.

All attack articles should follow the Attack template.

Examples:

- Brute Force: Is an exhaustive attack that works by testing every possible value of a parameter (password, file name, etc.) Brute_force_attack
- Cache Poisoning: Is an attack that seeks to introduce false or malicious data into a web cache, normally via HTTP Response Splitting. Cache_Poisoning
- DNS Poisoning: Is an attack that seeks to introduce false DNS address information into the cache of a DNS server, where it will be served to other users enabling a variety of attacks. (e.g., F

Note: many of the items marked vulnerabilities from CLASP and other places are really attacks. Some of the more obvious are:

- Log injection
- Resource exhaustion
- Reflection injection
- Reflection attack in an auth protocol

Subcategories

This category has the following 12 subcategories, out of 12 total.

A
- Abuse of Functionality (7 P)

D
- Data Structure Attacks (3 P)

E
- Embedded Malicious Code (4 P)
- Exploitation of Authentication (9 P)

I
- Injection (29 P)

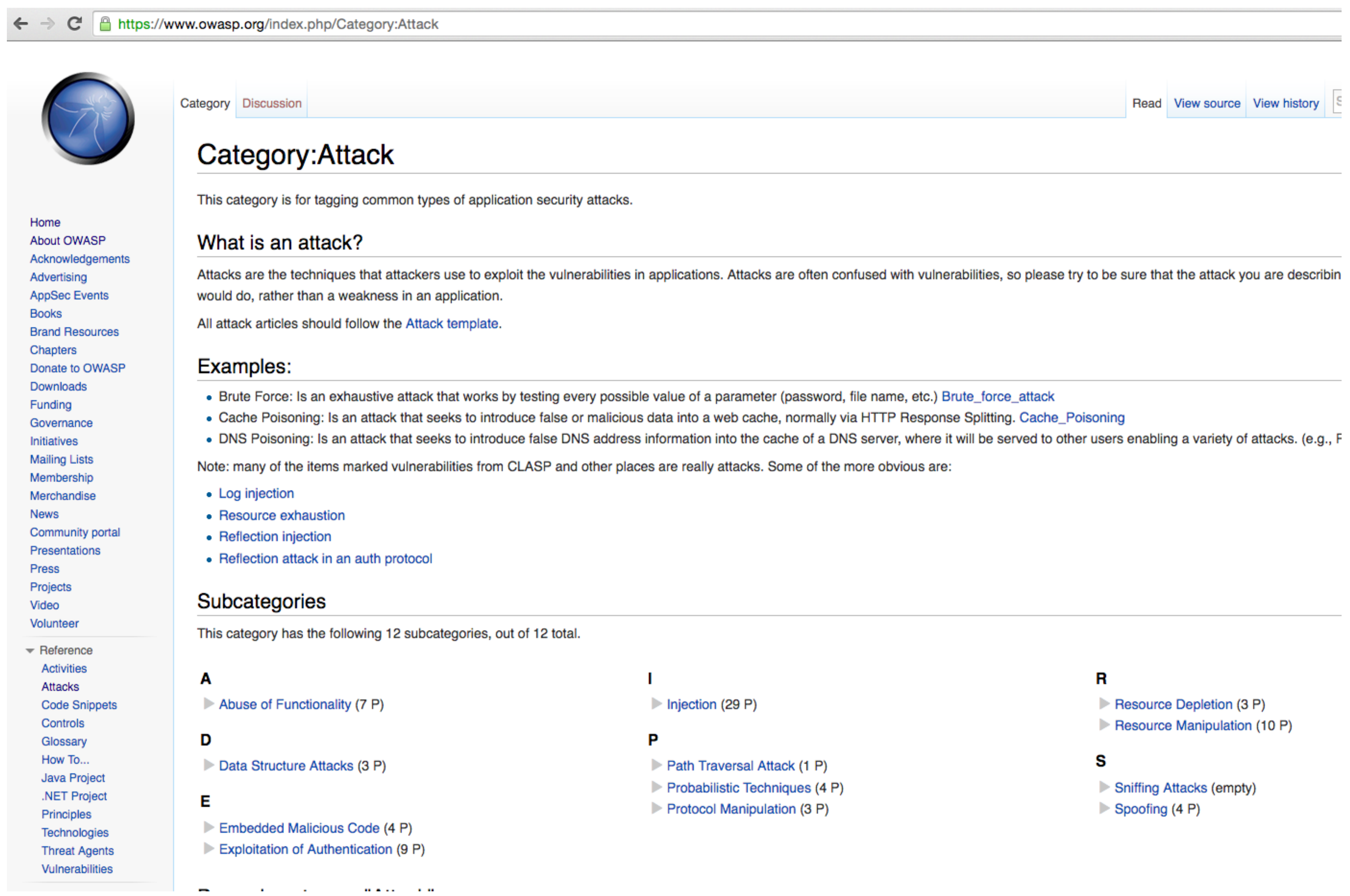
P
- Path Traversal Attack (1 P)
- Probabilistic Techniques (4 P)
- Protocol Manipulation (3 P)

R
- Resource Depletion (3 P)
- Resource Manipulation (10 P)

S
- Sniffing Attacks (empty)
- Spoofing (4 P)

OWASP

Not all attacks may have detailed write ups but it's a handy lookup


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

Category Discussion Read View source View history

Home
About OWASP
Acknowledgements
Advertising
AppSec Events
Books
Brand Resources
Chapters
Donate to OWASP
Downloads
Funding
Governance
Initiatives
Mailing Lists
Membership
Merchandise
News
Community portal
Presentations
Press
Projects
Video
Volunteer

Reference
Activities
Attacks
Code Snippets
Controls
Glossary
How To...
Java Project
.NET Project
Principles
Technologies
Threat Agents
Vulnerabilities

Category:Attack

This category is for tagging common types of application security attacks.

What is an attack?

Attacks are the techniques that attackers use to exploit the vulnerabilities in applications. Attacks are often confused with vulnerabilities, so please try to be sure that the attack you are describin
would do, rather than a weakness in an application.

All attack articles should follow the Attack template.

Examples:

- Brute Force: Is an exhaustive attack that works by testing every possible value of a parameter (password, file name, etc.) Brute_force_attack
- Cache Poisoning: Is an attack that seeks to introduce false or malicious data into a web cache, normally via HTTP Response Splitting. Cache_Poisoning
- DNS Poisoning: Is an attack that seeks to introduce false DNS address information into the cache of a DNS server, where it will be served to other users enabling a variety of attacks. (e.g., F

Note: many of the items marked vulnerabilities from CLASP and other places are really attacks. Some of the more obvious are:

- Log injection
- Resource exhaustion
- Reflection injection
- Reflection attack in an auth protocol

Subcategories

This category has the following 12 subcategories, out of 12 total.

A
- Abuse of Functionality (7 P)

D
- Data Structure Attacks (3 P)

E
- Embedded Malicious Code (4 P)
- Exploitation of Authentication (9 P)

I
- Injection (29 P)

P
- Path Traversal Attack (1 P)
- Probabilistic Techniques (4 P)
- Protocol Manipulation (3 P)

R
- Resource Depletion (3 P)
- Resource Manipulation (10 P)

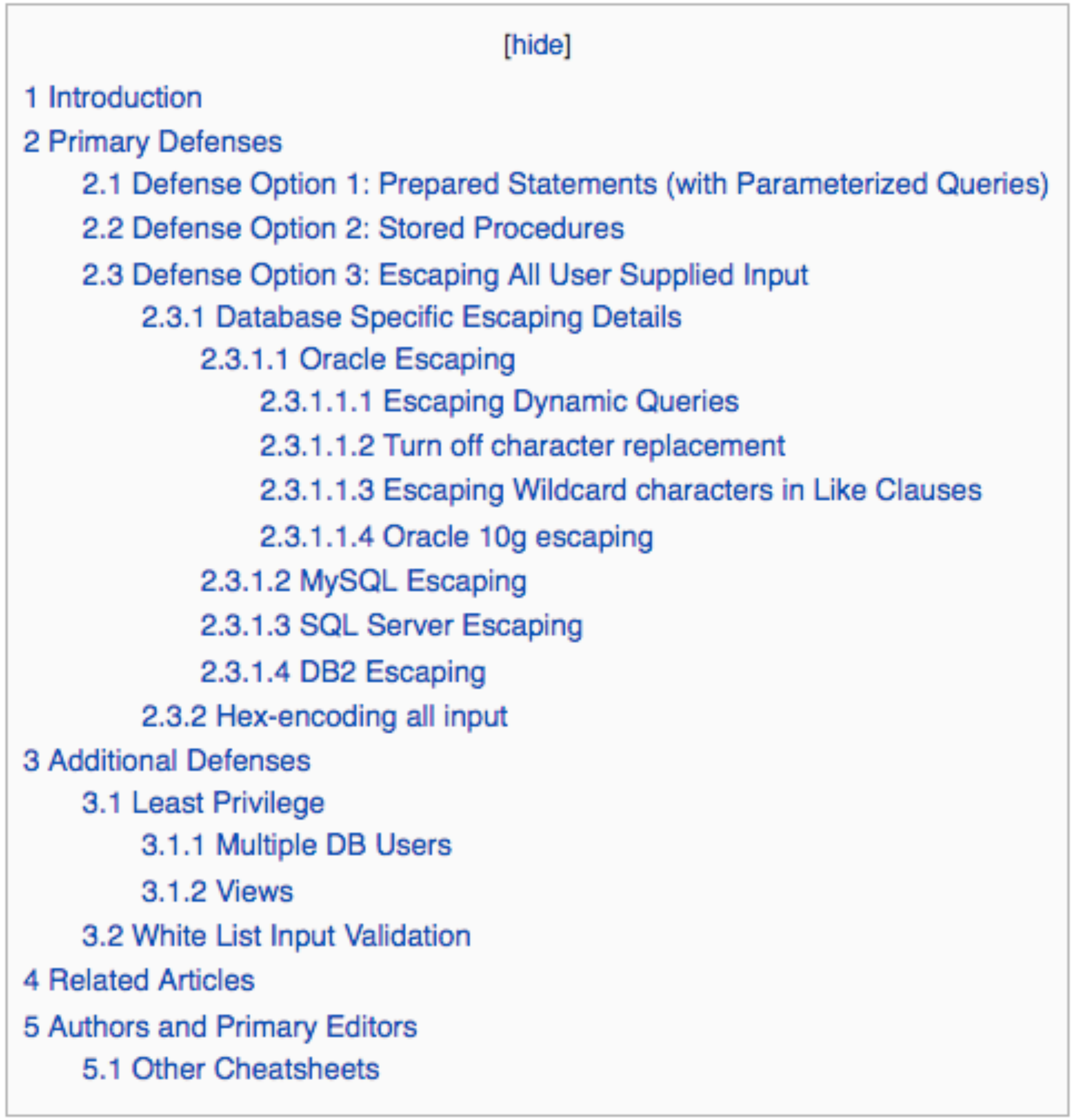
S
- Sniffing Attacks (empty)
- Spoofing (4 P)

OWASP

What is really useful though are the **cheat sheets**

Most attacks have a cheat sheet which has a **basic description** of the attack and the **defense** mechanisms to use

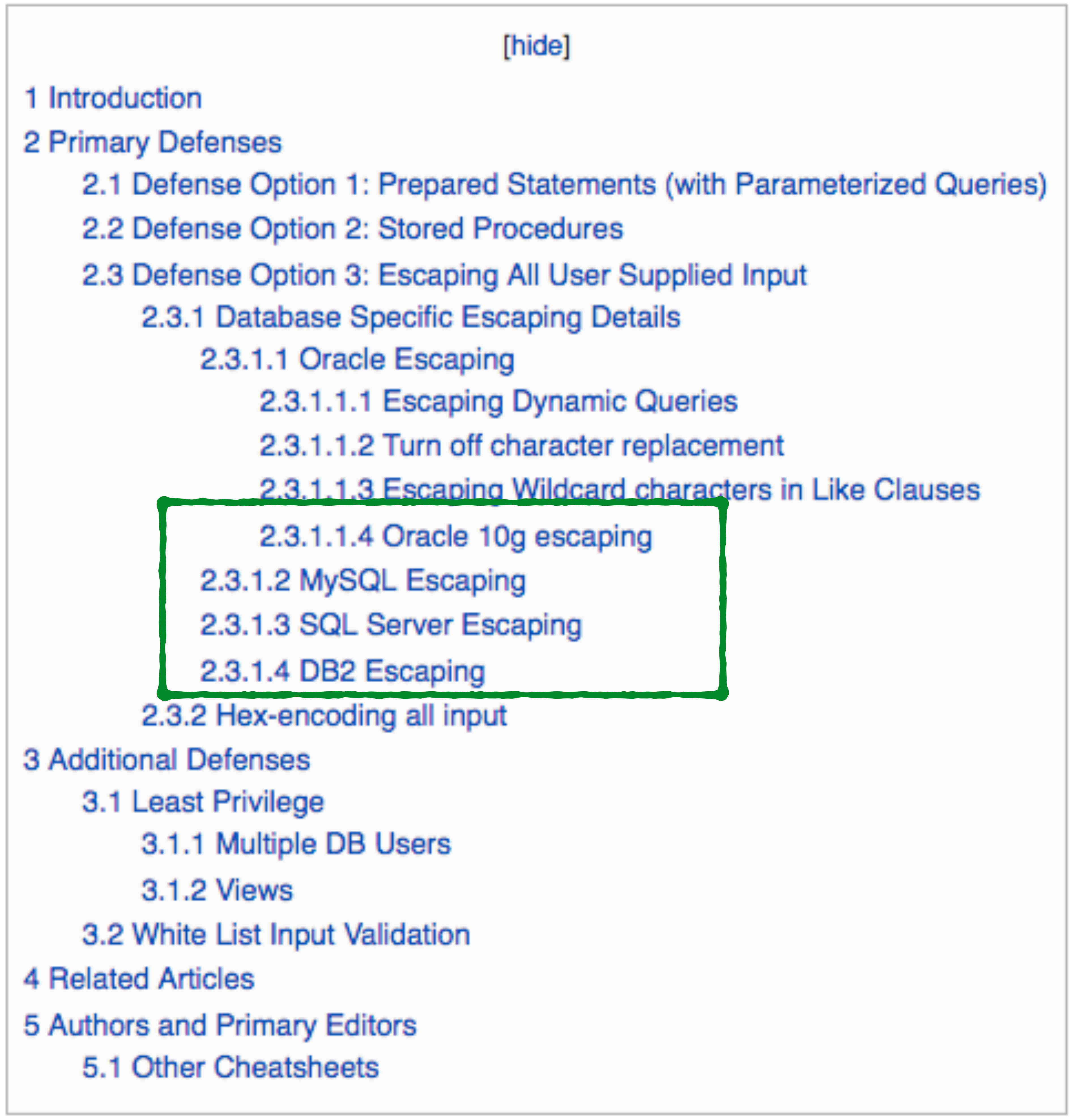
OWASP
SQL injection cheatsheet



[hide]

1 Introduction
2 Primary Defenses
  2.1 Defense Option 1: Prepared Statements (with Parameterized Queries)
  2.2 Defense Option 2: Stored Procedures
  2.3 Defense Option 3: Escaping All User Supplied Input
    2.3.1 Database Specific Escaping Details
      2.3.1.1 Oracle Escaping
        2.3.1.1.1 Escaping Dynamic Queries
        2.3.1.1.2 Turn off character replacement
        2.3.1.1.3 Escaping Wildcard characters in Like Clauses
        2.3.1.1.4 Oracle 10g escaping
      2.3.1.2 MySQL Escaping
      2.3.1.3 SQL Server Escaping
      2.3.1.4 DB2 Escaping
    2.3.2 Hex-encoding all input
3 Additional Defenses
  3.1 Least Privilege
    3.1.1 Multiple DB Users
    3.1.2 Views
  3.2 White List Input Validation
4 Related Articles
5 Authors and Primary Editors
  5.1 Other Cheatsheets

OWASP



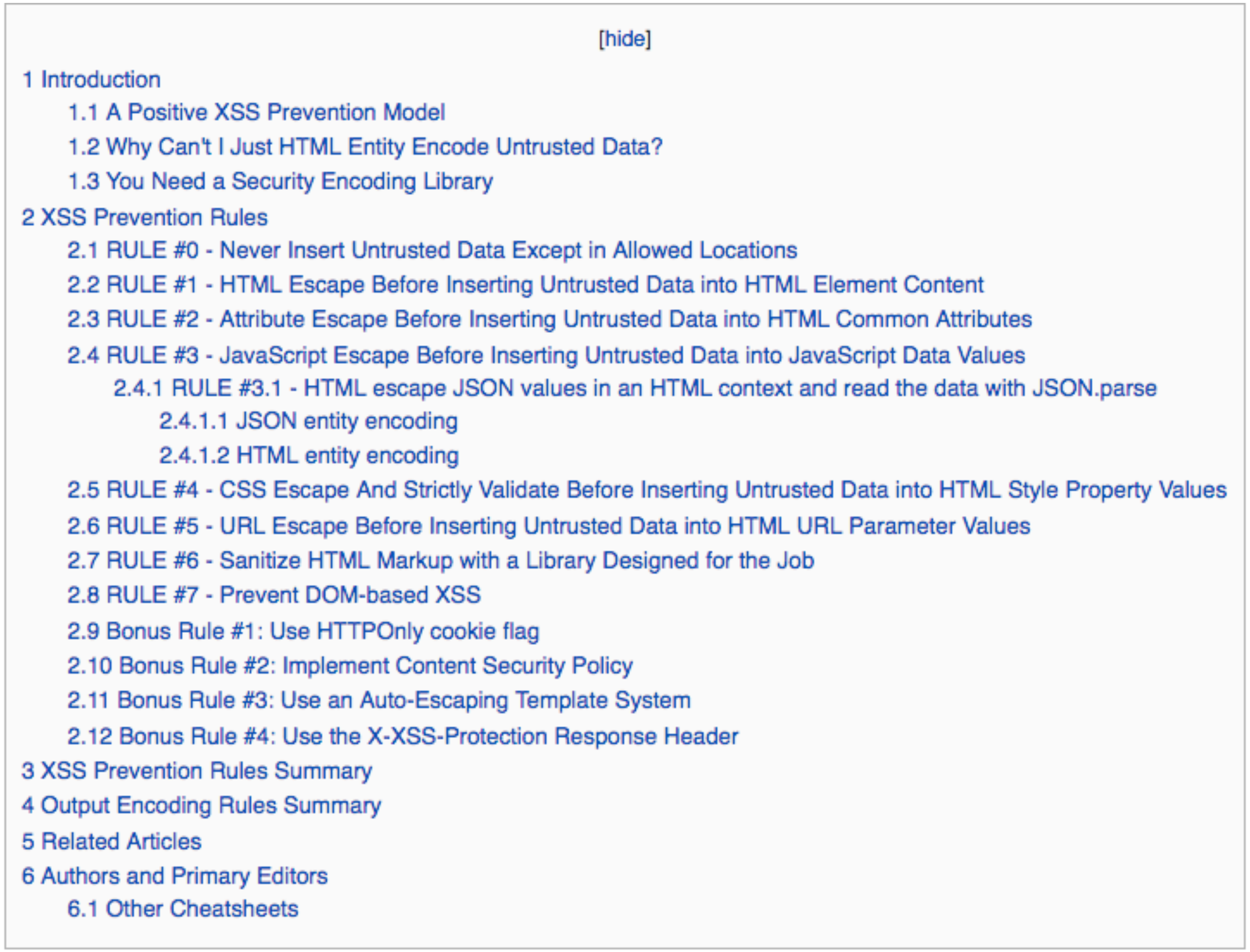
[hide]

1 Introduction
2 Primary Defenses
  2.1 Defense Option 1: Prepared Statements (with Parameterized Queries)
  2.2 Defense Option 2: Stored Procedures
  2.3 Defense Option 3: Escaping All User Supplied Input
    2.3.1 Database Specific Escaping Details
      2.3.1.1 Oracle Escaping
        2.3.1.1.1 Escaping Dynamic Queries
        2.3.1.1.2 Turn off character replacement
        2.3.1.1.3 Escaping Wildcard characters in Like Clauses
        2.3.1.1.4 Oracle 10g escaping
      2.3.1.2 MySQL Escaping
      2.3.1.3 SQL Server Escaping
      2.3.1.4 DB2 Escaping
    2.3.2 Hex-encoding all input
3 Additional Defenses
  3.1 Least Privilege
    3.1.1 Multiple DB Users
    3.1.2 Views
  3.2 White List Input Validation
4 Related Articles
5 Authors and Primary Editors
  5.1 Other Cheatsheets

The most widespread vulnerabilities have specific details which a developer would find useful

OWASP

Cross Site Scripting cheatsheet



[hide]

1 Introduction
- 1.1 A Positive XSS Prevention Model
- 1.2 Why Can't I Just HTML Entity Encode Untrusted Data?
- 1.3 You Need a Security Encoding Library

2 XSS Prevention Rules
- 2.1 RULE #0 - Never Insert Untrusted Data Except in Allowed Locations
- 2.2 RULE #1 - HTML Escape Before Inserting Untrusted Data into HTML Element Content
- 2.3 RULE #2 - Attribute Escape Before Inserting Untrusted Data into HTML Common Attributes
- 2.4 RULE #3 - JavaScript Escape Before Inserting Untrusted Data into JavaScript Data Values
  - 2.4.1 RULE #3.1 - HTML escape JSON values in an HTML context and read the data with JSON.parse
    - 2.4.1.1 JSON entity encoding
    - 2.4.1.2 HTML entity encoding
- 2.5 RULE #4 - CSS Escape And Strictly Validate Before Inserting Untrusted Data into HTML Style Property Values
- 2.6 RULE #5 - URL Escape Before Inserting Untrusted Data into HTML URL Parameter Values
- 2.7 RULE #6 - Sanitize HTML Markup with a Library Designed for the Job
- 2.8 RULE #7 - Prevent DOM-based XSS
- 2.9 Bonus Rule #1: Use HTTPOnly cookie flag
- 2.10 Bonus Rule #2: Implement Content Security Policy
- 2.11 Bonus Rule #3: Use an Auto-Escaping Template System
- 2.12 Bonus Rule #4: Use the X-XSS-Protection Response Header

3 XSS Prevention Rules Summary

4 Output Encoding Rules Summary

5 Related Articles

6 Authors and Primary Editors
- 6.1 Other Cheatsheets

OWASP

OWASP also publishes the **top 10** security vulnerabilities

Here is the last published list for 2013

OWASP

10. Unvalidated redirects and forwards

9. Using components with known vulnerabilities

8. Cross site request forgery (XSRF)

10. Unvalidated redirects and forwards
9. Using components with known vulnerabilities
8. Cross site request forgery (XSRF)

OWASP

7. Missing function level access control

6. Sensitive data exposure

5. Security misconfiguration

10. Unvalidated redirects and forwards
9. Using components with known vulnerabilities
8. cross site request forgery (XSRF)
7. Missing function level access control
6. Sensitive data exposure
5. Security misconfiguration

OWASP

4. direct object reference

3. Cross Site Scripting (XSS)

10. Unvalidated redirects and forwards
9. Using components with known vulnerabilities
8. cross site request forgery (XSRF)
7. Missing function level access control
6. Sensitive data exposure
5. Security misconfiguration
4. Direct object Reference
3. cross Site Scripting (XSS)

OWASP

2. broken authentication and session management

10. Unvalidated redirects and forwards

9. Using components with known vulnerabilities

8. cross site request forgery (XSRF)

7. Missing function level access control

6. Sensitive data exposure

5. Security misconfiguration

4. Direct object Reference

3. cross Site Scripting (XSS)

2. Broken authentication and session management

OWASP

1. injection (SQLi)

OWASP

1. Injection (SQLi)
2. Broken authentication and session management
3. Cross Site Scripting (XSS)
4. Direct object Reference
5. Security misconfiguration
6. Sensitive data exposure
7. Missing function level access control
8. Cross site request forgery (XSRF)
9. Using components with known vulnerabilities
10. Unvalidated redirects and forwards

OWASP

Overall a web developer looking to make her code secure will have lots to learn here

2 FACTOR AUTHENTICATION

2 FACTOR AUTHENTICATION

This is also known as **2FA** or **2 Step Verification**

This was patented way back in 1984 but found widespread use on web applications recently

2 FACTOR AUTHENTICATION

This enables confirmation of a user's identity by a combination of components

Something a user **knows**

Something a user **possesses**

Something **inseparable** from the user

2 FACTOR AUTHENTICATION

Something a user **knows**

Password, username, PIN, TAN

Something a user **possesses**

Something **inseparable** from the user

2 FACTOR AUTHENTICATION

Something a user **possesses**

Secret token USB, bank card, key

Something **inseparable** from the user

Something a user **knows**

2 FACTOR AUTHENTICATION

Something **inseparable** from the user

user biometrics such as fingerprint, eye iris, voice, typing speed

Something a user **knows**
Something a user **possesses**

2 FACTOR AUTHENTICATION

A very common example of 2 factor authentication is the use of **ATMs**

Withdrawing money from a bank account requires **2 pieces** of information

2 FACTOR AUTHENTICATION

Withdrawing money from a bank account requires **2 pieces** of information

The combination of:

1. A bank card i.e debit card or ATM card

2. A valid PIN number

2 FACTOR AUTHENTICATION

1. A bank card i.e debit card or ATM card
2. A valid PIN number

Only if the match is valid will the transaction be successful!

2 FACTOR AUTHENTICATION

2 factor authentication is a type of **multi-factor** authentication

Multi-factor authentication is a strong defense against online **identity theft and fraud**

2 FACTOR AUTHENTICATION

Multi-factor authentication is a strong defense against online identity theft and fraud

A password alone is no longer enough to get into a system which has sensitive data and perform actions

2 FACTOR AUTHENTICATION

Implementation considerations:

2 factor authentication may require **additional client software** to be installed to get things to work

e.g. software to use the token or smart card

2 FACTOR AUTHENTICATION

Implementation considerations:

Or a hardware based approach using **hardware token** products could be used

These provide a **logistical challenge** when they have to be issued in large numbers

2 FACTOR AUTHENTICATION

Implementation considerations:

hardware token

logistical challenge

They require additional **investment** for implementation and maintenance

2 FACTOR AUTHENTICATION

Implementation considerations:

hardware token

logistical challenge
investment

They also possibly require **support** - when users get locked out of their systems or lose their tokens

2 FACTOR AUTHENTICATION

Implementation considerations:

hardware token

- logistical challenge
- investment
- support

All in all 2 factor requires commitment and is not cheap by any means

2 FACTOR AUTHENTICATION

mobile phone based 2 factor authentication seems attractive

No additional hardware tokens are necessary and the the user is **always in possession** of her mobile phone

2 FACTOR AUTHENTICATION

mobile based 2 factor authentication

The user enters a password or a pin at a website

An additional **dynamic** passcode comprising of digits is sent to the user's mobile phone via **SMS** or an installed application

2 FACTOR AUTHENTICATION

mobile based 2 factor authentication

This passcode is called a **OTP** or a One Time Password

This is generated by a time-based one time password algorithm

OTP

2 FACTOR AUTHENTICATION

mobile based 2 factor authentication

The generation of the OTP uses a **shared secret key** and the **current time**

It's for **one time** use - if entered once it's no longer valid

If the token expires - it's no longer valid

2 FACTOR AUTHENTICATION

Advantages of mobile based 2FA

No additional hardware needed

Safer than static login information

Have fixed expiry and one time use

Easy to configure and easy to use

2 FACTOR AUTHENTICATION

Disadvantages of mobile based 2FA

Cellphone needs to be **charged** and needs to be **in range**

User needs to **share the phone number** with the OTP provider

Text messages are **insecure** and can be intercepted

Smartphones have **both email and SMS** so a loss of the phone means all accounts for which email is the key can be hacked - **2 factors become 1 factor**

Malware on the phone can steal credentials

SOCIAL ENGINEERING

SOCIAL ENGINEERING

Social engineering is the art of **manipulating** people so they give up **confidential** information

SOCIAL ENGINEERING

Social engineering is the art of manipulating people so they give up confidential information

Users can be made to give up passwords, banking information, install malicious software anything

SOCIAL ENGINEERING

Social engineering is the art of manipulating people so they give up confidential information

Users can be made to give up passwords, banking information, install malicious software anything

Attackers like social engineering because it is much easier to **exploit a user's trust** than to find ways to hack into software

SOCIAL ENGINEERING

Social engineering is the art of manipulating people so they give up confidential information

Users can be made to give up passwords, banking information, install malicious software anything

Attackers like social engineering because it is much easier to **exploit a user's trust** than to find ways to hack into software

It's easier to **trick** you into handing over your password than trying to figure out what it is

Security is all about:

Knowing who and what to trust

When to trust that the person you're communicating with is indeed the person he claims to be

When to trust whether the website is legitimate or not

Knowing when providing your information is a good idea

SOCIAL ENGINEERING

Security is all about:

Knowing who and what to trust

Whether to trust that the person you are communicating with is indeed the person he claims to be

When to trust whether the website is legitimate or not

Knowing when providing your information is a good idea

The weakest link in the chain is always the human being - this is why social engineering works!

SOCIAL ENGINEERING

Common social engineering attacks

Phishing

Baiting

Clickjacking

SOCIAL ENGINEERING

Phishing

Phishing is an attempt to get sensitive information from users by **masquerading as a trustworthy entity**

A bank, school, a friend

Clickjacking   Baiting

SOCIAL ENGINEERING

Phishing

You might receive an email from **your mail provider** asking you to mail your password to them

Or from **your bank** asking you to reset your password using a malicious link

Clickjacking Baiting

SOCIAL ENGINEERING

Phishing

Or from another country asking you to give your bank account number so they can transfer funds

Clickjacking Baiting

Phishing

When phishing uses your personal information it's infinitely more successful

It's termed **spear phishing**

Clickjacking Baiting

SOCIAL ENGINEERING

Phishing

Phishing mails might ask for your help, declare you a winner, or ask you to verify information

Clickjacking Baiting

SOCIAL ENGINEERING

Phishing

Clone phishing is when the mail mimics a legitimate mail which was sent earlier

The look and feel and the email address from which the phishing mail is sent is **very similar to the original**

Clickjacking Baiting

Phishing

Phishing attacks often target CEOs of companies or other highly placed officials - this is called **whaling**

Clickjacking Baiting

Baiting

This involves offering something the user would like to have to **bait** them to click on stuff

A new movie for download, free coupons etc

Baiting

Baiting could also pretend to be responding to your request for help e.g. for a software that you use

Clickjacking

This is a technique to get the user to click on something which is **different** from what the user **perceives** he is clicking on

Clickjacking

This is a version of the **confused deputy** problem where a user is fooled into misusing his authority

Clickjacking

Harmless features of HTML can be heavily misused

Clickjacking

A user clicks on a **concealed** link (matches with the page background)

Clickjacking

A user clicks on a concealed link
(matches with the page background)

Another page is loaded in a **transparent** layer over the existing page

SOCIAL ENGINEERING

A user clicks on a concealed link (matches with the page background)

Clickjacking

Another page is loaded in a transparent layer over the existing page

All actions on the page that you see are actually **malicious actions on the transparent layer**

Baiting Phishing

Clickjacking

A user clicks on a concealed link
(matches with the page background)

Another page is loaded in a transparent
layer over the existing page

All actions on the page that you
see are actually malicious actions
on the transparent layer