

# Sécurité des Applications

Partie WEB

---

Stéphane Küng

November 17, 2020

# OWASP Foundation Projects

---

# OWASP Projects (OWASP)

Open Web Application Security Project (OWASP) is a nonprofit foundation that works to improve the security of software.

- OWASP Top 10 Proactive Security Controls For Software Developers to Build Secure Software
- OWASP Web Security Testing Guide
- OWASP Top 10

# OWASP Top 10 Proactive Security Controls

## The Top 10 Proactive Controls

The list is ordered by importance with list item number 1 being the most important:

- C1: Define Security Requirements
- C2: Leverage Security Frameworks and Libraries
- C3: Secure Database Access
- C4: Encode and Escape Data
- C5: Validate All Inputs
- C6: Implement Digital Identity
- C7: Enforce Access Controls
- C8: Protect Data Everywhere
- C9: Implement Security Logging and Monitoring
- C10: Handle All Errors and Exceptions

## 4. Web Application Security Testing

### 4.1 Introduction and Objectives

#### 4.1.1 Testing Checklist

### 4.2 Information Gathering

#### 4.2.1 Conduct Search Engine Discovery and Reconnaissance for Information Leakage (WSTG-INFO-001)

#### 4.2.2 Fingerprint Web Server (WSTG-INFO-002)

#### 4.2.3 Review Webserver Metafiles for Information Leakage (WSTG-INFO-003)

#### 4.2.4 Enumerate Applications on Webserver (WSTG-INFO-004)

#### 4.2.5 Review Webpage Comments and Metadata for Information Leakage (WSTG-INFO-005)

#### 4.2.6 Identify application entry points (WSTG-INFO-006)

#### 4.2.7 Map execution paths through application (WSTG-INFO-007)

#### 4.2.8 Fingerprint Web Application Framework (WSTG-INFO-008)

#### 4.2.9 Fingerprint Web Application (WSTG-INFO-009)

#### 4.2.10 Map Application Architecture (WSTG-INFO-010)

# OWASP Top 10

OWASP Top 10 - 2013	➔	OWASP Top 10 - 2017
A1 – Injection	➔	A1:2017-Injection
A2 – Broken Authentication and Session Management	➔	A2:2017-Broken Authentication
A3 – Cross-Site Scripting (XSS)	➔	A3:2017-Sensitive Data Exposure
A4 – Insecure Direct Object References [Merged+A7]	U	A4:2017-XML External Entities (XXE) [NEW]
A5 – Security Misconfiguration	➔	A5:2017-Broken Access Control [Merged]
A6 – Sensitive Data Exposure	➔	A6:2017-Security Misconfiguration
A7 – Missing Function Level Access Contr [Merged+A4]	U	A7:2017-Cross-Site Scripting (XSS)
A8 – Cross-Site Request Forgery (CSRF)	☒	A8:2017-Insecure Deserialization [NEW, Community]
A9 – Using Components with Known Vulnerabilities	➔	A9:2017-Using Components with Known Vulnerabilities
A10 – Unvalidated Redirects and Forwards	☒	A10:2017-Insufficient Logging&Monitoring [NEW,Comm.]

# Rappel

---

# Escaping vs Encoding

Encoding is transforming data from one format into another format.

- **HTML:** `&lt;h1&gt;Hello&lt;/h1&gt;`
- **UTF8, Unicode**
- **Base 64, URL Encoding, ...**

Escaping : Changing the interpretation of the following characterer by adding an escape character (Subset of encoding)

- **SQL:** `SELECT * FROM a WHERE name='Sam\'s`
- **Python/Javascript:** `"\"hello\"","\"\"hello\"\""`,
- **Shell:** `cd My\ Folder`



# Injection

---

# Input Validation

Identification

← → ✕ 🏠  🔍

Name

Birthdate

Where the data comes from ?

- Form (maybe unintentional)
- Cookie
- URL
- Headers
- Another web site (ajax)
- ...

How to perform data validation ?<sup>1</sup>

- Data Type Validation (cast/parse)
- Range and constraint validation (min/max/neg)
- Code and Cross-reference validation (Rules, Object exist, )
- Structured validation (JSON/XML Schema)
- Whitelist : Array of allowed values (Days, Country)
- Regular Expression
- Blacklist : Need to be sure

---

<sup>1</sup>Input Validation Cheat Sheet, Data validation

# Input Validation 4

## Server Side



### File Upload:

- Size
- Filename
- Extension
- Mime Type
- Magic number
- polymorph images

Misdone, **Input** Data Validation can lead to :

- Corruption

Disable any **active** content by using appropriate **encoding** before transmitting it to the targeted interpreter.

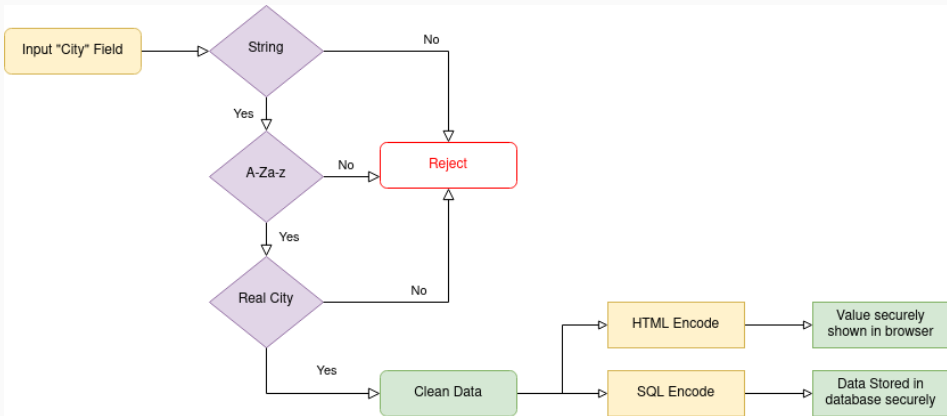


## Output Data Validation 2

What is the destination / targeted system ?

- SQL interpreter : ' " ;
- HTML browser : < >
- URL : : / @ & =
- XML file : < >
- Command Line : ; &
- **Use the correct Encoding**
- **Use SQL Prepared Statements**

# Output data Validation 3



Misdone, **Output** Data Validation can lead to :

- Corruption
- Injection
- RCE
- Gain information
- Gain privileges
- XSS

[https://owasp.org/www-project-top-ten/2017/A1\\_2017-Injection](https://owasp.org/www-project-top-ten/2017/A1_2017-Injection)

# SQL Injection 1

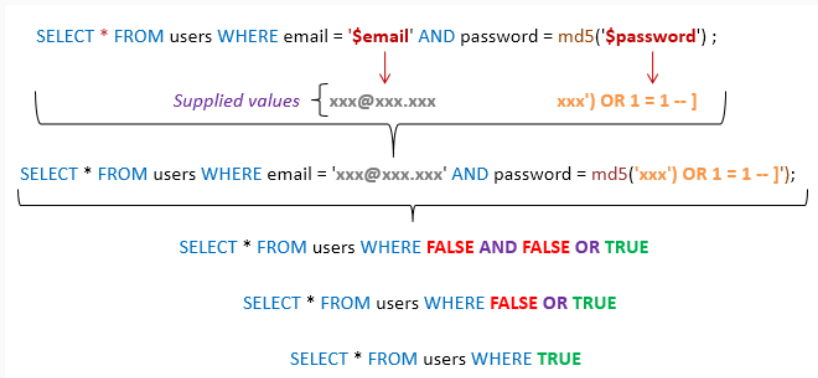


Figure 1: [guru99.com](http://guru99.com)

- Gain privileges, Corruption, ...

# LDAP Injection

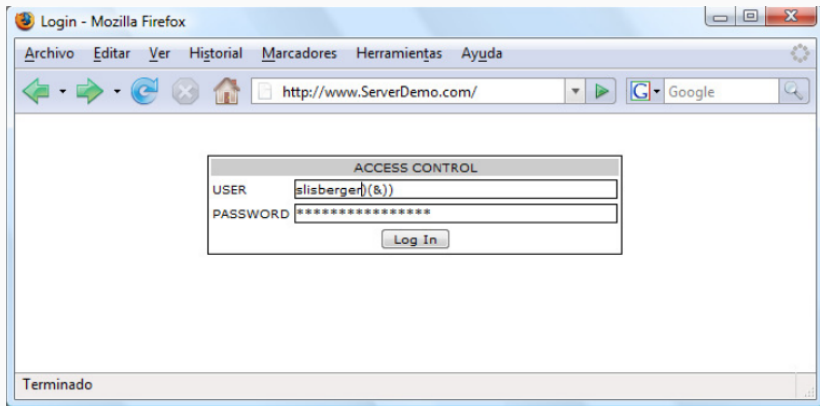


Figure 2: [repo.zenk-security.com](http://repo.zenk-security.com)

`(& (USER=slisberger) (&)) (PASSWORD=Pwd))`

- Gain privileges

## Vulnerability: Command Injection

### Ping a device

Enter an IP address:

```
PING 192.168.0.1 (192.168.0.1) 56(84) bytes of data.  
64 bytes from 192.168.0.1: icmp_seq=1 ttl=63 time=4.71 ms  
64 bytes from 192.168.0.1: icmp_seq=2 ttl=63 time=4.47 ms  
64 bytes from 192.168.0.1: icmp_seq=3 ttl=63 time=4.10 ms  
64 bytes from 192.168.0.1: icmp_seq=4 ttl=63 time=6.24 ms  
  
--- 192.168.0.1 ping statistics ---  
4 packets transmitted, 4 received, 0% packet loss, time 3007ms  
rtt min/avg/max/mdev = 4.106/4.884/6.248/0.819 ms  
/app/vulnerabilities/exec
```

Figure 3: [chris-young.net](#)

- RCE, Gain information, Gain privileges